MSC RESEARCH REPORT

Developing a Video Game Prototype to Test the Effectiveness of Intuitive Level Design

Abstract

Designing a game to be intuitive can allow new players to immediately engage with the gameplay, rather than becoming bored or frustrated because they don't understand it. This study delves into the theory behind what makes games intuitive and explores different methods for designers to subtly convey information to their players. An experiment is proposed to develop a game prototype that compares a level created using these intuitive design principles against one that does not. Data will be collected on both how well the player performs in each level and how much they enjoy it, to evaluate whether intuitive design concepts can improve player experience. A concept design for a 2D platformer is proposed for this prototype and the available technology for implementing it is analysed. The studies requirements, plan, ethical issues and risks are all assessed to prove it is feasible within the allotted timescale.

Stuart Paterson

Sp96@hw.ac.uk

1 Declaration of non-plagiarism

I, Stuart Paterson,
confirm that this work submitted for assessment is my own and is expressed
in my own words. Any uses made within it of the works of
other authors in any form (e.g., ideas, equations, figures, text,
tables, programs) are properly acknowledged at any point of their use.
A list of the references employed is included.
Signed: Stuart Paterson
Date: 03/04/2019

2 Contents

3 Abbreviations & Definitions

2D – Two dimensional

3D – Three dimensional

AWS – Amazon Web Services

FPS – First person shooter

GUI – Graphical User Interface

MoSCoW – Must, Should, Could, Won't requirements analysis

NES – Nintendo Entertainment System

PC – Personal Computer

Platformer – A genre of video game where the player character must travel from platform to platform, usually by jumping. They often include additional obstacles such as enemies or hazards.

UE4 – Unreal Engine 4

UI – User Interface

4 Introduction

An intuitively designed game is easy for new players to jump straight into and play without long-winded introductions. Long tutorial-sections or walls of instructional text can cause players to become bored before they even get the chance to play the game. Alternatively, players may struggle to understand a game that is not properly explained which will add unintended difficulty and may make them lose interest or grow bored. Designing a game to be intuitive means the designer can convey information and instructions to the player without them even realising it is happening. This helps the player stay engaged and immersed in the gameplay, rather than having to worry about the technicalities of the game.

Intuitive design is most important in the early stages of a game where the player is unfamiliar with the game mechanics. It can also be used later in the game when implementing new mechanics the player has yet to encounter. Many games develop intuitive designs by playtesting and observing how new players interact with the game (Fullerton, 2008). These games develop iteratively from this feedback to repeatedly improve how intuitive they are. This is effective but time consuming. This study aims to test whether applying intuitive design principles can offer immediate improvements to level design without the need for iterative development.

This paper will cover:

- The aims and objectives of this study.
- The different aspects of intuitive design for games and the benefits each of them can provide.
- Techniques for implementing intuitive design.
- Case studies of games making use of intuitive design.
- Technology options and justification of the selected technology for the game prototype.
- The requirements of this study and their MoSCoW priorities.
- The planned methodology for carrying out this study.
- The individual tasks that make up the study and a plan for their durations.
- Any professional, legal, social or ethical issues that may affect the study.
- The conceptual game design of the game prototype.

A game prototype will be developed to evaluate the effectiveness of intuitive level design. This prototype will provide a good basis for implementing intuitive design techniques. Two levels will be created and tested to offer a comparison, one using intuitive design and one without. A design concept for this game will be discussed and justification for the design choices are given.

5 Project Aim

The aim of this study is to test the effectiveness of intuitive level design techniques. Both the player's performance in the game and their enjoyment will be used to determine how effective the level design is. A game prototype will be developed as a platform for the testing. A level designed using intuitive practices will be tested against one without to establish if it offers any noticeable improvement. Data will be collected and analysed on; how well the player performs, how much they enjoy the game and how difficult they perceive the game to be.

6 Objectives

- Create a stable game prototype.
- Create a well thought out game design for the prototype. The design should provide the tools to allow testing intuitive level design with ease. The game mechanics, premise and technology will be included in this design.
- Create two levels within the game. One using intuitive design and one with a simple layout.
- To test if intuitive level design allows players to understand the game faster.
- To test if intuitive level design increases player enjoyment.
- To test if intuitive level design increases how intuitive the player finds the game.
- To test if intuitive level design decreases the player's perceived difficulty of the game.
- To analyse the data collected and evaluate whether there are any noticeable trends.
- To calculate whether any trends in the data are statistically significant.

7 Literature Review

7.1 What Intuition Means for Games

7.1.1 Defining Intuition for Games

If a game is designed to be intuitive it can allow the player to quickly understand the controls, mechanics, layout and many other features of the game. Allowing the player to quickly grasp the technicalities of the game allows them to focus on actually playing and enjoying the game. If the player has difficulty understanding the game this can cause frustration. It also may mean the player never develops an interest in the game and stops playing because they are bored. Toshio Iwai stated "*When I design something, I am trying to create it so that it is very attractive at first sight. If players touch it, it can be understood instinctively and the players can be pulled into it very strongly"* about his game Electroplankton (Centaur Communications Ltd, 2006). He acknowledges that designing a game that is immediately understood by the player allows them to dive straight into engaging with it rather than spending time puzzling over it.

What makes a game intuitive is a complex combination of many factors. Shirinian(2013) divided the intuitive "mapping" of controls to actions into 4 characteristics:

1. *Physical or spatial analogies*: The location in space gives away the function. For example, the left arrow key being used to move the player character left in the game.
2. *Cultural standards*: Preconceptions a user is likely to have based on societal norm. For example, the QWERTY keyboard is viewed as more intuitive than the Dvorak keyboard because QWERTY keyboards have been so widely used for many years (Shirinian, 2013).

3. *Biological aspects*: A parallel with a real biological relation may make certain controls intuitive.
4. *Principles of perception*: Physical arrangement can show relation. For example, grouping functionality relating to one object separately from other functionality.

Although Shirinian(2013) applied these principles purely to controls they can also be applied to gameplay mechanics.

7.1.2 Physical or spatial analogies

Physical analogies are important but are often so intuitive that they are not considered. For example, if the player triggers an explosion to their left they expect to be thrown right, due to their understanding of realistic physics. The Gestalt principles of proximity shows that people perceive objects as being grouped based on their physical location (Kubovy & Pomerantz, 2017). Take the example in Figure 1. There are 3 buttons and 3 objects. Most players would assume that the buttons correspond to the objects above them. The designer will often create physical analogies without even considering it, but if they are neglected this can result in extremely confusing features.

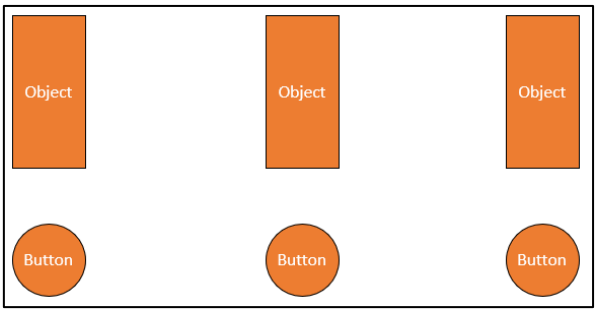


Figure 1: An example of the Gestalt principle of proximity

7.1.3 Cultural Standards

Cultural standards are important to consider in video games, especially for appealing to more experienced players. For example, it is almost universal in first person shooter games on PC that WASD is mapped to moving the player and left click is used to shoot (Shirinian, 2013). If a game were to be released with differing control schemes this would immediately be an additional obstacle for players trying the game for the first time. Adhering to these standards when possible can smooth the learning process for players who have these prior experiences.

Cultural standards are also important for the gameplay, not just the control schemes. For example, 2D platformer games such as *Super Mario Bros* (Nintendo Creative Department, 1985) have established a standard that the player should run from left to right to progress through levels (Brown, 2018). The game need not strictly follow these standards but instead the designer can make use of them. On beginning a game of *Metroid* (Nintendo R&D1 and Intelligent Systems, 1983) the player will likely run to the right as is common in other platformer games. They will then reach an impassable object and must retrace their steps to the start to explore the other direction. This immediately teaches the player they will encounter dead-ends throughout the game and will then have to explore, discover a new item and use it to overcome these previous obstacles.

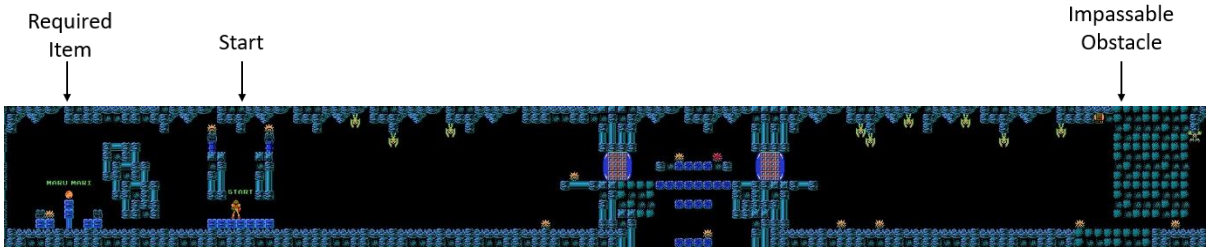


Figure 2:The first area of the map in Metroid (NES Maps, 2013)

7.1.4 Biological Aspects

The example Shirnian(2013) gives of biological aspects is volume vs frequency. When concerned with volume, most people instinctively understand that more volume means a louder sound. When it comes to frequency however, more is not necessarily so easily understood. People are less likely to associate more frequency with a higher pitched noise, despite that being the mathematical definition. Biological aspects are the least well understood of these mappings, and therefore least applicable to typical game developers.

7.1.5 Player perception

Player perception is very important for ensuring player understanding in video games. If the player cannot quickly make sense of what they're seeing they must spend time deciphering this rather than actually playing the game. The Gestalt principle of similarity states that people will assume objects which are similarly displayed will behave similarly (Kubovy & Pomerantz, 2017). Taking the example from Figure 1, if different colours are applied to each button in the same layout there is suddenly a shift in the perceived functionality. Figure 3 shows the updated image, which now gives the impression that a button interacts with the object of the corresponding colour, regardless of physical grouping.

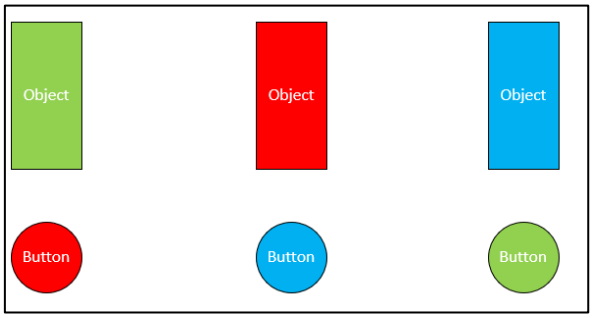


Figure 3: An example of the Gestalt principle of similarity

In *Portal* (Valve Corporation, 2007) the player must often trigger buttons to open doors. A visual path tracing from the button to the door is shown within the game to ensure the player can easily tell what each button does. This allows the player to concentrate on solving the puzzle rather than introducing needless confusion that may frustrate them. This is another example of perception being used to guide the player and an example is shown in Figure 4.

