CS3 CAPSTONE PROJECT

Final Report

Alka Baijnath - BJNALK001
Jarryd Dunn - DNNJAR001
Wilhelm Prins -PRNSAM002

Contents

Requirements Analysis

Functional

- Two independent players are able to play the game simultaneously. These players should be able to interact within the game world.
- The game takes place in a 3-dimensional environment making use of 3D assets. The players should be able to navigate within the world.
- The game must contain non-player controlled characters. These characters must be used to increase the difficulty and interest of the puzzles within the game.
- The theme for the game must be consistent thought the duration of the game.
- The game must include audio components to enhance the players' experience of the game.
- Visual effects must be incorporated into the game to add the aesthetics of the game.

Non-functional

- Keep frame rate at an acceptable level such that any lag is imperceptible to the user.
- The game should behave in a reasonable manner, with no unexpected/inexplicable behaviour.
- The game should respond to user inputs in a reasonable amount of time such that the user is not aware of the delay.
- The game should be reliable in that it reacts to user interaction in order to let the player know if their actions are being registered.
- The game should be able to handle the player's input.

Usability

- The movement of players should be controlled using common keys/buttons (e.g. 'A','S','W','D').
- The layout of the controls for the players should be such that the players can independently control their characters without getting in each other's way.
- The player's current status within the game should be clearly indicated and easily accessible to players.
- Common actions should be easy to complete and intuitive to the user.

