Computer Graphics Project Proposal

The Crypt of Thoth: Interactive 3D Tomb Exploration

Proposed By:

Forhad Islam Rony
Roll: 2007065

Iqbal Mahamud Moon
Roll: 2007093

Course: Computer Graphics
Department: Computer Science & Engineering

Academic Year: 2024-2025

February 15, 2026

Contents

1 Introduction

1.1 Project Overview

This project aims to design and develop an immersive Ancient Egyptian Tomb Exploration simulation using OpenGL. The virtual environment represents a "T-shaped" burial chamber containing corridors, transverse halls, pillars, and an interactive sarcophagus. This architectural design is historically accurate and represents the typical layout found in Egyptian royal tombs from the New Kingdom period (circa 1550-1077 BCE). The project demonstrates core computer graphics concepts such as hierarchical modeling, matrix stacks, point light attenuation, and normal mapping. All objects in the scene are constructed directly in `C++` code using basic geometric primitives, showcasing the power of procedural geometry generation without relying on external 3D modeling software.

1.2 Motivation and Objectives

The primary motivation for this project is to create an engaging interactive experience while demonstrating fundamental computer graphics principles. The Egyptian tomb setting provides a rich visual context that allows for the implementation of complex lighting scenarios, texture mapping on ancient stone surfaces, and atmospheric rendering techniques. **Key objectives include:**

- Implementing a complete real-time 3D rendering pipeline using modern OpenGL
- Demonstrating hierarchical transformations and scene graph management
- Creating realistic lighting with multiple point light sources and attenuation
- Developing interactive elements that respond to user input
- Applying texture mapping and normal mapping for enhanced visual fidelity
- Building a first-person camera system with smooth navigation controls

1.3 Scope and Limitations

The project focuses on rendering and interaction mechanics rather than game logic or narrative elements. Physics simulation is limited to basic collision detection for navigation boundaries. Advanced rendering techniques such as shadow mapping, ambient occlusion, and global illumination are beyond the current scope but may be implemented as future enhancements. The environment is designed as a single continuous space rather than multiple loaded levels, which simplifies memory management and state transitions. All assets are generated procedurally or loaded from standard image formats, ensuring portability across different development environments.

2 Tools and Libraries

2.1 Core Technologies

The following tools and libraries form the foundation of the project's technical implementation:

Library	Version	Purpose
OpenGL	3.3 (Core)	Core graphics API for rendering 3D objects
GLFW	3.3+	Window creation and input handling
GLAD	–	Loading OpenGL function pointers
GLM	0.9.9+	Mathematical operations and transformation matrices
stb_image	2.27	Image loading for textures

Table 1: Primary libraries and their roles

2.2 Development Environment

The project is developed using `C++17` standards with cross-platform compatibility in mind. The build system utilizes CMake for managing compilation across Windows, macOS, and Linux environments. Version control is maintained through Git, enabling collaborative development and iterative refinement. The shader code is written in GLSL (OpenGL Shading Language) version 3.30, matching the core OpenGL profile. All texture assets are stored in PNG format to ensure lossless quality and transparency support where needed.

3 Project Visuals

Figures 1 through 4 showcase the visual design and lighting of the proposed 3D environment. The exterior pyramid structure serves as the entry point, while the isometric view demonstrates the spatial relationships between different chambers and corridors. The lighting showcases highlight the atmospheric qualities achieved through careful light placement and attenuation parameters. The tomb's architecture follows authentic Egyptian design principles, with a descending entrance corridor leading to a pillared hall, which then opens into the main burial chamber. Hieroglyphic wall decorations are rendered using normal-mapped textures to simulate carved relief without the geometric complexity of actual 3D carving.

Figure 1: External view of the pyramid structure serving as the entry point for the 3D exploration environment

4 Dynamic Interaction Objects

4.1 Game Object & Mechanics Overview

The interactive elements within the tomb create an engaging exploration experience that goes beyond passive observation. Each dynamic object serves both aesthetic and gameplay purposes, contributing to the overall atmosphere while demonstrating advanced graphics programming techniques.

