|  |
| --- |
| **Investigation of Fluid Movement** |

**Course: Game Design and Programming**

**Name: Connor Jones Supervisor: James Vickers**

**Date: 14/10/2024**

# Abstract

AAA games often suffer from a lack of quality due to the push for more marketing and features which make money rather than crafting quality player controllers due to the increasing need to focus on achieving profit margins to make back profit for a massive project to pay off. Indie titles are taking over the market due to the quality and creativity of their titles, the key difference is the game feel of the player which boosts user experience as indie developers though lacking in resources often have time to focus on fun, but this is not always true. This paper discusses how movement should be implemented into a game and what aspects of movement are important, we also discuss the importance of game juiciness and compare indie-developed games to AAA.

The developed artefact to demonstrate an advanced player controller was developed using Unity version 6.1 for Windows computer systems. The developed project used a state machine to dynamically switch player states allowing for smooth movement and transitions between mechanics, some basic animation was also attempted and achieved. The paper will discuss the issue of developing some mechanics and challenges faced when implementing animations, including steps taken for the project to succeed. The end of the paper concludes the lessons learned while working on the investigation, including the efficiency of finding key components, setting conclusive questions and breaking them down to understand the direction of research.

# Contents

[Abstract 1](#_Toc191394352)

[Contents 1](#_Toc191394353)

[Introduction 3](#_Toc191394354)

[Aims and Objectives 4](#_Toc191394355)

[Literature Review 5](#_Toc191394356)

[Importance of movement in games 5](#_Toc191394357)

[History of video game movement 6](#_Toc191394358)

[2D Games 6](#_Toc191394359)

[3D Games 7](#_Toc191394360)

[Factors supporting movement 8](#_Toc191394361)

[Design methodologies 9](#_Toc191394362)

[Animation 11](#_Toc191394363)

[“Game Juiciness”: How does it influence the game's feel and player competence? 12](#_Toc191394364)

[Comparison of game quality between Indie Devs and AAA Companies 13](#_Toc191394365)

[Procedural camera animations (add to recovered version) 13](#_Toc191394366)

[State Machines 14](#_Toc191394367)

[Research Methodologies 15](#_Toc191394368)

[Results and Findings 17](#_Toc191394369)

[Development Lifecycle 17](#_Toc191394370)

[Coroutine Approach 17](#_Toc191394371)

[Finite State Machine(FSM) 19](#_Toc191394372)

[Testing Findings 21](#_Toc191394373)

[Discussion and Analysis 23](#_Toc191394374)

[Conclusion 25](#_Toc191394375)

[Recommendations 28](#_Toc191394376)

[Bibliography 29](#_Toc191394377)

# Introduction

Movement in video games is a key mechanic for a player controller, in the industry developers must produce quality movement systems to help support the player travelling from one point to another, as stated by (Terry Clinkenbeard, 2017) fundamentally video games are about movement from point and click games to card selection games they all share a type of movement. The problem with movement in current AAA game titles is that they fail to create fluid and unrestrained movement through their environment as suggested (Adderley, 2020) for example Grand Theft Auto tries to mimic realistic movements, but this can create annoying controls. Movement which is not fluid can create problems within the game itself for example if a turning animation is too slow this could ruin parts of the player experience which could lead to the player closing the game. The key factors of manoeuvrability in games are player acceleration and how well they can turn whilst moving because, without speed or control, players would not get anywhere as stated (Adderley, 2020).

The game Neon White implements mechanics which assist with control of the player controller, this mechanic is called wall control, and it allows the player to manoeuvre through the entire level whilst moving at a fast speed. This is a good example of a fluid movement because it will enable the player to precisely move through complex levels, if speed and control are managed correctly this will be received well by players as (Reilly, 2022) describes it as a marvellous sense of momentum.

 To back up these claims (IMDb, 2022) has listed awards Neon White has been nominated for and although this game did not win awards it still shows the game was recognised well by representatives at famous video game companies. This study aims to research the process of designing a good movement system and the logic behind fluid movement to create an artefact using the key factors of movement speed and control to create a fluid movement system with mechanics to support the player. The optimisation of games is also a factor of good movement systems because as stated by (Polydin, 2023) optimisation directly affects the game as a whole meaning if the game is not optimised the movement system will not feel good to play, ensuring the solution is optimised research will also delve into how we can optimise video games to enhance the player experience.

This should solve the problem of restrained movement as this research should outline the industry's mistakes during development and provide a functional solution. To ensure the data researched and artefact produced are accurate to suggested findings, data will be collected from selected testers to ensure issues are solved quickly. Finally, the question to answer at the end of this study to conclude the investigation is what design techniques and mechanics make up a successful movement system? Answering this question is important because once movement in games has been fully investigated the processes behind games such as Mirror’s Edge and other similar titles can be identified for future games ensuring movement can continue to be improved.

Key areas to focus on other than the physics of the world that control movement would be the concept of game juiciness as this aspect of gameplay can significantly affect the expression of a project to increase player experience. Further into the paper we will break this concept down into what components make up game juiciness and research which backs up the game juiciness concept as well as using this as inspiration for artefact testing.

# Aims and Objectives

This research paper will investigate movement systems of existing games to identify the problems of AAA title movement and to provide a solution to show how the movement should be implemented in games to emphasise freedom and movement. Supporting mechanics will also be investigated to ensure the solution can provide answers referring to gameplay implementations.

**Primary Aim -**

* The primary objective of this paper is to propose a solution for developing a fluid movement system that can improve gameplay and manoeuvrability. The paper will outline design processes for creating an effective movement system, to examine the current state of movement in games. The research will be utilised to develop an effective solution to the problem of mishandled movement, solutions produced will not lack defects but will aim to set a good example for future games. During the investigation of movement, an intriguing question that must be answered is why AAA companies fail to create competent movement systems. This is important because the root of the problem must be identified to provide a competent solution to show to fellow developers.

**Objectives –**

* Research existing systems using documentation, video talks and owned games to identify how the industry implements design techniques to create a fluid movement system that feels good to play. This research will then discuss how these systems can be improved if they were to be iterated upon and will help identify a common mistake between different studios.
* Research into the correlation between animations and effects to support movement believability, this will outline ways of helping players have a better experience with first-person free-running games which supports the nature of this study.
* Investigating mechanics that support the player's movement, such as wall-running and ledge climbing, will help this study, as mechanics play a big role in the fluidity of movement with player controllers.

# Literature Review

## Importance of movement in games

Movement in games is important and this is proved by (School of Game Design, n.d.) in games such as The First Tomb Raider programmers did not understand properly how to use a 3D space so each movement component such as grabbing a ledge mid-air had a button which these days as described by (School of Game Design, n.d.) would feel too clunky. (School of Game Design, n.d.) points out that over time the movement system in later Tomb Raider games became easier to use as your character would automatically grab ledges making the game more accessible. There are few mechanics in games which are as fundamental as movement (Rempton Games, 2019), most games other than decision-based games require a character controlled by the player typically running, jumping and in some cases climbing and although the importance of movement may seem obvious (Rempton Games, 2019) suggests it can be easy to look over this fact.

A good example of a bad feeling movement system would be as pointed out by (Dorset, 2024) in Grand Theft Auto 5 the movement system felt clunky because the focus was to capture realism so turning the character around took longer. After all, the animations were trying to copy real life and sometimes players can get stuck on objects. Another great example from (Dorset, 2024) would be Assassins Creed Unity as the animation is fast and diverse allowing for smooth and entertaining movement using simple inputs as X + right bumper would allow the player to parkour upwards and B + right bumper would allow the player to parkour downwards.

As stated by (Terry Clinkenbeard, 2017) a responsive and fluid movement system and a system with input lag are the main differences between a fun game and a frustrating one. Players should feel autonomous in their actions as this allows them to react and enjoy their environment without overthinking. (Terry Clinkenbeard , 2017) used Titanfall as an example and described the movement system as “Tony Haw’s Gunfighter “ as the player can learn how to string together movements to traverse through the world. Smooth gameplay such as this does not become stale as the sense of freedom will have players wanting to pick the game up a second time. Games should allow players to execute moves that are difficult or impossible to do in the real world as (Terry Clinkenbeard, 2017) suggested expanding beyond movement limitations in the real world will make the player feel incredible after pulling off complex movements.

