TOWER DEFENSE
PROGRAMMER'S MANUAL

The Tower Defense Game is a top down 2D Unity project developed by the Three Musketeers that has the player take control of a hero character, defending their castle against a horde of slime monsters. It is the goal of the player to destroy all of the monsters before reaching and being able to destroy the castle. The two ways to do this are for the player to attack the enemies themselves, using characters' own attacks, or by building archer towers to automatically attack the slimes. To update and add new features and levels to this software, a programmer needs to be aware of how this software was built.

Firstly, the software was built on the Unity Engine, so to update the code, one must first download the Unity Hub client. From there, you can set up the project with the Unity built in tools. To store past versions of the software and to collaborate with other team members seamlessly, we recommend using Github online with Github desktop to push and pull new features online to their servers. Here is a link to the Unity documentation on how to set up a Github repository for the game engine: https://docs.unity3d.com/560/Documentation/Manual/UnityCloudBuildVcsGitHub.html. Once getting this all set up, you will need to understand the code and how it functions. The game is divided into scenes, tile maps and objects. Each scene is the canvas that is used to make a part of the game. The tile map is the background of that scene, and the objects are the things that can move and interact with each other on screen. Most of these objects are put into prefabs (much like classes) so objects can be built with the same or similar constructors. The most important prefabs and objects are Player, Enemy, Tower, and Castle. All of these classes interact with each other through methods such as playerDamage() or enemyTakeDamage(). Each class is able to affect the other through collision by damaging the other. Documentation in the code lays out what every method and variable does. To make new levels or features, one must build a new scene and build on the existing objects and how they, with their scripts, affect each other.

REQUIREMENTS DOCUMENTATION

1. Introduction

1.1 Scope of the product - A single player Tower Defense game featuring a singular player character that can collect gold, build towers, and defend a castle from a swarm invading enemies.

1.2 Definitions, acronyms, and abbreviations - Here we try to clarify a differentiation between ‘user’ and ‘player’. The user is the operator reading this and playing the game. The player is within the game as the avatar (or icon) the user controls. The player has its own sprite, animations, and C# scripting.

1.3 References - The game engine used to build this game is Unity which allows for free download for student and personal use that allow for “Revenue or funding less than 100k in the last 12 months''. Unity also offers an asset store for creators to share their created sprite images and animations to be reused for other games following their “Standard Unity Asset Store EULA” license agreement. We utilized several free offerings including “Prototype Hero Demo - Pixel Art” by Sven Thole, “Slime Character” by NYKNCK, “Isometric Tower Defense Pack” by Artyom Zagorskiy, and “2D Animated Coin - 2D RPK'' by Marco Siino.

2. General Description

2.1 Product perspective - Our group collectively decided on building this game idea of a tower defense game not only in order to satisfy the school project requirement but also to advance our knowledge and skills in software development, coding in C#, working with the Unity game development engine, and most importantly have fun

making something fun to play. Other users will hopefully find enjoyment and inspiration from the game as well.

2.2 Product functions - The product will be an executable program launched by the user that loads a game consisting of a map, controllable player that can attack enemies and collect gold, enemies that damage the player and a castle at the end of the map, and towers that can be statically placed to attack enemies.

2.3 User characteristics - End users will consist of gamers interested in strategy and/or arcade type games, specifically tower defense.

2.4 General constraints - The software is a single player, offline experience that can run on Windows 10.

2.5 Assumptions and dependencies - Requirements include that the user is able to unzip and run the file that contains the software installer and all its associated documentation. As for a user that hopes to continue development of the software, Unity Hub will be required to be installed to manage and open the game's file for further editing and additions. In the event that Additional scripting is to be added, Visual Studio will also be needed (which is typically installed along with Unity) in order to open, edit, and create C# scripts.

3. Specific Requirements

Menus

1.0.0 Main Menu - The game should have a main menu scene that the application loads first containing 2 buttons: 'Play' & 'Exit'.

- 1.0.1 Pressing the 'Play' button should transfer from the MainMenu scene to the TowerDefenseGame scene.
- 1.0.2 Pressing the 'Exit' button should close the application.

