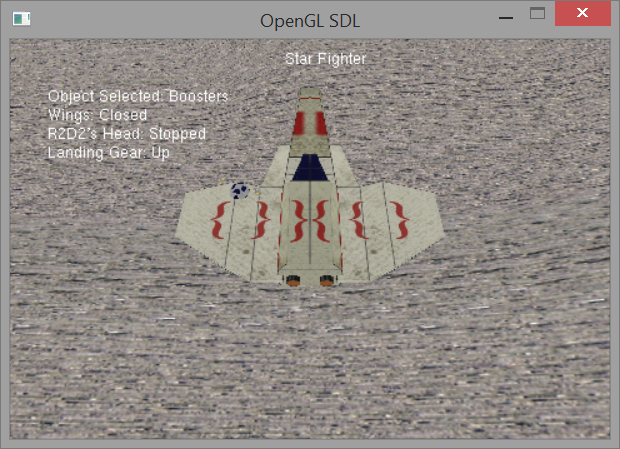
OpenGL Assignment - Jet Fighter



Description

The application loads in multiple objects to create a jet fighter which was made in Blender3D. The user can control different aspects of the spacecraft which includes, deploying wings and landing gear, moving the spacecraft forward and rotating, and stop and start R2D2's head spinning. The scene also has fixed animation, which is the stars rotation around R2D2's head. These move relative to each other. The objects are set in a scene graph so that when the spacecraft moves, so do all the other objects attached to it.

Controls - Camera

W - Pitch Down

S - Pitch Up

A - Yaw Left

D - Yaw Right

Q - Roll Left

E - Roll Right

Left Arrow - Strafe Left

Right Arrow - Strafe Right

Up Arrow - Move Forward

Down Arrow - Move Backwards

Controls - Jet Fighter

F - Object interact (Boosters or Landing gear)