## History of video game movement

### 2D Games

Tennis for Two was one of the first games ever released. Built in 1958 by William Higinbotham, it simulated a game of tennis (De, Cruz, and Ryan, n.d.). A ball would move across the screen, and the player would use a controller to hit the ball back over the other side of the court which will be launched back at the player's side. There were a lot of technological limitations due to the current advancement of technology achieved during the ’50s which consisted of huge computers made from a large array of vacuum tube amplifiers which were able to perform a wide variety of complex mathematic operations(De, Cruz and Ryan, n.d.). Although it is not possible to create a game of today's standards on one of these machines William Higinbotham still created a fun game with very basic movement, the movement was created by using math equations to find the trajectory of the ball and the velocity to measure the impact of the hit and where the ball will land as the key movement mechanic of the game is the ball itself.

In 1985 a console game called Super Mario Bros. developed by Nintendo was released for the Nintendo Entertainment System (Ray, 2019). Mario is a side-scrolling 2D platformer which has a unique but very fluid movement system in which the player accelerates to build up speed and this can be affected by the player pressing the opposite direction to cause Mario to skid to a halt (Patel and YouTube, 2018). The movement provides much control as the player controller utilises gravity and velocity to calculate when Mario reaches the apex of his jump to then add gravity bringing Mario back down, (Patel and YouTube, 2018) explained that this was to help players have an opportunity to crush enemies quickly or reach narrow platforms between gaps. Compared to predecessors this complex movement was a staple for the game industry for a while as Super Mario Bros was successful selling around 40 million copies (Ray, 2019).

Pac-man was a game published by Bandai Namco for arcade cabinets and table arcades alike in the 1980s. The movement in this game consists of 90-degree turns in a maze-like level in which ghosts chase the player, the movement is basic but flows through the session with no issue. The key mechanic which influenced the feeling of the game and movement was the tension caused by the player being cornered during a chase, on an article titled “Pac-Man: The Joy of Movement” the writer (Smith, 2024) describes the experience as a delightful nightmare and expressed that the movement of Pac-man created tension against the limits of a maze and the ghosts make the tension juicier.

### 3D Games

Mario 64 was released by Nintendo for the Nintendo 64 console (Goodall, 2021). Mario 64 was one of the games which marked the transition from 2D gaming to 3D gaming, technology was becoming more advanced being able to render 3D objects in virtual spaces. The movement in Mario 64 was considered to be excellent for its time as the character movement works on input intensity meaning depending on how far the analogue stick is pushed the player transitions from walking, jogging and sprinting which allows for precise movement control which players agree the game feel of Mario 64 is great (Swink, 2007). An interesting point was explained on Steven Swink’s article (Swink, 2007) they found that the developers of Mario 64 followed a process to ensure game feel felt great in their games which was called The Gardens Ecosystem which consisted of 6 pieces of game feel. Input represents how the payer can express their intent In the virtual space, Response represents the responsiveness of the system processes which respond to player input, and Context represents how constraints give spatial meaning to motion ((Swink, 2007), Metaphor gives emotional meaning to motion to mitigate learning frustration and Rules control variables to give additional challenges and more meaning to motion and control.

Spider-man 2 for the PlayStation 2 was developed by Treyarch in 2004. This game had an excellent example of a movement system due to the control the player has over Spider-man, webs attached to buildings which gave players the ability to zip around buildings with precise control and the momentum of each swing carries the flow of the movement. The wide control over the player controller created a sense of a high skill gap in which players would learn to combine mechanics to traverse Manhattan (Lectril, 2015). Spider-man’s character is known to get from point A to point B rather quickly so to replicate this in the game the developers had mechanics to help the player go from zero movement speed to high movement speeds by introducing mechanics such as the slingshot which sent the player flying through the air building momentum (Lectril, 2015). These mechanics made the player feel like the character they were playing; thus, this game achieved a great level of the game feel which still brings older players back even after all these years.

## Factors supporting movement

Game optimisation is the process of refining a game so that it can run as efficiently as possible to achieve maximum performance on the desired platforms(Polydin, 2023). Performance affects everything within a game including movement so it should be obvious that game optimisation is important for a successful fluid movement system. Optimisation also plays a role in immersion because if the game does not run smoothly the player will notice and the player experience will be tainted (Polydin, 2023). As well as optimisation other factors including speed and control are key to creating a good-feeling movement system, the factors tie into the creation of manoeuvrability(Adderley, 2020). Manoeuvrability dictates how well a player can turn when moving at certain speeds and how quickly the player can accelerate and decelerate over time (Adderley, 2020) so if these factors are unbalanced the player has no control whatsoever.

In the game Dying Light 2, there is an intentional change in gravity that players describe as ‘floatiness’ This feeling is used to help the player think about their next move (Techland, 2022). This feature is helpful because the gameplay in Dying Light 2 is fast and many challenges must be overcome so extra airtime makes the movement system more accessible. Touching upon physics again, physics is an important factor of movement as this controls how a player is affected by weight and momentum, (Ipacs, 2023) suggests taking real-world values into account will help immerse the player into the experience by building up believability.

Exaggeration of physics is when the game breaks some laws of real-world physics (Ipacs, 2023), but this allows for games to be more fun for example the double jump is fun because it provides a way of moving around quickly but is impossible in the real world. Running has a weight behind it so if the focus of a game is to be running around a sense of weight is a factor which caters to immersion but in games, weight cannot be created only simulated. (Taylor, 2021) made an example of which weight is simulated as the way developers can do this is with higher gravity, lowered velocity and visual/audio effects, this feels good because the player can almost feel their characters' movement which can cause adrenaline to be released, this is proved by (Young and Young, 2014) as games can affect the fight-flight response.

## Design methodologies

Mirrors Edge is a game which has inspired countless others such as Dying Light since its release, it is a game which simulates first-person free-running and the aim was to feel like a third-person adventure game translated into first-person (Aberg and Dahl, 2009). Mirrors Edge was designed in a way that rewards momentum (Aberg and Dahl, 2009) explains that the player is forced to assess their situation to either stop and fight enemies or keep the momentum and begin the chase, this also creates a sense of presence making the player feel chased.

Motion blur is an asset well used in this game as it creates a sense of heightened senses and so is the use of colour, (Aberg and Dahl, 2009) describe the use of red helps players understand the characters' perspective of the world hence why objects are highlighted red to help players focus on what is relevant in their environment and everything becomes peripheral(Aberg and Dahl, 2009).

 The way developers tested their movements was by creating a white room, (Aberg and Dahl, 2009) described it was a fast-loading level used to debug all their features including animation to not need an animation editor, this was a valuable tool especially for testing as this room helps narrow down issues without extra clutter. (Aberg and Dahl, 2009) made it clear that the key area of focus was movement, all moves were iterated 5-6 times and developers made sure to pay close attention to detail to give the moves the quality they needed to provide a fluid experience.

After an interview conference about the development of the game a question about root motion control vs physics-driven input controls, (Aberg and Dahl, 2009) explained that wall-running started off using root motion control, but this was deemed to be too restrictive, so this system was then used physics-driven movement for a believable transition of movement, this also includes the pivoting of the camera whilst running around corners. The player camera is very advanced as it creates the feeling of controlling an actual human head for example when the player breaks their fall with a roll the camera follows the player and the camera has constant small movements to mimic a person (Aberg and Dahl, 2009).

Designing a climbable environment is a complex task especially when dealing with a parkour system, the game Dying Light went through some design processes to get around this problem by creating hooks around the game's environment (Binkowski et al., 2016). The hooks were used to indicate which areas of the map were climbable and the design goal was that anything that looked climbable was climbable (Binkowski et al., 2016).

This approach then became a problem due to the amount of memory taken up so the lead programmer Bartosz Kulon came up with a system which scans the environment checking for interactable objects based on certain criteria to determine what actions can be performed i.e. climb, jump etc(Binkowski et al., 2016). When testing Dying Light’s movement system a problem with motion sickness was found as their tester was left feeling ill after their experience, (Binkowski et al., 2016) explained that the developers worked on levels of motion blur, adjusting camera movement, filters etc to try and deal with the motion sickness which turned out to help.