1. Swinging Blade Trap **Role:** Hazards requiring timed movement to bypass, adding challenge to corridor navigation. **Implementation:** The blade trap utilizes time-based sinusoidal rotation using a matrix stack to handle pivot points accurately. The transformation hierarchy consists of:

1. Ceiling mount (parent transformation)
2. Pendulum arm (child with animated rotation)
3. Blade geometry (grandchild inheriting all transformations)

The rotation angle follows the equation: $\theta(t) = \theta_{max} \sin(2\pi f t)$, where θ_{max} is the maximum swing angle (typically 45 degrees) and f is the frequency of oscillation. This creates a smooth, physically plausible motion pattern that players can learn to anticipate.

2. First-Person Camera System **Role:** Interactive exploration of the tomb environment with intuitive controls. **Controls:** WASD keys for movement (forward, left,

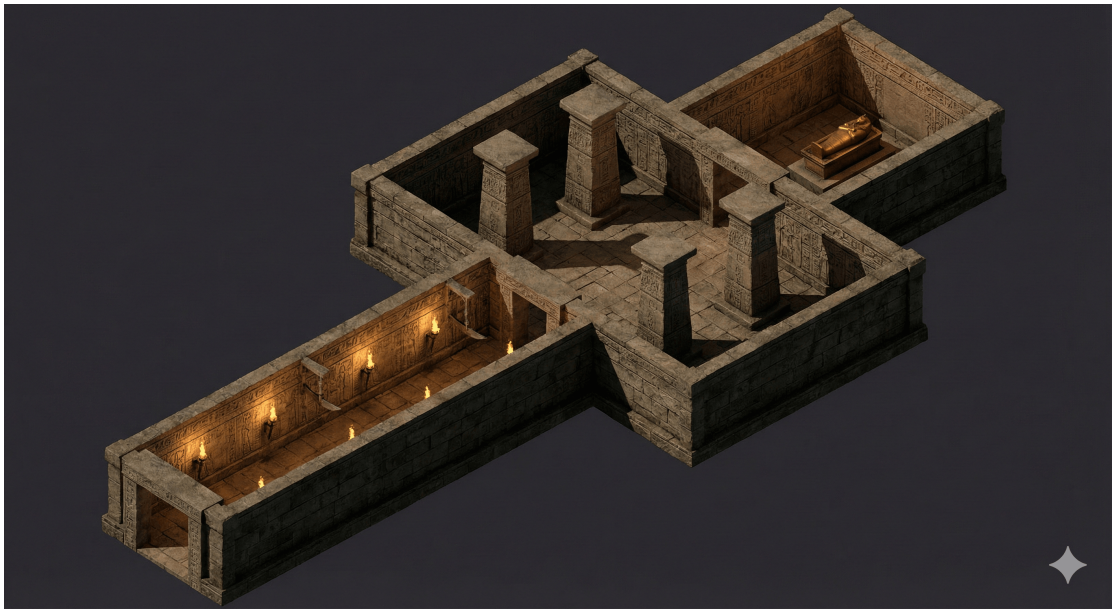

Figure 2: The Crypt Layout (Isometric View) – showing the T-shaped architectural design with main corridor, transverse hall, and burial chamber

backward, right) and mouse input for yaw/pitch rotation. The camera implements momentum-based movement for smooth acceleration and deceleration, preventing jarring stops and starts. **Technical Details:** The camera maintains three perpendicular vectors (forward, right, up) that are recalculated each frame based on mouse input. Movement speed is modulated by frame time to ensure consistent velocity regardless of frame rate variations. Pitch rotation is clamped to prevent gimbal lock, restricting vertical look angle to approximately ± 89 degrees.

Figure 3: The Trapped Corridor – demonstrating point light attenuation and hierarchical modeling of the blade trap mechanism

4.2 Interactive Features

3. Interactive Sarcophagus **Role:** The final objective within the burial chamber, serving as the focal point of the exploration experience. **Interaction:** Key-press (E key) triggers lid translation along its local Y-axis when the player is within interaction range (defined as 2.5 units from the sarcophagus center). The lid slides smoothly using linear interpolation over 2 seconds, revealing the interior. A proximity detection system uses distance calculation: $d = \sqrt{(x_p - x_s)^2 + (y_p - y_s)^2 + (z_p - z_s)^2}$ where (x_p, y_p, z_p) is the player position and (x_s, y_s, z_s) is the sarcophagus position. **Visual Feedback:** When within interaction range, the sarcophagus glows subtly (achieved through increased emissive color in the fragment shader), providing clear feedback to the player that interaction is possible. This prevents frustration from unclear interaction boundaries.

