Liam Daley

CS 330

December 6th, 2023

Kurt Diesch

Project Design Decisions

**Justify development choices for your 3D scene.**

I wanted to capture a still life that could, even with limited resources, tell some kind of story. I selected the city scene because the colorful and varied nature of the objects caught my eye. As I planned to implement the scene, I thought about which objects were the most important to capture for the atmosphere of the scene. The central building was a given. After that, I selected the tree to inject some natural life into the scene, and the car to add a distinctively human element. The sidewalk and road were necessary elements to set the scene, and finally I decided the trash can placed center frame was the final component that would tie the scene together.

After the scene was assembled, I decided to experiment with light and color to find what kind of atmosphere I could evoke. I found with the tools provided, I could most effectively create a darker, nighttime scene. What I settled on uses two light sources, one for the moon and one for the faint blue glow nighttime city ambiance. The moon can be seen reflecting in certain materials, such as the metal and glass on the car and the lacquered wood doors. This functionality to change the specularity of objects by material was not based on any provided examples and is a custom extension of the shader and rendering code.

**Explain how a user can navigate your 3D scene.**

Following the project requirements, the scene can be navigated using the W,A,S,D keys for horizontal translation and Q/E to move up/down. Additionally, the mouse cursor can be moved to swivel the camera around. The scroll wheel will change the rate at which the camera traverses, which is a useful feature for getting closer to objects without blasting through them. These controls are achieved using code created during milestones, linking a number of OpenGL library callback functions.

**Explain the custom functions in your program that you are using to make your code more modular and organized.**

I work professionally as a game developer, so I knew from the start I wanted to make some sort of modular framework for the entire rendering system to be based around. What I settled on in a previous milestone is the “GLObject” struct and “sceneObjects” structure array. An instance of GLObject contains all the data required to draw a primitive shape, including the category of shape itself, an enum representing which of a set of textures the shape would like to use, values for position, rotation and scale, and a separate UV scale value for finer tuned adjustment of UV maps.

These structs are defined in the code and then looped through in the “ConstructSceneObjects” method, which calls the proper mesh constructor to finish initializing the object. In the render method, we loop through the array of GLObjects, using the stored matrix and vertex information to render our scene objects more efficiently. This data-oriented approach to object construction made modifying the scene itself much easier, as all the data defining an object was stored in one easy place. It made, in particular, the final scene composite quite fast to pull together, as I could create new objects and move them around with very little work.

For your convenience, I have attached the original image.

