**Module 7: Project Reflection**

Shannon Musgrave

Computer Science Dept. of Southern New Hampshire University

CS-330 Computer Graphics

Professor Shaykhian

8/18/2024

To create the scene from the selected painting, several mesh shapes needed to be instantiated as a base for the objects in the scene. These shapes would not be adequate for the task in their raw form, but by scaling each shape in 3-dimensions, they can be modified for use in the scene. First, a plane is rotated on the x-axis and scaled to make a backplane. This would give the illusion of a wall behind the scene. Second, a box is created and lengthened on its x-axis to simulate a rectangular chest. This chest is the in the forefront of the scene and is used to table all the other objects in the scene. Two more small boxes were created and scaled on the y-axis and were used to simulate the illusion of straps on the front of the chest.

The main centerpiece of the scene is located on the chest. This is comprised of a vase, melon, and leaves. The vase is the most complicated object in the scene, it is constructed using a torus, tapered cylinder, and a half torus. The torus could be used because it was set in between the chest and the tapered cylinder, this gave it the illusion of being the rounded part of the vase. It was also scaled in the y-axis to be the appropriate height. The tapered cylinder is positioned directly on top of the torus, they were given the same texture so that they would blend together to look as if they were one object. Lastly, a half torus is positioned to look like the gold handle of the vase.

The melon was created by using a sphere, since a melon is not perfectly spherical, the sphere had to be scaled to different values in the 3-axis to give it an oblong shape. It was also rotated in such a way that it would mimic the position of the melon in the scene. It was placed to the left of the vase with just enough room to allow some leaves to be inserted between them. Finally, the leaves were created and distributed around the scene. These leaves were circular in shape but would be very thin in real life. To simulate this shape, a sphere was used, and its z-axis was scaled to be almost zero. This made the sphere very thin and leaf-like.

Although the camera’s initial position and direction is adequately placed to appreciate the scene as it was depicted in the painting, the ability to move the camera in virtually any direction was enabled. The controls for the camera were set up to use the keyboard to move forward, backward, left, right, up, and down (w, s, a, d, q, e). These directions are in relation to the camera view, As the camera view is changed, the directions will be applied in the new orientation. The mouse is used to change the orientation of the view. As the mouse is moved around, the direction of the view is moved with the mouse. The speed of the camera’s movement, however, is changed with the mouse wheel. In addition, the keyboard keys “p” and “q” are used to toggle between the orthographic and perspective views. This allows the scene to be rendered in 2D or 3D modes.

The code in the project is written in a way that helps to maintain order, especially if the project continues to grow. Each function follows the single responsibility principle, which states that “every class, module, or function in a program should have one responsibility/purpose in a program” (Abba, 2022). Specifically, within the “RenderScene” function, the code was removed and calls to specific functions were added, such as, “DrawBackdrop” and “DrawChest”. By writing the code in this way, it is much more manageable and less bloated, allowing future developers to have an easier time finding and modifying the code. For instance, if the melon needed attention, instead of sifting through the “RenderScene” function, a developer could simply find the “DrawMelon” function and go directly to the needed code. By keeping to this single responsibility principle, code will be more reusable, allowing another coding paradigm to be more closely followed “Don’t repeat yourself.”

**References**

Abba, I. (2022, April 26). *SOLID Definition – The SOLID principles of Object-Oriented design explained.* freeCodeCamp.org. https://www.freecodecamp.org/news/solid-principles-single-responsibility-principle-explained/