Munth

A Storytelling Role Play Game

Analysis----------

Problem

Currently, there are a number of games in the video game market with bad writing. Most video games generally have bland characters, huge plot-holes and hold little depth to the game "world" their character is living in.

With an overwhelming consensus from video game critics, journalists, and novelists, they believe most video game stories exist to support the gameplay, not the other way around. *In a 2018 essay, Falmouth University game design lecturer Rory Summerley wrote "fictions can often be interpreted as just an excuse" when referring to video game stories being placeholders for the gameplay they hold.*

In most games, we see the overall experience split into: cutscenes for the story and gameplay for the engagement. The dualism usually shatters the experience for the player.

I have decided to design a videogame which prefaces the story first, over any mechanical elements of the game design and in failing that, keeping all game mechanics centred around their ability to work with and contribute towards the story.

This video game will be able to display the literary devices and depth seen in books and the tv industry and incorporate them with interactive storytelling.

This problem was first brought to my attention by my client with the hopes of exploring the medium of game design to challenge the market and develop a new title with her knowledge of literary devices and story writing.

Client

Grace Blackshaw

My client is a member of the Twyford schools' champions club, a book club for literary enthusiasts. She came to me during a session I was attending to talk to me about her experience with recent video games she was playing and how poor the storytelling was.

Together we have decided to create a game she will be immersed in and be able to connect with the characters, also one which others will also enjoy and explore. My client has stated that the game will need to be played on her desktop computer with a mouse and keyboard as that is the only way in which she is able to access them.

The game's story will also need to be restricted, with a script length of a short tale, no more than an hour's long at most as to not drag out the experience and bore the user. My client is 16 years of age so more mature themes can be explored in the story if necessary, but it will also limit the age rating of the game to a 16. This restriction will have to be considered when script writing. As the client is new to the gaming industry, the save system must be accessed at any time to be able to pick up and put down at any moment like a book.

Transcript

==========

Adam: What type of genre will the story fit into?

Grace: I think we should use many elements of fantasy and anchor around adventure tropes as I have most experience in those genres.

==========

Adam: Will the focus be on more than one character?

Grace: No, it should be based off one main character with non-playable side characters.

==========

Adam: What time period might it be set in?

Grace: In my mind, I imagine this game to be set in a time period of which Andrzej Salkowski's series of books "The Witcher" is loosely based, so I guess medieval around 1400AD?

==========

Adam: Will the character have a backstory?

Grace: Yes, sort of. His motivations will play a role; however, his past is not the main plot device of the story.

==========

Adam: Will there be a prologue? What might happen in it?

Grace: Maybe... There should be a mini prologue where it shows the main character as a child seeing something like his hero or some motivation to become something greater i.e. a royal knight.

==========

Adam: Other than movement and interaction mechanics, what might you want to include? How might it affect the pacing of the story?

Grace: I think we should have an Undertale like turn based combat mechanic where you can either choose to attack, defend or converse in which we can learn more about the attacker and choose the right option to spare them. As it is a story focused game, a turn-based mechanic works best as it does not take away from the story in which a real time combat system would as it will have to focus solely on the combat because there is no time to read any text.

==========

Adam: What TV shows, books or films might you want to take from?

Grace: Obviously the Witcher series I mentioned before but also lord of the rings, and Sherlock because of the way it unravels the story, giving subtle hints to the audience about the revealing plot, but only piecing them together after the climax. The mystery keeping the watcher engaged is something we can use throughout the story as well.

==========

Stakeholders

Kornel paluszkiewicz:

- Age: 16
- Enjoys single player, story 2D games with survival elements as well as
- Programmer

I chose kornel to be one of my stakeholders as he has hundreds of hours of experience playing similar styled games and will be in touch with what mechanics work and those which do not.

Use: I believe that he would use my program as he enjoys the idea of new experiences and is growing bored with the small repertoire of games he plays and would leap at the chance to be involved in the development of one.

Max Brooker:

- Age: 16
- Enjoys strategic RPG games
- Has thousands of hours of experience

Similarly to kornel, I chose max to be one of my stakeholders because he has major experience with the genre of game I am basing mine around, he will be also useful as he plays other genres alongside meaning he can bring an outsider's perspective to keep the project grounded.

Use: Max Brooker is similar to Kornel in the aspect that he is eager to play new games and will relish the experience and the end product.

Emily Tagg

- Age: 18
- Has little experience with games
- Enjoys well crafted stories

I chose Emily to be one of my stakeholders because she is a regular consumer of books and stories. She will be able to inform my decisions and thoughts on story writing.

Use: Emily is very invested in the project and how it will turn out because she would like to see a mix of storytelling and game design as she has never ventured into the area of gaming and feels this would be a great way to introduce her.

Jonathan Bush

- Age: 16
- Creates poems
- Writes songs

I chose Jonathan to be one of my stakeholders as he is a writer. This insight will be useful to teach and guide me through my development and story writing.

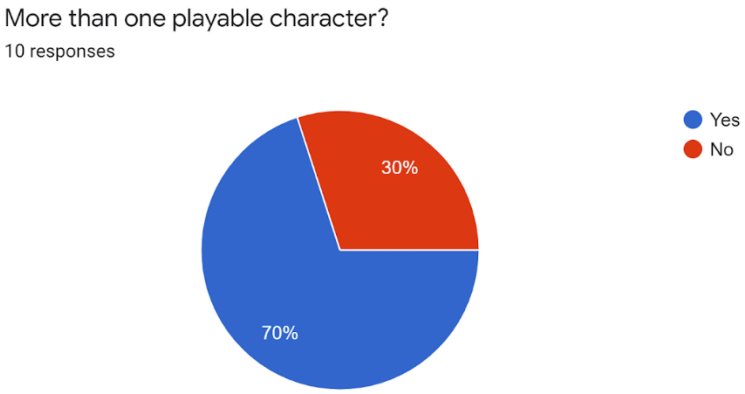
Use : Jonathan was searching for a project to work on and strengthen his literacy skills, when i proposed for him to be a stakeholder he was enthusiastic about the idea and believed that he could learn a lot from this project.

Users

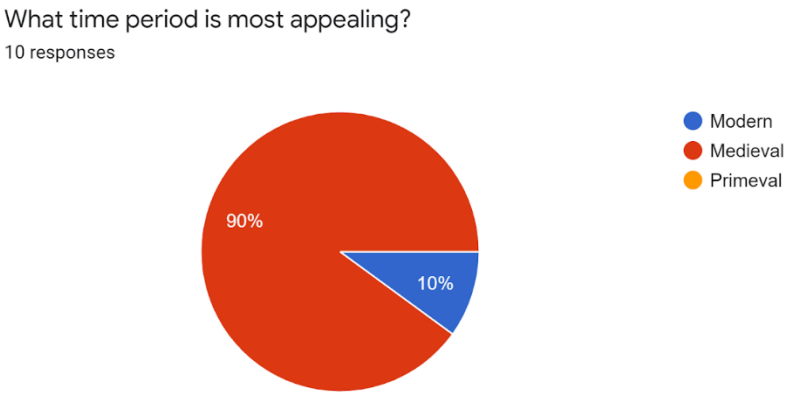
The people who may be interested in playing my game include people into RPG's, Single player stories and book readers. This audience could be within the age range of 14-30 as older users can appreciate the storytelling even if inexperienced because of the elegantly and simply designed mechanics. People younger than age 14 would not be interested in the game as that age range usually prefers fast paced reaction based gameplay to immersive due to their short attention spans.

Questionnaire

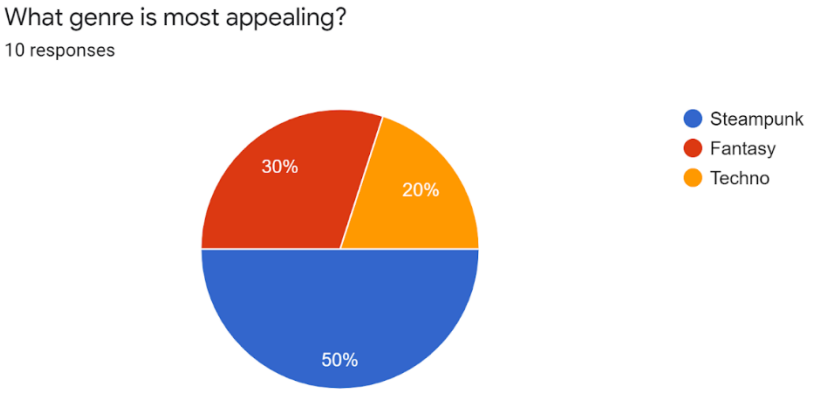
I sent out a questionnaire with similar questions which I asked my client, to my stakeholders and potential users. Here were the results:



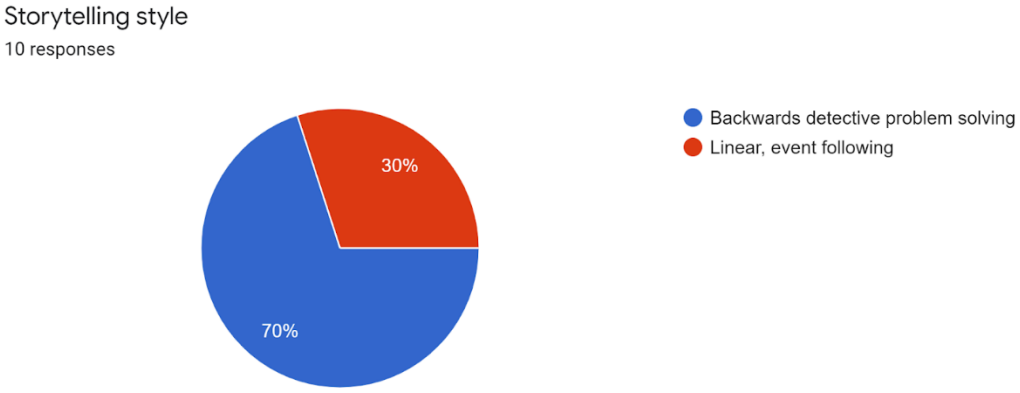
The majority of responses around the addition of more than one playable character being available in the game is mostly “yes” with 70% for and only 30% not. This would be a timely addition to the game but something users seem to want so it would need to be considered and brought up with the client.



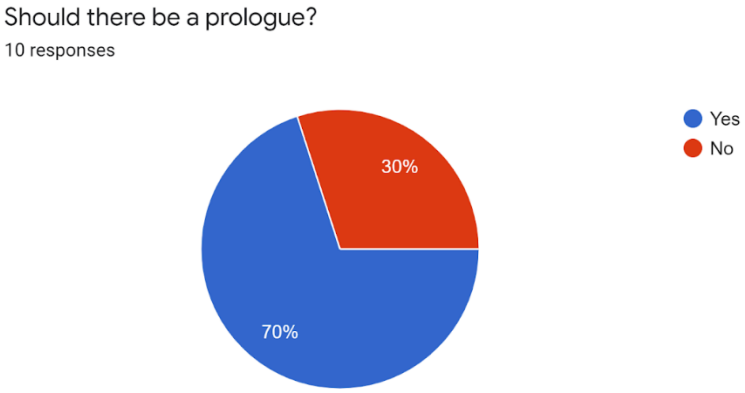
An overwhelming majority for a medieval setting of 90%, only 10% for modern and no primeval votes. This is clear that the audience of the game prefers a medieval setting for the story to take part in.



Votes are largely split on the genre in which the story may be set in, techno and fantasy getting both 30% and 20% of the vote respectively. Steampunk, however, has the overall majority of 50%. It may be worth considering a mix of genres to appease more users by combining strong elements of each.

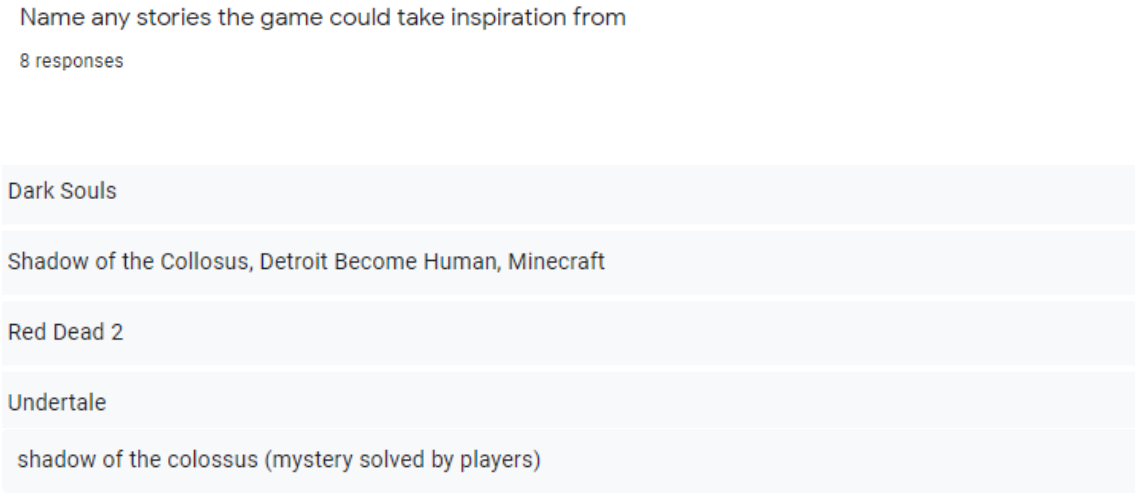


Here, 70% of users voted for a detective style mystery plot where clues are unravelled throughout until they all fit together like a jigsaw reaching a satisfying conclusion. Only 30% of users wanted to see a version of the tried-and-tested linear event path.



Most users think that prologue's are worthwhile additions to a storytelling game with a 70/30% split.

In this questionnaire I also asked for any suggestions of good storytelling they know of:



The response of Undertale and Shadow of the Colossus caught my eye in particular as they are industry defining games. Considering my clients mention of undertale as well, I have decided to include it in my research:

Research----------

To support my problem hypothesis, I have decided to research and break down 3 critically acclaimed games to understand what details, gameplay loops and story techniques they use and how I could use them to create a game of my own to blend the story and gameplay.

Undertale----------

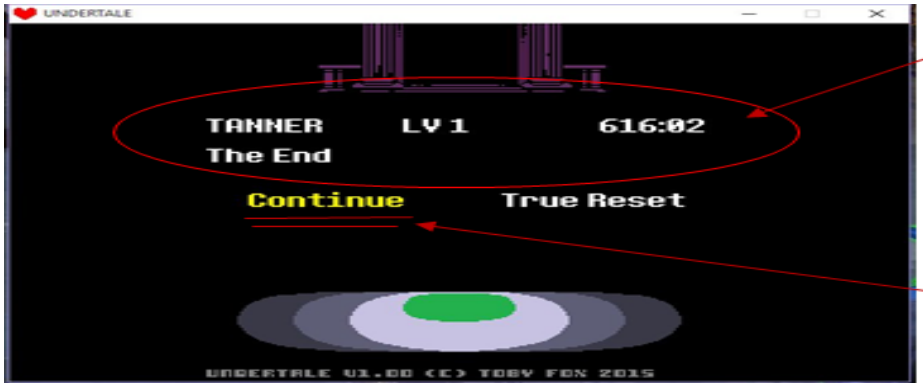
Undertale is the story of a human child who falls into the underground, which has become the home and prison of the monster race ever since humans drove them from the surface. Monsters have set up a life in this new place, but long to return to the surface. You are given the ability to spare or kill almost every one of these monsters on your own journey to return to the surface.

Undertale turns the medium itself into a critical piece of the narrative. Elements such as saving, loading, restarting, and the UI are all a part of the game's world. The very nature of a game and the control a player has over it is tied into the outcome of the player's actions. These actions persist through subsequent playthroughs, previous saves, and even erased saves, resulting in them permanently impacting the player's experience.

The big hook is that you can choose to never (physically) harm anybody. Over the course of a playthrough's 6-10 hour runtime, you'll encounter plenty of battles (random or otherwise), but you don't have to fight in any of them. You can always pick non-violent, contextual means of interacting with monsters—everything from petting dogs to initiating flexing contests with a douchebro horse man—and spare them in the end.

Main menu/Title screen

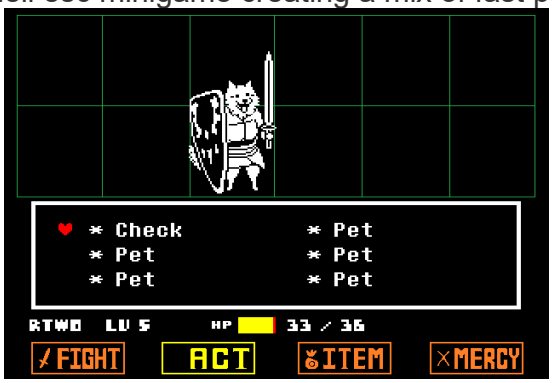
The main title has the player's **NAME**, **LEVEL**, **LOCATION** and **TIME** as un-interactive information and two simple options choosing either to **CONTINUE** or **RESET**.

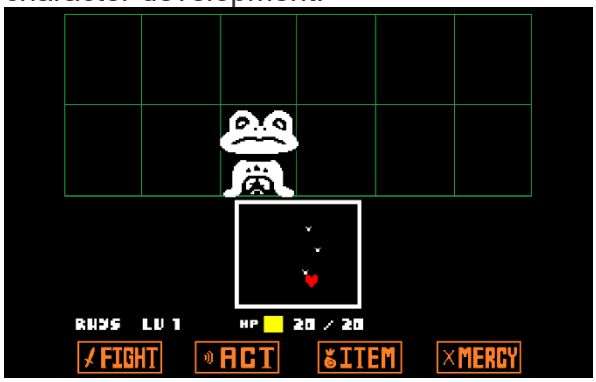


This minimal design is very elegant, requiring little inputs with simple outputs: start the game from where you left off (“continue”) or reset the game and start over. This could be useful to replicate for ease of understanding for the player. However, the lack of customisable options might make it worse for a wider consumer range as they cannot optimise the game for themselves.

Combat

The main focus of this game is centred around the interactions between the main character and all the entities surrounding them, the combat mechanic uses dialogue combined with a bullet hell esc minigame creating a mix of fast paced action and character development.





The combat user interface includes a **HEALTH BAR**, **ITEM INVENTORY** and an “**ACT**” button which initiates the dialogue options as seen above. There is also a “**MERCY**” mechanic which ends the combat peacefully after the dialogue puzzle is solved.

Enemies attack in a shoot-'em-up bullet-hell-like fashion, and you avoid their attacks. Then you whittle away at them with interactions or attacks. Which is your choice. Part of what makes Undertale so special is that these many interpersonal puzzles are more fun than combat. The game has a way of teaching you certain reliable techniques (if dialogue keeps changing, you're probably on the right track to solving the fight's puzzle), only to subvert them in sometimes unintuitive ways. Undertale is at its best when it's subverting itself. It just walks a very fine line between making you feel smart when you outsmart what is, essentially, the mechanical equivalent of an unreliable narrator.

World interaction

The world is also extremely interactive. For example these snow tufts each have dialogue boxes of the characters thoughts questioning the integrity of their existence until you progress to the next area and are stopped by a snow tuft which turns out to be an enemy.

I enjoy this level of exploration available as i feel it adds to the world, expanding things usually considered inanimate placeholders for items we are used to seeing in our everyday lives and showing them to be deeper than first expected.

Saving

Undertale's **SAVE** system is integrated into the gameplay. To save, the character must first reach a location of a **SAVE POINT** in the form of stars, which when interacted with; give the options: **SAVE** and **RETURN.** Pressing the return options leaves the dialogue box without saving. The dialogue box also includes the same static information as in the main menu.

The integrated save points mean that the game is only able to be restarted at certain points in the game's story, however saving at the same location after progressing is also possible.

This save system is unique as it contributes towards player immersion in the game world with a mechanic which would otherwise be considered a technical requirement.

The **ESCAPE** key options include only three options continuing with the minimalist design, **ITEM**, **STAT** and **CELL** as well as including a bunch of static information related to the gameplay such as **EXP** (experience points) **AT** (attack points) and your **LEVEL**.

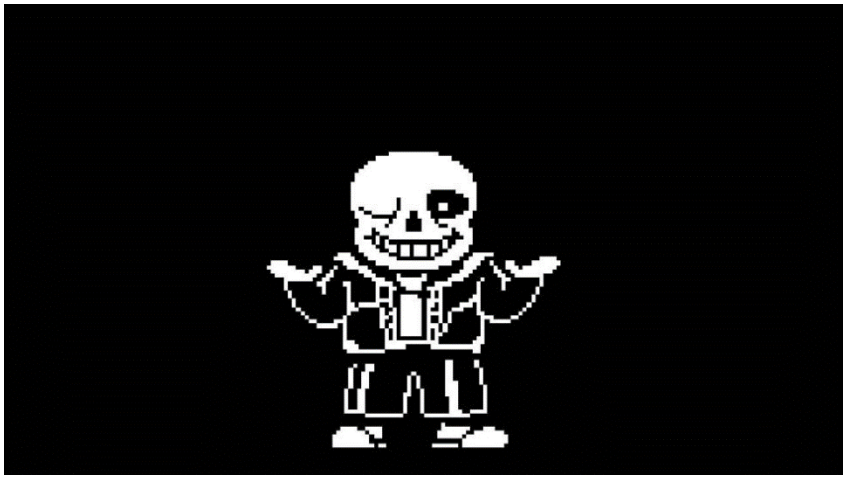
Dialogue

Dialogue within undertale is purely text based with no voice over. The dialogue appears on screen sequentially with repeating sound effects throughout to mimic the effect of speech. The main character in undertale is a silent protagonist making all speech within the game sub-character to sub-character or a monologue.

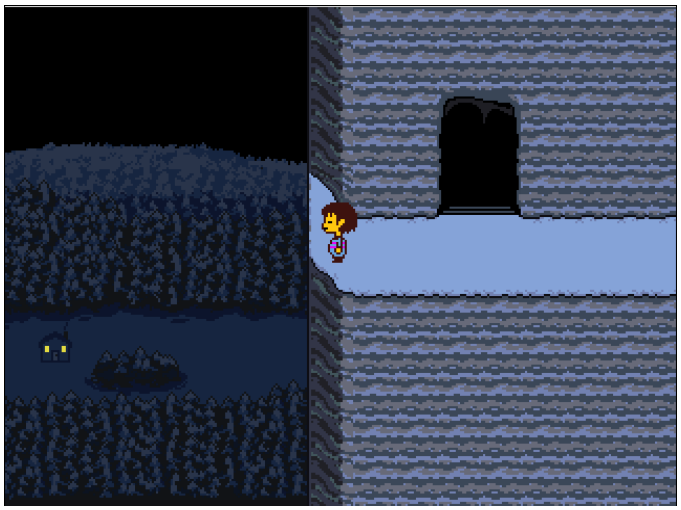
The silent protagonism affects the game world well, as it lets the player create his own unique points of view on the story instead of the game telling them how they feel in any specific instance. But when interacting with objects the player displays thoughts to explain what the object is and what it might mean to assist world building.

Graphics

Undertale is a top down 2D game purely made up out of pixel art, all characters are made up of a simple colour pallet when seen in the game world and during cutscenes are only black and white. The game has a blocky look to it created by the stark black outlines around all objects.

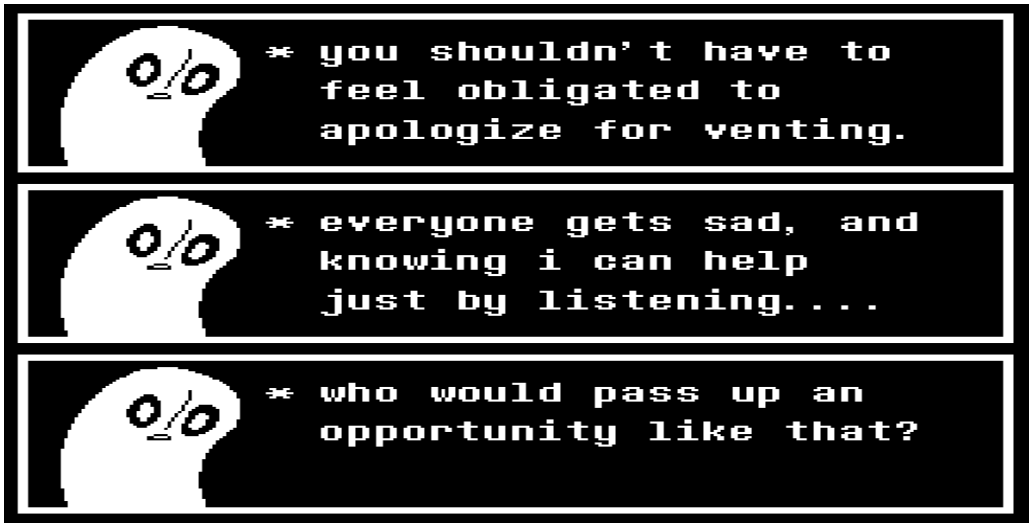


Outside of the playable area there is usually only black with very few areas using overviews of distant places to either foreshadow later in the game or create atmosphere, this solid black can sometimes create a claustrophobic feeling reminding the player that they are exploring a vast underground world hidden from the surface. The black outside makes the playable area vibrant with many areas using bright standout colours to contrast.



World Story and Characters

Undertale is a game about loneliness. The script does an excellent job at conveying this. The game world is a small enough place so therefore the NPC's interaction should be that they know each other by nature. But, at least initially, everyone is distant. Monsters are ensnared by problems that seem insurmountable. The world in their heads becomes bigger than the one right in front of their eyes. For example the encounter with the sad ghost "napstablook" has lines like these:



And then there's "Papyrus", who is "Napstablook's" polar opposite. He speaks in ALL-CAPS, ALL THE TIME. He's relentlessly upbeat as he viking-yells anecdotes from his life as an aspiring member of the king's guard—even as it becomes apparent that he doesn't really have any friends and sucks at his job. At one point you see the house that belongs to Papyrus and his brother, Sans. Sans' mailbox is overflowing with letters. Papyrus' is so empty that it echoes, hinting at the trouble this character faces about being liked.

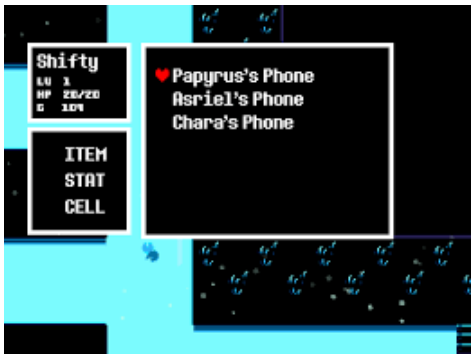
Other monsters come off as representations of anxiety, anger, repressed feelings, and self-doubt. It's hard to see at first—but all of these characters are coping, both with the larger tragedy of their situation and individual issues nearer to their hearts.

All these characters contribute to the tone of the world—which changes as you play when you spare them, giving them hope and a new attitude by solving their own personal dilemmas. Moving from a sad and enclosed world to a happy, vibrant and lived in one.

Call

Undertale has a "**CALL**" option in the **ESCAPE** menu. This is a mechanic which lets characters who you have already met, contact you through text dialogue. This is useful for worldbuilding as the silent protagonist can't enter a monologue about the area they are in, instead it is done by an acquaintance. They usually either have a

use to develop the story when they ring you, expand their characters (including when you call them) and explain a new mechanic such as lava.



The call mechanic also feels “alive” because when characters are either dead or not by their phones no one will pick up.

Conclusion

Points of interest:

- **Combat mechanics**—Do not take away from the story and have depth (MERCY option and dialogue puzzle)
- **Dialogue**—Text based with no voice acting like a book, useful for development speed.
- **Graphics**—Simple 2D bright artstyle
- **World interaction**—Objects and descriptions help develop the world. Increased development time.

Points of dislike:

- **Main menu**—Lack of optimisation options
- **Saving**—Does not let the user leave and come back when it suits

Overall UNDERTALE is a great game with charm, great story and innovative mechanics—such as the dialogue combat puzzles—which will be useful to take from and implement into my own game.

The Witcher 3----------

The Witcher 3: Wild Hunt is an action role-playing game developed and published by CD Projekt and based on “The Witcher”, a series of fantasy novels by Andrzej Sapkowski. It is the sequel to the 2011 game The Witcher 2: Assassins of Kings, played in an open world with a third-person perspective. Players control protagonist Geralt of Rivia, a monster hunter who is looking for his missing adopted daughter on the run from the Wild Hunt, an otherworldly force determined to capture her and use her powers. Players battle the game's dangers with weapons and magic, interact with non-player characters, and complete main-story and side quests.

The Witcher 3 has been praised by the gaming community earning a 9.3 on IGN 9/10 on gamespot and overwhelmingly positive overall reviews on its steam page. This huge success is deemed in every review to do with the huge immersive world and storytelling arcs. A quote from a PC world article published in june 2015 writes “The Witcher 3 is full of life, full of people commenting on your actions whether or not you pay attention, full of tiny stories and tiny details many players won’t even notice. It rewards exploration while maintaining a strong core story.”

Players of the witcher often find themselves straying from the main story quest for hours on end, traversing the game to the edges of its world and completing optional side quests because of the colossal amount of intrigue the hints of mystery in the stories presented to them. The time spent exploring however, is not wasted as it shows the player how vastly interconnected the game is—you can delve into the life of a simple NPC, read books left lying on countertops and understand their role within the world.

To emphasise this here is a snippet of a quest in the game:

A simple farmer has had his crops destroyed from a fight with a griffin (one of the game's monsters) which you had from an earlier quest and he now cannot afford to feed his family. In a desperate attempt to earn cash he turns to a gang of bandits to join their ranks, if you then explore further you come across this group hassling a lord at knifepoint. A situation you could have encountered at any time but now with the context about the bandit's background you may choose a different course of action; save the lord or let the bandits continue and pretend it's none of your business.

The Witcher presents the player with choice and context in vast explorable amounts which makes it one of the most immersive games of all time.

Main menu/Title screen

The main menu consists of **CONTINUE**, **NEW GAME**, **LOAD GAME**, **OPTIONS** and **EXIT** buttons with only the version number for static information. The continue and load buttons both load the game, the continue chooses the latest save, and load lets the player choose which save they want to start from. The new game button starts a new game, overwriting the old one and lets the player choose which difficulty they want. The exit button simply closes and stops the game and the options button heads into the settings menu.

The title screen also has a graphical rendering of Geralt, the main character sat by a fire shown with particles running along the screen and a bright orange glow. For a new player starting the game for the first time it sets an impression, the bird, camp and mountainous background hints at an exploration/adventure game and the Witchers sword, armour and scar hints at action and conflict.

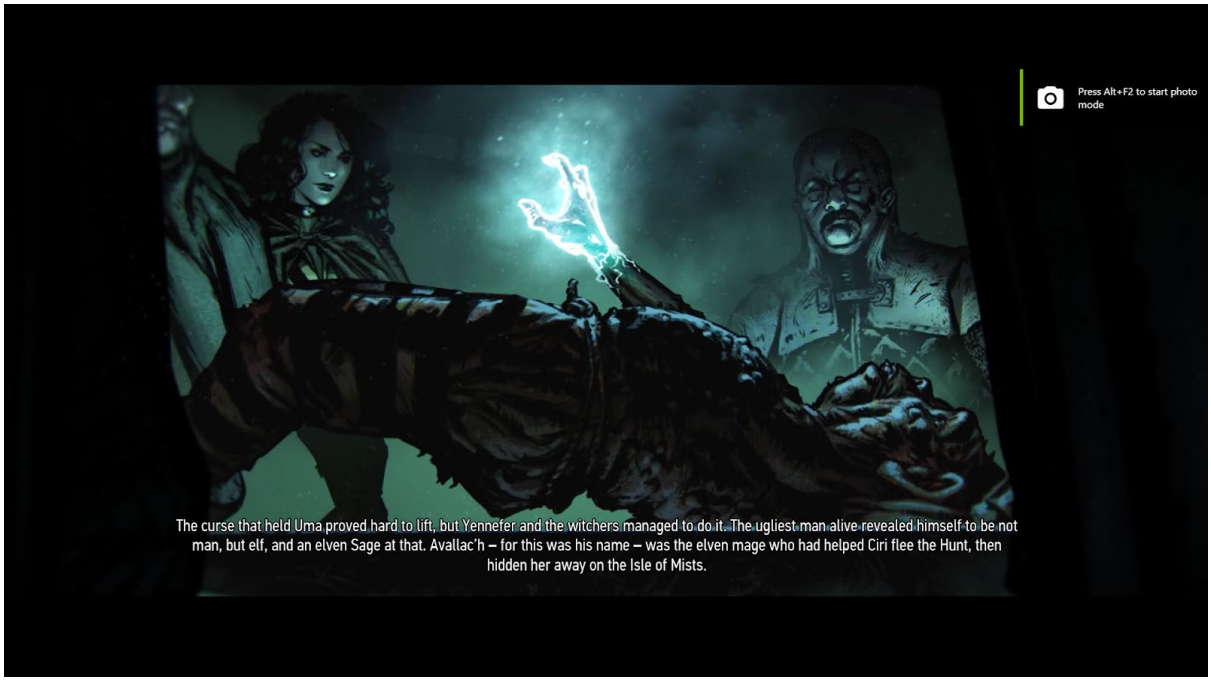
Settings menu

The settings menu has an immense amount of options. There is the first sub menu which includes the main parts of customisation. Audio, control settings, controller scheme, key bindings, gameplay, video and language.

Most parts of this settings menu would be extremely complicated to code. The languages for example would require a full translation of the game script into every language. This is also true for the key bindings and audio as it is an unnecessary addition to the workload when audio can be adjusted externally and the key binds will be optimal for the gameplay experience.

Loading introduction

When loading a save, the game passes the time by having a narrated two or three lines of dialogue explaining which part of the main story you are up to as a reminder. This is extremely useful if you might forget what you did last due to gaps between play sessions which is a problem my client might face being a full-time student.



Dialogue

The Witcher 3 uses voice actors for every single sub character you can interact with as well as having sub-titles in time with their speech. The NPC's you can interact with also have subtitles appear over their heads with mundane repeating dialogue.

Dialogue options

The witcher gives the player a choice of how to respond during a conversation, highlighted yellow text options progress the conversation where the bland brown text only includes extra information to the quest. Story quests have a set linear path the dialogue tree takes and when "important" decisions take place there will be a

time limit as to how long you can take to respond rushing the player and invoking a sense of urgency about the story immersing them further into the world.

Merchants in the game world are NPC's which you can return to at any time and therefore can't progress any dialogue. Their use is to buy and sell items from, to signify this option small icons appear next to the relevant word showing the player which option they are looking for.

Witcher Sense

Witcher sense is a game mechanic which highlights interactable objects. Whilst holding down the right mouse button on PC the screen slightly lowers the FOV(field of view) and shades the sides and muffles ambient noise. This creates the feeling of focus which gives explanation as to why this mechanic and how it might exist within the game world.

The orange highlighted objects are lootable interactions whereas the red highlights have relevance to any story quests active.

User interface

With such an immense game overflowing with content comes many things the user will want to know. The most important of those needing to be shown on screen at most/all the time.

Top left

- **HEALTH**—Combat systems require strategic fast thinking. The health bar is there to display how many more hits you can take before dying. This lets the player decide what to do: keep fighting, use a consumable or run.
- **TOXICITY**—The Witcher's potions are infinite with 4 uses at a time. To stop them from being used over and over in quick succession a "toxicity" mechanic is shown underneath the health bar which, if full, poisons the player and slowly depletes their health.
- **STAMINA**—The combat has a block option which reduces or completely stops all incoming damage, to stop this from being abused the stamina bar is depleted in time with blocked damage and when it reaches 0 the Witcher is stunned letting the enemy get a free hit. The stamina also acts as a timer until you can re-use your magic spell.
- **MAGIC SPELL**—The icon which looks like a glyph signifies which spell you have assigned to your cast spell button at the relevant time.
- **BUFFS**—Any active buffs appear underneath the health bar with a green circle timer showing how long is left.

Top right

- **MINIMAP** —The minimap shows an aerial view of your surroundings with icons of the local area such as dead bodies, quest locations, vendors etc… (the minimap is extremely useful in the Witcher as it is easy to get lost in the huge open world.
- **TIME**—Next to the minimap it shows the time and weather which is useful as some quests require a certain time of day to complete, for example a stealth mission must be done at night when the guards are sleeping.
- **ACTIVE QUEST DISTANCE + INFORMATION**—The number of meters to the tracked quest is shown to the left of the minimap and a brief overview of that tracked quest is displayed underneath as well as which part you are completing at that current time.

Bottom left

- **CONSUMABLE ITEMS**—The currently equipped items which you can use on the fly are shown here.

Bottom right

- **TIPS** and relevant **CONTROLS**—When playing a specific part of the game or hovering over an interaction, the buttons useful to that situation will pop up here as a small reminder of what to press. This is shown with text and small images.

Combat

The Witcher is a monster hunter so it makes sense for the game to have a large portion of combat. Within the world many random encounters may happen by just stumbling upon them, you might be exploring a wood and walk into a pack of wolves feeding on a corpse and decide to fight them to loot the body for yourself.

The combat system is a real time mechanic which uses skills and melee combat:

- *(Left click)* **light attack**—The light attack is a fast attack which deals moderate damage with a short charge up time of around 0.05 seconds.

(Light attack)

- *(Right click)* **heavy attack**—The heavy attack is a slower hard hitting attack which deals significantly more damage than the light attack but with a slower charge time of around 0.125 seconds leaving you open to attacks in the meanwhile.

(Heavy attack)

- *(Alt)* **dodge**—The alt dodge is a short, quick step in the direction you are moving which you can avoid incoming enemy attacks


(Sidestep dodging to the right)

- *(Space)* **roll**—The roll is another dodge mechanic, it covers a wider area but it takes longer to attack after doing so making it useful mostly for repositioning or retreating


(Rolling away)

- **Auto lock on**—during combat there is a soft lock on feature meaning that the player attacks either towards the closest enemy or the direction they are facing if none are near enough helping the player to not be surrounded when facing multiple enemies.
- **Weapon type**—The witcher holds three weapon types, Silver sword for monsters, Steel sword for men and a handheld crossbow for underwater combat or hunting (the underwater combat is unrealistic of real life physics).


(Shooting the crossbow)

- **Spells**—The witcher is not a sorcerer but can cast mini-spells called signs. These include:

  AARD *(a knockback wind push)* **IGNI** *(a small fire blast emitted from the hand)* **YRDEN** *(a magical trap for wraiths and ghosts)* **QUEN** *(a personal protective shield)* and **AXII** *(a mind control technique)*


(Using IGNI in combat)


(Using AXII in combat)

(Using YRDEN in combat)

Enemy design

The Witcher, being an already developed book series, has a host of enemies with set backstories and is able to design monsters out of descriptions.

The bestiary is a menu in the witcher which explains monsters you have encountered as well as their weaknesses.

Most monsters are loosely based on norse and greek mythology, where all the creatures are mixes and mutations of known animals such as the griffin, a lion with wings and talons for feet.

This mystical look of the creatures is something I enjoy about this game as it is evident in showing the fantasy setting. In particular I enjoy the look of these monsters:

Graphics

This game uses a 3D realistic art style with excessive use of ambient occlusion lighting and shrubbery making the game have a lifelike beautiful effect with sun rays shining through the trees into dark forests and sunsets over mountains casting huge shadows over castles in the valley.

The artwork and work needed to implement this 3D graphics style is extremely expansive and is something I would not be able to add into my project.

Conclusion

Points of interest:

- **Main menu + Save system** — The main menu shows exactly what the game is about and interests the player and the save system is as easy to pick up as a book, exactly as the client wished.
- **Witcher sense** — The witcher sense mechanic was integral to the mystery problem solving that the game presented and I enjoyed the premise of a muder mystery style backwards storytelling approach.
- **Loading introduction** — The short narration within the load screen helps set the scene for the next play session which could be useful within my storytelling game.
- **Dialogue** — The script in the witcher is extremely well crafted and the optional choices leading to consequences may be difficult to implement but will present the player with a rewarding story aspect.

Points of dislike:

- **Settings menu** — The settings menu is extremely complex for incredibly precise optimisation but is not easily used by relatively new, casual players.
- **Combat** — The witcher's fast combat proceeds nothing towards the story so wouldn't be useful to implement into my own game
- **Enemies** — Although the enemies in the witcher are unique in design, the stylistic design they have might not work with a 2D setting
- **Graphics** — Above my abilities and timeframe to implement

Hollow knight----------

Hollow Knight is a 2D Metroidvania action-adventure game, which takes place in Hallownest, a fictional ancient kingdom. The player controls an insect-like, silent, and nameless knight while exploring the underground world. The knight wields a nail, which is a cone-shaped sword, used both in combat and environmental interaction.

Hollow Knight's initial release received "generally favorable" reviews and the Nintendo Switch version was met with "universal acclaim", according to review aggregator Metacritic. Jed Whitaker of Destructoid praised it as a "masterpiece of gaming ..., and certainly art worthy of being in a museum" and, on PC Gamer, Tom Marks called it a "new classic". Reviewers spoke highly of Hollow Knight's atmosphere, visuals, sound and music, noting the vastness of the game's world.

Elegy

“In wilds beyond they speak your name with reverence and regret,

For none could tame our savage souls yet you the challenge met,

Under palest watch, you taught, we changed, base instincts were redeemed,

A world you gave to bug and beast as they had never dreamed.”

An elegy is a sad poem of reflection, usually written to praise and express sorrow for someone who is dead. The purpose of this kind of poem is to express feelings rather than tell a story.

The first verse of the elegy of hallownest is shown at the very start of a new game. It acts both functionally showing the player what their quest and purpose in this world is and narratively, setting the tone and feel of the world.

