Assignment 1 Logbook

COMP612 Graphics and Animation

Jinwoo Lee (18018154)

3/03/2020 (2.5 hours)

- Downloaded GLFW,GLEW, GLM libraries for the assignment. GLFW/GLEW are utility libraries for managing Opengl windows/contexts etc and GLM is a math library for calculating matrix transformations,dot products etc.
- Created CMake text file to load in dependencies. This will take care of all the linkers/library paths for the project as well as creating the project.
- The main.cpp file will be doing in the order of the following:
    - First check if GLFW can be initialized and initialize.
    - Set sampling rates per pixel and set the versioning to 3, which is the most current version of Opengl.
    - Set the window size and title, set the rest of the parameters to null as they are not needed.
    - Check if window has been initialized and not type of null.
    - Initialize GLEW and check for its error in initialization.
    - Set key capture to true, so that the window can be closed on ESC.
    - Set the clear color to white and clear clear buffer bit.
    - Nothing's drawn since we're just testing the window.
    - Terminate window when the loop is broken by ESC.

4/03/2020 (2 hours)

- Created an index buffer to factor out points that are redundant in drawing a square (3rd and 1st indices being repeated to close off the two triangles in building one)
- Created an error handle functions to handle errors. Previously, if an error/errors had occurred, the black window would show up with no errors in the console.
- Defined an assert function to call for compiler error function that are specific to gnu and msc.
- Created GLLogCall function, which will log what error had occurred at which function and which line.
- GLClearError clears previous errors from the console in a new loop
- GLCall function will pass in the string representation of the function name, file path, and the line in which the error had occurred.

- Now, wrapping the GLLCall function with all the functions in the code will allow me to see where and which function had the error occurred.

10/03/2020 (3 hours)

- Shader.cpp contains LoadShaders function, which loads shader as the name implies. This function will read two GLSL files, vertex shader and fragment shader. It will first create the shaders with glCreateShader() function with parameters that specifies types of shaders. It will then read the two files from the file path specified. Once read without errors, it will then compile the two shaders, attach them to the program, and detach/delete shaders once done.
- The base class for shapes such as Triangle, Circle, Lines are all implemented. The rectangle will be drawn from the Triangle class.
- GLBegin and GLEnd are deprecated functions. I had found out during the process on making base classes for shapes. It will affect the assignment as it will be much more set up to do to draw shapes. There are no reference as to how to assign buffers and bufferlayout etc. Therefore, I will have to spend more time to do research on my own. GlLineWidths is also a deprecated function.

12/03/2020 (2.5 hours)

- Bitmap fonts are also deprecated and cannot be imported in standard Opengl 3.3 as it does not use GLUT. GLUT had been unsupported for a decade. I don't see a reason for me to use or learn outdated information for this assignment.
- Moving on, I've added buttons and made a pointer to reserve the previous color state and toggle the color of buttons.
- The button listeners are implemented by GLFW library. I had written conditional statements to check if the user is clicking within the x and y coordinates.
- Making the sun shades was quite tough. It was not the same as drawing the circle. As line width function is deprecated, they are left as they are. On the online, it says that it was deprecated since it was expensive and Tessellation shader should be used instead. This is outside of the scope of this assignment as it was not taught at all.

15/03/2020 (4 hours)

- The trees are now implemented. The root is made off of a rectangle and the rest changes according to seasons. Randomizing spawn coordinates was done within the triangle class.
- The trees will receive seasonal information from button listeners and change accordingly.
- Added grapes for summer trees. It was little bit time consuming as grapes are made of lots of circles
- Enabled depth test. Lots of breaking due to coordinate systems being set in vec2. Fixed.

- Enabled alpha blend. Had to change lots of color values but now the sky seems like less of a blend sheet of blue paper

17/03/2020 (4.5 hours)

- Implemented toggle slider for the night – day shift extension. Using GLFW mouse release detection it wasn't bad making it to work initially. However, the click listener and the release has to have some sort of toggle boolean variables now to see if it was a initial toggle or it was a click to end a toggle. Lots of boolean algebra.
- Wrote a math operation to convert slider coordinates to the window coordinate so that the sun or the moon will be able to change its positions based on the slider's positions.
- The sun seems to be too fast, it was hardcoded with a value that seemed most optimal. Glfw delta time can be used but it was not optimal.
- The moon will pop out when the sun hits the middle of the screen. The sun will then disappear and when it does, it does not get drawn behind the scene; applies as well to the moon.

18/03/2020 (5 hours)

- Added effects for the sky to be brighter during the day by reducing alpha. I'm sure if I have more time I will be making shadows/light source, but again, not really the scope of this assignment nor do I have knowledge to do so.
- A lot of refactoring to be done as some of them are drawing hundreds of elements and it's taking up the memory quite of bit.
- Reduced stack variables and allocated all of vertex arrays to the heap.

19/03/2020 (7 hours)

- Particle system is deploying/emitting correctly but not between the change of season as the loop only checks for the unused and do not properly kill the particle on time to not show them onto the next season.
- Particle class seems redundant and can be a struct. However, it is only there for the assignment requirement.
- The particle class was initially initializing the shapes but moved it to the system as it should have been the system to change it.
- Slight bug fixes in the randomization of particle coordinates by using normal distributions instead of standard rand() function.
- Draw calls seem expensive, but batch calls(instancing) are also out of scope for this assignment at the moment.

Future Improvements:

- Particle system can be significantly improved by using instance calls to change buffer data and reduce its draw calls.
- Need to add texts but it seems quite hard to do it without bitmap texts. This can be done for the future improvements.
- Classes are little bit hardcoded for single draw calls. Indexing can be improved.
- Shaders could’ve been used to improve textures.
- The main class seem to be overblown. There can be a renderer class and abstract all the details out of the main or application file for better reading.