Game Engine Design Document

Ryan Hood

P1617694X

Contents.

Introduction.

This document will outline the main components which will form the game architecture assignment. The document will contain both an introduction and conclusion; following the introduction will be the design of the engine, this will include a flowchart and a description of how the engine will run, I will also include a section of how the engine will work with user interaction. Next, a plan for the architecture of the engine, how the files will be stored and how they will interact with each other. I also aim to include a section which will describe the inputs for the user, this will include a brief outline of which key corresponds to what function. The asset pipeline section will cover the use of the assets within the engine and how they can be expanded and placed within the virtual environment. My plan for the software testing section is to conduct an in-depth stage of testing following the completion of the engine; this will consist of unit testing where each function will be given a measurable piece of data and the output value checked to determine the correctness of the function. I will also create a list and outline of the coding standards which will take place in my code, increasing the readability of the game engine and allowing for easy maintenance at a later date. Also, I will include Doxygen within my game engine allowing for a set of class descriptions will be created for the project enabling the game engine to easily be evaluated and altered at a later date as well as, providing sufficient documentation outlining the functionality of the game engine.

Engine Design.

The main design for this engine is to make everything as modular as possible to ensure it's reusable, to this end; components are an important part of the design. The main idea is to create a class with the desired function (like controlling a camera) and creating a component which will control that class, this will allow for it to be attached to a game object allowing for the game object to having the desired class features and functionality.

Ryan Hood
P1617694X
Engine Design Document
Hoodrn03@gmail.com
p1617694x@my365.dmu.ac.uk
https://github.com/Hoodrn03

Architecture Plan.

Using a folder structure for the engine is important, it allows for the different objects to be managed more easily. Also splitting assets into sub-folders also makes their management simpler. This engine will follow a structure where the .cpp files will be stored in an src folder, .h files in an include folder and all of the assets in folders of their type (like a model folder for models, a level folder for levels, ext).

User Input Options.

For the current state of the engine, there will be a limited set of inputs and actions with those keys.

- MoveForward - W.
- MoveBackward - S.
- MoveRight - A.
- MoveLeft - D.
- Next Camera - Z.

And all of these keys will be changeable via a config file but the key codes for the key must be known.

Asset Pipeline Assesment.

Using the folder structure aforementioned this will make the asset pipeline more easily implemented. Meaning that models will be easy to add to the game, they just need to be loaded into the game engine in order to be loaded.

Levels would also be easy to load using the models, JSON files are a good way to load levels however being able to convert the level data from within a 3D modelling program (like Maya) is more difficult to create, but once a system for converting a scene into a JSON file is created then the loading into game part should be rather effortless.

Having a config file for the key codes is also a great plan for a game engine, it will allow rebinding keys a better task as they only need to be changed at a single place and only needs to be done once and loaded upon start-up.

Software Testing Strategy.

The plan for testing the game engine is to ensure that each component works independently, this will help to prevent one component from breaking another. So, unit testing is the best approach to the game engine. Unit testing follows a simple structure; where a test is created, followed and if the code passes then a new test is created otherwise the code is adjusted and fixed until it passes the test.

Coding Standards.

The game engine is a tool to use to other projects so, coding standards must be followed to ensure future development is possible. The coding standards I aim to use is Github to ensure that source control is held, allowing for removal of broken code and saving to an online repo (to view this repo https://github.com/Hoodrn03/Game-Engine).

Within the code, I aim to use Doxygen. This will allow me to create a set of pages showing the structure of the engine. It will allow for the improvement of the engine to be possible at a later date it could also allow for portions of the code to be identified and used in other places.

Within the code, I plan to use certain prefixes to make sure the code stays consistent.

- (m_) - For member items.
- (l_) - For local items mainly local variables.
- (v_) - For vector containers.