Ref. Ares(2026)2254189 - 01/03/2026

care, judgment, dexterity

CRAEFT

Craft-specific action simulations

Project Acronym	Craeft
Project Title	Craft Understanding, Education, Training, and Preservation for Posterity and Prosperity
Project Number	101094349
Deliverable Number	3.1
Deliverable Title	Craft-specific action simulations
Work Package	3
Authors	Xenophon Zabulis, Panagiotis Koutlemanis, Ioanna Demeridou, Nikolaos Partarakis, Peiman Fallahian, Nikolaos Nikolaou, Maria Eleni Choclidaki, Asterios Leonidis, Polykarpos Karamaounas, Vassiliki Manikaki
Number of pages	289

This project has received funding from the European Commission, under the Horizon Europe research and innovation programme, Grant Agreement No 101094349.
http://www.craeft.eu/

Executive summary

This deliverable (D3.1) documents Craeft's technical work on craft-specific action simulations: a set of reusable methods and software components that enable craft processes to be represented, parameterised, simulated, and visually evaluated. The deliverable is written as an engineering and research record, aimed at making the work traceable (what was simulated and why), reproducible (how it can be rerun or extended), and transferable (how the same components can support multiple craft domains and tools).

Motivation and problem framing

Craft processes sit at an awkward intersection of requirements. They demand physical realism (materials deform, fracture, flow, or change state), perceptual realism (appearance depends on illumination, thickness, translucency, gloss), and usability (design exploration must remain interactive). High-fidelity simulation methods such as FEM are essential for capturing many craft-relevant behaviours, but are typically too slow for interactive authoring and iteration. Conversely, real-time engines support exploration but can miss the constitutive richness needed for meaningful craft predictions. D3.1 addresses this tension by proposing and demonstrating a layered simulation pipeline that links meaning, physics, and appearance.

Approach and document rationale

The deliverable is organised by dependencies so that later chapters reuse earlier capabilities without circular definitions:

- Semantic layer (Section 3): a structured description of craft entities and actions to ensure that parameters and outputs remain interpretable, comparable, and reusable.
- Physics backbone (Section 4): FEM-based simulation for high-fidelity reference behaviour, complemented by interactive simulation and learning-based surrogates that make high-fidelity behaviour usable in exploration workflows.
- Appearance and evaluation layer (Section 5): light transport simulation as a reusable visualisation capability for controlled studies and perceptual evaluation, positioned immediately after FEM to make the *physics* → *appearance* pipeline explicit.
- Geometry-generation and modelling utilities (Sections 6–9): methods for producing and managing craft-relevant forms that exploit the earlier infrastructure.
- Application systems (Sections 10–11): two end-to-end demonstrations: stained-glass composition design (dependent on light transport) and plaster turning (dependent on solids by revolution).
- Synthesis (Section 12): conclusions, limitations, and forward directions.

Key results and contributions

D3.1 reports the following outcomes:

1. A traceable representation of craft actions and parameters. The semantic structures reported in Section 3 provide a consistent way to describe "what happens" in a craft process—actions, agents, workpieces, tools, constraints, and outcomes—so that simulation outputs can be interpreted and reused across tools and case studies.
2. A practical hybrid simulation strategy. Section 4 demonstrates how FEM can be treated as a reference capability rather than an interactive runtime, with pathways to compress high-fidelity behaviour into forms suitable for tool integration (e.g., calibrated approximations and surrogates). This supports a pragmatic balance between physical fidelity and interactive usability.
3. A reusable light-transport capability for appearance-sensitive evaluation. Section 5 establishes controlled rendering as an evaluation instrument, not merely a presentation step. This is particularly important for crafts where perception is central (e.g., glass and glaze) and where illumination/viewpoint changes materially affect judgement.
4. Reusable modelling methods for craft design and fabrication-aware workflows. Sections 6–9 describe geometry-generation and modelling capabilities—reliefs, axisymmetric forms, mould-related workflows, and multi-part composites—implemented and presented as reusable building blocks rather than one-off demonstrations.
5. Two application chapters showing end-to-end integration.
    a. Stained-glass composition (Section 10): demonstrates how structural design (e.g., segmentation and assembly constraints) can be coupled with light transport to support perceptual evaluation under realistic lighting.
    b. Plaster turning (Section 11): demonstrates how axisymmetric modelling supports the simulation and communication of a craft process grounded in rotational geometry, enabling structured exploration and educational/communication scenarios.

Value for the project and intended use

The deliverable strengthens Craeft's simulation portfolio in three ways:

1. It provides a coherent technical rationale for combining semantics, FEM, and rendering into a single workflow.
2. It establishes reusable components that can be integrated into future Craeft tools and demonstrators.
3. It produces evidence and documentation suitable for internal integration, external review, and handover to developers and partners.

The deliverable is intended to be read by (i) technical teams integrating simulation components into Craeft tools, (ii) researchers extending methods and validating assumptions, and (iii) craft/design stakeholders who need to understand what can be simulated and how to interpret results.

Limitations and next steps

The deliverable makes clear that simulation quality depends on (i) calibration of parameters and boundary conditions, (ii) coverage of the action/parameter space when building surrogates or approximations, and (iii) multi-criteria validation (physical plausibility, perceptual plausibility, and usability). Recommended next steps include strengthening measurement-driven calibration, expanding

action coverage with uncertainty-aware surrogates, improving authoring workflows for non-expert users, and ensuring interoperability/preservation of configurations, datasets, and semantic descriptors.

Document history

Date	Author	Affiliation	Comment
19/06/2024	Xenophon Zabulis	FORTH	Drafting
12/07/2024	Xenophon Zabulis	FORTH	Drafting
21/07/2024	Xenophon Zabulis	FORTH	Drafting
20/10/2025	Xenophon Zabulis	FORTH	Drafting
20/02/2026	Xenophon Zabulis	FORTH	Drafting
28/02/2026	Xenophon Zabulis	FORTH	Drafting

Abbreviations

ABS	Acrylonitrile Butadiene Styrene
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
AS	Acrylonitrile Styrene
AVI	Audio Video Interleave
BC	Before Christ
BRDF	Bidirectional Reflectance Distribution Function
BSDF	Bidirectional Scattering Distribution Function
CAD	Computer-Aided Design
CAP	CRAEFT Authoring Platform
CH	Cultural Heritage
CHI	Cultural Heritage Institutions
CFD	Computational Fluid Dynamics
CLI	Command-Line Interface
CLD	Cognitive Load Theory
CNC	Computer Numerical Control
CPU	Central Processing Unit
CSG	Constructive Solid Geometry
DLL	Dynamic Link Library
ENSAD	École nationale supérieure des arts décoratifs
EOS	Equation of State
FEA	Finite Element Analysis
FDM	Fused Deposition Modelling
FEM	Finite Elements Method
FPS	Frames Per Second

GPa	Gigapascal
GPU	Graphics Processing Unit
HCD	Human Centred Design
HCI	Heritage Computer Interaction
HDR	High Dynamic Range
HDRI	High Dynamic Range Image
HSV	Hue, Saturation, Value
ICH	Intangible Cultural Heritage
IPR	Intellectual Property Rights
IR	Infrared
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
KG	Knowledge Graph
LCP	Liquid Crystal Polymer
LOD	Level of Detail
MPa	Megapascal
MET	Material Engagement Theory
MoCap	Motion Capture
NeRFs	Neural Radiance Fields
NIR	Near-Infrared
NURB	Non-uniform rational B-spline modelling
OBJ	Object
PA	Polyamide (Nylon)
PBR	Physically Based Rendering
PBS	Physics-Based Simulation
PBT	Polybutylene Terephthalate
PC	Polycarbonate
PE	Polyethylene

PET	Polyethylene Terephthalate
PMMA	Polymethyl Methacrylate
POM	Polyoxymethylene (Acetal)
PP	Polypropylene
PPO	Polyphenylene Oxide
PS	Polystyrene
PSF	Polysulfone
PVC	Polyvinyl Chloride
Qs	Solid Transformation Matrix
Qt	Tool Transformation Matrix
RGB	Red, Green, Blue
SDK	Software Development Kit
SLA	Stereolithography
SPP	Samples Per Pixel
TXT	Text
UCD	User-Centred Design
UI	User Interface
UV	Ultraviolet
VR	Virtual Reality
WWI	World War I
XR	Extended Reality
2D	Two-Dimensional
3D	Three-Dimensional

Table of contents

1 Introduction

In recent years, the convergence of traditional craftsmanship with advanced computational simulation has opened new opportunities for preserving, exploring, and innovating craft practice. As these technologies mature, they enable increasingly faithful representations of how materials, tools, and techniques interact—supporting both a deeper understanding of traditional processes and new forms of digital-first experimentation.

This deliverable contributes to that trajectory by documenting a suite of methods and software components that allow craft actions and their outcomes to be studied, previewed, and communicated in silico. The aim is not to "replace" practice, but to provide makers, designers, and researchers with computational instruments that make craft behaviour easier to reason about: what changes when a parameter changes, what the plausible outcome space looks like, and which constraints must be respected for a design to remain physically realisable.

The work is motivated by a practical tension. On the one hand, high-fidelity simulation can capture complex phenomena relevant to crafts, but it is computationally expensive and poorly suited to interactive exploration. On the other hand, real-time engines and simplified models enable interactivity but often lack the constitutive richness needed for craft-relevant predictions. Craeft's approach, as reflected in this deliverable, is to treat these as complementary layers and to build a pipeline in which: (i) craft actions are described and parameterised in a traceable way, (ii) offline simulation supplies reference behaviour where needed, (iii) interactive approximations support exploration and iteration, and (iv) physically based visualisation supports evaluation and communication.

1.1 Objectives

The deliverable pursues four objectives:

1. Action-centric simulation: represent craft processes in terms of actions (who/what acts on what, under which constraints), with parameters that can be interpreted, varied, and reused across scenarios.
2. Physics-grounded prediction: use high-fidelity simulation where necessary to model craft-relevant behaviour beyond simple rigid/soft-body dynamics, and to provide reliable reference data.
3. Interactive exploration: support design-time iteration by enabling real-time or near-real-time approximations suitable for user-facing tools.
4. Perceptual/visual evaluation: provide physically based rendering capabilities to evaluate appearance-sensitive crafts (e.g., glass, glaze) under controlled illumination and viewing conditions.

1.2 Technical approach in brief

The document reports a layered technical stack that connects meaning, physics, and appearance:

1. A **semantic** layer to structure craft entities and processes (tools, workpieces, actions, constraints) so that simulation parameters and outputs remain interpretable and traceable.
2. A **physics** layer that combines (where appropriate) offline FEM "ground truth" with interactive simulation and surrogate approximations, allowing craft phenomena to be studied both accurately and interactively.
3. A **light-transport / rendering** layer enabling controlled appearance simulation for materials and artefacts, supporting comparative studies, presentation assets, and environment-dependent previews.

On top of these foundations, the deliverable develops and demonstrates craft-facing modelling workflows, including relief generation, axisymmetric solids, mould-related modelling, composite objects, stained-glass composition, and plaster-turning.

1.3 Scope and limitations

This deliverable should be read as a tool-and-method report. It focuses on a set of representative craft-relevant phenomena and workflows that are sufficient to (i) validate the approach, (ii) demonstrate integration across the stack, and (iii) provide reusable components for future Craeft tools. Where the underlying physics is inherently complex, the text makes explicit which assumptions are adopted (e.g., which material behaviours are captured by FEM versus approximated interactively) and how results should be interpreted.

1.4 Intended readership and use

The document is written for three, likely overlapping, audiences:

1. Craft and design stakeholders who need to understand what can be simulated, what the outputs mean, and how simulation supports ideation, communication, and training.
2. Technical developers and researchers who need sufficient detail to reproduce scenarios, extend the tooling, or integrate components into other Craeft utilities.
3. Project reviewers and integrators who need a coherent account of WP3 outputs, dependencies, and evidence of feasibility.

1.5 Software artefacts and supporting evidence

Where appropriate, implementation details, parameter definitions, and extended technical material are provided in annexes to keep the main narrative readable while preserving reproducibility. The annexes function as the "engineering record" for the reported capabilities (e.g., toolbox interfaces, training-data production and FEM definitions, material state descriptions, and user-facing manuals).

The document guide below explains how the sections are arranged and how dependencies flow across the deliverable (i.e., which chapters provide foundations for later ones).

1.6 Document guide and rationale

This deliverable is organised as a dependency-driven narrative: early sections introduce the conceptual and computational building blocks, and later sections present design tools and craft-facing applications that explicitly reuse those building blocks. The ordering, therefore, reflects what must be defined first for subsequent sections to be meaningful and reproducible.

At a high level, the document progresses from (i) motivation and context, to (ii) semantic and physical simulation foundations, to (iii) appearance simulation, to (iv) geometry-generation methods, and finally to (v) craft-specific workflows and case-study systems. This structure minimises forward references and makes it clear how results are composed from reusable components.

The section-by-section map of this deliverable is as follows.

1. Introduction. Defines the scope, audience, and overall contribution of the deliverable, and explains how the technical chapters relate to craft workflows and evaluation evidence.
2. Background. Establishes the prior art and technical context (simulation, representation, rendering, and craft digitisation), clarifying what is reused and what is novel in the present work.
3. Semantic Simulation. Introduces the semantic/knowledge layer used to parameterise simulations and to ensure that outputs are interpretable, traceable, and reusable across tools and use cases.
4. FEM simulations. Presents the core high-fidelity physical simulation capability. This section functions as a physics backbone: later geometry-generation methods and downstream applications draw their parameters, constraints, or validation logic from FEM outputs.
5. Light Transport Simulations. Introduces appearance simulation as a reusable infrastructure layer that benefits from, and integrates with, the physical grounding of Section 4. Placing light transport immediately after FEM makes the pipeline logic explicit: physics-informed shape/material states can be propagated into physically-based visual outcomes that are then reused by later design tools and demonstrations.
6. Reliefs. Describes relief-generation as a geometry-processing method that leverages the preceding simulation capabilities (notably the physical modelling introduced in Section 4, and the visual evaluation capabilities of Section 5 where relevant) to support controlled shape design and fabrication-aware variation.
7. Solids by revolution. Covers a second family of geometry-generation methods (axisymmetric modelling), positioned before craft applications that rely on rotational geometry.
8. Moulds. Presents mould-related modelling methods and workflows, building on the geometry-generation capabilities developed earlier (Sections 6–7) and, where relevant, using visual simulation (Section 5) for inspection and presentation.
9. Composite objects. Extends the modelling pipeline to multi-part assemblies and heterogeneous configurations, combining the earlier geometry methods with the simulation and rendering infrastructure established in Sections 4–5.
10. Stained glass compositions. Introduces the stained-glass design system as an application chapter that explicitly relies on light transport (Section 5) to support perceptual evaluation of compositions and illumination-dependent effects.
11. Plaster turning. Describes the plaster-turning workflow and system, grounded in the axisymmetric modelling methods of Section 7 (and supported, where relevant, by the simulation and visualisation infrastructure already established).

12. Conclusions. Synthesises findings across the technical foundations and the application chapters, and outlines limitations, transferable lessons, and next steps.

Annexes provide implementation details, data specifications, extended results, and evidence that would interrupt the main narrative if placed in-line. The main chapters, therefore, remain readable as a coherent methodological story, while annexes preserve traceability and reproducibility for technical readers who need to validate or reuse the reported work.

2 Background

Section 2 provides the background required to interpret the technical contributions in subsequent sections. These draw on methods and toolchains that originate in several distinct literatures: physics-based simulation, semantic modelling, physically based rendering, and domain-specific digital design software used by practitioners. The purpose of this section is therefore twofold: (i) to position the project's technical choices within the state of practice, and (ii) to establish a vocabulary that makes later sections comparable and traceable.

Digitisation of tangible heritage relies on photographic documentation and 3D reconstruction to record the asset's morphology and materiality, or in simpler terms, its geometry and appearance. While geometry is constant at a moment in time, appearance is variable because it is influenced by the environment. To move beyond this static snapshot, we rely on simulation. In this context, Physically Based Rendering (PBR) acts as a simulation of light transport. By applying the captured optical properties to the digital model, PBR calculates how light interacts with the material, allowing the object to be visualised realistically under any new environmental conditions.

Finally, Section 2.5 extends the background beyond simulation theory into the tool ecosystems that motivate the craft-specific modules later in the deliverable. Stained-glass composition design is used as a representative case: it demonstrates that many craft workflows are anchored in production-oriented software pipelines whose outputs are fabrication artefacts, and whose representations routinely move between raster and vector formats with non-trivial interoperability constraints. This perspective is important for Craeft because it clarifies what "integration" means in practice: later chapters do not merely propose new algorithms, but must interoperate with the software conventions, file formats, and workshop constraints that practitioners already rely on.

2.1 Appearance simulation

2.1.1 Digitisation

To date, CH research has focused on the visualisation of existing artefacts that have been digitised (or, following Computer Vision jargon, 'reconstructed'). This type of digitisation is based on scanning and modelling technologies, which have become valuable tools in the photographic and 3D documentation of cultural heritage artefacts. High-resolution 3D laser scanning and photogrammetry are commonly used to capture precise geometrical details of artefacts as well as their photorealistic appearance. A comprehensive review of 3D scanning technologies and approaches can be found in The Mingei Handbook on Heritage Craft representation and preservation, at Step 3 'Craft recording', in Section 3.2, 'Digitisation of endurant assets.'

Other digitisation methods focus on the material composition of artefacts and, in this case, employ imaging techniques that reveal properties imperceptible to humans, as they identify features outside the visible spectrum, i.e. IR and UV illumination. In [1], advances in multi-spectral and hyperspectral imaging for archaeology and art conservation are reviewed, informing on their use in the identification of the materials used and contributing to their conservation and preservation.

Despite these technologies, most of the visualisation techniques focus on Lambertian, or 'matte', surfaces. The rendering of shiny, transparent and translucent materials explicitly specifies the 3D model regions that have this property. The simulation of the appearance of these materials follows some simple rules, but does not simulate the interaction of light with individual types of materials.

During the last few years, Neural Radiance Fields (NeRFs) [5, 6] were introduced as a method for synthesising novel views of complex 3D scenes from a sparse set of 2D images. They leverage deep learning to model the volumetric scene representation by encoding the scene into a neural network, which can then be queried to render new views. This approach has gained significant attention due to its ability to produce highly realistic images and its applications in various fields, including cultural heritage visualisation. NeRFs can capture the appearance of challenging materials; however, they cannot be used to render scenes with different properties (i.e., illumination and décor) in the scene in which they were imaged.

2.1.2 Physically based rendering

The visualisation of cultural heritage artefacts has seen significant advancements in recent years, driven by technological innovations and interdisciplinary research. With the support of 3D graphics and immersive presentation technologies (i.e., AR, VR), interactive and educational experiences have been produced, making cultural heritage more accessible to broader audiences. Real-time rendering techniques are essential for interactive applications such as VR and AR. These techniques leverage the parallel processing capabilities of GPUs to render scenes efficiently [2].

Photorealistic rendering aims to create images that are indistinguishable from real photographs. This technique is valuable for producing accurate representations of artefacts. In [3], the principles of global illumination, which model the complex interplay of light in a scene to achieve photorealism, are presented. In [4], image-based lighting techniques are employed using photographs of real-world lighting conditions to illuminate 3D models, enhancing the realism of renderings.

In terms of material-specific appearance simulation, a technical approach is proposed that is based on advanced, physically-based rendering software infrastructure called 'Mitsuba 3' [16], which has been developed for research and educational purposes. This infrastructure simulates the interaction of light with materials, enabling the creation of photorealistic images. Domain-specific PBR tools also exist for craft materials whose appearance is tightly coupled to process constraints. In glass, VirtualGlass provides computer-aided design and visualisation specifically for blown-glass cane, enabling rapid exploration of cane layouts and their expected appearance on blown pieces before fabrication [147]. More broadly, modern renderers and engines have converged on practical physically-based shading conventions that influence how BRDF parameters are exposed and interpreted; Disney's "Physically-Based Shading" work is a canonical example that has shaped common parameterisations adopted in production pipelines and real-time engines [139]. This convergence is relevant in Craeft because it reduces the semantic gap between (a) offline, high-fidelity light transport used for appearance studies and (b) interactive preview in engine-based training or authoring tools.

In addition to general-purpose physically based renderers, craft-facing tools have been developed that embed domain knowledge into the authoring workflow. VirtualGlass supports computer-aided design and visualisation of blown-glass cane patterns, enabling rapid visual iteration before physical production [155]. While our focus here is plaster turning rather than glass, the same technical rationale applies: embedding

physically grounded visualisation into the workflow can reduce trial-and-error and shorten design–make cycles.

2.2 Action simulation

Interactive, physics-based simulations have become vital tools for the study and preservation of traditional crafts and mechanical operations. These simulations employ computational modelling of the physical behaviour of materials and mechanisms, aiming to provide immersive and educational experiences.

2.2.1 Physics-based simulation

Physics-based simulations focus on the physical behaviour of mechanical systems and have been pivotal in various fields of engineering and science. The Finite Element Method (FEM) is foundational to many mechanical simulations and was popularised by Clough and others in the 1950s [29, 30]. FEM allows the simulation of complex mechanical systems by breaking them down into smaller, simpler parts known as finite elements. Initially, FEM was applied primarily to civil engineering problems, such as analysing the stress and strain in bridges and buildings [31]. The aerospace and automobile industry played a crucial role in driving the early adoption of mechanical simulations [32].

The commercialisation of FEM software made these tools more widely accessible to engineers across various industries. Notable commercial FEM software included ANSYS [33, 36], NASTRAN [34], and Abaqus [35]. In parallel, Computational Fluid Dynamics (CFD) emerged as a critical tool for simulating fluid flows around mechanical structures, such as aircraft wings and automotive bodies, with software suites like FLUENT and CFX providing instrumental in advancing this field [37, 38].

Traditionally, generic and interactive simulations are based on ‘physics engines’ [10], which are software libraries which simulate the behaviour of physical objects under prescribed dynamic and kinematic configurations. The majority simulates rigid object kinematics and potential collisions [11].

Realistic simulations of actions upon materials are found in the field of scientific simulation. Finite Element Analysis (FEA) [85, 86] is a numerical technique that utilises the Finite Element Method (FEM) to simulate and analyse the behaviour of physical systems. FEA is the *de facto* standard in state-of-the-art physical simulation. The idea behind the FEA is to divide complex physical systems into smaller, simpler, and very local (or finite) elements. The behaviour of each element is predicted by mathematical equations that describe the physical laws governing an action.

Although widely adopted in modern mechanics and engineering, scientific simulation has not widely applied in the domain of crafts. In [87], the formation of knots is studied using FEM. Mechanical models for fibres are proposed that account for elongation, bending and torsion forces, and the frictional contacts between them [88]. In [89], the metalworking processing is studied to understand the quenching process, and results of a computer simulation based on metallo-thermo-mechanics are presented to know how the temperature, metallic structure and stress/distortion vary in the process.

A related (and methodologically informative) body of work comes from heritage engineering and archaeology, where finite-element methods are used to evaluate the plausibility of reconstructed

structures and processes under incomplete evidence. For example, finite-element analysis has been applied to analyse ancient architecture reconstructed from partial measurements, including a detailed case study on North American Arctic (Thule) whalebone houses [153]. Similarly, studies of historical forging devices have combined 3D reconstruction with FEA/CFD to assess whether reconstructed mechanisms operate within plausible mechanical regimes [154]. Although these works are not "craft simulators" per se, they exemplify a common technical pattern that is directly relevant to craft action modelling: translate sparse observations into physically plausible states, then test those states against measurable constraints.

Beyond industrial engineering, FEM has also been used as an interpretive method in heritage and archaeology, particularly when controlled physical testing or reconstruction is impractical. Levy and Dawson apply finite element methods to analyse ancient vernacular architecture in the North American Arctic, using Thule whalebone house structures as a case study to connect material properties and structural hypotheses to plausible load-bearing behaviour [153]. Rojas-Sola et al. similarly demonstrate how digitisation can be combined with FEA (and related simulation methods) to study historic forging devices, linking 3D reconstruction to mechanical interpretation of function [154].

These examples are directly aligned with our use of physics-based modelling in craft contexts: simulation is not only for optimisation, but also for reasoned reconstruction and comparative evaluation of craft outcomes against observed or digitised references.

2.2.2 Game engines based on physics simulation

The integration of physics simulation into game engines has advanced over the past few decades, evolving from basic collision detection to sophisticated simulations of rigid bodies, soft bodies, fluids, and other physical phenomena. In the early stages of game engine development, physics simulations were rudimentary and limited to basic collision detection. An example of this is id Tech 1, which featured simple collision handling without realistic physics [39]. More advanced physics simulations enabled interactions between game objects, especially in the context of rigid body dynamics [40].

As physics engines matured, they were increasingly integrated into mainstream game engines, such as Unreal and Unity. These engines support rigid body dynamics and are expanded to more complex simulations, including cloth, fluid dynamics, and destructible environments [41]. The Unreal and Unity 3D engines were the first to incorporate middleware physics engines like PhysX [42].

The application of physics simulations in game engines has expanded beyond traditional gaming into areas such as VR and AR, where physics-based interactions are essential for creating immersive and engaging experiences [43]. Additionally, game engines like Unreal Engine are increasingly used in virtual production for film and animation, where real-time physics simulations enable dynamic scene creation [44]. Furthermore, these engines are employed in serious games and simulations for engineering, architecture, and medical training, where accurate physics simulations are necessary for realistic virtual environments [45].

Balancing realism with performance remains a significant challenge on hardware-limited platforms, such as conventional computers and mobile devices. Techniques such as Level of Detail (LOD) for physics, hardware acceleration, and multi-threading are continually optimised to address these challenges [46]. Additionally, the demand for real-time simulations is growing, especially in VR/AR and multiplayer online

games, necessitating more efficient and scalable physics simulations. Future developments in game engines involve the integration of AI to dynamically adjust physics simulations, thereby creating more adaptive and responsive environments [47].

2.2.3 Material approximations

Real-time simulation of the manipulation of real-world volumetric deformable objects is a challenging task. Accuracy can be achieved with commercially available simulators [20] that rely on finite element methods. However, these simulators do not perform in real-time and are thus unsuitable for interactive applications. A recent review of simulators that are suitable for robotic applications and therefore typically perform in real-time can be found in [23].

To fulfil both simulation accuracy and real-time requirements, we investigate three different approaches, two based on learning and one that relies on a GPU-powered real-time simulator, PhysX [22]. Learning-based approaches rely on training data to model several aspects of a physical system. In this work, we test two such approaches, one that models the object geometry only, 3DNS [27] and one that models both the geometry and the dynamics, ACID [26].

Our progress is the following. We constructed 3D manipulation data using Simulia, which is considered the ground truth for all the methods. Initial experiments with this data for 3DNS and PhysX are shown in the experiments section. The experimental evaluation of ACID is pending.

3DNS

Implicit surface representations are becoming increasingly popular and achieve state-of-the-art results in several tasks, such as shape representation and reconstruction. 3DNS [27] proposes an approach for making local edits in objects that are represented using implicit surfaces. 3DNS uses a brush-based framework that is intuitive and can be used by sculptors and digital artists, which is in line with CRAEFT goals. The experimental evaluation results show that 3DNS is accurate in terms of modelling the desired edits, while preserving the overall object geometry outside the interaction areas.

ACID

ACID [26] is a dynamic model for deformable object manipulation based on implicit neural representations. Within ACID, two techniques are leveraged: implicit representations for action-conditional dynamics and geodesics-based contrastive learning. Deformable dynamics and occupancy representations are learnt from simulation data. More specifically, the framework is built with the NVIDIA PhysX simulator used to generate a deformable dynamics dataset. By learning the deformation dynamics from simulation, ACID can provide a more accurate simulation in an evolving manipulation setup compared to methods that only model geometrical deformations, such as 3DNS. We plan to experimentally validate its applicability in the following period.

PhysX

NVIDIA PhysX [21] can simulate complex physics interactions in real-time applications, particularly in the realm of soft-body dynamics. By leveraging the parallel processing capabilities of GPUs, PhysX achieves high-performance simulations, enabling real-time interaction and rendering of soft bodies. Soft body

dynamics involve the simulation of deformable objects that respond realistically to external forces, such as gravity, collisions, and pressure. PhysX employs the FEM to handle the complexities of soft body dynamics, ensuring that interactions between objects and the environment are faithfully represented. To achieve this, the objects are assigned properties such as elasticity, friction, and mass distribution. Given that the focus of PhysX is on real-time applications, the set of physical properties is kept small. In the scenarios considered in Craeft, the main external force is the one applied by the tool on the object that is manipulated.

2.3 Craft process simulations

Traditional crafts involve interactions between materials and tools, often requiring a deep understanding of physical properties and manual skills. Interactive, physics-based simulations can accurately replicate these processes, offering new ways to preserve and teach these crafts.

2.3.1 Presentation of crafting actions

In [53], a mobile Augmented Reality (AR) system that superimposes 3D craft objects in space is proposed. The system triggers the virtual demonstration of their usage using a multi-touch surface. A Head Mounted Display is used to present traditional craft objects with high presence and absorption in [54]. In [55], an AR system augments a given physical object with audio and visual digital assets relevant to its making.

Virtual Reality (VR) and handheld controllers are used for more realistic virtual handling and interaction with the presented 3D craft objects. VR demonstrations of traditional beverage production are provided in [56]. The demonstrations utilise pre-recorded interactions with tools and machines, which in VR can be viewed from any viewpoint. In [57], visitors are immersed in a VR environment where they can perform some indicative woodworking tasks, in the context of introducing the usage of dovetailing carpentry tools. As in the previous case, the simulation of potential interactions is pre-recorded and shown as animations.

2.3.2 Textiles

Textile weaving simulations regard the patterns and the interweaving of threads. In [7], a system for the interactive design of woven structures is proposed, enabling the simulation of the weaving process and the visualisation of the resulting fabric in real-time.

A broad range of studies exists on the mechanical characterisation of textiles [90, 91, 92, 93]. Several pertinent works also focus on how textiles are to deform and distort when worn, e.g. [94, 95]. TexGen is an open-source software for modelling the geometry of textile structures, including fibre and textile mechanics [97].

Works that predict the visual appearance of woven textiles are found in the industry. Prominent examples are WeaveIt [76], Weaving Design Software [77], ArahneWeave [78], pixeLoom [79], WeavePoint [80], and WIF Visualizer [81]. The 3D Knitting Simulation [82] was developed for flat and circular knitting technology. Given the fabric design, the simulator creates realistic visualisations for Jacquard Raschel, Multibar Lace, and warp-knit fabrics. More relevant to handcrafted textiles, a physics-based heuristic model is used in [83] to predict the visual results of painting on fabric, using thin-brush dyeing. The

simulator focuses on modelling 2D fluid simulation on fabrics to reduce computational burden. The dyeing algorithm is based on an ink-wash painting algorithm [84].

2.3.3 Robotic re-enactment

Automatic craft product manufacturing was initiated during the Industrial Revolution. Today, robotic automation is the norm in manufacturing industries [58], but for precisely predefined tasks. In this subsection, work on the recreation of human crafting motion is reviewed.

Mimicking human freehand motion using robots has been studied for carving tasks. As robotic motion has fewer and different degrees of freedom than humans, for a robot to achieve the same tool movement as a human, it is required to convert human kinematics to the available robotic embodiment. In [59], human movements are analysed into their principal components and then encoded into robot kinematic instructions. These components were approximated through machine learning in [60, 61, 62].

Taking an inverse approach, other studies focused on achieving the same result as human actions. Studies of carving strokes were conducted to establish a correspondence between human results and robotic motion that approximates the same results [59, 60]. In [63], a step forward was made by adding some sensor-based robotic autonomy in the construction of wooden structures.

2.3.4 Games

Some video games offer creative interaction by replicating basic crafting aspects in the contexts of virtual building and decoration.

Creating virtual pottery using wheel-throwing and subsequent decoration is found in several games engaging creativity (e.g. 3D Pottery [64], Pottery Master [65], and Pottery Simulator [66]), albeit not exhibiting high levels of realism, nor addressing practical constraints.

Beyond entertainment-oriented mobile games, the literature also includes research prototypes that treat *virtual pottery* as a sensing, interaction, and geometry-modelling problem. The Turn software couples a physical spinning wheel into a virtual clay model, providing a direct mapping between real rotation input and digital forming operations [148]. Virtual Pottery demonstrates real-time gesture-driven 3D “pottery” creation using natural hand motions as the primary control modality [149]. DigiClay further explores motion sensing for virtual pottery as an interactive installation, using tracked body/hand input to drive real-time mesh deformation [150]. Complementary work proposes explicit object models and deformation functions that support plausible wheel-throwing transformations (e.g., layered models with dedicated operations for outer shaping, inner deepening/widening, and height reduction) [151]. Finally, Wowtao introduces a manufacturing-aware, personalised pottery-making pipeline that connects user-facing design actions to feasibility-constrained output suitable for production workflows [152].

Technically, these systems are relevant to our “Plaster Throwing” simulator because they emphasise (i) input modality design (wheel/touch/gesture), (ii) computationally efficient deformation representations, and (iii) constraint handling when the digital model must remain manufacturable, concerns that translate directly to plaster-master shaping even when the underlying material model differs.

In the adventure game genre [67], the prerequisite of crafting or recipe materials is addressed by requiring them to be available for the execution of an action. Although recipes do not simulate material treatment, they pose constraints that the player must follow. A central concept in these games is the concept of recipes, or the knowledge necessary to transform a collection of ingredients (materials) into a new object. For example, a recipe for a pickaxe specifies two twigs and two flints as the necessary materials. Some recipes regard specific types of materials, such as a recipe for tailors that transforms fibres into fabric or a recipe for blacksmiths that transforms bulk metal into a sword.

The use of human motion in gaming interaction has proliferated since the Wii controller. The Knitting Simulator 2014 [68] requires the manipulation of controllers that resemble knitting needles. The usage is simplified as the needle motion is only used to advance a knitting animation. In [69], a VR controller is used to edit solids by revolution in a wood-turning lathe crafting simulation. An architecture for integrating the Unity game engine as a platform for craft simulation is proposed in [70].

PhysX [71] is a physics engine middleware SDK for the development of gaming applications that are based on hardware acceleration to achieve real-time performance. It supports rigid and body dynamics and volumetric fluid simulation. However, it cannot simulate the generality of phenomena in the context of this work, as it does not support all of the material properties and damage models of interest in this work.

2.3.5 Serious games

Woodwork Simulator [72] recreates the experience of working in a carpenter's workshop. It provides reasonable approximations of the effect of virtual saws, drills, glue, chisels, and sandpaper on virtual wood. Educational uses are found in workspace geography and safety, training in the use of tools, and the planning of woodworking processes that implement specific designs.

In [73], a blacksmith's forge is simulated in VR, providing simplified tool interaction that shapes metallic pieces using 3D controllers; the crafted structures can be 3D printed.

Counting and calculating tasks are intrinsic to the weaving of fabrics and wicker. In [74], these capacities are trained to make calculus for young students more interesting, intuitive, and educational on Native American Heritage.

In [75], glasswork actions are simulated to accustom to the weight, balance, and handling of a real blowpipe performed at a real glassblower's bench. A Mixed Reality system tracks human hands and illustrates the user against exemplar hand movements for that action.

Pottery simulations involve the manipulation of clay on a rotating wheel. In [8], a haptic simulation of pottery-making provides users with tactile feedback, enhancing the realism of the experience and enabling the replication of traditional techniques. Beyond casual or entertainment-oriented "pottery simulators", research prototypes have explicitly studied interaction and training for pottery forming. Representative examples include gesture-based virtual pottery interfaces [149], augmented-space and AR pottery-making experiences [141], training-oriented systems (PotteryGo) [143], and mobile/consumer-facing systems that connect virtual design to manufacture under constraints (Wowtao) [152]. Work on commodity hand sensing includes Leap Motion–enabled bare-hand shaping (zPots) [142]. These efforts motivate two design choices that recur in technical craft simulation: (i) explicit staging of actions into

discrete, learnable steps, and (ii) feedback that is intelligible as a “making cue” (geometric, kinematic, or perceptual), rather than as a purely numerical signal.

Virtual pottery has been explored in the literature under a variety of interaction modalities, spanning gesture-driven installations, augmented/virtual reality, and training-oriented desktop/mobile applications. Han and Han present “Virtual Pottery”, a virtual 3D audio-visual interface driven by natural hand motions, illustrating how hand kinematics can be mapped to plausible wheel-based forming operations in real time [149]. Complementary “augmented space” approaches demonstrate pottery making with spatially situated feedback and interaction, bridging physical space and virtual deformation [141]. Training and play have also been targeted explicitly: PotteryGo proposes a virtual pottery-training system with an emphasis on learnable stages of operation [143], while Wowtao provides an easy-to-use personalised pottery-design workflow that explicitly considers industrial constraints for downstream manufacture [152]. On commodity sensing, zPots demonstrates close-range bare-hand shaping enabled by the Leap Motion device [142], and DigiClay reports an interactive installation for virtual pottery based on motion sensing technology [145].

These systems largely focus on “clay-as-medium” and on wheel-based deformation under direct hand interaction. By contrast, the Craeft workflow targets a plaster master used for slip-casting mould making; this shifts the technical emphasis from modelling viscoplastic clay to staging discrete, tool- and posture-mediated actions (centring, profiling, calibration against templates) whose correctness can be assessed via geometric and perceptual cues. The literature above is therefore used here as an interaction and staging precedent, while the material state and manufacturing endpoint are distinct.

Metalworking processes such as forging, casting, and machining involve significant physical transformations. In [12], deformable models that simulate the plastic deformation of materials, which are essential for creating realistic simulations of metalworking operations, were developed.

Interactive simulations for mechanical assembly enable the virtual assembly of mechanical systems, providing insights into the assembly process and the interactions between components. In [9], Fang and Chang (2010) developed a system for simulating the assembly of mechanical parts, enabling users to explore different assembly sequences and detect potential issues.

2.4 Semantic simulation

Semantic simulation sits at the interface between knowledge representation and computational physics. Its purpose is to describe physical entities, processes, states, and causal relationships in a way that is both conceptually explicit and computationally actionable. In cultural heritage and craft contexts, this semantic layer is particularly valuable because it supports (i) the integration of heterogeneous evidence, (ii) the traceability of modelling choices, and (iii) the reuse of simulation descriptions across tools and applications.

In practice, semantic simulation is rarely built from scratch. Instead, it is assembled by combining (a) heritage-oriented reference ontologies with (b) foundational ontologies for general categories such as continuants/occurrents or endurants/perdurants, and (c) simulation-oriented ontologies that explicitly connect physical phenomena to their computational representations. The remainder of this subsection reviews the main ontology families relevant to this deliverable.

2.4.1 CIDOC CRM extensions

The CIDOC Conceptual Reference Model (CIDOC CRM) [178] provides a widely adopted semantic backbone for cultural heritage data integration, and is standardised as ISO 21127. Its extension mechanism enables specialised models to remain compatible with the core event-centric view while introducing domain-specific constructs.

The CRMsci (Scientific Observation Model) [179] extends CIDOC CRM to describe scientific observation, measurement, sampling, and processed data in descriptive and empirical sciences. It is particularly relevant when simulation is grounded in evidence, because it provides explicit constructs for observational activities (observations are treated as events), their procedures, and their results, enabling simulation inputs to be tied to documented measurements.

The CRMgeo (Spatiotemporal Model) [180, 181] extends CIDOC CRM to represent spatiotemporal properties of entities and events, including distinctions between phenomenal spatiotemporal extents and declarative extents. This is important for simulation because it provides a principled way to connect geometry, location, and time to evidence and to derived representations.

The CRMpe (PARTHENOS) entities [182] extend CIDOC CRM to represent the procedural and infrastructural context of research activity, including services, protocols, datasets, and project-level organisation. In the context of semantic simulation, this supports describing how evidence and digital assets are produced, curated, and linked. This way, simulation models, runs, and outputs can be placed within a traceable workflow of knowledge production rather than being isolated artefacts.

These CRM-compatible models provide a CH-aligned layer for describing what things in spacetime, events/processes, and measurements, are prerequisites for semantically grounded simulation.

2.4.2 Physics and engineering ontologies

While CRM extensions provide strong coverage for heritage entities and observation, simulation also benefits from alignment with foundational ontologies that clarify how physical reality is categorised at a high level.

The Basic Formal Ontology (BFO) [183] distinguishes continuants (objects as entities that persist through time) from occurrents (events as entities that unfold in time). This distinction is useful when modelling craft processes: the same workpiece persists as a continuant, while its forming actions are occurrents that transform its state.

The Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [184] similarly provides an endurant/perdurant distinction, and is often used to model qualities (e.g., temperature, velocity, hardness) and their values as regions in an abstract space. This offers a disciplined way to talk about “state variables” and their evolution without prematurely committing to solver-specific encodings.

Foundational ontologies help make simulation semantics coherent, avoiding category mistakes such as treating a process as a material object, and provide anchors for mapping between domain concepts, state variables, and computational artefacts.

2.4.3 Physics-Based Simulation Ontology

A persistent challenge in simulation is the gap between (i) the physical phenomenon one intends to model and (ii) the solver-specific representation used to compute it. The Physics-Based Simulation Ontology (PSO) [185] addresses this by explicitly separating two layers:

- PSO-Physics, which models the physical entities, processes, and relationships of interest independently of any particular solver; and
- PSO-Sim, which models the informational and computational artefacts that represent those physical entities inside a simulation workflow (e.g., boundary condition specifications, discretisations, solver inputs, and derived outputs).

This separation is valuable for two reasons. First, it supports reuse across tools: the same physical model can, in principle, be instantiated in different solvers by mapping PSO-Physics to different PSO-Sim encodings. Second, it enables explicit recording of assumptions and approximations: discretisation choices, constitutive simplifications, parameter ranges, and validation constraints can be represented as first-class objects rather than being buried in code or undocumented configuration files.

Finally, PSO is well aligned with broader engineering patterns such as digital twins and simulation-driven design exploration, because it supports representing not only outcomes but also the semantics of the model construction and execution that produced those outcomes [186].

2.5 Software for stained-glass composition design

Stained-glass composition design is a production-oriented design activity: the end-product is not only a visual layout, but a pattern system that can be fabricated. Digitally, this means a design must encode seam geometry (lead/came or foil), support piece identity (typically numbering), and support printing with exports that remain usable across a workshop toolchain. Dedicated stained-glass packages, therefore, prioritise pattern-centric operations rather than treating these as secondary export steps. [187]

A practical way to understand the software landscape is through the common raster-to-vector workflow. Raster imagery is a natural starting point when the input is a sketch, scan, or photograph; vector paths are essential when the goal is clean scaling, precise seam boundaries, and CAD-friendly exports. In low-cost workflows, this split is typically handled by a raster editor for intake/cleanup and a vector editor for seam geometry and final pattern layout. [189], [190]

2.5.1 Pattern correctness

Across tool families, the fabrication artefact is a pattern: piece boundaries with stable thickness conventions, unique piece identifiers, and a print/export workflow that preserves scale. Stained-glass-native tools explicitly implement this as a first-class feature set, which reflects the practical reality that workshop work depends on consistent, repeatable templates. [187]

2.5.2 Interoperability

Interoperability is typically framed as “supports DXF/DWG”, but in practice, the crucial question is whether exports round-trip reliably through downstream consumers, including CAD tools, cutters, CNC/laser pipelines, and print workflows. A known friction point is DXF curve representation: Inkscape’s DXF export is commonly discussed as emitting SPLINE curves and not being universally compatible with all downstream tools, which can matter if the fabrication chain expects polylines/arcs or imposes constraints on curve primitives. In those cases, users often need an explicit DXF interpretation step that includes curve conversion, discretisation, or routing via a CAD tool [190]. Commercial vector tools provide explicit guidance on supported formats and import/export constraints, and are often used when interoperability must be tightly controlled [191].

2.5.3 Preview realism

Preview requirements sit on a continuum. Many stained-glass-centric tools offer glass texture libraries, swatch-based colouring, and limited transparency preview [187, 188]. By contrast, physically based renderers simulate light transport and can support more faithful reasoning about translucency and illumination context, at the cost of additional modelling effort and more complex scene specification. This distinction motivates the separation in Craeft between pattern authoring (Section 10) and physically based appearance studies (Section 7). [192].

2.5.4 In-situ preview

A smaller, but increasingly relevant, workflow class supports in-situ preview, where a design is inspected in a target architectural context (window opening, view distance, lighting situation). Here, rapid 3D mock-up and AR viewing can support stakeholder communication; the workflow then returns to the 2D pattern environment for piece numbering, scale-true printing, and fabrication outputs. [193]

2.5.5 Cutter/plotter integration

When production includes plotters/cutters (or laser workflows for templates, masks, or related tasks), additional tooling becomes relevant: nesting to reduce waste and organise parts (often by colour), and machine handoff formats and constraints. Some stained-glass packages bundle this at higher tiers; open-source utilities can provide nesting, and laser-control software can serve as the final stage of a vector-to-machine pipeline [187, 194, 195].

This landscape clarifies why Section 10 treats stained-glass composition as a distinct module: stained glass design is constrained by pattern correctness, piece identity, scale fidelity, and export reliability, while also benefiting from optional preview layers (swatches, in-situ AR, or physically based rendering). Section 10 is positioned to operationalise these constraints in a Craeft-compatible way: composition and pattern generation as reusable tooling, with clear interfaces to downstream visualisation and fabrication workflows. [187, 188, 190].

3 Semantic Simulation

3.1 Rationale

Work Package 2 established a set of *archetypal* simulators: generic, parameterised action templates that describe how physical change is induced by canonical causes (e.g., contact, pressure, friction, heating) and how this change can be computed by an appropriate simulator. These archetypes are intentionally craft-agnostic: they define "what kind of physics is happening" without committing to a particular craft vocabulary, tool set, or workflow episode. In WP3 we turn these archetypes into *craft-specific* simulators by specialising them to concrete craft events, selecting the relevant entities (tools, workpieces, fixtures), and instantiating action parameters from observed practice. This specialisation step follows the same pattern: take an archetypal action and instantiate a specific scene and material regime.

The purpose of *semantic simulation* is to make this transition explicit and traceable. A high-fidelity solver does not operate on "marvering" or "blowing" as such; it operates on boundary conditions, constitutive laws, loads, contacts, and numerical schemes. To prevent the simulation from becoming an opaque collection of solver settings, we use a semantic layer, called Physics-Based Simulation Ontology (PSO). This way, we bridge physical phenomena and their computational representation, so that assumptions and approximations are recorded as *domain concepts* rather than as undocumented file syntax. Concretely, PSO is used so that a boundary condition written in a computer file is grounded in a shared conceptual model of physics (semantic grounding), enabling independent simulation steps to be connected by meaningfully typed outputs and inputs.

In this section, we analyse (a) the semantic representations of the actions and constraints and (b) their corresponding physical simulations. The semantic layer provides the glue that lets us compare simulations across materials, reuse archetypal templates across crafts, and integrate downstream components such as visualisation workflows that already consume these archetypal models in a material-specific way.

We present the glass case in depth because it is the most mechanically and semantically demanding: it exercises the full range of physics used in this deliverable, namely coupled mechanical and thermal behaviour. The workflow explicitly chains thermally driven state variables into mechanical response. In addition, the glass model requires temperature-dependent material properties updated volumetrically, which makes the thermal-mechanical coupling essential.

3.2 Use case demonstration

A two-step process comprising two fundamental glassworking actions, marvering and blowing.

The marvering technique is registered in this thesaurus and classified under the <glassworking processes and techniques> category in the Getty ATT: Its definition, in [aat: 300233474], is:

Marvering is the process of rolling and shaping a gather of molten glass on a marver to make it symmetrical, to centre it on the blowpipe, or to embed applied threads into the surface as decoration.

The marver is [aat: 300022860] *a hard, flat surface of stone, wood, or metal on which a mass of molten glass gathered on the end of a blowpipe or pontil is rolled and shaped in glassmaking*.

Getty does not have a specific definition for this action, only for glassblowing in general [aat: 300053932]. Our definition is:

Blowing, bubbling, insufflating, or inflating is the fundamental act of inflating molten glass into a hollow bubble by blowing air through a pipe. That is, expanding a gather by forcing air into it through a blowpipe, to create hollow vessels.

3.2 Our approach

We map the physical actions of a glassblower to physical mechanisms that, in turn, can be simulated by the Finite Element Method (FEM). This mapping allows us to correspond semantically represented physical phenomena to mathematical models.

Using semantic tags, we can connect independent simulation steps that are developed independently. In this case, the output of the marvering step becomes the input for the blowing step:

1. Marvering: outputs a pso:TemperatureField.
2. Transition: pso:TemperatureField is used to determine a pso:ViscosityMap for a given material via a modelling equation.
3. Blowing: this pso:ViscosityMap is used to determine how a given pso:InternalPressure will deform the parison.

That is, the purpose of PSO is to ensure that a boundary condition in a computer file is grounded in a domain concept. This 'Semantic Grounding' links the simulation to a shared understanding of physics.

3.3 Maker-Material Negotiation: Thermomechanical Shaping

We recall the analysis in D2.1, which identified the fundamental properties and computational models required for simulating each action type. This prior analysis enables us to shortlist the specific physical entities relevant to the shaping actions we study.

3.3.1 Temperature-dependent material properties

The material properties of both glass and steel are retrieved from the MatWeb database, specifically utilising temperature-dependent data. A critical temperature-dependent property of interest in this context is viscosity.

For such temperature-dependent properties, we utilise the full dataset to dynamically update the material properties of the bodies throughout the simulated event. This update is executed volumetrically, meaning it occurs locally within the material. Because the spatial distribution of stiffness directly influences the outcome of contact-based deformation, its volumetric representation is highly important.

To accurately represent the role of heat as an active agent in a shaping action, we must account for its influence on material stiffness. This stiffness—or conversely, its malleability—dictates the specific response of the workpiece under the pressure exerted by tools, human hands, machinery, or self-weight. Implementing a volumetric, per-element representation allows us to model this physical action with a high degree of realism.

The workpiece material is defined via a **`pso:ViscoelasticModel`** (denoted by the *VISCOELASTIC keyword in Abaqus notation) employing the Williams-Landel-Ferry (WLF) equation. This model establishes a link between the material's reference state and its viscosity, dictating that hotter elements require substantially less effort to deform than cooler ones. Based on these given material properties, the reference temperature (T_g) and shift constants (C_1, C_2) are utilised to calculate the dynamic viscosity (η) at any given temperature. The relevant semantic tags for this process are **`pso:ThermalField`** and **`pso:TemperatureDependentViscosity`**.

Refer to Annex D, Table 27 for the equivalent encoding in Abaqus notation

3.3.2 Causing entities

We instantiate the principles from D2.1 in this particular case. The primary entities responsible for driving change within the simulation are identified in Table 1 below.

Table 1. Common causal entities in glassblowing.

Causal Entities	Energy Type	Simulation
Glassblower's Hand/Tool	Kinetic (Motion)	Tool velocity, in soft pulses
Temperature Difference	Thermal (Heat)	Setup of initial temperature fields
Surface Roughness	Dissipative (Friction)	Tangential interaction
Mass / Gravity	Potential (Gravity)	Gravity Load

3.3.3 Affected entities

These entities are modelled in an event-centric manner, as follows.

Workpieces

The workpiece represents the primary entity of interest in the action and, consequently, its simulation. It is purposefully placed, specifically heated, and pressurised, resulting in massive changes to its state-dependent features. These features include its pose, shape, temperature, and localised material properties.

In all shaping, interlocking, and subtractive operations, the workpiece acts as a **`pso:DeformableBody`**. We treat deformable bodies as having spatiotemporally varying material properties that are

volumetrically represented. The mesh of finite elements representing the spatial extent of the workpiece undergoes significant kinematic and geometric changes due to contact interaction. These transformations include thickness redistribution, smoothing, centring, internal expansion, radial expansion, symmetrisation, and rimming.

Tools

The tools utilised in these actions are manually manipulated by practitioners. As discussed in D2.1, tools are treated computationally as rigid objects (**`pso:RigidBody`**), meaning they possess infinite stiffness. Because heat transfer is a primary concern, we also assume that these tools and environmental surfaces possess an infinite heat capacity, acting as perfect heat sinks.

Semantically, tools are "affected entities" because they receive the physical force of the practitioner. However, accurately simulating muscle forces and complex human tool grip is computationally challenging and falls outside our primary scope. Instead, we treat human action on tools phenomenologically, utilising the tool's movement over time (velocity) as a given 'fact,' exactly as it is recorded in ethnographic observations. By defining how the tool moves, the simulation can compute the physical consequences of this movement. The assigned mass and shape of a tool provide sufficient data to compute momentum, which is the exact quantity needed to predict the outcome of the tool's impact with the workpiece.[1]

Kinematics

To maintain intuitive logic and avoid complex vector decomposition in global coordinates, we represent tooling actions within a local coordinate system that is directly aligned with the tool in hand. This ensures that rotational inputs remain semantically linked to the tool itself, regardless of its location and orientation (pose) in the larger global workspace. Velocity is represented in this local coordinate system with translational components (V_1, V_2, V_3) and angular components (VR_1, VR_2, VR_3).

A highly common kinematic state is 'null' motion—occurring when a tool or an element of the environment is stabilised or naturally static, such as the ground plane or a heavy workbench. In the simulation, the object is firmly locked in place by prescribing zero velocity as a boundary condition (**`*BOUNDARY, TYPE=ENCASTRE`**).

Pneumatics

The practitioner's breath is utilised to apply internal pressure inside the cavity of a parison. This is represented numerically as a pressure field, functioning similarly to tool contact but in a much less localised, widely distributed manner. Much like the kinematic inputs, pneumatic load applications are defined using a smooth step amplitude to ensure physical realism and numerical convergence. This approach accounts for the compressibility of air and the finite amount of time required for pressure equilibration within a closed system.

[1] Quantitative measures of the work needed for the given motion can be calculated from the velocity and mass of the objects. Still, mapping these to human anatomy to infer individual muscle effort is beyond the scope of this work.

Amplitudes

To accurately model human-driven processes, we consistently use smooth step amplitudes, profiled as a 0 → 1 → 0 mathematical curve.

- The 0 → 1 upward ramp represents the gradual recruitment of human muscle force required to accelerate the tool.
- The 1 → 0 downward ramp represents the controlled deceleration required to stop the tool precisely without jarring the delicate workpiece.

This profiling is necessary because physical movement is governed by inertia and the biomechanical limits of human muscles. From a technical perspective, if a velocity boundary step jumps from 0 to 1 in an infinitesimal amount of time, the acceleration tends toward infinity, applying an infinite artificial force to the model. This artificial high-frequency pulse manifests in the computation as destructive stress waves, causing spurious mesh distortions. The step amplitude utilises a cubic polynomial spline, ensuring that both the start and the end of the motion possess a derivative of exactly zero. Consequently, the model successfully avoids the artificial numerical oscillations associated with discontinuous mass flow rates.

Figure 1 illustrates the practical consequence of this choice. When a kinematic boundary condition is applied as an abrupt step, the solver injects non-physical high-frequency content into the deformable model, which appears as stress-wave artefacts and spurious surface distortions. Using a smooth 0 → 1 → 0 amplitude suppresses these artefacts and produces a stable, interpretable deformation trajectory.

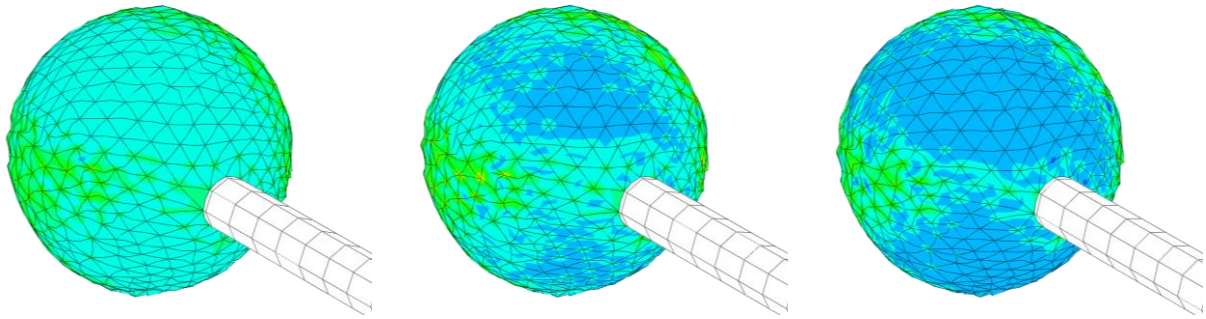


Figure 1. Numerical instability induced by discontinuous kinematic inputs. Three snapshots from the same forming setup show how an abrupt step in a velocity/displacement boundary condition can generate non-physical oscillations (stress-wave artefacts) that corrupt the deformation field, whereas a smooth 0 → 1 → 0 amplitude yields a stable and visually plausible evolution.

3.3.5 Shaping constraints

What is defined in FEM simulation as a spatial “constraint” is, in essence, the physical affordance that guides the practitioner's input effort. The shaping action inherently capitalises on the stability of a workpiece when pressure is exerted upon it. Several key shaping and stabilisation actions rely on the purposeful use of surface contact constraints. The general contact definition algorithm detects when the expanding glass nodes hit the mould surface, subsequently applying a hard contact penalty to prevent unphysical penetration. To simulate this accurately, we must rigorously characterise the contact that

occurs to exert pressure and the resulting interaction. In these interactions, the workpiece is modelled as deformable, while the tools and environment elements are modelled as rigid.

Contact

The physical interaction across tools, the workpiece, and the supporting surface depends entirely on their geometry and spatial arrangement, which determines how the workpiece deforms. General contact is computed dynamically by the FEM solver.

Pressure is applied as a distributed mechanical load across the contact interface. The solver calculates this pressure based on the spatial overlap (penetration) between the rigid tool and the deformable workpiece, effectively translating geometric interference into localised normal forces that push against the glass mesh.

Friction naturally occurs in any contact between surfaces, resisting slipping and greatly affecting the shaping process. When materials are heated, these effects become more significant because both friction and adhesion are fundamentally dependent on temperature; this friction is described as tangential behaviour (Coulomb friction). Surface-to-surface contact applies a mathematical penalty based on the friction coefficient of the surfaces (μ). When a tool slides over or compresses a body, the general contact algorithm adequately models these complex phenomena without needing specialised constraints. This is effectively simulated by the general surface-to-surface contact, using a penalty defined for the specific friction coefficient.

Transmission of motion

Although we have successfully idealised the transmission of human motion directly to a rigid tool, we cannot apply this same idealisation to the deformable workpieces. When a workpiece is manipulated by a tool, the tool's prescribed motion must be physically transmitted to the workpiece. This attachment relies on friction, and depending on the tightness of this contact, it may either allow or restrict slip.

When high heating is involved, we must account for the fact that molten bodies exhibit extremely high friction; they adhere to the blowpipe, rapidly cool, and mechanically lock with it. We simulate this phenomenon via a coupling constraint that rigidly links the bodies, ensuring they rotate and translate seamlessly together. In practical glassblowing, during the gathering phase, molten glass from the furnace is attached to the blowpipe in this exact manner, creating a rigid mechanical lock known colloquially as a 'moil'. This very high friction manifests computationally as a lock that prevents the molten glass from sliding off the blowpipe. Through this lock, the blowpipe effectively transmits its motion to the glass body.

The constraint technically links the two bodies at their conceptual surface of contact. It is crucial to note that linking the surface in this manner does not imply that the workpiece itself becomes rigid. Depending heavily on the applied angular velocity and the workpiece's own inertia, the workpiece may still slip, fall, or become severely twisted internally. The dynamic change of material properties according to heat gradients enables the simulation to accurately represent these differing action outcomes.

Figure 2 shows why we treat motion transmission to deformable workpieces as a constraint problem rather than a rigid attachment. A surface-based coupling can transmit angular motion from the blowpipe while still permitting shear and internal torsion within the viscous body. This distinction is important:

"attached" does not mean "rigid", and the simulation must preserve that separation if it is to reproduce realistic failure modes (slip, fall, twist).

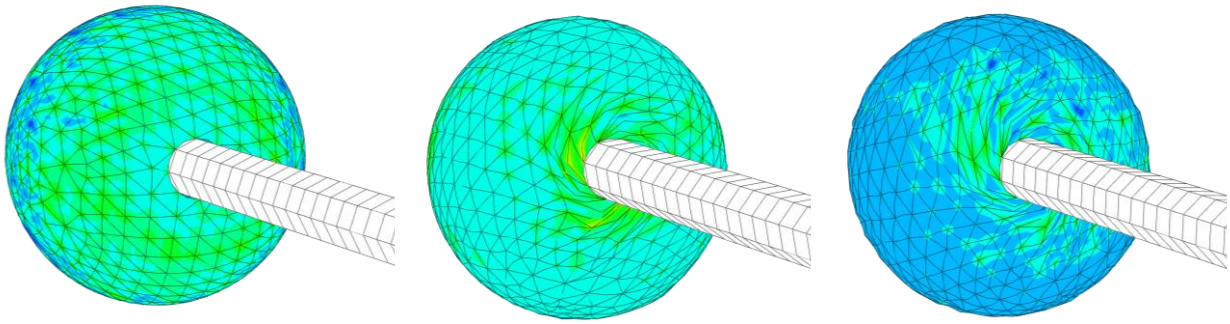


Figure 2. Twisting under rotational coupling. Example outcomes when angular motion is transmitted from the blowpipe to a deformable workpiece via a surface-based constraint. The sequence highlights that the coupling communicates rotation at the interface, but the workpiece can still develop non-rigid torsion and internal shear, producing pronounced twist rather than behaving as a rigidly co-rotating body.

Weight and Gravitational Sagging

While tools and surfaces provide discrete, localised constraints, the self-weight of the workpiece introduces a continuous, omnipresent environmental constraint driven by gravity. Due to the substantial density of soda-lime glass (approximately 2500 kg/m^3), the mass of the workpiece exerts a constant downward body force.

When the glass is in its highly heated, viscous state, its internal yield strength is frequently insufficient to resist this gravitational pull. If held in a static position, the workpiece will inevitably undergo self-weight deformation, sagging or elongating asymmetrically towards the ground. Consequently, gravity acts as a fundamental constraint that dictates the practitioner's kinematics; it forces the glassblower to employ continuous axial rotation (tourné) to counteract the sagging, thereby utilising centrifugal dynamics and constant re-orientation to maintain geometric symmetry.

Computationally, this phenomenon is represented as a global body force, or Gravity Load (**`*DLOAD, GRAV`** in Abaqus), applied to the entire deformable mesh. The FEM solver calculates this downward force per volumetric element based on the material's defined density and the standard gravitational acceleration. By incorporating this volumetric load, the simulation accurately predicts the rate of sagging over time, directly linking the temperature-dependent viscosity of the glass to its structural inability to support its own mass. This allows the model to accurately simulate the necessity of the practitioner's corrective rotational movements.

Figure 3 visualises the "always-on" role of gravity as a global constraint. Unlike tool contact, which is local and event-driven, self-weight acts continuously and couples directly to the temperature-dependent viscosity: as the glass softens, gravitational sag accelerates. In practice, this is exactly the effect that motivates continuous corrective turning by the practitioner.

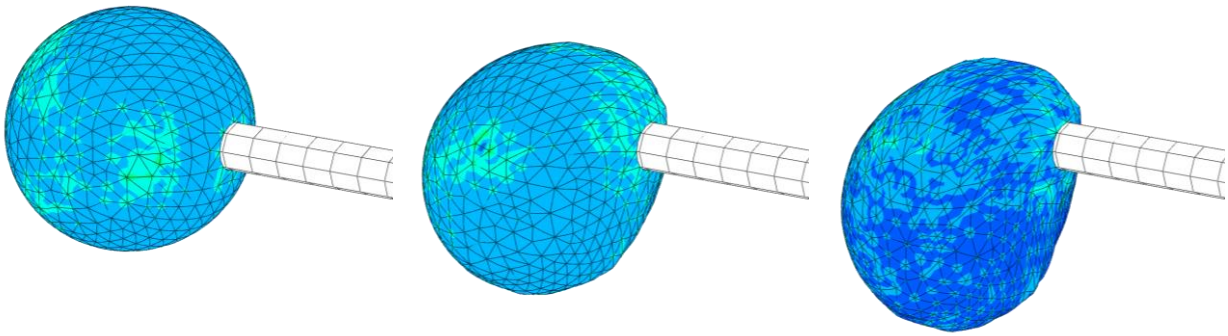


Figure 3. Gravitational sagging is modelled as a global body force. Progressive self-weight deformation of a hot, viscous workpiece under an applied gravity load (e.g., DLOAD, GRAV in Abaqus). The snapshots illustrate how sag develops over time when viscosity is low, motivating the compensatory rotations used in real practice to maintain symmetry.

3.3.5 Effects

The effects of actions are represented as a change of workpiece state. We proceed to identify these changes and select the physical models to approximate their occurrence.

Shape

The physical effects of the practitioner's actions are represented mathematically as a change in the workpiece state. We proceed below to identify these specific changes and select the appropriate physical models to approximate their occurrence.

Shape changes are represented numerically by coordinate mesh updates that physically reflect the new volume occupied by the workpiece. The primary updated variables in this process are the nodal coordinates.

It is important to note that we employ a solid mechanics Finite Element (FE) formulation rather than an Eulerian or computational fluid dynamics (CFD) approach. We acknowledge that we do not work with pure liquids in this simulation; rather, the molten glass is treated as a highly viscous, deformable solid (visco-plastic). This approach enables us to track nodal displacements, severe deformations, and structural integrity without the mesh degradation, which is typical of fluid solvers.

Stress

Internal stress maps precisely how much a volumetric element is being stretched or squashed. Measured in Pascals (Pa), we utilise it volumetrically to assess whether the density of the distortion energy exceeds the material's structural yield strength. Thus, stress analysis is central to predicting shape changes resulting from local pressure.

The FEM solver calculates this stress based on the combined pressure and weight per volumetric element measured. Volumetric representation remains critically important here because the actual yield strength

of the materials depends entirely on their current local temperature. Typically, every geometric change of shape inherently changes the internal stress distribution of a body.

Heat

During any surface contact, thermal energy is transferred between bodies. The simulator accurately calculates the conductive heat transfer (via the ***GAP CONDUCTANCE** feature) and subsequently updates the temperature of each element.

It is necessary to compute this localised heat exchange to model how rigid tools interact with deformable shapes. The rapid thermal drain at the contact patch creates a stiffer, highly viscous 'skin' on the workpiece surface. This chilled skin mechanically resists further deformation, behaving as a dynamic constraint that fundamentally guides the final geometric outcome of the glass.

3.4 Glassblowing negotiations

The glassblowing simulation models glass as the workpiece with temperature-dependent properties evaluated at 900 °C, where the glass is in a molten, formable state. At this temperature, the glass has a viscosity of 10^4 Pa·s and a reduced shear modulus of 10 Pa, allowing large deformations during blowing. The workpiece density is 2500 kg/m³, with a thermal conductivity of 1.6 W/m·K and a specific heat of 1100 J/kg·K. A tabular Equation of State (EOS) is used to model the volumetric response of the glass during contact with forming tools.

3.4.1 Step 1: Marvering

The goal of marvering is to shape the hot glass against a rigid, conductive surface (the marver) and create a cooled, viscous surface (skin) that encapsulates it.

In our case, the parison is manipulated by the blowpipe and constrained by the table. It physically moves and deforms from a bulb into a cylinder. When marvering, the parison is deformed externally when rolled against the marvering table: as the skin freezes, the glass builds up internal tension and resists the deformation. In blowing, the frozen skin acts as a cast that holds the bubble in place, while in breathed-in pressure deforms the glass body internally, thinning its walls.

During surface contact, the glass loses internal energy to the table, which acts as a heat sink. The simulator calculates the conductive heat transfer (*GAP CONDUCTANCE) and the temperature of each element. In our case, the workpiece temperature shifts from hot to cool locally at the surface regions in contact with the marver. In our case, because the workpiece surface cools locally, glass transforms from less to highly viscous, e.g. from honey-like to toffee-like. During marvering, the parison surface becomes significantly stiffer than the core. In turn, this local increase in stiffness influences the expansion in the upcoming blowing step.

Marvering involves a combination of vertical descent, lateral translation, and axial rotation.

- A prescribed motion is applied to the blowpipe.

- Surface-to-surface contact with the penalty method is defined between the blowpipe and workpiece. A surface-to-surface contact is also defined between the workpiece and the marvering table.
- The normal contact is defined as hard contact, and the tangential behaviour uses the penalty method with a coefficient of friction μ = 0.5. The workpiece material is initialised at 900 °C.

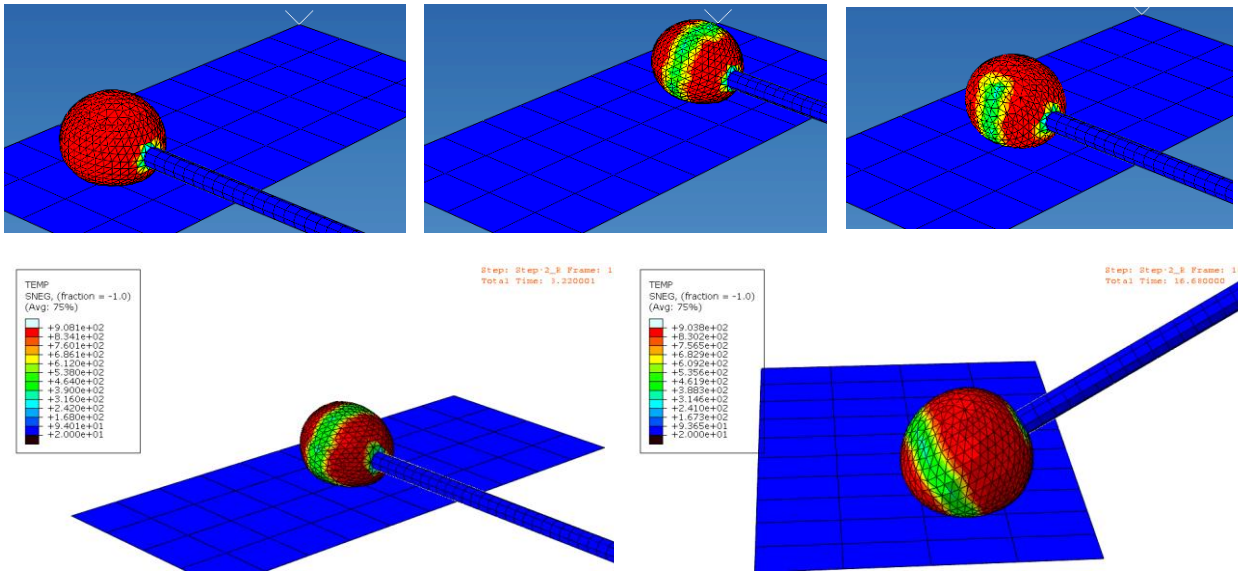


Figure 4. Simulation of the marvering process. Videos for the bottom row: https://youtu.be/hl1aeG36CcE (left) and https://youtu.be/2mL7HN1rZSM (right).

The simulation result is shown in Figure 4 for two inclinations of the blowpipe, one horizontal and one tilted by 30°.

Table 2. Negotiation.

The Glassblower (Cause) inputs motion, and the Marver (Environment) drains heat; the workpiece (Affected) responds by changing its shape and freezing into a solid skin. The Blowpipe Surface acts as both the Engine (driving the motion) and an Anchor (freezing the neck). Heat is transferred across the marver, workpiece, and blowpipe when in contact.		
	Blowpipe	Transmits motion to the glass body. (Tie)
Energy transfer	*Blowpipe*	Extracts heat from the glass body in contact.
	Marver	Extracts heat from the glass body in contact.
Tool(s)	*Blowpipe*	Movement as prescribed by velocity
Contact	*Marver*	Forces glass to flatten
Change	*Shape*	Parison to cylindrical shape.

	Viscosity	Unevenly chilled body, localised on the workpiece surface (skin).

Heat

Marvering cools the workpiece's surface. Heat transfer occurs between the viscous hot glass and the steel marver, which is a high-conductivity heat sink. Semantically, it represents a body whose temperature does not rise significantly despite the contact. Due to the temperature-dependent viscosity of glass, the outer layer of the parison becomes stiffer than the inner layer. The simulation involves two thermal agents draining energy from the glass via conductance, namely the marver and the blowpipe.

Motion

A trapezoidal motion height path of the tilted tool brings the revolving glob transiently in contact with the marver. Velocity is decomposed across axes V_1 and V_3 in the local coordinate system. During contact, the pipe is tilted at an angle $\phi = 30°$, influencing the workpiece's shape change. Axial rotation (VR_1) is coupled with lateral velocity (V_2). The roll is continuous and about the blowpipe axis. Ideally, the angular velocity should match the horizontal translation across the table, so that the workpiece does not slip and sustain unwanted deformations.

Contact

This geometric transformation is governed by the mechanical disparity between the two contacting bodies. The steel marver is idealised to possess infinite stiffness compared to the viscoplastic glass. This simulates the physical, geometric and mechanical constraints that force the glass mesh to displace and flatten upon impact.

Table 3. Abaqus mapping.

Event	Implementation	Role
Motion Transmission	`*TIE, *KINEMATIC COUPLING`	Locks the neck to the blowpipe so the pipe's motion/rotation drives the workpiece.
Conductive Cooling	`*SURFACE INTERACTION, *GAP CONDUCTANCE`	Heat flux based on contact pressure. Coupled temperature–displacement analysis.
Kinematic Drive	`*BOUNDARY, TYPE=VELOCITY`	Prescribes tool motion.
Marver Contact	`*CONTACT PAIR / *SURFACE`	Shapes the workpiece.

In Annex D, Table 28 shows the encoding in Abaqus notation.

3.4.2 Step 2: Blowing

The goal is to expand the parison volume using internal air pressure, relying on the "skin" created in Step 1 to control the shape. The glassblower blows air through the blowpipe to create an air bubble inside the molten glass. The glassblower blows into the hole in the pipe, then caps the hole with their thumb. This pressurisation of the air pushes the air into the soft molten glass, and the air expands inside the glass, creating a cavity in the glass.

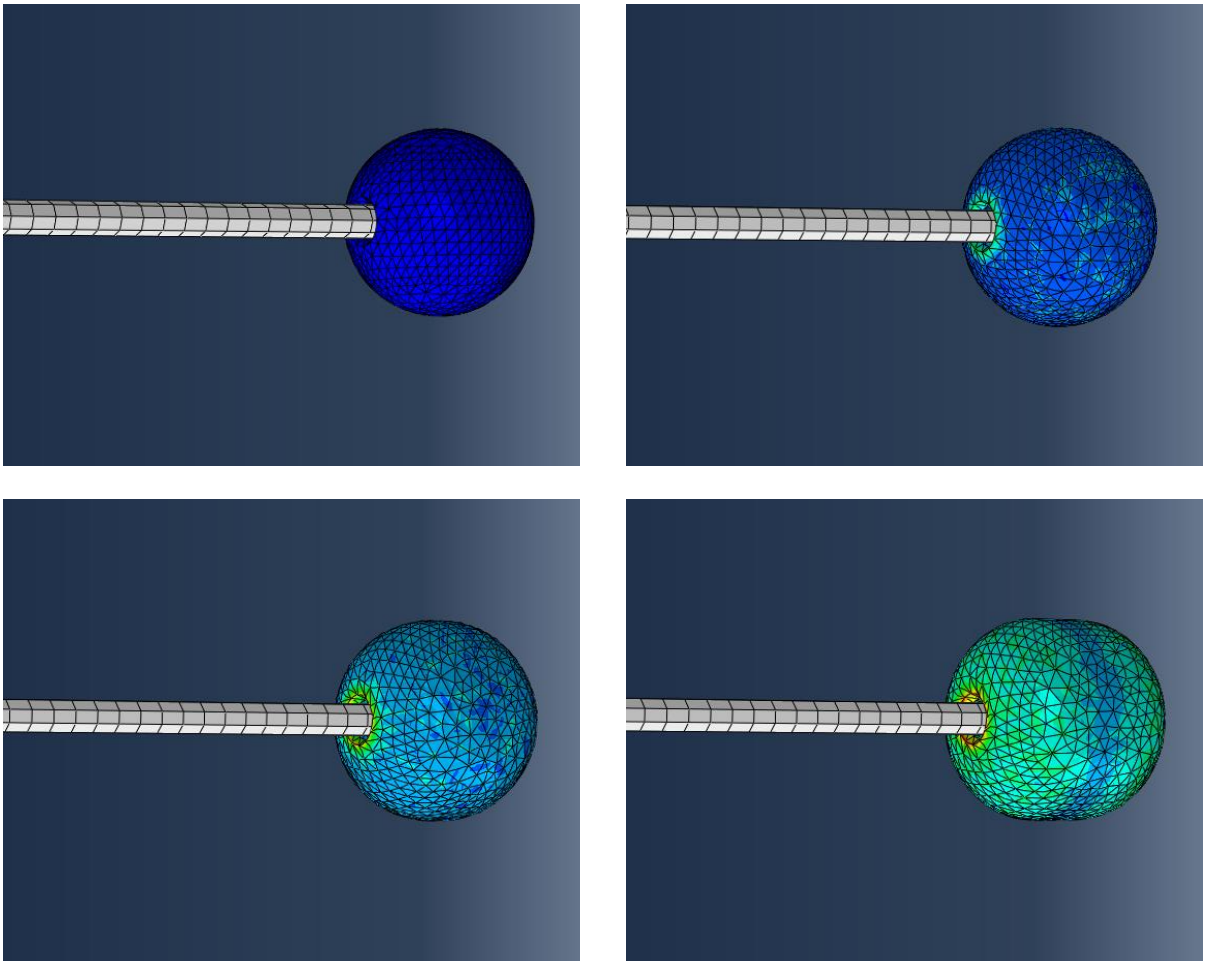


Figure 5. Air is introduced through the blowpipe using the blow-and-cap technique. A brief burst of air is blown into the pipe and immediately capped, trapping and pressurising the air. The pressurised air is forced into the still-molten glass, where it gradually expands to form an internal cavity.

Table 4. Negotiation.

The Glassblower (Cause) inputs pneumatic pressure, and the Viscous Skin (Constraint) guides the expansion; the Workpiece (Affected) responds by inflating its volume and thinning its walls.		
Energy transfer	*Pneumatic pressure*	Drives the expansion.
Tool	*Blowpipe*	Anchors the glass neck.

	Blowpipe	Transmits pressure.
Contact	*Workpiece (self)*	Viscosity gradient (cooler outer skin vs hotter core) governs where stretching concentrates and how wall thickness redistributes.
Change	*Shape*	inflation and wall thinning.

Motion

In blowing, the blowpipe becomes an anchor as the glassblower holds the pipe steady to blow. The pipe is locked, ensuring the neck of the attached glass remains fixed in space while the body expands.

Pressure

Pneumatic deformation transforms the workpiece from a solid parison into an expanding hollow shell. The air blown inside the pipe acts as a compressible fluid. The internal pressure load represents the force exerted by the gas on the internal boundaries of the viscoplastic container.

Table 5. Abaqus mapping.

Phenomenon	Abaqus Implementation	Role
Pneumatic Pressure	`*DSLOAD`	A follower load that remains normal to the surface as it expands.
Static Anchor	`*BOUNDARY, TYPE=ENCASTRE`	Keeps the blowpipe static during expansion.

In Annex D, Table 29 shows the encoding in Abaqus notation.

3.4.3 Step 3: Soufflé Tourné

The goal is to expand the parison volume using internal air pressure. The expansion is constrained externally by the mould, while rotation redistributes material and stabilises symmetry.

The Soufflé Tourné (blown and turned) is a cooperative action that uses a mould to precisely control the final shape, thickness and surface texture of the workpiece. The glassblower rotates the suspended blowpipe. The workpiece neck is tied to the blowpipe via a coupling constraint. This ensures the neck of the glass remains fixed in space while the body expands.

When the gob is ready, the glassblower places his rod vertically above the open mould, and the workpiece (hot glass) slowly stretches by gravity. It is at this point that the glassblower introduces the elongated gob into the mould, doing so slowly with a very slow rotation. The blowpipe is never inserted into the mould, but remains outside it. The assistant closes the mould in the same motion, holding it tightly while the glassblower slowly rotates and blows into the blowpipe. The purpose of this action is to expand the glass

in the mould, while simultaneously turning the glass so that it conforms to the shape of the mould and gradually smoothes its surface by friction. At the same time, the glass is refined internally.

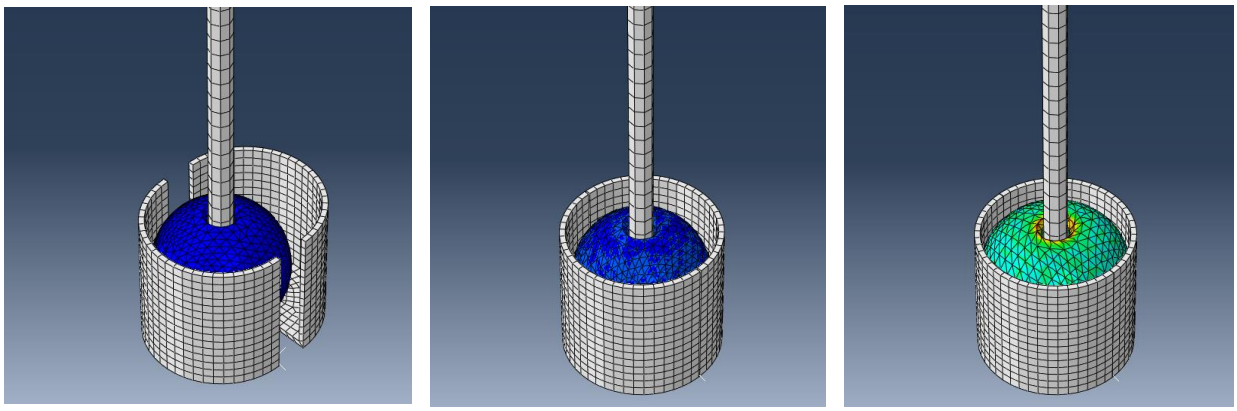


Figure 6: Soufflé tourné - expansion and shaping of the workpiece in the mould.

Table 6. Negotiation.

The Glassblower (Cause) inputs pneumatic pressure and rotation; the Mould (Constraint) defines the geometric limit; the Workpiece (Affected) responds by inflating and conforming to the cavity.		
Energy transfer	*Blowpipe*	Transmits torque to the glass neck.
	Pneumatic pressure	Inflates the glass body.
Tool(s)	*Blowpipe*	Movement as prescribed by velocity.
	Blowpipe	Pressure load as prescribed.
Contact	*Mould cavity*	Constrains the expansion of the glass body.
Changes	*Shape*	Wall thinning and mould surface replication.

Motion

Controlled rotation prevents gravity from sagging the hot glass to one side. The blowpipe is constrained in translation ($U_1=U_2=U_3=0$) but driven in rotation ($UR_3 = \omega$).

Pressure

The glassblower blows, then seals the hole with their thumb. Air is forced in, trapped, and then expands as it heats up/equalizes within the hot glass. Pressure represents the trapped air expanding against the soft glass.

Contact

The general contact definition detects when the expanding glass nodes hit the mould surface, applying a hard contact penalty to prevent penetration.

Table 7. Abaqus Mapping.

Phenomenon	Abaqus Implementation	Role
Pneumatic Pressure	`*DSLOAD`	Internal load expands the mesh.
Kinematic Drive	`*BOUNDARY, TYPE=VELOCITY`	Prescribed angular velocity.
Motion Transmission	`*TIE, *KINEMATIC COUPLING`	Kinematic coupling locks the neck to the blowpipe so the pipe motion drives the workpiece.
Mould Contact	`*CONTACT PAIR / *SURFACE`	Contact stops the expansion at the mould walls.

In Annex D, Table 30 shows the encoding in Abaqus notation.

3.4.4 Step 4: Blocking (Shaping)

The goal is to shape and centre the workpiece using a water-soaked wooden block while maintaining internal pressure to prevent collapse. This step also cools the surface of the workpiece and homogenises the quantity of glass in the axis of the blowpipe, to obtain symmetrical and centred objects.

The glassblower instils pressure into the glass by blowing. This pressurisation pushes the air into the soft workpiece, resisting the outer pressure of the block.

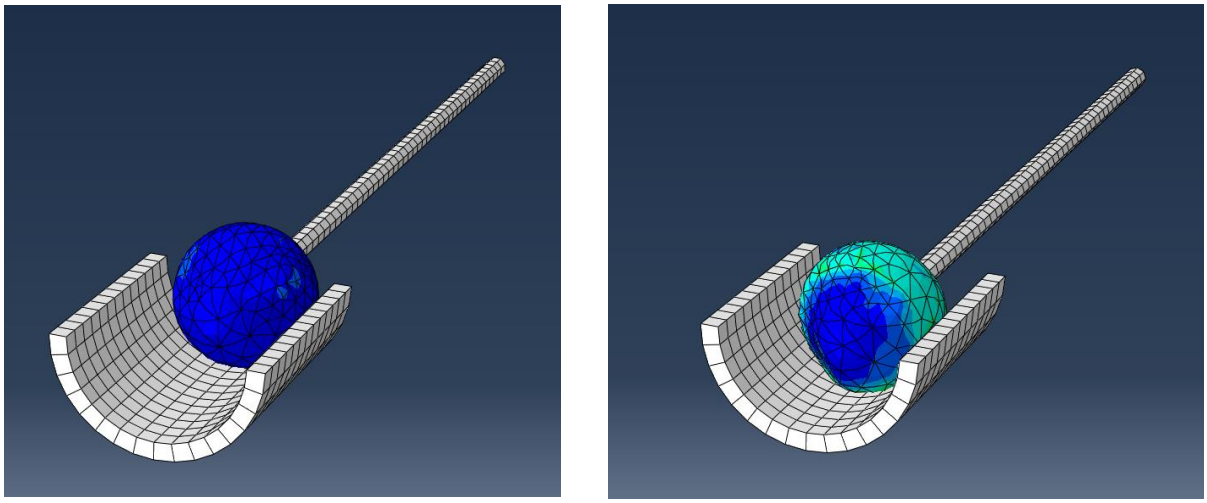


Figure 7: Simulation of the workpiece shaping with a rigid block.

Table 8. Negotiation.

The Glassblower (Cause) inputs rotation and pneumatic pressure; the Wet Block (Constraint) cools and shapes the surface; the Workpiece (Affected) responds by centring its mass and smoothing its skin.		
Energy transfer	*Blowpipe*	Transmits torque to the glass neck.
	Breath	Maintains internal support to resist block pressure and prevent collapse (stabilises during contact).
	Wet block	Extracts heat.
Tool(s)	*Blowpipe*	Movement as prescribed by velocity
Contact	*Block*	The water-soaked block constrains inflation.
Changes	*Shape*	Symmetrisation, surface replication, and smoothing.

Motion

The block is a curved tool used to centre the glass. The glassblower rotates the pipe back and forth on a metal rail while pressing the glass into the water-soaked block.

