Contents

Project Specification Report

3D Ray Tracing Engine with Professional Desktop Application

Course: Project CSI in Visual Computing

Program: Masters in Computer Science International (CSI)

Institution: Universität Rostock

Team Members:

Abu Bakar

Muhammad Shahman Butt

December 2024

1. Project Overview

This project develops a 3D ray tracing engine from scratch that generates photorealistic images of complex 3D scenes. The engine accepts XML scene descriptions, OBJ mesh models, and MTL material definitions, producing high-quality rendered images through physically-based rendering. Ray tracing simulates light behavior by tracing rays from camera through pixels, calculating interactions with surfaces, materials, and light sources, naturally handling reflections, refractions, soft shadows, and global illumination.

Objectives: Develop CPU-based ray tracing engine with core algorithms; support XML/OBJ/MTL formats; implement material models (Lambertian, Metal, Emissive, Dielectric); achieve photorealistic output via optimization (BVH, multi-threading); build professional desktop application with modern UI; provide real-time preview and scene editing capabilities.

1.1 High-Level Architecture Diagram

```
+------------------------------------+
|   Desktop Application UI Layer     |
|  +----------+  +----------+        |
|  | MainWindow|  | Viewport  |       |
|  | (Menus,  |  | (Preview) |        |
|  | Toolbars)|  +----------+        |
|  +----------+  +----------+        |
|  | Scene     |  | Properties|       |
|  | Hierarchy |  |  Panel    |       |
|  +----------+  +----------+        |
|  +----------+                      |
|  | Material  |                      |
|  |  Editor   |                      |
|  +----------+                      |
+------------------+------------------+
                   |
                   v
+------------------------------------+
|      Input Layer                   |
|  +----------+  +----------+        |
|  | XML Parser|  | OBJ Loader|       |
|  +----------+  +----------+        |
|  +----------+                      |
|  | MTL Parser|                      |
|  +----------+                      |
+------------------+------------------+
                   |
                   v
+------------------------------------+
|   Scene Representation             |
|  +----------+  +----------+        |
|  |  Camera   |  |  Objects  |       |
|  | (FOV,Pos) |  |(Spheres,  |       |
|  +----------+  |Triangles, |       |
|                |  Meshes)  |       |
|  +-----------+  +-----------+       |
|  | Materials |                       |
|  |(Lambertian|                       |
|  | Metal,etc)|                       |
|  +-----------+                      |
+------------------+------------------+
                   |
                   v
+-----------------------------------+
|   Ray Tracing Core Engine         |
|  +-----------+  +-----------+     |
|  | Ray Cast  |  |Intersection|      |
|  | Generator |  |  Testing   |      |
|  +-----------+  +-----------+     |
|  +-----------+  +-----------+     |
|  | Material  |  | Recursive |     |
|  | Scattering|  |  Tracing  |     |
|  +-----------+  +-----------+     |
|  +-----------+                    |
|  |  Shadow   |                      |
|  |   Rays    |                      |
|  +-----------+                    |
+------------------+------------------+
                   |
                   v
+-----------------------------------+
|   Optimization Layer              |
|  +-----------+  +-----------+     |
|  |  BVH/KD- |  | Multi-    |     |
|  |   Tree   |  | Threading |     |
|  +-----------+  +-----------+     |
+------------------+------------------+
                   |
                   v
+-----------------------------------+
|      Output Layer                 |
|  +-----------+  +-----------+     |
|  |  Tone    |  |  Gamma    |     |
|  | Mapping  |  | Correction|     |
|  +-----------+  +-----------+     |
|  +-----------+                    |
|  |  Image   |                     |
|  |  Writer  |                     |
|  |(PNG/PPM) |                     |
|  +-----------+                    |
+-----------------------------------+
```

2. Core Algorithms and Techniques

Ray-Object Intersection: Sphere (analytic quadratic, O(1)), Triangle (Möller-Trumbore, O(1)), Mesh (BVH-accelerated, O(log n))

Material Models: Lambertian (diffuse, cosine-weighted hemisphere), Metal (perfect/fuzzy reflection), Emissive (light-emitting), Dielectric (glass with Fresnel)

Optimization: BVH (binary tree AABB, 10-100x speedup), Multi-threading (tile-based, linear scaling)

UI Components: Qt6-based desktop application with dockable panels, real-time viewport with progressive rendering, scene hierarchy tree, properties editor, material editor with live preview, render settings panel

