README Submission for Orbital Project

Project: A Hero's Tale

INDEX

Proposed Level of Achievement: Apollo 11

Motivation

As students who enjoy playing games to relax, we are interested in learning what goes behind making one of those games and desire to make one ourselves for experience and learning. As such, we believe a 2D-platformer game is a good balance of being a challenging enough project to be proud of, but at the same time manageable enough for students like us with a lack of experience.

Additionally, when deciding on the type of project we wanted to embark on, we looked for an idea that not only hones our programming capabilities, but also other aspects like creativity, critical thinking, problem solving and possibly even some Math. As such, we feel that making a game is a perfect project to work on.

With web-based flash games being a big part of our childhood growing up, we wish to pay homage and make one of our own that hopefully would allow others to spend hours having fun playing it.

Lastly, most 2D side-scrollers are pretty much contained in that very genre of platforming. Therefore, we wanted to explore the idea of a 2D side-scrolling platformer with elements from other genres of games such as puzzle games, movement (parkour) games and role-playing games (RPGs).

Aim

We hope to be able to create an appealing and enjoyable 2D platformer game with a variety of chapters/levels of different types of missions to complete, with a linear story to go along with the gameplay.

User Stories

1. Players of the game will be able to play through different levels with increasing degree of difficulty as the story of the game progresses to test the player's hand-to-eye coordination skills.

2. Players of the game will be able to solve challenging puzzles in certain levels of the game to test the player's critical thinking and analysing skills.

3. Players of the game can experience a sense of achievement or competition as they try to beat their own personal best time, or compare with their friends to get a better time.

4. As a student stressing over exams, I empathise and want to make a platform where other students can take a break and relax.

Proposed Features (What we intend to produce)

In-Game features:
Classes the player can select to play as for the game which will impact what type of skills their character has.

Missions/Levels which will be the main challenge of the game, making up the majority of the gameplay available to the player. As the game progresses even deeper, the difficulty of the missions the player has to complete gets tougher. The missions will be a mix of 2D platforming levels and puzzle solving challenges.

In-Game Shop (TENTATIVE) to allow players to utilise in-game currency they earn to purchase items and skills that will help upgrade their character.

Game Design features:
Audio for the game, inclusive of background music, animation sound cues that sync up to certain actions and more.

Model designs which includes that of the main character the player plays as, recurring and side characters the player will encounter throughout the game and enemies which consist of small mobs to end level bosses the player will fight against.

Background designs which include varying backgrounds for different levels to indicate different settings, textbox pop ups for dialogue between characters and more.

Level concepts and designs which constitute what the aim of each level is and what the level will look like.

Storyline which is the main motive for the main character of the game and how the journey unfolds.

Completed by the end of Milestone 1:

1. Researched on the aspects needed to make a complete game.
    a. User interface
    b. Gameplay
    c. Functionality
2. Learned required skills needed to start the project. (i.e Unity, C#, etc)
3. Finalised the general theme of the game and figured out the storyline (subject to changes as the project goes along).
4. Designed the Warrior class character along with some of its animation (As of Milestone I, we have added the Idle animation with more animations to come).
5. Designed and programmed tutorial level inclusive of the dialogue and the layout of the level.
6. Designed and programmed a start menu for the game.
7. Uploaded the first version of the game into a cloud using Unity Collaborate for version control / source code control, such that changes will be published, and if need be, the game can be rolled back (reverted) to earlier versions.

Completed by end of Milestone 2:

1. Complete most of the character designs for the game, for example some characters, enemy mobs.
2. Programmed the auto-scrolling and parallax-scrolling background feature.
3. Programmed enemies into the game.
4. Programmed the feature of interactables in the game.
5. Programmed the health system in the game.
6. Programmed a basic combat system for the game.
7. Programmed “portal” to move to the next level.

Completed by end of Milestone 3:

1. Complete World 1: Level 1 and 2 and World 2: Level 1 and 2.
2. Redesign of the Mage and the Mage animations.
3. Design and animation of the enemies for World 1 and World 2 (both the common monsters and the boss).
4. Addition and programming of the Archer and Mage characters along with the six different enemy monsters for World 1 and 2 into the game.
5. Addition and programming of the Hero Select Menu.
6. Addition and programming of a Game Over screen.
7. Addition and programming of coins.
8. Addition and programming of the in-game shop feature and sprites.
9. Addition and programming of the in-game shop items and sprites.
10. Improvements on the movement and combat system (ie. Enemies will get knockback when the player attacks them), fixing any major bugs we could find from Milestone 2.
11. Addition and programming of the checkpoint feature.
12. Addition/Improvement and programming of other minor features like invisible walls and portals.

To be completed prior to end of submission:

1. Complete the remaining missions/levels and gameplay design.
2. Addition of music and sound effects into each level and scene.

For the month of May we focused mostly on brainstorming the ideas and features of the game and learning the required skills for implementation and designing of a game. We also did some early coding/designing for the game.

For the month of June we focused mostly on designing the game and programming of the most key features to us the game will need.

For the month of July we focused mostly on the execution of the level design and coding of multiple major and minor in-game features to make the game more expansive and complete.

Project Flow

Scope of Game/Background and Vision

A Hero's Tale will be a PC side-scroller 2D platformer game with influences of puzzle-solving set akin to games like Super Mario Bros. and Super Meat Boy.

Our game seeks to be an immersive game experience for teenagers who are interested in 2D-platformer fighting arcade games.

The game will be set in a fictitious world whereby the kingdom has been overtaken by "THE DEMON" sealed away in an alternate dimension. Following a forgotten prophecy, play as the protagonist in this game as he seeks out the hidden-away ancient artifacts which are required to reopen the portal to the alternate dimension "THE DEMON" is in to defeat him for good!

Game Flow

The goal for the game is to feature 3-4 different "world"/settings, whereby each setting will have approximately 2 levels with different enemy mobs. Each level mainly will require the player to complete a platformer stage, defeating enemy mobs to reach the end. Some levels will require the player to complete a sidequest or a puzzle in order to move on to the next stage. At the end of each world (the 3rd level), there will be one major boss fight which is much more difficult than the normal enemies found from the levels. After defeating the boss, the player will be rewarded with one of the ancient medallions to open the portal. Upon collecting all the ancient medallions, the player will be able to open the portal to meet "THE DEMON". The game will be completed after the player has successfully defeated "THE DEMON".

User Interface

Start Game Menu:

Upon entering the game, the player will be met with the Start Game Menu with the following options:

1) Start button that leads to the game

2) Quit button that leads to the end of the game

Game Scene:

1) Movement enabled controls

2) Combat system with built in visual health bar/health system

3) Interactable objects and characters

4) Auto-scrolling and Parallax-scrolling background

5) Invisible Walls at points of the game to prevent player from progressing prior to completing a task

6) Portals to enable the player to move from one level to the next (for transition purposes)

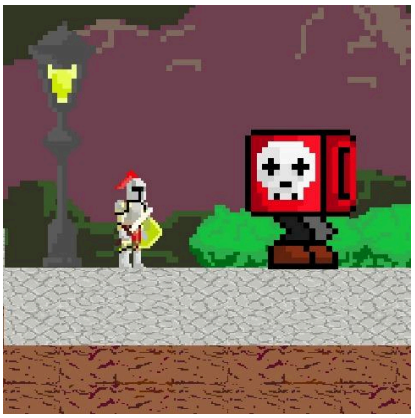
Game Over Scene:

When the character runs out of health or should they fall out of the level, the Game Over Menu will be presented to the player with the following options:

1) Option to restart the level (player starts from the beginning again) at no cost

2) Option to load the character back to the latest checkpoint the player interacted with at a cost