Pressure

The glassblower blows, then seals the hole with their thumb. Air is forced in, trapped, and then expands as it heats up/equalizes within the hot glass. Pressure represents the trapped air expanding against the soft glass.

Contact

Surface-to-surface contact with a penalty defines the interaction. The friction coefficient here is critical, as it represents the lubricated contact due to the steam cushion, not dry wood. Fluid Cavity: Modelled as a Fluid Cavity with a closed volume

Table 9. Abaqus mapping.

Phenomenon	Abaqus Implementation	Role
Block Contact	`*CONTACT PAIR, PENALTY`	Soft contact represents the steam layer.
Pneumatic Pressure	`*DSLOAD`	A follower load that remains normal to the surface as it expands.

In Annex D, Table 31 shows the encoding in Abaqus notation.

3.4.5 Step 5: Necking

The goal is to narrow the workpiece diameter at a specific location (usually the neck), using localised compression and rotation. The action narrows the glass at the base near the blowpipe, causing the surrounding material to elongate and form a more suitable shape for moulding. The jacks apply localised compressive forces that narrow the workpiece at the contact area while the surrounding material elongates axially. Simultaneous rotation ensures that the necking occurs symmetrically around the blowpipe axis.

Table 10. Negotiation.

The Glassblower (Cause) inputs compressive force and rotation; the Jacks (Constraint) apply localised pressure; the Workpiece (Affected) responds by narrowing radially and elongating axially.		
Energy transfer	Blowpipe	Transmits motion to the glass body.
	Jacks	Compression deforms the workpiece.
Tool(s)	Blowpipe	Movement as prescribed by velocity.
	Jacks	Movement as prescribed by velocity.
Contact	Jacks	The tool motion deforms the workpiece.
Changes	Shape change	Radial constriction and axial elongation.

Contact

Jacks are precision tools that apply force to a specific ring of the material. The jacks apply localised compressive forces that narrow the workpiece at the contact area. As the radius decreases, the volume of glass shifts.

Table 11. Abaqus Mapping.

Phenomenon	Abaqus Implementation	Role
Dynamic Inertia	`*DYNAMIC, EXPLICIT`	Captures rotational forces and complex contact evolution.
Jacks motion	`*RIGID BODY, *BOUNDARY`	Constricts the vessel neck.
Attachment	`*KINEMATIC COUPLING`	Locks the glass motion to the pipe.

In Annex D, Table 32 shows the encoding in Abaqus notation.

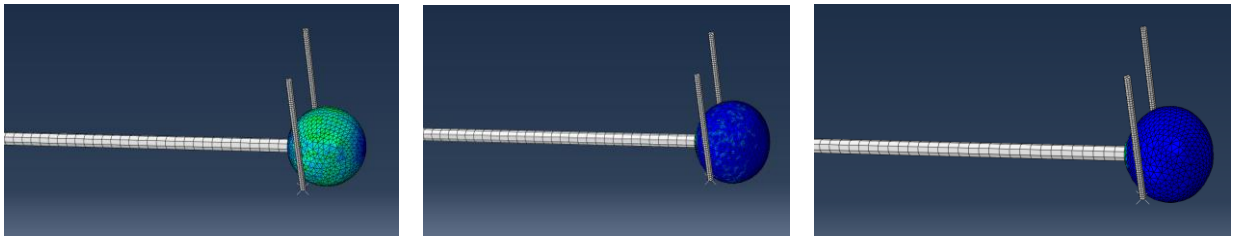


Figure 8: Constriction of the workpiece by jacks.

3.4.6 Step 6: Rim Forming

The goal is to flare open the rim of the vessel using jacks and rotation, transforming a cylindrical opening into a wider, controlled shape.

Surface-to-surface contact with the penalty method is defined between the punty and the workpiece, with a tangential friction coefficient of $\mu = 0.5$ to transmit forces and allow controlled shaping.

During this step, the punty applies localised compressive forces to open the lips of the vessel while the workpiece rotates, aligning the opening with the axis of the punty. The combined effects allow shaping and formation of the open lip.

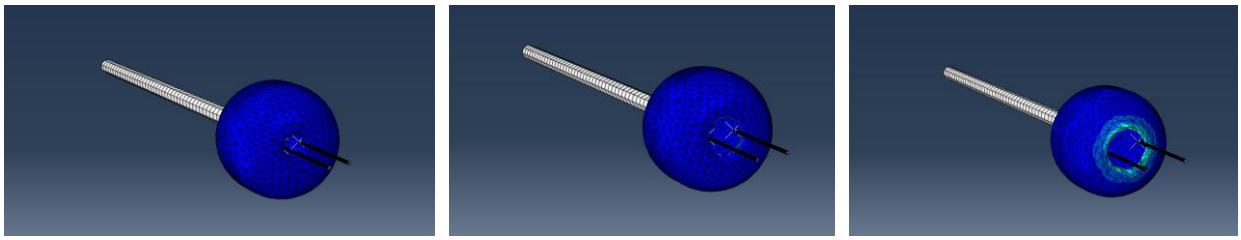


Figure 9: Opening the lips from the workpiece.

Table 12. Negotiation.

The Glassblower (Cause) inputs rotational torque and outward mechanical pressure; the Jacks (Constraint) guide the expansion; the Workpiece (Affected) responds by flaring radially while maintaining symmetry.		
Energy transfer	Punty	Transmits torque to the glass neck.
	Jacks	Compression deforms the workpiece radially outwards.
Tool(s)	Punty	Movement as prescribed by velocity.
	Jacks	Movement as prescribed by velocity.
Contact	Punty	Holds the base of the vessel.

Changes	Shape	Rim flaring.
	Stress	Stretching and inertia alter body stiffness.

Motion

Controlled rotation prevents gravity from sagging the hot glass to one side. The blowpipe is constrained in translation (U_1=U_2=U_3=0) but driven in rotation (UR_3 = ω).

Pressure

The jacks apply localised compressive forces to open the lips while the workpiece rotates.

Table 13. Abaqus Mapping.

Phenomenon	Abaqus Implementation	Role
Rotation	`*BOUNDARY, TYPE=VELOCITY`	Prescribed punty rotation for alignment.
Spreading	`*BOUNDARY, TYPE=DISPLACEMENT`	Jacks move outward to expand the rim.
Tool Grip	`*FRICTION, 0.5`	High friction transmits the shaping force.
Transfer	`*KINEMATIC COUPLING`	Coupling transferred from neck-blowpipe to base-punty.
Jacks contact	`*CONTACT, *SURFACE INTERACTION`	Enables rim shaping; friction acts tangentially.

In Annex D, Table 33 shows the encoding in Abaqus notation.

The outcome of this glassblowing process is a vessel, exhibiting a smooth profile and uniform wall thickness.

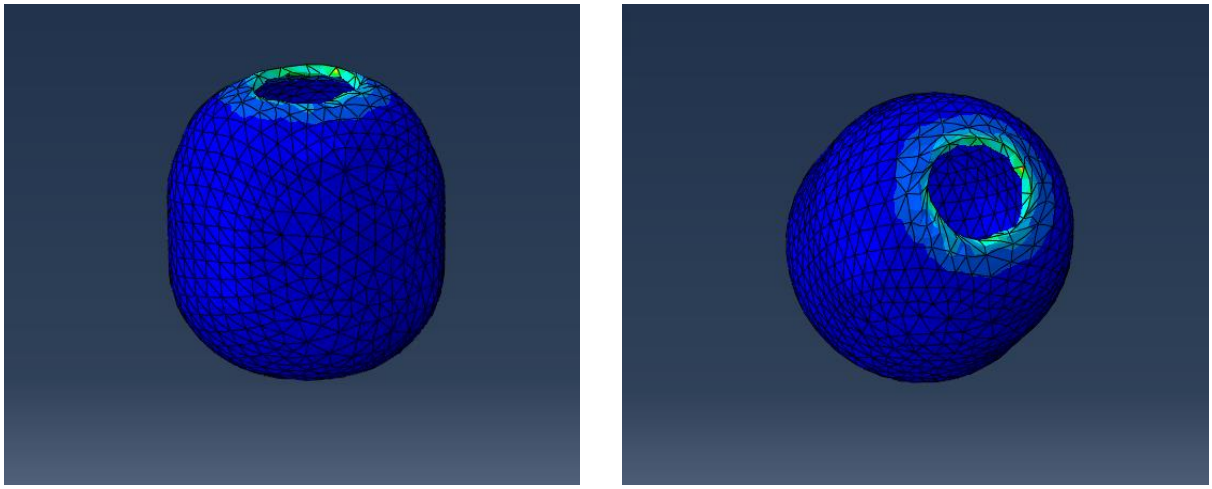


Figure 10: Final vessel.

3.4.7 Auxiliary simulations

Furnace door opening and closing

This simulation models the opening and closing of the glass furnace. The door is constrained to move along the defined opening direction, representing the real mechanical motion used to access the molten glass inside the furnace.

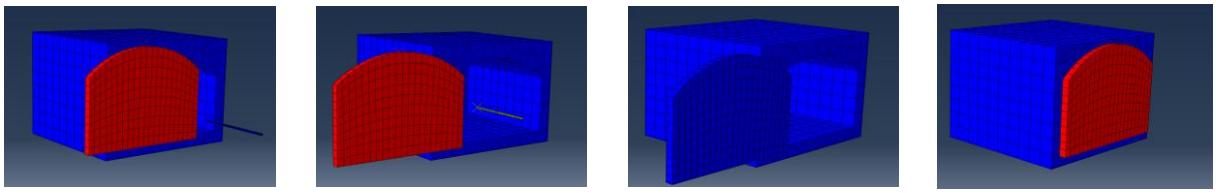


Figure 11. Glass furnace door (leftmost image pair) opening and closing simulations (rightmost image pair).

Similarly, once the workpiece has been shaped in the mould, the assistant releases the mould to remove the workpiece, which remains attached to the blowpipe.

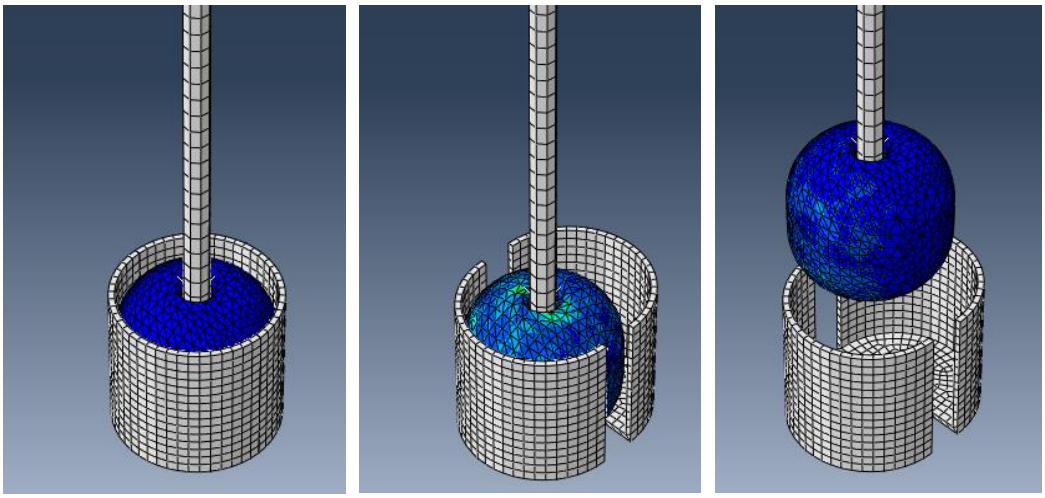


Figure 12: Open the mould to release the workpiece.

Blowpipe and workpiece attachment

The simulations include the extraction of the blowpipe and workpiece from the furnace. The workpiece is deformable, while the blowpipe remains rigid. Surface-to-surface contact with a high penalty is defined between the blowpipe and workpiece. The blowpipe is subjected to prescribed translation and rotation, driving the extraction of the workpiece, while contact prevents penetration and ensures realistic sliding behaviour.

The gathered glass workpiece remains attached to the rigid blowpipe during extraction due to contact friction between the glass and the blowpipe surfaces, preventing the workpiece from slipping. Still, the simulation can illustrate that abrupt motions may unintentionally deform the parison.

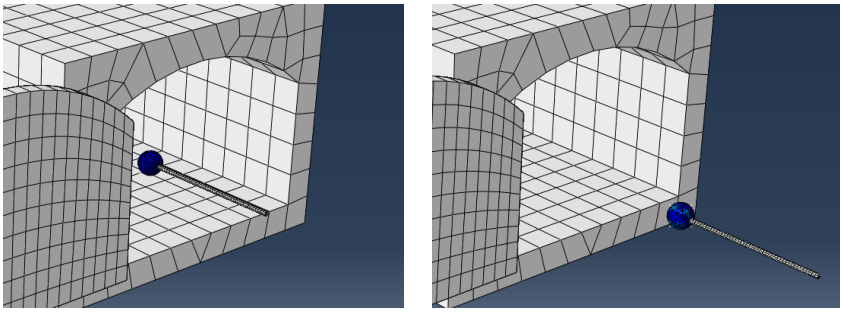


Figure 13: Extraction of the blowpipe and workpiece from the glass furnace.

A simulation model of the turning of the workpiece on a metal bar, when blocking or necking in the top row of the figure below. A surface-to-surface contact is also defined between the workpiece and the metal bar. The contact is defined as a hard contact, and the tangential behaviour uses the penalty method. Another simulation shown in the figure below (bottom row) models the transfer of the workpiece from the blowpipe to the punty. Surface-to-surface contact with friction defines the interaction between the workpiece and punty. Similarly, the interaction with the blowpipe is modelled using a penalty-based contact formulation. The movements of the punty and the blowpipe are prescribed.

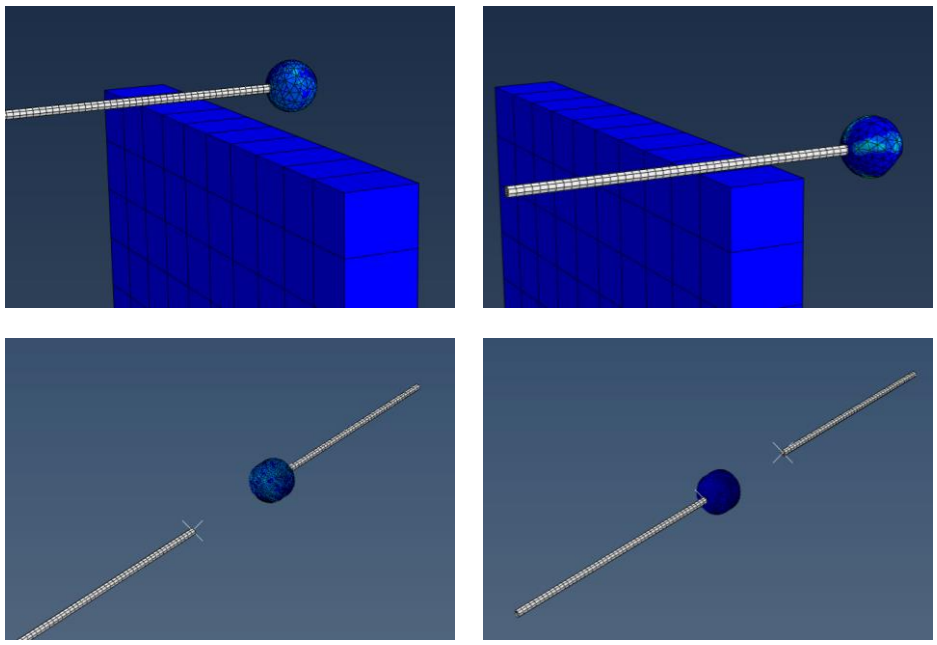


Figure 14: Top: A workpiece turning on a metal rig. Bottom: Detachment of the workpiece from the blowpipe and attachment to the punty.

3.5 Conclusion

This section establishes semantic simulation as a practical bridge between (i) the craft-level description of shaping actions and (ii) the solver-level specification of boundary conditions, constitutive laws, and contact. The core idea is semantic grounding: every operational element in the simulation file (e.g., temperature fields, viscosity maps, pressures, kinematic constraints) is linked to an explicit domain concept (via PSO-style tags), so that the model is preserved and simulation steps can be composed without ambiguity. In the glassblowing example, this enables a clean, machine-actionable chain where marvering yields a temperature field, which determines a viscosity map, which in turn governs the deformation response under blowing pressure.

We then framed glassblowing as a sequence of maker–material negotiations driven by a small set of reusable mechanisms: (a) temperature-conditioned stiffness/viscosity updated volumetrically, (b) contact and frictional interaction with tools and supports, (c) motion transmission that constrains without "rigidly attaching", (d) internal pressure as a pneumatic cause, and (e) gravity as a continuous global load that couples strongly to viscosity. Several modelling practices are not merely "implementation details" but semantically meaningful commitments—for example, using smooth 0 → 1 → 0 amplitudes to represent human force recruitment while avoiding non-physical solver artefacts, and treating attachment/transfer events (blowpipe ↔ punty) as changes in constraint structure rather than ad hoc geometry edits.

Taken together, the outcome of Section 3 is a modular, interpretable template for describing shaping actions: identify causing entities, specify affected entities and fields, state constraints and couplings, and record effects (shape, stress, heat) in a way that can be traced across steps. This semantic layer does not replace FEM; rather, it makes FEM configurations composable and reusable, and prepares the ground for

the subsequent sections where these grounded mechanisms are (i) exposed through interactive tooling and (ii) accelerated through surrogate modelling while retaining a clear link back to the underlying physics.

4 FEM simulations

Finite Element Method (FEM) simulations offer a significantly higher degree of accuracy and descriptive fidelity compared to standard Physics-Based Simulators (PBSs). However, full multiphysics FEM solvers cannot achieve real-time interactivity due to their immense computational overhead. Nevertheless, in complex scenarios where multiple physical phenomena interact, FEM remains the only viable means of accurate simulation. Such scenarios include temperature-dependent plasticity, severe material damage, and non-linear deformation.

To overcome this limitation and create highly realistic, interactive simulations of specific craft actions, we implemented a hybrid methodology parameterised by the action variables of interest. Within the Craeft framework, we employ two complementary mechanisms to satisfy real-time constraints:

1. **Interactive Real-Time PBS:** For many training scenarios, macroscopic deformation is computed directly via PhysX soft-body dynamics. By utilising a deliberately reduced set of material parameters, the simulation remains interactive, driven primarily by tool-applied forces, and outputs deformed meshes that are visualised instantly in the Unity engine.
2. **Offline FEM Ground Truth & Surrogate Learning:** When a craft phenomenon requires higher-fidelity constitutive behaviour than the real-time engine can natively support, we utilise offline multiphysics FEM (Simulia Abaqus) to generate a sparse "ground truth" dataset. We then train a lightweight machine learning surrogate that maps execution parameters to deformation-relevant quantities; these surrogate outputs subsequently parameterise or dynamically correct the real-time PhysX interaction at runtime.

The fundamental challenge in craft-specific simulation is computational time. A high-fidelity Abaqus simulation can require several hours to resolve the contact mechanics of a single tool strike, rendering it useless for an interactive learning environment. While previous iterations (D2.1) explored precomputing and storing every possible outcome in a vast database for "playback," this proved unscalable due to prohibitive storage and memory bandwidth requirements.

Instead, we leverage machine learning. By sparsely sampling the parameter space with high-fidelity FEM simulations, we generate a highly structured training dataset. This data trains a neural network that either directly approximates the geometric deformation or predicts the optimal real-time material parameters compliant with the FEM results.

4.1 Training data

Training data is systematically produced using the Simulia Abaqus solver, strictly constrained to the material properties relevant to the specific craft action. We present two primary cases illustrating this data production pipeline: structural contact interaction and thermal conditioning.

4.1.1 Contact Interaction and Tool Taxonomy

The relevant parameter space for contact interaction is defined by tool speed, angle of incidence, and material properties. A carefully selected, sparse subset of these parameters is simulated to enable the

downstream neural network to generalise effectively across untrained continuous variables. Each FEM simulation captures the stress distributions and geometrical deformations resulting from the tool-workpiece interaction, which are then labelled and structured as paired input-output training sets.

Different actions utilise different tool types because the geometry and application of a tool produce drastically different mechanical outcomes. We taxonomised (using D2.1) tool interactions to select the appropriate physical models for each type of action and simplify the downstream machine learning architecture.

- **Cutting Tools:** Defined by the sharpness of a linear edge. They achieve bifurcation or subtraction by concentrating stress along a narrow border to exceed the material's shear strength. Longitudinal motion results in incisions, while transverse motion results in paring or chip removal.
- **Forming Tools:** Characterised by radiused or hemispherical tips that facilitate plastic deformation. Lacking a severing edge, they force the material to flow and reshape through burnishing, indenting, or compressing without a loss of overall mass. Because they utilise point-contact interactions, their axial orientation is often negligible compared to a blade.

The data generated is used as training data through a structured and methodical process. First, the relevant parameter space is defined, using tool speed, incidence angle, and material properties. A representative subset of these parameters is for the neural network to generalise effectively across different conditions. This selection is essential for enabling the neural network to generalise effectively across different conditions.

Table 14 summarises the mechanical and geometric distinctions between tools designed to sever material versus those designed to reshape it.

Table 14. Comparison of Cutting vs. Forming Action and Mechanics

Action	Cutting Tool	Forming Tool
Mechanics	Shear (slicing)	Yield (bending/compression)
Material failure	Exceeds shear strength to break the workpiece.	Exceeds the yield strength to deform the workpiece.
Contact geometry	Line	Point
Orientation	Critical	Negligible
Motion	1. Longitudinal: incising 2. Transverse: paring, chipping	1. Lateral: burnishing, rubbing 2. Vertical: punching, indenting
Result	Separation (Subtractive)	Displacement (Mass-preserving)
Design	Sharp for penetration	Blunt for pressure distribution

To capture these mechanics, we simulated driving blunt forming tools (e.g., a nail or punch) into a wooden body at incidence angles of 0°, 30°, 45°, and 60° (Figures 14 and 15). Conversely, cutting actions were simulated using a sharp chisel executing transverse and longitudinal motions on metallic and wooden surfaces. To construct training data for contact-driven deformation, we systematically vary the incidence angle of the tool while keeping the workpiece geometry and boundary conditions fixed. This produces paired "before/after" states that can be labelled by execution parameters (here: angle of incidence) and used either directly for surrogate learning or for calibrating reduced real-time models (Figure 15 and Figure 16).

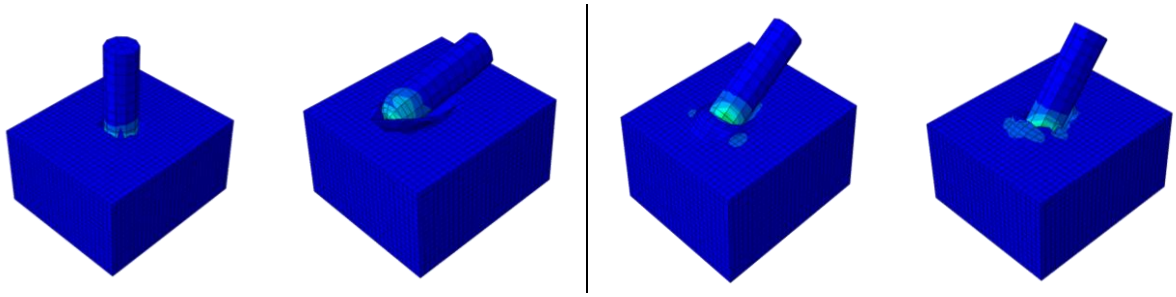


Figure 15. Parametric FEM training samples for tool-workpiece impact at two incidence angles (0° and 30°), shown as two image pairs. For each image pair, the left image shows the initial configuration immediately before contact, and the right image shows the resulting state after impact, illustrating how tool orientation differentiates deformation.

Because these simulators ultimately support training and communication, FEM outputs must be translated into a visual form that is interpretable to learners and credible to experts. We therefore couple FEM results with a rendering pipeline, enabling side-by-side comparison between engineering "ground truth" and the visual feedback used in interactive or explanatory applications.

Beyond continuous sweeps over angle, we also capture discrete "action segment variants" that correspond to qualitatively distinct executions of the same nominal action. In nail insertion, the distinction between a tilted and a perpendicular strike is pedagogically meaningful because it leads to different contact regimes and shows the results of erroneous execution (see Figure 16).

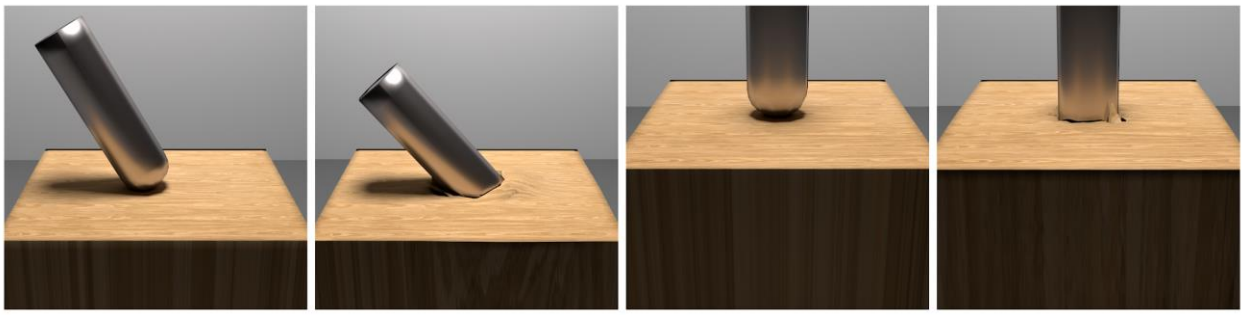


Figure 16. Two variants of a nail insertion action segment, shown before and after a mallet strike. Left: tilted insertion; right: perpendicular insertion. The paired states exemplify how execution variant (tool axis relative to the workpiece) produces measurably different outcomes for the same task.

Cutting

For cutting mechanics, the governing variables are not only orientation and motion direction, but also the effective applied load (or pressure) during the stroke. We therefore generate datasets that hold tool geometry and motion type constant while varying the intensity of interaction, producing a controlled set of outcomes spanning shallow to more aggressive material removal (see Figure 17).

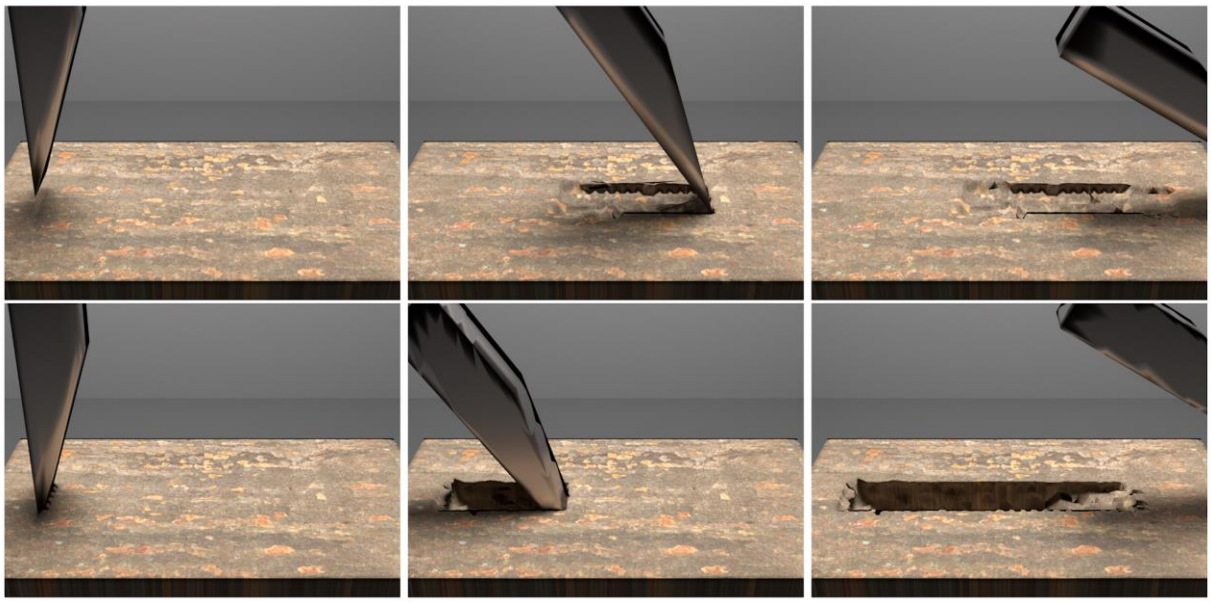


Figure 17. FEM-generated outcomes for a transverse wood-carving action executed with a chisel under varying pressure levels. The sequence illustrates the progression of deformation and material removal as interaction intensity increases, providing labelled samples for learning or calibrating cutting-related parameters.

Figure 18 shows the difference in the type of cut induced by swapping the orientation of the tool from longitudinal to transverse. The scraping action becomes an incision.

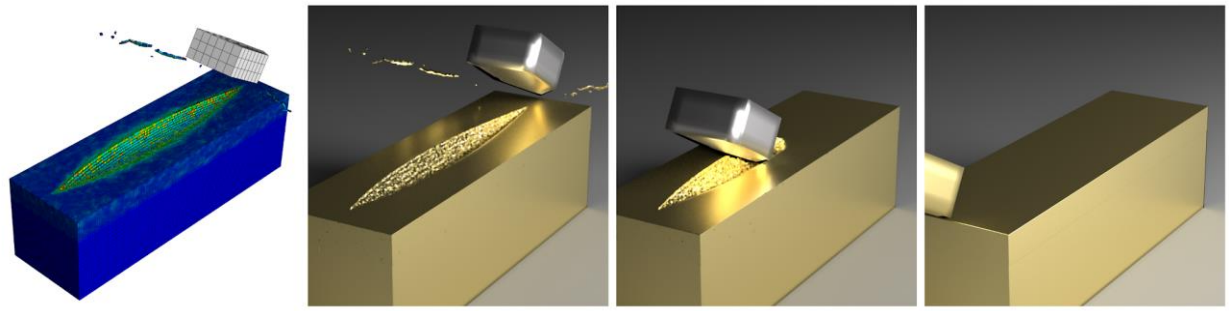


Figure 18. Comparison between an FEM simulation (rightmost) and its rendered visualisation for a longitudinal engraving action on a metallic surface (three leftmost images). Left: FEM result; right: rendered views of the corresponding deformation, demonstrating the translation from high-fidelity simulation output to training-oriented visual feedback.

4.1.2 Thermal conditioning

To investigate how temperature alters stress and pressure distribution during a forming action, we produced temperature-dependent datasets. The scene consists of a rigid tool applying mechanical pressure to an aluminium workpiece supported by a ground plane. The workpiece is subjected to two displacement levels (1 mm and 1.5 mm) across three distinct temperatures (20°C, 300°C, and 660°C). The element type utilised was C3D8R (an 8-node linear brick with reduced integration).

The simulation confirms a profound relationship between thermal conditioning and stress. The temperature increases to 660°C, and the stress values decrease significantly. Concurrently, the pressure distribution becomes highly uniform and widespread across the surface. This demonstrates the rapid reduction in yield stress and the increased viscoplasticity of the material at elevated temperatures, proving that thermal variables must be a core input in the training dataset.

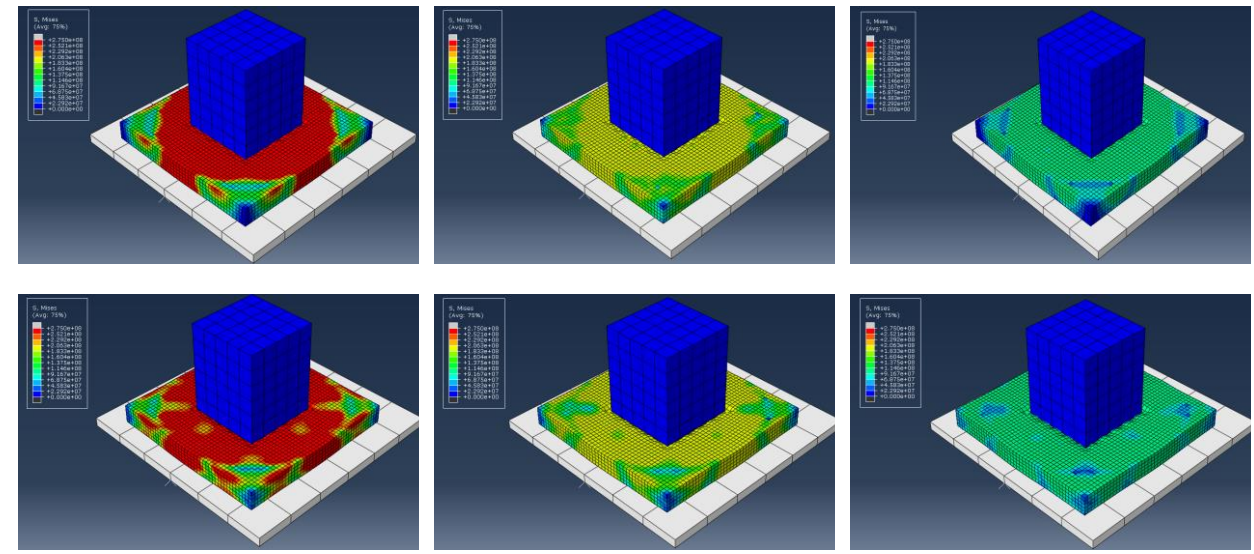


Figure 19. Stress spatial distribution on the workpiece at three different temperatures: 20°C (left), 300°C (middle), and 660°C (right), for two displacements of 1 mm (top) and 1.5 mm.

Figure 19 shows the six resultant states of the tool and workpiece. The colour bar at the top left of individual plots indicates the stress values, with higher stresses shown by colours close to red and lower stresses indicated by other colours. As the temperature increases, the colour of the workpiece shifts towards green, indicating that the stress within the aluminium decreases at higher temperatures. The pressure distribution is significantly more uniform and spread out across the surface. The workpiece shows an even greater reduction in pressure values, with a predominance of light blue colour. This indicates a decrease in pressure intensity, as the material deforms more easily under the applied load. The details of this experiment are reported in Annex B.

4.1.3 Surrogate Modelling and Parameter Mapping

The fundamental output of the Abaqus simulations comprises millions of nodal displacement vectors, stress tensors, and thermal gradients. This is far too dense for a real-time game engine to process. Therefore, we utilise this ground-truth data to train an intermediary machine learning surrogate model.

A lightweight statistical model is trained on this structured and labelled dataset to interpolate outcomes within the covered parameter ranges. In the current toolbox, such models are primarily used to calibrate PhysX material/solver parameters against FEM-derived targets, while PhysX remains responsible for real-time collision handling and deformation.

For 3DNS, we select a radially symmetrical brush template based on a smooth-step function. By modulating stroke intensity, the method can reproduce ground-truth deformations for a range of applied force levels.

Rather than pre-computing the full parameter space, we generate training data by conducting a sparse number of FEM simulations covering representative combinations of execution parameters and material properties (such as tool speed, angle of incidence, and stiffness).

This surrogate acts as the critical bridge mapping the theoretical physics to the interactive environment. The process operates as follows:

1. **Feature Extraction:** The execution parameters controlled by the user (e.g., tool vector, incidence angle, applied force, and localised temperature) are extracted as the *Input Features*.
2. **Surrogate Training:** A lightweight neural network (the 3DNS architecture) is trained to learn the non-linear relationship between these sparse input features and the resulting high-fidelity FEM deformation profiles.
3. **Real-Time Parameterisation:** At runtime in the interactive application, when a user strikes a virtual workpiece, the system does not solve complex differential equations. Instead, it queries the trained surrogate. The surrogate instantaneously predicts either the direct geometric deformation footprint or outputs a calibrated set of simplified stiffness/damping parameters.
4. **Engine Injection:** These predicted parameters are dynamically injected into the NVIDIA PhysX engine, forcing the real-time soft-body simulation to behave exactly as the offline Abaqus model dictates.

This technique enables training applications to benefit from computationally demanding material models while preserving interactivity. We performed PhysX experiments using two approaches. Initially, we used USD Composer from Omniverse [22]. Using its graphical tool, we conducted preliminary experiments to establish the proof of concept. Subsequently, we used the PhysX SDK to build a platform that controls the performed manipulations, to assess the results quantitatively in an automated fashion. In the following, we describe the basic elements of the PhysX SDK-based platform.

4.2 Interactive simulations

This section describes the engineering effort to deploy the mapped surrogate data into a real-time training application. Interactive simulation is treated not as a standalone demonstrator, but as a core component of a wider pedagogical pipeline. The interactions are procedurally scaffolded by a process schema encoded within a Knowledge Graph (KG), where every simulation node corresponds to a specific craft action enriched with pedagogically relevant constraints and conditions.

Building on initial PhysX experiments conducted in NVIDIA Omniverse (USD Composer) during Y1, we developed a standalone module that offers full control over manipulations and supports quantitative, automated assessment. The module is ready for integration with the Unity-based interactive tool.

The module takes physical properties, geometries, and initial states of objects as input and outputs deformed geometries. Physical properties and initial states of actors are defined in configuration files. Multiple material property sets are defined for soft bodies (workpieces) and rigid bodies (tools). Object geometry is imported using the Wavefront OBJ file format, and the resultant deformed meshes are exported in the same format.

4.2.1 PhysX integration and the SimplePhysX5 toolbox

To handle the real-time physics computation, we leverage NVIDIA PhysX [163], a GPU-accelerated engine that supports soft-body dynamics through approximations and constrained material models. Based on extensive investigations in Year 1, we developed a domain-specific software layer called SimplePhysX5, which acts as a high-level C# wrapper around NVIDIA PhysX v5.0, directly integrated into the Unity game engine.

SimplePhysX5 is designed explicitly to simulate the interactions between rigid tools and soft, deformable workpieces, satisfying three primary constraints:

1. **Knowledge Model Alignment.** Tools, workpieces, material properties, and execution parameters (forces, velocities) are derived directly from the Knowledge Graph to configure the Unity scene.
2. **Performance.** The core leverages GPU-accelerated soft-body computation, ensuring the responsive feedback required for motor-skill training.
3. **Integration.** The simulation runs as a native plugin. Unity is the rendering and interaction front-end. SimplePhysX5 computes collisions and mesh updates behind the scenes.

The API exposes a compact set of C# classes that call C/C++ functions from Dynamic Link Libraries (DLLs). The core manager is the **`PxPhysics`** class, supported by **`PxRigidDynamic`** for tool transforms, and **`PxSoftBody`** for retrieving deformed vertices. Because these interact with unmanaged memory, they implement the IDisposable interface, requiring memory management to prevent leaks upon termination.

In the implementation, each actor has a physical state and associated material properties. Actors' states evolve due to applied forces and interactions, and their state may influence their material properties, as in the case of temperature. Typically, two actors are defined in the simulation: the tool as a rigid body actor and the manipulated material as a soft body. Given the actor properties and initial states, simulation is performed using the appropriate PhysX SDK API calls. The GPU pipeline is used, as it is required for soft-body simulation. Figure 20 shows indicative results for chisel–wood interaction under varying contact forces and angles.

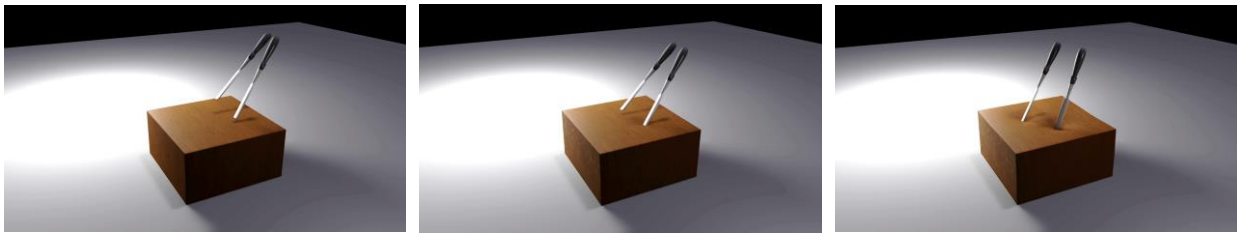


Figure 20. Indicative PhysX results for chisel–wood interaction. Each row shows exemplar frames from different simulations; contact forces and tool angles are varied.

Runtime Workflow & Mesh Cooking

During execution, Unity registers the initial states of tools and materials from JSON configuration files and imports geometry via the Assimp library. Every frame, Unity updates the kinematic transforms of the tools based on the user's VR controller input and forwards this to SimplePhysX5. The toolbox advances the simulation step, calculates the deformation, and returns the updated mesh vertices to Unity for instantaneous rendering.

We generalised our approach for generic deformations and subtractions. This was achieved by developing interfacing classes to PhysX (see Figure 21). On the left, the figure shows a human hand deleting a body of voxels, demonstrating real-time interaction with complex geometries. On the right, the demonstration illustrates the tooling and real-time shaping of wood.

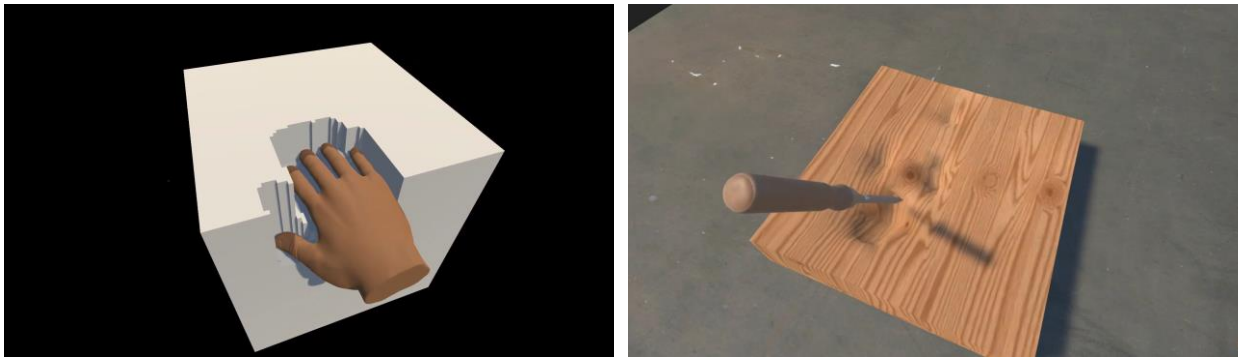


Figure 21. Using PhysX for real-time physics-based interaction. Left: voxel subtraction. Right: voxel deformation. Videos: https://youtu.be/9RR5hznqRP8 (left) and https://youtu.be/b7C_pfXaIyw (right).

Simulated object categories

The API allows the developer to instantiate three kinds of objects in the physics scene:

1. **Ground Plane:** A static planar object of infinite mass and heat capacity that simulates the ground and brings colliding dynamic objects to rest.
2. **Rigid Objects:** Objects whose shape remains constant. Their position and orientation are calculated and updated by the PhysX engine based on gravity, collisions, and other forces.

3. **Rigid Kinematic Objects:** Non-deformable objects whose movement and rotation are controlled externally (e.g., by the user, Unity's animation system, or scripting). They participate in collisions and interactions, simulating the tools being wielded.
4. **Soft Bodies:** Objects that deform upon collision, with the degree and permanence of deformation controlled by FEM-based material models. These represent the workpieces.

Parameters

Simulation parameters, including material models and tool trajectories, are typically stored in configuration files (e.g., JSON) to ensure systematic variation and repeatability. In the current implementation, object geometry can be imported using the Wavefront OBJ file format, and simulation parameters are recorded in JSON and acquired by the engine during execution.

Runtime and simulation workflow

The run-time loop is:

1. Registration: Tools, workpieces, and materials are registered with the SimplePhysX5 API during scene start-up.
2. Kinematic Update: Each frame, Unity updates the transforms (position/rotation) of kinematic objects (tools) based on user input or animation and forwards them to SimplePhysX5.
3. Simulation: The toolbox advances the PhysX simulation by one time step.
4. Read-back and Rendering: Updated transforms for rigid dynamic objects and the deformed vertices for soft bodies are read back from SimplePhysX5 and applied to the corresponding Unity GameObjects. The resulting meshes are then passed to Unity's rendering module for visualisation.

API classes and memory management

The API's core is the PxPhysics class, which manages materials, objects, and the main simulation-step method that must be called by a Unity script every frame. Other key classes include PxRigidDynamic (for rigid-body transforms) and PxSoftBody (for retrieving deformed vertices). Material properties are managed by PxMaterial (for rigid bodies) and PxFEMSoftBodyMaterial / PxFEMParameters (for soft bodies).

Most C# classes implement the IDisposable interface. Therefore, the Dispose() method must be called for all objects at program termination, in the reverse order of their creation, to prevent memory leaks.

Mesh cooking

To achieve interactive frame rates, the toolbox relies on the PhysX feature known as mesh cooking. This is a pre-processing step that builds spatial acceleration structures, enabling a mesh to participate efficiently in collision detection. SimplePhysX5 exposes methods for both in-memory cooking and for saving pre-cooked meshes to disk. In Craeft, mesh cooking is performed offline during lesson authoring, allowing simulations to start promptly even with geometrically complex workpieces.

Input/Output

Physical properties and initial states of the actors are defined in JSON files and imported using the boost property tree library [24]. Different sets of parameters are defined for the soft bodies (materials that are manipulated) and rigid bodies (tools). Object geometry is imported using the Assimp library [161]. Assimp supports several formats; in our case, the Wavefront .obj file format is used for mesh definition [28]. The deformed meshes that result from the manipulations are also exported in the same format.

4.2.2 Evaluation, validation, and calibration

To bridge the gap between high-fidelity physics and real-time interaction, the trained surrogate models (such as the 3DNS neural network) are used to calibrate the PhysX solver parameters against the FEM-derived targets.

For example, the 3DNS model utilises a radially symmetrical brush template based on a smooth-step function, modulating stroke intensity to reproduce ground-truth deformations. However, because retraining the 3DNS network takes approximately 10 seconds per unique deformation, it cannot support continuous manipulation directly. Thus, its outputs are used to calibrate the PhysX material parameters, allowing PhysX to handle the continuous real-time collision detection.

Validation of this hybrid approach occurs on two levels:

1. **Physical Plausibility:** We verify that the simulated responses (deformation and stress patterns) behave consistently with expected material trends under controlled execution variables.
2. **Training Efficacy:** We evaluate whether the simulator provides interpretable, pedagogically measurable signals. This includes assessing semantic guidance clarity, interaction ease, and expert judgment of fidelity.

In practical testing, the calibrated interactive simulation successfully differentiated skill levels: novices exhibited high variability in tool angle and force, improving over repeated attempts, while experts immediately reported stable performance and confirmed that the simulation's constraints aligned accurately with real-world physical practice.

4.2.3 Surface deformation results

The final integration was tested by comparing the real-time PhysX approximation against the offline Simulia ground truth. The scene featured a rigid rod striking a deformable timber block at angles of 0°, 15°, and 60°. As shown in comparative profiles (Figure 22), the 3DNS-calibrated real-time results (plotted in red) closely approximate the high-fidelity Simulia results (plotted in orange), successfully capturing the directional bias and volumetric displacement of the impact. The experiment compares the PhyX approximation.

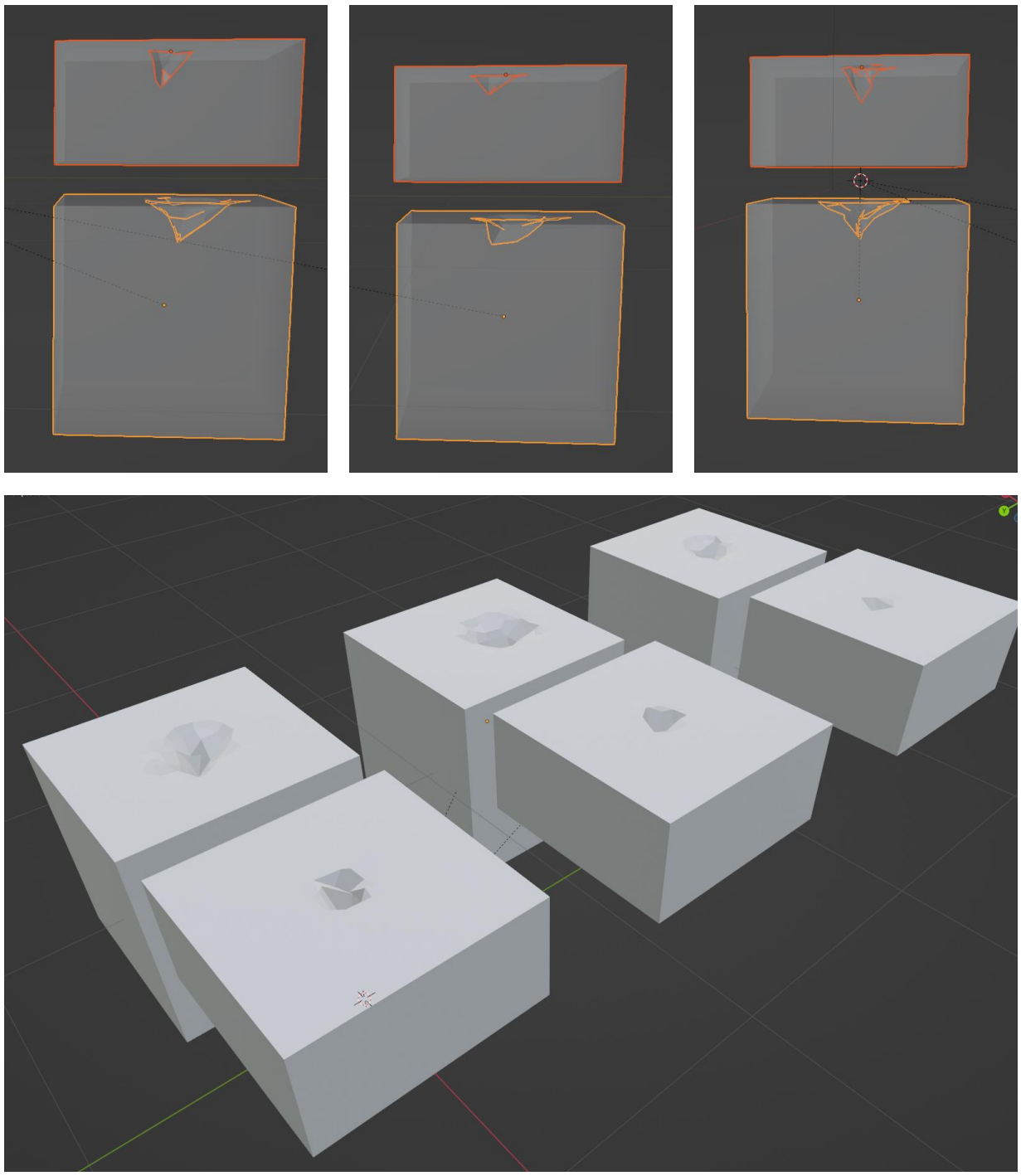


Figure 22. 3DNS results compared to Simulia, for three angles of incidence. Top: profiles of stroke results for 0°, 15°, and 60° (left to right, respectively). 3DNS results are plotted in red and Simulia results in orange. Bottom: 3D renderings of the results above, in the same order as above, from left to right; Simulia results in the back and 3DNS results in the front.

Subsequent real-time demonstrations successfully visualised complex voxel subtraction and surface deformations for timber and aluminium (Figure 23 and Figure 24), achieving the necessary balance of physical realism and instantaneous interactivity.

To validate, we verify physical plausibility by inspecting simulated responses under controlled variation of execution parameters and boundary conditions, and by checking that deformation and stress patterns behave consistently with expected material trends. Second, because the toolbox is designed for training scenarios, we validate that the simulator supports measurable learning-relevant signals: whether it can distinguish correct execution from typical errors, and whether the resulting feedback is interpretable to learners and credible to experts.

In practice, this training-oriented evaluation is operationalised using dimensions such as semantic guidance clarity, interaction ease, perceived learning support, and expert judgement of representational fidelity, complemented by objective logs (e.g., task completion time and mode-dependent differences in accuracy and error rates).

Following the quantitative profile comparison against the offline Simulia reference (Figure 22), we broaden the demonstration to a tool/material sweep that better reflects the variability of real workshop interactions. Figure 23 summarises a set of cylindrical-tool contacts executed on both timber and aluminium, spanning different strike angles and contact-force levels. The purpose is not to highlight a single “best” outcome, but to show that the calibrated real-time model produces stable, plausible surface deformation behaviours across a range of operating conditions, and that the response changes consistently with both tool geometry (blunt cylindrical contact) and substrate material.

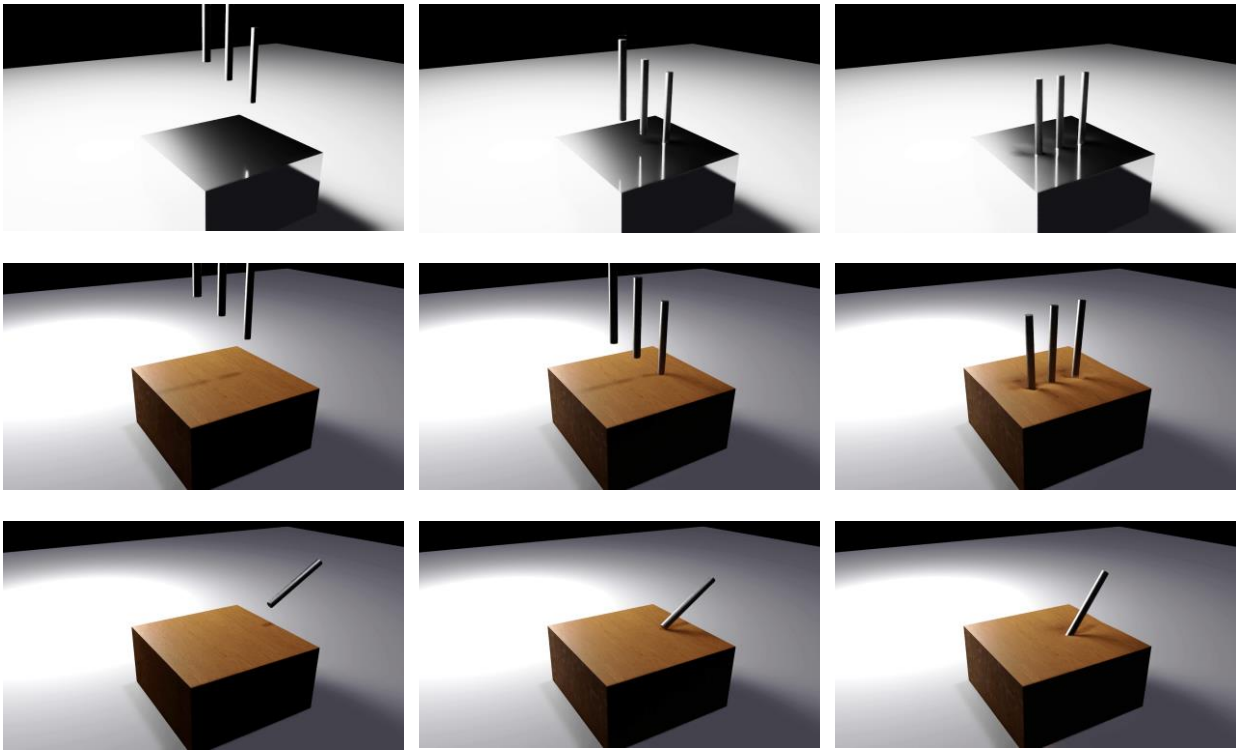


Figure 23. Simulation of the effect of cylindrical tools on different surfaces (aluminium and wood). Every row visualises exemplar frames from other simulations. Different contact forces and angles are simulated, illustrating the various impacts on the surface deformation.

We then repeat the same style of sweep using chisel-like tools, which impose a markedly different contact regime: the interaction is more localised and directional, and small changes in angle or load can produce qualitatively different surface effects. Figure 24 collects exemplar frames from multiple runs on a wooden workpiece, again varying incidence and force, to illustrate that the real-time simulation remains coherent when the contact patch becomes sharper and more anisotropic. This complements Figure 23 by showing that the approach generalises beyond blunt impacts to edge-dominated tool actions under comparable parameter variation.

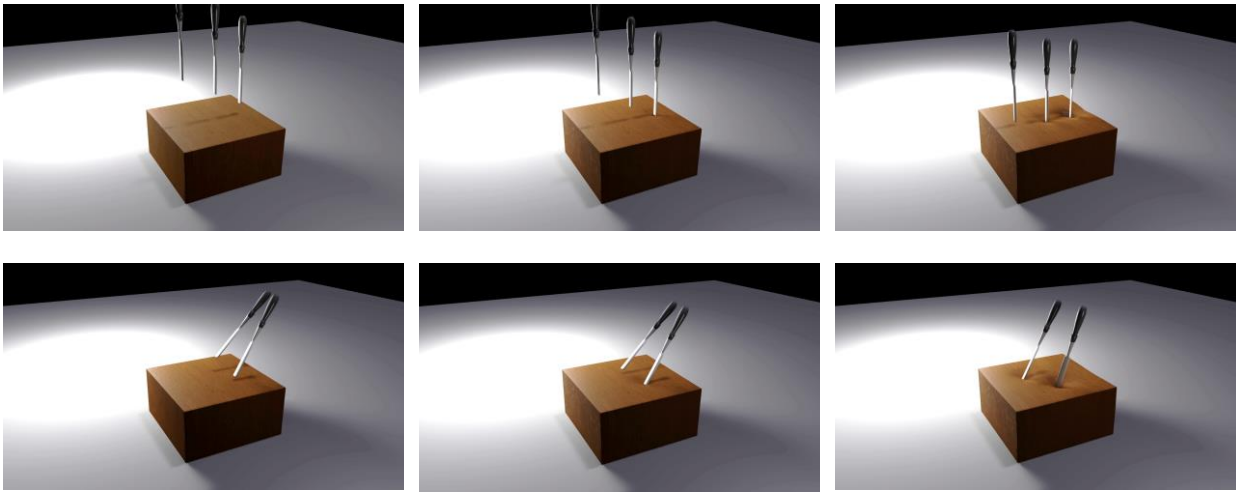


Figure 24. Simulation of the effect of chisel tools on a wooden surface. Every row visualises exemplar frames from different simulations. Again, different contact forces and angles are simulated.

4.3 Conclusion

This section positions high-fidelity FEM as the *reference layer* for craft action simulation: it is the mechanism by which tool–workpiece interactions are modelled with physically meaningful stress, displacement, and (where relevant) thermo-mechanical effects. To make this feasible at scale, we defined the action space in terms of a small set of execution parameters (e.g., tool vector, incidence angle, force/displacement, and material state), and generated paired input–output samples that expose how those parameters modulate deformation outcomes. A simple but important consequence is that "action types" are not merely semantic labels: the distinction between cutting and forming is reflected directly in contact geometry, orientation sensitivity, and failure/deformation regimes, and it therefore informs both dataset design and model selection.

We also showed that thermal conditioning is not auxiliary in hot-working scenarios. Varying temperature fundamentally reshapes the stress and pressure fields under the same mechanical input, indicating that temperature (or a proxy such as viscosity/yield stress) must be treated as a first-class input to any model intended to generalise across realistic operating conditions. In this sense, FEM provides not only geometry changes but also the physically grounded relationships between state variables (temperature, stiffness/viscoplasticity) and observable outcomes.

Because FEM outputs are too dense for interactive runtimes, the section then introduced the key engineering approach: a surrogate-to-engine mapping where a lightweight model (3DNS) learns the non-

linear relationship between sparse execution features and high-fidelity deformation signatures, providing runtime parameterisation for a real-time simulator (PhysX). The result is a practical hybrid: interactive soft-body behaviour remains responsive at millisecond timescales while being calibrated against offline "ground truth" through comparisons with Simulia profiles and qualitative consistency across tool types and materials.

Finally, we framed validation in training terms: the objective is not only physical plausibility, but the production of learning-relevant signals—consistent differentiation of correct execution versus typical errors, and feedback that is interpretable to novices and credible to experts. This completes the bridge from offline FEM to deployable interactive simulation, and prepares the ground for subsequent sections that scale these mechanisms across additional craft actions and integrate them into the broader authoring and training pipeline.

5 Light Transport Simulations

Section 5 focuses on appearance under physical optics constraints. The Light Transport Simulation capability is introduced as a reusable technical module that supports multiple downstream uses across Craeft.

Many craft outcomes are judged primarily by how they look under illumination: translucency, gloss, caustics, coloured absorption, internal scattering, and shadowing are not cosmetic effects but material signatures that inform making decisions. Light transport simulation provides a principled way to predict these signatures, enabling controlled “what-if” studies over material parameters, geometry, and lighting that are difficult or costly to perform physically.

Accordingly, we develop a pipeline that connects (i) material properties, (ii) geometry assets, and (iii) lighting/scene configurations, to produce physically-based renderings and measurable image-derived quantities. The intent is twofold:

1. explanatory: supporting communication and training through faithful visualisations and
2. design-supportive: for parameter exploration, selection among alternatives, and documentation of visually critical choices.

Finally, the same simulation core can be deployed in different modes: as an offline renderer for high-quality visual evidence, as a parametric study engine for material and lighting exploration, and as a component that can be invoked by craft-specific applications elsewhere in the project.

5.1 Visualisation Toolbox

To achieve a generic way of simulating the appearance of artefacts and materials, we created a programmable API and toolbox. This way, the functionalities of the developed utilities are available to multiple utilities developed in Craeft, as well as to third-party developers. Our toolbox is an integration of Mitsuba3, and its purpose is to simplify the usage of Mitsuba3 and automate some of its functionality in a more user-friendly way. In this section, we briefly describe and demonstrate this toolbox and its usage.

The toolbox can create both images and videos; it can be found here, along with usage instructions: https://github.com/andriani-st/mitsuba3-util/tree/GeneralUtil_Refactor. The toolbox is analytically demonstrated in Annex A. A detailed presentation of the toolbox capabilities can be found in the online manual of the toolbox.

Rendering Lambertian (or ‘matte’) surfaces under any given illumination and decor has been commonplace in Computer Graphics for decades. Today, several rendering libraries are available, with OpenGL and Open3D being the most widely used open-source ones. In these libraries, the rendering of textured matte surfaces has been optimised. However, more challenging phenomena, such as light absorption and scattering that occur in shiny, semi-transparent, and translucent materials, are not realistically modelled. To solve this problem, we use a Physically Based Renderer, specifically the Mitsuba 3 renderer. Besides conventional, texture-based renderings, this provides us with a realistic simulation of the appearance of:

- scanned artefacts made from challenging (shiny, translucent, and transparent) materials and
- artefact designs modelled in 3D for which a designer wishes to predict their appearance.

However, as this infrastructure is highly technical and research-based, we have developed a wrapper that simplifies its use for broader audiences. Specifically, we created a toolbox that simplifies the rendering of images and videos of arbitrary scenes, composed of 3D models and light sources, made from virtually any material, given its physical properties. In this way, we can create realistic renderings of craft artefacts. Moreover, we can simulate how these artefacts would appear when placed in an arbitrary environment. The latter capability is useful for realistic previews of craft products and typical environments, in which the buyers would place or use them.

The toolbox is operated using the Python scripting language and integrated into the Design Studio. Nevertheless, it can also be used independently. To further facilitate usage, we created a few wrapper applications that simplify the creation of image and video previews and aid their design. Utilising Mitsuba's state-of-the-art physically-based rendering (PBR) techniques, our software ensures that the light transport and material interactions are simulated with high fidelity. This allows for realistic previews of objects, capturing the nuances of reflections, refractions, subsurface scattering, and surface textures for materials such as conventional and challenging materials, such as glass, metal, plastic, wax, marble, and wood.

The rationale for the toolbox's functional design is the following. To use the toolbox, only a configuration file is needed as input. This file describes the scene elements (objects, materials, lights), the type of requested output (image or video) and its properties (resolution, field-of-view, camera trajectory, etc.). In other words, the toolbox is utilised as a ‘compiler’ which receives configuration files and outputs visual media. Our intention behind this choice is to make the toolbox available to multiple user interfaces that serve different purposes and applications: all that is needed is for the interface to generate the scene file and call the toolbox. Using this approach, we created a few utility applications that specialise in specific craft products.

Using the toolbox, users can define their scenes and adjust illumination settings to match specific environments. Whether it’s natural sunlight, indoor lighting, or complex studio setups, the software provides the flexibility to recreate the desired lighting conditions for the most accurate visualisation. Using 360 photography, users can also import their environments into the appearance simulation.

A software interface allows programmers to adjust parameters, experiment with different materials, and visualise the results. The toolbox is designed as middleware to be used by a GUI that implements an interactive design process, enhancing creativity and ensuring that the final product meets the desired aesthetic and functional requirements.

The toolbox creates high-resolution images and videos that showcase the intricate details of the simulated 3D models. This feature is aimed at presentations, client approvals, and marketing materials, providing a powerful tool to communicate the artistic vision of the practitioner. A basic application simulates the appearance of 3D models made from different material types, such as different types and colours of glass. The example below illustrates a glass body made from variations on the type of glass and metal from the same geometry (see Figure 25).

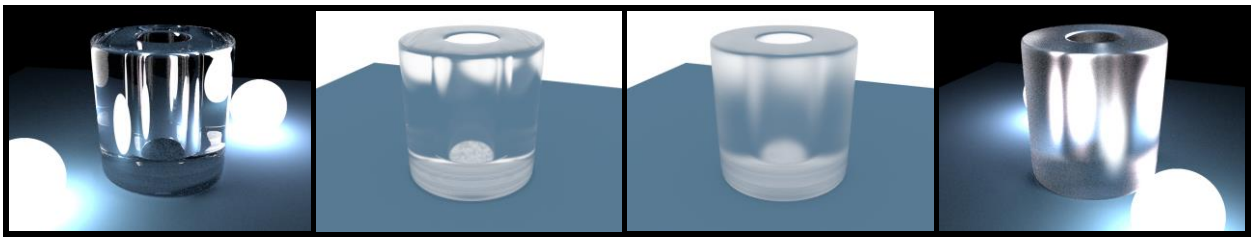


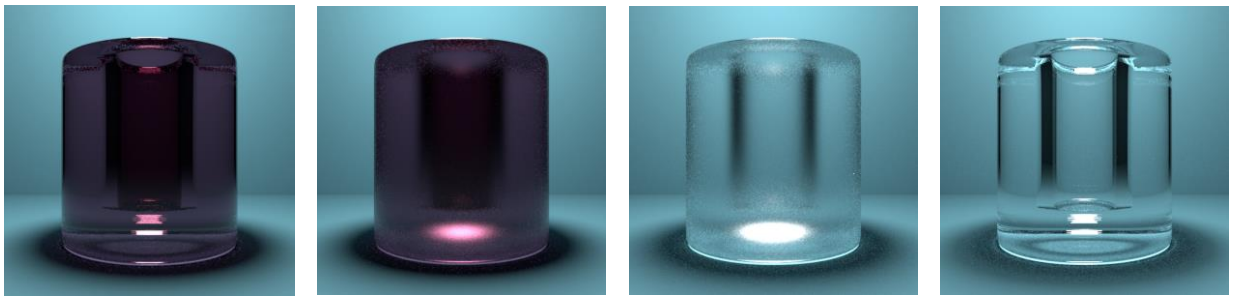
Figure 25. Simulation of materials. The first three illustrate different types of glass. The last simulates a metallic composition.

Next, an application renders a video from a 3D model as if it were placed on a turntable. This way, the user can inspect the appearance of a designed product from all viewpoints around it. In addition, this application can be constrained to rotate the object in a smaller range of angles. In this way, the user can inspect how light interacts with a designed object, such as when rotating a metallic anaglyph under the light to see how light is reflected upon it (see Figure 26).

Figure 26. Top: 360 video rendering of a glass body. Bottom: photorealistic rendering of the glass body.

All materials are described through their Bidirectional Scattering Distribution Functions (BSDFs) [17], which are mathematical functions that characterise how light is scattered from a surface. Given its BSDF, any material can be simulated. To facilitate frequently used materials, the utility toolbox provides some predefined materials. These are glass, plastic, and metal.

The rendering of the materials is demonstrated in a few examples below. Clear and tinted glass of various levels is simulated in the top row of Figure 27 below. The bottom row of Figure 27 demonstrates plastic and metallic materials, as well as the viewpoint modulation.



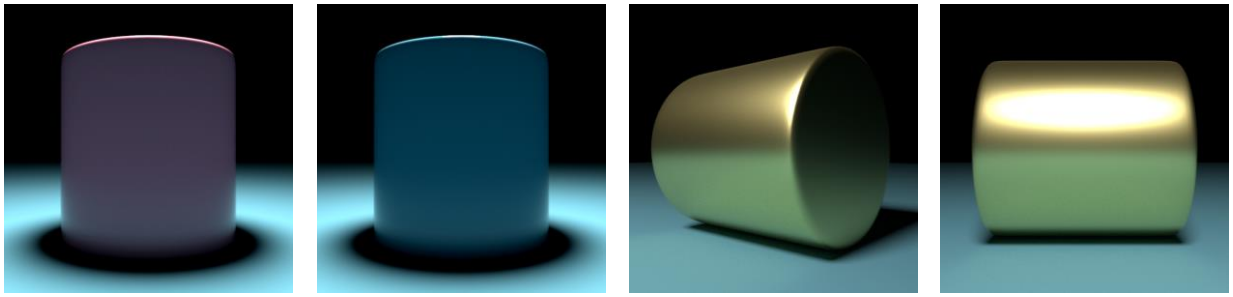


Figure 27. Material Rendering and Viewpoint Modulation

The differentiation between smooth and rough surfaces determines the type of light scattering. For smooth surfaces, any given incoming ray of light, the model always scatters into a discrete set of directions, as opposed to a continuum.

Lighting is necessary for the simulation to result in visible surfaces. Two lighting types are available:

1. Specific light sources. Light sources may be of ovaloid or paralepidid shape. Their size, colour, position, orientation and radiance are user-defined.
2. Environments which simulate ambient illumination. Environments are provided as 360-degree spherical images that define the incoming light from the environment. The toolbox is compatible with conventional High Dynamic Range (HDR) images.

The previous examples utilised the first method mentioned above, that is, the synthetic light sources. In the figure below, the use of HDRIs is demonstrated for an indoor and two outdoor scenes.

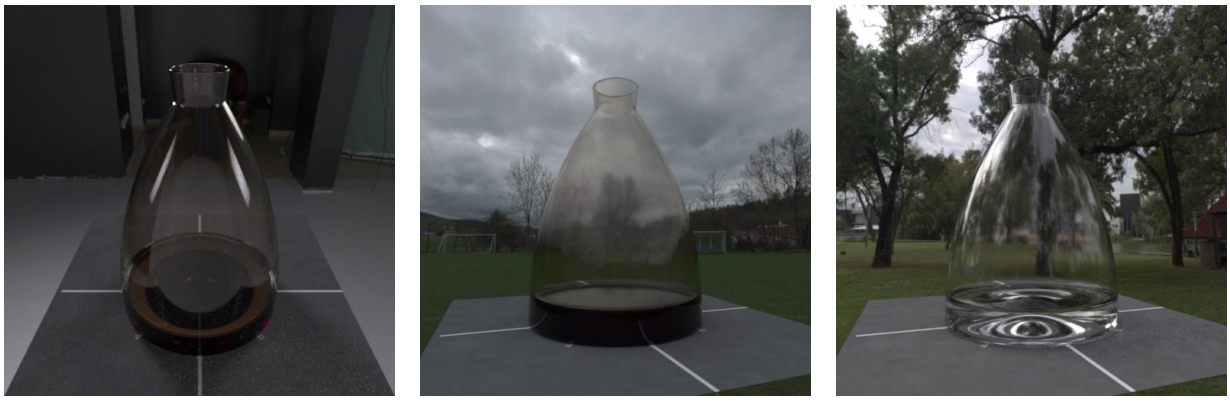


Figure 28. Lighting using prescribed light sources (left) and environment illumination (right).

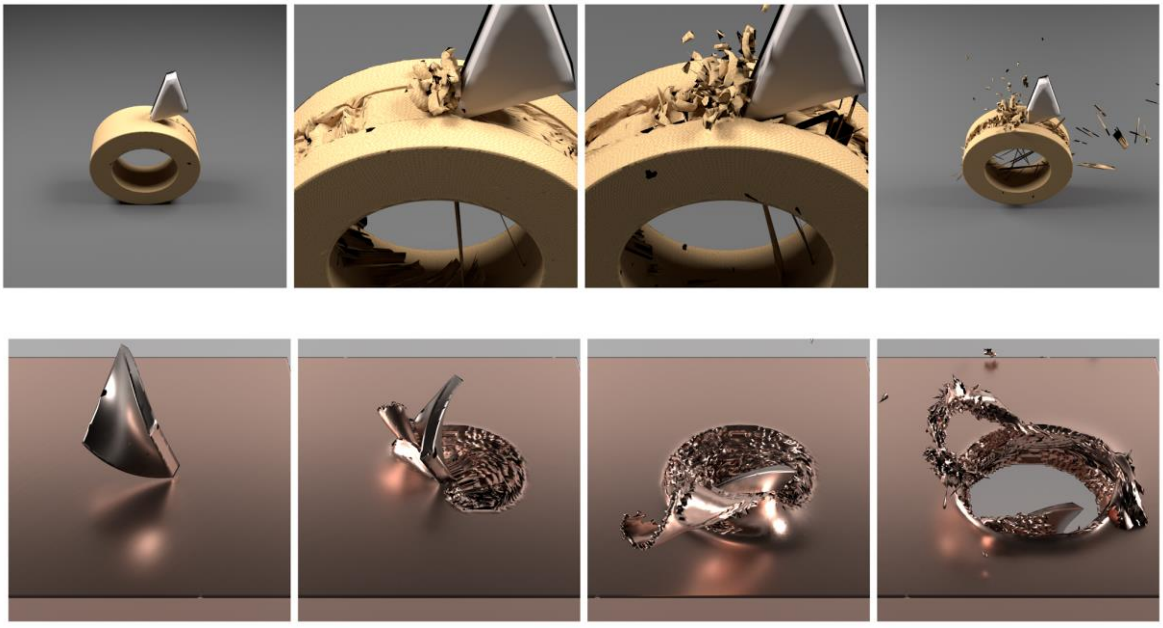
An arbitrary number of light sources can be added to the simulated scene. Moreover, the two types of lighting can be combined by adding multiple light sources to the simulated environment.

5.2 FEM – PBR Integration

We integrated our visualisation toolbox renderer with the FEM simulation engine (Simulia Abaqus) to create realistic renderings of processes that treat materials that are challenging to render. By merging the precision of FEM simulations with the visual fidelity of physics-based rendering, we produce detailed visual representations that better reflect the material behaviours and techniques inherent to these crafts.

Integrating realistic material and appearance properties in FEM simulation and rendering with a light transport simulation toolkit enables realistic visualisation of material deformation during processing. This level of realism is valuable for a variety of applications. It enables a thorough analysis of how different materials and techniques interact. These realistic simulations serve as teaching tools. The integration serves for cultural heritage and conservation. Accurate visualisations can support efforts to conserve and restore artefacts. This technology opens possibilities for innovation, enabling the exploration of new materials and techniques in a virtual environment before they are applied in practice. Thus, this integration offers opportunities for product design, prototyping, and artistic expression. By combining traditional craft techniques with modern technology, designers and artists can create novel forms of expression.

The integration works using file communication between the two software suites. We extended our toolbox to read Simulia Abaqus output files. From the input, we read the materials used and, in the visualisation toolbox, we specify their visualisation properties appropriately. Below, we present material-specific visualisations of archetypal simulators presented in D2.1 for wood, bronze, and plastic.



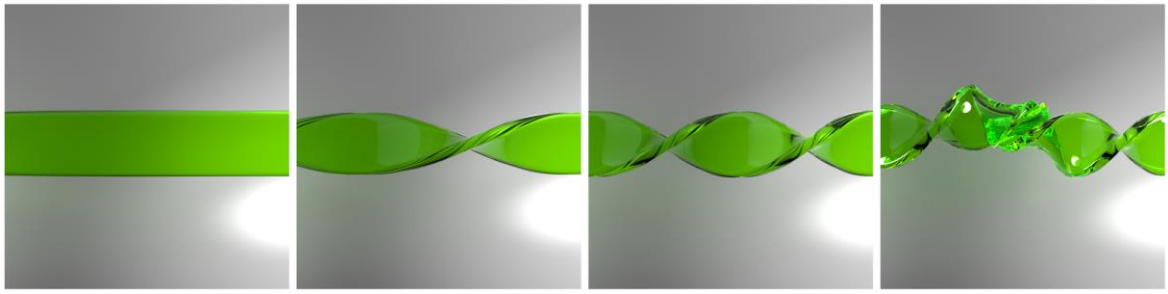


Figure 29. Realistic visualisation of crafting actions.

The examples below illustrate cases that are unsafe or expensive to test in the workshop. Figure 30 (top) shows a metal carving simulation, which is instantiated for copper cut by a Tungsten wedge. Figure 30 (middle) presents a metal drilling example using a workpiece and tools from the same materials. The middle row illustrates two instances of the process, as shown in the left and middle images. In the right image, we demonstrate another capability of our approach, namely, 'hyper-realistic' rendering. In this case, we modified the material parameters to create an explanatory illustration. The workpiece and supporting plane are rendered as glass to enable views that show the interaction of the tool with the material. Figure 30 (bottom) renders a hot-rolling process (performed under great heat and pressure) that thins a copper plate. Videos are provided for all three examples.

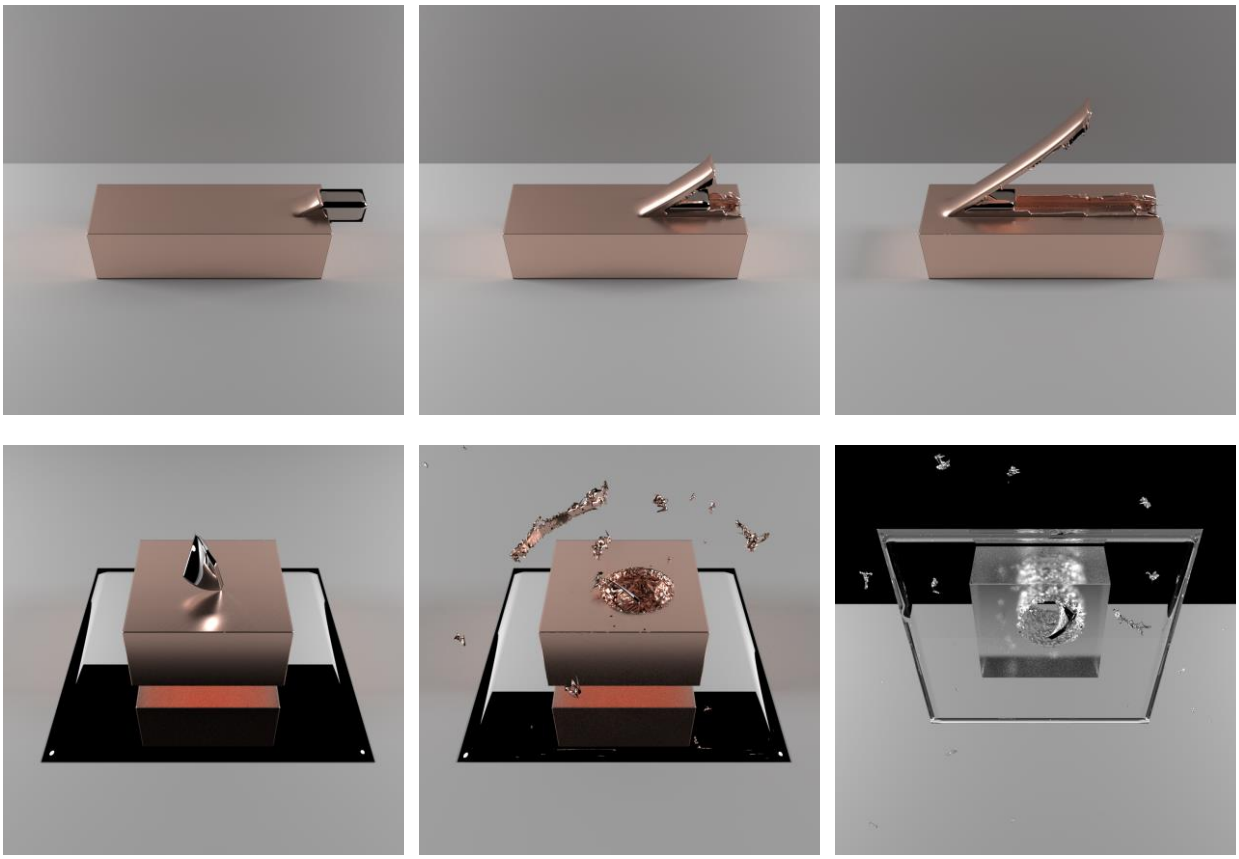


Figure 30. Top: Metal carving. Videos: https://youtube.com/watch/eKw25yT-UkA (same as above) and https://youtu.be/EpuJlB1bEdc (multiview visualisation). Middle: Metal drilling. The left and middle images show two top-view frames. The right image shows a view from the bottom, where the workpiece and ground plane are rendered as transparent materials to show the drilled structure. Videos: https://youtube.com/watch/q8i8R3cCFMs (left, middle) and https://youtu.be/OTW0CeY2O9w (right). Bottom: Metal hot-rolling. Video: https://youtube.com/watch/aus1FQQrNko.

5.3 Diffuse and stratified bodies

This subsection utilises the Material Rendering Toolbox to render a porcelain appearance under controlled viewing and illumination conditions. This targets two needs in digital representation and communication of material qualities:

1. Controlled comparison of appearance states, including unglazed, glazed, variants of gloss, opacity, and celadon-like states.
2. Ease of dissemination in simple formats, including HTML browsing.

We present a visual simulation of porcelain appearance, specific to the state of the workpiece at each stage, from baking to glazing. Additionally, some states offer alternative options, such as different glaze selections.

The adequacy of the proposed approach to differences in material composition. On a second dimension, we comparatively render the same processing states for different porcelain lineages, with correspondnly differnt material properties.

5.3.1 Material states

Eight examples demonstrate the impact of subtle alterations in the top layer on our perception (see Table 15). The states are identified semantically as glazed, celadon, and opaque. T1–T2 vary the subsurface scattering of the porcelain body. T3–T4 approximate thin glazes. T5–T8 model the glazed ceramic as a stratified object. The latter, modelled as a film coating the body, enables thickness-dependent absorption and scattering effects.

We separate two modelling families.

1. Unglazed porcelain (T1–T2) is modelled as a purely diffuse reflector (diffuse BSDF), reflecting the matte, porous surface behaviour expected for biscuit and Parian-like finishes.
2. Glazed porcelain (T3–T8) is modelled using the “roughplastic” BSDF model with the GGX microfacet distribution, capturing a specular clear-coat effect over a diffuse substrate. In glazed states, the internal and external indices of refraction are fixed, and the specular reflectance is retained constant (white) to keep comparisons focused on the controlled appearance variables. The principal state-dependent control is the microfacet roughness (α), which determines whether the glaze reads as 'glossy', 'satin', or 'hazy'.

The utility represents porcelain appearance using a range of material states, T1–T8, by modulating the parameters of surface scattering models. Renderings use a spectral-to-RGB path integrator with depth control to model multiple scattering, Fresnel reflection, and volumetric absorption. Colour is treated through the diffuse base reflectance used for the unglazed states and the diffuse substrate beneath the glaze.

Table 15. Porcelain conditions tested.

Category	Light transport mechanism	Visual sensation
Unglazed (T1–T2)	Subsurface scattering in the body; weak surface specular	Matte to semi-translucent; body-colour dominant
Transparent glaze, surface model (T3–T4)	Dielectric surface reflection (Fresnel) + roughness-controlled specular lobe	Gloss level changes; highlights sharpen/broaden
Explicit glaze film, clear (T5)	Explicit film + multiple internal bounces + body return	Deeper gloss; thickness-dependent depth cues
Coloured transparent glaze (T6, T8)	Absorption in the glaze film (participating medium)	Green tint with optical depth; saturation increases with absorption/thickness
Opaque scattering glaze (T7)	Strong volumetric scattering in glaze film	Milky/creamy appearance; suppressed detail beneath

Often, porcelain products (mainly tableware) are decorated. The utility provides three variants of the above states. These are:

1. CMP: plain (no decoration).
2. PTN: patterned using visual texture that determines the diffuse channel.
3. RNG: adds a gold-painted equator ring.

For the patterned mode, the diffuse reflectance or substrate is driven by a repeat-wrapped bitmap texture. The gold ring is simulated as a low roughness conductor with material preset ‘Au’ (gold).

5.3.2 Implementation

Fired porcelain body

The fired (biscuit) porcelain regime is intended to capture the perceptual characteristics that matter for craft communication and design assessment: a predominantly white, matte-to-satin body with soft diffusion cues that can be influenced by thickness and illumination. The simulator uses a physically based

appearance model in Mitsuba 3 that is appropriate for systematic comparisons and controlled sweeps. Where translucency is represented, it is expressed through a subsurface or volumetric proxy parameterisation that produces the expected qualitative response under fixed lighting (e.g., softening of contrast and light spread), rather than asserting absolute material calibration.

Glaze as a shader-layer approximation

The glazed regime models glaze as a dielectric surface layer applied via shading, controlling the dominant perceptual signatures of glazing: specular highlights, gloss variation, and view-dependent reflectance. The glaze is not represented as explicit thickness geometry; therefore, thickness-dependent phenomena that require geometric layering are outside the scope. This choice is deliberate and advantageous for D3.1, because it enables clean, interpretable sweeps over glaze finish and optical response without requiring additional geometry preparation.

Interpretation boundary

Results in this chapter are to be interpreted as controlled appearance simulations under a specified lighting and camera protocol. Claims are limited to reproducible differences attributable to explicit parameter changes (e.g., enabling glaze; increasing glaze roughness), rather than to absolute photometric fidelity.

5.3.3 Comparative presentation

We use these technical capabilities to set a comparison, implementing a virtual scene generator software and a viewer for this purpose. This enables us to comparatively see the bodies in a wide range of imaging conditions, where we can modulate the illumination and observation viewpoint. We present this software mechanism, using the porcelain example to explain it.

Having the porcelain states we wish to compare as input, the software generates views by constructing a scene dictionary that describes the geometry, materials, emitters, and sensors.

User controls

- Camera: radius, height, and FOV.
- Sampling and resolution: samples-pes-pixel, width, and height.
- Animation: angular step of camera or object rotation.
- Lighting mode: point light by default or HDRI environment map.

In the experiments below, the renderings use the same physically based light-transport configuration, the same camera model, and the same environment illumination. Images are rendered at 960 × 540 px with 256–512 samples per pixel (SPP) and a 35° field of view. The HDRI shown below excites specular and diffuse phenomena.

Virtual scene arrangement

The scene layout is designed so that differences in gloss, translucency, and colour are attributable to material state rather than shape or orientation.

