**Introduction**

The software I create is simply a “time-attack” based 2D platformer game that the player is supposed to kill the boss monster within a given limited amount of time. Before starting the game, it will shortly show the credits participated in the game, and will deliver the brief story of the setting that the player will be playing in. Once the game starts, the monster ai will start attacking the player, and in the middle of the screen, “the timer” displays the time left for the player to clear the stage (the initial timer is set to 5 minutes). The player character is equipped a gun and can attack the monster ai by making its hp 0 point before the time is up (the monster has 2400 hp at the start). The targeted users for this game can be anyone who wants to be entertained and curious enough to spend their time to see what the game is about. However, the game has only one stage to clear, thus it will lead the users to lose their interests shortly after some playing time, or if it is fun for them, they will want the game to be expanded into more stages or deeper level design. Fortunately, with the advantage of developing the game on Unity engine, it is quite easy to bring more updates or changes to the game responding to future demands from the users. In addition, I am planning to use this game as a prequel of the bigger project that I will be personally working on and will guide the users to more detailed worldview of this game in the later updates.

**User Requirements Specification**

1. 1.1 It simply needs to entertain the user. By saying “simply entertains,” I mean that it should be able to kill users’ time in a least tiring way. For users to spare certain amount of their time, the game will make the user to repeat a particular process to reach a set goal to move onto next stage or clear the game, which is called level design.

1.2 However, my game has **only single stage** and its **time limit is set to 5 minutes**, thus it won’t take that long time for the users to finish the game. To resolve the issue, I designed that single **stage relatively difficult** compared to other role-playing based action games.

1. I made the game to **store the fastest clear time and the last clear time**, so the users can be competitive on clear time with themselves or the other players, also you can see how worse/better you played this time compared to the best play. In that way, I could make the simple game little more interesting and time consuming.
2. It shall intrigue the curiosity of the users about the game’s worldview. It’s very important part of the game as it will be working as **a prequel of the bigger game project following up**. The music, background design, character design, story intro, and altogether, the vibe that the game delivers needs to be mysterious and interesting.
3. To keep the game fun but least tiring, it should not take the user more than **15 – 20 trials** to clear the game.
4. Visual rendering of player character and the monster shall be clear in the scene from the background image so the player would be able to distinguish each object clearly and have no issue clearing the stage.
5. Game interface needs to be **straight forward** and **clear** to look and indicate the information about the game (for examples, monster & player’s HP, menu options, score boards, and etc..).
6. Game objects will require to response correctly and correspondingly to the user input assigned to their particular behaviors. As well as the audio source should respond properly according based on the game objects behaviors or user interface selection.
7. The game application shall properly terminate upon the user selecting ‘Exit’ menu from any game scene.

**[1. Structured Tables of User Requirements]**

|  |  |
| --- | --- |
| **Function** | Entertains the users and plays a prequel of a bigger game project coming up. |
| **Description** | The player will be told about the game worldview and given 5 minutes to fight the monster. Upon clearing the game, it will display the clear time and stores the record in the score board as the last record and the best record if it’s faster than the previous best record. |
| **Inputs** | Player character’s movement and attack control key inputs, and menu selections. |
| **Source** | A computer keyboard and a mouse. |
| **Outputs** | Graphic rendered scenes and objects, and background music and sound effect. |
| **Destination** | Computer monitor and audio outputs(headphone/speaker). |
| **Action** | As running the game file, a player will be shown the main menu, followed after the splash screen, where you can select ‘play’, ‘score’, and ‘exit’. The player selects ‘score’, he/she will see the highest score and the last score, and ‘-‘ if no data has been saved. The player selects ‘play’, he/she will be led to the game stage and given 5 minutes to drop its HP bar down to 0 to kill it. If the player fails to kill it in 5 minutes or die from the monster ai’s attack, he/she will be led to ‘Game over’ scene. If the player succeeds to kill the monster ai in 5 minutes, he/she will be led to ‘You win’ scene and it will display your clear time. Additionally, if the player clears the game in 3 minutes, he/she will be led to ‘Easter egg’ scene (a hidden scene) that foreshadows there would be future update to the game. |
| **Requires** | User input to select game scenes to pull up and to control game object. |