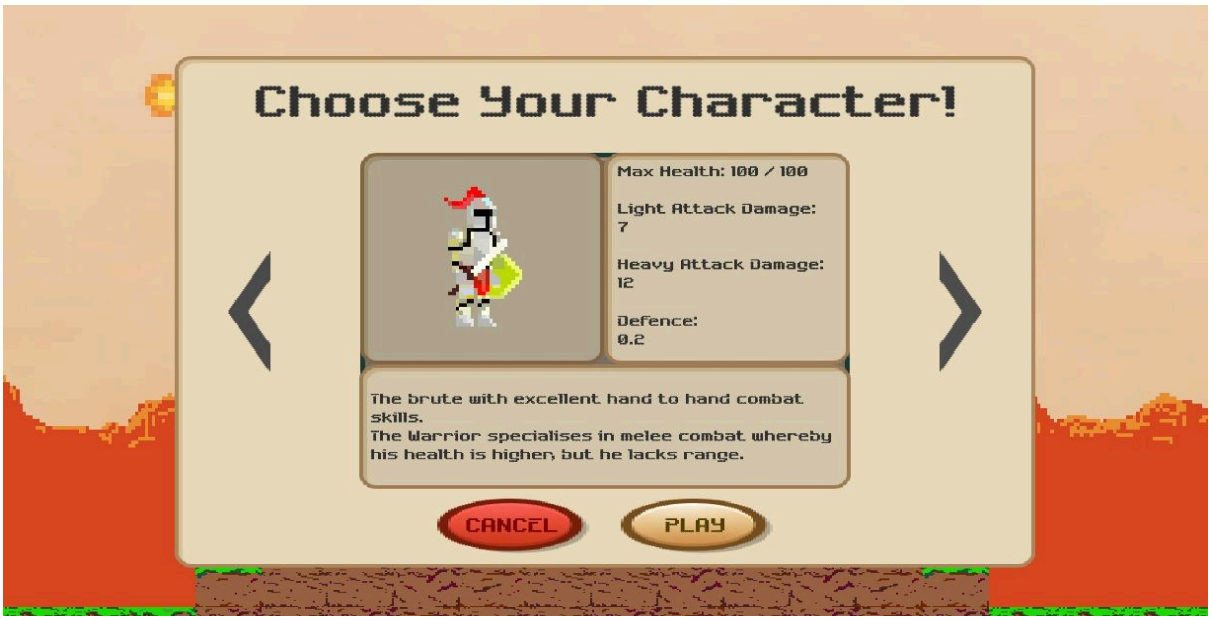
3) Option to quit the game which sends the player back to the Start Game Menu

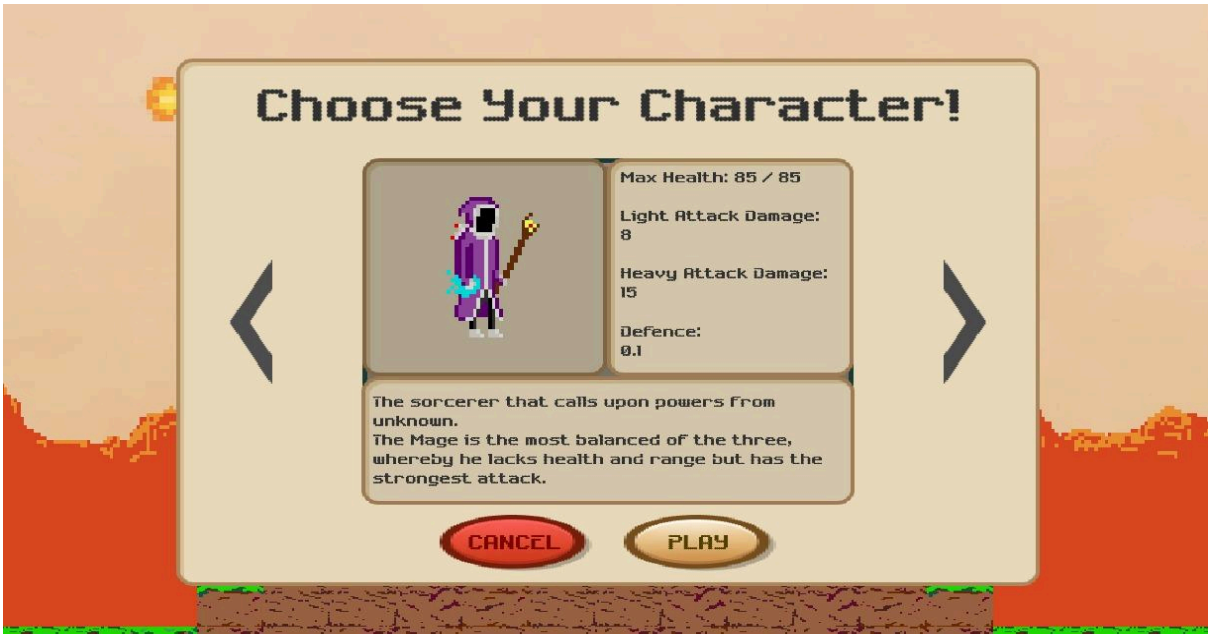


With reference to the checkpoint option in the Game Over Menu, checkpoint spawns are placed in specific parts of the levels to enable players to have an easier gameplay so they need not constantly restart the level should they lose. However, the player has to decide if they wish to utilise their coins to reload to the latest checkpoint or save them for other uses. Should the player not have the required amount of coins to respawn at the nearest checkpoint, the Load Checkpoint option in the Game Over Menu will not be selectable.

In-Game Features

Hero Select Menu:





At the start of the game, the player will be presented with an option to choose to play one of three classes, Warrior, Archer or Mage. Each class has their strengths and weaknesses which can be found further down in the README under the Characters, Enemies and Objects section. This is a crucial decision for the player to make as they will play as the class they choose as for the duration of the entire game.

The Hero Select Menu will present the player with information on the health, light attack damage, heavy attack damage and defence of each vanilla class (with no additional buffs), along with what each class is stronger and weaker in compared to the two other classes.

Players can use their mouse to navigate the Hero Select Menu to toggle between different classes or to select the Hero of their choice.

Coins:

Coins can be found in every level of the map and the total amount of coins a player has at any time will be shown on the top left of the screen for the player's perusal.

Coins are the main in-game currency whereby the player can utilise it to purchase items from the in-game shop or to help them respawn at a prior checkpoint to make gameplay easier should the player die in any point of the World.

Coins can be found placed throughout every level.

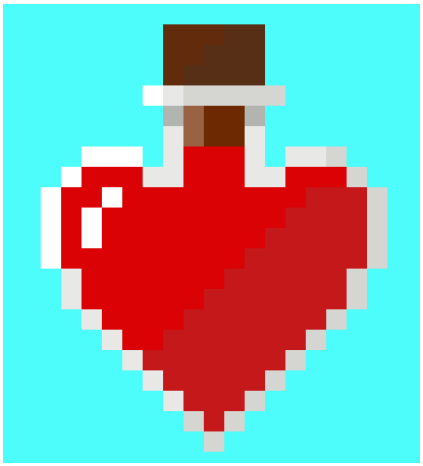
Shop:

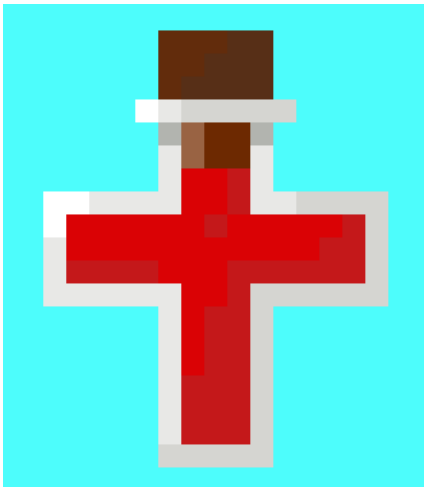
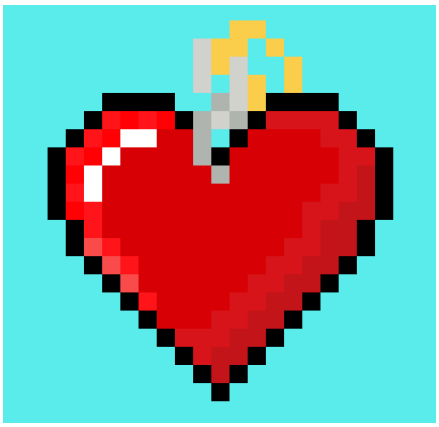
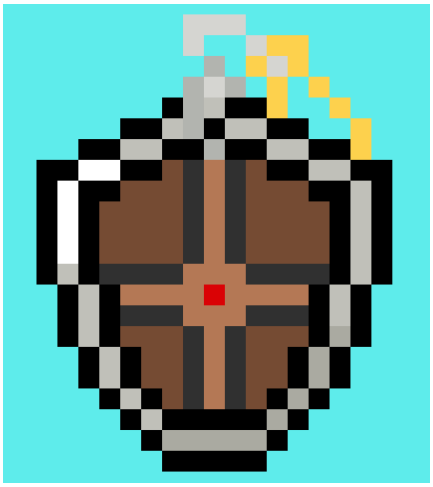
There will be one interactable Shop located at a specific position in every level for the player to purchase items for their perusal.

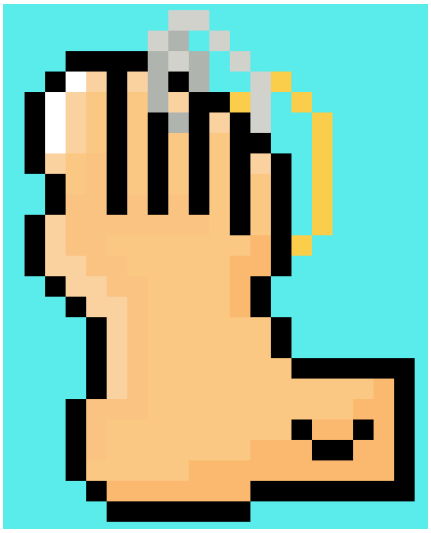
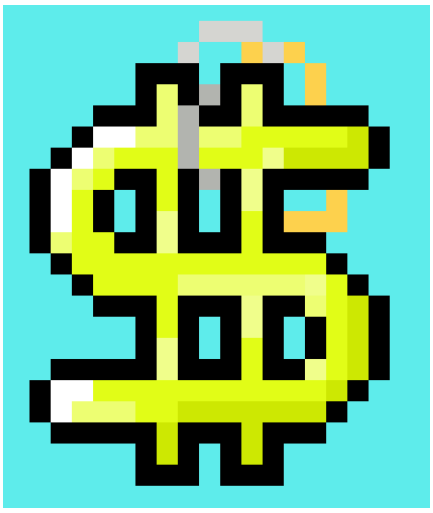
Interacting with the Shop will bring up the Shop Menu where all the shop items will be purchasable. Players can opt to purchase any item multiple times. Should an item not be purchasable due to the amount of coins the player has at the time they interact with the shop, the Shop Menu will prevent the player from buying the item. The cost and a short description of each item is also reflected in the Shop Menu for the players to decide if they wish to purchase the item.

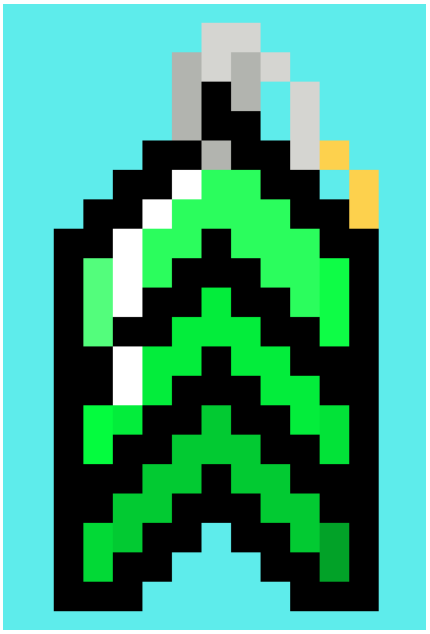
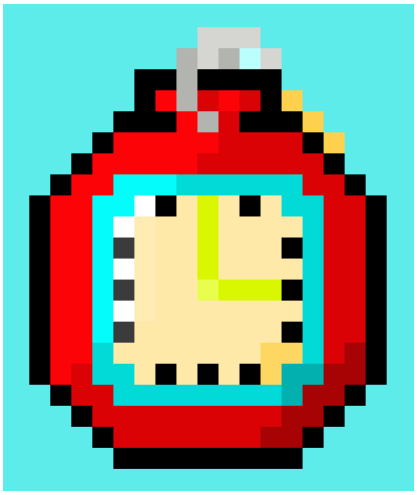
Shop Items:

The following items will be present in the Shop for the player to purchase, along with what each item does:

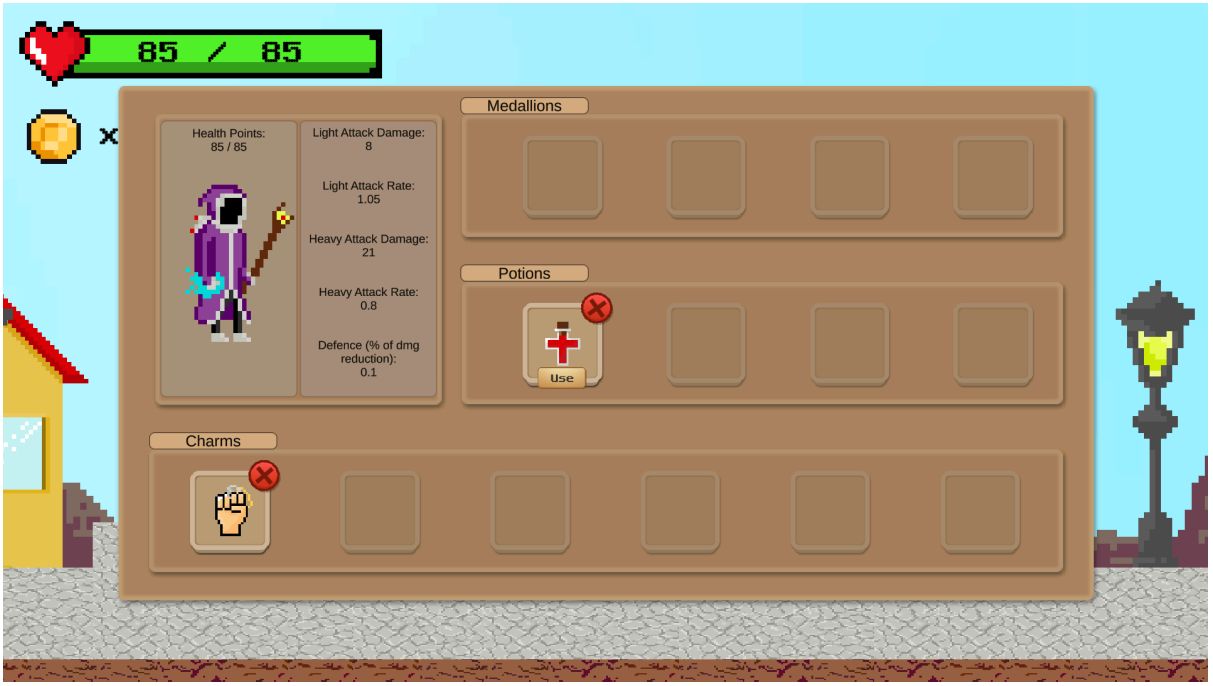
	<u>Large Health Potion</u> Cost: 20 Buff: Heals 50% of the current max character health