R - Deploy Wings

T - Stop/Start R2D2's head spinning

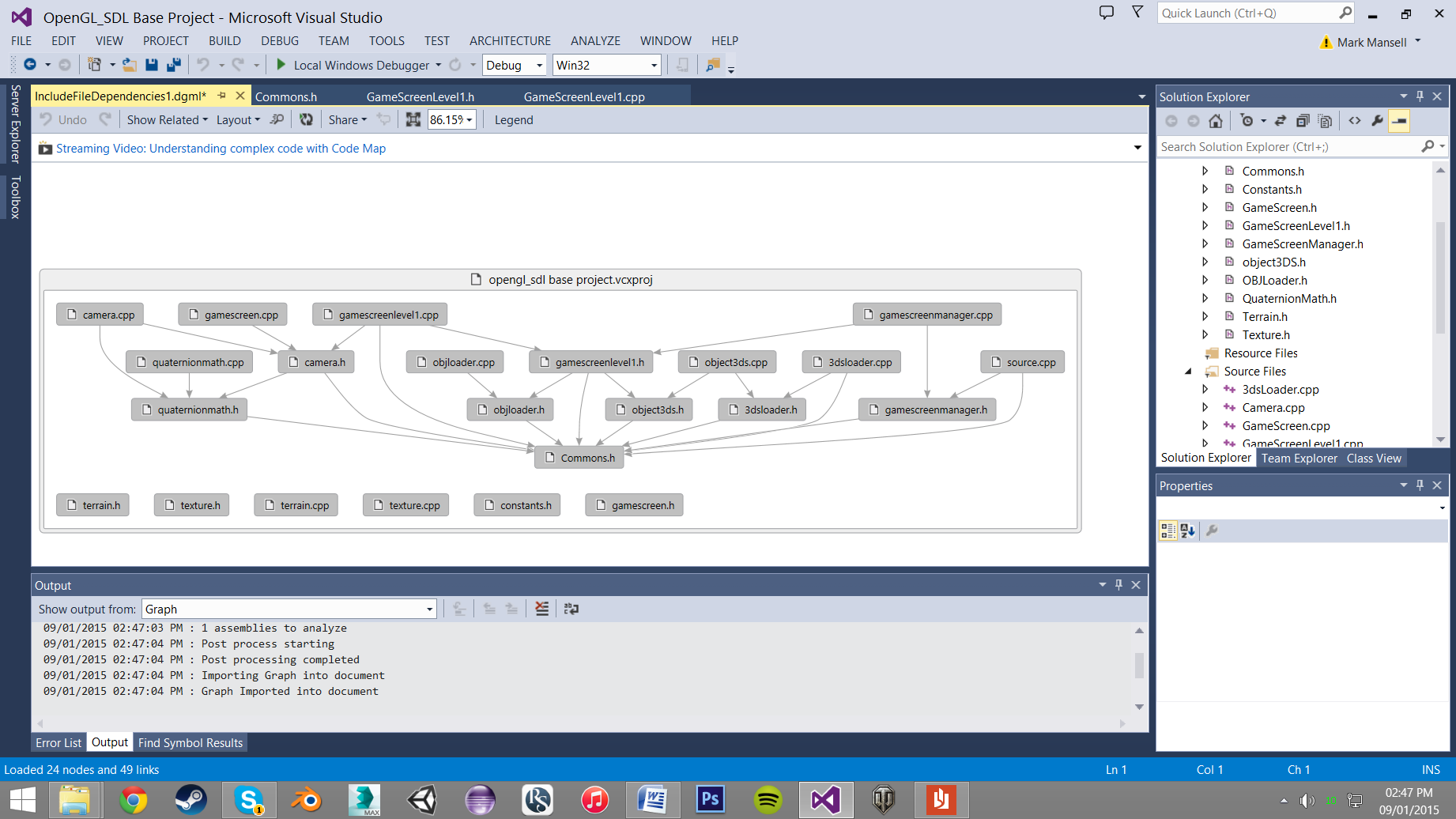
1 - Select Boosters

2 - Select Landing Gear

Z - Rotate Left

X - Rotate Right

UML Diagram



Critical Reflection

Texturing

When sorting out the texturing there were some problems with the alignment in the texture on the models. After some research I found out that the file format for the images (.raw) stores in data backwards. to solve this problem I had to flip vertically all my textures before exporting them to .raw.

Object Loader

After setting up the scene with the .3ds loader I attempted to code the .obj loader. I found some examples online that I adapted to with the code from the 3dsloader. It loaded in a cube but it didn't look correct (one of the vertices stretched off screen). After debugging I found out that the indices were not selected the right vertex in the array e.g. index 1 is the first vertex but in the array of vertices 1 is the second vertex, 0 is the first one. To solve this I just -1 from all the indices so that they select the correct vertices in the array.

Because the object loader used the same structure used by the 3DSLoader I didn't have to create new one and only added an extra array to the structure for storing UV indices which is needed for my obj loader.

Selecting Objects

For selecting objects I chose to swap between the landing gear and boosters. This was because they would rarely be changed at the same time. If I was to change something about the selecting of objects I would have added the wings as an extra selectable object or just have the wings and landing gear as the selectable objects as they do the same thing (rotate to a fixed angle).

Overall structure

To make the coding a bit more easier to read I would have put all the coding that loads all the models and textures into its own function so that the Initiate function in the level 1 screen is shorter and easier to read.

Camera

The camera in the scene works basically but cannot fly around the scene very well. This is due to gimble lock. I could not figure out how to fix this so I left the camera a basic one. If I would to change something I would fix the camera so that it is easier to fly around the scene and maybe have a separate camera that follows the spaceship.

Changes

Some of the changes I would make to the application would be the ability to fly the jet around the scene. The spaceship does have simple movement but doesn't have enough to make it practical to move around the scene with.

I would also code the scene graph differently as it is just all in the update function. I would probably have all the scene graph in its own function so that the update function isn't cluttered.