Title page goes here

Acknowledgements go here

Contents go here

## Introduction

The aim of this project was to use the Unity2D game engine in order to create a two dimensional platformer game. Due to the fact that the game engine handles things such as physics, the main focus of the project was on the level design and the artificial intelligence agents that would be opposing the player. It was decided early on that on each level of the game there should be a set of enemy AI agents for the player to overcome. One of the main objectives was to have a variety of AI agents throughout the game that are capable of interacting with the game world and the player in different ways.

Further to this, it was decided that there should be at least one “Boss” agent for the player to face in each level of the game. The AI for the Boss agent was to be more complex and mostly unique from other agents. The idea of using “phases” for the Boss fights was also considered during the planning stage of the project; in each phase the AI would operate differently from the previous one, a new phase would be triggered when the Boss agent’s hit points have passed below a certain threshold.

Aside from the main focus on AI, there were multiple secondary objectives for the project. One of which was to create a power-up system which alters the capabilities of the player-character. An early idea for how this system could work was that a power-up could perhaps be activated by the player character picking up or walking into an item within the game world.

Another requirement for the game was that it needed to be capable of switching between menus, levels and splash screens as necessary. This was required in order to be able to convey a story to the player, but also to allow the player to navigate through menus to save progress or perhaps load into a game that had been saved previously.

A checkpoint system was also deemed necessary so that the player character’s respawn point changes if they pass a certain point in the level. This would help to increase the flow of the game and to reduce the player’s frustration if they are having difficulty with a particular enemy or obstacle. Moreover, a life system would also need to be introduced so that the player is able to attempt to overcome an obstacle or enemy multiple times before having to restart the level from the first checkpoint.

Another objective that was not considered to be key, but was attainable and was likely to have a positive impact on the rest of the game was the introduction of a projectile system so that certain enemies could fire projectiles at the player and deal damage from a distance, and perhaps the player would be able to fire back too.

By the end of development, the project had achieved all of these objectives, with a focus on providing a fun and challenging experience to the player, while minimizing anything that could be frustrating or unfair. Based on play-testing and user evaluation, the project’s solution has been a success. Throughout the rest of this report, the different stages of the process will be addressed in detail.

Firstly there will be a discussion of related work; specifically there will be an examination of games of the same genre. This will be broken down into 2D platformer games made by the triple-A games industry, and 2D platformer games made by the independent developer industry.

Following this, there will be a review of the project specification. In this section, the project proposal will be described in detail and there will be an evaluation of how the specification was arrived at.

The next section of the report will discuss the game design. This is different from the design of the system in that rather than describing technical considerations this section will focus on the creative process of building the game, along with discussion of usability. Game design focusses heavily on the thought processes of the player and attempts to keep the player challenged and entertained while remaining fair at the same time. Often parts of game design can be subtle and unnoticed by the player, and in-fact, it is often when game design has been most successful that it goes unnoticed. Poor game design tends to draw more attention to itself than well thought out game design.

Further to this, there will be a section addressing the system design. This will be a more technical discussion entailing a description of the high level design of the project, evaluating the design method, the design process and the final design outcome. This report will then go on to discuss the design in more detail, taking certain aspects and breaking them down to the implementation level. Design decisions and trade-offs will be evaluated along with the tools selected to create the final product. Relevant qualities of the project such as maintainability, reliability, performance and user-friendliness will be discussed.

Additionally, this report will address the verification and validation strategies used to ensure the game satisfies the specification, covering play testing carried out both by myself and by others and the changes made to the game as a result of the analysis of any findings.

Towards the end of this report there will be a discussion of the final results of the project. This section will attempt to describe the final build in such a way as to convey a clear picture of how the entire system works. There will also be an analysis of the results of the questionnaire taken by all who volunteered to play the game as part of the evaluation.

## Related Work

In this section the various modern 2D platformer games will be examined and discussed. Both the triple-A games industry and the independent developer games industry create a large amount of 2D platformer games. There will be a comparison between the two industries and an analysis of any patterns that might emerge upon examination of the typical methods associated with each industry and its development process. The 2D platformer genre of gaming itself will also be reviewed.

A platformer is defined by Wikipedia to be “A video game which involves guiding an avatar to jump between suspended platforms, over obstacles, or both to advance the game.” And while that is true of the first few games in the genre, in recent years they have become so much more. Platform games originated in the 1980s and at one point they were the most popular genre of video game, at this point it was estimated that between a quarter and a third of all console games were platformers. No other genre of gaming has ever been able to achieve that market share.

Due to hardware limitations in the 1980s, early platform games were forced to have a largely static playing field with not a lot going on. “Space Panic” a game which is sometimes credited with being the first platformer game, had gameplay based on climbing ladders between different floors. More games like this emerged and eventually UK press began to refer to the floors as “Platforms” and to the games as “Platform games”. US press however referred to them as “Climbing games”. Eventually though “Platform games” became the internationally used name for the genre, however there is still debate as to whether or not games like “Space Panic” should be considered to be platformers in the modern sense of the term as there was no jumping swinging or bouncing involved.

Nowadays platformers are far less dominant, but they are still a perfectly viable genre of game to produce, economically speaking, with a number of modern platformers selling millions of copies. Modern smart-phones and other mobile devices have brought a renewed popularity to the platformer genre. The reason for this is that platformers often have very simple controls and so a mobile phone touch screen for example is an acceptable method of input. While other modern games like “Call of Duty” for example, require precise movement and aiming that is better captured by a keyboard & mouse or a gamepad. MMORPGs like “World of Warcraft” are infamous for having complex controls that can take players a very long time to truly master. At the peak of its complexity, competitive players of “World of Warcraft” could easily be expected to bind 25 to 30 different functions to their keys, often having to use modifiers to be able to cope (For example have one spell bound to the “x” key and another spell bound to “shift+x”).

So one major advantage the platformer genre has is clearly the fact that the games can reasonably be developed for a variety of platforms, whether it’s for the PC, a console, or a mobile device, each platform has adequate input devices to cope with what is required by the game. Not only this, but platformers generally are not particularly performance intensive games, the art style is often quite cartoon like which allows developers to use smaller textures and still have the game look good, while “realistic shooters” such as Call of Duty use a much more realism-focussed art style and as a result need to use larger textures for graphics. This means that graphical performance is quite intensive and unless a platform is specifically built to play games (such as a console or a gaming PC), it would be unlikely that it would have the necessary hardware to actually run the game smoothly, and this is just graphics, on top of that there is a need to perform a lot of physics calculations and there generally tends to be a lot more going on in a modern shooter than in a platformer.