A ground plane is included as a diffuse reference surface, scaled to provide a stable context for highlights, shadows, and contrast across the sequence.

The material samples are spherical bodies, because the geometry of a sphere allows the observer to perceive subtle differences in gloss, translucency, and colour under the same lighting conditions. The spheres rest on a diffuse ground plane and are evenly spaced along a horizontal ring.

For side-by-side comparison, we use a fixed arrangement of eight identical spheres (radius 0.12 m) placed on the ground plane and evenly spaced on a horizontal ring (radius 0.8 m). Each sphere corresponds to a making state.

Lighting

Contextual lighting is simulated with an accurate and realistic spectral energy distribution within an environment map (HDRI). We used a 'studio' HDRI that avoids colours and reflections to present the porcelain materials with as little ambiguity as possible.[2] This HDRI is shown below.

Figure 31. Studio HDRI Environment.

Camera and viewpoints

Two viewing regimes excite different perceptual cues:

1. **Turntable:** the camera remains fixed while the objects and substrate rotate through 360°, as if on a turntable. This highlights specular dynamics and makes glaze differences easier to see because the surface normal field sweeps through the illumination.

[2] The asset was downloaded from https://polyhaven.com/a/studio_small_08 at 4K resolution.

2. **Orbit mode:** the substrate and object constellation are static while the camera moves around circularly. The illumination relative to the objects is stable, supporting the observation of body tone, translucency, and form.

Both motions correspond to one full revolution (360°), sampling every 0.5° of rotation.

5.3.4 Visualisation

The results are provided in two formats: video animations and through an interactive viewer. The viewer structure is generic and can be reused for more comparisons, as explained below.

Video

The rendered frames are encoded into lossless MP4 sequences using the FFmpeg encoder (libx264rgb, CRF = 0). The eight spheres are arranged on a circular ring and ordered clockwise, starting from the sphere nearest the camera, as T3 → T4 → T5 → T6 → T7 → T8 → T1 → T2. This ordering groups the glaze variants first and places the unglazed bodies last, supporting a consistent reading as the ring rotates or the camera orbits.

Interactive visualisation

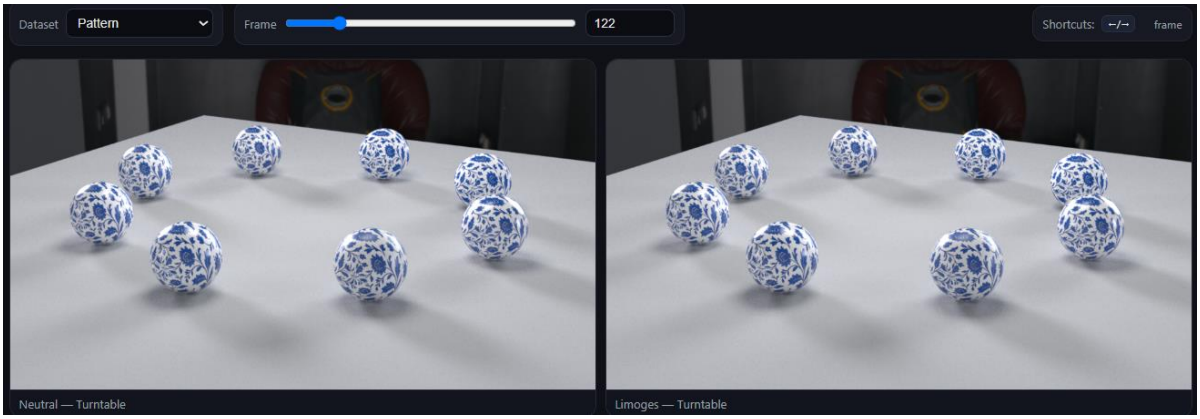
Interactive control includes pausing, scrubbing, reversal, and side-by-side comparison. It supports the inspection of subtle differences. The software emits a simple HTML viewer for interactive inspection of the results. This viewer assembles rendered frame sequences into an interactive layout for evaluating material appearance across material states and motion types.

The interactive viewer featuring the interactive examples below can be downloaded at: https://doi.org/10.5281/zenodo.18190038. The viewer is provided as a self-contained HTML page alongside the corresponding rendered-frame folders.

Screenshots of the viewer for the three conditions are shown in Figure 32. Below each screenshot, a link to the corresponding video is provided. The three conditions, namely Plain, Rings, and Pattern, are selectable through the 'Dataset' menu. The slider runs through animation frames, and the frame number is displayed. The keyboard shortcuts '←' and '→' move a frame backwards or forward, respectively. At the bottom, a footer summarises the dataset, the current frame number, and the source folders from which images are loaded.

https://youtu.be/ETse0vCMWNs https://youtu.be/GJBy7W_JCOM

https://youtu.be/Rt698abrEtQ https://youtu.be/28w8jAHDKEg



https://youtu.be/nW1dyWevf4M https://youtu.be/mq3uJkpycgI

Figure 32. The HTML viewer UI and examples for the three conditions investigated. Left column: orbital camera motion. Right column: turntable subject motion.

5.3.5 Ceramics and glazes

Glazing in ceramics enhances both aesthetic and functional qualities. It provides a wide range of colours and finishes, enabling intricate decorative effects such as glossy, matte, or textured surfaces. Functionally, glazing makes ceramics waterproof, more durable, and easier to clean by creating a non-porous surface that resists scratches, stains, and bacteria. Additionally, glazes offer chemical and thermal resistance, protecting ceramics from acidic or alkaline substances and enhancing their suitability for culinary, laboratory, and oven use. Properly formulated glazes ensure food safety by preventing harmful substance leaching. Artistically, glazing allows for creative expression through techniques like layering and sgraffito, producing unique effects such as crackling and drips.

To produce the appearance of glazed ceramics, we have created a software utility that operates as follows. The input is a 3D model of the ceramic artefact. The model may be textured or not, in which case its absorption spectrum or material must be defined.

In both cases, the procedure is the same and requires the following parameters:

1. the thickness of the glaze (typically in the order of one to three millimetres)
2. the roughness of the glaze

Our software utility scales the 3D input mesh to inflate according to the required glaze thickness. This results in a second 3D model, which is larger than the original and is in the same coordinate frame. Since the two models are aligned, we proceed to use the algorithm in [15] to perform their Boolean subtraction. This results in a new 3D mesh, representing the difference between the two meshes.

Since the two meshes are aligned and the one is a scaling of the other, the difference is a “hollow” mesh that represents the structure of the glazing material. A simple example below illustrates the procedure.

1. **Original Mesh**: Let a 3D sphere be the original mesh.
2. **Inflated Mesh**: Scaled the original mesh uniformly in all directions. In this example, this is simply a larger sphere.
3. **Difference of Meshes**: The hollow mesh is created by subtracting the original mesh from the inflated mesh. This means the final object will look like a shell with the thickness determined by the scaling factor.

Figure 33 shows the visualisation of this process:

1. **Original Mesh**: The blue sphere represents the original mesh.
2. **Inflated Mesh**: The red transparent sphere is the scaled version of the original mesh.
3. **Hollow Mesh**: The final image shows the hollow mesh, where the blue sphere (original mesh) is subtracted from the red sphere (inflated mesh), resulting in a spherical shell.

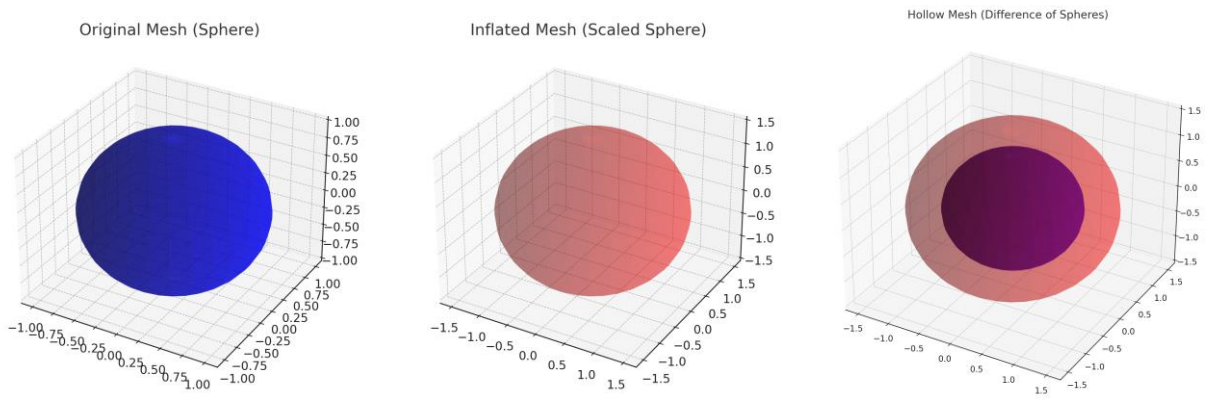


Figure 33. Example illustration of mesh subtraction. Left: original mesh. Middle: inflated mesh. Right: "hollow" mesh.

In Figure 34, we demonstrate the technique for different types of glazes on a clay body, painted with underglaze colours (typically made from metal oxides). In the first row, no glazing is applied to the painted body, which is treated as a diffuse (or Lambertian) surface. In the middle row, a thin layer of glaze is applied. In the bottom row, a thick layer of glaze is applied.

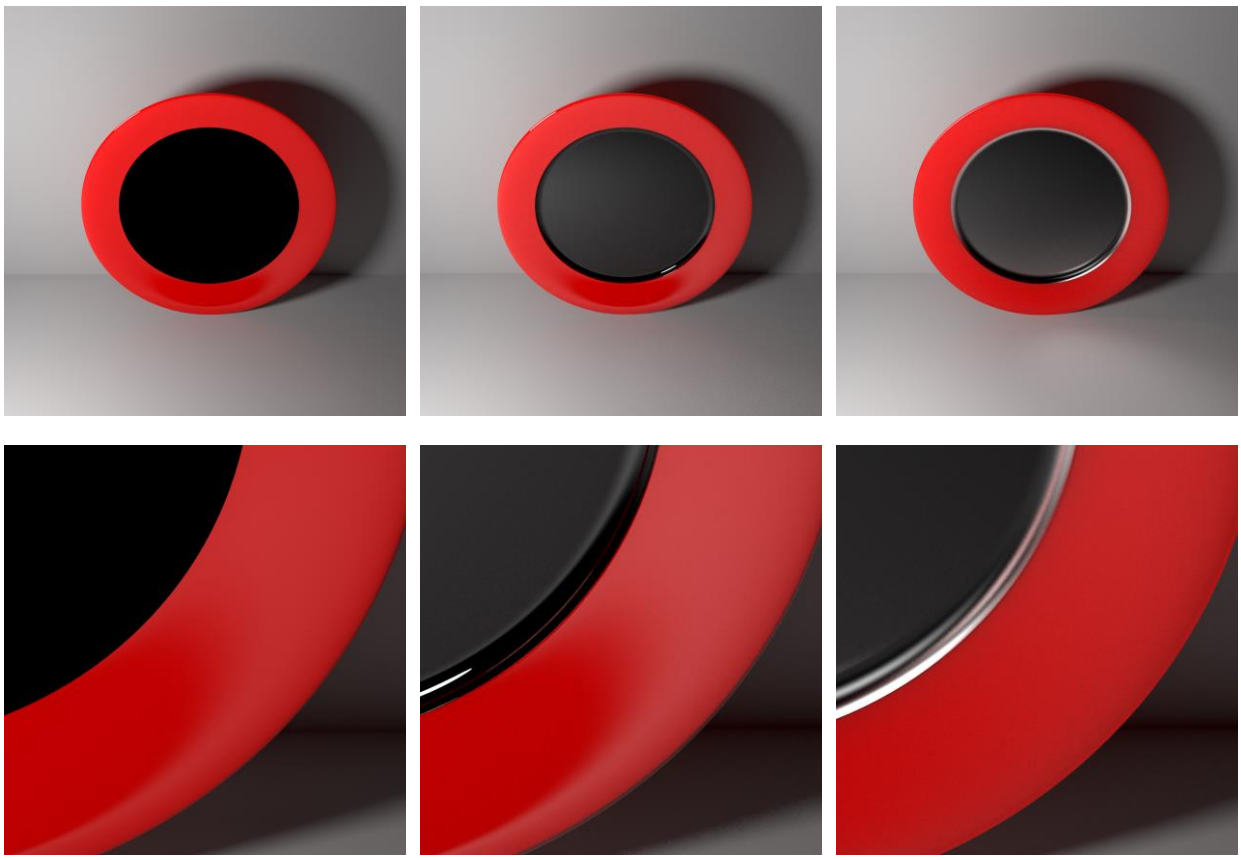
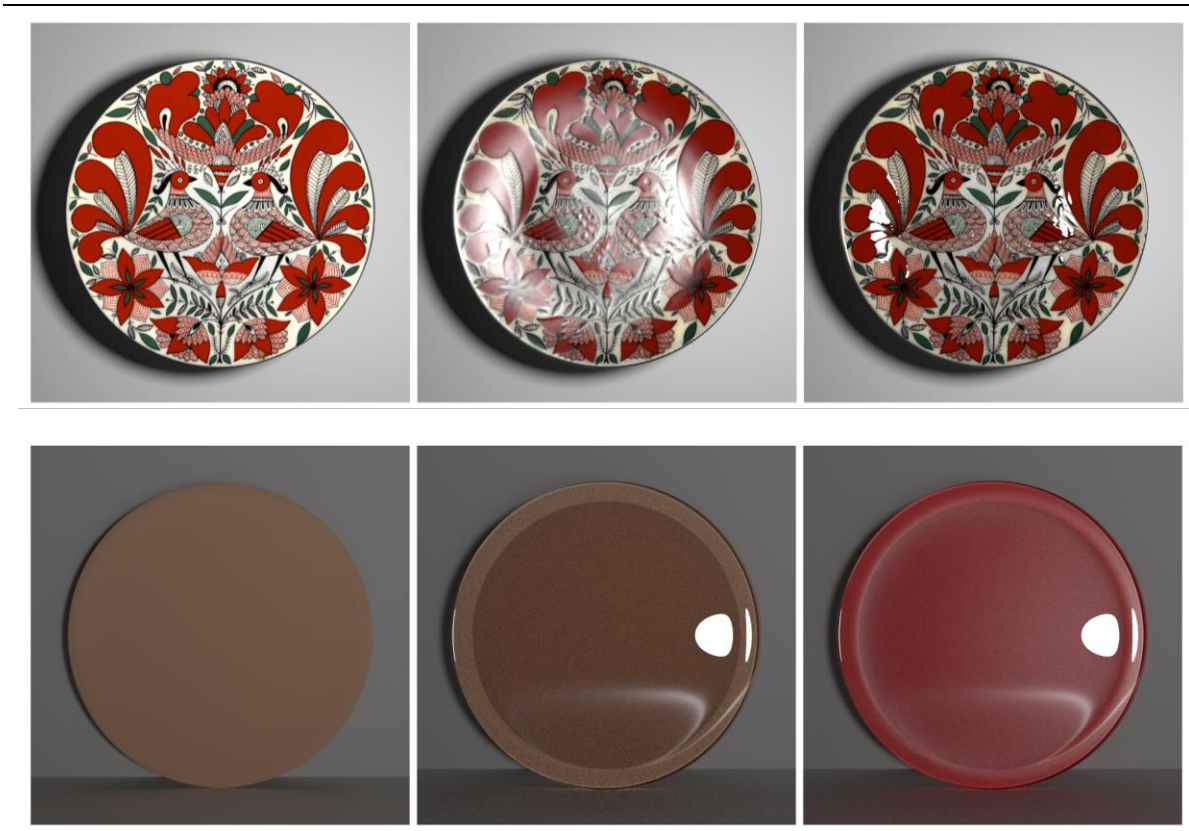


Figure 34. Visual simulation of glazed appearance. Top: no glazing. Middle: thin glazing. Bottom: thick glazing.

Part of the craft is the appearance of the intermediate and final product, potentially previewing it at the installation site and optimally during its creation. The visualisation toolbox is used to realistically predict the appearance of craft products from their designs or 3D models. This capability serves the product designer to envisage how the prospective creation would look before embarking on its implementation.

Glazing is important in crafting utilitarian and artistic pottery. The final texture of a pottery piece depends on glaze composition, application thickness, and firing temperature. The simulator takes a textured model of a pottery piece and creates 3D, realistic visualisations depending on the specification of the glaze. These visualisations capture light scattering due to glaze, enabling the refinement of textures digitally before committing to physical trials.



5.3.6 Discussion

The simulations demonstrate that even within identical geometry and illumination, subtle modifications of optical microstructure produce markedly different perceptual outcomes. The turntable view demonstrates the movement of illumination across a stationary camera, highlighting specular reflections and the glossy quality of the surface. The orbit view illustrates camera movement with consistent lighting, highlighting geometric curvature and material thickness. Comparison allows distinguishing subtle colour temperature and translucency differences, as well as changes in gloss, highlight spread, and internal scattering among the eight material states.

This porcelain rendering approach models glaze as a shader-layer approximation, which does not represent spatially varying glaze thickness or micro-defects such as crazing, bubble inclusions, or pooling. The fired body model similarly prioritises controlled appearance behaviour under fixed lighting over full calibration to measured reflectance or scattering parameters.

5.4 Transparent and Translucent Bodies

5.4.1 Motivation

We created software for previewing glass bodies. Because this is challenging, is identified as useful by stakeholders, and does not exist, we created a standalone utility.

Two operational modes correspond to physical interpretations in the rendering pipeline:

1. Solid glass: a single watertight mesh represents a solid object filled with glass medium.
2. Hollow glass: an outer mesh and an inner mesh define a thin glass shell enclosing a cavity.

The approach allows for the comparative simulation of an identical object geometry, modelled either as a solid, moulded component or as a hollow, blown structure. The technical contribution is a unified rendering interface that:

- Distinguishes solid glass over hollow glass geometry, so the physical interpretation is controlled by the user rather than implicitly encoded in the 3D mesh file.
- Separation of controls enables studies by independently manipulating the interface and bulk appearances of the rendered objects. The surface finish (e.g. frosted glass) is varied independently of the medium characteristics.
- Enables exploration of physically based bulk behaviour by incorporating volumetric light transport and spectral extinction input. This allows for the independent investigation of thickness-dependent attenuation and internal scattering.

The software produces reproducible datasets by generating frame sequences and machine-readable metadata, together with a lightweight HTML viewer for inspection and comparison.

The ShellGen software and its manual are provided in https://zenodo.org/records/18659950. The manual can also be found in Annex F.

Specifically, ShellGen is a utility that generates physically based, reproducible rendered datasets of craft objects. It focuses specifically on objects whose appearance is determined by volumetric light transport, such as moulded solid glass and blown hollow glass. ShellGen couples measured wavelength-dependent extinction $\sigma_t(\lambda)$ with unit-consistent scene scaling, in metric units. It separates interface roughness from bulk absorption and scattering controls, enabling systematic exploration of transparency, translucency, and tint under controlled illumination. Batch runs produce frame sequences, as well as an interactive HTML viewer and machine-readable manifests, supporting comparative studies, reporting, and downstream analysis.

ShellGen functions as a dataset generator. Each operation yields consistent multi-view results, such as camera orbits and turntables, and logs the parameter set necessary for replication.

5.4.2 Implementation

ShellGen is a Python command-line utility that programmatically assembles and renders Mitsuba 3 scenes to render glass appearance data. Each invocation resolves geometry and operational mode constraints, constructs a scene with fixed illumination and camera protocol, assigns surface and bulk optical properties, and renders two standard motion presets into a self-contained dataset folder. Optical material specifications are provided via XML, including spectrometry data where available, and are applied consistently with the metric unit convention.

3D model input

Representing wall thickness is often counterintuitive when modelling hollow objects.

- For a solid object, a mesh of the outer boundary is usually sufficient: the renderer implicitly assumes that the region inside the closed surface is filled with material.
- For a hollow object, thickness must be represented explicitly. One option is to ask the renderer to treat a single surface as a shell of a specified thickness. However, this would require special-case programming in the renderer and is incompatible with our goals, because we wish to retain software generality.

For this reason, we represent hollow bodies by modelling both interfaces explicitly: an outer surface and an inner surface, which together define the material layer.

A useful distinction is between open and sealed hollow objects. Many blown-glass objects (e.g., vases, bottles, drinking glasses) are hollow in the physical world, yet are modelled as a single watertight surface.

The mesh already contains the outer wall, inner wall, and the rim that connects them, so it is still the boundary of a solid glass volume.

By contrast, a sealed hollow object (e.g., a bubble) encloses an air cavity completely. The glass forms a thin layer between two air regions, which creates two distinct interfaces. In this case, the geometry must be represented by two surfaces (inner and outer), or an equivalent volumetric shell representation.

In other words, if the inner and outer walls meet somewhere (e.g. at the rim or lips of the vessel), we can represent thickness with one watertight mesh. If they never meet, such as in sealed cavities, we need two disconnected surfaces.

Material properties and light-transport simulation

Glass is simulated using volumetric light transport with a participating medium. The user controls absorption, scattering, and spectral attenuation. This enables thickness-dependent phenomena to emerge naturally: thicker regions appear darker or more saturated (tinted glass) and increasingly diffuse (scattering-dominated glass). Bulk appearance is controlled through parameters that regulate:

- the absorption–scattering balance (clear/tinted glass vs haze/milkiness proxies), and
- the scattering phase behaviour (e.g., forward scattering to approximate haze).

Where spectral extinction $\sigma_t(\lambda)$ curves are provided, ShellGen supports thickness-dependent, wavelength-dependent attenuation, enabling systematic studies of tinting and darkening with thickness.

Surface appearance is controlled through a dielectric interface model with a roughness parameter that spans polished, typical craft finish, and frosted glass. This supports craft-relevant outcomes such as grinding, etching, sandblasting, and glaze micro-roughness effects. Surface finish is treated independently from the bulk medium via a dielectric interface model with a controllable roughness. This supports craft-relevant surface regimes spanning:

- polished glass (near-specular transmission/reflection),
- realistically finished glass (mild roughness), and
- frosted/ground glass (high roughness; reduced interior visibility).

This separation is central to craft interpretability: it allows one to change the finish without changing the material, and vice versa.

Software architecture

The implementation separates responsibilities:

1. A configuration layer validates inputs and records a complete parameter snapshot,
2. A scene construction layer instantiates cameras, emitters, shapes, and material-media bindings,
3. A render driver generates frame sequences for motion presets, and
4. A dataset layer emits frame manifests and a lightweight viewer for inspection.

This organisation supports controlled ablation studies, as changes in surface finish and bulk medium parameters can be isolated while holding illumination and camera protocol constant.

5.4.3 Usage

The user controls are similar to the porcelain case (see previous subsection). For the same reasons as in that case, the utility produces two animation protocols that separate the perceptual effects of camera motion versus object rotation, namely:

- Turntable (TT): The model rotates around the vertical axis while the camera remains fixed.
- Orbit (CAM): The camera orbits around the object while the scene remains fixed.

Illumination is treated similarly, utilising an HDRI environment map, which is retained constant across sweeps for comparability.

The main difference is that in this case, the optical data are defined spectrally by the $\sigma_t(\lambda)$ function defined in the XML input.

Additional parameters control mode, finish, and bulk behaviour. Specifically,

- Mode: determines the geometry, that is, whether the body is solid or hollow; in the latter case, two meshes are required as input.
- Interface: determines the roughness of surface finish, across the polished ↔ frosted range.
- Bulk: determines medium transmission and internal scattering, enabling the simulation of transparent and translucent materials.

5.4.4 Results

This subsection presents a controlled set of renders that validate the spectral, thickness-dependent behaviour of our glass model. The demonstrations use the Stanford bunny, and compare solid and hollow (shell) configurations under identical illumination and camera settings; throughout, the solid rendering is shown on the left, and the hollow rendering on the right. The purpose is to establish a compact, reliable palette of reference glasses that will be reused in downstream applications, most notably the stained-glass composition system.

We selected eight SCHOTT reference glasses that span (i) near-colourless transmission, (ii) extreme absorption ("near-black"), and (iii) saturated warm and cool tints. This spread is intentionally close to the chromatic vocabulary used in conventional (especially medieval) church stained glass: clear/neutral glass, deep blues, greens, ambers/yellows, and strong reds.

The demonstrations use the widely adopted Stanford bunny model, scaled to occupy approximately 10 cm^3. For hollow variants, the inner surface is generated by uniformly scaling down the model to simulate a 2.5 mm shell thickness. The optical properties are taken from Schott reference glasses, using their published spectral measurements.

To isolate the perceptual effects of geometry mode and material parameters, all renderings are produced under identical conditions. In each figure, the hollow body is rendered on the left, and the solid on the right.

Group A — Clear baseline vs strongly absorbing (near-black) (BK7, UG11)

BK7 is the canonical clear reference in the glass industry. It provides a near-neutral transmission baseline against which we can read geometry, refraction, and Fresnel effects without strong chromatic bias. UG11 is used as the extreme absorber: it intentionally drives the model towards a "dark glass" regime, revealing whether our implementation correctly produces strong optical-depth effects, that is, rapid darkening with path length and thickness.

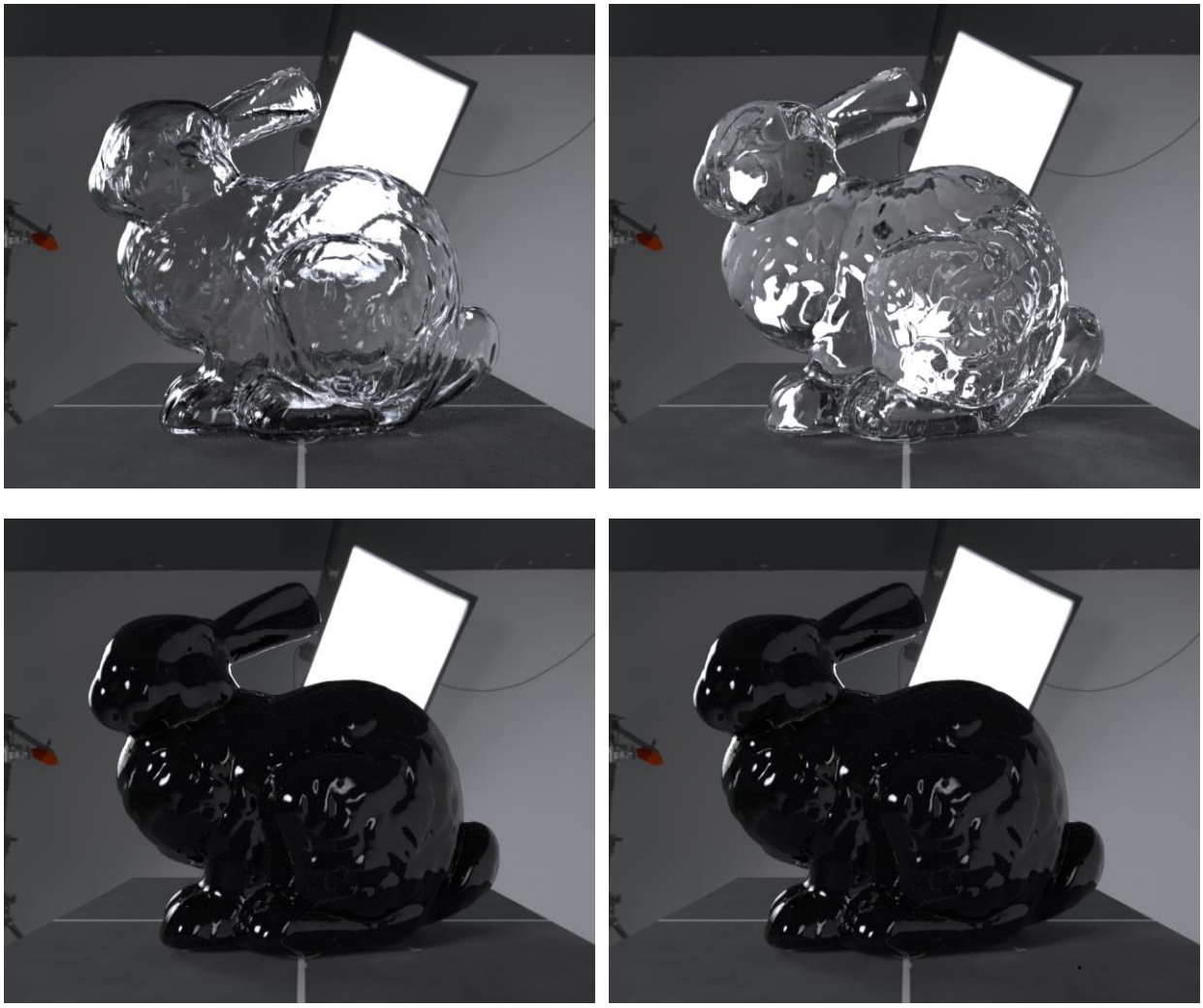


Figure 35. BK7 (top) and UG11 (bottom) rendered under the same protocol. BK7 serves as the canonical clear baseline. UG11 demonstrates the upper extreme of absorption (near-black) under visible and bright illumination. Videos: BK7: https://youtu.be/yE2RHs5vq0I, UG11: https://youtu.be/sKQj3kxrbiM.

Group B — Warm palette: yellow, amber/orange, deep red (GG495, OG530, RG610)

The warm set targets the "sunlit" end of the stained-glass spectrum. GG495 and OG530 form a progression from yellow to amber/orange, while RG610 provides a deep red reference. Together, these glasses test whether the renderer preserves saturated warm hues without collapsing them into flat colour, and whether thickness variations produce the expected deepening of tone.

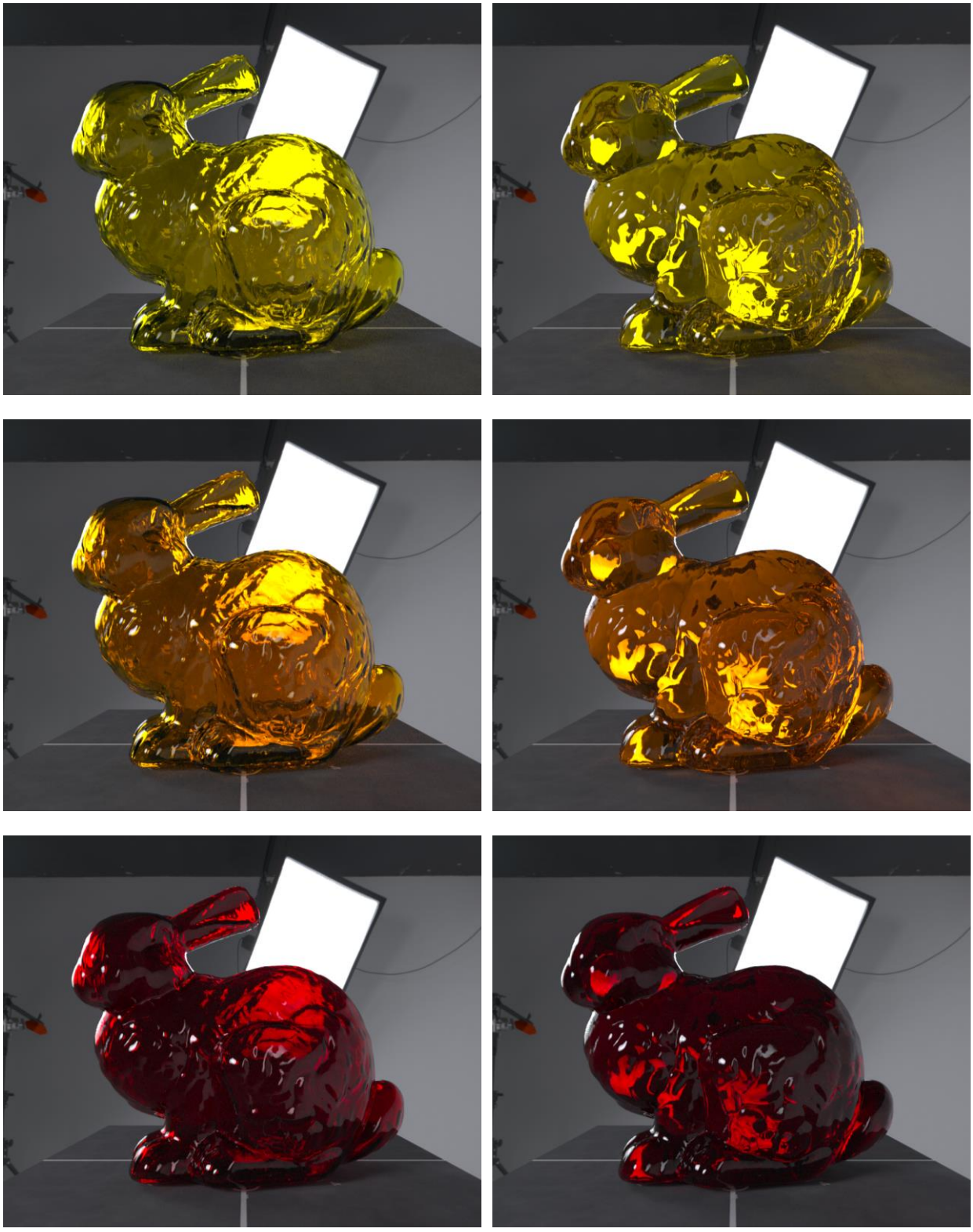
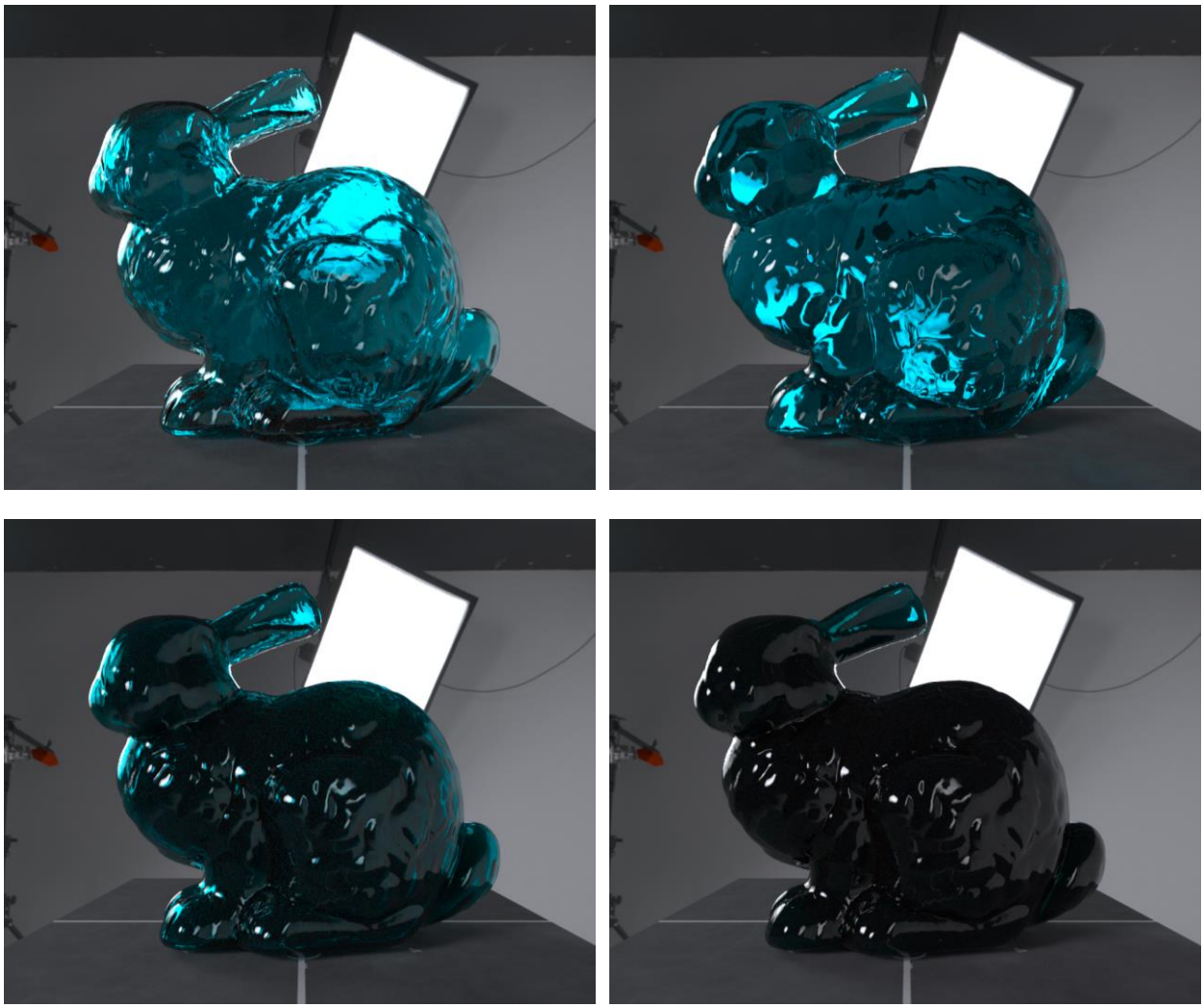


Figure 36. Warm stained-glass palette rendered as solid (left) versus hollow (right), under identical settings. Rows show GG495 (yellow), OG530 (amber/orange), and RG610 (deep red). The set spans luminous yellows through dense reds, representative

of conventional stained-glass "warm" chroma. Videos: GG495: https://youtu.be/DKrkaidtp2c, OG530: https://youtu.be/2BgRu1U8K5g, RG610: https://youtu.be/OL5eAq5Mmvg.

Group C — Cool palette: blue, green, deep blue–green (BG3, VG20, BG38)

The cool set targets the characteristic blue and green components of ecclesiastical glazing. BG3 provides a blue reference, VG20 provides a green reference, and BG38 supplies a darker blue–green reference that helps test the transition from vivid cool tints to more strongly attenuating, "heavy" glass. This group is particularly informative in shell mode, where thickness and internal path length vary strongly across the form.



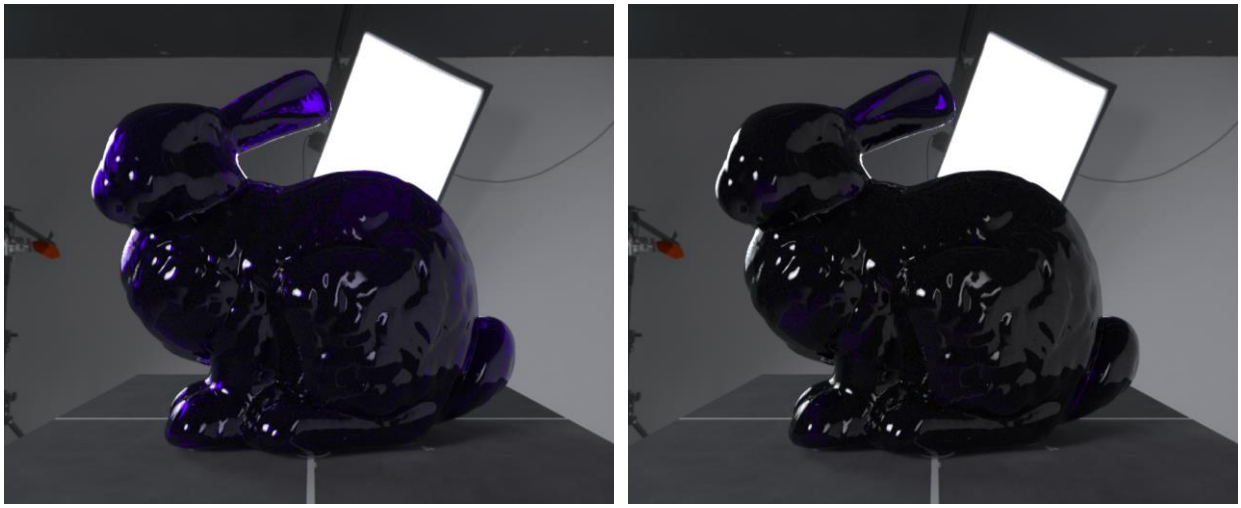


Figure 37. ***Cool stained-glass palette rendered as solid (left) versus hollow (right), under identical settings. Rows show BG3 (blue), VG20 (green), and BG38 (blue–green / strongly attenuating cool glass). This set probes saturation and thickness dependence in the cool chromatic range.*** **Videos: BG3: https://youtu.be/nF5eZfSubQM, VG20: https://youtu.be/PF36guklivo, BG38: https://youtu.be/QDedm2t4vEY.**

Discussion

These results establish a physically grounded palette that is suitable for compositional work: the colours are not arbitrary RGB picks, but emerge from spectral attenuation and geometric thickness, so they respond correctly to illumination changes and viewing conditions. This is essential for stained-glass compositions, where perceptual plausibility depends on the coupled effects of material spectrum × thickness × daylight spectrum.

Besides testing the extremes of the chromatic spectrum, the above colour palette is aligned with the dominant colours of historical and conventional church glazing; the subsequent stained-glass composition previews can remain both aesthetically authentic and optically consistent. In modern designs, particularly in indoor decoration, lighter tints are used, as demonstrated for very light green, blue, and red below.

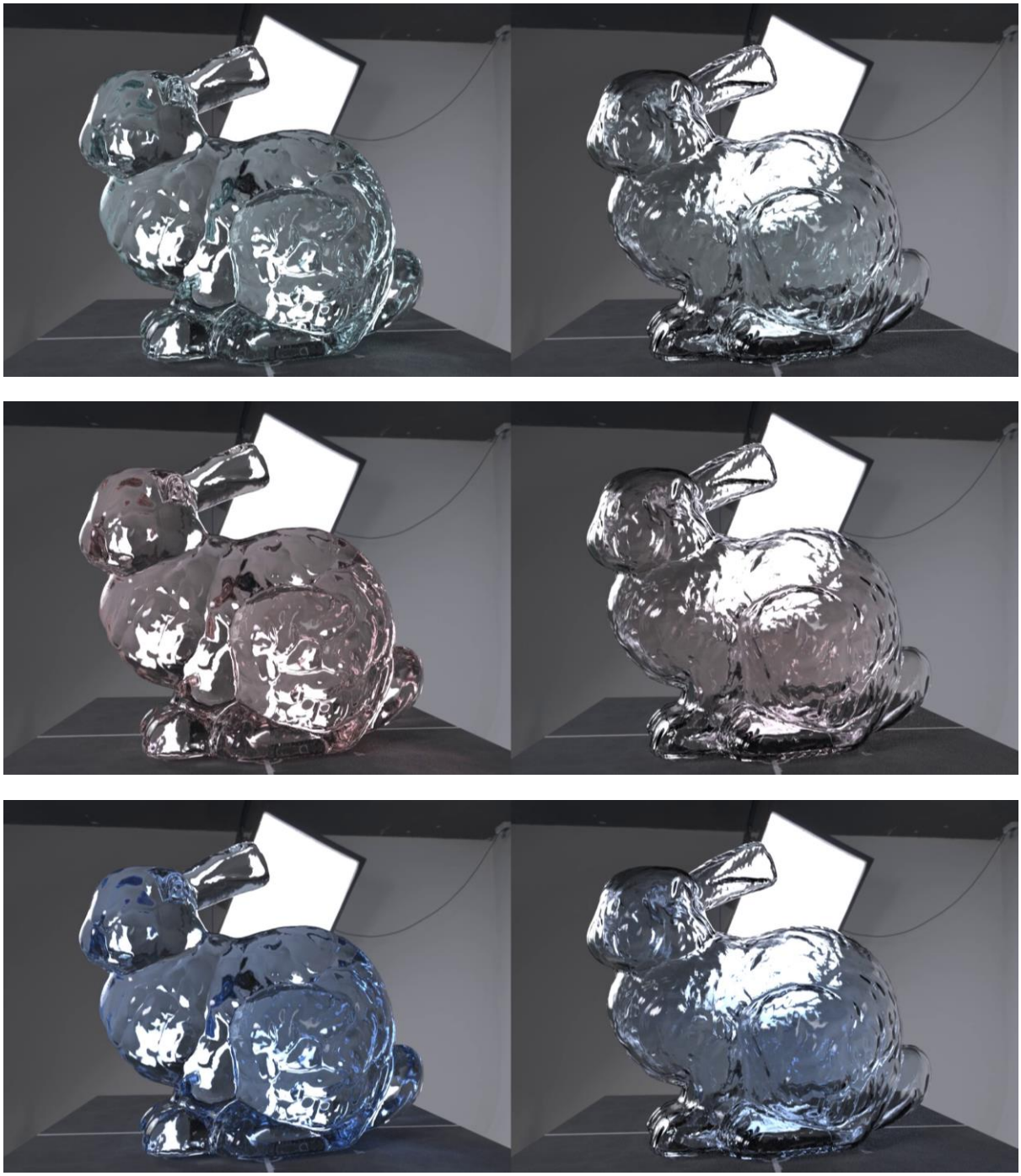


Figure 38. Red tint (solid vs hollow). RG610, with a solid shown left and a hollow 2.5 mm shell shown right, under identical conditions. Video:. Light: green: https://youtu.be/4Y8puW5iT30 Pink: https://youtu.be/32L8-_XZkt8 Light blue: https://youtu.be/pyhfXR6vL0E

Interface roughness (specular lobe size)

Increasing roughness progressively blurs environment reflections, softens caustics, and shifts the overall impression from polished to frosted/etched glass. Crucially, this parameter is largely orthogonal to bulk absorption/scattering: it provides an independent control axis that corresponds well to craft finishing operations (polishing, sandblasting, acid etching).

pecular lobe width at the air–glass interface. Increasing roughness progressively blurs environment reflections, softens caustics, and shifts the overall impression from polished to frosted/etched glass. Crucially, this parameter is largely orthogonal to bulk absorption/scattering: it provides an independent control axis that corresponds well to craft finishing operations (polishing, sandblasting, acid etching).

Figure 39 solates interface finish by holding geometry and bulk properties fixed and sweeping surface roughness from polished to frosted, using a deliberately simple shape to make the reflection blur and highlight spread easy to read.

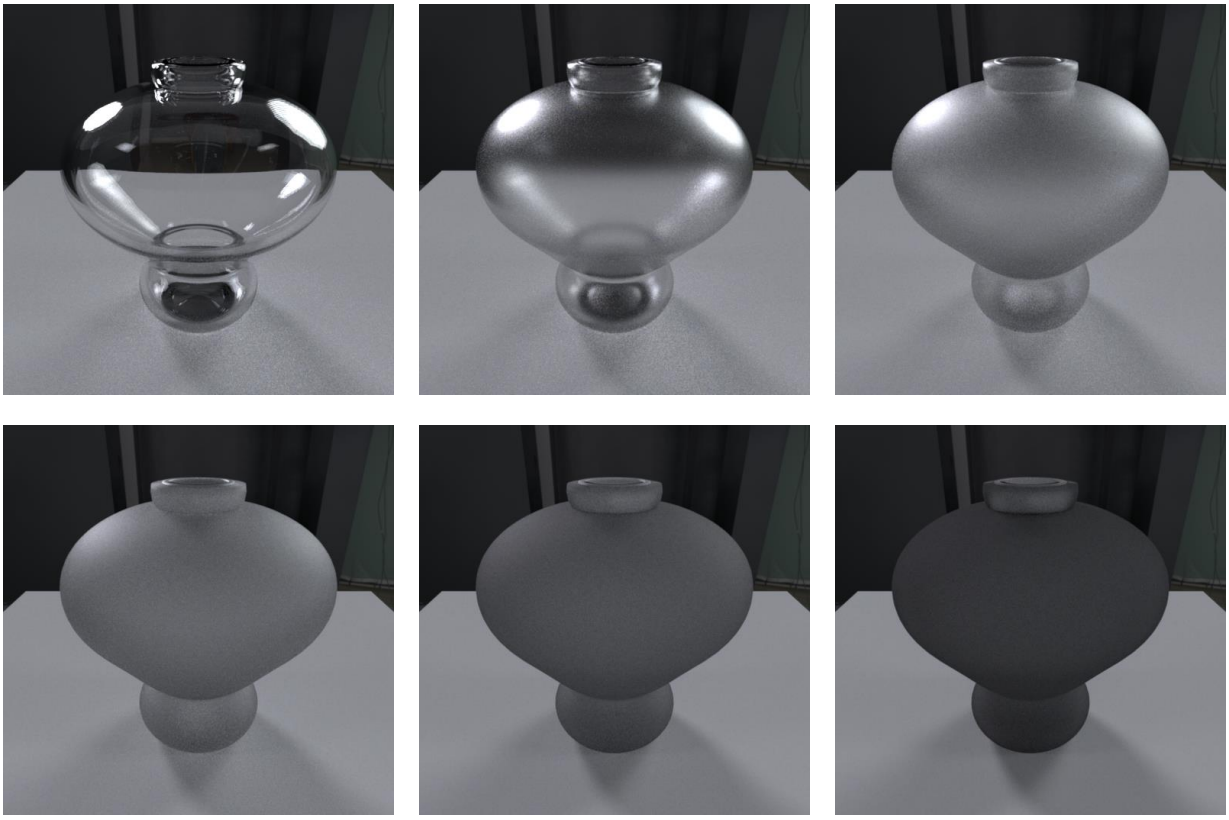


Figure 39. Effect of interface roughness (from polished to frosted). Same geometry, illumination, and bulk parameters; only surface roughness is varied. Top: progression from: polished, intermediate roughness, frosted appearance. Bottom: extreme roughness progression.

Scattering

Bulk scattering controls the appearance of haze and subsurface light redistribution inside the medium. We sweep the anisotropy difference for backward vs isotropic vs forward scattering. In the solid object, longer internal travel typically amplifies the visual impact of scattering (more opportunities for redirection), while the thin shell tends to preserve more direct transmission and boundary-driven effects.

Figure 40 varies scattering anisotropy to show how the same nominal material can shift between a milky, haze-dominated look and a clearer, more directional transmission, again with a systematic contrast between solid and hollow due to differing internal path lengths.

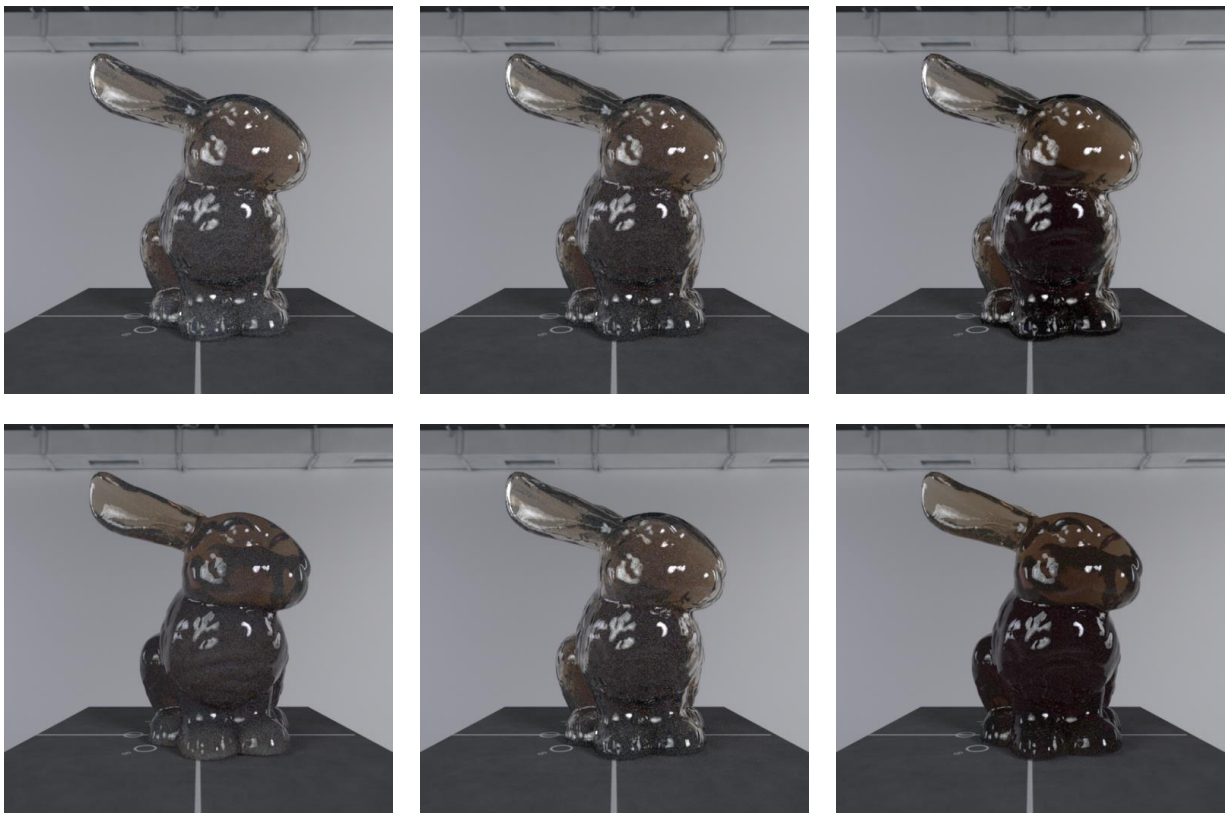


Figure 40. Effect of bulk-scattering anisotropy on glass appearance (solid vs hollow). Top row: hollow 2.5 mm shell; bottom row: solid. Left to right: g = −0.5 (backwards-biased), g = 0.0 (isotropic), g = 0.5 (forward-biased). Geometry, illumination, and interface finish are held constant to isolate the perceptual impact of bulk scattering.

These result sweeps validate that (i) changing the strength of absorption-driven darkening and tint saturation; (ii) interface roughness provides an independent control that maps directly to craft finishing operations and strongly affects highlight structure and reflection sharpness; and (iii) bulk scattering parameters control the transition from clear transmission to haze-dominated appearance, with the solid variant typically exhibiting stronger scattering effects due to longer internal transport. Together, these controls support systematic, craft-relevant studies of transparent and translucent materials under comparable rendering conditions.

5.4.5 Discussion

The results illustrate the central rendering principle for transparent and translucent materials: appearance is governed jointly by interface behaviour and volumetric transport, and both are modulated by the effective optical path length created by the object's geometry. This is why the same external shape can exhibit markedly different colour intensity and brightness when rendered as a solid volume versus a thin hollow shell; see Figure 35 through Figure 38. The proposed approach renders realistically solid bodies, where rays frequently traverse longer distances through the medium and bounce internally; thus, absorption-driven effects accumulate to produce a darker, more saturated appearance. In the hollow case, the shorter path length reduces attenuation, yielding a brighter and less saturated result, with boundary-driven effects, i.e. highlights and Fresnel reflections, becoming prominent (as expected).

This observation is informative for tinted glasses (e.g. Figure 37, and Figure 38). Tints are not "surface colours"; they are consequences of wavelength-dependent attenuation integrated along ray paths. As a result, perceived hue and saturation are not only material-dependent but also geometry-dependent: the same glass can appear "washed out" in thin regions and much denser in thick regions. From a craft perspective, this supports controlled "what-if" studies that separate *material choice* from *design choice*, reducing the requirements for physical samples.

The interface roughness modulation (Figure 39) highlights an orthogonal control axis that maps well to craft finishing operations. Increasing roughness broadens the specular lobe and progressively blurs environment reflections, shifting the appearance from polished to frosted/etched. Importantly, this behaviour is largely independent of bulk absorption: it is an interface phenomenon that can be tuned without changing the underlying volumetric parameters. This separability is practical in authoring: users can reason about finish separately from glass type, and the renderer exposes both dimensions explicitly.

The scattering anisotropy sweep (Figure 40) demonstrates that translucency is not a single knob. By varying anisotropy, the medium transitions between visually distinct regimes: from haze that redirects light more evenly to forward-biased transport that preserves directional transmission while still softening internal contrast. As with absorption, the solid-versus-hollow contrast amplifies the perceptual impact: longer internal travel in the solid increases the opportunities for scattering interactions, so scattering effects typically read as stronger in the solid than in the thin shell.

These results justify the two modelling choices adopted that we adopted:

1. Explicitly modelling hollow objects, rather than treating a single surface as implicitly thick, is essential for varying thickness, which is common in handcrafted products.
2. Our toolbox can provide control of separable families of material properties independently of shape. In subsequent chapters, this separation supports both design exploration and documentation.

Moreover, although demonstrations standardise illumination for comparability, illumination is part of the specification: the same material can appear substantially different under different environments. For this reason, we adopt environment maps, enabling repeatable comparisons under controlled lighting while also allowing the user to swap lighting presets to test the robustness of the intended appearance.

5.5 Conclusion

Section 7 introduces light transport simulation as a reusable capability for predicting the visual appearance of craft materials under controlled illumination. This section addresses the visual experiences of craft products featuringchallenging materials, as characterised by their colour due to absorption, translucency due to scattering, and surface finish due to interface roughness. The corresponding optics cannot be reduced to surface-only shading without losing essential dependencies on thickness, curvature, and internal transport.

Technically, the section established a practical pipeline that connects (i) geometry assets, (ii) physically grounded material parameterisations, and (iii) repeatable scene and illumination settings, to produce comparable render studies. The controlled experiments show that the toolbox is sensitive and demonstrates realistic rendering in phenomena relevant to the visual appreciation of craft products, and specifically:

1. **Solid vs hollow geometry** strongly modulates appearance through *effective optical path length*, changing tint saturation and perceived darkness/brightness;
2. **Interface roughness** provides an orthogonal “finish” axis (polished → frosted) that directly maps to workshop operations; and
3. **Bulk scattering** and its anisotropy control the transition between clearer transmission and haze-dominated looks, again interacting with path length and thus with geometry.

From an integration viewpoint, the main outcome is not a single set of images but a toolbox: light transport simulation becomes an analysis and communication component that can be invoked wherever visual appearance is decisive. It supports design exploration, documentation, and downstream craft-specific simulators that require credible rendering of transparent or translucent artefacts. In this sense, Section 7 plays the same structural role as Section 6: it defines a standalone technology with clear inputs/outputs that is intended to be reused across multiple demonstrators, enabling consistent appearance prediction across the Craeft toolchain.

6 Reliefs

Section 3 introduced semantic simulation as a disciplined way to connect craft intent to computational operations, by making explicit *what acts on what*, *through which parameters*, and *with which observable effects*. Section 4 then demonstrated how this principle can be grounded in physics through FEM-generated reference data, enabling robust parameter studies and supporting downstream acceleration (e.g., surrogate modelling and real-time approximations). Against that backdrop,

This section converts a 2D drawing, often the most accessible digital artefact produced by a practitioner, educator, or curator, into a 3D relief mesh that evokes engraving on a surface. Conceptually, the input image functions as a compact semantic specification of intended form (edges, silhouettes, linework), and the method operationalises this specification by constructing an explicit depth field (a scalar "height map") and then a polygonal surface. This is aligned with the earlier chapters' emphasis on (i) making intermediate representations explicit (fields, constraints, parameters), and (ii) keeping the transformation chain inspectable and reproducible.

From a Craeft perspective, this utility supports two complementary goals. First, it enables rapid design exploration: users can vary a small, interpretable set of parameters (thresholding, smoothing, depth scaling) and immediately obtain alternative relief outcomes without specialised 3D modelling skills. Second, it enables re-use in the wider simulation/rendering toolchain: the produced meshes can be rendered for communication and training, archived as digital craft assets, or passed downstream as inputs to later modules (e.g., mould generation, composite assemblies, or optical studies), thereby illustrating how the technical contributions of Sections 3-4 are instantiated in concrete authoring workflows.

Finally, Section 5 should be read as a deliberately pragmatic module: it does not attempt to replicate the full physics of engraving, but rather provides a controlled and semantically transparent geometric approximation that is suitable for education, documentation, and craft-oriented design pipelines.

This software is provided in https://github.com/ytzortz/Anaglycraft, under the title *"A Technical Approach for the Generation of 3D Anaglyphs from Images, for the Preservation of Traditional Crafts,"* as part of the Bachelor's Thesis of Yiannis Tzortzakis at the University of Crete (UoC), supervised by Xenophon Zabulis. The online repository contains a user's manual.

6.1 Motivation

The approach to automated polygonal mesh generation based on pixel intensity and depth is situated at the intersection of these two areas. By automatically generating a 3D mesh from a 2D image, it bypasses the need for manual input and specialised expertise required by most traditional 3D modelling methods. The ability to generate reliefs based on 2D image data is related to the engraving domain, particularly to 2D-to-3D techniques such as height maps and distance-based reliefs, offering a simpler, more accessible tool suitable for the target users: artisans, educators, and digital heritage applications.

Conventional digital conversions mimic this by taking the initial binary design (the silhouette) and filtering the image. This is inadequate for generating true relief geometry. Smoothing filters create flat forms that do not reflect the increased depth of coarse structures. The resulting surfaces often look artificial.

We overcome this limitation by employing the Distance Transform (DT), a powerful and precise technique from morphological image processing. The DT offers a principle that perfectly models the sculptor's intent because it calculates the shortest distance from every pixel inside the desired form to the form's edge. By interpreting this distance as depth, the result is a structure where pixels closest to the centre of the form are automatically assigned the greatest depth, while those at the edges are assigned the least.

The proposed method calculates the distance of each pixel to the nearest edge, creating reliefs that highlight contours rather than uniformly filling interior regions. This results in depth distributions that mirror real engraving, where edges and linework dominate visual impact. The process generates smooth, continuous, concave slopes for every feature. By using the DT, we bypass the visual guesswork and inefficiency of simple smoothing, ensuring that the generated depth map distributes the available depth to mimic real engraved structures.

6.2 Method

The system follows a step-by-step process, including loading the image, applying thresholding, generating depth information, and exporting the final 3D model as an OBJ file. Users can customise various settings, such as smoothing methods and depth levels, to influence the final output. This chapter explains the methodology and technical details of the implementation.

6.3 Preprocessing

Before relief generation, the system converts the input image to grayscale and applies optional padding. This step ensures that the design is cleanly separated from its background and that sufficient margins are preserved for depth calculations. Thresholding distinguishes foreground from background, establishing the pixel intensity map that drives the subsequent depth assignment. From a craft perspective, this stage is the digital equivalent of preparing a clean working surface, so that the relief corresponds to the intended drawing rather than photographic or printing artefacts. Typical filtering methods (Gaussian, median) are available for this purpose. The choice of blur method can serve as a means for stylistic exploration before engraving. For educators, comparing outcomes can demonstrate how tool selection influences design interpretations.

6.4 Depth Mapping

Relief generation begins with the construction of a depth map, which acts as the foundation for the final 3D model. In this phase, the system translates the visual information from the 2D input image into physical height values. To support various artistic goals, the system provides three selectable preprocessing methods.

Each method encodes a distinct assumption regarding how visual details, such as lines, shading, or silhouettes, should be converted into depth. This choice allows the user to determine the stylistic character of the final engraving, ranging from photographic realism to sharp, graphic stencils.

6.5 Conventional Method

Pixel intensity values are interpreted as proportional depths: darker tones indicate deeper incisions; lighter tones rise closer to the surface. The result is a continuous mapping that preserves tonal gradients in the source image. This approach aligns with photographic logic, where shading is taken as a cue for relief. It is well-suited for motifs that rely on gradual transitions rather than sharp boundaries. Typically, this method is applied to a thresholded, two-level mask. The relief depth is uniformly assigned to both the foreground and background regions of this mask. The image is then smoothed to generate grayscale values, which subsequently guide the creation of the depth map.

6.6 Proposed Method

The DT computes, for each pixel, the distance to the nearest edge. When applied to engraving simulation, the DT generates depth maps where relief is emphasised at contours and gradually diminishes toward interior regions. Unlike smoothing, which distributes intensity smoothly across the whole image, the DT prioritises edges, creating reliefs that closely resemble line-based engraving practices. The Distance Transform generates outcomes that mirror traditional engraving methods, where edges dictate depth. The resulting map emphasises boundaries: contours are raised more prominently than flat areas. In effect, the DT approach reproduces a principle familiar to engraving practice, where the line and its edge define visual weight, while interiors recede into secondary importance.

6.7 Comparative Demonstration

The two methods above are exemplified for two designs.

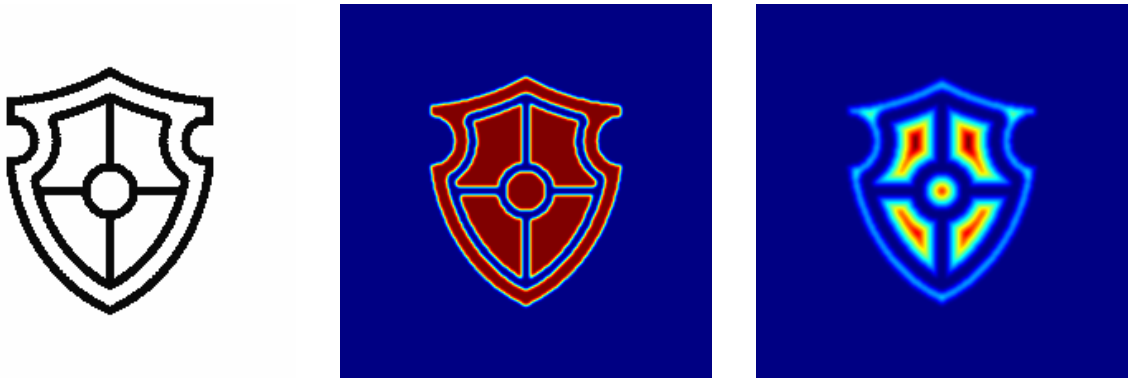
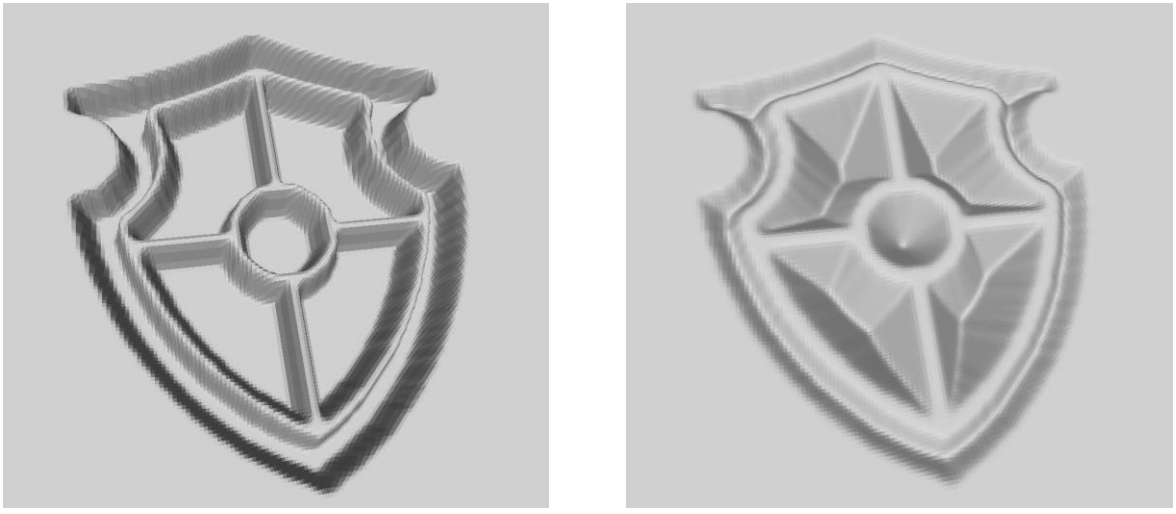


Figure 41. Comparative illustrations of the conventional and the proposed image processing methods for two designs (top and bottom). Left: original pattern. Middle: conventional method, Right: proposed method.

6.8 Meshing

Once the depth map is calculated, it is converted into a polygonal mesh by generating vertices for each pixel and connecting them into triangular faces. The resulting mesh encodes both the geometry of the surface and the stylistic features of the original design, transformed into three-dimensional form. The mesh is exported as an OBJ file, a widely supported standard that can be opened in 3D modelling software, fabrication workflows (CNC, 3D printing), or digital preservation platforms. In practice, this step provides the bridge between a digital preview and tangible application, enabling artisans to fabricate, educators to demonstrate, and archivists to preserve engravings in durable 3D formats. The conversion of depth maps to surfaces is exemplified in Figure 42 below.



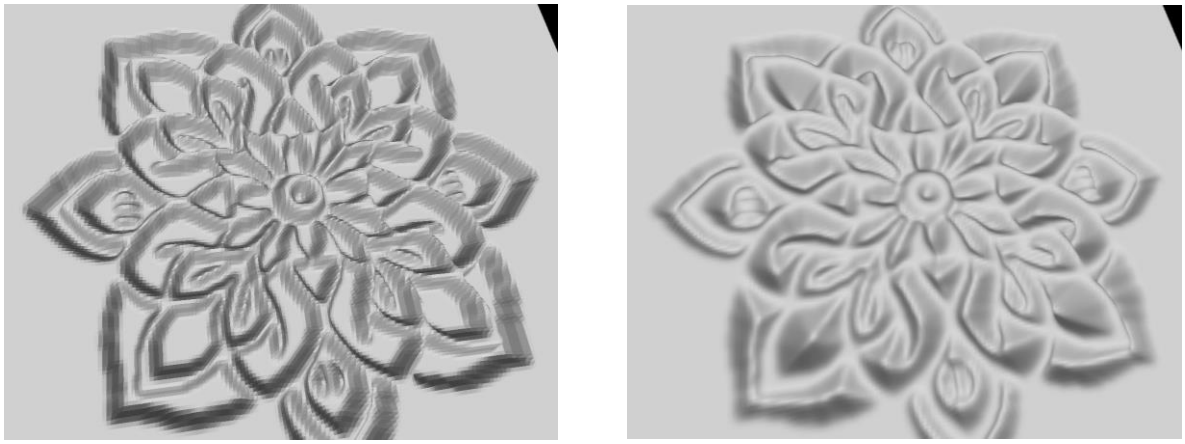


Figure 42. Conversion of the depth maps of the previous example to surface reliefs using the conventional (left) and the proposed (right) methods.

6.9 Usage

The utility processes the image based on the configuration parameters.

Once the configuration file is set, the program processes the image through several steps. First, it validates the configuration file and its contents. Next, it loads the image and applies a greyscale filter, followed by adding white padding around the image based on the padding parameter specified in the configuration.

The generator then calculates the depth map and constructs the 3D mesh. Finally, it exports the 3D model as an OBJ file. During this process, the appropriate vertices are grouped into faces, and a final list of triangles is created. Finally, the program writes all the vertices and faces to an OBJ file, named according to the configuration, using the proper syntax for formatting the OBJ file.

The program calculates the depth of each pixel (vertex) using the depth parameter of the configuration file, which defines the maximum depth that can be 'carved' into the 3D model. Additionally, the threshold and blurring methods are applied to generate a 2D matrix (x, y) containing normalised intensity values. This matrix determines which areas of the image will be carved into the 3D model and to what extent, based on the calculated depth.

The software operates in two modes: Automatic mode, where the parameters are read from a configuration file, and Manual mode, where users can input their settings through a graphical user interface (GUI). In programming mode, a configuration file is generated based on the user's input and used in the conversion process.

The GUI (see Figure 43) is optional and facilitates parameter tuning. It is an intuitive means of configuring the image path, output path, depth, padding, threshold, and blur method. The GUI makes the tool accessible to non-technical users.

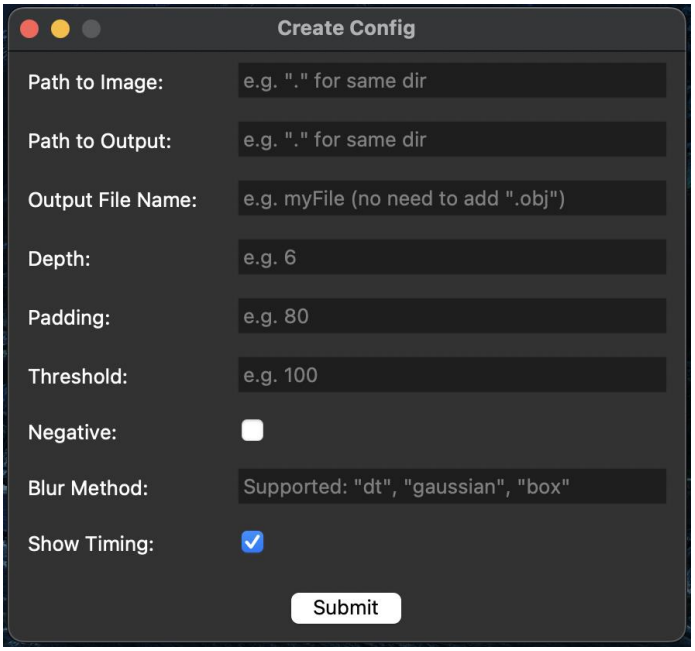


Figure 43. Graphical User Interface (GUI) of the engraving previewer.

Once model generation is complete, the final output is an OBJ file that can be used in various 3D modelling applications.

6.10 Validation

The valuation was conducted on two different computers, a MacBook and a Windows Laptop. It demonstrated consistent and predictable performance across both hardware configurations, despite the MacBook being significantly faster in absolute terms. The testing of 17 configurations revealed that the tool's computational behaviour is primarily influenced by image dimensions, with the effect of parameters remaining consistent across platforms. This strong proportionality and platform-agnostic reliability enable the tool to be safely adopted by institutions with varied hardware resources, allowing for longer runtimes when processing high-resolution archival designs, without sacrificing image quality.

The configurations varied the following parameters:

- Method: Gaussian Blur, Distance Transform.
- Intensity threshold: Values of 70, 100 (default), 150, 190, and 210.
- Padding: Values of 30, 80 (default), 150, and 300 pixels.
- Depth: Values of 3, 6 (default), 10, 15, and 20.
- Negative Filter: Enabled or Disabled (default).

These configurations were designed to explore the system’s sensitivity to key parameters for different images. For example, the blur method was varied to evaluate the impact of different edge-smoothing techniques. The threshold was adjusted to test the system’s ability to handle varying levels of detail and contrast. The padding parameter was used to assess the effect of border size on processing time and model quality. The depth parameter was modified to determine how the number of layers influences the

complexity and processing time of the 3D model. The negative filter was toggled to explore the impact of inverting image intensities on the final output. The images were chosen to represent a wide range of use cases, from small icons (e.g., Image 1) to higher-resolution images (e.g., Image 2). The configurations were systematically selected to isolate the impact of each parameter on performance and output quality.

The threshold parameter regulates the system's sensitivity to fluctuations in pixel intensity. A heightened threshold value (e.g., 210) yields models with greater detail, thereby capturing more intricate features. Conversely, a diminished threshold value (e.g., 70) results in smoother models, albeit potentially at the cost of subtle details. The default threshold (100) strikes a balance between detail preservation and noise reduction. The validation confirmed that higher intensity thresholds preserve fine details, while lower thresholds smoothed results. Increasing the threshold value preserves finer details but may introduce noise, while lower values smooth the model at the cost of subtle features.

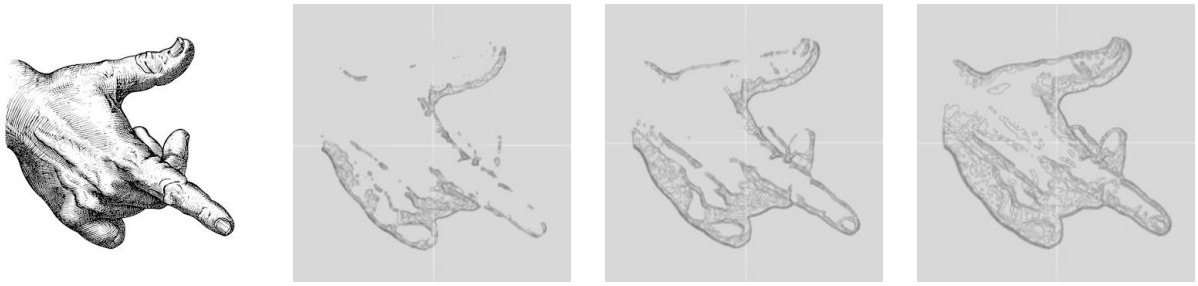


Figure 44. Effect of intensity threshold parameter. Values, from left to right: 100 (Default), 150, and 210.

The depth parameter determines the number of layers in the 3D model, directly influencing its verticality. Greater depth values yielded more pronounced reliefs, while shallow depths produced flatter, subtler surfaces. Higher depth values produce more pronounced reliefs with sharper vertical variation, while lower depths create flatter models. This allows for previewing carving depth and understanding material implications. A higher depth value (e.g., 20) results in a more pronounced 3D effect, with greater vertical variation and finer details in the raised areas. Conversely, a lower depth value (e.g., 3) produces a flatter model with less vertical variation, smoothing out the finer details.

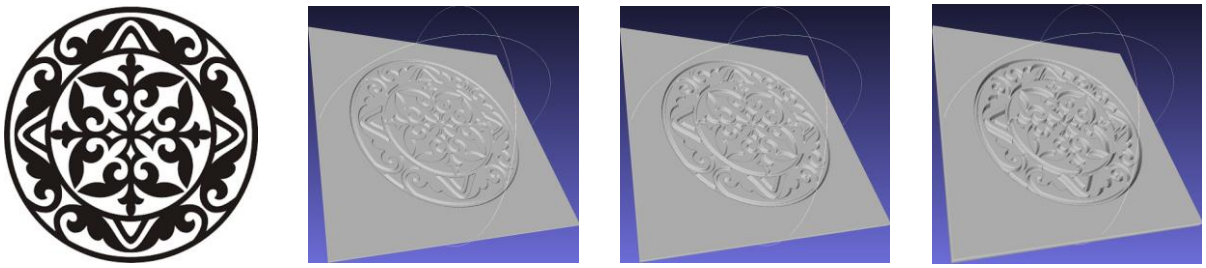


Figure 45. Effect of depth parameter on 3D relief outputs, image #5. Depth = 3, 6 (Default), 10, and 20.

Inverting the depth allowed dark regions to appear raised and light recessed areas. Enabling the negative filter inverts depth interpretation, causing dark regions to be raised and light areas to be recessed.

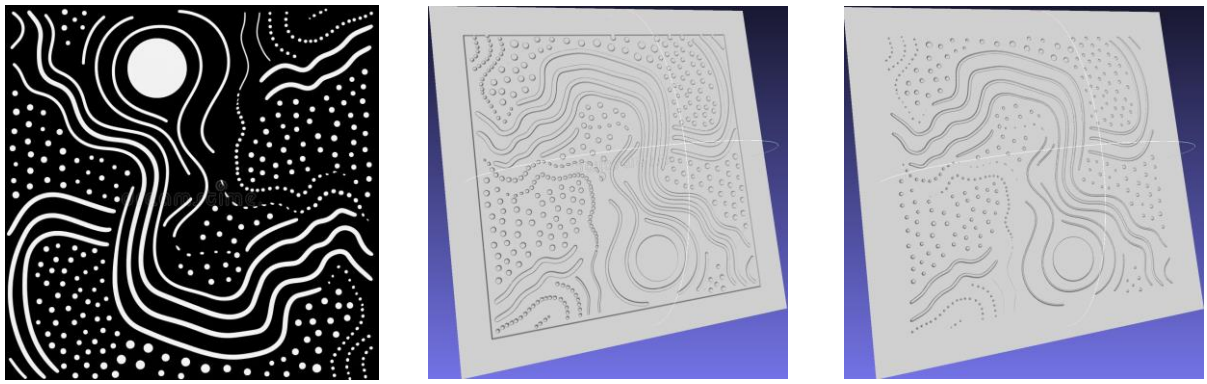
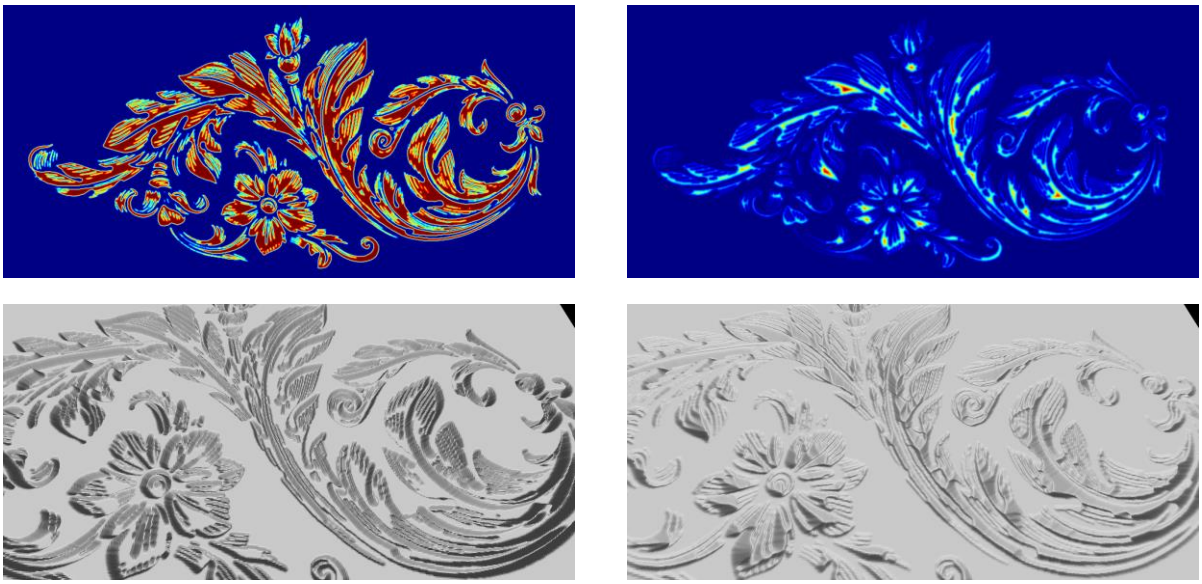


Figure 46. Effect of the negative filter on image #2. Negative Filter = Off (Left), = On (Right)

When the negative filter is enabled, the pixel intensities are inverted, causing the black regions to appear raised and the white areas to appear recessed.

Visual inspection confirmed that the system reliably translated 2D designs into coherent 3D reliefs across parameter variations. In the example below, we passed the results as input to the visualisation toolbox to simulate the metallic appearance of the engraving.

To conclude the parameter study, we provide a compact side-by-side comparison between a conventional relief-from-image pipeline and the method proposed in this section. The aim is to make differences visible both at the intermediate representation level (the depth mapping) and at the user-facing outcome level (a rendered “engraving preview”). In addition, this example demonstrates the intended integration with the visualisation toolbox introduced earlier: the same generated relief can be rendered under a metallic material to communicate the final appearance to makers and educators.



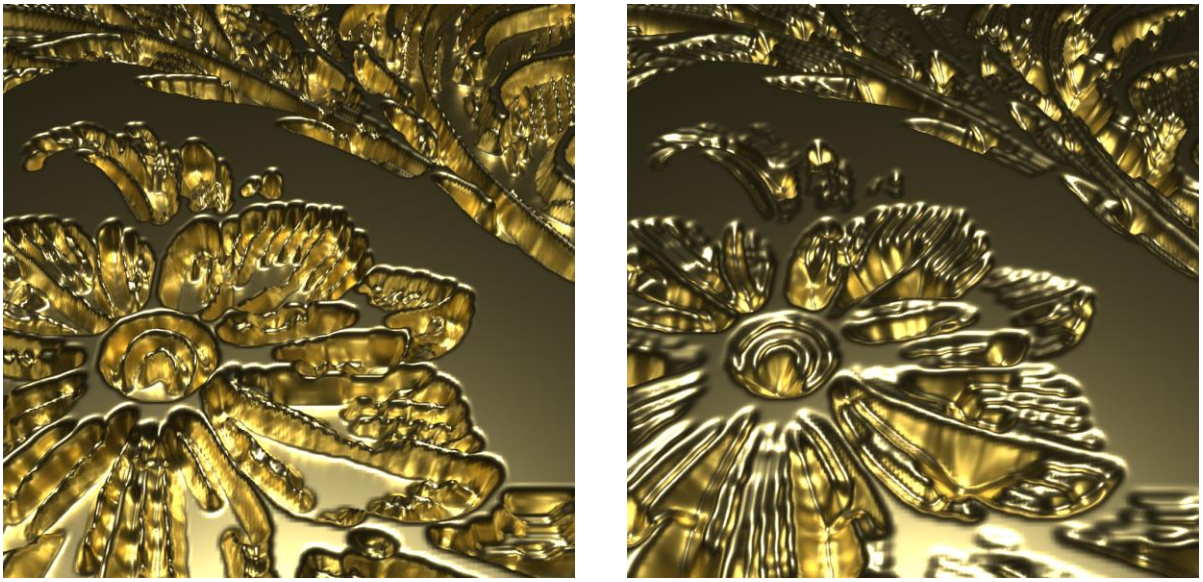


Figure 47. Comparative assessment of the conventional and proposed relief pipelines. Left: conventional method; right: proposed method. The top row shows the resulting depth mapping, and the bottom row shows the corresponding engraved appearance when the relief is rendered as a metallic surface (gold).

While Figure 32 focuses on method comparison, Figure 33 illustrates the practical "preview" capability: once a relief is generated, it can be rendered as an engraving across different metal appearances and lighting conditions. This supports rapid design iteration (e.g., checking whether linework survives the chosen depth/blur settings), and it provides a consistent visual language for documentation and teaching. A texture key is included to clarify the range of engraved finishes that can be communicated through the renderer.

Tin punching (also encountered as *punched tin*, *pierced tin*, and in tinsmithing traditions) is a canonical surface-deformation craft: the maker transfers a 2D motif onto a thin metal sheet, then uses a punch, with a hammer and a supportive backing. The goal is to produce localised indentations and/or perforations that collectively form a decorative pattern. This is exploited in lanterns and panels because the geometry interacts strongly with grazing light and (when perforated) transmitted light.

From a modelling perspective, tin punching is an instructive stress-test for our relief method because its "meaning" is carried by *local depth discontinuities and edge-defined features*. This uses the rationale for using a distance transform (DT) in depth mapping: DT computes, for each pixel, the distance to the nearest edge and thereby allocates depth in an edge-governed manner, producing relief that emphasises contours and progressively diminishes toward interiors. This effect that mirrors how punch marks and chased boundaries dominate visual weight in real metalwork.

In the validation figurgns as *inputs* to the same relief pipeline. The intermediate depth maps make the transformation legible, while the final PBR views demonstrate the perceptual consequence: specular highlights and shadowing on a metallic surface strongly amplify small

geometric differences, making this an effective “generality check” for surface-deformation modelling beyond engraving-style examples.

The relief software is not specific to engraved lookups but generalises to other surface-deformation crafts. Here it is applies it to tin punching motifs. Tin punching is produced by striking repeated tool marks into thin sheet metal (often forming perforations or embossed dimples), and the resulting appearance is governed by fine-scale geometry under specular reflection and grazing illumination. The following figure presents (i) representative tin punching designs, (ii) intermediate depth-map products of our pipeline, and (iii) photorealistic (PBR) renders of the resulting reliefs as metallic sheets, illustrating how the same method supports design preview and parameter inspection for an adjacent craft domain.

