# Software Requirements Specification

Two-dimensional video game

**<u>Group-4398-RR-2</u>**

Maria Sanchez

Muhammed Shiykh

Christopher Stearns

David J. Nelson

## 1. Introduction

Video gaming has become extremely prevalent in our society and players are always looking for a new game to play.  This document is an overview for a 2D sidescrolling platformer aimed at all audiences.  The game will be created in Unity and shared with all members using Unity Collab so that each member can have access to the current state of the project.  The final release should have a finite ending with a boss that tests the skill level of players that make it to the end of the game.

### 1.1. Purpose

The purpose of this application is to provide the player with a challenging experience while still being completable.  The goal is for the player to navigate a series of platforms that are on levels of variable height while simultaneously killing enemies, eventually progressing to a boss where the player wins if they beat the boss.

### 1.2. Problem statement

The main reason platformers are fun for gamers is that most rely on skill to complete levels or the entire game.  Navigating through complex areas and dodging incoming threats is the basis of what makes a platformer fun.  There is not much RNG in most platformers which makes the player's skill a heavy influence in how the game is designed.  This game aims to be challenging for players with a wide skill set which might mean hardcore players might not have as much fun with it but such is the nature of appealing to a wide audience. This project is a desktop application two-dimensional side scrolling game built on the programming language C# through Unity.  The style of the game will be Pixel art, and will feature parallax scrolling, gravity physics, player controlled horizontal and vertical movement modified by friction of the ground, and modular abilities. The objective of the game is to survive against enemies of increasing difficulty. The game will include progression for the player character, allow them to increase their capabilities, effectively simulating growth in skill and power.

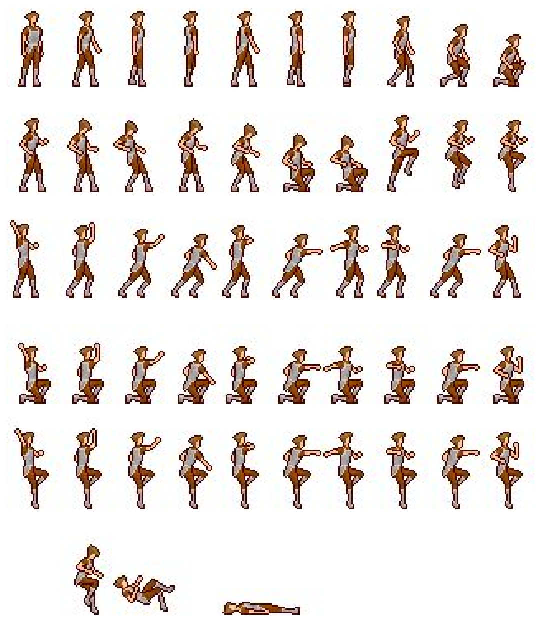
## 2. Statement of Functional requirements

### 2.1 Background

The background should have a parallax effect to create a simulated 3D environment, giving the game a bit more depth.  For a greater effect, multiple layers of the background can be rendered.  Background tiling should be implemented so that only one real piece of background is needed and the rest is rendered when the camera is near the edge of the current tile which should create a seamless effect for the player progressing through a level.

### 2.2 The Player

This is the character controlled by the person playing the game.  They should be able to attack enemies, move left and right, and jump to progress through the level.  If they player kills the boss at the end they beat the game.  The picture below features the player's sprite sheet representing him in multiple positions or movements.