Other modern genres cannot boast the same advantages, and due to this, the mobile market is limited in what genres can appear for sale, and this obviously allows platformer games a larger share of the mobile market as a result.

Platformer games are still made today, both by the triple-A games industry, and by the independent developer industry. The term triple-A is applied to games with large amounts of funding for development, and lots of advertising and promotional power. Due to this, when triple-A games are released, players expect the game to be of a very high quality both functionally (released with very few bugs and solid game mechanics) and graphically.

After the video game crash of 1983 (caused by companies churning out games at a very high rate often riddled with bugs, or unplayable, or impossible to complete) gaming companies needed to find a way to allow players to distinguish between high quality games and badly developed ones. An example of this was Nintendo using the “Nintendo Seal of Quality” on games to show that they had been properly tested before release. This caused a shift in development attitude in the games industry where developers would be striving to release games with absolutely no bugs whatsoever, and as such the term triple-A came about to describe these games.

However, many gamers today have a feeling of distaste towards triple-A games companies due to the business strategies they employ. For example, Electronic Arts is a games publisher perhaps best known for games such as FIFA or Mass Effect, both of these series are of a high quality but it is widely known throughout the industry that Electronic Arts often buys over small successful developers just to shut them down. This of course upsets many gamers as it stifles creativity in the industry, and this is the main problem the triple-A industry faces today.

For the most part games released by triple-A companies are of a high quality but they often stick to a “winning formula” and offer nothing particularly new or interesting to gamers. Publishers are also guilty of rushing developers to release the game so triple-A games are starting to release with less and less content and in some cases (For example, Ubisoft’s Assassin’s Creed: Unity) the games are released filled with bugs.

However the most popular 2D-platformer series was created by the triple-A industry. The Mario series published by Nintendo has been running for over 25 years and is the best-selling video game franchise in history, with over 262 million combined units sold. As such it is a good place to look for inspiration. Obviously this project could not aim to compete directly with the series as the games are produced by teams of people with large amounts of time and funding available to them.

The Super Mario games follow Mario through the “Mushroom Kingdom” with the player usually controlling Mario. He is usually accompanied by other characters such as Luigi and Princess Peach. Common game mechanics are running and jumping across platforms as well as jumping on top of enemies in order to defeat them. The plot generally consists of Mario and Luigi attempting to rescue Princess Peach from Bowser, the main antagonist.

Super Mario 3D World, the latest pure platformer instalment of the series was released in November 2013. The game was largely positively received, scoring 9.6/10 on IGN’s review. It was praised for its good looking levels and orchestral soundtrack. The game introduced multiplayer cooperative play to the series and was praised for doing so in a seamless way. According to IGN it “takes slightly smaller steps in the grand scheme of the platformer genre” but the “light-hearted co-op play proves that Nintendo still knows exactly how to tweak the Mario formula in fun ways.”

Super Mario games are widely regarded to be good games that are well made and robust, upon release game mechanics work well and the games always look good, they provide a reasonable challenge to the player and run smoothly on whichever platform they are released for, and as such most of these games are well enjoyed by the gaming community.

Another successful platformer series created by the triple-A industry is “Sonic the Hedgehog” developed by Sonic Team and published by SEGA. The first game in the series was released in June 1991 and featured gameplay involving collecting rings as a form of health and simple controls where jumping and attacking were both controlled by a single button. The game was a large success and established the SEGA Genesis console as a big player that could compete with the Super Nintendo.

The series is still running with the most recent release being “Sonic Dash 2: Sonic Boom”, released for android and iOS systems in 2015. Clearly SEGA wanted to take advantage of the mobile market, and it turned out to be a good decision as over 100 million copies of the game were downloaded. The game is actually free to play, with revenue instead being generated by optional micro-transactions to unlock other playable characters or power-ups.

In reviews the game was praised for being “perfectly competent” and for the fact it had plenty of content but was criticised for not bringing anything new to the platformer genre and for the fact that it showed SEGA still don’t know how to bring anything fresh to the series.

So it could be argued that while technically these games are very good, they are somewhat bland compared to other modern day games. Any negative reviews the games receive usually target the fact that series seems to stick with the same characters, same settings and frustratingly similar plots, and while some new game mechanics might be introduced in new instalments there generally aren’t many. What this tells us is that due to the success of the series and the fact that the current model works well financially, publishers aren’t willing to take creative risks, and this frustrates some gamers.

Conversely, on the opposite end of the spectrum there are games made by independent developers (also known as “Indy” games). Generally these games don’t have a lot of funding and this means developers are more likely to take creative risks. Also due to the fact that they usually don’t have an existing brand name to appeal to fans, they usually need to provide something unique in order to attract people to the game. These types of games are probably closer to what could be produced with this project due to the time constraints and the lack of funding.

A good example of a 2D platformer that was released fairly recently by an independent developer is “Super Meat Boy”. It receives praise for precise controls and unique level design. The main selling point of Super Meat Boy is its difficulty; the game is extremely hard and was designed this way on purpose. A lot of gamers like a challenge and with the triple-A industry these days trying to target such a broad audience they often reduce the challenge meaning people have to look to the Indy market in order to find a game that is hard to beat. The aesthetics of the game compliment this difficulty as often the animations are excessively violent, but not without showing self-awareness, the violence isn’t supposed to be taken seriously. Overall this game achieved great success and received multiple awards from IGN, GameSpot and GameTrailers.

According to GameSpot “Super Meat Boy is the digital embodiment of the idea that pleasure can spring from pain” and this is evidence of the fact that the game was designed to be extremely difficult in the hopes of bringing something unique to the genre. It is this high level of difficulty that actually makes the game worth playing in many people’s eyes, because although playing against an excruciatingly difficult game can be frustrating, there can be a great deal of satisfaction gained from beating a particularly hard level.

The developers reinforce this satisfaction by playing a unique sort of replay once the player has cleared a level. The replay is an amalgamation of all of the player’s previous attempts at clearing the level, all played at once. So a player who has failed a level numerous times will see multiple avatars on screen all dying to whichever obstacle the player failed to overcome on that particular attempt. It makes for quite a cathartic experience and is a rather funny little addition to the game.

It is the sum total of unique artwork, precise and rewarding controls, high difficulty and a cathartic experience that make Super Meat Boy worth playing. The game has achieved great success both commercially and critically and should be an inspiration to anyone who wishes to develop a platformer game.

Another fantastic example of a successful Indy platformer game is Braid, which could really be considered to be on the opposite end of the spectrum from Super Meat Boy in that Braid takes itself really quite seriously. The game is praised for its plot, gorgeous visuals and difficult puzzles. However there have been plenty of platformer games that have offered these things before. What separates Braid from the pack is a unique time reversing mechanic that allows the player to turn back time with the push of a button.