Ghost Runner is a fast-paced FPS game with a fluid movement system, the design focus during experimentation flipped between level design and game design to ensure the environment worked well with the mechanics ensuring the size of gaps and platforms were suitable etc (Game Industry Conference, 2023). The level design is based around the movement system, what this means is the platforming sections aim to incorporate a combination of moves as demonstrated by (Game Industry Conference and Wabik, 2023) some levels follow a pattern as the player is shown wall-running, sliding, jumping, dashing and then back into a wall-run, this sets up a fluid route for the player to follow.

When developing an encounter with the warden enemy players did not enjoy the way it was designed because the developers broke a design convention, (Game Industry Conference, 2023) discusses that the warden enemy is fast and forces the player to halt movement to safely react, but this interrupted the flow of movement. These design conventions are as follows explained by (Game Industry Conference and Wabik, 2023) levels should be built around movement, and levels are divided into small sections with checkpoints and a mix of platforming and combat. But even without enemies present in big arenas the environment should always be fun to play as the speaker (Game Industry Conference and Wabik, 2023) explains the structure of the level is like a skatepark, this way the level should be fun to move around obstacles with little to no dead ends allowing for environmental loops.

## Animation

Traditionally in games the gaming industry handles animations using static animations, this refers to character movement an artist has drawn particularly for a required action either through drawings or motion capture, these animations are predefined(Zucconi, 2017). These animations typically require a large amount of storage so another way of creating realistic animations is by using procedural animations, a common way of achieving this is by simulating physics (Zucconi, 2017). Many games have implemented a form of procedural animation for example the likes of Grand Theft Auto use ragdoll physics when the character falls over which saves on creating a separate animation, ragdoll physics successfully simulates how a person might fall (Zucconi, 2017). But ragdolls are highly unpredictable typically resulting in humorous behaviours (Zucconi, 2017) or sometimes causing a player’s loss in some cases.

Rigid body animations are programmed movements in which the character walks by itself without the need for predefined movement using the same constraints as a ragdoll(Zucconi, 2017). The problem with ragdolls is that they lack plausible mobility so with rigid body simulations the character's limbs are forced in a way to simulate movement procedurally (Zucconi, 2017). A good example of rigid body simulation is the game Grow Home which (Wiltshire, 2016) talked about how the game used to be a failed development tool which aimed to make the design process of games easier by rapidly prototyping characters without getting animators involved. This resulted in a “drunk” "-looking movement in which one could walk and grab objects (Wiltshire, 2016), but to ensure that the character felt good to play senior developer Andy Buck stated they used “Little Nudges” (Wiltshire, 2016) implying some predefined animations were used to straighten up the procedural animations.

Rigid body simulations allow for simple movement but do not accurately simulate realistic movement as rigid body simulations only consider things such as mass and gravity, but lack any contextual knowledge (Zucconi, 2017). Inverse kinematics solve this as it can find out how the character can reach a target and force them into a desired stance (Zucconi, 2017), this allows for characters to do things such as naturally interacting with a desired object, this is done by using inverse kinematics to find the most natural way an arm joint can move (Zucconi, 2017). Unity has an animation tool called Mecanim which allows developers to utilise inverse kinematics, this tool provides character configurations using scripts and the animator controller to set up inverse kinematic animation (Technologies, n.d.), the scripts are used to set up the target positions and rotations of the character needs to make to complete an action.

## “Game Juiciness”: How does it influence the game's feel and player competence?

Game juiciness refers to feedback across all the communication channels such as visual, haptic, and audio feedback (Hicks, 2024) but this also details the coherence of game mechanics, theme and looks (Hicks, 2024). Examples of juiciness in the real world include electric cars as they produce different sounds compared to other petroleum engine cars and to enhance user experience designers added sound effects or “juiciness” to make the experience of driving better (Hicks, 2024). A research study was conducted to test how gamification and juiciness could affect player competence while playing Predator! This virtual reality title was used as part of the research study and findings found that only juiciness alone affected player competence positively during gameplay (Hicks et al., n.d.).

This is because game juiciness makes the player feel powerful and in control (Kao, 2020) which in turn provides a boost in confidence, confidence has been proven to increase performance in humans for instance in athletes’ self-confidence facilitates concentration, goals, an increase in effort and more (Trine University, 2023). This proves that if the player feels good about their gameplay by receiving juicy feedback from their actions, their performance will increase through confidence and juiciness. It also helps coach the players through the game's rules, letting them know how they are doing per interaction (Kao, 2020).

In an interview with Kieran Hicks, he was asked about this finding and he explained if a participant were to play a base version of a game with no juiciness and then played a juicy version, the juicy version would feel better to play because they have experienced the worst version (Hicks, 2024). The combined version (gamification and juiciness) did not add to player competence because gamification adds extrinsic goals such as badges players have not achieved yet or levelling up a character etc. In contrast, juiciness does increase competence (Hicks, 2024).

Juiciness can be used as part of many areas of a game, this is because juiciness does not have to be positive. For instance, in the game Dark Souls, when the player dies the screen fades to black with text stating the player has died (Hicks, 2024) described the sound effects to be almost oppressive to make the player feel negative feedback this adds weight behind the player's death encouraging the player to become better at the game. Focusing on the components that make games juicy for instance audio, designers when creating juicy audio have this idea of creating a sense of synaesthesia which creates audio which feels good to listen to for instance (Hicks, 2024) noted that the game Destiny they use a full frequency spectrum which makes the audio feel juicy.

When it comes to creating a sense of visual juiciness for instance applying juiciness to animation or adding particle effects, the 12 principles of animation Disney closely rely on can be used in game development. The 12 principles include exaggeration of movement, squash and stretch etc these principles are relevant as the goal of these principles is to make the animation feel good to watch (Salsberg, 2019). Looking at the 12 principles is relevant to juiciness in game development because as (Hicks, 2024) described it helps with the process of applying juiciness by guiding the process in a more structured way. For instance, the game Mirrors Edge adds visual juiciness by using the camera as it functions to act like a human head so when the character rolls the camera follows the flow of movement.

Although juiciness refers to feedback from actions (Hicks, 2024) mentioned that juicy games use ambient feedback elements for situations such as players leaving the game idle. Developers add in certain idle animations or wind blowing In the background so that even if an interaction does not take place things can be implemented to make the player feel great.

## Comparison of game quality between Indie Devs and AAA Companies

The AAA game development in today's era is quite advanced as developers can create games which when compared to older titles look extraordinarily better. AAA developers have a high source of budget which means these companies can hire hundreds of developers to work on a given project which at first seems like a great step up for the history of the industry (Kovanto, 2013). Although a higher budget would be expected to produce higher quality products, the facts are that AAA studios are more concerned about making money which makes aspects such as the quality of player controller have a significant quality loss. The difference between indie studios and AAA studios is the level of creativity and freedom developers have with the project, for example, indie developers can focus on aspects the team deems most important so if they want the best movement they can focus on that thing and make it perfect. AAA studios must follow tasks set up by the company and are managed by lead developers who guide staff on their jobs for that given development period (Kovanto, 2013).

## Procedural camera animations (add to recovered version)

As previously discussed, procedural animation is animation which has not been hand-animated by an animator, meaning the animation is created through code or by using animation curves. Procedural animation is a great tool for game feel as it can be used on the player's camera to create realistic movements such as swaying arms when walking or simulating head movements by moving the camera. In order for the animations to play there must be a way to track the state of the player, so the camera manager knows when to play an animation (Lewis, 2019) for example a Boolean can be used when the walk button is pressed to check if the player is walking.

The complex part is applying the procedural element to the animation as the position of an animated object must be calculated for the correct results for example some values such as transforms are manually configured such as the target location. A blog written about how procedural animation can be used for first-person aiming animation (Lewis, 2019) discussed an example of code which checks if the player is in the aiming state and lerps from the non-aiming state to the players' sightlines and manually entered locations using the hands alpha and then applies this to the transform location variable, to reverse this the opposite is done and a check is put in place to make sure the animation does not continuously play.