1.1.0 Pause Menu - Users should be able to pause the game during gameplay by pressing the escape key on the keyboard loading a transparent menu of 3 buttons: ‘Resume’, ‘Main Menu’, and ‘Exit’.

1.1.1 Pressing the ‘Resume’ button should hide the transparent pause menu and resume time and movement within the game.

1.1.2 Pressing the ‘Main Menu’ button should switch from the TowerDefenseGame scene back to the MainMenu scene.

1.1.3 Pressing the ‘Exit’ button should close the application.

1.2.0 Game Over Menu - User should be able to finish the game by having a menu appear with win/lose description and 3 buttons: ‘Replay’, ‘Main Menu’, and ‘Exit’.

1.2.1 Losing the game by letting the castle be destroyed by enemies will show “Play was killed, try focusing on surviving more”.

1.2.2 Losing the game by letting the player be destroyed by enemies will show “Castle was destroyed, try focusing on protecting it more”.

1.2.3 Winning the game by destroying all the enemies will show “All enemies destroyed, good job defending”.

1.2.4 Pressing the ‘Replay’ button should reload the TowerDefenseGameScene resetting all game objects position and existence to their initial values.

1.2.5 Pressing the ‘Main Menu’ button should switch from the TowerDefenseGame scene back to the MainMenu scene.

1.2.6 Pressing the ‘Exit’ button should close the application.

Map

2.0.0 Using a tile map, textures are placed to build the map and world

2.0.1 Path texture and waypoints are placed for the enemies to follow

2.0.2 Foliage and lake are placed around the map to make it more natural

2.0.3 Invisible walls are placed around the map and inside the lake to keep the player from venturing where the hero cannot go

Castle

3.0.0 Castle model is placed at the end of the path, and it is the goal for the player to defend and the enemies to attack

3.0.1 Castle is damaged a specified amount when collided by the enemy, enemy model is then destroyed

Player

4.0.0 Player should display the hero sprite on screen over the background layer, and the player should be given control over it

- 4.0.1 Using “WASD”, the player can move the character around the map and trigger the proper animations
- 4.0.2 When colliding with gold, the player should add that gold to his pouch to be used to build towers
- 4.0.3 By pressing the Left Mouse Button, the player is able to attack enemies in his front arc, doing a set amount of damage

Enemy

5.0.0 Enemy model moves over a specified path across the map to end at the castle and damage it

- 5.0.1 Enemy follows preset path of waypoints on path
- 5.0.2 Enemy stops and attacks the player when colliding with him, doing a set amount of damage
- 5.0.3 Enemy animates while walking and spins while attacking the player
- 5.0.4 An enemy spawner at the start of their path causes new enemies to emerge, allowing for multiple enemies

