Phase 2 - Design SDD

OSRS Clone

Mitch - Kyle - Malik - Laith

Table of Contents

1.0 Introduction

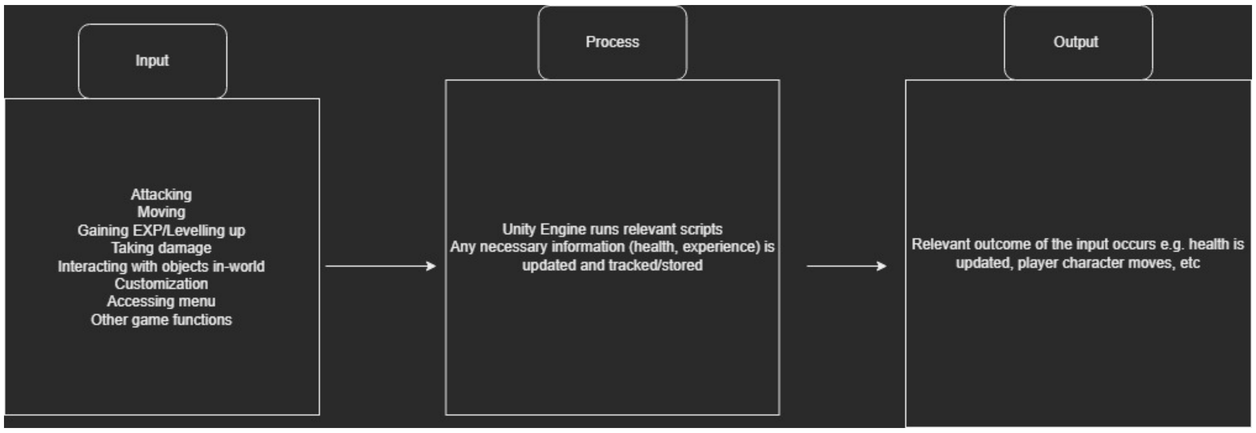
This document will briefly cover the design of our program and discuss limitations. The requirements, goals, and scope of the project laid out in our previous documentation have not changed. We are still aiming to make an old school runescape inspired game that has the basic functionality to be played as a game. So far, we have not made use of data structures outside of arrays, and our UI is in early phases of development. The restrictions on our team remain the same, but we will discuss these restrictions more in the context of program design.

2.0 Data Design

We are using arrays in our level-up system. The array is used to store how much experience the user needs per level, and how much accuracy they have per level. Our camera uses Vector3, which is a vector in UnityEngine that represents 3d vectors and points. The Vector3 stores camera coordinate positions.

A full description of the data structures as well as a complete diagram of them would be easier to create when the project nears completion.

3.0 Architectural and Component-Level Design


Input
Attacking
Moving
Gaining EXP/Levelling up
Taking damage
Interacting with objects in-world
Customization
Accessing menu
Other game functions
Process
Unity Engine runs relevant scripts
Any necessary information (health, experience) is updated and tracked/stored
Output
Relevant outcome of the input occurs e.g. health is updated, player character moves, etc

4.0 User Interface Design

The UI for our game will be inspired by the UI for old school RuneScape. There will be 3 separate panels: one with tabs focused on stats, game info, options, and such, another as a text box to recall game dialogue and what quests the player is in/has completed, and a final panel with a minimap and settings that work along with the minimap.

5.0 Restrictions, Limitations, and Constraints

Some issues we have experienced in the design of the program have to do with dealing with dependencies, as well as tackling the general scope of the project within the amount of time and resources available. Problems can occur when a component of the code, such as a certain class, script, or method, is implemented but is later needed to be iterated on and a dependency is found. This issue arises because of the large scope of the project and our limited experience. In terms of design, this means that we are not always capable of implementing the best way of doing something the first time, and that we may not recognize these flaws while planning. The amount of time and collaborators available also can cause issues like dependencies to arise during project design. Overall, however, the level of resources available online help a lot with creating a good design, especially for making an old school runescape game, which is not as complex as other games. We also plan for the project design to be constantly changing, which helps make errors mean less due to the fact that it was planned with the intent to be changed.

6.0 Testing Issues

The previous documentation described how we would test, but issues related to testing were not really explored. While we are doing test-driven development and can check if our code is working as we go, it is not the same as how an actual organization would test their code. We have limited experience with automated testing, as well as other forms of testing outside of just checking if it works or is breaking in any obvious ways. This may lead to problems as more components are implemented because bugs will become more obvious.

7.0 Appendices

7.1 Packaging and Installation issues

N/A

7.2 References/Supplementary Information

N/A