3. Feature Development Layers

Layer 1: Functional Minimum (50-60 hrs) - Ray-sphere intersection, Lambertian material, directional light, hard shadows, basic anti-aliasing (1-4 samples), PPM output, XML scene loading, basic Qt window with viewport display. *Success: Renders 2-3 spheres with basic lighting, displays in UI viewport.*

Layer 2: Minimum Goal (80-100 hrs) - All Layer 1 + triangle intersection, OBJ mesh loading, multiple materials, MTL support, recursive reflections (3-10 depth), improved anti-aliasing (10-20 samples), PNG output, multi-threaded rendering, scene hierarchy panel, properties editor, basic material editor. *Success: Complex scenes with meshes, reflections, < 5 min for 800x600, full UI functionality.*

Layer 3: Desired Goal (100-120 hrs) - All Layer 2 + BVH acceleration, soft shadows, dielectric material, texture mapping, Monte Carlo path tracing, tone mapping/HDR, performance profiling, real-time preview mode, advanced material editor, render settings panel, export functionality. *Success: Photorealistic images, 1000+ triangles, < 2 min for 1920x1080, professional UI.*

Layer 4: Maximum Goal (150-200 hrs) - KD-Tree, advanced materials, environment maps, depth of field, motion blur, volumetric rendering, denoising, SIMD optimizations, interactive camera controls, transform gizmos, undo/redo system, project templates.

Layer 5: Extras (250+ hrs) - GPU acceleration (CUDA/OpenCL), real-time capabilities, advanced global illumination, bidirectional path tracing, plugin system, scripting support.

4. Design Decisions

Object-Oriented Design: Polymorphic base classes (`hittable`, `material`) enable extensibility. **XML Scene Description:** Human-readable, hierarchical, industry-standard. **Tile-Based Parallel Rendering:** Maximizes cache locality, scales with CPU cores. **Recursive Ray Tracing:** Natural reflections/refractions with configurable depth. **Material System:** Physically-based rendering model. **Qt6 Framework:** Professional cross-platform UI, industry-standard, excellent OpenGL integration for viewport. **Progressive Rendering:** Real-time preview with low samples, full render on demand. **Dockable UI:** Flexible workspace similar to professional 3D software.

5. Tools and Technologies

Language: C++20 (GCC 10+/Clang 12+) | **Build:** CMake 3.16+ | **GUI Framework:** Qt6 (Widgets, OpenGL) | **Libraries:** GLM, TinyXML2, OBJ_Loader, stb_image_write | **Tools:** Git/GitHub, CI/CD, GDB/LLDB

6. Task List and Schedule

6.1 Task Breakdown (Layers 1-3)

Phase	Tasks	Who	Est. Hrs
Foundation	Setup, architecture design, core math library (vec3, ray), camera system	Abu Bakar	28
Foundation	Ray-sphere intersection, Lambertian material, basic image I/O (PPM)	Muhammad Shahman Butt	16
UI Foundation	Qt6 project setup, MainWindow, basic viewport widget, menu system	Abu Bakar	20
Core Features	XML scene parser, OBJ mesh loader, MTL material parser, anti-aliasing	Abu Bakar	42
Core Features	Ray-triangle intersection, Metal material, Emissive material, PNG output	Muhammad Shahman Butt	24
UI Core	Scene hierarchy panel, properties panel, render integration	Abu Bakar	24
UI Core	Material editor UI, image export functionality	Muhammad Shahman Butt	16
Integration	Basic ray tracing loop, recursive tracing, shadow rays	Both	18
Optimization	BVH acceleration structure (design & implementation), performance profiling	Abu Bakar	32
Optimization	Soft shadows, texture mapping support	Muhammad Shahman Butt	20
Advanced	Monte Carlo path tracing, dielectric/glass material, tone mapping/HDR	Abu Bakar	28
UI Advanced	Real-time preview mode, progressive rendering, render settings panel	Abu Bakar	20

Phase	Tasks	Who	Est. Hrs
Multi-threading	Tile-based parallel rendering, progress reporting, UI thread safety	Both	20
Documentation	Testing, validation, report, presentation	Both	40