Figure 4: Screenshot from Portal showing buttons connected to door (Steam, 2007)

7.2 Intuitive Level Design

Intuitive design can be applied to several different aspects of game design. Some of these aspects are listed below.

- Controls/Input
- Game mechanics
- Level design
- User interface (Both gameplay overlays and menus)

This study specifically focuses on intuitive level design. Many of the practices explored later in this literature review are also applicable to other areas of game design. However, this study only aims to test their effect when applied to level design. The designer can use intuitive level design to convey information to the player and smooth their playing experience. This is particularly important at the beginning of games, or when introducing new mechanics (Desurvire & Wilberg, 2008).

The way a level is laid out can teach the player about the game without them realising. Take the first level of *Super Mario Bros* (Nintendo Creative Department, 1985) as an example. When the player first encounters the mushroom power up, they will be unsure whether this will be positive or negative. The designer ensured due to the confined space and high movement speed of the mushroom that even if they player tries to avoid it, they will not be able to (Anthropy & Clark, 2014). Being forced to grab the mushroom the player will then see that it is a power-up and the game has taught this without any explicit instructions.

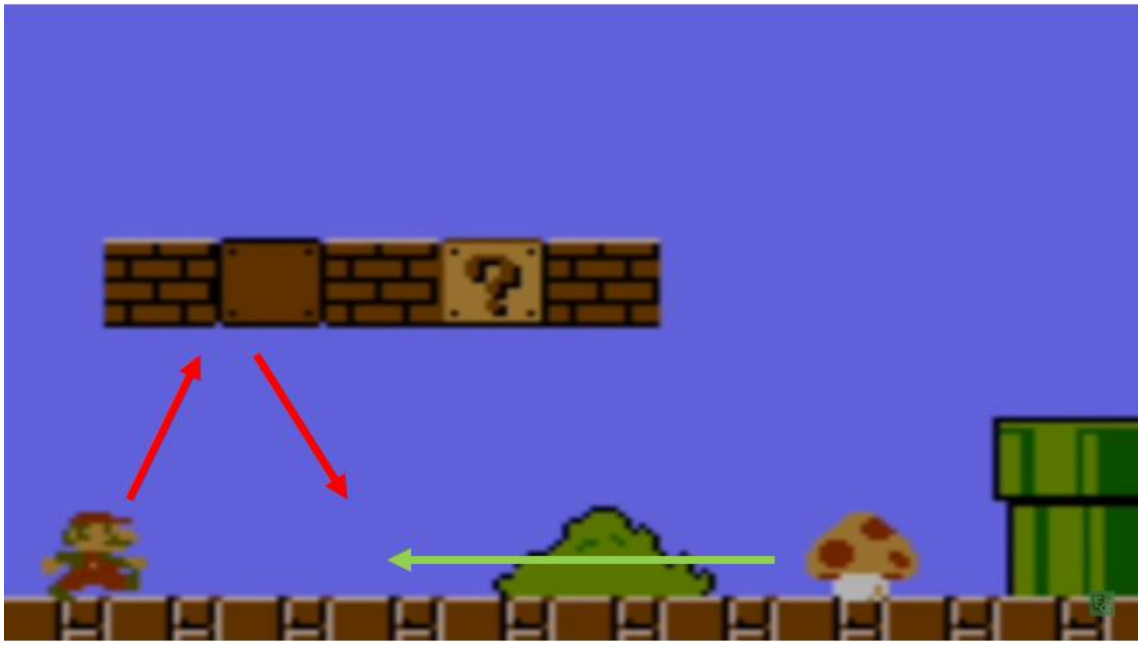


Figure 5: The beginning of the first level of Super Mario Bros. The blocks above ensure the player cannot jump over the mushroom and will likely collide with it. Image taken from (Iyer, 2017) and modified

Super Mario 3D Land (Nintendo EAD Tokyo, 2011) uses intuitive design to introduce a new core mechanic in each level and uses a 4-step system to do so (Nutt, 2012).

1. Introduce the mechanic in a simple situation.
2. Use the mechanic to complete a more complex challenge.
3. Throw a very different challenge at the player that requires them to utilise the mechanic in a different way.
4. Challenge the player to use everything they have learned in combination and demonstrate their skill.

This shows how intuitive level design can be important throughout the entire game rather than just the early stages. None of the levels in *Super Mario 3D Land* feature lengthy textual descriptions explaining the new mechanic. The simple introduction of the mechanic teaches the player the basics and then they build upon this in the more complex challenges.

7.3 Benefits of Intuitive Design in Games

7.3.1 Initial Player Engagement

Intuitive design eases the player into understanding the technicalities of the game like the controls and game mechanics. In many games these are addressed with text-based tutorials or dedicated tutorial levels that instruct the player at each step. This type of tutorial is often slow and boring, running the risk of disrespecting the player's intelligence by explaining every simple detail (Anthropy & Clark, 2014). When a player decides they wish to try a game, they have been enticed in by snippets of the gameplay. On starting the game they want to jump right in, not sit and read a wall of text or move through a slow and unchallenging tutorial level.

7.3.2 Challenge

Challenge is a very important part of player engagement and tutorial levels often offer no-risk to the player which can lead to boredom (Hoffman & Nadelson, 2010). Hoffman & Nadelson(2010) discovered that balancing the degree of challenge is very important for keeping players interested, as both too easy or too difficult a game resulted in a negative response from players. Figure 6 shows how challenge must be based on player ability to keep the player interested (Chen, 2007). This is not only true for the first levels, it also means challenge should increase as the player's ability increases when the game progresses. Intuitive design can play a key part in making the game easy to understand, which means the designer can control player challenge more easily. If the player understands the tools they can use in the game then the challenge can come from the game's content rather than "unintended difficulty" created by the player not understanding the game (Desurvire & Wixon, 2013).

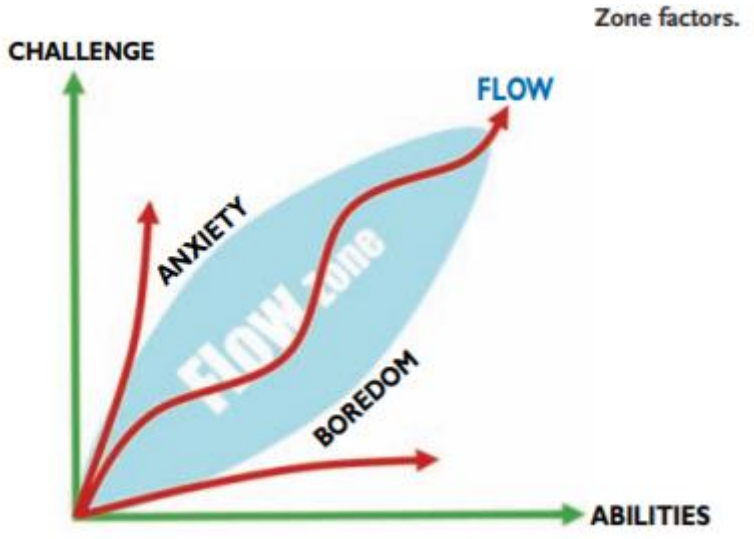


Figure 6: A graph depicting how challenge and player ability must be balanced to maintain flow (Chen, 2007)

7.3.3 Flow

Intuitive design increases player engagement by balancing challenge but it also encourages players to enter a flow state. Csikszentmihalyi (2014) describes flow as a state of "deep concentration" where the person is only responding to a "limited set of stimuli". A common sign of flow is the person losing track of time due to their complete absorption in the task (Finneran & Zhang, 2003). Player engagement and immersion are essential parts of achieving flow for video games and intuitive

design can help both. Reading tutorial text or receiving instructions can break the flow of the game and interrupt the player's immersion. These types of instruction remove the player from the game world and remind them they're playing a game. Reducing the player engagement like this is bad for achieving flow whereas making the game easy to understand intuitively will allow the player to be drawn in and remain immersed. It has been shown people who reach this flow state show enhanced learning so players in flow will understand the game faster (Webster, et al., 1993) (Kiili, 2005).

7.3.4 Accessibility

Intuitive design is not only good for engagement but also accessibility. It provides benefits for both hardcore gamers and more casual players. Intuitive design can allow hardcore gamers to feel immediately familiar with the game while also making the game easier to understand for new players. Allowing less-experienced players to understand the game faster makes the game accessible to a larger audience by allowing the player to feel competent immediately, regardless of skill (Shirinian, 2013). Casual players are more likely to quit the game if they struggle to understand it, so intuitive design can ensure better player retention and a larger player base (Desurvire & Wilberg, 2008).

Removing textual instructions is not only advantageous for player immersion, it also transcends language barriers. Without as much text essential to the game more players from different cultures across the world will be able to access the game. The first level of *Shovel Knight* features no textual instructions but still teaches the player how the main mechanics work. Figure 7 shows the screen where the player must use the downwards aerial attack for the first time. The player cannot progress without using the ability, and the breakable blocks that have been previously introduced make it clear the player must somehow attack downwards. This combined with the fact that players lock into a down attack if they press down at any point in the air means the player will quickly discover this ability even if they are not aware it exists (Grow, 2014). By increasing accessibility, intuitive design can allow a game to reach larger markets and appeal to a wider player demographic.

Figure 7: The first level of shovel knight requires the player to use the downwards aerial attack to progress (Grow, 2014)

7.4 Implementing Intuitive Game Design

Intuitive game design offers a lot of benefits but implementing it comes with challenges. Different techniques can be used to make the game intuitive and to teach the player about the game.

7.4.1 Experiential Learning

Experiential learning means learning from past-experiences (Kolb, 2014). Experiential learning is useful in games as the designer can control what the player experiences. Kolb (2014) theorises that the optimal learning pattern is when people; experience something, reflect on their experience then form conclusions based on those experiences.

Experiential learning can be very useful for introducing game mechanics and allowing the player to understand the overall structure of the game. *Super Mario Odyssey* (Nintendo EPD, 2017) uses experiential learning in each level to introduce new mechanics in a way that is easy for the player to understand. Each level begins by introducing the mechanic and its basic functionality in a very simple situation. This lets the player gets some practice using the mechanic and ensures they understand it's basic function. Then the difficulty is ramped up as the stage progresses and the mechanic is often used in multiple different ways (Brown, 2017). If immediately after acquiring the new mechanic, the player had to use it in several different ways to solve a complex puzzle this would be extremely difficult. By controlling the player's previous experience with the mechanic, Nintendo ensure that the player is never overwhelmed, and by ramping the difficulty up they keep the player engaged.

Figure 8: The uproot mechanic in Super Mario Odyssey is initially used for climbing simple steps (left). Later in the level it must be used to attack the underside of the level boss(right). Images from (Brown, 2017).