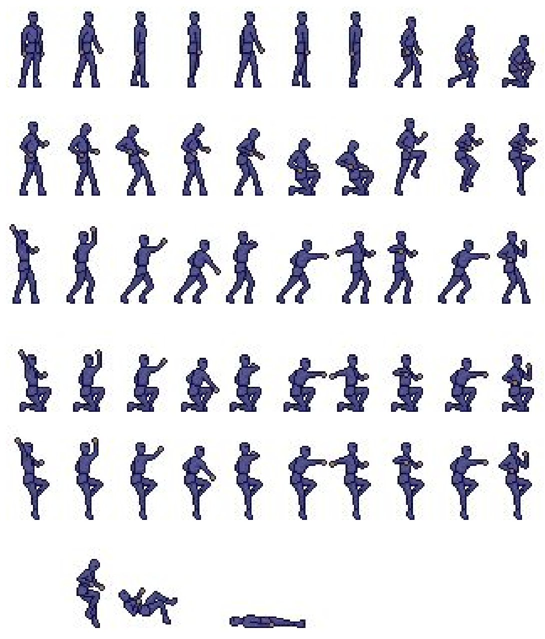


## 2.3 Platforms

The platforms should be spaced apart and at different heights so that the player has to jump between platforms to continue.  They should act as ground for the player as well as enemies.  All platforms should be accessible to the player and enemies although not all should be easy to access.  There should be difficult jumps for the player to keep them interested in the game and pose a challenge while they are progressing.

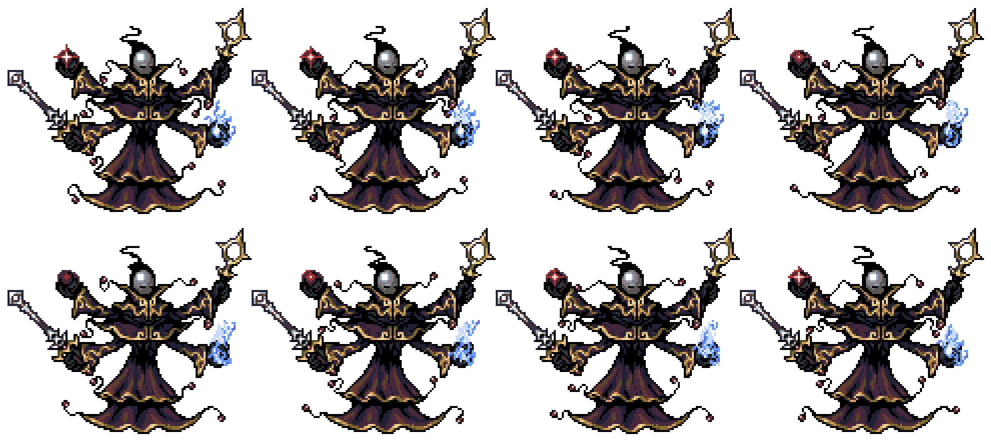
## 2.4 Enemies

The enemies the player encounters on the way to the boss should be fairly easy and predictable.  They should attempt to attack the player or knock them off the platforms.  They should also be able to be killed by the player.  Below is a picture of the enemies that will show up to hinder the player in a variety of positions and movements.