This mechanic can be used to avoid being killed by an enemy or to retry a mistimed jump, but this is not the first time that players have seen a time reversing mechanic in games, “Prince of Persia: The Sands of Time” immediately springs to mind as a game that heavily features a time reversal mechanic. However the unique aspect of it in Braid is how it is used to solve puzzles, because some objects and enemies in the world are unaffected by the player reversing time, for example the player could unlock a door and then reverse time and the door would still be unlocked. This provides a unique challenge to the player, giving them puzzles that need to be solved a way that makes them think outside the box.

It is this unique experience that the game is so often praised for. Braid reached both commercial success and critical acclaim, with IGN scoring it at 8.8/10. They say that “Part of the reason that Braid sticks with you is it’s also one of the most sombre games in recent memories.” Which is certainly true, but perhaps a lot of players found this quite refreshing as the 2D platformer genre is quite well known for its upbeat adventurous attitude, or in other cases a rather comedic tone is used instead, so Braid offers a new experience. The IGN review really highlights this, stating “In an age when many games feel like clones of one another or are designed to be easy to beat, Braid sticks to its guns” “It’s like an invigorating breath of fresh air”. It seems that offering something new and unique is central to success in the Indy market.

Limbo is another Indy platformer that was released in July 2010. It was an extremely successful game, receiving the VGX Award for Best Independent Game and a score of 9/10 on IGN. Limbo is a rare game in the sense that there is no text or dialogue to explain to the player what is going on, instead communicating the story organically through the setting. There are also no cut scenes or loading screens and according to reviews this all helps add to immersion. Another unique aspect of Limbo is that it is in black and white, this helps add to the gloomy tone of the game.

The game becomes quite difficult but avoids being frustrating by having the player character respawn right next to where he died so that the player doesn’t feel the irritation of losing progress every time they make a mistake. Another aspect of the game that is used in a refreshing way is the sound, instead of simply playing background music while the player works their way through the game, sound is used to give cues to the player that could help them solve a puzzle. There are also ambient noises in the background that help to add to the overwhelmingly bleak atmosphere. It is this atmosphere that Limbo is most often praised for and it is likely a large part of its success. Again this is something rather new to a genre that is usually light-hearted and fun.

It is clear based on the success of these Indy games that they target specific audiences and that their success is based on standing out from the crowd with unique mechanics. While the time constraints for this project mean that it cannot compete with these games, it is certainly worth drawing inspiration from them in order to produce a high quality and fun game.

## Problem Description and Specification

The main goal of this project was to create a two dimensional platformer game using the Unity 2D game engine. Due to the fact that a large amount of the complexity in the back end of a game is actually in the game engine itself with things like physics and collision needing to be calculated in a way that is both resource and time efficient, it was decided that since the use of Unity removes this complexity, this projects main focus should be on creating AI agents to populate the game world.

In order to reach a specification, this problem had to be refined so that success was actually measureable; as a result one of the main objectives of the specification was to create five standard enemy types, each with distinct simple AI patterns. This objective created a clear and reachable goal to strive towards.

Along with AI, the project proposal also stated that time should be spent adding other features to the game such as a power up system that changes the state of the player character, and perhaps even a level up system in which a player could alter the attributes of their avatar.

From this project proposal, several main objectives were drawn up that the project was expected to have achieved by the end of the development cycle. There were also several additional objectives that could be tackled if there was enough time to take the necessary steps to implement them. Each objective is addressed below:

1. “Create a system that allows the switching of menus, levels and cut-scenes or splash screens.”

In order to be able to provide save/load functionality to the player, or the possibility of letting them adjust settings such as resolution, or controls, there needed to be a menu system. Obviously for a game with multiple levels there also needed to be some sort of functionality that allowed the game to transition between levels. It was also thought that in order to provide context to the player (i.e. justify the actions of the avatar in terms of a plot) it might be necessary to have a system that could control the use of any cut-scenes or splash screens.

1. “Create a menu screen that has the options of beginning the game, changing the settings or exiting the game”

It is necessary to have a landing screen for the player to choose whether they want to start a new game, load a previously saved game or change settings, this is without a doubt preferable to just instantly starting a level.

1. “Use the Unity2D sprite animator to render an animated character on screen. The character should be able to switch sprites depending on user input.”

The reasoning behind this is fairly clear, the player will need an avatar to control, and in a 2D game the best way to create an animated character is through the use of a sprite sheet. Of course, there will need to be different sets of sprites for different animations (i.e. running vs idling) and how the game engine selects the animation to play will need to depend on user input. Animating enemy A.I. will work in a similar fashion.

1. “Use the Unity2D game engine to implement physics and collision detection into the game.”

While the game engine does the back-end work for physics and collision, it is still up to the developer to put these tools to use. Obviously the actual force of gravity and other physics along with the areas that have collision are dictated by the developer.

1. “Create at least five standard enemy types, each with distinct simple AI patterns.”

This was partially addressed earlier, there needed to be a measurable objective for the AI so that the project could be deemed successful or unsuccessful in fulfilling the criteria in a way that is not subjective. The reason it was decided that there should be multiple agents each with their own simple AI is so that there would be more variety in the game in order to keep players entertained.

1. “Create at least three levels for the game.”

While it’s true that perhaps all of the enemies and obstacles could have been put in one large level for the player to face, it made more sense to have separate levels where players could learn how to overcome certain obstacles and enemies separately so that it would be perhaps less overwhelming. Also it could be argued that having multiple levels gives a greater sense of achievement to the player as they will feel a sense of accomplishment each time they complete a level.

1. “Create at least one boss type enemy per level with more complex AI.”

In order to keep the game interesting and challenging to the player it makes sense to have more complex AI agents for them to face now and again to put their skills to the test. Having one boss agent per level is also a good idea as it poses the challenge to them after they have learned from enemies and obstacles throughout the level.

1. “Create a checkpoint system so that the player’s respawn point changes if they pass a certain point in the level”

This is arguably not necessary as older arcade platformers did not have this functionality. However part of the appeal of arcade games was the relentless challenge they offered and this helped draw players in through the competition to see who could be first to beat the game. In recent years, with there being less local gaming and more online gaming, players are less likely to be motivated by trying to be the first to beat the game, so due to this it is best to attempt to reduce frustration caused by difficulty through systems such as checkpoints.

1. “Create a life system so that the player can attempt a level multiple times before having to restart”

This is largely similar to the reasoning behind implementing a checkpoint system. Players should perhaps be punished for mistakes in order to keep the game challenging, but not so harshly that it takes the fun out of the game.

