1 Introduction

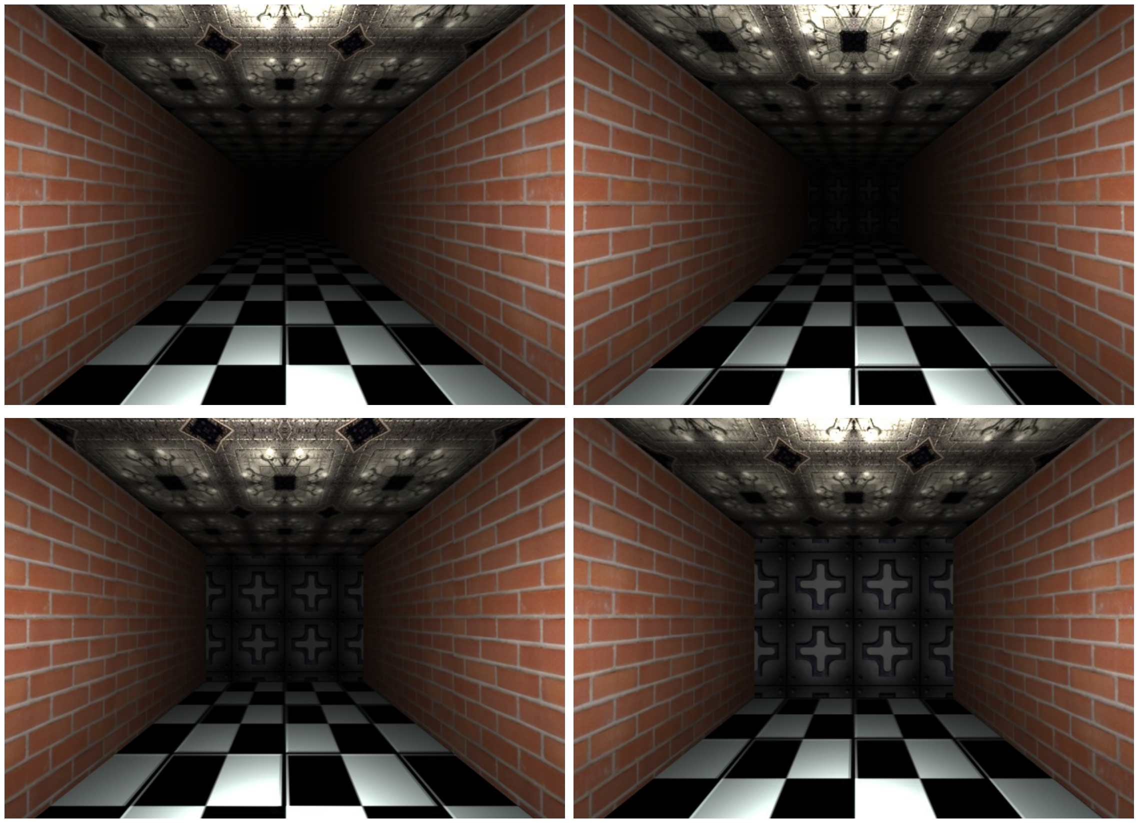
Write each program using either Windows or Linux using Code::Bolocks or using the Mac and XCode, zip up all your work and upload the projects to the Assignment #7 page of the MyClasses site for this class.

In these exercises you need to update the DoxyGen documentation. Fully document all new data members and methods to any of the classes, and update the documentation for anything that has changed, including the documentation in the main that gives an overview of the program.

2 Exercises

1. In this exercise you will be playing with the attenuation aspect of lighting to produce a spooky hallway. Create a hallway with the shown textures in the ceiling, walls and floor. Make the hallway long. You may use whatever dimensions you would like, mine is 500 units in length. Using the YPR camera, create a camera that can move only forwards and backwards, using the up and down arrow keys. You do not need to alter the YPR camera, just only implement forward and backward movement in the UI. The movement should stop before the user backs up out of the hallway. The end of the hall should contain a wall with the shown texture. Forward movement should stop before hitting the wall.

   Use only one light that is always positioned either at the camera position or slightly in front of the camera. Putting it in front of the camera tends to make the lighting on the side walls, floor, and ceiling look more natural. Play with the attenuation to produce the candlelight/lantern affect of the program. This type of visual works better if you do not do the full lighting calculations, update the fragment shader to take the texture color and simply multiplying it by the attenuation calculation. So you would not do the diffuse, specular or spot calculations, just attenuation times texture. In the shaders, strip off all of the unnecessary calculations and just use what is needed.



2. In this exercise you will be implementing full lighting and textures. Back in the 70's or 80's one popular item was an acrylic photo cube that doubled as a paperweight. This exercise is to program one of these. The user interface specifications and several screen shots are below. Each face of the cube has a different face on it. Each face is of a famous person either in Mathematics or in Computer Science. You will probably recognize some of them and you should recognize the ones you do not.

   As always I want you to design and implement the program as best you see fit but I will give you a couple pointers on how I implemented the program. Fell free to ignore these if you have a different solution.

