Scriptable 2D Game Engine

Course-Specific learning outcomes

Throughout the course of this project several learning outcomes will be covered as well as potentially others.

- Use knowledge, abilities and skills for further study and for a range of employment in areas related to scientific and technical computing.
- Interpret legislation appropriate to computer professionals and be aware of relevant ethical issues and the role of professional bodies;
- Analyse, design, and implement algorithms using a range of appropriate languages and/or methodologies;
- Apply the principles and operation of languages, compilers and interpreters;
- Demonstrate effective communication, decision making and creative problem solving skills, and identify appropriate practices within a professional, legal and ethical framework;

If the engine is deemed during research to require an AI system the below would be applicable.

- Critically appraise and apply suitable artificial intelligence techniques for a variety of software systems.

Project Background

Game development can be very difficult without the correct tools, especially the prototyping phase. This project would allow for the creation of a game engine that is fit for prototyping an idea as well as being adapted for the final production version.

Prototyping is difficult as engines with excellent performance such as Unreal Engine or even Unity are very heavy and slow to work in and would take a vastly larger amount of time to get to a high-quality working example, especially in 2D. LOVE2D is excellent for quickly prototyping a game but the performance would suffer when the game began to grow in scope, a great way to remedy this is to have most of the heavy-duty code running on the C++ side. The game engine that needs to be created is one that is easy to prototype in as well as have the performance to handle a fully-fledged game.

Similar to how Unreal Engine 4 has high level drag and drop programming in the form of blueprints as well as full engine access in C++, an engine that is much lighter weight and easy to develop in is yet to be made especially for 2D game development.

Aim

Design and implement a viable open source 2D game engine using C++ that implements an interface to the backend for use in a high-level embedded scripting language for rapid and powerful development.

Objectives

Research and review what existing 2D game engines use for rendering, sound and which scripting language was used, as well as design decisions such as how much control the embedded language gets and how much of the game engine is written in the embedded language versus C++. Research into game engine design techniques will also be a necessity as engine design is a very deep and complex subject.

Allow interaction with game objects from both C++ and the embedded high level scripting language. This will be extremely important for ensuring that the designed game engine will be highly functional and easy to use.

Have various entry points that should be synonymous with what other game engines have such as standard entry points like initialise, update and draw.

Ensure full expandability. This will be a major factor when it comes to designing and implementing the core engine as the engine will need to be as general purpose as possible and should be tweakable for any use case.

Improve performance beyond LOVE2D by using techniques such as moving more of the core processing to the C++ side.

A maintainable codebase is very important for expandability and will ensure a fluid development process through the project as well as into the future beyond the project.

Document the engine and scripting language API. Implementing a game in a short space of time is very difficult without documentation as developers without any intimate knowledge of the engine will still be required to fully utilise the various features of the engine.

Compete in a game jam to verify how effective the various aspects of the game engine are under high stress rapid development.

Problems

There are many potential problems that could be run into. With each objective comes potential failure points that would need to be worked through as they are encountered.

Interaction with game objects from both C++ and the high-level scripting language could be very difficult and would take an excellent infrastructure plan to allow for it to be entirely implemented especially in a fluid and simple way.

Expandability may also prove to be a problem when it comes to the implementation as engine design is a very complicated subject, a lot of research will have to go into this part of the project to ensure that the design allows for the required feature set.

Ensuring that the documentation is all up to scratch could be very time consuming and could end up pushing the timetable out.

Required Resources

For the completion of this project no additional resources are required.

Timetable and Deliverables

Research – In depth research will be required before starting any of the other objectives to ensure that the design is on target.

Engine Implementation – The entire development process should be entirely possible within approximately 20 weeks unless there are any major hold ups.

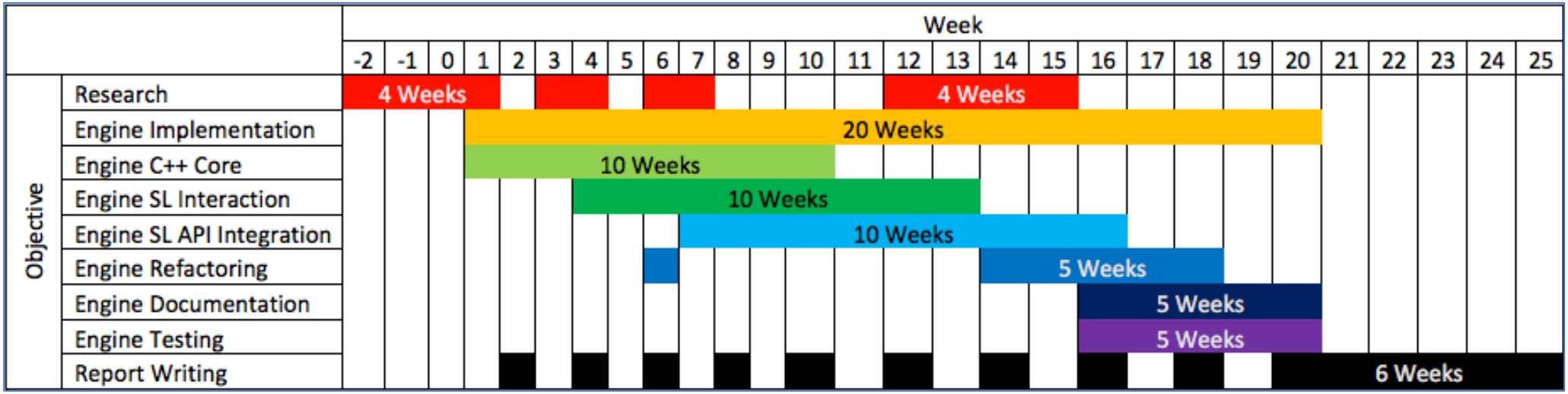
Engine C++ Core – The core of the engine will be very important for ensuring that the performance will be better than equivalents.

Engine SL Interaction – Ensuring that the scripting language has enough core engine interactions through things like keypresses and other hooks should take around 10 weeks.

Engine SL API Integration – Designing and implementing the scripting language API should take around 10 weeks.

Engine Refactoring – Refactoring will be completed in minor cases throughout the project, but the major refactoring that will be required after the engine is implemented should take around 5 weeks.

Engine Documentation and Testing – Documenting the engine will take place while testing every part of the engine and should take around 5 weeks.



Cyrus Hanlon 14061650

Huw Lloyd (supervisor) 29/10/2017