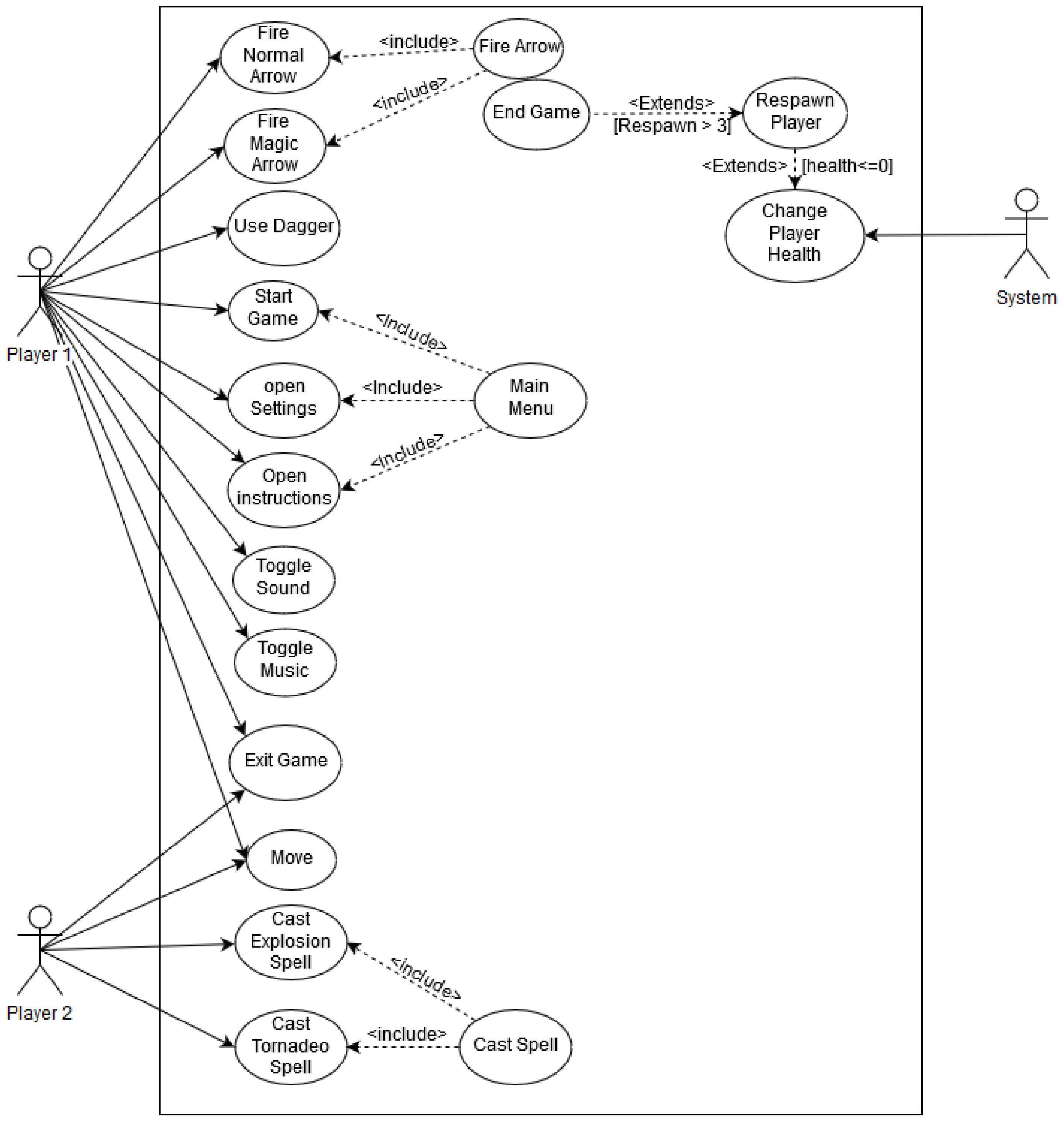
Use Cases

- Two players should share a monitor viewing their own characters actions independently through a split screen. The two characters may interact with each other within the game world in order to solve puzzles and complete the game.
- The player controls the movements and actions of a game character in three-dimensional space. The players provide input to the game through either the keyboard and mouse or an x-box controller. The actions of the character provide feedback to the player as well as allowing the player to progress through the game.
- While playing the game the player receives visual and audio feedback from the game. These may result from the player's actions, such as when the player's arrow hits a target and explodes. They may also be used as part of the game world such as items that need to be destroyed in order to gain access into different parts of the dungeon. One player (Persephone) is equipped with a bow and arrow (normal or magic) as well as a dagger, she can use these to

interact with objects in the game. The second player (Demeter) has two types of magic which can be used to destroy or activate various objects throughout the game.

- The player needs to avoid and overcome any NPC enemies they encounter. This can be done by avoiding the enemies' line of sight. Alternately the player may try to destroy the enemy with their weapon (dagger, sword or bow) however the game is biased to make avoidance a better tactic then engaging with the enemy.
- Through the game there are tasks that only one of the players can perform. Thus the players must cooperate in order to progress through the game.

Use Case Diagram



HIGH LEVEL DESIGN

Components

AI

There are four different types of A: soldiers, sentries, light guards and ghosts. Its component function is to provide the players with some obstacle/challenge and increase the 'fun' factor of the game.

Players

There are two characters (Persephone and Demeter) each with different weapons and abilities. Their relation and significance to each other is explained during the introductory story.

Inventory

The inventory collects and stores any collectable items in the game. It also manages the use of these items. The inventory system enables the players to view what collectables they have acquired as to decipher what they still have to get in order to finish the game.

Health

Both NPCs and players have a health script which manages their health. If a player's health is less than or equal to zero they will respawn at the last checkpoint they reached, however a player may only respawn five times after that if the player dies the game is over. The player can also increase their health system by picking up health gems.

Weapons/Spells

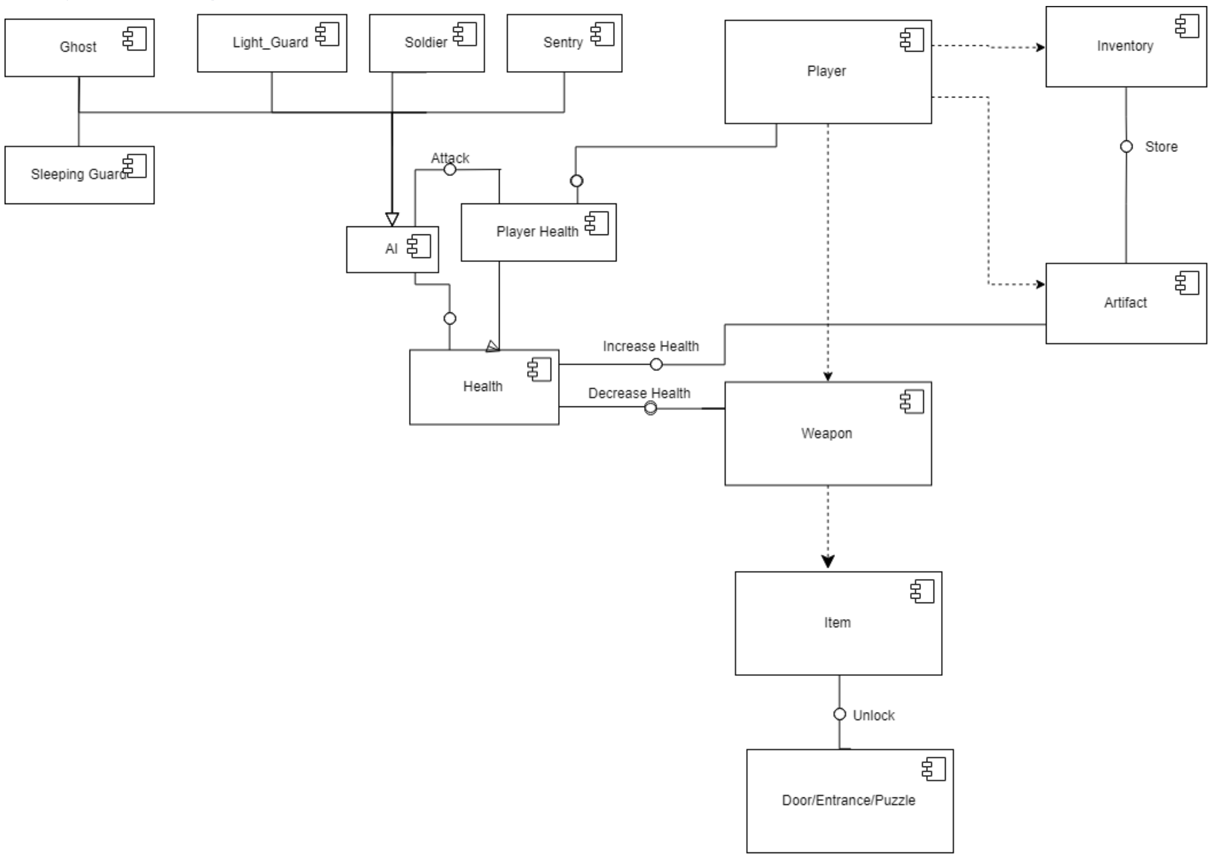
The players are provided with different abilities. They need to exploit their strengths and powers in order to solve puzzles.

