Real-Time Rendering of Molten Glass with Raymarching Technique on Point Clouds

Félix Chevalier
Manon Méhalin
Maxence Retier

Game Programming
Option GPU

September 2024 - June 2025

Tutor ISART DIGITAL:
Maël ADDOUM
m.addoum@isartdigital.com

Tutor enterprise:
Sylvain
CONTASSOT-VIVIER
sylvain.contassotvivier@loria.fr

Abstract

Glassblowing is a culturally significant and technically demanding craft whose preservation faces modern challenges, including environmental concerns and limited accessibility to training.

The LORIA aims to create some virtual simulations around glass blowing ; like a virtual simulation of glass blowing with VR for example. Rendering motlen glass in real time is ver

This project explores a real-time simulation of molten glass using raymarching techniques applied to implicit surfaces and 3D point clouds. A custom rendering pipeline was implemented in Vulkan, leveraging compute shaders for efficient parallel execution and enhanced control over memory access. Blending operations, Fresnel-based lighting, and reflection models were integrated to visually reproduce the semi-fluid, reflective nature of molten glass. A bespoke space partitioning algorithm was also developed to optimise point cloud traversal.
Preliminary results confirm the feasibility and visual realism of the approach, establishing a solid foundation for future integration of physical simulation and user interaction.
Beyond its technical contributions, this work illustrates how immersive technologies can support the transmission of traditional craftsmanship while promoting more sustainable practices.

Table of contents

1 Introduction

Glassblowing is an art form with deep cultural and historical significance, particularly in France, where it embodies centuries of craftsmanship and tradition. However, preserving this heritage presents modern challenges, including the environmental impact of energy-intensive training methods and the need to adapt artisanal practices to digital technologies.

This work is conducted within a broader research initiative led by the LORIA (Laboratoire Lorrain de Recherche Informatique et ses Applications), which aims to develop an immersive augmented reality experience for virtual glassblowing. The present contribution focuses specifically on the real-time visual rendering of molten glass. No physical modeling of the material is performed ; the project is limited to graphical representation, with the objective of producing a visually convincing depiction of molten glass in an interactive context.

Real-time visualization of molten glass remains a complex task due to its semi-fluid behavior and optical properties. Achieving a balance between realism and performance is essential to enable integration into interactive environments.

To this end, a simulation framework based on raymarching techniques applied to 3D point clouds has been developed. Raymarching—an algorithm that advances rays incrementally to detect surfaces—offers a flexible approach to visualizing deformable materials. In this context, it enables the dynamic visualization of molten glass surfaces without relying on explicit geometry.

This method departs from traditional mesh-based rendering. Instead, implicit surfaces and raymarching are employed for their suitability with point cloud data, allowing dynamic deformation to be integrated with advanced visual effects. Although molten glass exhibits both fluid and solid characteristics depending on temperature and motion, these physical behaviors are not simulated. Rather, the rendering method approximates the visual result in real time.

Beyond its technical scope, the project also addresses sustainability. By offering a virtual training tool, reliance on physical furnaces may be reduced, thereby lowering the environmental footprint associated with traditional training. This dual objective—technological innovation and ecological responsibility—forms the foundation of the work presented in the following sections.

2 The methodology

To address the challenge of representing molten glass in real time, a structured and experimental methodology was adopted, drawing from graphics programming and real-time rendering techniques. This project does not aim to reproduce the physical behavior of molten glass but focuses exclusively on its visual representation. The objective is to develop a convincing and responsive graphical system suitable for immersive environments. This required a balance between artistic fidelity, technical feasibility, and performance constraints.

2.1 Technical Scope and Approach

The visual rendering of molten glass presents several unique challenges : it must appear fluid, semi-transparent, and dynamically deformable. Traditional mesh-based rendering methods are generally inadequate for representing such materials, particularly when geometry evolves continuously. To address these limitations, a **raymarching-based rendering technique** was selected, enabling the direct rendering of **implicit surfaces** without predefined geometry.

Raymarching is a rendering approach in which rays are cast through a scene, and the algorithm incrementally advances along each ray until a surface is detected (see Fig.1). This technique is particularly effective for rendering mathematically defined surfaces such as metaballs, soft shape blends, or deformable volumes—features that are well suited for conveying the visual characteristics of molten glass.