7.4.2 Productive Negativity

Productive negativity is a type of experiential learning and can be thought of as learning from mistakes (Gauthier & Jenkinson, 2018). By allowing the player to fail, the designer can allow the player to draw their own conclusions from this failure.

Controlling the impact of failure is essential when using productive negativity. If failure is too significant then the player will avoid it too much. If failure is not significant enough then the game may struggle to engage the player. As discussed earlier, failure must remain significant enough to balance the difficulty of the game. If the game is too easy players may lose interest (Hoffman & Nadelson, 2010) so steps must be taken to prevent this when encouraging productive negativity.

Super Meat Boy (Team Meat, 2010) is an example of a game that allows the player to use productive negativity and maintains suitable challenge. It achieves this by keeping levels short and increasing the number of hazards as the game progresses. Early levels in super meat boy are extremely short and only feature a small number of hazards for the player to pass. In these early levels if the player fails, returning to the start of the level is not much of a setback. Later in the game the levels feature many more hazards and returning to the beginning becomes a much larger penalty. An increased number of hazards means more challenges the player must overcome successively to reach the end of the level, greatly increasing the difficulty. Reducing the significance of failure in the early levels ensures the player won't become frustrated when failing to new hazards. Instead they learn about

the game's mechanics through these failures, which provides them the skillset they need to complete the more complex challenges later in the game.

Figure 9: A comparison of two levels in Super Meat Boy. (Left) A simple level from early in the game to introduce hazards. (Right) A more complex level later in the game (Valve, 2010)

Portal (Valve Corporation, 2007) also has an excellent example of productive negativity in a different manner to *Super Meat Boy*. There is a mechanic known as the "fling" in portal where the player falls through a portal and then comes out of another, retaining their forward momentum (RPS, 2013). When the player enters the first room where the fling mechanic is required they will not have any prior experience of it. The level guides the player to the solution by limiting where the player can place portals and keeping the layout very simple. The player is immediately confronted with a diving board styled platform over a pit with a surface they can place portals on at the base. The only other area they can place portals in the room is a vertical wall above the door they entered through. A single panel protrudes from this wall, hinting the player should use it. The use of productive negativity comes into play if the player jumps into the pit. It is likely many players will either jump or fall into the pit when entering the room; either mistakenly or while attempting to explore. To get out of the pit the player must place a portal on the floor and on the only other surface they can see outside the pit, the protruding wall panel. When passing through this portal the player will be slightly flung forwards and see the exit door in front of them. The player can then conclude they could use the same technique to reach the other side if they had a little more speed. To add to this, the player's portals are now in the correct position for them to do so. The player can then jump from the platform into their existing portal and reach the goal, as shown in Figure 10.

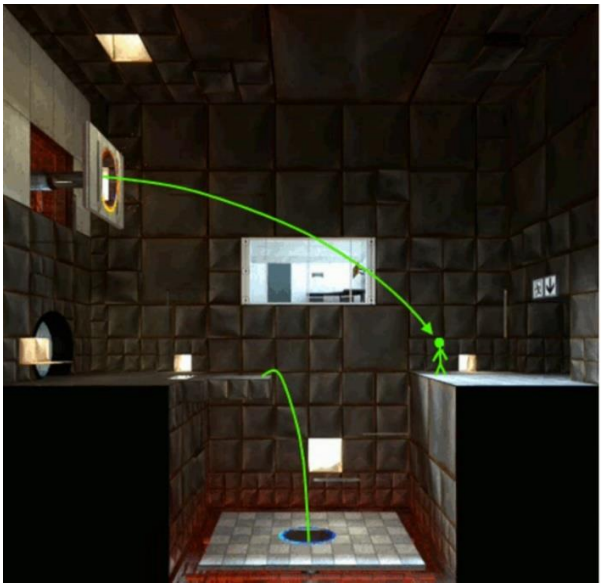


Figure 10: The first level the "fling" mechanic is required in portal (RPS, 2013)

Super Meat Boy takes away the significance of failing in early levels to allow the player to make mistakes and learn from them without getting frustrated. In contrast *Portal* cleverly lures the player into failing and in the process of returning from this failure shows them exactly how to complete the puzzle. Both techniques teach the player through failure, but in very different manners.

7.4.3 Reinforcement Learning

Reinforcement learning involves letting the player experiment to determine which actions offer a reward or penalty (Abe, et al., 2011). The player can view the outcome of actions they take then adjust their understanding of the game based on the result. There are two types of reinforcement learning; model-free and model based.

Model-free reinforcement learning involves the player only drawing conclusions from the results of their own actions. Model-based reinforcement learning extends this to include the player drawing conclusions based on observing the environment. This allows them to deduct the hypothetical result of an action without taking it (Abe, et al., 2011). Using level 1-1 of *Super Mario Bros* (Nintendo Creative Department, 1985) as an example, at the start of the level when the player first encounters an enemy they may run into it and fail. They can then draw the conclusion that that enemy is dangerous which is model-free reinforcement learning. When the player encounters the first mushroom from a question-mark block it appears on a platform above the player. It then travels forwards, falls and bounces off the green pipe. After bouncing off the pipe it moves towards the player. The player witnesses the mushroom change direction when colliding with the pipe and can draw the conclusion that any moving object will change direction when they collide with other objects (Iyer, 2017). This is true for the enemies too and so the designer has used model-based reinforcement learning to teach the player how moving objects behave.

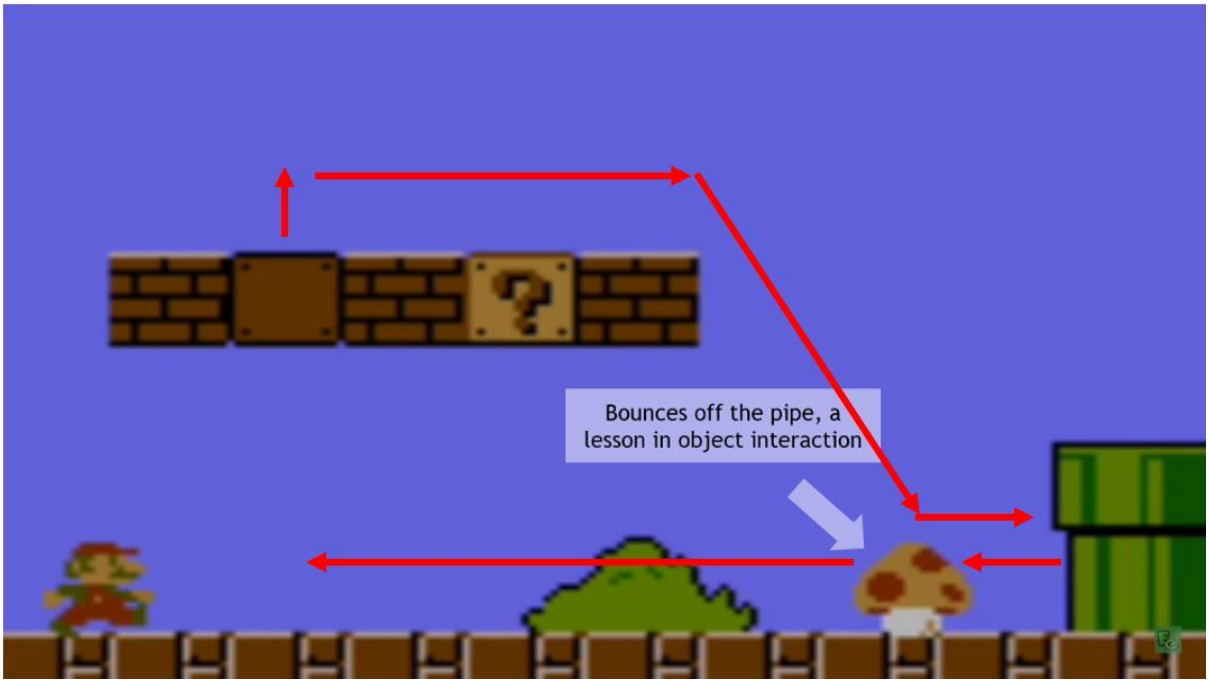


Figure 11: The first mushroom power up in Super Mario Bros demonstrates how moving objects interact with obstructions (Iyer, 2017)

7.4.4 Transformative Learning

Transformative learning involves taking previous knowledge and then adjusting it to reveal something not previously perceived (Podleschny, 2012). It takes prior experience and not only builds upon it but changes the player's understanding of the game. As discussed earlier, *Super Mario 3D Land* (Nintendo EAD Tokyo, 2011) uses a 4-step level design pattern which involves introducing a mechanic then twisting it to surprise the player (Nutt, 2012). Transformative learning is used for this

to show the player the mechanic can be perceived differently. For example, in World 1-4 the player must use tilting platforms to travel along rails through the level (Nintendo EAD Tokyo, 2011). At the start of the level the mechanic is that if a player is standing on one side of a platform it moves in that direction. Part way through the level it is revealed to the player that the platforms can in fact move up and down, adding a new dimension to the gameplay. The player must change their existing understanding of the platforms being constrained to their rails and utilise rising water spouts and falling to complete the remaining puzzles. After their first encounter with vertically moving from one set of rails to another the player's perception of the mechanic is permanently changed to include the additional movement dimension.

Figure 12: Tilting platform mechanic in Super Mario 3D Land. Initially the player is constrained to a set of rails (left) but later they realise they can move between rails by falling or riding water spouts (right). Images taken from (JapanCommercials4U2, 2012)

7.4.5 Iterative Learning

Iterative learning (also called recursive learning (Mitgutsch & Schirra, 2012)) is another useful tool for intuitive game design. Iterative learning involves introducing mechanics to the player a small amount at a time (Fischer, 2016). A typical iterative learning design uses two repeating steps; a new mechanic is introduced that builds on the player's previous knowledge (if any) and then the player gets to use it and learn how it works. Portal (Valve Corporation, 2007) makes an obvious use of iterative learning to introduce the Portal Gun and the steps are outlined below.

1. Player must exit first room through static portals that appear.
2. Player must solve puzzle using moving portals controlled by game.
3. Player must solve puzzle using a portal gun capable of shooting one colour of portal. The other portal is already placed in level.
4. Player gains ability to shoot both colours of portal and must solve the remaining puzzles using this.

Recognising that moving through space using the portals may be difficult for the player to understand, Valve cleverly introduced the mechanic in iterations, with each building upon the last and adding new depth.

7.4.6 Antepieces

An antepiece is a simple task that precedes a more complex challenge to give the player hints on how to solve it (tvtropes, 2019). The antepiece itself offers no challenge but by completing it the player gains knowledge they can use later. This can be used in game design to make difficult challenges easier to approach. In *Portal* (Valve Corporation, 2007) the player is forced to incinerate the "weighted companion cube" before approaching the final boss GLaDOS. When the player fights the final boss they must use an identical incinerator to destroy parts of GLaDOS, a mechanic they have just learned to use through the antepiece (RPS, 2013).