Puzzles

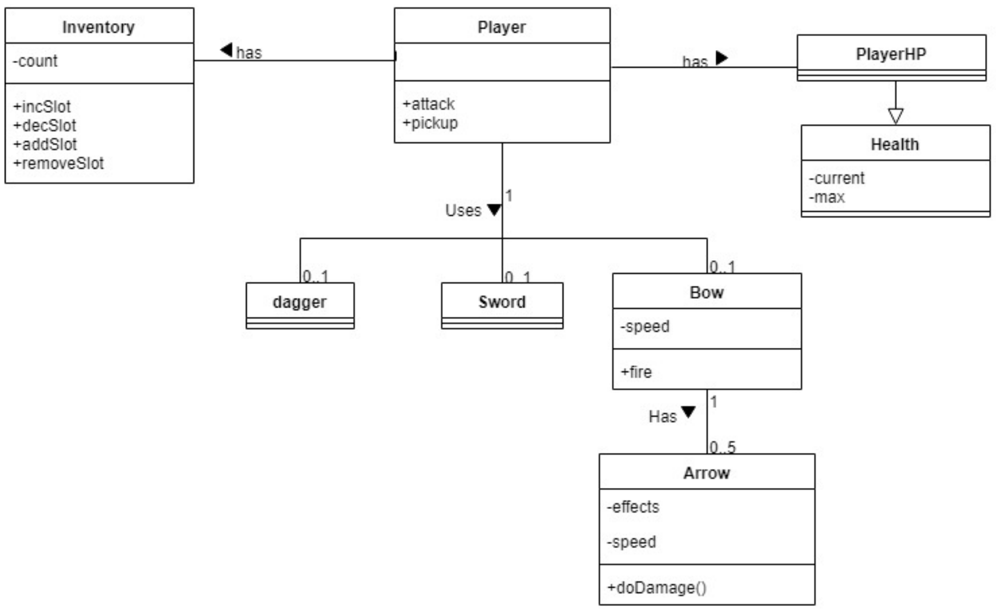
The puzzles are the core component of our game and provide the gameplay aspect. The players will have to complete all unique puzzles in order to win the game. Each puzzle has a distinct set of actions that must be completed in order to move onto the next puzzle.