4. Torchlight System **Features:** Multiple point lights strategically positioned along corridors and chambers, each with quadratic attenuation to simulate realistic light falloff. **Attenuation Formula:** Light intensity follows the equation:

$$I = \frac{I_0}{K_c + K_l d + K_q d^2}$$

where I_0 is the base intensity, d is distance from light source, and K_c, K_l, K_q are constant, linear, and quadratic attenuation coefficients respectively. Typical values used: $K_c = 1.0$, $K_l = 0.09$, $K_q = 0.032$ for medium-range torchlight. **Flickering Effect:** Each torch implements a Perlin noise-based flicker using time-varying intensity modulation in the fragment shader. This creates natural-looking flame movement without the computational cost of particle systems: $I_{final} = I_{base} \cdot (1.0 + 0.1 \cdot \text{noise}(t))$.

Figure 4: The Burial Chamber – showcasing high-detail texturing with normal mapping on hieroglyphic walls and the central sarcophagus with metallic material properties

5 Objects with Complex Shapes

5.1 Hierarchical Modeling Approach

All complex objects in the tomb are constructed using hierarchical modeling principles. This approach treats complicated structures as assemblies of simpler components organized in parent-child relationships. Basic primitives such as cubes, cylinders, and pyramids are reused with different transformations, demonstrating efficient geometry management. The hierarchical structure offers several advantages:

- **Transform Propagation:** Child objects automatically inherit parent transformations
- **Code Reusability:** Base primitives are defined once and instantiated multiple times
- **Logical Organization:** Complex objects are conceptually divided into meaningful components
- **Simplified Animation:** Animating a parent automatically affects all children appropriately

The matrix stack implementation uses GLM's transformation functions combined with explicit push/pop operations to maintain transformation context. Each object's local transformation is multiplied with its parent's accumulated transformation to produce the final world-space position and orientation.

5.2 Implementation Details

1. Corridor Pillars **Why it is complex:** Requires instanced rendering of repetitive tapered structures with decorative capitals and bases, positioned at regular intervals along the corridor. **Implementation:** Each pillar consists of three main components:

1. **Base:** A wider cuboid providing stability (1.5 unit width)
2. **Shaft:** Multiple stacked cuboids with decreasing scale factors to create tapering effect. Each section is 0.98 times the width of the section below it, creating a smooth taper from 1.2 units at bottom to 0.8 units at top over 6 units of height.
3. **Capital:** A decorative top element with lotus-inspired geometry (wider than shaft, 1.3 units)

Instancing is achieved through a loop that positions identical pillar assemblies at intervals of 8 units along the corridor's length. Each instance shares the same vertex buffer object (VBO) but has a unique model matrix, significantly reducing memory usage and draw calls compared to creating separate geometry for each pillar.

2. Pendulum Traps **Why it is complex:** Multiple articulated parts (ceiling mount, suspension chain, pendulum arm, blade) that must move in synchronized motion while maintaining correct spatial relationships. **Implementation:** The hierarchical structure is organized as follows: **Parent:** Ceiling beam (stationary, defines global position)

- Fixed horizontal beam spanning ceiling width ($0.3 \times 0.3 \times 4$ units)
- Serves as anchor point for all moving components

Child 1: Pendulum arm (animated rotation)

- Long narrow cylinder (0.1 unit radius, 3 units length)
- Rotation pivot at top, connecting to ceiling beam
- Oscillates according to time-based sine function

Grandchild: Blade attachment (inherits arm rotation)

- Large flat rectangular prism ($2 \times 0.1 \times 0.5$ units)
- Positioned at bottom of pendulum arm
- Metallic texture with specular highlights to emphasize danger

The transformation chain multiplies matrices in the order: $M_{world} = M_{ceiling} \cdot M_{rotation}(t) \cdot M_{arm} \cdot M_{blade}$, ensuring that the blade maintains proper orientation relative to the swinging arm regardless of rotation angle.

3. Hieroglyphic Walls **Why it is complex:** Requires depth simulation on flat surfaces to create the illusion of carved reliefs without the geometric complexity of actual 3D carving. This technique is essential for achieving visual richness while maintaining rendering performance. **Implementation:** Walls are modeled as scaled cuboids (typically $0.2 \times 8 \times 6$ units for a corridor wall segment) with carefully crafted texture and normal maps. The normal mapping technique uses tangent-space calculations to perturb surface normals based on the normal map's RGB values, creating the illusion of surface detail. **TBN Matrix Construction:** For each vertex, three vectors are calculated:

- **T (Tangent):** Aligned with texture U coordinate direction
- **B (Bitangent):** Aligned with texture V coordinate direction
- **N (Normal):** Perpendicular to surface

The TBN matrix transforms normal map data from tangent space to world space: $N_{world} = \text{TBN} \cdot N_{tangent}$, where $N_{tangent}$ is the normal read from the texture ($2 \cdot \text{RGB} - 1$ to convert from [0,1] to [-1,1] range).