Objectives that were considered to be attainable but not crucial will be addressed below:

1. “Add more variety in standard enemies and bosses.”

Basically variety is always a good thing in a game and if time allows it adding more would certainly be beneficial as long as any new AI agents aren’t rushed.

1. “Add more levels and transitions to the game.”

This is similar to the previous objective, adding more length and variety to the game could add to the player’s experience.

1. “Add a projectile system so that certain enemies can fire at the player (and perhaps the player can fire back).

The addition of a projectile system would make the game more dynamic, both by increasing the potential variety of enemy AI agents and also by possibly allowing the player a new method of attack. This objective would certainly improve the player’s experience if time allowed it to be implemented.

1. “Add the option to save or load previous games.”

This objective is all about player convenience. If someone doesn’t have the time to beat the game in a single play through, it is unlikely that they would want to play through the levels that they have already beaten previously. So a system that allows them to save their progress and load it later would undoubtedly improve player experience.

There were also a number of non-functional requirements that needed to be addressed in the specification, the first of which was framerate. It is important for the game to run smoothly, a low or jumpy framerate can make it very difficult for the player to play the game properly, it is also detrimental to the amount of fun a player can have. As such a minimum of 24 frames per second is required, with the framerate remaining consistent throughout the game.

The next requirement discussed was usability, it is important that the player experience isn’t degraded by difficult controls or unintuitive level design or AI mechanics. It is also important that any HUD is shown clearly and any menu screens are easy to navigate. This was to be evaluated through the use of different play-testers.

Additionally, response time was important to keep in mind. The time between the player pressing a key, and the game responding to it should be kept low. A latency of 500ms is considered very high when gaming so I would like to keep response time well below that.

Another issue to think about was the amount of system resources the game would require. It was decided that the game should not need to use any more than 1024MB of RAM while running. The game also should only require a reasonable amount of Hard Disc space. Some modern games can take up well over 20GB of Hard Disc space, but a small platformer game would not be expected to exceed more than 5 GB.

Finally, it was decided that the platform the game needed to run on was Windows 7. This was largely down to the fact that it was being developed on Windows 7 and so this was the platform that the game could be most thoroughly tested on. It was also the platform most widely available amongst the people who agreed to play-test the game so windows 7 simply made the most sense, due to the fact that the time and resources to develop for another platform were not available.

Overall the project problem was tackled in an iterative manner, often during implementation or play testing, some issue would arise and the design stage would need to be revisited in order to resolve the issue. This was to be expected however as games are never developed simply by working through each stage of the software development cycle without ever revisiting decisions. The iterative approach worked well as it allowed a certain part of the game to be developed, tested and if necessary redesigned before moving onto the next objective.

## Game Design

In this section, various game design choices will be discussed and justified in terms of how they would affect the game, and the player’s experience. This section will not address technical design choices such as design patterns used for implementation. Instead this section will discuss issues like level design.

The first decision that was made during the development process was on which collection of sprites / tiles to use for the game. To clarify, these are art assets that are used to render characters onscreen. They are a series of 2D images that can be used to create an animated Player Character or an enemy agent. The tile-set is what is used to craft the level. Due to time constraints, art assets were bought online.

After searching through many different sets of sprites, the one that was finally selected was called “Alien sprites”. This was due to the fact that this sprite package contained a large variety of sprites that could potentially be used as AI agents. On top of this there were huge number sprites for animating the player character, meaning it would be capable of anything that would need to be implemented into the game later. Specifically, there was a sprite sheet for firing a rifle, which would allow for the development of a projectile system later should there be time.

Another point that further cemented this decision was the fact that there doesn’t seem to be many modern platformer games where the player is fighting off aliens (though there were plenty during the genre’s early stages). So this would help separate the game from anything else currently on the market. To compliment the “Alien sprites” it was decided that a “sci-fi” tile-set should be used. This would help to contextualise the enemies, but also the power ups. The sci-fi setting would easily allow for power-ups to be justified as some sort of “technical marvel” within any sort of plot.

Creating the first level also came with a great number of design choices. The first of which was how to introduce the player to the controls of the game. It was decided early on that there would be a dialogue that would explain to the player how to control the avatar; however it was also necessary to ensure that the player had a basic grasp of what they were doing before introducing them to too much adversity.

In order to do this, it was decided that the first obstacle the player should face is a “spike pit”. This is because a spike pit will not actively try to kill the player, it is a static obstacle that the player can prepare for and take their time to overcome. In this way it was a perfect test to ensure that the player has a basic grasp of the controls before allowing them to advance on to the rest of the level, as it would require them to have a grasp of running and jumping, and how to coordinate them both in order to achieve the timing necessary to cross the spike pit safely.

After this, to further ensure that the player had a good grasp of the controls and the fundamentals of jumping to overcome obstacles, the next obstacle they faced was a series of platforms that they would have to jump on-top of in order to reach the next floor to advance in the level. This was done in such a way that if the player failed to navigate the platforms they would not be punished, as the only thing below the platforms was solid ground and the player character would not take damage from falling. This would allow the player to attempt the obstacle and fail it as many times as necessary for them to overcome it. Once they had overcome it, the assumption could be made that they have a firm grasp of how to move their character in an efficient manner. The only controls left to teach the player were the combat controls.