Lore

Summary:

Within the world of Hallownest, higher beings are primordial, god-like creatures that exist above all others, possessing certain abilities and powers that regular bugs do not have. Higher beings were worshiped as gods by other bugs. Before the start of the game, a higher being known as the Wyrm had died near Hallownest in order to transform into a new shape. His name is the Pale King. Before the Pale King's arrival, the bugs of Hallownest worshiped a Higher Being known as the Radiance, a powerful moth whose mere presence could sway the denizens of Hallownest to mindless obedience; however, the Pale King was able to expand the minds of the bugs of Hallownest, leading them to accept him as their ruler, and leaving the Radiance as a distant memory. With his newfound control, the Pale King went on to construct a great kingdom that sprawled over and throughout the lands of Hallownest. However, after some time, the bugs of Hallownest began hearing the whispers of the Radiance in their dreams, and this soon led to an infection of madness and mindlessness that affected many. In order to combat the growing threat, the Pale King created shadowy beings called Vessels that lack a mind and will, and he attempted to contain the Infection by sealing it within a single hollow Vessel. After many attempts, a suitable Vessel, called the Hollow Knight, was created by the Pale King and used to contain the Infection. The Hollow Knight was placed within the Black Egg Temple and sealed by three powerful elder bugs known as the Dreamers to be locked away forever.

At the game's outset, the player character, the Knight, arrives in Dirtmouth, a small town above Hallownest's ruins, seeking to venture within. As it journeys through the forgotten kingdom, the Knight encounters the possessed remnants of Hallownest's former residents and other creatures, who are slowly being overcome by the mysterious Infection. It also encounters Hornet, the protector of Hallownest's ruins, who attempts to stop the Knight's quest on multiple occasions. Through learning the history of Hallownest on its journey, the Knight discovers that it is a failed Vessel. It also becomes clear that the Pale King's attempt at sealing away the Infection using the Vessel known as the Hollow Knight did not work and would soon completely fail, allowing the Infection to consume all of the remaining inhabitants of Hallownest.

Hollow Knight has multiple endings. In the first ending, called, the player defeats the Hollow Knight and absorbs the Infection into itself to become the new Hollow Knight. As chains manifest to hold the Knight in place, the Temple is once again resealed and the plague is held at bay, though its true source still survives. The second ending occurs if the player collects an item called the Void Heart before fighting the Hollow Knight. Halfway through the battle, Hornet arrives to offer assistance and briefly holds the Hollow Knight at bay. Choosing to continue fighting normally causes her to be knocked unconscious. The ending then plays out like the first, except Hornet is sealed in with the player Knight and her mask is used as the new Dreamer seal upon the Temple door.

The third ending occurs if the player collects the Void Heart and uses the Awakened Dream Nail ability to enter the Hollow Knight's mind when Hornet arrives to help. Here, the player challenges the Radiance, the moth-like goddess who was the true source of the Infection and who sought to return to power in Hallownest and eliminate the Pale King's influence. With the power of the Void Heart, the Knight commands the forces of the Void, with the aid of the Hollow Knight and the shades of the failed Vessels, to consume the Radiance. As the darkness from the Temple fades, Hornet awakens to find the chamber empty, except for the player Knight's broken and empty shell.

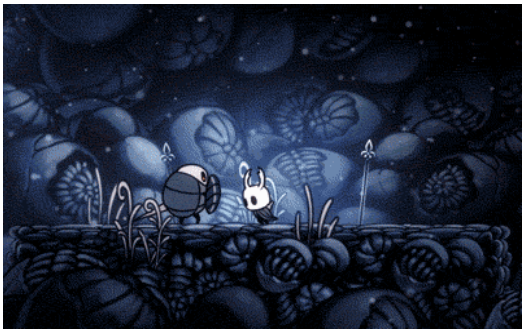
The exploration of hollow knight allows the player to travel around the world in any way they want to collect the item which would give you a better ending. This is something hollow knight does constantly, it gives the player information and reasons to discover but doesn't force it into them giving the player freedom instead of constraint. This is something worth considering adding into my own game.

Core mechanics

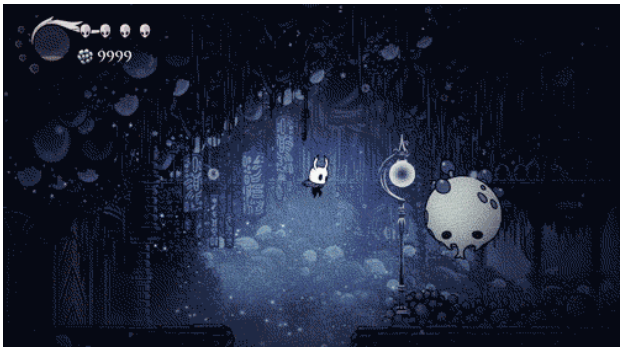
The movement mechanics in Hollow Knight change and develop as you progress through the game. This happens through the discovery of items such as the monarch wings which allow a double jump. This progression is locked behind terrain which cannot be passed until the correct movement mechanic is acquired, this also lets the game set barriers to story progression to implicitly push the player in the right direction.

You start the game with an initial moveset for platforming and combat:

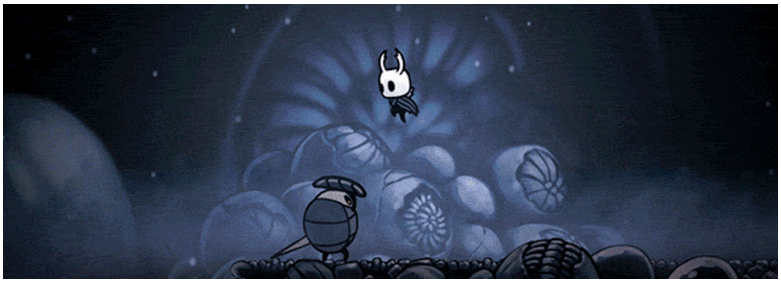
- **Walking** (S +D) — Hollow knights movement is linear, moving from left to right (up and down is done with a gravity mechanic) therefore in the WASD fashion the player is able to move along screen both to the left and to the right.



- **Jump** (space) — Hollow knight has many platforming elements to the game as well as floating enemies and hazardous drops to overcome. Pressing space makes the knight jump up into the air until gravity pulls him back down again. You are also still able to move whilst in midair for directional control.

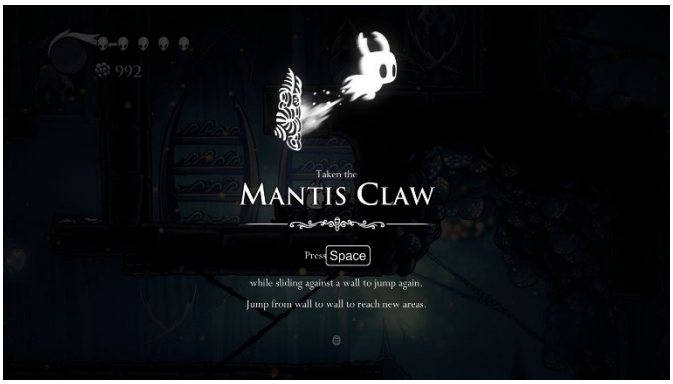


- **Bouncing** (left click) — Attacking enemies or spikes whilst in midair will give you a secondary boost as if you are ricocheting off of it.



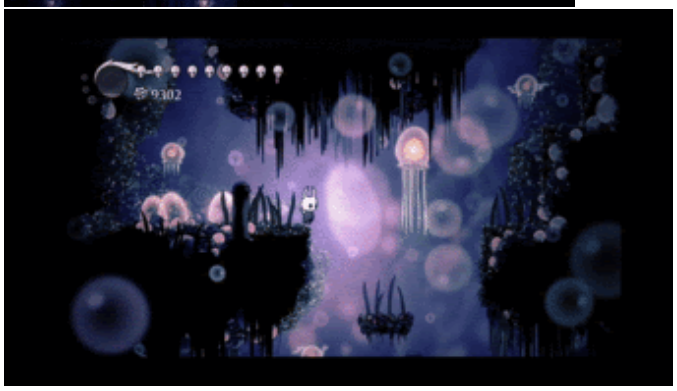
Then the game upgrades to more developed movement:

- Wall grab (Mantis claw)—While close to a wall while in midair, you can press in the direction of the wall to cling to it and slide down. Pressing JUMP to launch the player diagonally away from the wall. This can be repeated an infinite amount of times, allowing The Knight to climb up walls.

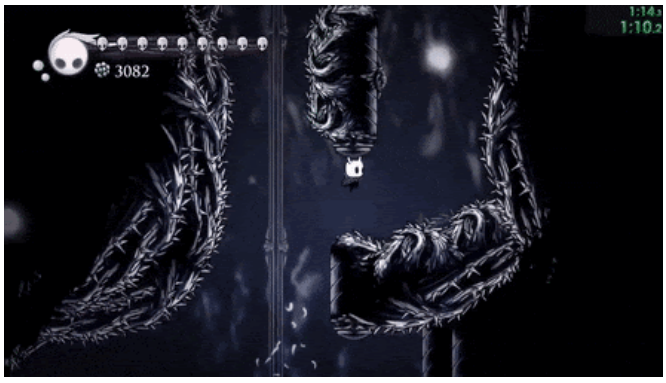


- Dash (Mothwing cloak) - (Shade cloak)—Press DASH to dash a short distance in the direction the Knight is facing. This can be used in mid-air. It refreshes when the Knight lands on the ground, takes damage, bounces off an enemy/object with their Nail, or clings to/jumps off a wall using Mantis Claw (first picture).

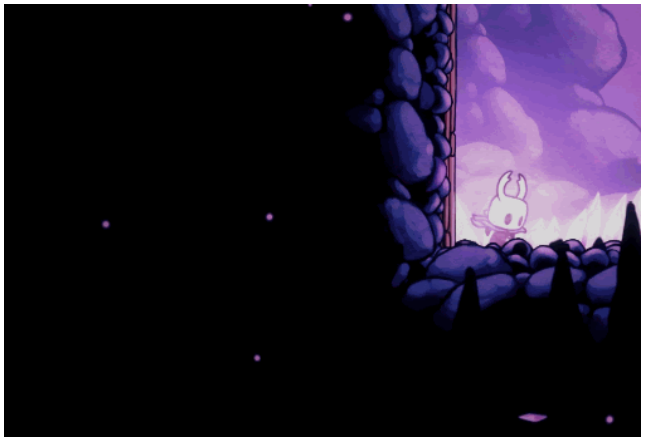
The Shade Cloak (second picture) acts as an upgrade to the Mothwing Cloak. It allows the Knight to dash through enemies and their attacks, as well as Shade Gates. Explosions and environmental hazards will still deal damage. Dashing through most projectiles will also destroy the projectile.



- Double jump (Monarch Wings)—Pressing JUMP while in the air to perform a double-jump. The jump refreshes when the Knight lands on the ground, takes damage, bounces off an enemy/object with their Nail, or clings to/jumps off a wall using Mantis Claw. Similarly to the regular jump the height of the second jump scales with how long the JUMP button is held.



- Launch (Crystal Heart)—Hold SUPER DASH (E) while on the ground or clinging to a wall to concentrate force. Release the button at full charge to blast forwards and fly through the air. the Knight will continue moving forward until interrupted by damage or the environment, by pressing JUMP or by tapping SUPER DASH again.



Other gameplay mechanics

- Knockback occurs when the player character is damaged throwing them back into the opposite direction and interrupting their movement for a brief moment.

- Fall damage is not present in hollow knight even with a gravity like fall mechanic meaning that puzzle and platforming sections feel less harsh.

Artstyle

The world of Hollow Knight is alive with its vivid, moody detail and its bizarre and terrifying creatures, each animated by hand in a traditional 2D style. The look of Hollow Knight is reminiscent of a 2D animated movie. Vibrant characters, creatures and detailed environments draw players deep into the game world. Each new area discovered in the players travels is unique, strange and teeming with interesting creatures. Hollow knight is a game designed for exploring, even just to take in the sights or discover hidden secrets off the beaten path. The art and gameplay intertwines to create a powerful sense of wonder and discovery as you venture further and further into the game.

Animation

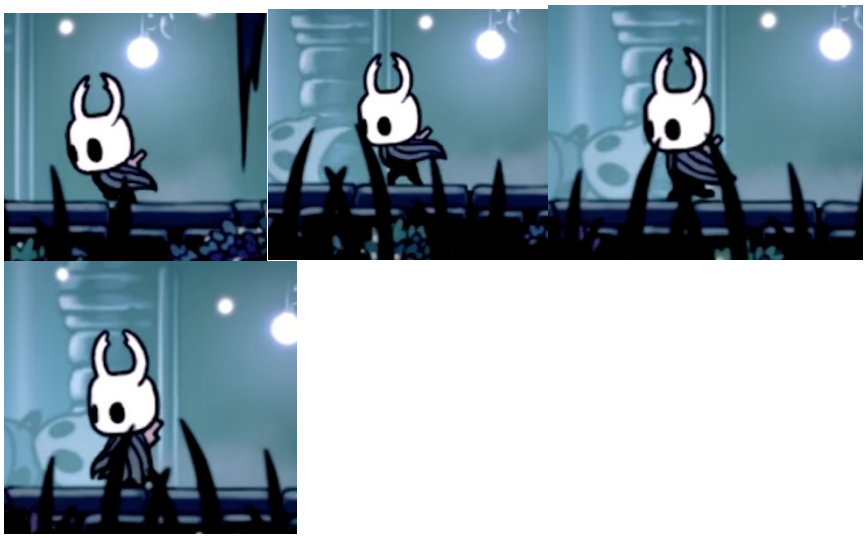
Hollow knight is a 2D side scrolling game which meant that the game devs had to focus on having their animations for running and jumping perfect. The jump in hollow knight takes inspiration from the tight control scheme from games like Mega Man X, in being able to control your character midair to pixel-perfect precision. The running action became an extension of this design philosophy. A simple way to explain it would be that the form needed to fit the function.

For instance, when you swing your nail, the action happens immediately, consisting of one tiny frame of anticipation, with the next frame being the full arc of the swing. The bright smear which represents the swing is filling in the gaps of what you didn't see. But, while the form is serving the function, the form isn't lifeless.

Let's look at running and jumping to start. They too, are simple, basic actions but there's tiny little moments of charm. Whenever you start running, the character does this slight little lean into the direction its moving, like a little indication of effort.

Your actual horizontal traveling speed is constant, so as not to mess up the tight gameplay, but just that little addition gives your avatar a little more life. Also, you have a small turning animation if you start running the opposite way you're facing, which isn't common in many games, but adds a great effect.

The running animation is subtly versatile, too. If you stop running while you're in that first leaning part, you'll gradually straighten back to your starting pose,



whereas if you stop while you've been running for a while, you're already in that sort of pose. It's just little moments like these which aren't noticeable, really, but they're there, smoothing out all of the action.

What I love with the jumping, aside from the great little follow through animations which you get to enjoy throughout the whole motion, is the dust that follows your jump arc.

It's made up of both hand drawn frames and more physics based, generated particle animation. You might be quick to assume that there's two dust animations for when you jump; one for straight up, and one for when you jump when running. But, upon closer inspection, you'll see that the dust follows your exact path. And this is because the dust trail is a generated effect in Unity.

The animation in Hollow Knight, generally, is fairly simple. There's nothing extravagant, there's nothing particularly special, but it's consistent and confident. Any character you interact with, whether it's friend or foe, is always rendered with a thick black outline, bold colours and tends to move with restraint. Enemies tend to have more anticipation to aid in telegraphing attacks, so you get lots of nice, striking battle animations,

Whereas NPCs tend to have more thoughtful staging, like this lovely jubbly character, your map maker who you can hear humming to himself in the area, the bank teller, the fast travel stag, and many, many others. They all tend to look at you when you speak to them or move past them, too. Which is a really nice thoughtful touch.

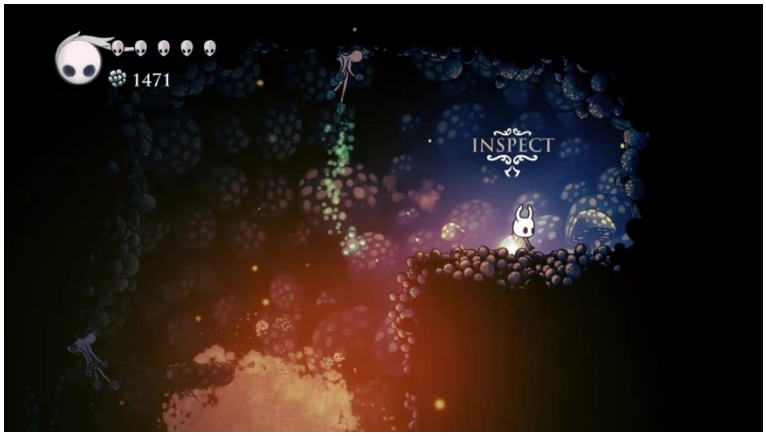
A lot of things in hollow knight’s design move at very different frame rates, depending on whether it's in the foreground or the background. One of the first examples is from kingdom's edge; the leaves are a particle effect that acts as an overlay to the game these leaves move pretty steadily and fluidly making it easy to see their falling movements seamlessly.

On most computers this will be displayed at 60 fps. But when you look at the characters specifically your character, it is rendered out in a noticeably slower frame rate than the rest of the game (which is in 60fps) this is not just your character but every single NPC you can interact with and any enemies you can kill. The enemy's being rendered in the same frame rate as the player character is an obvious design choice by the game devs helping the player to recognise that you are probably able to kill or hit it. Furthermore if there is an inanimate object in the game light will usually be shining on it at, again, the same frame rate as the characters so to subconsciously hint to the player that it is interactable.

Hollow knight's animation achieves a lot through a little.

Colours

Each area in hollow knight comes with its own unique color pallet. The fungal wastes has a prominent orange theme inundated with variations of yellow and likewise colours to give the appearance of natural fungi.



The queens gardens on the other hand has heavy contrasts of greens and reds. The **greens** make it immediately apparent to the player that they're entering an area with a surplus of vegetation

whilst the **reds** communicate challenge and danger, which is fitting because this area contains some of the most convoluted platforming segments and some of the hardest enemies to kill.

The colour scheme of each area is recognisable enough to where you probably don't even need the text which appears when you enter to help you out (shown below).

This is indicative of exception colour usage and it is something that art directors of this game obviously had quite a grasp on. A colour pallet of the game itself helps

you with navigation because when you're about to enter another area the music sometimes gets silent and the colours are slowly transitioned from one pallet to another.

The infection (the main disaster of the game) however, is represented by a bright citrus orange, an orange that if it had been used anywhere else would look friendly but instead, as a stand out colour, it is one of the most ominous looking things in the game.

Seeing warrior husks with their insidious manic orange glow in their eye or seeing the infection spread like a fire in the crossroads is something that fills the player with dread. Artists and designers will use complementary colour schemes to give the appearance of imbalance which is the ring of harsh contrast between two colours which are opposite sides of the colour wheel.

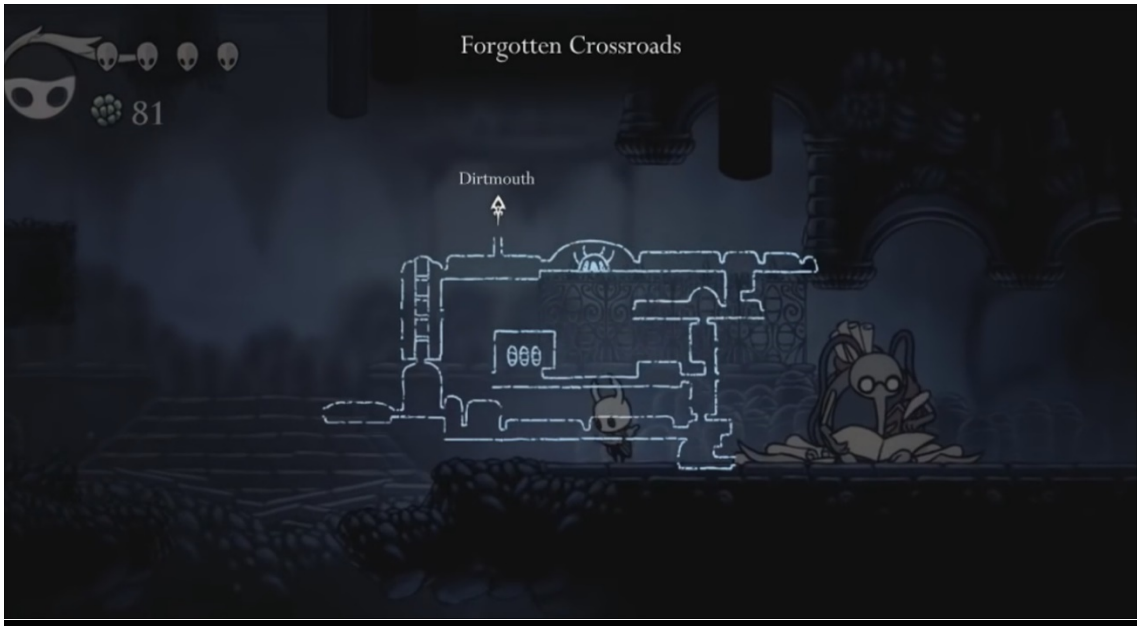
(before the infection)

Using this colour scheme can sometimes create an off putting sensation, a sensation that immediately tells you “something is wrong here” which is exactly what the infection does to the player. It contrasts the orange with the dull blue environment in the crossroads to give the player the effect.

Map

The map in hollow knight has a variety of features. To start, when the player accesses the menu by pressing the TAB button the menu hovers over the screen as an overlay. During this the player is still able to move left and right slowly keeping the game's fluidity.

The map is also integrated in the game's story, exploring the caves of hallownest is confusing and the player is sure to get lost. In the initial area of the game, the forgotten crossroads, the player encounters an NPC named Cornifer surrounded by parchment. Cornifer offers to sell the player a map of the area they are currently exploring. This map contains vague outlines of the rough area which when explored become more detailed.



Cornifer moves from location to location as the game progresses selling players maps to the new locations, if the player progresses past an area without using a map and cornifer moves location it can be acquired from the start town letting the game flow naturally with the players random movements and progression paths.

Geo

Geo is the main currency of the game. It is collected by defeating enemies, finding relics or as a reward for various goals. It can be spent in shops to buy charms, items, etc. or to unlock benches/stag stations.

At the beginning of the game, geo is scarce and there are many things you need to buy. The player is presented with many shops with crucial items (quill, compass, geo magnet, extra HP) and enemies that drop only a few geo. This can make the early game somewhat tough as the player is missing crucial mapping tools when they are needed the most as they are still getting used to the game.

In the midgame, geo is used well to soft-lock areas with the expensive lamp and limit your health and soul until you play more and naturally collect more geo.

In the late game, players could have 20,000+ geo and nothing to spend it on as there are no expensive items by the end.

The player loses all geo upon death, but they can retrieve it by going back to the place they died and killing their shade:

This is a good system, punishing the player by taking the time to run back to the place they died. Losing a third of their soul storage is also necessary as it requires the player to go back. I might argue that losing all geo is not too necessary. In the late game, the player faces virtually no danger in going back and recovering geo. On the other hand, players only learning the controls are set back even more on the essential items they have to buy. I would be interested to see what would happen if the geo loss was partially or fully removed. A better option would be to have an easily accessible bank where the player can store geo.

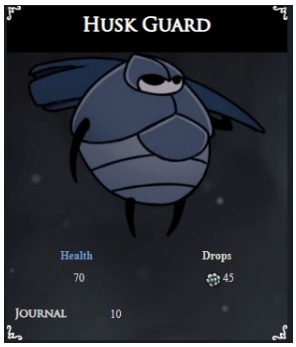
Enemies & Bosses

Enemies

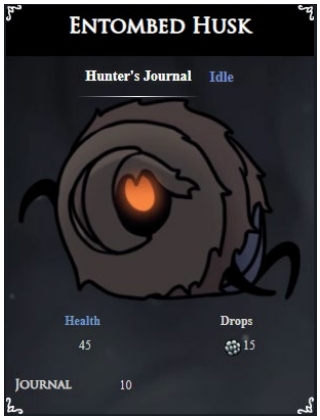
Hollow Knight has a good variety of different enemies that challenge the player to try new strategies and techniques. While the variety is a bit deceptive as there are a lot of reskins, there are enough to keep the game interesting. Here are some thoughts about certain enemies that I would like to highlight.

Crossroads Husks: these are the basic enemies in the first area of the game. It is good that they are simple and easy, but at least one of the three “run at the player” enemies could use a new mechanic.

Husk Guard: these large foes not only have a fast, long strike but deal two damage. Even having completed the game, I struggled with these enemies more than False Knight. Their damage should be lowered to one mask, heavy-hitting enemies are for the late game.

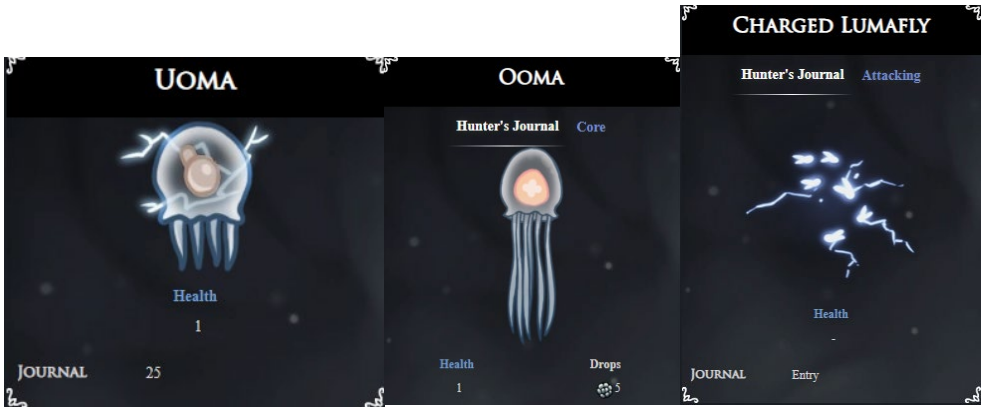


Entombed Husk: the first time I encountered them, they scared me more than Deepnest. In the dark underground, their panting is frightening - great sound design.

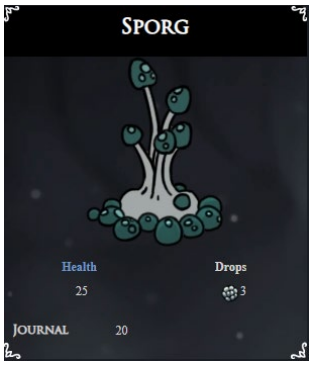


Volatile Mosskin/Fungified Husk: these enemies can be boring at first – hit, move away, hit. If you are more experienced, you can sneak in two or three hits before they explode, making this enemy a great game of risk-reward.

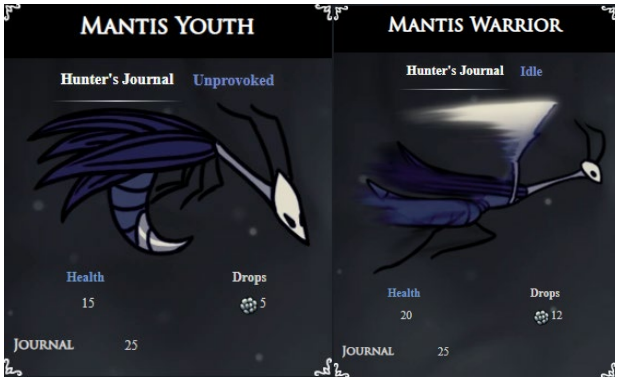
Uoma, Ooma and Charged Lumafly: these ‘enemies’ float around passively and the Ooma are clearly meant to be avoided. I would even consider adding some extra mechanic to make the Fog Canyon even more dangerous as it is still relatively easy to traverse.



Sporg: They take four hits to kill, are often in hard to reach positions and shoot homing explosives hat deal two damage. A tough enemy because of positioning.

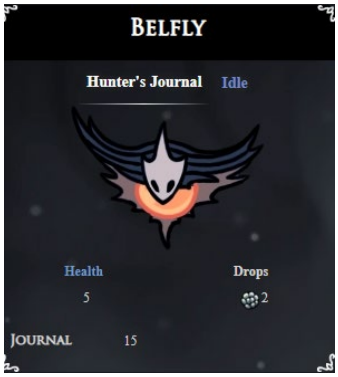


Mantis enemies: these are good enemies and I enjoy their lore, but–there are only two variations and can become overused

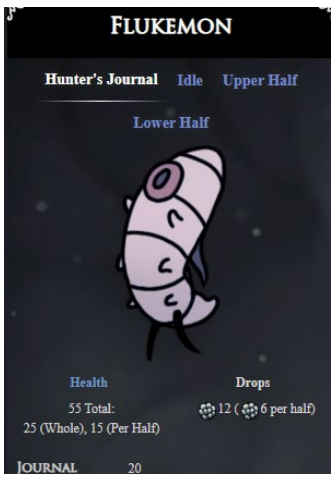


Soul Twister: They follow you around and teleport constantly, making them hard to hit. If you get close, their homing missiles are almost guaranteed to hit you. Their mechanics make them extremely difficult to kill.

Belfly: not very fun or engaging. They are either a frustrating surprise two damage or you can just walk past and they kill themselves.

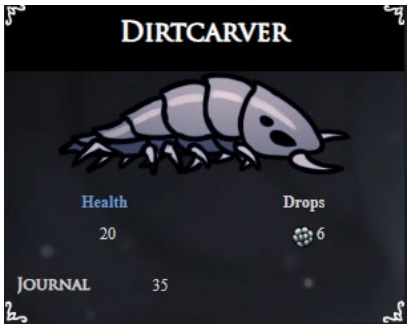


Flukemon: a fast and maniacal enemy that really supports the dangerous atmosphere. I like how its split body comes back to attack you.



Furious Vengefly & Volatile Gruzzer: they are supposed to be infected versions of the Vengefly and Gruzzer, but for me, they seem too large compared to the Violent Husks, who did not change in size. I will also mention again the lack of infected horned husks.

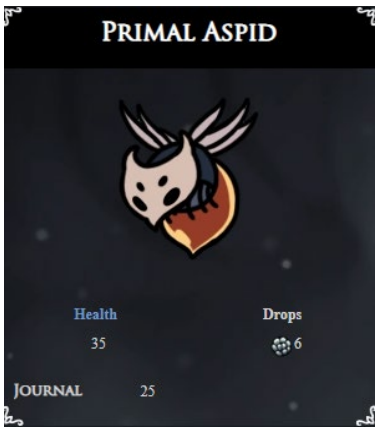
Dirt Carver: the trademark enemy of Deepnest, they ambush you by burrowing underground. The oppressive atmosphere and the feeling of no escape is supported by the leap and wall-climb abilities.



Shadow Creeper: is the only enemy found in the basin but it is not challenging or creative enough for a mysterious place like this.

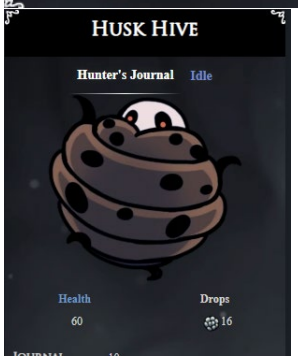
Infected Balloon: these are alright enemies, but the happy fat look doesn't fit with lore and atmosphere of the dangerous infection.

Primal Aspid: These enemies are extremely frustrating: they shoot three fast projectiles with little warning. I think maybe making them shoot more frequently but with a slower telegraph would change this.



Hiveling: it would have been nice to see more of these small enemies. Bees move in large swarms and it would have been more atmospheric if 30 or so attacked the player at once.

Hive enemies: a great combination of unique larger and smaller foes and a fun breakable platform mechanic.



Loodle: these enemies are too frustrating. They are tedious and random, which is not a great combat experience.

Mantis Traitor & Petra: I like how these enemies are evolved from the ones in Mantis village, which even fits the lore of the infection. They are difficult, but engaging to fight.

Overall, a good variety of enemies with some interesting mechanics. I enjoyed finding and taking on new enemies when I explored new areas. The combat is definitely one of the strong sides of Hollow Knight.

Bosses

I loved a lot of these unique and fun battles, most of which took some time to learn and master. The game has many different, fun bosses that wow you with their dramatic entrances, unique styles, cool abilities and immersive atmosphere and music. They are also a good example of the number of skills you learn playing this game. After I beat the second, more difficult version of Hornet, the first fight feels easy and predictable.

Unlike some other games, the bosses don't take up all the screen and dodging through attacks is not the only focus. The interwoven systems of offense and defence, risk and reward and the integration of all the combat systems make these fights all the more addictive.

Most bosses have a stun mechanic that is quite complicated to understand. To put it simply, most bosses will be knocked down for a few seconds if they are hit enough times. This is an appreciated and sometimes necessary break to heal or deal extra damage if you are already doing well. I do wish the stun took a bit more time to take effect: I missed the animation many times and accidentally hit the boss, ruining my healing opportunity.

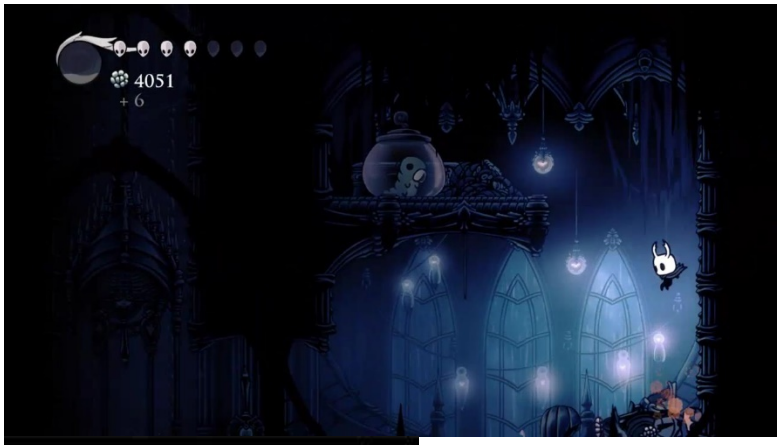
Bosses also lack any sort of health bar in this game. There is some indication with staggers, but it is not very reliable. This is not an issue with smaller enemies who take a few hits to kill, but it can change the mood of a boss fight entirely. The uncertainty the player has of the amount of health a boss has makes it all the more gripping and tense.

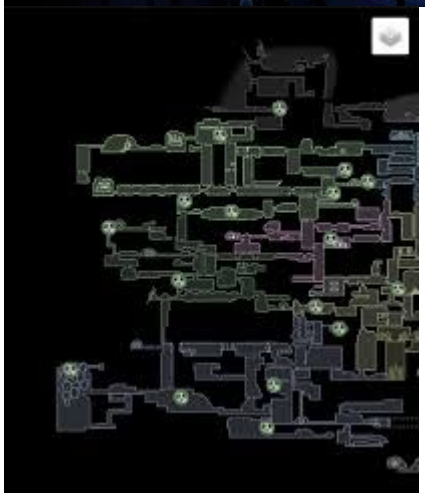
Collectibles

Hollow Knight has a vast underground world with around 15 areas for you to explore. These are all meticulously designed, with branching paths, hidden areas and many secrets.

Many players enjoy getting lost in the caverns and wandering around until they stumble upon some hidden-away treasure. This can be really fun but also tense as you have no idea what will be in the next room. This feeling of mastery when coming back to older areas with more experience and stronger gear was also great. Hollow knights often surprise the player by finding even more secret areas with my new abilities.

Grubs and relics are a good example of how exploration is done well in Hollow Knight. Grubs can be found sometimes hidden, but mostly with a small challenge. This is usually platforming or enemies, but there are also some unique challenges. I estimate that most players rescue about 2/3 of grubs before the late game. After enough items are acquired, a boss can be fought, after which the approximate locations of the grubs are revealed on the map. I find this a perfect exploration system: the player finds most secrets with challenges and after enough exploration and a battle, they can track down the remaining secrets more easily but will still have to complete the challenge.



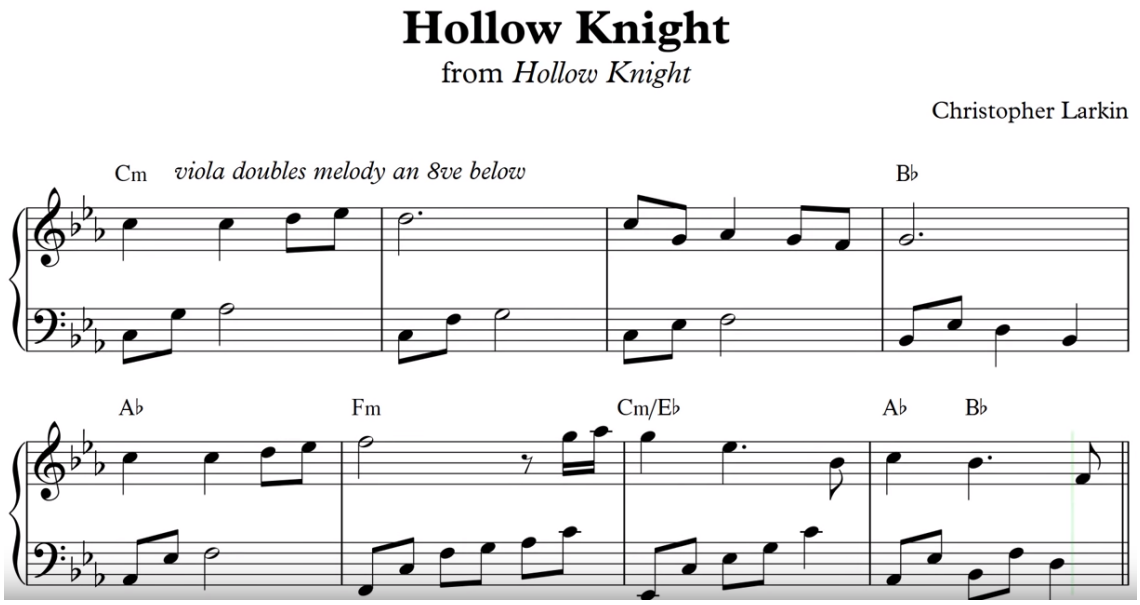


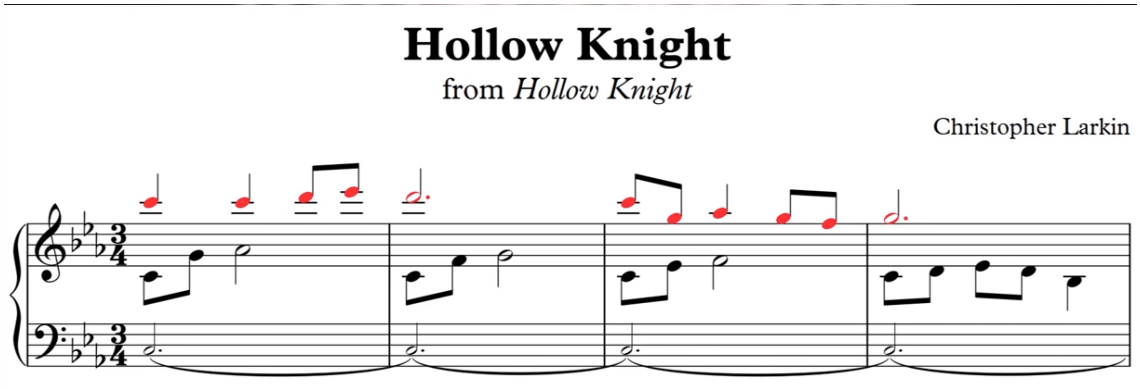
Relics are the game’s way of rewarding extra exploration and curious players. Players that wonder what would happen if they wall-climbed up the well or pogoed on a spiked roof will be rewarded with some extra geo that they won’t lose when they die. These relics can be hidden anywhere that would be too obscure for a normal player but are a good reward for the dedicated explorer.

Sound design

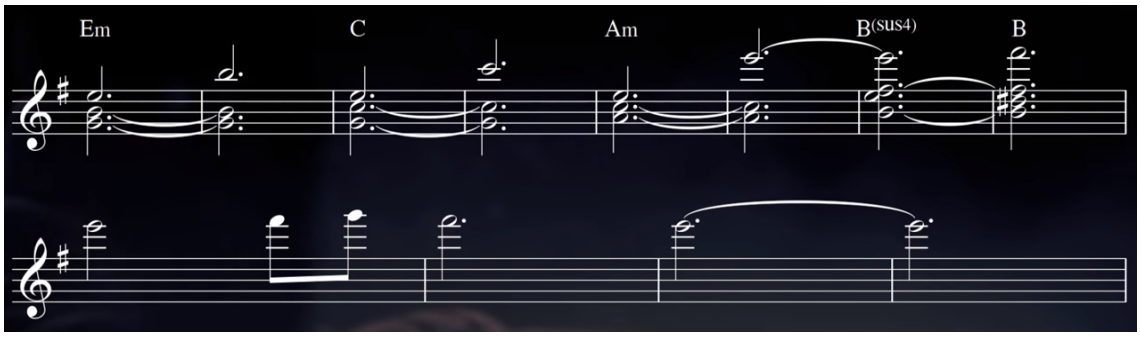
Hollow Knights soundtrack develops themes across multiple pieces of music. The consistency of the soundtrack’s atmosphere and use of leitmotifs to reflect specific characters and places is in perfect balance with the wide variety of styles and arrangements that adapt to different parts of the game, making it feel like you're experiencing one huge living piece of music that grows and evolves around you as you explore the game's world.

The piece named Hollow Knight on the official soundtrack, and arguably the main theme of the game, is the music that plays over top of the main menu. This might make it seem unimportant, but as the very first thing that you encounter when you boot up the game it serves a vital role in setting the tone for the adventure you're about to embark on. The instrumentation of piano and viola duet coupled with the slow tempo and natural minor harmony, set the stage for an intimate yet solemn dark yet elegant adventure.