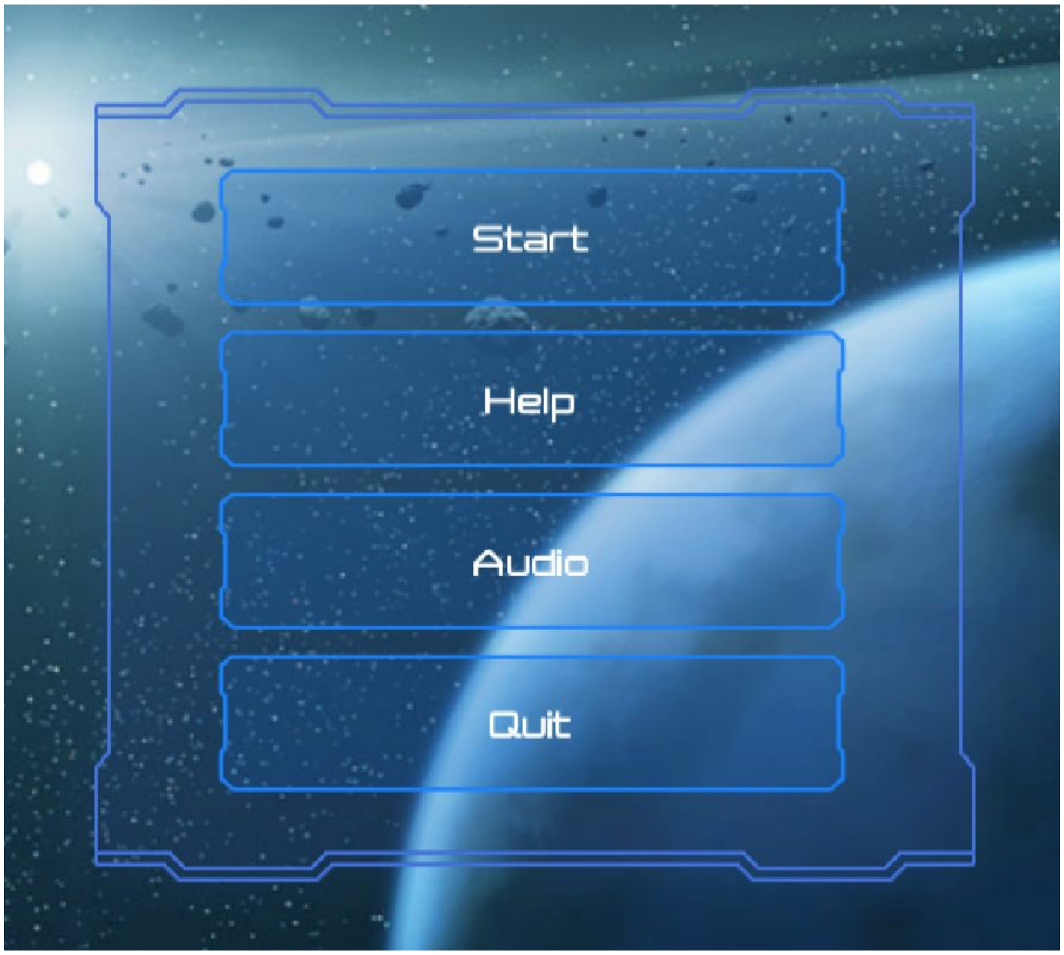
## 2.5 Boss

The boss should be at the end of the game.  It should be more difficult than the previous enemies.  To beat the game the player must beat the boss.  The boss should have a higher health pool than regular enemies as well as more attacks to create a greater challenge for the player.  When the player beats the boss they either move on to the next level or the game is over.



## 2.6 Menu

When the game is started up there should be a menu for the player to decide what they want to like starting the game or exiting.  When the game has started another menu should be accessible by pressing Escape that should give similar options such as restart or to exit the game.



## 3. Non-Functional Requirements

### 3.1 Reliability

Collision is important here and should be the most reliable part of the system. The player should be able to rely on where their character is on the screen and that when they are jumping from platform to platform, what is rendered to the screen is actually what happened. It can be pretty infuriating if what the player sees on the screen is them making a jump but the game indicates that they didn't and the player loses. The same thing can be said for any enemies that might attack the player. Proper collision is important here as it must feel like the player has been hit by an attack. If the attack hits and the enemy or projectile is nowhere near the player it can be difficult to dodge and therefore complete the game.

### 3.2 Robustness

In unforeseen circumstances, the game must be able to make decisions that steer the gameplay back into something that is expected. For example, the player should only die if they have taken too much damage or fall off a platform. To die by falling, a zone underneath the platforms should initiate the game being over, but this zone should never be able to contain platforms otherwise an unexpected death could occur.  Enemies should also be required to check if a hit is detected before any damage is done, not after an attack is made.

### 3.3 Performance

Ideally the game should run at a smooth 60 fps which is pretty standard for most PC games.  The most intensive parts of the game are most likely the boss fights but being fairly small and only rendering a select few sprites the game should have no trouble running on most PCs.

### 3.4 Maintainability

Code should be written so that it is easily maintainable and new features can be added. Possible new features may contain new enemies, different players, new levels, or new bosses.  Code should be implemented so that these features can be implemented smoothly and based off of code from a previous release.

### 3.5 Usability

An intuitive control scheme will be used.  With many platformer PC games, it is common practice for WASD to be used for movement.  A and D are typically left and right and W and S are up and down.  If W is not jump, Spacebar is typically used and can be used here.  Escape will be used for the menu with buttons that can be clicked on for different options.  For other non-intuitive controls a control menu can display which keys do what actions.

## 4. Design and Implementation Constraints

The only constraints are from using the game engine Unity and it's limitations as well as our knowledge of the engine because we are all new using it although the game should not exceed any of Unity's capabilities.

## 5. References

Character and Enemy Sprites:
https://opengameart.org/content/mv-platformer-male-32x64

Boss Sprite:
https://opengameart.org/content/bosses-and-monsters-spritesheets-ars-notoria