Tower

6.0.0 Tower model is placed on the map in front of the path to be able to attack enemies

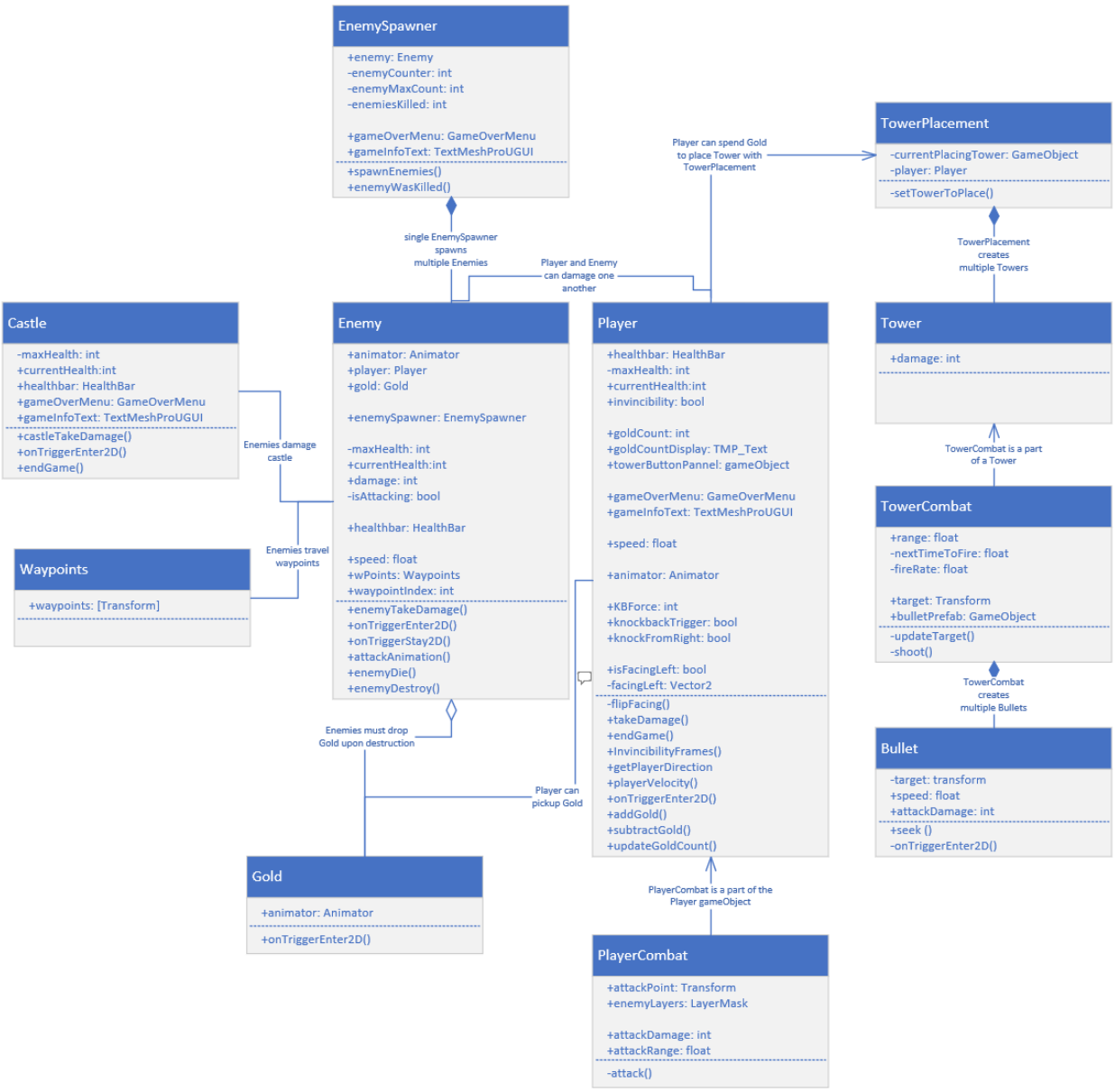
- 6.0.1 Tower is able to create a bullet object that seeks and moves toward the nearest enemy model to the tower
- 6.0.2 Tower bullets do a certain amount of damage to enemies when colliding with them
- 6.0.3 The Tower is made into a class, allowing multiple towers to be made at a time and attack enemies simultaneously
- 6.0.4 The special cursor was built a tool to help the player easily place towers on the map where they decide