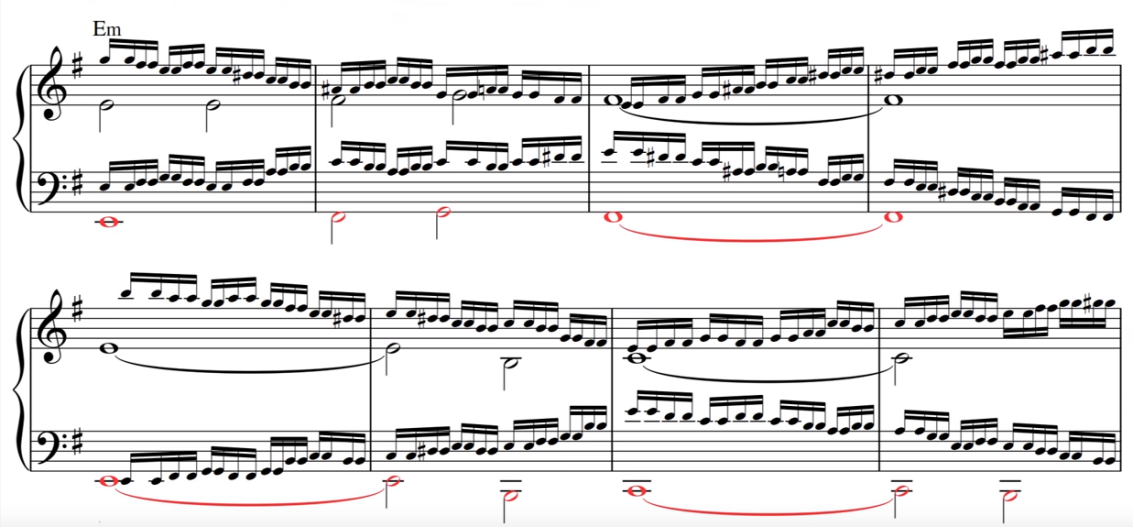


Here, the melody develops out in an extended call-and-response pattern. The first two phrases work wonderfully as leitmotifs for two reasons, they're simple and they're recognizable. For a motif to work in multiple musical environments it needs to be malleable enough to fit into a variety of settings. To function as a leitmotif a listener has to be able to recognize that even when it's stretched into a totally different form than initially presented, even if this recognition only happens subconsciously. Our first example of this motif being used in game comes in the game's intro cinematic where it's plunked out almost as an afterthought high up on the piano as the player character arrives at hallonest.

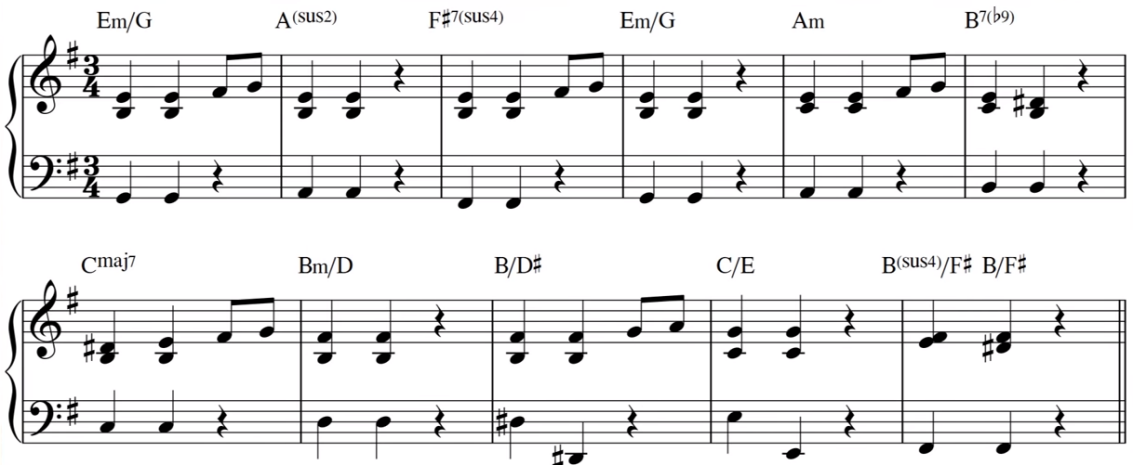


This leitmotif is used to represent not just the literal Hollow Knight, but the vessels which were beings created by The Pale King and filled with void in his attempts to create a pure vessel that could contain the radiance that had been spreading the infection all across colonists. Our hero called the Knight is one of these vessels. Besides the intro cutscene, the Hollow Knight leitmotif is used a couple times throughout his travels through the environment. For example, the battle with the Hollow Knight himself.

The music in this fight is a truly masterful conclusion to the Hollow Knights story combining the bombastic thrill and intensity of battle with a poignant reflection. The whole leitmotif is stretched out into long and slow phrases to contrast an accompanying frenzied string part. This two-part sixteenth-note line jaggedly outlines a harmonic minor sound on our tonic A minor chord. However, instead of sounding like your typical boss fight theme this piece also retains a serious quality.



This is attributed to the doubling of the melody in the bass part even with the swirling hectic string runs going on around it granting the music with long slow bass notes playing. This solemn melody colors the whole piece with a grave tone which is fitting for a final boss fight closing out the games story.



After a certain point in the boss fight, the hollow Knight begins to stab himself with his nail and the music completely shifts. The chaotic string runs of the first phrase are replaced with these slow string chords, taking the first three notes of the Hollow knight motif and cutting them off with a return to the tonic, refusing to let the full melody play out. This, combined with the uneven eleven bar length of the loop, makes it feel like the melody is trapped in place, playing out a cycle that is destined to repeat itself.

This incredible use of music to tell stories is where Hollow knight's design is unique and it's something which should be replicated in more games.

Conclusion

Points of interest:

- **Elegy** — The way Hollow knight's scene is set at the start of the game with a simple poem is elegant and is something I would consider starting my game with.
- **Core mechanics** — Hollow knight, being a 2D side scroller has mastered the control handling and feel of a platforming game with smooth unrestricted movement.
- **Artstyle** — Both Hollow knight's colour scheme and animation styles are incredibly simplistic, smooth and effective: something worth examining and trying to incorporate into my own project.
- **Map** — The way that the map is displayed as an overlay does not take away from immersion for the player and would be something to consider programming.
- **Enemies and bosses** — All designs are stylised and have a reason for being included within the game world which is useful for immersive and believable stories.
- **Sound design** — The use of music to help with storytelling is incredible, something which i will have to consider in my limitations but would be a great addition to a story based game.
- **Collectables** — This gives the player an optional secondary objective which can be rewarding and give the player a sense of achievement.

Points of dislike:

- **Combat** — Hollow knight uses fast paced reaction time based combat which is something that my client would not want in their game.
- **Currency (geo)** — Having a shop system in a storytelling based game is an unnecessary addition.

Main conclusion----------

Points to develop:

From my research, the game which most suits the storytelling narrative my client is looking for is Hollow knight. From this game I would want to take **Core mechanics** for platforming, **Artstyle and Map,** for player immersion and **Enemies and bosses,** for their cartoonish design and simplistic movements.

Secondly, Undertale's **Item menus** and **title screen** would be easy, simplistic and effective designs to use in my own game as it suits my client's request of a similar **Dialogue based combat system** as it will be a good way to develop characters.

The Witcher 3 doesn't have many points for me to incorporate into my own game except from the **Loading introduction** which is useful to pick up play sessions where you left off and a **Dialogue system** which adds complexity and gives choice to the player.

Potential ideas:

After researching games based off of my interview and questionnaire, I have developed a few basic thoughts of what the game could include:

Firstly, the game needs to retain its story-based focus with all design ideas included, to make the game feel like a story in which the player is interacting with. A **platforming mechanic** like the one found in Hollow knight could be considered as it lets the player feel progression. The hardships of the main character are projected onto the player as it is in their control how well they complete platforming segments. All failures and achievements are both the player's and the character's, connecting the two together.

There is also room for a **choice-based dialogue system** like the one seen in the Witcher 3 with converging story arcs resulting in the same outcome eventually, but letting the player decide which route to take. This would be a worthy addition as it: gives control to the player, helps the game world feel alive resulting in immersive storytelling and it adds complexity to the plot.

An idea inspired by Hollow Knight's introduction elegy is that the game's story would be told in **poems**. This could be replicated in the dialogue with limmerics and with ballads for the friendly creatures who cannot speak.

As mentioned in the last idea, the game might include a number of **friendly NPC's**. This would help with storytelling as they can act as dialogue boxes and develop world coherence.

Developing on the platforming idea, the game world could be designed as **a climb**, for example the main character lives at the foot of a mountain and at the top there is riches or a potion which the main character needs to help save his loved one. On the way up he finds ruins, old civilisations, native species, ruins, etc.

Transcript with client:
To help decide on these ideas, I have contacted my client and arranged a feedback session.

==========

Adam: Have you ever had any experience with platforming style videogames? If so, what are your opinions, likes/dislikes?

Grace: Yes, growing up I played a lot of super mario bros, I personally very much enjoy the style of game as it is very beginner friendly and there is no need for the player to have a high level of competency as they can take their time playing. I feel however, that many of its advantages can be disadvantages as it can be simple. It also comes across as patronising, holding the player's hand and it does not utilise any depth of the game world in this.

==========

Adam: Would you prefer the dialogue in the game to be choice based or linear as to replicate more like a book?

Grace: I would much prefer the dialogue to be linear as it solidifies the story into being told from the complete control of the designer and storywrite. Although the dialogue choices may be more appealing to the potential users.

==========

Adam: Do poems interest you as a medium for storytelling? If so, which style and do you have any examples?

Grace: Yeah!! I think poems are often superior to traditional dialogue as they can be much more emotive through their use of language forms, metre and rhyming scheme. These skills make it much more interesting as it gives much more freedom to the writing and ultimately gives a more interpretive dialogue to players.

O what can ail thee, knight-at-arms; Alone and palely loitering?; The sedge has withered from the lake; And no birds can sing;

This poem I read recently has a cyclical structure, this means that the last line "And no birds can sing" is used over in other verses of the poem. From a narrative point this can be used to give a view into the character's thoughts, as you can tell that they are stuck on an important idea, cycling through it over and over again. This cyclical language device could show the characters drive or motivation inside the game to encourage the players gameplay loop.

However this style may confuse other users.

==========

Adam: How do you think the character would be affected by a continuously climbing and developing story? Would his personality be changed by what he sees? Are we set on a theme of a mountain climb with ruins and old civilisations etc?

Grace: The character should experience a change in development, he should see and experience things which opened his eyes to the world around him and he should be shown a new side to his life. Yeah i think the mountainous climb idea is a set one.

==========

Adam: Are we certain of the non combat - combat system? And should it be done through set boss battles or a selection of random encounters?

Grace: Yes, the non combat will be a refreshing change to the industry standard combat. And as for the random encounters or boss battles, either fits but boss battles may be more impactful and you can spend more development time on a select few.

==========

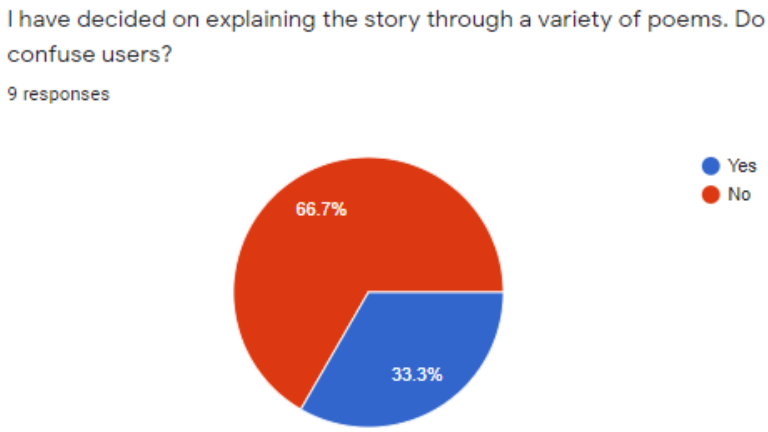
Adam: Do you have any other ideas to add to this?

Grace: No, I think the ideas we have are sufficient in creating a game which combats the problem I am having.

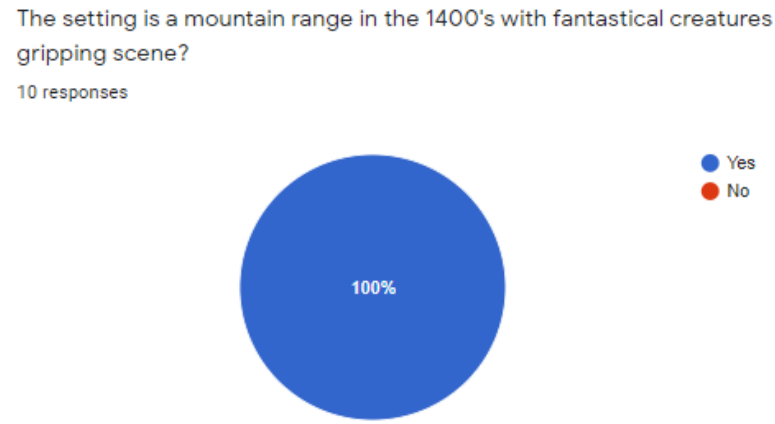
==========

Questionnaire feedback:

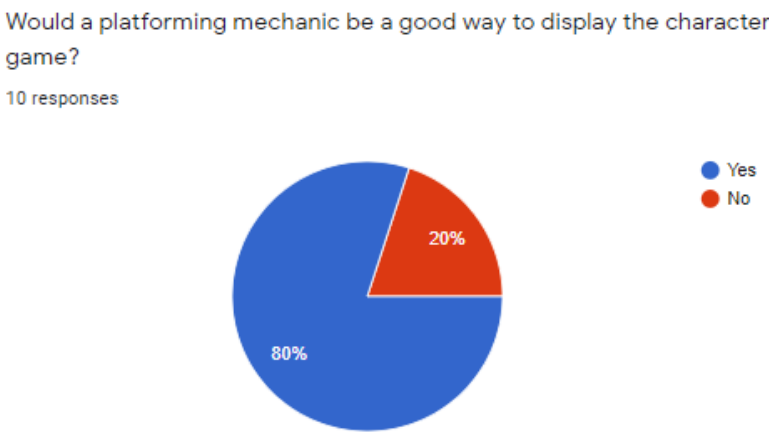
In addition to a client-developer dialogue, I have set up a secondary questionnaire for the user base to gather feedback furthering my decisions.



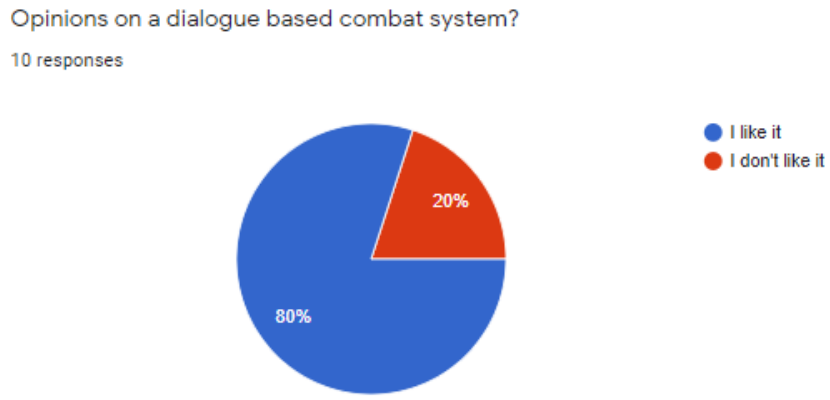
The client has stated that she wants the game to be told through poems and the majority of users believe that it will not confuse them which was her main concern.



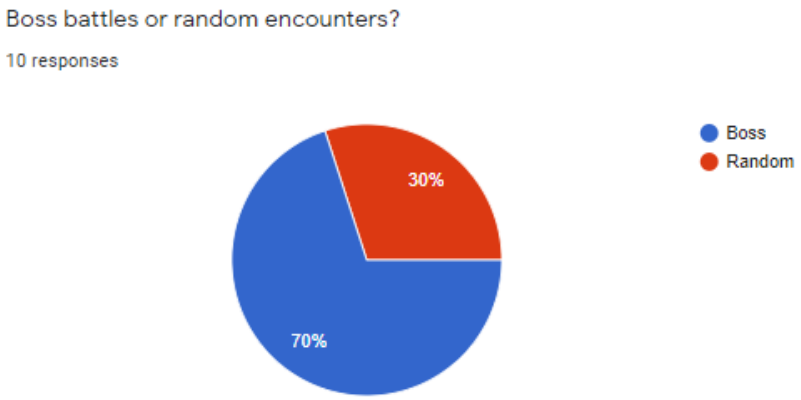
There is a resounding agreement of the setting and scene being gripping for the game design.



Most users agree with the client for there being a platforming mechanic in the game to help with player character immersion.



Most users also like the idea of a dialogue combat system as does the client.



With a 70/30 split, most users want the combat to happen with set boss encounters.

Proposed solution

After concluding and talking with my users, client and stakeholders I have designed a proposed solution and design for the game.

For my proposed solution, I have stated what my game is going to include as well as where the idea originated from within my research where you can find where I justify its use.

The game will be a 2D platformer RPG (Role Play Game). - ***Hollow knight and Undertale 2D design***

There will be dialogue between the main character and individual NPC's, enemies and interactable objects within the world. - ***Client's request***

Any and all text will be there to build the plot of the story and world atmosphere the game is portraying. - ***Problem***

The world's story will be told through poems created with the help of my stakeholders and client. - ***Own idea from Hollow knight elegy backed by client***

An introduction will play when you first enter the game to display the style the game involves. - ***Hollow knight, idea backed by client***

The game will be set in a fantastical mountain range with odd creatures and locations. - ***Hollow knight, idea backed by client***

The simplified goal of the game is to reach the top of the mountain. - ***Own idea backed by client***

Platforming segments will be the game's main gameplay loop as well as a boss encounter after each one. - ***Hollow knight, idea backed by client***

The boss encounters will have no combat but instead use decision based dialogue puzzles. - ***Undertale, idea backed by client***

The game will use original pixel art and simple animations with bright and vibrant colours. - ***Hollow knight side scrolling and Undertale colour pallet, idea backed by client***

There will be simple, catchy and repetitive melodic phrases playing in the background when you play - ***Hollow knight, idea backed by client***

Limitations

Time

The project in hand came to life on the 12th of january where I started to work on analysing my clients problem she presented to me the week before. This project will need to be finished between 20th june - 22 september before the end of summer as that is the stated deadline.

This restricts much of the development i can include in my game as only a certain amount of time can be spent on it, 254 days maximum to be exact.

This time will not only be needed to be spent on making the game but also **Documenting, Research, Designing, Client & User feedback waiting times and Learning**.

My time available to me is also further reduced by the fact that i am a full-time student studying for 3 A-levels 5 days a week for at least 6-7 hours in and out of school with extra work on the weekends. Not only being a student but also having a part-time job reduces my available hours per week.

Learning

For my proposed solution, I shall have to learn many skills to complete each aspect of the game effectively. Everything which is needed to be learnt shall cost me development time.

Programming: I already have basic knowledge of the C# language, but a more adequate fluency will be needed for more complex solutions I will face in my project.

Photoshop: To design my own pixel art and animations, I will have to learn how to design characters, animate them and implement correct file sizes into the game engine from Adobe Photoshop CC 2019.

Unity: I have zero experience with using the unity game engine except basic concepts and general knowledge of how game engines work, therefore adequate learning time will be needed to be accounted for.

Story Writing: In order to create coherent plot, characters and world I will need to do extra research into story writing.

Age Rating

The game is being designed for a 16 year old client, legally this means that the game will have to be within the rules to have an age rating of 16 years and below.

Here is the quote I found from parentinfo.org on the PEGI guidelines for 16 rated games:

PEGI 16

The game can feature death and injury to humans, including gory and bloody violence if the game is 'arcade style' (ie: not too realistic.) Smoking, drinking alcohol, the use of illegal drugs, glamorized representation of crime and strong bad language can be shown.

It can contain erotic nudity and sexual activity, excluding the showing of genitals.

Software

There are many game development softwares available for use, most of them cost money to use. This is not something the client has a budget for and so the available softwares is limited to "free to use". This means that the optimal game engine cannot be used and the software chosen may not be able to perform the actions I want/need. This is also true for photo editing software.

For the game engine im using Unity as it is free and uses C# which is the syntax I would like to code in

For the photo editing, I have a free photoshop license from school so I will be using that.

Hardware

Hardware is also a limitation to the user base of the game, so development will have to take into account the capability of lower-end machines. A minimum specification will be needed to be presented with the game as to be played, a minimum requirement will be met. The research below is to help me design a specification (shown at end of segment).

Minimum system requirements for Adobe Photoshop

- Processor: Intel® or AMD processor with 64-bit support; 2 GHz or faster processor
- Operating system: Microsoft Windows 7 with Service Pack 1 (64-bit), Windows 10 (Version 1709 or later)
- RAM: 2 GB or more of RAM (8 GB recommended)
- Hard disk space: 3.1 GB or more of available hard-disk space for 64-bit installation; additional free space required during installation
- Monitor resolution: 1024 x 768 display (1280×800 recommended) with 16-bit color and 512 MB or more of dedicated VRAM; 2 GB is recommended
- Graphics processor acceleration requirements: OpenGL 2.0-capable system
- Internet: An Internet connection and registration are necessary for required software activation, validation of subscriptions, and access to online services.

Official unity 2D Hardware minimum requirements

- Processor: Intel Core i7-3770 @ 3.4 GHz or AMD FX-8350 @ 4.0 GHz or better.
- RAM: 8GB.
- Video Card: NVIDIA GeForce GTX 780 or AMDRadeon R9 290X (3 GB VRAM)

Visual studio 2019 system requirements

- 1.8 GHz or faster processor. Quad-core or better recommended

- 2 GB of RAM; 8 GB of RAM recommended (2.5 GB minimum if running on a virtual machine)

- Hard disk space: Minimum of 800MB up to 210 GB of available space, depending on features installed; typical installations require 20-50 GB of free space.

- Hard disk speed: to improve performance, install Windows and Visual Studio on a solid state drive (SSD).

- Video card that supports a minimum display resolution of 720p (1280 by 720); Visual Studio will work best at a resolution of WXGA (1366 by 768) or higher.

Undertale

Minimum requirements

- CPU: (Any)
- RAM: 2 GB RAM
- GPU: 128MB
- OS: Windows XP, Vista, 7, 8
- Store: 200 MB available space

Recommended requirements

- CPU: Any @ 2GHz+
- RAM: 3 GB RAM
- GPU: 512MB
- OS: Windows XP, Vista, 7, or 10
- Store: 200 MB available space

Hollow knight

Minimum system requirements

- CPU: Intel Core 2 Duo E5200
- CPU SPEED: 2.8GHz
- RAM: 4 GB
- OS: Windows 7
- VIDEO CARD: GeForce 980GTX (1GB)
- FREE DISK SPACE: 9 GB
- DEDICATED VIDEO RAM: 1 GB

Recommended Requirements

- CPU: Intel Core i5
- CPU SPEED: 3GHz
- RAM: 8 GB
- OS: Windows 10
- VIDEO CARD: GeForce GTX 560
- FREE DISK SPACE: 9 GB
- DEDICATED VIDEO RAM: 1 GB

The Witcher 3

Minimum system requirements

- CPU: Intel CPU Core i5-2500K 3.3GHz / AMD CPU Phenom II X4 940
- CPU SPEED: 3.2GHz
- RAM: 6 GB
- OS: 64-bit Windows 7 or 64-bit Windows 8 (8.1)
- VIDEO CARD: Nvidia GPU GeForce GTX 660 / AMD GPU Radeon HD 7870
- FREE DISK SPACE: 40 GB
- DEDICATED VIDEO RAM: 1.5 GB

Recommended system requirements

- CPU: Intel CPU Core i7 3770 3.4 GHz / AMD CPU AMD FX-8350 4 GHz
- CPU SPEED: 3.4GHz

- RAM: 8 GB
- OS: 64-bit Windows 7 or 64-bit Windows 8 (8.1)
- VIDEO CARD: Nvidia GPU GeForce GTX 770 / AMD GPU Radeon R9 290
- FREE DISK SPACE: 40 GB
- DEDICATED VIDEO RAM: 2 GB

Note: This is a 3d game and wont be an accurate representation of the required specs to run my game

---

My personal specs

- Processor: Intel Core i5 7500 @ 3.8 GHz
- RAM: 8GB
- Video Card: NVIDIA GeForce GTX 1070
- 64 bit Windows OS 10
- 1TB HDD storage
- 255GB SSD storage

I have above minimum requirements for the three main programs in use so I should encounter no problems with software or hardware.

For the users of my project, a low-end machine may be used to play the game creating accessibility for potential users.

My recommendation for minimum hardware

- Processor: Intel Core i3 6300 and above
- RAM: 8GB
- Video card: GTX 1050ti
- 10GB storage space
- Keyboard
- Mouse
- Monitor

My recommendation for software

Windows, mac or linux operating system as these are supported by Unity and C#

Computational thinking

Throughout my project I will be using computational thinking to achieve the desired outcome and here are a few examples of where they will be used.

Abstraction

Abstraction is the principle of reducing the complexity of an item to its essential features to allow for efficient design and implementation of system softwares. Because humans cannot think about all of the things that go on in a program at once. We need to divide programs into separate pieces that we can understand - ensuring that these pieces only interact with each other in very simple ways. When you write code that uses an abstraction, you can write code once that will be reusable against any new code that implements that abstraction. You can write less code to do more.

Incorporation

- Coding segments - To simplify the programming, separate scripts will be used for different segments to make it easier to access, identify and edit when troubleshooting.
- Success criteria - The tasks needed to solve the core components of the problem

Thinking ahead

Thinking ahead is the planning of input outputs and preconditions that the design will adhere to before it is made. The aim for most designs in computing is reusability for efficiency, both for the designer and computer. At its most basic, thinking ahead is a set of conditions which need to be met irrelevant of *how* they are met.

Incorporation

- Event design - When designing the dialogue combat, different occurrences will need to be planned out for different answers
- Success criteria - For the project to be completed to the correct standard, a set of goals must be created and met

Thinking logically

Thinking logically identifies the points in a solution where decisions have to be made, the conditions that affect the outcome of this decision and how they affect the flow through a program. The purpose is to understand how the design needs to branch out to work and include all aspects of the design.

Incorporation

- Dialogue - When choosing different occurrences the design needs to
- Client interviews and questionnaires
- Main gameplay loop

Thinking concurrently

Thinking concurrently is the process of defining which objectives need to occur at the same time with each task being executed separately to maximise efficiency within the design. This is used for the design of programs and events to be made efficiently and accurately.

Incorporation

- Movement mechanics
- Sound design
- Visual art

Decomposition

Decomposition is the process of taking a large task/problem and breaking it down into smaller sub-problems/tasks. Depending on the size of the task, the already smaller sub-problems can be broken down further into sub-problems of their own. This is to create more efficient problem solving as it is easier to manage and comprehend

Incorporation

- Success criteria
- Hardware
- Research

Success Criteria

(Red represents necessary objectives, yellow shows objectives which should be met but are not critical, green shows extension tasks)

Criteria number	Explanation	Reasoning	Need

1	There must be a main menu with selectors for the player to navigate in and out of the game and to adjust settings if any are available.	This is so that the main parts of the game are created and the player can move change between them	Must
1.1	There must be Contine, New game, Options and Exit game buttons	This is so that the player can access every part of the game	Must
1.2	There may be background art to show the theme of the game	This is so that the atmosphere for the game is set	Should
1.3	There could be background music to help set the scene	This is so that the atmosphere for the game is set	Extra
2	It must include an introduction which introduces the story, grips the user and displays all gameplay mechanics (aka movement and dialogue)	This is so that the player understands how to play the game and does not get bored before the game can hook them in	Must
3	The game must have fully functioning dialogue menu with branching choice paths	This is so that the player has choice in how to explore the story	Must
3.1	It must have buttons for responses, an exit button and a place to display the text	So that the dialogue is interactable for the player as it is how the game is differentiated from a book	Must
3.2	There should be a simple transcript design layout which can be easily implemented into the game	This is so that development is easier and more organised	Should
3.3	It could show who the player is talking to	For the player to understand who the conversation is between	Extra
4	There must be platforming elements which challenge the player	To create challenge which will hook the player into playing	Must
4.1	It must have player directional control	This is so that the player can move and platform	Must
4.2	It must have a jumping mechanic	This is so that the player can jump between platforms	Must

4.3	It may have a wall climbing mechanic	This is to add more depth to the movement and keep the player more engaged	Extra
5	There must be a coherent story with a complication, climax and denouement	This is so that the story is like that seen in books	Must
6	There should be a soundtrack with available volume adjustments	This is so that there is music which sets the mood for the story and can suit the players sound levels	Should
7	There must be characters and items to interact with	This is so that the story can be told in numerous ways	Must
8	There must be bosses to encounter and defeat	This is so that there is meaning and motivation to interact with dialogue	Must
9	There should be animation	This is so that the game looks the part	Should
10	The movement should be smooth	This is so that the game is fun to play and does not feel jagged	Should
11	It could have original artwork	This is so that the game has its own distinct style and looks good	Extra
12	There may be a save system	This is so that the player can save their progress and come back to where they left off at a later date like a bookmark	Extra

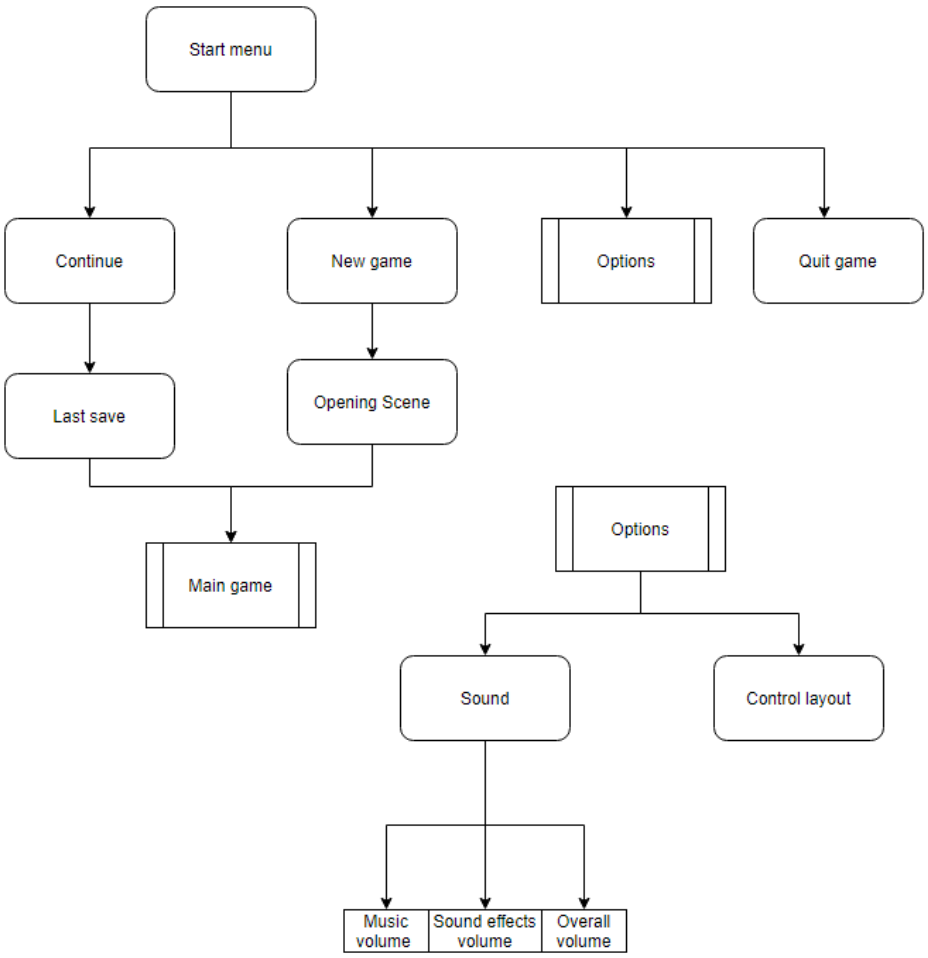
Design

Systems diagram

The systems diagram is used to give the design of my program structure. The diagram breaks down the design into smaller sub-problems, in which I can tackle individually.

This will allow me to use “thinking ahead” to determine inputs and outputs and allow me to use “thinking logically” to look at the where decisions are required and what

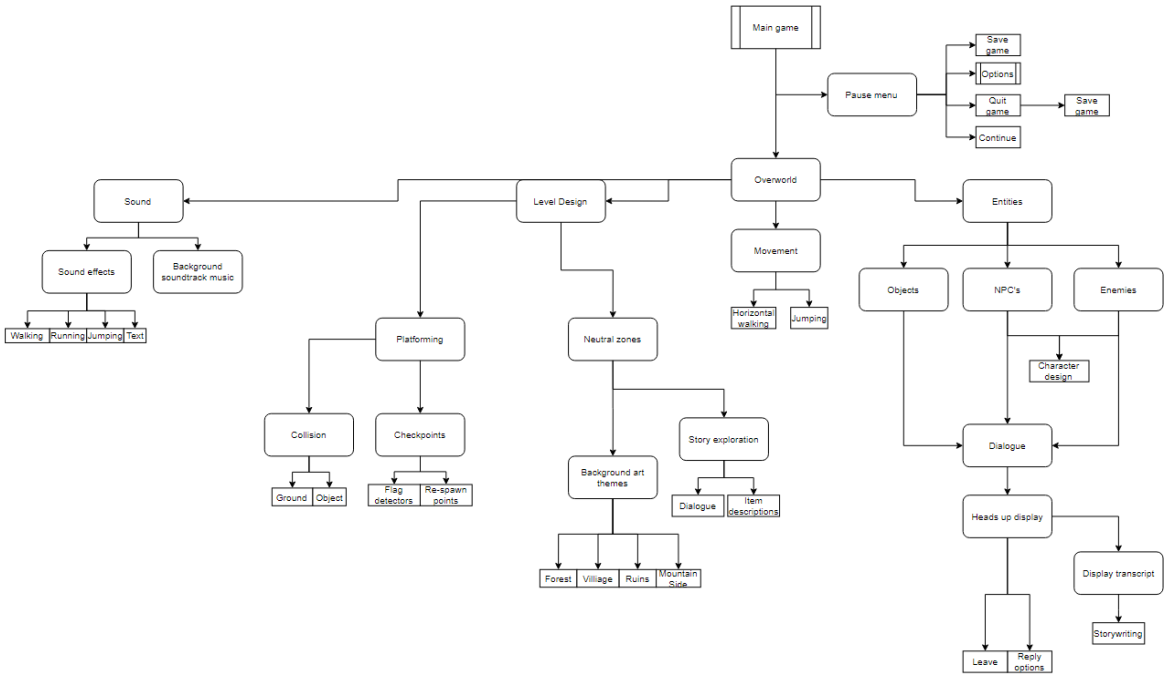
effects it will have on other parts of the solution. This will give my working solution some structure letting me work towards my solution systematically.



This is my start menu broken down to its core components and then linked to the other components it leads to, ending up at the “Main Game” component (only needing a separate structure for simplicity purposes).

The main game has a few core components which will need to be focused on to develop the majority of the game. These include the **Level Design, Movement, Entities and**

Pause Menu also **Sound** as an extension task.

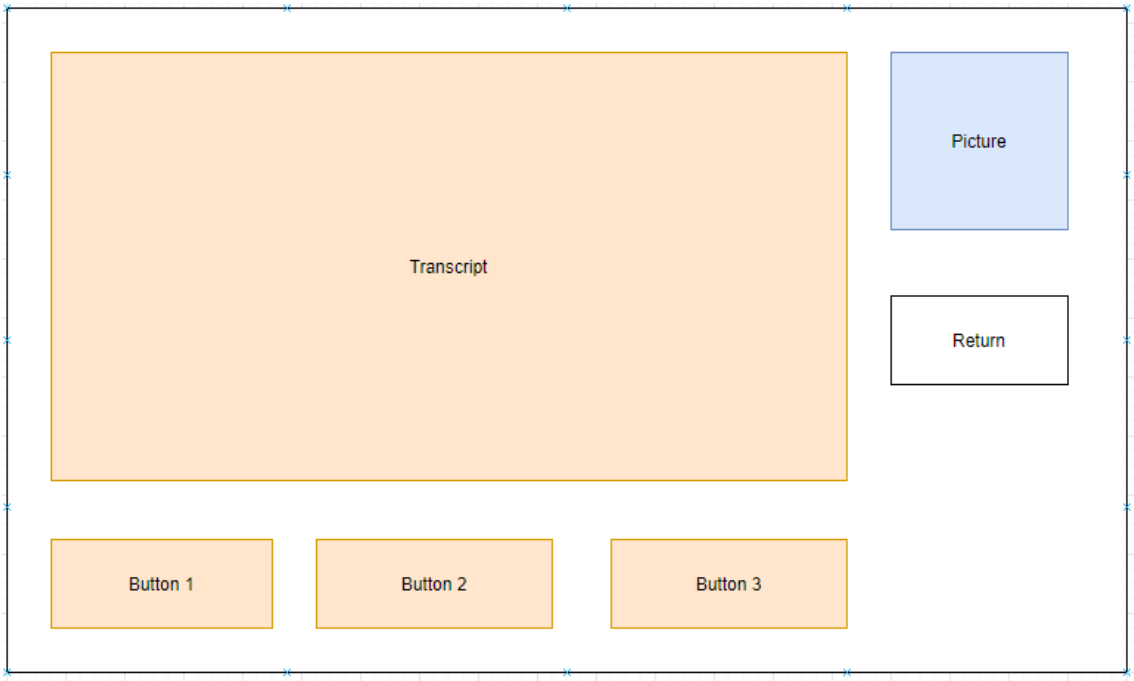


I will explain how these modules need to look and act in the rest of the design below:

Dialogue

User interface design

The user interface completes a fully functioning dialogue menu in my success criteria.



This is the main user interface design where all interactions between the player and entities will happen in the game. It allows the user to read the story text, return to the overworld and reply. The most important part of the design is the displayed transcript using most of the space available to be able to read easily.

Transcript

The transcript box will contain the story text relative to the entity which the transcript belongs to. The text will be displayed word by word in sequence to make it seem as if the conversation is progressing also making it easier to follow for the reader. The next part of the text will not be shown until a button is pressed.

Link to success criteria

- There must be a coherent story with a complication, climax and denouement.
- The game must have fully functioning dialogue menu with branching choice paths

Return

The return button will be available when the interaction or conversation is complete. It will return the player to the overworld and return movement controls. When the interaction is still in progress, the button will be blacked out:

Link to success criteria

- It must have buttons for responses, an exit button and a place to display the text

Button 1, 2 & 3

These buttons will relate to the choices and paths in the transcript. They will be labeled with their correct numbers which correlate to the choices set in the text. If only one or two choices are available, the leftover buttons are blacked out signifying that they are not in use:

Buttons 1 and 2 are in use unlike button three as only two options are available.

Link to success criteria

- It must have buttons for responses, an exit button and a place to display the text

Picture

This will be a close up of the object or character the text is between to keep the player oriented with who they are talking to.

Link to success criteria

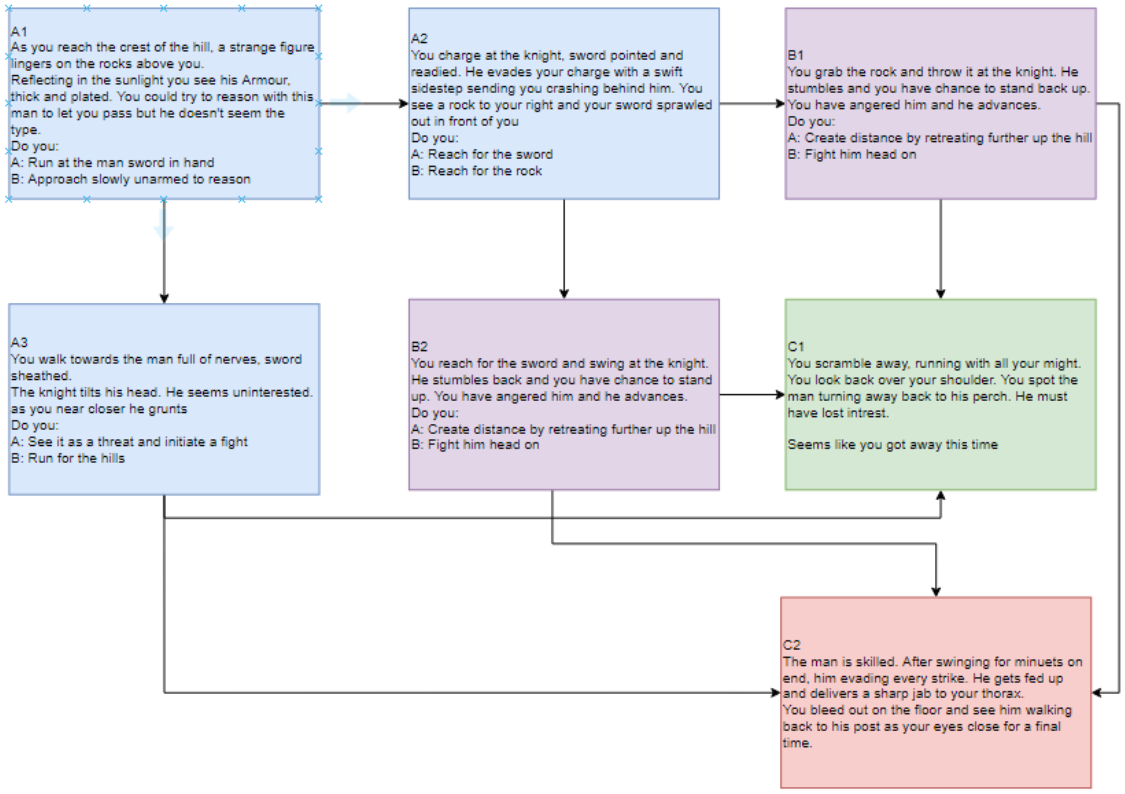
- It could show who the player is talking to for coherency

```
SUBROUTINE dialogue()                                   #calls and names the subroutine

open(dialoguetree.txt)                                  #opens the transcript
activeConvo = TRUE                                      #sets the while loop condition

WHILE activeConvo = TRUE:                               #starts while loop
        state = line.read(dialoguetree.txt)             #stores the state from the file
        if buttonPress = 1:                             #checks if 1 button is pressed
                nextState = state + 1                   #stores the next state variable

        else if buttonPress = 2:                        #checks if 1 button is pressed
                nextState = state + 2                   #stores the next state variable

        else if buttonPress = 3:                        #checks if 1 button is pressed
                nextState = state + 3                   #stores the next state variable

        else:                                           #when there are no more options
                activeConvo = FALSE                     #closes loop
        OUTPUT nextState                                #displays the read state on screen
```

Here is the pseudocode for the dialogue scene

Transcript

The transcript design is a layout of branching pathways connecting scenario with scenario. The story will be written like this so when completed it can be easily placed into the game.