Total Estimated Hours (Layers 1-3): ~328 hours

Task Distribution Summary: - Abu Bakar: Architecture, core math, camera, XML/OBJ/MTL parsers, BVH acceleration, path tracing, dielectric material, tone mapping, performance profiling, Qt6 UI framework, MainWindow, viewport, scene hierarchy, properties panel, real-time preview, render settings, basic materials (Lambertian, Metal, Emissive) - **Muhammad Shahman Butt:** Ray intersections, shadows, texture mapping, image I/O, material editor UI, export functionality - **Both:** Integration, recursive tracing, multi-threading, UI thread safety, documentation

6.2 Milestones and Schedule

Milestone	Week	Deliverables	Success Criteria
M1: Basic Ray Tracer + UI	5	Ray-sphere intersection, simple scene rendering, basic Qt window with viewport	Renders 2-3 spheres, displays in UI
M2: Complete Core Engine + UI	9	Full material system, mesh loading, recursive reflections, multi-threading, scene hierarchy, properties panel	Renders complex scenes, full UI functionality
M3: Optimized Renderer + Advanced UI	14	BVH acceleration, advanced materials, path tracing, real-time preview, material editor	Fast rendering, photorealistic quality, professional UI
M4: Final Submission	18	Complete project, final report, presentation	All deliverables submitted

Schedule: Weeks 1-5 (Foundation + Basic UI), 6-9 (Core Features + UI Integration), 10-14 (Optimization + Advanced UI), 15-17 (Polish & Testing), 18 (Finalization)

7. UI Features and Components

MainWindow: Professional application window with menu bar (File, Edit, Render, View, Help), toolbars for quick actions, dockable panels, status bar with render progress.

ViewportWidget: Real-time preview of scene using OpenGL, progressive rendering display, interactive camera controls (mouse drag to rotate, scroll to zoom), render progress visualization.

SceneHierarchy: Tree view of all scene objects (spheres, meshes, lights), drag-and-drop reordering, object selection, context menu for add/delete/duplicate operations.

PropertiesPanel: Edit selected object properties (position, rotation, scale), material assignment, transform gizmos, real-time property updates.

MaterialEditor: Visual material creation with color pickers, sliders for roughness/metallic properties, live preview, material library (save/load materials), texture assignment.

RenderSettings: Resolution presets (720p, 1080p, 4K), samples per pixel slider, max depth control, thread count selection, output format selection, render queue management.

8. Success Metrics

Functional: Render scenes with 1000+ triangles; support all material types; produce photorealistic images; handle complex lighting (shadows, reflections, refractions); professional UI with all core features.

Performance: Render 1920x1080 scene in < 2 minutes (Layer 3); 10-50x speedup with BVH; linear scaling with thread count; real-time preview at 1-4 samples per pixel.

UI Quality: Intuitive interface comparable to commercial 3D software; responsive interactions; smooth preview updates; professional appearance.

9. Conclusion

This specification outlines development of a comprehensive 3D ray tracing engine with a professional commercial-grade desktop application. The system combines advanced computer graphics algorithms with modern UI design, providing an intuitive interface for scene creation, material editing, and photorealistic rendering. The layered approach ensures functional system at each stage while allowing ambitious enhancements. Focus on fundamental algorithms, optimization techniques, modern C++ practices, and professional UI development will deliver a production-quality application capable of generating photorealistic images through an accessible graphical interface.

10. References and Bibliography

Books

Shirley, P., & Morley, R. K. (2003). *Realistic Ray Tracing* (2nd ed.). A K Peters/CRC Press.

Pharr, M., Jakob, W., & Humphreys, G. (2016). *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann.

Glassner, A. S. (1989). *An Introduction to Ray Tracing*. Academic Press.

Online Resources

Shirley, P. "Ray Tracing in One Weekend" series. Retrieved from https://raytracing.github.io/

Scratchapixel.com - Computer Graphics Programming. Retrieved from https://www.scratchapixel.com/

NVIDIA OptiX Documentation. Retrieved from https://developer.nvidia.com/optix

Qt6 Documentation. Retrieved from https://doc.qt.io/qt-6/

Software and Libraries

GLM - OpenGL Mathematics Library. Retrieved from https://github.com/g-truc/glm

TinyXML2 - XML Parser Library. Retrieved from https://github.com/leethomason/tinyxml2

stb_image_write - Image Writing Library. Retrieved from https://github.com/nothings/stb

Qt6 - Cross-platform Application Framework. Retrieved from https://www.qt.io/