AYDIN ADNAN MENDERES UNIVERSITY

ENGINEERING FACULTY

COMPUTER SCIENCE ENGINEERING DEPARTMENT

COMPUTER GRAPHICS FINAL PROJECT

CSE411 COMPUTER GRAPHICS, FALL 2025-2026

STUDENT'S NAME SURNAME:

231805003 – ÖZCAN ERDEM TOSUN

LECTURER:

ASST. PROF. DR. SAMSUN MUSTAFA BAŞARICI

CSE 411 - PROJECT #2: ADVANCED UTAH TEAPOT SIMULATION REPORT

1. Introduction

The objective of this project is to implement a dual-window OpenGL application centered around the "Utah Teapot" model, demonstrating proficiency in geometric transformations, user interaction, and advanced rendering techniques. As outlined in the project requirements, the application features two distinct viewing environments: an interactive perspective view controlled by mouse inputs and a multi-view orthographic/perspective control panel. Furthermore, to satisfy and exceed the bonus requirements, the project implements a procedural high-definition texture generation algorithm and a complex lighting model to simulate a porcelain material.

2. Implementation Details

2.1. Multi-Window Management and Architecture

The application is built using the GLUT library, utilizing its ability to manage multiple contexts. As specified in the requirements, two separate windows were created using **glutCreateWindow.**

Window 1 (Interaction Window): Dedicated to user manipulation. It utilizes a perspective projection matrix.

Window 2 (Control Window): Displays fixed camera angles. To ensure smooth animation and prevent flickering, GLUT_DOUBLE (double buffering) and GLUT_DEPTH (depth testing) were enabled for both contexts, as recommended in standard OpenGL practices [1]. The window callbacks (display, reshape, mouse) are registered separately for each window ID to ensure event isolation [2].

2.2. Procedural Texture Generation (Bonus Implementation)

Instead of loading a static bitmap image, which limits resolution and quality, I implemented a **Procedural Texture Generation** algorithm. This approach calculates pixel colors mathematically at runtime, creating an "infinite resolution" effect.

- **Algorithm:** The makeHDFlorealTexture() function synthesizes a circular floral pattern (Mandala).
- **Mathematical Foundation:** The algorithm converts pixel coordinates (x, y) from Cartesian to Polar coordinates (r,θ) using the **atan2** function [7].
- **Pattern Synthesis:** A sinusoidal wave function is applied to the angle and distance:

$$P = \sin(\theta \times 12 + r \times 0.05) + \cos(r \times 0.1)$$

This technique, inspired by procedural modeling methods [4], allows for the creation of organic, seamless patterns without the artifacts associated with image scaling.

- **Color Mapping:** The generated scalar values are mapped to a specific color palette (Cobalt Blue and Porcelain White) to mimic the style of traditional Chinese porcelain.

2.3. Texture Mapping and Mipmapping

Since the **glutSolidTeapot** geometric primitive does not generate default texture coordinates (UVs), standard texture mapping would result in a solid color. To solve this, I utilized Automatic Texture Coordinate Generation:

- **Object Linear Mapping:** The **glTexGen** function was configured with **GL_OBJECT_LINEAR** mode [5]. This projects the generated procedural texture onto the teapot's geometry as if projected from a plane, ensuring the pattern wraps around the curvature of the object effectively.
- **Filtering:** To ensure high visual quality at various distances, `gluBuild2DMipmaps` was used with `GL_LINEAR_MIPMAP_LINEAR` filtering. This eliminates aliasing artifacts (moiré patterns) when the object is zoomed out [3].

2.4. Lighting and Material Properties

To achieve a realistic "Porcelain" look, a specific material property set was defined:

- **Specular Highlight:** The **GL_SHININESS** value was set to 120.0 (near the maximum of 128.0) to simulate a highly polished, glassy surface [1].
- **Dual Lighting Setup:**
  - **GL_LIGHT0:** Acts as the key light (sun).
  - **GL_LIGHT1:** Acts as a fill light from the bottom to illuminate the underbelly of the teapot, ensuring the texture details are visible even in shadowed areas.

3. User Interaction

3.1. Mouse Rotation (Window 1)

In the first window, the user can rotate the teapot freely. This was implemented using glutMouseFunc to detect clicks and glutMotionFunc to track movement. The difference between the current mouse position and the last click position is calculated ($\Delta x, \Delta y$) and added to the global rotation angles (angleX, angleY).

3.2. Pop-up Menu System (Window 2)

The second window features a hierarchical pop-up menu attached to the right mouse button. The menu structure includes:

- **Camera Angles**: Allows switching between Top, Bottom, Left, Right, and Perspective views using gluLookAt to reposition the camera dynamically.
- **Texture/Color Modes**: Allows the user to toggle between the generated HD Texture and solid material colors (Ruby, Emerald, Gold, etc.). This interaction logic relies on GLUT's menu management system [2][6]

4. Experiences and Conclusion

Developing this project provided deep insights into the difference between fixed-function pipeline rendering and procedural generation. One of the main challenges was mapping the texture onto the complex geometry of the teapot without explicit texture coordinates. I overcame this by researching and implementing OpenGL's `glTexGen` functionality. Additionally, tuning the mathematical parameters for the procedural flower pattern required significant experimentation with trigonometric functions to achieve an aesthetically pleasing result.

The final executable successfully meets all core requirements: dual-window rendering, interactive rotation, and menu-based view switching. Furthermore, the inclusion of procedural texturing and advanced material properties fulfills the bonus criteria, resulting in a visually distinct and technically robust application.

5. References

1. **Shreiner, D., et al. (2013).** ***OpenGL Programming Guide*****. Addison-Wesley.**
2. **Kilgard, M. J. (1996).** ***The OpenGL Utility Toolkit (GLUT) Programming Interface*****. Silicon Graphics, Inc.**
3. **Hearn, D., & Baker, M. P. (2004).** ***Computer Graphics with OpenGL*****. Pearson Prentice Hall.**
4. **Ebert, D. S., et al. (2003).** ***Texturing and Modeling: A Procedural Approach*****. Morgan Kaufmann.**
5. **OpenGL.org. "glTexGen - OpenGL 2.1 Reference Pages." [Online].**
6. **Lighthouse3D. "GLUT Tutorial - Pop-up Menus." [Online].**
7. **Bourke, P. "Polar Coordinates and Texture Generation." [Online].**
8. **Course Material: CSE 411 Projects .pdf.**