Diagrams of our overall architecture

Component Diagram



Class Diagram



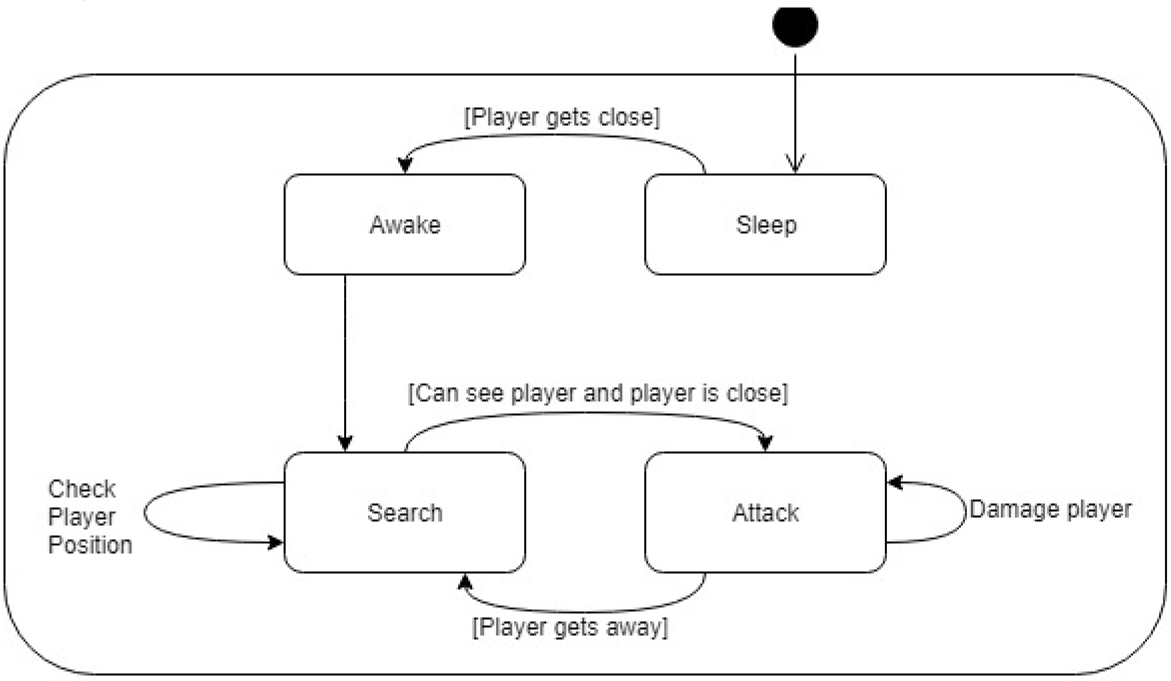
IMPLEMENTATION

AI

There are four different types of AI: soldiers, sentries, light guards and ghosts. All four inherit from the Enemy script. Each of the enemies are use state transition machines to move between sleep, search, chase and attack states. These states determine the actions of the AI. A nav-mesh is used to allow the AI agents to navigate through the world towards their goal.