	<u>Small Health Potion</u> Cost: 10 Buff: Heals 25% of the current max character health
	<u>Health Charm</u> Cost: 35 Buff: Increases the current max health of the character by 33% Stackable: Yes
	<u>Defence Charm</u> Cost: 25 Buff: Reduces damage taken from enemies by 20% Stackable: Yes to a certain extent as we want to prevent the scenario where the player takes only 1 damage per hit from the normal enemy minions

	<u>Light Attack Charm</u> Cost: 30 Buff: Increases the current Light Attack damage done by the character by 40% Stackable: Yes
	<u>Heavy Attack Charm</u> Cost: 35 Buff: Increases the current Heavy Attack damage done by the character by 40% Stackable: Yes
	<u>Money Charm</u> Cost: 200 Buff: Makes every coin the character picks up worth twice the value Stackable: No as we want to prevent the character from getting too much coins so easily

	Useless Charm Cost: 0 Buff: Does nothing
	Light Attack Speed Charm Cost: 30 Buff: Increases the current Light Attack speed taken by the character by 40% Stackable: Yes
	Heavy Attack Speed Charm Cost: 35 Buff: Increases the current Heavy Attack speed taken by the character by 40% Stackable: Yes

Inventory Menu:



The game has an Inventory feature. When the Inventory Menu is opened up, the player can see the current stats the character has, along with the current Medallions the player has collected, and the Potions and Charms the character is holding.

The inventory enables the character to hold up to 4 different Potions and 6 different Charms. The player can utilise the Potions directly from the Inventory Menu by clicking on the Potion and hitting the Use button. Similarly, the player can opt to throw away a certain Potion and Charm they currently have by clicking on the specific Potion and/or Charm and hitting the X button.

Health and Combat System:

The game features a health and combat system. Present at all times is the character's current and maximum health at the top left of the screen. The character will have their own health, heavy attack, light attack and defence stat, which can be changed with Potions and Charms.

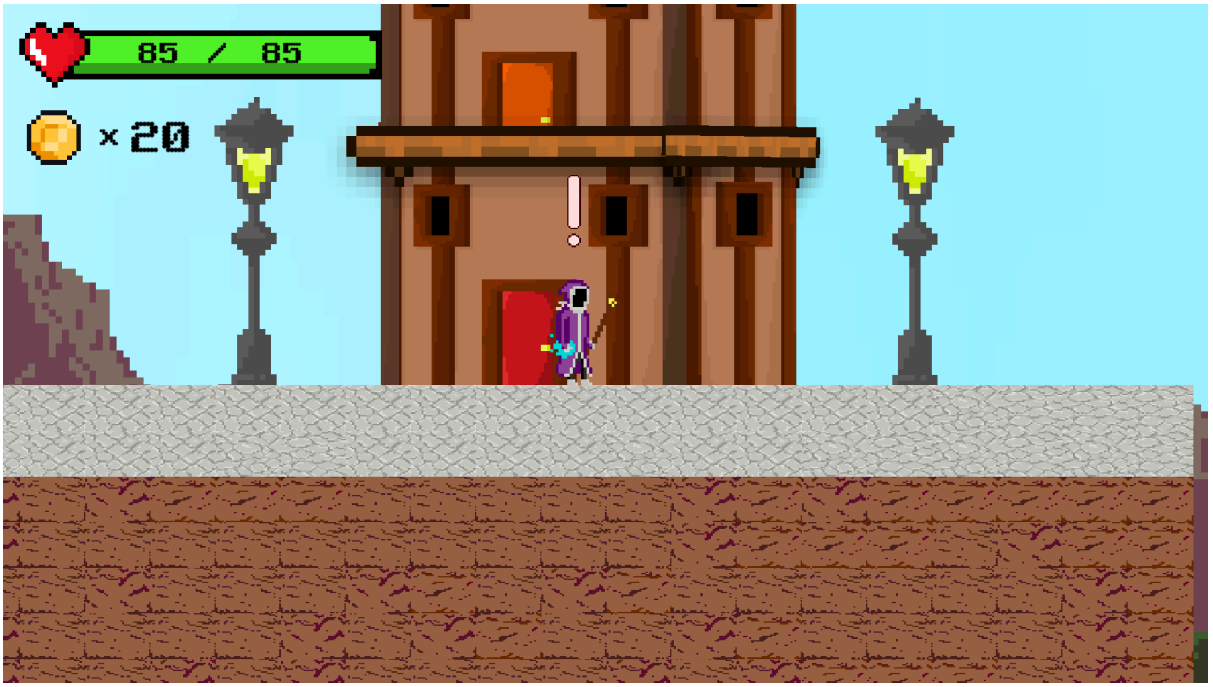
All the enemies in the game will also have their own individual health and damage stat, wherein enemies of the same kind will have the same stats, whereas enemies of different kind will have different stats. When the player nears/attacks an enemy, their health bar will be shown in the top right of the screen. Enemies can be found in multiple places in the level.

The health system works wherein should the character or the enemy health drop to 0, they will die. For the character, the Game Over Menu will be invoked whereas for the enemies, their death animation will be played.

The combat system works on the fact that different attacks of the players have different values. The light attack will do lower damage but it takes less time to execute, whereas the heavy attack does higher damage but it will take more time to execute. As such, it presents the player with the option to decide what is the best move to make when engaging in combat with the enemy. When the character attacks the enemy, damage will be done to the enemy which will deplete their health and vice versa. Additionally, when attacked, the player will be knocked back a short distance to prevent repeated damage taken. The hurt animation for the character and enemy will be played should they take damage to visually allow the player to be aware of the current state of the battle.

We decided to utilise the keyboards for the movements of the character, whereas the attacks utilise the mouse. This to use felt that the gaming experience would feel smoother and more comfortable.

Interactable Objects:



The game also features an Interactable feature wherein the player can interact with innumerable objects or characters.

When the character is able to interact with a certain object, a "!" icon will be shown above the player. Doing so will trigger an action be it a pop up text box/dialogue or items obtained, corresponding to the object interacted. Examples of interactable items currently in the game include chests, NPCs and some background art.

Settings:


PAUSED
RESUME
SETTINGS
QUIT


PAUSED
AUDIO
VIDEO


85 / 85
x 25
PAUSED
Master
100
SFX
100
MUSIC
100


85 / 85
x 25
PAUSED
1920 x 1080
FULLSCREEN
GRAPHICS
ULTRA

The game also has a settings menu which can be accessed and navigated using the mouse.

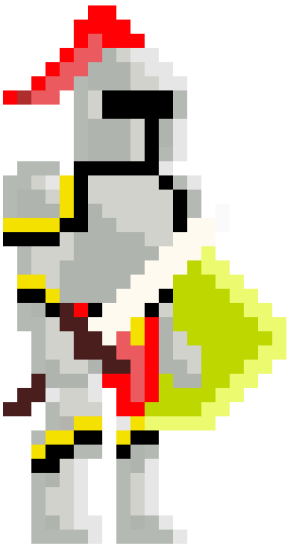
The player can adjust the audio and video settings of the game to their liking. For the audio portion, the player can adjust the overall Master volume, or individually the SFX and Music volume. For the video portion, the player can adjust what size they wish to play the game by and the graphics quality of the game. However, being a 2D game, there is not much of a difference between the different graphics qualities.