4. The Sarcophagus **Why it is complex:** Multi-part assembly representing one of the most detailed objects in the scene, consisting of base container, decorative trim, and sliding lid component with distinct material properties. **Implementation Structure:** **Base (Parent Object):**

- Rectangular prism ($2.5 \times 1.2 \times 5$ units) representing the main container
- Textured with aged stone material (diffuse map with wear patterns)
- Decorative gold inlay simulated through additional texture layer with higher specular values
- Serves as origin for child transformations

Lid (Child Object):

- Slightly larger rectangular prism ($2.6 \times 0.4 \times 5.1$ units) to create overlap
- Initial position offset by +0.8 units in Y-axis to sit atop base
- Animated translation along local Y-axis when interaction is triggered
- Transformation: $M_{lid} = M_{base} \cdot T(0, y_{offset} + y_{slide}, 0)$ where y_{slide} interpolates from 0 to 2.5 over interaction duration

Decorative Elements (Grandchildren):

- Corner posts (4 small cylinders, 0.15 radius, 1.5 height)
- Central cartouche (flattened sphere, scaled $1.5 \times 0.2 \times 0.8$)
- All inherit both base and lid transformations, moving with lid during interaction

The hierarchical organization ensures that when the lid translates upward, all decorative elements attached to it move together naturally. Material properties differentiate components: base uses high roughness (0.8) for aged stone appearance, while metallic decorations use low roughness (0.2) and high metallic factor (0.9) for polished gold effect.

6 Technical Implementation

6.1 Lighting System

The lighting system implements the Phong reflection model with three components: **Ambient:** $I_a = k_a \cdot L_a$ provides base illumination (typically 20% of total) **Diffuse:** $I_d = k_d \cdot L_d \cdot \max(N \cdot L, 0)$ simulates matte surface reflection **Specular:** $I_s = k_s \cdot L_s \cdot \max(R \cdot V, 0)^{shininess}$ creates highlights Total illumination: $I = I_a + I_d + I_s$, calculated per-fragment for smooth results.

6.2 Shader Programming

Two primary shaders are implemented: **Basic Shader:** For simple geometry (walls, floors) with texture mapping and Phong lighting. **Normal-Mapped Shader:** Extends basic shader with TBN matrix calculations for detailed surfaces. Both shaders receive uniform variables for light positions, camera position, and material properties, enabling dynamic updates without shader recompilation.

6.3 Texture Mapping

All textures are loaded using stb_image library and bound to OpenGL texture units. Mipmaps are automatically generated using `glGenerateMipmap()` to prevent aliasing at distance. Texture coordinates are carefully assigned during geometry construction to avoid distortion on non-uniform primitives.

7 Conclusion

7.1 Summary

The "Crypt of Thoth" simulation effectively demonstrates fundamental computer graphics concepts including lighting models, hierarchical assembly, shader-based effects, and interactive camera systems. The project highlights the power of modern OpenGL to create atmospheric and visually compelling environments using procedurally generated geometry and carefully crafted textures. Through the implementation of dynamic objects like swinging traps and interactive sarcophagi, the project showcases not only rendering techniques but also game development fundamentals such as collision detection, user input processing, and state management.

7.2 Future Enhancements

The modular structure of the codebase allows for several potential enhancements:

- **Shadow Mapping:** Implementing shadow volumes or shadow maps would dramatically increase visual realism, particularly in torch-lit corridors
- **Particle Systems:** Adding torch smoke, dust motes, and debris effects would enhance atmospheric quality
- **Collision Detection:** Implementing proper physics-based collision would prevent wall clipping and enable more complex trap interactions
- **Sound Integration:** Audio feedback for footsteps, trap mechanisms, and ambient environment sounds
- **Advanced Post-Processing:** Bloom effects for torches, depth-of-field for cinematic moments, and color grading for mood enhancement

The foundation established in this project provides a solid basis for these future expansions while maintaining clean, maintainable code structure.