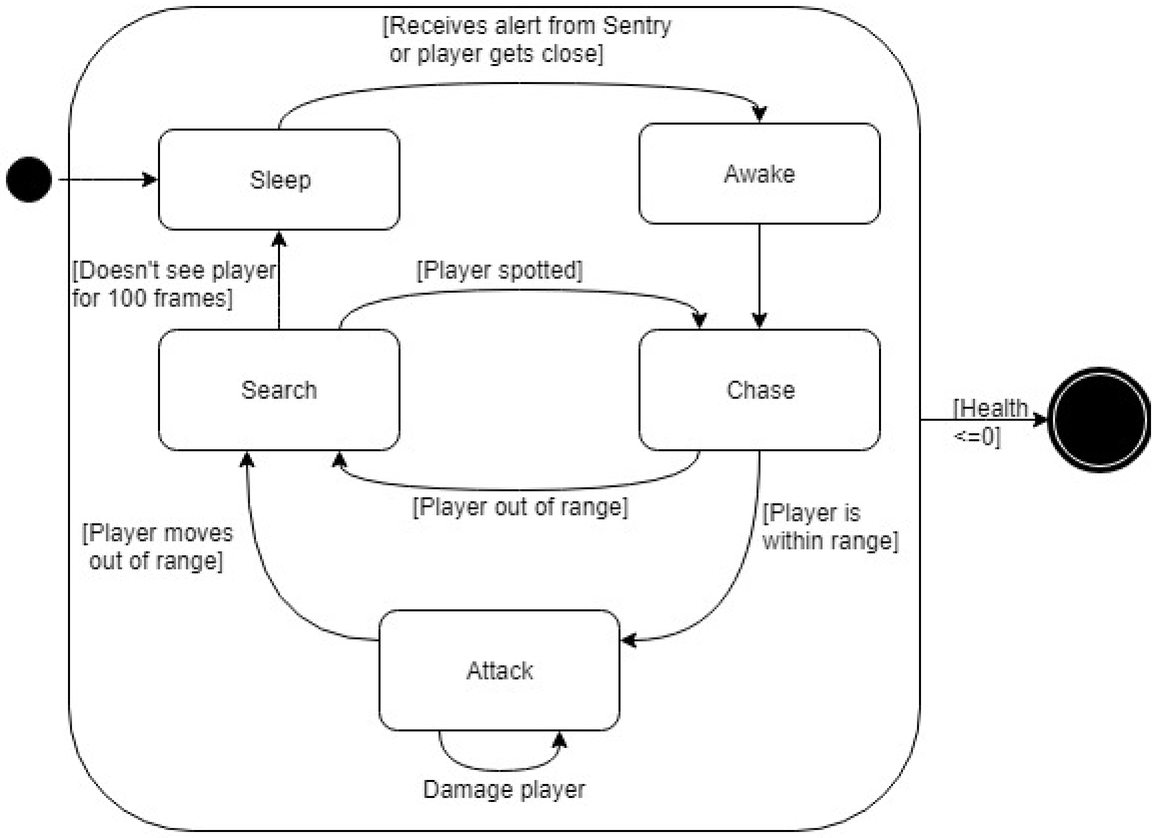
Sentry

Sentries appear throughout the game and may be an indispensable part of a puzzle or just there to add difficulty. The sentry NPCs will chase and attack a player however their primary purpose to alert soldiers to the presents of a player as well as sending the soldier the position of any player they can see. Sentries start in a stationary "on-guard" state and pursue the player once they have spotted them, though they can lose sight of the player causing them go into a search sate looking for the player.



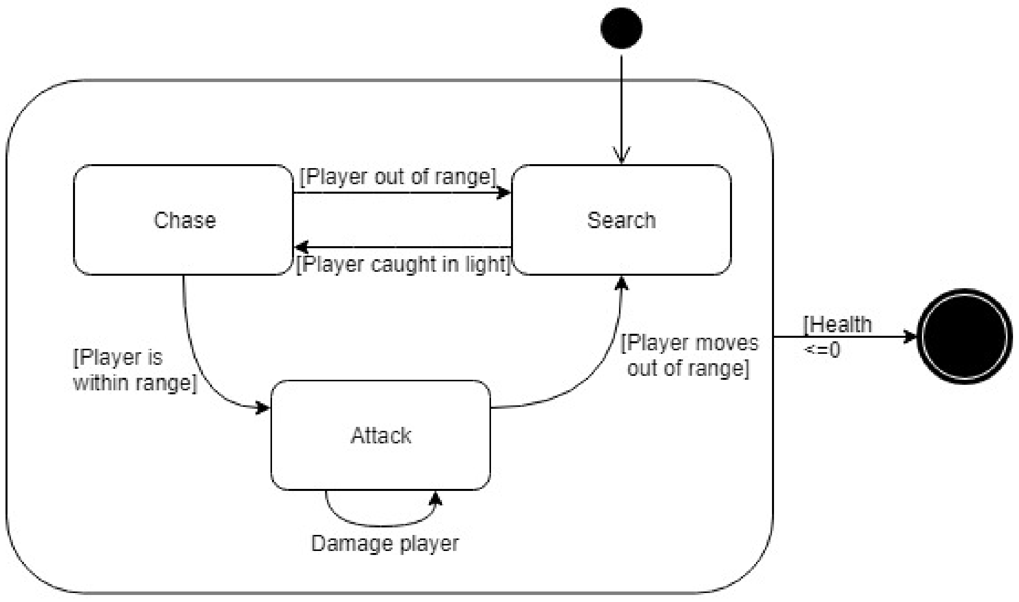
Soldiers

Like sentries the soldiers appear throughout the game. The soldiers are “woken” and instructed by the sentries though they may wake up if a player gets too close. Soldiers do more damage than sentries.



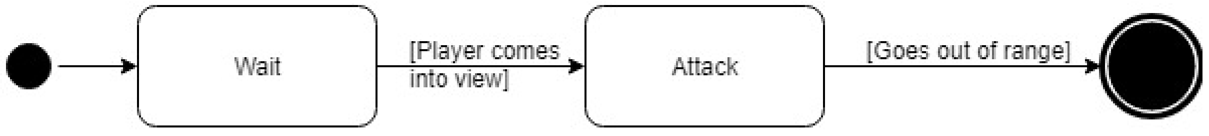
Light Guard

The light guards form part of one of the puzzles, they patrol a dark room through which the players must pass. Each guard carries a lantern, if a player is caught in the light the guard peruses the player though the guard will lose track of the player if the player goes out of sight (is no longer in the light).