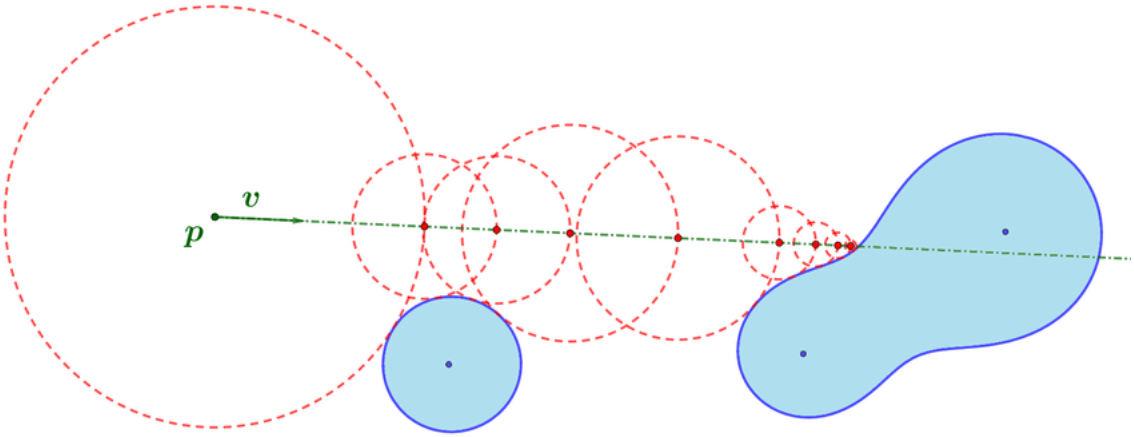

FIGURE 1 – Raymarching system

2.2 Project Breakdown

The technical development was structured into several incremental phases :

1. **Initial Raymarching Setup**
   A minimal raymarching loop was first implemented to render a basic torus shape using Signed Distance Functions (SDFs).
   This step enabled validation of the rendering pipeline and early experimentation with implicit surface definitions (see Fig.2).

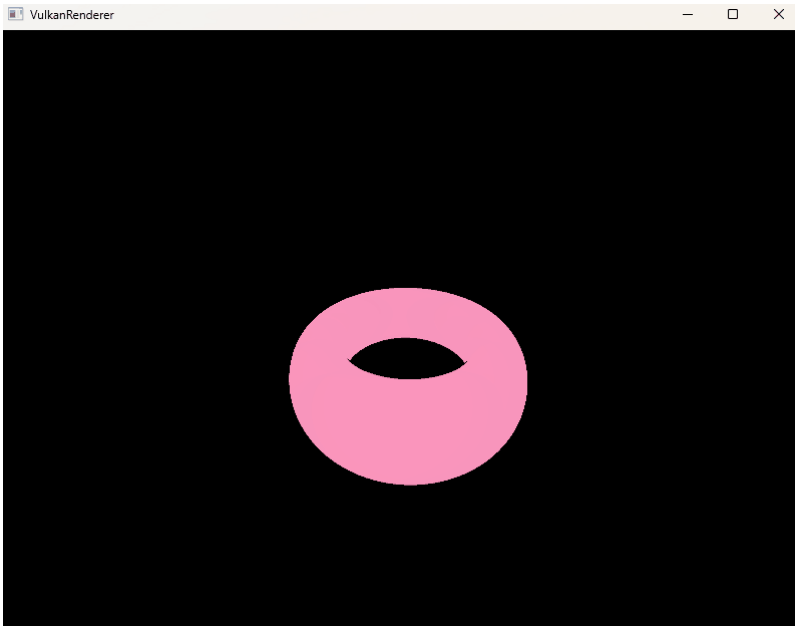

FIGURE 2 – Raymarching torus

2. **Implementation of Shape Blending**
   To approximate the soft, flowing appearance of molten glass, blending operations between multiple SDFs were introduced.
   By combining a torus and a sphere with smooth transitions, organic shapes were generated to reflect visual deformations observed in real glassblowing (see Fig.3).

