Mehdi Mihir

Southern New Hampshire University

Professor Battersby

CS 330: Computer Graphics and Visualization

12/16/24

Final Project Design Decisions Document

Development Choices Analysis

The development of this 3D Zen garden scene required careful consideration of both artistic and technical elements to create an authentic representation while maintaining performance and code maintainability. The scene's primary elements were strategically chosen to demonstrate mastery of various 3D graphics concepts while creating a cohesive, contemplative environment.

The container design employs a flattened cylinder with a precise rim detail created using a torus. This approach was chosen over alternative methods (like combining multiple boxes) because it provides smoother curvature and better represents the circular nature of traditional Zen gardens. The container's dimensions were carefully calculated to provide an optimal viewing area while maintaining realistic proportions.

For the wooden bridge, I implemented a half-torus for the main arch combined with cylindrical support posts. Suggest by Professor Battersby, I agreed that this approach offers several advantages over using a series of boxes or a single curved mesh:

- Provided natural curvature that matched a very traditional Japanese like bridge design
- Allows for precise control over the arch's height and span
- Good texture mapping across the curved surface since it was already provided in ShapeMeshes.cpp
- Has a lot of visual consistency when viewed from different angles

---

Scene Navigation Implementation

The navigation system was designed to provide intuitive camera controls while maintaining the contemplative nature of a Zen garden viewing experience. The implementation uses a combination of keyboard and mouse inputs:

1. Primary Movement Controls:
   - WASD keys control horizontal movement (forward, left, backward, right)
   - QE keys handle vertical movement (up and down)
   - Mouse controls camera orientation with smooth interpolation
   - Mouse scroll adjusts movement speed for precise positioning

2. Camera System Features:
   - Dynamic perspective switching between orthographic (O key) and perspective (P key) views

- Smooth transition handling between view modes
- Collision-free movement for unobstructed viewing
- Speed adjustment system that prevents abrupt camera movements

The orthographic view was specifically implemented to provide architectural-style viewing angles, which is particularly useful for appreciating the garden's geometric layout and symmetry.

Code Modularity and Organization

The project's code architecture was designed with modularity and reusability as primary considerations. Several key custom functions demonstrate this approach:

1. Scene Component Management:

```
void SceneManager::RenderScene() {
    // Organized into clear sections for base components, bridge structure,
    // and decorative elements for improved readability and maintenance
}
```

2. Material System: The material system was designed to be highly reusable through a structured approach:

```
void SceneManager::DefineObjectMaterials() {
    // Each material (marble, container, rim, etc.) is defined once and
    // can be reused across multiple objects, reducing code duplication
    // and allowing for consistent visual properties
}
```

3. Object Creation:

```
struct Rock {
    glm::vec3 position;
    glm::vec3 scale;
    float rotation;
};
// This structure allows for easy creation and modification of multiple
// rock formations
```

I believe the code's modularity is particularly evident in the texture and material handling systems, where I created reusable functions that can be applied across different objects. This approach not only reduces code duplication but also ensures consistency in visual appearance and simplifies future modifications.

For example, the SetShaderMaterial() function already created encapsulates all material-related logic, making it easy to apply consistent material properties across different objects while maintaining clean, maintainable code. Similarly, the bush creation system uses a layered approach that allows for creating complex organic shapes from simple primitives, demonstrating how modular design can create sophisticated visual results.

This modular approach not only improves code organization but also facilitates future enhancements and modifications to the scene. Each component can be modified independently without affecting other parts of the system, providing a robust foundation for further development.