**[2. Behavioral Model of User Requirements]**

Run

Exit the game

Main menu

Intro Story

Splash Screen

Go to menu

Displays Scores

Reset Scores

Exit

Menu

Exit

Menu

Play

Game Over Scene

Play

Time over

Player dies

Easter Egg Scene

You win scene

Killing the monster ai

Clear time > 3min

Clear time < 3min

Score

Exit

Play

**System Requirements Specification**

1. Player Control

* Horizontal movements will be controlled by the arrow keys on the keyboard by manipulating the transform velocity of the player object. In addition, ‘z’ key input will change the player object’s x-axis of the transform leaving fading away prefab in the initial location, so it looks like it teleports along the x-axis.
* Attack will be controlled by ‘x’ key input, as the key inputs instantiate the bullet prefab travels the horizontal axis with given initial velocity.
* Vertical movements will be controlled by ‘c’ key input and ‘v’ key input, where ‘c’ key input triggers upward vector impulse on the player object as it looks like it jumps along y-axis, and ‘v’ key input works similar as ‘z’ key input but it works along y-axis downward only.
* As ‘sniper gage’ (UI shall indicate level of the gage) reaches the full, the player can input ‘f’ key to change the player game sprite to the one with stronger looking weapon and while that sprite is up, the player game object should instantiate stronger bullet prefab when ‘x’ key input is triggered.
* The player game object shall start the game with 100 HP points, and when it reaches 0 by getting attacked by the monster ai, the player has to lose its control and the game object should play its ‘death effect’ animation before the scene into the game over scene.

1. Monster AI

* Once the game scene starts, the monster ai script shall be able to track the player object’s local position and set its velocity vector to move toward the player, and keep tracking the player object’s next position to change the velocity based on it.
* Monster AI object starts the game scene with 2400 HP points, and when it reaches 0 by players attack, the game scene turns to ‘You win’ scene.
* Monster AI will have two phases during the game scene, each phase has different looking game sprite playing different animation. The monster ai starts with the first phase and once its hp points drops below 1000, it triggers transition animation going from the 1st to 2nd phase, and when the animation is over, the second phases starts.
* Each phase of the monster ai has 3-5 attack patterns that are triggered by ‘invoke’ method, which instantiate a particular method in given random time range. Attack patterns during the second phase shall deliver more difficult experience to the player.

1. User Interface

* Each menu button shall be able to load proper scene by using scene management library.
* Text, color, image, and sprites must be carefully selected and oriented in a manner that each interface stands out clearly to look organized to the player.
* User interface shall conveniently provide assistance to the player so the player can easily understand the game and manage the game application.
* Game interface such as HP bar of player/monster-ai, ‘rage gage’ of monster ai, and the timer must accurately display the proper data/information of the current game scene.

1. Audio

* All the background music and sound effect shall be managed by ‘audio manager’ script during the entire game experience.
* Some additional sound effect can be attached to the game object/prefab so it plays the audio sound as it is instantiated.

1. Graphic Rendering & Animation

* Every game sprite for background animation, player object, monster ai object, platformer, and prefabs shall look unique and neat, and need to be designed proportional to the others.
* All the game sprites shall deliver appropriate image corresponding to its object’s behavior/method calls.
* Each game sprite should be imported with appropriate compression, texture type, sprite mode, filter mode and etc.. to deliver clear image and proper size.
* Each game sprite should be sorted correctly in the sorting layer, thus they do not overlap with the others inappropriately.
* Each animation assigned to method/object calls needs to be played with appropriate playing time and should exit with appropriate exit time to display smooth and natural object movements.

**Platform Requirements Specifications**

This game will be developed on the game engine called “Unity.” Unity accompanies C# for the programming language to implement behavior/method scripts to all the game objects and the user interface. Fortunately, Unity supports the most OS that people generally play the game on such as Macintosh, Window, Linux, iOS, and Android. For the most parts, the project can be rebuilt for different platform, but if the input device differs (from computer keyboard to joystick or from the keyboard to touch screen), you will just need to modify input key setting in the project setting inside Unity. Especially this project has only few key inputs that will be used during the entire game, thus it will be almost trivial process to be worried. Similarly, difference between capacity of memory/CPU power of different devices is neglectable, since it’s offline based and using only few amount of memory storage (because of the size of the game).