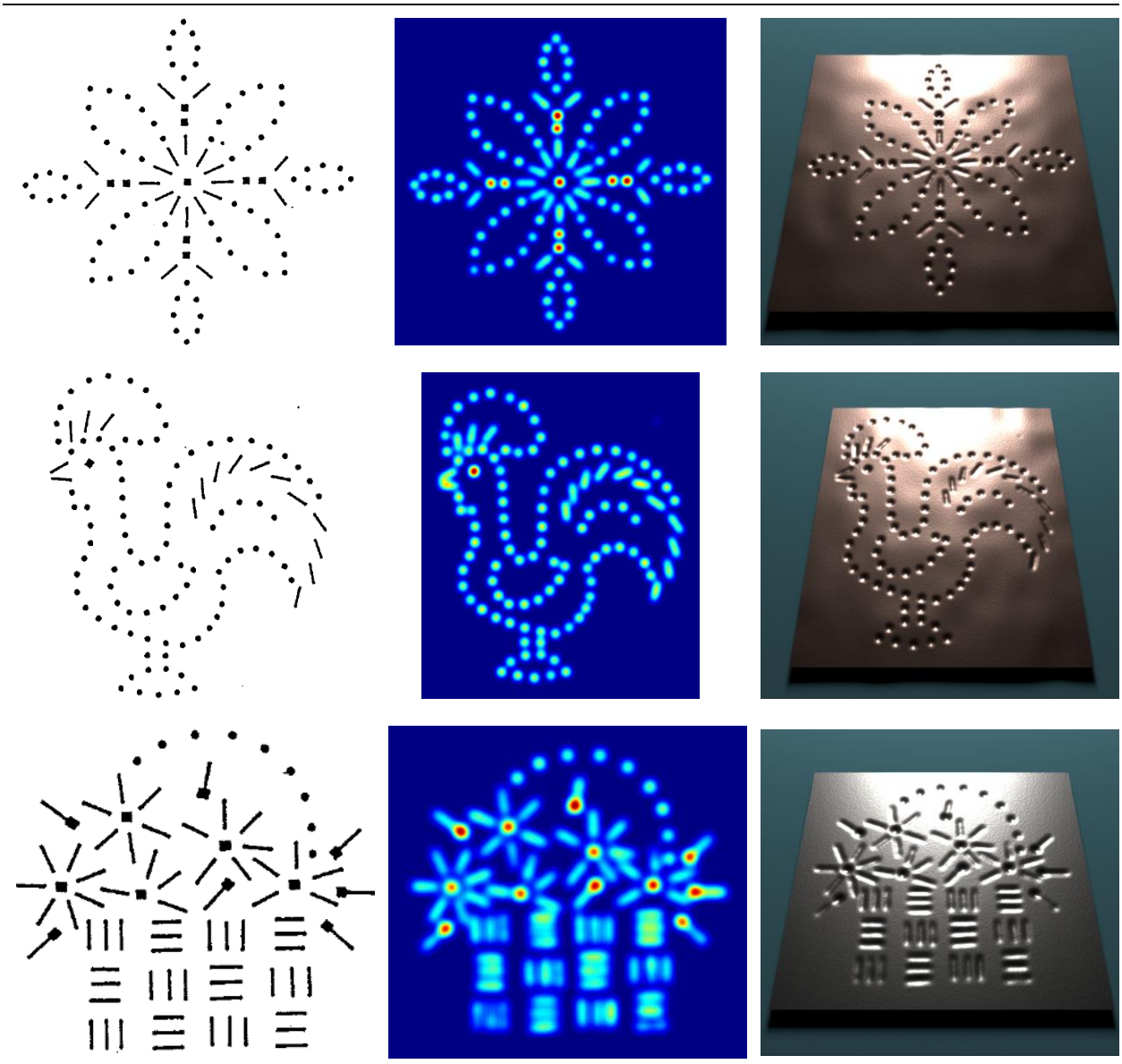


Figure 48. Tin punching validation. Left: example tin punching motifs (2D designs). Middle: intermediate depth maps produced by the relief pipeline (showing the edge-conditioned depth structure that drives the surface deformation). Right: PBR

renderings of the generated reliefs on a metallic material, illustrating how small geometric differences translate into perceptually salient highlights and shadows—supporting the generality of the method for surface-deformation crafts..

The preceding examples demonstrate the qualitative plausibility of the generated reliefs. The operational envelope is shown in following figure which functions as a palette of relief outcomes, where we systematically vary width and intensity. The method supports controlled parametric modulation of a single design without changing its topology. The range shown illustrates the breadth of achievable surface deformations under consistent rendering conditions.

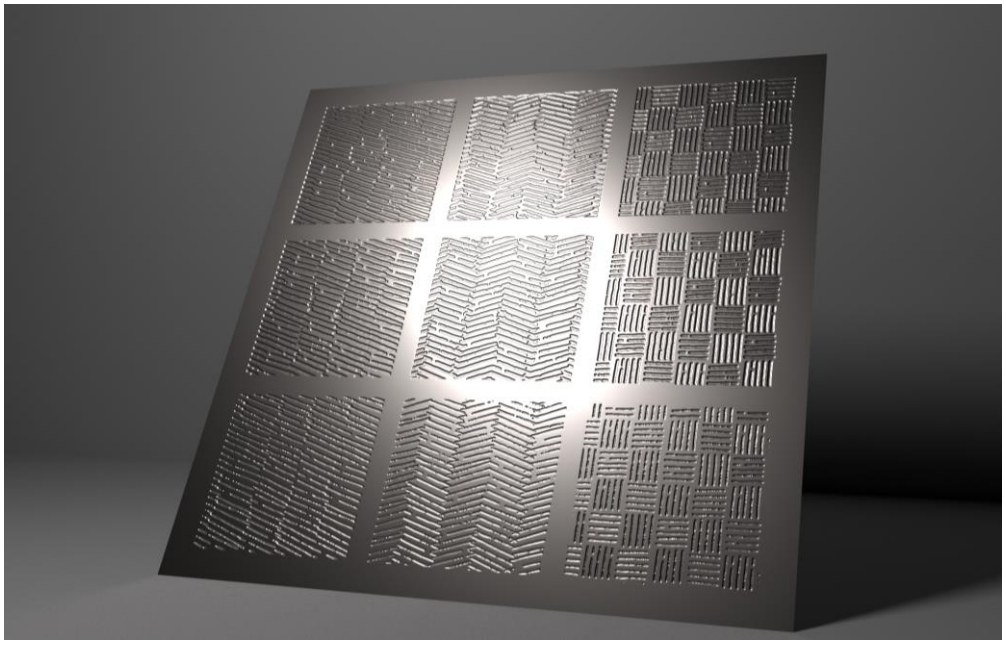


Figure 49. Rendered outputs of the engraving previewer. Top: Grid of relief variants generated from the same design by varying the carving width and carving intensity: a texture key showing representative engraved texture patterns. Bottom: example metallic engravings on different materials and finishes.

These renderings illustrate how threshold, depth, blur, and polarity variations affect the final engraved appearance. For artisans, they provide a preview of design outcomes before material production; for educators, they serve as teaching aids in demonstrating engraving principles; and for preservation specialists, they capture digital surrogates of stylistic variations for archiving.

6.11 Conclusion

This section presents a complete and reproducible pipeline for generating 3D relief geometry from 2D images, with the explicit goal of supporting craft-oriented design exploration and communicative visualisation. In contrast to treating “relief generation” as a single opaque conversion, the approach makes each intermediate step semantically interpretable: an input drawing is converted into an explicit depth field, the field is filtered and scaled through user-controlled parameters, and the final mesh is produced as a surface suitable for downstream use. This mirrors the broader technical stance of the deliverable: transformations should be modular, parameterised, and traceable so that results can be explained, repeated, and adapted.

A key outcome is that the pipeline supports rapid iteration without requiring specialist 3D modelling expertise. The parameters exposed (e.g., thresholding/edge emphasis, smoothing/blur, depth scaling, and optional post-processing) correspond to choices that practitioners and designers can reason about directly: they control how strongly a line becomes a ridge, how continuous the transitions appear, and how prominent the engraving reads at typical viewing distances. The comparative examples show that these controls improve stability and visual legibility of the produced reliefs, particularly when the input drawings contain thin strokes, high contrast edges, or heterogeneous line density.

Finally, the section demonstrated how the generated reliefs integrate with the earlier contributions on visualisation and simulation. The relief meshes can be rendered as engravings using physically based materials to provide credible previews for documentation, teaching, and design review, and they can be treated as geometric assets for later modules (e.g., mould generation, composite assemblies, or further physical simulation). In this sense, Section 5 functions as the first concrete craft application of the semantic and computational foundations established in Sections 3–4: it operationalises a craft-relevant input (a 2D drawing) into a reusable 3D representation, with a transparent chain of decisions and outputs suitable for the broader Craeft toolchain.

7 Solids by revolution

Many artefacts central to craft practice are approximately axisymmetric: vessels thrown on a wheel, turned wooden forms, plaster masters, blown-glass profiles, and a wide class of mould-ready geometries. Modelling these objects as general 3D meshes is possible, but it is often unnecessarily expensive for interactive workflows, because the user's edits and the object's salient shape variation occur primarily in a 2D radial profile. This section, therefore, exploits rotational symmetry to obtain a compact representation that is computationally efficient, easy to edit, and straightforward to convert into 3D geometry when needed.

A solid of revolution can be described by revolving a planar region around an axis. In practical terms, this means that the entire 3D object can be generated from a single template curve (or silhouette) in 2D: the curve defines the outer surface profile, and a 360° rotation produces the final volume or shell. This reduction is particularly valuable in craft simulation because it supports interactive manipulation: the user edits one contour instead of a full mesh, while remaining compatible with downstream tasks such as mesh generation, collision handling, fabrication-oriented checks, and physics-based or appearance-based evaluation.

An algorithmic approach is a proposed method for generating 3D models that represent the geometry of solids by revolution. The approach adopted here specialises a generic 3D representation into a 2D binary template: a matrix in which pixels encode material occupancy (material vs empty space). Interaction is intentionally constrained to the outer contour of this template, reflecting the common situation in turning- and wheel-based processes where the practitioner primarily controls the exterior profile. Once edited, the template can be revolved to produce a 3D surface (or volume), enabling subsequent processing and visualisation using the shared infrastructure of the deliverable.

This section is also a methodological bridge to later chapters. In particular, the axisymmetric modelling formalism is a direct prerequisite for the plaster turning workflow (Section 11), where the craft action is naturally expressed as successive modifications of a profile under rotation.

7.2 Method

7.2.1 Representation

We specialise the generic 3D model into a computationally efficient 2D representation. Since the solids are generated by revolution, they exhibit vertical symmetry. This property is utilised by performing all editing on a hypothetical planar segment, known as the template. The template is a 2D binary image that, when conceptually revolved 360 degrees around its first column (which aligns with the Y-axis in 3D space), produces the 3D solid. Interaction is limited to the outer surface, which means editing a single contour within the template image.

The central data structure is a 2D binary matrix (T), where white pixels represent material, while black pixels indicate unoccupied space. The generated 3D model is symmetrical about the Y-axis. The final 3D model has a fixed height of one world unit, with X and Z dimensions proportional to the template's aspect ratio. The model is centred at the origin (0, 0, 0) of the coordinate system.

Volumetric

The template is a Boolean matrix T, where white pixels define the profile to be revolved. Workpiece editing is volumetric, taking place directly in T and propagating symmetrically in 3D. The representation includes calculating the gradient ∇T, which is later used to compute the surface normal vectors of the 3D model. A scaling factor s = 1/H links the real world to pixel dimensions, where H is the height of T.

Surface

The outer contour of the white form in T is traced, using [14], which returns the contour points in spatial order. This contour is Gaussianly smoothed and interpolated. When spun around its left vertical border by a full circle, its trace defines the workpiece's symmetrical volume. Using v rotational steps (default v = 72), a 3D triangle mesh is obtained. Each 2D contour point $[u, v]^T$ and its normal $[n_u, n_v]^T = \nabla T(u, v)$ are converted to 3D points $[u \cdot s, v \cdot s, 0]^T$ and normals $[n_u, n_v, 0]^T$, respectively. These oriented points are serially rotated about the Y-axis in v steps. Triangles are then formed from adjacent points in adjacent slices, with care taken to ensure consistent triangle winding across the entire solid.

7.2.2 Editing

When a user makes a change in the virtual environment, this is translated into a modification of the 3D representation via editing the template T.

Volumetric Edits

Subtractive, additive, and mass-preserving (shaping) operations are implemented by subtracting, adding, or moving white pixels in T. All tools affect the pixels within their circular vicinity, denoted as P. Additive and subtractive tools paint pixels white and black, respectively. Shaping tools 'push' (move) pixels near the circumference of P in adjacent locations, with the effect attenuating with distance from P.

Tool Representation

Internally, each tool is represented as a sphere in the 3D world, which is visualised as a circle on the template image. This spherical shape is chosen for its efficiency in collision detection (determining contact with the material). Tools of more complex shapes can be created by combining multiple spheres. Users can apply translations, rotations, and uniform scaling to these tools, which adjust the size and position of the circles on the template.

Shavings treatment

For subtractive and shaping operations, a connected component labelling process is run on T to detect shavings or split pieces. If the template contains multiple connected components, only the one in contact with the axis of revolution (the "base," determined by the bottom-left pixel) is retained. All other components are painted black (discarded) to ensure a single, attached solid.

7.2.3 Tools

Subtractive

The default operation is to paint all template pixels inside the tool's circle black. This action is skipped under two special collision-based conditions, which are intended to prevent the creation of internal holes in the solid volume:

- No Collision: If the circle contains no white pixels (no collision), the change is skipped, and it is noted that the tool made no change (this information can then be used to skip mesh generation).
- Tool Inside Solid: If all pixels inside the circle are white (the tool is fully "inside" the solid), the operation is skipped. This prevents the creation of internal voids, which would lead to unpredictable user interaction.

After the changes, the connected component labelling procedure is run to ensure any resulting split pieces are discarded, retaining only the component connected to the solid's base.

Additive

The default operation is to paint all template pixels inside the tool's circle white. This action is skipped under two special collision-based conditions to conserve computing power:

- No Collision: If all pixels inside the circle are black (no collision), the change is skipped.
- Tool Inside Space: If the template contains no black pixels inside the circle (the tool is entirely within the existing solid volume), the operation is skipped.

After the changes, connected component labelling is run. If the template has ended up with internal 'holes' (e.g., the tool connected previously separated edges of the solid), the tool fills these holes with white pixels.

Mass-Preserving

This tool initially behaves like the subtractive tool (including the special cases and subsequent component discarding if the solid splits).

If only one connected component remains, the tool adds white pixels to restore the solid's mass:

1. The edge pixels (the "surface" of the solid) are identified.
2. These edge pixels are sorted based on their geodesic distance from the pixels where the tool initially collided with the solid.
3. White pixels are added adjacent to the edge pixels, starting with those closest to the collision points and progressing outwards. This approach ensures the solid "thickens" faster near the points of interaction than at other points.
4. The process stops when there are no more edge pixels to add to, or when the template's surface area (i.e., the number of white pixels) equals a variable ε.

The variable ε is initially set to the surface area of the template when the program starts, but it can be changed by certain tools (e.g., when a split component is discarded). Suppose the process runs out of

edge pixels before reaching the ε threshold. In this case, the remaining mass is added the next time the mass-preserving tool is used, ensuring the solid eventually reaches the target area ε.

7.2.4 Mesh Generation and Output

At the end of each frame, the 3D triangle mesh is generated from the smoothed 2D contour.

Algorithm
1. For each 2D contour point p2D = $[x, y]^T$ and its normal N2D = $[nx, ny]^T$, the initial 3D point and normal are calculated as p3D = $[x \cdot s, y \cdot s, 0]^T$ and N3D = $[nx, ny, 0]^T$. 2. These points and normals are rotated around the Y-axis across \nu slices to create the final mesh. 3. The triangle indices array is created by traversing the generated points while maintaining consistent triangle winding. 4. The final 3D triangle mesh is returned to the rendering environment (Unity) as the arrays of points (p), normals (N), and indices (I).

7.2.5 Interfacing for 3D interaction

This subsection defines the interaction mapping between the user's motion in a 3D virtual workspace and the 2D template curve that parametrises a solid of revolution. The core problem is geometric: the user manipulates a controller (or hand proxy) in 3D, but the modelling primitive we require is a planar silhouette, the profile curve that, when rotated about a fixed axis, generates the target volume. The interface, therefore, implements the projection from a 3D interaction point to a 2D point (r,z) on the silhouette plane.

We define a local coordinate frame attached to the modelling widget. Let the axis of revolution be the unit vector (a) passing through the origin point (o). The system also defines a "silhouette plane" spanned by two orthonormal directions. Given a user interaction point (p), the controller tip, we compute its displacement relative to the origin (d = p - o). The axial coordinateis the signed projection onto the revolution axis, $z = d \cdot a$, and the radial coordinate is the distance to the axis, $r = |d - (d \cdot a)a|$.

This ((r,z)) pair is the point that the user "draws" on the 2D template. Intuitively, (z) expresses how far along the axis the user is, and (r) expresses how far from the axis the point lies; rotating this profile around the axis reconstructs the corresponding 3D locus.

In practice, users benefit from a stable editing surface. The interface, therefore, constrains interaction to a designated working region and applies clamping and snapping rules: ($r > 0$) is enforced by construction, (z) is clamped to the template's bounds, and optional snapping aligns samples to existing control points or a discretisation grid. Each interaction sample modifies a piecewise curve representation of the silhouette (e.g., polyline or spline). Once the silhouette is edited, the geometry kernel generates the 3D shape by revolving the curve, producing a watertight mesh or implicit volume depending on the downstream requirements.

Computationally, the goal is to map the user's 3D tool manipulation onto the 2D template T. The sphere S (or sphere cluster) used by the tool is transformed by the matrix Q_t from its local coordinate frame to the 3D world space. For collision detection and modification, the sphere's 3D projection onto the 2D profile T is approximated by a 2D circle.

Algorithm:
The key transformation matrix M takes the tool from its local frame to the solid's local frame: $M = Q_s^{-1} \cdot Q_t$ where Q_s is the transformation matrix of the solid. Using M, we calculate the centre and radius of the sphere in the solid's local coordinate frame: • The sphere's centre in the solid's local frame is $C = M \cdot [0, 0, 0]^T$. • A point on the sphere boundary is $p = M \cdot [0.5, 0, 0]^T$. • The radius of the sphere in the solid's local frame is $R = d(C, p)$, where d is the Euclidean distance. The 2D circle parameters (C, ρ) that define the effective area in the template image T are calculated: • The height of T in pixels is H. • The Y-coordinate of the circle's centre (vertical axis) is $cy = Cy \cdot H$. • The X-coordinate of the circle's centre (horizontal distance from the axis of revolution, exploiting symmetry) is $cx = hypot(Cx, Cz) \cdot H$. • The radius of the circle in pixels is $\rho = R \cdot H$. This circle (C, ρ) defines the specific pixels in T that the tool's operation will modify, dependent on the tool's type (subtractive, additive, or mass-preserving).

7.2.6 API Interfacing

To make the solids-by-revolution module reusable beyond a single demonstrator, we package its functionality behind a clean application programming interface (API). The intent is to separate (i) the *craft-specific interaction logic* and user experience from (ii) the *core geometric operations*—template editing, profile validation, surface-of-revolution construction, meshing, and export. This modularisation allows the same engine to be embedded in multiple craft-oriented applications (e.g., pottery profiles, turned plaster masters, glass forms, ornamental mould components), while keeping a consistent data model and reproducible processing chain.

At the API level, the module exposes a small set of stable primitives: initialise a workspace, define the axis and template frame, ingest interaction samples (e.g., the ((r,z)) points produced by the mapping described in 6.2.5), edit and query the silhouette representation, and generate outputs (mesh/volume, metadata, and optional intermediate artefacts). The API also standardises how parameters are supplied and recorded, such as smoothing strength, sampling density, curve type (polyline/spline), meshing resolution,

and export formats, so that simulator instances remain comparable and results can be traced back to configuration choices.

This API framing supports composition with other Craeft components. The generated geometry can be passed downstream to modules for mould generation, composite assembly, rendering, or physics-based simulation; conversely, upstream tools can provide sketches, constraints, or semantic annotations that drive the same underlying generator. In this sense, 6.2.6 is not an implementation detail but a scalability mechanism: by exposing solids-by-revolution as an API, we enable rapid creation of multiple craft-specific simulators that share a common, tested geometric core while differing in interaction metaphors, templates, constraints, and domain semantics.

Initialisation

The initialisation process configures the environment based on user preferences:

1. Template Allocation: The template image is allocated, with its size and resolution determined by user preferences.
2. Downscaling (Optional): The template can be uniformly downscaled based on an API 'resolution' parameter, reducing execution time and the complexity of the generated 3D model.
3. Tool Setup: Simulated tools are initialised with user-specified types and active status. Each tool is assigned a unique integer identifier for later modification of its parameters.
4. Tool Parameters: Each tool is associated with a 4x4 transformation matrix (Q_t). This matrix maps the tool's 3D shape from its local coordinate frame, with the sphere centre at (0, 0, 0), and a radius of 0.5 units. This transformation can include any combination of translations, rotations, and uniform scaling. An active Boolean flag indicates whether the tool should modify the template at the current frame. The API stores these parameters and allows the user to change them at any time.

Execution Flow

At each simulation frame, the active tools modify the template's pixels according to their type. The user is responsible for replacing the existing 3D solid's triangle mesh in the rendering environment (e.g., Unity) with the new one generated by the API, simultaneously applying the solid's transformation (Q_s). The API can also generate a 2D image showing intermediate results (e.g., smoothed contours, 2D normals) for debugging purposes.

Computational Efficiency

To conserve computational power, the API returns a Boolean flag that indicates whether the mesh has changed since the last frame. The mesh is not recalculated if:

- Algorithm parameters (e.g., resolution, slices) have not changed.
- The transformations of the tools and solids remain unchanged.
- All tools were inactive.
- There were no collisions between the tools and the solid.

In these cases, the last calculated mesh is returned.

7.3 Validation

This section validates the solids-by-revolution module as an interactive modelling component, rather than as an offline geometric converter. Validation therefore focuses on three practical criteria: (i) interaction correctness, i.e., 3D user motion is mapped consistently onto edits of the 2D template T (Section 6.2.5); (ii) representation coherence, i.e., the internal template/contour state and the generated 3D surface remain synchronised under continuous editing; and (iii) real-time performance and usability, i.e., the workpiece can be shaped while the system updates the mesh and render feedback at interactive rates. We demonstrate these properties in two craft contexts, pottery and glassblowing, that share axial symmetry but differ in typical shaping metaphors, tools, and user expectations.

7.3.1 Pottery

Pottery is an appropriate validation domain because the intended interaction is continuous shaping: users expect to “push” and “pull” a clay body while observing immediate geometric feedback. In our implementation, this is realised by updating the 2D template T (binary profile image) according to the tool’s effective footprint and then regenerating the corresponding surface of revolution for rendering.

Figure 50 illustrates the system’s dual representation during interactive modelling. The left panels show the real-time 3D rendering of the evolving clay body, while the right panels show the corresponding internal state maintained by the toolbox (the 2D template/profile representation from which the surface of revolution is generated). The key validation point is that edits applied through the interaction loop update TTT, the contour extraction/smoothing, and the generated mesh in a tightly coupled manner, allowing the user to inspect the result from arbitrary viewpoints while the internal representation remains consistent with the rendered shape.

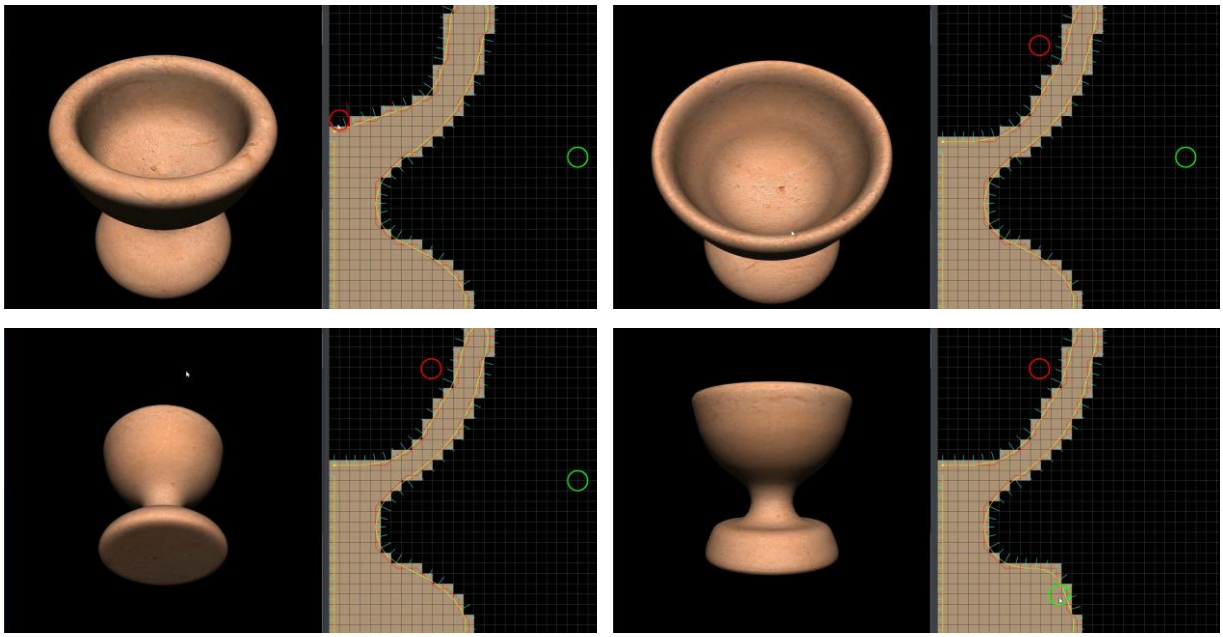


Figure 50. Real-time pottery shaping with synchronised internal representation. Left: interactive 3D render of the workpiece updated frame-by-frame. Right: the corresponding internal template/profile state used by the toolbox to generate the surface

of revolution. The paired views demonstrate coherent updates of representation and rendering during continuous editing in the Design Studio. Video: https://youtu.be/Yc7FtCdOeSs.

To validate the interaction from the user's perspective, Figure 51 shows the first-person view of shaping in which the abstract spherical tool proxy can be replaced by a more legible embodiment—a tracked 3D hand model driven by a controller. This does not change the underlying tool computation (still implemented via sphere/sphere-cluster footprints projected onto T), but it improves interpretability: users see a familiar shaping gesture and can better anticipate the direction and locality of material change. The demonstration, therefore, validates both the robustness of the mapping and the usability of the interaction metaphor in a pottery setting.

Figure 51. First-person interaction view using a hand-shaped tool proxy. The user manipulates a controller to drive a 3D hand model that acts as a shaping tool for the clay body, while the system applies the corresponding template edits and regenerates the surface of revolution in real time. Video: https://youtu.be/W2tzByCnlpc

7.3.2 Glass

The same geometric core applies naturally to glassblowing whenever the target form is approximately axisymmetric (e.g., bulbs, beakers, vases, and intermediate preforms). The validation objective here is cross-domain applicability: the module should support shaping actions that are conceptually aligned with glass practice (pressing, constraining, and adjusting a hot workpiece), while preserving stable behaviour under the continuous rotational symmetry assumptions.

Figure 52 demonstrates the application of the module to glassblowing shaping actions. As in the pottery case, the user's 3D manipulation is mapped onto edits of the 2D template TTT, from which the surface of revolution is generated. The important validation point is that the interaction remains well-behaved under typical glassblowing-style contact and shaping sequences: the workpiece remains coherent (no unintended internal voids), the contour/smoothing pipeline yields stable geometry, and the rendered feedback supports rapid iteration on form.

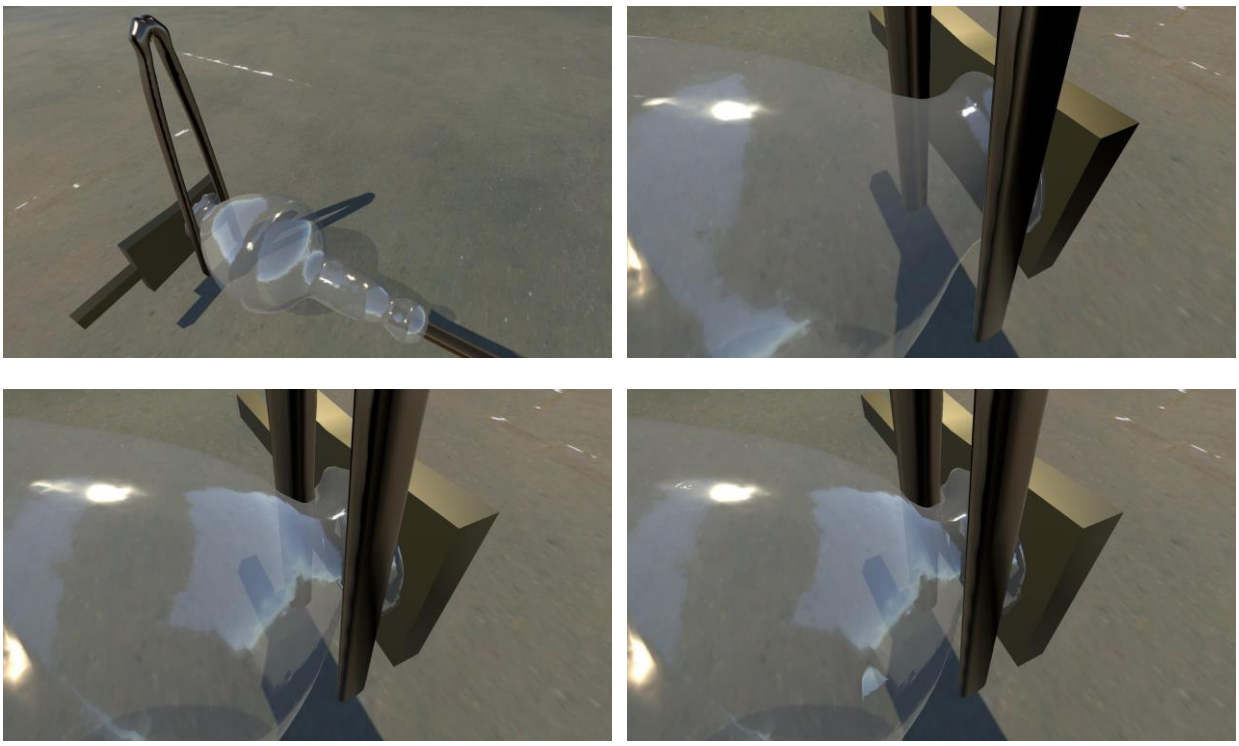


Figure 52. Application to axisymmetric glassblowing forms. Example shaping outcomes generated by mapping 3D interaction to edits of the 2D template/profile and revolving the resulting contour to obtain the workpiece surface. The examples illustrate the portability of the same interaction-and-representation loop across pottery and glass contexts.

7.4 Lathes

This subsection validates the “solids by revolution” interaction model in a realistic lathe setting, where geometry emerges through continuous tool-workpiece contact and where practitioner judgement is explicit. We ground the validation in an ethnographically recorded woodturning episode, then reconstruct it as a Woodturning Virtual Process with traceable workflow nodes, assets (videos), and decision transitions. The resulting package shows how a craft narrative can be turned into (i) a semantic process specification and (ii) an interactive re-enactment, so that each simulated step remains accountable to recorded evidence and to an explicit process logic.

To validate the method on a lathe-centred craft, we used the recording of a complete woodturning episode and re-enacted it in the simulator, maintaining traceability between observed actions, workflow nodes, and simulated operations

7.4.1 Process

This workflow outlines a defines a woodturning virtual process for producing a honey picker, progressing from stock preparation to finishing, with explicit quality-control decision points.

1. Preparation (Risk - Dexterity)

Before it is shaped, the raw wood must be prepared.

- Bark Removal: Removing the outer layer of the wood.
- Left-end Smoothing: Levelling the edges.
- Initial Thinning: Reducing the overall diameter.

This phase is about removing the protective bark to reveal the workable material underneath, then getting the raw material into a workable shape.

- The Craft Action: Preparing the lathe and the initial "roughing out" to create a cylinder.
- Documentation Nodes: wt_v_p_2 (Preparing), wt_v_p_8 (Bark Removal).
- The Digital Twin: Simulations focus on the impact of the gouge against the irregular bark surface.

Primary Assets:

- Real Capture: Video 02 (Lathe Setup), Video 08 (Bark Removal).
- Simulation: Simulated Video 08 (Mimicking bark impact).

2. Sizing (Certainty - Judgement)

Two conditional checks gate the transition from a generic cylinder to a functional blank: if the workpiece is too long, the process branches to shortening; if it is too thick, it branches to additional thinning/detail passes. If neither condition holds, the workflow proceeds directly to detailing.

Condition	Action if True	Action if False
Is it too long?	Shortening: Includes ridge making and specific cutting.	Skip to the next stage.
Is it too thick?	Thinning: Includes further smoothing and detailed carving.	Skip to the next stage.

This phase documents the transition from a generic cylinder to a tool-ready blank.

Action: Evaluating length and thickness; trimming the piece to size.

- Documentation Nodes: wt_v_p_13 (Cutting), wt_v_p_5 (Detailed Thinning).
- Digital Twin: Simulations mimic the "parting tool" actions, calculating the stress required to cleanly cut through the grain to shorten the piece.

Primary Assets:

- Real Capture: Video 13 (The Cut), Video 14 (Thinning).
- Simulation: Simulated Video 13 (The physics of the parting cut).

3. Detailing (Risk - Dexterity)

Once the dimensions are correct, the workflow moves into the specific components of the tool:

- Making the Picker Head: The primary functional end of the tool.
- Making the Handle: Involves two distinct thinning stages to ensure a comfortable grip.
- Refining the Head: A final detail pass on the picker head.

The workpiece takes form. This involves carving the head and the handle.

Action: Making the picker head and sequentially thinning the handle for grip.

- Documentation Nodes: wt_v_p_6 (Handle), wt_v_p_11 (Ridge Making).
- Digital Twin: Detailed simulations of the "ridge making" process, documenting the precision required to create decorative and functional structural lines.

Primary Assets:

- Real Capture: Video 06 (Handle creation), Video 11 (First ridge).
- Simulation: Simulated Videos 11 & 12 (Detail carving re-enactments).

4. Finishing (Certainty - Care)

Before the item is complete, it goes through a multi-stage polishing sub-process to ensure a smooth surface:

1. Brushing: Removing debris.
2. Sandpapering: Smoothing the grain.
3. Clothing: Buffing the surface.
4. Saw dusting: A traditional technique using fine wood dust for a final polish.

The final stage is about the surface texture. This is a multi-sensory part of the craft, using friction and progressively finer abrasives.

Actions: A 4-step sequence: Brushing → Sand-papering → Clothing → Saw dusting.

- Documentation Nodes: wt_v_p_15 through wt_v_p_21 (Smoothing, Carving, Polishing).
- Digital Twin: The simulations document surface friction. They mimic the heat and pressure applied during the "saw dusting" technique.

Primary Assets:

- Real Capture: Videos 15-21 (The Polishing Sequence).
- Note: Many simulations in this phase document the "effort" of achieving the high-gloss finish.

7.4.2 Reenactment

We re-enacted the woodturning workflow in the simulator, producing a step-aligned sequence of interactive states that mirrors the recorded episode.

To validate the lathe module as a *process re-enactment* (not merely a shape generator), we replay the woodturning workflow as a sequence of interactive states. At each step, the user applies a tool proxy to the rotating workpiece; the system updates the underlying silhouette/template accordingly and regenerates the corresponding solid of revolution for immediate visual feedback. Figure 37 shows representative frames from this reenactment, spanning the progression from rough shaping through sizing and detailing to surface finishing, and illustrates how the simulator can be aligned to the observed sequence in the ethnographic capture.

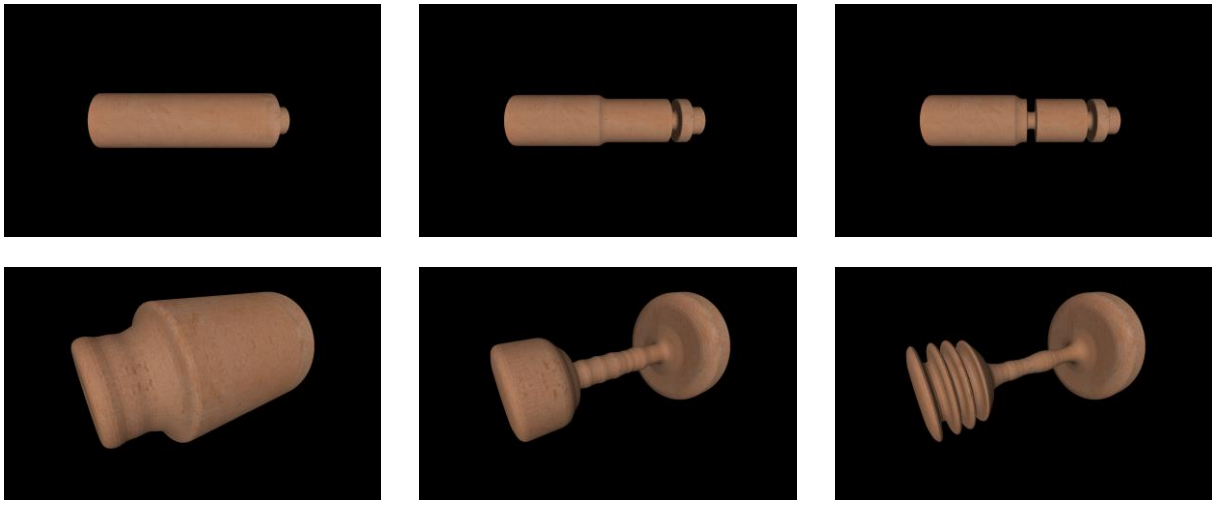


Figure 53. Representative frames from the interactive woodturning re-enactment. A 2×3 set of screenshots from the lathe simulator, covering key phases of the honey-picker workflow (rough shaping, sizing, detailing, and finishing). The workpiece is modelled as a solid of revolution generated from an editable silhouette/template; user tool interaction updates the template and the resulting 3D geometry in real time, enabling step-aligned comparison with the recorded process.

Data Parsing

1. Using the hierarchical schema of the workflow, we parsed the simulation actions into sub-process IDs (e.g., wt_v_p_x), action nodes (wt_v_a_x), and conditional logic (Decision Transitions).
2. A SPARQL result-derived XML was exported from the knowledge base containing metadata for 34 video entities. This included unique resource identifiers (URIs), semantic labels, and Zenodo-hosted locations.

Semantic Mapping

We align each workflow node wt_v_p_X with its corresponding recording Woodturning_video_X by matching the shared index X. Simulated video is treated as a digital re-enactment and is linked to the same node, allowing side-by-side comparison of real and simulated performance.

Synthesis

The transformation from raw data to instructions followed a four-stage synthesis:

1. The flowchart's directed graph was used to establish a linear timeline for the reproduction guide.

2. Abstract action IDs were replaced with descriptive craft terminology (e.g., "wt_v_p_8" became "Phase 1: Bark Removal").
3. Discrete actions were grouped into functional "Craft Phases" (Preparation, Geometry, Detailing, Finishing) to improve cognitive load management for the end-user.
4. Each instruction was annotated with its corresponding recording (real videos) and digital reenactment (simulation video), providing a 360-degree view of the action.

The resulting instructions represent a fusion of ontological engineering and ethnography. This documentation is a gallery of files and a knowledge base that can support reproduction and analysis. Given this analysis, we can now write the process in pseudocode (see Table 16).

Table 16. Woodturning pseudocode.

```
→ Preparing
      → Roughing out → Bark removal → Left-end smoothing → Thinning
→ IF too long THEN
      → Shortening
            → One ridge making (1st) → One ridge making (2nd)
→ IF too thick THEN
      → Reducing width
            → Thinning → Smoothing → Carving → Thinning
→ Making
      → Making the picker head
      → Making the picker handle
            → Thinning (pass 1) → Thinning (pass 2)
      → Refining the picker head
→ Finishing
      → Brushing → Sand-papering → Clothing → Sawdusting
```

The process is formally represented in Annex D and in our online knowledge base at: http://mop.mingei-project.eu/resource/?uri=http://www.craeft.eu/resource/a574985a-d44b-434f-893a-722bdf195018.

We now validate the reenactment at the level that matters for realism and traceability: does each simulated action segment correspond to an observed action segment in the ethnographic recording? Figure 54 provides representative pairings between *real* and *simulated* video excerpts. The pairings are indexed consistently (real video *X* ↔ simulated video *X*), so the viewer can compare motion intent, tool approach, and the resulting evolution of the axisymmetric profile across corresponding steps. The complete set of correspondences and file-level links is provided in Annex E, while the compiled video sequence is available via the online demonstration link.

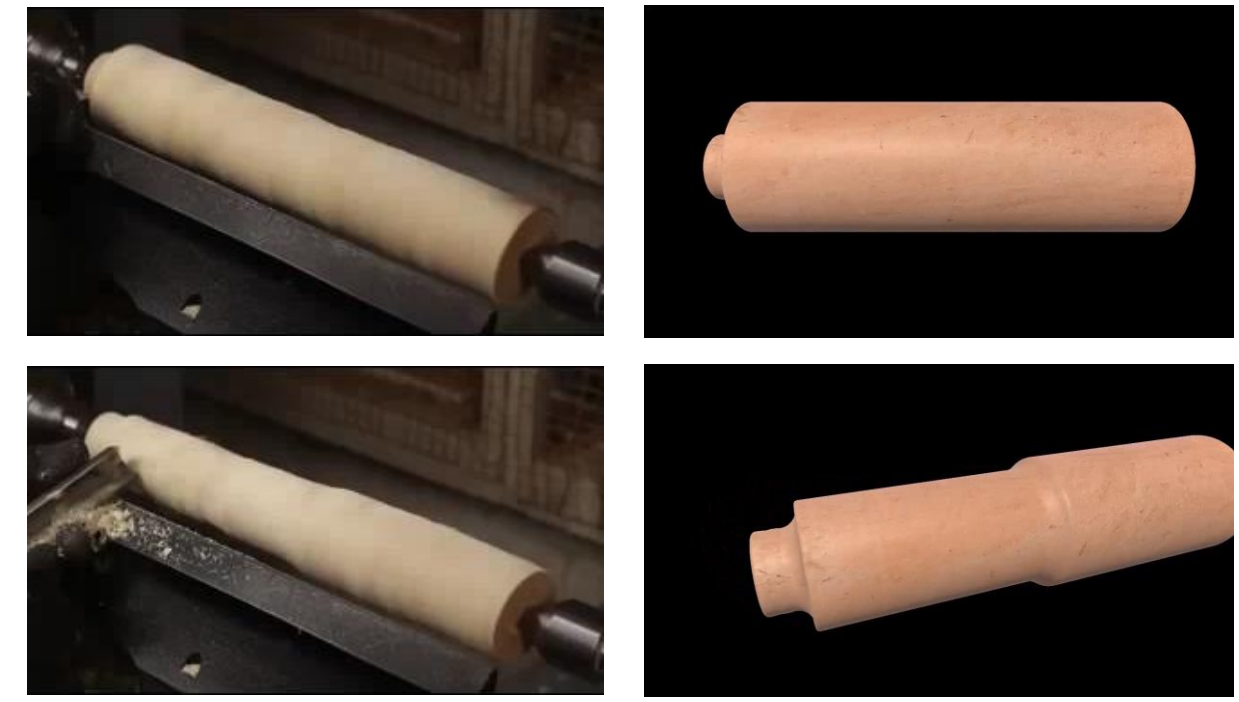


Figure 54. Step-aligned comparison of real and simulated action segments. Selected pairs of excerpts from the ethnographically recorded woodturning process (left column) and the corresponding digital reenactment produced with the lathe simulator (right column), aligned by shared step index (real video X ↔ simulated video X). The comparison illustrates how the simulator reproduces the process sequence and its characteristic action segments while maintaining traceability to the recorded evidence. Video: https://youtu.be/U99i7hgC0g0

7.4.3 Discussion

The woodturning case demonstrates that the toolbox is not limited to freeform sketching of rotational solids: it can be coupled to a formally structured workflow and to ethnographic media, producing a simulator whose steps are both interactive and traceable. Concretely, the workflow schema provides a logic layer (sub-process IDs, action nodes, and decision transitions), while the knowledge base provides an asset layer (video metadata and URIs); these are cross-referenced automatically through ID-asset correspondence, and then synthesised into an executable, user-facing reproduction guide and compact pseudocode. In this sense, the lathe evaluation serves as an end-to-end proof that semantic modelling and simulation can be fused into a reusable craft-specific simulator pattern, where process structure, evidence, and digital re-enactment cohere.

7.5 XR game for pottery

This subsection demonstrates that the solids-by-revolution module and its API packaging (Section 6.2.6) can be deployed as the geometric backbone of a craft-specific XR simulator implemented in Unity and delivered to a standalone headset. The case study is a VR-first pottery training game (developed in the associated Bachelor thesis), designed for a modern XR headset and built around an interaction loop familiar from wheel throwing: the workpiece rotates, the learner applies shaping tools, then explicitly transitions to decoration and exports the result.

The purpose here is to integrate evidence. The more comprehensive presentation of game features, user scenarios, and pedagogical framing is provided in D4.3. In D3.1, we focus on showing that the module can be (i) packaged as an API, (ii) bound into Unity as a reliable runtime component, and (iii) exercised through XR interaction patterns that remain stable, reversible, and exportable.

7.5.1 Architecture and API integration

The XR application is intentionally built around a **single authoritative clay mesh**. Rendering, tool interaction, painting/decoration, and export all operate on this same mesh instance. This architecture is the practical expression of the API boundary introduced in Section 6.2.6: the Unity/XR layer orchestrates interaction and presentation, while the geometry core remains the **single source of truth** for the current vessel state and the mesh updates that derive from user actions.

In the reference implementation, the mesh is produced by a native **RevolutionSolid** geometry generator and accessed from Unity via managed bindings on both desktop (Editor) and headset (Android). This is precisely the kind of packaging that allows the same geometric engine to be reused across multiple craft simulators: Unity provides the XR runtime and interaction vocabulary; the API provides deterministic geometry construction and update.

Figure 55 provides a compact “at a glance” view of the Unity/XR deployment. The key integration points are visible: the wrist menu exposes session-level controls, the clay mesh is the central live artefact, the wheel/base governs rotation state (which gates sculpting), and the tools constitute the XR-facing façade over the underlying geometry operations.

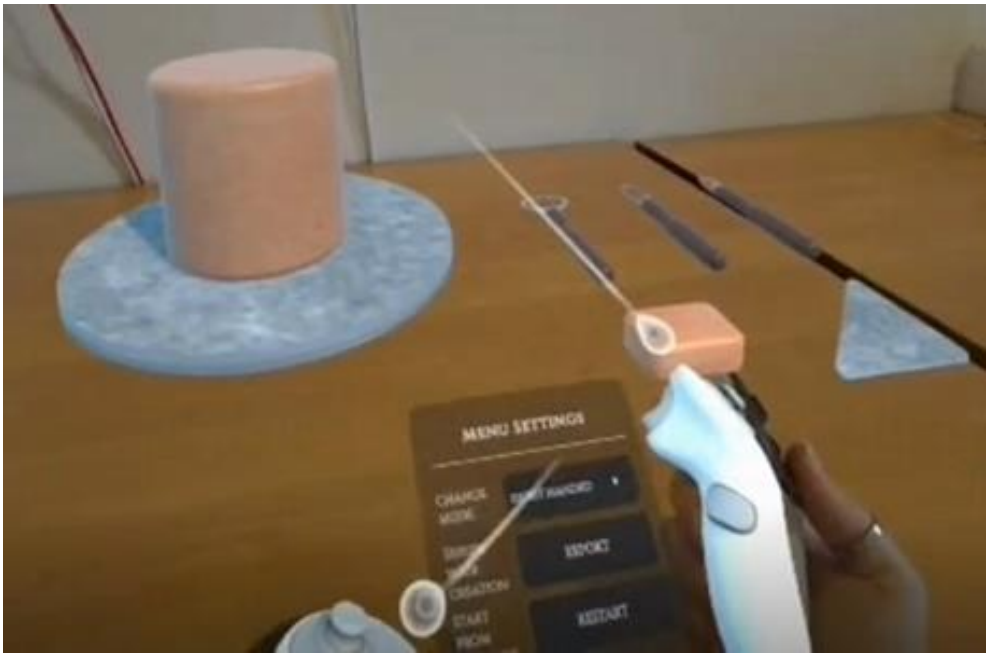


Figure 55. Unity/XR integration of the solids-by-revolution pottery module (workspace overview). In-headset workspace showing (1) wrist system menu, (2) the authoritative clay mesh, (3) the base/wheel controller that governs rotation state, and (4) pottery tools for shaping and decoration. The Unity layer drives interaction and feedback, while the underlying generator/API maintains the vessel geometry.

7.5.2 Sculpting loop: wheel-gated shaping and tool operations

The pottery simulator is wheel-centric and rotation-gated: shaping actions are enabled only while the wheel/base is spinning. This encodes an important studio constraint ("spin first, then shape") directly into the interaction logic and prevents a common novice error: attempting to deform the form without rotational reference. The base controller supports smooth acceleration and deceleration for comfort and precision, and the clay inherits the base transform each frame so that the visible rotation and geometry updates remain synchronised.

A compact set of tools provides the core manipulations required for practice-oriented vessel formation:

- an Additive tool ("block of clay") for build-up,
- Ribbon tools for subtractive shaping and refinement,
- a Mass-preserving tool for moving material without changing total mass, and
- a Rib tool for smoothing and controlled profile refinement.

To improve usability in XR, brief floating labels reduce tool-selection errors, and the application includes clear recovery pathways (e.g., reset after over-reduction).

7.5.3 Explicit mode boundary: shaping versus decoration

A recurrent requirement in craft simulators is to prevent accidental cross-contamination between phases that have different intents and constraints. The XR pottery game implements this as an explicit mode boundary between:

- **Sculpting mode** (geometry-changing operations permitted), and
- **Painting mode** (geometry locked; decoration operations permitted).

Painting is entered only after the user picks up the brush and confirms a prompt. On entry, the clay becomes slightly lighter to signal a "dry" stage appropriate for finishing, and sculpting tools are disabled. This is a direct example of an API-level separation: the same mesh remains authoritative, but the set of admissible operations is restricted to ensure predictable outcomes.

Figure 56 shows the explicit transition that enforces the separation between forming and finishing. The confirmation prompt functions as a safety gate: it reduces mode errors and ensures that once decoration begins, geometry edits are temporarily locked, preserving the integrity of the shaped form.

Figure 56. Explicit Sculpting → Painting transition (mode boundary enforced in XR). A confirmation pop-up gates entry to Painting mode; upon acceptance, sculpting tools are disabled, and the application switches to decoration operations on the same authoritative mesh, preventing accidental geometry edits during finishing.

7.5.4 Decoration

It is important to distinguish geometry generation from decoration in this XR prototype. The solids-by-revolution module (Section 6.2) is responsible only for the shape of the vessel: it maintains the silhouette/template representation and produces the triangulated surface of revolution that defines the clay mesh. Decoration does not modify this representation and does not stem from the shape-from-revolution pipeline. Instead, decoration is implemented as an *orthogonal* rendering attribute layer applied on top of the already-shaped mesh. In other words, the vessel geometry is *frozen* during decoration, and only per-vertex appearance attributes change.

Technically, the implementation uses vertex colours as the minimal, game-ready approach that remains lightweight on a standalone XR device and avoids UV unwrapping. Each vertex of the clay mesh carries a colour attribute ($c_i \in [0,1]^3$) with an alpha channel, stored in Unity's mesh colours array. The painting brush is modelled as a moving 3D influence region, as a sphere around the brush tip. At runtime, when the brush is active and a pigment is selected, the system performs a local update:

1. **Contact/proximity test.** Determine which vertices fall within a brush radius (R) of the brush tip position (b) in world space: $|v_i - b| \leq R$, where (v_i) is the vertex position (transformed to world space).
2. **Falloff weighting.** Compute a weight (w_i) based on distance for a soft brush edge, e.g. w_i = smoothstep(R, 0, $|v_i - b|$) (or a linear/quadratic falloff).
3. **Colour blending.** Update each affected vertex colour by blending the current colour with the active pigment colour c*: $c_i \leftarrow (1-\alpha w_i)c_i + (\alpha w_i)c^*$, where ($\alpha$) is an application strength parameter (paint "opacity").
4. **Erasing (water jar).** Erasing is implemented as the same operator but with (c*) set to the base clay colour (or by blending towards neutral), again with falloff. Clearing the brush simply resets the active pigment state so that painting has no effect until a colour is selected.

The rendering side uses a shader that combines the base clay material with the vertex-colour attribute. This can be as simple as: finalColour = lerp(clayBase, vertexColour, β), where (β) is a user-tuned mixing factor; alternatively, vertex colour can modulate albedo directly. The key property is that no geometric recomputation is required during decoration: the mesh topology and vertex positions remain constant, and only the per-vertex colour buffer is updated, which is efficient and predictable for XR.

From a systems viewpoint, this creates a clean separation of concerns:

- **Shape layer (API):** silhouette/template editing → surface of revolution mesh generation.
- **Appearance layer (Unity/XR):** vertex-colour painting and erasing as a process over the final mesh.

This separation is intentional: it demonstrates how the API-backed geometric engine can be integrated into Unity while allowing additional craft-relevant layers (such as surface decoration) to be implemented independently, without entangling them with the rotational-solid construction.

Figure 57 illustrates the decoration layer applied after shaping: geometry is locked, and only vertex colours are updated and rendered. Painting updates per-vertex colour attributes on the already-generated vessel mesh; the solids-by-revolution geometry remains unchanged during finishing. A water jar clears the brush and erases colour to support rapid correction during practice.

Figure 57. Decoration as an independent appearance layer. Painting updates per-vertex colour attributes on the already-generated vessel mesh; the solids-by-revolution geometry remains unchanged during finishing. A water jar clears the brush and erases colour to support rapid correction during practice.

6.5.5 Export and reuse of outcomes

The application exposes export via an in-headset wrist menu to keep the session self-contained. In Sculpting mode, export produces an OBJ geometry snapshot. In Painting mode, export produces OBJ and PLY, preserving both geometry and vertex-colour decoration. Files are timestamped on the headset to support critique workflows, portfolio capture, and downstream fabrication pipelines (e.g., printing or mould-related workflows), consistent with the deliverables' broader emphasis on traceability and reuse.

6.5.6 Summary: what this validates for Craeft

As an integration demonstrator, the XR pottery game provides evidence that the Section 6 module can be packaged as an API and embedded in a modern XR stack with:

- a stable single-source-of-truth mesh,
- continuous updates driven by wheel-gated interaction,
- explicit, reliable mode boundaries (forming vs finishing),
- reversible decoration suitable for training, and
- exportable outcomes for external review and reuse.

The pattern of Unity/XR front-end, combined with craft-specific interaction rules and a shared geometric core via API, generalises beyond pottery and supports the broader Craeft goal of creating multiple craft-specific simulators from reusable technical components.

7.6 Conclusion

Section 6 introduces and validates a craft-oriented modelling capability centred on solids by revolution, designed to bridge intuitive interaction with a mathematically well-defined construction. The section first establishes the core representation and interaction mapping: users act in 3D, but their actions are consistently mapped onto a 2D template, which is then revolved about a specified axis to generate a watertight workpiece. This design choice yields two practical benefits that matter for craft simulation: it preserves the structural constraints of rotational manufacturing (lathe turning, wheel throwing, blowpipe-centred forming), and it enables stable real-time updates because the modelling primitive is compact and interpretable.

A central contribution of the section is the API packaging of this module. By exposing silhouette editing, profile validation, mesh generation, and export through a clean interface, the same geometric core becomes reusable across multiple applications without entangling interaction logic with geometry generation. The validation results demonstrate this reuse in two complementary directions. First, within the Design Studio, the API supports interactive authoring of axisymmetric forms in a way that is intentionally constrained by real processes: designs remain compatible with the manufacturing assumptions and affordances of rotational crafts, rather than being unconstrained freeform modelling. Second, in Unity/XR, the same API is reused to construct an XR pottery training game, where the geometry engine acts as the single source of truth for the evolving vessel while the XR layer provides embodied interaction, feedback, and session control.

Importantly, Section 6 also clarified a separation that is critical for scalable craft simulators: shape generation and surface decoration are orthogonal layers. The solids-by-revolution module governs geometry; decoration in the XR prototype is implemented independently as a lightweight appearance layer (vertex colours) applied to the already-generated mesh while geometry is locked. This separation keeps the geometric kernel general and reusable, while allowing craft-specific finishing interactions to be added without altering the underlying construction logic.

Taken together, Section 6 shows a coherent pattern for craft-specific simulators: a process-constrained geometric core (solids by revolution), a reusable API boundary that supports multiple front-ends, and validation across both (i) ethnographically grounded, process-faithful reenactment (lathes) and (ii) modern interactive deployments (Design Studio and XR). This pattern prepares the ground for further craft modules that require tight coupling between realistic process constraints, interactive authoring, and deployable training experiences.

To close the section, we return to the Design Studio context to emphasise the broader role of the module: it is not merely a geometric generator, but a design instrument constrained by real process structure. The Design Studio uses the same API to let users explore forms interactively while remaining within the representational envelope of rotational manufacture. The figure below illustrates this authoring workflow: the user's edits act on the silhouette/template representation, and the corresponding solid of revolution is regenerated in real time for inspection and iteration.

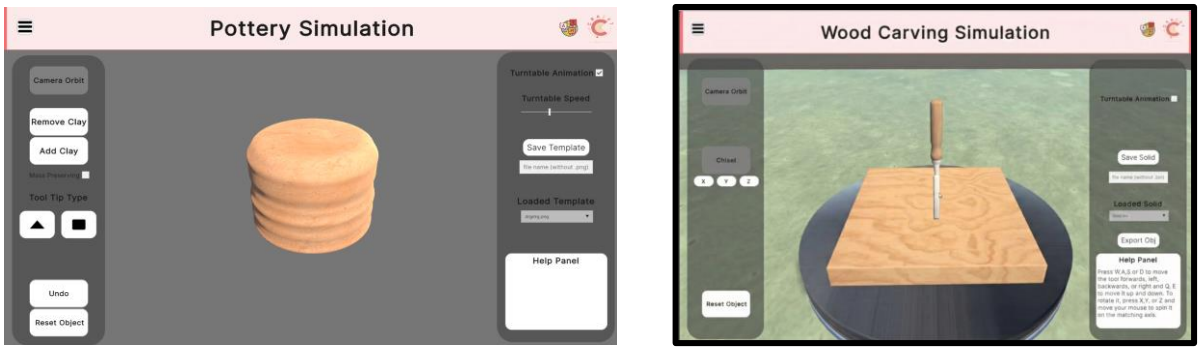


Figure 58. API reuse in the Design Studio for process-constrained authoring. Screenshot of the Design Studio interface using the solids-by-revolution API to edit an axisymmetric body and regenerate the corresponding 3D workpiece in real time. The workflow supports rapid design iteration while keeping outputs aligned with the constraints and affordances of rotational craft processes.

8 Moulds

Mould-making is a fundamental process for producing repeated objects in craft, design, and manufacturing. Mould design requires CAD expertise and labour-intensive preparation, making it inaccessible for many practitioners. Artists, designers, and small workshops often face technical barriers when attempting to create moulds for experimentation or production.

Moulds can be expendable or permanent. Their primary difference lies in their lifespan and the method by which the final object is retrieved. An expendable mould is designed for a single use; it must be broken away to release the casting inside. This type is made of granular materials such as sand or plaster, allowing for complex geometries without the need for a separation seam. A permanent mould is rigid and intended to be reused. Because it cannot be broken to retrieve the part, it must be designed as a split assembly that opens along a parting line, allowing the cast object to be ejected intact before the mould is closed again for the next pour. For reusability and material conservation, we focus on permanent moulds.

A software system addresses that bottleneck by introducing a pipeline that converts arbitrary 3D models into print-ready negative moulds. This system reduces the need for manual CAD intervention, generating mould pieces that are structurally sound and ready for fabrication. Features such as adaptive wall thickness, alignment keys, draft angles, pouring cups, sprues, and vents are generated automatically. The method supports bipartite (two-part) and quadripartite (four-part) moulds, reflecting established industry standards.

The **Automated 3D Mould Generator** is distributed as an open-source software package. The download location is the project's public GitHub repository, which contains the source code and examples for reproducing the results reported in this section. User instructions are provided in the project's README manual, as well as in Annex M. **Link:** https://github.com/Lion4re/automated_3d_mold_generator.

By lowering the technical threshold, the system enables rapid iteration and personalised production. It allows a wide variety of materials and supports sustainability through material-efficient design principles. The result is an accessible tool that enhances experimentation, facilitates teaching, and strengthens links between digital design and physical making.

8.1 Approach

In principle, the problem of mould design for arbitrary solids is highly complex. A general solution would require adaptive placement of cutting surfaces and specialised handling of undercuts, often involving multi-part mechanisms beyond simple planar separation. Such generality is difficult to automate and unnecessary in many practical contexts. To achieve a tractable solution, we restrict attention to solids that can be produced either by a bipartite mould, separated by a single vertical plane, or by a quadripartite mould, divided by two perpendicular vertical planes, as illustrated below.

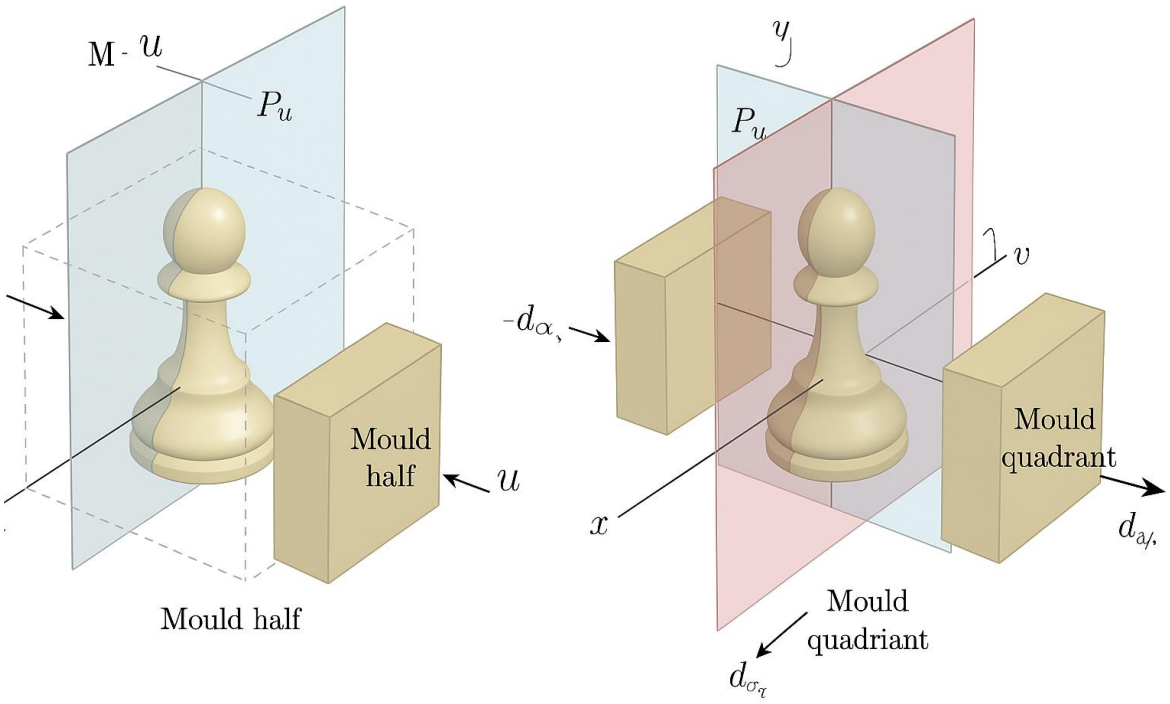


Figure 59. The two mould partitioning strategies that were implemented.

This restriction was chosen to maximise utility. The selected scenario represents a common geometric case that combines simplicity and broad relevance. The initial vertical cut divides the object into left and right sections, allowing each half to slide laterally away from the other. That usually works well, because most objects 'stand up' vertically and their details are stacked along height, not locked sideways. If the second cut is also vertical, the object is divided into four vertical slices, like cutting a cake into quarters. The design allows each quarter to slide freely outward, sideways, or horizontally. This flexibility accommodates a wide variety of shapes, as no element restricts their lateral movement. But if instead the second cut is horizontal, then the mould pieces would have to slide upward or downward to come free. The problem is that most objects that 'stand up', such as bottles or figurines, have changes in diameter or protrusions along the vertical axis, e.g. shoulders, rims, or bulges. When pulling a mould piece straight up or down, those features catch it and lock it in place.

In other words, vertical features are common, and they block vertical removal. By contrast, sideways removal (from vertical cuts) avoids these locks because objects rarely have features that 'hook' sideways across the entire height. For instance, let a chess pawn. If we cut it into left/right halves with a vertical plane, each half can slide out sideways: the curved outline never blocks it. But if we cut it into top/bottom halves with a horizontal plane, the top half is stuck on the pawn's bulging 'shoulders.' You can't slide it straight up without colliding with something.

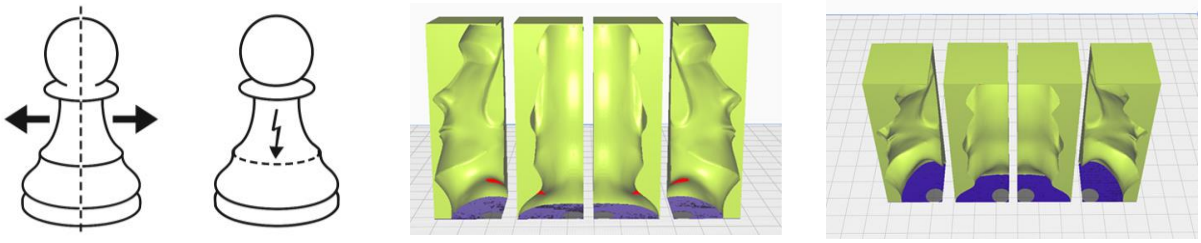


Figure 60. Left: Illustration of a chess pawn and two hypothetical cuts along the vertical and horizontal directions, plotted as dashed lines. For the horizontal cut, the pawn's concavities would lock the pieces in place. Middle, right: 3D models of mould pieces for a quadpartite mould cut vertically twice, in perpendicular directions.

By covering bipartite and quadripartite cases, the method handles both simple convex shapes and a wider class of more intricate cross-sections, without introducing unmanageable complexity. In this sense, the chosen scenario is 'broadly useful' because it captures a prevalent geometric case encountered across both industrial and craft moulding, while remaining tractable for automation.

The goal is a 3D-printable representation of moulds that will be used to manufacture a given 3D design. The computational method is conceptualised and implemented to be provided in the service of the Apprentice and Design Studios. In this section, we describe the technical modules that implement the moulds for the 3D design.

- Given the real-world implementation of abstract digital designs into specific moulds, we explore associated implications that lead us to practical additions to the mould implementation.
- Moreover, we want permanent moulds (green).
- We need to be simple, not for geometry theorists.

The computation of the mould is conceptualised and implemented as a sequence of Constructive Solid Geometry (CSG)[3] operations, converting 3D models into print-ready negative moulds in bipartite or quadripartite form. In 3D computer graphics and CAD, CSG is often used in procedural modelling. CSG can also be performed on polygonal meshes, and may or may not be procedural and/or parametric. In our case, we use a Cartesian voxel space to parallelise the computation on the GPU.

The program calculates a box that fits around the object. The middle of this box becomes the reference point for how the mould is cut. The program splits an object into pieces so that a mould can be built around it and later opened without breaking. There are two main ways the program can split the mould. Bipartite mould: The object is divided into two halves, left and right, with a single vertical cut down the middle. The mould opens sideways, like a book. Quadripartite mould: The object is split into four quarters using two vertical cuts at right angles. To protect the user from unfeasible manufacturing, a computational safeguard is provided. The mould opens like four doors swinging outward. The key condition is that each piece of the mould must be able to slide straight off the object without getting stuck.

[3] Constructive solid geometry uses solid modelling to create complex objects using Boolean operators.

8.2 Mould Representation

We first conceptualise the 'mould block' as if it were a cast. Then we slice it into pieces to create permanent moulds.

A uniform, orthotropic voxel grid is employed to represent volumetric occupancy and facilitate logical volumetric operations. The principal object dimension determines the resolution of this grid, although users can refine it. The process involves:

1. Conversion of the input model into a padded voxel grid.
2. Introduction of a hole for pouring the material.
3. Computation of the mould cavity using logical operations on the voxel grid.
4. Computation of mould pieces.

A bounding box is computed around the object and expanded by an offset thickness to form a mould block. The cavity is created by subtracting the object mesh from this block. This negative space represents the casting volume. In the example below, a voxel grid is visualised by displaying slices along the Z-axis.

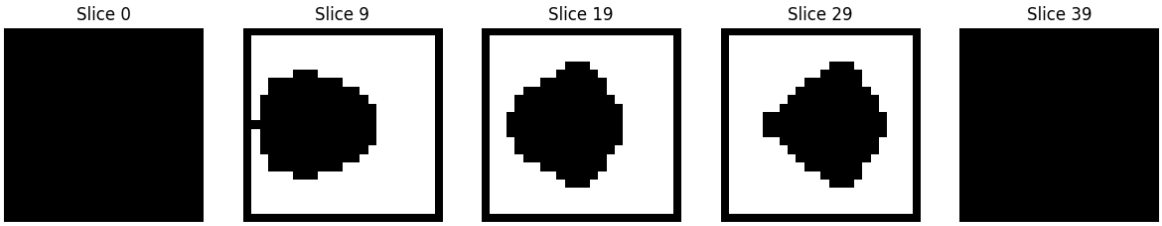


Figure 61. Voxel grid sections across zz'. Black and white pixels denote empty and occupied voxels, respectively.

We illustrate the outcomes of our mould-generation process using a sample model.

Figure 62. Left: Original model. Middle: Visualisation of the corresponding mould as transparent. Right: Predicted cast. This middle visualisation is animated in https://youtu.be/MFrC8m9ddCU to demonstrate the quality of the mesh created using the marching cubes algorithm.

8.3. Moulded, cast, and sculpted objects

Texture-based renderings of matte objects are supported by conventional, off-the-shelf rendering engines and exhibit no challenge. The greatest challenge is the treatment of dielectric materials, such as metals, porcelain, and glass, which cannot be rendered using conventional texture-based methods. The result is the visualisation of the final appearance of custom moulded, cast, and sculpted pieces, from vases and sculptures to jewellery and decorative items, across a variety of materials. The benefit is the feedback during the design process. Visual updates enable quick iterations and refinements of the design, reducing the time and cost associated with physical prototypes. This is particularly important in sculpted, cast, and moulded products, as the mistakes cannot be remedied.

The examples below demonstrate this for two 3D models. Figure 63 shows a 3D model rendered as it is made from glossy plastic.

Figure 63. A 3D model of a sculpture rendered as made from glossy plastic and shown from three viewpoints.

8.4 Implementation

8.4.1 Overview

This process generates a mould based on a digital 3D model. By convention, the system assumes that the input 3D object is provided in an upright orientation.

1. Input and Preprocessing. The system imports object geometry from standard file formats, such as STL, OBJ, PLY, and GLB. Because many meshes are imperfect, a repair pipeline ensures they are watertight.
2. Bounding Box Computation and Padding. The process calculates the model's bounding box using the extreme coordinates along each axis. This volume is expanded (padded) to ensure there is sufficient material thickness for the subsequent morphological operations.
3. Voxel Grid Generation. The volume is discretised into an isotropically spaced voxel grid. The grid represents occupancy and is initialised as 'full' with a value of one (1); a value of zero (0) indicates empty space. The user specifies the resolution to balance geometric accuracy against processing time. The boundaries of this grid are set to zero to define the outer limits of the mould.

4. Internal Structure Detection. To create the mould cavity, the system uses a Constructive Solid Geometry (CSG) operation to subtract the 3D model from the solid voxel block. Additionally, a funnel-shaped opening is carved into the top padding to serve as the sprue (or pouring basin). The resulting voxel representation is converted into a 3D triangle mesh.
5. Parting. To create a permanent mould, the mesh is segmented into separate pieces using cutting planes. The mesh of each segment is capped (sealed) at the intersection region so that every piece remains watertight (manifold).
6. Finishing. Structural features, e.g. alignment keys, are added to each part to enhance its utility.
7. Visualisation and Export. The voxel grid is visualised to allow inspection of the internal and external structure. Finally, the mould segments are exported as separate meshes, ready for 3D printing.

8.4.2 Preprocessing

Reliable CSG operations require watertight meshes. The preprocessing applies conventional repair operations.

Repairing	Hole filling, normal correction, and removal of degenerate and duplicate faces.
Cleaning	Removal of non-manifold edges and duplicate vertices.
Validation	Watertightness is checked; if not satisfied, DT-based filling is applied.

4.4.3 Voxel and Mesh Processing

The cavity is represented computationally as the logical difference C=B\M, where B is the bounding box, and M is the repaired, watertight mesh of the object. In both cases, the separation ensures that mould parts assemble into a parallelepiped surrounding the solid.

Division into mould parts is achieved by applying orthogonal planes through the centroid of C. Each mould part is then computed as the intersection of the cavity with the corresponding half-space.

This method reduces the task to efficient clipping operations, guaranteeing watertight outputs. Watertightness means that every edge belongs to exactly two faces. Volumetric logical operations are reliable only on watertight meshes; otherwise, the subtraction is ambiguous and may fail to produce accurate results. Since many scanned models are not watertight, mesh-repair methods are applied to enforce this condition before further processing.

The surface of the voxel structure is converted into a mesh representation, using the Marching Cubes algorithm [174]. For a triangle mesh with n nodes, the dominant cost is the logical kernel $O(n \cdot \log n)$.

8.5 Parting

The parting method is based on planar partitioning and is accompanied by a feasibility analysis. This analysis admits solids by their ability to be enclosed within a two- or quadripartite parallelepipedal mould, so that each part can be removed by straight-line translation without intersecting the solid.

Parting is performed by hypothetical planes that intersect the 'negative' solid. Two cases are implemented, both utilising planar slicing to divide the block.

- Bipartite mould: One vertical symmetry plane through the geometric centre partitions the block into two halves.
- Quadripartite mould: Two perpendicular vertical planes through the geometric centre partition the block into four quadrants.

In both cases, each part is removable by a straight horizontal translation.

A feasibility analysis checks for undercuts by evaluating surface normals relative to extraction directions. If any normal points against the intended extraction, the solid cannot be demoulded under the straight-line assumption.

The feasibility test checks if the input solid is partitionable into mould segments such that each segment can be removed by a straight motion, without intersecting the solid. This protects the practitioner from creating a mould that is not recyclable and must be destroyed (or would destroy the product during demoulding).

8.5.1 Nomenclature

Let $S \subset R^3$ be a compact solid with C^1 boundary ∂S; a piecewise C^1 is permissible.

Let $c \in R^3$ be the geometrical centre of the bounding box of S.

Straight-line extraction along d means translating a mould piece rigidly by $t \cdot d$ with $t \rightarrow +\infty$, without intersecting S.

The outward unit normal to ∂S at $p \in \partial S$ is n(p).

The input solid $S \subset R_3$ is provided in upright orientation, with its dominant axis denoted by yy' aligned with gravity. The geometry is assumed to be draft-compatible, i.e. no re-entrant features or negative draft angles are present.

The axis-aligned bounding box of the solid is computed. The geometric centre c of the bounding box serves as the reference for subsequent partitioning operations.

8.5.2 Case A. Bipartite Mould

The object is enclosed within a bounding box and divided into two halves by a single vertical plane that passes through the centre of the box. The two mould halves form complementary blocks that meet along a plane. During demoulding, each half slides away in a straight line, in opposite horizontal directions (+u and -u). This configuration is suitable for a wide range of upright objects that can be split cleanly into two parts without undercuts. See Annex M.3, Theorem A, for a proof.

Intuitively, on the '+' side, all outward normals must have non-negative components along +u. On the complementary '−' side, they must have non-positive components along +u. Points whose normal points in the opposite direction correspond to undercuts and signal a violation of straight extraction.

8.5.3 Case B: Quadripartite Mould

The object is enclosed in the same way but divided by two perpendicular vertical planes through the centre of the bounding box. This creates four mould quadrants, each covering one quarter of the object. This configuration enables the production of more complex cross-sections than in Case A, while retaining demouldability. See Annex M.3, Theorem B, for a proof.

Intuitively, in each quadrant, the outward normal must not point against its outward bisector. This ensures there is no re-entrant feature that would 'hook' the mould as it translates out along direction $d_{\sigma,\tau}$. As in Case A, bilateral or dihedral symmetry is sufficient but not required. These criteria depend only on the direction of the surface normals relative to the extraction directions, not on global symmetries.

8.5.4 Geometric Feasibility Test

For each mould part, we evaluate the no-undercut condition. The outward normal n(p) at each boundary point $p \in \partial S$ must not point against the intended extraction direction d. Formally, $\langle n(p), d \rangle \geq 0$, with equality permitted only along parting lines or surfaces. Violation of this condition indicates that demoulding would trap the solid, rendering the partition infeasible. See Annex M.3, Theorem C, for a proof.

8.5.5 Insight

Convex solids are universally admissible under both bipartite and quadripartite separations. Non-convex solids may be admissible provided their concavities do not create undercuts relative to the extraction directions. For solids aligned with gravity, vertical-plane separations, cases A and B, succeed more often than horizontal separations. Introducing a horizontal cut perpendicular to gravity significantly reduces the admissible class of solids, since vertical features (e.g. shoulders, rims) tend to obstruct upward or downward extraction.

But why does a horizontal second cut reduce admissibility?

Let $v = e_y$ (vertical). A 'horizontal cut' uses $P_v = \{y = \text{const}\}$ and would assign extraction along $\pm v$. Most technical solids 'stand straight' (balance and rest when placed vertically) with significant vertical relief. On large portions of the surface, n(p) has both positive and negative e_y-components, e.g., near shoulders, rims, steps. Thus, the condition $\langle n(p), \pm e_y \rangle \geq 0$ fails on substantial regions, creating unavoidable undercuts

for straight vertical withdrawal. Hence, the admissible class shrinks relative to purely vertical (horizontal-direction) openings.

8.6 Finishing Features

Once geometrical reasons have provided the mould pieces, we consider the practicalities of using their 3D printed implementations. To ensure manufacturability, the system incorporates the following finishing features:

1. Draft angles are applied to vertical walls to reduce friction during demoulding.
2. Wall thickness balances structural integrity, printability, and material economy.
3. Alignment keys are placed at the sides of parting planes for reliable assembly.
4. In addition to the sprue, vents are added to permit air escape.

8.6.1 Wall Thickness

Wall thickness, t, is the distance between the mould's inner cavity and its outer surface. Wall thickness is critical for maintaining structural integrity, optimising material economy, and the placement of alignment keys. In the inset, the orange head shape is the cavity, while the surrounding grey block shows the piece's body. The space between the cavity and the outer block conveys the controlled wall thickness.

Wall thickness is calculated to strike a balance between manufacturability, strength, and efficiency. A wall that is too thin risks breakage during printing or demoulding, while a wall that is too thick wastes material. By determining an appropriate thickness, the system produces moulds that are structurally sound, economical to fabricate, and capable of accommodating alignment keys. A suitable value balances four requirements:

1. Printability: meeting the minimum wall limits of SLA/FDM technologies.
2. Structural integrity: providing strength at split planes.
3. Key accommodation: ensuring sufficient clearance for alignment keys.
4. Material economy: avoiding unnecessary bulk.

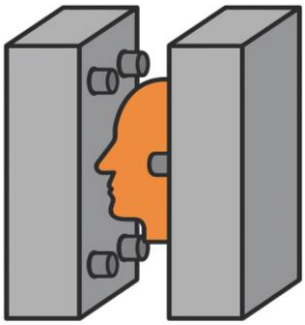
An empirical base factor β, tuned through extensive testing, is applied $t_0 = \beta \cdot d$. The base factor, β, is a piecewise base factor used to determine the initial wall thickness of the mould. The initial thickness, t_0, is the product of the base factor by the mean dimension, $t_0 = \beta \cdot d$. This value is later refined based on the number of mould pieces and their interfaces.

Modifiers adjust the initial thickness. The interface area factor, α_s, increases the thickness for large contact surfaces. The piece number factor, α_p, increases the thickness for four-part moulds. The refined value is $t_c = t_0 \cdot \alpha_s \cdot \alpha_p$.

A minimum and a maximum thickness value bound the final thickness. The minimum, t_0, ensures printability, key clearance, and a 2 mm floor thickness. The maximum $t_1$prevents wasteful material use and excessive bulk. The selected value t always satisfies $t_0 \leq t^* \leq t_1$.

8.6.2 Alignment Keys

Alignment keys are small interlocking features added to the contact surfaces of mould pieces to guarantee a precise fit during assembly. They stabilise mould assembly and guarantee accurate reproduction. Without them, even minor shifts between parts can cause leaks, offsets, or distortions in the final cast. This makes the mould easy to align, reduces errors during casting, and supports consistent reproduction of complex forms.



This step calculates the alignment key size and placement, so that they are durable for repeated use and do not interfere with the cavity.

Geometry

Keys are right cylinders, extruded normal to the parting plane. Each protrusion has a matching recess. Radius, r_k, and height, h_k, scale with wall thickness t: as $r_k = k \cdot t^*$, $h_k = \lambda \cdot t^*$. Clearance γ is applied to ensure fit during printing.

Placement

We use two methods to determine key placement. A primary and a fallback. Both methods guarantee that keys do not intersect the cavity.

The 'corner-first' strategy is the primary method. The diagonal corners of the parting plane are tested for clearance; if suitable, they are chosen. Let the split plane P_i in R^2. Four candidate centres are given by (u, v) = ($\pm(½ \cdot w - c)$, $\pm(½ \cdot h - c)$), where w and h are the mould-face width and height, respectively and $c = \eta \cdot t^*$ ($0.5 < \eta < 0.8$). A candidate is safe if dist((u, v), P_i) ≥ $r_k+\gamma$, and dist((u, v), corner)<ρ, where $\rho = 1.5 \cdot t^*$. If two diagonal safe corners are found, they are accepted. For quadripartite moulds, corners are chosen by descending clearance.

The fallback method is based on the Distance Transform [173], and is invoked if corners are unsuitable. Local maxima of the Euclidean Distance Transform provide alternative placements. When fewer than the required safe corners are found, the face is rasterised to an N×N binary image. Let I denote the occupancy matrix, where I(i, j) = 0 inside S and 1 elsewhere. The Euclidean Distance Transform DT(I) is an image where

each pixel is assigned a value corresponding to its distance from the object boundaries in the binary image. Table 17 summarises these features.

Table 17. Fallback strategy spatial features.

Local Maxima	Candidate local maxima are filtered to be $D \leq d_0$, where $d_0 = (r_k + \gamma) / p_x$, and p_x denotes pixel spacing. A 2D max-filter (30 × 30 pixels) extracts local peaks, sorted in descending order.
Separation	Select peaks iteratively, enforcing pairwise distances s_0, where $s_0 = \max(0.2 \cdot N, 1.2 \cdot s_{min})$, once for two pieces and twice for four pieces.
Back-projection	Map pixel coordinates (i, j) back to the reference frame via inverse linear interpolation, to find the placement centres on the interface.

Figure 64 shows the alignment keys and their sockets for a biparite (left) and a quadparite (right) mould.

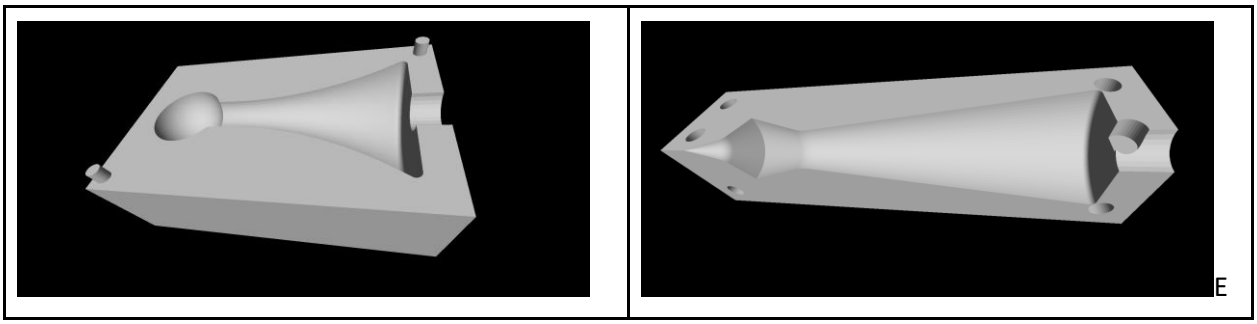
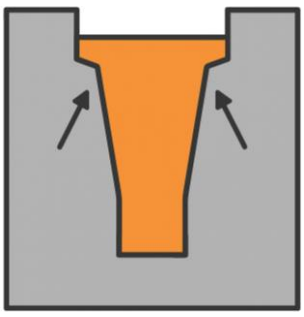


Figure 64. Examples of alignment keys are places at the corners of two pieces of a biparite (left) and a quadparite (right) mould.

8.6.3 Draft-Angles

Perfectly vertical walls create friction and suction, which can trap the cast and damage either the mould or the object. A draft angle θ is a slight taper applied to the vertical walls of a mould cavity to make demoulding easier. It prevents undercuts and reduces friction during demoulding. By introducing a controlled slant (typically between 0.5° and 3°), the pipeline reduces these forces, allowing the cast to be released smoothly.



Method

The draft is applied by volumetric scaling in the XY plane, $M_d = T^{-1} \times S(1 + \tan\theta) \times T(M)$, where T translates the centroid of M to the origin and S applies radial scaling. This scaling avoids self-intersections in concave regions and maintains the exact Z-height.

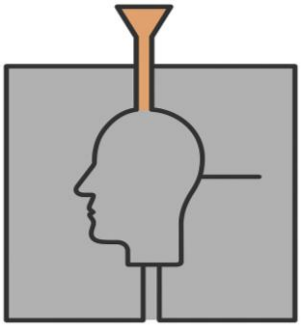
The drafted mesh, M_d, is obtained by the similarity transform $M_d = T^{-1} \times S \times T(M)$, where $T(p) = p\text{-}c$, which scales radially around the centroid in the XY plane. The cavity is refined, replacing M as $C_d = B \setminus M_d$. This is a similarity transform and, thus, M_d remains watertight.

Limitations

Deep recesses invisible from the parting direction cannot be drafted automatically and require redesign. The uniform scaling model cannot add draft to deeply recessed features whose surfaces are not visible from the +Z direction (the 'top' view). Such undercuts need manual redesign. Scaling enlarges all XY cross-sections, so the outer mould footprint grows. When the bounding-box size is critical, the user may disable draft. Runtime overhead is negligible.