Unity has plenty of documentation for camera animations as their Cinemachine is a game object which is used as a placeholder for the standard camera object which can be manipulated using scripts for purposes such as camera shake, responding to inputs etc. Unity refers to these animations as procedural motion and comes with behaviours and extensions to help developers create dynamic cameras. Position and rotation control manipulates the camera position and rotation (Unity, 2024) which allows developers to follow a target which can include third- or first-person player controllers or targets with fixed offsets.

Noise behaviours refer to applying shake to the camera to simulate realistic physics responses for immersion, For each frame update the Cinemachine can add noise which is separate from the camera movement which follows a target which means noise does not affect the camera's future position (Unity, 2024). Cinemachine also comes with several extensions which can be useful for purposes such as moving the camera out of the way of game objects which bloc camera view (Unity, 2024).

Using these tools developers can create convincing player cameras which can add to the game feel of a project massively, bad camera animations can cause plenty of issues down the line so if left underdeveloped or neglected entirely a company can lose its player base as terrible cameras ruin a game’s experience.

## State Machines

A state machine is a behaviour model which consists of a finite number of states (AG, n.d.), also referred to as finite state machines (FSM) based on the current state and a change in input from the player or another system such as AI decision functions the FSM is responsible for transitioning between states which produce new outputs (AG, n.d.). States can be switched using multiple methods including utilising timers to trigger a state such as the player falling, as we can time the moment the player leaves the ground and use a variable to track a falling threshold and with these conditions met the state can switch.

State machines which only have one transition from an input are referred to as deterministic FSM meaning one can determine the next state the machine will transition to (tutorialspoint, n.d.), fundamentally the output of a state machine is the final state. This means the machine goes through all the processing and reads the final state which in turn follows with an action, the machine does not do anything during the transitions between the states (Shead, 2018).

Non-deterministic state machines refer to FSM’s which given an input can lead to multiple different states. Problems start to arise when trying to manage multiple different outcomes because sometimes the machine does not know which path to take, so a way to solve this issue is through backtracking. Backtracking is the process of taking all possible outcomes and then ignoring or backing out of paths which end up blocking transitions, research compared this method to chess machines as they consider the different possibilities of the match and choose the path which returns the greatest number of advantages (Shead, 2018).

Further studies in automata theory suggest there are two basic types of state machines, one of the machines is referred to as the Moore Machine (AG, n.d.) created by Edward F. Moore in 1956. Moore machines are finite-state machines whose output value depends on the current state and is not determined by any inputs. For example, a state machine for a simple light switch which produced different brightness levels would have an on and off button which when the on button is pressed the light transitions from different brightness levels and when off is pressed the lights turn off (AG, n.d.).

Mealy machines were created by Geroge H. Mealy in 1955 and in comparison, to Moore machines the Mealy machine output results based on its current state and current inputs instead of solely the current state, mealy machines are deterministic meaning only one transition is possible (Wikipedia Contributors, 2019). So, for the light example, we would be taking into account the on-button pressed event and the brightness level of the light allowing us to transition from one brightness level directly instead of pressing on multiple times to transition.

# Research Methodologies

Throughout the research process of the investigation many sources will be utilised to solve the problem of bad movement in games, these sources include but are not limited to previous research papers, previous thesis or dissertation papers. Sources used vary in presentation this should be considered and separated into sections of importance, this is because some sources include reviews of games to grasp how players feel about the state of movement in games, but more formal sources should be used to provide concise insight into the issue. Currently, research has been conducted into movement by using a thesis which discusses the importance of control and speed of movement in games as it covers the main key factors of a good movement system, this helps the investigation as this provides references used by this individual which could be used as a good starting point. Combining this research with IGN reviews to grasp players' perceptions provides a broader insight into how players want to feel when playing games and how movement should be handled. Design documentation will be desired, if possible, to retrieve as these papers hold the processes developers went through to achieve the produced game, these processes could not only help research how good movement is designed but also help with the artefact.

Before working on the artefact some planning will be required to flesh out a design document to help guide the project. The design document should include the breakdown of the required mechanics, this breakdown should simplify the mechanics down to how the mechanics should function and what variables or functions are needed. Diagrams could be used to help visualise mechanics and what checks are needed for each interaction, this is necessary as individuals who review this investigation will need to clearly understand how the movement system works to grasp the goals of the investigation and to understand the importance of movement in games. Project management practices will be used to help make sure the investigation is on track. Gantt chart has been produced to set tasks in order of importance to help track goals which must be completed and goals which should be left for late-stage development for example optional mechanics can be implemented closer to deadlines.

The engine utilized for the development of the artefact will be Unity version 2022.3.16f1, this engine is being used because Unity has tools which can be used to make the development of the movement system easier for example Unity has a better movement system which can be modified. Many indie developers choose to use Unity for example the developers of Neon White used Unity to create their game, this is because Unity is flexible and easier to use compared to larger engines like Unreal Engine. Unity also has a large community which means that problems and solutions can easily be researched because there are lots of tutorials and forums which could help the development of the artefact. Unity is also being used due to the skillset currently possess as this engine has been used myself amounting experience of 2 years, so with confidence Unity is the right engine to use to create the project.

During development, many challenges will be faced when creating the movement system and to overcome these challenges sources such as videos, forums and Unity documentation will be used to help understand how to approach the challenge. All sources used must be recorded to reference where the information came from, tutorials will be used but must not be followed as this project must reflect the findings of the investigation and not someone else’s findings.

Once the artefact has been developed into a minimal viable product standard project testing of the solution can occur to identify problems within the code, and each error should be recorded to keep track of the development process. Personally, regression testing will take place during user testing after a change or improvement has been implemented, this will make sure that the updates have not affected the artefact’s functionality in a way that hinders user experience. GitHub will be utilised to ensure version control practices are taking place so that if an error breaks the project, previous versions will be preserved in the event of any issues which may harm the investigation.

Testing will also be asked of volunteers who will be asked to conduct some play testing and white box testing to help find hidden problems that may ruin the experience, these volunteers will be emailed a questionnaire to collect data about potential bugs and artefact effectiveness. The data collected is justified because all playtesting will be conducted anonymously to ensure the privacy of participants and questions will not try to collect personal data as data collected will purely be for the betterment of the artefact. The project test build will be shared through emails and participants will be fellow students personally known.

After testing the artefact errors should be reported by both me and the participants, reports may not exactly lead to the problem as some participants may not understand programming, but all reports will be analysed and investigated for the betterment of this investigation. Once reports are received or an issue is found immediate action will be taken to deal with the problem swiftly and keep track of progress, if the problem is complex research will be done to learn how to solve the problem and the process should be recorded. Throughout the development of the artefact, the quality of the mechanics will change with time, the first iterations will have obvious issues such as the movement being unstable at times or miscalculations of vectors. Understanding this the artefact should slowly be improved with time, the way improvements will be researched like how the programming solutions will be found as other developers would have struggled similarly with similar issues and that knowledge can be adapted.

The style of testing to be carried out during this investigation will be the Mann-Whitney U test, which will allow for the collection of non-parametric data (McClenaghan, 2022). This means the test will be divided into separate groups, these participants will have different versions of the artefact to test to collect various sets of data which can be compared. Visualising the results on a graph would show the skewed data (McClenaghan, 2022), this opens up the results for reflection and identifies the strengths and weaknesses of each version.

For this to work properly the number of participants required will be 6 – 10 participants as this would be an even split in the data, this should simplify data analysis after the testing period. The main difference between versions should be the game feel, this means anything that makes the game feel good should be removed from one version. Too many differences between the versions could dirty the data because this could lead to similarities between versions being lost, the games should be similar but different perhaps new bugs will be found between versions.

If these requirements cannot be met, different testing will be done to run a similar test to collect data like the data that was needed. The same data can still be collected but if time becomes too precious to waste, then the alternative of fewer participants but the same differences could be the option to take.

# Results and Findings

## Development Lifecycle

During development, the project had many changes, including different approaches and many cases of code being refactored or completely remodelled. The goal of the artefact was to showcase an industry-standard minimal viable product artefact which would represent how a movement system for a free-running style game should feel and what systems are being used to make that experience. This project also showcases many attempts to enhance the experience using a few different means including introducing state-based animation and programmed animation, These attempts were rough but well-adjusted enough to improve the feel of the game as without them the movement would not feel as impactful or enjoyable.