DESIGN DOCUMENTATION

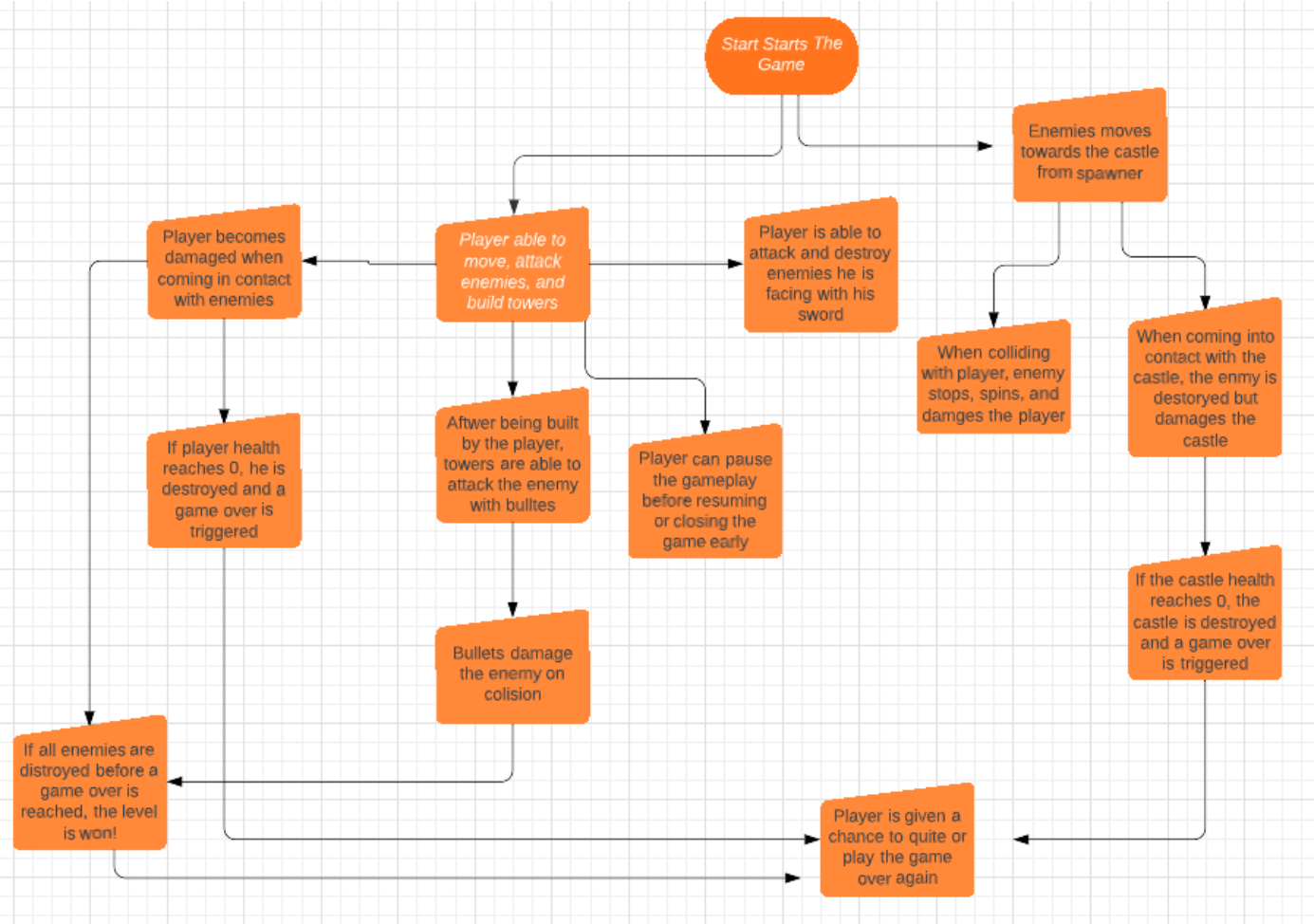
The game runs on state and decisional “catch” logic. All characters in the game, like Enemies and Towers, act in a predetermined way, and it is the action of the player that changes the state of the program. The basic design followed by the game is as follows:

- Enemies move on their predetermined path to attack the castle, ending their journey when either killed by a tower or the player or when attacking the castle.
- Towers can be placed to automatically attack members of the Enemy class as they move across.
- The Player starts off near the castle, and they can choose how the game continues forward
    - The player can affect the game by attacking enemies with his sword
    - The player can collect gold, and once enough is made, build a tower to automatically attack the enemies
    - Move in front of the Enemies to slow them, but take damage from their attack
- The game can conclude based on these conditions
    - The player and/or towers successfully defend the tower by destroying all the enemy objects
    - Too many enemies collide with the castle, destroying it and losing the game for the player
    - The player can take too much damage from enemy attacks and cause a game over due to player death

UML Class Diagram



Architecture Diagram for Object and Player Interactions



Psuedocode Documentation

Player Movement:

//Defines the logic of how player input will affect their character's movement

Default player state: standing still, awaiting input from player

If (player holds "W"):

    Character Model with transform a certain amount up

If (player holds "S"):

Character Model with transform a certain amount down

If (player holds “A”):

Character Model with transform a certain amount left

If (player holds “D”):

Character Model with transform a certain amount right

//All inputs are put into district if statements, since multiple inputs can be put in at once to move the character diagonally

Tower Attacks:

//Defines the logic of how the tower detects and attacks the enemies

Tower object sends out a scanner to detect objects within its attack radius

If(object is Enemy):

If(Distance to object < other Enemies):

Tower.attacksObject

attacksObject()

Tower calculates the position of the object

Tower creates a bullet object and passes the position of object to the bullet script Bullet

Bullet seeks the enemy object and moves towards it

If(bullet collides with enemy)

Enemy takes damage

Bullet is Destroyed

If(bullet.distance from tower > attackRadius) => Destroy Bullet

Enemy Movement:

//Defines how enemies move across the path

An array of waypoints in set into an array of points on the map

The array of points is passed to the Enemy Object

Enemy starts to move towards the first point in the array

if(Enemy collides with point)

Active waypoint is set to the next point in the array

Enemy begins movement towards that point

if(Enemy collides with the last point in the array) //collision with the castle

Enemy damages castle

Enemy is destroyed

Player Attack:

//Defines the logic for the player sword attack

If(leftClick is pressed)

Player.Attack

Player.attack

A circle is drawn in front of the player //sword hitbox

If(hitbox collides with an Enemy Object)

Enemy is added to an array of HitEnemies

HitEnemies is passed to a damageEnemies method

damageEnemies

Foreach (enemy in hitEnemies)

Damage enemy the attack damage value

SOURCE CODE

The source code for this project consists of C# scripts that are attached to various game objects within Unity to have game objects function and react as desired. Consequently there are 17 separate C# scripts to handle all the moving parts within the game ranging in length and complexity.

In an effort to keep things organized and not overflow this paper we have elected to keep a dedicated folder in our submission called “SourceCode”. They will contain the following code, annotated with comments, and can be opened with notepad for easy viewing in a txt format:

- Bullet.cs
- Camera.cs
- Castle.cs
- CustomCursor.cs
- Enemy.cs
- EnemySpawner.cs
- GameOverMenu.cs
- Gold.cs
- HealthBar.cs
- MainMenu.cs
- PauseMenu.cs
- Player.cs
- PlayerCombat.cs
- Tower.cs
- TowerCombat.cs
- TowerPlacement.cs
- Waypoints.cs

The C# scripts can also be viewed at this public Github repository, along with release notes, READ ME, and past versions: https://github.com/Dmass4/ThreeMusketeers

TEST PLAN & TEST RESULTS

White Box Testing (Traceability of Source Code)

Test Case #	Requirement/ Source Code Involved	Rationale	Input(s)	Expected Output	Pass /Fail
101	1.1.0 1.1.1 1.1.2 1.1.3/ PauseMenu.cs	The pause menu should suspend gameplay, and give the player options on how to continue	Escape button followed by choosing the appropriate button	The escape button and all the options held within function as expected, taking the player to their chosen point	Pass
102	1.2.1 1.2.2 1.2.3/ GameOverMenu.cs	Game Over Menu should appear with winning/losing conditions description	Player destroyed, Castle destroyed, Or all Enemies destroyed	Descriptive text appears on menu informing player of win or loss situation	Pass
103	3.0.1/ Castle.cs	Castle takes damage from enemies, eventually being destroyed	Enemies collide with Castle	Castle's health values decrease until 0, at which point castle is destroyed	Pass
104	4.0.1/ Player.cs	Using character speed values, the player model should animate to demonstrate the character starting and stopping running	Player moving their the character and stopping movement	Player changes from running to idle animation, based on the character's speed ie start run to run at certain speed ups and run stop to idle when slowed down	Pass
105	4.0.2/ Player.cs	Player should gain coins	Player collides with Gold on map	Player's goldCount increases	Pass
106	4.0.3/ Player.cs	The player, enemies, and castle all take damage when coming under attack	Various methods of attack caused by differing collisions	Player and towers can damage enemies with their attack. Enemies can damage the player and castle with their attacks	Pass
107	5.0.1/ Enemy.cs	Enemy follows set path of waypoints on	Invisible waypoints coded	The enemy follows those waypoints, one to another, until reaching the	Pass