8.6.4 Vents

The sprue is the main entry path for pouring material into the cavity. Vents are channels added to the mould to control how material flows in and air escapes during casting. Vents allow trapped air to exit, preventing bubbles and incomplete fills. If they are absent or poorly placed, casts suffer from voids or surface defects. The automated pipeline generates sprues and vents adaptively, scaling their size to the object and mould geometry, so that filling is smooth, air pockets are minimised, and the final cast reliably captures the intended form.



Vents are represented as cylindrical subtractions from the mould block. Radius r_s is proportional to the minimum mould dimension: $r_s = 0.1 \cdot \min(dx, dy)$. For handheld objects, penetration $z \cdot \Delta z$ scales with part height, clipped between $z_0 = 2$ mm and $z_1 = 25$ mm.

The pouring basin geometry is $\text{cyl}(r_s, h_s) = (x, y, z) \mid (x - m_x)^2 + (y - m_y)^2 \leq r_s^2,\ z_0 \leq z \leq z_0 + h_s$, where $h_s = z_c - \Delta z - z_0$. The cylinders are subtracted after the cavity, and alignment keys are defined, ensuring watertightness of the final geometry. After alignment keys are in place, subtract the cylinder from every mould piece: $\forall i,\ P_i \leftarrow P_i \setminus \text{cyl}(r_s, h_s)$. Because the cylinder axis lies entirely inside and its top terminates at least Δz_0 below the cavity midpoint, intersections are robust and cannot break watertightness.

8.6.5 Output

At this stage, the pipeline has fully specified the mould geometry: the cavity is partitioned into mould parts; keys and the parting interface are instantiated; and a sprue-vent system is generated with controllable sizing parameters. The implementation then converts these constructive definitions into explicit meshes and performs a final consistency pass (mesh repair where required, Boolean assembly checks, and watertightness checks of each mould component).

The output of this stage is (i) a set of watertight STL files for the mould pieces and auxiliary channels, and (ii) JSON logs that record the run configuration and any mesh-repair actions. These artefacts provide the

basis for validation: the STL files enable fabrication and casting tests, while the logs support reproducibility and debugging when an input geometry requires special handling.

8.7 Validation

The automated pipeline outputs (i) a set of watertight STL files representing the mould pieces (including parting, keys, sprue, and vents) and (ii) JSON logs that record mesh repair actions and the parameter values used. This dual output is important for validation: the STL files validate manufacturability, while the logs support reproducibility and troubleshooting when an input mesh requires repair or when a parting choice needs revision. The workflow can be executed either interactively or via the command line, completing in under a minute on standard hardware for typical inputs.

Validation is presented in three layers: (1) conceptual manufacturability, i.e., whether the chosen parting enables straight extraction without undercuts; (2) practical fabrication, i.e., whether the produced parts can be manufactured and used to obtain a cast; and (3) robustness by design choice, i.e., how increasing the number of mould parts can reduce demoulding risk in fragile cases.

8.7.1 Conceptual validation

Before fabricating a mould, the key question is whether the cavity can be demoulded by a straight pull along the intended extraction direction. For bipartite moulds, this is equivalent to an undercut check with respect to a sideways translation: surfaces that “face backwards” relative to the pull direction will mechanically lock the cast. Figure 65 illustrates both the geometric construction (parting plane through the mould block) and a normal-orientation-based check that confirms the absence of undercuts for the selected parting.

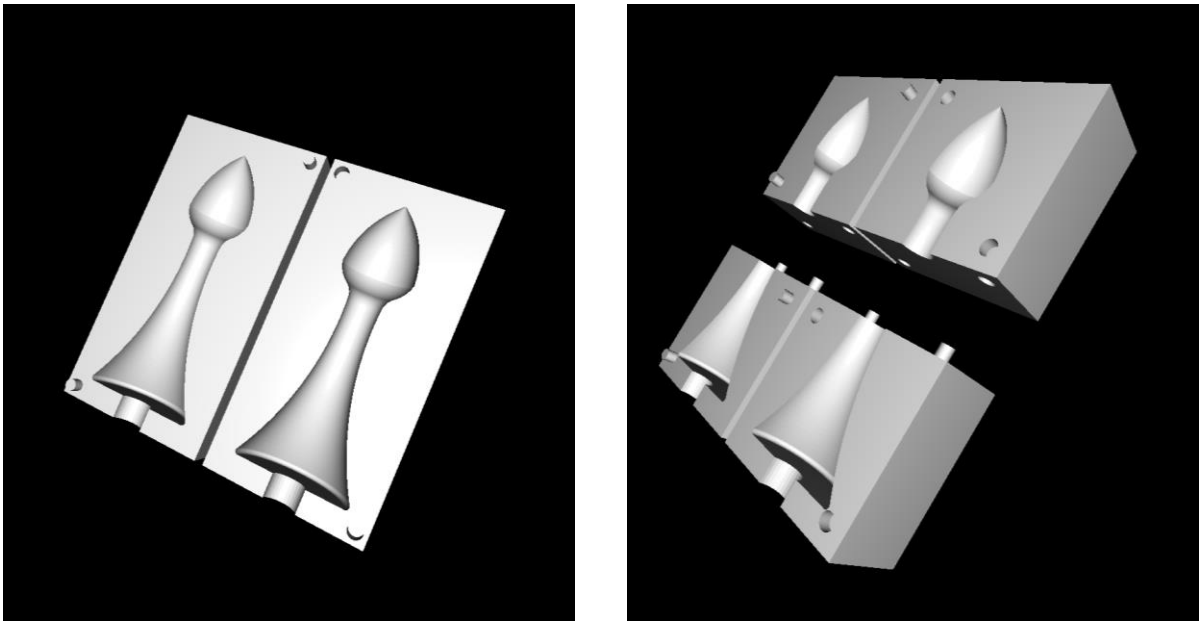


Figure 65. Conceptual demouldability check for a bipartite mould. Left: the cavity is enclosed in a bounding box and split into two complementary halves by a central vertical parting plane: each half is extracted by a straight sideways translation. Right:

an undercut check based on surface-normal orientation with respect to the pull direction confirms that the part can be demoulded without mechanical locking.

8.7.2 Practical validation

Conceptual demouldability is complemented by a practical test: the generated pieces must assemble correctly, remain watertight as mould blocks, and support a full casting/demoulding cycle. Figure 66 shows two representative outcomes, a bipartite and a quadripartite mould, together with their casts. The quadripartite case is included to demonstrate that the same pipeline generalises to multi-part moulds when a two-part split would be risky or when the user prefers to reduce demoulding forces by distributing them across smaller part interfaces.

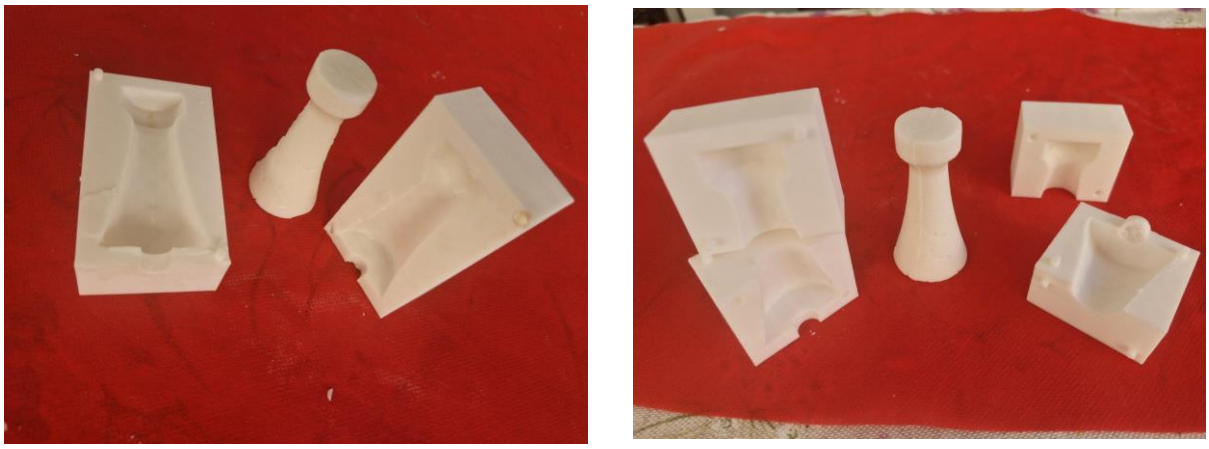


Figure 66. Practical validation with fabricated moulds and resultant casts. Left: bipartite mould and cast. Right: quadripartite mould and cast. The examples demonstrate that the pipeline produces assembleable mould pieces that can be used in an end-to-end casting workflow.

8.7.3 Robustness

Quadripartite moulds are not always necessary; many axis-aligned shapes can be demoulded reliably from two halves. However, increasing the number of parts can make demoulding **simpler and safer**, particularly for slender features or fragile cast materials, because it reduces the required extraction force and the risk of chipping at edges.

Figure 67 illustrates this trade-off on a rook: although a bipartite split is feasible, a four-part mould reduces the mechanical “prying” required to release the cast. This is a pragmatic robustness mechanism: it safeguards delicate pieces and supports user preference for lower-risk demoulding, at the cost of additional mould parts.

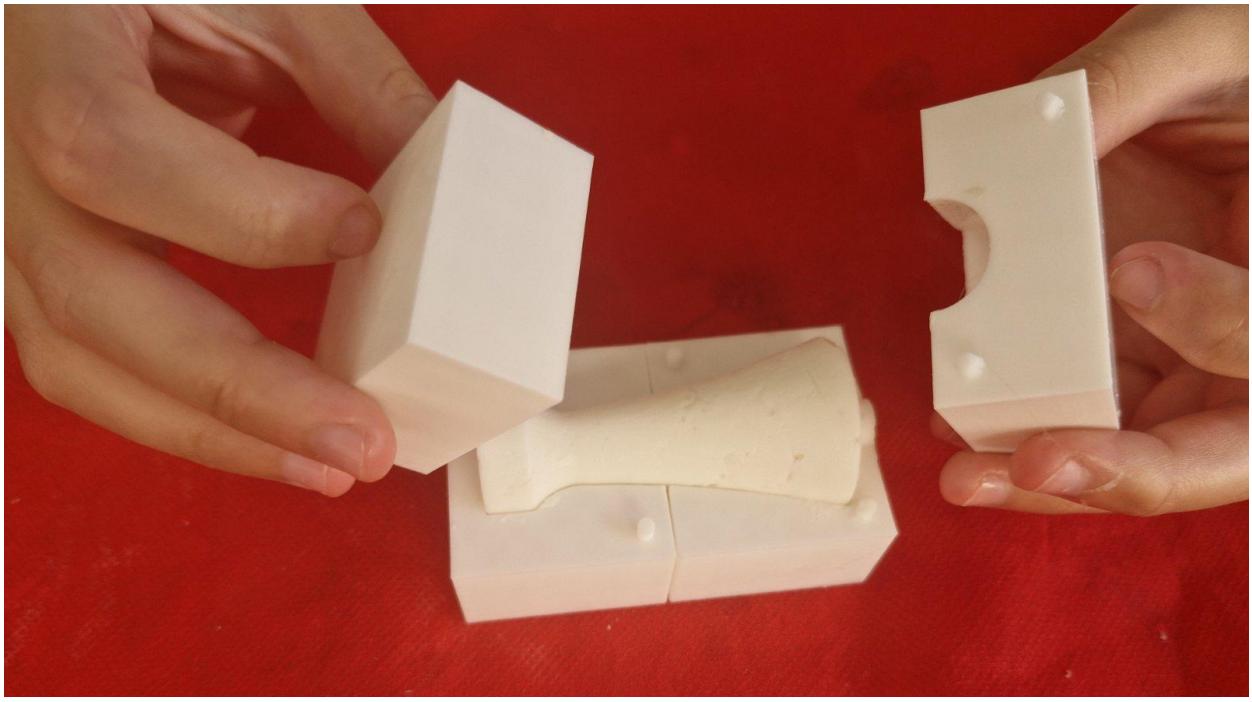


Figure 67. Quadripartite demoulding as a robustness choice. A rook can be demoulded from fewer parts, but a four-part split reduces extraction stress and makes release easier, which is advantageous for fragile materials and fine details.

8.8 Conclusion

We presented an end-to-end pipeline for automated mould generation from input geometry, to produce moulds that are manufacturable and usable in real craft workflows. The section formalises a sequence of constructive operations and then translates these into watertight meshes suitable for digital fabrication.

A key contribution is that the pipeline treats mould generation as a process-constrained design problem rather than a purely geometric one. Demouldability is addressed through parting choices and undercut reasoning, while assembly reliability is supported through parameterised keys and controlled clearances. The sprue and vent designs operationalise casting requirements as first-class geometric features, allowing the pipeline to produce moulds that can be used immediately in an end-to-end casting cycle.

Validation showed that the method supports practical fabrication and casting. The examples demonstrate bipartite and multi-part moulds, highlighting a robustness mechanism: increasing the number of parts can reduce demoulding stress and improve reliability for delicate forms. The pipeline records its operations and parameters so that results are reproducible and diagnosable, which is essential when the same method is reused across diverse craft geometries and materials.

Section 8 establishes automated mould generation as a reusable Craeft capability that connects digital design assets to fabrication. It complements the earlier sections on geometry construction and simulation by providing a concrete route from a digital workpiece to a manufacturable mould, thereby supporting rapid iteration, repeatability, and craft-aligned production constraints within the broader Craeft toolchain.

9 Composite objects

Sections 5–8 established a set of reusable capabilities for craft-oriented design: geometry generation, physically grounded appearance prediction, and manufacturable outputs. Section 9 brings these strands together in a pragmatic direction: it targets composite artefacts as objects made from multiple materials and multiple parts that are fabricated separately and then assembled.

The motivating observation is that many craft products are not single-material monoliths. They are layered, inlaid, framed, or modular by construction: colour regions become separate pieces; transparent elements interact with light; decorative sub-components must fit within an assembly logic; and practical constraints shape what designs are feasible. Accordingly, Section 9 treats composition as both a design representation problem and a manufacturing problem.

Technically, the section introduces lightweight utilities that start from conventional images, generate segmented regions and part definitions, and then convert them into planar or 3D components suitable for fabrication. These pipelines deliberately incorporate craft constraints early: morphological clean-up to avoid holes and overly intricate boundaries, and assembly-aware decomposition to reduce the number of tiny pieces. The resulting parts can then be previewed using the visualisation and light-transport machinery, including environment-based illumination and explicit light sources when the artefact is itself emissive (lamps).

The subsections develop this theme progressively: planar composites, 3D composites, lamps, and cane-work structures. Across these examples, the common deliverable contribution is a coherent pattern for turning a high-level design into an assembly-ready set of components with predictable appearance, bridging creative intent and practical realisation.

9.1 Planar objects

New types of products can be designed using the visualisation toolbox. We target objects composed of multiple materials that can be assembled. The individual pieces can be printed or handcrafted, given their 3D models. This utility creates planar and 3D composite objects that can be assembled from multiple parts, taking as input conventional images, such as photographs or drawings.

A software utility was developed that takes as input an image and performs colour quantisation according to several colours determined by the user. It then distinguishes the masks corresponding to each colour. However, the quantisation result will typically contain holes, which would make the physical implementation of the design impossible for some materials or, at least, extremely tedious for a practitioner to craft or assemble. An example[4] is given in Figure 68, showing the original image and the quantised regions for the facial and hair-coloured pixels. Due to the intricate details, the 3D printing of such components would result in numerous small pieces, making the assembly task difficult or impossible to achieve.

[4] The example is inspired by the “Shot Marilyns”, a series of silkscreen paintings produced in 1964 by Andy Warhol, each canvas measuring 40 inches square, and each one portraying actor Marilyn Monroe. See https://en.wikipedia.org/wiki/Shot_Marilyns for more information.

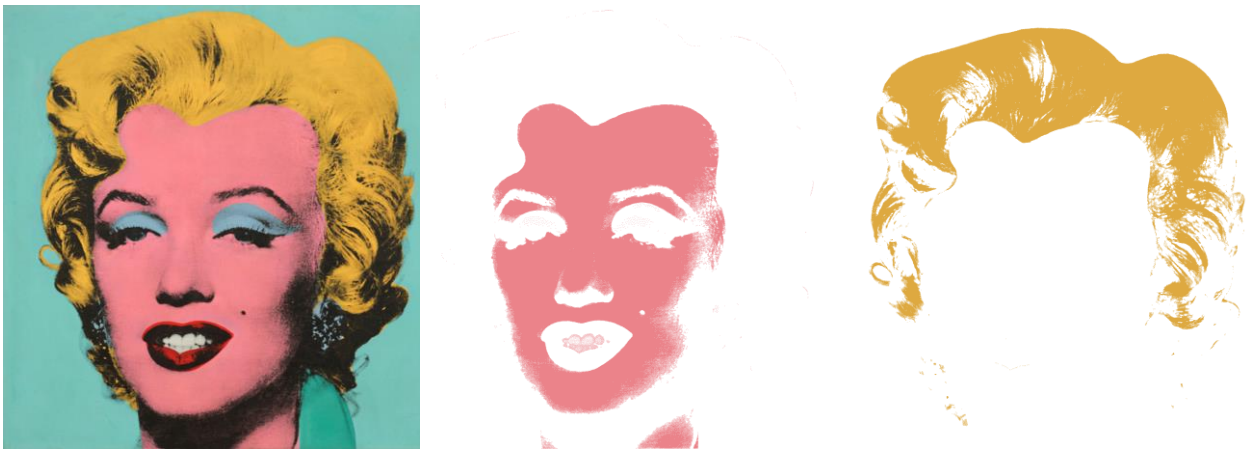


Figure 68. Original image and selection of pixels that correspond to two colours after colour quantisation.

To address this problem, the utility performs a morphological transformation (an "open and close image" operation [286]) on each quantised layer to close small holes and smooth intricate boundary details that would complicate the physical crafting of the final artefact. The computation implements a simple image segmentation method based on colour quantisation and morphology; depending on style, numerous other segmentation variants can be utilised [134, 135]. For this approach and seven colours, the results are shown in Figure 69. In this case, the segmentation process results in 44 individual pieces. The hair (yellow_ and shadow (dark grey) layers give rise to multiple small pieces.

Figure 69. Original image and binary masks obtained from colour and morphological image segmentation.

Thereafter, we used the same method as for the stained-glass windows to create the pieces to be printed. However, this time we did not include the metallic skeleton rig, since the pieces should be glued together. The image below illustrates the expected result. Numerous materials and illumination configurations can

be used to investigate the interaction of light with semi-transparent and translucent materials that are printed or moulded and used for internal decoration. In Figure 70 (top two rows), we illustrate the generality of the approach by simulating a light source, a tilted colour pane, and two planar and grey surfaces. Clear-coloured glass is simulated in this case to create the coloured projection on the background wall. As demonstrated, the simulation is capable of predicting the illumination effects introduced by the designed artefact. In Figure 70, we changed the glass surface to be rough and, thus, scatter light more. In the bottom row, real 360 images are used for the environment illumination to predict how the design would suit specific environments.

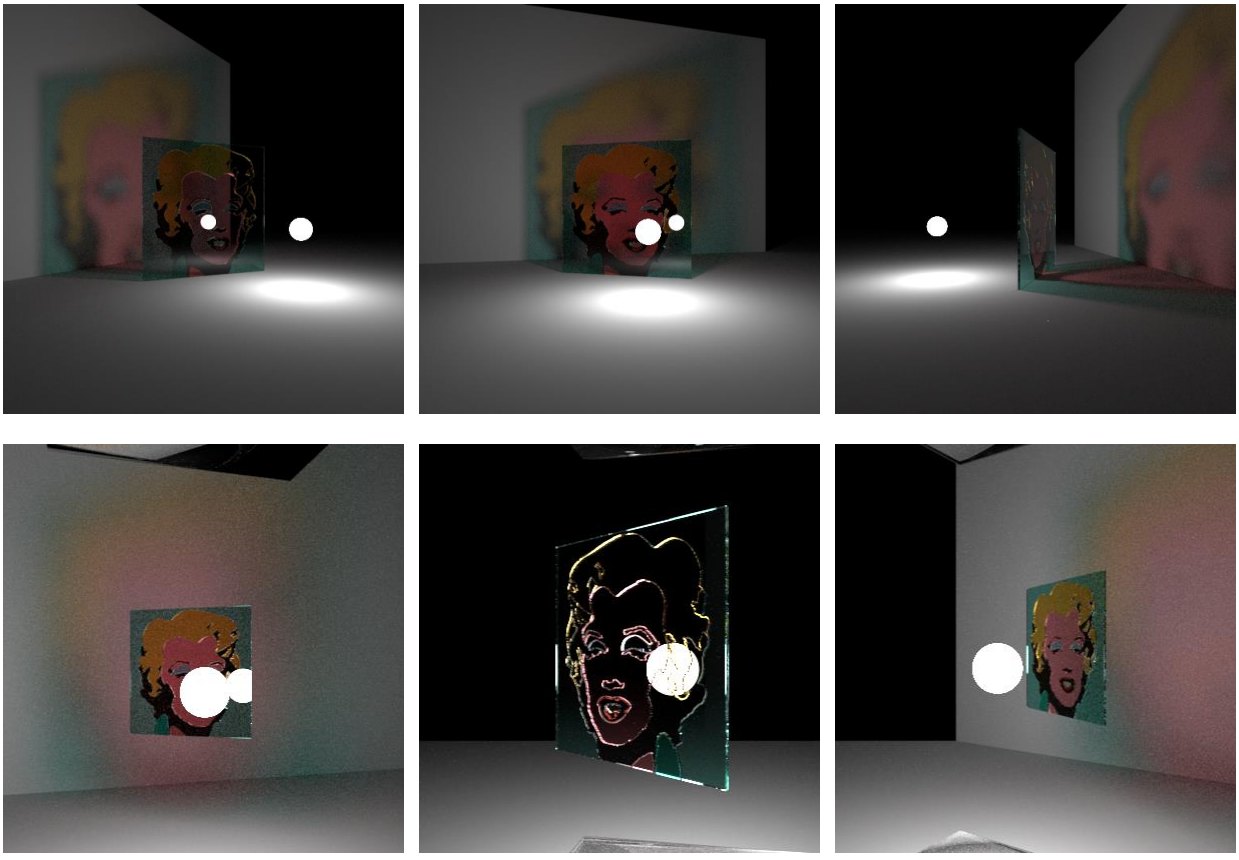


Figure 70. Top two rows: simulation experiments that illustrate the interaction of light with semi-transparent objects (see text). Bottom row: photorealistic previews of the designed artefact, in specific environments, from both sides. Video: https://www.youtube.com/shorts/vKlDdEHxMqA.

9.2 3D objects

Using the same artwork and the original photograph[5] used to create the silkscreens, we extend the composite-object utility from planar assemblies to 3D, multi-piece sculptures that a practitioner can fabricate and assemble. Each piece corresponds to a colour region in the artwork, so the final artefact is an assembly of separately manufactured parts rather than a single monolithic print. The key additional ingredient required to "lift" the 2D artwork into 3D is a depth field: we obtain this by applying a monocular

[5] Publicity portrait of Marilyn Monroe as Rose Loomis in the 1953 film Niagara. Photograph by Gene Kornman.

depth-estimation method to the original photograph, and then transferring the resulting depth map onto the stylised artwork via image registration.

9.2.1 Depth from photograph

To obtain a plausible depth structure, the original photograph is processed by the monocular depth estimation method referenced in [286]. This yields a dense depth map aligned with the photograph, which can be interpreted as a height/relief prior for subsequent 3D construction.

Figure 71 shows the source photograph and the estimated depth map. This pair defines the photometric-to-geometric step that underpins the entire 3D composite pipeline: once depth is available in the photograph's coordinate system, it can be transferred (warped) to other image representations of the same subject.

Figure 71. Monocular depth estimation is used as a 3D prior. Left: original photograph used as input to the depth-estimation method. Right: the generated depth map (relative depth field) produced by the monocular approach referenced in [286]. This depth map is later transferred to the stylised artwork via 2D image registration.

9.2.2 Depth transfer

Because the photograph and the artwork are two different images of the same subject, we apply conventional 2D image registration to align them. The estimated depth map is then "transferred" to the artwork by applying the same alignment transform, yielding a depth value for each location in the artwork image. Next, each segmented colour region is converted into a separate 3D solid by combining its 2D footprint with the transferred depth field. The result is a set of colour-indexed parts that can be printed individually and assembled as a physical object.

A practical advantage of this representation is that it naturally imposes assembly constraints, similar to a jigsaw puzzle: some parts must be placed before others can be inserted. In the example shown below, three pieces must be placed before the eyelids can be inserted, illustrating that “composite” here is not just a visual layering but a manufacturable multi-part structure with a real assembly order.

Figure 72 visualises the conversion from segmented colour regions to individual solids and demonstrates the assembly logic in a physical-print workflow. The top panels emphasise that each colour region becomes a separate printable part; the intermediate panels illustrate the assembly sequence and its ordering constraints; and the final panels provide a preview of how the same geometry would appear under different fabrication/material assumptions (e.g., matte filament versus resin-like appearance).

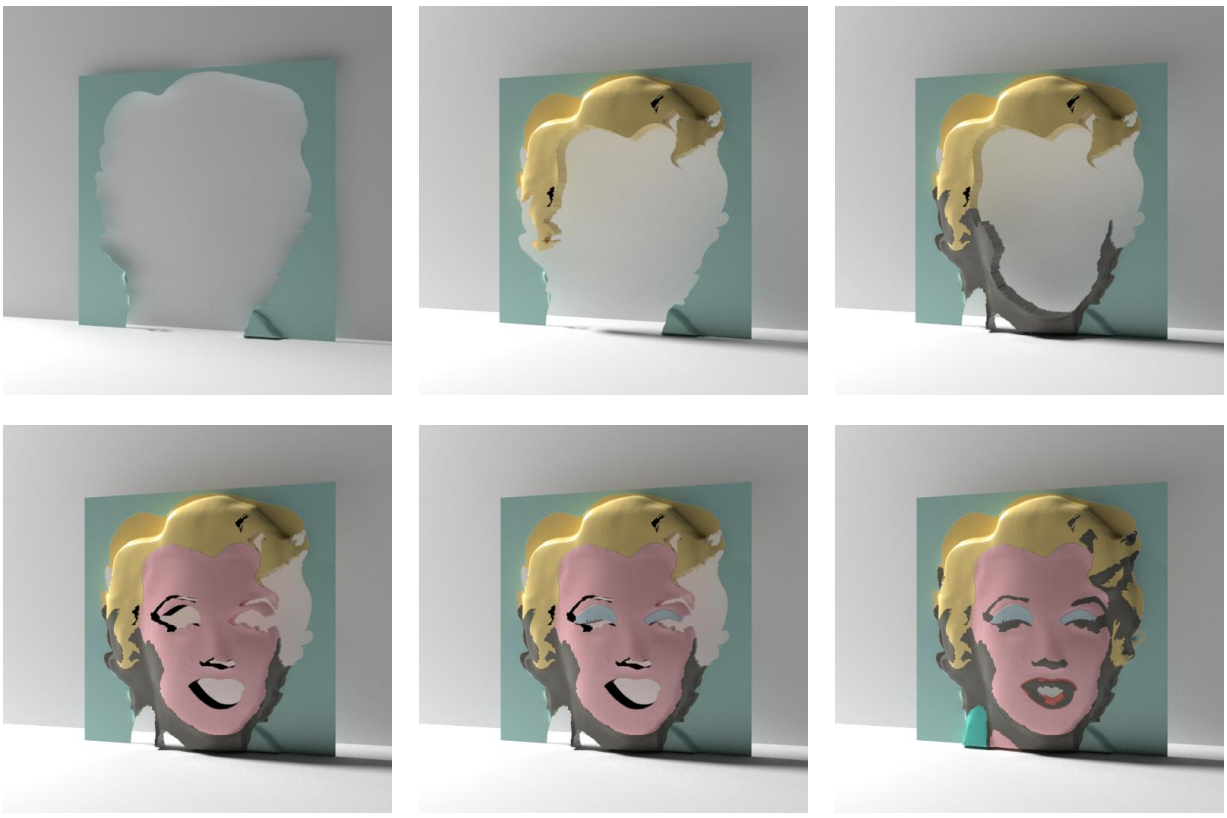


Figure 72. From colour segmentation to a multi-part 3D statue and its assembly sequence. The segmented colour regions are transferred into 3D by combining their 2D footprints with the registered depth field, producing one solid per colour. The figure illustrates representative assembly stages (with ordering constraints analogous to a jigsaw), and shows the final assembled result; the final row provides a material/print-style preview (e.g., resin-like finish) of the same assembled geometry.

Beyond printability and assembly, the same composite representation supports appearance prediction for artefacts made from mixed materials. Once the object is represented as a set of solids with explicit part identities, the designer can assign different materials per part (e.g., plastic, metal, coloured glass) and render the assembled object under controlled environments, reusing the light-transport/rendering capabilities introduced earlier in the deliverable. Figure 73 demonstrates this idea by rendering the same assembled composition with an assortment of materials from multiple viewpoints.

Figure 73 shifts the validation from “can we manufacture and assemble the parts?” to “can we preview plausible material outcomes before fabrication?” The renderings illustrate how part-wise material assignments change the perceived balance of colour, gloss, and translucency, which is particularly useful when a practitioner is choosing between low-cost prototyping materials and higher-end final materials.

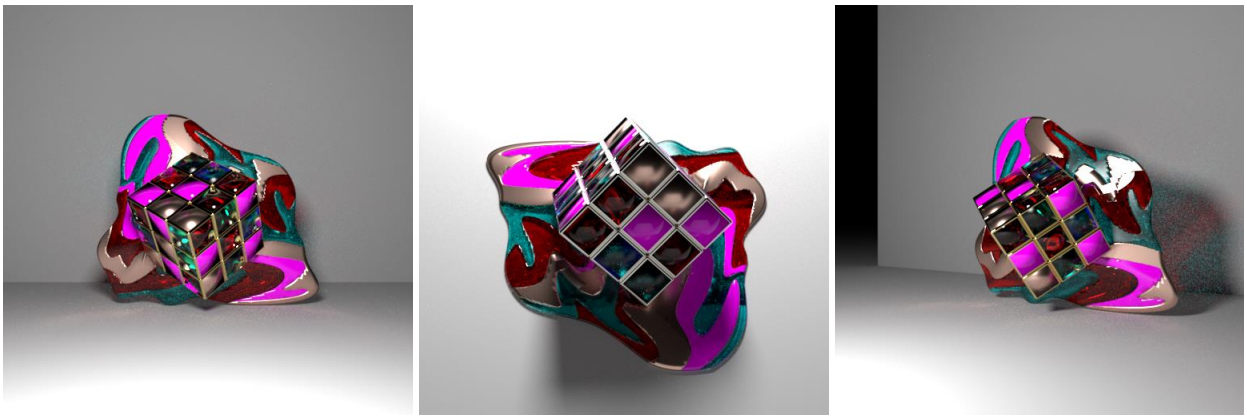


Figure 73. Multi-material appearance preview of an assembled 3D composite. The assembled sculpture is rendered with an assortment of materials (e.g., plastics, metals, and coloured glass) and shown from three viewpoints, illustrating how the same geometry can be evaluated under alternative per-part material assignments before fabrication.

The 3D composite workflow in this subsection demonstrates how a conventional 2D artwork can be lifted into an assembly-ready, multi-part sculpture by combining three ingredients: part definition, monocular relief/volume field, and the transfer of depth from a photograph onto the stylised artwork. The resulting parts are not merely visual layers: they become separate solids whose interfaces impose an explicit assembly order (akin to a jigsaw puzzle), embedding a practical manufacturing constraint directly into the digital design. This is valuable for craft-facing pipelines because it turns a design into fabricable components with an implicit assembly logic, while still allowing rapid exploration of alternative material choices through rendering. This establishes a modular route from image → parts → solids → assembly → appearance preview.

9.3 Lamps

Lamps are a natural “stress test” for composite-object design because their appearance is determined by both the material composition of the object and the lighting regime. Unlike passive artefacts that are viewed under external illumination only, a lamp must be assessed under (i) external lighting and (ii) internal lighting emitted by the light source and transmitted through the shade. This makes lamps an ideal application of the Craeft visualisation toolbox: by explicitly modelling light sources in the scene, we can preview how a design will read in realistic viewing contexts before committing to fabrication; in this context, Tiffany-style designs, where coloured glass and seams are central to the aesthetic.

9.3.1 External versus internal illumination

In the renderer, lamps can be studied under two complementary configurations. In the external-illumination configuration, the lamp is treated as a passive object lit by the environment (e.g., an HDRI), which makes it easy to evaluate surface finish, specular highlights, and the overall silhouette. In the

internal-illumination configuration, one or more emitters are placed inside the shade (approximating a bulb/LED source), so that the rendered appearance includes transmitted colour, glow gradients, and shadowing caused by seams and structural elements. Comparing the two is important: a design that looks balanced under external illumination may look overly dark, overly saturated, or uneven once backlit.

Figure 74 contrasts these two configurations on representative 3D lamp models. The top row shows the lamp under external/environment lighting only; the bottom row adds an internal light source, revealing how the same geometry and materials can produce a qualitatively different visual outcome once the shade is backlit.

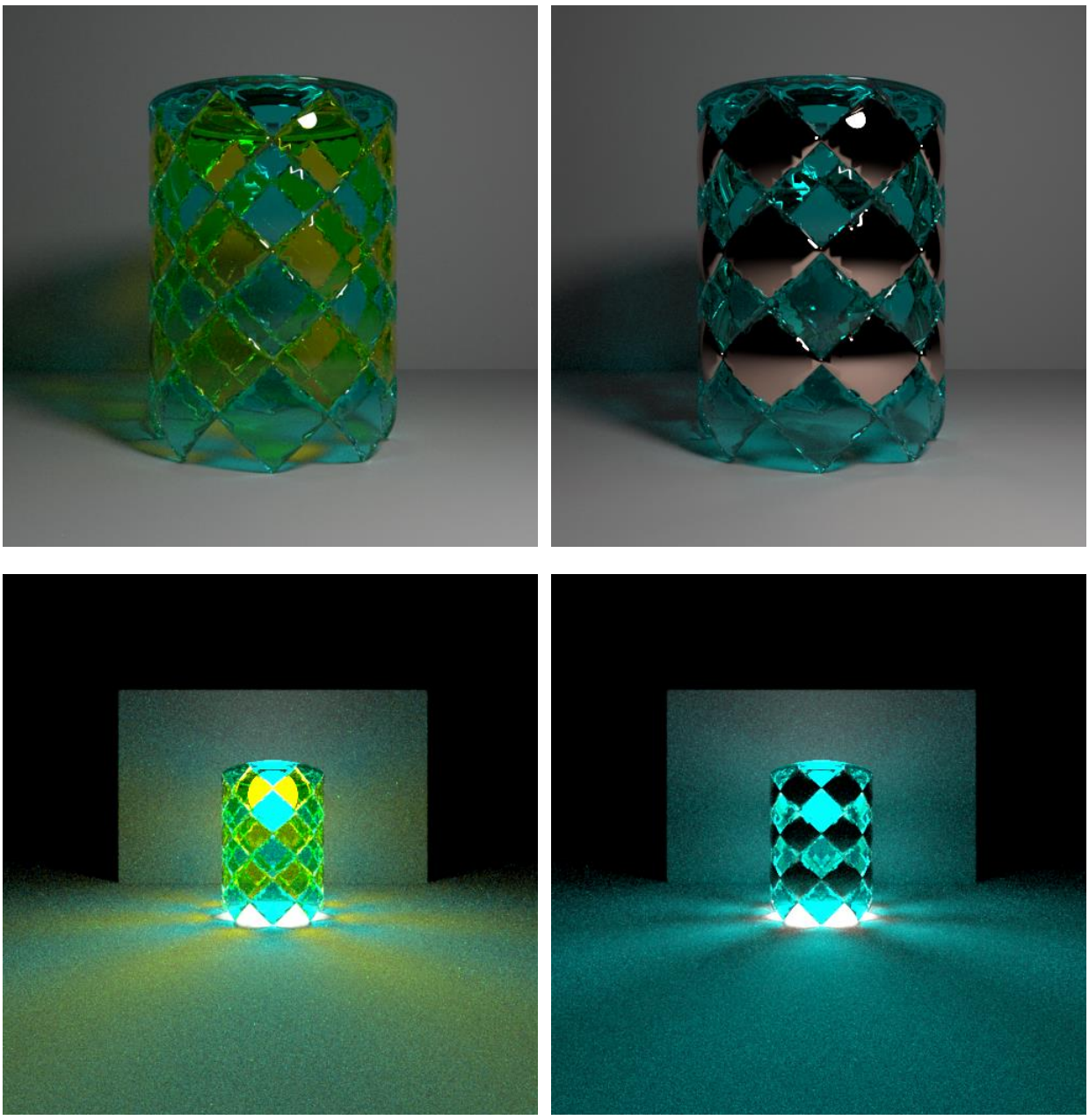


Figure 74. Lamp appearance under external versus internal illumination. Renderings of representative 3D lamp models under two lighting regimes: external illumination (top row; environment light only) and internal illumination (bottom row; emitter placed inside the shade). The comparison highlights how backlighting changes perceived colour, contrast, and seam visibility.

9.3.2 In-situ previews

Because lamps are typically installed in lived-in spaces, the perceived result depends strongly on the surrounding environment (room colour, nearby surfaces, and ambient light directionality). A practical way to study this is to render the same lamp in multiple environment maps captured with a 360° camera. This allows controlled "what-if" comparisons of the same design across contexts without re-modelling a full room.

Light sources can be added to the virtual environment to simulate the appearance, but the light emanates from lamps. The figure below shows the same lamp rendered in different HDRI environments captured with a 360° camera. All lamp parameters are held constant; only the environmental illumination is changed. This isolates the contextual effect of the installation setting on the perceived balance of colour and brightness.

Figure 75. Same lamp in different captured environments (HDRI-based preview). The lamp is rendered with the same geometry, materials, and internal light settings, while the surrounding illumination is varied using 360° HDRI environment maps. This provides a fast, repeatable way to assess how a Tiffany-style design will read in different installation contexts. Video: https://youtu.be/Crfac6MLmyQ

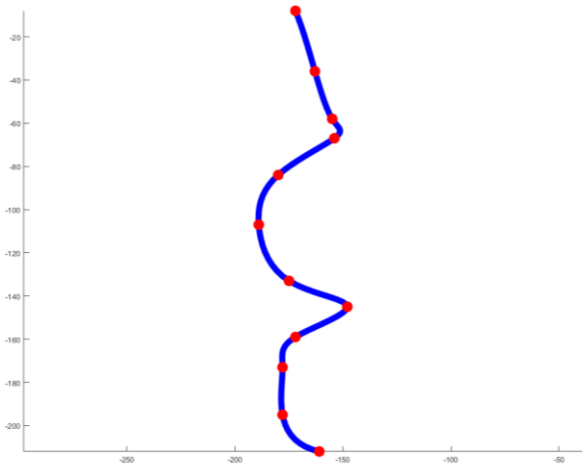
Though light sources can be added to the virtual environment to simulate the appearance, the light emanates from lamps. Following the design paradigm of the so-called "Tiffany lamps", we can preview

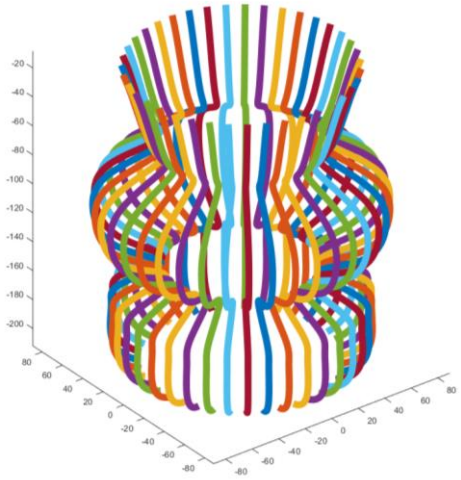
how the implementation of designs will appear in a certain environment. The figure below illustrates the appearance of the same lamp in different environments, acquired using a 360° camera.

9.4 Cane work

In glassblowing, cane refers to rods of glass with colour; these rods can be simple, containing a single colour, or they can be complex and contain strands of one or several colours in the pattern. Cane working refers to the process of making a cane, and also to the use of pieces of cane, lengthwise, in the blowing process to add intricate, often spiral, patterns and stripes to vessels or other blown glass objects. Cane working is an ancient technique, first invented in southern Italy in the second half of the third century BC and elaborately developed centuries later on the Italian island of Murano.

An application is provided that simulates the appearance of cane work compositions.[6] The user specifies the number, shape and dimensions of the canes that will comprise the result. In addition, a twisting parameter is provided that determines the final shape of the simulated artefact.





[6] This application is part of the Visualisation Toolbox in Section 7.1; see manual for usage.

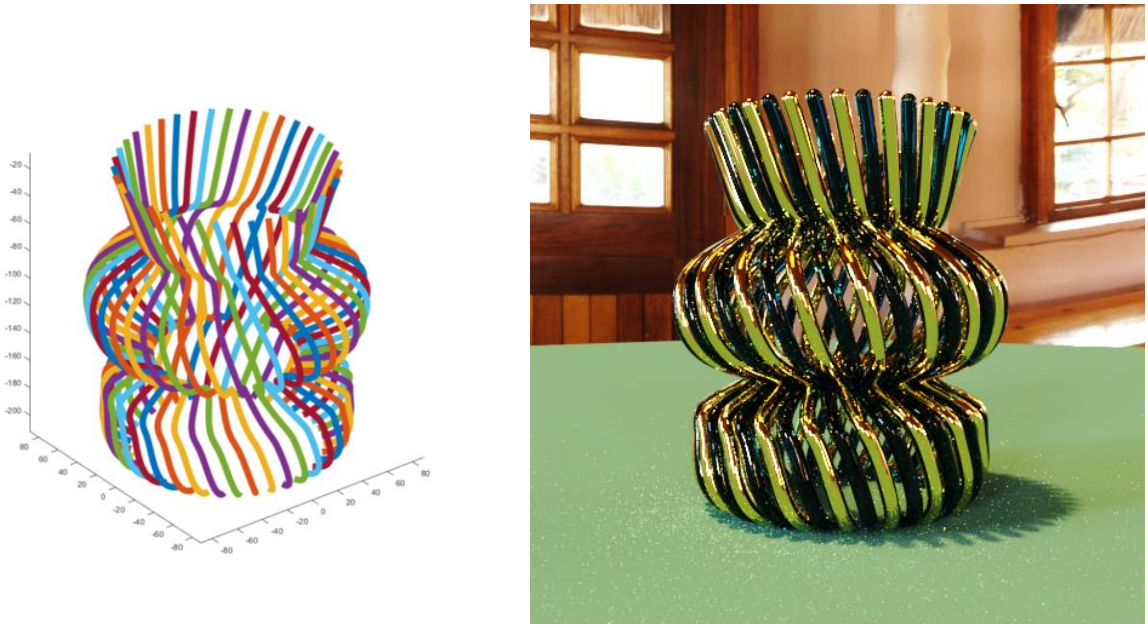


Figure 76. Numerical design of cane work structures and rendering (see text).

In glassworking, cane work refers to a technique involving the use of long, thin rods of glass, called "canes," which are created by stretching molten glass into slender strands. When these canes are then incorporated into the blown piece, they produce decorative effects. A cane-working simulator predicts the appearance of artefacts created from glass and metal canes, using nominal descriptions of the number of canes, their twisting, composition, and thickness. In the figure below, the appearance of cane work artefacts is predicted, created from glass and metal canes. The left and middle images show the same design implemented with different glass colours. The image on the right shows a composition of glass and metal.

Figure 77. Photorealistic rendering of canework designs.

Traditional stained-glass windows used the "came glasswork" process of joining cut pieces of stained glass. Copper foil is a versatile alternative to the traditionally used lead. To facilitate the design of stained-

glass windows, a software utility was developed. Given a colour image, this utility creates the 3D models of the parts of a composition, that is, the metallic framework and the glass pieces.

9.5 Conclusion

Section 9 demonstrates how the Craeft methods can be used to design composite artefacts. The core technical idea is to treat composition as a controlled pipeline from conventional imagery to part definitions, and then to fabrication-ready geometry that can be previewed under physically meaningful rendering conditions.

For planar composites, the section showed that naïve colour quantisation produces highly fragmented regions with holes and intricate boundaries that are impractical to manufacture or assemble. The introduced morphological clean-up step operationalises a craft constraint (assemblability) by smoothing boundaries and closing holes, yielding a manageable set of pieces that can be produced and bonded. For 3D composites, the same decomposition is extended by transferring a monocular depth estimate to the segmented artwork, converting each region into a solid part and validating the feasibility of assembly through an explicit sequence. This makes the dependency chain explicit: segmentation controls part count and interface complexity; depth estimation controls relief/volume; both directly affect manufacturability and assembly burden.

Beyond geometry generation, the section emphasised the value of appearance prediction for composite designs. Multi-material render previews enable early evaluation of material choices, and the lamp examples illustrate how the same design behaves under external versus internal illumination and across different environments, making lighting part of the specification rather than an afterthought. The cane-work simulator illustrates another composition mode, as a parametric structure synthesis that is previewed through the same rendering pipeline.

10 Stained Glass Compositions

Section 10 addresses stained-glass compositions as a craft domain where digital design can contribute to coping with production constraints. Unlike freeform illustration, a stained-glass design must terminate in a buildable partition: a set of pieces separated by came or foil lines, each piece uniquely identifiable, printable at scale-true dimensions, and exportable in formats that remain reliable across workshop toolchains. Section 2.5 reviewed this software landscape and showed that practitioners routinely work in raster-vector pipelines and rely on pattern-centric functions. This section builds on that background and positions Craeft's contribution as a reusable, craft-aware workflow that can support both design exploration and documentation.

Technically, the section focuses on the computational problem that underlies most stained-glass authoring tools: how to convert an intended motif into a partition of the plane that satisfies competing constraints. The partition must be visually faithful, structurally sensible, and manufacturable. In addition, stained glass is appearance-sensitive: material choice and lighting context determine whether a composition "reads" correctly once installed. For this reason, Section 10 also connects composition authoring to the project's visualisation capabilities, enabling design previews under controlled illumination where appropriate.

Finally, Section 10 is written with reuse in mind. The outputs of stained-glass authoring are images but also structured design artefacts that can feed later modules: fabrication templates, composite assemblies, light-transport studies, and documentation assets. In this sense, stained-glass compositions serve as a representative case of Craeft's overarching approach: integrate craft constraints early, maintain traceable intermediate representations, and produce outputs that can be reused across authoring, simulation, and communication workflows.

10.1 Application

We present a design utility that takes an input image of a stained-glass composition, such asa drawing, a scanned sketch, or a coloured design, and produces a fabrication-oriented digital model of the panel. The output is a 3D assembly comprising (i) the individual glass pieces and (ii) the lead came framework that separates, supports, and constrains them. This way, the tool supports rapid prototyping and production preparation.

A practical motivation is that much conventional stained-glass software assumes the user will manually draw the lead-line pattern (the stencil) and then use the software primarily to refine, number, and print that pattern. By contrast, the Craeft utility aims to provide design synthesis: it automatically derives a manufacturable camework partition from any source image by performing colour-quantisation-based segmentation and subsequent geometric processing. This dramatically reduces the time from artistic inspiration to a buildable structure, and opens the way for the utilisation of stock designs or customer-specific orders.

This automation introduces a specific CAD-for-fabrication challenge: direct segmentation and boundary tracing often preserve high-frequency jaggedness and create narrow inward cusps. In stained glass, these become fragile points that are difficult to cut, grind, and assemble reliably. For that reason, the pipeline does not treat segmentation as the final artefact. Instead, segmentation is followed by quality testing and

regularisation steps that detect and characterise intrusions and then apply remedies to yield boundaries that are structurally viable for glass cutting and came/foil assembly.

Beyond geometry and manufacturability, the outputs are designed to integrate with Craeft's visualisation capabilities. Once a composition is represented as explicit glass pieces and cames, it can be rendered photorealistically and combined with lighting configurations to support appearance studies. This includes predictive visualisation of how daylight direction changes affect the panel's projected patterns, and the inclusion of internal light sources to preview how a design reads under different environments. These demonstrations position stained-glass composition design as a workflow that links image-based authoring → manufacturable partition → 3D assembly → lighting-aware preview, reusing the same project infrastructure rather than relying on isolated, tool-specific representations.

Operationally, the application is organised as a two-step workflow:

1. **Input image → segmentation**: convert the input design into labelled regions corresponding to candidate glass pieces and the separating came network.
2. **Segmentation → 3D model**: generate the 3D geometry of glass tiles and cames from the segmentation, mapping colour information (when present) onto the corresponding 3D glass components.

The remainder of Section 10 details each stage: segmentation and its parameterisation, geometric quality testing and boundary regularisation, the construction of the 3D tiles and came framework, and representative results that illustrate manufacturability and lighting-aware preview.

10.2 Segmentation

The segmentation step partitions a design image into defined regions representing the individual glass pieces and the lead came framework.

The method handles two types of input: specific leaded light designs or arbitrary images. For the latter, the system converts the image into a design by generating the necessary lead-came framework. In both, a colour image segmentation algorithm is used.

10.2.1 Segmentation of leaded light designs

In this type of input, the images are in vector format and include the lead came framework and are authored on a computer. This segmentation process is, thus, streamlined by the input format. Since the designs are vector graphics, they provide precise boundaries and solid, homogeneous colours for the glass pieces. This eliminates the noise and gradients found in standard photographs, making the segmentation of regions computationally simple.

However, the system remains robust when inputs are not perfectly homogeneous, such as images affected by JPEG compression artefacts. The colour segmentation algorithm is designed to tolerate these minor inconsistencies, grouping slightly varying shades into coherent regions.

The system implements a deterministic, unsupervised segmentation pipeline to partition a raster image I into a set of discrete, perceptually uniform chromatic regions. The workflow comprises the following steps.

Let the input 3-band colour image be defined as I.

1. Lead came framework extraction

The method isolates 'black pixels' that represent the came framework. A binary mask is produced that maps the 2D structure of the framework.

2. Perceptual Colour Space Transformation

To align the segmentation with visual intuition, the masked image is mapped from the RGB source space to the CIELAB colour space [156]. The interesting feature of this space is that Euclidean distance in CIELAB space corresponds (approximately) to felt (perceptual) colour difference. This way, we have a metric measure of dissimilarity, which we will need in the subsequent clustering phase.

3. Colour quantisation

The aim is to reduce the number of colours in the image while preserving the essential visual characteristics. This step is crucial for simplifying the segmentation process. Pixels are grouped into k clusters based on colour similarity. The centroid of each cluster represents the colour for that segment.

Distinct from the chromatic segmentation to preserve high-frequency information that might be lost during colour quantisation. This is performed in the HSV colour space, as $M_{dark}(i) = I(V(x_i) < \tau_v \wedge S(x_i) < \tau_s)$, where τ_v denotes the brightness threshold and τ_s denotes the saturation threshold. For our case, these thresholds are tuned to mild values because our goal is to counter minute colour artefacts due to image compression or poor digitisation of the design. Morphological operations (opening and small-object removal) are applied to M_{dark} to reduce noise [157].

3. Unsupervised Colour Clustering via K-Means

The core segmentation utilises the K-Means algorithm [158, 159] to partition the pixel set $X = \{x_1, ..., x_N\}$ (where N is the number of non-background pixels) into k clusters $C = \{C_1, ..., C_k\}$. The algorithm minimises the within-cluster sum of squares (variance), that is, the sum of all pixelwise colour differences from the 'mean colour' μ_j, where μ_j is the centroid of cluster C_j. If the number of segments k is not user-defined, the system iteratively evaluates a range of k values using the Silhouette Score [160]. The silhouette coefficient s(i) for a sample i is s(i) = (b(i) - a(i)) / (max{a(i), b(i)}), where a(i) is the mean intra-cluster distance and b(i) is the mean nearest-cluster distance. The optimal k maximises the mean global silhouette score.

4. Post-Processing and Layer Reconstruction

Upon convergence, every pixel x_i is assigned the colour value of its corresponding centroid μ_j. To mitigate the 'hard assignment' artefacts typical of K-Means, the system applies post-processing filters. Specifically,

median filtering, closing, and dilation are employed to smooth boundaries and merge small, incoherent regions [161].

5. Connected-Component Labelling

The binary mask is analysed to identify all contiguous foreground pixel groups, each of which is initially treated as a separate region. The result is a labelled image in which each distinct region carries a unique identifier for further processing.

10.2.2 Segmentation of generic colour images

This method differs from the previous one in two features. The first is that colours are allowed to vary more. This is to cater to cases of paintings or photographs of the designs. The second is that some designs do not explicitly model the came framework; sometimes, because they were not drawn as came framework designs. In this case, the user specifies the thickness of the glass pieces and that of the metallic rig.

1. Colour-based clustering and segmentation

As above, we quantise the image and then segment it using the same algorithm into distinct regions based on colour. The image is segmented into areas that share a similar colour. The method used for this segmentation is that described in (Balasubramanian et al, 2008), although other segmentation techniques are also applicable. Starting from seed points, regions are grown by adding neighbouring pixels that have similar colour values.

In this case, we are very strict when merging pixels into the same cluster threshold, resulting in an 'oversegmented' image. This way, a seemingly homogeneous colour region may be segmented into multiple sub-regions that exhibit similar but not identical colours.

To merge over-segmented regions and enforce spatial coherence, a statistical region-merging step is applied that efficiently merges adjacent areas based on their intensity distributions [162].

The mean colour is assigned as the colour of the piece of stained glass needed to represent the corresponding image region. The mean colour of each region is calculated in the Lab domain, taking care of the circularity of the 'hue' (L) component.

2. Pieces and came skeleton

Neighbouring regions in the segmentation result are 'connected' by their boundary pixels, with their union covering the template image. This leaves no space for the metallic skeleton intended to hold them together. Consequently, each region must be uniformly reduced in size (or 'shrunk') to create a consistent gap thickness between pieces. Simple scaling is not viable due to varying shapes, as it leads to unequal gaps. Therefore, morphological operations are employed to refine the segmented regions. Opening, which removes small foreground objects, involves erosion followed by dilation. Closing, which fills small foreground holes, involves dilation followed by erosion. These operations define segment boundaries, preparing them for conversion into glass pieces.

10.2.3 Segmentation Results

This subsection summarises the behaviour of the two segmentation pipelines introduced above and illustrates their typical outputs on representative inputs. The purpose is twofold: (i) to confirm that the segmentation step produces a piece-wise partition of the template image, and (ii) to show that the produced regions are immediately usable as geometric input for the subsequent 3D modelling stages, where segmented regions become glass tiles and their separating gaps become the came framework.

Figure 78 presents a side-by-side comparison. In each row, the left image is the input, and the right image is the corresponding segmentation output expressed as a piece boundary map.

- Top row (leaded-light design input; Section 10.4.1): the input is a digitally authored design in which the came framework is explicitly present, and colour regions are largely homogeneous. The method therefore yields a clean partition: the came is extracted directly, and the remaining pixels are clustered into perceptually uniform regions, producing sharp boundaries aligned with the authored design.
- Bottom row (generic colour image input; Section 10.4.2): the input is a photographic/painted image in which colours vary continuously, and the came framework is not explicitly given. The method first over-segments the image and then merges regions to enforce spatial coherence; finally, each region is uniformly "shrunk" to reserve a consistent gap for the came. The resulting boundary map provides a feasible stained-glass tessellation that approximates the dominant chromatic structure of the image while suppressing small-scale texture.

These examples demonstrate that the segmentation stage can accommodate both clean design templates and more challenging real-world imagery, while producing outputs in a consistent format suitable for downstream geometry generation and visual evaluation.

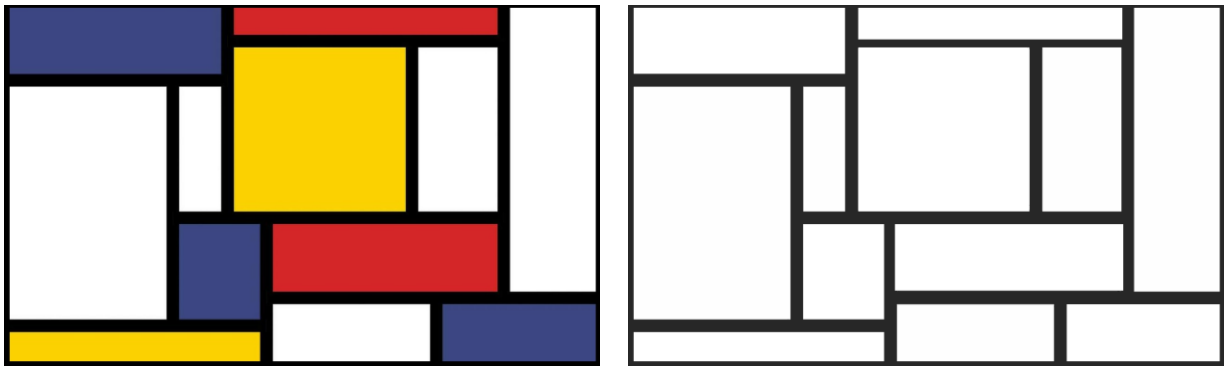


Figure 78. Segmentation outputs for the two supported input classes. Left column: input images. Right column: segmentation result as a stained-glass piece boundary map (with gaps reserved for the came framework). Top row: vector-/design-style leaded-light input with explicit came, yielding sharp, design-aligned regions. Bottom row: generic colour image processed via over-segmentation, region merging, and uniform shrinking to synthesise a feasible piece layout and came spacing.

10.5 Quality Testing and Parameter Tuning

Stained glass designs take into consideration the sensitivity and fragility of glass components. Practitioners do not cut deep concavities in a single piece, as it would break. When a complex shape is needed, they split it into two or more simpler pieces. This process is essential because of the material properties of glass. Intrusions result in a glass piece with a sharp, fragile point protruding inward. When the glass is cut, these narrow points concentrate stress and are extremely prone to breaking or chipping, rendering them unusable or difficult to join. In Pye's terms, the practitioner interprets the design under the limits of the making process: complex outlines are decomposed into simpler pieces, because deep concavities in a single glass piece would almost certainly break.

This subsection offers a safety test to alert users to the potential fragility of their proposed designs. This assessment detects inward intrusions that prevent the robust implementation of a design. In addition, the

method provides a remedy suggestion by replacing sharp inward structures with a straight-line bridge. The technique operates on the image contour.

10.5.1 Intrusion point detection

First sharp intrusions are detected. We do not wish to eliminate *all* intrusions but only the sharp ones that inhibit manufacturability. As such, we need to characterise and quantify intrusion sharpness to decide if it shall be annulled or not. Similar phenomena hold for sharp extrusions. As glass pieces complement their neighbours, the avoidance of deep recesses in any one piece also suppresses sharp projections in the others.

The method detects intrusions by assessing the local curvature and position of each contour point in an arclength-oriented scale-space fashion.

The contour, C, is represented by an ordered set of N coordinate pairs $C = \{(x_i,y_i) \mid i = 1, 2, \ldots, N\}$ that interpolate the contour in equal arclength steps. The i-th point of C is denoted as P_i. The centroid of contour C is denoted as C_K.

We first detect contour points that belong to intrusions. We use integer $k \in [1, N-1]$ as the arclength that determines the 'observation scale' of the candidate point.

For each P_i:

1. A hypothetical, straight anchor line L is established between two points, $P_s = P_i - k$ and $P_e = P_i + k$. Indices are calculated modulo N to handle the closed contour. L serves as a 'stability baseline' for the examined feature.
2. Only inward cusps are addressed via a directional filter. Deviations are characterised as intrusions or extrusions by checking if P_i lies on the same side of the L as C_K.

To avoid selecting a particular observation scale, the method runs this detection operation for a range of k values and collects all the detected intrusion points. Since a P_i may be characterised as an intrusion point at multiple scales, the accumulated points are made unique by excluding repetitions.

Then, contiguous points that satisfy the deviation and intrusion criteria are grouped into an intrusion segment. Intrusion segments are ordered using the point indices.

We did not use morphological operations to solve this problem. The reason is that they are unsuitable because their effect is uniformly distributed across the boundary; as a side effect, they would modify the entire design rather than the particular intrusions. Aggressively tuning morphological parameters to target a large, deep intrusion would result in the undesirable erosion or dilation of all large, important features in the image, thereby distorting the overall geometry of the stencil. In contrast, the constrained bridging method is a point-list manipulation that only modifies the specific, validated segment of the contour, leaving the rest of the shape's definition mathematically precise and untouched.

10.5.2 Intrusion characterisation

Next, we characterise the sharpness of intrusions. Let the endpoints of the examined segment be P_s and P_e. These endpoints define another hypothetical line, L_s, in the same way as above. Let also P_m be the farthest segment point from L_s.

The sharpness of the intrusion is quantified by the distance from P_m to L_s. This distance must be less than a tolerance threshold to qualify as sharp and significant. To cope with scale, this distance is normalised with the contour area.

If the condition is satisfied, all interior points of the intrusion segment are removed from the contour list, and P_s and P_e become neighbouring points. The operation essentially bridges the edges of the intrusion, replacing the high-frequency curvature with a straight-line segment.

10.5.3 Intrusion repair by contour bridging

In practice, the bitmap-to-contour conversion may produce intrusions: narrow inward dents or gaps in an otherwise smooth region boundary. These artefacts are undesirable because they create (i) spurious small pieces, (ii) locally self-intersecting or jagged outlines after offsetting, and (iii) unstable came placement when the contour is later converted to geometry. We therefore apply a simple geometric repair that removes an intrusion by bridging its boundary points, i.e., by completing the contour across the mouth of the dent.

Conceptually, an intrusion is detected as a pair of boundary locations that are close in Euclidean distance yet far along the contour order: the contour "walk" must travel deep into the dent and back.

Let the ordered boundary samples be (p_i, $i \in [1, N]$). We identify candidate pairs (p_i, p_j) such that:

- $| p_i - p_j | \leq \tau_d$ the two sides of the dent are proximate, and
- $| i - j | \geq \tau_s$ along the cyclic contour indexing; that is, they are separated by a substantial arc length on the boundary,

where τ_d controls how "narrow" a dent must be, and τ_s prevents trivial near-neighbour pairs from being selected.

Once an intrusion mouth has been identified by its two border points p_i and p_j, we modify the contour by inserting a bridge segment that directly connects them, and by removing the boundary chain that runs through the interior of the dent. Specifically, we replace the contour subsequence (p_i, p_{i+1}, …, p_{j-1}, p_j) with the single segment ($p_i \rightarrow p_j$). Operationally, this "cuts off" the indentation and yields a repaired contour that is simple, locally convexified at the former mouth, and better conditioned for subsequent offsetting and polygon-to-mesh conversion.

Algorithm
1. Input: ordered contour points p_i, $i \in [1, N]$; thresholds: τ_d, τ_s. 2. Find candidate mouth pairs: search for index pairs (i,j) satisfying the spatial proximity and contour-separation criteria above.

3. Select intrusion chain: for each candidate, evaluate the two possible chains between i and j on the cyclic contour; select the chain whose enclosed area (or maximum inward depth) indicates an indentation.
4. Bridge: insert the segment (p_i, p_j) and delete the selected intrusion chain.
5. Post-process: re-sample/smooth the updated contour if needed, then pass it to the offsetting stage used to reserve came thickness.

This repair is conservative: it removes only those intrusions whose rim is narrow (τ_d) but whose boundary detour is long enough to represent a genuine indentation (τ_s). This way, we suppress segmentation artefacts without erasing intentional concavities that encode the intended glass piece geometry.

10.5.4 Validation

The intrusion-remedy step (Section 10.5.3) deliberately changes the geometry of *one* glass piece by replacing a sharp inward cusp with a straight bridge between the intrusion's border points. While this local modification improves manufacturability, it can also invalidate the *global* assembly: a tile boundary is shared with the came framework, and any change in one component must be propagated so that (i) the lead came remains continuous, (ii) neighbouring tiles still meet the framework without overlap or gaps, and (iii) the resulting stencil remains a consistent partition of the panel image.

For this reason, validation is performed at the *assembly level*, not only at the level of a single contour. We maintain a global binary occupancy matrix, with the same pixel dimensions as the input image, that encodes which pixels belong to each segmented component. When an intrusion is repaired on a given tile, the corresponding region mask is updated, and the framework is recomputed from the revised global occupancy. This ensures that the camework geometry remains compatible with the modified tile set and that downstream 3D extrusion (tiles and came) operates on a coherent, collision-free partition.

Figure 74 demonstrates this validation on a design that was not authored as a clean, fabrication-ready lead-line stencil but instead provided as a *cartoon*: a full-scale preparatory drawing intended to guide the construction of a stained-glass window.[7] The example is a cartoon by the Jersey-born stained-glass artist Henry Thomas Bosdet (1856–1934) for a window depicting the Marriage at Cana, designed for St Aubin on the Hill in Jersey. The scene is visually dense: multiple figures in period dress and banquet details such as a peacock platter, which is precisely the kind of ornamental drawing that tends to produce narrow cusps and deep concavities when converted to piecewise boundaries. In our pipeline, these cusps are detected, characterised, and where they exceed the sharpness threshold, repaired by bridging. Validation then updates the global assembly representation so that the repaired tile remains compatible with its neighbours and with the recomputed came.

Finally, the validated assembly is converted into a 3D composite: each glass piece is extruded into a tile with controlled thickness, the lead came is generated as a separate rigid component, and the full window

[7] The photograph of the design we use is credited to Matt Hotton in the original post. Megan Davies, 'Exhibition celebrates Jersey artist's stained-glass windows', Jersey Evening Post, 2 April 2025: https://jerseyeveningpost.com/news/2025/04/02/exhibition-celebrates-jersey-artists-stained-glass-windows/ The photograph of the design we used in the example is credited to the exhibition photographer, Rob Currie.

is rendered using the Craeft PBR workflow. This closes the loop from a historically meaningful, “non-CAD” design artefact to a physically plausible 3D model whose appearance can be explored under alternative illumination and viewing conditions. Beyond its immediate technical value, this capability supports heritage interpretation: cartoons and other preparatory designs often survive independently of workshop records, and re-instantiating them as validated 3D assemblies makes them inspectable as geometry, not only as drawings.

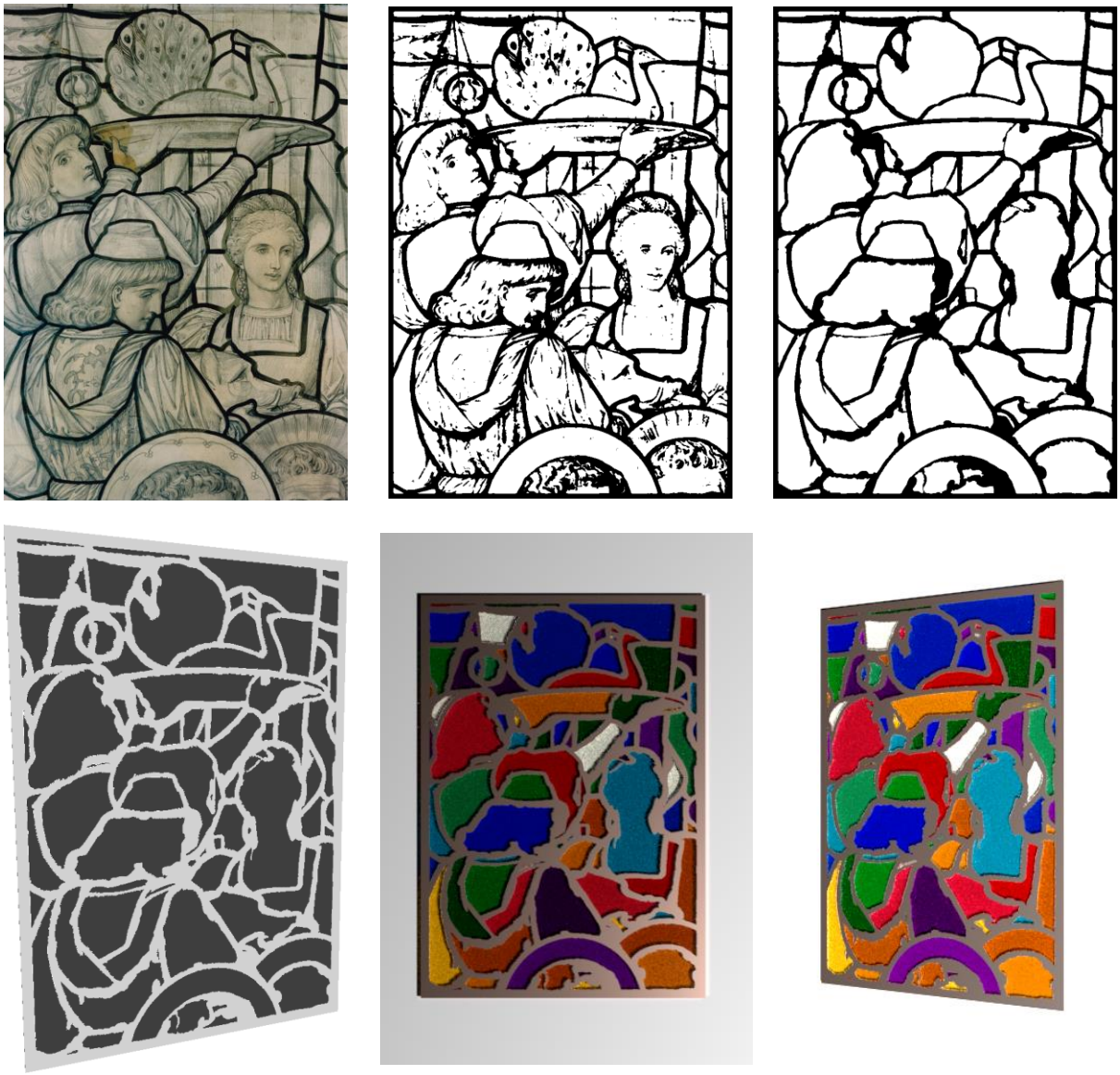


Figure 79. Validation of intrusion repair on a historical cartoon design. (Left) Source cartoon (Bosdet) used as input. (Middle-left) Extracted piece boundaries showing sharp inward intrusions that would yield fragile glass points. (Middle-right) Intrusion repair by constrained bridging of the intrusion border points, removing the high-curvature cusp while preserving the surrounding contour. (Right) Resulting validated 3D assembly (glass tiles + recomputed came), rendered with the Craeft PBR toolbox to preview the visual outcome under physically based light transport.

Unimplemented designs hold a special cultural significance because they testify to lost opportunities and the passage of time in history. They indicate commissions that failed due to financial collapse, shifts in patronage, or major historical events that suspended construction or artistic work. An unpublished design expands the known body of their work, revealing phases, styles, or subjects that were otherwise unknown. We are not just reproducing the image, but completing a historical artefact that was literally waiting for the technical means (or patron) to be realised.

10.6 3D modelling

The provided application takes as input an image and transforms it into a came glasswork composition. The result is a set of 3D models of the pieces that would be needed for the composition.

10.6.1 Tiles

Each image segment represents a piece of stained glass and is extruded into a 3D model. The 2D segment is converted into a 3D object by assigning a uniform thickness to it. A 3D mesh of triangles is generated for each segment to represent it in 3D space.

This section details the procedure for converting individual 2D stained glass image segments into watertight 3D models. The algorithm extrudes each 2D segment to a uniform thickness, denoted as τ, creating a solid object suitable for rendering and light-transport simulation.

The generation process consists of four stages:

1. Boundary Extraction and Triangulation

First, the algorithm traces the contour of the input region to establish its 2D perimeter points, utilising the method described in [14]. Within this boundary, the internal area is meshed into a network of non-overlapping triangles using Delaunay triangulation [19]. This method is selected to ensure the mesh maintains robust connectivity and consists of well-shaped elements (Lee & Schachter, 1980).

2. Vertex Lifting

To transition the geometry into 3D space, the 2D vertices generated in the previous step are converted into two distinct sets of 3D coordinates:

- Base Layer: The original 2D points are assigned a Z-coordinate of zero (0).
- Top Layer: The points are replicated and assigned a Z-coordinate equal to the user-defined thickness parameter (τ).

3. Volume Creation and Sealing

The triangular topology generated in Step 1 is applied to both point sets to form the bottom and top faces of the glass piece. To ensure the model is watertight, the algorithm identifies the boundary vertices of both layers. It serves to 'seal' the object by generating side walls, creating two triangles for every pair of

consecutive boundary points to bridge the gap between the base and top layers. The operation is described below.

1. Let the ordered set of boundary vertices on the base layer (where z=0) be denoted as B = {b0, b1, …, bN-1}, where N is the total number of boundary points. Similarly, let the corresponding vertices on the extruded top layer (where z = τ) be denoted as T = {t0, t1, …, tN-1}. By construction, ti is the vertical projection of bi.
2. Since the contour forms a closed loop, the indices are treated cyclically. For any index i, the subsequent vertex index j is defined as: j = (i + 1) mod N.
3. This ensures that the final point bN-1 connects back to the initial point b0.
4. For each pair of consecutive indices (i, j), the vertices {bi, bj, tj, ti} define a quadrilateral in 3D space representing a segment of the side wall.
5. To tessellate this surface, each quadrilateral is decomposed into two triangles, Δ1 and Δ2. The set of side-wall triangles S is generated by the union of subsets, {(bi, bj, tj), (bi, tj, ti)}, where i ∈ [0, N-1].
6. This operation produces a contiguous strip of 2N triangles that seamlessly bridges the gap between the base and top layers into a watertight model.

4. Surface Properties and Export

Finally, surface normals are calculated across the mesh to facilitate accurate light interactions during rendering. The resulting closed mesh is exported in OBJ format, adding it to a library of individual tile files ready for the colour-mapping stage.

10.6.2 Came framework

To create the skeleton rig, we start with the binary image of the template. A padding is first made to represent the outer part of the rig.

We create the 3D model of the skeleton rig from the template image. This corresponds to creating a 3D model for the region occupied by black pixels. This is achieved as follows. For each point $p = (p_x, p_y)$ of the template except for points in the last column and last row, we define a 2×2 kernel with its top-left point at coordinates (p_x, p_y), which has points (p_1, p_2, p_3, p_4).

We examine the number of black pixels in this kernel and define triangles by reviewing the number of black pixels in this kernel:

1. If this number is 4, then we define 2 triangles (p1, p2, p3) and (p3, p4, p1)
2. If this number is 3, a single white pixel occurs in the kernel. We define a triangle, according to the following subcases, depending on the location of the white pixel (p1, p2, p3), (p1, p2, p4), (p1, p3, p4), or (p2, p3, p4).

Case 1 and the four subcases of case 2 are illustrated in Figure 80, along with the triangle(s) defined in each case.

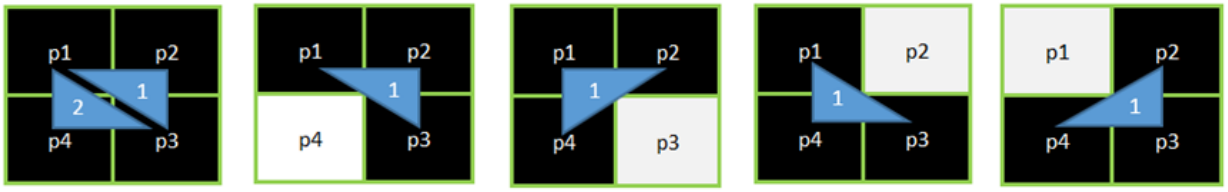


Figure 80. Triangulation cases.

We duplicate the created triangles to provide 'thickness' to the skeleton rig and upgrade them from 2D to 3D, in the same way that we did for the glass pieces. That is, one triangle has Z = 0, and the other has Z equal to the thickness of the rig.

We define the vector normal for each triangle by ordering (p_1, p_2, p_3) so that the normal points outwards to the skeleton. For each triangle, we order its vertices so that they denote normals towards the external part of the skeleton. For the upper triangles (where z = thickness), vertices are ordered so that their normal vector is $[0, 0, 1]^T$, and the opposite ($[0, 0, -1]^T$) for the bottom triangles.

Using the same method as for the glass pieces, we collect all outline border points. As we did for the triangles, we duplicate these points for the upper and lower parts of the rig. This task is illustrated in Figure 81 below.

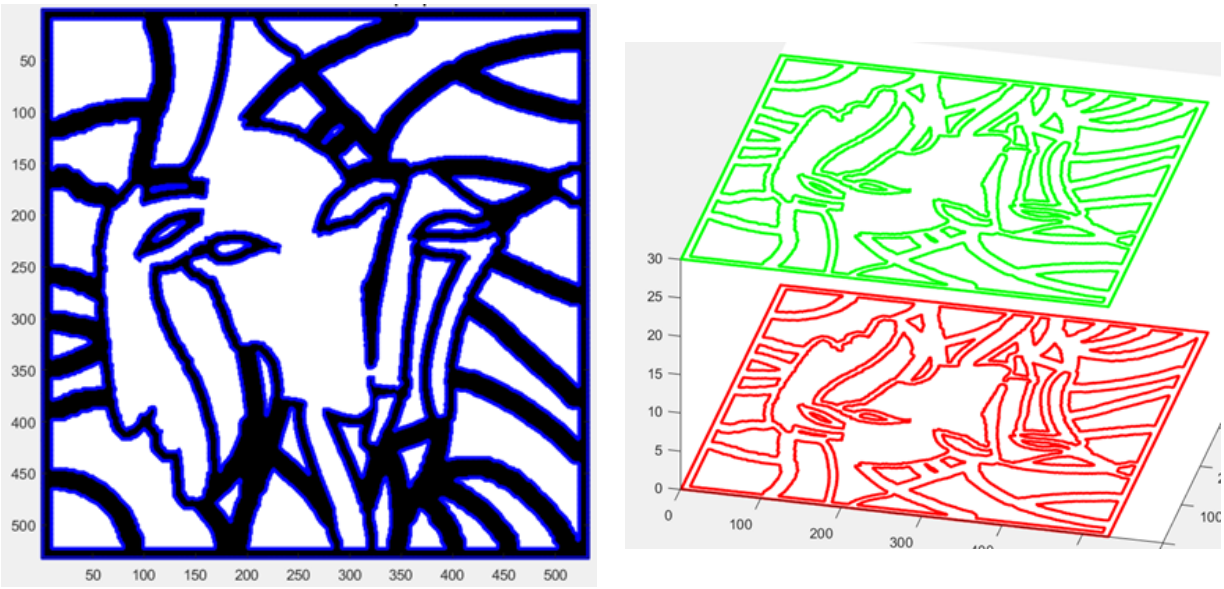


Figure 81. 2D boundary points of the skeleton rig and their conversion to two layers of 3D points.

Then we need to define the vector normal for each triangle and, thus, we order ($\mathbf{p}_1$, $\mathbf{p}_2$, $\mathbf{p}_3$) so that the normal points outwards to the skeleton. For this running example, the result is shown in Figure 82.

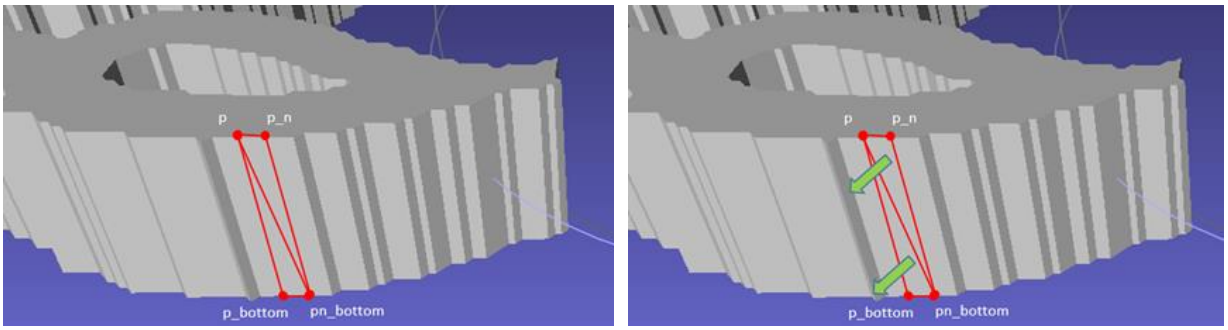


Figure 82. Triangulation of lateral faces of the skeleton rig (left) and definition of their vector normal (right).

Figure 83 shows the 3D model of the result (left) and the 3D models of the glass pieces to be placed within the skeleton.

Figure 83. 3D model of a came framework (left) and its glass pieces (right).

10.7 Results

This subsection demonstrates end-to-end outputs of the stained-glass composition pipeline introduced: starting from a colour source image, the system segments the design into glass-piece regions and then generates a 3D assembly comprising (i) the individual glass tiles and (ii) the came framework that structurally binds them. The resulting geometry supports both fabrication-oriented inspection and photorealistic preview under controlled lighting.

We report three representative source images (Figure 84). For each, we show multi-view renderings of the produced 3D assembly to make the spatial relationships between tiles and cames visually explicit. For two examples, we also provide turntable-style videos (Figure 85 and Figure 86) to support continuous inspection of the generated structure. A third example demonstrates the pipeline on a canonical "flat-colour" artwork (Mondrian-style), and includes a comparative video illustrating how different parameter settings change the perceived outcome (Figure 87). Finally, Figure 88 provides an additional static view pair for closer inspection of one result.

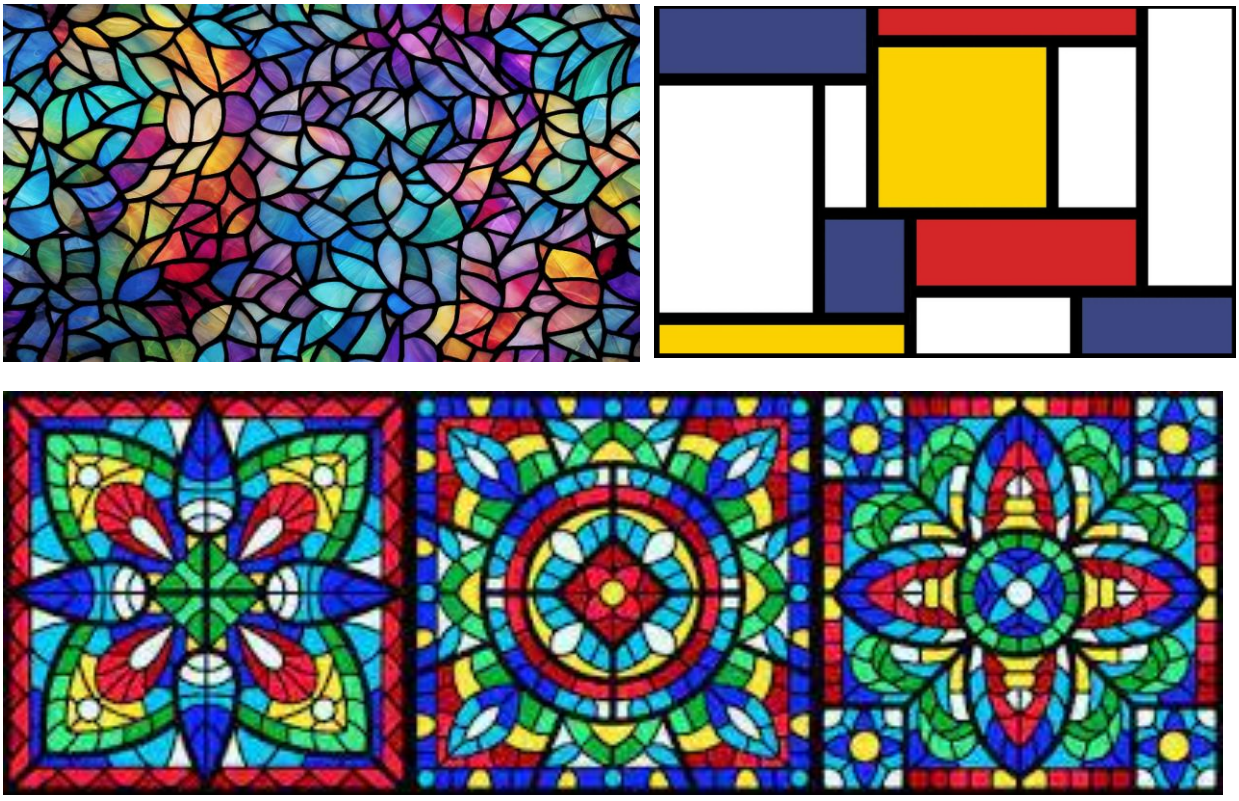


Figure 84. Source images used in the results. Three input images were used to evaluate the presented method. Two images are shown in the top row (A left, B right). The bottom row contains a single elongated image (C) with three motifs. Images A-C are the sole inputs; subsequent figures visualise the corresponding generated 3D assemblies.

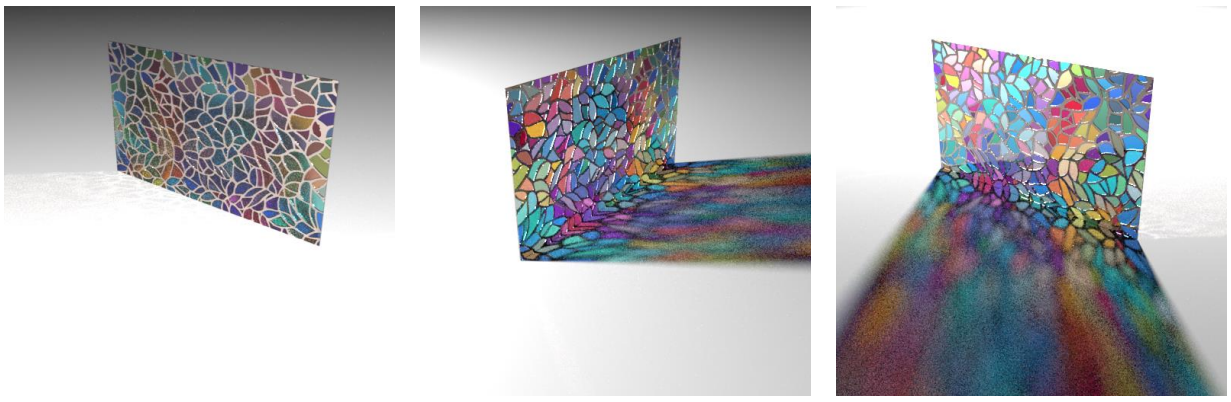


Figure 85. Generated stained-glass assembly for Source Image A. Three viewpoints of the resulting 3D composition, rendered to expose the tile layout and came continuity cues introduced by the 3D modelling. Video: https://youtu.be/wAmowQ6GE4U.

The example below (Figure 86) shows a "walk-through" by a simulated stained glass wall.

