Project report – Journey to the sun roguelike

Candidate Number:Centre Number:Qualification Code:Date: 17/12/23

Rohan Sim

Contents

[2 Introduction 1](#_Toc153742887)

[3 Analysis 1](#_Toc153742888)

[4 Bibilography 1](#_Toc153742889)

# Introduction

This report will serve as the primary destination for everything related to this project. Its purpose will be to provide information for the progress of the project throughout its lifecycle. This report will also discuss the requirements for the project and analyse the different features and computational techniques that will be required before even beginning development. This will take up a large proportion of the document, as it is important that the requirements have been fully analysed and investigated, so that they are clear, unambiguous, and able to be easily implemented into the solution when development begins, with minimal issues and delays due to unforeseen events that could have been mitigated had more time been spent planning.

Once development has begun, we will be using this report to document the process from start to finish. This may come in the form of screenshots taken from the solution itself, with descriptions on the purpose of the code shown, and information on any issues that may have been encountered when writing it and how they were resolved. This will be beneficial as if similar issues are encountered in the future, we will be able to reference this document and come to a fix faster as we will know the process required to fix it. It will also be useful to show the stakeholders how the project is developing to keep them informed and have something to reference when discussing future improvements or testing.

# Stating the Problem

This section will overview the problem that we are trying to solve with this project. It will discuss the things relating to the problem such as the gathering of requirements in order to get a full view of what the solution will entail and how it will be accomplished, along with identifying the key stakeholders of the project who will have the best idea as to what they want to gain from the solution.

## The Problem

This game will be a roguelike twin-stick shooter game, where the player will complete levels on each of the planets in the solar system, starting on Neptune, and ending the game in the final level which will take place on the Sun. Each level will start the player with a basic gun, firing projectiles at nearby enemies. Items that power up the player can be found throughout the level in randomised locations that may give the player a better weapon, or improve their stats. At the end of each level, the player will collect an armour piece that will be able to absorb the energy of the sun, that will prevent it from becoming a supernova. All armour pieces will have to be collected from each planet in order to achieve this and beat the game. To my knowledge, this game concept has not been created before, and for that reason I believe it will be an interesting and unique project that has a solid selling point.

## Computational Methods

The core of this game will be the different planets that the player will explore, and the maps that will be used to represent them. In order to create these levels procedurally, while still keeping correct theming, object-oriented programming will be used. This will allow us to create classes for important things such as enemies, weapon archetypes, and the maps themselves that will be generated based on a class containing the generic features and functions for each level. This will result in code that is much easier to read, and be more expandable to make the creation of new levels much simpler and easier to implement into the game while keeping them unique and allowing the developers to add new content to the levels to make them different and unique to each other.

Other methods, such as abstraction, may be used during the development of the game, and can be used in cases such as the spawning of enemies. It may be beneficial, for initial testing, to create code that allows for entities to spawn outside of a given radius, at random intervals, and in random spaces. This will get the core functionality of the enemy spawning system in place, which can then be later expanded upon to include the sprites and AI that will be used to make the enemies move, follow, and attack the player. The method of abstraction can be used throughout the project to ensure that all systems implemented work on a purely functional level, such as having a sprite move around a plane with the WASD keys, and colliding with a wall to prevent further movement in that direction. This will be able to be expanded upon by implementing the player sprite, along with the required perspectives for the player character, and creating walls around the player to define the play-space. By abstracting these functions, we can be sure that the systems work as intended before fully implementing them into the game.

## Stakeholders

The stakeholders of this project are the people who we must create the game in mind for. We will base all requirements off of their feedback, and must ensure that throughout development, that they are being catered for in the content that is added to the game. For example, if one of the stakeholders believes that the projectiles fired by the enemies are too fast, we must conduct testing on the speed, and possibly allow the play-testers to run through some levels with projectile speeds adjusted to different speeds, from which they can then help us to finetune the gunplay of the game to be as fair as possible while still keeping the player engaged, and with a certain level of challenge. The stakeholders for this game will be my classmates, as they are a part of the key demographic for this game, as well as my future colleague, who will provide aid and feedback on the technical aspects of the project, as their primary requirements for the project are that is runs optimally on all hardware, so that it remains accessible to all players, regardless of their PC specifications.

# The Approach

The following headings will discuss how the project will be approached, and the software-based requirements that will be used, such as the IDE (integrated development environment) and the programming languages that will help us to create the game. We will also decide on the development methodology that we will use for the project, between ones such as Waterfall, Agile, and Spiral.

## Game Engine

A couple of game engines were considered for use in this project. Those were Unity and Monogame. Monogame was considered due to it being easy to set up, simply installing it onto Visual Studio and creating a solution with it like any other project. It would allow us to create a simple 2D game, which would be beneficial for this project, however one of the drawbacks that made us unlikely to use it was the fact that there was not a preview window of the game available outside of runtime, where objects and assets could be moved graphically, rather than by code. This was available in Unity, making the creation of levels and layouts much more manageable and easy to create in our short timeframe.

Another benefit of Unity was that we would be able to easily import assets from the Unity store directly into our project, which would not be possible with Monogame, with us either having to create them ourselves, or find them online and then import them ourselves. The Unity Asset Store will help greatly with intuitive imports, and also allow us to purchase higher quality assets than those we would likely come across searching elsewhere.

Unity also allows us to modify the properties created in the code directly from the engine, and modify numeric values such as speed and attack damage without having to go into the code itself for the object we are modifying.

## IDE

For this project we will use Visual Studio for the IDE. This is because it is well integrated with both Windows machines, which this game will be created in mind with, as well as Unity, the game engine. It has a wide range of libraries that can be imported and used that will make various aspects of development significantly easier, as we will be able to use pre-made functions specific to what we want to achieve instead of having to create them from scratch. This will greatly cut down on development time, and there is also a good chance that the functions from the libraries will be better optimised and less prone to issues and bugs than ones we may make ourselves for the same purpose.

Visual Studio as an IDE has other features, such as Intellisense, break points, and unit testing capabilities that will make general development and code-writing much easier. Intellisense can be used in all aspects of development to autocomplete lines of code. As you begin writing code, intellisense will attempt to guess the code you are trying to write, to which you can then press the tab key to automatically enter it into the code. This minimises the risk of misspelling words or missing out other syntax, as the computer writes the code itself, and can be beneficial in suggesting likely functions that will be used that you may not have known about to achieve the goal you were working towards.

# Analysis

This section, the analysis phase, will

# Bibliography