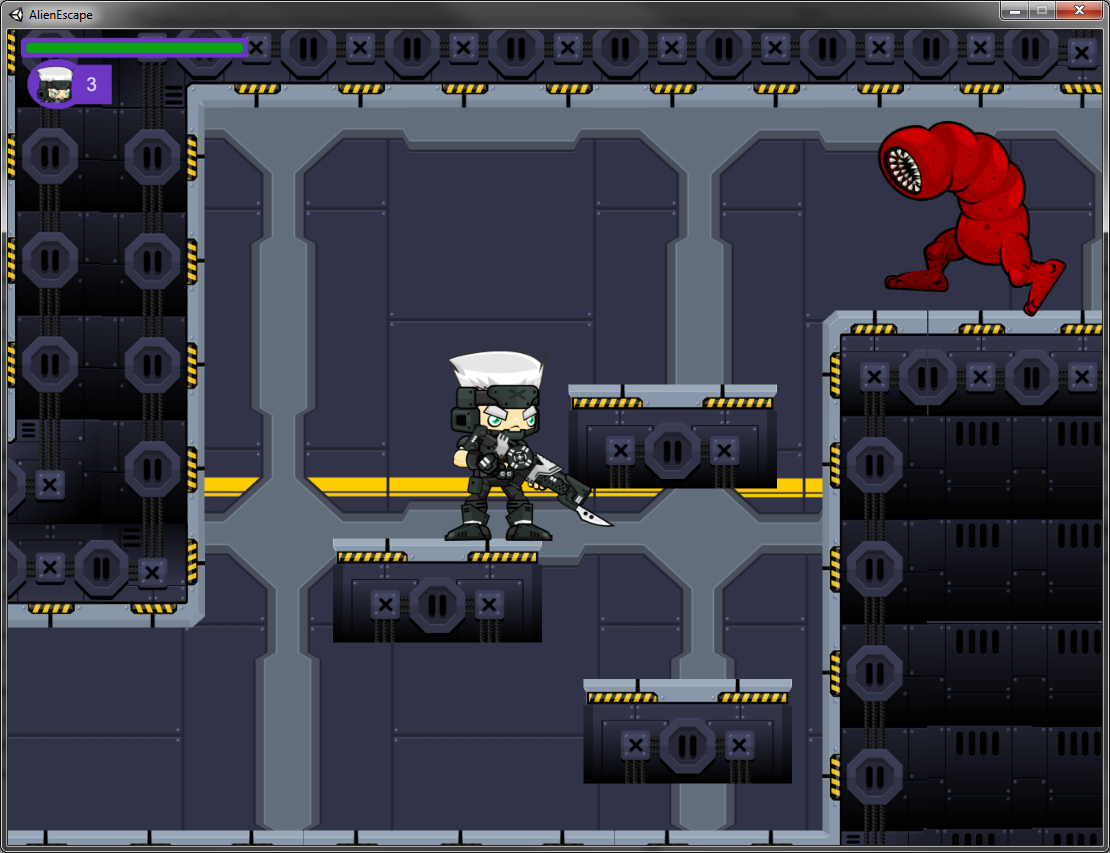
Similarly to the movement controls, the combat controls were conveyed to the player at the start of the level using a dialogue box. However before the player could face a real threat, it was necessary to ensure that they had a grasp of these controls. As such it was decided that the first enemy AI agent they should face, should be the one with the simplest AI. This also made sense in the context of any plot, and in the overall difficulty curve (obviously it makes no sense to have the player face the toughest enemies at the start of the game and the weakest enemies at the end).

So the first AI needed to be fairly simple, but still pose a reasonable threat to the player. The first step was to select a sprite from the “alien sprites” package. Most of these sprites looked quite menacing and so they weren’t suitable for a simple AI that the player would be facing at the start of the game. One of them however was a small slime that looked slightly dangerous but mostly unthreatening. This was perfect for the job of conveying to the player that they have come across a threat, but nothing that they should not be able to overcome.



The player was first introduced to this AI after the player character drops down onto a flat platform. There are no other obstacles or enemies on this platform; it was simply a flat platform with this single “Slime” AI on it. This AI can of course deal damage so if the player is incapable of defeating it, they will lose a life and have to make their way back to that point. However the AI can be easily overcome with a few melee attacks so as long as the player remembers what the controls are, they should be able to beat it.

The player then faces a number of more difficult jumping obstacles and one more “Slime” AI before meeting the second AI of the game. It was necessary to convey to the player that this AI was more dangerous so a more threatening looking sprite was chosen. This AI was to be slightly more intelligent. As such it was decided that it should attempt to hide from the player whenever it is unable to reach the player to attack them. This would mean that the player would be incapable of simply shooting at the AI from another platform while it is helpless. This will be demonstrated in the following screenshots.



In the first screenshot it can be seen that while the player is on a platform that is too low to allow line of sight to fire his rifle at the enemy, the “Worm” AI will proceed to patrol to the end of the platform at the upper right, because it is safe to do so. However if the player was to jump onto the next platform, they would be at a height that would allow them to fire their rifle at the “Worm” AI. So the AI detects this and attempts to hide by running off-screen to the far end of its patrol and remaining there until the player either moves away, or comes into reach.



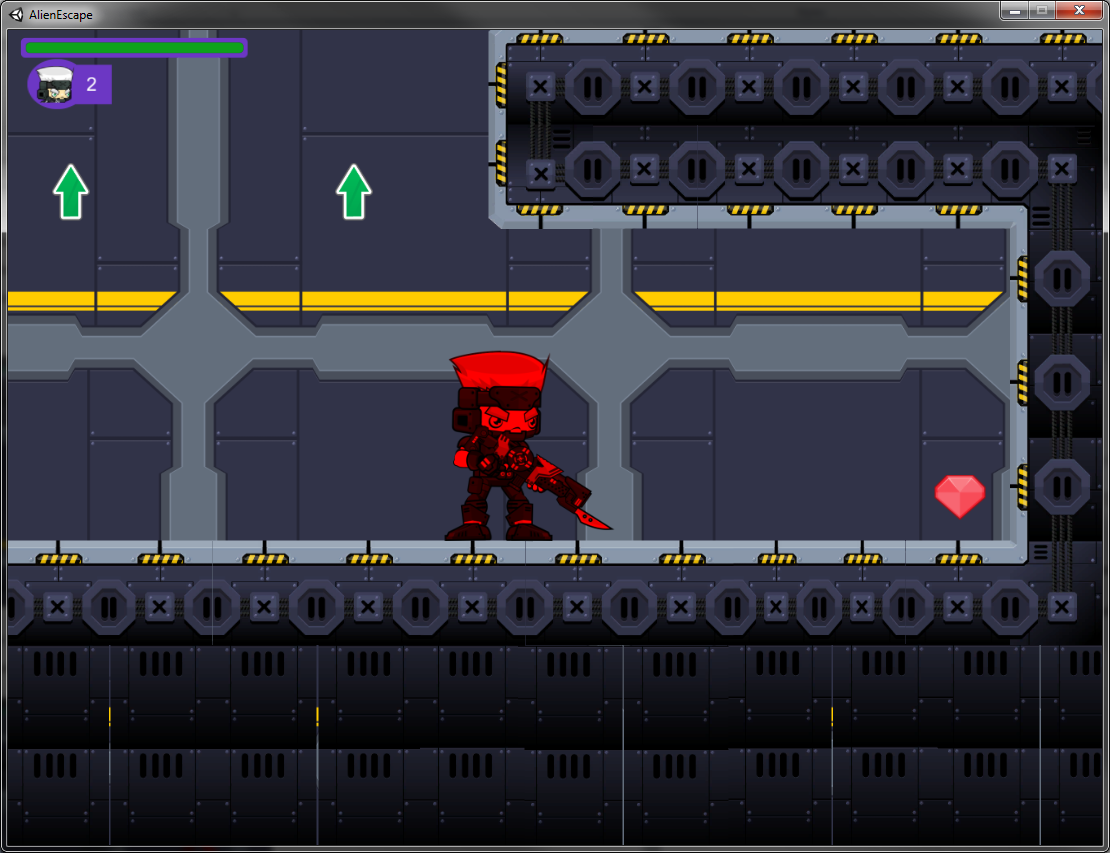
Since the “Worm” AI isn’t able to jump across platforms, or use ranged attacks, its “reach” is defined as the platform that it is patrolling on.



