Angel Zhang
ECE 462 Computer Grahpics
Prof. Sable
05/02/2017

Implement Environment Mapping

Project Description

This project demonstrates several perfectly reflective objects, one at each time, surrounded by a skybox and moving objects. A skybox is a large cube that surrounds a scene and is textured with images that form a background for that scene, in all directions. In this project, the reflective object is centered at the skybox, which represents a room in Lancellotti Chapel in Rome, Italy [1].

Controls and Options

There are three options available to choose from the reflective object menu: they are a cube, a sphere, and a teapot. There is an additional option, which says “No object,” allowing the user to see the surrounding environment without placing any reflective object inside. Also, the user can change the size of reflective object, which is only available for cube and sphere, by choosing either “Small,” “Medium,” or “Large” in the “Size of The Object” menu, which has a default value of “Small.” When the user selects “No object” or “Teapot” already, the size menu will disable accordingly.

Next to the size menu, there is text filed that allows the user to specify how many dynamic objects that he wants to place into the environment. There are four different geometries for the dynamic object: a solid cube, a solid cone, a solid cylinder, and a hollow cylinder. After write down the number, simply press the “Enter” button, that many objects would first line up and appear behind the reflective object, assuming the “Animating” checkbox is unchecked. However, each time the user hits “Enter,” parameters of the reflective object, its size, and the

animation setting will be restored to default. This is not a bug, and it would not happen when you just change the geometry or size of the reflective object. Because all the models of the reflective objects are initialized already in the WebGL graphics context, but not for the moving objects, changing the number of moving objects essentially changes the entire environment, so such change would require to reset everything first. There will be more explanation about how to actually map the dynamic environment in next section.

Once the user enables the animation by clicking the checkbox, “Animating,” these objects would start rotating around its own object coordinate system while orbiting the reflective object, which is centered at the origin. For each moving object, all parameters including rotation axis and angular velocities in local and global are randomly chosen as well. Moreover, these objects’ angular rotation speeds are randomly assigned, and so do their colors since the main purpose of creating these objects is to demonstrate how they are mapped into the reflective object, rather than focusing their motions in great precisions.

Next to the “Animating” checkbox, there is a reset button, which would restore the default settings: the reflective object is the cube, the number of the moving object is 5, and the animation is disabled. To rotate the view, simply drag with the mouse on the picture. To rotate the reflective object, the user could use the arrow keys to control.

Explanations

The challenge in this project is how to render the six images of the scene that are needed for the cubemap texture. The main idea is to point the camera, placed at the center of the object, in the six directions of the positive and negative coordinate axes and snap a picture in each direction. To get all the details correct would require the view from the very point on the surface of the object, not the view from the center of the object, but realistically we can't make a

different environment map for each point on the surface. Thus, the approximation, placing the camera in the center, will gives good enough results as long as other objects are not too close to the reflective surface, and that's why all the moving objects' initial positions are given specifically. What I mean by a "camera," it really means a projection transformation and a viewing transformation. The projection needs a ninety-degree field of view, to cover one side of the cube, and its aspect ratio will be 1, since the sides of the cube are squares. Also it is needed to aim the camera in different directions first before taking the pictures. Because of the details of how the images must be stored for cubemap textures, we need to apply a transformation so that the image snaps taken by the camera inside would match the sides in the layout are viewed from the outside of the cube as needed [2].

In this project, there are two sets of cubemap textures. This means that the cubemap texture used for the skybox is not the same as the one for reflection map because the moving objects are introduced. Therefore, in a dynamic environment, all six faces of the reflection cubemap are redrawn for each frame of the animation. This projects uses render-to-texture for a cubemap texture. It is a technique, in which the output of a rendering operation is written directly to a texture, and it is implemented by attaching the texture as one of the buffers in a framebuffer. In this case, to render a 3D scene to a framebuffer, a color buffer and a depth buffer attached to the framebuffer are used.

There are more comments throughout the code to explain what each function does and how it is used. Please enjoy playing around with my project.

[1] Lancellotti Chapel, in a room in Lancellotti Chapel in Rome, Italy.
http://www.humus.name/index.php?page=Textures&start=24
[2] Eck, David J, *Introduction to Computer Graphics*