Evaluation:

I believe that I have managed to hit most of my objectives that is set out in the analysis of this project.

My objectives

* Users will be able to download this project easily:
  + I have achieved this by posting the project on github.com. this will allow people to easily access the code and download it from any device connected to the internet. This will also allow people to give feedback on the project with improvements that I could make in the future. This also allows people who find bugs in the code to report them so I can fix them.
* Users will be able to save the progress of their game:
  + This has been achieved by saving all the objects in the game to a Json format. This can be easily converted back to objects in the editor when the user wants to load the saved project. This is done by using a library called Gson, created by google, which simplifies the serialization and deserialization of objects.
* Users will be able to edit the code to add extra features:
  + I have successfully achieved this by making this project open source so people can download the engine and see all the code, being able to edit/add to it. I have named all the variables and methods in this code relevant to what it stores/does. This is so people going through the code can see how each method works and how each variable is used. I have also included a readme.txt file when you download this project. It contains links:
    - docs.gl, which is the documentation for OpenGL. This is so if the user wants to add anything to how the objects are rendered then they have an explanation for all the functions that can be used in OpenGL.
    - ImGui github, which is the documentation for ImGui. this is if the user wants to add features to the windows.
    - JetBlains IntelliJ IDEA. This is the code editor I used when coding. I have provided a link to this a I feel it was a helpful IDE, with helpful feature and plugins that you can use to help when coding.
* Users should be able to save on one device and load on another device loading from this engine:
  + I have done this by when loading a game, you must select the folder it is saved in. this filepath is then saved as a root folder. This can be used when loading textures as when saved, textures store their filepath to that texture. If the folder is moved, then the filepath to get to the image will change. The root folder is used if this error occurs when loading a texture, with /assets/ + filename added to the end for the full filepath to the file. This is done for all the files loaded into the game engine.
* If there is an error, a relevant error message is output:
  + This has been achieved. when doing major/important tasks, I have put it in a try catch clause. This catches the error caused, in most cases printing the line that caused this and, in some cases, asserting false and halting the program (only when there is an error in an integral part of the game engine, meaning the game engine will not work without it functioning properly). This also outputs a line of text telling the user what the most likely cause is. For example, if there was a problem finding a file, the code would output “there was an error loading ‘filepath’.”.

User Feedback:

I think it met most of the criteria set out for this project. I found it easy to download but there is no executable or compiled file to run from so you need an IDE to compile and run the project. this is helpful if you want to edit the code but if you just want to run it and create a game then having to use an IDE is an extra step people may not know they need to do if they don’t read the readme.txt file included in the download. This project can save a game and load them easily with little user input, only needing to select the main folder of game. I tried moving a game save to a different hard drive on my computer and it seemed to load this with no errors. When an error came up, I found it helpful that there was an error message telling me what was wrong. There is a problem with the code as when you create a new game, a lot of errors are shown telling me that there was a problem loading the game. This confused me as I created a new game so there shouldn’t be anything to load from. I also think there should be a help button explaining how each part of the game engines UI works as when I first started messing around with it, I didn’t really understand how to use the windows and what they do. After getting used to how the UI works, I found that it was a good engine, being quite easy to use when you know what you are doing.

Feedback response:

I think they made good point about how to improve the code. I thought that if I left it as the main source code, people would be able to go through the code, edit it and add functions if they wanted to. I didn’t think that I would exclude many of the potential users of this game engine as you need to download an IDE to open, compile and run the code. Most people may find this a pain to do and get confused on what IDE to use etc. I should have included an .exe to run a compiled version of the code. The error messages can be helpful but also confusing if everything is working correctly and still displaying error messages. I could have added checks to stop outputting errors if everything is working as intended like in the example in the feedback. As I knew what each part of the engine does and how to use the UI, I didn’t think about users who haven’t used the engine before and didn’t give them any support when trying to use the project for the first time. I could have added a help panel users could click bringing up information on how to use each panel and what they do.

How I would take this project forward:

* I could add a way to export games so they can be played without using the game engine
* Add more features to edit the window like changing the resolution of the viewport, its zoom using the mouse
  + Add a setting panel you can use form a dropdown. These settings can be changed there to what ever the user wants. You could also store variables for the custom key binds the user has set. In OpenGL key codes are stored as integers so you would store the key bind value as an integer and when changed is set to the integer of the new key. This variable can be referred to by other classes to get these key binds
* Add a way to group objects together and select multiple objects at once to move/scale/rotate them.
* Add ways to save objects individually and import them from a separate folder
  + I would do this by letting the user select a folder with an object in. this will contain the images used in that object and all the necessary data to create that object in a text file. This will be saved in the Json format so I can easily import it into the engine.