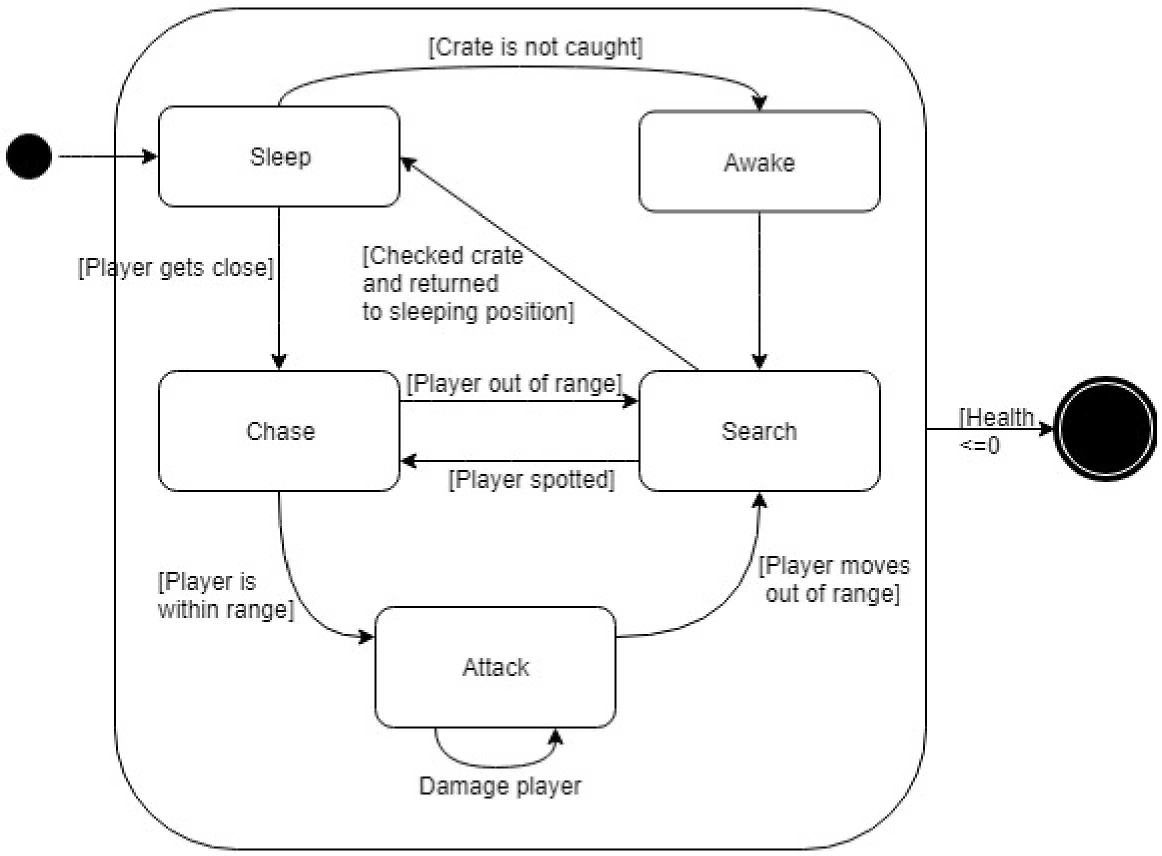
Ghost (Shriek Skeleton)

The ghosts hide in corners and rush at the player if they come too close inflicting a small amount of damage before disappearing into the depths of the dungeon. The main purpose of this AI is to add to the spooky atmosphere of the game.



Sleeping Guard

The Sleeping guard is used for one of the puzzles. Its responsibility is to wake up and attack the player if the make too much noise while completing the puzzle.



Players

There are two characters (Persephone and Demeter) each with different weapons and abilities. Both players have a health and inventory system to track their health and the artefacts that they have collected. Demeter is equipped with two different types of magic, an explosion spell and a tornado spell. Persephone has a bow, which fires magic or regular arrows, as well as a dagger. Players may use their weapons/abilities to solve puzzles and so progress through the game world.

Health

Both NPCs and players have a health script which manages their health. If a player's health is less than or equal to zero they will respawn at the last checkpoint they reached, however a player may only respawn three times after that if the player dies the game is over. Players can see their health via health bars.
If an enemy NCPs health drops below 0 then the enemy is destroyed.

Weapons

Each weapon was implemented uniquely. They all behave differently and are assigned to either player. There are three types of weapons we have implemented, namely melee, magic and ranged. It is apparent in the way they are coded to behave.

