Muhammad Hilman Beyri

Completed

- generate meshes from obj models
- geometry buffer
- directional light
- directional light shadow
- spot light
- spot light shadow
- point light
- point light movement
- animate slerp between two orientations for spot light

1. Build

- -This program can only be run on windows with SFML installed.
- -cmake ../src on folder build
- -open p4.sln
- -set p4 as startup project
- -set p4 debugging command arguments to ../../scenes/cube.scene or ../../scenes/toy.scene
- -Build and run

2. Camera controls

W, A, S, D for three major axis movement and I, J, K, L for changing the camera's orientation

3. Design Overview

- The meat of the program is in Renderer::render.
- Flow :
    - create shadow map for directional light
    - render the directional light on all pixels
    - for each point light, do stencil pass of the sphere and then render the point light
    - for each spot light, do stencil pass of the cone, create shadow map, and then render the spot light
    - render the light accumulation buffer to the screen

4. Scene Geometry

After loading meshes to static models, I created a linked list of render data that holds informations that are needed to render that static model such as vertices open gl id, indices open gl id, etc

5. Shadow Mapping

- The shadow mapping of this renderer is sharp shadow. If i have more time, probably I could make it prettier by sampling around the pixel and average it to make it soft shadow
- In order for directional shadow mapping to work correctly in this renderer, you need to put bounding box specification in the scene file.

6. Changes to scene format

- rotation or orientation, now uses quaternion
- you can specify the bounding box for the scene to help with the directional light shadow mapping
- you can specify the camera properties
- added slerp properties for animating the spot light
- added correction to easier adjustment of the light intensity of the spot light

7. Solved Problems

- The code is hosted on my github (https://github.com/bysreg).
- First I thought vertices from different groups in a OBJ file is strictly separate. That causes the toy.scene was loaded incorrectly. Later, I found out that all the groups shared the same vertices.
- Initially, I spent a very long time figuring out how to compile with SFML and messing around with CMake on windows. The starter code did not compile succesfully
- I also solved the problem of the mesh loaded cant load face that has four vertices. I think mesh loading is not really relevant in this project and it would be wonderful if it is already handled in the first place
- implement camera movement with still using fixed pipeline
- implement shader
- using VBO
- transition to dynamic pipeline from fixed pipeline
- problem figuring out how to use GLM, and spent long time to figure out how to use quaternion with GLM
- problem with flipped texture. seems like sfml texture's first pixel is from top left while the open gl expects it to be from top bottom
- transition to new GLSL version 330. previously using GLSL ES 2.0
- bugs using open GL API. took a very long time and hard to debug. Thing that scares me the most is black screen. The cause of the problem could be anything like forgot to bind shader program, forgot to specify GL_TEXTURE_MIN_FILTER and GL_TEXTURE_MAG_FILTER (the sampled texture will be black on some computer), forgot to clear bit, and already done clear bit but forgot to enable depth mask, wrong parameters, wrong size of indices / vertices (should be number of vertices times the size of the vertex structure itself rather than just the number of vertices), etc
- if i have more time, I would also clean up the code, make sure no memory leak, remove code duplication, etc

8. Screenshots