Devin White

CS 5610

Interactive Computer Graphics

Project 7-ShadowMapping

- **What was implemented:**
  - The plane was implemented and the sized and positioned properly
  - Started on shadow buffers
- **What you could not implement.**
  - Shadows not implemented
- **Additional functionalities beyond project requirements.**
  - No additional functionalities at this time
- **How to use your implementation.**
  - Right click and drag to zoom in and out (the teapot)
  - Left click and drag to rotate. Left and right rotates left and right, up and down rotates up and down(the environment)
  - P switches between orthogonal and normal transforms
  - F6 recompiles the shaders.
  - ESC exits the window
  - CTRL and left click slightly moves the lighting
  - Alt + Left click rotates the teapot
- **What operating system and compiler you used**
  - Operating system: Windows 11
  - Programmed in Visual studio: used VS internal compiler(gl????)
- **External libraries and additional requirements to compile your project.**
  - Same as previous projects:
  - Required libraries:
    - FreeGlut
    - Glew
    - CyCodeBase
    - LodePNG

In addition, FreeGlut.dll and Glew.dll were required to be put in System 32. Libraries were put in folders next to the project along with the associated headers. Both LodePNG.h and LodePNG.cpp need to be included in the headers as well.
A header file with the functions in the program is also included. Main.h
In addition, I used:

```
#include <string.h>
#define _USE_MATH_DEFINES
#include <math.h>
//I had to put this to make it work...for some reason
#pragma comment(lib, "glew32.lib")
```

There were really no other changes from the things required for previous projects. Everything should be run using the exe in the zipped folder in either debug or release

Figure 1. Plane rendered and in proper place

Figure 2: Started on shadows(doesn't work yet. ☹)

```
//Render to depth texture
void initDepthTexture()
{
    //initialization
    shadowProgram = glCreateProgram();
    //Create frame buffer
    glGenFramebuffers(1, &frameBuffer);
    glBindFramebuffer(GL_FRAMEBUFFER, frameBuffer);
    //Create depth buffer
    glGenTextures(1, &depthMap);
    glBindTexture(GL_TEXTURE_2D, depthMap);

    glTexImage2D(GL_TEXTURE_2D, 0, GL_DEPTH_COMPONENT, shadWidth, shadHeight, 0,
        GL_DEPTH_COMPONENT, GL_FLOAT, 0);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_NEAREST);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_NEAREST);

    //configure frame buffer
    glBindFramebuffer(GL_FRAMEBUFFER, frameBuffer);
    glFramebufferTexture(GL_FRAMEBUFFER, GL_DEPTH_ATTACHMENT, depthMap, 0);
    glDrawBuffer(GL_NONE);
    glReadBuffer(GL_NONE);

    if (glCheckFramebufferStatus(GL_FRAMEBUFFER) != GL_FRAMEBUFFER_COMPLETE)
    {
        //return false
    }
    //glUniformMatrix4fv(glGetUniformLocation(shadowProgram,"mvp"),1,GL_FALSE,matrixMLP);

}

void renderCameraView()
{
    //RENDER CAMERA VIEW
    //glBindFrameBuffer(GL_DRAW_FRAMEBUFFER, origFB)
    //glViewport(0, 0, windowWidth, windowHeight);
   //glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    //gldrawArrays();
}
```