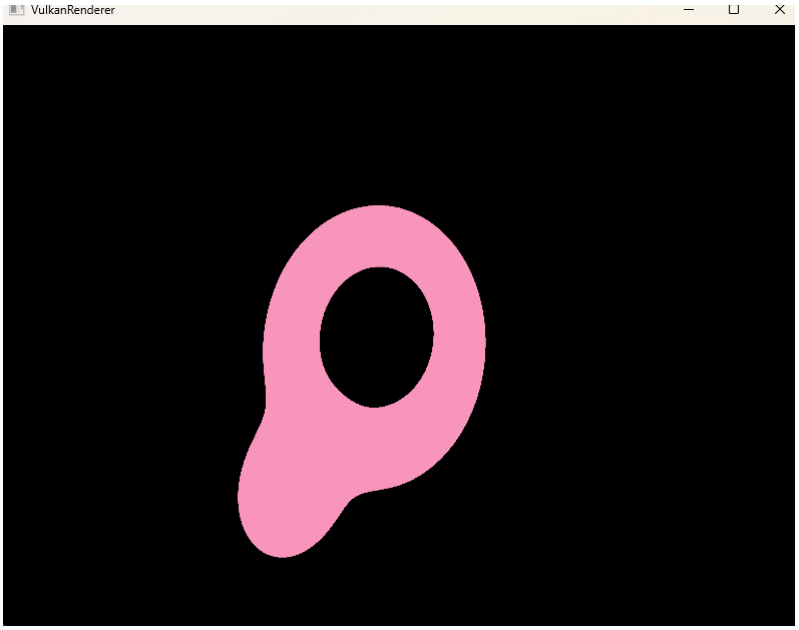

FIGURE 3 – Raymarching blending

3. **Lighting and Reflection Enhancements**
   Realistic shading is essential for depicting transparent and reflective materials such as glass. Phong-based lighting, Fresnel effects, and surface normals derived from SDF gradients were incorporated to enhance visual realism (see Fig.4).

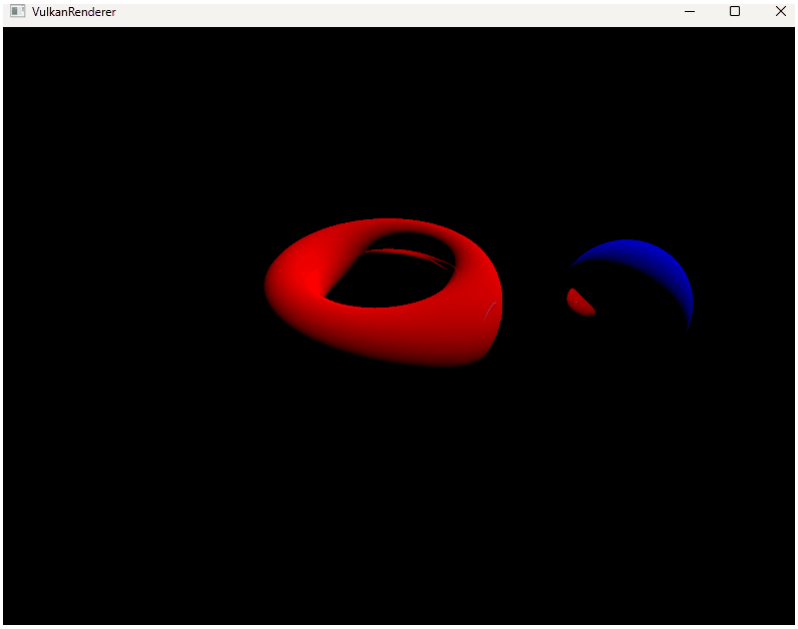

FIGURE 4 – Raymarching Phong light

4. **Transition to Compute Shaders**
   The initial implementation relied on fragment shaders, but performance and flexibility limitations led to a migration to compute shaders. This transition provided enhanced control over parallel execution and memory access, enabling better performance and supporting future extensions involving point cloud data.

5. **Space Partitioning algorithm**
   Distance evaluations for multiple spheres were identified as a computational bottleneck in the raymarching process. To improve performance when processing large point clouds, a custom space partitioning algorithm was developed, prioritizing balanced subdivision and traversal efficiency over conventional methods such as k-d trees [1].

6. **Custom Space Partitioning algorithm**
   The algorithm begins by computing an Axis-Aligned Bounding Box (AABB) around the point cloud, followed by recursive axis-aligned subdivisions along X, Y, and Z axes based on median point positions. This ensures an even distribution of points across subregions and minimizes the depth of the resulting binary tree.

7. **Binary tree**
   Each AABB and its subdivision is represented as a node in a binary tree. The root corresponds to the bounding volume of the entire point cloud, while child nodes contain recursively defined subspaces (see Fig.5).

