Project Proposal

Software Engineering

Professor Do-Young Park

Team Members:

Emiz Intriago, Gurkiran Kaur, Wardah Aziz, Miranda Figueras

Team Leader:

Andrew Stephens

Our team will create a computer-based video game that will merge the greatest elements of several classic games with more recent games in the same genre. Our goal is to make a Super Mario Brothers-style two-dimensional platformer game. Previous platform games, such as Super Mario Brothers, were designed with continuous side-scrolling levels for the Nintendo Entertainment System. The player controls a main character in Super Mario Brothers, and the player's camera remains fixed on the character throughout the game. When collecting coins, the character is able to run, leap, and strike enemies while the player tries to get to the end of the level as quickly as possible. In addition to that, eliminating enemies raises the player's score.The Super Mario game allows the player to progress through the level while avoiding backtracking.

Our team is focusing on making a 2D platform game that is temporarily called "2D Platformer." The game will be suitable for all ages and will adhere to the PEGI 7 guidelines (with mild violence for a younger audience). We will try to improve the functionality of Super Mario Brothers by allowing the player to freely maneuver around a level. This means that the game will allow for level backtracking and traversal of both right and left. Backtracking will be a crucial aspect of the game because it will allow for the incorporation of puzzle mechanics. Within the classic platforming experience, puzzle mechanics will be nested. The player must acquire unique treasures scattered around the level in order to continue to the next section. The main collectibles, known as Keys, will be hidden behind static barriers throughout the level.

Our goal will also be to provide the user with a more dynamic gaming experience. The technology at the time constrained Super Mario Brothers' development, which walked a fine line between optimization and fidelity. We will strive to provide a more efficient gaming experience to the player as technology evolves. This includes allowing the user to choose their desired refresh rate, which will result in a better comprehensive visual experience. We will also give the user the option of choosing their ideal window dimensions or choose to display fullscreen.

The functional requirements for our game will include a 2D game engine built from the ground up. It will include both vector motion mechanics, a hitbox-based physics simulation, graphical rendering and animations, and controller input. Controlling the main character will be done via keystroke. The user will be able to control the character's movement and actions using simple keystrokes of the W, A, S, D, and Space keys, as well as other keys should more actions be implemented. The user will be able to control the character – to walk left and right throughout the level and jump between platforms and over obstacles. The items able to be picked up by the character will include Keys that are used in opening a door to complete the Level. Levels will contain static obstacles, which will prevent the character from walking in the user’s preferred directions. Some of such obstacles will damage the user. For the sake of simplicity, these will be considered hostile obstacles. It will be required that the user navigate the character around hostile obstacles. Touching hostile obstacles such as spikes would cause damage to the character which can lead to failing an attempt to completing a Level. These hostile obstacles will be placed strategically to make it more

difficult for the player to reach important collectibles. If the player is successful in their playthrough, they will proceed to another level. However, upon failure, the player will be returned to the ‘Main Menu’ and level progress will be permanently lost.

The game will load with a ‘Start Screen’. Upon pressing ‘Start’ while on that screen, the frame rate will be initialized and the ‘Main Menu’ screen will load. From the ‘Main Menu’ screen, the player will be able to start a ‘New Game’, select ‘Continue Game’, or visit an ‘Options Menu’. Selecting ‘New Game’ will cause the first Level to load by default. By choosing the ‘Continue Game’ option, the user will be brought to a ‘Level Select’ menu where they will be prompted to select all completed levels or the level they’re currently attempting. We have added the ‘Level Select’ menu so that there will be a simple avenue to increase the options for gameplay. (Our initial goal will be to achieve one complete Level for the user to play, with the ability to expand the number of levels should time allow for it.) After selecting a Level, the ‘Game Activity’ will start and the user can begin playing. If the player completes the Level, the next Level will load. If there are no more Levels to progress to, the user will be returned to the ‘Main Menu’. The ‘Options Menu’ will allow the user to select the volume for any audio played. From the ‘Main Options Menu’, the user will be able to return to the ‘Main Menu’ and continue by selecting the ‘New Game’ or ‘Level Selection Menu’.

Our system will be deployed through IntelliJ IDEA using the Java programming language. We will allow for Maven support should external libraries be required for more advanced implementations, however, the goal is to create a game completely from

scratch. Due to our game running on Java, it will be able to be run on any PC, Mac, or Linux operating system that has the Java JRE installed. Our game engine will be developed using standard Java libraries for the display and standard classes, including the Java Swing environment. Our group will be using GitHub in order to keep all code and documents available to all group members throughout game development. We will also use Discord as our primary form of communication to maintain regular asynchronous information sharing as well as to utilize the voice calls feature for scheduled project review sessions.

We expect to deliver a complete game on May 15, 2022. At this time we will have a functional .jar where a user can play the first level of our game program with all expected functionality without the program crashing. Our team also aims to create a tertiary Level Creation software as a supplement to building the Levels directly in the game environment. It should hopefully allow us to dynamically add more Levels by externally creating Level files and moving them into the game's source.

Our criteria for acceptance will be the running JAR file. All functionality of user controls and game object interactions must work as expected. A satisfying release must procure enjoyable playtest experiences from multiple test users.