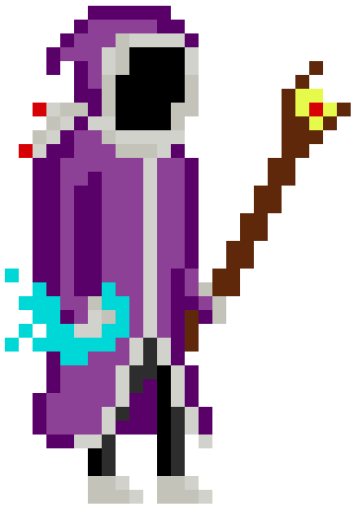
Audio:

The game also features some audio to make the game feel better to play. Simple sound effects have been binded to actions like jumping, clicking of buttons, etc. Background music has also been added to portions of the game.

Characters, Enemies and Objects

All sprites are hand-drawn in GraphicsGale and have fully animated idle, movement (run, jump, crouch), attack (light attack, heavy attack), hurt and death animations. Each hero will have different aspects they are better at. The player will get to choose one of three different characters they wish to play as for the whole game at the start.

	<u>Warrior</u> Health Points: 100 Light Attack Damage: 7 Light Attack Rate: 2 Heavy Attack Damage: 12 Heavy Attack Rate: 1.2 Defence: 0.2 Others: Knockback from attack enabled The brute with excellent hand to hand combat skills. The Warrior specialises in melee combat whereby his health is higher, but he lacks range.
	<u>Archer</u> Health Points: 70 Light Attack Damage: 5 Light Attack Rate: 1.05 Heavy Attack Damage: 10 Heavy Attack Rate: 0.8 Defence: 0.15 Others: Knockback from attack not enabled The sharpshooter that can’t miss. The Archer specialises in long range combat whereby his health is lower, but his range is the largest.

	<u>Mage</u> Health Points: 85 Light Attack Damage: 8 Light Attack Rate: 0.95 Heavy Attack Damage: 15 Heavy Attack Rate: 0.7 Defence: 0.1 Others: Knockback from attack not enabled The sorcerer that calls upon powers from unknown. The Mage is the most balanced of the three, whereby he lacks health and range but has the strongest attack.

There will also be Non-Playable Characters (NPCs) present to help progress the story. The NPCs minimally will have their own idle animations.

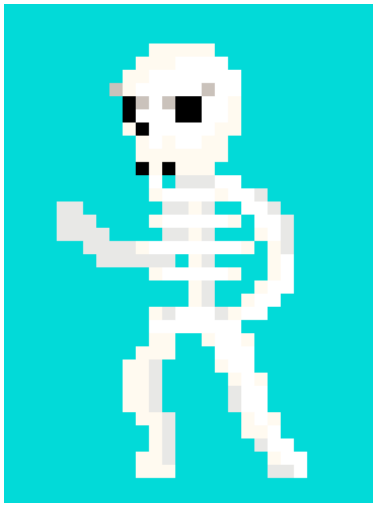
Non-Playable Characters	
	<u>Exech, The Oracle</u> The guide of the story

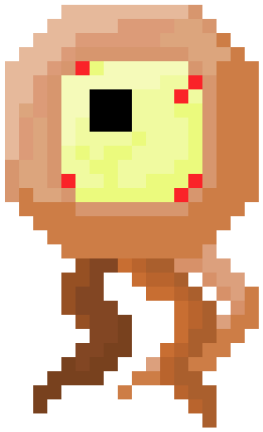
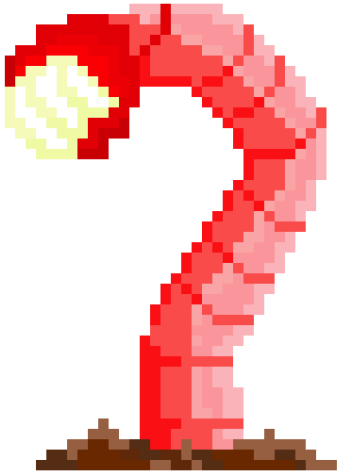
	<u>Shop Merchant</u> The owner of all the shops the player encounters in each level. Sells the player a wide assortment of items which can be useful for the player.

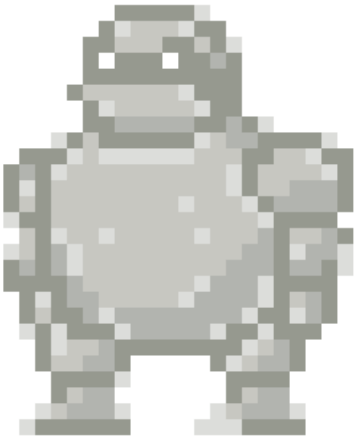
Each level will feature their own set of enemy monsters for the character to fight. Each enemy monster also has their own idle, hurt and death animations.

Different enemy monsters have different movement and attack speeds (time between each attack), health points, attack damage and attributes. Moreover, with each passing World, the enemy monsters get stronger and harder to fight.

Enemy Monsters	
	<u>Zombie</u> Health Points: 30 Attack Damage: 8 Attack Speed: 3 Movement Speed: 1 Zombies will chase the player when the player gets close to it.

	Skeleton Health Points: 35 Attack Damage: 6 Attack Speed: 2 Movement Speed: 2 Skeletons will chase the player when the player gets close to it.
	Imp Health Points: 25 Attack Damage: 9 Attack Speed: 1 Movement Speed: 3 Imps will not chase the player when the player gets close to it.
	Slime Health Points: 45 Attack Damage: 12 Attack Speed: 3 Movement Speed: 2 Slimes will chase the player when the player gets close to it.

	Eye Health Points: 40 Attack Damage: 16 Attack Speed: 2 Movement Speed: 2 Eyes will chase the player when the player gets close to it.
	Worm Health Points: 43 Attack Damage: 14 Attack Speed: 2 Movement Speed: 0 Worms are stationary in the position they are in.
	Lava Golem Health Points: 45 Attack Damage: 25 Attack Speed: 1 Movement Speed: 3 Lava Golems will chase the player when the player gets close to it.

	Stone Golem Health Points: 55 Attack Damage: 20 Attack Speed: 2 Movement Speed: 3 Stone Golems will chase the player when the player gets close to it.
	Rock Golem Health Points: 50 Attack Damage: 23 Attack Speed: 2 Movement Speed 3 Rock Golems will chase the player when the player gets close to it.

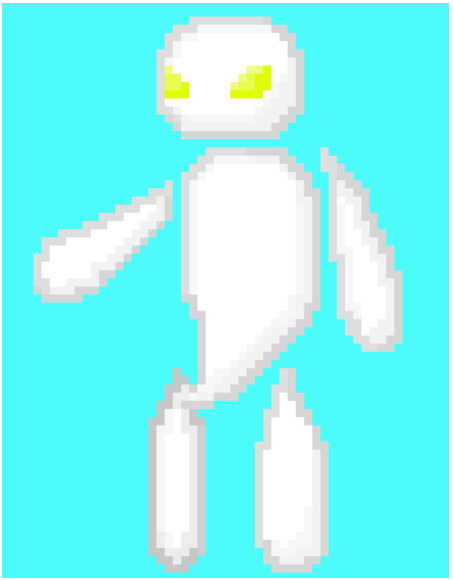
PLANNED (BUT UNABLE TO COMPLETE DUE TO TIME CONSTRAINTS):

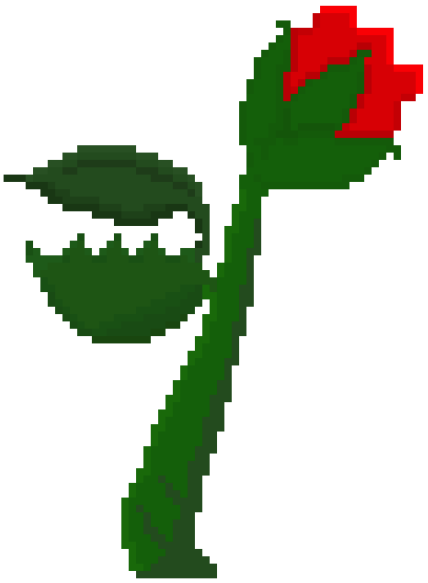
The third level of each World will feature a Boss fight. The Boss is significantly harder to defeat compared to the normal enemies found in level 1 and 2 of each World. The health and attack damage of the Boss will be much higher than the normal enemies.