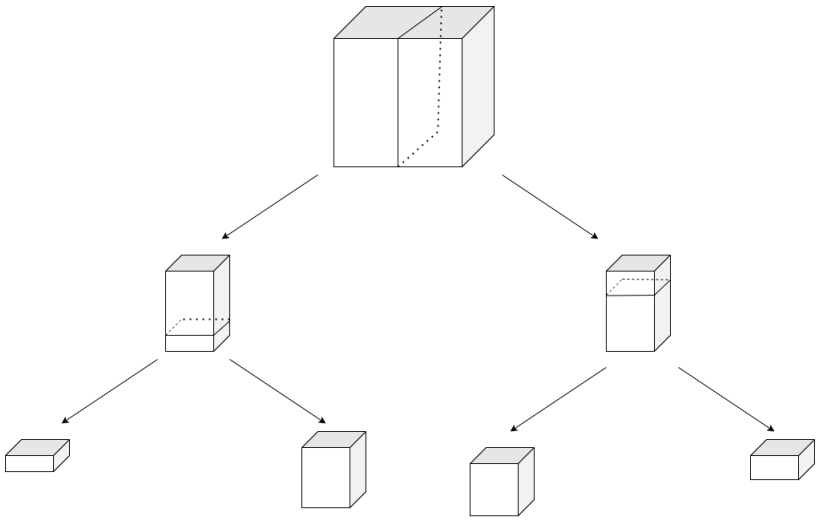

FIGURE 5 – Binary tree

8. **Unique index, morton number**
   Unique indices are assigned to each node using a Morton code. The root node is indexed as 1, and each child inherits an index extended by 0 or 1 depending on its position relative to the cut. This facilitates hierarchical traversal and memory alignment.

9. **Space Partitioning implementation in raymarching compute shader**
   TODO

2.3 Development Environment

The system has been implemented using Vulkan, a low-level graphics API offering fine-grained control over GPU resources and pipeline stages. Vulkan was preferred over OpenGL for its performance, multithreading support, and native compatibility with compute shaders.

The rendering logic is written in C++, while GPU operations—such as raymarching, lighting, and memory management—are implemented in GLSL, with dedicated modules for fragment and compute shaders. Version control was managed through Git, and the codebase was structured into modular components for maintainability and scalability.

2.4 Key Technical Choices

Several key decisions shaped the development approach :

— Adoption of raymarching : Enabled the rendering of smooth, deformable shapes without relying on mesh geometry.

— Use of implicit surfaces and blending techniques : Provided a way to visually represent molten glass in a fluid-like state through continuous transitions between forms.

— Migration to compute shaders : Offered improved performance and flexibility by decoupling rendering from the rasterization pipeline.

— Preparation for volumetric and point-based rendering : Established a foundation for future integration of 3D data structures, while remaining within the scope of real-time visualization, not physical simulation.

2.5 Performance Evaluation Method

To assess the performance of the rendering system, a dedicated profiling methodology was established using the Tracy profiler. This tool enabled detailed monitoring of execution timings, memory usage, and GPU/CPU workload distribution, providing valuable insights into the internal behavior of the application.

1. **Comparison between fragment shader and compute shader pipelines**
   Performance was measured for both a traditional rasterization-based pipeline using fragment shaders and a compute-shader-based rendering pipeline. This comparison aimed to evaluate differences in control, parallelization efficiency, and resource usage.

2. **Impact of the binary tree structure**
   Tests were conducted with and without the custom space partitioning algorithm based on a binary tree, to determine its influence on distance computations and traversal efficiency within point cloud datasets.

3. **Metric collection**
   The profiling focused on collecting quantitative data such as frames per second (FPS), GPU utilization, and CPU utilization. To ensure consistency across tests, the same scene and parameters were used in all configurations, isolating the impact of the rendering pipeline itself.

4. **Multi-system testing**
   In order to ensure generalizability and better understand performance variations across hardware, tests were conducted on multiple machines with different specifications. This approach provided a broader basis for comparison and highlighted the scalability of the system.

It should be noted that the use of ImGui for interface and debugging purposes introduces a dependency on the traditional graphics pipeline. As a consequence, the rendering system does not operate in a fully compute-only configuration. This factor was taken into account during performance measurements and may influence the interpretation of the results.

3 Results and discussion

3.1 Achieved results

At the current stage of the project, a fully functional raymarching rendering loop has been implemented, enabling real-time visualization of implicit surfaces. Basic geometric shapes, such as spheres and tori, are rendered using Signed Distance Functions (SDFs), and blending techniques have been developed to produce smooth transitions between them—effectively mimicking the visual behavior of molten glass (see Figs.2–3).

In addition to shape blending, dynamic lighting and reflection models have been integrated. Surface normals are computed from the gradient of the SDFs, and lighting effects—such as specular highlights and environmental reflections—are applied to enhance the appearance of a transparent, reflective material. These enhancements contribute to a visually convincing representation of molten glass (see Fig.4).

A significant milestone was the transition from a traditional fragment shader pipeline to compute shaders. This change provided improved control over execution, memory management, and thread distribution, resulting in better performance and greater scalability for future extensions (see Table 1).