Example text is not accurate to the relevant story

All points lead to one of 2 or more outcomes, at least one of which will make you restart the interaction. They use a number, lettering and colour system to keep the diagram easy to interpret and keep track of when it becomes more complex.

Link to success criteria

- There should be a simple transcript design layout which can be easily implemented into the game

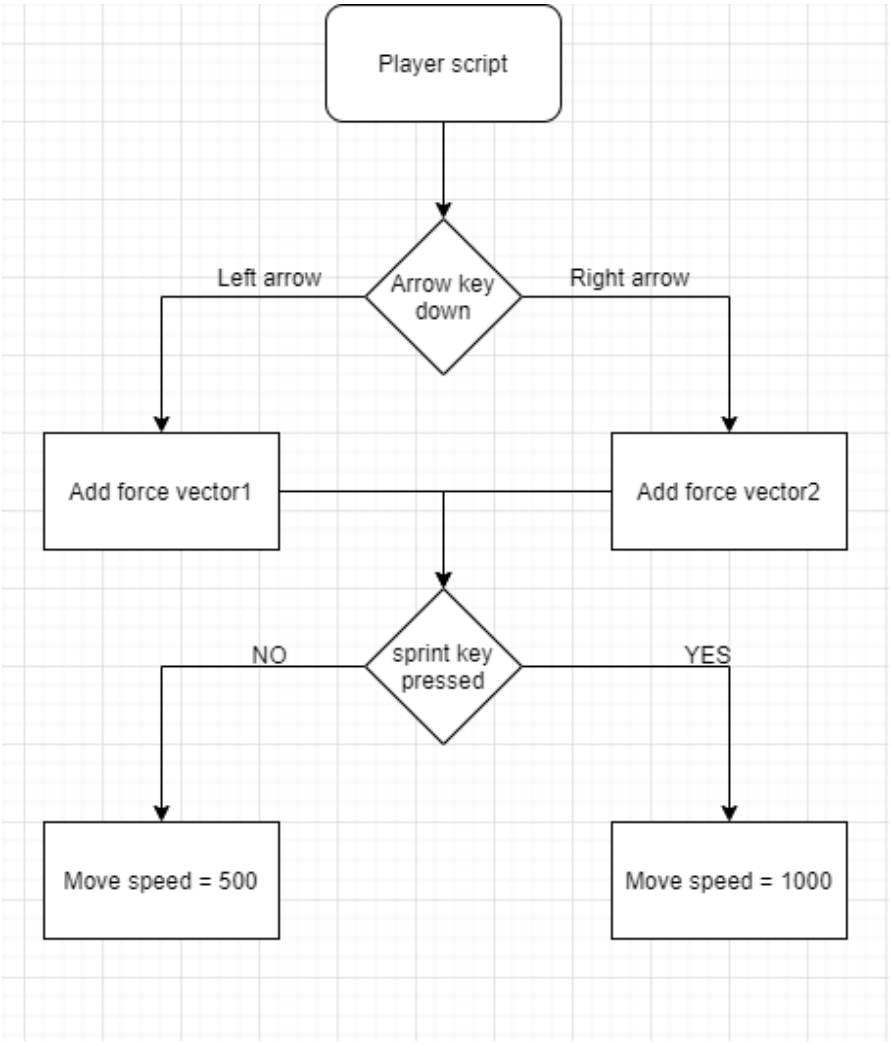
****IMPORTANT NOTE****

The story writing and script is not relevant to the code of the program and will be shown in its entirety at the end of the project.

Movement

Horizontal movement

For the player to traverse the world, the horizontal movement must work smoothly and the player will be able to move at different paces throughout the game.



If the player presses either left or right arrows, force is added to the corresponding direction and then a check to see if the character is sprinting. Then move speed is added to the direction to allow the player to travel.

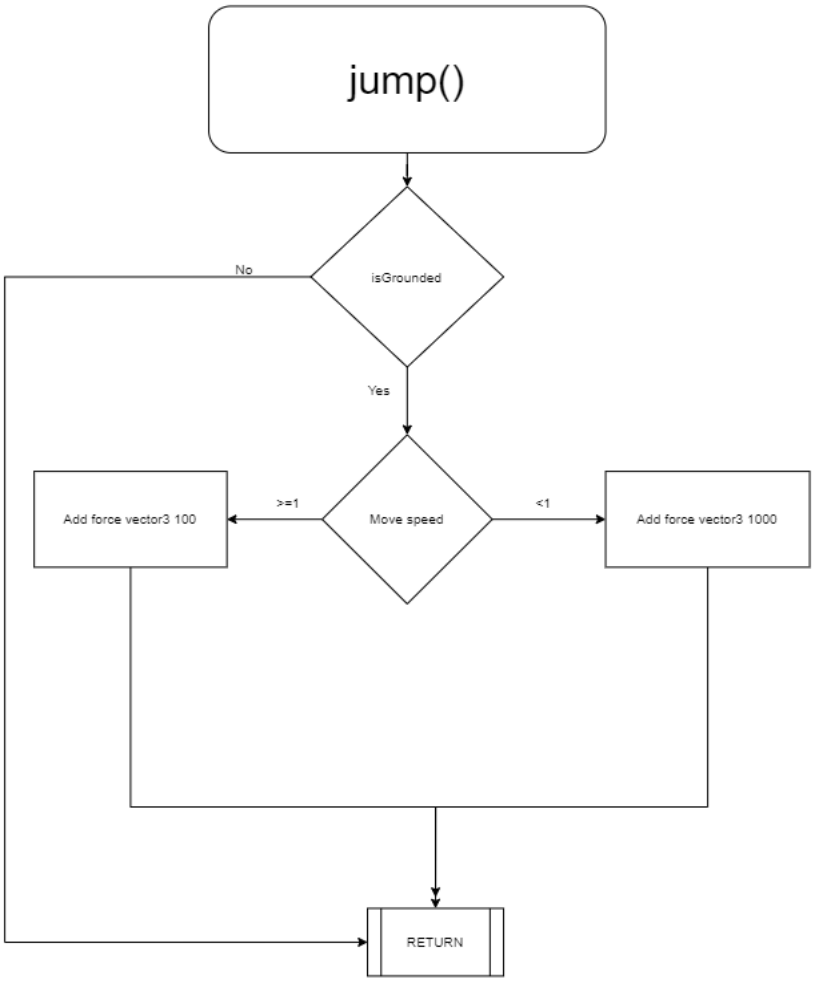
```
SUBROUTINE playerMove(vector1,vector2,moveSpeed)                #calls vector and movespeed variables the player move subroutine
        movespeed = 100                                         #sets the walk movement speed
        if keydown == (SHIFTKEY)                                #checks the sprint key
                movespeed = 1000                                #sets the sprint movment speed
        else:
                if keyDown == (LEFTARROW):                      #checks if the player wishes to move left
                        vector1 = vector1*moveSpeed             #adds movment speed to direction

                else if keyDown == (RIGHTARROW):                #checks if the player wishes to move right
                        vector2 = vector2*moveSpeed             #adds movment speed to direction

                else:                                           #if no key is pressed
                        vector1 = 0                             #resets the vector1 variable
                        vector2 = 0                             #resets the vector2 variable
        moveSpeed = 0                                           #sets the moveSpeed to IDLE speed
        RETURN                                                  #ends the subroutine
```

This is the pseudocode for the movement subroutine

Jumping

As a main mechanic for platforming the jump algorithm must be efficient and have no bugs to not disturb the flow of the program.

Here is the main flow of the function:



To have the jump mechanic work effectively parameters must be set. For example, not being able to infinitely jump and how high and far the jump will allow the player to travel.

The player should be unable to jump again whilst in the air. To stop this from happening a boolean variable with a collision detector should work in unison updating and returning the output which then allows the jumping function to be called upon again.

The isGrounded parameter will be coded with its relevant collision detection later in the design.

Here is the code and flow of parts of the function:

```
FUNCTION jump()                         #states the name of the function
        if isGrounded == TRUE:          #parameter check if the player is already jumping
                                        #rest of the code
        else:                           #if the player is jumping, the function is finished
                RETURN                  #returns to the main code flow
```

The pseudocode for implementing the isGrounded parameter

```
if USERINPUT == SpaceKeyPressed:        #checks if spacebar is pressed
        jump()                          #calls jump function
```

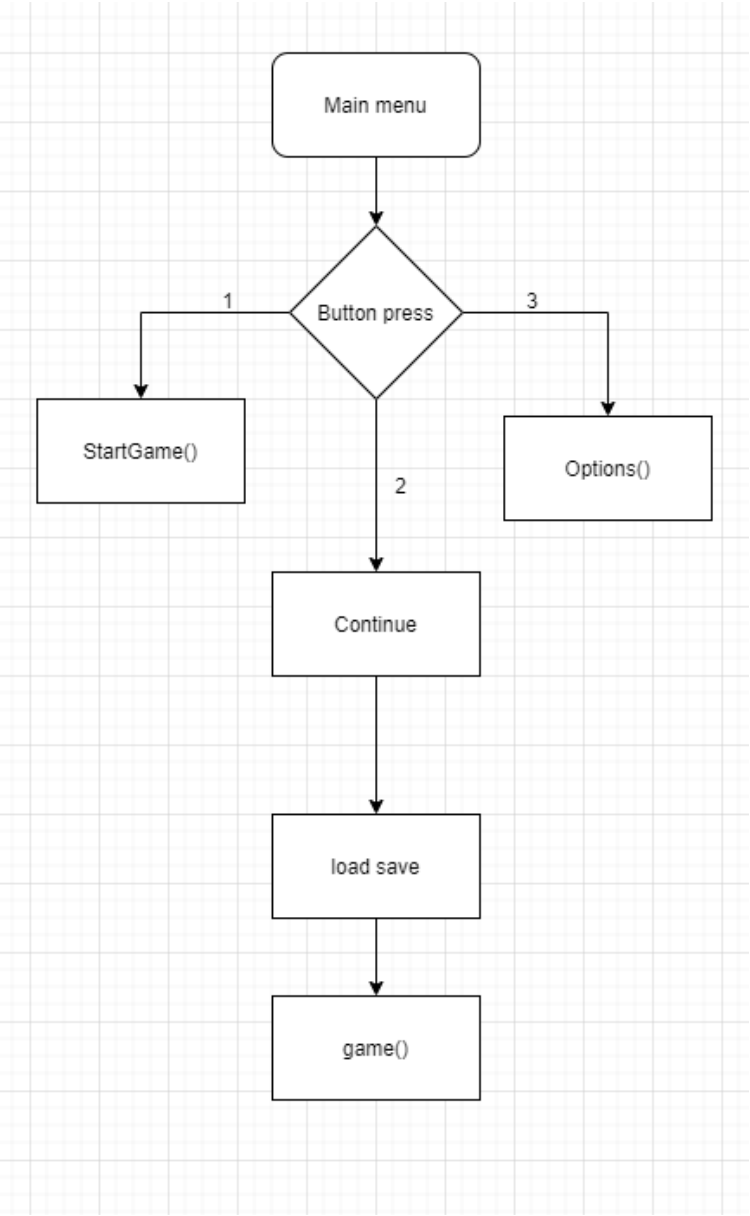
The pseudocode for calling the function

```
#rest of the code
if movespeed < 1:                       #checks if the player is moving at sprint speed
        vector3*1000                    #sets the vertical movement to 1000

else if movespeed >= 1                  #checks if the player is moving at walk speed
        vector3*100                     # sets the vertical movment to 100

else:                                   #if all else fails
        catch                           #catch the code
```

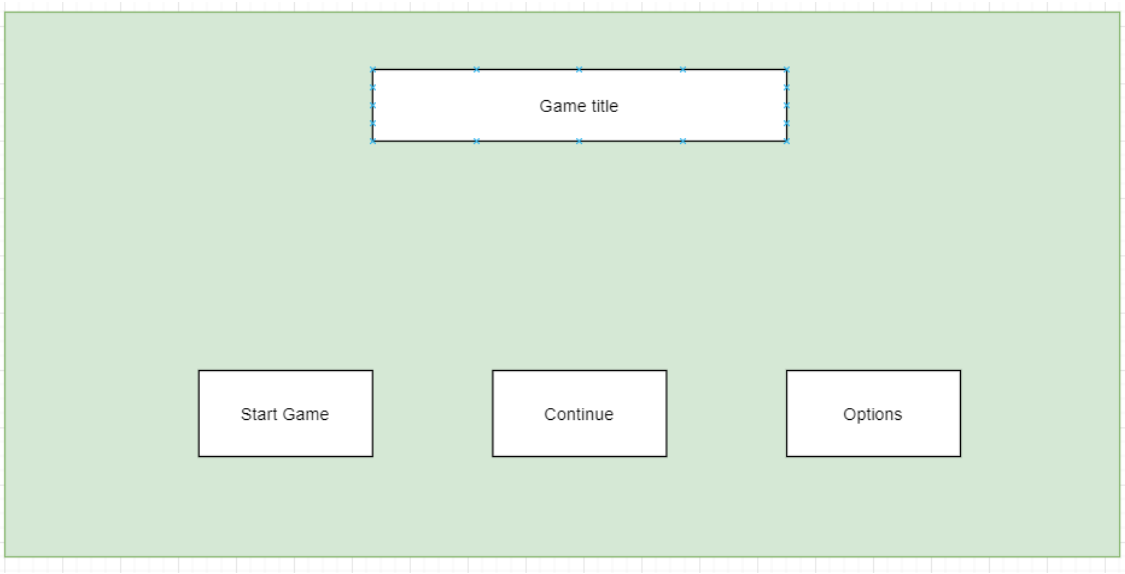
Pseudocode for the rest of the function

Main menu + Options

Flow of the subroutine:


Main menu
Button press
1
3
StartGame()
Options()
2
Continue
load save
game()

Design



The design of the main menu includes 3 buttons and the game title. The green background represents background art which depicts the game in hand.

The options menu above has the same art background with two volume sliders and a non-interactable list of controls (jpeg) for the player to remind themselves of how to play.

Pseudocode

```
SUBROUTINE mainMenu(lastSave)                   #start of mainMenu subroutine
        IF userinput = (buttonPress1):          #checks button press
                game()                          #starts the game subroutine
        IF userinput = (buttonPress2):          #checks button press
                gameState = lastSave            #sets the game to the last save
                game()                          #starts the game subroutine
        IF userinput = (buttonPress3):          #checks button press
                options()                       #starts options subroutine
```

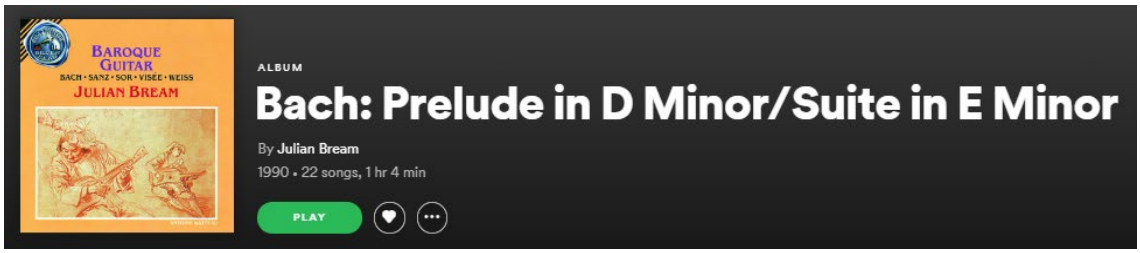
code for the main menu subroutine

```
SUBROUTINE options()
        volume = userinput                      #changes the volume with the user input
        if userinput = (keyDownESCAPE):         #if the user presses escape key
                mainMenu()                      #loop back to the main menu
```

code for the options subroutine

Sound

Soundtrack



The game's soundtrack will be a lute prelude suite which is fair use, so no fees will have to be paid to use it. Tracks will be selected to fit areas of the game, so not all may be used. For example, the 51 second "Suite in D Minor: Prelude" will suit perfectly for the main menu due to its short repetitive motifs.

The songs will use the volume variable from the options menu to address how loud it plays. The same applies for any sound effects.

The songs will play when the character passes a flag and will repeat until the next flag is crossed. These flags will be placed at main points in the game such as boss encounters or when introducing a new character.

Here is the pseudo code:

```
soundtrack = [track1, track2, etc...]
playSong = TRUE

WHILE playsong = TRUE:
        if flagPosition = 1:
                PLAY soundtrack[0]
        etc:
|
```

Ambient sound

Similarly to the soundtrack the ambient noise will play depending on the area the character is located. The flags however will be placed in quieter sections of the game.

Sound effects

The sound effects such as footsteps, leaf rustles will be played when the player passes over certain terrain

Here is the pseudocode:

```
soundEffects = [sound1, sound2, etc...]
terrain = [grass, leaves]
IF terrain = [leaves]:
        PLAY soundEffect[sound1]
```

World design

Collision

For the character to stay within the world, physics will need to be in place such as when the character collides with an object such as the ground, the character does not fall through.

Here is the pseudocode:

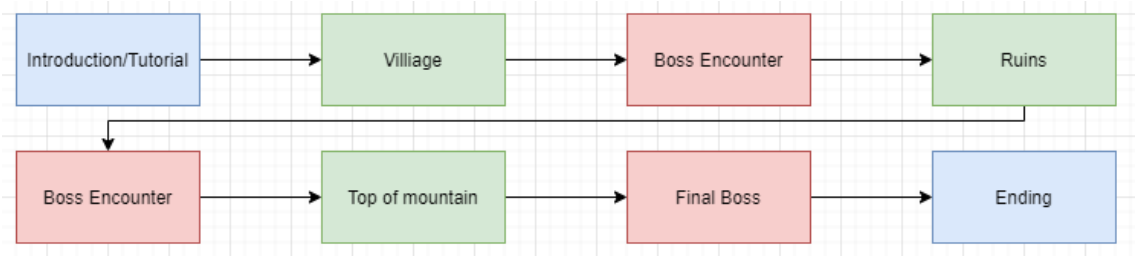
```
while characterPos == entity :  #checks if the character is touching the entity
        collision = TRUE        #sets collision to true
```

The ground the character walks on will need to output a terrain variable for the correct walking sound effect to play.

Here is the pseudocode:

```
terrain = [1]           #The terrain type for the entity
if collision = true     #checks if the player is colliding
        OUTPUT (terrain)#outputs the terrain variable
```

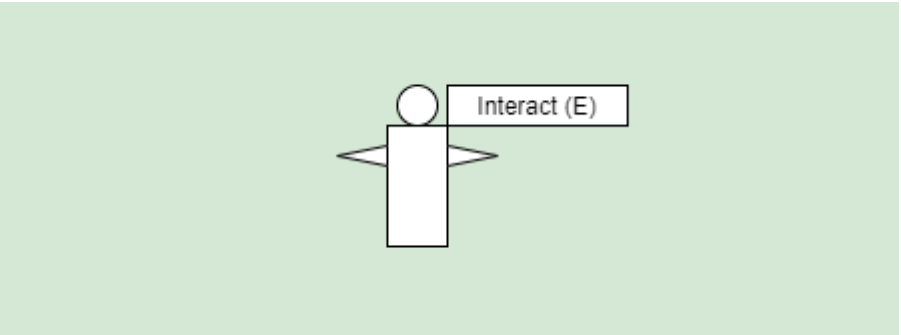
Layout



This is the layout of the game segments and how the game will flow.

Entities

Interaction



To interact with the entities in the world, there needs to be a simple pop up telling the player that they can interact with them. Also when the player presses the key the dialogue menu subroutine will need to be executed

Here is the pseudocode:

```
if interact == playerLocation:          #checks if the player is stood next to the entity
        OUTPUT (popUpInteraction)       #displays the popup
if USERINPUT = (buttonPress 'E')        #checks if the user interacts
        dialogue()                      #executes dialogue subroutine
```

Key variables and data structures

Method	Data Name	Type	Explanation	Justification
Variable	activeConvo	Boolean	Boolean True/False if the dialogue subroutine is in use	This will be used to take the player in and out of the dialogue
Game state	state	N/A	Holds the state of the dialogue transcript	This will be used to display the dialogue rooms
Variable	buttonPress/keyDown	Char	Reads which key is being outputted	This will be used to move the player
Game state	nextState	N/A	Holds the integer of which state is next to be active	This will be used to change the dialogue rooms
Variable	moveSpeed	Integer	How fast the player is travelling	This will be used to adjust the speed of the player

Variable	vector1	Integer	Horizontal left movement number	This will be used to determine which way the player is travelling
Variable	vector2	Integer	Horizontal right movement number	This will be used to determine which way the player is travelling
Variable	vector3	Integer	Vertical upwards movement number	This will be used to determine which way the player is travelling
Variable	isGrounded	Boolean	Boolean True/False if the player is touching the ground or not	This will be used to determine if the player can jump
Game state	lastSave	N/A	Holds the last save data	This will be used to load the game from a save
Game state	gameSave	N/A	Holds the current save data	This will be used to save the game and overwrite old saves
Array	soundtrack	N/A	List of songs available to the game	This will be used to choose which song is playing
Procedure	playSong	N/A	This will play the songs	This will be used to play the songs
Array	soundEffects	N/A	List of sound effects available to the game	This will be used to choose which sound effect needs to play
Array	terrain	N/A	List of all terrain types available to the game	This will hold all of the terrains to be placed in the game which are traversable
Variable	characterPos	Float	Holds the XYZ coordinates of the player position as an integer	This will be used to check where the player is in the world to use in the save files

Variable	Colliding	Boolean	Boolean True/False if the player is colliding with an object	This will be used to make sure the player does not fall through objects and gives them permanence
Procedure	popUpInteraction	N/A	Makes the popup window for interaction appear	This will be used to display text letting the player know it can interact with the object

Test data

Test data	Type	Part of the program
Button click on new game	Valid	Start menu
Button click on continue	Valid	Start menu
Button click on options	Valid	Start menu
Button click on quit	Valid	Start menu
Mouse click on screen	Invalid	Start menu
Volume 100	Borderline	Sound
Volume 50	Valid	Sound
Volume 350	Invalid	Sound
Volume -2	Invalid	Sound
“A”	Valid	Movement
“D”	Valid	Movement
Space bar	Valid	Movement
Left arrow	Valid	Movement
Right arrow	Valid	Movement
“W”	Invalid	Movement
Backspace	Invalid	Movement
“ESC”	Valid	Pause menu

“P”	Invalid	Pause menu
“Enter”	Invalid	Pause menu
Button press 1	Valid	Dialogue
Button press 2	Valid	Dialogue
Button press 3	Valid	Dialogue
Return press	Valid	Dialogue
“1”	Invalid	Dialogue
“2”	Invalid	Dialogue
“3”	Invalid	Dialogue

Acceptance testing

These are the tests which I expect to perform in my development.

No.	Requirements	Input	Expected outcome
1	The user can start a new game	Valid: Button click new game Invalid: The user presses 1 on the keyboard	Valid: The user is taken to the start of the game Invalid: Nothing happens until the valid input is pressed
2	The user can continue the game from where they left off	Valid: Button click continue Invalid: The user presses 2 on the keyboard	Valid: The user is taken to their las save Invalid: Nothing happens until the valid input is pressed
3	The user can enter the options menu to change any settings	Valid: Button click options Invalid The user presses 3 on the keyboard	Valid: The user is taken to the options menu Invalid: Nothing happens until the valid input is pressed
4	The user can press the quit button to terminate the program	Valid: Button click quit Invalid The user presses 4 on the keyboard	Valid: The program is terminated Invalid: Nothing happens until the valid input is pressed

5	The player can adjust the volume in the options menu of all sounds in the game	Valid: Volume set to 50% Invalid: Volume set to -10% Borderline: Volume is set to 0%	Valid: The volume is set to 50% as inputted Invalid: The volume stops at the lowest volume possible Borderline: The volume stops at the lowest volume possible
6	The player can move left and right on the screen	Valid: “A” or “D” Invalid: “8”	Valid: The player moves left or right corresponding to the a or d buttons pressed Invalid: The player does not move
7	The player can jump on the screen	Valid: “spacebar” Invalid: “J”	Valid: The player jumps up Invalid: The player does not move
8	The player can interact with entities	Valid: “E” Invalid: “O”	Valid: The player interacts with the entity and the dialogue is loaded Invalid: Nothing happens until the valid input is given
9	The player can return from the dialogue	Valid: Return button press Invalid: “backspace”	Valid: The player returns to the level Invalid: Nothing happens until the valid input is given
10	The player can choose option 1	Valid: Button 1 press Invalid: “1” on the keyboard	Valid: The dialogue moves to the next dialogue scene under the response number 1 Invalid: Nothing happens until the valid input is given
11	The player can choose option 2	Valid: Button 2 press Invalid: “2” on the keyboard	Valid: The dialogue moves to the next dialogue scene under the response number 2 Invalid: Nothing happens until the valid input is given
12	The player can choose option 3	Valid: Button 3 press	Valid: The dialogue moves to the next dialogue scene

		Invalid: “3” on the keyboard	under the response number 3 Invalid: Nothing happens until the valid input is given

Development

The development process will be documented (as much as possible) linearly, in time with when aspects are developed. Milestones will, however, be revisited to make the development references more coherent and there will be stated in the section, what was changed/added.

Milestones:

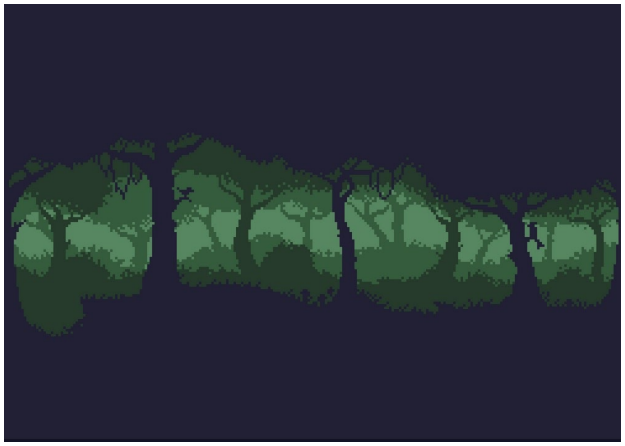
Main menu + Options

20th June - Start

Explanation

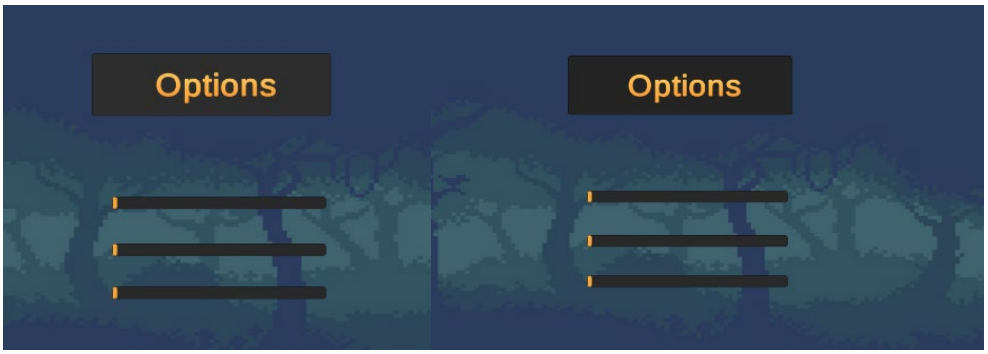
My first goal of my development process is to have a functioning main menu. With working buttons leading to their respective unity scenes.

I conferred with my client and we decided on a free 8-bit stock image for the background of the main menu.



I then set about working the text and button design:

The buttons are programmed so that when they are clicked or hovered above, the rectangular background becomes more apparent for a visual response.



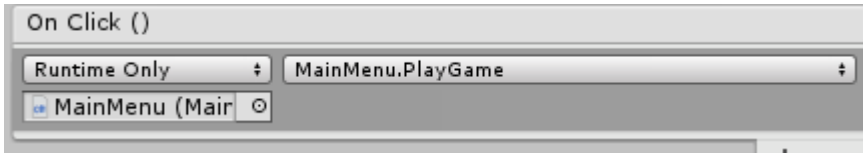
This is the main menu design:

After this I created a script for the main menu to give the buttons functionality.

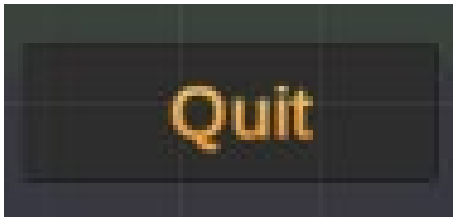
The PLAY button heads to the next scene.

I made this line of code a little more complex than it needed to be for a reason. Instead of moving to the next scene directly, the build index is increased. This leaves it open to including saves and loads later on:

```
public class MainMenu : MonoBehaviour
{
    public void PlayGame()
    {
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
    }
```



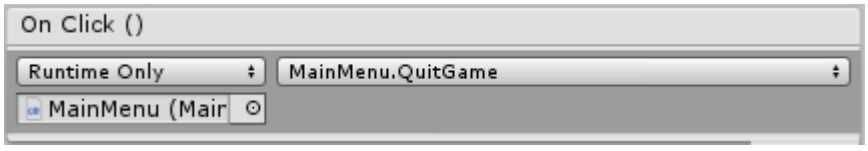
This is simply attaching the subroutine with the “on click” function of the button labeled “Play” so that it is called when the button is pressed.



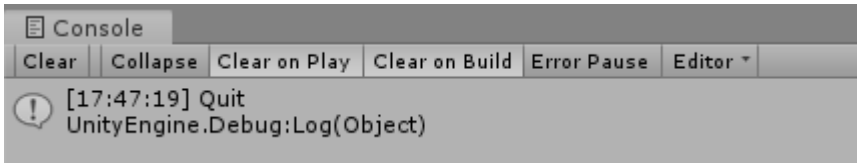
The subroutine below terminates the program and sends a debug message called “Quit” to the console log (this is made more complex so that it is easier to test that it is functional).

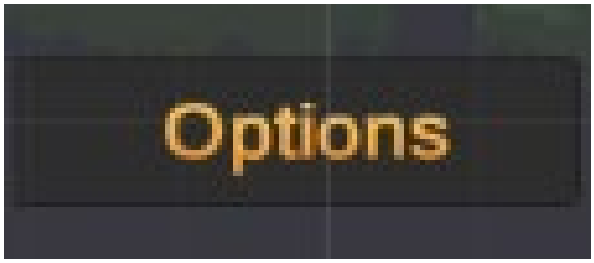
```
public void QuitGame()
{
    Debug.Log("Quit");
    Application.Quit();
}
```

The next screenshot is also simply the attachment of the subroutine with the “on click” function of the button labeled “Quit” so that it is called when the button is pressed.

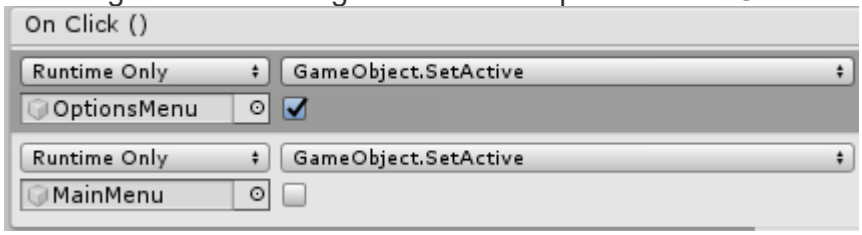


Testing for the the quit debug in the console:

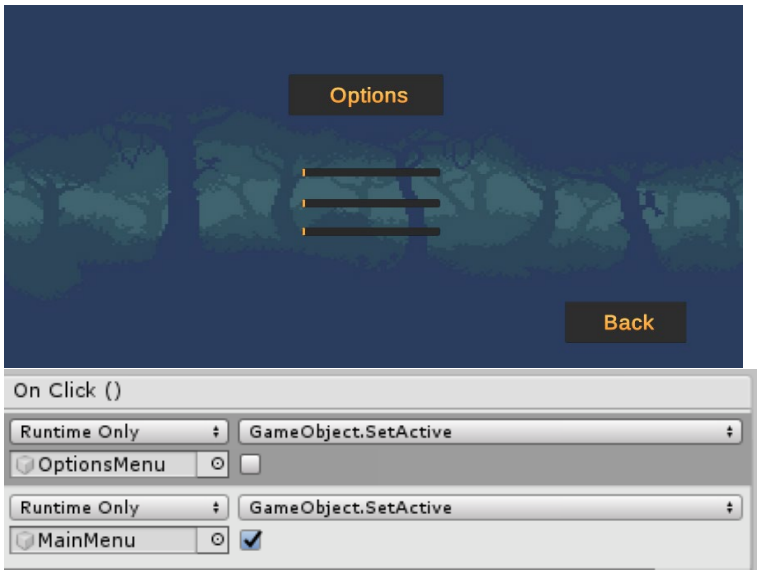




The options button's "on click" function is used to switch the UI to the "options UI" by selecting and deselecting the main and options menu UI elements:



Also, within the options UI I added some placeholder sliders for when sound is implemented and a back button which does the reverse of what the options button did (shown below).



Testing

Testing is done throughout the development, stated in the explanation and put into this table.

Test number	Test	Success criteria link	Expected outcome	Actual outcome
1	Play button	1.1	Enters the first scene	Starts the game
2	Options	1.1	Hides the first main menu buttons	Shows the options menu
3	Exit	1.1	Terminates the program	Exits the game
4	Background	1.2	The menu has a background	The background appears behind the UI

Variables

No variables were used when creating the UI

Conclusion

What went well

All development of the main menu was done in one long run. This was a swift development cycle and no bugs were encountered. The time saved can be used to polish other aspects of the game later down the line.

Even better if

The main menu has not yet had any music added to the scene and therefore does not meet success criteria point 1.3, however I may remedy this if there is spare development time later in the process.

Changes from original

The control layout is not included in the options and there is no continue button.

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
1	There must be a main menu with selectors for the player to navigate in and out of the game and to adjust settings if any are available	Yes	There is a main menu with an options menu and the relevant placeholders for options to be added later down the line as well as a quit button which ends the program and finally a play button to take the player into the game.
1.1	There must be Contine, New game, Options and Exit game buttons	Somewhat	The new game and continue buttons were merged into one as there is no loading system but all other buttons are present and working.
1.2	There may be background art to show the theme of the game	Yes	There was a pixel art image that the client and I decided on to use as a background which fit the mood well
1.3	There could be background music to help set the scene	No	At this point in time, the music has not been put into the game as it is an optional success point and takes up too much time at the start of development, this will most likely be completed at a later stage.

Client sign off

28th June

I had already conversed with the client at the start of this milestone about the use of a free image for the background. Once the development was finished I presented it back to her and she was happy with the fast progress I had made with all functionality in place. We agreed that sound could be implemented at a later stage when polishing and refining the program.

The client gave me the go ahead to move to the next stage.

References

URL	What it was used for
https://www.youtube.com/watch?v=OD-p1eMsyrU	Learning about UI design features in Unity

Movement

29th June

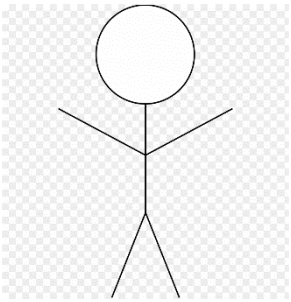
The next milestone was to get my character movement ready for platforming.

From the design, the movement and platforming has been merged into one milestone as they encompass the same parts of the program.

Explanation

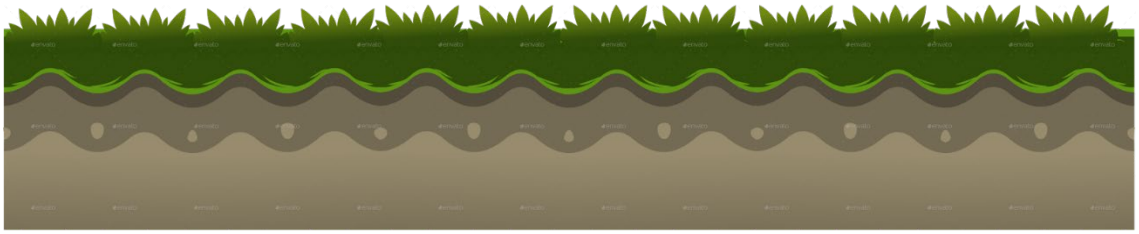
As i had not yet designed a character, i took a stock jpeg stickman character off of google and used it as my player sprite(this will not be used in the final game).

Placeholder sprite:

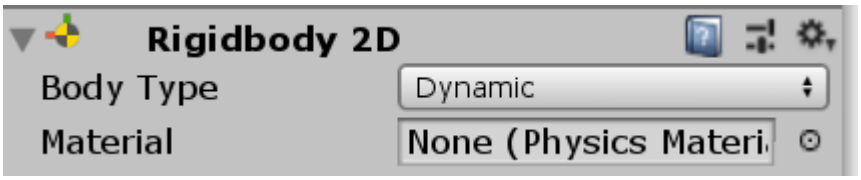


The player has nothing to stand on as well, so I took a stock “ground” image and used it for my character to test the movement on (this will also not be used in the final game).

Ground sprite:



For my player to be able to “move” it needed a 2D rigidBody. Rigidbodies are components that allow a GameObject to react to real-time physics. This includes reactions to forces and gravity, mass, drag and momentum. This can be attached to the entity in the unity editor.



Testing

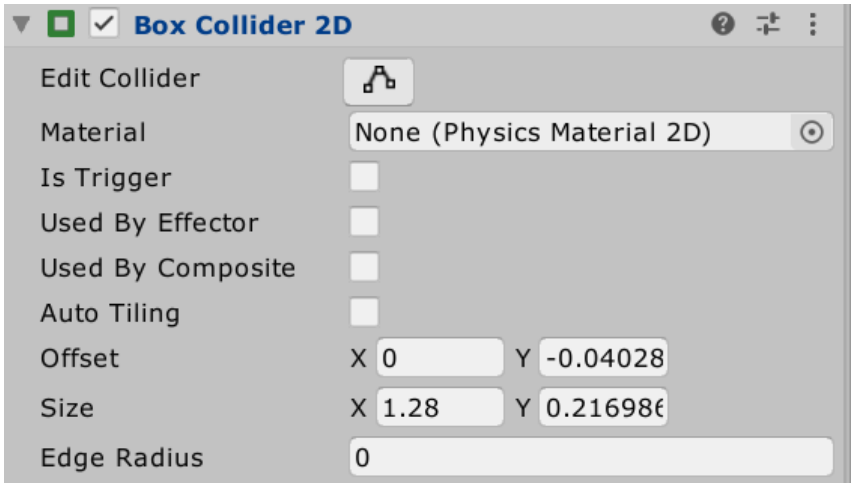
Test number	Test	Success criteria link	Expected outcome	Actual outcome
5	Mass	4	The player has mass	The player fell endlessly

Collision

After attaching the rigidbody to the sprite, the player was now falling. Showing me that it had mass. However, it fell through the ground as the ground had no collision.

To remedy this, I added a 2D box collider to the ground sprite and a 2D capsule collider to the Player (as it fits better to the sprite).

The 2D box collider will be added to any entities the player can move on or are physical such as walls, floors and platforms.



Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
6	Collision	4	The player has collision with rigidbody entities	The player landed on the ground sprite

Control

Now I could get to work on the movement script:

```
Unity Message | 0 references
private void Start()
{
    canJump = true;
    rb = GetComponent<Rigidbody2D>();
}
```

In the start subroutine I set the boolean variable “canJump” to True which is defined when the subroutine is called and set the rigidbody variable called “rb” to call on the RigidBody attached to the sprite. This was put into a variable so that it is quicker to type out to make the development more efficient.

```
public class Move2D : MonoBehaviour
{
    public float movementSpeed = 1;
    public bool canJump;
    private Rigidbody2D rb;
    private float jumpForce = 9f;
```

This is where I define all of my variables. I keep all of my variables in one place as much as i can so that it is more organised and easier to read.

In the jumpForce variable , the “f” after 9 makes the float a force.

Next, the “update” subroutine gets called every frame. Therefore, this is where I put the movement inputs and what they do as they are constantly changing depending on the valid inputs.

```
18      Unity Message | 0 references
        private void Update()
19      {
20          var movement = Input.GetAxis("Horizontal");
21          transform.position += new Vector3(movement, 0, 0) * Time.deltaTime * movementSpeed;
22
23          if (Input.GetKeyDown(KeyCode.Space) && canJump)
24          {
25              rb.AddForce(new Vector3(0f, jumpForce, 0f), ForceMode2D.Impulse);
26          }
```

Firstly, within the update subroutine, I created a “movement” variable which is set to whatever keyboard input pressed which relates to horizontal movement: aka “W” and “D” and the arrow keys. This is so that I can add directional force/movement relating to the direction the user has inputted.

I then transformed the position of the player depending on the movement variable and the vector3 which represents a point in space or a relative direction. I then multiplied it by Time.deltaTime as the movement needs to update relative to the current frame rate so that movement is smoother at lower frames. I then also multiplied it by the movement Speed float variable to give it momentum which will give the sense of speeding up and slowing down. I aimed for this to reach the success criteria of point10 making the movement mechanics smooth.

The If statement checks if the spacebar is pressed AND if the player is touching the ground with the boolean ground check variable “canJump”.

If these conditions are met it adds an upwards force to the rigidbody (variable made before to save long complex lines of code) making it jump.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
7	Left and right movement	4.1	The player character is moved depending on the valid inputs.	The player moved left and right.
8	Jump	4.2	The player character is given vertical directional force if space is pressed.	The player jumped
9	Jump validation (Ground check)	4.2	The player cannot jump if they are not touching the ground	The player cannot jump in midair.

Ground check

Note: The ground check testing happened above^^

The reference to the “canJump” boolean variable in the control section is explained here.

Here is the groundCheck script which is placed on any ground or platform entity which the player can jump off of.