### Coroutine Approach



Figure 1: Coroutine Logic

The first approach consisted of using coroutines to control the behaviour of the player this meant using the new input system Unity introduced to assign an input action to the player which would read player inputs from the keyboard. As seen in Figure 1 the coroutine when started would check a while statement if applicable, then if the condition was met force would be applied to the player if the conditions were not met the coroutine would end.

This was a simple approach which was easy to implement and test into the artefact. Over the first weeks of development, movement was created to a very simple degree which consisted of adding force to the player when a key had been pressed by reading the value of the callback context. At first, this system worked fine the player would as expected jump when told and move to where they needed to be, so the logic of basic movement was correct, but despite this, there were problems when introducing more complex behaviour such as wall running and climbing.

Introducing wall running/climbing was a great challenge as a lot of complicated mathematics was used in the process of developing the correct behaviour the surface area of a ray cast had to be calculated to find not only the dot product off the wall which dictated the player wall running left or right but also the players forward dot which calculated the forward vector of the player to check their intended actions, these intentions need to be checked as if the player is facing directly at the wall then they intend to run up it, but if they're facing to the side we need to check if we are closer to the left side of the wall or the right side and then activate wall run in that direction.

Problems started to occur because unpredictable behaviour was starting to happen for example if the player wanted to run up the wall sometimes, they would erratically be shot up the wall, be shot in the wrong direction by getting confused about which method should be running or nothing would happen. A lot of time was spent on this problem as the solution was hard to find but suddenly a solution was found which solved the problems occurring but also brought new challenges. These new challenges included refactoring the behaviour and how it worked, and the time constraint already put on the project as by this point there already wasn't enough time to waste.

To make sure that the best artefact is created from this research a decision was made to go through with the refactoring and remodelling of the code base which has proved to be highly beneficial and knowledgeable.

### Finite State Machine(FSM)



Figure 2: Finite State Machine

Finite state machines are a very simple concept once broken down, it consists of setting up enumerations which represent each state that the player can enter when a condition is met. As seen in Figure 2 the state machine runs on a method which checks the different states, it would check different states because the program needs to find out if the current state is equal to the previous state because we want to stop any behaviour which should not be running as this causes clashes with code or could cause unpredictable behaviour. If the current state is not equal to the previous state, then it would check if it can in fact change states in the first place, the check for changing states has been encapsulated into another method but on the diagram, this method is represented as a decision due to its Boolean nature.

When a verdict has been reached the state machine checks if the current state equals the state being used and it also works out if the coroutine for that behaviour is equal to null as we want to find out if the current coroutine is already being used. Once these checks have been fulfilled the coroutine attached to the decision made will run and then the behaviour stored in the coroutine will apply the action.

Creating a state machine has proven to be highly optimal when it comes to running the program optimally as the state machines allow the behaviour to be switched dynamically with complex decision-making. Using coroutine alone was not enough to make a convincing player movement script because, without the state machines, the program does not know when a specific behaviour is running so we cannot create a condition to stop behaviour that should not be running. Booleans were used to try and simulate this dynamic behaviour switching, what this means is some variables called “is moving” etc. were being checked on and off to try and stop the coroutines when we wanted them to end but this caused issues as some of them would flicker on and off when they shouldn’t, and others would not work at all.



Figure 3: Timers for state cooldown, grounded cooldown and falling condition

For the state machine to work as predicted timers were also used in the development of this system as we had to check when a player left the grounded state or when the player was falling. We would also use timers to check what state changed to add a cooldown to prevent other states from overwriting each other, as seen in Figure 3 there are multiple examples of timers used for different purposes.

The first time represents the state cooldown, this was created for each state to wait a couple of milliseconds before switching states to prevent overwriting and unpredictable state changes. How this works is it simply checks when the last state was changed and checks again if the state can be switched, then we do mathematics on the state checker Boolean, which checks if the cooldown has started. The timer is then reset in the state to make sure the cooldown does not get in the way of activating that specific state.

The next represents the ground check, we are checking the last time we were grounded by making it equal Time.time this is because we want to time when the player has left the ground, if the last time the player was grounded is greater than the ground check delay this must mean that the player has left the ground.

Finally, the last timer used in this program is the fall timer, as we want to check if the player is falling below a certain point on the Y axis, and we also want to check if the player is grounded. If these conditions are met, we apply the state change timer and setting falling to true as if you're below that threshold you are falling. But if you're grounded, we are setting the time on the ground to be time delta time as we want to track how much time the player is spending on the ground to see when exactly you are leaving the ground. Once this condition is finally met you are no longer falling so falling equals true and the time on the ground is reset to 0.

Although we are making a new time of four falling, we cannot use the same timer we used for grounded as this has been tried in a different test build of the game and it does not prove to have optimal behaviour, so a new one was created for this purpose.

## Testing Findings

Due to the time left to develop the artefact the testing conducted was not the same test as the one previously planned. A compromise was created and put in place to receive well-needed feedback; fewer people than expected were available to do the tests as only 4 different results were collected but useful information was collected. By creating a point system based on the positivity of keywords used in feedback a graph was created to show the rate of change in user satisfaction between different versions of the same artefact.



This visual display shows how distinct the changes between the two versions tested are as the satisfaction of the participants using the good version of the artefact was significantly higher than users on the bad version of the game.

The lines tracking the rate of change make sharp changes in direction between each participant's experience which could mean there are major flaws present in the bad version.

Although participants had issues with one of the versions of the artefact, the good version also had problems which impacted satisfaction.

Mechanical differences were pointed out between versions suggesting the existence of inconsistent data which may have affected the results of this graph.

Figure 4: Positivity Graph



Figure 5: Keywords Collected

Above are words used within the feedback form, these words are keywords which identify the positivity level of each comment made while testing. Positive words are added to a score and Negative words are subtracted from that score which creates even results easily analysable, some of the keywords represented describe how participants feel about the state of the game and the direction development is headed.

A participant claimed they were hopeful as they had excitement while testing the game stating they look forward to the next version of the artefact. Somewhat positive comments such as these highlight the artefact does indeed make players feel positive while playing it, other participants described the artefact to be exciting and freeing. These very positive comments were very good to see as this artefact aimed to create a freeing player controller to make the player feel excited while moving through the environment. Participants feeling a form of this proves the success of the development of the artefact.

Although the artefact was positively received participants reported a lack of control at some point while playing through the level which would affect participant satisfaction as players should have consistent control to ensure movement accuracy. Another participant noticed the movement was sometimes inconsistent stating the character sometimes felt robotic, this points out a major flaw as the movement has been reported to be fluid but also not fluid.

After testing many major problems and major positives about the artefact have been pointed out. The key areas which caused issues were the jump mechanics and wall-running mechanics as they had inconsistent functionalities meaning they sometimes worked and other times did not, movement was fluid at times but must be fluid throughout the experience. Participants did enjoy the experience and enjoyed the aspects of the game such as the feeling of being free and the smooth gameplay which showed the strengths of what was achieved within the given time.

A noteworthy comment one participant kept mentioning was the environment and the texture. They noticed that there were not many elements within the artefact that conveyed movement, this lack of information made the player sometimes feel as if they were not moving at all which would throw them off breaking fluidity. The participant suggested using decals, adding objectives or more interesting environments as this would have helped the artefact succeed more in the long run.

The received data was hypothesized to be skewed data as the expectation was for the data to be lopsided. The data received included values of 3,2,-1,-3 which is not an example of skewed data as the data does not show any positive skew or negative skew meaning the data received is balanced or bimodal.



This graph is another way of looking at the data as now the negative data is represented as larger blocks making the distinctiveness between versions more visible and representable.

From this graph you can see the data overall is quite balanced as the values all round up to a value of 1. If rounded to 0, the data would suggest perfect balancing of feedback.

Although the data is useful it would have been more beneficial to have had 10 participants or 6 as the data may have been skewed.