The Boss fights will feature a three stage fight. In each stage the Boss will take on a different look and have different attacks. The health and damage of the Boss will increase with each increase in stage, making it challenging for the player to defeat the Boss. Only by defeating all three stages of the Boss, the player will win the fight.

Each stage of the Boss will have their own idle and attack animations. Stage 1 and 2 of the Boss will also have a transitional animation to the next stage, while Stage 3 of the Boss will have their own death animation.

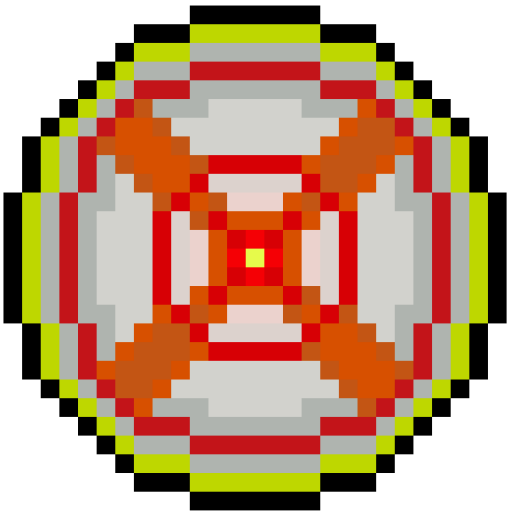
Enemy Bosses

	Ghost - Stage 1 Health Points: Attack Damage:
	Ghost - Stage 2 Health Points: Attack Damage:
	Ghost - Stage 3 Health Points: Attack Damage:

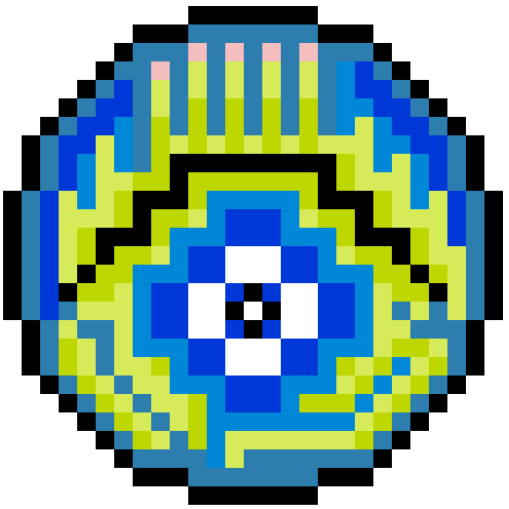
	Flora - Stage 1 Health Points: Attack Damage:
	Flora - Stage 2 Health Points: Attack Damage:

	Flora - Stage 3 Health Points: Attack Damage:
	THE DEMON Health Points: Attack Damage:

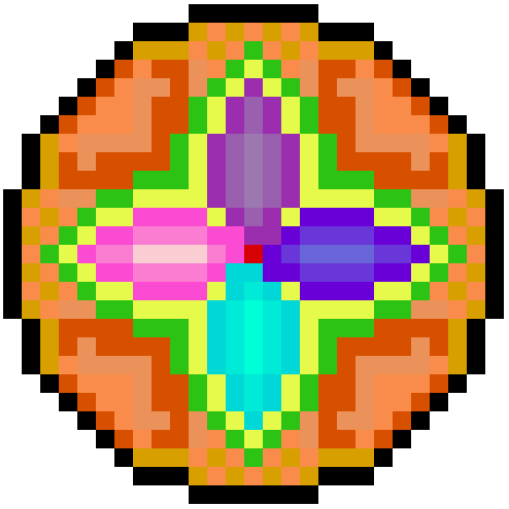
The game also has additional art for some other important items like the Portal to “THE DEMON” and the four medallions that are required to unlock the Portal. The portal in particular has its own animation for its activation.



Medallion of Protection



Medallion of Observation



Medallion of Eternity

Medallion of Balance

Portal to Void

Programming Principles and Framework

Overview of the key fragments of our programme

OBJECT ORIENTED PROGRAMMING (OOP)

==========

PLAYER AND ENEMY

The ***Player Game Object*** is mainly run by four scripts:

1) **Player**: The stats of the player including health, damage, defence, attack rates, and methods that affect the entire player such as healing, taking damage, dying, etc
2) **CharacterController2D**: This script controls the physics of the player's character. This includes the movement of the character (via changes in velocity), the jumping (via adding force), the crouching (which includes changes in the object's colliders), etc
3) **PlayerCombat**: This script is the main chunk of the combat system of the player. This includes methods such as light attack, heavy attack, and checking whether enough time has passed between attacks.
4) **PlayerMovement**: This script manages player input, and accesses the appropriate Classes based on the input.

The ***Enemy Game Object*** is mainly run by two scripts:

1) **EnemyStats**: Similar to the **Player** script, this script manages the enemies' stats.
2) **EnemyAI**: This script is what drives the behavior of the enemies such as when it moves, attacks, and patrols.

In order to make different character classes of similar formats to the above mentioned ***Player Game Object***, as well as different types of enemies that utilises a similar format to the above mentioned ***Enemy Game Object,*** we used the *Prefab Variant* feature of the Unity Engine. This allows us to change different fields of both objects for customisation.

This section showcases one of the 4 Pillars of OOP - Encapsulation. For example, with respect to the ***Player Game Object,*** there is modularity in the program whereby each script is in charge of certain functionalities, each of which are encapsulated into their own class.

==========

INTERACTABLES

This feature makes use of 2 of the 4 Pillars of OOP - Inheritance and Polymorphism.

Interactable: This script is an abstract class that will be the parent class from which all interactable items inherit from. It has the abstract method *Interact* which all interactable items will have to implement. As a result, each interactable item can do something different even with the same function *Interact* being called. We decided that this should be an abstract class instead of an interface, so that we could also add default methods and fields that all interactable items should have. For example, 2D Colliders are required, as well as the methods *OnTriggerStay* and *OnTriggerExit* so that the player can detect the item upon colliding with the item's colliders, in order to indicate that it's interactable.

==========

INVENTORY AND SHOP SYSTEM

This feature is a good example of the last Pillar of OOP that is not yet mentioned - Abstraction.
Whenever an item is to be bought from **ShopManager** and placed into **Inventory**, the **ShopManager** methods do not directly access the fields of **Inventory** (such as inventoryItems or inventorySlots), but instead use the methods available in **Inventory** such as *AddItem* or getters such as *GetNumOfPotions* and *GetNumOfCharms*.

==========

FUNCTIONAL PROGRAMMING
When building the message pop-up boxes (i.e the message that pops up when you pick up chests, or when you activate checkpoints) and the yes-or-no boxes (i.e the boxes that ask you if you are sure you want to delete an inventory item, or if you want to leave the level), we did not want to hard code a new box for every scenario. Therefore, abstraction was done in a way where functions were treated as first-class citizens, and given as an argument into a method to determine what the buttons of the boxes would do via code. This made the process of making a new variation of pop-up box simple, as a single method could be used to determine if we wanted the buttons to for instance give the player coins, or transport the player, etc.