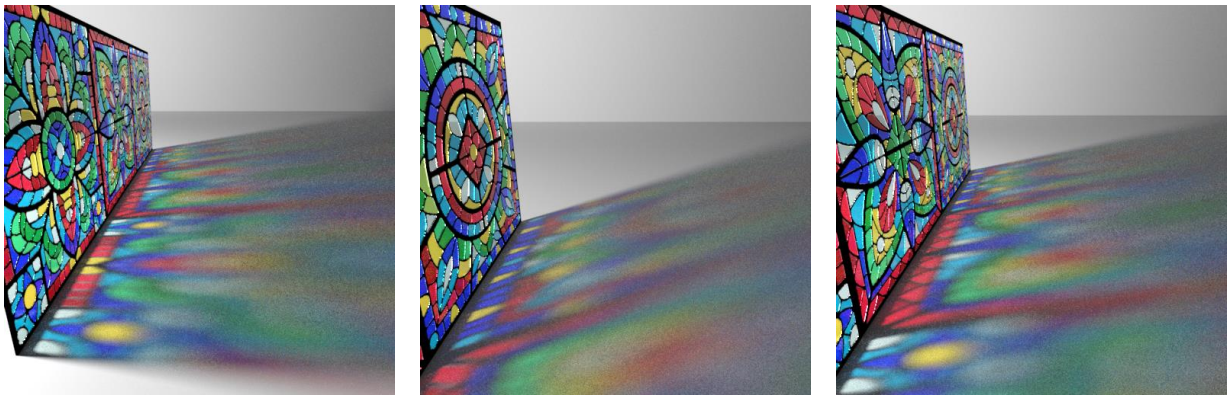
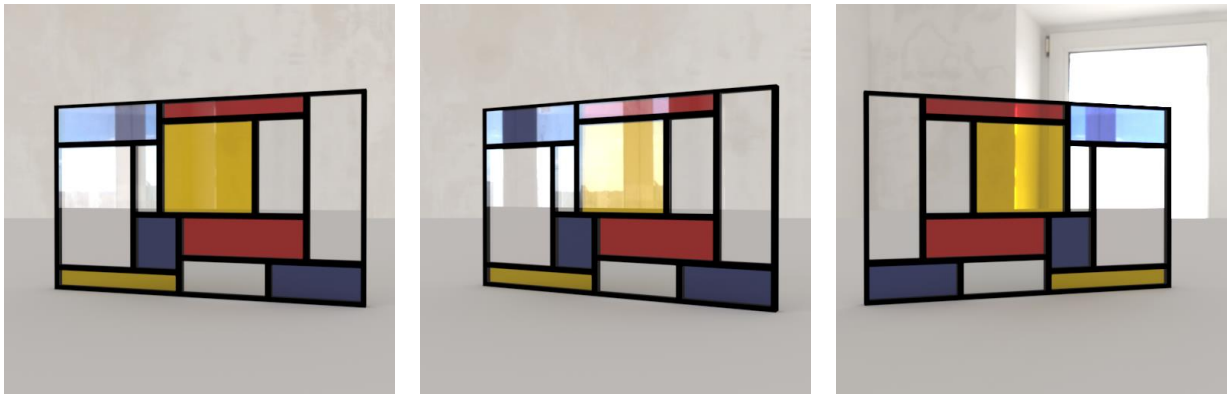


Figure 86. Generated stained-glass assembly for Source Image C. Three viewpoints of the resulting 3D composition for a second input, demonstrating robust piece extraction and consistent came formation on a different design. Video: https://youtu.be/YQ4UHi5apfc

The next example (Figure 87) shows the same structure but using different glass types (clear and frosted). We notice the realistic behaviour of the glass interface simulation, showing the difference between transparency and translucency. In the top row, the clear glass interface exhibits transparency as well as reflections of the opposite side of the windowed room. In contrast, in the bottom rows, we see that the translucency of frosted (rough) glass diffuses the light.



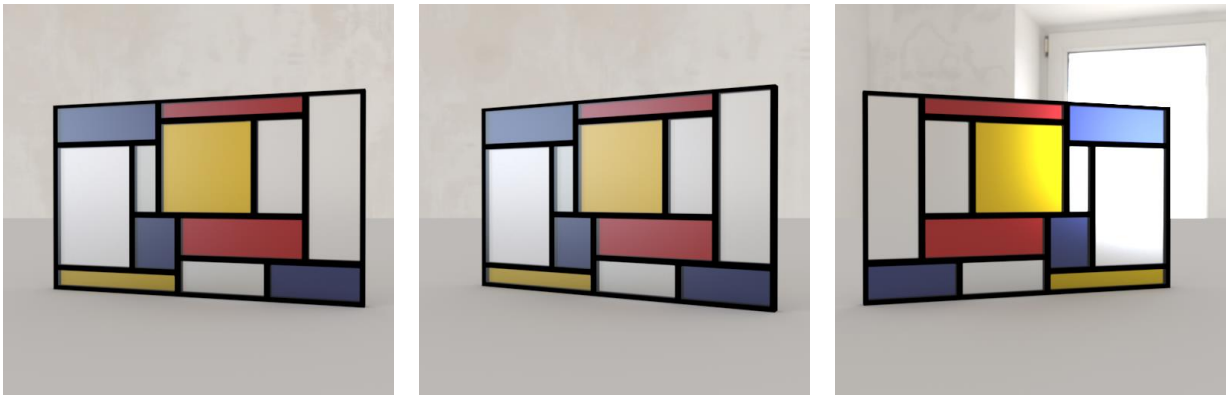


Figure 87. Mondrian-style example, illustrating the difference between transparency and translucency. Two sets of multi-view renderings of the same source artwork produced under different parameterisations, simulating clear and "frosted" (rough) glass. The comparison highlights how modelling parameters influence the visual balance between coloured tiles and the metallic framework. Comparative video: https://youtu.be/AfUsOCcTDKA.

The next example demonstrates the capacity for individualised previews. The stained glass composition is instantiated with a thick copped framework. Moreover, it is installed as a window in a virtual room with an HDRI simulated environment outside it (a field with grass). This shows the capacity of prevailing compositions for a particular place.

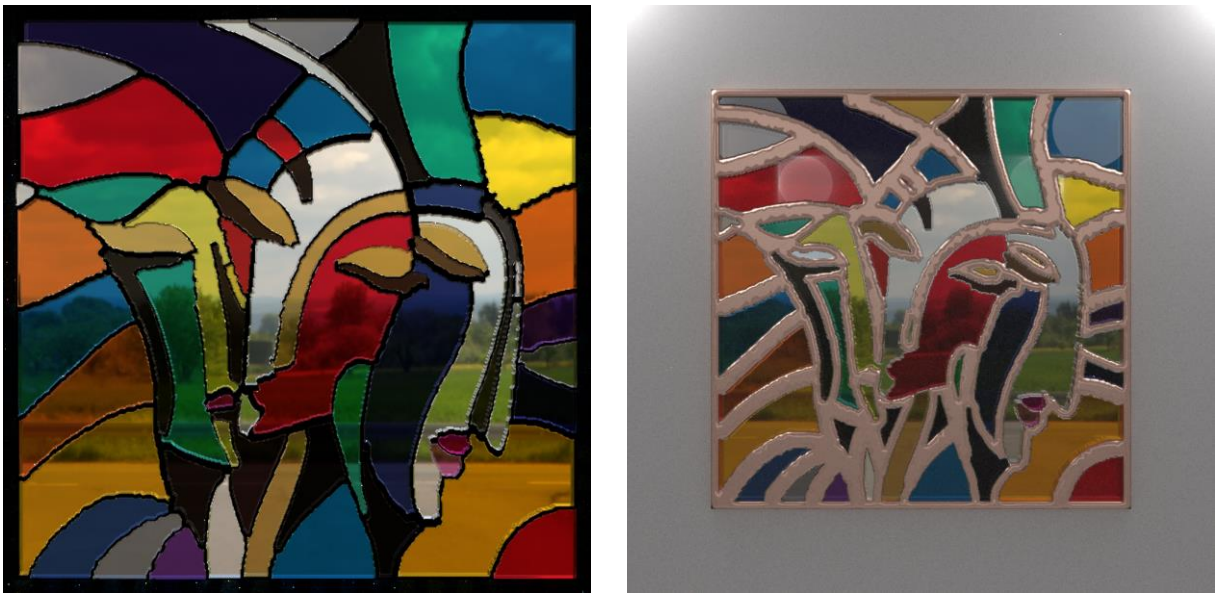


Figure 88. Inspection views. Two views of a generated stained-glass assembly for in situ visualisation of a transparent composition, allowing the outside landscape to be visible.Visualisation with black plastic (left) and copper (right), as a window allowing view to the landscape outside.

Using this method, we can create predictions of how light will interact with the installation throughout the day. Figure 89 shows the projected luminous patterns on the floor as the direction moves throughout a summer day. The artwork used in this simulation is down on the top row of the figure. The corresponding video shows a "walk-through" of the installation. Notice that, given the geographical orientation of the room, the direction of sunlight can be predicted.

Figure 89. Demonstration of stained glass installation. Video: https://youtube.com/shorts/KqolDghB1G4

10.8 Discussion

In conclusion, the key result of this section is that a stained-glass composition can be synthesised directly from an arbitrary raster source (photo, sketch, or painting), eliminating the time-consuming manual authoring of a lead-line stencil that conventional tools typically require, while still aiming for geometric regularity and craft feasibility.

The pipeline explicitly addresses manufacturability constraints by (i) creating consistent inter-piece gaps for the metallic skeleton via morphology-driven boundary refinement, and (ii) providing quality testing for fragile intrusions that would be prone to breakage during cutting and assembly, together with a corrective remedy strategy.

Finally, by extruding each segmented region into a watertight solid with a user-defined thickness (τ) and combining it with a corresponding framework model, the system supports photorealistic rendering and light-transport-driven previews of installed appearance, including scenarios that depend on illumination conditions.

11 Plaster Turning

This paper presents the design and evaluation of an interactive simulator dedicated to the craft of plaster turning for porcelain slip-casting. Unlike existing virtual pottery systems that focus on clay deformation, this tool models the subtractive shaping of rigid plaster blanks, a critical intermediate step in industrial ceramic production. Developed through user-centred design with expert practitioners, the simulator replicates the specific workflow of the workshop, from preparing the blank to the geometric constraints of the turning wheel. We describe the evolution of the system across five prototypes, highlighting how expert feedback drove the transition from generic sculpting metaphors to a high-fidelity representation of workshop practice, including correct hand positioning, rotation-dependent turning, and authentic preparatory rituals. The results demonstrate that the system not only supports the acquisition of tacit procedural knowledge but also generates geometric data compatible with physically-based rendering workflows. This work contributes to the digital preservation of intangible cultural heritage by making the materiality and rationale of porcelain manufacture accessible in a virtual environment.

Related work on virtual pottery demonstrates multiple viable interaction paradigms, that is, physical wheel coupling, gesture-driven shaping, and motion-sensing installations, together with efficient geometric deformation strategies [148, 149, 150, 151, 152]. Our “Plaster Throwing” simulator differs in objective and reference frame: we prioritise controlled, measurable geometric transformation of a plaster master (with direct comparability to digitised states) rather than attempting to model clay rheology for expressive free-form outcomes.

11.1 Introduction

Plaster turning for porcelain slip casting spans a wide range of operations. A designer draws the target form. A practitioner produces a master model made of plaster. The finished master is then used to cast a (negative) mould, where liquid porcelain is poured into. The resultant cast is dried, fired twice, and glazed. The quality and reproducibility of the produced artefacts are directly determined by the accuracy with which the plaster master has been shaped, or, otherwise, the practitioner's turning skill.

The quality and reproducibility of the resulting artefacts are therefore sensitive to the geometric accuracy of the plaster master, and hence to the practitioner’s skill. Figure 90summarises this workflow and locates the plaster-master preparation and making stages addressed by our simulator, and shows their positioning within the overall porcelain artefact production process. Each step is accompanied by one or two images from our ethnography.

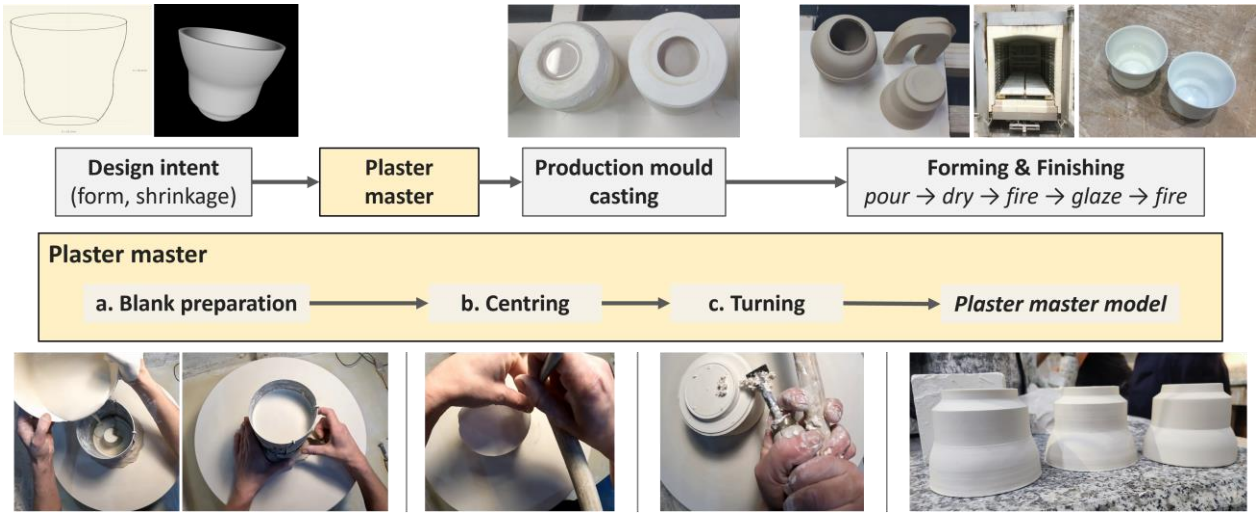


Figure 90. Porcelain slip-casting workflow and scope of this work. A designer specifies the intended target geometry. A practitioner produces a plaster master model by (a) preparing a blank, (b) centring it to establish the true axis of rotation, and (c) turning it on the rotating plaster to achieve the intended shape. The finished master is then used to cast a negative production mould, into which porcelain slip is poured, dried, trimmed, fired, and glazed to produce the final artefact. The plaster master stage is highlighted and analytically illustrated to indicate the stages addressed and simulated in this deliverable.

Modern digital-twin concepts include dynamic features and functional modelling of cultural objects and activities. In crafts and making, a representative model should convey the object geometry, the practitioner's motion, but also the relations between the tools, the practitioner's actions, and the intended result.

Within this context, the simulation of crafting actions as a digital twin serves three goals. First, to organise crafting knowledge in an action-centric fashion, which included the workspace stimuli and affordances available to the practitioner [164]. Second, the first-person and active perspective is more intuitive and informative than the conventional, third-person documentary views. Besides, virtual interaction provides an environment in which learners can safely err and endlessly rehearse crafting actions with reduced material consumption and workshop usage. Third, computer-aided design is an established way to reduce design costs through the digital preview of designs; the challenge here is to realistically account for the influence of the crafting process in the rendering of the design.

Plaster turning is a suitable craft instance to study digital craft representations because it combines structural tractability with rich variability. The shaping of plaster models involves coordinated perception and action to monitor the evolving geometry, process sensory cues, and use this feedback to adjust tool actuation and turning speed [165, 288, 167]. The sensory cues are generated by the monitored interaction between tool and material, and are perceived either directly or mediated through the tool.

At the same time, plaster turning offers a way to study the interplay of risk and certainty in making [48]. In freehand operations, quality is not determined by a machine or a fixed procedure, but depends on ongoing judgement while the work is underway. The process includes ways that certainty is `manufactured' by risk mitigation schemes. The simulator enables the understanding of a crafting process as the `interpretation' of an intended design into a working plan (or `umbrella plan' [168]) realised through situated micro-decisions, influenced by the progress and material response to the preceding decisions

and the actions currently in hand. This way, a simulator exposes not only how experts move, but also how experts think.

The cues generated at the tool--material interface include visual, haptic, auditory, and proprioceptive sensations that, interpreted, enable the monitoring of the evolving geometry of the tool-plaster interface. This real-time monitoring and understanding of the working space enables the practitioner to adjust actuation by modulating tool incidence posture and wheel turning speed. Figure 91 visualises the sensory feedback loop during plaster turning.

Figure 91. A practitioner engaged in plaster turning (left). The diagram (right) illustrates the continuous feedback loop, where the maker modulates tool actuation in real-time response to sensory cues.

The simulator was developed iteratively with an expert practitioner to address the following objectives:

1. To capture and document the skills and knowledge involved in plaster turning for porcelain slip casting, transforming an intangible craft skill into measurable, archivable, and inspectable data. This includes audiovisual documentation, process analysis, and alignment of simulated states with the physical workflow.
2. To create a tool for teaching and rehearsing freehand turning that enables practitioners and learners to explore turning strategies, observe geometric consequences, and discuss tool use in a shared visual environment, including remote or hybrid settings where access to workshops is limited.
3. While computationally tractable on commodity hardware, to serve as a representation of plaster turning that is realistic enough to prepare learners for subsequent engagement with the physical process.

In pursuing these objectives, this work makes three contributions. First, an ethnographic analysis of plaster turning for porcelain slip casting, including terminology and workflow. Second, a user-centric design and implementation of an interactive simulator of plaster turning. Third, a way to integrate expert insight and feedback into a digital experience that goes beyond cinematographically presenting the kinetic and physical dimensions of practitioner actions, but also conveys the knowledge and affordances that intuitively explain the actions of the practitioner.

11.2 Ethnographic Interaction Design

We first present an overview of our ethnography of the plaster turning process. We then identify the critical and characteristic elements of the plaster turning tasks and discuss their digital representation.

11.2.1 Ethnography

Fieldwork was conducted at the Ceramic Workshop of ENSAD Limoges, France, between January 2024 and November 2025. The focus was on producing plaster master models for porcelain slip-casting moulds. Crafting was carried out by an expert plaster turner with over twenty years of experience in this role. A follow-up conversation and semi-structured interview were conducted to clarify terminology, decision points, and workshop-specific conventions. All recordings were made with the practitioner's informed consent. The practitioner is a co-author of this work.

To capture the use of handheld tools, an egocentric camera was mounted on the practitioner's head. This was a GoPro Hero 11 device, recording at 1920 × 1080 pixels and 30 frames per second. The egocentric video was segmented into episodes corresponding to recurring actions. Representative frames from these episodes were exported as still images and annotated with action labels, tool use, and relevant parameters. Still photographs of key stages were acquired using a conventional digital camera. Indicative video segments from this session are provided in the supplementary material and at the repository of the Craeft project.

11.5.1 Terminology

The ethnography informed our understanding of the crafting process and equipped us with terminology. The video annotations informed the analysis and the structuring of the simulator into a preparation (A) and a turning (B) scene. Table 18 summarises the derived terminology for real practice. Within the virtual environment, the virtual representation of the plaster workpiece is referred to as the virtual workpiece, and correspondingly for the rest of the physical entities and simulated actions, e.g. turning the wheel, etc.

Table 18. Terminology of physical craft objects and simulation elements.

Term	Definition
Formwork	A temporary retaining structure made of metal or PVC is used to contain and shape liquid plaster while it sets to become an initial blank. Typically, it has a cylindrical shape.
Blank	The initial cylindrical cast of plaster was produced by pouring liquid plaster into a formwork. It comprises the initial state of the workpiece that will be turned on the wheel.
Workpiece	The plaster body that was worked on by the practitioner.
Master Model	The turned plaster was positive, embodying the target geometry of the piece, including allowances for drying and firing. It is used to produce moulds for production.
Production Mould	The plaster mould cast from the plaster master model is used to shape liquid porcelain slip into repeatable porcelain pieces.
Porcelain Artefact	The ceramic product is made by casting porcelain slip into a production mould, then drying, and firing twice at two different temperatures.

Using this vocabulary, we read the sequence of photographs in Figure 90 as follows.

The practitioner pours liquid plaster into a *formwork* to create a plaster *blank*. When the blank dries, the formwork is removed, becoming the *workpiece* in its initial state. The practitioner marks the centre of rotation and turns the rotating piece. The practitioner cuts plaster to achieve the intended shape. The resultant plaster body is the *master model*. The master model is used to cast a negative *production mould*. Porcelain slip is poured into this mould. The cast porcelain body is dried, trimmed, fired, and glazed to produce the *porcelain artefact*.

11.2.1.2 Centring

The blank diameter and height are chosen with foresight of the subsequent material removal during turning and the succeeding porcelain shrinkage during firing, so that the porcelain artefact meets the intended dimensions. In pottery, the plastic clay body is pushed into alignment; however, in this case, the plaster blank is fixed upon the spinning wheel. Alignment is, therefore, achieved by removing material and cutting the plaster blank until its symmetry axis coincides with the rotational axis.

To minimise the risk of inaccurate cutting, practitioners mark a circle where the correct cut is expected beforehand. Using the wheel's spin, a circle is drawn by holding a pencil still at a constant radius, securely braced upon the rest by the practitioner's fingers. This circle is, by construction, centred at the centre of revolution. The practitioner repeats the drawing at successively smaller radii that converge towards the centre of rotation (`true centre'), which typically differs from the geometric centre due to formwork irregularities and placement errors. Practitioners inspect the estimated centre with a compass to verify that it is equidistant from the drawn circle, that is, it is `true', and can thus be safely used for subsequent operations (Figure 92).

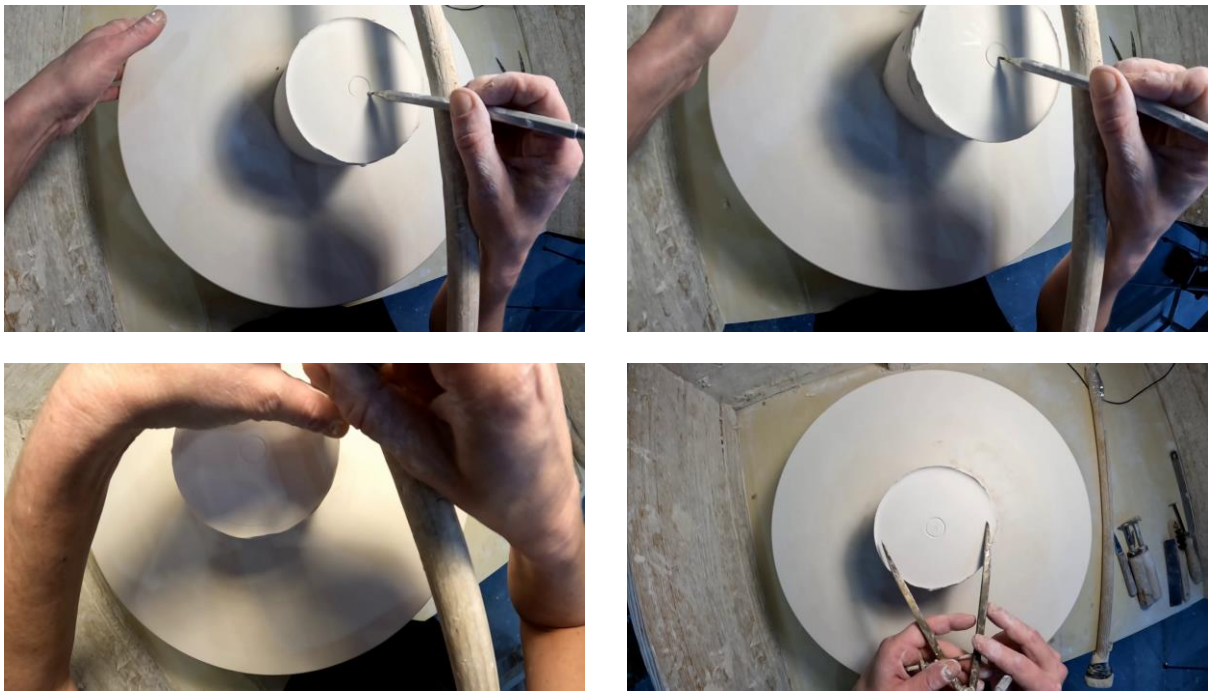


Figure 92. Centring workflow. Top: Locating the rotational centre on the spinning blank. Concentric circular neighbourhoods are marked (left) to close in and accurately pinpoint the rotational centre (right). Bottom: An outer circle is inscribed (left) to guide plaster subtraction and provide a rotationally symmetric, centred blank, and visually verified using a compass (right).

Figure 93. Egocentric views demonstrating the use of handheld turning tools, cutting the plaster.

The actions are identified in the table below.

Schema	The actor aligns the material through iterative fine observations and adjustments in a feedback loop until convergence to a symmetrical body. After each adjustment, the alignment is assessed through observation and readjusted until the body is running 'true'.
Tools	A rigid handheld 'guide' facilitates precise observation. It is planar and shaped like a ruler. It is designed to contact the surface of the spinning plaster gently, without deforming it.
Perceptual task	The block is rotated slowly to check its alignment. The discrepancy to be adjusted is fine. An auxiliary tool facilitates precise observation. The practitioner brings a scribing tool into contact with the spinning cylinder's surface. The tool in contact amplifies minute surface anomalies into noticeable vibratory-like tool movements. If the block is off-centre, the tool undergoes a significant wobble.
Manual task	This is a delicate process of physical adjustment. The actor stops the wheel and gently taps the side of the block with a wooden mallet or presses it firmly by hand to shift its position slightly on the wheel head. If wet clay is used to secure the block, the practitioner uses firm, controlled pressure to nudge the block into the correct position, re-sealing the clay base to hold the new position.

Only once the plaster block is running true is the actual turning process initiated. The first cut with the turning tool is often made immediately to create a perfectly flat top surface, which serves as a level reference point for all subsequent measurements. The difference between a wobbly piece and a perfectly symmetrical piece begins with this initial centring step.

11.2.1.3 Turning

Material is removed from the rotating cylinder to produce the designed master model. The practitioner cuts, observes the evolving shape, and adjusts tool motion or wheel speed in a feedback loop. Freehand and template-based methods are used in plaster turning.

In freehand cutting, the practitioner shapes the rotating plaster directly using a handled turning tool (see Figure 94). As the wheel turns, the cutter comes into contact with the surface, removing material in a highly controlled manner. This includes continuously adjusting the tool position and pressure based on sensory assessment and internal expectations. Freehand methods offer greater flexibility than template-based methods, enabling practitioners to achieve expressive variations by dynamically adjusting wall thickness and curvature.

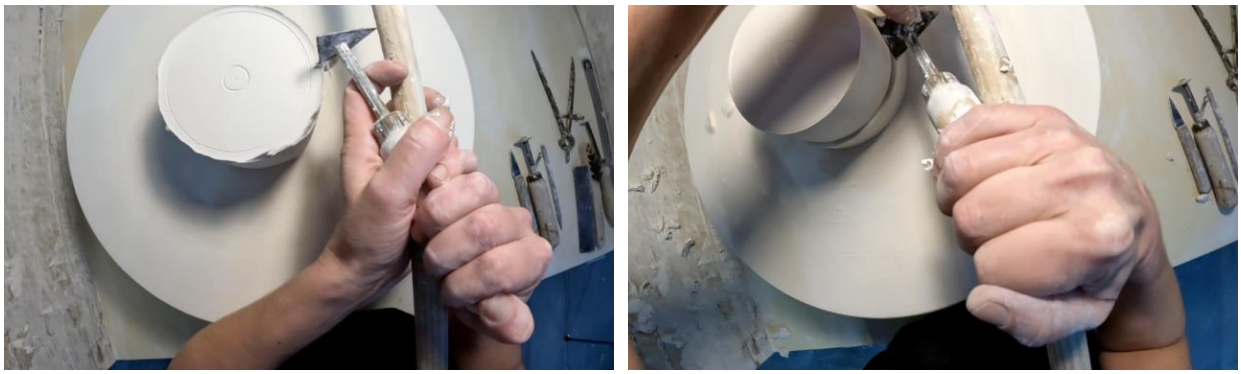


Figure 94. Egocentric views demonstrating the use of handheld turning tools, cutting the plaster workpiece.

The turning tools are short and light. For precision cutting, practitioners lean against the wooden rest and hold it firmly in both hands, braced against the body for stability (Figure 95). This shifts their weight to the rest, which also absorbs most of the forces generated at the cutting edge. The practitioner grips the tool and braces it against the rest, using it as a stable fulcrum. The rest affords stability because it alleviates the influence of weight and vibrations. Practitioners use their fingers only to control the tool and not to support their weight. This affords precise modulation of the tool, enabling the inscription of intricate forms.

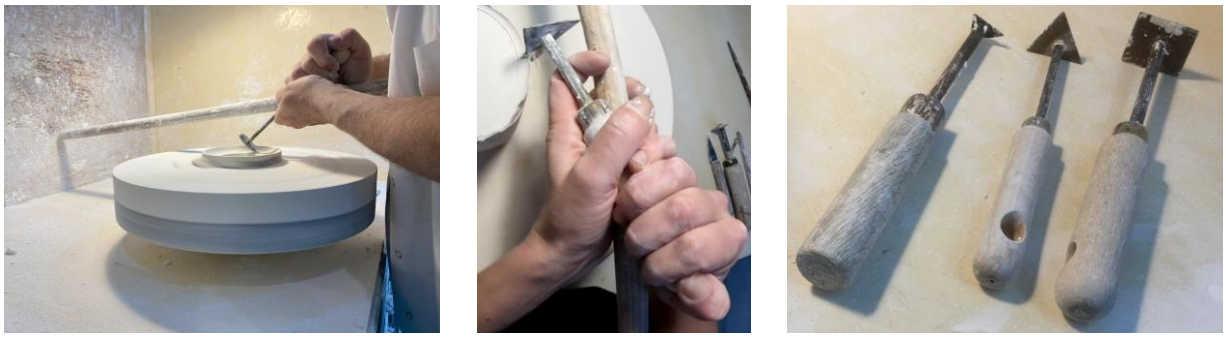


Figure 95. Ergonomics and tool usage. Left, centre: The tool is braced against the rest to separate support from guidance, allowing the hands to focus on steering. Right: a set of turning tools.

Template-based methods are used in industrial production. A rigid metallic template fixed against the spinning blank acts as a profile scraper. This work regards freehand methods, as they support explorations of design possibilities and, as works of risk, warrant the most practice. Nevertheless, similar shaping principles are found in alignment verification and finishing tasks (Figure 96).

Figure 96. Egocentric views from straightness and dimensional verification (left), planing with a handheld metal strip (centre), and surface finishing with an abrasive pad (right).

Figure 97. Egocentric views from straightness and dimensional verification (left), planing with a handheld metal strip (centre), and surface finishing with an abrasive pad (right).

Schema	There are two primary methods for shaping the plaster. The Template method is the factory standard for high accuracy and mass repeatability. A rigid object template is created based on the design. This template represents the object's profile and is fixed next to the spinning plaster cylinder. The stationary template acts like a knife, shaving the spinning plaster until the block's profile exactly matches the template. Artists favour the Freehand method as it increases shaping possibilities and allows for intuitive sculpting. This approach enables the creation of subtle, fluid, or expressive curves that are difficult to achieve with a rigid template.
Tools	The artist uses tools to cut the plaster, which is tough but brittle. Griffon hooks are strong rods with a sharp, curved hook or loop at the end, used for the rapid removal of bulk plaster. The hook allows for controlled, deep cuts that produce smooth, continuous curved lines. Chisels are used for finishing and precision cutting. During the freehand process, cutting actions are stabilised by using a rigid rest. This stability is crucial for coordinating visual monitoring and precision control.
Perceptual task	The artist uses visual monitoring and haptic feedback (via the feel of the tool) to guide the cut. They constantly look at the spinning surface, checking for symmetry and anticipating where the next cut will be made. By resting the tool on the spinning plaster, they monitor the resistance and texture to judge the depth and angle of the cut, ensuring the form develops smoothly.
Manual task	The artist modulates the pressure applied to the tool during the action. This pressure is precisely controlled, as too little pressure fails to cut, while too much can cause the brittle plaster to chip or crack. The tool's angle and horizontal position are also modulated to

dictate the curve being sculpted, requiring coordinated visual monitoring and hand control.

11.2.1.4 Relation to pottery

Pottery and plaster turning both begin with a bulk workpiece mounted on a rotating wheel, and both require *centring* before any shaping operation can succeed. If the workpiece is not centred about the rotation axis, it wobbles, periodically "runs into" the tool or hand, and quickly becomes unworkable.

During early prototyping, we were initially misled by pottery-oriented descriptions of wheel work, where *centring is performed by hand* through plastic deformation of the clay body. This is a common misconception when transferring knowledge between wheel-based crafts: the shared tool (the wheel) suggests a shared centring technique, but the material regime is fundamentally different.

In **clay pottery**, the workpiece is soft and continuously deformable (Figure 92, left and middle). The potter centres by applying pressure: one hand stabilises the side while the other compresses from above, forming the clay mass into alignment with the rotation axis. Importantly, this is not only an actuation strategy; it is also a *sensory* one. As documentary frames show, the stabilising hand functions as a tactile probe: if the clay body is off-centre, the hand feels a periodic lateral oscillation ("knock") as the wobble passes under the palm. Centring is therefore an active, closed-loop process: the potter continuously applies corrective forces while simultaneously reading the residual wobble through touch. By contrast, in **the porcelain mould-manufacturing workflow**, centring is performed on a **rigid plaster blank** that will serve as a precision master for mould creation. The blank cannot be "pushed into place" as clay. Practitioners locate the true rotational centre by drawing concentric circles while the wheel spins, then inscribe a target circle and remove material exactly up to this pre-scribed boundary, producing a rotationally symmetric blank whose symmetry axis coincides with the wheel axis. In that context, centring is a geometric, subtractive operation: accuracy comes from reference marks and controlled removal, not from redistributing material through hand pressure.

As reported in the evaluation below, this was a point of initial confusion, due to the stereotype brought from pottery, where centring is manual.

Figure 98. Centring in clay pottery versus plaster turning. Stills from a clay pottery documentary illustrate hand-based centring through pushing of the plastic clay body, where the stabilising hand provides feedback (left, middle). Rejected prototype implementing pottery-style centring (right). The prototype modelled centring as a hand-driven force interaction that attempts to push the rotating workpiece into alignment.

The lesson was valuable in two ways. First, it clarified the centring operation to be modelled and the downstream requirements. Second, the technical component developed for the initial pottery-inspired centring remains useful: it will be repurposed for pottery demonstrators.

11.2.2 Interaction design

We extend our observation beyond kinematics to better understand the rationale of practitioner actions within a crafting process. The practitioner operates under an umbrella plan, which is a goal-oriented overview that remains open to emergent contingencies [168]. In contrast to 'mentally-rehearsed' short-span action plans [169], this plan is overarching and realised through a succession of short-span actions [168]. The umbrella plan manages the uncertainties stemming from the environment and outcomes of crafting actions [48]. Because the volume of potential contingencies precludes exhaustive mental rehearsal, the artisan utilises the umbrella plan to maintain strategic continuity while delegating tactical adjustments to the sensory-driven perception-action cycle.

The perceptual information fuelling this cycle is multimodal and active [165, 166, 167]. Practitioners focus their attention on the sensory feedback generated by the cutting of the spinning plaster to extract the information it contains. Visually, the practitioner discerns transient shifts of the rotating profile to detect asymmetries, as they signal surface irregularities. Audio provides feedback on the quality of the cut; a clean, continuous sound indicates cutting stability, whereas rhythmic or harsh noises signal tool chatter or misalignment. Haptic resistance and vibration reveal surface coarseness or texture difficult to perceive visually. The attended sensory inputs are compared against `mental sensory images' [168], or the anticipated sensations from the intended forms and qualities. When felt stimuli differ from those anticipated, practitioners correct errors to converge back to their plan [170]. Experienced practitioners use this feedback, and tutors transmit this knowledge by bringing it to the attention of apprentices.

Practitioner actions are conceived within the range of risk and certainty [48]. Centring measurements and supporting structures are constraining `certainties' that reduce the risk of centring failure, thereby lending greater certainty to the actions executed. In freehand cutting, the outcome is continually at risk, a vulnerability intensified by the irreversible nature of the subtractive process. The armrest exemplifies how practitioners balance freehand risk with tooled certainty even within individual actions. Bracing the cutting tool against it limits the degrees of freedom and constrains the uncertainty against vibrations and hand tremor. Correspondingly, interaction design is justified in supporting relatively stable tool positioning over short time intervals.

These observations guide us to conceptualise the simulator not as imposing a static plan but as a dynamically coupled system, comprising the maker and the material [171]. For this reason, the system has no prefabricated `template' to follow, as often found in educational applications. Interaction supports the interpretation of designs into a plan of situated actions, where the final form integrates the intended form, the crafting method, the properties of the materials used, and the care, judgment, and dexterity by which actions were carried out [48].

11.5 System Architecture & Implementation

Traditionally, the creation of a porcelain artefact by the slip-casting method begins with a plaster model. This model is the 'intermediary' that enables repeatability. The use of plaster is key because it absorbs water from the liquid porcelain.

This process has three main stages:

1. Master Prototyping: A model of the final object is sculpted in plaster.
2. Working Mould: This model is used to cast plaster moulds that copy its form.
3. Slip-Casting Liquid porcelain (slip) is poured into the mould.

Additional treatment and firing are required for glazed coatings.

Our focus is on the first stage: shaping the plaster prototype, known as Plaster Turning. This subsection outlines the design rationale and core system requirements derived from the plaster-turning workflow. We describe how these requirements are realised in Unity in a virtual workspace, and detail the design of the two actions represented in the simulator.

11.5.1 Requirements and Motivating Rationale

Three factors shaped the design of the simulator. First, there was a need for a workflow representation that explains the spatial and functional dependencies involved in creating master models for porcelain slip-casting. Second, the design had to respect the computational limitations of real-time simulation on commodity hardware. Third, the simulator had to provide a coherent learning experience that reflects workshop practice, trains users to notice and interpret helpful sensory cues, and intuitively presents the semantics and dynamics of crafting elements.

In response to these challenging considerations, the simulator features a modular, scene-based architecture that organises the virtual process into two phases:

- Scene A (Centring): The mould is placed on the wheel, liquid plaster is poured and allowed to set, resulting in a cylindrical plaster blank.
- Scene B (Turning): The user works with rotating cylindrical blanks using tools that cut the revolving solid to its desired shape.

This division mirrors the slip-casting workflow, where a plaster blank is first produced and then refined. It preserves the interdependence of the crafting actions and supports knowledge transfer between phases.

User interaction begins in Scene A by selecting the handedness settings and plaster blank type. These selections prescribe the configuration of the subsequent scene. The initial state of Scene~B comprises a blank of corresponding shape and dimensions, as well as wheel direction and an armrest placement compatible with the user's handedness.

At runtime, the simulator executes a per-frame loop that links user input, rotational motion, tool--material interaction, and mesh deformation. The system maintains a set of state variables that are updated every

frame, including wheel speed, tool pose and activation state, as well as the geometry of the virtual workpiece. Audiovisual feedback is updated in step with this loop. All outputs are derived from the same data structure to keep visual, numerical, and auditory feedback synchronised.

During turning, a rotation controller and a mesh-generation module manage the core data structure. The controller updates wheel angle according to user input, and the mesh-generation module reads tool state to update the solid-of-revolution representation. Occupancy changes trigger surface regeneration and update, as needed, per frame. The activity log contains the sequence of user actions and their effect on the workpiece shape.

Figure 99 summarises the states encountered by the user in the simulation and indicates the data flow between scenes. On the left, shown is the configuration sequence that initialises the workspace and material according to user handedness and session selection. On the right is a sequence of states encountered in the interactive use of the workshop. In turning, rotational velocity is user-controlled and drives tool impact. The user approaches the turning plaster, makes a cut, and observes the tool manipulation and interaction in real-time. Additionally, the UI can be used to read, inspect, and measure the shaping outcome.

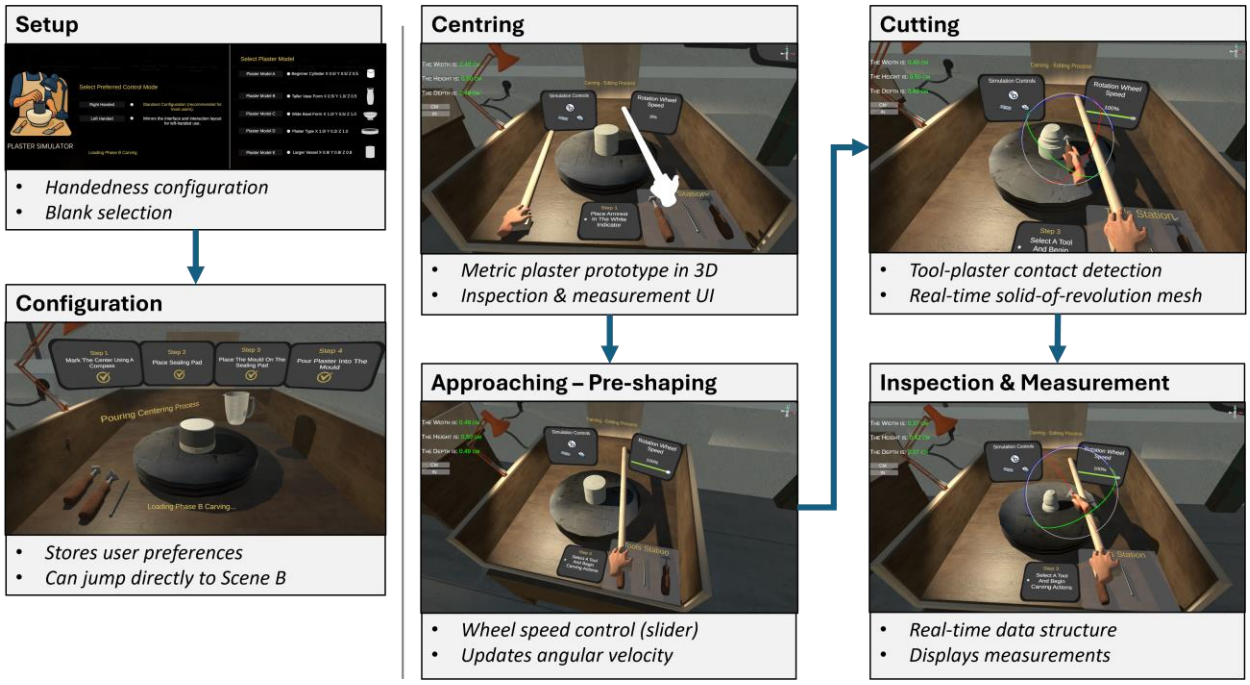


Figure 99. System data-flow pipeline.

11.5.2 Implementation

The simulator was developed using the Unity game engine (version 2022 LTS), chosen for its real-time rendering performance and extensible C\# framework and editor environment. This enables access to mesh manipulation, tool interaction, and data persistence. Its physics engine provides a tool-material contact detection via collision events.

This platform allows the same virtual environment to be deployed as a 3D desktop application, a browser-based experience, or a head-mounted VR installation. In this work, participants interacted with the desktop 3D configuration. In all cases, the simulator provides a first-person view aligned with the practitioner's viewpoint.

The conventional UI (menus, preferences, and settings panels) was designed in Figma and imported via the official Figma-to-Unity bridge, allowing pixel-accurate reproduction of the layout. The 3D models and textures of the cutting tools and virtual furniture assets were modelled in Autodesk Maya (version 2024) to support high-quality geometry and shading.

Geometric Representation

The central data structure in the system is a volumetric representation of the virtual workpiece. It is encoded as a planar binary occupancy field T, which is conceptually revolved about its vertical axis. From this representation, a surface mesh is derived for appearance rendering, equipped with surface normals for correct reflectance. The two representations are coupled and kept synchronised to support real-time interaction. The implementation of this data structure is based on the toolbox presented in Section 6; it is specifically described from Section 6.2.1 through 6.2.3.

Mapping Tool Interaction to Geometry

In the virtual world, 3D tools are approximated by spheres (or small clusters of spheres) to enable efficient collision detection, as described in Section 6. The established geometric mapping projects user interactions in 3D space onto the 2D occupancy field. The specifics for tool interaction and 3D mesh update are provided in Sections 6.2.4 and 6.2.5, respectively.

11.5.3 Staging Virtual Actions

The simulation is organised into sequential scenes that mirror the workflow of plaster turning. Each scene isolates a coherent craft action and provides a digital counterpart of the practitioner's physical operations. All scenes are presented from the practitioner's viewpoint.

Lighting is provided by a single directional source, mimicking the practitioner's deliberate use of a dominant lamp. In essence, this constitutes a raking-light configuration: illumination arrives at an oblique angle, enhancing visual cues from surface texture, as even slightly raised loci cast elongated shadows that are easier to discern. The simulator adopts this configuration to train the practitioner's reading (interpretation) of these diagnostic shadows.

All scenes are staged from a practitioner-like viewpoint and use workshop-informed viewing conditions to support the perceptual demands of plaster turning. Illumination is provided by a single dominant directional source arranged as **raking light** (oblique incidence), so that slight deviations in the surface produce elongated, diagnostic shadows that are easier to read during symmetry and smoothness assessment. Live measurements are displayed in-scene, and tool pose is made explicit via a local red–green–blue orthogonal frame aligned with the tool’s coordinate frame, reducing ambiguity during freehand cutting and fine adjustment (see Figure 100).

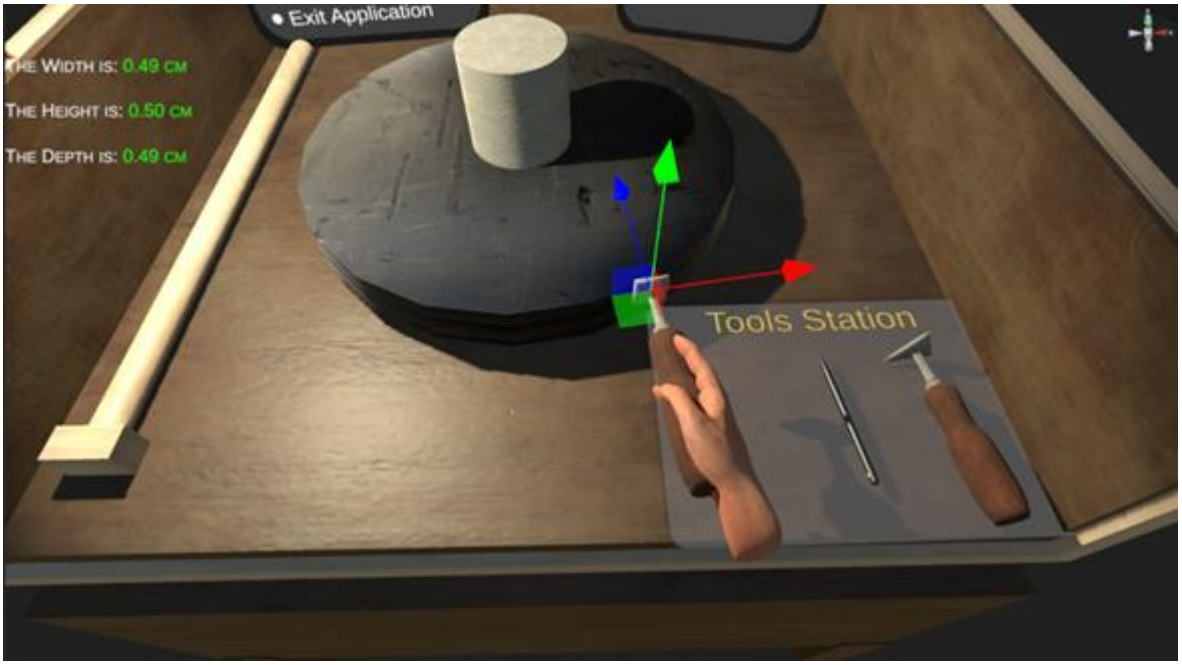
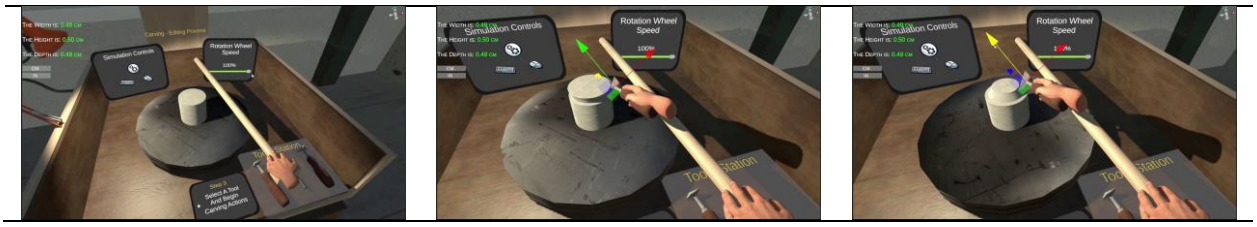


Figure 100. Virtual workspace overview for plaster turning. The scene shows the scene lighting and an explicit local coordinate GUI component for placement and manipulation input in the virtual world. Tool rest placement, light direction, and wheel spin are adjusted for handedness.

Each scene is presented from the practitioner's viewpoint and uses workshop-informed visual conditions to support the sensory input available for judgment. Light is provided by a single directional source, arranged as in a raking-light configuration: light arrives at an oblique angle, enhancing surface relief cues, so that even subtle raised loci cast elongated shadows that are easier to discern when assessing symmetry and smoothness. To aid tool manipulation, a superimposed tool-orientation GUI input component is optionally available. The resulting geometry is presented in a stable inspection view and can be exported with an optional scale adjustment to account for shrinkage allowances, producing metric, 3D-printable models suitable for downstream production and rendering workflows.

Figure 101 shows interaction in the virtual workspace. Two turning tools rest on the table, and one is gripped by a virtual hand. Tool orientation aids are superimposed as a local red--green--blue (RGB) coordinate frame that moves with the tool's local coordinate system. Live measurements are displayed, and a blank is placed on the turning wheel. In the figure, the top row shows the workspace setup and the tool pose adjustment configuration. The middle and bottom rows show successive frames from interaction with a coarse and fine cutting tool, respectively.



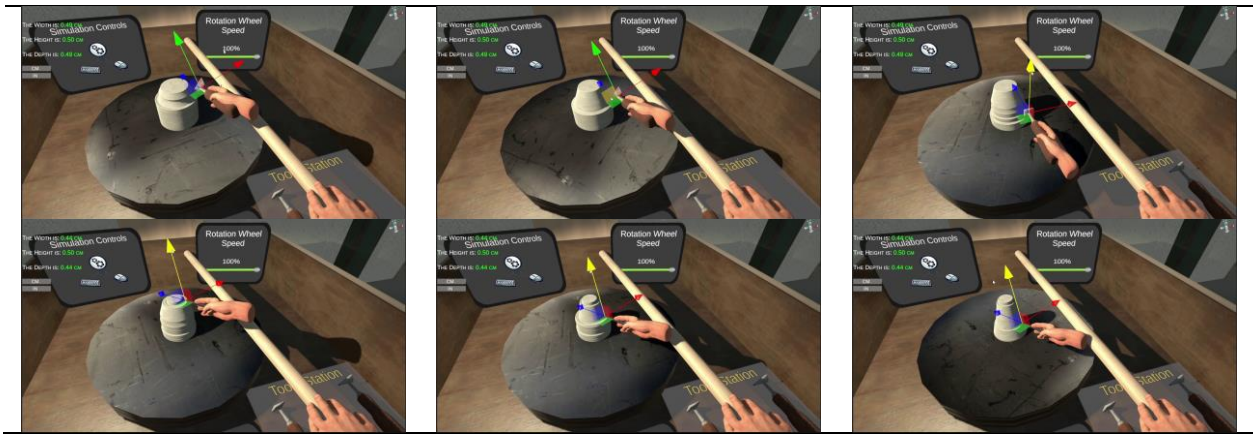


Figure 101. Interaction instances from the turning process in the virtual workspace (see text for details).} Video https://youtu.be/i6CXUOkA4Ww

Centring (Scene A)

Scene A represents the preparation phase. In workshop practice, the practitioner mounts and centres the formwork on the wheel, secures it, pours liquid plaster, and allows it to set, producing a cylindrical blank aligned with the axis of rotation. In the simulator, this phase is condensed into an ordered placement-and-pouring sequence that establishes correct starting conditions for turning.

Scene A focuses on preserving procedural order: a stable formwork, correct placement on the wheel, and a plaster blank prepared for turning. This ensures that Scene B begins from a dynamically stable condition consistent with real plaster-turning practice.

Turning (Scene~B)

In Scene B, the plaster master model is refined using handheld tools while the wheel rotates (see Figure 102). The user works in real time with the revolving workpiece. Tool motion and material response are tightly coupled. Each tool reflects a specific role in plaster turning:

- Rectangular tool: Produces planar cuts and flattens surfaces, establishing reference planes.
- Triangular tool: Supports sharper, deeper cuts for defining inflexion points and articulated profiles.
- Scriber: A fine-point instrument for delicate detailing and controlled point pressure.

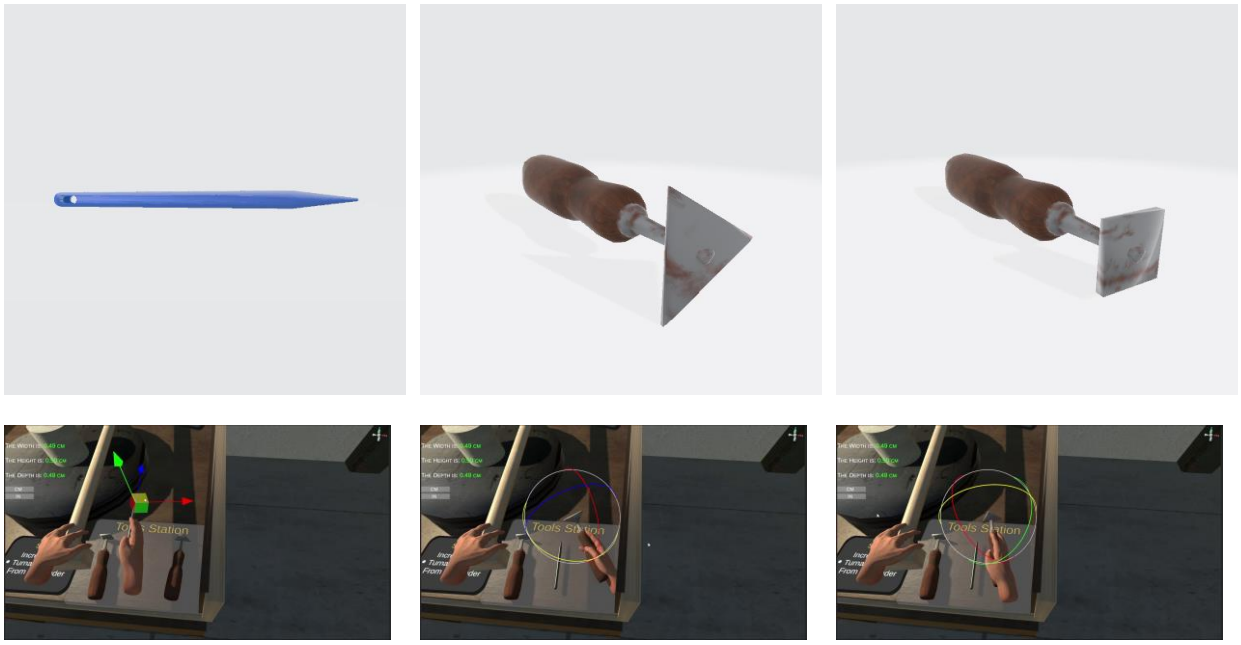


Figure 102. Virtual cutting tools. Top row: Left: Scriber tool. Centre: Rectangular turning tool. Right: Triangular turning tool. Bottom row: Inworld tool selection and handling. Video: https://youtu.be/Uo19tK2eGnY

Rotation is the central mechanical concept of this phase. In workshop practice, the practitioner continuously adjusts wheel speed to stabilise cutting and control the depth of cut. The simulator couples wheel rotation, plaster body rotation, and tool responsiveness to a continuously modulated angular velocity. A minimum speed threshold prevents subtraction when the wheel is stopped or rotating too slowly, reflecting real-world physics where the tool cannot remove plaster without sufficient rotational momentum.

Audio Feedback

Sound is integral to replicating the sensory experience of plaster turning. The simulator implements a dynamic audio system using two independent looping sources:

1. **Mechanical rotation.** Triggered when the wheel exceeds a minimum speed threshold, providing immediate confirmation of motion and speed.
2. **Scraping/cutting**. Activated when a tool contacts the surface *and* the wheel is rotating sufficiently. The amplitude conveys the intensity of the turning action.

This synchronisation of mesh deformation, tool motion, and auditory feedback closes the perception--action loop and supports the learning of tacit craft knowledge, where sound is used to judge material stability and cutting continuity.

11.4 User Experience and Interaction Design

The development followed a User-Centred Design (UCD) methodology, with practitioners of porcelain slip casting participating throughout all stages of concept development, prototyping, testing, and refinement. Because our objectives involve real-time 3D interaction, conventional low-fidelity software prototyping methods were insufficient for evaluating the core behaviour of turning tools, rotation dynamics, and mesh responses.

To address this, this work adopts an iterative prototyping strategy: early builds focused exclusively on validating functional mechanics, while later builds introduced a structured user interface (UI), feedback systems, and workflow accuracy. This approach enabled continuous expert verification of interaction correctness, craft authenticity, and usability across four development iterations.

To enable continuous expert verification of interaction correctness, craft authenticity, and usability, we adopted an iterative prototyping strategy across five development iterations. Early builds focused on validating the interaction mechanics, while later builds introduced a structured user interface (UI), feedback systems, and closer alignment with workshop workflow. Figure 103 illustrates the evolution of the UI and user experience (UX), in (approximate) temporal alignment with iterative prototyping and expert evaluation of intermediate prototypes, which are elaborated in Section~\ref{sec:ucd}. The remainder of this section regards the design of the UX and its principles.

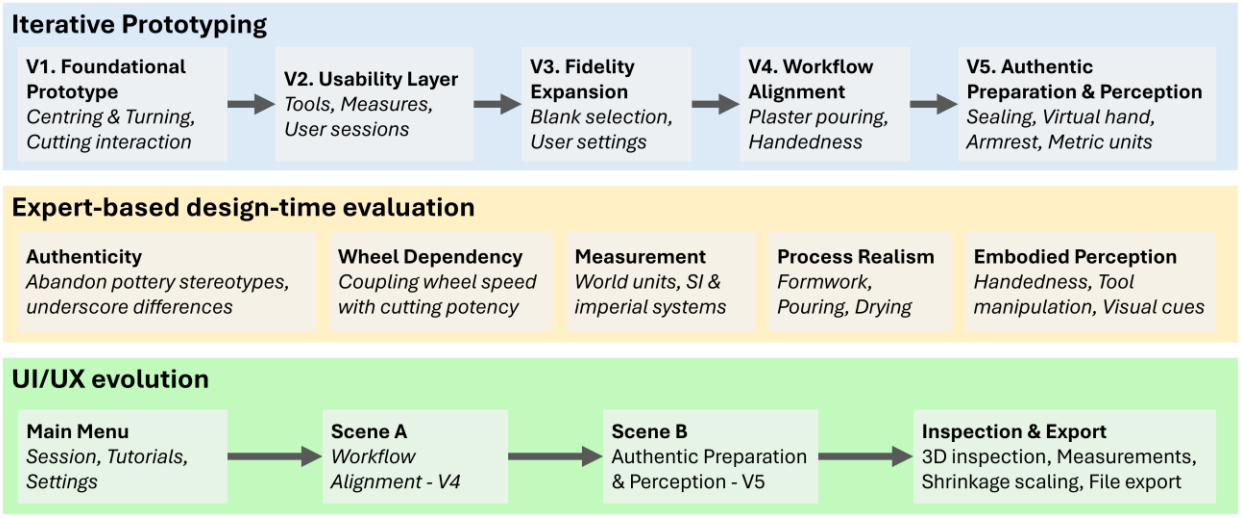


Figure 103. Evolution of the UI and UX, along with prototype version development and expert evaluation.

11.3.1 UI Design Principles

The user interface and interaction design of the system were guided by a set of principles intended to balance accessibility for novice to expert plaster-turning practice with fidelity. Rather than aiming for a feature-rich general-purpose modelling tool, the system foregrounds a single, well-defined workflow in which each interface element is anchored in a specific craft action.

1. To serve simplicity and maintain low cognitive load, the UI exposes only the necessary controls at each stage. When crafting, the user sees only the immediately relevant set of controls, reducing cognitive load.
2. To maintain a sense of material engagement, the simulator provides real-time feedback across multiple channels.
3. The visual design evokes the constraints of a plaster workshop. The camera framing and tool representation reflect the experiences documented in the ethnography. The plaster is rendered with sufficient detail to suggest a turned, slightly chalky material.
4. The simulator supports exploration. Users can adjust speed and tool position and reset the workpiece if a cut goes wrong. The system does not forbid `incorrect' cuts; instead, it allows users to see the consequences. This mirrors the pedagogical role of sacrificial plaster blanks in the workshop, where material is expended in the service of skill development.
5. The interaction scheme requires no prior experience with plaster turning or complex 3D software. Core actions are achieved with standard mouse and keyboard input. Visual cues, such as tool highlighting and axis emphasis, help users understand what can be manipulated.
6. The ordering of stages, the toolset, and the speed parameters are derived from real constraints and from insights of professional turners. Where simplifications are introduced, they are negotiated with experts to ensure the resulting environment reflects meaningful variations in practice.

11.3.2 User Experience

The user experience mirrors the workflow described in Section 9.2. Users are guided through four stages: entering the system via the main menu, preparing the plaster master model, turning and shaping the rotating workpiece, and finally inspecting and exporting the result. This structure minimises navigational complexity to foreground the process of plaster turning.

Main Menu

The main menu provides access to all system functions. From this screen, users can start a new turning session, access tutorials, and adjust user settings. The default path leads to a new session for first-time users to begin with minimal effort.

Centring (Scene A)

This preparation scene reproduces the ritual of the porcelain workshops. The user begins by selecting a hand preference and a plaster blank. The user must first activate a compass tool to mark the true centre of rotation on the wheel head. Following this, a sealing pad is applied to create a stable, leak-proof gasket. The mould is then placed at the marked centre. The plaster master model is mounted on the virtual wheel and begins to rotate about its symmetry axis.

The pouring process is explicitly modelled: the user triggers an animation of a jug filling the mould with liquid plaster. A simulated drying period follows, during which the plaster sets and becomes workable. This sequence replaces the instant materialisation of earlier versions, emphasising the temporal constraints of manufacturing the plaster blank.

Turning (Scene B)

The turning scene hosts the core of the interaction. Users control the rotation speed through a dedicated interface element and manipulate turning tools by moving the cursor or controller within the scene. To improve spatial orientation, the selected tool is now accompanied by a virtual hand model, making the tool's position and rotation immediately readable in 3D space. When a tool is brought into contact with the rotating plaster surface, material is removed according to the solids-of-revolution model described in Section 6, and the geometry of the workpiece is updated in real time. The measurement interface supports a dual-unit system, allowing users to toggle between centimetres and inches.

Finishing and Exporting

Once centring and turning are complete, the resultant plaster geometry is presented in a stable, non-interactive view that supports a) profile and key dimensions, and b) 3D inspection for user reflection. The user can examine the resulting profile and review the key measurements.

A scaling option is provided to incorporate shrinkage allowances so that exported geometry remains production-ready after firing-related dimensional change. Exported models are metric and suitable for downstream use, such as in 3D printing, archival storage, or 3D rendering.

11.5 Iterative Design and Expert Evaluation

The development of the simulator followed an iterative, UCD methodology shaped by continuous collaboration between developers and porcelain craft experts. From the outset, the goal was to create a serious game for crafting knowledge transmission. A simulation-oriented approach offers more training opportunities in mechanical and perceptual tasks than an entertainment-oriented gameplay.

The process combined technical prototyping within Unity with structured feedback loops informed by craft knowledge. The methodology can be summarised in two key pillars:

1. Iterative Prototyping: The simulator was developed through repeated cycles of implementation, critique, and refinement.
2. Expert Collaboration: Each cycle involved identifying realism gaps and workflow inconsistencies, with targeted revisions based on expert feedback.

Early iterations focused on verifying the core mechanics of the system; later iterations refined authenticity and cultural alignment.

11.4.1 Versioned Software Prototypes

In the following subsections, we describe the five successive versions (V1--V5) of the simulator. Each version introduced targeted changes in response to expert feedback, gradually improving the authenticity of the plaster-turning workflow, the coherence of interaction, and the quality of the digital models.

Version 1 --- Foundational Software Prototype

The first build established the essential two-scene structure of the simulator. Scene A provided a preliminary workspace in which a plaster blank could be placed and inspected, while Scene B implemented basic turning actions on a solid-of-revolution representation of the blank. Users could navigate the camera, select tools from an initial toolset, and deform the profile of the rotating blank through direct manipulation of the solids-of-revolution mesh (Figure 104). Real-time performance is achieved with free viewpoint rendering at 60 fps while editing the mesh for an angular sampling of v = 72, on a conventional GPU.

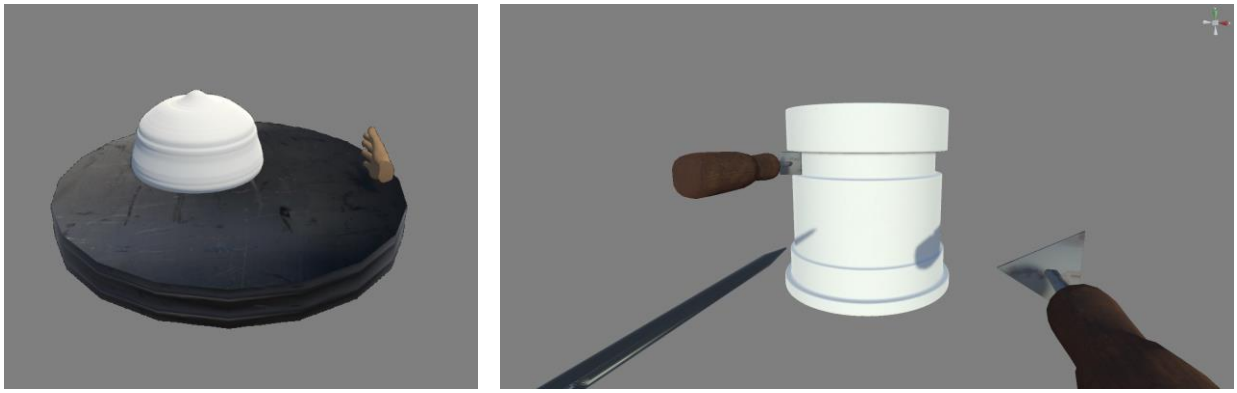


Figure 104. Version 1. Left: Preliminary centring workspace with a single plaster blank placed on the wheel. Right: Early cutting scene with a basic toolset and direct profile editing on a solid-of-revolution blank.

Expert feedback on Version 1 was cautiously positive, but framed it as a proof of concept rather than a usable simulator. Practitioners recognised the potential of the core metaphor of a rotating blank shaped by a cutting tool, but noted the absence of a structured user interface, the generic surface appearance, and the lack of a clearly defined plaster-turning workflow. These observations motivated a second iteration focused on usability and basic interaction scaffolding.

Version 2 --- Usability and Interaction Layer

The second build introduced a comprehensive usability layer for repeated expert testing. Tool selection was moved into dedicated panels, measurement overlays were added, and firing-related functions were implemented so that users could complete a session and begin a new trial. A firing animation, restart options, and OBJ exporting made it possible to resume work, compare results, and run repeated trials under controlled conditions (Figure 105).

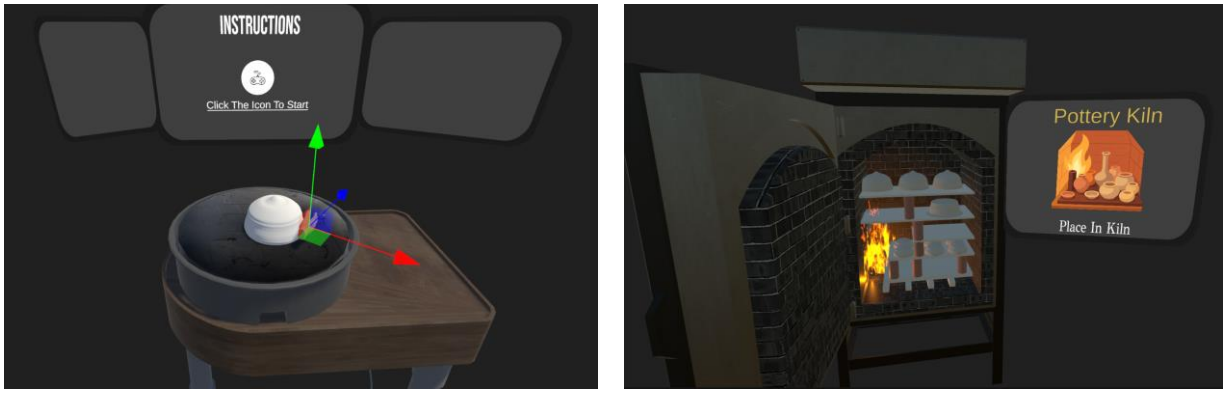


Figure 105. Version 2. Left: Updated centring scene with a more realistic wheel surface and tool interaction via a 3-axis gizmo. Right: Extended interface with firing controls, session management, and export options supporting repeated expert trials.

Testing Version 2 revealed improved interaction coherence and allowed experts to engage with the tool for longer sessions. However, the material behaviour and workflow still reflected a pottery-inspired sequence more than plaster turning. Turning could be performed on a stopped wheel; centring was not grounded in plaster practice, and the blank material continued to resemble clay. These limitations motivated a third iteration aimed at richer geometry and surface fidelity.

Version 3 --- Expanded Models and Improved Surface Fidelity

Version 3 extended the simulator in two main directions: model variety and visual realism. At start-up, users could choose between multiple initial blank geometries, moving beyond a single default shape. A dedicated tool station was added, with dimensional readouts and overlays for key measurements. Surface materials and lighting were refined to better evoke the look of turned plaster, while user preferences (for example, camera options) were made persistent across sessions (Figure 106).

Figure 106. Version 3. Left: Start-up interface allowing users to select from multiple plaster blank geometries. Centre: Tool station with dimensional readouts and measurement overlays. Right: Scene overview with a refined wheel, improved surface rendering, and an expanded toolset.

Expert feedback at this stage provided critical guidance on authenticity. While the increased fidelity of geometry and surface made the simulator more engaging, practitioners noted that several starting shapes were decorative rather than cylindrical, and the overall workflow remained underspecified. Handedness was still unsupported, and the sequence of actions did not yet map cleanly onto the established plaster-turning procedure. These observations motivated a major redesign of the preparation and turning stages for the next version.

Version 4 --- Realism and Workflow Alignment

The fourth build implemented substantial changes, with the explicit goal of aligning the simulator with authentic plaster-turning practice. A pouring stage was introduced in Scene A: the user places the model into a mould, pours plaster from a jug, and initiates a simulated drying period before the blank can be turned. The wheel was reconceived as a mould-support wheel, and the initial blanks were restricted to cylindrical forms that match the requirements of plaster turning. Modes for left- and right-handed operation were added, and cutting actions were constrained to occur on the rotating blank rather than on a static surface (Figure 107).

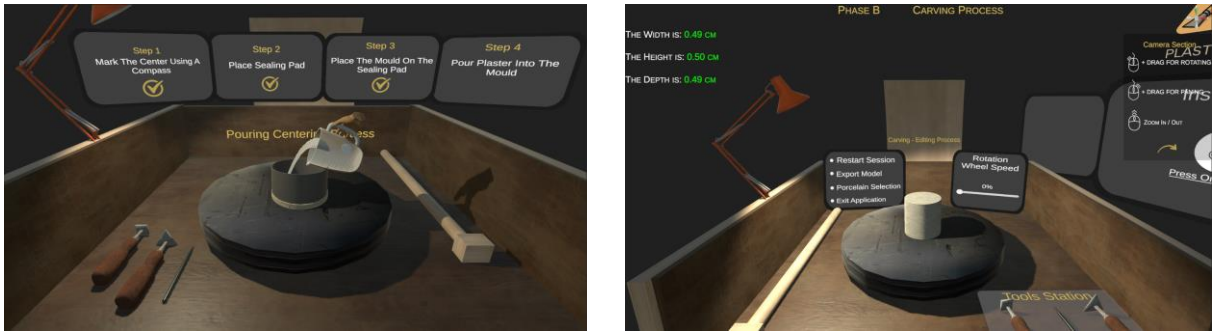


Figure 107. Version 4. Left: Scene A (Preparation). Redesigned workspace showing mould placement and plaster pouring from a jug. Right: Scene B (Turning). Updated workspace with a cylindrical blank supported by the mould on a rotating wheel.

In Scene B, a virtual hand operation mode was added. A semi-transparent hand and tool-handle overlay provides artificial visual feedback on tool pose and rotation, encouraging users to develop an active perceptual understanding of how the tool engages with the rotating surface (Figure 108).

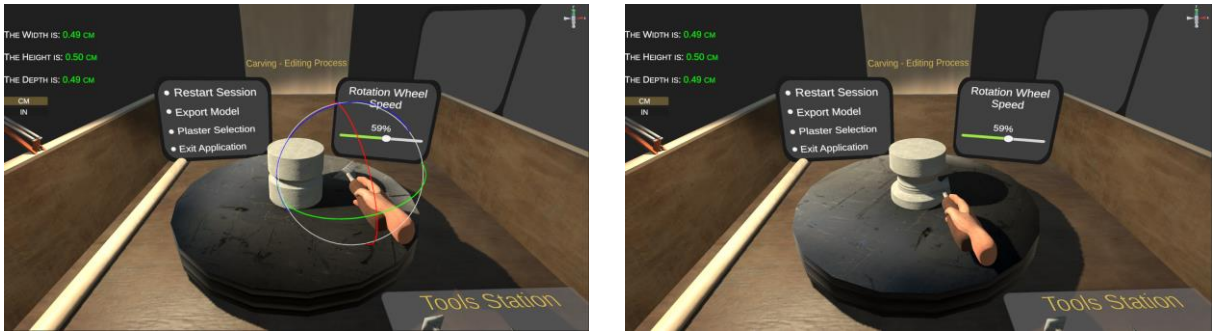


Figure 108. Version 4. Scene B (Turning). Left: Tool positioning with orientation aid superimposed. Right: Cutting action with the virtual hand.

Experts recognised Version 4 as the first build that convincingly captured the sequence of plaster turning. The addition of the mould-supported wheel, cylindrical blanks, and handedness options improved ergonomic realism and established a credible workflow for validation and teaching. At the same time, the pouring and drying stages were still presented conceptually rather than through physically accurate simulation, and practitioners requested additional cues for the preparation steps. These issues informed the final refinement.

Version 5 --- Authentic Preparation and Active Perception

The fifth build refined both scenes, focusing on preparation detail, active perception, and professional measurement practice. In Scene A, compass-based centre marking and a sealing pad were introduced to mirror the physical preparation of plaster models. The pouring and drying sequence was enhanced with animations and information displays that explain the underlying process without attempting a full physical simulation. Additionally, dual-unit (centimetre/inch) support was introduced to accommodate professional conventions in different workshops (Figure 109).

Figure 109. Version 5. Scene A (Preparation). Left: Compass-based centre marking of the model. Centre: Sealing pad placement between the model and mould. Right: Information display accompanying the animated drying process.

More importantly, the stabilisation armrest was placed at a realistic configuration according to the handedness of the simulator user. In this configuration, the hand had to be supported by the armrest, which correspondingly limits the freedom of the tool. In the real workshop, this configuration enforces a particular field of view, which the simulator user can assume.

This visual skills training regards not only the occluded hand-tool-armrest system, but also the workpiece. The practitioner learns to attend to workpiece formation not by looking at the tool's cutting point, but at the workpiece profile where the spotlight shines upon it. The profile view is unobstructed and taskwise preferable, and explains the selection of cutting side: it is such that it does not cast a shadow on this profile (Figure 110).

Figure 110. Version 5. Scene B (Turning). Left: Tool positioning with a virtual hand overlay that visualises handle rotation. Right: Carving action with the virtual hand superimposed, providing artificial 3D feedback for tool orientation.

Expert feedback on Version 5 indicated that the simulator now supports both explanatory use in teaching and exploratory use by practitioners. The preparation stage no longer resembles pottery but plaster turning; the turning actions are appropriately constrained by rotation, and measurement support aligns with professional practice. Across these five versions, three themes emerged: improved workflow authenticity, increased interaction coherence, and enhanced model quality and responsiveness. In the next subsection, we reorganise the version history along these dimensions and summarise the evolution.

11.4.2 System Evolution and Value Gains Across Iterations

Across the five development versions, the system gained value through three largely independent but interacting improvements: increasing workflow authenticity, strengthening interaction coherence, and enhancing model quality and responsiveness. Whereas Section 9.4.1 described each version chronologically, this subsection regroups the same trajectory along these three dimensions to clarify how expert feedback translated into design decisions.

Workflow Authenticity

Early versions contained pottery-based elements that blurred the distinction between plaster turning and wheel-thrown ceramics. Centring was treated generically; the wheel could be stopped without affecting turning operations, and the preparation stage did not include mould placement or plaster pouring. From Version 3 onwards, expert comments consistently highlighted these mismatches.

Version 4 addressed workflow authenticity by introducing a mould-supported wheel, restricting blanks to cylindrical forms, and enforcing rotation for turning actions. Version 5 further aligned the workflow with professional practice by adding compass-based centre marking, sealing pads, and an explicit pouring-and-drying sequence. Together, these changes repositioned the simulator from a general turning toy towards a credible representation of the plaster-turning procedure.

Interaction Coherence

Interaction coherence concerns the extent to which users can form a stable mapping between interface controls and craft actions. In Version 1, controls were minimal and largely implicit, which made the system difficult to operate beyond short demonstrations. Version 2 introduced dedicated tool panels, camera controls, measurement displays, and firing/export functions, enabling experts to complete full sessions and restart them under comparable conditions.

Subsequent versions refined this interaction layer rather than replacing it. Version 3 added persistent preferences and clearer feedback on tool selection and measurement, while Version 4 integrated handedness modes and ensured that turning actions were tied to wheel rotation. Version 5 extended coherence into the perceptual domain by adding a virtual hand overlay that makes handle rotation and tool orientation explicitly visible. As a result, experts reported that once they had learnt the controls, they encountered a stable and predictable mapping between interface actions and craft operations.

Model Quality and Responsiveness

Improvements to model quality and responsiveness targeted both the underlying solids-of-revolution representation and its visual presentation. Version 1 demonstrated that real-time deformation of a rotating profile was feasible in Unity but offered only a basic mesh and generic materials.

Version 2 stabilised performance and provided clearer measurement overlays, while still presenting a clay-like surface.

Version 3 introduced multiple starting geometries, refined shading to evoke turned plaster, and extended measurement support for experts to evaluate the resulting forms more critically. Version 4 ensured that the blanks and mould support matched the mechanical constraints of plaster turning, and Version 5 added dual-unit measurement support to match workshop conventions. Across these iterations, experts

increasingly treated the digital models as adequate surrogates for physical blanks when discussing geometry and tolerances.

This combination of authentic workflow, coherent interaction, and responsive models raised the perceived value of the simulator. Experts moved from viewing it as an experiment in interaction design to considering it as a credible environment for analysing and teaching plaster turning.

Table 19 summarises the evolution across versions by listing the main additions, the issues they addressed, and the principal outstanding issues that motivated subsequent design decisions.

Table 19. Iterative development summary. Additions, resolved issues, and remaining issues across five versions of the simulator.

Ver.	Additions	Issues Addressed	Issues Remaining
V1	Basic two-scene flow; initial turning tools; solids-of-revolution deformation.	Validated the feasibility of real-time turning/rotation in Unity; established core digital metaphor.	No structured UI; generic materials; no pouring/drying; no handedness; turning unconstrained by rotation.
V2	Tool panels; measurement UI; firing animation; restart/export functions.	Strong usability improvements, enabled repeated expert trials, and clearer session structure.	Material remains clay-like; no explicit pouring stage; hand orientation not represented; turning possible on a stopped wheel; workflow still pottery-like.
V3	Model selection, improved surface materials, and persistent user preferences.	Increased fidelity of form and surface; enriched user control; suitable for extended evaluation.	Several starting shapes are unrealistic (decorative rather than cylindrical); workflow is under-specified; handedness is still unsupported.
V4	Plaster pouring; mould-support wheel; cylindrical blanks only; left/right-hand modes; rotation-dependent turning.	Achieved alignment with authentic plaster-turning sequence; improved ergonomic realism; established credible workflow for expert validation.	Pouring and drying represented conceptually rather than physically; material behaviour and sensory cues still simplified; further visual/tactile realism recommended.
V5	Compass centre-marking; sealing pad; animated pouring/drying with explanations; dual-unit (cm/in) measurement; virtual hand visualisation.	Replaced pottery-style centring with authentic preparation; strengthened explanatory value of the pouring/drying stage; improved tool-orientation feedback; supported professional measurement standards.	Fluid dynamics and drying are animated rather than physically simulated; no haptic feedback; scope for further refinement of material appearance and sound.

11.4.3 Evaluation

HCI research emphasises that the usability, learnability, and user experience of interactive systems emerge through situated interaction, and are best assessed with representative users rather than by inspection [175]. For educational and creative systems, evaluation prioritises workflow comprehension, perceived exploration support, and pedagogical alignment, rather than task efficiency [176, 177]. Following this perspective, we report a learner-centred, formative evaluation designed to validate whether the simulator is usable and learnable by its intended audience (art and design students), and whether the interaction and workflow are coherent with the conventions of the workshop context.

To avoid conflating design rationale with empirical assessment and make the evaluation scope explicit, we separate (i) upstream expert-informed participatory design, which shaped the interaction metaphors and pedagogical objectives, from (ii) post-design evaluation with the target user population.

Methodology: post-design formative validation with learners

The simulator was developed through iterative participatory design involving porcelain-related experts across five design iterations. Given this upstream expert involvement, the purpose of the evaluation reported here is post-design validation: (a) to assess whether students can understand and use the simulator with minimal facilitation, (b) to identify remaining usability breakdowns and workflow inconsistencies, and (c) to evaluate perceived pedagogical adequacy and creative affordances in an educational setting.

Participants, consent, and data handling

Across two workshop deployments (Phase 1: November 2024 and Phase 1: November 2025), we recruited N=23 art/design students (aged 18--30; first--third year) from ENSAD Limoges (Phase 1: n_1=11; Phase 2: n_2=12). Additionally, two teaching assistants with expertise in plaster turning and ceramic 3D printing contributed contextual and pedagogical feedback through observation and debrief interviews (not included in N).

All participants provided informed consent before participation, including specific consent for audio recording where applicable. Data were analysed and reported in anonymised form. Audio recordings and visual documentation were stored on secure institutional storage and accessed only by the research team for analysis.

Study context and session structure

The evaluation was conducted as educational workshops, in two phases:

For Phase 1, a four-day workshop was conducted with an early prototype. Participants engaged in two-hour sessions of open-ended interaction. The goal was diagnostic: to surface major usability barriers, mismatches between intended workflows and student expectations, and early evidence of how learners interpret the interaction metaphors. In parallel, a structured expert inspection (walk-through) was conducted by a researcher to probe edge cases, functional limits, and breakdown points. This was used to prioritise revisions before task formalisation.

For Phase 2, a two-day workshop was conducted with a refined simulator version that incorporated feedback from experts and Phase 1. Participants used the system for 8 hours per participant, in two sessions of 4 hours each. This phase evaluated learnability, workflow coherence, and functional completeness under structured tasks aligned with educational use.

Table 20. Overview of the formative evaluation design.

Phase	Participants	Format	Primary purpose
1	• Students: 11 (aged 18--30; art and design). • Teaching staff: 2 (observers, debrief).	Open exploration sessions with an early prototype; parallel structured inspection by a researcher.	Diagnose usability breakdowns and workflow mismatches; identify revision priorities before task formalisation.
2	• Students: 12 (aged 18--30; art and design). • Teaching staff: 2 (observers, debrief).	Task-based workshop deployment with a refined version. Tasks: reconstruction from drawing, OBJ export, and open-ended modelling.	Assess learnability and interaction coherence under structured tasks; validate readiness for educational deployment and collect remaining suggestions.

Tasks

We used a two-stage task design, which is commonly adopted in formative HCI evaluation. Phase 1 emphasised unconstrained exploration to reveal spontaneous interaction strategies and breakdowns. Phase 2 employed structured tasks to evaluate learnability, workflow coherence, and functional adequacy after refinement.

In Phase 1, participants were invited to explore the system freely, with no predefined goals beyond ``use the simulator as you think appropriate''. This was intended to (i) observe initial interpretation of interaction metaphors, (ii) identify recurring usability errors and points of confusion, and (iii) detect mismatches between the intended workflow and student mental models.}

In Phase 2, participants completed the following four tasks.

1. Attend a brief interaction to re-establish orientation and confirm the clarity of updated interaction mechanisms.
2. Reconstruct a staircase-shaped ceramic piece from a provided printed drawing, starting from a cylindrical primitive, to assess procedural understanding and learnability.
3. Export the resulting model in OBJ format to evaluate functional completeness and technical usability, including potential downstream use for ceramic 3D printing.
4. Create a ceramic form while explicitly considering constraints and affordances of ceramic 3D printing, to assess creative autonomy and pedagogical transfer.

Data collection

We adopted qualitative triangulation across multiple data sources, consistent with formative evaluation practice for interactive educational and creative systems.

In Phase 1, educators and designers documented observations on recurring breakdowns, hesitations, and navigation difficulties. Photographs and screenshots captured interface states and interaction sequences. A researcher performed a systematic walk-through, producing a structured written log of limitations and revision priorities. This phase was diagnostic, aimed at forming actionable revision hypotheses rather than measuring performance outcomes.

Phase 2 included the following feedback collection modalities:

1. Audio recordings: participant commentary during task execution to capture reasoning and evaluative judgements.
2. Observation and notes: facilitators documented task progress, points of friction, and successful strategies.
3. Visual documentation: screenshots/photographs of intermediate and final models.
4. Post-session questionnaire: an online questionnaire using a 5-point Likert scale to assess perceived usability, learnability, workflow clarity, pedagogical relevance, and creative potential.
5. Interviews and debriefings: individual interviews and collective debrief discussions at the end of the workshop to elicit reflective feedback and improvement suggestions.

Data analysis

Data from both phases were analysed using thematic analysis. We first organised observational notes, audio recordings, and debrief material, then iteratively consolidated these into a codebook organised around usability, learnability, workflow comprehension, perceived adequacy of constraints, and pedagogical relevance. Themes were refined through comparison across data sources to support triangulation and reduce reliance on single observations.

Phase 1 analysis prioritised identification of recurrent breakdowns, misunderstandings of interaction metaphors, and workflow inconsistencies, directly motivating design revisions between iterations. Phase 2 analysis focused on learnability, interaction coherence, functional adequacy (including export), and perceived pedagogical usefulness under structured tasks.