The “Worm” AI moves fast and closes the gap to the player fairly quickly, this can cause newer players to panic, but overall the worm has the same health as the slimes and deals the same amount of damage. Its advantages are its speed and intelligence but the player should still be able to overcome it fairly quickly. If not though, there is actually a checkpoint on the second of the three platforms leading up to the “Worm” so even if the player character is killed, they will not have to make their way there from the start of the level so this should help minimize any frustration that could arise.

The next game mechanic the player is introduced to is the power-up system. The first power-up the player comes across is the jump power-up. This power-up allows the player to jump whilst already in the air, essentially allowing them to jump infinitely to reach heights they would normally be incapable of reaching. Of course, the player doesn’t know all this yet so the game design challenge here is coming up with a way of communicating that information to them, but it should also be kept as a puzzle so that the player feels a sense of accomplishment once they have worked out what it is that they need to do.

The first thing that needed to be tackled was how to allow the player to activate the power-up. There were a few different possibilities: Firstly, the power-up could be activated automatically when the player reached a certain position in the level, but while this would certainly work for any jump puzzles, it would be less ideal for any other power-ups like a melee-attack power-up for example so this was ruled out. Another option was to allow the player pick up some sort of item, which would then despawn but would grant the player the power-up for a temporary amount of time. This seemed better but it had a problem, if the player collected the power-up and then failed the jump puzzle and then ran out of time on the power-up, they would be unable to complete the level. So this idea was modified so that the power-up item would not despawn once collected.



The second issue that arose was how to communicate to the player that the player-character was powered-up. The idea that was settled on was changing the colour of the sprite, to match the colour of the power-up crystal that the player had picked up, and this seemed to work fairly well. Along with this, the game needed to be able to convey to the player when the power-up was about to wear off, particularly for jump puzzles as it was possible that if the player overestimated the amount of time remaining on their power-up they could get themselves killed on obstacles like spike traps, this would be particularly frustrating so it needed to be prevented. The solution was to make the player character sprite change colour between what it normally is and the colour of the power-up at a high rate to make it appear as if the colour of the player is flashing. This would occur three seconds before the power-up wears off in order to warn the player that they should take care not to overestimate the time remaining.

So now that the player knows when a power-up is active on them, the next thing they needed to learn was what the power-up actually does. This was done through a combination of dialogue hints and level layout. When the player nears the jump power-up crystal, a dialogue box will appear hinting at the fact that the player might be able to use it to reach higher places.

The first level was laid out in such a way that when the player reaches the first power-up it should be fairly obvious to them that they haven’t missed some other path before that point. The layout should hint to the player that they have reached a new kind of obstacle and that they need to find a new way to deal with it. This paired with the dialogue hints should be enough for players to work out what needs to be done without being directly told.

Due to a slight flaw in level building, there are certain points where a player might have to make a leap of faith. This seemed an unfair thing to expect from a player in a game that has spike traps and other obstacles. The solution to this (aside from rebuilding the level as that would cost too much time) was to have an arrow in the background of the level, showing the player the way to go. Of course, if the first time the player encountered an arrow was next to a leap of faith, they might be hesitant to trust it, to solve this arrows were put all throughout the level, even in places where the path should be obvious to the player, the hopes here were that this would help to gain the player’s trust in them so that by the time they encountered the leap of faith, they would see the arrow and know that it was safe to make the jump.

The next AI the player faces is the “Bat” AI. Due to the fact that the other two AI agents in this level so far are unable to leave the platform they are patrolling on, it seemed like a good idea to create an agent who could chase the player across platforms in order to keep a good variety in the game. The “Bat” sprite was perfect for this as it was a flying creature so it could simply move along the x and y axis without the need to attempt any jumping across platforms.

The player is introduced to an obstacle involved a series of vertically moving platforms. To get past this obstacle the player needs to time their jumps correctly, otherwise they risk falling into a spike pit.



Most players should not find this to be too challenging as they have already proved their proficiency with jumping before this point and the main reason for this obstacle existing is to keep things fresh in the game as repetition generally isn’t fun. Players are first introduced to the “Bat” AI towards the end of this obstacle, the reason for this is that players might be starting to get into the swing of things and might be quite relaxed while completing the platform obstacle, in a game this can lead to boredom. As such the “Bat” appears at a moment when the player isn’t expecting, and this helps to keep the game feeling dynamic. It also helps to test the player’s reactions.

The next obstacle the player should meet is the “Exit Door”. This is where the player would leave the level and the game should load them into the next one. However, in 2D-platformer games, puzzles and exploration are commonly used tools to keep to player engaged. So when the player first encounters the “Exit Door” it is locked, and a dialogue informs them that to unlock it they must find and pull two levers which correspond to each of the switches next to the door. Once the door levers have been pulled, the door will unlock and the player will be allowed to advance to the next level.

Due to the fact that the player needs to backtrack through the level to reach the exit door once the levers have been pulled, it was necessary to implement some way of allowing them to ascend back up any drops that they jumped down. The vertical moving platforms would have been capable of this but as they have already been used for an obstacle previously in the level, using them again as a means of transport could become slightly boring. Instead a ladder was used. This involved putting the ladder sprite in place, giving it a collider and writing a script for when the player climbs it. It also involved creating a new climb animation for the player character.



The rest of the first level consisted of more “Slime”, “Worm” and “Bat” AI agents and other jump obstacles, with each of the two levers being at the end of two different jump obstacles. Arrows lead the way to each of the levers in order to avoid frustration if the player is unable to find them.

Once the player uses the exit door to start the level transition, they face the first boss. The level on which the fight occurs is kept fairly simplistic; this is because the main focus should be on the combat between the player and the boss. The key here was to have the encounter be quite challenging to the player, but still on par with the game’s difficulty curve. It was important not to make the fight too hard as the player might get the idea that the rest of the game is even harder and they might lose the will to continue playing.

The first boss was designed to work in two phases, a melee attack phase and a ranged attack phase. The boss starts the fight off in the melee attack phase, so here the AI attempts to close the gap to the player and attack them in melee. The boss will continue to do this until either the player dies, or the boss’ health passes below a certain threshold. Once the bosses health is below this threshold, the AI will transition to the ranged attack phase. The boss receives a burst of speed, and jumps onto the next platform, and begins to throw daggers at the player. If the player then closes the gap, the boss will go back to melee attacking.