Figure 6: A new look at the graph

# Discussion and Analysis

This investigation has shown that creating an advanced player controller is more complex than it once seemed before conducting the investigation. From a technical standpoint, a good player controller is a script that runs optimally and produces smooth gameplay. Still, the most important aspect is the sense of motion and simulating this to create a more enjoyable user experience. Game juiciness makes a huge impact on the effectiveness of conveying the feeling of motion as this feedback immerses the player into the motion of their character, the tests conducted tried to create a similar environment to the research study conducted by (Hicks et al., n.d.). The findings within the tests found that versions without any effects to create feedback were not as enjoyable with players feeling a lack of control as seen in Figure 5 or the player experiencing a lack of direction meaning the character did not feel like they were moving.

Motion can be created through many aspects of game juiciness because there are many elements we expect when a character moves on the screen, footsteps should be heard with each step and should change, for first-person games head bobbing is a common feature attached to the player's camera. The research found by (Wiltshire, 2016) discussed how developers found ways to animate their characters without the need for animators during early testing, this finding was utilised within the artefact to achieve a head bob effect. This effect was found to increase user enjoyment and was described to help make the participants feel the act of moving within the digital world.

Applying animation was found to be an essential part of creating a convincing game feel, for example, the creators of Mirrors Edge utilised first-person animations by making the camera follow the player's head movement while also using a 3D full-body model (Aberg and Dahl, 2009). Participants referenced the artefact's use of animations, some claimed they enjoyed how the limbs sometimes moved in front of the camera creating a sense of control or movement which proved to benefit the project, but problems arise from using full-body models.

One participant noticed straight away that they could see through their body, the creators of Mirrors Edge (Aberg and Dahl, 2009) created a player controller which utilised their model more efficiently without looking through their body and if given more time this issue may have been fixed. Problems such as these break immersion the goal of the investigation was to avoid this outcome, but unexpected outcomes arise while dealing with vectors and other complex factors such as states.

A Finite state machine was created for the artefact solution to be able to manage the different states of the player without other behaviours interfering with method execution. The state machine could be referred to as a Mealy state machine because of the research conducted, a Mealy machine is stated to be a state machine whose output is determined by the input and current state, at most there can only be one transition at a time (Wikipedia Contributors, 2019). With this knowledge and considering each transition within the artefact is determined by the current state and the player’s input, this must mean the developed artefact has a state machine which resembles that of a Mealy machine.

The developed state machine did impact the game experience heavily because if the state machine did not work then the mechanics would not function as expected. Players reported some mechanics would either not work or function oddly, mentioning the wall-running would not work or would simply make the player float. These issues may have been due to the conditional checks either not returning the correct values and although the developed machine is deterministic there could be issues with states unsure when to transition as discussed in research (Shead, 2018).

The investigation collected many different concepts around the idea of movement in games, these concepts included the importance of control and speed, the importance of conveying motion through game effects and animations and much more. The goal of the project was to identify how developers can create systems to support the creation of an industry-standard player controller, this was achieved as we have discovered how complex movement is. Testing revealed how smaller details can add up due to negative and positive feedback with most stating the player felt motionless or jittery.

The main question of the investigation was why player controller quality differs from AAA games and indie games. The argument was that indie games create much better feeling experiences than most modern AAA company titles, we have found out that this statement is true to some extent. Findings suggest that bigger AAA companies often focus on creating large profit margins due to high costs and not meeting these expectations would cause the company to lose money and developers unpaid. This often leads to mechanics being misunderstood which causes neglect for mechanics which might seem like it does not impact the experience greatly, but the little things often do.

Indie developers often have smaller cheaper teams but this also means that the budget for assets is very low. Although these developers struggle with these factors, teams such as these often have time to work on the smaller details which ends up creating a much better product. Games such as Neon White produced some of the best movement systems seen in years, these games are built for movement because they are speed runner games which means the point is to be fast.

Overall, this project has done well. However, this investigation has not achieved all of its goals. For example, the artefact developed could not be considered industry standard because of the many inconsistencies with movement and the lack of further advanced mechanics or refined animation for game feel. This made the experience less enjoyable than it could have been possible, but what was achieved is comparable to a minimal viable product.

The question of the investigation was unclear meaning the research was not narrowed down to specific areas. The research seems inconsistent, meaning the topics jump from aspect to aspect as we discussed many technologies around animation and fundamental rules of movement but if narrowed down to specific elements of movement such as game feel, the goals of the project could have been more clearly defined. This project did meet some of the goals because an artefact was created which displayed an advanced controller but could not be considered industry standard, so we did not achieve all of them.

# Conclusion

This investigation has not been an easy journey, although at first researching the basic concept of movement and the components which transform it into a worthy player experience seemed simple at first glance. It became clear mid-way through that movement in games is far more advanced than once hypothesised, the question at hand was what makes up movement to create an experience that feels good to play and why indie titles which revolve around movement seem better in quality and game feel than higher budget AAA titles. The pathways of research were ambiguous as movement is an immensely large concept and must be narrowed down into key concepts which support the investigation, at first, I felt the research was branching out too much into different concepts and seemed to jump straight from vector calculations to kinematic animations.

These subjects have relevancy, and aspects were used from all research into the artefact, but further into the paper the research tries to narrow down to more relevant concepts, but if this was done sooner this feeling of over-branching might have been fixed.

This fact is not inherently wrong because we have achieved some of the goals set out at the start of the investigation. After all, I have proved how complex movement can be by taking a deep-dive approach to try and break down these systems. We found out how developers of games considered to have the best movement controllers overcame the challenges of creating believable systems. For example, the creators of Dying Light discussed how their system became more complex as their climbing system started from an anchor system to environmental decision-making (also ledge detection)(Binkowski et al., 2016).

A good observation is that the paper has a strong take on game feel, what I mean is this concept is mentioned throughout the entire paper as it is referenced on most headliner topics. A good justification to make for this is that movement cannot be considered well done if it does not feel great to play, this is referenced within [Importance of movement in games](#_Importance_of_movement) as systems which felt “clunky” do not feel great to play, although older games are iconic for their systems developed at the time.

Sections of the paper may seem irrelevant to some so to make things clear, some points are to be made. To understand the process games, go through to become refined and nice to play, I delved into some of the history of older games to discuss some of their systems to have some sort of perspective on older game systems and how the developers made the experience have juicy elements for better user experience. Making these seemingly redundant research sections tie into the investigation at hand opened perspective to the client side of the project, testers were expected to play the produced artefact and understanding the needs a user wants satisfied by a game through history was a good way of breaking down what the key aspects of the artefact.

Key aspects of the artefact were certainly the implementation of game juiciness mechanics as discussed within [“Game Juiciness”: How does it influence the game's feel and player competence?](Juiciness#_Game_) and the balance of speed and control of the player while dealing with vectors and possible physics as discussed within [Factors supporting movement](#_Factors_supporting_movement). This was made abundantly clear while testing the artefact, testers claimed on one version the movement felt motionless and was getting stuck on parts of the level which gave a sense of no control, while on others the gameplay felt smooth which made the experience better with positive feedback, but issues still were made clear. This highlights how important a sense of control over the character is as it can impact the experience of the user, the artefact developed reflects a flawed movement system which is developed well but could be better with further improvements.

Development was a struggle as the project transformed from basic concepts such as using coroutines to complex but simple to conceptualise finite state machines to dynamically transition player behaviour. This benefitted the project in many ways but the lack of direction of where to start and further mechanics to result in wanted behaviours hindered development and I was left stuck on problems for weeks. The mistake was not researching state machines sooner, but although I lacked direction even with the abundance of research, the produce state machine is not far off from completion. I have created a dynamically transitioning player controller, currently, it has experience hindering flaws but with some tweaks will become a fundamentally successful prototype.

Parkour mechanics were difficult to understand due to the mathematics involved in calculating surface areas within digital spaces as well as calculating intention from player direction. The wall-running and climbing mechanics created do work as intended, although this cannot be accurately demonstrated with the current state of the artefact I can be confident in stating the logic is sound. During my own individual testing over development these mechanics were working as the player glided across and up the wall in whichever direction the player intended to go, the problem which breaks the parkour mechanic are when we are trying to store a jump input to track when the player wants to wall-run through the jump mechanic.