Figure 13: The player incinerating the weighted companion cube in Portal. This teaches the player how to use the incinerator which they will require in the upcoming boss battle. Image taken from (astral, 2014)

7.4.7 Designing around a core mechanic

Designing a game around a single core mechanic can help make it more intuitive to the player. In this model the core mechanic is the central feature of the game and other mechanics and the narrative are all built around it (Kim, 2012). *Super Mario Odyssey* (Nintendo EPD, 2017) is an excellent example of this structure of game design. In *Super Mario Odyssey* the core mechanic is the hat throw (Nintendo EPD, 2017). Mario's hat can be used to jump, attack enemies, possess creatures, collect coins and many more actions. To add to this, possessing creatures is an extremely detailed mechanic. There are a huge number of different creatures Mario can possess and each one has different abilities that are used to solve different challenges. The many different mechanics in *Super Mario Odyssey* would confuse the player if Mario was always capable of doing them. The game's design choice to make Mario interact with everything through his hat means if the player encounters an object or situation where they don't know what to do, they know that they must use the hat and this helps them to understand the situation quicker (Brown, 2017).

Although the hat in *Super Mario Odyssey* is a single core mechanic, it still holds a large amount of complexity due to the many different interactions. Designing around a core mechanic does not mean the mechanic must have a huge number of different possibilities and interactions. The game *VVVVV* (Cavanagh, 2010) uses the core mechanic of being able to flip gravity. The game's creator describes it as *"a platform game all about exploring one simple mechanical idea"* (Cavanagh, 2010). The game prides itself on using a simple single mechanic and introducing complexity through level design. The wide range of possibilities result from combining the games limited number of mechanics together to result in differing challenges in each level (Kayali & Schuh, 2011).

7.4.8 Signifiers

Signifiers are a technique used to highlight objects of importance to the player. They can show the player that something is interactable and encourage them to investigate further. For example, the question mark blocks in *Super Mario Bros* (Nintendo Creative Department, 1985) use signifiers to make it obvious to the player that these blocks differ from typical bricks. The combination of the question mark symbol and the flashing animation make these blocks stand out compared to the other static elements in the scene (Iyer, 2017).

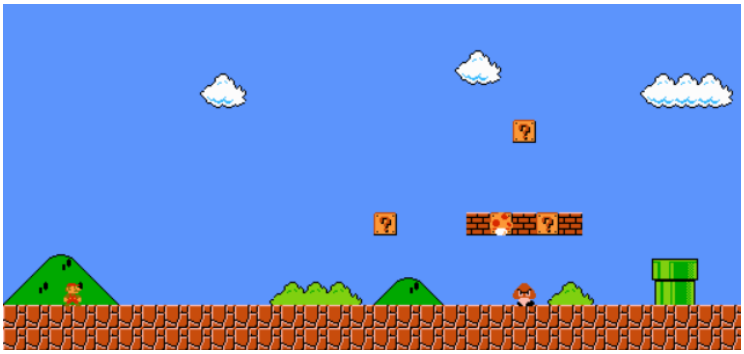


Figure 14: The question-mark blocks stand out in the initial scene of Super Mario Bros. Their flashing animation and mysterious question-mark symbol pique the player's interest. Image from (NES Maps, 2008)

7.5 Integrating Intuitive Design

The techniques discussed previously are excellent at teaching the player how to play the game without explicitly instructing them. However, these techniques must be applied in ways that integrate them into the game naturally. If they are not, then rather than feeling intuitive to the player, the player will see that the game is leading them. This may break their immersion or reduce their satisfaction when they make progress in the game. Therefore, methods must be devised for integrating intuitive design into the gameplay seamlessly.

Intuitive design should give the player hints and point them in the correct direction for understanding the game. The goal is to give the player the necessary tools they require to work it out for themselves. Spelling it out too obviously will decrease the challenge of the game which will mean players might start to lose interest (Fischer, 2016). *Half-Life 2* (Valve Corporation, 2004) uses an excellent example of integrating intuitive design into the narrative naturally so that the player will not realise the designer is guiding their hand. In *Half-Life 2* the player possesses a gravity gun which can be used to move objects around. The gravity gun can be used to fire saw blades at enemies to defeat them, and Valve used a very clever technique to introduce this mechanic.

The first time the player encounters saw blades they are embedded in a doorway and block the player from passing. When the player uses the gravity gun to remove one so that they can pass, this triggers a zombie to stumble around the corner. The player, who is already aiming at the zombie due to its placement, will most likely hit the fire button and shoot the saw blade through the enemy (Brown, 2015). This is a much more eloquent solution than say, locking the player in a room with a zombie and a saw blade and not letting them progress until they figure it out. This type of seamless design makes the game truly intuitive as the player create conclusions on their own and doesn't realise the designer is leading them.

Figure 15: The introduction to the saw blade mechanic in Half-Life 2. Images taken from (Brown, 2015)

7.6 Selection of Technology

7.6.1 What is a Game Engine?

Intuitive design has been discussed in depth, but in order to implement these design techniques a prototype is required. There are many technological options when developing a game, and this section describes the best methods for this study.

The game could either be created entirely from scratch, or by using a game engine. There are many game engines and they offer a wide range of different functionality. However, they all have the same core purpose, to provide commonly used functionality so that the developer doesn't have to create it themselves. This usually includes support for; physics, 2D and/or 3D graphics, animation, sound, GUI, networking and many other features. Using a game engine saves a lot of time compared to programming the game from the ground up (Gregory, 1970).

7.6.2 Selecting a Target Platform

A lot of different platforms support video games and using an engine can make many of them easy to develop for. Table 1 shows a comparison of the three main platforms for video games. Players typically use one platform to play. Therefore releasing a game on multiple platforms will allow it to reach the largest market. Porting the game to more than one platform requires additional development time so this should only be done when the benefit outweighs the cost.

Table 1: A comparison of different video game platforms

Platform	Advantages	Disadvantages
PC (Windows, Linux, Mac)	• Allows keyboard or controller for input. • Can be portable if run on a laptop. • Easy to save data (e.g stats for evaluation). • Easy to develop for. • Can be most computationally powerful platform with correct hardware.	• Performance can vary a lot due to wide range of hardware specifications. • Separate builds needed for different operating systems. • Compatibility depends on operating system version.
Mobile (Android, IOS)	• Allows touch controls. • Portable for testing. • Most people have mobile phones.	• Only touch controls. • Computing power may be a limitation.
Console (Xbox, Playstation, Switch)	• A lot of players only use consoles. • Platform is dedicated to games.	• Difficult to develop for. • Not easily portable. • Difficult to save statistics on. • One type of controller per platform.

Developing for PC is the simplest and most versatile platform, but care must be taken to ensure compatibility. The huge range in performance specifications, operating systems and software versions means usually games will need to be designed to be adaptable to ensure they run. Mobile offers a completely different experience with portability and touch controls. Mobile games typically appeal to a more casual audience than PC or consoles. Consoles can be difficult to develop for, but

the advantage is that a large audience use them exclusively to play games. Therefore, developing for console makes the game available to a large number of potential players.

7.6.3 Selecting a Game Engine

It can be very beneficial to use a game engine to produce a game, saving time and effort. Some game engines are very high level and may even require no programming, such as Scratch. At the other end of the spectrum, Unreal Engine 4 supports coding in C++ and the engine source code can be accessed by the developer to tweak the behaviour to their exact specification. The engines that offer the most flexibility and complex features often have the steepest learning curve whereas the higher-level options can be easier for beginners.

A game engine usually has its own strengths and weaknesses meaning different engines can suit different games. To give an extreme example, Ren'Py is an engine which can be used to easily create visual novels and only requires some basic python programming (Ren'Py, 2019). However, it cannot be used to produce any other type of game. The more common engines like Unity and UE4 can be used to make a wider variety of game genres. Table 2 shows the advantages and disadvantages of some of the most popular game engines.

Table 2: A comparison of game engines (Christopoulou & Xinogalos, 2017) (Burrows, et al., 2019) (GameDesigning.org, 2019)

Engine	Advantages	Disadvantages
Unity	• Exports to a lot of platforms. • Free to use. (Only charged when game makes over a $100k a year) (Unity, 2019). • Good documentation. • Large number of features. • Suited to small teams. • 2D and 3D.	• Performance may be an issue (higher overhead cpu cost and less space for optimisation). • Source code isn't open. • Inferior graphics to UE4.
Unreal Engine 4 (UE4)	• Excellent audio-visuals. • Free to use. • Open source. • 2D and 3D. • Non-programmers can use blueprints to edit code. • Programmers use C++ so easier to optimise code.	• Suited to larger teams with individuals for each role. • Engine is complex and takes time to learn. • Documentation is good but inferior to Unity.
Lumberyard	• Amazon web services (AWS) integration. • Good for cloud AI and multiplayer. • Free to use.	• Web services cost money. • Less features than UE4 and Unity.
GoDot	• Open source and free to use. No royalties. • 2D and 3D.	• Less features than UE4 and Unity. • Documentation inferior to UE4 and Unity.
GameMaker	• Doesn't require programming knowledge so more accessible.	• Imposes a lot more limitations than programming engines. • Free version doesn't include most features.

The two most powerful engines with the most comprehensive list of features available are Unity and UE4. These two are distinguished by their complexity and ease of use. Unity is excellent for rapid prototyping and developing with small teams. There is excellent documentation and the engine is easy to understand and use. UE4 is more difficult to use than Unity but provides additional development power for this cost. UE4 is best suited to a game that requires the ability to tweak the engine code and have full control of the technology behind the game to achieve the required results. UE4 works best with larger teams that consist of individuals specialised in different areas (Christopoulou & Xinogalos, 2017). This is because the additional complexity requires more effort and time. To summarise, Unity is well suited to teams where manpower and time are limited, and the game specifications are within the functionality that Unity provides. UE4 is suited to larger teams and projects that require modification to the engine tools in order to achieve the precise desired results.

Rather than attempting to compete with Unity and UE4, most other engines try to offer a more specialised niche. For example, Amazon Lumberyard's main marketing point is the AWS integration and GoDot is like Unity but open source. These engines can save even more development time than a generic engine like Unity when making games that require their niche role. However, this specialisation makes these engines poorly suited to use for other types of games.

7.6.4 Technology for this Study

An engine should be used for the prototype required in this study due to the timescale of the project. The tools provided by an engine will be more than adequate provided the right engine is picked, meaning there is little to no benefit to creating an engine from scratch. A lot of time will be saved not developing the engine which allows more time for developing the game and carrying out the testing.

Based on the findings in Table 2, Unity is the most appropriate engine for the prototype. The application will be developed by a single developer which means Unity is more appropriate than UE4. The lack of access to the engine's source code will not be an issue for this prototype because the supplied features in Unity will be adequate. The game does not intend to include any revolutionary features and the default Unity physics system will be suitable. Although UE4 offers better audio-visuals than Unity these are also not important for this project, especially for a 2D game. If the audio-visuals are good enough to ensure the player can understand what is happening in the game, then they are sufficient. Unity is easily capable of providing the necessary audio-visuals.