Here it was important to consider game balance. If the boss threw the daggers at the player too quickly they would be unavoidable and so this would be quite unfair and would leave the player feeling frustrated, however if the boss threw them too slowly they might be too easy to avoid and so there would not be enough of a challenge. For this fight and for subsequent boss fights the philosophy was that a skilled player should be able to avoid all damage, or if there is any unavoidable damage, it should not be enough to kill the player on its own.

In this first boss fight, taking some damage from the boss’s melee attacks was unavoidable. However if the player is also melee attacking, they should be able to force the boss into the ranged attack phase before it kills them. The player should then be able to use this phase to their own advantage. If they time their jumps correctly, they should be able to avoid the throwing knives completely, and if the player hasn’t taken damage after a period of time, their health starts to regenerate. So the idea of this phase was to provide a challenge to the player, but also to give them a chance to regenerate their health before attacking the boss in melee again.

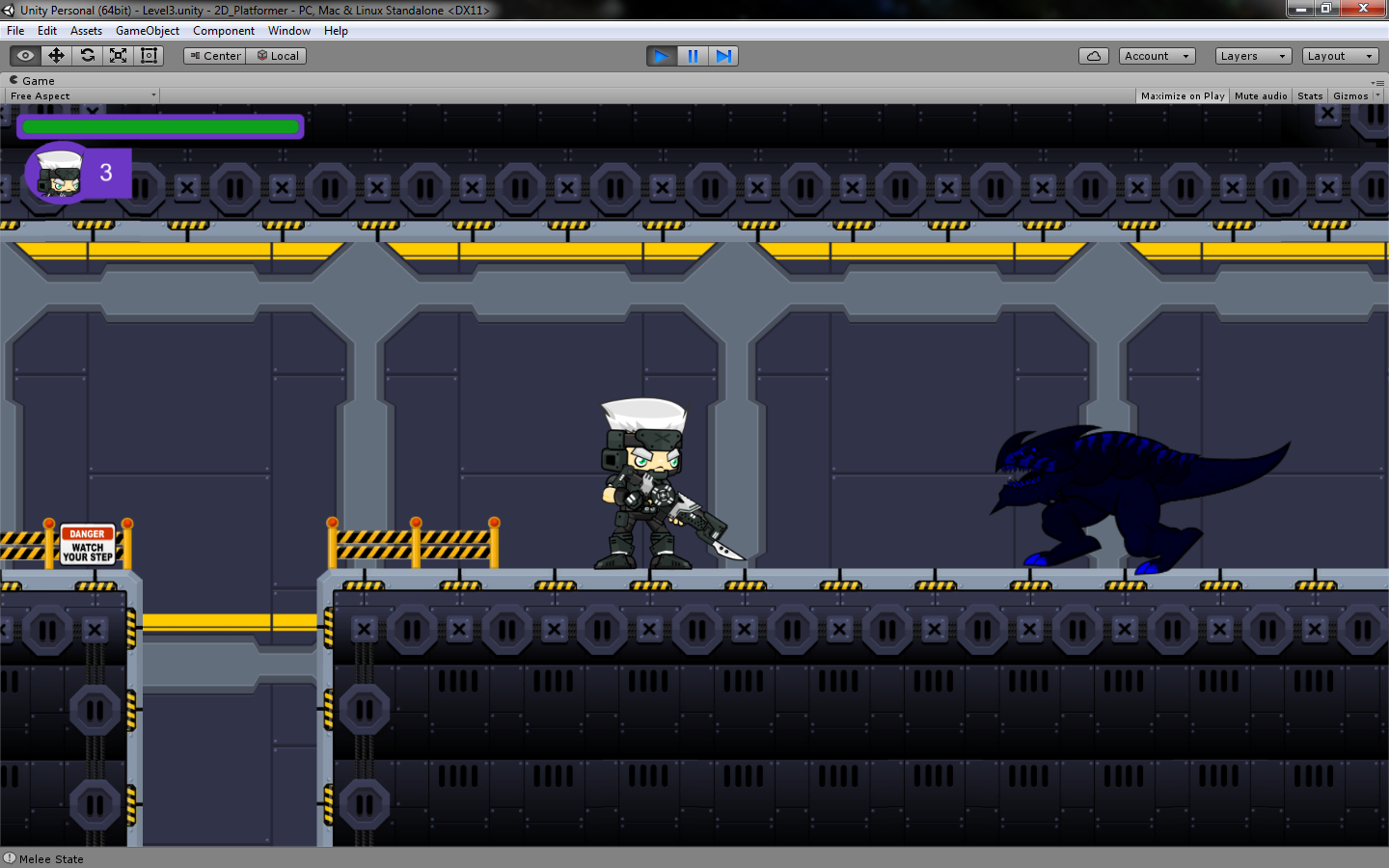


Using this tactic the player should be able to whittle the boss’ health down and win the fight.

On the second level the player is introduced to a new power-up and a new enemy AI. This new power-up boosts the player’s melee attack so that it does more damage and the player is capable of killing enemies in fewer hits. The new AI, the “Seeker” can make use of these power-ups too so it was important that the player was introduced to the power-up first so that later on when they encounter a “Seeker” they understand what is happening.

In order to introduce the player to the new power-up in a relatively safe environment, the power-up crystal is placed right at the start of the level. The player will activate it before running into range of a “Slime” AI agent. Similarly to the jump power-up, the player’s sprite will change colour to help demonstrate that they have received the power-up. Once the player attacks the AI, due to their increased damage, the “Slime” should die in one or two blows, and the player should realise that their melee attacks are stronger. There is also a dialogue hint to help explain this.