The jump mechanic went through many phases as the problem with the jump was the revolving problems. My definition of a revolving problem is an issue which you fix once, progress further and come back where you started on a loop, many attempts were made to overcome this issue from Boolean checks to jump buffer timers. At first the button input was tracked with a jump-pressed check, once the jump is pressed this check becomes equal to true and once completed or the falling state is the current state this jump-pressed check becomes equal to false. The problem with this was the wall-running not being able to track this Boolean correctly as the jump-pressed check was always false.

To fix this a jump buffer timer was created so the raw input of the jump could be checked to store when the jump was pressed and keep this value available for other methods to read in as a check of their own. This idea is great as this could allow for double jumping or applying wall jumping as an extension of wall-running, but the execution ended up breaking the state machine transitions due to logic being not quite right and returning odd values. So, between artefact versions the jump mechanic needs to be adjusted in the future as this was implemented late in development which lead to the jump buffer suffering in quality.

The movement itself created for the project was good technically as it was smooth on the better version of the artefact as the player would run along the level with a basic animation and head bob script quite smoothly. The strength of the movement was the ability to walk around the level and how smoothly the transition of the states was as the main four states once worked perfectly including idle, moving, jumping and falling. The output of each state was as expected as idle would only transition if the player completely stops moving and even if the player falls without the jump, it can still be tracked.

The weakness of the movement was the ability it had to convey motion as I struggled with implementing game juiciness, tweaking the head bobbing script was a struggle as some testing reported back rubber-banding referring to how the camera sometimes had erratic movement. Another problem was trying to make the forces applied to the player feel natural, as this is difficult to accurately simulate and this is where the project fell short, not enough time was spent on the feeling of motion, which was the whole point of the investigation, this is because of the fixation I had when trying to get each parkour mechanic to work.

Animation was utilized as part of development as this was necessary to link back to my animation research to prove how relevant animation can be when working with first-person controllers. Without previous animation experience, I worked out how to use Mixamo using sources such as YouTube and experimentation which allowed me to apply basic animation which only got developed to where the run and idle work. Implementing the fall and jump animations was difficult due to trying to use similar checks to the actual methods used for the animation parameters but this led to behaviour falling apart.

Overall, we have found out that movement calculations are really important as the speed of the player can affect the control they have over the controller and we have also found out that a player controller without the feeling of motion cannot be considered a viable system to implement. Research goals were achieved as many questions such as what makes up a player controller movement system to simulate advanced movement because we know camera animations and juiciness such as motion blur or sound can drastically change the experience from mediocre to a positive version of its former self. This is further proved while looking at research covered by (Hicks et al., n.d.) and test results found in [Testing Findings](#_Testing_Findings) because we found out players found the experience lacking without the implementation of camera animation or other effects.

Another research goal achieved but with some holes was the comparison between indie developers and AAA company projects. We discovered why indie games can spend so much time refining their mechanics to produce a much better game, this is because AAA companies must meet certain profit margins for the huge project to pay off. The focus on marketing and making back all the money spent on the project led to some games suffering from problematic movements or poorly implemented solutions. The reason this section has holes is because there could be more information surrounding decisions made during development and a reason why the developers went with that approach to the problem.

In conclusion, not all of the goals were met as the outcome of the artefact does not represent what the investigation covers because of the massive flaws in the system I have crafted. This does not mean that this outcome is a failure but means it is a success in itself because of the information learned during both development and research time. The main lessons learnt include the complexity of movement systems and approaches used to create dynamic player behaviour using state machines. In the future finding key systems such as state machines should be the first focus as if the state machine developed had more time to be developed then the true outcome of a successful industry standard prototype could have been achieved.

Another lesson I have learnt is to approach the problem with more clear questions as this approach left me with ambiguous research goals which led to confusion and slowed the investigation's progress greatly. Next time I should create a breakdown of problems to properly plan my next steps to manage the amount of time between start and finish more efficiently.

# Recommendations

During my time with this project, I have learned many things including how to make a state machine and how player characters can be programmed optimally. If I were to do this project again, I would focus on making the state machine first before making coroutines as beforehand I had no idea that I would need a state machine, if I had done a bit more research into the development side of movement maybe I would have found the answer quick enough to notice this missing system.

The research is very flawed, what I mean is the research feels scattered which is not good for an investigation as this could lead to confusion or irrelevant information. This happened because of the ambiguity of the project title which derails the focus of the research, next time I would focus on a key aspect of movement or play a gameplay and work to improve that specific area of movement instead of looking at the vast variety of topics movement consists of.

Lastly, too much time was spent on debugging if I had spent less time worrying about a broken mechanic, I think I would have had more time to make the important mechanics count.

# Bibliography

Aberg, J. and Dahl, T. (2009). *Creating First Person Movement for MIRROR’S EDGE*. [online] www.gdcvault.com. Available at: https://www.gdcvault.com/play/1387/Creating-First-Person-Movement-for [Accessed 17 Nov. 2024].

Adderley , T. (2020). *Maneuverability in Games: The Importance of Speed and Control*. [online] Game Developer. Available at: https://www.gamedeveloper.com/design/maneuverability-in-games-the-importance-of-speed-and-control [Accessed 14 Oct. 2024]. (Adderley , 2020).

AG, itemis (n.d.). *What is a state machine?* [online] www.itemis.com. Available at: https://www.itemis.com/en/products/itemis-create/documentation/user-guide/overview\_what\_are\_state\_machines [Accessed 24 Feb. 2025].

Ahearn, N. (2008). *Mirror’s Edge Review*. [online] IGN. Available at: https://www.ign.com/articles/2008/11/11/mirrors-edge-review [Accessed 11 Oct. 2024].

Amanda (2024). *What exactly is ‘game-feel’? - Game Development Diary - Medium*. [online] Medium. Available at: https://medium.com/teeny-tiny-game-dev-essays/what-exactly-is-game-feel-f2eee9c5dbf5 [Accessed 26 Nov. 2024].

Binkowski, M., Kulon, B., MilewskiJune 02, P. and 2016 (2016). *Game Design Deep Dive: Dying Light’s Natural Movement system*. [online] Game Developer. Available at: https://www.gamedeveloper.com/design/game-design-deep-dive-i-dying-light-i-s-natural-movement-system [Accessed 16 Nov. 2024].

Bycer, J. (2015). *The Importance of Feedback to Learning in Game Design - Game Wisdom*. [online] Game Wisdom. Available at: https://game-wisdom.com/critical/feedback-game-design [Accessed 9 Nov. 2024].

De, A., Cruz, L. and Ryan, J. (n.d.). *Tennis For Two*. [online] Available at: https://web.mit.edu/6.101/www/s2016/projects/jackryan\_Project\_Final\_Report.pdf [Accessed 17 Dec. 2024].

Developer Nation Community (2023). *Unity, Unreal Remain Popular Game Engines Among Developers But What Do Indie Studios Use?* [online] Hackernoon.com. Available at: https://hackernoon.com/unity-unreal-remain-popular-game-engines-among-developers-but-what-do-indie-studios-use [Accessed 28 Oct. 2024].

Dorset, C. (2024). *LinkedIn*. [online] Linkedin.com. Available at: https://www.linkedin.com/pulse/designing-enjoyable-movement-systems-games-christopher-dorset/ [Accessed 9 Nov. 2024].

Game Industry Conference and Wabik, Ł. (2023). *Level Design of Ghostrunner - Creating Cyberninja Skatepark - Łukasz Wabik || One More Level S.A. |*. [online] YouTube. Available at: https://www.youtube.com/watch?v=UBln-JtD7hM [Accessed 17 Nov. 2024].

Goodall, R. (2021). *The Boar*. [online] theboar.org. Available at: https://theboar.org/2021/01/concept-to-console-super-mario-64/ [Accessed 18 Dec. 2024].

Hicks, K. (2024). *What is Game Juiciness*. 22 Nov.

Hicks, K., Dickinson, P., Holopainen, J. and Gerling, K. (2024). *View of Good Game Feel: An Empirically Grounded Framework for Juicy Design*. [online] Digra.org. Available at: https://dl.digra.org/index.php/dl/article/view/936/936 [Accessed 26 Nov. 2024].