Unity is free for any game making less than $100000 per year (Unity, 2019). The prototype in this study will not be published commercially so Unity can be used free of charge. Unity will require a greater performance overhead than making the game without an engine, but this should not be an issue. The game will be run on one machine rather than being distributed to many users so the computing power will be known beforehand. During development the performance can be monitored and tweaked to ensure the game runs adequately for testing.

Windows is the ideal platform to build the prototype for. This will make development of the game simpler and ensure frequent testing is easy. The issues with compatibility and performance will be circumvented because the game will only be required to run on one machine. The game will be built to run on the laptop that will be used for collecting test data, meaning any issues can be discovered during development and fixed before testing. Running the game on a laptop allows it to be portable which makes travelling to test participants to collect data easy. Windows allows a range of input devices to be used to control the game. The game will probably be designed for a game controller, but keyboard and mouse could also be opted for meaning the choice of input device is flexible.

7.7 Review Conclusions

This review has explored what intuitive design means for video games. Allowing the player to immediately understand a game by providing the tools for them to work it out themselves can provide many benefits. The main goal is to ease new players into the mechanics of the game and allow them to focus on the gameplay rather than the technicalities. In a similar vein, intuitive design can be used later in the game when introducing new mechanics. Several intuitive design techniques have been explained and a summary is shown in Table 3.

Table 3: A summary of the intuitive design techniques explained

Design Technique	**Summary**
Experiential learning	Shape the player's experiences to teach them about the game.
Productive negativity	Allow the player to fail and learn from their failures. Control the impact of failure to prevent frustration or boredom.
Reinforcement learning	Let the player experiment in the game and provide positive or negative feedback to their actions.
Transformative learning	Take the player's existing understanding and re-shape it to include additional information.
Iterative learning	Introduce features in small pieces at a time, building off previous knowledge.
Antepieces	Let the player complete trivial tasks that teach them the basis for solving a more difficult task.
Designing around a core mechanic	The game is based upon a single mechanic. The player then knows when they encounter a new situation that they must use this mechanic somehow.
Signifiers	Use visual or audio cues to give the player hints about important objects.

The importance of seamlessly integrating intuitive design into the game was explained. For the design to be truly intuitive the player must not see the designer's intent and should instead think they are working out the game themselves. The discussed techniques can be applied to many aspects of game design, but this study will be focusing on level design. To test the effectiveness of these techniques a game prototype will be developed. Technological options were explored for both the platform and game engine. Windows was selected as the target platform due to the simplicity of development and flexibility of input devices it provides. Unity will be used to develop the game prototype. Using a game engine will save time developing the prototype and will free up this time to perform testing and collect more data.

8 Requirements Analysis

The main requirements of the study are shown in Table 4. These are the minimum possible requirements to be fulfilled for the study to be successful. The requirements have been defined from the project aim and objectives. These main requirements have been split into further sub-requirements to outline the full planned scope of the study. Tables 5-9 show the sub-requirements for each main requirement. Their MoSCoW prioritisation is displayed, showing how important each requirement is for ensuring the study is successful.

MoSCoW prioritisation has been used to account for not having enough time to complete every requirement fully. The priorities assigned will dictate which requirements are most important and should be completed first to ensure the study is successful. “Must” priority refers to requirements which are critical to the study being completed at all. Without these the study would be incomplete and it would be impossible to draw any conclusions from it. “Should” requirements are important and need to be fulfilled for the study to be conducted properly. Without “should” requirements it may prove difficult to complete the study, or data collected may be invalid. “Could” requirements are optional and may provide some added value if completed. “Won’t” requirements are features that are irrelevant or unnecessary for completing the study and should never take priority over other categories of requirements.

Table 4: The main requirements of the study

Requirement	Prioritisation	Explanation
Have a playable prototype to play	Must	There must be a playable game prototype that is capable of supporting the study of intuitive level design
Have at least 2 level designs to compare	Must	Require comparison to show whether altering level design has an affect.
Have a suitable number of playtesters to collect data	Must	More playtesters means the data is more likely to be statistically valid. An insufficient amount means no conclusions can be drawn from the data.
Game is thoughtfully designed	Should	Thoughtful design of the game will enhance experience and be more representative of a full game.
Game collects data for analysis	Must	Data is required for analysis. Without gathered data no conclusions could be drawn from the study.

Table 5: Sub-requirements for the main requirement "Have a playable prototype to play"

Requirement	Prioritisation	Explanation
Have a playable prototype to play	Must	There must be a playable game prototype that is capable of supporting the study of intuitive level design
Prototype uses game engine to save time	Must	Using a game engine rather than developing from scratch will greatly reduce development time
Prototype uses Unity as the game engine	Could	Unity is a game engine well suited to small scale projects produced by small teams or solo developers
Basic sound effects to accompany mechanics	Should	Required to give player feedback on gameplay and aid their understanding
Basic animations to accompany player mechanics	Should	Required to give player feedback on gameplay and aid their understanding
Have a complete set of sound effects	Could	Enhance player engagement and immersion
Have a complete set of animations	Could	Enhance player engagement and immersion
Be available on major distribution platforms	Won't	Players will need to be interviewed while playing. Must use same machine for all tests
Be multiplayer	Won't	Only testing on one individual at a time
Be monetised	Won't	Not necessary

Table 6: Sub-requirements for the main requirement "Have at least 2 level designs to compare"

Requirement	Prioritisation	Explanation
Have at least 2 level designs to compare	Must	Require comparison to show whether altering level design has an affect
Have a control level that doesn't use intuitive design	Must	This level will be used as baseline
Have an intuitively designed level	Must	This level will attempt to use intuitive design techniques to improve player learning and performance
2 Levels to be compared have similar length and difficulty	Should	Will ensure results are fairer and data is valid
Have multiple consecutive levels for each test	Won't	Only one level for each test is required

Table 7: Sub-requirements for the main requirement "Have a suitable number of playtesters to collect data"

Requirement	Prioritisation	Explanation
Have a suitable number of playtesters to collect data	Must	More playtesters means the data is more likely to be statistically valid
Playtesters should be from target audience	Should	This will ensure they have similar experience and interest in the game
Same number of playtesters for each level	Must	Same amount of data for each level means fair comparison
Candidates are interviewed on their experience	Must	Must establish candidate experience level is within a given range so that data can be grouped based on candidate experience.

Table 8: Sub-requirements for the main requirement "Game is thoughtfully designed"

Requirement	Prioritisation	Explanation
Game is thoughtfully designed	Should	Thoughtful design of the game will enhance experience and be more representative of a full game
Game has a basic premise which is implemented in game	Should	Premise can make game more engaging and help players get immersed.
Be designed around a core mechanic	Could	Designing around a core mechanic could help make the main mechanics easy to understand
Use design patterns to structure software	Should	Design patterns lead to better structures and more efficient code (Nystrom, 2014).
Integrate feedback early on to improve game design	Could	Can help address issues designer may be missing (Fullerton, 2008).

Table 9: Sub-requirements for the main requirement "Game collects data for analysis"

Requirement	Prioritisation	Explanation
Game collects data for analysis	Must	Data is required for analysis. Without gathered data no conclusions could be drawn from the study.
Game stores data about playtest for analysis later	Should	Would be more efficient and accurate than inspecting the test and manually recording data.
Have fully automated exporting and analysis of test data from game prototype	Won't	Not necessary for a test of this scale. Development time would be greater than time saved.

9 Methodology

The goals of the study are:

- To establish if intuitive design allows players to understand the game faster.
- To establish if intuitive design increases player enjoyment.
- To establish if intuitive design increases how intuitive the player finds the game.
- To establish if intuitive design decreases the player's perceived difficulty of the game.

In order to establish the effectiveness of the intuitive design, participants will be required to play the game prototype and data will be collected. There will be two levels; one created using intuitive design and one with a simpler design as a control. The levels should be as similar in length and challenge as possible to ensure a fair comparison. Participants will only play one of the two levels as they must have no prior experience with the game for the data to be valid. An equal number of participants should play each level.

Both levels will be split into the same number of sections. Each section will contain one challenge which is as similar as possible between the two levels. Data will be collected for each challenge rather than the entire level which will give a fairer comparison. For example, if the completion time of the entire level was compared this could be heavily skewed by a single challenge and would be difficult to draw any conclusions from. The data to be collected and how it will be obtained is shown in Table 10.

Table 10: Information about the data to be collected during the study

Data to collect	How data will be collected	Data Type
How fast the player completes each section.	Stored automatically by game.	Ratio
How many times the player fails in each section	Stored automatically by game.	Ratio
How fun the player found the game when asked after playing.	Player survey after playing. Rate from 1-10.	Ordinal
How easy the player found the game when asked after playing.	Player survey after playing. Rate from 1-10.	Ordinal
How intuitive the player found the game when asked after playing.	Player survey after playing. Rate from 1-10.	Ordinal

The completion time and number of failures form the quantitative data collected in this study. The mean and standard deviation of the quantitative data will be calculated from the raw data for each level. T-tests will be carried out on the quantitative data to gauge whether the results shown are statistically significant. Two separate t-test will be carried out; one for completion time and one for number of failures. The following hypotheses will be used for the tests:

Null hypothesis: Intuitive design has no effect on completion time or number of failures.

Alternative hypothesis: Intuitive design results in a faster completion time and a lower number of failures.

It is important to try and control as many of the variables, other than the level design, that may affect the results. Below is a list of things that must be kept as similar as possible between the two test set-ups.

Table 11: The variables that must be controlled during the study and measures taken to control them

Variable to be controlled	Control measures
Game mechanics and audiovisuals.	Use same game design and implementation for both levels. Build levels from same components just structured differently.
Level Length	Level consists of several consecutive individual challenges. Same number of challenges and each challenge is similar in style and length between two levels.
Level Challenge	Try and keep each challenge as similar as possible between two levels, with slight changes to make one more intuitive. Keep order of challenges the same for difficulty progression.
Player experience (how frequently they play games, what types of games they play, how good at this type of game they are, how much they enjoy this type of game)	Try and recruit candidates with similar amount of experience playing games. Use candidates who regularly play games and are reasonably capable.
Bugs affect gameplay in unintended ways and change results. E.g player fails a section due to a bug.	Game must be thoroughly tested during and after development before data is collected. Testing by developer should be sufficient.

It is important to collect enough data to draw conclusions, so a suitable number of participants must be recruited. A minimum of 8 testers per level is the aim for this study. This will result in 16 overall testers playing the game, as player's can only play one of the two levels. It would be ideal to recruit as many participants as possible to improve the results. However, care must be taken when recruiting candidates to ensure they are similar in skill level. Player experience is the most difficult variable to control and is the most likely factor to invalidate the results. By recruiting more participants this will help average out the effect of player experience and means results are more likely to be reliable. One of the requirements for the study is that the participants should be from the game's target audience. Recruiting many participants that are not similar in experience level will not produce better data. This would be poor control of the player experience variable which will affect the data collected significantly.

10 Project Plan

10.1.1 Introduction

Expanding on the defined methodology, a plan has been created for this study to ensure it can be realistically completed in the allotted time period. The deliverables have been defined so that progress can be tracked while carrying out the study. The project has been split into tasks and completion durations have been estimated. These have been displayed in a Gantt chart to highlight the critical path.