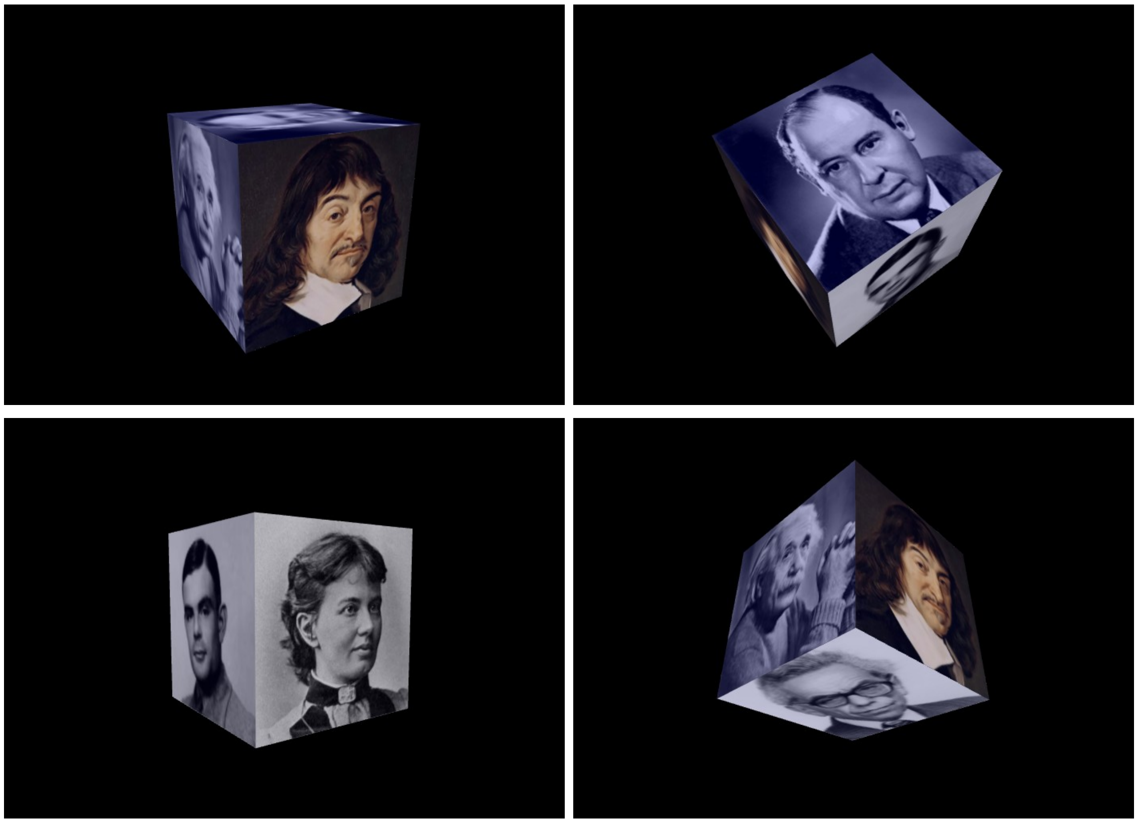
   (a) The shaders could be part of the Cube class but I put them in the Graphics Engine. I had it take care of loading all 6 images to the graphics card and then selecting which to draw depending on a uniform integer.
   (b) I updated the Cube class to incorporate texture coordinates on each face.
   (c) I made it so that each face was in its own VBO, and updated the draw function to draw each face independently.
   (d) Considering the above setup, this meant that I needed to have some communication with the shader and the cube, since the cube draw function would be switching between the images in the shader. All I did to make that happen was send in the shader program address to the draw function of the Cube.
   (e) For the lighting I used three light sources at the following $(r, \theta . \phi)$ locations, in degrees, $(50, 45, 45)$, $(50, -45, 100)$, and $(50, -100, 60)$.

   User Interface:

   - Escape: Ends the program.
   - M: Toggles between fill mode and line mode to draw the triangles.
   - F5: Turns the cube lights on.
   - F6: Turns the cube lights off.
   - F7: Turns the cube texture on.
   - F8: Turns the cube texture off.
   - F10: Saves a screen shot of the graphics window.
   - F11: Turns on the spherical camera.
   - F12: Turns on the yaw-pitch-roll camera.
   - If the spherical camera is currently selected,
     - Left: Increases the camera's theta value.
     - Right: Decreases the camera's theta value.
     - Up: Increases the camera's psi value.
     - Down: Decreases the camera's psi value.
     - A click and drag with the left mouse button will alter the theta and psi angles of the spherical camera to give the impression of the mouse grabbing and moving the coordinate system.
     - A click and drag with the left mouse button while the Control key is pressed will alter the radius (using the change in the y coordinate) for zooming in and out.
     - Ctrl+Up: Decreases the camera's radius.
     - Ctrl+Down: Increases the camera's radius.
   - If the YPR camera is currently selected,
     - Left: Increases the yaw.
     - Right: Decreases the yaw.
     - Up: Increases the pitch.
     - Down: Decreases the pitch.
     - Ctrl+Left: Increases the roll.
     - Ctrl+Right: Decreases the roll.

- Ctrl+Up: Moves the camera forward.
- Ctrl+Down: Moves the camera backward.
- Shift+Left: Moves the camera left.
- Shift+Right: Moves the camera right.
- Shift+Up: Moves the camera up.
- Shift+Down: Moves the camera down.

Different views of the cube.



Below are the different lighting and texture modes for the cube. The upper left is with the lighting off and textures off, notice it is just a pass-through shader. The upper right is with lighting on and textures off, blue plastic cube. The lower left is with lighting off and textures on, note the blending of the colors and textures. The lower right is with lighting on and textures on, note the blending of the material color and the textures.