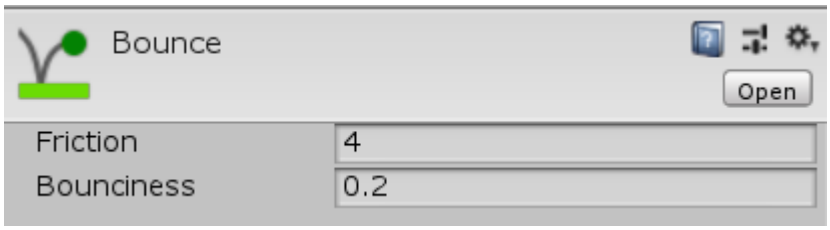
```
public class GroundCheck : MonoBehaviour
{
    Unity Message | 0 references
    private void OnCollisionEnter2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            collision.gameObject.GetComponent<Move2D>().canJump = true;
        }
    }
    Unity Message | 0 references
    private void OnCollisionExit2D(Collision2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            collision.gameObject.GetComponent<Move2D>().canJump = false;
        }
    }
}
```

The groundCheck script uses two subroutines called when any game object enters and exits collision, IF the player is the one colliding with another entity then it sets the boolean function “canJump” (used in the movement script for jump) to true or false to its respective use. (If the player is on the ground it can jump and if not it can't).

I created this in a separate script so that it can be used modularly. This is placed on any entity that the player can jump off.

Bounce

To make the character controls feel more energetic I added a 2D physics component called bounce which allows the character to bounce along the floor after applying jump force. This was a premade component.



The parameters can be adjusted when testing the component to smoothen out the movement.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
10	Bounce	10	The player bounces	The player bounced high and stopped the player from progressing quickly as they cannot jump.
11	Bounce (adjusted)	10	The player bounces at a reasonable level	The player bounced once shortly and the player could progress

Variables

These are the variables used in all parts of the milestone.

Name	Variable type	Reasoning for name	Reasoning for use
canJump	Boolean	It is checking whether jumping is allowed	It is used to validate if the player can jump
rb	Object	Short for rigidbody	To make coding faster by making it shorter to type out.
movementSpeed	Integer	The pace of the player movement	To determine the speed at which the player moves
jumpForce	Integer	It is the force of the jump	To determine the amount of upward force/height the player is given
movement	Implicit local variable (VAR)	Which direction the player is moving	It holds the directional input from the player

When naming my variables, the format I use is a lowercase first word and uppercase for any additional words. This is known as camelCase formatting.

Conclusion

What went well

The adjustments of the bounce were accurate. All movement flows together well and all success criteria points were hit except 4.0 which adds challenge. However this will be made when the levels are designed.

Even better if

The movement, although smooth enough and working, could still be a little less rigid however, this will take a complete rework of the directional control. This would mean an addition of much more math which the development deadline does not have time for.

Changes from original

An additional component of bounce was added to make the movement feel more alive, the client will have to sign this off to finalise the addition.

The optional wall climbing mechanic was skipped.

The movement and platforming has been encompassed over one milestone just called movement

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
4.1	It must have player directional control	Yes	The player is able to move left and right in the game using either the A and D keys or left and right arrows.
4.2	It must have a jumping mechanic	Yes	The player can jump off of surfaces using the spacebar and cannot jump endlessly in midair.
4.3	It may have a wall climbing mechanic	No	The wall climbing was skipped to save time as it is an optional point of the success criteria to reach

Client Sign off

13th July

During a meeting with the client I asked firstly, their thoughts on the bouncing mechanic. They said that it seemed to compliment the original movement platforming loop and character of the player. With this, the addition was finalised.

I also enquired about not including the wall climbing mechanic. She was hesitant at first as she was hoping that it would add to the interactiveness of the game and

make it more enjoyable for players. However, I countered that with the argument for simplicity making it easier for more inexperienced players combined with the faster development speeds. She agreed and signed off.

I was given the go-ahead to move on to the next stage.

References

URL	What it was used for
https://www.youtube.com/watch?v=UbPiCgCkHTE&list=PL3-vkaonFpj9hr1LXge2D0CddnaB8AHUo&index=16&t=1038s	Collision and movement

Camera movement

COMPLEX

13th July

Rather than attaching the camera to the player and have it follow statically I created a script which has the camera separate and follows the player on a delay. This decision was made with the client at the last meeting after she signed off to move onto the next milestone.

Explanation

```
Unity Script | 0 references
public class CameraFollow2d : MonoBehaviour
{
    public Transform target;
    public float damping = 1;
    public float lookAheadFactor = 3;
    public float lookAheadReturnSpeed = 0.5f;
    public float lookAheadMoveThreshold = 0.1f;

    float offsetZ;
    Vector3 lastTargetPosition;
    Vector3 currentVelocity;
    Vector3 lookAheadPos;
```

These are the variables (most of them) defined at the start of the script as common practice in all of my code. This is for organisation and efficient development.

This script was much more complex than the others and therefore my organisation was extremely helpful.

Note: All of the force variables are made public for ease of change in the Unity editor.

Note: I tend not to comment on my code as I find it clutters up the view of the code and confuses my development process.

```
Unity Message | 0 references
18  private void Start()
19  {
20      lastTargetPosition = target.position;
21      offsetZ = (transform.position - target.position).z;
22      transform.parent = null;
23  }
```

This is the start subroutine which runs when the script is called (this runs only once). It puts the target's (the players) position into a variable, sets the offsetZ variable to the distance between the camera and target, and nullifies transforming the parent (camera) for the first instance.

I created this subroutine to put all the variables in the states which they need to be in at the start of the script. This is useful so that if the script is run a second time, the variables revert back to the starting states so that no bugs are encountered involving old data.

The rest of the script is within the update subroutine, updating every frame and running the entirety of the code again. As previously mentioned in the moment milestone, I use the update subroutine for the "moving parts" of the game as it is run constantly.

```
24  private void Update()
25  {
26      float xMoveDelta = (target.position - lastTargetPosition).x;
27      bool updateLookAheadTarget = Mathf.Abs(xMoveDelta) > lookAheadMoveThreshold;
28
```

Starting the update subroutine, a float variable called xMoveDelta identifies the difference in distance between the target's current position in that frame and its position in the last (How far it has moved). A boolean variable called updateLookAheadTarget checks if the distance moved is greater than the threshold for looking ahead.

These two variables are important. They are included in the update subroutine as they need to be reset every frame, they are used in the equations to move the camera and therefore cannot contain old data.

```
29      if (updateLookAheadTarget)
30      {
31          lookAheadPos = lookAheadFactor * Vector3.right * Mathf.Sign(xMoveDelta);
32      }
33      else
34      {
35          lookAheadPos = Vector3.MoveTowards(lookAheadPos, Vector3.zero, Time.deltaTime * lookAheadReturnSpeed);
36
37      }
```

These statements use the last boolean variable “updateLookAheadTarget” to determine if it should adjust the look ahead position. This happens **IF** the player has moved further away from the camera’s last position. **ELSE** it will adjust it back towards the player if they have moved closer to the camera to return it to its default position.

This code and the next took the longest to figure out due to its complexity and extensive research was used. The references were not all documented but the most visited websites will be referenced.

```
39          Vector3 aheadTargetPos = target.position + lookAheadPos + Vector3.forward * offsetZ;
40          Vector3 newPos = Vector3.SmoothDamp(transform.position, aheadTargetPos, ref currentVelocity, damping);
41
42          transform.position = newPos;
43          lastTargetPosition = target.position;
44      }
45
46  }
```

This last part of the script includes a lot of math and adding and multiplying variables. Put simply, it creates a vector3 variable (aheadTargetPos) of the target's position ahead of the camera and then uses that variable in the math to create the newPos vector3 variable which uses the dampening and velocity variables to smoothen out the speed of the camera.

Then it makes “transform.position” to actually move the camera in game. It also updates the camera with “transform.position = newPos”. This uses the just made variable “newPos” and the last target position to the current frame for the next update frame too.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
12	Dampening	N/A	The higher the dampening, the smoother the movement of the camera	The camera is less jagged when moving
13	Look ahead threshold	N/A	The higher the look ahead factor, the further the camera looks ahead of the characters position it is moving towards	The camera looks further ahead of the player
14	Look ahead factor	N/A	The speed of the camera moving to look in front is increased	The camera looks in front of the player quicker when the player starts the move

15	Look ahead return speed	N/A	The speed of the camera returning to the player when there is no movement is increased	The camera returns faster

Variables

Name	Variable type	Reasoning for name	Reasoning for use
damping	float	It controls the dampening	Stores the value which the camera should be dampened by
lookAheadFactor	float	It is the amount the camera looks ahead	Stores the distance which the camera can look in front by
lookAheadReturnSpeed	float	It controls the speed of the camera returning to the player	Stores the number which determines the speed at which the camera returns to the player
lookAheadMoveThreshold	float	It controls the velocity of the camera moving to look ahead of the player	Stores the velocity at which the camera heads to the maximum distance which it can look ahead by
offsetZ	float	It is the offset of the z coordinate	Stores the coordinates of the offset
lastTargetPosition	Vector3 (float)	It is the last position the camera was aiming for	Stores the coordinates of where the camera was last targeting
currentVelocity	Vector3 (float)	It is the velocity at the point in time it is created	Stores the velocity of the camera at the point in time it is updated (every frame)
lookAheadPos	Vector3 (float)	It is the position the camera is looking ahead to	Calculates the overall aim of where to look ahead

xMoveDelta	float	The movement along the x coordinate (delta to show a 2d movement)	Calculates the distance between the players current and last position
updateLookAheadTarget	bool	It updates whether the	Checking if the target has moved or not
aheadTargetPos	Vector3(float)	The target position ahead	Calculates the actual position the camera is aiming for
newPos	Vector3(float)	The final new position	Holds the final calculated coordinates for the camera to move to.

Conclusion

What went well

The camera is extremely smooth, it flows with the movement of the game. This helps me reach my target of smooth movement which was point number 10 in the success criteria.

Even better if

The time taken to research and develop, what in a non-essential part of the game, was much longer than anticipated, next time I would have focused on the integral parts of the game before

Changes from original

The camera movement was not originally in the design phase but was a necessary addition to hit the success criteria.

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
10	The movement should be smooth	Yes	The camera flows with the character movement rather than rigidly to smoothen the movement feel

Client Sign off

3rd August

The client was happy with the extra work and complexity I used in this milestone. There was little to discuss as it matched all expectations.

The client signed off and I could move to the next stage.

References

URL	What it was used for
https://answers.unity.com	This has past problems and fixes to questions on the site and I could learn from other peoples issues and fixes.
https://stackoverflow.com	Similar to above, however I used this site more.
https://www.youtube.com/watch?v=u0DTaaUtRYQ	Example of camera movement- wasn't used
https://www.youtube.com/watch?v=BQEsbOALKhc	How to do long arithmetic and use variables inside of them

Sprite design

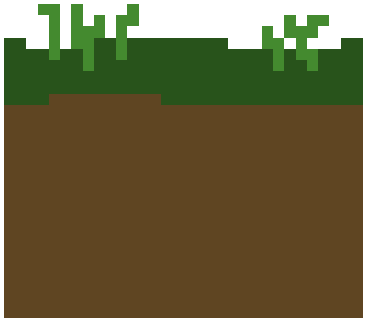
10th August

Explanation

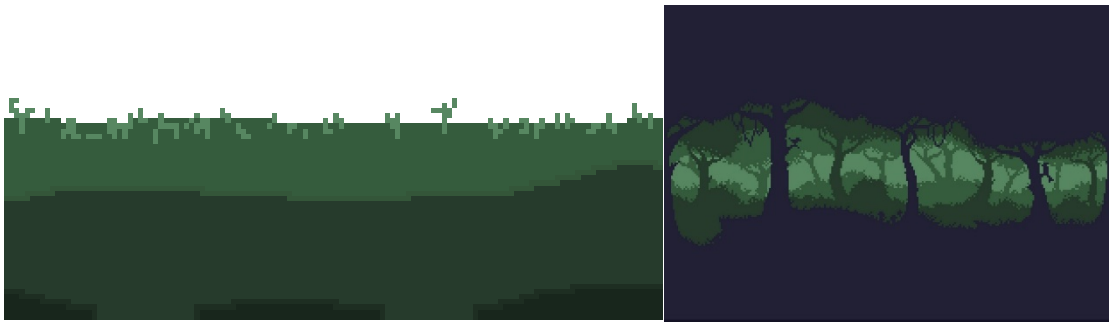
Ground

The first area of the game is going to use the same background of the main title so that it is coherent to the player when they first enter as there is no contrast in the graphic design from the boot-up. This is important for first impressions. Therefore, I needed to make a custom ground sprite for the character to walk on.

I designed it on photoshop and created a pixel grid. However my first implementations pixel grid was too small and it needed to be wider, it looked like this:



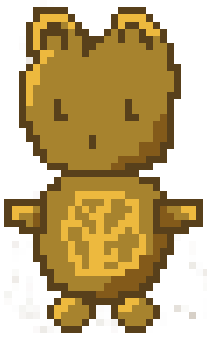
This also did not fit the theme of the first scene as the colour pallet was incorrect. With the next implementation i remedied this and also widened the pixel grid and designed this which I was happy with:



Above is the background for demonstration of the better colour pallet used.

Player character

Next I decided to create the player character sprite.

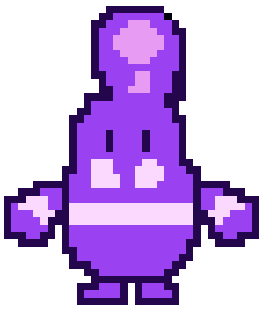


The design for the player character was inspired by the background of the game. Climbing out of a wooded forest made me think of the nursery rhyme “teddy bear’s picnic” and the lines from it “If you go down in the woods today, You're sure of a big

surprise”. On that basis I created a round bodied teddy bear sprite with a design on his chest representing a tree.

However.

I wasn't very happy with the teddy bear character as it seemed very generic and didn't stand out from the game enough. Therefore, I went back into photoshop to design a second character and this is what I came up with:

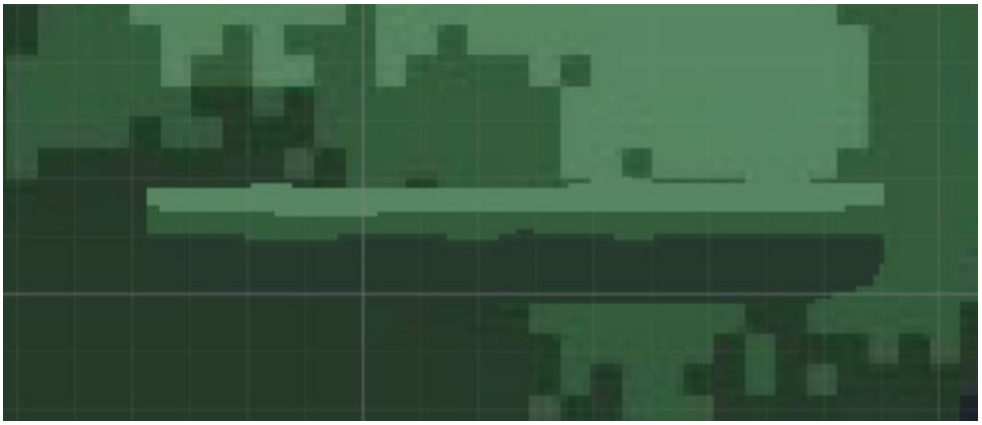


I liked this design much better as the purple coloring stood out from the game’s colour scheme but did not take away from it. It seemed to subtly stand out but not to the point where it became a distraction from the gameplay. This design also had much more personality to it in my opinion so I decided to swap the character designs and make this the main character.

Jump Platforms

I then needed platforms for the player to use to traverse the level and jump from.

I first designed a platform similar to the ground. However this turned out to be a bad decision as the colour scheme was exact to the background which made the relative colours appear transparent when placed over the background:

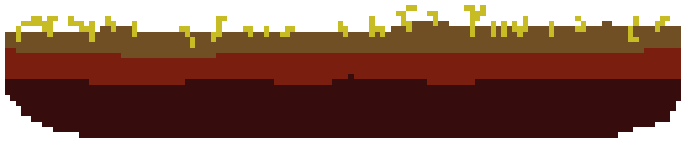


I was happy with the shape of the original platform, so I kept that for the second version to save time and used a new colour pallet which contrasts with the background more to recolour the sprite and this is the end result:

I will most probably keep the 1st iteration to be used in a different scene.

Additions

For the later levels of the game, I edited the platforms giving them other distinct features and colour schemes to suit the backgrounds that they met. These are the designs:



Boss

The game needs to have a boss which the player can converse with.

This is the design I came up with:

The character is meant to look raged and like part of the environment to make the player imagine that they have been there for a long time and have blended into the environment through long periods of evolution

Campfire

This is the sprite design for the campfire for the boss to be sat by to make the game feel a little more alive.

Testing
Testing is not applicable for the sprites as they have no code attached to the design.

Variables
No variables were used

Conclusion

What went well
The art style is consistent and is all completely original (hitting point 11 of the success criteria). All the sprites are easily understood in regards to their purpose which is good for players as it makes the game more accessible and easier to learn.

Even better if
There could be more sprites if the game was longer and introduced more characters. This would “flesh” out the game and make it come alive more.

Changes from original
There were no plans on how to design the sprites which was an oversight in the design section, therefore all concepts had to be created during development which ate away at the time that could have been spent developing them quickly.

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
11	It could have original artwork	Yes	All of the artwork in the game bar the backgrounds are palletized and designed by me
7	There must be characters and items to interact with	Somewhat	I designed the characters which will be interacted with later but no items as I have not
8	There must be bosses to encounter and defeat	Yes	I created one boss sprite which will be reused for each end segment of the platforming

Client sign off
The client was much more involved in this stage of the development as it was not discussed in design. Many times we came together to talk about the time being spent on the designing. The client was happy for me to use more time than she hoped on this stage as the game will be jarring if the artwork is mismatched.

She was happy with the final result and signed me off to the next milestone.

References

URL	What it was used for
https://www.youtube.com/watch?v=rLdA4Amea7Y&list=PL3-vkaonFpj9hr1LXge2D0CddnaB8AHUo&index=4	Learning the basics of creating pixel art in photoshop
https://www.youtube.com/watch?v=k0jIBbmwvGk&list=PL3-vkaonFpj9hr1LXge2D0CddnaB8AHUo&index=35	Learning how to import sprites into Unity

Animation

Explanation

Animation is not very code-heavy but is needed to give the project an element of polish and hit point 9 as well as helping towards point 10 of the success criteria. Animation was a point identified in my analysis of Hollow Knight which I found made the game flow and feel extremely coherent, this was a part to develop the solution to the problem my client presented me with.

The main necessary animation is the player character movement, if I have extra development time on my hands, I may consider animating swaying grass on the platforms and ground.

Player character

An idle and running animation is needed for the player character. So let's start with the idle:

This animation runs when the player character is receiving no inputs. It’s needed to make the player seem alive (they bob up and down as if they were breathing) rather than being completely still. I used adobe photoshop for this and edited the original player character to move slowly up and down over different layers. To then export

this into the game, I first need to turn the different layers into a sprite sheet. I did this with this photoshop script I found on the internet:

```
if (documents.length > 0)
{

        // ---------------------------
        docRef = activeDocument;
        var activeLayer = docRef.activeLayer;

        numLayers = docRef.artLayers.length;
        var cols = docRef.width;

        var spriteX = docRef.width;

        // put things in order
        app.preferences.rulerUnits = Units.PIXELS;

        // resize the canvas
        newX = numLayers * spriteX;

        docRef.resizeCanvas( newX, docRef.height, AnchorPosition.TOPLEFT );

        // move the layers around
        for (i=0; i < numLayers; i++)
        {
                docRef.artLayers[i].visible = 1;

                var movX = spriteX*i;

                docRef.artLayers[i].translate(movX, 0);

        }
}
]
```

The finished sprite sheet looked like this:

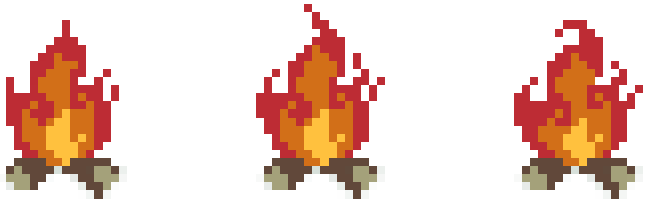
I then created a left running animation:

And then flipped it to create the right running animation:

I then imported these sprite sheets into unity.

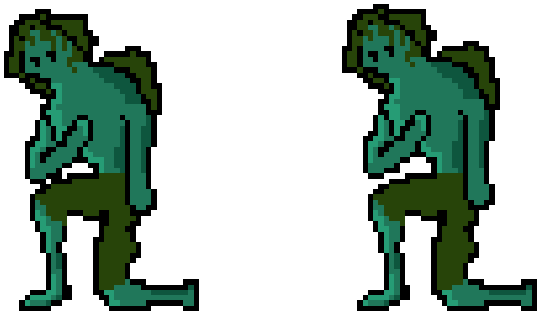
Campfire

I also decided to animate this campfire which needs to look alive for the atmosphere of the game. It was a simple animation as no input coding needs to occur and the fire simply needs to curl upwards. Here is the sprite sheet i created:



Boss

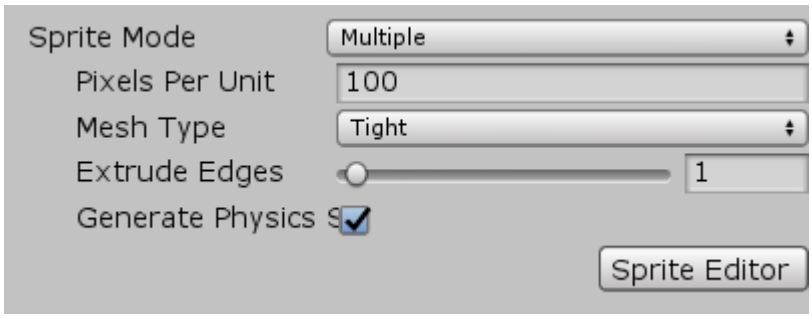
For the boss I created a simple two frame idle animation as he does not need to move but needs to look like he is alive.



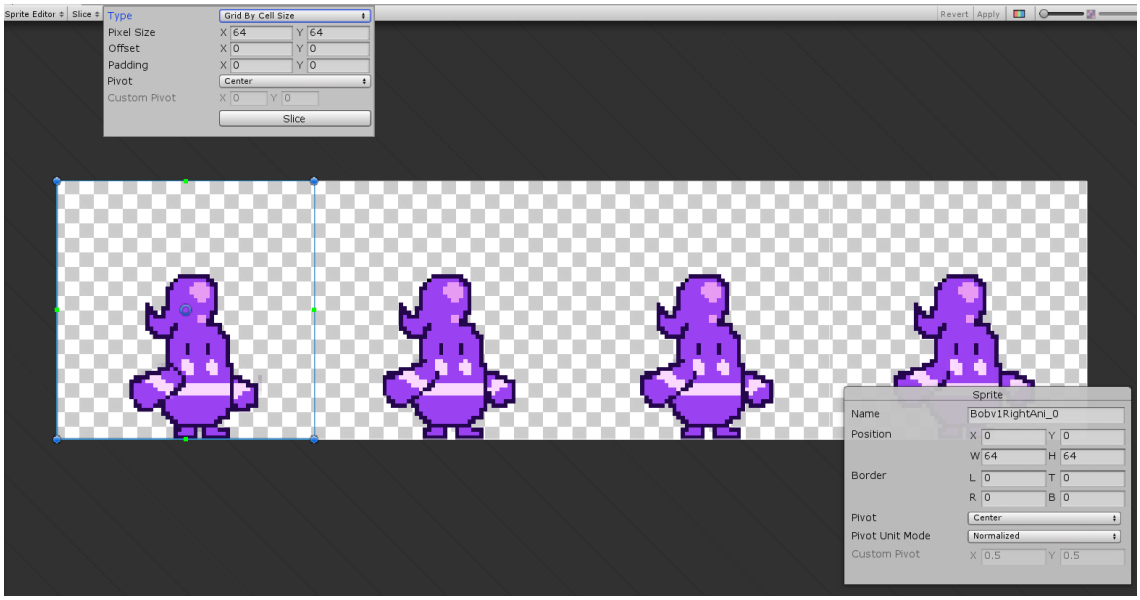
Unity animator

Now that the sprite sheets are imported into unity I am able to start animating and implementing the animations into the game. I started with the player sprite.

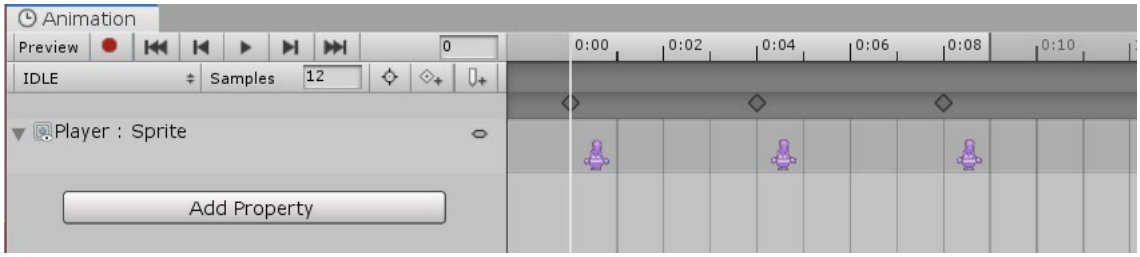
Firstly I needed to split up the sprite sheet into the individual sprites representing each state of the animation. I selected the multiple sprite mode on the asset editor:



Then went into the sprite editor and sliced it into 64X64 pixel grids:



Then I was ready to animate. I entered the unity animation window, imported the sprite sheet and named it to the relevant animation (IDLE in this instance):



I changed the sample rate to 12 so that it fits pixel art animation speeds. This was done so that the player does not look like they are moving at a faster pace than the game (this is helpful for making the game feel more alive and coherent). The individual sprite states were given keyframes so that the sprite being rendered changes at that point in the animation timeline. This creates the illusion of movement and with that the animations were done.

Now that the animations were created, I could attach them to the player character and play them in response to valid inputs. This turned out to be trickier than anticipated. I planned to have the character simply play each animation depending on the player input, however this was inefficient to code. Instead it was done using a new variable called “horizontal”.

The “horizontal” parameter is related to which direction the player character is moving. It is a float variable which I added to the player movement script to pick up the players direction of travel to use for selecting the correct animation. This works better than checking each player input as I originally intended because it uses one simple process to update the data and change the animation.

Here are the additions to the script to implement this:

```
10      public Animator animator;
```

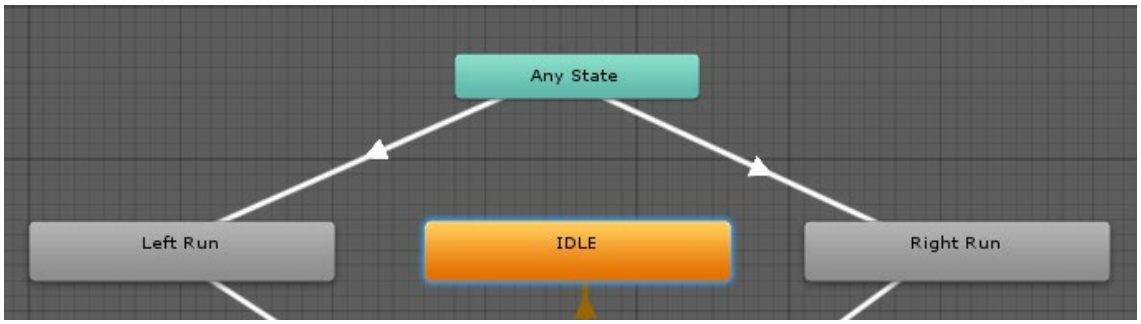
```
19      private void Update()
20      {
21          animator.SetFloat("Horizontal", Input.GetAxis("Horizontal")); //animator horizontal
```

On line 10, the animator in unity is turned into a variable which can be used in the animator:

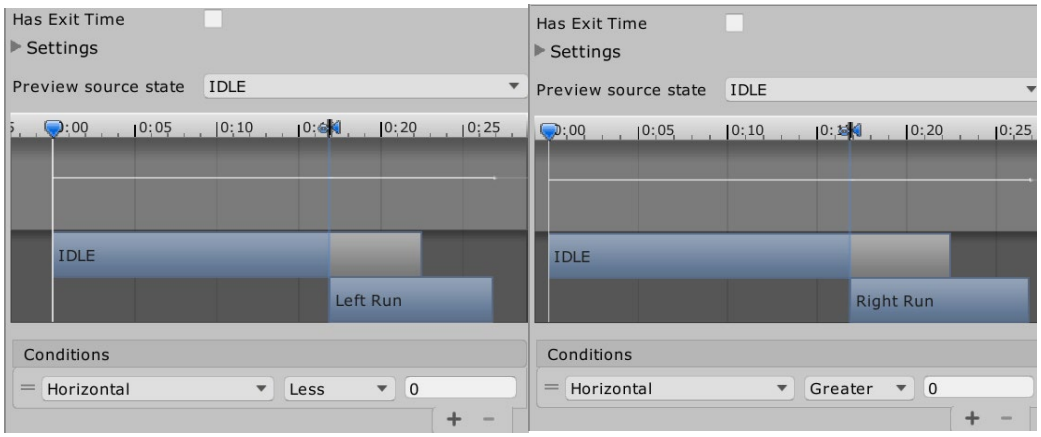
And on line 21 it takes the players horizontal movement and sends the “horizontal” variable to the animator through the just created “animator” variable.

Next, from the starting state (which is when no other states are in use) the player is set to play the idle animation.

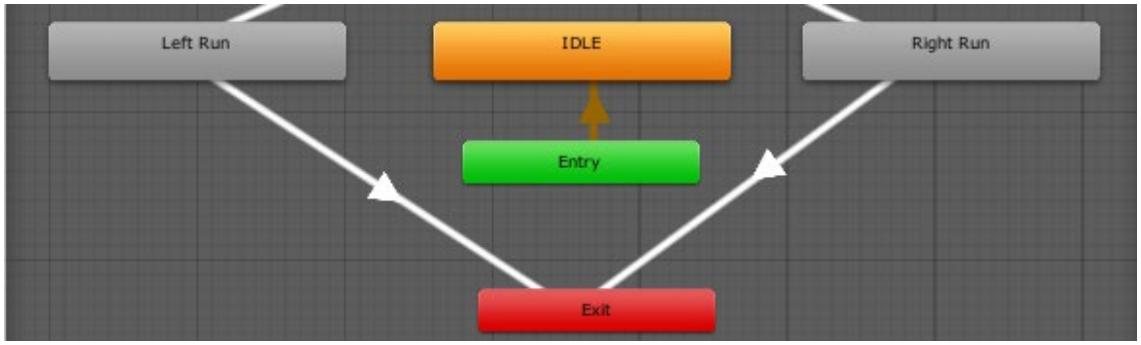
And then from the “any state” option, depending on the player's direction of travel (the horizontal float) it will either play the right or left run animations:



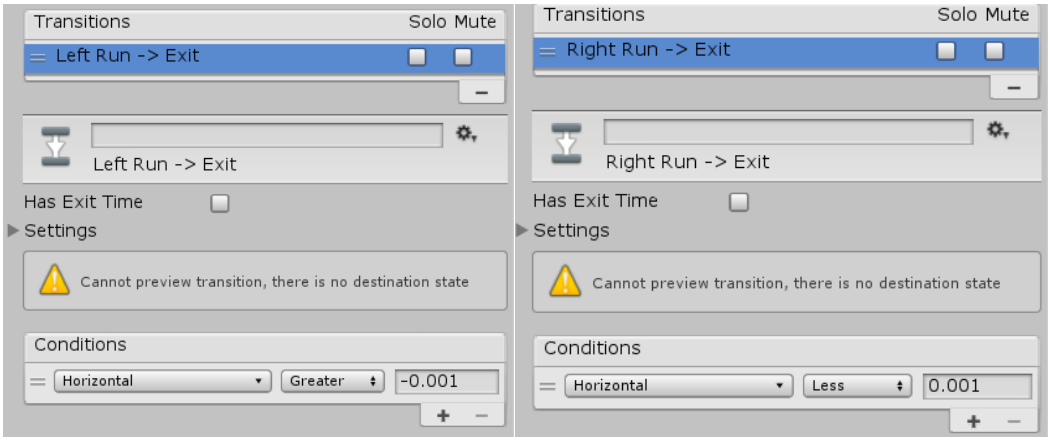
This happens through transitions (the arrows) declared with conditions. The conditions are a way of telling the game engine to run the animation which the transition is linked to IF the condition is met. A simplified IF statement.



The conditions take the horizontal movement number and checks if it is greater/less than 0. This acts as a way of telling the engine that the player is moving left or right (the 0 relates to the x axis).

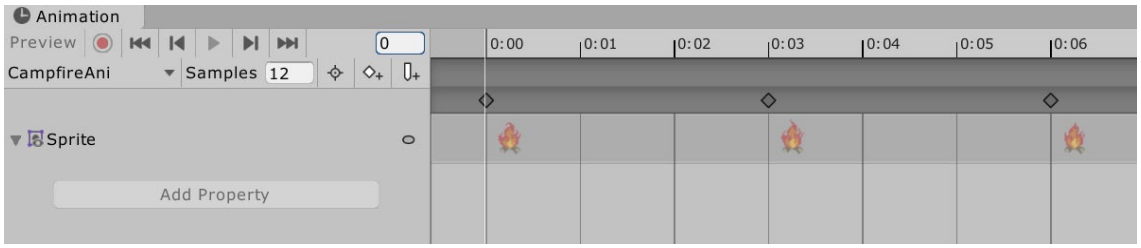


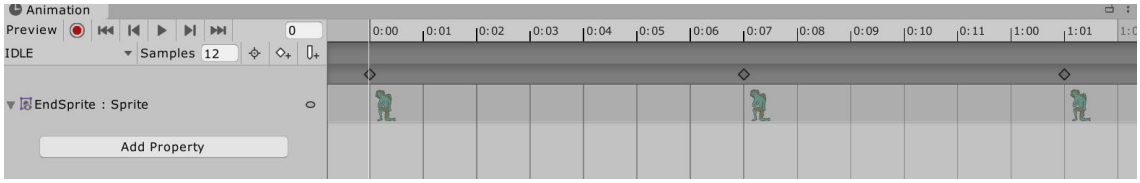
To have the player stop running at some point, I must have transitions with conditions opposite the entrance conditions leading to the EXIT state which simply acts as a stop function:



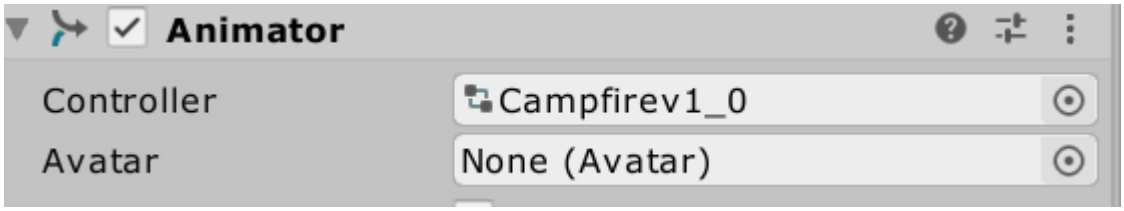
A point of note here is that, instead of simply setting it equal to or less/greater than 0, I set it to 0.001 or -0.001. This was because there is no equality operator available.

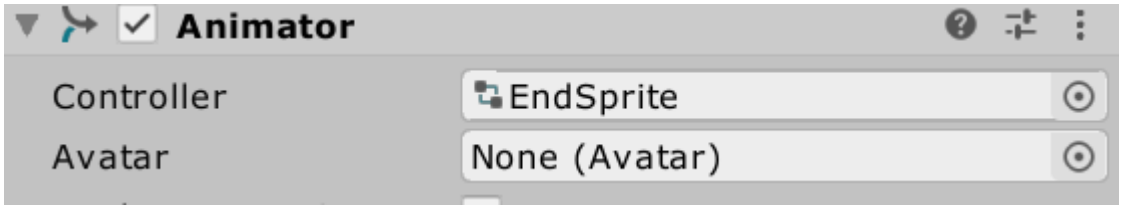
For the campfire and boss it was a similar process, import the sprite sheet to unity, create an animation with the different sprites: (this was called CampfireAni and EndSprite)



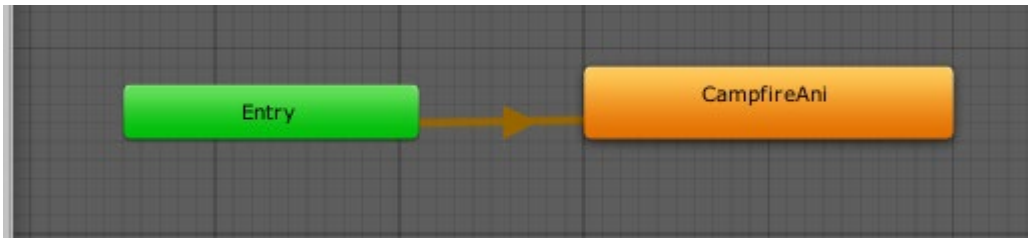


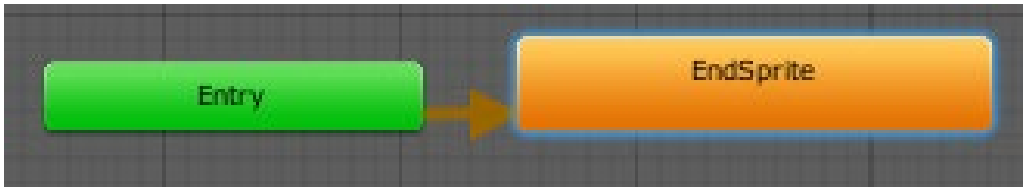
Then attach the animator to the sprite:





And lastly attach the animation to the “entry” statement in the animator controller. As there was no directional movement needed for the campfire, the animation can run constantly from its entry state.





Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
16	Animation stop	9	The animations do not play endlessly when the inputs are stopped	The animation stopped when the movement stopped

17	IDLE	9	The idle animation plays when there are no movement inputs	The player character bobbed up and down
18	Left run	9	The leftRun animation plays when there is left horizontal movements	The player character ran to the right
19	Right run	9	The rightRun animation plays when there is right horizontal movements	The player character ran to the right

These tests will be shown in the evaluation as it is easier to show it working in video form.

Variables

Name	Variable type	Reasoning for name	Reasoning for use
animator	structure	Holds the animator	To reference the animator component and assign variables inside
horizontal	float	Holds the horizontal movement	To check the directional movement for selecting the correct animation

Conclusion

What went well

Completing this milestone marks the end to developing the aesthetics of the main gameplay. This allows me to focus on the internals of the game in the next milestone. All animations turned out well and no bugs were encountered.

Even better if

If time restraints did not exist, I would have liked to develop a jump animation as well so that all movement elements present in the game are covered.

Changes from original

Different approach to playing the animations depending on the horizontal movement as a float rather than the individual inputs.

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation

9	There should be animation	Yes	The animations for all moving objects in the game have been created and implemented

Client sign off
17th August

After showing it to the client, she was happy with the way the animations looked and had no concerns.

She signed me off to the next milestone.

References

URL	What it was used for
https://gist.github.com/jessefreeman/870172	Sprite sheet photoshop script
https://www.youtube.com/watch?v=rycsXRO6rpI&list=PL3-vkaonFpj9hr1LXge2D0CddnaB8AHUo&index=36	Learning how the animator works in regards to movement

Respawn
20th August

This was not planned in the design phase.

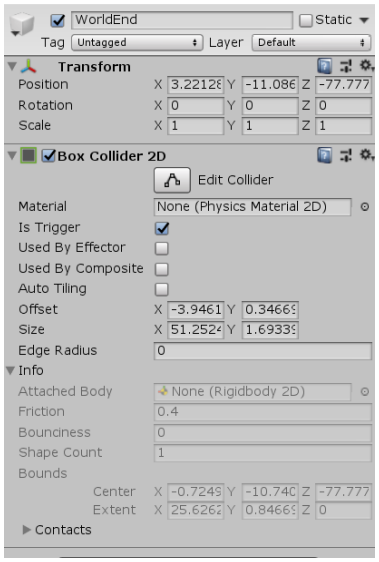
Explanation
For the player not to fall into eternity when they fall off of the platforms, I needed a quick and effective way of returning the player to the start while keeping the player engaged and the gameplay flowing.

I did little research for this part as I believed it to be quite straightforward and wanted to test my coding capabilities. Hence, it took two iterations of this milestone to get it working.

Attempt 1

First I created a box collider trigger and placed it underneath the world and called it “WorldEnd” (this will be used in both solutions). This will act as a “catch” or “death zone” for the player where, if they fall into it they are out of the bounds of the game and a script will be attached to respawn them.

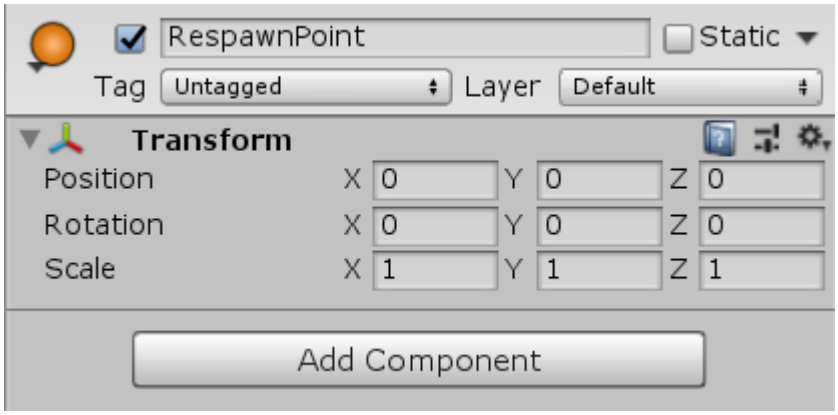
Here is the entity in the Unity editor with the box collider attached:



And here is what the collider looks like in the game:

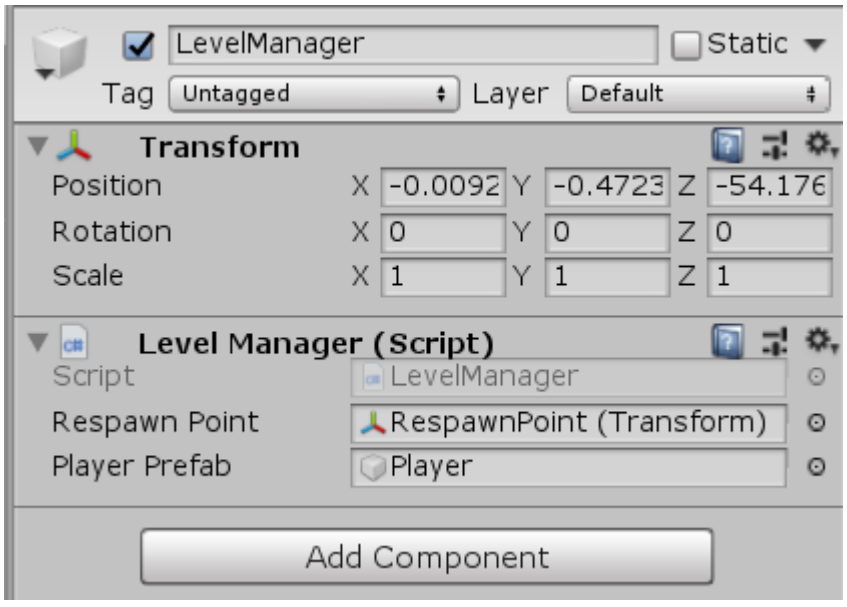
Note: This outline will be invisible to the player.