COMPOSITION OVER INHERITANCE
Due to the complexity of the overall program, the only instances of inheritance we used were classes that inherited from **Interactable**, which made up the interactable items, and classes that inherited from **ScriptableObject**, which include shop items and character classes. Everything else follows the Composition Over Inheritance principle such that complex classes contain instances of other classes with their own individual behaviours, instead of inheriting from that complex class and adding new behaviours (methods).

YAGNI ("You aren't gonna need it")
Our code also follows YAGNI, which states that a programmer should not add functionality that is not needed, on the off chance that it may be needed in the future. This was achieved by constantly testing the programs, classes and functionality of the code, and deciding what needs to be added.

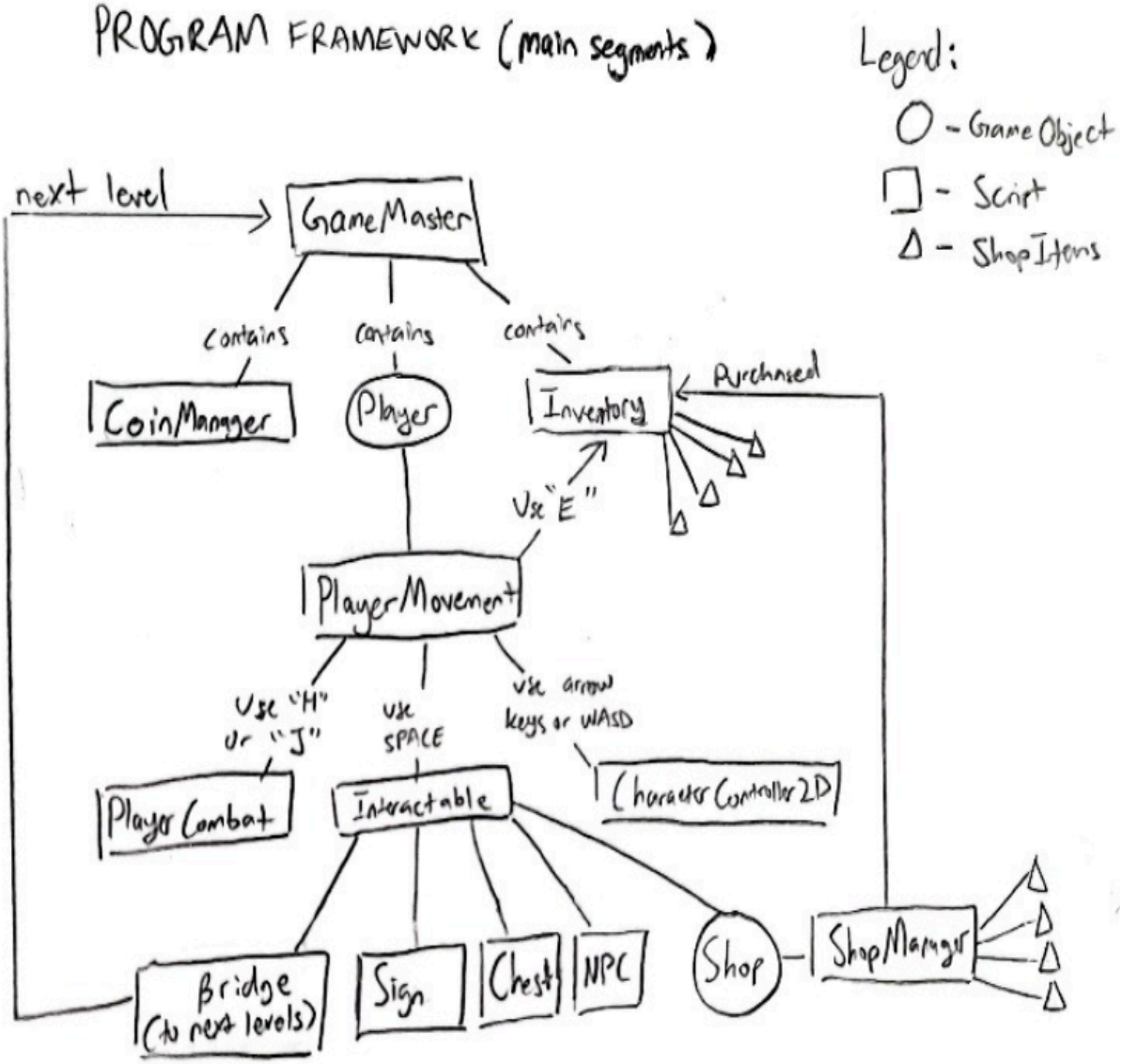
CODE DOCUMENTATION AND CLEAN CODE
This programming practice is self explanatory. We documented our code, particularly the more complex classes such as **CharacterController2D**, **Inventory**, **PlayerCombat** and UI boxes: **YesNoBox** and **MessageBox**.

We also assigned each shop item an itemType number and a codeNumber. This was documented in the **ShopItemSO** class as well:

```
ITEM TYPES
-1 => Empty Placeholder
0 => Medallions
1 => Potions
2 => Charms

CODE NUMBERS
100 => World 1 medallion
101 => World 2 medallion
102 => Puzzle Medallion
103 => World 3 Medallion

10 => Small Potion
11 => Large Potion

20 => Useless Charm
21 => Defence Charm
22 => Health Charm
23 => Heavy Attack Charm
24 => Light Attack Charm
25 => Money Charm
```

By assigning such numbers to each item, it is much easier to handle the data. For instance, everytime I want to check what item I just purchased, I do not need to compare the item's name (a string), and additionally, I do not need to hard-code a new if-block to compare that string. Instead, I could filter the data using math (i.e modulus) and switch statements. Furthermore, it is much easier to filter the data. For example, if I wanted to know whether a certain shop item is a potion, I could just compare either its item type, or apply a math function to its code number. Either way, both are more seamless and cleaner than comparing strings. Lastly, a side perk of using numbers is that it takes a shorter amount of time to compute the comparisons of small integers compared to long strings.

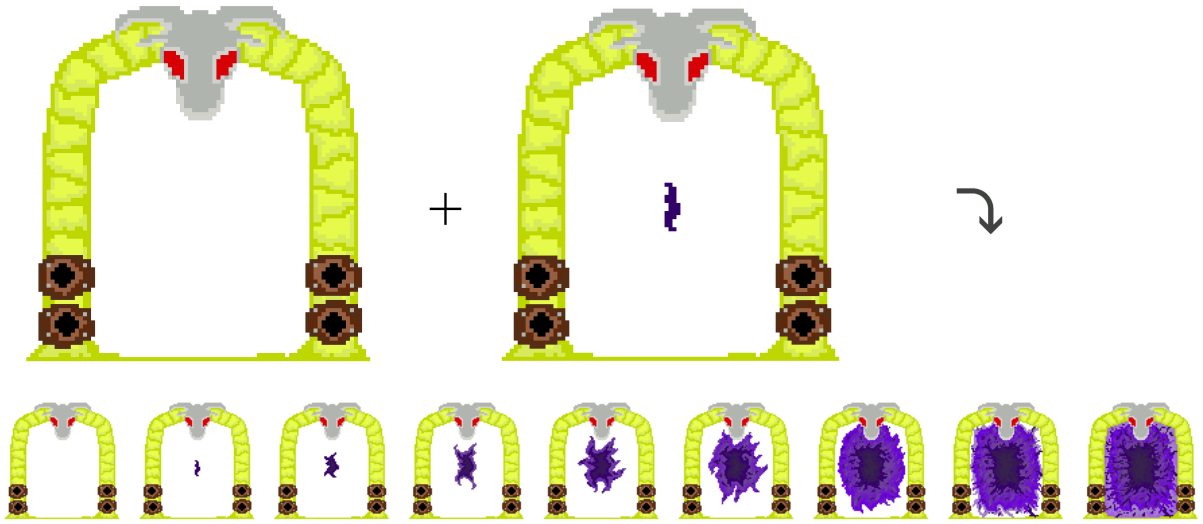
PROGRAM FRAMEWORK (main segments)