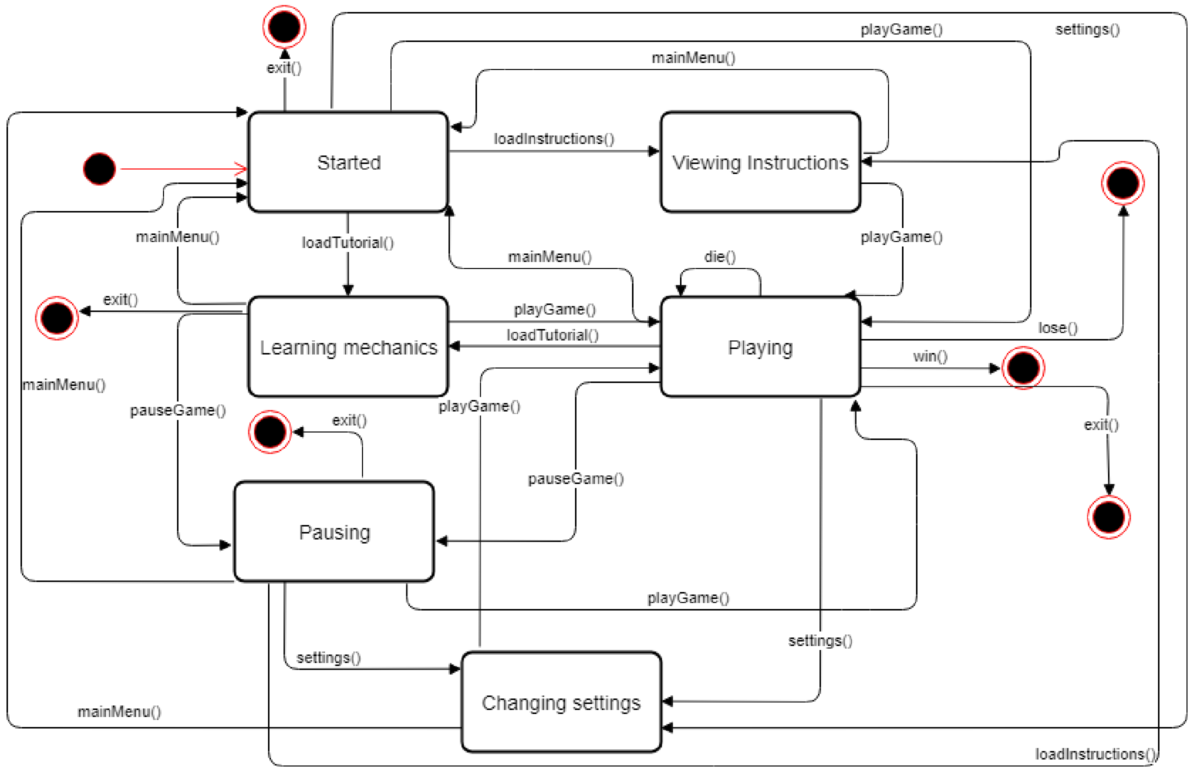
Inventory

The inventory collects and stores any collectable items in the game. it also manages the use of these items. Each player has their own separate inventory displayed on the screen. The inventory shows any collection items that the player may have found. The internal mechanics of the inventory are made up of an array containing the number of items in each slot and a pair of dictionaries, one to find the index of an artefact within the inventory and another to get the image to be displayed in the inventory. Then inventory class contains methods to add and remove artefacts. The artefacts are always stored such that they are displayed from top to bottom in the inventory panel.

UI

Our UI implementation during the gameplay comprises of the information that needs to be displayed i.e. health, inventory box. Our UI is consistent, neat and visually appealing. It matches our theme and has precise behaviour.

State Transition Diagram to show how we implemented the functioning of our game



PROGRAM VALIDATION & VERIFICATION

The majority of our program validation and verification took the form of a play testing and debugging. We first developed a separate testing game world for both the AI and the players. This separation allowed us to carefully control and test the interactions with the players and AI. These “testing grounds” comprised of a few simple elements in order for us to see how the elements interacted with the surroundings and vice versa. Once we were satisfied with the development of these components in the test environment, they were transferred into the game environment.

We then incrementally and iteratively developed the puzzles. We first tested how the puzzle environmental components behaved and then progressed to incorporating the players and AI into the puzzles.