To make sure that the player can respawn I created an empty game object and gave it a position of where the player should return to when they die and named it RespawnPoint.



I made the respawn point an object rather than a bunch of coordinates because it becomes modular. Changing the location of the object will hence update the new coordinates automatically. This future-proofs the system for maintenance, updates and changes.

The respawn method needed to be on a script. I created a new game object called “level manager” where I thought ahead for future coding as any changes made to the level in runtime can be placed here. For now only the respawn subroutine is located here on the attached script.



Below is the aforementioned script:

As is common practice most variables are declared at the start to keep the code organised.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 2 references
public class LevelManager : MonoBehaviour
{
    public static LevelManager instance;
    public Transform respawnPoint;
    public GameObject playerPrefab;

    Unity Message | 0 references
    private void Awake()
    {
        instance = this;
    }
    1 reference
    public void Respawn()
    {
        Instantiate(playerPrefab, respawnPoint.position, Quaternion.identity);
    }
}
```

The method I took with this script is that it's formed on the basis; when the “respawn” subroutine is called, the player is recreated at the respawn point and in the collision script below, the old player is destroyed.

When this script is called the “awake” subroutine runs which initializes variables and states. This is because instantiation is used to create a new game object and when instantiated objects are created, the variables and states need to restart to run properly.

When the respawn subroutine is called from the script below, an object is instantiated (made) and given the attributes of the player, the position of the respawn point and the quaternion.identity makes it so that the object is perfectly aligned with the world. I included the last attribute to avoid any bugs which may occur when placing the new player.

```
C:\Users\adamt\Desktop\Munth\Assets\Scripts\PlayerDeath.cs
1   using System.Collection
2   using System.Collections.Generic;
3   using UnityEngine;
4
        Unity Script | 0 references
5   public class PlayerDeath : MonoBehaviour
6   {
            Unity Message | 0 references
7       private void OnCollisionEnter2D(Collision2D collision)
8       {
9           if (collision.gameObject.CompareTag("WorldEnd"))
10          {
11              Destroy(gameObject);
12              LevelManager.instance.Respawn();
13          }
14      }
15  }
16
```

This is a collision subroutine with calling conditions identical to the groundCheck script in the movement milestone.

Inside the subroutine I created an if statement which checks if the collision was with the “WorldEnd” object. IF true, the player is destroyed and the respawn() subroutine is called inside the LevelManager. This then goes through the aforementioned code and spawns a “new” player at the respawn point.

This in theory should work, however when testing it did not.

Testing

The link of this milestone to the design success criteria is, “There must be a coherent story with a complication, climax and denouement”. I feel that the respawning of the player is important to the pacing of the game and hence should count towards making a coherent story.

Test number	Test	Success criteria link	Expected outcome	Actual outcome
20	Respawn point	5	The coordinates update to a move in the location	The coordinates updated
21	Destroying the player	5	The player object is destroyed	The player disappears from the screen
22	New player	5	A new player is created	No new player appeared

This last test showed that the code did not work and I could not find any fixes, therefore a new approach had to be taken.

Variables

Name	Variable type	Reasoning for name	Reasoning for use
instance	object	It is the instance in which the script is called	To run the script and initialise variables inside
respawnPoint	Transform (object)	It is where the player will respawn	It holds the respawn point objects “transform” attribute which is the coordinates of the player spawn
playerPrefab	object	It is the players prefabricated attributes	It is an empty game object which

Solution

My first attempt was way too complex in spawning the player and I believe that to be why it was not working, so I tried another approach with similar elements to the first.

I first deleted all Level management scripts and added this code to the players movement script instead of a seperate one to reduce the complexity.

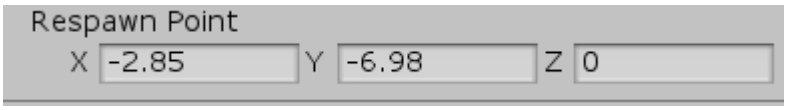
```
Unity Message | 0 references
private void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.tag == "WorldEnd")
    {
        transform.position = respawnPoint;
    }
}
```

The IF statement checks when the object collides with any box collider with the tag “world end”. IF these conditions are met the players position is reset to the same as the variable respawn point which I set as sa vector3 public variable here to be able to adjust in the unity editor:

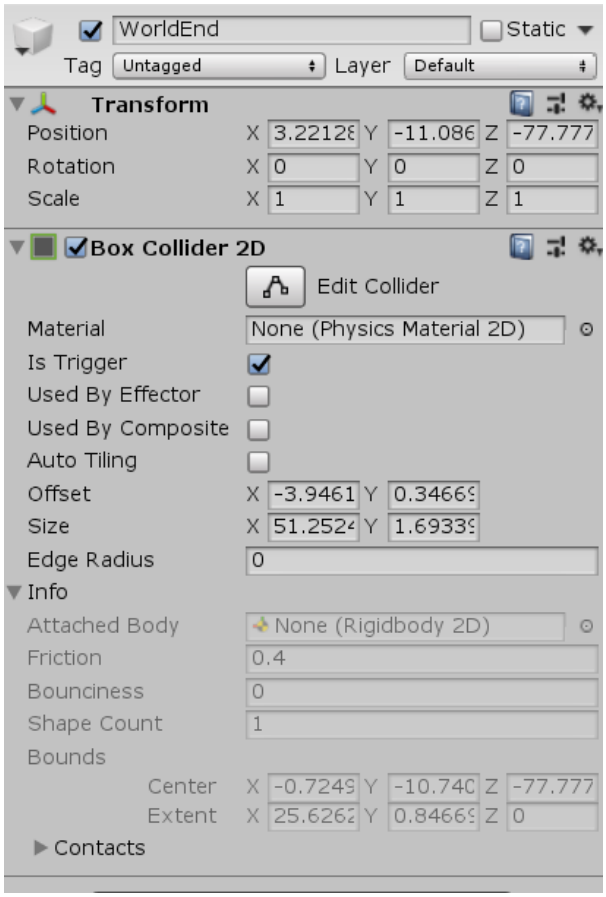
```
public Vector3 respawnPoint;
```

This was not as modular as the solution of putting the respawn point on top of a game object as its location. But as the last iteration did not work, I did not want to take any risks and instead took this linear approach where if any changes need to be made to the respawn location, they will need to be manually entered. This is worse for the future maintenance of the game so a note will need to be made when handed on to future developers.

Although, to help the future developers i made this variable public so that it can be adjusted through the editor. This means that they do not need to dig into the scripts.



I also kept the world's end, collider and the tag WorldEnd so it can be called upon in the respawn function:

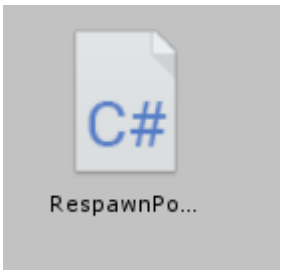


Here's what it looked like in game: (the same as attempt 1)

This method worked much better functionally and was much more efficient than the complex design of the last. Only one script was needed and three lines of code.

One issue with the code; it was inside the movement script. This meant that when the next scene was loaded, the character would respawn in the wrong place related to the position set.

To remedy this, I placed the code in a separate script which contains the respawn for this scene only and then attached it to the player



This has no outstanding issues.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
23	Collision respawn	5	When the player collides with the worldEnd, the players position is transformed	The player is “teleported”
24	Accurate respawn	5	When the player respawns it moves back to the specified coordinates	The player moves instantly to the place specified
25	Respawn Point change	5	When the coordinates of the respawn point are changed, the player is spawned at the new coords.	The player moves instantly to new specified coordinates.

Variables

Name	Variable type	Reasoning for name	Reasoning for use

respawnPoint	Vector3(float)	It's the point in the world the player respawns	Holds the coordinates of the point to transform the player to

Conclusion

What went well

The final solution was extremely efficient using only 3 lines of code. The process I used tested my knowledge and skills extensively which forced me to be more detailed and precise when coding which is good practice for the rest of development

Even better if

It would be better if the modular solution was implemented in the second solution but a working milestone outweighs the modularity.

Changes from original

This is a new milestone created so that the game was playable.

Success criteria review

There were no points to meet on the original success criteria which was an oversight on my behalf.

Client sign off

31st August

When meeting with my client, she showed concerns that there was no downtime for the player after they fail. We came up with the solution of adding a respawn menu alongside the pause menu milestone. This will slow down the gameplay without causing the problem identified in my analysis of disjointed gameplay.

The client was happy with this solution and signed me off to start developing the next milestone

The respawn menu will be developed later alongside the pause menu.

References

URL	What it was used for

https://www.w3schools.com/cs	Learning and checking a few times about the operators I was choosing

Entity interaction

1st September

Explanation

This milestone is to complete point 7 of the success criteria: “There must be characters and items to interact with”.
In the entities section of my design, the interaction mechanic states:

“To interact with the entities in the world, there needs to be a simple pop up telling the player that they can interact with them. Also when the player presses the key the dialogue menu subroutine will need to be executed”

The code below is what I wrote to hit all of these points.

Code

The first piece of code I wrote is the change to the dialogue scene. Many elements in this need to be explained.

```
        if (Input.GetKeyDown(KeyCode.E))
        {
            switch (CurrentTrigger)
            {
                case 1:
                    SceneManager.LoadScene("Dialogue");
                    break;
                default:
                    break;
            }
        }
    }
}
```

Firstly, the code is an IF statement which, if the “e” key is pressed, a similar conditional statement called “switch” checks the cases against a currentTrigger variable which is referenced in another script. If case one is true, the scene is switched to the first dialogue scene.

This is different to the design, as I referenced a subroutine, instead the scene manager is called. This is because I found it much easier to develop the dialogue in a separate unity scene.

The reason I used a switch statement instead of a traditional IF statement is because it is better for reusable programming, more efficient and any number of cases can be added later on in the development which is useful for the expansion and maintenance of the game.

I put this piece of code in the player controller as it requires the “e” input from the player.

The code below is in the same script and allows unity to change scenes, without this the code will break.

```
4   using UnityEngine.SceneManagement;
5
```

This is the nextSceneTrigger script which I placed on the end sprite. The script does also does a number of things:

Firstly, to have the text pop up as stated in the design, the script must have this line of code to use the text element.

```
5   using TMPro;
```

```
public class NextSceneTrigger : MonoBehaviour
{
    public int TriggerNumber;
    public TextMeshProUGUI Message;
    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.tag == "Player")
        {
            collision.gameObject.GetComponent<Move2D>().CurrentTrigger = TriggerNumber;
            if (Message != null)
            {
                Message.color = new Color(0, 0, 0, 1);
            }
        }
    }
```

As always, I declare my variables at the start of the script which helps me keep organisation and clarity through my program. Looking back on the script, I notice that the formatting for the variable names is inconsistent with the rest of the code. This may need to be changed later.

The public integer called TriggerNumber is there so that If i place this code on another object later down the line I can decide its number in the Unity editor. This is useful for reusable programming. And the text mesh pro GUI variable is there so that the text element mentioned earlier can be referenced and changed in the script.

When the player enters the box collider (using it as a trigger) the collision is checked against the player and in the next IF statement, the currentTrigger variable (located in the player script) is assigned to the trigger number of this instance. The script also makes the TMPro text opaque so it gives the impression of appearing when you come near the script and it is used as a prompt to move to the next scene.

The if does not equal null statement is simply a bug preventative as no changes can be made to text which does not exist.

I decided to change the opacity of the text rather than destroying and creating new text elements as the complexity of the code which it would require is unnecessary.

The code below is the inverse of the code above so that when there is no collision, the input does nothing and the message does not appear. It is still in the same script so that only one script needs to be added if more messages need to do the same thing. (All reusable code)

```
Unity Message | 0 references
private void OnTriggerExit2D(Collider2D collision)
{
    if (collision.gameObject.tag == "Player")
    {
        collision.gameObject.GetComponent<Move2D>().CurrentTrigger = -1;
        if (Message != null)
        {
            Message.color = new Color(0, 0, 0, 0);
        }
    }
}
```

As I mentioned in this milestone, the text element I was using was called TextMeshPro, this is a free unity asset which is a replacement for Unity's UI Text and the legacy Text Mesh prebuilt into the engine.

I used this because it has many text appearance and formatting options, and is an easy way to add a professional touch to my user interface. At the client sign off I will need to finalize this decision with my client.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
26	Message appear	7/10	When the player collides with the box collider, the text is made opaque	The text appears

27	Message disappear	7/10	When the player collides with the box collider, the text is made transparent	The text disappears
28	Scene switch	7/10	When the player presses “E” whilst colliding with the box collider, the scene is changed to dialogue1	The scene changes

Variables

Name	Variable type	Reasoning for name	Reasoning for use
CurrentTrigger (wrong formatting)	integer	Holds the current trigger number	To select the case of which scene to switch to
TriggerNumber (wrong formatting)	integer	Holds the trigger number of that specific popup	To change the current trigger in reference to the trigger that is decided in relation to the cases.
message	object	Holds the text message that appears	To reference and change the text displayed as a popup

Conclusion

What went well

All coding in this segment has reusable elements for further development of the game which is useful for expansion if there is time at the end of the project.

Even better if

I should have formatted the variables correctly with the rest of my code as the inconsistency is detrimental to the futureproofing and organisation of the code.

Changes from original

There is a scene change rather than a subroutine being called concerning the dialogue section of the game as I quote “*Also when the player presses the key the dialogue menu subroutine will need to be executed”* . This is still loosely followed and the functionality stayed the same.

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
7	There must be characters and items to interact with	Yes	The interaction between the player and entities has been created and can be put on any new entities added to the game later
8	There must be bosses to encounter and defeat	Yes	The boss is now interactable

Client sign off

4th September

This client sign off was important as I needed to finalize my decision to use text mesh pro. If the client was not happy with this, the milestone will need to be edited and the functionality may change.

However, this was not the case. She approved my decision and was happy with the milestone overall. I was signed off to the next milestone.

References

URL	What it was used for
None	N/A

Dialogue

5th September

The next few milestones are all sub-milestones relating to the larger “dialogue” milestone I planned to do in my design. This is simply to split up this larger task into smaller tasks for me to tackle (decomposition).

States

Complex

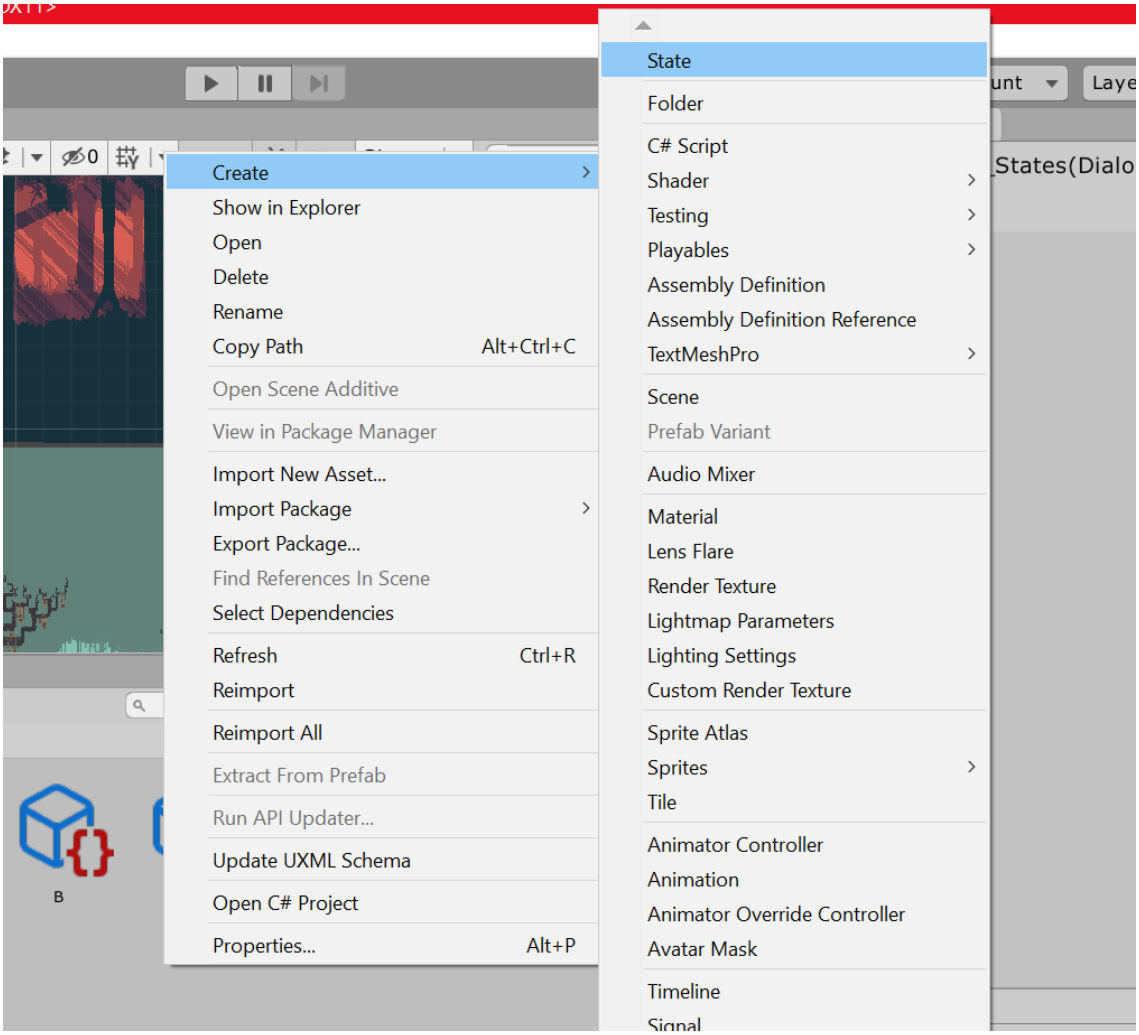
Explanation

Now that the scene has changed to the dialogue scene, I need to design an interactive, text display which moves from one dialogue “state” to another to change the displayed text acting like a new part of the story.

The design I came up with is a reusable programmable game object which I called “state”. Here is the code:

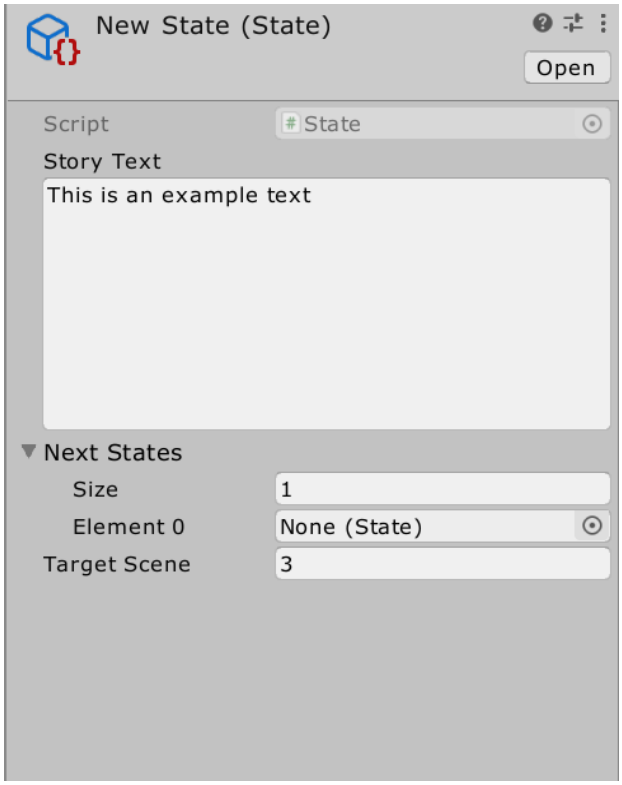
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   [CreateAssetMenu(menuName = "State")]
        Unity Script | 4 references
7   public class State : ScriptableObject
8   {
9       [TextArea(10, 14)] [SerializeField] string storyText;
10      [SerializeField] State[] nextStates;
11      [SerializeField] public float TargetScene;
12
13
            4 references
14      public string GetStateStory()
15      {
16          return storyText;
17      }
18
            1 reference
19      public State[] GetNextState()
20      {
21          return nextStates;
22      }
```

Line 6 creates an asset out of this script which I can use in the unity editor: (reusable programming)




```
9     [TextArea(10, 14)] [SerializeField] string storyText;
10    [SerializeField] State[] nextStates;
11    [SerializeField] public float TargetScene;
```

Lines 9, 10 and 11 all create part of the game object and are assigned variables. Line 9 creates a text space for the story text to be written. Line 10 creates a link to the next state(s) the story goes to and line 11 creates a float for the option to change the scene. Below is an example state.



Lines 14-17 is a subroutine which returns the story text box, which can be written into (shown above) , as a string called storyText..

```
4 references
14  public string GetStateStory()
15  {
16      return storyText;
17  }
18
```

Lines 19-22 is a subroutine which returns the next state which the initial state is linked to.

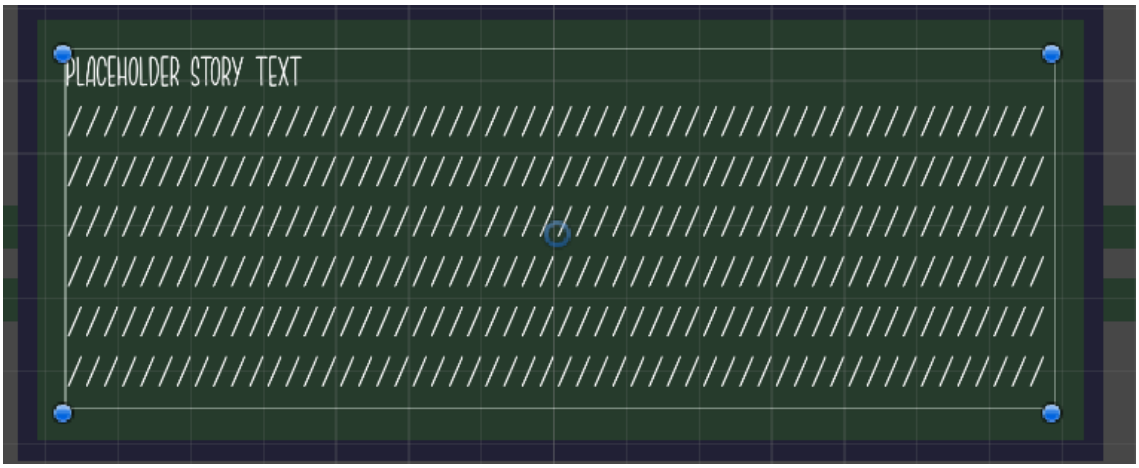
```
19  public State[] GetNextState()
20  {
21      return nextStates;
22  }
```

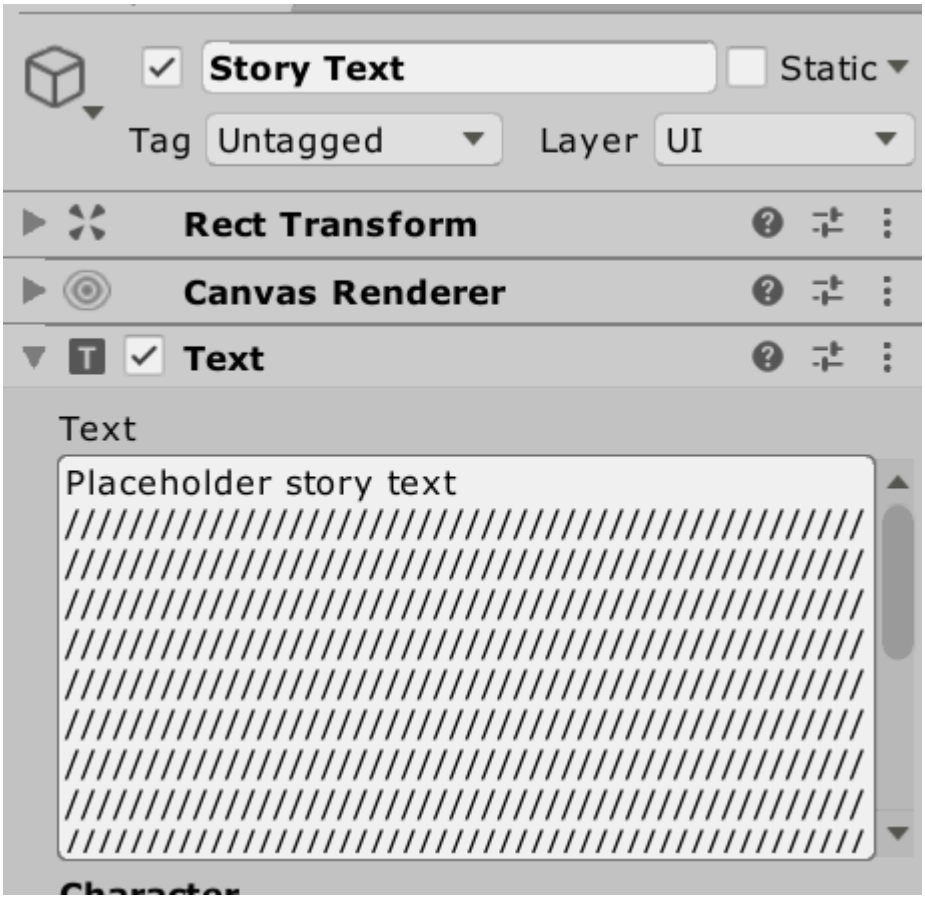
These are simply subroutines which can be called to change the scene (which links to the target scene float variable. They are called in the “Adventure game” script which is simply the script that manages all user interactions within the dialogue part of the game.

```
24      public void ChangeScene1()
25      {
26          SceneManager.LoadScene("Game");
27      }
        1 reference
28      public void ChangeScene2()
29      {
30          SceneManager.LoadScene("Game2");
31      }
32  }
```

Before I explain the adventure game script which is the next milestone, I will show how the UI is set up to display the dialogue as this is relevant to some of the errors I incurred in the next few milestones.

This is where the text is displayed, just as the design stage planned.





And this is the UI component which is edited by the game manager (adventure game script) to fit the story text written in each of the states.

Testing

This script does nothing when run on its own, the testing of all of the code will occur in the next milestone. The one test I can run is whether the asset can be created inside the Unity editor and whether the correct fields are shown (evidence in the screenshots above).

Test number	Test	Success criteria link	Expected outcome	Actual outcome
29	Asset creation	3	The asset can be selected from the asset creation menu and modified	The asset was created and usable
30	Asset fields	3	All of the fields are present and are correct and modifiable	The fields can be written to and modified and all are as intended

Variables

Name	Variable type	Reasoning for name	Reasoning for use
storyText	string	Holds the writing of the story	Text space for the story to be written inside
nextStates	state (object)	Holds the states that this state is linked to	To reference the states which the player chooses to go to
TargetScene (wrong formatting)	float	Holds the number relating to the scene change	To update an if statement in the dialogue manager which changes the scene

Conclusion

What went well

The states are designed around being reusable and are working as intended.

Even better if

I need to remember to format my variables correctly as after looking back it seems I have not.

Changes from original

This section is completely new.

Note: the success criteria review will be included in the last part to encompass the entire milestone

Client sign off

The meeting with the client went well. She was extremely happy with the way I made the creation of new states simple and easy. This meeting was shared with the dialogue manager part of the dialogue milestone and therefore she signed me off at the end of the whole milestone instead of each individual part.

References

URL	What it was used for

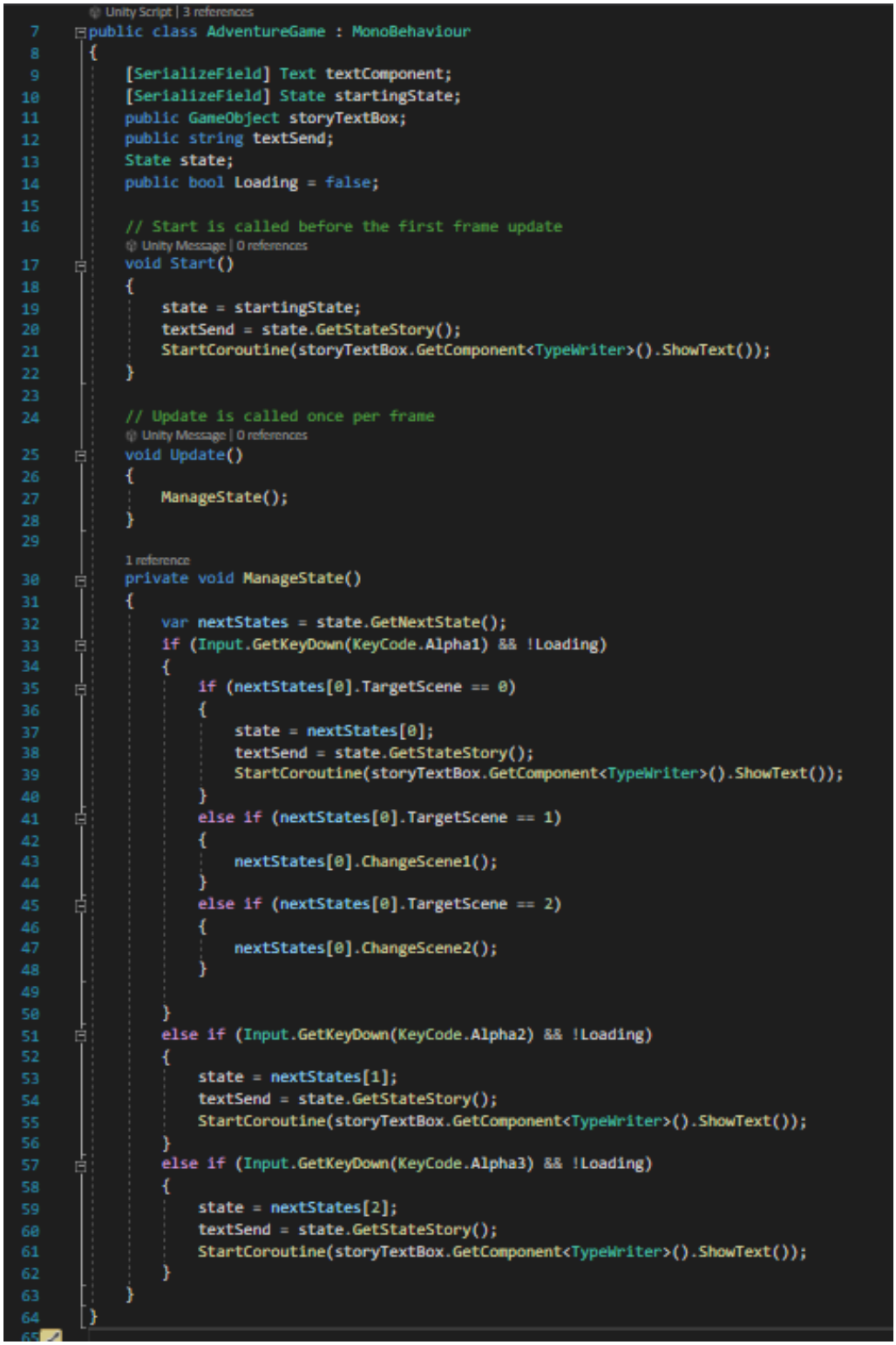
https://docs.unity3d.com/ScriptReference/CreateAssetMenuAttribute.html	Finding the command to create a scriptable object

Dialogue manager (Adventure Game)

Explanation

As mentioned before, the Adventure game script is simply a script that manages all the user interactions in that scene.

This is the full code in context:



```
Unity Script | 3 references
public class AdventureGame : MonoBehaviour
{
    [SerializeField] Text textComponent;
    [SerializeField] State startingState;
    public GameObject storyTextBox;
    public string textSend;
    State state;
    public bool Loading = false;

    // Start is called before the first frame update
    Unity Message | 0 references
    void Start()
    {
        state = startingState;
        textSend = state.GetStateStory();
        StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
    }

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        ManageState();
    }

    1 reference
    private void ManageState()
    {
        var nextStates = state.GetNextState();
        if (Input.GetKeyDown(KeyCode.Alpha1) && !Loading)
        {
            if (nextStates[0].TargetScene == 0)
            {
                state = nextStates[0];
                textSend = state.GetStateStory();
                StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
            }
            else if (nextStates[0].TargetScene == 1)
            {
                nextStates[0].ChangeScene1();
            }
            else if (nextStates[0].TargetScene == 2)
            {
                nextStates[0].ChangeScene2();
            }

        }
        else if (Input.GetKeyDown(KeyCode.Alpha2) && !Loading)
        {
            state = nextStates[1];
            textSend = state.GetStateStory();
            StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
        }
        else if (Input.GetKeyDown(KeyCode.Alpha3) && !Loading)
        {
            state = nextStates[2];
            textSend = state.GetStateStory();
            StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
        }
    }
}
```

This is one of the longer pieces of code which I wrote, therefore I will discuss a general overview of what it does and then explain why I designed it this way.

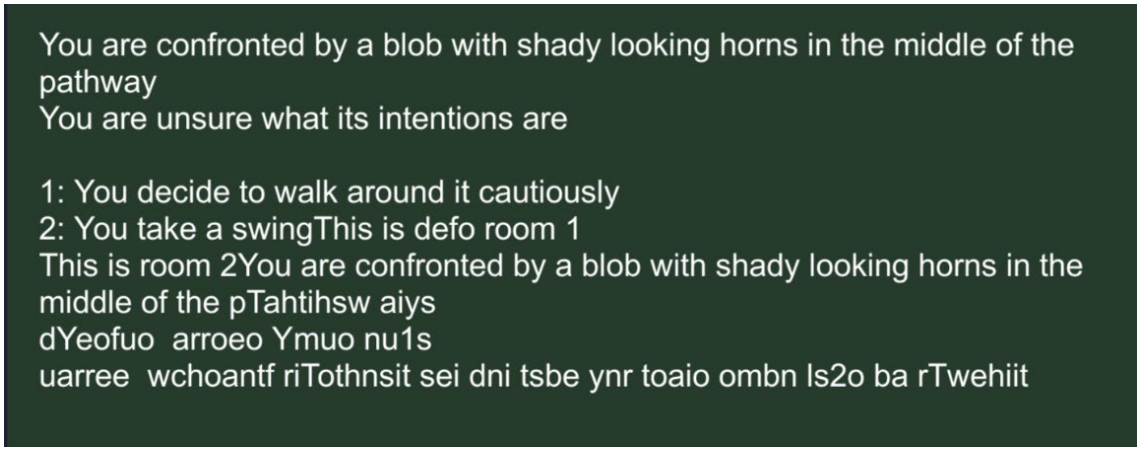
The variables are declared at the start all in a group as I usually do. As the script is first run, the variables are placed in their starting conditions so that if the script is re-run, no bugs regarding old data are encountered.

I thought ahead in regards to the starting state variable. I could have made the state that the scene started at simply declared by name, instead I made it an object variable in which the editor can choose which state to start from as a second scene would start from a different state.

```
17      void Start()
18      {
19          state = startingState;
20          textSend = state.GetStateStory();
21          StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
22      }
```

The main subroutine of the script is called in update so that the inputs are validated every frame (if there are any). Simply what it does is change the state depending on what button is pressed which moves the story onwards.

One of the tests that I performed on the code, I found that the player could spam through the dialogue scene:



To fix this I added the loading variable as a condition which needs to be met to move to the next state. It is boolean and turns to true in the textwriter and false once the coroutine is finished. Acting as a barrier for the player to have to read the text.

```
33          if (Input.GetKeyDown(KeyCode.Alpha1) && !Loading)
34          {
35              if (nextStates[0].TargetScene == 0)
36              {
```

After this implementation all of the code worked perfectly.

However I was not happy with the way it was written. It used many conditionals inside of each other. This is a very inefficient way of coding and uses up more memory than I would want. Luckily the game is small in size and it makes no

noticeable difference but if the game was to be developed in the future, this may need to be reworked.

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
31	Spam	3	The user could not progress through the story without reading	The user was able to spam through all of the scenes
32	Loading	3	The user could not progress through the story without reading because there is now a loading condition	The user could not move on until the choices and story was written out
33	Button 1	3.1	When the user enters 1, the corresponding state is loaded and written out	The user chooses option 1 and the linked state is loaded
34	Button 2	3.1	When the user enters 2, the corresponding state is loaded and written out	The user chooses option 2 and the linked state is loaded
35	Button 3	3.1	When the user enters 2, the corresponding state is loaded and written out	The user chooses option 3 and the linked state is loaded
35	Invalid press	3.1	When the user enters an option which is not specified in the choices, the game does not break and carries on until a valid input is pressed	Nothing happens unless a valid button is pressed
35	Scene change	3	When the state is loaded which has a corresponding case to load the next scene, the correct scene is loaded	The user is taken to the correct scene
36	No state	3	If there are no states linked to a button, when it is pressed the game carries on until a linked button is pressed	Nothing happens unless a link button is pressed

Variables

Name	Variable type	Reasoning for name	Reasoning for use
textComponent **not used**	object	Holds the text object which displays the text	N/A
startingState	object	Holds the state that the scene starts in	To start the scene from the correct state
storyTextBox	object	Holds the written text which goes in the text component	To change the text displayed on screen
textSend	string	Holds the written text as a string	Given to the text writer to write out
state	object	Holds the state asset	To reference states
Loading	boolean	Loading true or false condition	To make the user wait until the text is finished

Conclusion

What went well

The code works as intended and, in combination with the “states” milestone before, another success criteria point is complete.

Even better if

The code was inefficient in this milestone and if I was to do it again, I would find a way around using so many conditionals inside each other

Changes from original

The method of using a game manager to change between parts of the script is different from the design.

Client sign off

Grace (the client) unlike myself, was not worried about my concerns around efficiency as there was no immediate effect on the performance of the program.

There was still no sign off as this was only a chunk of the overall milestone.

References

URL	What it was used for

None	N/A

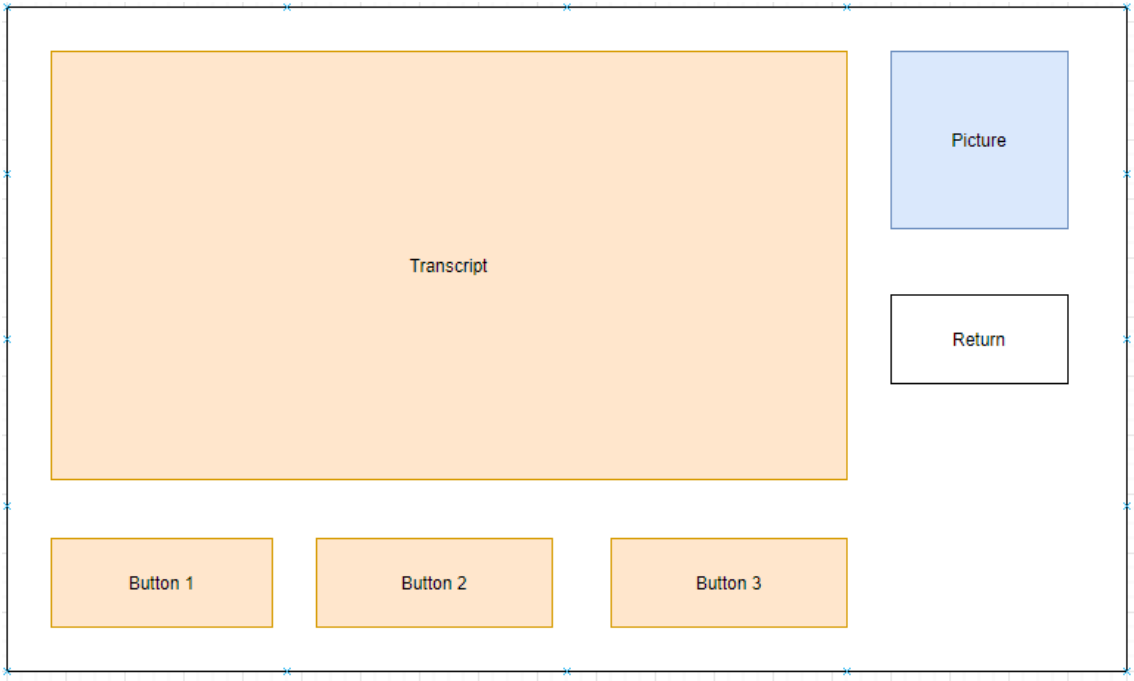
Dialogue UI design

Explanation

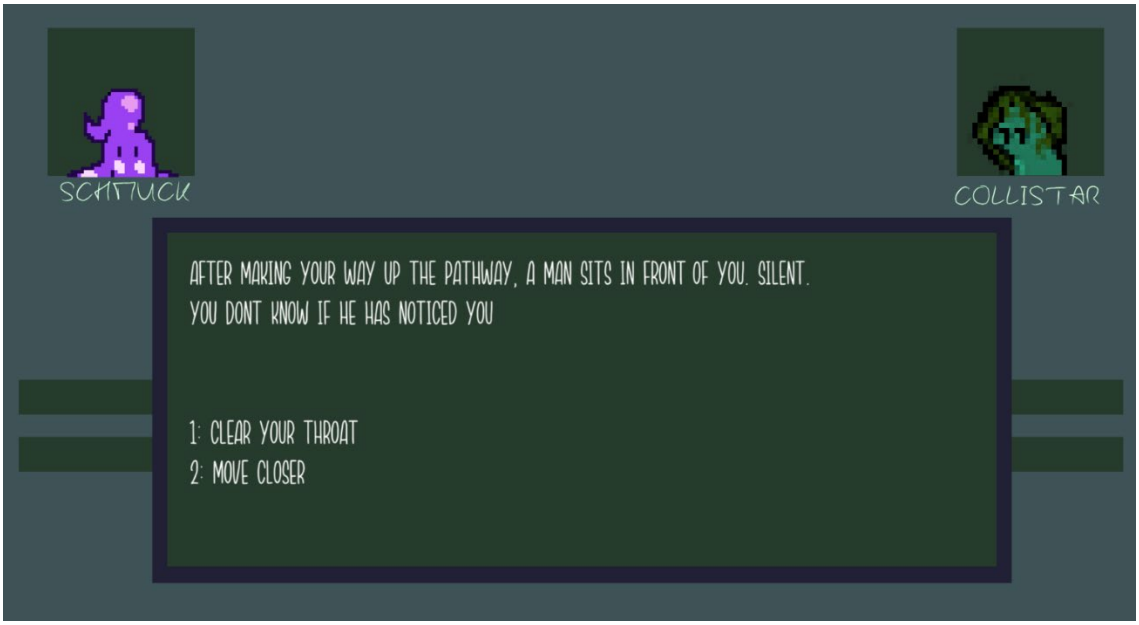
This turned out differently from the originally stated design.