10.1.2 Deliverables

1. Game design document.
    a. Mechanic design
    b. Premise design
    c. Level design
    d. Software design

2. Game prototype
    a. Game loop and main mechanics
    b. Control level
    c. Intuitive level
3. Collecting results
    a. Recruit participants
    b. Participants test game, collect data
4. Documentation and Report
    a. Methodology
    b. Results
    c. Evaluating results
    d. Conclusions
    e. Commenting code

10.1.3 Tasks

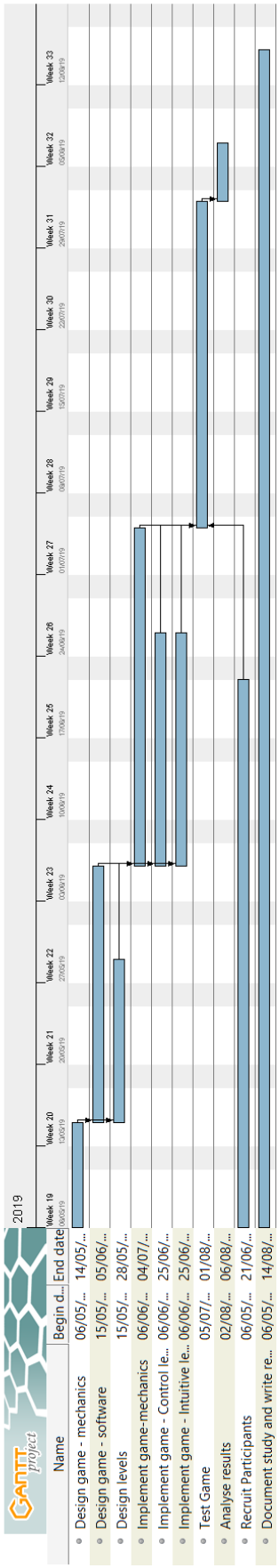
Table 12 shows the tasks that must be completed and their estimated duration. Some durations, such as the time for testing and recruiting, have been overestimated to account for the difficulty to predict these exactly. Overestimating is preferable here to underestimating to ensure the study can be completed on time. If the project relied on an underestimated task duration to be completed on time and it took longer than expected this would make the project run late. Planning around these overestimated times ensures the tasks will be completed on time or early, allowing additional time for other tasks.

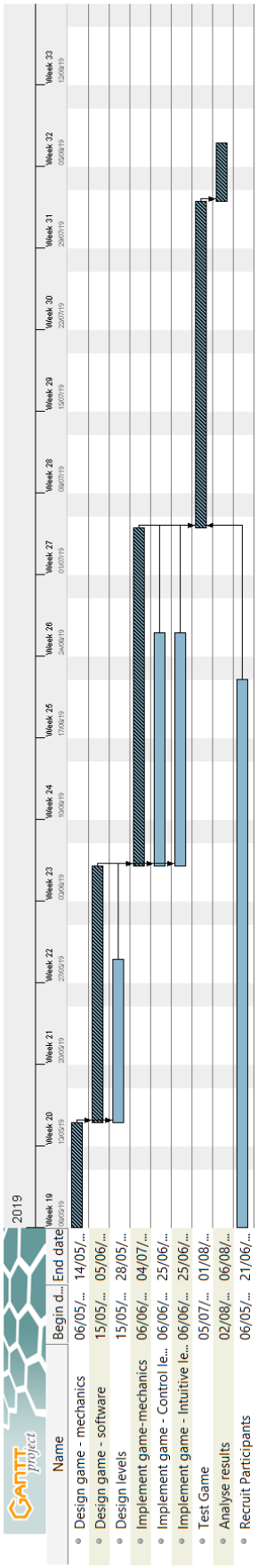
Table 12: The tasks required to complete this study

Task	**Duration**
Design game – Mechanics, features, premise, background	7 days
Design game – Software side. (Classes, patterns, libraries)	16 days
Design game – Levels	10 days
Implement game – Mechanics and game loop	21 days
Implement game – Create control level	14 days
Implement game – Create intuitive level	14 days
Recruit Participants	35 days
Test game: Get feedback and stats, use about 8 people from target audience per level	20 days
Analyse Results	3 days
Documentation + Report	Project duration

In section 10.1.4 the Gantt chart shows the dependency between these tasks. Where possible, tasks have been planned to be completed in parallel. This allows for float on some tasks, meaning if they run slightly late it will not affect the overall project. The critical path highlights which tasks must be completed on time for the project to remain on time. During the development process this Gantt chart can be used to track progress. Tracking the project progress means if the project starts to fall behind action can be taken immediately to bring it back on track.

10.1.4 Gantt Chart





10.1.5 Risk Assessment

There are risks associated with this study that may jeopardise the plan and result in the study running late or not being completed on time. Table 13 shows the risks that have been identified for this study and how severely they would affect it. Mitigation methods have been defined for each to lessen their chance of causing issues or delays. Although risk cannot be completely eliminated from the project, the chances of issues arising after the mitigation methods is low.

Table 13: The risks and their corresponding mitigation strategies for this study

Risk	**Severity**	**Probability**	**Mitigation**	**Severity after Mitigation**	**Probability**
Unity licensing changes.	H	L	Unity has been free for personal use since 2009 (Helgason, 2009) and has not announced this will change. No elements of game design should be dependant on Unity specific features so that it could be easily ported to another engine.	M	L
Game is buggy and bugs interfere with testing results.	M	M	Game should be frequently tested during development to identify bugs. These should be fixed whenever possible or mitigated somehow. Worst case players should be informed so they know what to expect.	M	L
Main mechanic is difficult to understand and interferes with level testing.	H	M	Main mechanic should be developed early and iteratively to refine it.	H	L
Hardware failure causes loss of study work e.g. game prototype and level designs.	H	M	Work for this study should be backed up to a cloud service like Github.	H	L
Hardware failure causes loss of test data.	H	M	Test data must be backed up. Care must be taken when backing up test data to make sure that it is still secure.	H	L
Unity does not provide the correct functionality to create the game prototype as intended	M	M	Unity features have been researched in advance to ensure it should be able to support the intended design. Concept prototype was created to test basic functionality/	M	L

Cannot find enough participants to test game, result in insufficient data	H	M	Recruiting will begin at the start of the task. Recruit more participants than required if possible. Leave extra time for testing so that can work around people's schedules	H	L
Underestimated prototype development time	M	M	Prototype has been planned and split it into sub-tasks. Lengths for each sub-task have been estimated and structured with a Gantt char	L	L
Game prototype has poor performance and this affects testing results	M	M	Game should be tested during development on machine that will be used for test participants. Only one machine will be used for collecting data so that performance is always identical	L	L
Recruited participants are unavailable when the time comes for testing.	M	M	Participants will be recruited early and informed of the testing dates. A booking schedule for participants should be set up to ensure everyone fits in. Extra-time should be allowed for testing in plan to account for unforeseen circumstances	L	L

11 Professional, Legal, Ethical and Social Issues

There are many professional, legal, ethical and social issues to consider when this study is carried out. In order to ensure these are addressed suitably, plans have been established for any foreseeable issues. The precautions taken to avoid any issues with the study have been split into their respective sections and described below.

In the following sections "Participants" refers to people who have consented to take part in the study and "Candidates" refers to potential participants that have not yet agreed to take part.

11.1 Professional Issues

- Test participants should be treated with care during the study. Any requests or questions they have should be treated with utmost importance to ensure their comfort when taking part in the study.
- Candidates and participants will always be treated with respect.
- All software used during the study will be free, open-source or a relevant license must be obtained.
- If the license requirements of any software used change during the study action must be taken immediately to acquire the relevant license or find an alternative software package to use.

- If any assets are supplied externally to be used in the game prototype full consent must be obtained stating what assets are being supplied and agreeing how they are to be used.
- All intellectual property used in the study must be original or acquired with consent.
- The game prototype will be developed using version control to ensure data is never lost. If changes are made that negatively affect the prototype these can be reverted easily.

11.2 Legal Issues

- Test participants will be required to give consent for their data to be used in the study.
- Test participants have the right to view data stored about them and correct any mistakes in accordance with GDPR.
- Data regarding test participants should be as adequate, relevant and correct as possible in accordance with GDPR.
- Data regarding test participants will not be stored longer than required in accordance with GDPR.
- Data regarding test participants will be stored securely to prevent their data being used for anything other than this study in accordance with GDPR.
- Data regarding test participants will be stored with the utmost care to prevent unlawful or accidental loss of data.
- Copyrighted assets must not be used in the game prototype without the correct intellectual property agreements.

11.3 Ethical Issues

- Ideally test participants will be from the game prototype's target market. This only refers to the participants frequency of playing games and preferred game genres. Candidates will not be discriminated on any other factors.
- Candidates for the study will only be contacted through means they have provided permission for.
- Only questions relevant to the study should be asked in the questionnaire.
- Only data relevant to the study should be captured during the testing of the game prototype.
- Candidates will be given 2 days to review the information about the study and decide whether they wish to participate.
- Participants will be allowed to opt-out of the study at any point while it is ongoing and have their data removed. This right will be revoked when the data is published in the final report to prevent amendments to the publication being required.

11.4 Social Issues

- Candidates should be fully informed about the contents of the game and questionnaire before they decide whether to participate so that they can make an informed decision.
- Data will be stored anonymously as information about the participants identity is not required for this study. This protects the participants data if the data is illegally accessed.
- Participants will only be required to use standard hardware to complete the study. This will most likely be a laptop or PC with the option of using a games console controller or mouse and keyboard.

12 Game Prototype Design Concept

12.1 Mechanic Design

A concept has been developed for the game prototype that will be used to test intuitive design practices. It was important to create a concept early so that the design could be iterated on before the final prototype (Fullerton, 2008). The concept was shown to a focus group so that feedback could be received from potential players and incorporated into the design.

The game is a 2D platformer based around the core mechanic that a player can shoot a projectile out of the soles of their feet. Using this ability in different situations can result in different interactions. For example, shooting downwards in the air can allow for an additional jump. A 2D platformer was selected due to the linearity of this type of game. The player will only be able to progress along one path, which more easily allows the designer to influence their experiences using level design. This linearity constraint will mean every player who tests the game will have the same experience and this makes the results more reliable.

Player Mechanics: Move, jump, shoot forwards on ground, shoot upwards on ground, shoot downwards in air, shoot forwards in air, shoot upwards in air.

As stated, the game will be designed around this core mechanic, meaning other mechanics must build upon it. Below is a list of the secondary mechanics planned, and how they interact with the main mechanic.

