Overview

This Project creates a floor of cube objects the serve as platform where a boat sails across water tiles and juts across mossy tiles. The aim of the game is to get as far as possible with out damaging the boat on mossy tiles. This is achieved by “Jumping” with the Spacebar. The Game is made using C++ and OPENGL using a wrapper encouraged by the labs.

Research

Researching into different lighting techniques, learn OPENGLs website was useful in diffuse lighting. This addition to the existing methods of lighting given in the labs uses the mathematical concept that the light reflected from a surface is brighter if the surface in question is flat on. This is because light hitting a surface diffuses out spreading its intensity. The OPENGL resource explained how diffuse lighting worked and with implemented it spit the process up into different sections such as finding the normalised normal and the dot product of normal and light directions.

High-level description

The program works mainly around the wrapper built in the 3D labs which allows .obj file to convert to a series of triangles and for the low level OPENGL functions to take these and draw them on to the screen. This is done in a more efficient manner by taking the data of the triangles and putting it onto the memory of the graphics card through the vertex buffer. This is done as traveling to the graphics card is the slowest part of the process. In terms of sorting objects, the object files and coordinates are stored in Game Objects and theses are stored in an Object Manager. These nesting of classes and vectors is useful for organisation and code neatness and furthermore makes updating all the very easy. Having all of the in one easily itteratable vector means updating all objects is done in one function. One limitation and workaround for this I found was having the ability to update a specific section of objects and creating a struct “Groups” was a nice work around. Having a separate vector for a selection of Game Object pointers meant the objects were in both vectors but also had the ability to be called separately. The downside of my implementation of doing this was having to delete an object or create and object with in this function required a lot of framework and therefore functions telling classes to find game objects in vectors. Another thing I did that could have been improved was the physics. Initially I had a large array of physics classes and functions. This was too complicated and was ultimately added nothing to the gameplay. Instead I replaced it with one modular physics class in Game Object. This was null unless otherwise specified easily distinguishing between physics and non-physics objects. This class dealt with the more complicated moving object of the boat. It has velocity and gravitys that make the boat have an arc in the air. In the .zip there is a Dia diagram a .png with the diagram on it.

Analysis

One thing that I have done very well is my implementation of the object manager and the “Group” struct. This is because these sections scale up and are reasonable featured in that the positions, rotations physics and Boolean states of these objects and be edited at within the manager and with in groups is an easy way to sperate objects but not split them into groups having any number of objects in any number of groups. The downside of this aspect of the program is that if the objects are not “named” the functions have a hard time to find specific objects. One thing I would do differently would be in corporate groups closer into the objects manager as well as using a namespace for nested vectors. This is because it was very annoying to use a very confusing looking nested statement every time, I would want to call objects. This way conceptualising a vector of vector as say a “manager” would be easier to explain to others. The weakest part of the project was the player class and camera class. These classes, in my opinion, could be done much better in future projects by utilising inheritance and other Object orientated programming concepts. This is because lots of functions were duplicate code from Game Object. One other thing I will look into will be the ability to call the class “above” a classes own class. Rather than having to store a pointer to that class inside the nested class. This in my opinion could be very dangerous as it has the potential to be an infinite loop and cold crash the program very easily. Finally, the something I was implemented poorly on my part is how the program creates and uses textures and models. Ideally, I only need one of each object and texture to render all of the objects. In my program on the other hand, every object has a texture and model. This is means that the game will have extremely bad performance at high model counts. In future I will experiment with issuing a texture and model manager. This will store textures in an efficient way and give out pointers to these on need basis. Overall there is a lot I could improve on. The most important thing I have learnt from this assignment is how structure is the most important thing when doing large projects and everything else surfers when it is done poorly.

Referencing

Learnopengl.com. (2019). *LearnOpenGL - Basic Lighting*. [online] Available at: https://learnopengl.com/Lighting/Basic-Lighting [Accessed 2 Jan. 2019].

Anon., 2019. *free 3ds mode rowboat boat* [online]. Turbosquid.com. Available from: https://www.turbosquid.com/3d-models/free-3ds-mode-rowboat-boat/675083 [Accessed 29 Dec 2018].

Brown, A., 2019. *Huntington photo by Aidan Brown (@aidanbvisuals) on Unsplash*[online]. Unsplash.com. Available from: https://unsplash.com/photos/eti4sSPrVO4 [Accessed 31 Dec 2018].