However, the point in the success criteria are still reached: “It must have buttons for responses, an exit button and a place to display the text”

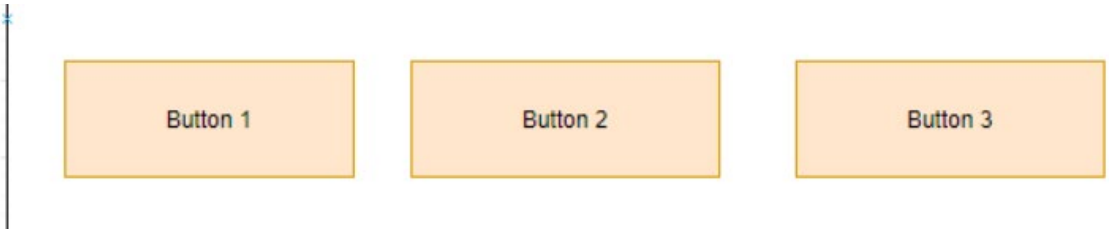
For reference:



Vs



One aspect was the three buttons; the development time needed to create buttons which interact with the states was not necessary when simple button press prompts within the text suit the purpose with no errors. This was passed by the client who gave it the go-ahead as it has nearly the same functionality with only a different design look:

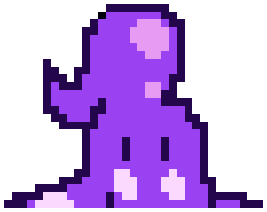


Vs

The return button has gone from the original, this is because it made little sense storywise to simply walk away from a conversation. Furthermore, the objective of player immersion will be broken by the time traveling - backtracking - aspect of the return function. Therefore after conversing with the client, this was scrapped.

However, a compromise of having a similar function in the pause menu was chosen as it is optional to press meaning that the player opts to break their immersion and it provides them with accessibility options if they need to leave the game quickly. Extra aim.

Lastly, I created head sprites for each character which is just the same as their sprite without the body and put them in a box.



This somewhat differs from the original design as both the player character and NPC are included as images. However, it makes the menu look more symmetrical.

Testing
No code testing is applicable for this part of the development

Variables
No variables were used in this part of development

Conclusion

What went well

The design of the UI has all the functionality features needed excluding the return button which was compromised for. The interface is sleek and simple which does not overwhelm the player and is easy to understand.

Even better if

If I was better at pixel art, next time I would design a custom background as the block colour seems mundane.

Changes from original

The UI has keyboard prompts instead of interactable buttons, no return button and two character graphics instead of one.

Addition to pause menu: similar return function.

Client sign off

In the meeting with my client, I confirmed the decision to remove the return button and have it placed in the pause menu. I then showed her the user interface which she responded well to. I asked her to tell me how she thinks she would progress to see if it was simple enough to pick up easily. She found it easy to use.

With that she signed me off to the next part of the dialogue milestone.

References

URL	What it was used for
https://www.youtube.com/watch?v=VHFJgQraVUs	To understand the basics of creating a UI in unity (similar to the main menu)

TextWriter

Complex

Explanation

Text writer is a completely aesthetic addition to the project, during the design phase some stakeholders such as Max Brooker noted that “instead of a hard information dump it would be much easier to read if it was written alongside the reader”. Therefore, I discussed with the other stakeholders and client and they decided that a text writer would be a good direction to take for game cohesion.

Therefore this is not meeting any success criteria that was stated in the design.

This is the TextWriter script:

```
Unity Script | 4 references
public class TypeWriter : MonoBehaviour
{
    public float delay = 0.001f;
    public string fullText;
    private string currentText;
    private List<char> charArray = new List<char>();
    public GameObject gm;
    Unity Message | 0 references
    void Start()
    {

    }
    4 references
    public IEnumerator ShowText()
    {
        gm.GetComponent<AdventureGame>().Loading = true;
        fullText = gm.GetComponent<AdventureGame>().textSend;
        foreach (char c in fullText.ToCharArray())
        {
            charArray.Add(c);
        }
        currentText = "";
        for (int i = 0; i < charArray.Count; i++)
        {
            currentText = currentText + charArray[i];
            GetComponent<Text>().text = currentText;
            yield return new WaitForSeconds(delay);
        }
        charArray.Clear();
        gm.GetComponent<AdventureGame>().Loading = false;
    }
}
```

The first few lines in the type-writer subroutine define all the variables used.

```
public class TypeWriter : MonoBehaviour
{
    public float delay = 0.001f;
    public string fullText;
    private string currentText;
    private List<char> charArray = new List<char>();
    public GameObject gm;
```

The start routine has no use, but is kept in if future additions or tweaks need to be made to the script. This is noted in the document for future maintenance.

```
Unity Message | 0 references
13  void Start()
14  {
15
16  }
```

This is the start of the coroutine which writes out the text:

```
4 references
17  public IEnumerator ShowText()
18  {
```

This is called on within the game manager shown here:

```
39      StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
```

In the first part of the coroutine a boolean variable called “loading” from the game manager is set to true so that the player cannot choose an option without having the text written out. This was done to remedy the error found in the testing of the dialogue manager.

```
18  {
19      gm.GetComponent<AdventureGame>().Loading = true;
20      fullText = gm.GetComponent<AdventureGame>().textSend;
21      foreach (char c in fullText.ToCharArray())
22      {
23          charArray.Add(c);
24      }
```

The text send variable from the dialogue manager script which holds the story text is put into a variable in this script named “fullText” just so that the coding is easier. This is because in unity, referencing another script requires a long line of code, it is much simpler for me to put it into its own variable. I did not do this for the loading variable as it was not referenced much.

The char array is given all of the characters of full text for the next part of the code to write out.

```
25          currentText = "";
26          for (int i = 0; i < charArray.Count; i++)
27          {
28              currentText = currentText + charArray[i];
29              GetComponent<Text>().text = currentText;
30              yield return new WaitForSeconds(delay);
```

I clear the current text variable so that if the code is being run a second time, there is no old data which could cause bugs.

The for loop writes out the code character by character and waits for the delay before looping back until all of the text is written.

Then the array is cleared for the next time it is run and the loading is reset so that the decisions can be made by the player in the dialogue manager.

```
31          }
32          charArray.Clear();
33          gm.GetComponent<AdventureGame>().Loading = false;
34      }
```

There was an issue that the text was being written out twice. This was fixed before it could be documented, however, the root of the problem was that the coroutine was being called twice. Therefore, I merged the two lines of code and that resolved the issue.

```
39              StartCoroutine(storyTextBox.GetComponent<TypeWriter>().ShowText());
```

This was that line of code.

Testing

There were two testing cycles during this stage due to the bugs I encountered with the dual running coroutines.

Test number	Test	Success criteria link	Expected outcome	Actual outcome
37	Write out	N/A	The text is written out letter by letter at the speed set in the editor	The text is written out character by character

38	Expected text	N/A	The text being written out is the exact same as that stored in the state	Multiple characters were being written at once

This test table is after the fix was implemented:

Test number	Test	Success criteria link	Expected outcome	Actual outcome
39	Expected text 2	N/A	The text being written out is the exact same as that stored in the state	The text was correct in compliance to the state.

Variables

Name	Variable type	Reasoning for name	Reasoning for use
delay	float	The amount of delay between each character being written	Is the waiting time in point seconds before the code is looped
fullText	string	Holds the imported text from the dialogue manager	To give each character of the script to the character array
currentText	string	Holds the text currently displayed	To print out the text each time a character is added giving the illusion of it being written out
charArray	character	Holds each individual character in an array	To add on each character to the currentText
gm	object	Short for Game manager	To reference the dialogue manager script

Conclusion

What went well

The end result is a great solution of giving the illusion of each word being written one by one and the code is well organised and as efficient as I could make it.

Even better if

If I was to do this milestone again, I would learn from my coding errors and be more careful when planning out the milestone so that I include more validation conditions and make less simple mistakes like I did running the coroutine twice.

Changes from original

This was not part of the original design but was requested by stakeholders.

Client sign off

After completing this part, I hosted a larger meeting with both my client and two of the stakeholders. The reasoning for a larger meeting than normal is because this part of the dialogue was a request by the stakeholders. Therefore, I needed to check that this meets the expectations of what they were imagining AND to run it past the client to finalize the addition.

The stakeholders and client were happy. I could now work on the writing of the story.

References

URL	What it was used for
https://www.tutorialspoint.com/declare-char-arrays-in-chash	How to make a regular array into a character array

Story

Explanation

In my design, point 3.2 states “There should be a simple transcript design layout which can be easily implemented into the game.”

I designed the transcript in Draw.Io to show the flow of the story as well as the text. Each box represents a “room”/”state” in the program.

For this milestone, I involved and worked closely with my stakeholders Jonathan and Emily.

As stated in my analysis, “I chose Emily to be one of my stakeholders because she is a regular consumer of books and stories. She will be able to inform my decisions and thoughts on story writing.” and “I chose Jonathan to be one of my stakeholders as he is

a writer. This insight will be useful to teach and guide me through my development and story writing.”

In my proposed solution it states “Any and all text will be there to build the plot of the story and world atmosphere the game is portraying”. My stakeholders will help me not to stray from this point and solve the problem my client has rather than create it.

The theme of this game's story is about repetition and finding meaning in a world that you cannot control .

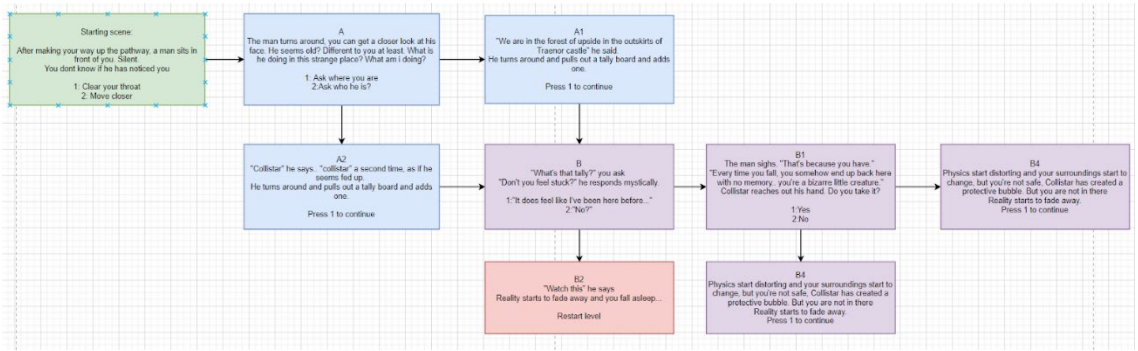
The stakeholders Emily, Jonathan and I came up with this idea to revolve the story around as it gives the mechanics of the game an explanation of why they are re-doing the level if they fail and is relatable to the audience who, like many people may feel like they are doing the same thing over and over again on a day to day basis like a job but still search for meaning in happiness, peace or joy.

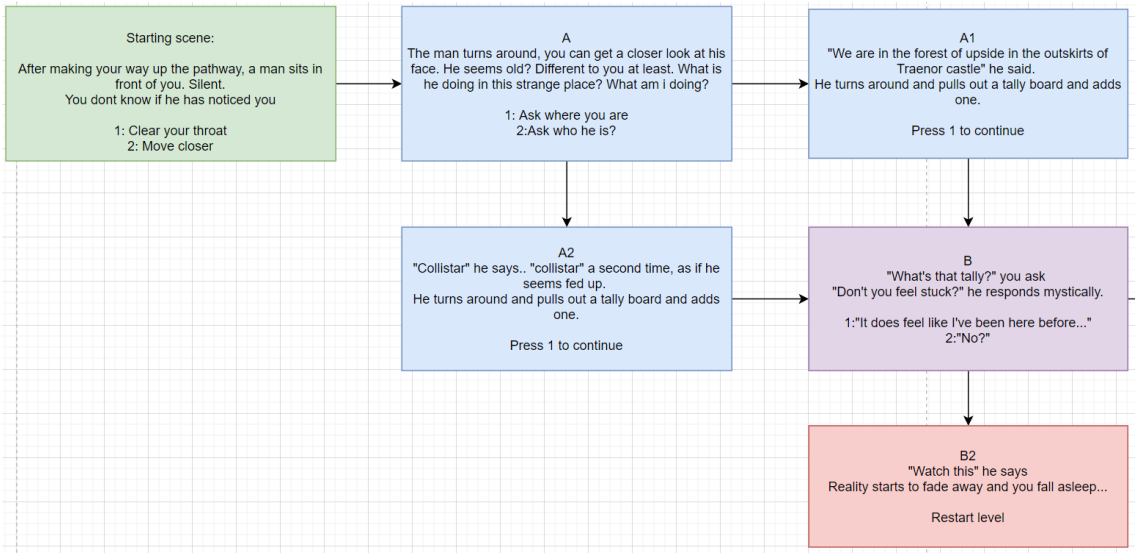
The emotion that relatability invokes is important to making the game memorable to the player. People do not remember conversations word by word but only the emotions that they associate with it and they felt at the time. The same goes for any story or film and now for a game.

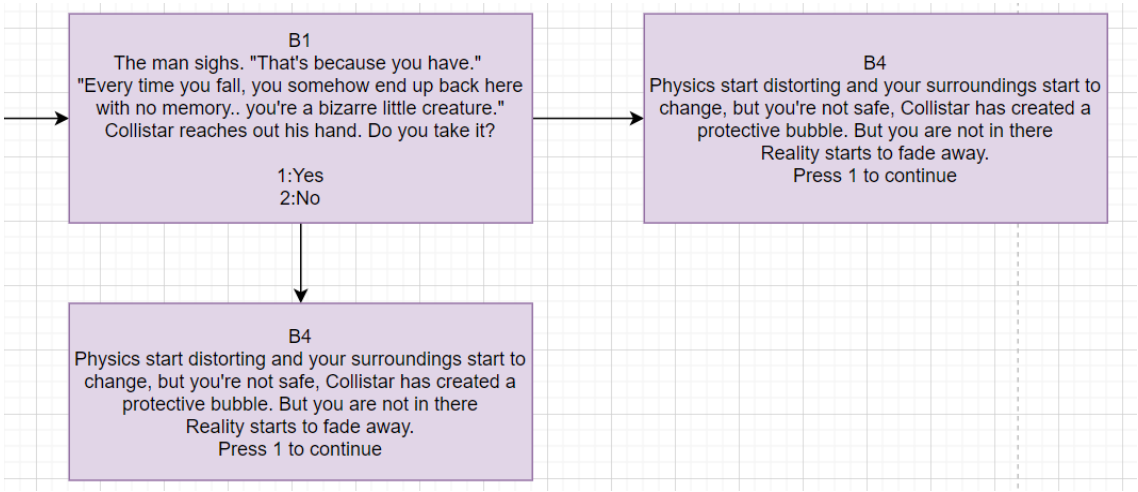
Dialogue(1)

The focus with dialogue 1 was to build suspense in the game and have an air of mystery surrounding the plot as a complication arises.

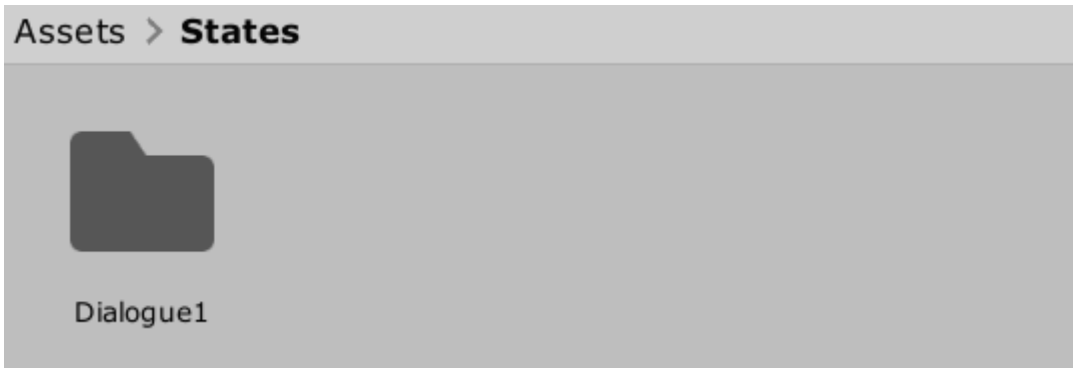
The end product was this flow diagram of all the rooms for the first dialogue segment.

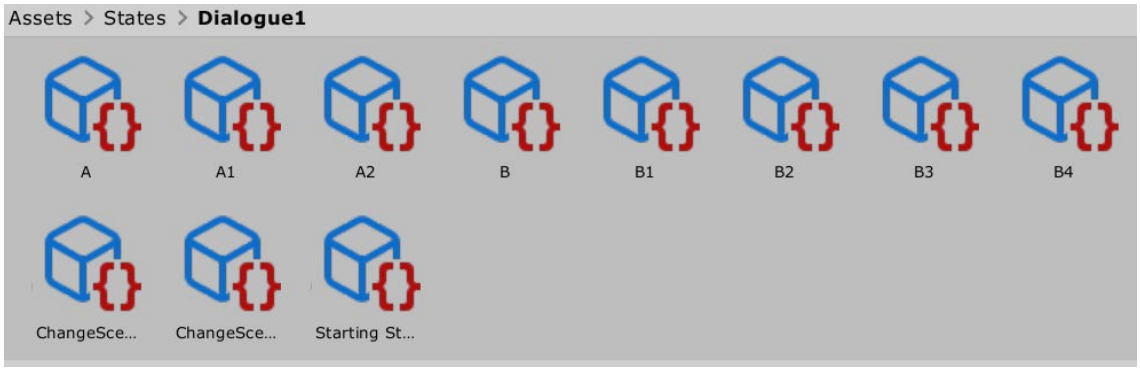






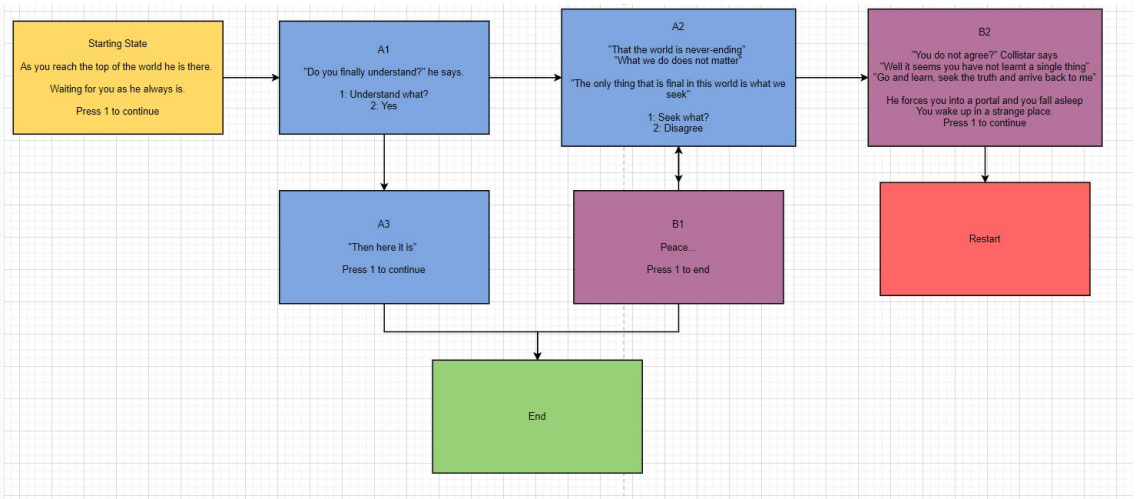
For organisation, I created a folder in the editor and placed all of the states in the relevant dialogue scene. I also named the states after the points on the draw.io chart so that it stayed consistent from the planning.



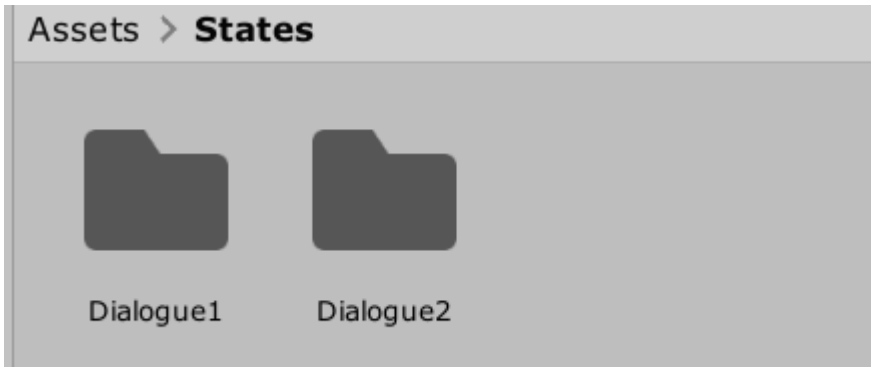


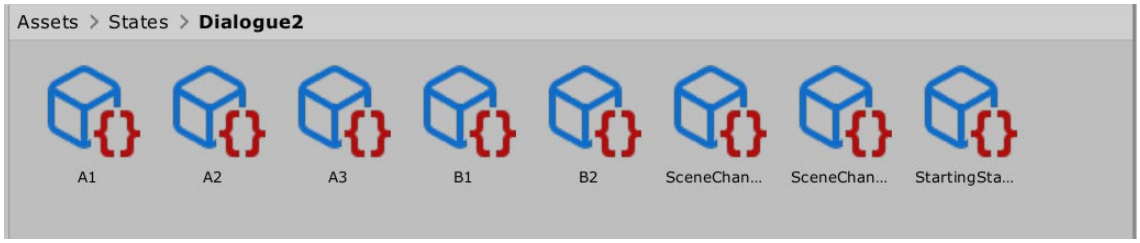
Dialogue(2)

Dialogue 2 uses the gameplay as an experience and the branching paths allows it to be the climax as well as the denouement. This dialogue tree is special because one option does not lead to the end of the game. Instead it leads the player to restart the game. Which adds to the theme of the story.



This was organised with the same method as dialogue 1:





Testing
Not applicable

Variables
Not applicable

Conclusion

What went well
The design for the script relating to each state is easy to read, follow and implement into the game. The story overall has a very interesting concept and is written well.

Even better if
If I had more time dedicated to writing the story we could expand on the concept we created a lot more which would improve the end product exponentially.

Changes from original
None

References

URL	What it was used for
https://www.youtube.com/playlist?list=PL3-vkaonFpj9hr1LXge2D0CddnaB8AHUo	Learning how plots are made

Dialogue end review
14th September

What went well

The entire milestone is working functionally as intended and the code has no bugs. The story is written well and solves the problem which the client has with normal games nowadays having bad writing. The elements of the dialogue blend well with each other, the story is complimented by a book-like presentation being printed letter by letter and the choices make the player involved so that they become invested in the characters journey.

Even better if

If there was music that went with the writing there would be more ambience and help the game to invoke greater emotion.

Changes from original

The text is written out word by word and there are some changes to the UI design (referenced in the dialogue ui design section).

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
3	The game must have fully functioning dialogue menu with branching choice paths	Yes	The states are linked, progressable and lead to the next scene (win) and last scene (loose)
3.1	It must have buttons for responses, an exit button and a place to display the text	Somewhat	There are no buttons that can be clicked but keyboard prompts which relate to the responses available, a box where the text is written. The exit button will be in the pause menu.
3.2	There should be a simple transcript design layout which can be easily implemented into the game	Yes	Flow diagrams for the script have been created and put into the states inside of the game
3.3	It could show who the player is talking to for coherency	Yes	There are headshots of the player character and boss in the corners of the UI

Client sign off

This part of the development was the part the client was most anxious about, this is where her problem with the gaming industry is solved. Grace was involved at every stage of the dialogue milestone and her end conclusion was that it fits perfectly with what she envisioned. She said that the story that I and the stakeholders created is

perfect for people who enjoy reading and that the gameplay created alongside it is great for introducing them to videogames as a form of storytelling.

Grace signed me off to finish off parts of the game as we near the end of the development.

Menu's

15th September

For the player to be able to respawn and pause the game, an interactable UI needed to be implemented. This was not stated in the success criteria but was identified in design and I was asked to by the client to develop it.

Pause Menu

Explanation

Firstly the pause menu has to be accessible at all times during the game, except from the main menu. Therefore I plan on creating an overlay which stops the game and gives the player all of the relevant choices: Resume, Quit and Main menu. I decided not to add the options as at this point I have not yet created any.

I designed three buttons over a semi-transparent black panel to still have the game visible but it is clear that the game has been paused.

To give functionality to the menu, I created this script:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

Unity Script | 0 references
public class PauseMenu : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;

    // Update is called once per frame
    Unity Message | 0 references
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }
            else
            {
                PauseGame();
            }
        }
```

This code is self explanatory, the only point of note is where it checks against the GameIsPaused variable to see if the player is already in the menu and if it should pause or resume. The relevant subroutines are shown below:

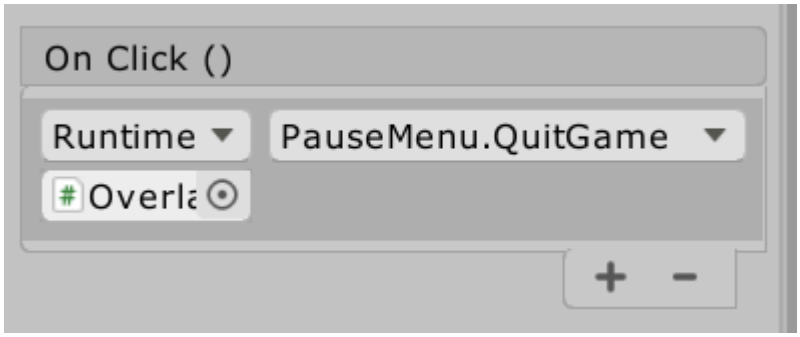
```
25          }
26
            1 reference
27          public void Resume()
28          {
29              pauseMenuUI.SetActive(false);
30              Time.timeScale = 1f;
31              GameIsPaused = false;
32          }
            1 reference
33          void PauseGame()
34          {
35              pauseMenuUI.SetActive(true);
36              Time.timeScale = 0f;
37              GameIsPaused = true;
38          }
            0 references
39          public void LoadMenu()
40          {
41              Time.timeScale = 1f;
42              SceneManager.LoadScene("MainMenu");
43              Debug.Log("Loading");
44          }
            0 references
45          public void QuitGame()
46          {
47              Debug.Log("Quitting game");
48              Application.Quit();
49          }
50      }
51
52
```

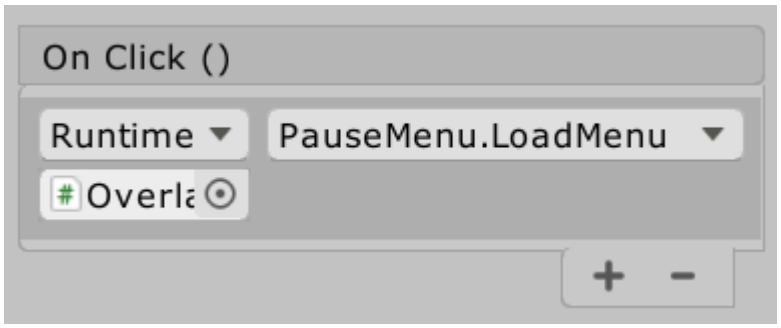
The pause and resume subroutines stop the game by setting the time progression to 0 or start it by setting it to 1 (normal) respectively. They also show and hide the menu UI and state if the game is paused or not.

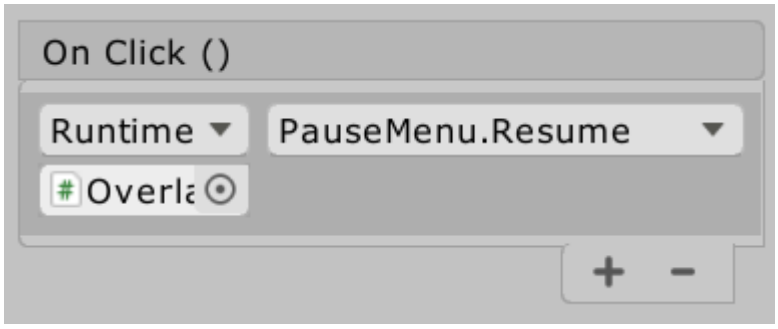
The other two subroutines will be programmed into the buttons. The quit button will stop the application from running and the main menu will take the player back to the

main menu scene. The finished code also includes a line which resumes time, this will be explained in the testing.

This is the programming of the buttons:







Simply, as the buttons are clicked, the subroutines from the script are run.

Testing

The code above was not the first iteration, one line was added to the main menu subroutine:

“Time.timeScale = 1f;”

The reason why is shown in the first test table:

Test number	Test	Success criteria link	Expected outcome	Actual outcome
40	UI load	N/A	When the escape key is pressed the UI is shown	The pause menu appears when the player presses the escape key
41	UI close	N/A	When the escape key is pressed and the UI is active, the UI disappears	The pause menu disappears when the menu is active and the player presses the escape key
42	Time Stop on UI load	N/A	When the UI is on the screen, the time flow is set to 0	The game stops playing
43	Time resume on resume press	N/A	When the UI is on the screen, the time flow is set to 1	The game restarts playing
44	Game Quit	N/A	When the quit button is pressed, the application closes and in the console, the debug log is printed to show that it has run	The game closes
45	Main menu	N/A	When the main menu button is pressed, the scene is switched to the main menu	The scene changes to the main menu
46	Play game from main menu	N/A	After entering the game all of the buttons work and the next scene can be entered and works	The buttons can be pressed and the next scene can be entered but not played

The time did not restart when the player entered the main menu, therefore when they entered the first scene the game was still paused with no way of resuming.

After fix:

Test number	Test	Success criteria link	Expected outcome	Actual outcome

47	Resume time main menu	N/A	When the 1st level starts the time is flowing at a regular speed	The game is playable

Variables

Name	Variable type	Reasoning for name	Reasoning for use
GameIsPaused	boolean	It checks if the game is paused	To pause and unpause the menu depending on its state
pauseMenuUI	object	Holds the UI element of the pause menu	To turn the menu on and off

Respawn menu

In the respawn milestone, the client identified that a menu should appear and pause the game until the player presses respawn for some downtime between each try at the platforming segment.

Explanation

I wanted a similar design to the pause menu, where there is a button for respawn and an overlay which darkens the screen.

This is the design I came up with:

And this is the code which is attached:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity Script | 0 references
public class RespawnMenu : MonoBehaviour
{
    public static bool GameIsPaused = false;
    private bool respawning;
    public GameObject player;
    public GameObject respawnMenuUI;
    Unity Message | 0 references
    void FixedUpdate()
    {
        respawning = player.GetComponent<RespawnPoint>().respawning;
        if (respawning)
        {
            PauseGame();
        }
    }
    1 reference
    public void Resume()
    {
        respawnMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
    }
    1 reference
    void PauseGame()
    {
        respawnMenuUI.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
    }
    0 references
    public void Respawn()
    {
        respawning = false;
        player.GetComponent<RespawnPoint>().respawning = false;
        player.GetComponent<SpriteRenderer>().enabled = true;
        Resume();
        Debug.Log("working");
    }
}
```

The pause and resume subroutines work the same as the previous script only under a different condition.

There is a boolean variable called respawning that I placed in the respawn point script (shown later) which is used to tell the script to pause and unpause.

The actual respawning subroutine is attached to the respawn button. When clicked it changes the respawning variable and unpauses the game.

This then allows the respawning to continue as has already been documented in the respawn milestone.

```
37            respawning = true; // heads to the respawn menu
```

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
48	Respawn pause	N/A	When the player dies the game stops	The game is not playable
49	Respawn button	N/A	When the player presses the respawn button, the game is unpaused and the player respawns	The player respawns and the game carries on - the player falls through the floor

The player is falling through the floor which is not good.

After some debugging, I found that the root of the problem is that the player is keeping its velocity from the fall and glitching through the floor due to the speed it is travelling at. This is a simple fix and I placed this line of code before the player is placed at the respawn point. (this is in the “respawn point” script)

```
38            rb.velocity = new Vector2(0, 0); // the character plummeted through the floor 24/7
```

Test number	Test	Success criteria link	Expected outcome	Actual outcome
50	Clear velocity	N/A	The players velocity is set to none so that they do not glitch	The player does not fall through the floor

Variables

Name	Variable type	Reasoning for name	Reasoning for use
GameIsPaused	boolean	It checks if the game is paused	To pause and unpause the menu depending on its state
respawning	boolean	It checks if the player is in the middle of respawning	To display the menu before the player is respawned
player	object	It holds the player character	To reference and change variables attached to the player character
respawnMenuUI	object	It holds the respawn Ui	To turn the menu on and off

Conclusion

What went well

The menu looks sleek and does not remove the player from the game but gives them a break which lasts as long as they wish.

Even better if

If I had done this again, I would have liked to use more efficient code which uses less subroutines.

Changes from original

This was not in the original design; it was requested by the client.

Success criteria review

Not applicable

Client sign off

When the client saw my finished product she had some concerns:

When the game paused and waited for the player to press respawn, it did not look like the player had died as he was just floating in midair.

Grace told me to find a fix to this or restart the milestone as it completely broke immersion.

Fix

My thoughts on how to fix the issue were to make the player disappear when it hit the world end and reappear once the player clicks respawn.

To do this, I referenced the sprite renderer in hopes to mimic what I once tried with destroying the player sprite but this time instead I just deactivate its sprite and reactivate it as it is respawned rather than destroying the whole character. This is the code that does that:

```
25          SpriteRenderer sr = GetComponent<SpriteRenderer>(); // the player did not look like it died
26          sr.enabled = false;
```

Deactivation (This is in the respawn point script)^^

```
35          player.GetComponent<SpriteRenderer>().enabled = true;
```

Reactivation (This is in the respawn subroutine so happens when the player presses respawn)^^

Testing

Test number	Test	Success criteria link	Expected outcome	Actual outcome
51	Sprite deactivation	N/A	The player sprite is deactivated	The player becomes invisible
52	Sprite re-activation	N/A	The player sprite is reactivated	The player becomes visible
53	Checks	N/A	The player is still controllable	The player moves as normal

Variables

Name	Variable type	Reasoning for name	Reasoning for use
sr	object	Short for sprite renderer	To pause and unpause the menu depending on its state

Client sign off

18th September

The client was much happier with this and said that the rest of the code was perfect as well.

She signed me off to the next milestone.

Sound

19th September

The sound for the game was a refinement objective hence, this milestone is one of the last. As researched in my analysis, the sound for a game sets the atmosphere and amplifies storytelling. This is mainly done with the music. Sound effects are mostly there for game coherency.

As the deadline for the game is coming up fast I am going to only add music into the game as the sound effects will take much more work. Here is my process:

Explanation

Music

The choice of music was easy. Whilst researching the hollow knight soundtrack, I found an album by Julian bream. This is a cover of Bach: Prelude in D Minor/Suite in E Minor. It was copyright free music and therefore can be used in my game.

I decided that there would be a song played on loop for each scene: main menu, level one etc to set the mood:

Main menu

This song will play on loop over the main menu. This will be the first thing that the player hears. I chose this piece in particular because of its short melodic tune. It is mysterious but also welcoming. I feel this creates a good atmosphere to introduce the game. The song also loops well into itself which is good as the player may spend some time in the main menu before they play.

Level one

This song plays whilst the player traverses the first platforming level. It has many ostinato which build up and up. This creates a rushed feel and coincides with the players climb to the top of the level making the player feel pressure to reach the top.

Dialogue one

Similarly to the last piece, fantasie has many repeating ostinato which build up, but towards the latter half of the song, it starts working its way down. This is perfect when paired with the story; after the climb up and mystery of what the player is doing here, as things start to unravel so does the music and the player's tension is released. It is at this point where the player finds enjoyment from the game. Just like a small rising action and climax all in one.

Level two

The next piece is a little different, it still has the theme of climbs and repeating ostinato included however, it also has many false cadences which are sets of chords which signify the end of a phrase. These create small moments of tension and relief as the player climbs further and further, the song playing in time with their falls and successes.

Dialogue two

I needed something calmer for the next scene to signify that the player was at the top of the world. None of the music suited in Julian Bream's album, so I searched elsewhere.

I searched for lute music to fit with the rest of the music and found this:

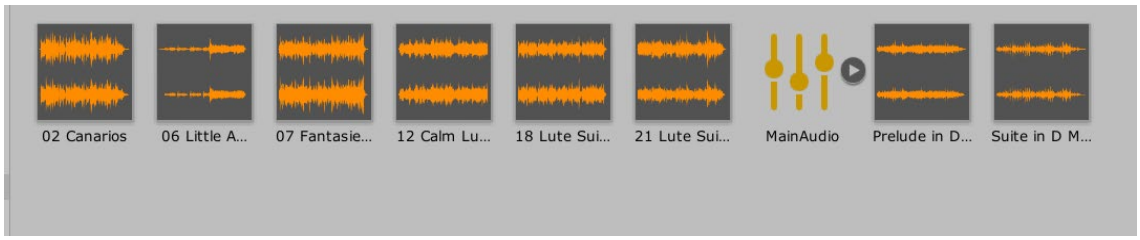
This piece has a calming ambiance and fits the theme I was looking for perfectly.

End screen

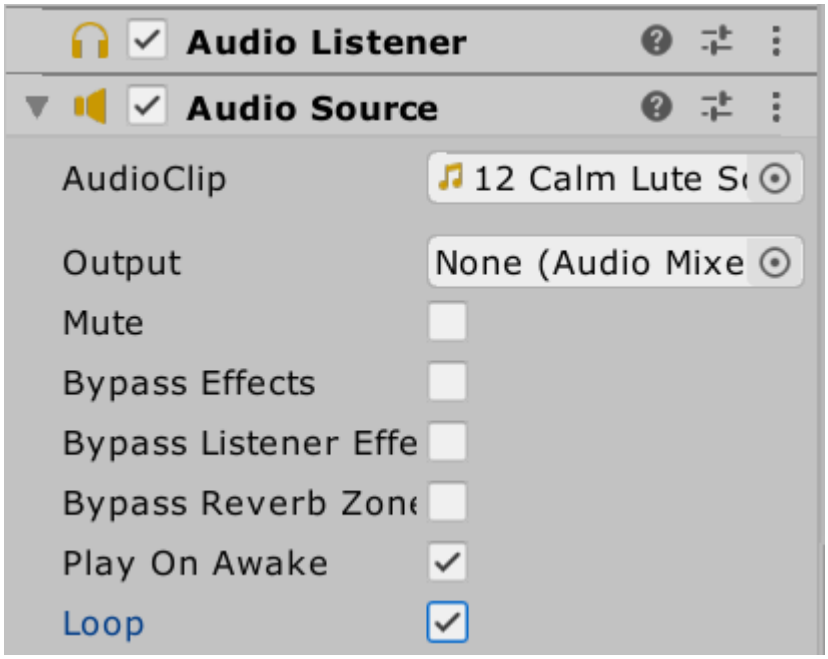
The end screen also needs to fit with the calming nature of the end of the game and I found this piece by murray gold

It is similar to the last song but has vocal chords which leave the game in a good feel and ends it perfectly.

I downloaded all of these songs and imported them into unity as MP3 files:



The process to make them play in the game was simple, I add each song to the main camera of each scene respectively as it controls the audio that the player hears and set it to loop:

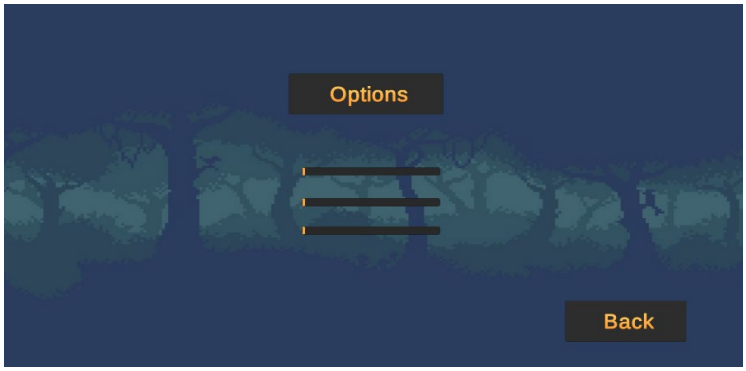


I decided to implement it like this instead of a script with coroutines because of the limited time I had available. The alternate method would have more functionality such as being able to play different songs in the same scene using event triggers, however I deemed it unnecessary.

Volume control

Next, I need to allow the player to adjust the volume of the music if it is too loud or just to suit their preferences.

