Software Requirements and Design Document

For

Group 9

Version 1.0

Authors:
Alice Bishop
Andrew Eikman
Christopher SanGiovanni
Jon Guzman
Mathew Paravila Jose

1. Overview

The project is a 2D platformer titled “The Astro Nots” in which the player can choose from a number of different protagonists to play and must navigate across several obstacles and traps to reach the end of the level and start the next. While navigating the foreign landscapes, players can choose to collect the bits and bobs they find along the way for an extra challenge which will reward them with points.

Players can choose to customize their experience and follow their own path by choosing from a number of playstyles and aesthetic differences depending on the hero they choose. Additionally, they can use the treasures they find along the way to unlock more features, giving completionists and aestheticians alike a goal beyond the main campaign to strive for.

2. Functional Requirements

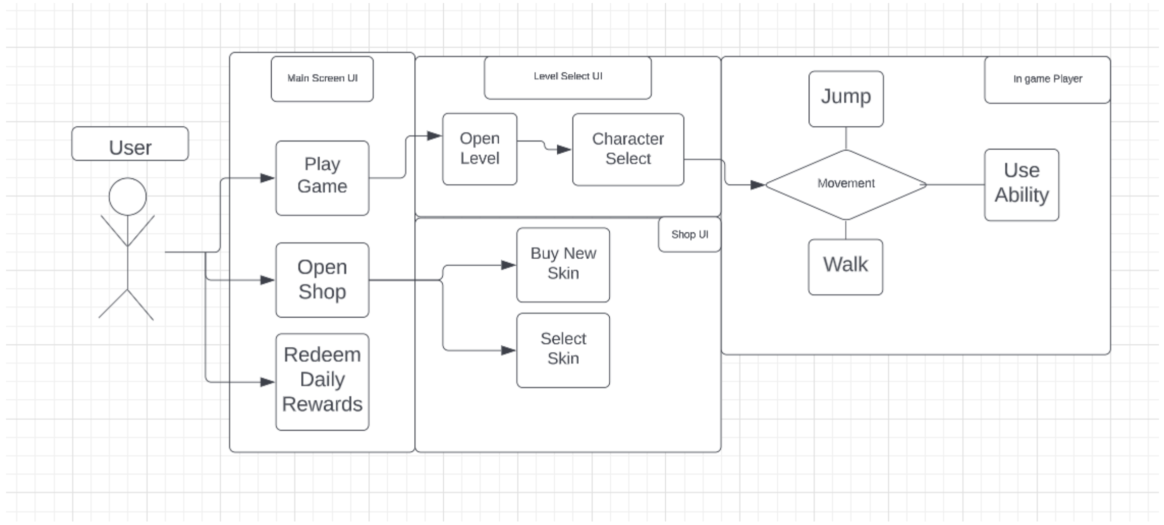
- Player Movement (HIGH)
    - The player should be able to use the keyboard to move their character left and right, and jump. After the character has jumped, they should still be able to influence their horizontal movement by using the keyboard while they are in the air. All of these movements should have animations to go along with them: Running left, running right, jumping up, falling down, standing still
- Player life (HIGH)
    - When the player’s character collides with a trap, their character should die. This death should be made obvious with a death sound and animation, followed by a brief pause before the level that the player is currently on is reloaded by reverting to the state the scene was in at the very beginning.
- User Interface (HIGH)
    - When the player first opens the game, they should be met with a main menu that will allow them to do any of the following
        - Start the game from the level of their choice
        - Open the shop, and return from it back to the main menu when they are ready
        - Quit the application
    - While the player is in the game, their user interface should provide them with information such as their score and/or how many items they have collected. This part can be flexible depending on if features such as a timer are added in the future that would warrant expansion of the in-game UI.
- Item Collection (MEDIUM)
    - The player’s character should have the ability to collect items, such as cherries, when walking over them. After collecting the items, the items should disappear from the game and be added to the player’s collection of all the things they have collected so far for that level.
    - While not having this feature would not destroy the game or diminish gameplay significantly, it is a staple feature that is observed by almost all 2D platformers and adds a dimension of extra challenge to the user if they choose to take it.
- Character selection (MEDIUM)
    - Players, at the start of the game, can choose which character they wish to play as. Each character will have different characteristics that change their playstyle, and can allow you to reach areas that may have been previously unreachable with the character you were playing before. The criteria that needs to be met, if any, before characters can be used has yet to be determined.

3. Non-functional Requirements

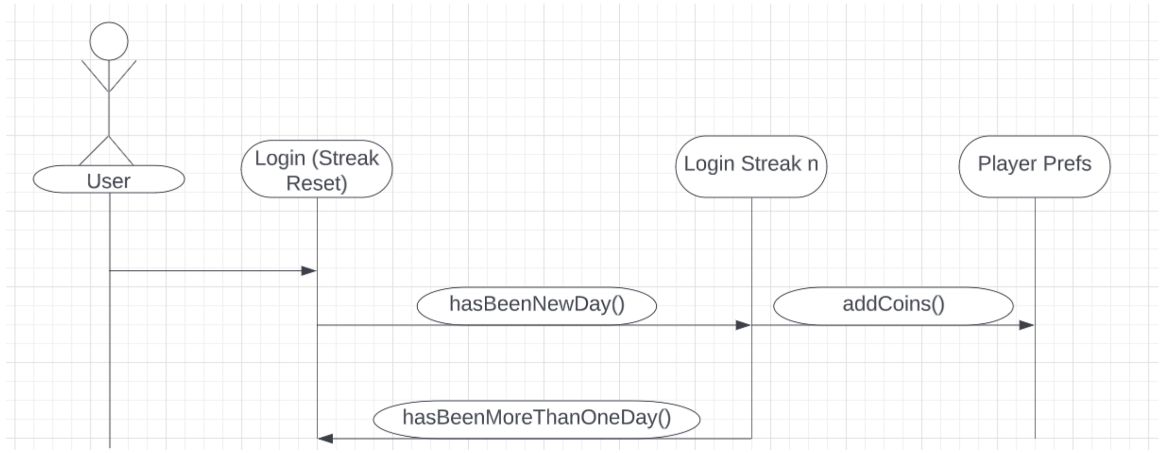
- File Organization

    - Making sure that the developers can easily navigate their way around the project directories.
- Code Readability
    - Cleaning up code and making implementing and debugging easiest for developers.
- Code documentation
    - Adding commenting throughout the project to make the logic of the code easier to follow.
- Game performance
    - Steady framerate while playing the game, no noticeable drops in response time or performance lag.

4. Use Case Diagram



5. Class Diagram and/or Sequence Diagrams



This diagram is explaining the daily reward system which we are using to give players coins which they can use on cosmetic items in the shop.

6. Operating Environment

The software will be running on the Unity Engine, which has a built-in tool to build deliverables for any of the most frequently used operating systems such as Mac, Windows, or Linux. Depending on the user's operating system, they would need to download the appropriate deliverable for their operating system. So long as the user has enough space on their drive of choice, and has one of the previously mentioned operating systems installed on their device, they should be able to use our application with no issues.

7. Assumptions and Dependencies

Unity, our main platform for developing our program, is the main dependency for our final product. If Unity were to undergo a major change that would prevent the software from running as intended, or shut down as a business, leaving us with no license to continue using the software, we would not be able to finish making our product.