11.4.4 Evaluation findings

This subsection reports findings from the learner-centred formative evaluation. We organise results thematically, consistent with qualitative reporting practice in HCI for educational and creative systems. Findings are grounded in triangulated evidence from (i) observation and facilitator notes, (ii) participant commentary and debrief discussions, (iii) workshop artefacts (intermediate and final models), and (iv) post-session questionnaire responses (Phase 2).

Initial comprehension and learnability

Across both phases, participants established a usable mental model of the simulator's core interaction logic after a short exposure. During Phase~1, learners were able to initiate rotation, select tools, and attempt subtractive shaping by relying on interface affordances and prior experience with 3D interaction conventions.

In Phase 1, a recurrent breakdown was that several participants conflated centring with turning and attempted shaping before establishing a stable axis reference; this motivated a redesign that strengthened the step structure and guidance cues so that centring is completed before turning actions become effective.

In Phase 2, the familiarisation segment was sufficient for participants to re-establish orientation and proceed into the reconstruction task without extended instruction.

A recurring learnability indicator was the emergence of purposeful parameter adjustments: participants tended to treat wheel speed as a controllable variable rather than a fixed setting, altering it in response to the visual state of the evolving form. This behaviour aligns with the simulator's intended operationalisation of the turning process as a feedback loop governed by visual and geometric cues.

Usability and interaction coherence

Phase 1 revealed usability breakdowns related to navigation, tool positioning, and the visibility of interaction constraints. Participants sometimes struggled to infer (i) the effective cutting zone of a tool, and (ii) the geometric consequences of sustained contact with a rotating surface. These observations motivated interface refinements between phases.

In Phase 2, observation and debrief feedback indicated more coherent action--effect mappings: once participants understood the basic control scheme, interactions were generally experienced as predictable and consistent. This reduction in unexpected shape changes or inadvertent tool engagement is important in learning systems, as stable causality supports procedural reasoning and self-directed exploration.

Workflow understanding and procedural awareness

A central outcome of both phases was that learners began to reason about turning as a sequence of operations rather than as isolated tool actions. During Phase 1 exploration, participants frequently discussed the ordering of steps and the need to preserve material for later adjustments. Such strategies mirror conventional workshop practice and suggest that the simulator supports procedural thinking rather than merely geometric editing.

During Phase 2, the reconstruction task encouraged deliberate planning. Participants' intermediate models and verbalised strategies commonly reflected staged removal instead of attempting to reach the final shape in a single pass. This behaviour is consistent with making the craft workflow intelligible as a controlled process of successive approximations.

Perceived realism within scope and constraint awareness

Participants consistently identified the simulator's constraints as consequential for decision-making. The irreversibility of material removal, the dependence of cutting on rotation, and the restriction to axisymmetric geometry were repeatedly described as differentiating the system from general-purpose 3D modelling tools. Rather than perceived as limitations, these constraints were interpreted as pedagogically valuable because they encourage anticipation of consequences and foster reflective adjustment.

Perceived realism was discussed in procedural terms, rather than in terms of physical force, vibration, or sound. The discussions concerned what can be inferred from the evolving profile and measurements, and how actions must be sequenced. This matches the current operationalisation of cues: participants predominantly relied on on-screen visual readout and geometric guides when monitoring progress and deciding subsequent actions.

Pedagogical and creative affordances

In the open-ended task, learners used the simulator to explore form under manufacturing constraints. Observations indicated reflective interaction: participants frequently paused to inspect the profile and measurement cues before continuing, suggesting iterative evaluation rather than exploratory manipulation. In debrief discussions, the simulator was positioned as complementary to physical workshop instruction: useful for rehearsal, comparison of alternative shaping strategies, and structured discussion of constraints.

A salient pedagogical advantage was the low cost of iteration: the ability to reset and retry without material waste encouraged experimentation and reduced the penalty of failed attempts. This aligns with the simulator's role as an instructional medium for understanding process logic and constraint-driven design decisions, rather than as a substitute for embodied material practice.

Limitations raised by learners

Participants articulated limitations consistent with the system's scope. The absence of physically grounded cues, such as resistance or tactile guidance, was noted as a missing component for judging cutting depth and surface quality. Additionally, the restriction to solids of revolution was seen as a useful simplification for learning rotational shaping, but also as an expressive boundary. Participants expressed interest in future extensions that would support post-turning modifications or non-axisymmetric features.

11.4.5 Summary, scope and limitations of the evaluation study

The formative evaluation provides converging qualitative evidence that the simulator is learnable and usable by its intended audience in an educational setting, and that it supports procedural understanding of centring and turning through visual and geometric feedback. In keeping with the study design, these findings should be interpreted as evidence of design adequacy and readiness for pedagogical deployment, rather than as a measure of learning gains, skill transfer, or comparative performance.

This evaluation is a post-design formative validation of an educational simulator. Its scope is intentionally limited to assessing usability, learnability, workflow coherence, perceived realism within the represented cue set, and pedagogical relevance for art and design students in an authentic teaching context.

The study supports claims that (i) learners can understand and operate the simulator with limited facilitation, (ii) the interaction conveys key procedural dependencies of centring and turning (notably the role of rotation and irreversible removal), and (iii) the system is perceived as pedagogically useful for exploring form under process constraints.

The study does not aim to: measure learning gains, skill transfer, or long-term retention; provide controlled comparisons against alternative tools or instructional methods; revalidate craft authenticity or domain correctness; assess realism in terms of physical/haptic cues, which are not operationalised in the current system.

The simulator represents cues that are accessible through visual inspection and geometric measurement (e.g., evolving profile, diameters, depths). It does not realistically simulate the preparatory blank preparation stage; that stage is included only for contextualisation. Moreover, the shaping model is restricted to axisymmetric forms (solids of revolution). These constraints are appropriate for the current research focus, but delimit the generality of conclusions.

Workshops were conducted in authentic educational settings, which strengthens ecological validity but may introduce variability in participant behaviour, facilitation style, and task interpretation. As with many qualitative HCI evaluations of creative and learning systems, findings should be interpreted as evidence of design adequacy for the studied context rather than as statistically generalisable outcomes [175, 177].

11.8 Discussion

11.8.1 Realism

Realism emerged as a central requirement throughout all development iterations. Expert feedback consistently demonstrated that even small deviations from authentic plaster-turning practice created conceptual and procedural confusion. Early prototypes included pottery-oriented elements such as hand-centring, decorative initial shapes, and firing animations, which experts rejected as incompatible with porcelain slip casting.

Moreover, realism proved necessary not only at the visual level but also at the process level. This included conceptual steps such as pouring liquid plaster, drying, and restricting blanks to cylinders. Experts highlighted the appropriateness of the audio feedback, noting its positive impact on engagement and its utility in conveying tool contact.

These corrections transformed the simulator from a generic 3D sculpting tool into a meaningful approximation of an established craft practice. The application serves as an introductory passage that foretastes, rather than replaces, physical practice.

11.8.2 Relevance

While the current work evaluates the simulator in a workshop setting with craft experts, its deployment in non-formal learning environments, such as decorative arts museums and exhibitions, is straightforward. Unity's multi-platform export pipeline allows the same turning environment to be presented as a 3D desktop application, a browser-based experience, or a VR installation. In these contexts,

the system could function as an educational aid installed near the exhibits, illustrating their making. Coupling embodied interaction with visual feedback makes hidden workshop practices legible, supports the interpretation of heritage, and is a flexible platform for communicating tacit knowledge without exposing fragile objects to risk.

Beyond rehearsing motor routines, the simulator enables the training of perceptual skills. The simulator encourages active visual exploration, training the user to find the viewpoints that best reveal invariants such as symmetry and profile continuity. The dynamic audio feedback facilitates learning to attune attention to a specific sound. In Gibsonian terms, this practice 'educates attention' [164]: the user learns to attend to the characteristic sensations that signal a successful turning action.

The redesign demonstrated that the simulator is a candidate educational tool for porcelain artefact designers. Because the system enforces correct process sequencing, it exposes learners to the constraints of the craft without requiring physical materials or equipment. The simplified solids-of-revolution model ensures that users focus on the conceptual constraints of cylindrical shaping, while the measurement UI enables precision training that parallels real workshop tasks.

11.8.3 Visualisation of Downstream Processes

To demonstrate that the simulator produces geometry compatible with professional rendering workflows, we exported the virtual workpiece for offline visualisation. Using the Mitsuba 3 rendering engine [172], we simulated the optical and geometric evolution of the ceramic object through three distinct production stages: the sealed plaster master, the calcined (fired) plaster, and the final glazed porcelain.

The visualisation allows for a direct comparison of material properties (Figure 111). The sealed plaster exhibits a satin finish due to the application of separators. The calcined plaster appears matte and chalky following dehydration. Most importantly, the final porcelain model incorporates a 14% uniform shrinkage to simulate the densification of the clay body, alongside a high-gloss vitreous glaze. This workflow confirms that the geometric data is compatible with physically-based rendering (PBR), effectively bridging the gap between digital design and the prediction of material outcomes.

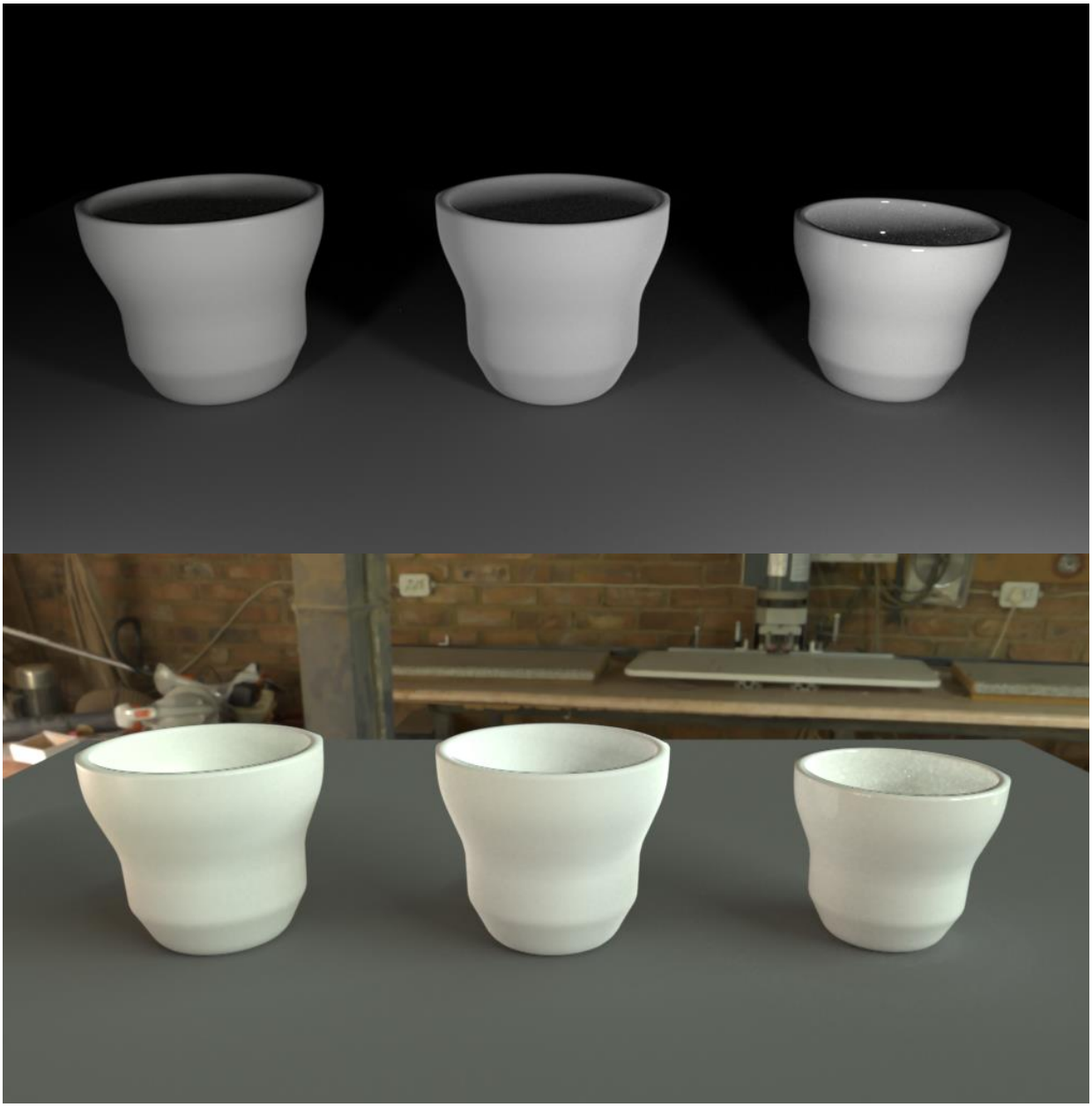


Figure 111. Physically based rendering (PBR) of a porcelain cup, comparing simulated material properties and geometric shrinkage. Left: The plaster master exhibits a satin finish. Centre: The calcined plaster master shows a characteristic matte, chalky surface. Right: The fired porcelain piece demonstrates a 14% volumetric shrinkage and a high-gloss glaze. Renders were generated simulating indicative studio lighting (top) and workshop illumination (bottom).

11.9 Conclusions

This section presents an interactive simulator of plaster model turning for porcelain slip-casting. The goal was not to replicate the full physics of cutting plaster, but to provide a faithful, workshop-informed environment in which the logic of the process, the constraints of rotation, and the perceptual cues used by experts can be communicated and explored.

Ethnographic documentation shaped both the vocabulary and the structure of the system. It informed the division into a preparation scene and a turning scene, the identification of key checkpoints (e.g., centring), and the emphasis on the practitioner's viewpoint and sensory feedback. In doing so, the simulator treats craft work as more than visible motion: it makes the ordering of actions, the role of stabilising structures (such as the armrest), and the practitioner's attention to diagnostic visual and auditory cues explicit.

Through five successive software versions (V1--V5), developed and refined with expert feedback, the system progressed from a feasibility prototype to a workflow-aligned representation of plaster turning. Key changes included restricting initial blanks to cylindrical forms, enforcing rotation as a prerequisite for cutting, introducing an explicit preparation stage (compass marking, sealing pad, pouring and drying), supporting left- and right-handed configurations, and adding a virtual-hand overlay to make tool orientation and handle rotation legible during interaction. These refinements were driven by repeated expert critique and were essential for authenticity.

The resulting system provides two complementary values. First, it offers an environment for teaching and rehearsal: learners can practise sequencing, speed control, and the interpretation of perceptual cues without consuming workshop materials. Second, it produces explicit, inspectable digital traces of turning actions and geometrically consistent models that can be exported for downstream use, including offline visualisation and physically based rendering workflows.

11.8.1 Limitations and Next Steps

The limitations outline a roadmap from the current visually oriented, desktop simulator towards richer, physically informed, embodied, and immersive configurations.

Tool handling is currently mediated without haptic feedback or physical coupling. Users do not feel the forces, vibrations, or resistance that arise in plaster turning. Moreover, tools may be approached to the workpiece without spatial constraints induced by human dimensions and articulation. Future work will utilise interaction props, such as force- or vibration-feedback styluses, and a foot pedal for adjusting wheel speed. These low-cost extensions support simplified embodied interaction and, importantly, enable the recording of coordinated actions.

A VR export of the simulator is available using Unity's cross-platform export facilities. The effect of immersion on learning remains to be evaluated. Future studies will compare desktop and VR usage to examine when the additional complexity of VR is beneficial, depending on the training, documentation, or engagement goals.

Pouring and curing are depicted as scripted events, not as time-dependent hardening or rheological actions. Users therefore do not experience the consequences of working too early or too late in the curing process. A natural next step is to introduce simplified, parameterised process dynamics, such as adjustable curing windows and changes in tool response over time. This way, timing skills are trained.

The system is restricted to solids of revolution, i.e., axisymmetric forms. This covers a large class of simple plaster models but excludes experimental or composite shapes with handles, undercuts, or local structures that depart from pure rotational symmetry, such as surface patterns or engravings. The

integration of interactive editing methods found in conventional 3D sculpting editors to support such post-turning and pre-firing tasks is warranted.

Important limitations remain. Interaction currently relies on standard desktop input without haptic forces or vibration, and preparation stages are represented as scripted events rather than as time-dependent material processes. The geometric representation is restricted to solids of revolution and therefore does not cover non-axisymmetric features such as handles, undercuts, or local sculpted detail. Future work will investigate low-cost physical props and haptic devices, evaluate the trade-offs of immersive VR use compared to desktop interaction, and explore extensions that introduce simplified process timing and broader shape classes.

Beyond the specific case of plaster turning for porcelain slip-casting, the work demonstrates a practical pathway for developing craft simulators that balance computational tractability with process fidelity: start from ethnographic structure, iterate with expert critique, and design interaction around the constraints and perceptual invariants that experts actually use. This approach is applicable to other workshop practices in which skill depends on active perception, staged routines, and controlled risk.

12 Conclusions

This deliverable has reported a dependency-driven set of methods and software components for simulating craft actions and previewing their outcomes. The central idea is compositional: craft-relevant meaning (what an action is, and which parameters matter) is made explicit through semantic structures (Section 3); craft-relevant physics is captured through a hybrid strategy that combines interactive simulation with high-fidelity FEM and learning-based surrogates (Section 4); and craft-relevant appearance is addressed through light-transport simulation and a reusable visualisation toolbox (Section 5). On top of these foundations, the deliverable has demonstrated a family of craft-facing utilities and systems (Sections 6–11) that exploit the same pipeline logic rather than being isolated prototypes.

12.1 Summary of achievements

Three technical contributions anchor the work.

First, the deliverable establishes a semantic framing for craft simulation (Section 3) that supports traceability: actions, parameters, workpieces, tools, and outcomes can be described in a way that remains interpretable when simulations are reused, compared, or extended. This layer is essential whenever simulation outputs must be communicated beyond the immediate developer team (e.g., to practitioners, educators, or curators) and whenever results must remain reproducible across versions.

Second, it introduces a hybrid simulation methodology (Section 4) that confronts the main practical bottleneck in craft action simulation: high-fidelity multiphysics FEM remains too expensive for interactive use, yet purely real-time physics engines cannot capture all constitutive behaviours of interest. The proposed approach uses (i) interactive real-time physics where acceptable, and (ii) offline FEM sampling to generate structured ground truth that is then compressed into lightweight machine-learning surrogates. In this way, high-fidelity behaviour can inform interactive experiences without resorting to exhaustive precomputation or unmanageable databases.

Third, the deliverable provides a light-transport and visualisation infrastructure (Section 5) that turns simulation outputs into controlled, physically based visual evidence. This is not merely “rendering for presentation”: appearance simulation is treated as a design and evaluation instrument, particularly important for crafts where perception depends on illumination, material thickness, translucency, gloss, and viewpoint.

Building on these foundations, Sections 6–11 demonstrate how reusable infrastructure supports diverse craft tasks: relief generation from 2D input (Section 6), computationally efficient handling of axisymmetric solids (Section 7), mould-related modelling workflows (Section 8), the design of composite multi-part objects (Section 9), a stained-glass composition pipeline that reconstructs came and glass assemblies and can enforce historically grounded constraints (Section 10), and an interactive plaster-turning simulator developed with practitioner input (Section 11).

12.2 Integration conclusions

Across the reported systems, two patterns emerge.

1. Interactivity requires abstraction. Craft simulation becomes interactive when geometry, parameters, and state are chosen to match the structure of the craft (e.g., solids of revolution, limited but meaningful action parameters, and targeted material descriptions). This abstraction is not a loss of rigour; it is the mechanism that makes controlled experimentation feasible.
2. Fidelity is most valuable when it is targeted. High-fidelity FEM is most effective when used as a sparse "truth generator" for the aspects of behaviour that real-time engines cannot express, rather than as a universal runtime. The deliverable's approach therefore treats FEM as a reference layer that informs approximation, calibration, or correction—an engineering compromise that aligns with practical requirements in training and design contexts.

12.3 Limitations

Limitations follow naturally from the problem setting.

- Calibration and parameter identification remain challenging. Whether for FEM, real-time simulation, or learned surrogates, the quality of outcomes depends on how well material properties, boundary conditions, and action parameters match the real workshop situation. In many crafts, these quantities are difficult to measure directly, and practitioners' tacit adjustments may not be captured by simple parameter sets.
- Coverage of the action space is necessarily incomplete. Even with sparse FEM sampling and surrogate learning, there is a limit to how broadly a model can generalise without additional data, especially in regimes with discontinuities (fracture, chipping, complex contact, or process-dependent material state changes).
- Validation is inherently multi-criteria. "Correctness" in craft simulation is not only geometric or mechanical; it also concerns perceptual plausibility and usability. Some reported systems (notably plaster turning) address this through iterative prototyping and practitioner feedback, but systematic benchmarking across all utilities is an ongoing requirement rather than a completed outcome.

12.4 Future work and exploitation directions

The work reported here establishes a platform that can be extended along several axes:

1. Broader craft coverage and action taxonomies. Additional craft actions (e.g., metalworking, textile processes, advanced joinery) can be incorporated by reusing the same semantic → physics → appearance pipeline, provided that action parameters and validation criteria are carefully defined.
2. Improved surrogate modelling and uncertainty awareness. Learning-based approximations will benefit from explicit uncertainty estimates and from mechanisms that detect when a query lies outside the surrogate's reliable domain, triggering either conservative behaviour or targeted new FEM sampling.
3. Stronger measurement-driven calibration. Integrating workshop measurements (geometry scans, force/trajectory traces, material characterisation, and optical measurements where relevant) would reduce ambiguity in parameter identification and improve comparability across sites.
4. Usability-focused tooling and authoring workflows. Wider adoption depends on lowering the cost of setup (scene authoring, parameter selection, and result interpretation). Craft-specific

interfaces, templates, and documentation—especially those co-designed with practitioners—are likely to yield the highest impact.

5. Sustained interoperability and preservation. Ensuring that semantic structures, datasets, simulation configurations, and resulting assets remain discoverable and preservable will be important for long-term reuse, including educational deployment and integration with broader cultural-heritage infrastructures.

In conclusion, this deliverable demonstrates that craft-specific simulation can be treated as an integrated engineering problem—combining semantic traceability, targeted physical fidelity, and perceptually grounded visualisation—rather than as a sequence of isolated demonstrations. The reported components provide a practical basis for continued development, extension to additional crafts, and the creation of robust, reusable simulation-driven tools for craft design, learning, and documentation.

Annex A. Visualisation Toolbox

This annex documents the visualisation toolbox used throughout the deliverable to assemble renderable scenes and produce consistent visual outputs (single images and videos). The toolbox is designed to minimise scene-setup overhead while retaining physically based control over geometry, materials, lighting, and camera settings.

A.1 Objects

A scene is specified by populating an **`object's`** array. Each entry describes a 3D object and (optionally) its material assignment. Objects can be added in two ways:

- **Per-object** specification. Each object is listed explicitly by providing the path to a mesh file in the **`filename`** field and setting its material options individually.
- **Batch** specification (shared material). Multiple objects that share the same material can be added by providing a path to a folder of mesh files in the **`filename`** field. If the selected material supports colour, an optional **`.txt`** file may be provided to assign an RGB colour per object.

Objects are placed according to the coordinates defined in their mesh files. The toolbox supports Wavefront OBJ and Stanford PLY formats.

To simplify scene authoring, the toolbox can optionally add two auxiliary primitives:

- **Floor** (ground plane). When enabled, a rectangular ground plane is placed automatically. Its position is computed from the camera pose and the centre of the bounding box that contains the scene objects, ensuring a stable "resting" surface without manual tuning.
- **Background plane**. When enabled, a planar "wall" is placed behind the objects. Its position is computed in the same way (camera pose and scene bounding box centre), providing a simple, controllable background for consistent renders.

These auxiliary objects are optional and can be disabled when a fully custom scene is desired.

A.2 Materials

The appearance of materials is defined via Bidirectional Scattering Distribution Functions (BSDFs) [17], i.e., functions that describe how light is scattered at a surface (and, where applicable, within a medium). In principle, any material can be simulated given an appropriate BSDF. For convenience, the toolbox provides a set of predefined material presets that cover the most frequent use cases in this deliverable:

- **Glass** and **rough glass**, with an optional colour parameter, for solid dielectric glass objects.
- **Thin glass**, with an optional colour parameter, for hollow (shell-like) glass objects.
- **Plastic** and **rough plastic**, with an optional colour parameter.
- **Metal** and **rough metal**. Metals are supported by acquiring their BSDFs from [18]. If a specific alloy is needed, its BSDF must be provided as input.

To illustrate these presets, the figure below renders the same reference test object under a consistent lighting setup while varying (i) material family and (ii) surface micro-roughness (where relevant).

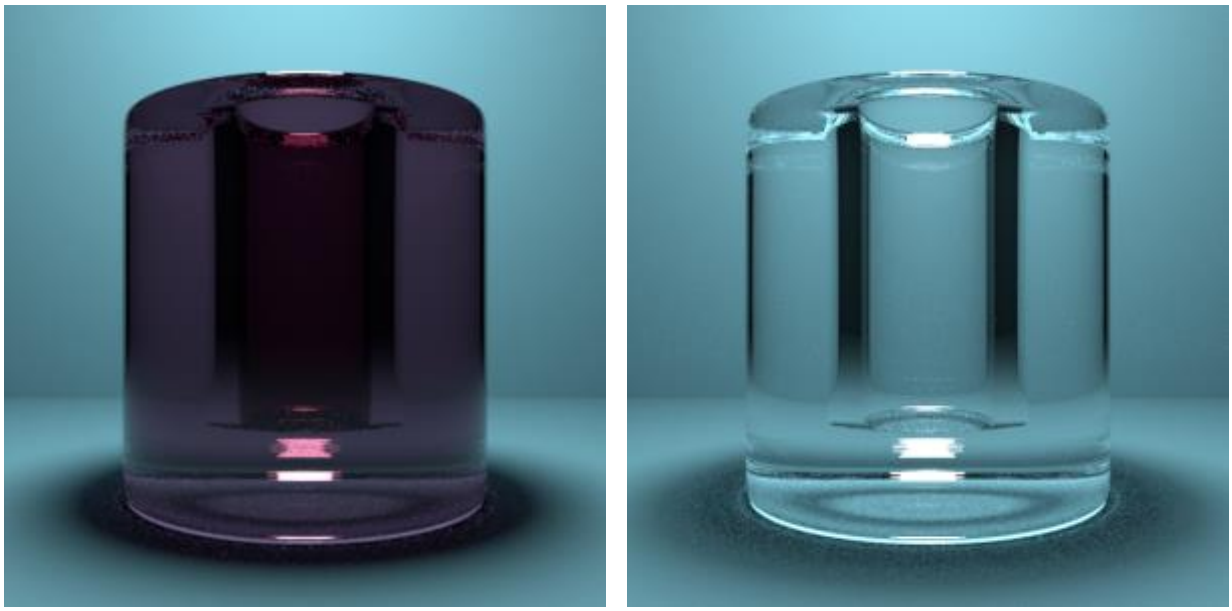


Figure 112. Smooth glass (solid dielectric). Reference test object rendered as (left) coloured glass and (right) colourless transparent glass.

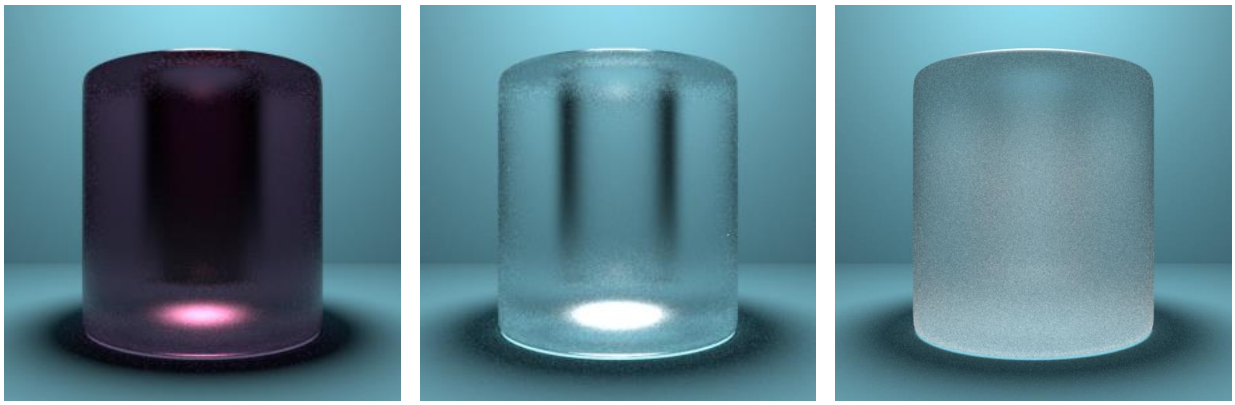


Figure 113. Rough glass (solid dielectric). Increasing surface micro-roughness from left to right, showing the transition from sharp refraction/specular reflection to progressively more diffused appearance.

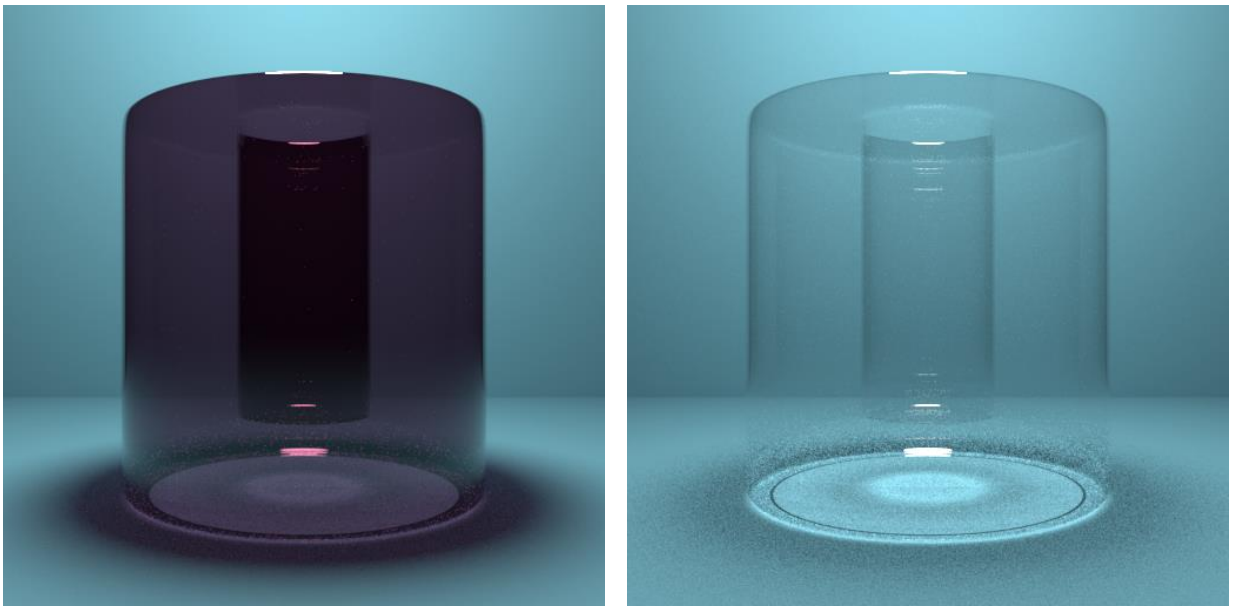


Figure 114. Thin glass (hollow dielectric). Reference test object rendered as (left) coloured thin glass and (right) colourless thin glass, illustrating the shell-like interpretation used for hollow objects.

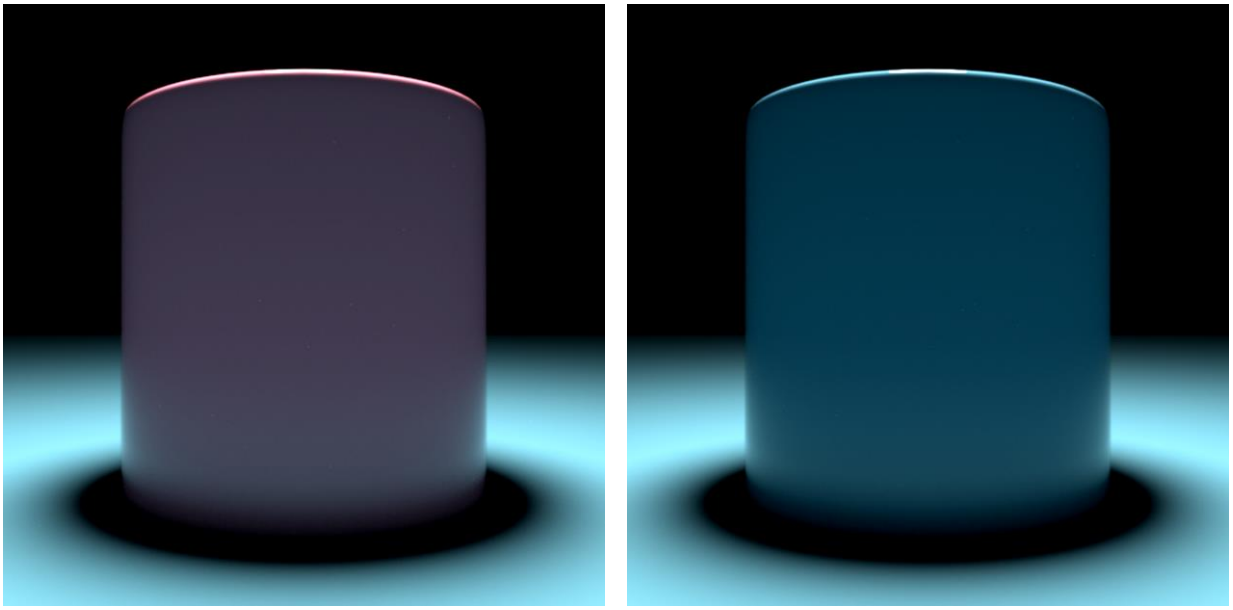


Figure 115. Smooth plastic. Reference test object rendered with the plastic preset as (left) colourless/neutral plastic and (right) coloured plastic.

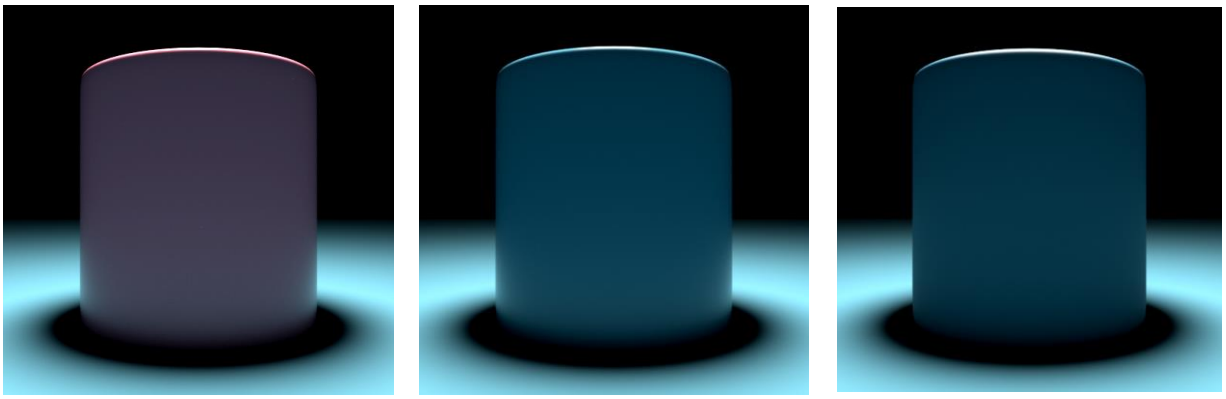


Figure 116. Figure 108. Rough plastic. Increasing surface micro-roughness from left to right, demonstrating broadening highlights and more diffuse reflections.

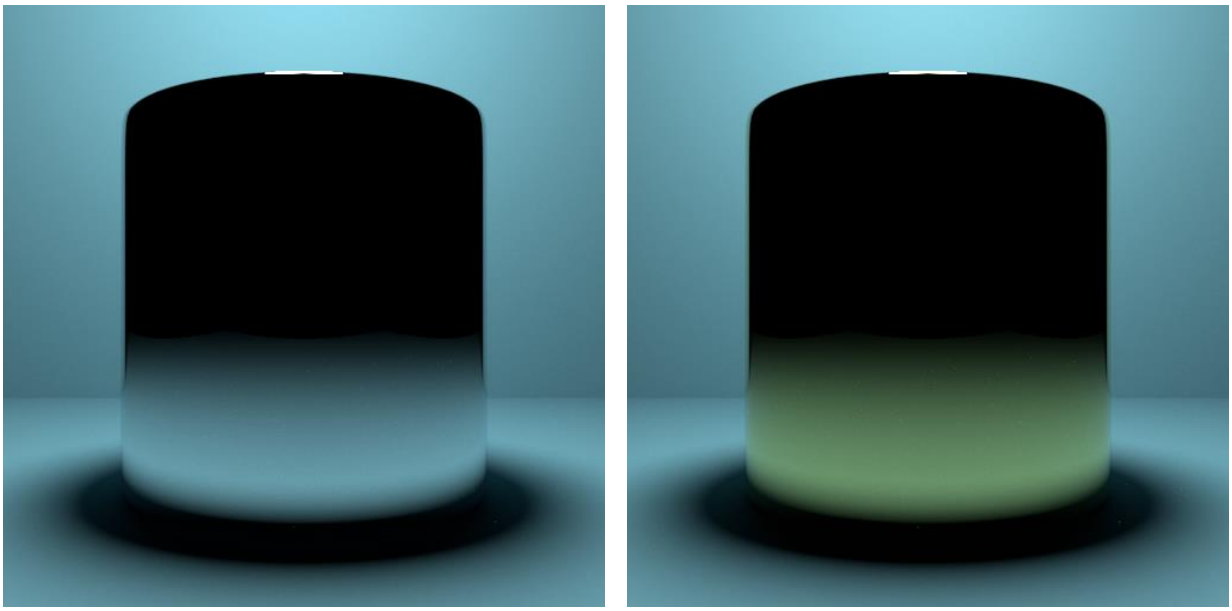


Figure 117. Smooth metal (conductor). Reference test object rendered as a reflective metal under the same scene illumination, illustrating the dominance of environment and scene reflections in conductor appearance.

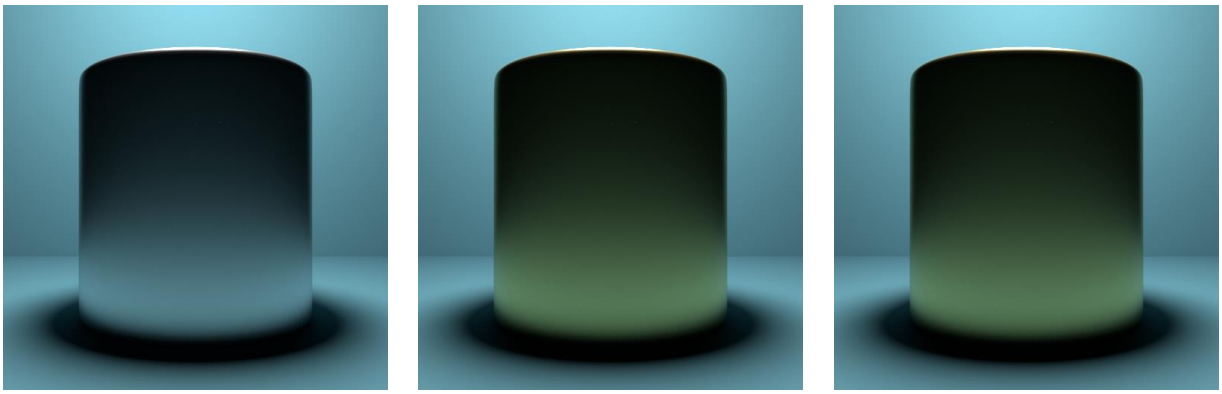


Figure 118. Rough metal (conductor). Increasing surface micro-roughness from left to right, producing progressively blurrier reflections and a more "brushed/matte" metallic look.

Two modelling distinctions are particularly important for the presets above:

1. **Solid versus thin (hollow) glass.** Glass is transparent/translucent, and hollow objects are often better represented as a thin dielectric interface embedded in another dielectric (e.g., glass surrounded by air). In the thin-glass model, the interior is assumed sufficiently thin that transmitted rays are not significantly deflected by a volumetric path, although specular reflection at the interface remains visible. This produces a practical approximation for shell-like glass while avoiding the complexities of explicitly modelling inner/outer surfaces in some workflows.
2. **Smooth versus rough surfaces.** Roughness controls how scattering spreads around the ideal specular direction. Smooth surfaces concentrate scattering into a narrow lobe (sharp reflections/refractions), whereas rough surfaces broaden the lobe (blurred highlights, more diffuse appearance). Where supported by the underlying BSDF, internal scattering parameters can also be adjusted (see the user's manual on the toolbox website) to model translucent behaviour in which a portion of energy remains within the material before exiting.

A.3 Lighting

Lighting is required for the simulation to produce visible surfaces. The toolbox supports two lighting modes:

1. **Explicit light sources**. Area lights with user-defined size, colour, pose (position/orientation), and radiance. (The toolbox supports common analytic shapes, including ellipsoidal and box-like emitters.)
2. **Environment illumination**. An ambient lighting field defined by a 360° spherical image (HDRI). HDR environments can be rotated arbitrarily and scaled uniformly in radiance. Such spherical images can be captured with 360° cameras or obtained from online HDR vendors.

Figure 119 compares typical lighting configurations used in this deliverable and illustrates their practical effect on appearance (especially for specular materials).

Figure 119. Lighting configurations supported by the toolbox. (Left) scene illuminated using prescribed area light sources; (middle) scene illuminated using an HDR environment map; (right) example combining environment illumination with additional explicit lights.

An arbitrary number of light sources can be added to a scene, and explicit lights can be combined with HDR environments when both directional control and realistic ambient illumination are desired.

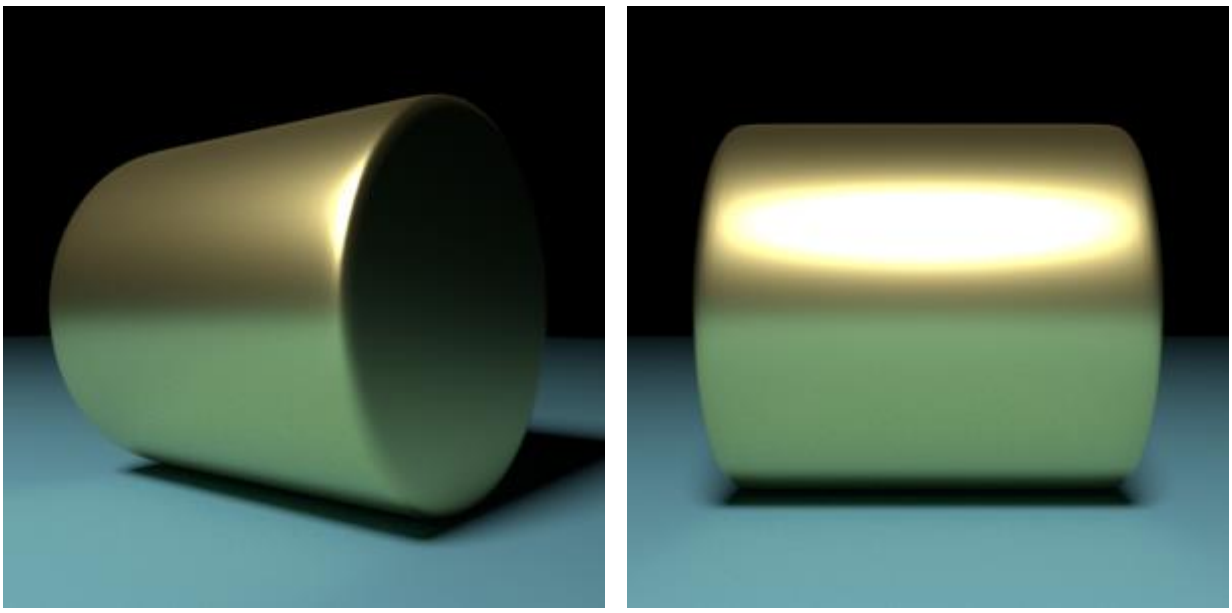
A.4 Viewpoint

Once geometry, materials, and lighting are defined, the rendered viewpoint is set by specifying conventional camera parameters:

- Extrinsics: camera pose (position and orientation).
- Intrinsics: field of view and output resolution.

In addition, the toolbox exposes a sampling parameter that controls the number of samples used by the rendering integrator. Rendering solves a high-dimensional numerical integration problem over geometry, materials, lights, and sensor response. Increasing the sample count typically reduces noise and improves image quality, at the cost of longer render time.

Figure 120 shows the same object rendered from two different camera viewpoints, illustrating how pose changes affect perceived shape and highlight structure.



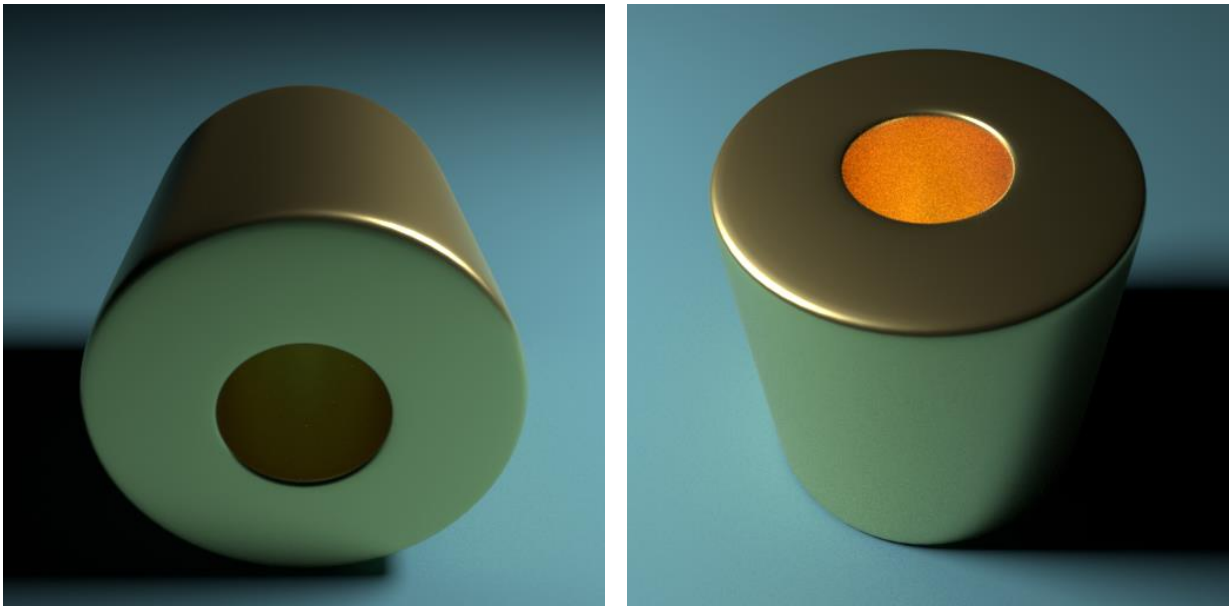


Figure 120. Viewpoint control. A metallic cylinder rendered from two different camera poses within the same scene configuration.

A.5 Outputs

The toolbox supports three output types:

1. Single image. Render a still image from the defined camera viewpoint.
2. Rotational video. Render a camera orbit about a user-defined axis, with specified step and angular range, and export an AVI video. This is useful for presenting artefacts from multiple views.
3. Animation video. Render a sequence of scenes (provided as separate 3D files) and export a video of the resulting animation. This is useful for visualising simulation results with consistent materials and lighting.

A.6 Execution

Execution can be selected according to available hardware:

1. Serial CPU execution.
2. Parallel CPU execution (multi-core).
3. Parallel GPU execution (graphics card).

These options are ordered by typical speed (slowest to fastest). Actual runtime depends on the specific scene complexity, sample count, and hardware capabilities.

A.7 Reference glasses

To evaluate tinted-glass behaviour in a controlled and repeatable way, we use a small set of reference SCHOTT optical filter glasses. These materials are well characterised, and therefore provide convenient known points for verifying that the rendering pipeline reproduces plausible colouration as a function of illumination, view direction, and glass thickness.

This annex documents the reference glass set used in our controlled render studies. The set spans (i) clear / weakly tinted glasses to test baseline absorption, and (ii) filter glasses (bandpass and longpass/edge filters) to test wavelength-dependent attenuation and its interaction with thickness and illumination spectrum. For each reference glass, the accompanying figure shows the same geometry rendered in solid mode (left) and hollow mode (right), to expose how a cavity and shell thickness modulate apparent colour and brightness under otherwise identical conditions.

Table 21. Reference glass set (SCHOTT catalogue items).

Glass	SCHOTT classification (datasheet)	Practical role in our toolbox
BG38	Bandpass / shortpass; NIR cut-off oriented	Low absorption clear-ish reference within the filter-glass family; baseline for thickness-driven darkening and NIR sensitivity.
BK7	Optical glass (boron crown), n_d ≈ 1.5168, v_d ≈ 64.17	Canonical *clear optical glass* reference for solid vs hollow comparisons and for checking that geometry/scale and IOR handling do not introduce spurious colour.
UG11	Bandpass filter (UV-transmitting filter glass)	High absorption / UV-pass reference: strongly thickness-sensitive; useful for diagnosing whether the pipeline preserves expected extinction trends.
BG3	Bandpass / shortpass filter; noted as “sensitive glass”	Cool-tinted UV–blue bandpass reference; strongly illuminant- and thickness-dependent.
VG20	Bandpass filter; explicitly described **as an** NIR cut-off filter	Green bandpass / NIR-cutoff reference for tests under illuminants with non-trivial NIR energy.
GG495	Longpass (edge) filter; 50% cut-on at ~495 nm	Yellow longpass reference; mid-spectrum edge case for thickness-dependent saturation.
OG530	Longpass filter; 50% cut-on at ~530 nm	Amber/orange longpass reference; complementary to GG495 with a more aggressive cut-on.
RG610	Longpass filter; 50% cut-on at ~610 nm	Deep red longpass reference; highly thickness-sensitive and useful for “solid vs hollow” consistency checks.

The figures below provide catalogue samples for these glasses.

Figure 121. UG11

Figure 122. BG3

Figure 123. VG20.

Figure 124. BG38.

Figure 125. RG610.

Figure 126. OG530.

Figure 127. GG495

Annex B. Training Data Production

A first investigation was performed to qualitatively extend the generated data by making them both material-specific and temperature-dependent. We present a preliminary study on how we intend to treat this task in our next steps.

In this case, we are studying the plasticity of metals concerning temperature. The objective of the simulation study is to analyse the mechanical behaviour of aluminium under varying temperatures and mechanical loading conditions. Specifically, the study investigates how temperature affects the stress and pressure distribution within an aluminium rectangular parallelepiped (workpiece) subjected to mechanical pressure. In the simulation, an imprinting (debossing) tool is moved along the Y-axis to apply pressure on the workpiece, which is positioned on top of a rigid ground plane. The tool has the shape of a parallelepiped.

The focus is on analysing the plastic properties of the workpiece at a displacement of 0.001 m and 0.0015 m and three temperatures: 20°C, 300°C, and 660°C. The simulation model includes gravity as a fundamental external force acting on the system. The simulation is conducted using Abaqus to observe how temperature influences the stress and pressure distribution within the workpiece.

Following the approach presented in "D2.1. Action and affordance modelling", we specialise the archetypal action for deformation and instantiate the following scene elements:

- Tool: The tool has dimensions of 0.04 m × 0.05 m ×0.04 m.
- Workpiece: The workpiece has dimensions of 0.09 m × 0.09 m × 0.01 m
- Ground Plane: The ground plane is a rigid body that acts as a fixed support for the workpiece and does not deform. The ground plane has dimensions of 0.12 m × 0.12 m × 0.005 m

The tool is made of steel, and the workpiece is from aluminium. The material properties are defined as follows. For the tool, we use the environment temperature (i.e. 20 °C) and its properties are defined at that temperature. The workpiece is heated, and as such, more material properties are needed to describe its expansion due to heat and, most importantly, the change of plasticity and elasticity as a function of material temperature.

As such, for steel, we use the following properties.

- Density: 7850 kg/m3
- Elastic Modulus: 200 GPa
- Poisson's Ratio: 0.3
- Yield Stress: 355 MPa with Plastic strain 0, and Yield Stress: 470 MPa with Plastic strain 0.178

For aluminium, besides density (2700 kg/m^3), all other properties are temperature-dependent and provided in Table 22, Table 23, and Table 24.

Table 22. Young modulus and Poisson ratio of Aluminium as a function of temperature.

Young's Modulus (GPa)	Poisson's Ratio	Temperature (°C)

70	0.33	20
69	0.33	100
68	0.33	200
67	0.33	300
66	0.33	400
65	0.33	500
64	0.32	600
63	0.32	700

Table 23. Expansion coefficient of Aluminium as a function of temperature.

Expansion Coefficient ($10\times^{-5}$)	Temperature (°C)
2.3	20
2.5	100
2.6	200
2.7	300
2.9	400
3.2	500
3.5	600
3.7	660

Table 24. Yield stress and plastic strain of Aluminium as a function of temperature.

Yield Stress (MPa)	Plastic Strain	Temperature (°C)
275	0	20
255	0.005	100
230	0.01	200
200	0.015	300
175	0.02	400
150	0.025	500
120	0.03	600

In the simulation, the debossing moves in the Y-axis with displacements of 0.001 m and 0.0015 m. The simulations are conducted at three district temperatures for each displacement: 20°C, 300°C, and 660°C. Gravity is implemented in the simulation to represent the effects of the gravitational force on the parts. The magnitude of gravity is set according to standard gravitational acceleration, typically 9.8 m/s^2. The element type used is C3D8R (an 8-node linear brick, reduced integration).

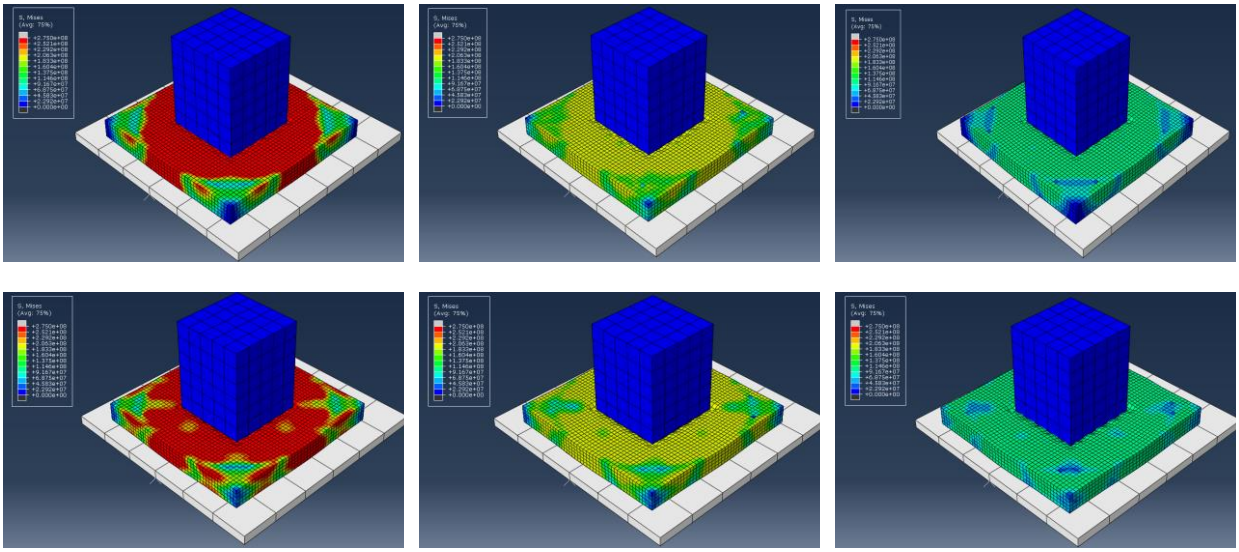


Figure 128. Von Mises Stress spatial distribution on the workpiece at three different temperatures: 20°C (left), 300°C (middle), and 660°C (right), for two displacements of 1mm (top) and 1.5mm.

In Figure 128, Figure 19stress distribution on the workpiece is depicted for three different temperatures: 20°C, 300°C, and 660°C and the two displacement values. The colour bar located at the top left of Figure 19 indicates the stress values, with higher stresses represented by colours near red and lower stresses represented by colours near blue or green. As the temperature increases, the colour of the workpiece shifts towards green, indicating that the stress within the aluminium decreases at higher temperatures.

Observations:

- 20°C: The pressure is highly concentrated around the contact region with sharp pressure gradients.
- 300°C: The pressure distribution at 300°C is more uniform compared to 20°C. At this temperature, the aluminium becomes more flexible, so the pressure spreads out more evenly. Although the pressure is still high near the contact area, it decreases in intensity, with more areas showing light blue colours.
- 660°C: The pressure distribution is significantly more uniform and spread out across the surface. The workpiece shows an even greater reduction in pressure values, with a predominance of light blue colour. This indicates a decrease in pressure intensity, as the material deforms more easily under the applied load.

Across all three temperatures, pressure waves are created over the surface of the workpiece. These waves are indicative of the distribution and transmission of pressure from the point of contact where the debossing applies the load. The waves propagate outward from the contact area, showing how the pressure is dispersed across the plate. The simulation results for 0.0015 m demonstrate that the fundamental observations regarding stress and pressure distributions in the workpiece remain consistent across different displacements. With greater displacement (0.0015 m), the magnitude of pressure increases.

As the temperature increases, the overall colour of the workpiece shifts towards light blue. This change illustrates the decrease in pressure intensity, which is a direct result of the reduction in the yield stress and increased plasticity of aluminium at higher temperatures.

The stress distribution results demonstrate a clear influence of temperature on the von Mises stress within the workpiece. As the temperature increases from 20°C to 660°C, the stress values decrease significantly. This stress reduction is due to the decrease in the yield stress of aluminium and the increased plasticity at higher temperatures.

The observations of the pressure distribution results demonstrate the formation of pressure waves across the aluminium surface and the increase in pressure with greater displacement of the debossing. Additionally, the influence of temperature is evident, with higher temperatures leading to a more uniform and widespread pressure distribution and an overall decrease in pressure intensity.

Annex C. Porcelain states

The states are defined not only by craft semantics (“glazed”, “celadon”, “opaque”) but also by the level of physical explicitness in the simulation. T1–T2 vary the porcelain body (subsurface scattering). T3–T4 approximate glaze as a surface-only dielectric layer (a BSDF with controllable roughness). T5–T8 model glaze as an explicit thin film above the body (a geometric layer with a participating medium), enabling thickness-dependent absorption and scattering effects.

Eight examples demonstrate the impact of subtle alterations in the top layer on our perception. Starting with bare porcelain, they progress through clear coatings of varying sheens to glazes that either tint or cloud the underlying body.

Table 25. Perceptual definitions.

T1. Biscuit	This is porcelain after its first firing, before any glaze is added. The surface is dry and matte, like a fine eggshell. Light does not just bounce off the top; only a little seeps under the surface and comes back out, which softens shadows and gives a gentle, chalky look.
T2. Parian	Parian, while also unglazed, is designed to mimic the appearance of marble. Light travels a bit further under the surface, so edges look softer, and the object appears glowing from within.
T3. Clear glaze (surface model)	The body is covered with a transparent, glassy coating. That coating makes the highlight crisp and bright, like a faint reflection on polished China, while still letting you see the porcelain beneath.
T4. Satin clear glaze	This is the same clear coating, but the surface is microscopically rougher. The bright point of reflection spreads out and becomes silkier, less mirror-like, more like a satin finish on paint.
T5. Clear glaze (explicit film layer)	This one shows the clear glaze as a separate, thin layer on top of the body. It behaves like a real layer of glass: light enters, travels a short distance, and leaves again.
T6. Celadon Light	Celadon is a clear glaze with a faint green tint. The colour comes from tiny amounts of colouring in the glaze that gently absorb some wavelengths of light. The result is a cool, watery depth while details of the body still show through.
T7. Opaque	The glaze is not fully transparent. It scatters light inside itself, like milk or frosted glass. That softens contrast and mutes what's underneath, producing a creamy, even tone with very gentle highlights.
T8. Celadon Strong	This version of Celadon uses a stronger tint, resulting in increased light absorption within the glaze layer. This creates a deeper, slightly darker colour while preserving the glassy surface.

The table below summarises the physical mechanisms that determine the phenomena mentioned above.

Category	Light transport mechanism	Visual sensation
Unglazed (T1–T2)	Subsurface scattering in the body; weak surface specular	Matte to semi-translucent; body-colour dominant
Transparent glaze, surface model (T3–T4)	Dielectric surface reflection (Fresnel) + roughness-controlled specular lobe	Gloss level changes; highlights sharpen/broaden
Explicit glaze film, clear (T5)	Explicit film + multiple internal bounces + body return	Deeper gloss; thickness-dependent depth cues
Coloured transparent glaze (T6, T8)	Absorption in the glaze film (participating medium)	Green tint with optical depth; saturation increases with absorption/thickness
Opaque scattering glaze (T7)	Strong volumetric scattering in glaze film	Milky/creamy appearance; suppressed detail beneath

The appearance of a surface, ranging from chalky and matte to deep, glossy, and luminous, is determined by how it influences the probability distribution of photon path lengths and the angular redistribution of emergent radiance. This concept is demonstrated across eight states, which progressively simulate the radiative transport space. These states vary from a purely diffusing medium, such as unglazed ceramic, to a strongly refractive and absorptive film, like a coloured glaze.

Table 26. Simulation mechanisms.

T#	Optical and light transport mechanisms
T1	The unglazed porcelain acts as a diffuse subsurface scatterer. Light enters the porous ceramic surface, undergoes multiple scattering events in the fine-grained kaolin-feldspar-quartz matrix, and exits with a Lambertian-like distribution. The residual porosity and open microcracks introduce microcavity scattering, which randomises photon paths. The chalky, matte look is due to increased backscattering, which is a result of the significant difference in refractive index between the air within the pores and the ceramic grains. Absorption is weak but nonzero, slightly warming the reflected light.
T2	The Parian body is more vitrified and less porous, leading to longer photon mean free paths and a subsurface glow reminiscent of marble. Even with a more optically uniform effective medium due to reduced porosity, scattering remains significant, albeit with a forward bias. The result is a soft translucency where illumination penetrates deeper and diffuses volumetrically before re-emerging. This volumetric subsurface scattering gives the surface a warm, waxy luminance and suppresses sharp surface reflections.
T3	The clear glaze introduces a dielectric interface with strong Fresnel reflection and partial transmission into the glaze. The glaze itself is optically smooth ($\alpha \approx 0.05$) and weakly absorbing (or effectively non-absorbing in the visible band). When light strikes the surface at a shallow angle, it creates clear, mirror-like reflections (specular highlights) at the boundary between the air and the glaze. Light that passes through this boundary, however, bends as it enters the glaze, interacts with the porcelain underneath, and then scatters back out as diffuse light. The combined effect is a gloss over a diffuse substrate; the visual signature of a thin dielectric layer over a scattering base. In this state, the glaze is treated as a surface interface (no explicit thickness).

T4	The satin finish results from microroughness of the glaze surface or dispersed crystalline phases (e.g., zircon, quartz). The expanded specular lobe and subsurface forward scattering within the glaze reduce the coherence of Fresnel reflections. This shifts the energy balance from specular to diffuse components, resulting in a soft lustre. Angular microfacets randomise local normal, giving a velvety sheen where highlights are broader and less intense.
T5	Here, the optical system comprises three interfaces: air–glaze, glaze–body, and body interior. Photons may undergo multiple internal reflections within the glaze film (a "shell"), producing light-trapping and thickness-dependent depth cues. The underlying body contributes diffuse backscatter, while the smooth shell preserves directional specularities. The result is a deeper gloss than the surface-only approximation, particularly where the glaze thickness is non-negligible. In this state, the glaze is modelled as an explicit film with thickness.
T6	Celadon glazes contain Fe^{2+} ions dissolved in the glassy matrix, creating wavelength-selective absorption primarily in the red portion of the spectrum. This selective attenuation, combined with volumetric scattering within the glaze, yields the characteristic blue-green transmission tint. Because the absorption coefficient is low, a significant portion of light penetrates to the body and returns, giving the impression of depth and internal glow. The result is an optically participating medium where absorption and scattering jointly shape a translucent greenish film over the porcelain body.
T7	The milky glaze owes its appearance to Mie scattering by suspended microcrystals (e.g., zirconium silicate, titania). The high refractive index contrast between inclusions and the glassy matrix produces multiple elastic scattering that randomises photon directions, reducing transmission and suppressing subsurface contrast. Because absorption is negligible and scattering albedo is near unity, reflectance is high and diffuse, resembling enamel. Highlights are subdued since specular reflection competes with intense volumetric backscattering. The surface, therefore, appears creamy and dense, with little translucency.
T8	The same mechanism as T6, but with a higher Fe^{2+} concentration or thicker glaze layer, increasing the absorption optical depth. Blue-green light penetrates deeper than red-yellow wavelengths, which are almost completely absorbed. This results in a darker, more saturated glaze, as less light is backscattered to the observer. Because scattering remains moderate, internal reflections within the glaze create subtle light-dark gradients. The perceived effect is a deep, luminous green with lower overall brightness but pronounced chromatic depth.

The visual simulation generates eight types of porcelain conditions. Starting with bare porcelain, they progress through clear coatings of varying sheens to glazes that either tint or cloud the underlying body.

The appearance of a surface, ranging from chalky and matte to deep, glossy, and luminous, is determined by how it influences the probability distribution of photon path lengths and the angular redistribution of emergent radiance. This concept is demonstrated across eight states, which progressively simulate the radiative transport space. These states vary from a purely diffusing medium, such as unglazed ceramic, to a strongly refractive and absorptive film, like a coloured glaze.

T1. Biscuit	This is porcelain after its first firing, before any glaze is added. The surface is dry and matte, like an eggshell. Light does not just bounce off the top; only a little seeps under the surface and comes back out, which softens shadows and gives a gentle, chalky look.
T2. Parian	Parian, while also unglazed, mimics the appearance of marble. Light travels a bit further under the surface, so edges look softer, and the object appears to glow from within.
T3. Clear glaze	The body is covered with a transparent, glassy coating. That coating makes the highlight crisp and bright, like a faint reflection, while still letting the porcelain beneath visible.
T4. Satin clear glaze	Same clear coating as T3, but the surface is rougher. Reflections spread out and become silkier, less mirror-like, and more like a satin finish on paint.
T5. Clear glaze	Shows the glaze as a separate, thin layer on top of the body. Behaves as a layer of glass: light enters, travels a short distance, and leaves again.
T6. Celadon Light	Celadon is a clear glaze with a faint green tint. Its colour is due to small amounts of colouring in the glaze that absorb some wavelengths of light. The result is a cool, watery depth while details of the body are still visible.
T7. Opaque	The glaze is not fully transparent, but scatters light inside itself, like milk or frosted glass. That softens contrast and mutes what is underneath, producing a creamy and even tone with gentle highlights.
T8. Celadon Strong	This version of Celadon uses a stronger tint, resulting in increased light absorption within the glaze layer. This creates a deeper and darker colour while preserving the glassy appearance of the surface.

Each material state is actualised as a distinct Bidirectional Scattering Distribution Function (BSDF) configuration, thereby illustrating the microstructural evolution of porcelain throughout its creation. Annex B describes the optical model used for each state and justifies its selection.

Annex D. Abaqus definitions

Table 27. FEM definition of a soda-lime glass as a thermorheologically simple material.

```
*MATERIAL, NAME = Glass_SodaLime

*DENSITY 2500.0,

*ELASTIC, MODULI=INSTANTANEOUS 70.0E9, 0.23

** WLF SHIFT FUNCTION (The Semantic Constraint)

*TRS, DEFINITION = WLF

** Ref Temp (Theta_0), C1, C2 550.0, 17.44, 51.6
```

Table 28. Marvering: Contact Mechanics & Conductive Cooling.

```
*STEP, NAME=Marver_Roll, NLGEOM=YES

*COUPLED TEMPERATURE-DISPLACEMENT

** INTERACTION: GLASS TO STEEL (Heat Transfer)

*SURFACE INTERACTION, NAME=Steel_Contact

*GAP CONDUCTANCE

 2000.0, 0.0

 0.0, 0.001

** KINEMATICS (Movement)

** PSO: Rigid Body Motion defined in Local CSYS

*TRANSFORM, NSET=Set_Pipe_Control, TYPE=R

 0.866, 0.0, -0.5, -0.5, 0.0, -0.866

*BOUNDARY, TYPE=VELOCITY, AMP=Soft_Pulse

 Set_Pipe_Control, 2, 2, 0.2

 Set_Pipe_Control, 6, 6, 1.5
```

```
*END STEP
```

Table 29. Blowing: Pneumatic Volumetric Expansion.

```
*STEP, NAME=Blow_Expansion, NLGEOM=YES

*COUPLED TEMPERATURE-DISPLACEMENT

** ANCHOR (Hold Pipe Steady)

*BOUNDARY, TYPE=ENCASTRE

 Set_Pipe_Control

** GAS DYNAMICS (Breath)

*AMPLITUDE, NAME=Breath_Ramp

 0.0, 0.0,  1.5, 1.0,  3.0, 0.0

*DSLOAD, AMPLITUDE=Breath_Ramp

** Apply pressure (Pa) to the internal surface

 Surf_Glass_Inner, P, 5000.0

*END STEP
```

Table 30. Soufflé tourné (Rotational Mould Blowing): Constrained Volumetric Expansion.

```
*STEP, NAME=Souffle_Tourne, NLGEOM=YES

*COUPLED TEMPERATURE-DISPLACEMENT

** KINEMATICS (Rotate Pipe while holding position)

*BOUNDARY, TYPE=VELOCITY

** Fix Position (1,2,3), Rotate around Z (6)

 Set_Pipe_Control, 1, 3, 0.0

 Set_Pipe_Control, 6, 6, 1.5

** MOULD INTERACTION
```

```
*SURFACE INTERACTION, NAME=Mould_Friction
*FRICTION
 0.1,
*CONTACT PAIR, INTERACTION=Mould_Friction
 Surf_Glass_Outer, Surf_Mould_Inner
** PNEUMATIC LOAD
*DSLOAD
 Surf_Glass_Inner, P, 8000.0
*END STEP
```

Table 31. Blocking (Block Shaping): Hydro-Thermal Shaping & Pneumatic Expansion.

```
*STEP, NAME=Blocking_Step, NLGEOM=YES
*DYNAMIC, EXPLICIT
** Time period roughly 5.0s for the operation
, 5.0
** KINEMATICS (Rotation on Rail)
*BOUNDARY, TYPE=VELOCITY
** Rotate around Pipe Axis (VR3)
 Set_Pipe_RP, 6, 6, 3.14
** CONTACT (Glass to Wet Block)
*SURFACE INTERACTION, NAME=Steam_Lubrication
*FRICTION
** Low friction due to the steam cushion
 0.05,
```

```
*CONTACT PAIR, INTERACTION=Steam_Lubrication, MECHANICAL CONSTRAINT=PENALTY

 Surf_Glass_Outer, Surf_Block_Inner

** PNEUMATIC (Blow-and-Cap)

*AMPLITUDE, NAME=Cap_Pressure

** Time, Amp: 0->1 (Blow) -> 0.8 (Cap/Trapped)

 0.0, 0.0,  0.5, 1.0,  5.0, 0.8

*DSLOAD, AMPLITUDE=Cap_Pressure

 Surf_Glass_Inner, P, 3000.0

** GRAVITY

*DLOAD

 All_Elements, GRAV, 9.81, 0.0, 0.0, -1.0

*END STEP
```

Table 32. Necking. Localised Compression & Axial Elongation.

```
*STEP, NAME=Necking_Step, NLGEOM=YES

** KINEMATICS (Pipe Rotation)

*BOUNDARY, TYPE=VELOCITY

** Rotate Pipe (VR3)

 Set_Pipe_RP, 6, 6, 3.14

** KINEMATICS (Jacks Constriction)

** Jacks move inward (Prescribed Displacement)

*BOUNDARY, TYPE=DISPLACEMENT, AMP=Ramp_Inward

** Move Jack 1 towards the centre

 Set_Jack_Top, 2, 2, -0.02
```

```
** Move Jack 2 towards the centre

 Set_Jack_Bottom, 2, 2, 0.02

** CONTACT (Dry Metal Friction)

*SURFACE INTERACTION, NAME=Jack_Friction

*FRICTION

** High friction coefficient

 0.5,

*CONTACT PAIR, INTERACTION=Jack_Friction, MECHANICAL CONSTRAINT=PENALTY

 Surf_Glass_Outer, Surf_Jacks

*END STEP
```

Table 33. Rim Flaring. Radial Expansion & Centrifugal Shaping.

```
*STEP, NAME=Opening_Lips, NLGEOM=YES

*DYNAMIC, EXPLICIT

, 1.0

** KINEMATICS (Punty Rotation)

*BOUNDARY, TYPE=VELOCITY

** Rotate Punty (VR3)

 Set_Punty_RP, 6, 6, 4.0

** KINEMATICS (Jacks Spreading)

*BOUNDARY, TYPE=DISPLACEMENT, AMP=Ramp_Outward

** Move Jack Tips Outward (Radial Expansion)

 Set_Jack_Tips, 1, 1, 0.05

** CONTACT (High Friction Shaping)
```

```
*SURFACE INTERACTION, NAME=Jack_Grip
*FRICTION
 0.5,
*CONTACT PAIR, INTERACTION=Jack_Grip, MECHANICAL CONSTRAINT=PENALTY
 Surf_Glass_Rim, Surf_Jacks
*END STEP
```

Annex E. Woodturning process

E.1 Semantic model

Process Schema: Wood Turning Virtual Process Schema. CRAEFT Resource URI.

Table 34. Wood Turning (Virtual Process).

START

-> Preparing (schema placeholder; no actions, only start/end: wt_v_pe_2)

-> Roughing out

-> Bark removal (wt_s_p_8): wt_v_a_8 -> wt_v_a_9 -> wt_v_a_10 (observe)

-> Left-end (wt_s_p_9): wt_v_a_11 (rotate) -> wt_v_a_12 (smooth) -> wt_v_a_13 (observe)

-> Thinning (wt_s_p_10): wt_v_a_14 -> wt_v_a_15 -> wt_v_a_16

-> Decision 1 (Length?) (wt_v_dt_1)

IF too long **THEN**

-> Shortening

-> One ridge making (1st) (wt_s_p_11): wt_v_a_17 -> wt_v_a_18 -> wt_v_a_19

-> One ridge making (2nd) (wt_s_p_12): wt_v_a_20 -> wt_v_a_21 -> wt_v_a_22

-> (merge: wt_v_mt_1)

ELSE (not too long) -> (skip shortening; merge: wt_v_mt_1)

-> Decision 2 (Thickness?) (wt_v_dt_2)

IF too thick **THEN**

-> Reducing width (process start labelled:wt_s_p_5)

-> Thinning (wt_v_p_14): wt_v_a_23 -> wt_v_a_24 -> wt_v_a_25

-> Smoothing (wt_s_p_15): wt_v_a_26 -> wt_v_a_27 -> wt_v_a_28

-> Carving (wt_v_p_16): wt_v_a_29 -> wt_v_a_30 -> wt_v_a_31

-> Thinning (wt_v_p_17): wt_v_a_32 -> wt_v_a_33 -> wt_v_a_34

-> (merge: wt_v_mt_2)

ELSE (thickness ok) -> (skip reducing width; merge: wt_v_mt_2)

-> Making picker head: wt_v_a_41 -> wt_v_a_42 -> wt_v_a_43

-> Making picker handle (wt_v_p_6)

-> Thinning (pass 1) (wt_v_p_18): wt_v_a_35 -> wt_s_a_36 -> wt_s_a_37

-> Thinning (pass 2) (wt_v_p_19): wt_v_a_38 -> wt_s_a_39 -> wt_s_a_40

-> Refining picker head: wt_v_a_44 -> wt_s_a_45 -> wt_s_a_46

-> Polishing (wt_v_p_7)

-> Brushing: wt_v_a_47 -> wt_v_a_48 -> wt_v_a_49

-> Sand-papering: wt_v_a_50 -> wt_v_a_51 -> wt_v_a_52

-> Clothing: wt_v_a_53 -> wt_v_a_54 -> wt_v_a_55

-> Sawdusting: wt_v_a_56 -> wt_v_a_57 -> wt_v_a_58

-> **LOOP**: Extracting

Repeat 4x: 'Extracting the picker head' (transitions wt_v_st_34 -> wt_v_st_37)

-> Finishing

END

E.2 Annotations

Table 35. Real and simulated actions.

Recording	Simulation
Woodturning_video_02	Woodturning_simulated_video_02_associated_with_real_video_02
Woodturning_video_03	Woodturning_simulated_video_03_associated_with_real_video_03

Woodturning_video_04	Woodturning_simulated_video_04_associated_with_real_video_04
Woodturning_video_05	Woodturning_simulated_video_05_associated_with_real_video_05
Woodturning_video_06	Woodturning_simulated_video_06_associated_with_real_video_06
Woodturning_video_07	Woodturning_simulated_video_08_associated_with_real_video_07
Woodturning_video_08	Woodturning_simulated_video_09_associated_with_real_video_08
Woodturning_video_09	Woodturning_simulated_video_10_associated_with_real_video_09
Woodturning_video_11	Woodturning_simulated_video_11_associated_with_real_video_11
Woodturning_video_12	Woodturning_simulated_video_12_associated_with_real_video_12
Woodturning_video_13	Woodturning_simulated_video_13_associated_with_real_video_13
Woodturning_video_14	Woodturning_simulated_video_14_associated_with_real_video_14

These videos document the master artisan's performance, capturing the fluid motion and tactile feedback of the craft.

Asset ID	Process Stage	Ontological ID	Repository Link
Video 02	Initial Preparation	wt_v_p_2	Access Video 02
Video 03	Roughing Out	wt_v_p_3	Access Video 03
Video 04	Shortening (Decision)	wt_v_p_4	Access Video 04
Video 06	Handle Construction	wt_v_p_6	Access Video 06
Video 08	Bark Removal	wt_v_p_8	Access Video 08
Video 11	First Ridge Detailing	wt_v_p_11	Access Video 11
Video 13	Cutting / Parting	wt_v_p_13	Access Video 13
Video 15	Brushing Phase	wt_v_p_15	Access Video 15
Video 18	Sawdusting Phase	wt_v_p_18	Access Video 18

These assets represent the digital efforts to mimic real actions, documenting the physics of the process.

Asset ID	Simulation Target	Repository Link

Sim Video 02	Prep Simulation	Access Sim 02
Sim Video 04	Length Adjustment	Access Sim 04
Sim Video 08	Bark Impact Physics	Access Sim 08
Sim Video 11	Ridge Stress Analysis	Access Sim 11
Sim Video 13	Parting Cut Physics	Access Sim 13

E.3 Semantic organisation

Phase 1: Shaping (Risk)

Judgement & Dexterity: Outcome is uncertain; requires constant artisan care to handle material variance.

Step	Sub-Process ID	Description	Recording	Simulation
02	wt_v_p_2	Initial Preparation & Mounting	Real	Sim
08	wt_v_p_8	Bark Removal (Roughing)	Real	Sim
09	wt_v_p_9	Left-end Smoothing	Real	Sim

Phase 2: Sizing (Judgement)

Step	Sub-Process ID	Description	Recording	Simulation
04	wt_v_p_4	Shortening (If too long)	Real	Sim
13	wt_v_p_13	Cutting / Parting Cut	Real	Sim
11	wt_v_p_11	Detailing: One Ridge Making	Real	Sim

Phase 3: Certainty (Finishing)

Focus: Care & Attention. Standard Operating Procedure: the result is predetermined by following the sequence.

Step	Sub-Process ID	Technique	Recording	Simulation
15	wt_v_p_15	Brushing	Real	

16	wt_v_p_16	Sand-papering	Real
18	wt_v_p_18	Sawdusting (Friction Polish)	Real

Remainder carving non-actions (idle):

- Woodturning_video_01
- Woodturning_video_10
- Woodturning_video_15
- Woodturning_video_16
- Woodturning_video_17
- Woodturning_video_18
- Woodturning_video_19
- Woodturning_video_20
- Woodturning_video_21

Annex F. ShellGen Manual

Chapter 0 – Introduction.

Distribution and licensing

ShellGen is intended to be released as open-source software and publicly archived (e.g., via Zenodo). It was developed within the Craeft project and is intended to be openly shared with the wider research and practice community. The definitive licensing terms are those distributed with the software repository (LICENCE file) and the archival record (DOI metadata), which take precedence if they differ from earlier drafts of this manual.

How to use this manual

This manual is written for readers who need to operate ShellGen reproducibly and to generate auditable evidence bundles for technical reporting (deliverables, papers, internal reviews).

A recommended workflow is:

1. Validate the asset (Chapter 2) before attempting final renders.
2. Select the regime (bulk vs shell) (Chapter 3).
3. Choose a preset intent (clear/tinted/frosted, etc.) and record it in the run identifier (Chapter 4).
4. Use the canonical lighting and camera protocol for comparability (Chapter 5).
5. Produce an evidence package and cite it using a stable Run ID (Chapter 6).

Dataset

The example demonstrations in this manual use the Stanford bunny dataset (Stanford 3D Scanning Repository). The dataset page and license/usage terms are the authoritative reference. Repository: https://graphics.stanford.edu/data/3Dscanrep/

Glossary

Term	Meaning in this manual
BULK	A solid dielectric object represented by a single closed boundary (no inner cavity surface).
SHELL	A dielectric shell is represented by an outer and inner boundary enclosing a cavity region (typically air).
Evidence package	The complete set of files emitted by a ShellGen run (plus optional external mesh diagnostics) is sufficient to reproduce and audit the result.
Preset	A named intent category (e.g., clear, tinted, frosted) is used to standardise reporting and comparisons.

Kit	A named, fixed configuration for lighting or camera settings, used to ensure runs are comparable.

Parameter mapping note

This manual uses reader-facing terminology (outer/inner surface, cavity medium, absorption, roughness). The mapping to ShellGen CLI flags is provided in Appendix A.

Scope and Operating Contract

Purpose

ShellGen is a utility for producing reproducible, physically grounded renders of dielectric objects (glass-like materials). Its primary purpose is to support technical communication: each run is designed to generate both a visually interpretable result and the associated configuration artefacts needed to reproduce and audit it.

Assumptions and invariants

- Scale matters. ShellGen does not impose physical units; however, optical effects (especially absorption/tint) are scale-dependent. The operator must ensure the asset scale is plausible and consistent across runs, and that the run record (Chapter 6) captures the relevant settings.
- Mesh validity is a gate. Dielectric behaviour is highly sensitive to normals, manifoldness, and self-intersections. Asset validation (Chapter 2) is mandatory before producing final evidence renderings.
- Determinism policy. A run is considered reproducible if it records (a) the full set of CLI parameters and (b) the random seed used. ShellGen provides an explicit --seed parameter (default 0).

Chapter 1. Conceptual Model: Dielectrics in ShellGen

1.1 Media and interfaces

- Surrounding medium (typically air)
- Object medium (dielectric)
- Cavity medium (typically air)

1.2 What changes between the bulk and the shell

- Bulk: single boundary surface (outside)
- Shell: two boundaries (outside + inside) + thickness effects

1.3 Roughness as 'frosting'

- How micro-roughness modulates visibility through the object

- Typical artefacts (noise/fireflies) and what mitigates them (without prescribing CLI yet)

1.4 Absorption/tint

- Path-length dependence: thicker regions appear more saturated/darker

Chapter 2. Assets and Validation

2.1 Objective and gatekeeping principle

Dielectric rendering is unusually sensitive to small geometric and conventional errors. For this reason, ShellGen should be operated with a strict gate: no 'final' dielectric renders are produced until the asset passes validation and the diagnostic evidence set is complete.

This chapter defines:

1. Asset requirements for (a) bulk solids and (b) shell-with-cavity models.
2. A validation checklist that must be passed before an asset is admitted to the rendering pipeline.
3. A minimal diagnostic render set designed to reveal the most common failure modes early.

The intent is not to enforce aesthetic consistency, but to enforce physical and geometric plausibility sufficient for reliable dielectric behaviour, including refraction, Fresnel reflection, roughness-driven blur, and absorption/tint.

2.2 Geometry requirements: bulk solids

A bulk solid is a dielectric object with no internal void. In this regime, the mesh should behave as a single closed boundary between the surrounding medium (typically air) and the object medium (glass or similar).

Bulk asset requirements

- Closed surface: the mesh must be watertight (no holes, no open boundaries).
- Manifoldness: the mesh should be a valid 2-manifold (avoid non-manifold edges/vertices).
- Consistent normal orientation: normals must be consistently oriented (typically outward).
- No self-intersections: interpenetrating faces often produce unstable refraction paths and noise.
- Scale declared and plausible: dielectric effects (especially tint/absorption and caustics) are scale dependent. The pipeline must record a unit convention, and the asset must be consistent with it.
- Clean topology in critical regions: avoid degenerate triangles and very thin sliver faces near rims and corners, as they frequently cause sparkle-like artefacts in specular paths.

Practical note: A bulk solid offers fewer 'escape routes' for rays than a diffuse object. Minor topology defects can manifest as major optical artefacts because refraction and total internal reflection amplify geometric inconsistencies.

2.3 Geometry requirements: shell with cavity

A shell-with-cavity asset represents a vessel-like object defined by two boundaries: an outer surface and an inner surface enclosing a cavity medium (typically air, or a fluid if supported). This regime is both more realistic for vessels and more fragile.

Shell asset requirements

- Two non-intersecting surfaces:
  - Outer surface: closed, watertight.
  - Inner surface: closed, watertight.
- Orientation convention is explicit and consistent:
  - Outer surface normals: outward (towards the surrounding medium).
  - Inner surface normals: oriented towards the cavity medium (i.e., 'inward' relative to the object material), depending on your convention. What matters is that the two surfaces are oriented consistently with the medium assignment.
- Minimum thickness is enforced:
  - Shell thickness must not collapse to near-zero anywhere.
  - Thickness discontinuities (sudden jumps) should be avoided unless intentionally modelled.
- No self-intersection and no surface crossovers:
  - The inner surface must never cross the outer surface.
  - Local self-intersections (e.g., at rims, spouts, handles) are a primary cause of bright leaks and fireflies.
- Rim and mouth integrity:
  - Open vessels must be modelled so that the inner and outer boundaries meet in a geometrically consistent rim structure (avoid ambiguous 'paper-thin' rims).

Practical note: In a shell, refraction occurs at two interfaces, and rays may bounce multiple times between them. Any inconsistency can create pathological ray paths and visually severe artefacts.

2.4 Validation checklist

The following checklist is the formal admission gate for assets in this manual. If any check fails, the asset is not admitted for final evidence rendering.