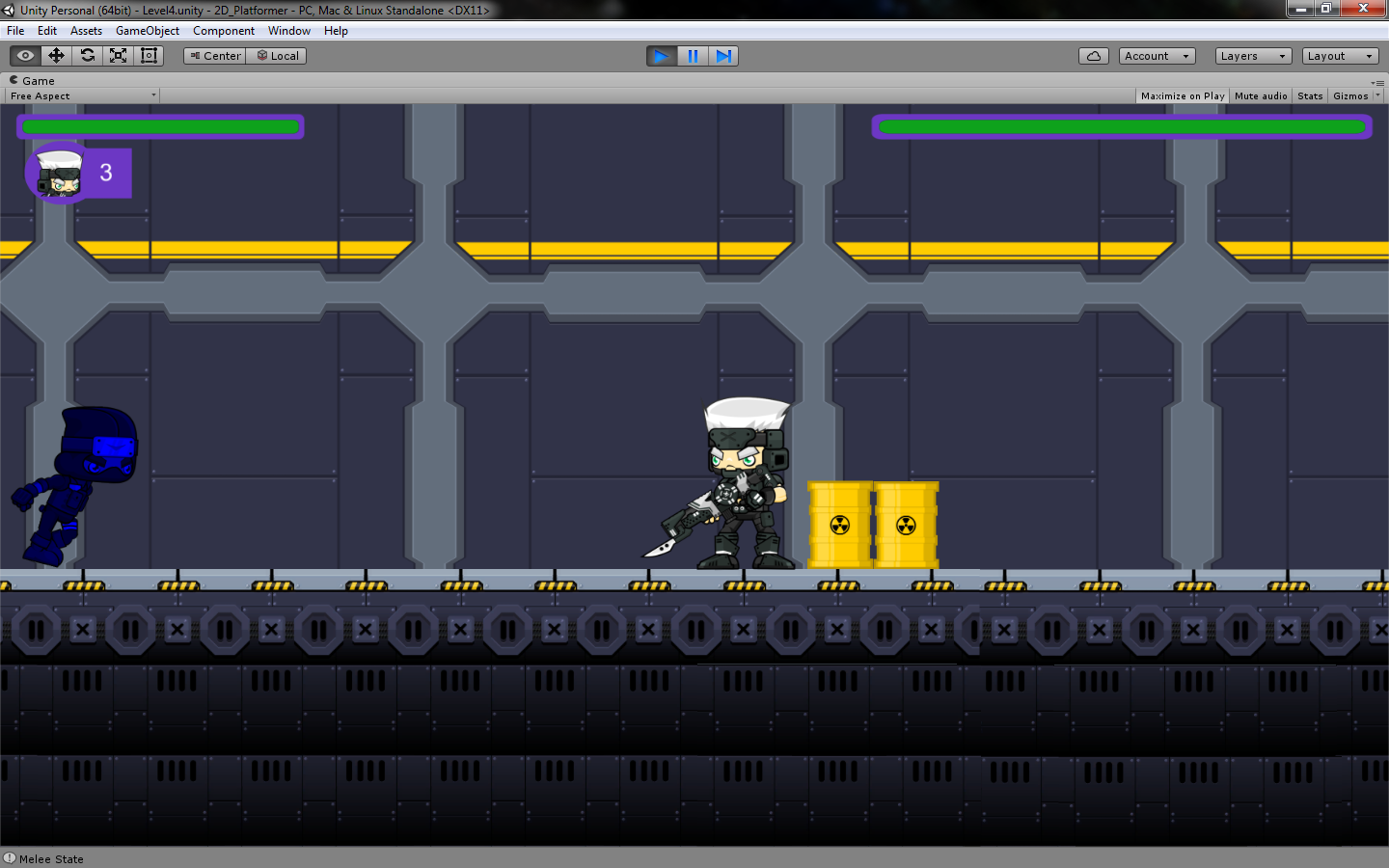
Once the player has been successfully introduced to the melee attack power-up they are ready to face the new “Seeker” AI. This AI will patrol an area similarly to the “Worm” or the “Slime”. If the “Seeker” notices that the player is nearby but out of range, it will hide in a similar fashion to the “Worm” AI. If the player comes into reach, the “Seeker” will scan the nearby area for any melee power-ups. If there is one within reach it will attempt to obtain it before attacking the player. The player of course is expected to attempt to keep the “Seeker” agent at range while it has the melee attack power-up. Like the player, the “Seeker” changes colour while it has the power-up so the player should be able to spot this fairly easily.



The rest of level two consists of an assortment of things the player has already been introduced to, along with the new melee attack power-ups and the “Seeker” AI agents.

After going through the exit door at the end of this level, the player meets the second Boss. This boss also makes use of the melee-attack power-ups in a way similar to that of the “Seeker” AI. However the Boss obviously has far more health and is capable of killing the player in a single blow if it has the power-up active. As such in order to deal with this the player is expected to either:

1. Compete with the boss for the power-ups. A new power-up manager script was created so that when either the boss or the player receives the power-up, the crystal despawns for a small amount of time and then respawns in a new location somewhere onscreen. So, if the player receives the power-up not only will they hit harder, but they will have deprived the boss of the opportunity to get a power-up until the crystal respawns again.
2. Keep the boss at range. Due to the fact that the boss is capable of killing the player in a single blow while under the effects of the power-up, this needed to be balanced in some way so the boss’s movement speed is reduced while powered-up so that the player is able to run away to avoid being killed. The level is actually large enough that the player should be able to open enough distance between themselves and the boss that they should have time to fire a few shots of their rifle at the boss while he is attempting to close the gap. So this is another possible tactic that could be used to defeat him.



The third level introduces a new AI agent to the player, the “Crawler” agent. This agent uses ranged attacks similarly to how the first boss uses them. To maintain game balance, the “Crawler” will not shoot at the player while they are on different platforms, instead it will choose to hide in a way similar to the “Worm” agent if the player is out of reach.

The rest of the third level comprises of obstacles, power-ups and AI agents that the player has been introduced to previously. After completing the third level, the player will come to the final boss encounter.

The final boss is a mixture of the first two. It will attempt to collect any melee power-ups that aren’t blocked by the player, and while it is powered-up it will attempt to user melee attacks to defeat the player. However, if the power-up is out of reach, or if the player is in the way, the Boss will ignore it and attack the player. Without the power-up, if the player is in melee range the boss will use melee attacks or if the player is attempting to keep the boss at range, the boss will throw knives instead. (This is a dynamic decision made by the boss as opposed to a scripted decision based on phases like the first boss).

To defeat this boss the player is expected to combine tactics for the previous two bosses. They should be attempting to take power-ups and engage the boss in melee, while moving to range and jumping the knives if they need to regenerate health. They are also expected to attempt to keep the boss at range while it is under the influence of the power-up.

The final boss is of course the last adversity the player faces in the game. The hopes of much of the game design was to introduce players to new mechanics in such a way that they are in a relatively safe environment to learn before then putting those mechanics to real use in a boss fight or something similar. Generally the expectations of design were to be able to communicate things to the player organically through level design and setting however on some occasions that was not possible and dialogue boxes were necessary in order to explain things to the player properly.

## Technical Design

Break down all objectives what where who when why how

Talk about power-ups keeping the game dynamic. Talk about walking into, vs activating randomly in the world.

Talk about each ai having a separate sprite in game design section

Time span limitations include unresolved bugs, and lack of unique gameplay mechanics.