Mechanic	Interaction with core mechanic
Enemies	Enemies will be defeated by projectiles. Different enemies will be invulnerable from different directions, requiring the player to use the core mechanic differently to defeat them.
Switches and switchable objects (e.g doors)	Projectiles will trigger switches, swapping the state of the switchable object linked to them.
Reflectors	Reflectors will reflect projectiles in differing directions based on their angle.
Grappling points	Shooting a grappling point with a projectile from below will allow the player to tether themselves to it if they are airborne. From the grappling position they will be able to jump again.
Grappling lines	Grappling lines will connect two grappling points. If the player tethers to one of the grappling points they will travel along the line to the other point, then be released.

A “core mechanic” based design (Kim, 2012) was chosen to ensure that the mechanics of the game were easy to understand. Rather than the player having to learn several isolated mechanics the game will build upon a single core mechanic and use other mechanics that interact with it.

Another advantage of the “core mechanic” design is that the secondary mechanics chosen can be part of the levels. This mechanic structure has been chosen to make the game easy for testing intuitive level design. When the game begins the player will only have one character mechanic to learn. Once they grasp this, they must learn the secondary mechanics built into the level which can be taught to them with or without intuitive design techniques. Each of these secondary mechanics

can be compared in isolation to see if intuitive level design affected how well the player understood them.

12.2 Premise

The premise of a game creates the context or setting that the game is taking place in (Fullerton, 2008). Without a premise it is unlikely player's will emotionally engage with the game. Premise can vary vastly in complexity and scale, but even a simple premise is often enough to draw a player in. The goal of a good premise is not just to establish a meaning to the game, it is to engage the player's emotions and evoke them into becoming invested in the game's setting. Some of the tools that can be used to establish a premise are; time period, location, characters, goals, conflict and history. To give an example of a simple premise and a more complex one we can look at *Space Invaders* and *Fire Emblem: The Blazing Blade* (Taito, 1978) (Intelligent Systems, 2003). *Space invaders* has a very simple premise, the player must take the role of an anonymous ship defending against an incoming horde of alien invaders. The simplicity allows new players to immediately understand the premise and become involved in the game. In contrast, *Fire Emblem: The Blazing Blade* begins the game by explaining to the player the history of the world they are entering. The game features a main dramatic arc, many different playable characters with individual backstories and interaction between many of the characters. The premise of *Fire Emblem: The Blazing Blade* is huge, complex and player's will undoubtedly not uncover the entirety of it during a single playthrough of the game. Space Invaders was still a hugely popular game despite this huge difference in scale, showing a simple premise can be sufficient to make a game engaging.

Figure 16: (Left) Space invaders has a simple and immediately obvious premise (Fullerton, 2008). (Right) Fire Emblem: The Blazing Blade has a much more in depth and complex premise. This image shows an unlockable interaction between two of the many characters (Kantopia, 2018).

Although a full premise has not yet been established for the prototype, a basic idea has been created which can be built upon. Several simple components have been formulated and these are shown in Table 14. The combination of these components will form a strong but simple premise to accompany the gameplay mechanics. The goal for the prototype would be to have a 10-20 second introduction explaining the history, and for the rest of the premise to be obvious through the gameplay. The combination of the goals and conflict aim to keep the player emotionally engaged and interested to discover the outcome.

Table 14: The premise designed for the game prototype

Premise Component	**Explanation**
Location	Space. Foreign fictional galaxy.
Main Character	Human female. Androgynous space suit/armour with no helmet. Long hair in a ponytail. Ex-policing force.
Goals	Recover a missing ally who previously attempted to infiltrate villain's gang. Their fate is unknown but will be discovered at end of level. Also aiming to damage our destroy the villains' organisation if possible.
Villains	A gang of cyber-terrorists operating an intergalactic black-market. Members consist of many different races of aliens and includes humans too.
Conflict	Main conflict between policing force and terrorist gang. Villains are immediately hostile to player.

A foreign fictional galaxy was chosen to give the developer creative freedom when designing the setting. Using a setting based on reality can save time on developing the setting but care must be taken to stick to it, or the player may notice any errors. Long hair was chosen for the character because it can be used in the animations to provide feedback to the user. Hair can be easily used to convey movement, which will be important for enhancing the player's core mechanics of moving, jumping and shooting. Using a gang made up of different races allows a variety in the type of enemies encountered in the game. This will make it easy for the player to distinguish enemy types and enemy aesthetics can be designed to hint at their functionality.

12.3 Concept Prototype

A small test prototype was created in Unity to allow players to experiment with the core mechanic. This lacked some of the secondary mechanics, but the main goal of this prototype was to prove the core mechanic. Free to use assets where used for the sprite art and sound effects and these are not representative of the premise or final game.

The mechanics included in the concept prototype were:

- Player moving and jumping.
- Player shooting projectiles forwards and upwards from standing.
- Player shooting projectile forwards and upwards in air.
- Player shooting projectile downwards in air to double jump.
- Enemies that move left and right and defeat player. No invulnerabilities to any direction.
- A triggerable switch connected to a platform that will appear and disappear.

The concept game could be played with keyboard or controller and was built for Windows. Rather than progressing through a level the player was constrained to one screen and simply had to defeat as many of the endless enemies as possible. A screenshot of the concept prototype is shown in Figure 17.

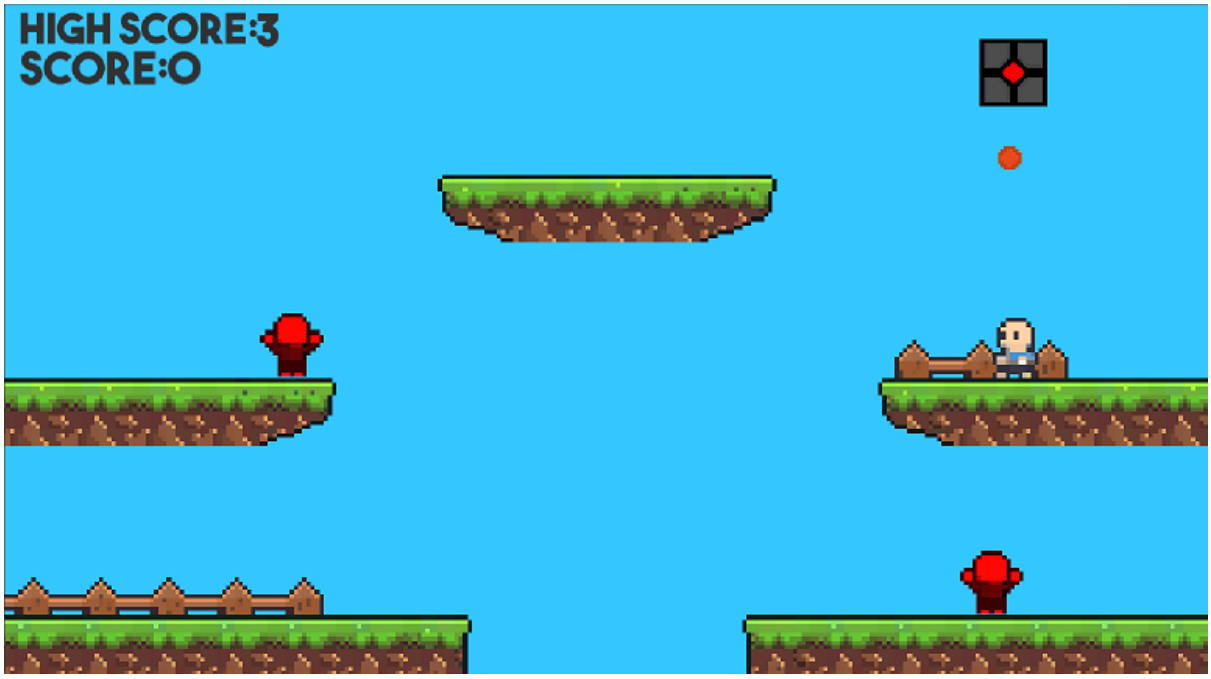


Figure 17: The concept prototype that was tested by the focus group. On the right the player can be seen shooting a projectile upwards towards a switch. The red humanoids are enemies.

12.4 Design Concept Review/Focus Group

The Heriot Watt University game development society was used as a focus group to receive feedback on the design. Playtesting is an essential part of game development and is important for both mechanic and level design (Kayali & Schuh, 2011). The game was built and run on a Windows laptop.

A presentation was created showing the core mechanic and ideas of other features to interact with this. This presentation included all features planned for the full game prototype, not just what was included in the concept prototype.

After the playtesting and presentation, the focus group was asked if they had any feedback on the game design. The goal was to establish whether the mechanics were robust and whether they would provide sufficient complexity for the study. As a whole, the group gave similar feedback and the concept was well received. The feedback received is listed below:

- The mechanics are strong and will provide enough complexity for the study.
- The concept prototype is missing a premise and this should be in the final prototype. A proper background and character are necessary for player engagement.
- The mechanics seem fun.
- The main mechanic is easy to understand.
- The controls in the concept prototype have flaws and should be changed for the final game.

Overall the focus group helped to prove the mechanics of the game which was the main aim of this concept prototype. The feedback taken from here can be used to develop the main prototype and should ensure a better designed game.

13 Conclusions

13.1 Literature Review

The aim of this study is to investigate if intuitive design principles can be applied to level design to improve player experience. Both player performance and player enjoyment will be tested to evaluate whether intuitive design has a statistically significant effect. The literature review went into depth about what intuition means for video games, and what benefits it can offer. The goal of intuitive design is to allow new players to easily understand games and to give them the tools to work things out for themselves. Unintuitive games can lead to players becoming frustrated or losing interest. Some intuitive design techniques were explored with relevant examples from commercial games.

As part of this study a game prototype will be developed and used to test intuitive level design. The different development technology and build platforms available were compared to evaluate which best suited this project. Development of the game prototype will use a game engine to save development time and Unity was selected due to its comprehensive features, good documentation and its compatibility with solo developers. Windows was chosen as the build platform as the game will be tested and used to collect data on a single Windows laptop.

13.2 Requirements Analysis

Following the literature review, the requirements for this study were defined. The requirements were expanded in detail and assigned MoSCoW priorities. It is important to establish the most important requirements so that these can be focused on during development. The main requirements for this study are; to produce the game prototype, create two levels to be compared, recruit participants and gather sufficient test data to analyse.

13.3 Methodology

Gathering the test data is a key topic of this study and must be done correctly to ensure valid conclusions can be evaluated. The planned methodology for carrying out this study was discussed, along with factors that must be controlled for the results to be valid. Test participants will play one of the two levels and data about their playthrough will be collected. Participants will also be interviewed about how much they enjoyed the game and how difficult they found it. The quantitative data collected about the participants performance in the game will be evaluated using a t-test to determine if any trends are statistically valid.

13.4 Project Plan

A comprehensive project plan was created to ensure the study will be completed on time. The deliverables have been defined, along with tasks to achieve them. A Gannt chart has been used to map out these tasks over the study timescale. The critical path has been shown which helps to define which tasks are at risk of delaying the project. Risk assessment has been carried out, and mitigation methods have been defined to de-risk the project.

13.5 Professional, Legal, Ethical and Social Issues

There are several professional, legal, ethical and social issues which may affect the study while it is being carried out. Care has been taken to address these and provide information about how they have been accounted for in the planning stage. The main issues are related to the test participants and their data. Test participants must be well treated during the study and their data must be handled securely, in accordance with GDPR.