Table 36. Asset validation checklist (Bulk vs Shell).

Check	Bulk solid	Shell with cavity	Pass criteria	Typical failure symptom
Closed surface(s)	Required	Required (outer & inner)	No holes; no open boundaries	Light leaks, black regions, unstable refraction
Manifoldness	Required	Required	2-manifold topology	Local spikes, shading breaks, and inconsistent refraction
Normal orientation	Required	Required (outer/inner consistent)	Orientation consistent with the medium assignment	Inside-out look, inverted highlights, 'wrong glass'

Self-intersection	Not allowed	Not allowed	No interpenetrations	Fireflies, bright leaks, discontinuities
Surface crossover	N/A	Not allowed	Inner never crosses outer	Severe leaks, impossible refraction patterns
Minimum thickness	N/A	Required	Thickness above declared threshold	Sparkles, leaks, 'tearing' refraction
Scale declared	Required	Required	Unit convention recorded and plausible	Over-strong tint, unexpected caustics, unstable noise
Degenerate faces	Not allowed	Not allowed	No zero-area faces/slivers	Pixel sparkles, local noise bursts
Rim integrity	Recommended	Required for open vessels	Inner/outer join consistently	Unexplained bright edges, broken transmission

Implementation rule: The run manifest (see Chapter 6) must record the validation status and any repair steps applied to the asset (e.g., 'recomputed normals', 'removed self-intersections', 'thickness clamp applied in modelling').

2.5 Diagnostic render set

ShellGen v24.02 focuses on dielectric rendering and does not emit dedicated AOVs for normals or thickness. Therefore, the diagnostic evidence set is split into:

- Mesh diagnostics (external, MeshLab): normal orientation and basic mesh integrity checks.
- Optical diagnostics (ShellGen): rim highlights and transmission/refraction checks using a reference background.

The diagnostic set is not a “beauty output”. Its role is to reveal, quickly and unambiguously, whether the asset is compatible with dielectric behaviour.

Table 37. Mandatory diagnostic render set.

Diagnostic	Purpose	Bulk	Shell	How to interpret
Silhouette	Verify shape, scale, gross defects	Yes	Yes	Look for spikes, holes, and unexpected topology
Normal visualisation	Verify normal orientation and discontinuities	Yes	Yes	Any flipped region is a hard fail
Rim highlight test	Verify interface correctness (Fresnel behaviour)	Yes	Yes	Highlights should be plausible and continuous
Transmission test (with chart/grid)	Verify refraction/transmission logic	Yes	Yes	Look for inverted refraction, sudden discontinuities
Thickness evidence	Verify the shell thickness behaviour	No	Yes	Identify collapsed or discontinuous thickness zones

MeshLab procedure

The following procedure is sufficient for the admission gate in Table T2.1.

1. Load the mesh(es). Import the outer mesh (and inner mesh if present). For shells, load both and inspect them together.
2. Inspect shading. Rotate the object under MeshLab's default shaded render. Sudden shading flips typically indicate normal issues or non-manifold regions.
3. Visualise normals. Enable face normals or vertex normals rendering (menu names vary by MeshLab version; typically under Render). Increase the normal length/scale until directions are clear.
4. Check orientation.
    a. Bulk: normals should be consistently outward.
    b. Shell: outer normals should point outward (towards surrounding air); inner normals should be consistently oriented with respect to the cavity boundary (i.e., opposite to the outer surface across the wall).
5. Repair if needed (MeshLab filters). If normals are incorrect, use MeshLab's normal/orientation filters (typically under *Filters → Normals, Curvatures and Orientation*) to re-orient or invert faces coherently.
6. Save evidence. Capture at least one screenshot for the normals check and one for the shaded/silhouette view. Recommended location: a mesh_diag/ subfolder next to the ShellGen evidence bundle.

ShellGen optical diagnostics (recommended)

For optical diagnostics, use the canonical lighting kit (Chapter 5) with a reference background (plane texture grid). Then extract one or more representative frames:

- Rim highlight: pick a frame where the object boundary is viewed at a grazing angle.
- Transmission: pick a frame where the plane grid is visible through the object, and refraction can be assessed.

2.6 Validation outcomes and operator actions

If all checks pass, the asset is admitted for evidence rendering under the presets in Chapter 4 and the evidence bundle in Chapter 6.

If any check fails, the operator should stop and repair the asset before proceeding. In dielectric rendering, 'rendering through' a defect is rarely efficient: it produces images that are difficult to interpret and expensive to compute, while providing weak evidence.

Chapter 3. Mode Selection: Bulk vs Shell

ShellGen does not require an explicit 'mode' switch for bulk versus shell-with-cavity rendering. Instead, the regime is inferred from the geometry inputs:

- SHELL (shell-with-cavity) is selected when an inner surface is provided via --inner together with --outer. In this regime, ShellGen treats the object as a dielectric shell bounded by two interfaces (outer and inner), with a cavity region enclosed by the inner surface.
- BULK (bulk solid) is selected when --inner is absent and only --outer is provided. In this regime, ShellGen treats the object as a single closed dielectric boundary between the surrounding medium and the object medium.

In the current pipeline, the cavity medium is always air. There is no explicit control to specify a different cavity medium; therefore, for all SHELL runs in this manual, the cavity is assumed and recorded as air.

For the mapping between reader-facing terminology (e.g., 'outer surface mesh', 'inner surface mesh', 'cavity medium') and the corresponding ShellGen CLI controls, see Appendix A.

Below are two reference invocations for dielectric rendering: one for shell-with-cavity assets and one for bulk assets. These examples are provided to document parameter names and typical scene/camera settings; users may vary parameters as required, and may optionally enable video generation (e.g., via --make-videos) when producing turntable evidence.

3.1 Reference invocations

The following two commands are the canonical baseline runs used throughout this manual. They are designed to be comparable across machines and to produce a complete evidence bundle (Chapter 6).

BULK baseline (solid dielectric)

```
python3 shellgen_viewer_v24_02.py    --outer objs/bunny.obj    --hdri
envs/studio.hdr              --mat-label     CLV              --mat-xml
media/colver_glass_data.xml    --mat-eta 1.5    --body-type solid    --
plane    --plane-size-mm    150    --plane-grey    0.5    --plane-texture
media/pattern_21.png    --step-deg 1    --cam-radius 160 --cam-height 40
--fov 60    --cam-look-at-x 0 --cam-look-at-y 54 --cam-look-at-z 0    -
-w 512 --h 512 --spp 256    --integrator volpathmis    --env-gain 0.4
--jobs 1    --results-dir bunny
```

SHELL baseline (absorption-only shell; no scattering)

In v24.02, the shell regime defaults to absorption-only when --medium-albedo is not explicitly set.

```
python3 shellgen_viewer_v24_02.py   --outer objs/bunny_50__outerTY.obj
--inner objs/bunny_50__innerTY.obj    --hdri envs/studio.hdr    --mat-
label CLV    --mat-xml media/colver_glass_data.xml    --mat-eta 1.5    -
-body-type shell    --plane --plane-size-mm 150 --plane-grey 0.5 --
plane-texture media/pattern_21.png    --step-deg 1    --cam-radius 160 -
-cam-height 40 --fov 60    --cam-look-at-x 0 --cam-look-at-y 54 --cam-
look-at-z 0    --w 512 --h 512 --spp 256    --integrator volpathmis    -
-env-gain 0.4    --jobs 1    --results-dir bunny_shell_outerTY_innerTY
```

Optional named variant (power-user): shell with volumetric scattering enabled

This variant is optional and is documented as a power-user extension. It changes the shell interior from absorption-only to a participating medium with controlled scattering:

```
python3 shellgen_viewer_v24_02.py  --outer objs/bunny_50__outerTY.obj
--inner objs/bunny_50__innerTY.obj  --hdri envs/studio.hdr  --mat-
label CLV  --mat-xml media/colver_glass_data.xml  --mat-eta 1.5  -
-body-type shell  --medium-albedo 0.1 --medium-g 0.0  --plane --
plane-size-mm  150  --plane-grey  0.5  --plane-texture
media/pattern_21.png  --step-deg 1  --cam-radius 160 --cam-height 40
--fov 60  --cam-look-at-x 0 --cam-look-at-y 54 --cam-look-at-z 0  -
-w 512 --h 512 --spp 256  --integrator volpathmis  --env-gain 0.4
--jobs 1  --results-dir bunny_shell_scatter_outerTY_innerTY
```

Reference baseline commands (v24.02)

Bulk (solid) baseline:

```
python3 shellgen_viewer_v24_02.py --outer objs/bunny.obj --hdri
envs/studio.hdr --mat-label CLV --mat-xml media/colver_glass_data.xml
--mat-eta 1.5 --body-type solid --plane --plane-size-mm 150 --plane-
grey 0.5 --plane-texture media/pattern_21.png --step-deg 1 --cam-
radius 160 --cam-height 40 --fov 60 --cam-look-at-x 0 --cam-look-at-y
54 --cam-look-at-z 0 --w 512 --h 512 --spp 256 --integrator volpathmis
--env-gain 0.4 --jobs 1 --results-dir bunny
```

Shell (cavity) baseline (absorption-only by default; no explicit scattering controls):

```
python3 shellgen_viewer_v24_02.py --outer objs/bunny_50__outerTY.obj -
-inner objs/bunny_50__innerTY.obj --hdri envs/studio.hdr --mat-label
CLV --mat-xml media/colver_glass_data.xml --mat-eta 1.5 --body-type
shell --plane --plane-size-mm 150 --plane-grey 0.5 --plane-texture
media/pattern_21.png --step-deg 1 --cam-radius 160 --cam-height 40 --
fov 60 --cam-look-at-x 0 --cam-look-at-y 54 --cam-look-at-z 0 --w 512
--h 512 --spp 256 --integrator volpathmis --env-gain 0.4 --jobs 1 --
results-dir bunny_shell_outerTY_innerTY
```

Shell variant. mild “haze” (optional, power-user; enables volumetric scattering in the shell):

```
python3 shellgen_viewer_v24_02.py --outer objs/bunny_50__outerTY.obj -
-inner objs/bunny_50__innerTY.obj --hdri envs/studio.hdr --mat-label
CLV --mat-xml media/colver_glass_data.xml --mat-eta 1.5 --body-type
shell --medium-albedo 0.1 --medium-g 0.0 --plane --plane-size-mm 150 -
-plane-grey 0.5 --plane-texture media/pattern_21.png --step-deg 1 --
cam-radius 160 --cam-height 40 --fov 60 --cam-look-at-x 0 --cam-look-
at-y 54 --cam-look-at-z 0 --w 512 --h 512 --spp 256 --integrator
```

```
volpathmis --env-gain 0.4 --jobs 1 --results-dir bunny_shell_haze_outerTY_innerTY
```

Note: if --medium-albedo is not provided, ShellGen defaults to 0.1 for bulk solids and 0.0 for shells (absorption-only). The shell variant above overrides the default to demonstrate scattering-enabled shells.

3.2 Decision rule

- Use bulk when the object is physically solid (paperweight, solid block)
- Use a shell when the object is a vessel/container with a cavity

3.3 Bulk mode: what the operator sets conceptually

- Surrounding medium, object medium
- Roughness, absorption/tint (if any)
- Lighting and camera evidence

3.4 Shell mode: what the operator sets conceptually

- Inner/outer boundary integrity
- Cavity medium specification (typically air)
- Thickness-sensitive effects (tint, caustics, distortion)

3.4 Common mode mistakes and how to detect them

- 'Looks hollow' vs 'looks solid'
- Inverted refraction
- Excessive blackening or unexplained bright leaks

Chapter 4. Dielectric Material Presets

4.1 Purpose and use of presets

Dielectric appearance is governed by a small set of physical controls (media assignment, refractive indices, surface micro-roughness, and—optionally—absorption/tint). However, operators typically reason in terms of observable behaviour ('clear', 'tinted', 'frosted', 'high-gloss vs satin') rather than raw parameters.

This chapter defines dielectric presets as named configurations of intent that:

- clearly state what visual behaviour they target,
- specify what evidence is required to substantiate that behaviour, and
- Record the minimal set of controls that must be tracked in the manifest.

Presets are not mandatory. They are a documentation device that improves comparability across runs and reduces ambiguity in reporting.

These presets apply to both regimes (BULK and SHELL). Where shell-specific considerations apply (e.g., thickness dependence), they are stated explicitly.

4.2 Common controls

All presets draw from the same conceptual control set:

- Medium assignment
  - Surrounding medium (typically air)
  - Object medium (dielectric)
  - Cavity medium (shell only; typically air)
- Refractive index (RI)
  - RI of the object medium
  - RI of the surrounding medium
  - RI of the cavity medium (shell only)
- Surface micro-roughness
  - Controls the breadth of specular reflection and the sharpness of transmission/refraction.
- Absorption/tint (optional)
  - Produces thickness-dependent colouration/attenuation along the optical path.

Reporting requirement: For any preset run, the manifest must record these controls in reader terms (see Chapter 6). This is more important than which internal parameter names were used to implement them.

4.3 Preset catalogue overview

Table 38. Preset summary.

Preset ID	Name	Intended dielectric behaviour	Bulk	Shell	Primary required evidence
D-CLEAR	Clear dielectric	Crisp transmission and refraction with sharp interface highlights	Yes	Yes	Transmission + rim
D-TINT	Tinted dielectric	Thickness-dependent attenuation/colouration while preserving dielectric behaviour	Yes	Yes	Transmission & thickness/tint comparison
D-FROST	Frosted dielectric	Roughness-driven blurred transmission and broadened highlights	Yes	Yes	Transmission & rim
D-SATIN	Satin/low-gloss dielectric	Moderately broadened highlights, slight transmission blur; 'softened' look	Yes	Yes	Rim & transmission
D-HIGH-ROUGH	High-roughness dielectric	Strong blur in transmission/refraction	Yes	Yes	Transmission & manifest

	dominated by micro-roughness	roughness declaration

Note: D-HIGH-ROUGH is included as a behavioural endpoint of the same control axis as D-FROST and D-SATIN. It is not privileged; it is documented because it is commonly encountered when exploring roughness ranges.

4.4 D-CLEAR. Clear dielectric

Intent

A clear dielectric exhibits:

- crisp transmission of background structure through the object,
- well-defined refraction (distortion consistent with shape),
- sharp Fresnel-driven highlights at grazing angles.

Applicable regimes

- Bulk: baseline case; simplest validity check for dielectric behaviour.
- Shell: requires correct inner/outer surfaces and cavity medium assignment.

Required evidence

- Rim highlight diagnostic (Chapter 2)
- Transmission diagnostic with background grid/chart
- Surface normals visualisation (asset gate)
- Silhouette (asset gate)
- Thickness evidence (shell only)

Expected outcome

- Transmission shows coherent distortion with no sudden discontinuities.
- Highlights are continuous along edges and consistent with curvature.
- Shell assets show plausible behaviour at rims and through thickness transitions.

Common pitfalls

- Inside-out appearance: typically normals or regime confusion (see Chapter 7).
- Unexpected darkening: likely scale inconsistency if absorption/tint is non-zero, or overly aggressive post-processing (if any).
- Bright leaks (shell): thickness collapse or surface crossover.

4.5 D-TINT. Tinted dielectric

Intent

A tinted dielectric exhibits:

- colouration/attenuation that increases with optical path length, and
- preserved dielectric interface behaviour (still reflective and refractive; not a diffuse medium).

Tint is treated here as an absorption effect (path-length-dependent), not as a surface colour.

Applicable regimes

- Bulk: tint correlates with local thickness along view rays.
- Shell: tint often emphasises thickness variations; therefore, thickness integrity matters.

Required evidence

All D-CLEAR evidence, plus:

- A thickness/tint comparison view (shell strongly recommended; bulk optional but useful)

Expected outcome

- Thicker regions appear more saturated/darker than thin regions.
- The tint does not 'float' on the surface; it should read as volumetric attenuation along refracted paths.
- Refraction remains coherent; tint should not introduce discontinuities.

Common pitfalls

- Over-strong tint: often caused by incorrect scale (units). Correct scale first, then retune tint.
- Non-physical surface colouration: may indicate the implementation behaves like a surface albedo rather than absorption; document the limitation explicitly if applicable.
- Shell leaks become more apparent: tint can make small geometry errors visually obvious; treat this as a diagnostic advantage.

4.6 D-FROST. Frosted dielectric

Intent

A frosted dielectric exhibits:

- blurred transmission (background detail becomes progressively less sharp),
- broadened specular highlights (wider, softer reflections),
- maintained overall dielectric identity (still reflective/refractive, not matte diffuse).

Frosting is modelled via surface micro-roughness. It is not, by itself, a volumetric scattering model.

Applicable regimes

- Bulk: frosting softens refraction and internal visibility.
- Shell: frosting affects both outer and inner interface contributions; shell integrity remains essential.

Required evidence

- Rim highlight diagnostic (to show highlight broadening)
- Transmission diagnostic with background chart/grid (to show blur)
- Manifest must state the micro-roughness level used (reader term)
- Shell: thickness evidence (as usual)

Expected outcome

- Transmission is visibly blurred relative to D-CLEAR under identical lighting/camera.
- Highlights are wider and less intense at the peak, consistent with roughness.
- Shell rims and edges remain stable (no leaks, no discontinuities).

Common pitfalls

- Noise amplification: Rough dielectrics can increase variance in specular paths. Record sampling and denoiser status; avoid interpreting noise as 'material texture'.
- Perceived 'milkiness': frosting can resemble a milky material, but the mechanism is still interface roughness. If the user requires true internal scattering, that is outside this manual's scope and should be stated explicitly.
- Loss of diagnostic power: very high roughness can hide refraction errors. If troubleshooting, temporarily revert to D-CLEAR.

4.7 D-SATIN. Satin / low-gloss dielectric

Intent

A satin dielectric is intermediate between clear polished and strongly frosted:

- highlights broaden modestly (softer reflections),
- transmission remains present but slightly softened,
- The overall look is 'softened' without becoming aggressively blurred.

Applicable regimes

- Bulk and shell are equally applicable.

Required evidence

- Rim highlight diagnostic
- Transmission diagnostic

- Manifest roughness declaration
- Shell: thickness evidence

Expected outcome

- Compared to D-CLEAR, highlights are visibly less sharp; transmission is slightly less crisp.
- Compared to D-FROST, blur and highlight broadening are milder.

Common pitfalls

- Ambiguity with lighting: satin appearance can be confounded by lighting changes. For evidence, keep the lighting/camera fixed when comparing with D-CLEAR or D-FROST.
- Over-interpretation: if differences are subtle, do not force claims; record the configuration and let the images speak.

4.8 D-HIGH-ROUGH. High-roughness dielectric

Intent

This preset documents the high-roughness end of interface modelling:

- transmission and refraction become strongly blurred,
- highlights become very broad and subdued.

It is included to provide coverage across the controllable range of micro-roughness, not to privilege a particular use case.

Applicable regimes

- Bulk and shell; shell artefacts may be harder to see at high roughness.

Required evidence

- Transmission diagnostic
- Rim highlight diagnostic
- Manifest roughness declaration
- Shell: thickness evidence

Expected outcome

- The background chart/grid appears substantially blurred through the object.
- Refraction reads as a broad distortion rather than a crisp lensing effect.

Limitations and caveats

- This remains an interface roughness model. If the visual goal is truly particulate or volumetric scattering, that is outside the scope of this manual.

- High roughness can conceal geometry problems; validation must be completed at a lower roughness (Chapter 2 gate).

4.9 Preset evidence protocol

For any preset, evidence is strongest when the operator can show that the observed behaviour is attributable to the intended control axis rather than uncontrolled changes.

When comparing presets (e.g., D-CLEAR vs D-FROST), keep the following fixed:

- asset (and its version),
- regime,
- lighting kit,
- camera set,
- resolution and sampling policy (or record differences explicitly).

If any of these differ, the report should state it plainly; otherwise, the comparison may be misleading.

4.10 Preset reporting template

Preset run summary (for reports)

- RUN_ID:
- Regime: BULK / SHELL
- Preset: D-…
- Intent: (1 sentence)
- Validation status: T2.1 passed; diagnostics produced: [list]
- Observed outcome: (3–5 bullets)
- Notes/limitations: (1–3 bullets; include noise or approximation notes)

Chapter 5. Lighting and Camera Evidence

5.1 Rationale

Dielectric appearance is highly sensitive to lighting geometry and camera placement. For this reason, ShellGen evidence should be generated using named kits that capture stable lighting and camera settings. The kit identifiers are used in the Run ID (Chapter 6) and provide a compact way to assert comparability across runs.

5.2 Standard lighting kit (default)

LIGHT_STUDIO_PLANE_GRID (default evidence kit)

This kit is the default for evidence runs in this manual. It uses a studio HDRI and a textured ground plane that provides a stable reference for transmission/refraction.

Element	Setting (ShellGen v24.02)
HDRI	--hdri envs/studio.hdr
Environment gain	--env-gain 0.4
Reference plane	--plane
Plane size	--plane-size-mm 150
Plane reflectance	--plane-grey 0.5
Plane texture (grid/reference)	--plane-texture media/pattern_21.png

Operator rule: for comparisons (e.g., D-CLEAR vs D-FROST), keep the lighting kit unchanged.

5.3 Standard camera set (default)

Notes:

- This camera set is tuned for the bunny baseline runs. Other assets may require an asset-specific camera set; if so, give it a new CAM_… identifier and encode it in the Run ID.
- ShellGen emits two motion sequences for each run: CAM (orbit) and TT (turntable). These sequences are part of the evidence bundle (Chapter 6) and serve as the primary “views”.

5.4 Canonical evidence view selection from sequences

Because ShellGen emits sequences rather than separate per-diagnostic renders, this manual standardises evidence selection as follows:

- Turntable keyframe: use anim_<MAT>.TT.<ENV>/frame_000.png as the default keyframe for reporting unless a different frame is explicitly justified.
- Rim highlight evidence: select a frame (TT or CAM) where a representative rim/edge region is near-grazing to the camera; record the chosen frame index in the report.
- Transmission (grid) evidence: select a frame where the grid plane is clearly visible through the object and refraction is diagnostically informative; record the chosen frame index in the report.

If a report includes extracted frames, they should be named using the Run ID and frame index to remain traceable (e.g., RUN_ID__TT__frame_000.png).

5.5 Noise and sampling notes

Rough dielectrics and caustics can increase Monte Carlo variance. When comparing runs, keep-- spp fixed (or report differences explicitly).

- ShellGen v24.02 provides an explicit --seed parameter. For evidence runs, use a fixed seed (default 0) unless you intentionally study variance.
- Optional outputs such as --make-videos and --exr-sidecar are supported for power users; they are not required for baseline evidence in this manual.

Chapter 6. Evidence Packages and Reporting

6.1 Purpose of evidence packages

ShellGen outputs are only useful to collaborators if they are reproducible and auditable. Dielectric rendering, in particular, can change materially with small differences in scale, medium assignment, sampling, or lighting. Therefore, each run must produce a standard Evidence Package that includes:

- A small set of images that substantiates the claimed behaviour.
- A manifest that records the run configuration in reader-facing terms.
- Provenance links to the asset version used.

This chapter defines what constitutes a 'complete run' in this manual.

6.2 Run identity and naming

A run identifier should encode the smallest set of attributes needed to interpret results at a glance. It should not depend on renderer jargon; instead, it should use the manual's conceptual vocabulary (regime, preset, lighting kit, camera set).

Recommended RUN_ID format
<ASSET>__<REGIME>__<PRESET>__<LIGHT>__<CAM>__<RES>__SPP<NNN>__REV<NN>

Example
VASE_01__SHELL__D-FROST__HDRI_A__CAM_CANON__1920x1080__SPP512__REV03

Table 39. Run ID fields (recommended).

Field	Meaning	Example
ASSET	Geometry asset identifier	VASE_01
REGIME	BULK or SHELL	SHELL
PRESET	Dielectric preset ID	D-FROST
LIGHT	Lighting kit identifier	HDRI_A
CAM	Camera set identifier	CAM_CANON
RES	Resolution	1920x1080
SPP	Samples per pixel	512
REV	Manual/pipeline revision tag	REV03

Operational rule: RUN_ID must be used consistently in filenames and manifest entries. A run that cannot be unambiguously identified from its files is considered incomplete.

6.3 Evidence Bundle contents

Each run writes a self-contained results folder specified by --results-dir (recommended: set this equal to RUN_ID). The folder contains:

- anim_manifest.json. sequence index used by the HTML viewer and for programmatic inspection
- params_v24.02.json. complete parameter snapshot for the run (including regime inference, camera look-at, angles, frame counts)
- viewer.html. lightweight, offline viewer (CAM and TT side-by-side)
- Sequence folders (PNG frames):
    - anim_<MAT>.CAM.<ENV>/frame_###.png (orbit motion)
    - anim_<MAT>.TT.<ENV>/frame_###.png (turntable motion) where <MAT> is --mat-label and <ENV> is the HDRI stem (e.g., studio).

Optional outputs:

- If --exr-sidecar is enabled, each sequence folder also contains frame_###.exr.
- If --make-videos is enabled, derived videos may be written (document the exact filenames if you rely on them).

6.4 Manifest minimum fields

In this manual, the "manifest" is defined strictly as the **tool-emitted provenance artefacts** produced by ShellGen:

- params_v24.02.json. the complete CLI argument snapshot (including resolved defaults such as --medium-albedo when not explicitly provided)
- anim_manifest.json. the sequence index used by viewer.html

No additional human-authored metadata file is required or assumed.

Minimum recorded fields (policy)

To be considered complete, a run must have enough recorded information to be reproducible without relying on operator memory. The required fields are therefore those that must be present either:

- directly in params_v24.02.json (as CLI arguments / resolved defaults), and/or
- encoded in the operator-chosen results folder name (--results-dir, recommended to be the Run ID; see Section 6.2).

Required fields (must be recoverable from the run folder):

- ShellGen version (implicit via params_v24.02.json filename and/or recorded fields)
- Geometry provenance: --outer path and (if applicable) --inner path
- Regime / body type: --body-type and the presence/absence of --inner
- Evironment: --hdri, plus environment controls (--env-gain, and if used --use-constant-env, --env-radiance)
- Optical controls: --mat-xml, --mat-eta, --surface-roughness, and medium controls (--medium-albedo, --medium-g)
- Camera protocol: --cam parameters, --fov, and --step-deg (sequence definition)
- Render protocol: --integrator, --max-depth, --w, --h, --spp, --seed, --jobs

Recommended (for interpretability, not required by the tool):

- Encode the conceptual attributes (Asset ID, BULK/SHELL, Preset ID, Lighting Kit ID, Camera Set ID, resolution, SPP, and revision tag) in the Run ID/results directory name (Section 6.2). This provides a reader-facing index without requiring an extra metadata file.

Validation evidence

ShellGen does not record MeshLab validation outcomes in the JSON files. If mesh diagnostics were performed, keep the evidence (screenshots) alongside the run folder (recommended: a `mesh_diag/` folder placed next to the run folder, or inside it if you prefer). Any repair steps applied to the mesh should be described in the report that cites the run.

Chapter 7. Troubleshooting Atlas

7.1 How to use this atlas

This atlas is organised for operational speed: symptom → likely cause → what to check → corrective action → required evidence. In dielectric rendering, the fastest path to resolution is usually to re-run the relevant diagnostic view(s) rather than repeatedly adjusting material controls.

A recurring pattern is that many artefacts are not 'material problems' at all, but asset or convention problems (normals, self-intersections, thickness collapse, scale errors). The diagnostic set in Chapter 2 exists precisely to separate these.

7.2 Troubleshooting atlas (dielectric bulk + shell)

Table 40. Troubleshooting Atlas.

Symptom	Likely cause	What to check	Corrective action
Object looks 'inside-out'	Flipped normals or inconsistent outer/inner normals	Normal visualisation	Reorient normals; ensure inner/outer are consistent with medium assignment
Shell shows bright leaks or glowing seams	Inner/outer surfaces intersect; thickness collapses locally; rim ambiguity	Thickness evidence & silhouette	Repair mesh; remove intersections; enforce minimum thickness; fix rim topology
Sudden refraction discontinuities	Non-manifold edges; self-intersections; degenerate faces	Silhouette + mesh audit	Clean topology; remove degenerates; remesh problematic regions
Transmission looks 'blocked' when it should be clear	Roughness too high; preset mismatch; accidental absorption/tint	Preset ID & manifest optical controls	Switch to D-CLEAR; reduce roughness; disable tint for diagnosis

Object becomes implausibly dark with tint/absorption	Scale inconsistency (units wrong); overly strong absorption	Scale declaration in manifest; compare with known dimensions	Correct asset scale; retune absorption/tint after scale is correct
Excessive fireflies / sparkling pixels	Difficult specular paths (caustics), strong refraction with sharp features, self-intersections, and very small triangles	Rim & transmission; check for intersections and degenerates	Fix geometry; adjust sampling strategy and lighting design; avoid pathological light-object configurations in evidence kits
Strange black patches at grazing angles	Total internal reflection regimes combined with roughness, normal discontinuities, and shading normal issues	Rim test + normals	Fix normals/discontinuities; confirm roughness is intentional; validate that behaviour persists after geometry correction.
Shell behaves as if 'solid' (cavity not evident)	Incorrect regime selection; cavity medium mis-specified; inner surface missing or mis-oriented	Regime in manifest; thickness evidence; silhouette	Confirm SHELL regime; ensure inner surface exists and is valid; set cavity medium explicitly
Bulk behaves as if 'hollow'	Wrong regime selection; asset unintentionally includes inner surface; normals inconsistent	Regime, silhouette, and normals	Use the BULK regime; remove the unintended inner surface or correct interpretation.
Highlights are broken into bands or faceted artefacts	Low mesh quality; smoothing/normal issues; uneven tessellation on curved surfaces	Silhouette close-up; normals	Improve mesh quality; repair vertex normals; remesh curvature regions
Thin features disappear or look unstable	Geometry below practical scale; thickness too low; degenerate triangles	Scale declaration; thickness evidence	Increase thickness; rescale asset; simplify or reinforce thin regions

7.3 Diagnostic-first remediation protocol

When a symptom appears, apply the following protocol in order:

1. Confirm regime correctness (bulk vs shell) in the manifest.
2. Run/inspect normals and silhouette diagnostics (fastest hard failures).
3. For shells, inspect the thickness evidence (second-fastest hard failure).
4. Only after geometry and regime are correct, adjust material intent (preset choice and roughness/tint).
5. Record the change in the manifest and regenerate the minimum evidence set.

This protocol prevents 'parameter chasing' and ensures that each change is auditable.

Annex G. Moulds

G.1 Geometrical claims

Theorem A. Bipartite demouldability with straight extraction.

Let a unit horizontal vector u and the plane P_u. Set $S^+ = S \cap H^+$ and $S^- = S \cap H^+$. Let the mould halves be separated along P_u, and the intended extraction directions be +u for the S^+ half and -u for the S^- half. For a unit vector u, the vertical plane $P_u = \{x \in R^3: \langle x\text{-}c, u\rangle = 0\}$, with complementary half-spaces $H^+ = \{\langle x\text{-}c, u\rangle \geq 0\}$ and $H^- = \{\langle x\text{-}c, u\rangle \leq 0\}$. Then the following are equivalent:

1. Demouldability. Each half-mould can be removed from the cast by straight translation along its assigned direction without intersection with S.
2. No undercut in the normal condition. For every $p \in \partial S \cap \text{int}(H^+)$, $\langle n(p), u\rangle \geq 0$, and for every $p \in \partial S \cap \text{int}(H^-)$, $\langle n(p), u\rangle \leq 0$; equality holds only on the parting set $\partial S \cap P_u$.

Thus, an undercut relative to direction d is precisely a patch where $\langle n(p), d\rangle < 0$.

Theorem B. Quadripartite demouldability with straight extraction.

Let (u,v) be orthonormal horizontal vectors. Let planes $P_u = \{\langle x\text{-}c,u\rangle = 0\}$, $P_v = \{\langle x\text{-}c,v\rangle = 0\}$. Let the four quadrants be denoted as $S_{\sigma,\tau} = S \cap \{\sigma\langle x\text{-}c,u\rangle \geq 0\} \cap \{\tau\langle x\text{-}c,v\rangle \geq 0\}$, where $\sigma = \pm 1$ and $\tau \pm 1$. Let the extraction direction for quadrant (σ,τ) be the outward bisector $d_{\sigma,\tau} = \frac{\sigma u + \tau v}{\sqrt{2}}$. Then, the following are equivalent:

1. Demouldability. Each quadrant $S_{\sigma,\tau}$ can be removed by straight translation along $d_{\sigma,\tau}$ without intersection with S.
2. No undercut normal conditions in quadrant directions. For every $p \in \partial S \cap \text{int}(S_{\sigma,\tau})$, $\leq\langle n(p), d_{\sigma,\tau}\rangle \geq 0$, with equality permitted only on the parting sets $\partial S \cap (P_u \cup P_v)$ and measure-zero features consistent with draft.

Theorem C. Feasibility Test

Given a triangulated ∂S:

- Case A: pick unit horizontal u. For every triangle centre p on the + side, require $\langle n(p),u\rangle \geq 0$; on the − side, $\langle n(p), u\rangle \leq 0$.
- Case B: pick orthonormal u,v. For each quadrant (σ,τ), require $\langle n(p), d_{\sigma,\tau} = \frac{\sigma u + \tau v}{\sqrt{2}}\rangle \geq 0\rangle$ on all its faces.
- Allow a small tolerance >0 to encode draft.

If any face violates the sign condition away from the parting sets, that solid is not demouldable under the chosen case/planes with straight extraction.

G.2 User Guide and API

This annex documents the public functions provided by the mould generator. All methods are implemented in src/mould_generator.py and operate on meshes represented with the trimesh library.

G2.1 Usage

Installation

Prerequisites: Python 3.7 or later, pip.

Cloning the repository:

```
git clone https://github.com/Lion4re/automated_3d_mould_generator.git

cd automated_3d_mould_generator
```

Dependencies: Install in a virtual environment with:

```
pip install -r requirements.txt
```

Editable installation (optional):

```
pip install -e
```

Interactive mode (Beginners)

Simply run the program and follow the guided setup: **`python makeMold.py`**

Example Interactive Session:

🏭 STL MOLD MAKER - Interactive Setup

💡 Tip: Press ENTER to use default values shown in parentheses

📂 Available STL Models (3 found):

1. chess_knight.stl (1.2 MB)
2. miniature_house.stl (856.3 KB)
3. decorative_vase.stl (2.1 MB)

📋 Step 1: Select your STL model

Enter model number (1-3): 1

✅ Selected: chess_knight.stl

📋 Step 2: Configure mould parameters

Wall thickness (mm) - leave empty for auto-calculation (default: auto):

Split axis (x=left-right, y=front-back, z=top-bottom) [x/y/z] (default: x): z

Number of mould pieces [2/4] (default: 2): 2

Number of alignment keys [2/4] (default: 2): 2

Mesh repair method [auto/trimesh/open3d/hybrid/none] (default: auto):

Draft angle in degrees (0.5-3.0 for easier demolding, empty for none) (default: none): 1.5

📋 Configuration Summary:

📦 Model: chess_knight.stl

🔧 Wall thickness: Auto-calculated

⚡ Split axis: Z-axis

🧩 Mould pieces: 2

🔑 Alignment keys: 2

🛠️ Mesh repair: auto

📐 Draft angle: 1.5°

✅ Proceed with mould creation? [Y/n]: y

Command-Line Interface

The generator can be run from the terminal:

```
python src/mould_generator.py --visualize
```

```
--input models/MAOI03b.stl --output output2101

--padding 0.1 --hole_positions bottom

--split_mode quarters

--draft_angle 0.0
```

Parameters:

```
--input (-i): Path to the input STL file or directory

--output (-o): Directory to save output files

--padding: percentage value (default: 0.1)

--hole_positions: List of hole sites

--split_mode: halves or quarters (default: quarters)

--draft_angle: Draft angle in degrees (default: 0.0)

--visualise: visualise the mould before export

--h: inline help
```

Help is available with: **`python src/mould_generator.py -h`**

The conversion can be automated (see README 'Batch Processing Script') with no interactive prompts, for batch processing on server computers. Every configuration is accessible through CLI.

The interactive notebook provides control for parameter tuning and visualisation. An interactive Jupyter Notebook is available in the notebooks/ directory.

1. Launch Jupyter Notebook as: jupyter notebook
2. Open notebooks/demo.ipynb to use interactive widgets for parameter tuning.
3. Execute cells to visualise and generate the mould.

The pipeline uses Python's logging module. Logs are human-readable and machine-parsable JSON summaries with the information below.

- Logging warnings if the input mesh is not watertight.
- Results of mesh repair attempts.
- Stepwise records of cavity creation, hole subtraction, and mould parting.

For automation and batch processing:

```
# Basic 2-piece mould

python makeMold.py chess_knight.stl



# Custom configuration

python makeMold.py chess_knight.stl --wall_thickness 3.0 --split_axis z
--mold_pieces 2 --num_alignment_keys 4



# 4-piece mould with draft angles

python makeMold.py large_sculpture.stl --mold_pieces 4 --draft_angle 2.0
--repair_method open3d
```

Usage Examples

Example 1: Simple Chess Piece Mould

Input: chess_knight.stl (small object, 25mm tall)

python makeMold.py chess_knight.stl --split_axis z

Output:

```
output/chess_knight_2-part/chess_knight_mold_bottom.stl

output/chess_knight_2-part/chess_knight_mold_top.stl

output/chess_knight_2-part/chess_knight_2-part_settings.txt
```

Result: 2-piece mould split horizontally, 2.1mm wall thickness (auto-calculated), 2 alignment keys, vertical pour spout.

Example 2: Large Decorative Object

Input: decorative_vase.stl (large object, 150mm tall)

python makeMold.py decorative_vase.stl --mold_pieces 4 --split_axis z --num_alignment_keys 4 --draft_angle 1.0

Output:

```
output/decorative_vase_4-part/decorative_vase_mold_bottom_left.stl
```

```
output/decorative_vase_4-part/decorative_vase_mold_bottom_right.stl

output/decorative_vase_4-part/decorative_vase_mold_top_left.stl

output/decorative_vase_4-part/decorative_vase_mold_top_right.stl
```

Result: 4-piece mould for easier handling of large objects, 1° draft angles for easier demolding.

Example 3: Problematic STL File

Input: broken_model.stl (non-watertight mesh with holes)

python makeMold.py broken_model.stl --repair_method hybrid

Process:

1. Detects non-watertight mesh
2. Applies hybrid repair (trimesh + Open3D)
3. Creates a mould with repaired geometry
4. Logs all repair steps for review

Example 4: Tiny Miniature

Input: miniature_detail.stl (very small object, 8mm)

python makeMold.py miniature_detail.stl --split_axis y

Automatic Adaptations:

- Extra-small object detection
- Reduced wall thickness (1.8mm)
- Smaller alignment keys (0.3mm radius)
- Conservative safety margins
- Precise positioning algorithms

⚙ Configuration Options

Split Axes

- **X-axis**: Left-right split (good for tall objects)
- **Y-axis**: Front-back split (good for long objects)
- **Z-axis**: Top-bottom split (good for flat objects)

Wall Thickness

- **Auto**: Intelligent calculation based on object size (recommended)
- **Manual**: Specify exact thickness in millimetres

- **Range**: 1.5mm - 25mm (automatically constrained)

Mould Pieces

- **2-piece**: Simpler printing and assembly
- **4-piece**: Better for large objects, easier handling

Alignment Keys

- **2 keys**: Diagonal placement for optimal stability
- **4 keys**: Maximum alignment precision

Draft Angles

- **None**: Most accurate reproduction
- **0.5-1.0°**: Slight taper for easier demolding
- **1.5-3.0°**: Easier demolding, less precision

Mesh Repair Methods

- **auto**: Smart detection and repair (recommended)
- **trimesh**: Basic built-in repair
- **open3d**: Advanced repair algorithms
- **hybrid**: Both trimesh and Open3D
- **none**: Skip repair (use only for perfect meshes)

🗁 Output Files

Each mould generation creates an organised output directory:

```
output/
└── model_name_2-part/
    ├── model_name_mold_bottom.stl    # Bottom mold piece
    ├── model_name_mold_top.stl       # Top mold piece
    └── model_name_2-part_settings.txt # Complete settings summary
logs/
└── moldmaker_model_name_20240101_123456.log  # Detailed technical log
```

Settings File Example

```
STL MOLD MAKER - SETTINGS SUMMARY
```

```
Generated: 2024-01-01 12:34:56


INPUT OBJECT

File: chess_knight.stl

Dimensions: 23.1 x 15.6 x 25.3 mm

Volume: 2847 mm³

Size Category: SMALL


MOLD CONFIGURATION

Type: 2-piece mold

Split Axis: Z-axis

Wall Thickness: 2.10 mm (CALCULATED)

User Input: Auto-calculated

Alignment Keys: 2

Mesh Repair: auto

Draft Angles: 1.5 degrees (Z-axis)


OUTPUT FILES

- chess_knight_mold_bottom.stl

- chess_knight_mold_top.stl


PRINTING TIPS

- Use 0.1-0.2mm layer height for smooth surfaces

- Print with 100% infill for strength
```

```
- Spray with oil or silicone spray to reduce friction

- Pour material through the bottom spout

- Alignment keys ensure proper assembly
```

🖨 *3D Printing Tips*

Print Settings

- **Layer Height**: 0.1-0.2mm for smooth cavity surfaces
- **Infill**: 100% for maximum strength
- **Support**: Usually not needed (moulds print cavity-up)
- **Print Speed**: Moderate (40-60mm/s) for quality

Materials

- **PLA**: Easy printing, good for low-temp casting
- **PETG**: Chemical resistance, higher temperature tolerance
- **ABS**: Acetone smoothing, very smooth surfaces
- **ASA**: UV resistance for outdoor use

Post-Processing

1. **Remove supports** if any were used
2. **Light sanding** of parting lines
3. **Apply release agent** (silicone spray, oil, or soap)
4. **Test fit** mould pieces before casting

🔧 *Troubleshooting*

Common Issues

"No STL files found"

- Place STL files in the current directory, models/, or Python/models/
- Check file extensions (.stl)

"Mesh is not watertight"

- Use --repair_method auto or open3d
- Check log files for repair details
- Consider manual mesh repair in Blender/Fusion360

"Alignment keys too close to cavity"

- The object might be too complex for automatic positioning

- Try a different split axis
- Check log files for detailed positioning analysis

"Boolean operation failed"

- Usually caused by mesh issues
- Try different repair methods
- Simplify the original STL if possible

Out of memory errors

- Reduce STL file complexity
- Close other applications
- Use 64-bit Python

Getting Help

1. **Check log files** in the logs/ directory for detailed error information
2. **Try different settings** (repair method, split axis)
3. **Simplify the STL file** if very complex
4. **Report issues** with minimal examples and log files

Advanced Examples

Batch Processing Script

```
import os

import subprocess

stl_files = ['piece1.stl', 'piece2.stl', 'piece3.stl']

for stl_file in stl_files:

    cmd = [

        'python', 'makeMold.py', stl_file,

        '--mold_pieces', '2',

        '--draft_angle', '1.0',

        '--repair_method', 'auto'

    ]

    subprocess.run(cmd)
```

```
    print(f"Completed {stl_file}")
```

Custom Wall Thickness by Size

```
# Small objects (< 20mm): thin walls

python makeMold.py small_detail.stl --wall_thickness 1.5

# Medium objects (20-50mm): standard walls

python makeMold.py medium_part.stl --wall_thickness 3.0

# Large objects (> 50mm): thick walls

python makeMold.py large_sculpture.stl --wall_thickness 5.0
```

Complex Multi-Part Projects

```
# Create complementary moulds for a multi-part assembly

python makeMold.py part_A.stl --split_axis x --mold_pieces 2

python makeMold.py part_B.stl --split_axis x --mold_pieces 2

python makeMold.py part_C.stl --split_axis y --mold_pieces 4
```

Best Practices

1. **Start with interactive mode** to understand options
2. **Use auto wall thickness** unless you have specific requirements
3. **Choose the split axis** based on object geometry and the desired parting line
4. **Enable draft angles** for easier demolding (1-2° is usually sufficient)
5. **Use mesh repair** unless you're certain your STL is perfect
6. **Print test pieces** with cheap filament first
7. **Keep detailed records** using the generated settings files

Specifications

- **Maximum STL Size**: Limited by available RAM (tested up to 50MB files)
- **Minimum Object Size**: 5mm (smaller objects may need manual thickness)
- **Wall Thickness Range**: 1.5mm - 25mm
- **Draft Angle Range**: 0.5° - 3.0°
- **Alignment Key Count**: 2 or 4
- **Split Axes**: X, Y, or Z
- **Mould Pieces**: 2 or 4

Contributing

Contributions are welcome! Please:

1. Fork the repository
2. Create a feature branch
3. Make your changes
4. Test with various STL files
5. Submit a pull request

License

This project is licensed under the MIT License - see the LICENSE file for details.

Acknowledgments

- Built with Trimesh for mesh processing
- Open3D for advanced mesh repair
- NumPy and SciPy for numerical computations
- OpenCV for image processing in EDT algorithms

G2.2 API Quick Start

Pipeline Walkthrough

The CRAEFT pipeline supports a staged learning experience:

1. Input STL - Load a model (e.g., a chess knight).
2. Geometry Analysis - Detect features that challenge manufacturability.
3. Wall Thickness - Compute optimal t∗t^* using dimensional heuristics.
4. Draft Angle - Apply adaptive tapering for smoother ejection.
5. Cavity Generation - Subtract the object from a padded block.
6. Splitting - Generate halves or quarters.
7. Alignment Keys & pouring basin- Add features for repeatability and casting flow.
8. Output - Export printable moulds and review logs.

Key API Calls

The key functions in the module src/mould_generator.py are:

`mesh_repair(M)`	**Repairs non-watertight meshes using Open3D routines.**
Parameters	• trimesh.Trimesh: input mesh.
Returns	trimesh.Trimesh repaired.
`apply_draft(M, θ)`	**Applies a taper to cavity walls.**

Parameters	• trimesh.Trimesh input mesh. • draft_angle (float): Draft angle in degrees.
Returns	A new trimesh.Trimesh object with the applied draft.
`process_model()`	**Processes an STL file.**
Parameters	• trimesh.Trimesh input mesh. • draft_angle (float): Draft angle in degrees. • Mould parts as STL files.
Returns	Exports 3D printing files of mould parts.
`process_models()`	**Processes all STL files in a directory.**
Parameters	• Directory name • draft_angle (float): Draft angle in degrees.
Returns	mould parts as STL files.
`create_mould()`	**Computes the mould part representations. 1) Loads and repairs the input mesh. 2) Applies draft. 3) Suggests venting/injection placement. 4) Parting the mould representation in halves or quarters.**
Parameters:	• mould (trimesh.Trimesh): The complete mould. • input_model_path (str): Path to the input STL file. • padding (float): Extra space added around the model. • hole_positions (list): Positions for holes (e.g., 'bottom'). • split_mode (str): 'halves' or 'quarters'. • draft_angle (float): Draft angle in degrees.
Returns	A tuple containing: parts (list): List of mould parts.

Examples

Example 1: Command-Line Usage for a Single File

```
python src/mould_generator.py --input models/MAOI03b.stl --output output2101 --padding 0.1 --hole_positions bottom --split_mode quarters --draft_angle 0.0 --visualize
```

Example 2: Processing All STL Files in a Directory

```
python src/mould_generator.py --input models/ --output output2101 --padding 0.1 --hole_positions bottom --split_mode quarters --draft_angle 0.0
```

Draft Angle Reference by Material

Material	Recommended Draft Angle
PE, PP, PVC	~1° (standard range: 0.5°-2°)
ABS, PA, POM, PPO	~1.5° (1°-2° typical)
PC, PSF, PMMA, AS	~2° (increase if textured)
PET	0.5°-1°
PS	0.5° minimum, 1° typical
LCP	Liquid Crystal Polymers ≥0.5° (avoid inverted designs)
PBT	0.5°-1°, add +1° if etched
General guidance	Add 1° draft for every 0.025-0.03 mm of texture depth. Increase draft for rough, etched, or patterned surfaces.

G.3 Size Calibration Quick Guide

Metric values for the quantities employed in the experimental configuration are provided.

The value of β varies according to the printer profile or user choice. This value was tuned based on over 200 tests. The table shows the values used in the presented experiments.

Category	d (mm)	β
Tiny	d<8	0.25
Small	8≤d<15	0.15
Medium	15≤d<50	0.12
Large	d≥50	0.10

The values of α_s are modulated according to d.

Model size	α_s

Large	$A \geq 2d^2$	1.2
Small	$A < ½d^2$	0.9
Medium	else	1.0

The values of α_s are modulated according to the number of pieces.

Parting	**α_p**
Quadripartite	1.15
Bipartite	1.0

The values of κ and λ are modulated according to the number of pieces.

	κ	**λ**	**d (mm)**
Tiny	⅙· t	0.6	d < 8
Small	⅕· t	0.8	8 ≤ d < 15
Other	¼· t	1.0	d ≤ 15

The printing clearance, γ, is modulated according to the key radius.

γ (mm)	**r_k (mm)**
0.15	$r_k < 1.0$
0.30	1.0 $r_k < 2.0$
0.40	$r_k > 2.0$

These values trade demoulding ease (small parts are fragile) against dimensional fidelity (large parts tolerate steeper angles). When the user leaves the draft unspecified, the default draft angle θ_a is:

θ_a (°)	**d (mm)**
1.0	d < 15
1.5°	15 ≤ d < 50
2.0°	d ≥ 50

Industry guidelines recommend 0.5°-3°.

Penetration Δz\Delta zΔz scales with part height, clipped between 2 mm and 25 mm to remain printable.

Wall-thickness adaptation was tested on more than 200 parts. The method adapts to size category and mould configuration, applying modifiers for large split planes and multi-piece assemblies. Bounds derived from manufacturing constraints ensure that the final value t does not intrude into the cavity. N = 400 in this work. Specifically, the system determines the following outputs.

Quantity	Value	Note
d	25.3 mm	medium
t_0	3.0 mm	$\beta = 0.12$
A_s	560 mm^2	medium plane $\alpha_s = 1$
p	2	$\alpha_p = 1$
t_c	3.0 mm	
t_0	2.8 mm	printability prevails
t_1	5.7 mm	0.25· d
t	3.0 mm	Accepted

This example illustrates how the system calculates and applies the wall thickness, which then feeds into downstream calculations such as key radius and pouring basin radius. The example illustrates how the algorithm balances printability, strength, and material economy, while guaranteeing that $B(t^*) \cap M = \emptyset$.

References

1. Liang, H. (2012). Advances in multispectral and hyperspectral imaging for archaeology and art conservation. Applied Physics A, 106, 309-323. https://doi.org/10.1007/s00339-011-6689-1
2. Aila, T., & Laine, S. (2009). Understanding the efficiency of ray traversal on GPUs. Proceedings of the Conference on High Performance Graphics 2009, 145-149. https://doi.org/10.1145/1572769.1572792
3. Jensen, H. W. (2001). Realistic Image Synthesis Using Photon Mapping. A K Peters. ISBN: 1568811470
4. Debevec, P. (2008). Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In ACM SIGGRAPH 2008 Classes (pp. 1-10). https://doi.org/10.1145/280814.280864
5. Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., & Ng, R. (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Proceedings of the European Conference on Computer Vision (ECCV), 405-421. https://doi.org/10.1145/3503250
6. Park, J. J., Sinha, S. N., Efros, A. A., & Snavely, N. (2021). Deformable Neural Radiance Fields. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 5791-5800. 10.1109/ICCV48922.2021.00581
7. Igarashi, T., Igarashi, Y., & Zorin, D. (2007). Interactive design of period styles for textile patterns. ACM Transactions on Graphics, 26(3), 1-7. https://doi.org/10.1145/3272127.3275105
8. Baxter, W. V., Scheib, V., Lin, M. C., & Manocha, D. (2001). DAB: Interactive haptic painting with 3D virtual brushes. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 461-468. https://doi.org/10.1145/383259.383313
9. Yunsheng Tian, Jie Xu, Yichen Li, Jieliang Luo, Shinjiro Sueda, Hui Li, Karl D. D. Willis, and Wojciech Matusik. 2022. Assemble Them All: Physics-Based Planning for Generalizable Assembly by Disassembly. ACM Trans. Graph. 41, 6, Article 278. https://doi.org/10.1145/3550454.3555525
10. Eberly, D.H. 2004. 'Game Physics', Morgan Kaufmann Publishers. ISBN 0123749034.
11. Kaufman, D., Edmunds, T. and D. Pai, 2010, 'Fast Frictional Dynamics for Rigid Bodies', ACM SIGGRAPH 2005. https://doi.org/10.1145/1073204.1073295
12. Trzepieciński T, dell'Isola F, Lemu HG. Multiphysics Modeling and Numerical Simulation in Computer-Aided Manufacturing Processes. Metals. 2021; 11(1):175. https://doi.org/10.3390/met11010175
13. Guru Prashanth Balasubramanian, Eli Saber, Vladimir Misic, Eric Peskin, Mark Shaw, Unsupervised color image segmentation using a dynamic colour gradient thresholding algorithm, Volume 6806, Human Vision and Electronic Imaging XIII; 68061H (2008) https://doi.org/10.1117/12.766184
14. Satoshi Suzuki and others. Topological structural analysis of digitised binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985. https://doi.org/10.1016/0734-189X(85)90016-7
15. Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. ACM Trans. Graph. 35, 4, Article 39. https://doi.org/10.1145/2897824.2925901
16. Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr Jit: A Just-In-Time Compiler for Differentiable Rendering. In Transactions on Graphics (Proceedings of SIGGRAPH) 41(4). https://doi.org/10.1145/3528223.3530099

17. F O Bartell, E. L. Dereniak, W. L Wolfe, 'The Theory and Measurement of Bidirectional Reflectance Distribution Function and Bidirectional Transmittance Distribution Function' Proc. SPIE 0257, Radiation Scattering in Optical Systems, (3 March 1981); https://doi.org/10.1117/12.959611
18. Seung-Hwan Baek, Tizian Zeltner, Hyun Jin Ku, Inseung Hwang, Xin Tong, Wenzel Jakob und Min H. Kim. 2020. Image-based acquisition and modeling of polarimetric reflectance. ACM Trans. Graph. 39, 4, Article 139 (August 2020), 14 pages. https://doi.org/10.1145/3386569.3392387
19. Delaunay, Boris (1934). 'Sur la sphère vide' [On the empty sphere]. Bulletin de l'Académie des Sciences de l'URSS, Classe des Sciences Mathématiques et Naturelles (in French). 6: 793–800.
20. Dassault Systèmes, SIMULIA Simulation Software. https://www.3ds.com/products/simulia 2024.
21. NVIDIA PhysX developer site. https://developer.nvidia.com/physx-sdk 2024
22. NVIDIA Omniverse: The platform for connecting and developing OpenUSD applications. Available at: https://www.nvidia.com/en-us/omniverse/ 2024.
23. Collins, J., Chand, S., Vanderkop, A., Howard, D., 2021. A Review of Physics Simulators for Robotic Applications. IEEE Access 9, 51416–51431. https://doi.org/10.1109/ACCESS.2021.3068769
24. Marcin Kalicinski and Sebastian Redl, 2024. Boost Property Tree Library, https://github.com/boostorg/property_tree.
25. Open Asset Import Library, https://github.com/assimp/assimp, 2024.
26. Shen, B., Jiang, Z., Choy, C., Guibas, L.J., Savarese, S., A. Kumar, A., Zhu, Y., 2022. ACID: Action-Conditional Implicit Visual Dynamics for Deformable Object Manipulation. https://doi.org/10.15607/rss.2022.xviii.001
27. T. Tzathas, P. Maragos, and A. Roussos, 3D Neural Sculpting (3DNS): Editing Neural Signed Distance Functions, IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), January 2023. https://doi.org/wacv56688.2023.00450
28. Wavefront .obj file, 2024. https://en.wikipedia.org/wiki/Wavefront_.obj_file. Wikipedia.
29. Courant, R. (1943). Variational methods for the solution of problems of equilibrium and vibrations. Bulletin of the American Mathematical Society, 49(1), 1-23.
30. Clough, R. W. (1960). The finite element method in plane stress analysis. Proceedings of the 2nd ASCE Conference on Electronic Computation, 8, 345-378.
31. Zienkiewicz, O. C., & Taylor, R. L. (2000). The Finite Element Method. Butterworth-Heinemann.
32. MacNeal, R. H. (1994). NASTRAN Theoretical Manual. MacNeal-Schwendler Corporation.
33. Bathe, K. J. (1996). Finite Element Procedures. Prentice Hall.
34. Cook, R. D., Malkus, D. S., & Plesha, M. E. (1989). Concepts and Applications of Finite Element Analysis. John Wiley & Sons.
35. Hibbitt, H. D., & Karlsson, B. I. (1978). ABAQUS User's Manual. Hibbitt, Karlsson & Sorensen, Inc.
36. Moaveni, S. (1999). Finite Element Analysis: Theory and Application with ANSYS. Prentice Hall.
37. Anderson, J. D. (1995). Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill.
38. Versteeg, H. K., & Malalasekera, W. (2007). An Introduction to Computational Fluid Dynamics: The Finite Volume Method. Pearson Education.
39. Dawson, C. (1993). Doom Game Engine. id Software.
40. Havok. (2004). Havok Physics Engine. Havok.
41. Hecker, C. (2010). Physics simulation in games. In Game Developer Conference Proceedings.
42. NVIDIA. (2008). NVIDIA PhysX Technology Overview. NVIDIA Corporation.
43. Slater, M., & Sanchez-Vives, M. V. (2016). Enhancing our lives with immersive virtual reality. Frontiers in Robotics and AI, 3, 74.
44. Mendiburu, B. (2020). 3D TV and 3D Cinema: Tools and Processes for Creative Stereoscopy. CRC Press.
45. Susi, T., Johannesson, M., & Backlund, P. (2007). Serious games: An overview. Technical Report HS-IKI-TR-07-001. University of Skövde.

46. Thompson, J. (2019). Level of Detail for 3D Graphics. Morgan Kaufmann.
47. Waveren, J. V. (2016). Real-time AI for Games: Techniques and Applications. CRC Press.
48. D. Pye, The nature and art of workmanship, Cambridge University Press, London, 1968.
49. Keller C, Keller J. Imagery in cultural tradition and innovation. Mind Cult Act. 1999;6(1):3-32. https://doi.org/10.1080/10749039909524711
50. Aktas B, Mäkelä M. Negotiation between the Maker and Material: Observations on Material Interactions in Felting Studio. International Journal of Design. 2019.
51. Cutsuridis, V, Taylor, J. A Cognitive Control Architecture for the Perception-Action Cycle in Robots and Agents. Cognitive Computation, 2013;5:383–395 https://doi.org/10.1007/s12559-013-9218-z.
52. Raspall F. Design with Material Uncertainty: Responsive Design and Fabrication in Architecture. In: Thomsen M, Tamke M, Gengnagel C, Faircloth B, Scheurer F, editors. Modelling Behaviour. Cham: Springer; 2015. https://doi.org/10.1007/978-3-319-24208-8_27
53. Iyobe M, Ishida T, Miyakawa A, Sugita K, Uchida N, Shibata Y. Development of a mobile virtual traditional crafting presentation system using augmented reality technology. Int J Space-Based Situat Comput. 2017;6(4):239–251.
54. F Fuchigami R, Ishida T. Proposal of a Traditional Craft Simulation System Using Mixed Reality. In: Barolli L, Takizawa M, Yoshihisa T, Amato F, Ikeda M, editors. Advances in P2P, Parallel, Grid, Cloud and Internet Computing. Cham: Springer; 2021. https://doi.org/10.1007/978-3-030-61105-7_32
55. Edlin L, Liu Y, Bryan-Kinns N, Reiss J. Exploring Augmented Reality as Craft Material. In: Stephanidis C, Chen JYC, Fragomeni G, editors. HCI International 2020 – Late Breaking Papers: Virtual and Augmented Reality. Lecture Notes in Computer Science, vol 12428. Cham: Springer; 2020. https://doi.org/10.1007/978-3-030-59990-4_5
56. Agelada A, Rizopoulos G, Flamos I, Kasapakis V. Users as Craftspeople: Demonstrating Traditional Crafts using Interactive Immersive Virtual Reality. 2022 IEEE International Conference on Artificial Intelligence and Virtual Reality; 2022. p. 245-247.
57. Rossau I, Skovfoged M, Czapla J, Sokolov M, Rodil K. Dovetailing: safeguarding traditional craftsmanship using virtual reality. International Journal of Intangible Heritage. 2019;14:104-120.
58. Hägele M, Nilsson K, Pires JN, Bischoff R. Industrial Robotics. In: Siciliano B, Khatib O, editors. Springer Handbook of Robotics. Springer Handbooks. Cham: Springer; 2016. https://doi.org/10.1007/978-3-319-32552-1_54
59. Guo Z, Zhang Z, Li C. Robotic Carving Craft, Research on the Application of Robotic Carving Technology in the Inheritance of Traditional Carving Craft. Proceedings of the CAADRIA Conference; 2022 Apr 9-15; Sydney. p. 747-756. https://doi.org/10.52842/conf.caadria.2022.1.747
60. Shaked T, Bar-Sinai KL, Sprecher A. Adaptive robotic stone carving: Method, tools, and experiments. Automation in Construction. 2021;129:103809. https://doi.org/10.1016/j.autcon.2021.103809
61. Brugnaro G, Hanna S. Adaptive Robotic Training Methods for Subtractive Manufacturing. In: Acadia 2017 Disciplines & Disruption: Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture. Cambridge, MA: Acadia Publishing Company; 2017. p. 164-169 https://doi.org/10.52842/conf.acadia.2017.164
62. Brugnaro G, Hanna S. Adaptive Robotic Carving. In: Willmann J, Block P, Hutter M, Byrne K, Schork T, editors. Robotic Fabrication in Architecture, Art and Design 2018. Cham: Springer; 2019. https://doi.org/10.1007/978-3-319-92294-2_26
63. Brugnaro G, Figliola A, Dubor A. Negotiated Materialization: Design Approaches Integrating Wood Heterogeneity Through Advanced Robotic Fabrication. In: Bianconi F, Filippucci M, editors. Digital Wood Design. Lecture Notes in Civil Engineering, vol 24. Cham: Springer; 2019. https://doi.org/10.1007/978-3-030-03676-8_4
64. Google, 3D Pottery, Available from: https://experiments.withgoogle.com/3d-pottery (Accessed on 29 July 2024).

65. Eyewind, Pottery Master, Available from: https://play.google.com/store/apps/details?id=com.create.pottery.paint.by.color (Accessed on 29 July 2024).
66. AZ Games. Master of Pottery, Available from: https://store.steampowered.com/app/1160490/Master_Of_Pottery/ (Accessed on 29 July 2024).
67. Grow, A., Dickinson, M., Pagnutti, J., Wardrip-Fruin, N., Mateas, M. (2017). Crafting in games. Digital Humanities Quarterly, 11(4).
68. Karastone, Knitting Simulator 2014. Available from: https://karastonesite.com/2014/07/19/knitting-simulator-2014/ (Accessed on 29 July 2024).
69. Chotrov D, Uzunova Z, Maleshkov S. Real-time 3D model topology editing method in VR to simulate crafting with a wood-turning lathe. 2019.
70. Xu R, Xu J. Research on Interactive Traditional Craft Diagram Model and Simulation System: Take Nanjing Rong Hua Craft as an Example. In: Proceedings of the International Conference on Advances in Energy, Environment and Chemical Engineering; 2016. p. 261-265. https://doi.org/10.2991/aeece-16.2016.54
71. NVIDIA, PhysX System Software, Available from: https://developer.nvidia.com/physx-sdk (Accessed on 29 July 2024).
72. The Irregular Corporation, Woodwork Simulator, 2019. Available from: https://steamdb.info/app/510230/info/ (Accessed on 29 July 2024).
73. Murray J, Sawyer W. Virtual Crafting Simulator: Teaching Heritage Through Simulation. In: Proceedings of the International Conference on Education and New Learning Technologies; 2015. p. 7668-7675.
74. Eglash, R. Ethnocomputing with Native American Design. In: Dyson LE, Hendriks M, Grant S, editors. Information Technology and Indigenous People. IGI Global, 2007. p. 210-219. https://doi.org/10.4018/978-1-59904-298-5.ch029
75. Carre A, Dubois A, Partarakis N, Zabulis X, Patsiouras N, Mantinaki E, et al. Mixed-Reality Demonstration and Training of Glassblowing. Heritage. 2022;5(1):103-128. https://doi.org/10.3390/heritage5010006
76. Canyon Art, WeaveIt, Available from: http://www.weaveit.com/ (Accessed on 29 July 2024).
77. Fiberworks, Fiberworks PCW, Available from: http://www.fiberworks-pcw.com/ (Accessed on 29 July 2024).
78. Arahne, ArahneWeave, Available from: http://www.arahne.si/ (Accessed on 29 July 2024).
79. pixeLoom, Available from: http://www.pixeloom.com/ (Accessed on 29 July 2024).
80. WeavePoint, Available from: http://www.weavepoint.com/ (Accessed on 29 July 2024).
81. Eisenstein, J., Softweave, WIF Visualizer, https://softweave.com/software/wif-visualizer/ (Accessed on 29 July 2024).
82. EAT - The DesignScope Company, Available from: https://www.designscopecompany.com/simulation-knitting/77/the-art-knitting (Accessed on 29 July 2024).
83. Torii T. Hand-painted yuzen-dyeing simulation for online handicraft experience. Computer Animation and Virtual Worlds. 2023;e2200. https://doi.org/10.1002/cav.2200
84. Xu S, Mei X, Dong W, Zhang Z, Zhang X. Real-time ink simulation using a grid-particle method. Computer Graphics. 2012;36:1025-1035.
85. Nam-Ho K, Bhavani S, Ashok K. Introduction to Finite Element Analysis and Design. Wiley; 2018.
86. Fish J, Belytschko T, A First Course in Finite Elements, 2007, Wiley.
87. Baek C, Johanns P, Sano TG, Grandgeorge P, Reis PM. Finite Element Modeling of Tight Elastic Knots. Journal of Applied Mechanics. 2021;88(2):024501. https://doi.org/10.1115/1.4049023

88. Crassous J. Discrete-element-method model for frictional fibers. Physics Review E. 2023;107(2):025003. https://doi.org/10.1103/PhysRevE.107.025003
89. Inoue T. Tatara and the Japanese sword: the science and technology. Acta Mechanica. 2010;214:17–30. https://doi.org/10.1007/s00707-010-0308-7
90. He Z, Xu J, Tran KP, et al. Modeling of textile manufacturing processes using intelligent techniques: a review. International Journal of Advanced Manufacturing Technology. 2021;116:39–67. https://doi.org/10.1007/s00170-021-07444-1
91. Orlik J, Krier M, Neusius D, Pietsch K, Sivak O, Steiner K. Recent Efforts in Modeling and Simulation of Textiles. Textiles. 2021;1(2):322-336. https://doi.org/10.3390/textiles1020016.
92. Xie J, Guo Z, Shao M, Zhu W, Jiao W, Yang Z, et al. Mechanics of textiles used as composite preforms: A review. Composite Structures. 2023;304(2):116401. https://doi.org/10.1016/j.compstruct.2022.116401.
93. Fang G, Liang J. A review of numerical modeling of three-dimensional braided textile composites. Journal of Composite Materials. 2011;45(23):2415-2436. https://doi.org/10.1177/0021998311401093.
94. Li Y, Du T, Wu K, Xu J, Matusik W. DiffCloth: Differentiable Cloth Simulation with Dry Frictional Contact. ACM Transactions on Graphics. 2022;42(1):2. https://doi.org/10.1145/3527660.
95. Boisse P, Hamila N, Vidal-Sallé E, Dumont F. Simulation of wrinkling during textile composite reinforcement forming. Composites Science and Technology. 2011;71(5):683-692. https://doi.org/10.1016/j.compscitech.2011.01.011
96. Durville D. Simulation of the mechanical behaviour of woven fabrics at the scale of fibers. International Journal of Material Forming. 2010;3(Suppl 2):1241–1251. https://doi.org/10.1007/s12289-009-0674-7
97. Brown L. TexGen. In: Kyosev Y, Boussu F, editors. Advanced Weaving Technology. Cham: Springer; 2022. p. 253-291.
98. Research & Education Unit, Cal/OSHA Consultation Service, California, Department of Industrial Relations and the National Institute for Occupational Safety and Health. Easy Ergonomics: A Guide to Selecting Non-Powered Hand Tools. California Department of Industrial Relations and the National Institute for Occupational Safety and Health; 2004. Publication No. 2004-164.
99. Radwin R, Haney J. An Ergonomics Guide to Hand Tools. 1996. https://doi.org/10.13140/RG.2.1.4761.5446
100. Dababneh A, Lowe B, Krieg E, Kong Y, Waters T. A Checklist for the Ergonomic Evaluation of Nonpowered Hand Tools. Journal of Occupational and Environmental Hygiene. 2005;1. https://doi.org/10.1080/15459620490883150
101. Uicker J, Pennock G, Shigley J. Theory of machines and mechanisms. New York: Oxford University Press, 2011. ISBN 978-0-19-537123-9.
102. Bowser EA. An elementary treatise on analytic mechanics: with numerous examples. New York: D. Van Nostrand Company; 1884. p. 202–203.
103. Lemaître J, Desmorat R. Engineering Damage Mechanics: Ductile, Creep, Fatigue, and Brittle Failures. Springer; 2005.
104. Hashin Z. Failure criteria for unidirectional fiber composite. Journal of Applied Mechanics. 47(2):329-334. https://doi.org/10.1115/1.3153664
105. Anderson T. Fracture Mechanics: Fundamentals and Applications. 4th ed. CRC Press; 2017.
106. Xu X, Needleman A. Numerical Simulations of Fast Crack Growth in Brittle Solids. Journal of the Mechanics and Physics of Solids. 1994;42(9):1397-1434.
107. Virigiri VKR, Gudiga VY, Gattu US, Suneesh G, Buddaraju KM. A review on the Johnson-Cook material model. Materials Today: Proceedings. 2022;62(6):3450-3456. https://doi.org/10.1016/j.matpr.2022.04.279

108. Johnson GR, Cook WH. Fracture Characteristics of Three Metals Subjected to Various Strains, Strain Rates, Temperatures and Pressures. 1985.
109. Chen W, Han D. Plasticity for Structural Engineers. Springer; 1988.
110. Drucker D, Prager W. Soil Mechanics and Plastic Analysis or Limit Design. Quarterly of Applied Mathematics. 1952;10:157-165.
111. Wood DM. Soil Behaviour and Critical State Soil Mechanics. Cambridge: Cambridge University Press; 1990.
112. Frost H, Ashby M. Deformation-Mechanism Maps: The Plasticity and Creep of Metals and Ceramics. Pergamon Press; 1982.
113. Nabarro F, de Villiers H. The Physics of Creep: Creep and Creep-Resistant Alloys. Taylor & Francis; 1995.
114. Versteeg H, Malalasekera W. An Introduction to Computational Fluid Dynamics: The Finite Volume Method. 2nd ed. Pearson Education, 2007.
115. Glotzer S, Rapaport D. Computer Simulation of Liquids. Oxford University Press; 2004.
116. Dantzig J, Rappaz M. Solidification. EPFL Press; 2009.
117. Chao-Yang W, Beckermann C. A two-phase mixture model of liquid-gas flow and heat transfer in capillary porous media—I. Formulation. International Journal of Heat and Mass Transfer. 1993;36(11):2747-2758. https://doi.org/10.1016/0017-9310(93)90094-M
118. Ferziger J, Perić M. Computational Methods for Fluid Dynamics. Springer; 2002.
119. Lewis R, Morgan K, Thomas H, Seetharamu K. The Finite Element Method in Heat Transfer Analysis. John Wiley & Sons, 1996.
120. Çengel Y, Ghajar A. Heat and Mass Transfer: Fundamentals and Applications. 5th ed. McGraw-Hill Education; 2014.
121. ISO. ISO 10303-21:2016 Industrial automation systems and integration – Product data representation and exchange – Part 21: Implementation methods: Clear text encoding of the exchange structure. Geneva: International Organization for Standardisation; 2016.
122. Library of Congress. Document ID: fdd000507. 2020 Jan 23.
123. Tsai M, Prindible M. Wood Gouge-Capturing Human Skill. 2019. Available from: https://courses.ideate.cmu.edu/16-455/s2019/index.html%3Fp=973.html (Accessed on 29 July 2024)
124. Harrison C, Childs H, Gaither K. Data-Parallel Mesh Connected Components Labeling and Analysis. Eurographics Symposium on Parallel Graphics and Visualization; 2011. https://doi.org/10.2312/EGPGV/EGPGV11/131-140
125. He L, Ren X, Gao Q, Zhao X, Yao B, Chao Y. The connected-component labeling problem: A review of state-of-the-art algorithms. Pattern Recognition. 2017;70:25-43. https://doi.org/10.1016/j.patcog.2017.04.018
126. Seymour J. The Forgotten Arts: A practical guide to traditional skills. Angus & Robertson Publishers; 1984. p. 54. ISBN 0-207-15007-9.
127. Khaledi K, Rezaei S, Wulfinghoff S, Reese S, A microscale finite element model for joining of metals by large plastic deformations, Comptes Rendus Mécanique, Volume 346, Issue 8, 2018, Pages 743-755, ISSN 1631-0721, https://doi.org/10.1016/j.crme.2018.05.005
128. Kumar RK, Babu AS. Finite element analysis and experimental study on metal joining by mechanical crimping. International Journal of Service and Computing Oriented Manufacturing. 2014;1:295-306. https://doi.org/10.1504/IJSCOM.2014.066489
129. Mui G, Wu X, Hu K, Yeh C, Wyatt K. Solder joint formation simulation and finite element analysis. 1997 Proceedings 47th Electronic Components and Technology Conference; 1997; San Jose, CA, USA. p. 436-443. https://doi.org/10.1109/ECTC.1997.606207

130. Anca A, Cardona A, Risso J, Fachinotti V. Finite element modeling of welding processes. Applied Mathematical Modelling. 2011;35(2):688-707. https://doi.org/10.1016/j.apm.2010.07.026
131. Cao J, Akkerman R, Boisse P, Chen J, Cheng H, de Graaf E, et al. Characterization of mechanical behavior of woven fabrics: Experimental methods and benchmark results. Composites Part A: Applied Science and Manufacturing. 2008;39(6):1037-1053. https://doi.org/10.1016/j.compositesa.2008.02.016
132. Vilfayeau J, Crépin D, Boussu F, Soulat D, Boisse P. Kinematic modelling of the weaving process applied to 2D fabric. Journal of Industrial Textiles. 2015;45(3):338-351. https://doi.org/10.1177/1528083714532114
133. Digital Image Processing (Third Edition) by Rafael C. Gonzalez and Richard E. Woods, ISBN 978-93-325-7032-0 (2008).
134. Nikhil R Pal, Sankar K Pal, A review on image segmentation techniques, Pattern Recognition, Volume 26, Issue 9, 1993, Pages 1277-1294, ISSN 0031-3203, https://doi.org/10.1016/0031-3203(93)90135-J.
135. S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz and D. Terzopoulos, 'Image Segmentation Using Deep Learning: A Survey,' in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 7, pp. 3523-3542, 1 July 2022, https://doi.org/10.1109/TPAMI.2021.3059968
136. Yang, L., Kang, B., Huang, Z., Xu, X., Feng, J., & Zhao, H. (2024). Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data. ArXiv, abs/2401.10891.
137. Lloyd, Stuart P., 'Least squares quantization in PCM.' Information Theory, IEEE Transactions on 28.2 (1982): 129-137.
138. (2008). Median Cut Algorithm. In: Furht, B. (eds) Encyclopedia of Multimedia. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-78414-4_36
139. Burley, B. Physically-Based Shading at Disney. In: Practical Physically Based Shading in Film and Game Production (ACM SIGGRAPH 2012 Course Notes). 2012. Available from: https://blog.selfshadow.com/publications/s2012-shading-course/ (Accessed on 20 January 2026).
140. Han, Y.C.; Han, B.-J. Virtual pottery: a virtual 3D audiovisual interface using natural hand motions. Multimedia Tools and Applications. 2014;73(2):917–933. https://doi.org/10.1007/s11042-013-1382-3
141. Han, G.; Lee, S.; Lee, J.; Park, Y. Experiencing Pottery Making in the Augmented Space. In: Shumaker, R. (ed.) Virtual Reality (ICVR 2007). Lecture Notes in Computer Science, vol. 4563. Springer, Berlin, Heidelberg; 2007. https://doi.org/10.1007/978-3-540-73335-5_69
142. Krishnamurthy, V.R.; Ramani, K.; Jasti, K.L.R. zPots: A Virtual Pottery Experience with Spatial Interactions Using the Leap Motion Device. In: CHI '14 Extended Abstracts on Human Factors in Computing Systems. 2014:371–374. https://doi.org/10.1145/2559206.2574834
143. Chiang, P.-Y.; Chang, H.-Y.; Chang, Y.-J. PotteryGo: a virtual pottery making training system. IEEE Computer Graphics and Applications. 2018;38(2):74–88. https://doi.org/10.1109/MCG.2018.021951634
144. Cai, R.; Lin, Y.-Y.; Li, H.; Zhu, Y.; Tang, X.; Weng, Y.; You, L.; Jin, X. Wowtao: A personalized pottery-making system. Computers in Industry. 2021;124:103325. https://doi.org/10.1016/j.compind.2020.103325
145. Gao, Z.; Li, J.; Wang, H.; Feng, G. DigiClay: An interactive installation for virtual pottery using motion sensing technology. In: Proceedings of the 4th International Conference on Virtual Reality. 2018:126–132.
146. McDonnell, K.T.; Qin, H.; Wlodarczyk, R.A. Virtual Clay: A real-time sculpting system with force feedback. In: Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D '01). 2001:179–190. https://doi.org/10.1145/364338.364395

147. VirtualGlass. VirtualGlass software (glass cane design and visualisation). Available from: https://virtualglass.org/ (Accessed on 20 January 2026). See also: MIT CSAIL Alliances, "Virtual Glass / VirtualGlass" (Accessed on 20 January 2026): https://cap.csail.mit.edu/members/research/open-source-technologies/virtualglass
148. Cho, S., Heo, Y., & Bang, H. (2012). Turn: A virtual pottery by real spinning wheel. In ACM SIGGRAPH 2012 Emerging Technologies. Association for Computing Machinery. https://doi.org/10.1145/2343456.2343481
149. Han, Y. C., & Han, B.-J. (2014). Virtual pottery: a virtual 3D audiovisual interface using natural hand motions. Multimedia Tools and Applications, 73(2), 917–933. https://doi.org/10.1007/s11042-013-1382-3.
150. Gao, Z., et al. (2018). DigiClay: An Interactive Installation for Virtual Pottery. In ICVR 2018 (Feb 24–26, 2018, Hong Kong). Association for Computing Machinery. https://doi.org/10.1145/3198910.3234659.
151. Matsumaru, T., & Morikawa, A. (2020). An Object Model and Interaction Method for a Simulated Experience of Pottery on a Potter's Wheel. Sensors, 20(11), 3091. https://doi.org/10.3390/s20113091.
152. Cai, R., Lin, Y., Li, H., Zhu, Y., Tang, X., Weng, Y., You, L., & Jin, X. (2021). Wowtao: A personalized pottery-making system. Computers in Industry, 124, 103325. https://doi.org/10.1016/j.compind.2020.103325.
153. Levy, R. M., & Dawson, P. (2009). Using finite element methods to analyze ancient architecture: an example from the North American Arctic. Journal of Archaeological Science, 36(10), 2298–2307. https://doi.org/10.1016/j.jas.2009.06.014.
154. Rojas-Sola, J. I., Bouza-Rodríguez, J. B., & Comesaña-Campos, A. (2019). Study of ancient forging devices: 3D modelling and analysis using current computer methods. Bulletin of the Polish Academy of Sciences: Technical Sciences, 67(2), 377–390. https://doi.org/10.24425/bpas.2019.127963.
155. VirtualGlass Team. (n.d.). VirtualGlass (software project site). Retrieved 20 January 2026, from VirtualGlass. & Winslow, A., Baldauf, K., Lee, B.L., McCann, J., Demaine, E.D., Demaine, M.L., & Houk, P. (2015). Virtual Cane Creation for Glassblowers. AND https://erikdemaine.org/theses/kbaldauf.pdf
156. Wyszecki, G., & Stiles, W. S. (2000). Color science: Concepts and methods, quantitative data and formulae (2nd ed.). Wiley-Interscience.
157. Serra, J. (1982). Image analysis and mathematical morphology (Vol. 1). Academic Press.
158. MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281–297. University of California Press.
159. Ikotun, A. M., Adebisi, B., Hammoudeh, M., & Anoh, K. (2023). K-means clustering algorithms: A comprehensive review. Information Sciences, 622, 178–210. https://doi.org/10.1016/j.ins.2022.12.070
160. Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics, 20, 53–65. https://doi.org/10.1016/0377-0427(87)90125-7
161. Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4th ed.). Pearson.
162. Nock, R. & Nielsen, F. 'Statistical Region Merging,' IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 11, pp. 1452–1458, 2004areas.
163. NVIDIA PhysX developer site. https://developer.nvidia.com/physx-sdk 2024
164. J. J. Gibson, "The Ecological Approach to Visual Perception". Boston, MA, USA: Houghton Mifflin, 1979.

165. J. Aloimonos, I. Weiss, and A. Bandyopadhyay, "Active vision," International Journal of Computer Vision, vol. 1, pp. 333–356, 1988. https://doi.org/10.1007/BF00133571.
166. R. Bajcsy, "Active perception," Proceedings of the IEEE, vol. 76, pp. 966–1005, 1988. https://doi.org/10.1109/5.5968
167. R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos, "Revisiting active perception," Autonomous Robots, vol. 42, pp. 177–196, 2018. https://doi.org/10.1007/s10514-017-9615-3.4
168. C. M. Keller and J. D. Keller, Cognition and Tool Use: The Blacksmith at Work. Cambridge, UK: Cambridge University Press, 1996.
169. U. Neisser, Cognition and Reality: Principles and Implications of Cognitive Psychology. San Francisco, CA, USA: W. H. Freeman, 1976.
170. D. M. Wolpert, Z. Ghahramani, and M. I. Jordan, "An internal model for sensorimotor integration," Science, vol. 269, pp. 1880–1882, 1995.
171. L. Malafouris, How Things Shape the Mind: A Theory of Material Engagement. Cambridge, MA, USA: MIT Press, 2013.
172. Jakob, W.; Speierer, S.; Roussel, N.; Nimier-David, M.; Vicini, D.; Zeltner, T.; Nicolet, B.; Crespo, M.; Leroy, V.; Zhang, Z. Mitsuba 3 Renderer, 2022. https://mitsuba-renderer.org
173. Felzenszwalb, Pedro F.; Huttenlocher, Daniel P. (September 2012). "Distance Transforms of Sampled Functions". Theory of Computing. 8 (19): 415–428. https://doi.org/10.4086/toc.2012.v008a019
174. William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high-resolution 3D surface construction algorithm. In Proceedings of the 14th annual conference on Computer graphics and interactive techniques (SIGGRAPH '87). Association for Computing Machinery, New York, NY, USA, 163–169. https://doi.org/10.1145/37401.37422
175. Hornbæk, K. Current Practice in Measuring Usability: Challenges to Usability Studies and Research. International Journal of Human-Computer Studies 2006, 64, 79–102. https://doi.org/10.1016/j.ijhcs.2005.06.002.
176. Shneiderman, B.; Plaisant, C.; Cohen, M.; Jacobs, S.; Elmqvist, N.; Diakopoulos, N. Designing the User Interface: Strategies for Effective Human-Computer Interaction, 6th ed.; Pearson, 2016.
177. Remy, C.; MacDonald Vermeulen, L.; Frich Pedersen, J.; Biskjaer, M.M.; Dalsgaard, P. Evaluating Creativity Support Tools in HCI Research. Proceedings of the 2020 ACM Designing Interactive Systems Conference (DIS '20). Association for Computing Machinery, 2020, pp. 457–476. https://doi.org/10.1145/3357236.3395474.
178. CIDOC CRM SIG (2024). ISO 21127:2023 has been released (announcement and mapping to CIDOC CRM community v7.1.3).
179. Doerr, M., Hiebel, G., Kritsotaki, A., Rousakis, Y., Schmidle, W., Theodoridou, M., Velios, A. (eds.) (2025). Definition of CRMsci: An Extension of CIDOC CRM to Support Scientific Observation, Version 3.0 (April 2025). https://cidoc-crm.org/crmsci/ModelVersion/crmsci-3.0
180. Hiebel, G., Doerr, M., Eide, Ø., Theodoridou, M. (2015). CRMgeo: A Spatiotemporal Model. An Extension of CIDOC CRM to link CIDOC CRM to GeoSPARQL through a Spatiotemporal Refinement, Version 1.2.
181. Hiebel, G. (2017). "CRMgeo: A spatiotemporal extension of CIDOC CRM." International Journal on Digital Libraries. https://doi.org/10.1007/s00799-016-0192-4
182. Bruseker, G., Doerr, M., Kritsotaki, A., Theodoridou, M. (2017). Parthenos Entities: Research Infrastructure Model (CRMpe), Version 3.1. https://cidoc-crm.org/Resources/parthenos-entities-research-infrastructure-model
183. Arp, R., Smith, B., Spear, A. (2015). Building Ontologies with Basic Formal Ontology. MIT Press. https://basic-formal-ontology.org/

184. Masolo, C., Borgo, S., Gangemi, A., Guarino, N., Oltramari, A. (2003). WonderWeb Deliverable D18: Ontology Library (final). IST Project 2001-33052 WonderWeb. https://www.loa.istc.cnr.it/old/Papers/D18.pdf
185. Cheong, H., Butscher, A. (2019). "Physics-based simulation ontology: an ontology to support modelling and reuse of data for physics-based simulation." Journal of Engineering Design. https://doi.org/10.1080/09544828.2019.1644301
186. Liu, X., Jiang, D., Tao, B., Xiang, F., Jiang, G., Sun, Y., Kong, J., Li, G. (2023)."A systematic review of digital twin about physical entities, virtual models, twin data, and applications" Advanced Engineering Informatics, 55, 101876. https://doi.org/10.1016/j.aei.2023.101876
187. Dragonfly Software. Glass Eye 2000 Enterprise Edition. https://www.dfly.com/
188. Rapid Resizer https://www.rapidresizer.com/
189. GIMP Software https://docs.gimp.org/
190. Inkscape Software https://www.inkscape.org
191. Adobe Illustrator Software https://www.adobe.com/gr_en/products/illustrator.html
192. Blender Foundation https://www.blender.org/
193. Trimble. SketchUp Viewer. https://sketchup.trimble.com/
194. Deepnest: Open source nesting software. https://deepnest.io/
195. LightBurn Software https://lightburnsoftware.com/