Table showing various test cases that we conducted

Test Case Number	The inputs, the behaviour being tested and the expected outputs.
1	Player uses keyboard/controller to move a character onto a pressure plate expecting a door too open. This is meant to test if the pressure plate is working as intended and if its animations are functioning.
2	Player navigates around game world, picking up health gems. This is meant to test if the player is colliding with the health gem and if their health is being increased accordingly.
3	Player uses keyboard/controller to shoot an arrow expecting the arrow to do damage to an enemy. This is meant to test if arrows are damaging enemies.
4	Player uses keyboard/controller to use expecting the spell to do damage to an enemy. This is meant to test if magic are damaging enemies.
5	Player uses keyboard/controller to interact with our orb system expecting a game event to occur when appropriate attribute is applied to a specific orb. This is meant to test if our orbs are reacting to the correct attributes.
6	When a player collects a previously unfound artefact it should appear in the inventory with a count of one. Then when a player collects an artefact that is already in the inventory the count for that item should increase. This is testing to see if our inventory system is working correctly.
7	If the player uses an item from the inventory the count should decrease. If the count is zero the slot is removed and slots below it move up. This is testing to see if the inventory behaves correctly when removing artefacts.
8	Player approaches/is spotted by an enemy and player’s health depletes. This is testing that the enemies are inflicting a considerable damage on the players.
9	Player depletes their health five times and the game is over. This is testing that the amount of time a player dies is being kept track off.

10	Player finds a key during a puzzle and approaches a door, the door should open with the key. This is testing whether the doors are being opened correctly during the puzzles.
11	Player switches between their weapons to do certain puzzles. This is testing whether all the various weapons are working and if they are being used optimally.
12	Players work together to do a sequential set of actions in order to complete a certain puzzle. This is testing to see whether the puzzles have been implemented correctly.

DISCUSSION

At the commencement of our game one of our major downfalls was our inflated scope/feature list. Shortly after beginning our game development process we realized this and switched our focus to identifying the most important features and devising a plan to implement them.

During the early stages of development we attempted networking for our game. After a week of trying we had gotten it to work but decided to opt for split screen as we were unsure of the errors and problems networking could bring. This decision helped us reduce lag and gave us more confidence in having control of our game. We also feel like the use of split screen makes it a more sociable experience for the players.

Unfortunately due to time constraints we were unable to implement a saving state and game economy (like we have previously stated in our game design document). We transformed our game economy to an inventory system which served as a tool for storing artefacts and other collectables. The result of using an inventory system worked better with our game's theme and style. We decided to remove a previously stated feature of having an outfit change as we felt like this feature was unnecessary and served no other purpose besides appearance. However, we did implement a weapon system. Each player had access to two types of weapons which may have included more variations. The outcome of focusing more of our time on developing a weapon system is having weapons that function accurately, precisely and are aesthetic.

We have implemented a wide array of different functioning AI. Our AI system is elegant in that it was well thought out and developed. Our NPCs provide a specific purpose to the game and enhance the gameplay greatly. One of our more impressive levels comprises of NPCs which use a light source to locate the player. These AI function with great precision and have a somewhat intelligent feel to them.

The beginning of our environmental development was a bit tricky in terms of deciding how we wanted it to look and then using those ideas to implement good design principles. In spite of that, the more we designed, the clearer our goal became and the easier it was for us to make decisions about our environment. Our end result is something we are all proud of as we feel we produced a great game world.

We had actually implemented all our puzzles quite early but decided to increase the complexity of each puzzle. Therefore most of our time was spent of this puzzle development and fine tuning them to work exactly how we wanted them to.

The visual effects we have used are apt, fit our theme well and are luminescent. We tried to develop a few of our own or find ones that we thought would work well in our game.

The process of trying to find players was challenging as we had a limited number of assets to choose from. Our players are functioning well. We spent a large amount of time developing them as they are the central aspect of our game.

Though we may have not met all our goals or implemented all of our wanted features, we have created a game that we feel is fun and has a high quality standard. The game development process was fun; we worked well together, collaborated on a lot of ideas and all contributed to the end product.

User Manual

Below are the set of controls for each player. Player 1 uses the keyboard and mouse controller while Player 2 makes use of an Xbox controller. These two controls are used in order to create a more comfortable gaming environment for the players as they are each provided with their separate controlling stations.

Player 1 - Persephone

- A,S,D,W: Forward, Back, Left, Right
- Mouse: Move Camera
- Q: Switch Arrow
- E: Change Weapon
- Alt/Left Click: Aim
- Right Click: Fire [If Aiming]

Player 2 - Demeter

- Left Analogue: Move
- Right Analogue: Rotate Camera
- A: Jump
- Y: Switch Weapon
- Left Trigger: Aim
- X: Use Tornado Spell [While Aiming]
- Right Trigger: Shoot Explosion [While Aiming]

References/Credits

Sound On/off: https://www.flaticon.com/free-icon/speaker_149139
Music On/off: https://www.flaticon.com/free-icon/musical-note_126493#term=music&page=1&position=6
Zombie Scream: http://soundbible.com/1459-Psycho-Scream.html
Skeleton: https://assetstore.unity.com/packages/3d/characters/creatures/dungeon-skeletons-demo-71087