Shader	**Nb spheres**	**FPS**	**GPU/CPU usage (%)**	**Notes**
Fragment Shader	400	30	GPU : 20 / CPU : 5	Stable
Fragment Shader	2500	2	GPU : 100 / CPU : 10	Saturate render
Compute Shader	2500	30	GPU : 50 / CPU : 2	Gain fluidity
Compute Shader	5000	10	GPU : 80 / CPU : 5	Reach limit

TABLE 1 – Comparison between fragment shader and compute shader

Furthermore, a complete binary tree has been constructed from the point cloud using a custom space partitioning algorithm. Each node in the tree is assigned a unique index to allow efficient traversal between spatial subdivisions (see Fig.5).

3.2 Performance and Realism Evaluation

The transition to compute shaders has led to improvements in frame rendering times and a reduction in GPU workload by optimizing memory access patterns and the execution of raymarching steps. Unlike fragment shaders, compute shaders decouple rendering from the rasterization pipeline, enabling more direct and efficient computation of per-pixel color values.

Although no formal benchmarking has yet been performed, a clear increase in visual fluidity and responsiveness has been observed, particularly when rendering large numbers of blended shapes.

From a visual realism standpoint, the blending technique and lighting model have proven effective in conveying the semi-solid, semi-fluid appearance of molten glass. It should be emphasized that this effect is purely graphical ; no physical simulation of material behavior is included. The material responds convincingly to lighting, and the reflections contribute strong visual cues about shape and curvature.

3.3 Limitations and Ongoing Work

Despite the promising results obtained so far, several essential features remain under development :

— Glass deformation is currently represented only at the visual level, without any physical modeling or real-time physical interaction.

— The integration of point clouds as a geometric foundation for molten glass remains in progress and has not yet been fully connected to the raymarching framework.

— Live user interaction, such as real-time shaping of the glass, is not yet supported and remains a long-term objective.

Additionally, the accurate rendering of transparency and internal refraction—both critical for realistic glass appearance—continues to present significant challenges. These aspects are intended to be explored in future stages of the project.

3.4 General Discussion

The results obtained thus far demonstrate that the real-time visual representation of molten glass using raymarching constitutes a feasible and effective approach. The combination of implicit surface rendering and compute shader execution offers a flexible framework capable of reproducing complex visual phenomena. Importantly, the implemented system is not intended to simulate the physical behavior of glass, but rather to approximate its appearance in a real-time, interactive context.

A solid foundation has been established for further development, including user interactivity, volumetric rendering, and integration into augmented and virtual reality environments. More broadly, this work illustrates how digital tools can support the preservation of traditional craftsmanship, offering new possibilities for education, training, and cultural dissemination—while contributing to more sustainable practices.

4 Conclusions, future prospects

This work was conducted within the framework of a broader research initiative led by the LORIA, aiming to create an immersive augmented reality experience for glassblowing. The contribution presented here focused exclusively on the real-time visual rendering of molten glass, with the objective of reproducing its appearance in a convincing and responsive manner. No physical simulation was implemented ; the scope remained limited to the graphical representation of the material.

To meet this objective, a raymarching-based rendering pipeline was developed, using implicit surfaces and compute shaders to visualize deformable shapes in real time. Lighting and blending techniques were introduced to emulate the semi-fluid, semi-transparent qualities of molten glass. In addition, a custom space partitioning algorithm was designed to optimize point cloud traversal, and a binary tree structure was successfully generated from the dataset. These developments demonstrate the feasibility of achieving realistic visual representations of molten glass without relying on mesh geometry.

The system offers a solid foundation for further research. Future work may include the integration of user interaction, allowing real-time shaping of the glass ; improvements in optical realism, such as internal refraction and transparency handling ; and the implementation of visual effects like thermal gradients to enrich the perception of material behavior. On a technical level, building the binary tree directly within the compute shader is a promising step toward full GPU-side computation. Ultimately, the vision is to transform the viewer into a true real-time simulation, where deformations applied to point clouds are computed interactively, with the spatial structure being updated dynamically. In the longer term, the incorporation of physically-based simulation models could extend the system beyond visual rendering toward behavioral realism.

5 Bibliographical references

[1] Bruce Naylor. Constructing good partitioning trees. *Graphics Interface*, September 2001.

[2] Floney Yang. Raymarching collision, November 2024.

[3] Hecomi. Raymarching collision project (github), November 2024.

[4] Sakri Koskimies. Raymarching engine (github), November 2024.

[5] Adrian Biagioli. Raymarching presentation, November 2024.

[6] Nabil Mansour. SDF raymarching, November 2024.

[7] Wojciech Grega, Adam Pilat, and Andrzej Tutaj. Modelling of the glass melting process for real-time implementation. *International Journal of Modeling and Optimization*, 5(6) :366–373, 2015. Number : 6.