		map	into the level	tower or being destroyed	
108	5.0.2/ Enemy.cs	Enemy damages Player	Enemy collides with Player	Enemy freezes movement to attack and damage player, decreasing Player's health	Pass
109	5.0.3/ Enemy.cs	Enemy does spin animation when attacking	Enemy collides with player and is attacking	Enemy animates spin attack while colliding with Player	Pass
110	5.0.4/ EnemySpawner.cs	Enemies are spawned on map	Time (3.5 seconds) pass	New enemies are spawned until a set maxCount is reached	Pass
111	6.0.1/ TowerCombat.cs	Towers able to scan the area around them to target and attack the closest enemy model to them	Enemies enter a towers attack range	Tower will attack the nearest enemy to them, and they will move to a new target when that enemy is no longer in range, is dead, or a new one becomes closer	Pass
112	6.0.2/ Bullet.cs	Bullets created by TowerCombat.cs damage enemies	Bullet collides with enemies	Enemy health is reduced	Pass
113	6.0.3/ TowerPlacement.cs	Multiple towers can be created in the world	Player left mouse clicks on "Click here to build tower" button	Tower created as a new gameObject in Unity	Pass
114	6.0.4/ TowerPlacement.cs	When user clicks to build tower, cursor turns into image of Tower to visualize action of placement	Player left mouse clicks on "Click here to build tower" button	User's cursor turns into sprite image of Tower	Pass

Black Box Testing

(Traceability of Requirements)

Test	Requirement	Rationale	Input(s)	Expected Output	Pass

Case #	Tested				/Fail
201	1.0.1 1.0.2	Main Menu buttons should allow player to enter/exit the game application	Left mouse clicking the 'Play' and 'Quit' buttons	Play button changes from Menu scene to TowerDefenseGame Scene. Quit button closes application.	Pass
202	1.1.0	User should be able to pause game and decide to resume, exit, or return to main menu	Escape button during gameplay	Pause menu appears with transparency to still show the player the game frozen in the background. Resume, main menu, and quit buttons appear.	Pass
203	1.1.1 1.1.2 1.1.3	Pause Menu buttons should allow user to resume game, return to main menu, and exit the application	Left Mouse clicking the 'Resume', 'Main Menu', and 'Quit' Buttons	Resume button, unfreezes time/movement within the game. The main menu button changes the scene back to the main menu. Quit button closes application.	Pass
204	1.2.4 1.2.5 1.2.6	Game Over Menu should allow user to Replay, return to main menu, or quit the application	Left mouse clicking the 'Replay', 'Main Menu', and 'Quit' buttons	Replay button reloads the scene of the game, resetting everything back to initial conditions/values. Main menu button changes the scene to the main menu. Quit button closes application.	Pass
205	6.0.4	User should have a "Tower Placement" button to click when enough gold is collected which can be pressed to drag a tower into the game	Left Mouse clicking the top left on-screen UI when image/button appears after collecting 30 gold	User's cursor turns into an image of a tower that can be moved about the map. Clickin the left mouse button then instantiates (Creates the game object and all its associated properties) at the user's cursor's location permanently.	Pass
206	4.0.1	User's player character should be able to move around the map 2 dimensionally	WASD movement or arrow keys	User's player character should move up, down, left, and right corresponding to user direction of WASD/arrow keys	Pass
207	4.0.3	User's player character should be able to damage enemies via melee attack	Left Mouse click	Enemies within proximity/direction of Player receive damage and their health bar decreases.	Pass
208	2.0.3	Player is in bounds within the map and	WASD or arrow keys	Player reaches edge of water or map and can no longer keep moving	Pass

		cannot walk into areas such as water			

LIST OF KNOWN BUGS AND ISSUES

- Animation transitions on the hero model will not change in time on certain occasions (mostly when the player changes movement inputs very quickly). This does not affect gameplay, but will sometimes result in the character being stopped but continuing to have the running animation for a half second, or move while in idle for a short period.

- Player does not animate properly when attacking, making the attack range of the sword vague and unsatisfying for the player. Work is being done to fix both animation issues.

- In an effort to prevent the player taking too much damage or being knockbacked too much by multiple enemies, when the player takes damage, it receives a brief second of invincibility. Thus the player may not seem to be decreasing in health or being affected, this is intentional and was designed for future balancing in mind. Given the limited set of enemies though, it doesn't appear as valuable at this stage.