Asset Creation Workflow

Designing

Look for inspiration on the design of characters/world we want → Sketch a mock up of it in GraphicsGale → Make edits until the finalised design is obtained



Creation of Animation

Following the design, draw multiple frames of the character, each differing slightly such that when put together there is a flow of an action → Create a sprite sheet in GraphicsGale by combining all the different frames onto one elongated artwork → Upload into our game in Unity and slice the elongated artwork into its individual frames with the sprite editor → Create a new animation by putting each frame at a specific time and setting the sample rate such that the animation moves smoothly

Quality Assurance/Testing

Development Issues Encountered and Solutions

This section records the notable issues or bugs we encountered over the course of the development of the project and our solutions to solving the problems.

1. **Tutorial trigger issue**

After we finished the tutorial, we intended for it to be a step by step level to slowly expose the player to the main controls of the game, namely running, jumping, crouching and the attacks. As such, we programmed it such that there are specific triggers whereby only when we reached a specific area and started the dialogue for that portion, a certain key would be unlocked. However, after we completed the entire tutorial, we found that despite us specifically programming the trigger system, the player could still press the "locked" keys and the action would still be carried out.

Solution: After re-examining all the code at that point, we found that the issue was due to the sign present at the end of the tutorial. The Update function in Unity will be called after every frame. We were utilising the same dialogue box for all the tutorial dialogue (to unlock the keys) and for the interactable sign at the end. As the sign was at the end of the tutorial, there was no need to lock any keys there, hence when the update function was called, the sign's Update is also called upon and it activates all the locked keys, hence negating our trigger system. As such, we made it such that the sign's script only works when interacting (by utilising a boolean to keep track) such that its Update function does not constantly run in the background.

2. **Ability to Airsurf/walk on air**

After we completed the combat system for Milestone 2, we found that when jumping and attacking, the player does not freeze in the air, but instead glides in the air.

3. **Ability to Superjump**

After we completed the combat system for Milestone 2, we found that attacking and pressing mid jump animation will result in the player jumping really high.

Solution: Both problem 2 and 3 were combat-related problems. We made it such that mid-attack animation the player's position will be frozen, similar to the situation where the player is in dialogue.

4. **Combat System**

After we completed the combat system for Milestone 2, we realised that the enemies had no gap in timing between attacks.

Solution: We fixed it such that there is a customizable cooldown between attacks, so that some enemies will have a higher attack rate, while others will have slower attack rates.

5. **Inability to transfer the Inventory items between levels**

After we finished the Inventory feature, we playtested to test the Inventory. We found that when we progressed from one level to the next, the items we had in the prior level would not be brought along to the new level which was not intended. We looked into the code and found that there was trouble referencing the gameobject (inventory) from a different scene.

Solution: Since static fields retain data through different scenes, we decided to save the inventory data as a static field in the gamemanager, then apply that data to the inventory in the new scene upon loading, thereby ensuring the inventory items could be brought along to new levels (scenes).

6. **HP always regenerates to full after entering the next level**

With the implementation of the Health Potion system, we wanted the players to have to work for their health and to make the game harder. As such, we intended for the current amount of health the player has to remain constant until the player uses a Potion. However, after we playtested our game, we found that despite taking damage in one level, upon reaching the next, the health of the character is fully regenerated.

Solution: After examining all the code, we found there to be one extra line of code(which we probably added initially without realising or for testing purposes) which was currentHealth = maxHealth. As a result of this line, whenever the level is spawned in, the player's health will be regenerated completely. As such, the fix was simply to just remove that line of code.

7. **The Health Potions not working**

After we finished implementing the Health Potions and Charms, one notable issue we first noticed was that the Health Potions were not working. Despite being able to utilise them in-game just fine, the health of the character was not being regenerated.

Solution: After examining all the code, we found the issue to once again to be a small mistake on our part. We had a line of code where for the Small Health Potion, it would add (int)0.25f * maxHealth, which ensured that it essentially would still be adding health (thus the Potions were working), just that it was adding 0 health due to the typecast to int, causing the line to essentially be adding 0 * maxHealth = 0 health. To fix this we simply typecasted the whole line (int) (0.25f * maxHealth).

8. **Character couldn't move**

While programming something, we decided to playtest the thing we just programmed, hence we switched to Unity's play mode to try it. However, for the first time, the character couldn't move at all in the level we were in, as well as all other levels. However, the character seemed to move just fine in the tutorial.

Solution: By comparing each component of the player game object in the tutorial against the player game object in any one of the other levels, we found that the player game objects in the other levels had their X-position frozen as well as having their PlayerMovement script disabled. These were the exact effects of being in a dialogue. This took a long time to find but after fixing each player game object manually in every level, the problem seems to be fixed.

<u>Manual User Testing</u>

This section records the responses we received from our peers after sending them the game to playtest.

For **Milestone 2**: We sent the link for the current state of the game to 5 of our friends to try and survey them for their opinions thus far.

Key feedback:

- The movements works fine
- The combat system could be better
- There are certain unintended actions that can occur when pressing different buttons at the same time (ie pressing the attack button and jump button at the same time will cause a super-jump)

For **Milestone 3**: We sent the link for the current state of the game to 5 of our friends again to try and survey them for their opinions thus far.

Key feedback:

- The movements work fine as last time
- The combat system is much improved compared to last time, no obvious bugs that cause significant problems
- Lack of audio
- The Mage and Archer class feels stronger than the Warrior
- The many features in the game (ie shop, inventory, interactables) make the game feel more complete and expansive

We noted their opinions and made the following notes:

1. **Lack of audio**

There was no audio in the game thus far.

Solution: We aim to add it into the game before Splashdown.

2. **The Mage and Archer class feels stronger than the Warrior**

The Mage and Archer class are ranged classes hence they need not be directly next to the enemy minions in order to deal damage to them as compared to the Warrior. At this point, all three classes have knockback enabled after the character attacks the enemy. This was to accommodate the Warrior class such that the enemy could not repeatedly do damage to the Warrior by continuously walking into him.

Solution: We decided to disable knockback for the Archer and Mage classes so as to increase the difficulty of combat for these ranged classes. As a result of this, the monsters would still be able to advance towards the Archer and Mage and increase the potential for them to be damaged. On top of that we further decreased the attack range of the Archer and Mage to prevent them from being too much stronger than the Warrior.

3. **There are moments when you die where the game over screen does not come up**

The hotkeys “R” for restart, “L” for load, and ”Q” for quit still work, but the game over screen sometimes does not pop up.

Solution: This is likely caused by the death collider being too thin so if the player falls at too high of a velocity, it could pass by the collider. So we made the collider much thicker to prevent the chance of the player missing the collider.

Tech Stack

Unity
We utilise Unity as the platform to make our game due to it being very beginner friendly and allows for ease in construction of the game.

GraphicsGale
We utilise GraphicsGale as the platform to create our character and game art as it is catered to pixel art drawing which was the direction of artstyle we wanted to go for. Moreover, it is very simple to use and has key features like a wide palette of colours and saving the drawings with the transparent background.

Link to download for A Hero's Tale:

https://github.com/BryannYeap/a-heros-tale