13.6 Game Prototype Design Concept

A design concept has been created for the game prototype that will be used for testing during this study. Although not completely developed yet, this concept serves as a base to be developed upon as the study moves forward. A 2D platformer based around a core mechanic of the player shooting a projectile from their feet was chosen. The linearity of a 2D platformer, and the simplicity of a game designed around a core mechanic make this design perfect for gathering the data required for this study. Ensuring the player progresses along a single path means that all participants will go through the same experience, and a simple core mechanic means the player mechanics should not hinder the player and the focus can shift to the level. A basic premise has been formulated to ensure the player engages emotionally with the game. A premise is necessary to provide a realistic representation of a commercial game.

A concept prototype was created and tested by the Heriot Watt game development society to receive feedback. This concept implemented the main mechanic but did not include the premise and some of the secondary mechanics. The concept was well received and some of the feedback recorded can be used to improve the game design moving forwards.

14 References

Abe, H., Seo, H. & Lee, D., 2011. The prefrontal cortex and hybrid learning during iterative competitive games. *Annals of the New York Academy of Sciences,* 1239(1).

Anthropy, A. & Clark, N., 2014. *A Game Design Vocabulary.* 1 ed. Crawfordsville: Addison Wesley.

astral, l., 2014. *That time when you incinerate the companion cube.* [Online]
Available at: https://www.youtube.com/watch?v=rq1EVrduBDI
[Accessed 25 03 2019].

Brown, M., 2015. *Half-Life 2's Invisible Tutorial | Game Maker's Toolkit.* [Online]
Available at: https://www.youtube.com/watch?v=MMggqenxuZc
[Accessed 10 02 2019].

Brown, M., 2017. *The Design Behind Super Mario Odyssey | Game Maker's Toolkit.* [Online]
Available at: https://www.youtube.com/watch?v=z_KVEjhT4wQ
[Accessed 22 02 2019].

Brown, M., 2018. *The World Design of Metroid 1 and Zero Mission.* [Online]
Available at: https://www.youtube.com/watch?v=kUT60DKaEGc
[Accessed 02 2019].

Bruns, R. N., 2008. *Super Mario Brothers - World 1-1 Labeled Map.* [Online]
Available at: http://www.nesmaps.com/maps/SuperMarioBrothers/SuperMarioBrosWorld1-1Map.html
[Accessed 08 03 2019].

Burrows, J., Jackson, W. & Rideg, M., 2019. Unity versus Unreal: Choosing your Game Engine. *3D World,* Issue 242, p. 86.

Cavanagh, T., 2010. *VVVVV on Steam.* [Online]
Available at: https://store.steampowered.com/app/70300/VVVVVV/
[Accessed 26 03 2019].

Cavanagh, T., 2010. *VVVVV. Game [Microsoft Windows].* London: Terry Cavanagh.

Centaur Communications Ltd, 2006. GAMES: Use your intuition. *Design Week,* 21(23), p. 16.

Chen, J., 2007. Flow in Games & Everything Else. *Communications of the ACM,* 50(4), pp. 31-34.

Christopoulou, E. & Xinogalos, S., 2017. Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *International Journal of Serious Games,* 4(4).

Csikszentmihalyi, M., 2014. *Learning, "Flow," and Happiness.* 3 ed. Claremont, USA: Springer, Dordrecht.

Desurvire, H. & Wilberg, C., 2008. *Master of the game: assessing approachability in future game design.* Florence, Italy, CHI '08 Extended Abstracts on Human Factors in Computing Systems.

Desurvire, H. & Wixon, D., 2013. *Game Principles: Choice, Change & Creativity: Making Better Games.* New York, CHI '13 Extended Abstracts on Human Factors in Computing Systems.

Finneran, C. M. & Zhang, P., 2003. A person–artefact–task (PAT) model of flow antecedents in computer-mediated environments. *International Journal of Human-Computer Studies,* 59(4), pp. 475-496.

Fischer, F., 2016. *Why We Need Challenge.* [Online]
Available at:
http://www.gamasutra.com/blogs/FabianFischer/20160224/266405/Why_We_Need_Challenge.php
[Accessed 02 03 2019].

Fullerton, T., 2008. *Game Design Workshop.* 2 ed. Burlington: Morgan Kaufmann.

GameDesigning.org, 2019. *The Top 10 Video Game Engines.* [Online]
Available at: https://www.gamedesigning.org/career/video-game-engines/
[Accessed 19 02 2019].

Gauthier, A. & Jenkinson, J., 2018. Designing productively negative experiences with serious game mechanics: Qualitative analysis of game-play and game design in a randomized trial. *Computers & Education,* Volume 127, pp. 66-89.

Gregory, J., 1970. *Game Engine Architecture.* 2 ed. Boca Raton, Florida: CRC Press.

Grow, B., 2014. *Good Game Design - Shovel Knight: Teaching Without Teaching.* [Online]
Available at: https://www.youtube.com/watch?v=cYvPdEyTXUc&t=1s
[Accessed 10 02 2019].

Helgason, D., 2009. *A free Unity?.* [Online]
Available at: https://blogs.unity3d.com/2009/10/29/a-free-unity/
[Accessed 31 03 2019].

Hoffman, B. & Nadelson, L., 2010. Motivational engagement and video gaming: a mixed methods study. *Educational Technology Research and Development,* 58(3), pp. 245-270.

Intelligent Systems, 2003. *Fire Emblem: The Blazing Blade. Game[Game Boy Advance].* Kyoto: Nintendo.

Iyer, A., 2017. *Analyzing Super Mario's level and tutorial design.* [Online]
Available at: https://medium.com/@abhishekiyer_25378/the-perfect-game-tutorial-analyzing-super-marios-level-design-92f08c28bdf7
[Accessed 20 03 2019].

JapanCommercials4U2, 2012. *Super Mario 3D Land Playthrough Part 1.* [Online]
Available at: https://www.youtube.com/watch?v=wdKnUqaHSmc
[Accessed 17 03 2019].

Kantopia, 2018. *FE7 Blazing Blade Localization: Karel's "Sword" – Physical Blade or a Technique? [JPN vs ENG].* [Online]
Available at: https://kantopia.wordpress.com/2018/06/07/fe7-blazing-blade-localization-karels-sword-physical-blade-or-a-technique-jpn-vs-eng/
[Accessed 02 04 2019].

Kayali, F. & Schuh, J., 2011. *Retro Evolved: Level Design Practice exemplified by the Contemporary Retro Game.* Hilversum, The Netherlands, DiGRA 2011 Conference: Think Design Play.

Kiili, K., 2005. Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education,* 8(1), pp. 13-24.

Kim, C., 2012. *Designing around a Core Mechanic.* [Online]
Available at:
https://www.gamasutra.com/blogs/CharmieKim/20120612/172238/Designing_around_a_core_mechanic.php
[Accessed 20 02 2019].

Kolb, D. A., 2014. *Experiential Learning.* 2 ed. New Jersey: Pearson Education Inc..

Kubovy, M. & Pomerantz, J. R., 2017. *Perceptual Organisation.* 1 ed. London: Routledge.

Mitgutsch, K. & Schirra, S., 2012. *Subversive Game Design and Meaningful Conflict.* [Online]
Available at: http://gamelab.mit.edu/research/subversive/
[Accessed 05 02 2019].

NES Maps, 2008. *Super Mario Brothers - World 1-1.* [Online]
Available at: http://www.nesmaps.com/maps/SuperMarioBrothers/SuperMarioBrosWorld1-1Map.html
[Accessed 28 02 2019].

NES Maps, 2013. *Metroid - All of Planet Zebes.* [Online]
Available at: http://www.nesmaps.com/maps/Metroid/MetroidCompleteMap.html
[Accessed 08 03 2019].

Nintendo Creative Department, 1985. *Super Mario Bros. Game [NES].* Kyoto: Nintendo.

Nintendo EAD Tokyo, 2011. *Super Mario 3D Land. Game [Nintendo 3DS].* Tokyo: Nintendo.

Nintendo EPD, 2017. *Super Mario Odyssey. Game [Nintendo Switch].* Kyoto: Nintendo.

Nintendo R&D1 and Intelligent Systems, 1983. *Metroid. Game[NES].* Kyoto: Nintendo.

Nutt, C., 2012. *The Structure of Fun: Learning from Super Mario 3D Land's Director.* [Online]
Available at:
http://www.gamasutra.com/view/feature/168460/the_structure_of_fun_learning_.php?page=4
[Accessed 20 02 2019].

Nystrom, R., 2014. *Game Programming Patterns.* 1 ed. Seattle, WA: Genever Benning.

Podleschny, N., 2012. *Games for change and transformative learning: An ethnographic case study,* Queensland University of Technology: Queensland University of Technology.

Ren'Py, 2019. *What is Ren'Py?.* [Online]
Available at: https://www.renpy.org/
[Accessed 10 03 2019].

RPS, 2013. *Untold Riches: An Analysis Of Portal's Level Design.* [Online]
Available at: https://www.rockpapershotgun.com/2013/09/20/untold-riches-an-analysis-of-portals-expressive-level-design/
[Accessed 14 03 2019].

Shirinian, A., 2013. Intuitions, Expectations and Culture. *Game Developer,* p. n/a.

Steam, 2007. *Portal.* [Online]
Available at: https://store.steampowered.com/app/400/Portal/
[Accessed 07 03 2019].

Taito, 1978. *Space Invaders. Game [Arcade Cabinet].* Tokyo: Taito.

Team Meat, 2010. *Super Meat Boy. Game [Microsoft Windows].* Asheville: Team Meat.

tvtropes, 2019. *Antepiece.* [Online]
Available at: https://tvtropes.org/pmwiki/pmwiki.php/Main/Antepiece
[Accessed 0 2019].

tvtropes, 2019. *Antepiece.* [Online]
Available at: https://tvtropes.org/pmwiki/pmwiki.php/Main/Antepiece
[Accessed 16 03 2019].

Unity, 2019. *The world's leading real-time creation platform.* [Online]
Available at: https://unity3d.com/unity?_ga=2.138526643.1471254040.1553439662-321221326.1538304610
[Accessed 24 03 2019].

Unity, 2019. *Unity Personal.* [Online]
Available at: https://store.unity.com/products/unity-personal
[Accessed 25 03 2019].

Valve Corporation, 2004. *Half-Life 2. Game [Microsoft Windows].* Belevue, Washington: Valve Corporation.

Valve Corporation, 2007. *Portal. Game [Microsoft Windows].* Bellevue, Washington: Valve Corporation.

Valve, 2010. *Super Meat Boy - Steam.* [Online]
Available at: https://store.steampowered.com/app/40800/Super_Meat_Boy/
[Accessed 13 03 2019].

Webster, J., Trevino, L. K. & Ryan, L., 1993. The dimensionality and correlates of flow in human-computer interactions. *Computers in Human Behaviour,* 9(4), pp. 411-426.