I am going to place this in the options menu as a slider as I have already made placeholders:



I need to create a script which I can place on the sliders. This is what I came up with:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Audio;

Unity Script | 0 references
public class SetVolume : MonoBehaviour
{


    public AudioMixer mixer;


    0 references
    public void SetLevel (float sliderValue)
    {
        mixer.SetFloat("MainVolume", Mathf.Log10(sliderValue) * 20);
    }
}
```

I decided to have the volume slider control the main volume instead of the music individually. This is because no other sounds have been created and won't be as I am nearing the end of development, therefore this is much quicker and more concise to perform. The concisity of the code also helps the limitation of hardware identified in my analysis as it takes up less memory.

This code is simple; it takes the audio mixer and then when the slider is changed, the SetLevel subroutine is run to set the mixer level to the same value of the slider.

There is a little bit of complexity in the logarithmic maths I had to apply. To explain simply: decibels are on a logarithmic scale where every 10 decibel increase is about twice as loud as the last. Therefore without using Log10, it would increase exponentially and break people's eardrums.

Testing

Test Number	Test	Success criteria link	Expected outcome	Actual outcome
54	Music play + loop	6	The music plays on loop in each scene	The right music plays constantly in each scene
55	Volume control	6	The volume level correlates to the percentage of the slider	The volume level can be adjusted perfectly by the slider

Conclusion

What went well

The soundtrack compliments the story well and creates and ambience and atmosphere which is intriguing and stand out from other games

Even better if

If time allowed it, I would have liked to create my own soundtrack which gives me complete freedom over the type of music I would like in the game.

Changes from original

No sound effects

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
6	There should be a soundtrack with available volume adjustments	Yes	There is a musical soundtrack and a volume slider which controls the volume levels

Client sign off

21st September

This was the last meeting with the client as the last thing to document was the level design which had been made all through the development and hence does not need a sign off.

The client was very happy with the sound and after the last milestone an end review will occur and then the evaluation.

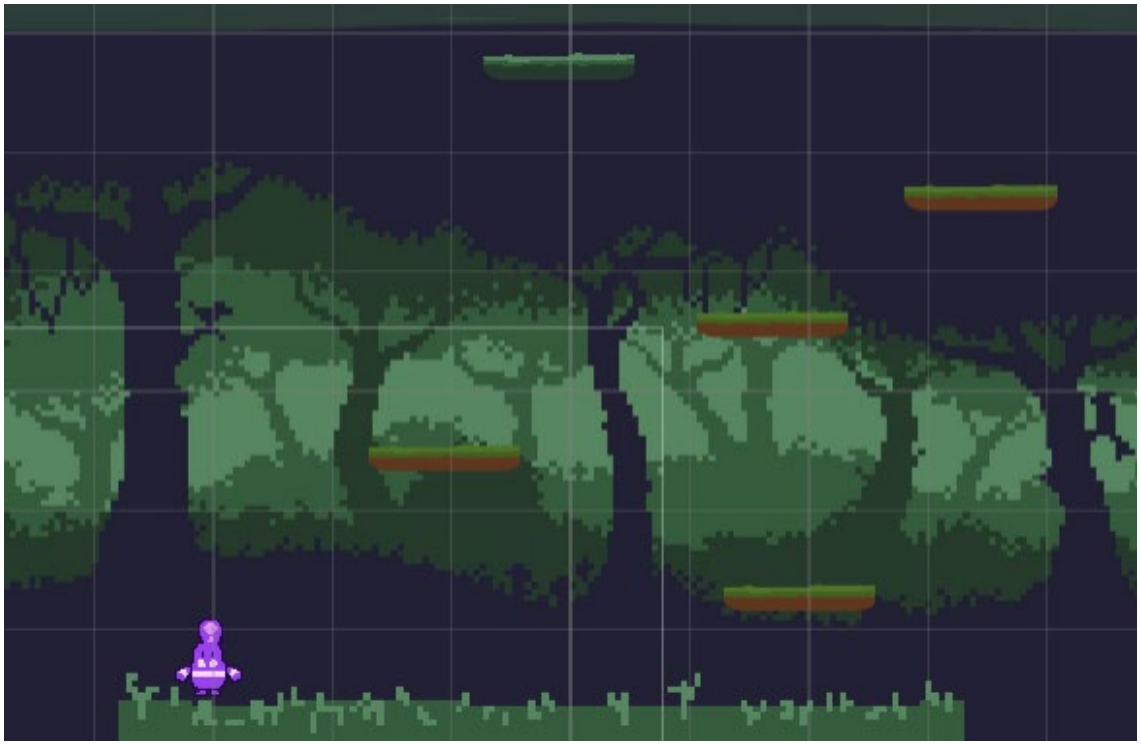
Level design

This was done throughout the development and the finished levels will be explained here. This milestone is to meet the requirement point 4 of the success criteria “There must be platforming elements which challenge the player” as well as just generally make the game feel clean to traverse and view.

Game 1

This is the first platforming segment and it is laid out as such:

The first few jumps are simple with the last of these being a bit harder to challenge the player before the rest of the platforms:



This level starts in the same background as the main menu as this will be what the player first sees.

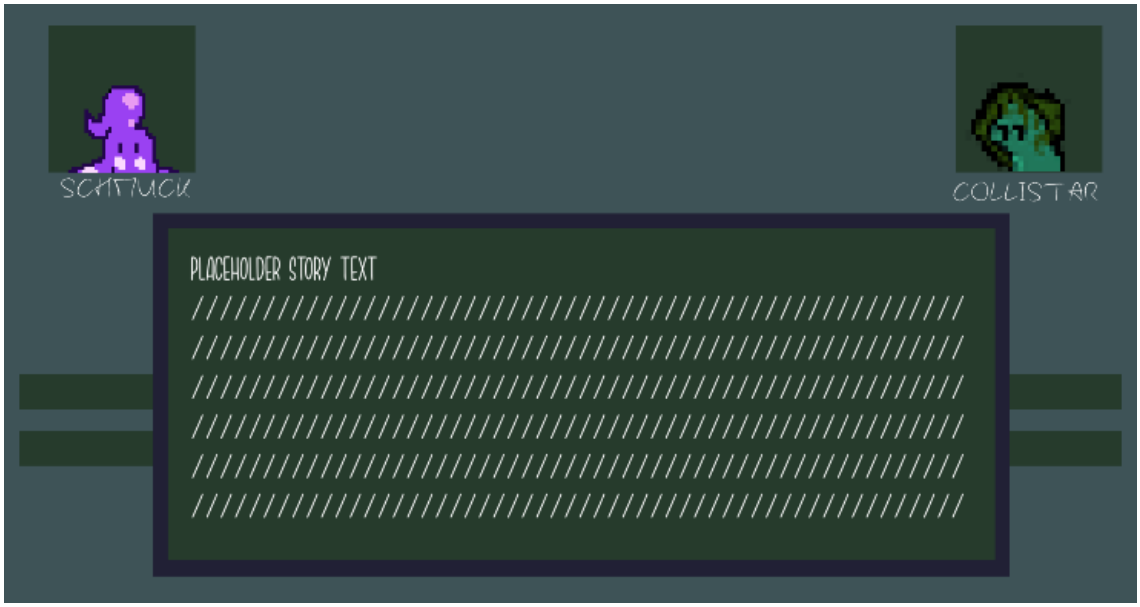
As the player makes his way out of the woods, the next background is upside down which hints at the difficulty to come:

There are a few level jumps which are made hard by the bounce mechanic requiring the player to take their time else they will fail.

The platforms then lead to the encounter with the boss. He is sat next to the campfire:

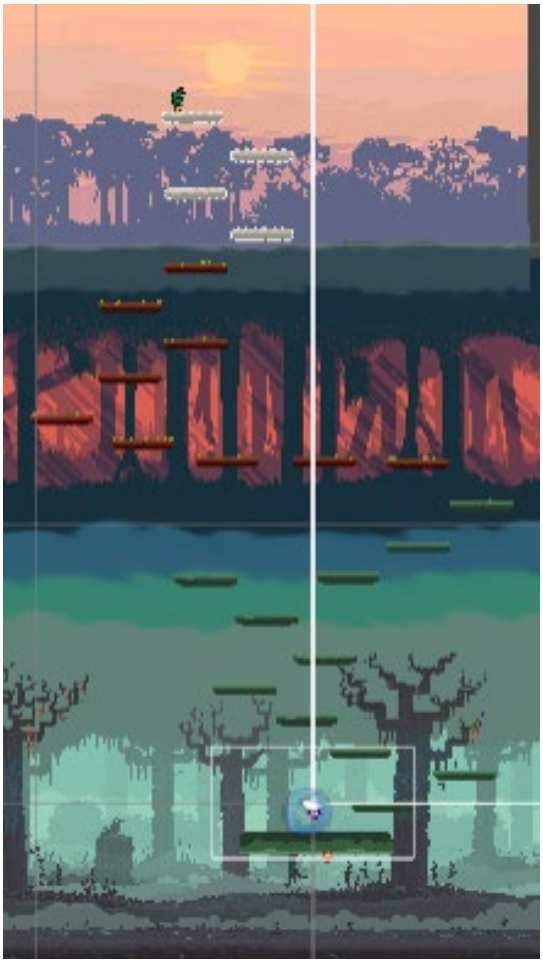
He will lead the player to the first dialogue:

Dialogue 1



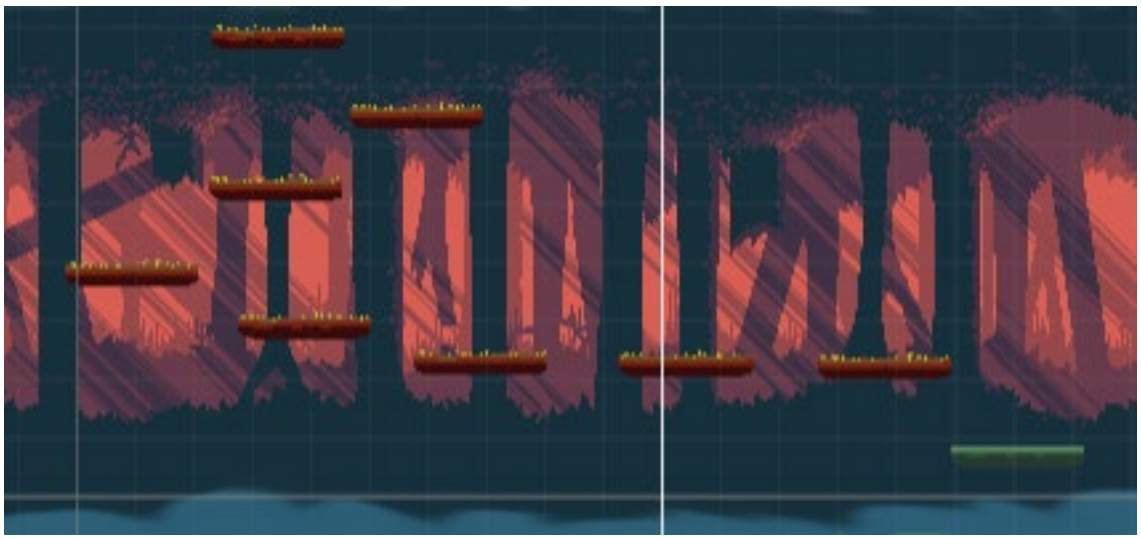
Completing this, the player will spawn in the next level.

Game 2



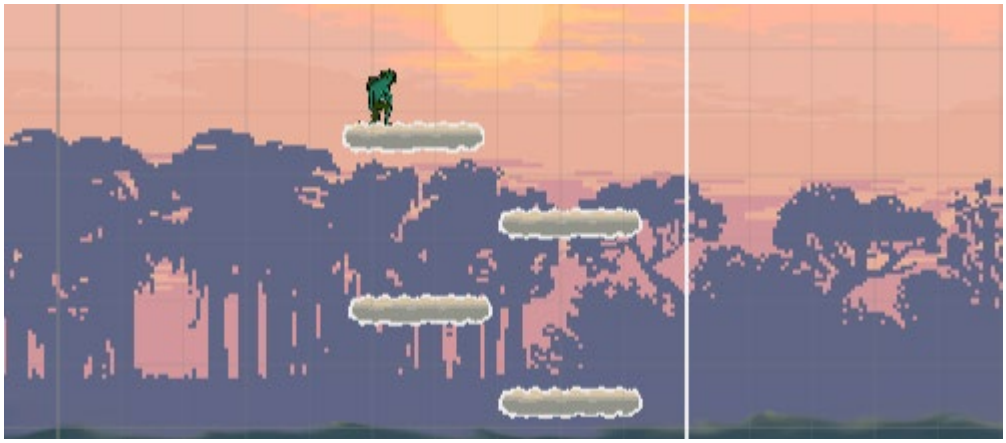
The next level starts with the player being on the underside of the platform they were just on. They are now the right way up and seems like they are going to traverse back the way they came

After some challenging jumps, the next area that the player enters is the same as the first but at a different point in time.



Again more horizontal jumps here force the player to be careful else they will have to restart the level.

The player then climbs up through this scene and into the sky. This is where they meet the boss for the final time.



Dialogue 2

If the player does not complete the dialogue segment, they are sent back to the first level. This harsh do or die aspect of the game not only builds the difficulty but also invokes emotional response from the player and helps the worldbuilding.

End

However, if they do complete the final dialogue segment, they are sent to the end screen which says:

There is also a reused “main menu” button (reused from the pause menu) which allows the player to return to the main menu and brings the game full circle.

Conclusion

What went well

The game comes full circle and all of the platforming is challenging. The platforming and dialogue work together to create a well rounded game.

Even better if

If there was more time, I would have liked to make an introduction scene which allows the players to learn the controls and get a feel for the game and its story.

Changes from original

No introduction

Success criteria review

Criteria No.	Success criteria reference	Was it met?	Explanation
4	There must be platforming elements which challenge the player	Yes	The platforms are laid out where the player has to reach the extent of their capabilities to complete the levels which makes the game challenging and rewarding to play

2	It must include an introduction which introduces the story, grips the user and displays all gameplay mechanics (aka movement and dialogue)	No	This should have been an optional success criteria as it is not needed for the game to work. The introduction and extra polish was not able to be done as the development had run out of time. There is an argument that the controls are intuitive and do not need to be explained. And the story is self explicable

References

URL	What it was used for
None	N/A

Evaluation

Contents

Post-Development Testing

Development test table:

The tests in red are where I encountered errors.

Test NO.	Test	Success criteria link	Expected outcome	Actual outcome
1	Play button	1.1	Enters the first scene	Starts the game
2	Options	1.1	Hides the first main menu buttons	Shows the options men
3	Exit	1.1	Terminates the program	Exits the game
4	Background	1.2	The menu has a background	The background appea behind the UI
5	Mass	4	The player has mass	The player fell endlessl
6	Collision	4	The player has collision with rigidbody entities	The player landed on th ground sprite
7	Left and right movement	4.1	The player character is moved depending on the valid inputs.	The player moved left a right.
8	Jump	4.2	The player character is given vertical directional force if spa is pressed.	The player jumped
9	Jump validation (Ground check)	4.2	The player cannot jump if they are not touching the ground	The player cannot jump midair.
10	Bounce	10	The player bounces	The player bounced hig and stopped the player from progressing quickl as they cannot jump.
11	Bounce (adjust	10	The player bounces at a reasonable level	The player bounced on shortly and the player could progress
12	Dampening	N/A	The higher the dampening, the smoother the movement of the camera	The camera is less jagg when moving
13	Look ahead threshold	N/A	The higher the look ahead fac the further the camera looks ahead of the characters positi is moving towards	The camera looks furth ahead of the player

14	Look ahead fac	N/A	The speed of the camera mov to look in front is increased	The camera looks in fro of the player quicker wh the player starts the mo
15	Look ahead ret speed	N/A	The speed of the camera returning to the player when th is no movement is increased	The camera returns fas
16	Animation stop	9	The animations do not play endlessly when the inputs are stopped	The animation stopped when the movement stopped
17	IDLE	9	The idle animation plays when there are no movement inputs	The player character bobbed up and down
18	Left run	9	The leftRun animation plays w there is left horizontal moveme	The player character ra the right
19	Right run	9	The rightRun animation plays when there is right horizontal movements	The player character ra the right
20	Respawn point	5	The coordinates update to a move in the location	The coordinates update
21	Destroying the player	5	The player object is destroyed	The player disappears from the screen
22	New player	5	A new player is created	No new player appeare
23	Collision respa	5	When the player collides with worldEnd, the players position transformed	The player is “teleporte
24	Accurate respa	5	When the player respawns it moves back to the specified coordinates	The player moves insta to the place specified
25	Respawn Point change	5	When the coordinates of the respawn point are changed, th player is spawned at the new coords.	The player moves insta to new specified coordinates.
26	Message appe	7/10	When the player collides with box collider, the text is made opaque	The text appears
27	Message disappear	7/10	When the player collides with box collider, the text is made transparent	The text disappears
28	Scene switch	7/10	When the player presses “E” whilst colliding with the box collider, the scene is changed dialogue1	The scene changes

29	Asset creation	3	The asset can be selected fro the asset creation menu and modified	The asset was created usable
30	Asset fields	3	All of the fields are present an are correct and modifiable	The fields can be writte and modified and all ar intended
31	Spam	3	The user could not progress through the story without read	The user was able to s through all of the scene
32	Loading	3	The user could not progress through the story without read because there is now a loadin condition	The user could not mov on until the choices and story was written out
33	Button 1	3.1	When the user enters 1, the corresponding state is loaded written out	The user chooses optio and the linked state is loaded
34	Button 2	3.1	When the user enters 2, the corresponding state is loaded written out	The user chooses optio and the linked state is loaded
35	Button 3	3.1	When the user enters 2, the corresponding state is loaded written out	The user chooses optio and the linked state is loaded
35	Invalid press	3.1	When the user enters an optio which is not specified in the choices, the game does not br and carries on until a valid inp pressed	Nothing happens unles valid button is pressed
35	Scene change	3	When the state is loaded whic has a corresponding case to l the next scene, the correct sc is loaded	The user is taken to the correct scene
36	No state	3	If there are no states linked to button, when it is pressed the game carries on until a linked button is pressed	Nothing happens unles link button is pressed
37	Write out	N/A	The text is written out letter by letter at the speed set in the editor	The text is written out character by character
38	Expected text	N/A	The text being written out is th exact same as that stored in t state	Multiple characters wer being written at once
39	Expected text 2	N/A	The text being written out is th exact same as that stored in t state	The text was correct in compliance to the state

40	UI load	N/A	When the escape key is press the UI is shown	The pause menu appea when the player presse the escape key
41	UI close	N/A	When the escape key is press and the UI is active, the UI disappears	The pause menu disappears when the m is active and the player presses the escape key
42	Time Stop on U load	N/A	When the UI is on the screen, time flow is set to 0	The game stops playing
43	Time resume o resume press	N/A	When the UI is on the screen, time flow is set to 1	The game restarts play
44	Game Quit	N/A	When the quit button is presse the application closes and in th console, the debug log is print to show that it has run	The game closes
45	Main menu	N/A	When the main menu button is pressed, the scene is switched the main menu	The scene changes to t main menu
46	Play game from main menu	N/A	After entering the game all of t buttons work and the next sce can be entered and works	The buttons can be pressed and the next scene can be entered b not played
47	Resume time m menu	N/A	When the 1st level starts the ti is flowing at a regular speed	The game is playable
48	Respawn paus	N/A	When the player dies the gam stops	The game is not playab
49	Respawn butto	N/A	When the player presses the respawn button, the game is unpaused and the player respawns	The player respawns a the game carries on - th player falls through the floor
50	Clear velocity	N/A	The players velocity is set to n so that they do not glitch	The player does not fal through the floor
51	Sprite deactivat	N/A	The player sprite is deactivate	The player becomes invisible
52	Sprite re-activa	N/A	The player sprite is reactivated	The player becomes visible
53	Checks	N/A	The player is still controllable	The player moves as normal
54	Music play + lo	6	The music plays on loop in ea scene	The right music plays constantly in each scen

55	Volume control	6	The volume level correlates to percentage of the slider	The volume level can b adjusted perfectly by th slider

Post development tests:

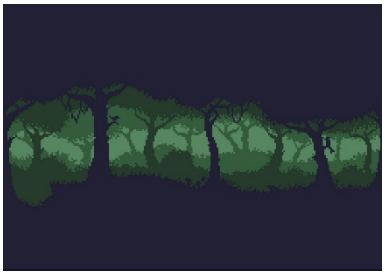
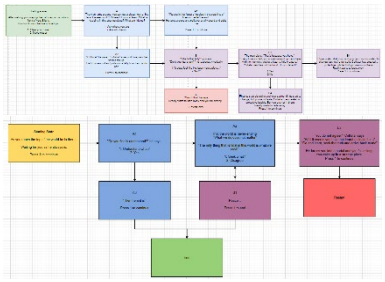
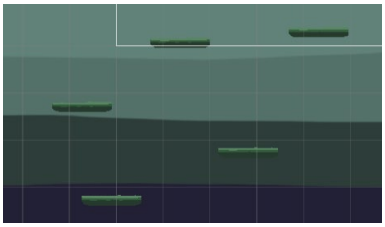
The post development tests are videos which are recorded after the completion of the project. They are stored in a folder which can be accessed to review the tests.

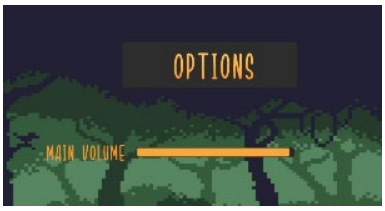
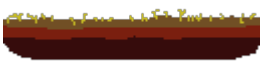
Test no.	Success criteria points	Development test numbers	File Name
1	1-1.1-1.2-1.3	1-4 , 54-55	Test 1 - Main menu
2	3-3.1-3.3	29-36	Test 2 - Working dialogue and UI
3	3.2-5	23-25	Test 3 - Transcript
4	4-4.1-4.2	5-9	Test 4 - Movement
5	6	54-55	Test 5 - Music and sounds
6	7-8	26-28	Test 6 - Interaction
7	9-10	12-19	Test 7 pt1 - Animation and camera Test 7 pt2 - Animation and camera
8	11	N/A	Test 8 pt1 - Artwork Test 8 pt1 - Artwork
9	N/A	N/A	Test 9 - Respawn + pause
10	N/A	N/A	Test 10 - Robustness testing

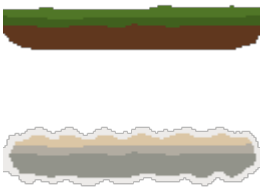
Success of the project

Successes

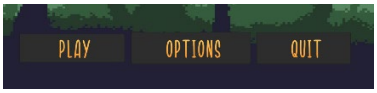
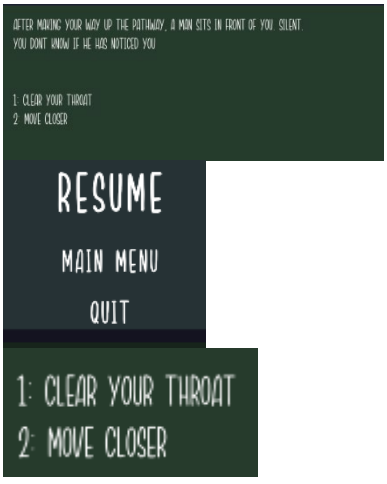
Success criteria point	How it was achieved	Test numbers	Evidence

1	There is a main menu which the player enters at the start of the game. There are buttons which the player can enter the game from and adjust settings	1-3	
1.2	There is an 8-bit image of a wood behind the main menu which is the opening setting for the game	4	
1.3	There is music approved by the client which plays over each part of the game	54-55	
3	There are dialogue scenes in the game which the player can choose responses and explore different parts of the story	29-32, 35-36	
3.2	There is a draw.io diagram with each part of the dialogue script placed into different boxes and they are linked with arrows to show the flow and links between the “rooms”	N/A	
3.3	There are images in each top corner of the characters in the conversation. They are only of their heads so that they do not take up the screen	N/A	
4	The platforms are placed in such a way that in each level the player will struggle to make jumps if they do not time them correctly	5-6	

4.1	The player can move left and right across the screen	7	N/A
4.2	The player can jump up using the spacebar and will fall when the height of the jump is reached. The player is unable to jump if they are not on top of a platform	8-9	N/A
5	The story makes sense in terms of the world. It builds up and presents a problem in the first dialogue scene and climaxes in the second and the denouement is experienced in the last scene of the game	23-25	
6	The music which plays over every scene can be volumetrically adjusted from the options in the main menu	54-55	
8	There is a boss called “collistar” which the player encounters at the end of each level and has to defeat to progress	N/A	
9	The player character and boss is animated as well as the campfire he sits at. Everything which moves in the game has animation.	16-19	
10	The movement is seamless between jumping and moving left and right, furthermore there is a smooth camera script which helps to smoothen the feel of the player movement	12-15	N/A
11	All of the sprites and artwork except the background is designed by me	N/A	 

Partial successes

Success criteria point	How it was partially achieved	Test numbers	Evidence
1.1	The options and exit buttons are fully operational, however the Continue and New game buttons have been merged into one Play button which takes the player to the start of the game. This is because the save system success point was not achieved	1-3	
3.1	There is a box where the text is shown and a way to exit in the pause menu, however there are no buttons, instead prompts to press on the keyboard to choose which response the player wishes. This works the exact same but requires less coding	33-35	
7	There is only one character which can be interacted with and no items in the game. The system for interaction can be reused if items were to be added into the game.	26-28	

Failures

Success criteria point	Why it was not achieved	What could have been done
2	There was not enough time to complete this step after all of the other systems were in place if I was going to meet the deadline. The movement is easy to pick up and the dialogue has prompts therefore it is less necessary for this point to be reached over others	The control scheme could be shown in the options menu which tells the player how to play. But this doesn't solve the issue of introducing the story. If there were more time I would have liked to add a small text scene before the player is thrown into the game
4.3	The wall climbing added too much complexity to the game for new players. As the audience have little experience with video games, this criteria was skipped	If the game was longer, there could be a level where wall climbing is introduced to add more difficulty to the later stages
12	It did not make sense for such a short game to have a save system in place	If the game was longer the player location and scene data could have been saved so that when it was loaded they could resume from where they left off.

Extra points reached

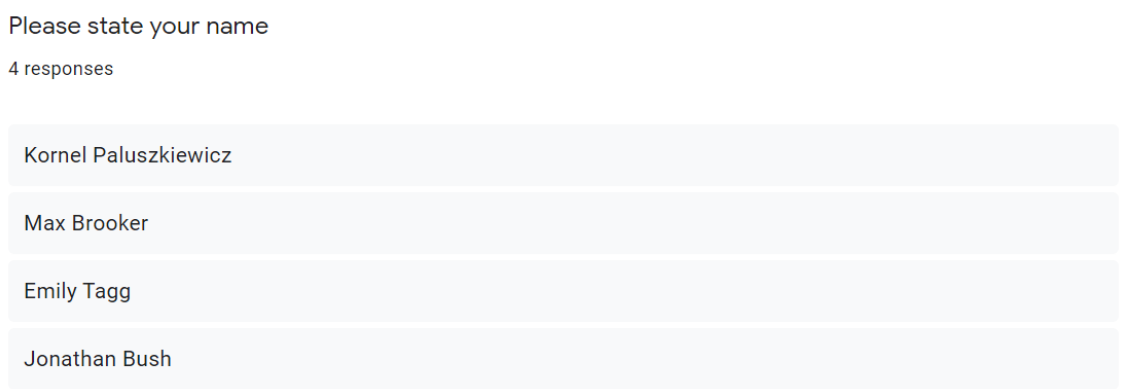
Identified point	What was achieved	Test numbers	Evidence
Bounce	A simple mechanic was added to the movement where the character bounces once when it hits the ground at speed, this can be off putting and adds more challenge into the platforming	11	N/A
Smooth camera	The camera was given dampening to smoothen how it follows the character. This adds to the aesthetic of the game, making it look more sleek and also makes the movement feel smoother	12-15	N/A

Respawn	The player was falling into eternity if they fell of the platforms and now has a way of going back to the start	20-25	
Respawn menu	There is a menu that appears after the player falls of which they have to press respawn to continue. This gives the player a break from the game	48-53	
Pause menu	There is a menu which stops the game when the escape key is pressed. It gives options to the player. They can go to the main menu, resume or quit the game entirely. This is so that the player can exit at any time they wish	40-47	
Text writer	The text in the dialogue box is written out one letter at a time at the request of the client. It gives the player time to read the text and forces them to engage in the story	37-39	N/A

Usability and stakeholder testing

Questionnaire

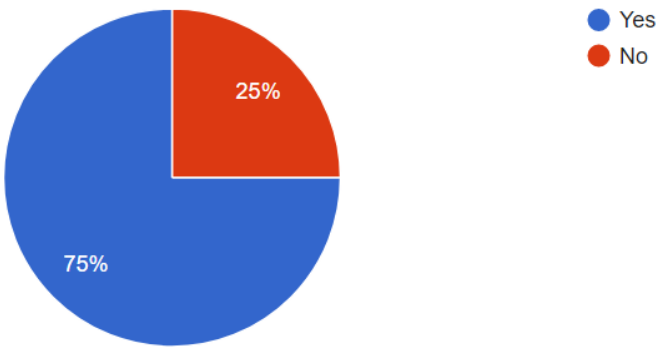
To evaluate the usability of my program, I created a questionnaire for the stakeholders to fill out after they have played the product. These are the results:



The names of the stakeholders who participated

Was the movement easy to learn

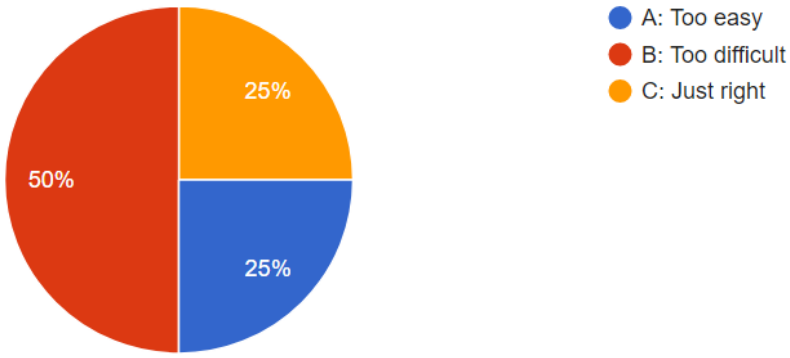
4 responses



The majority of stakeholders found the movement easy to learn

Was the difficulty of the platforming

4 responses



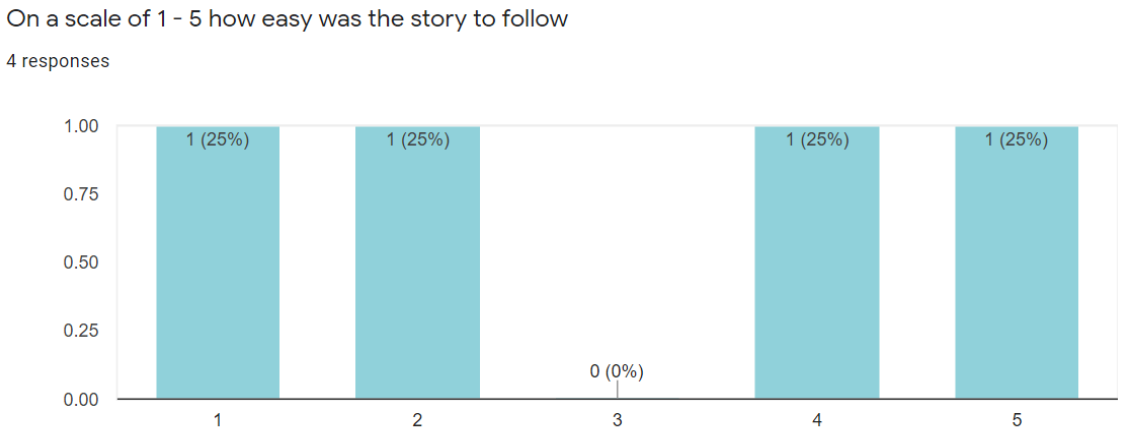
Most of the stakeholders found the platforming too difficult, but all completed the game

When you entered the game for the first time, please describe the atmosphere you felt in three words or less

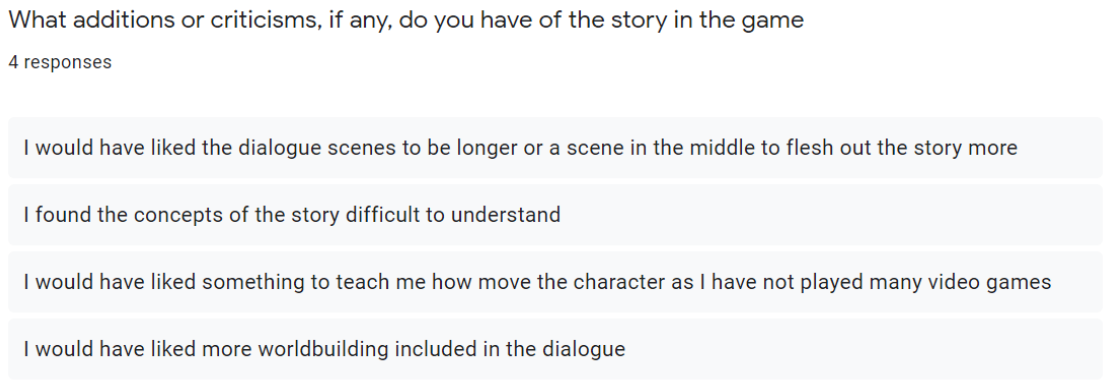
4 responses

Eerie and mysterious
Ominous
Intriguing and relaxing
Scary but exciting

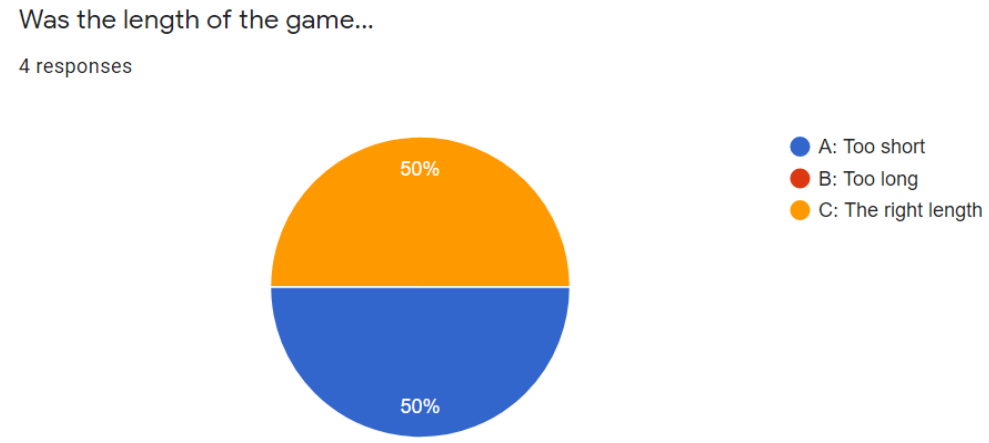
The stakeholders all had similar atmospheres around the games feel at the start



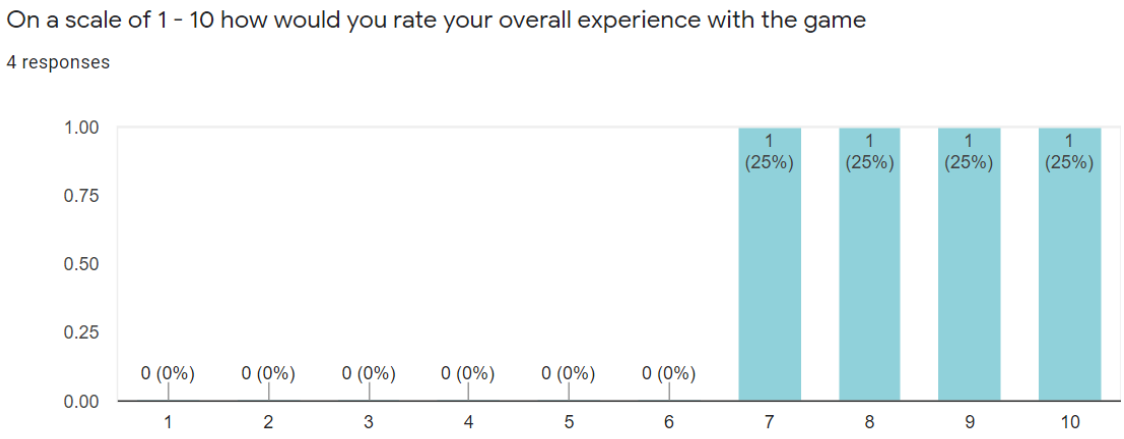
There was a split between the story being hard to understand and easy to follow



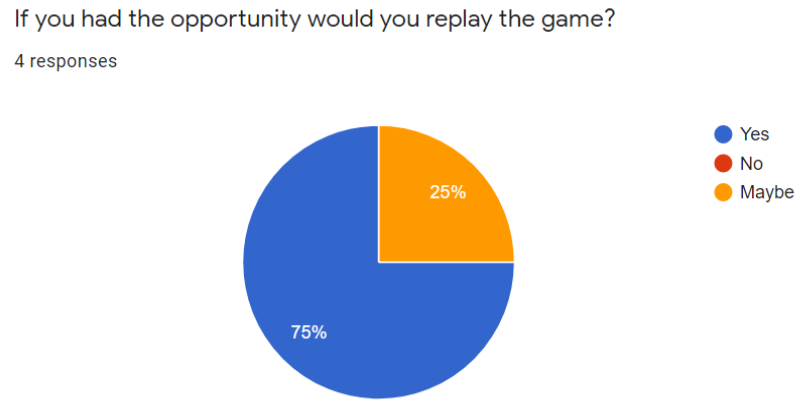
These are short answers to what my stakeholders would have liked to see from the story



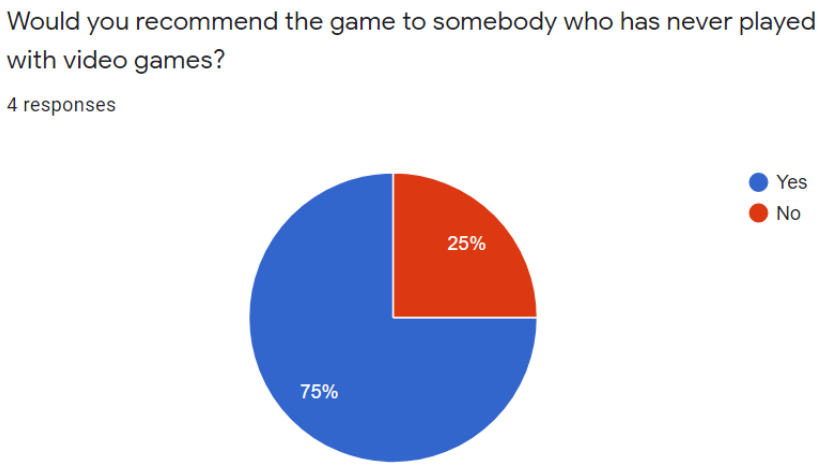
The stakeholders were split between the game being too short and just right



All of the stakeholders rated the game highly



Most of the stakeholders would replay the game



The majority of the stakeholders would recommend this game to new people

Meeting

After the questionnaire I conducted a meeting with the stakeholders to gather their opinions in a qualitative manner which I can analyse further to gain an understanding of the downfalls and successes of the project. These are the meeting minutes:

STAKEHOLDER FEEDBACK MINUTES

Date: 26th September
Time: 18:00
Facilitator: Adam Tagg

In Attendance

Grace Blackshaw

Kornel Paluszkiewicz

Max Brooker

Emily Tagg

Jonathan Bush

Introduction

The stakeholders were seated, and formal introductions were made.

Questionnaire

The project leader Adam Tagg asked discussed each question in the questionnaire.

Question 1: The only struggle with the movement was projected by the Emily Tagg who had the least previous experience with video games. The consensus was that it was easy to learn.

Question 2: This response was mixed, when asked the stakeholders who had the most experience had either no trouble with the difficulty or found it suitable. However, the majority leaned to the platforming being too difficult.

Question 3: All stakeholders were in agreement that the opening of the game worked perfectly, and the atmosphere suited

Question 4: The stakeholders were in disagreement of how easy the story was to follow, some of them found it difficult to grasp the concept of repetition and the existential meaning which the story was portraying

Question 5: Three of the stakeholders shared criticisms of the dialogue and one of the lack of introductory content for the movement

Question 6: Half of the stakeholders found the game to be too short and said that the story would be easier to understand if it was longer and had more exposition

> Question 7: The stakeholders universally agreed that they had a fun time playing the game and did not regret being a part of the process
>
> Question 8: The majority of the stakeholders would like to replay the game. Only one was on the fence, when questioned they answered that they thought that the experience was best going into it blind
>
> Question 9: Nearly all of the stakeholders wanted to share this game with their friends, only one of them believed that it did not suit their friendship group as they do not play games or read

After the meeting, the Client and I discussed our findings. Our summary can be found in the further development.

Maintenance

As the project is a video game, the end product is usually offline and needs few updates. Here are the few cases which may arise where maintenance is needed.

Bug fixes:

Any bugs that have not been found through testing during development will need to be fixed if they are found by users. I propose that the client invests in a website for the game on which there can be a place for users to report any bugs they find. Furthermore, if the game is developed further in the future, more bugs are likely to be present in the new code, therefore maintenance will be needed to rectify this.

Software updates:

As the software that the game was built upon ages, the Unity development team will release updates to the software. This will mean that the maintenance team will need to update the game and release new versions for download. Also, the updates may cause unpredicted issues which will need to be fixed before it is released to the customers

I propose that a very small maintenance team consisting of one or two employees works on the game. If the client plans to update in the future a larger maintenance team may be required.

Further development

My main constraint in this project was the limited time I had available. The time would have been sufficient for a professional game developer, however as I needed to learn how to code in unity, create pixel art in photoshop, animate etc. The game, as it stands, is very short and open ended. This leaves room for future additions to the game to be made. After the meeting and data I collected from my stakeholders here are some new features which would improve the program if it is developed further:

- Extension of the game (More platforming and story levels)
- Introduction scene explaining the controls - To help people with little experience
- Wall climbing - Make the later levels harder if they are added
- More exposition in the dialogue - Makes the story easier to understand

The client also came up with the idea to move the game to other platforms. As it stands, people can only play the game on windows operated computers. The one of the aims of the game was to represent video games like a book. It would be a smart choice to move the game to mobile. People would be able to pick the game up and down and be able to play it on the go. It is offline so no internet access is needed. The extra development needed would include things like touch controls and volume adjustment from the phone buttons. Other than that, the process should be rather simple.