Hicks, K., Gerling, K., Richardson, G., Pike, T., Burman, O. and Dickinson, P. (n.d.). *Understanding the Effects of Gamification and Juiciness on Players*. [online] Available at: https://ieee-cog.org/2019/papers/paper\_13.pdf [Accessed 9 Nov. 2024].

IMDb (2022). *Neon White (Video Game 2022) - Awards - IMDb*. [online] IMDb. Available at: https://www.imdb.com/title/tt20672664/awards/ [Accessed 25 Oct. 2024].

Ipacs, D. (2023). *Game Physics: Everything You Need to Know*. [online] Bluebird International. Available at: https://bluebirdinternational.com/game-physics/ [Accessed 9 Nov. 2024].

Iris (2023). *Forward and Inverse Kinematics: Explained*. [online] irisdynamics.com. Available at: https://irisdynamics.com/articles/forward-and-inverse-kinematics [Accessed 28 Nov. 2024].

Issuu (2022). *Dissertation Participant Information*. [online] Issuu. Available at: https://issuu.com/hwdissertation2022/docs/participant-information-sheet.docx\_14c8b24c0b6084 [Accessed 16 Dec. 2024].

Kao, D. (2020). The effects of juiciness in an action RPG. *Entertainment Computing*, 34, p.100359. doi:https://doi.org/10.1016/j.entcom.2020.100359.

Kovanto, A. (2013). *Business Information Technology, Game Programming The Improvements for Indie Game Development*. [online] Available at: https://www.theseus.fi/bitstream/handle/10024/55620/Antti\_Kovanto\_The\_improments\_for\_Indie\_Game\_development.pdf?sequ [Accessed 18 Dec. 2024].

Lectril (2015). *Locomotion and Momentum in ‘Spider-Man 2’*. [online] On the CRT Screen. Available at: https://onthecrtscreen.wordpress.com/2015/07/07/locomotion-and-momentum-in-spider-man-2/ [Accessed 18 Dec. 2024].

Lewis, K. (2019). *Designing an Immersive FPS in Unreal Engine 4! Part 3: Aiming Down Sights with Procedural Animation...and Math! - KARL LEWIS*. [online] KARL LEWIS. Available at: https://karllewisdesign.com/unreal-immersive-fps-part3/ [Accessed 9 Jan. 2025].

McClenaghan, E. (2022). *Mann-Whitney U Test: Assumptions and Example*. [online] Informatics from Technology Networks. Available at: https://www.technologynetworks.com/informatics/articles/mann-whitney-u-test-assumptions-and-example-363425 [Accessed 22 Feb. 2025].

Patel, A. and YouTube (2018). *The Satisfying Motion of Super Mario Bros | Game Feel Deep Dive*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=EwONt4r2rMM [Accessed 17 Dec. 2024].

Piquet, L. (2012). *[Breakdown] Shadow of the Colossus (PAL - PS2) | Froyok - Fabrice Piquet*. [online] www.froyok.fr. Available at: https://www.froyok.fr/blog/2012-10-breakdown-shadow-of-the-colossus-pal-ps2/ [Accessed 28 Nov. 2024].

Polydin (2023). *The Essentials of Video Game Optimization*. [online] Polydin. Available at: https://polydin.com/video-game-optimization/ [Accessed 25 Oct. 2024].

Ray, M. (2019). Super Mario Bros | electronic game. In: *Encyclopædia Britannica*. [online] Available at: https://www.britannica.com/topic/Super-Mario-Bros [Accessed 17 Dec. 2024].

Red Apple Learning (2023). *Red Apple Learning*. [online] Red Apple Learning. Available at: https://redapplelearning.in/list-of-disneys-12-principles-of-animation-you-need-to-know/ [Accessed 26 Nov. 2024].

Reilly, L. (2022). *Neon White Review*. [online] IGN. Available at: https://www.ign.com/articles/neon-white-review [Accessed 14 Oct. 2024].

Rempton Games (2019). *Run, Jump and Climb: Designing Fun Movement in Games*. [online] Repton Games. Available at: https://remptongames.com/2019/06/29/run-jump-and-climb-designing-fun-movement-in-games/ [Accessed 9 Nov. 2024].

Sailer, M., Hense, J.U., Mayr, S.K. and Mandl, H. (2017). How Gamification motivates: an Experimental Study of the Effects of Specific Game Design Elements on Psychological Need Satisfaction. *Computers in Human Behavior*, [online] 69(69), pp.371–380. doi:https://doi.org/10.1016/j.chb.2016.12.033.

Salsberg, A. (2019). *The 12 Principles of Animation | Lesley University*. [online] Lesley.edu. Available at: https://lesley.edu/article/the-12-principles-of-animation [Accessed 24 Nov. 2024].

School of Game Design (n.d.). *The Mechanics of Movement in Games | SCHOOL OF GAME DESIGN*. [online] School of Game Design. Available at: https://schoolofgamedesign.com/project/the-mechanics-of-movement-in-games/ [Accessed 9 Nov. 2024].

Shead, M. (2018). *Understanding State Machines*. [online] freeCodeCamp.org. Available at: https://www.freecodecamp.org/news/state-machines-basics-of-computer-science-d42855debc66/ [Accessed 24 Feb. 2025].

Smith, J. (2024). *accordion sprout*. [online] accordion sprout. Available at: https://www.accordionsprout.com/blog/gpusu79xtbqhjoq1v0iqvk2q0et2ro [Accessed 18 Dec. 2024].

Swink, S. (2007). *Game Feel: The Secret Ingredient*. [online] Game Developer. Available at: https://www.gamedeveloper.com/design/game-feel-the-secret-ingredient [Accessed 18 Dec. 2024].

Taylor, V. (2021). *‘Game Feel’: The Impression Of Weight - Vincent Taylor - Medium*. [online] Medium. Available at: https://vintay.medium.com/game-feel-the-impression-of-weight-b80f39767e6b [Accessed 9 Nov. 2024].

Techland (2022). *Dying 2 Know More Let’s Talk About Parkour With Tymon Smektała*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=URlBlMfzEnU [Accessed 28 Nov. 2024].

Technologies, U. (n.d.). *Unity - Manual: Inverse Kinematics*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/Manual/InverseKinematics.html [Accessed 16 Nov. 2024].

Terry Clinkenbeard (2017). *Movement Mechanics – The First Building Block of Gaming*. [online] Hopeful Homies. Available at: https://hopefulhomies.com/2017/02/18/movement-mechanics/ [Accessed 25 Oct. 2024].

Trine University (2023). *The Relationship Between Self-Confidence and Performance*. [online] www.trine.edu. Available at: https://www.trine.edu/academics/centers/center-for-sports-studies/blog/2023/the\_relationship\_between\_self-confidence\_and\_performance.aspx [Accessed 26 Nov. 2024].

tutorialspoint (n.d.). *Deterministic Finite Automaton - Tutorialspoint*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/automata\_theory/deterministic\_finite\_automaton.htm [Accessed 24 Feb. 2025].

Unity (2024). *Procedural Motion | Cinemachine | 3.1.2*. [online] Unity3d.com. Available at: https://docs.unity3d.com/Packages/com.unity.cinemachine@3.1/manual/concept-procedural-motion.html [Accessed 9 Jan. 2025].

Wikipedia Contributors (2019). *Mealy machine*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Mealy\_machine [Accessed 24 Feb. 2025].

Wiltshire, A. (2016). How Grow Home Uses Maths To Generate A Personality. *Rock Paper Shotgun*. [online] 23 Sep. Available at: https://www.rockpapershotgun.com/how-grow-home-uses-maths-to-generate-a-personality [Accessed 16 Nov. 2024].

Young, C. and Young, C. (2014). *Adrenaline Rush: The Effects of Video Games*. [online] prezi.com. Available at: https://prezi.com/4cyzozrs14r5/adrenaline-rush-the-effects-of-video-games/ [Accessed 9 Nov. 2024].

Zucconi, A. (2017). *An Introduction to Procedural Animations*. [online] Alan Zucconi. Available at: https://www.alanzucconi.com/2017/04/17/procedural-animations/ [Accessed 16 Nov. 2024].

Adobe (2019). *Mixamo*. [online] Mixamo. Available at: https://www.mixamo.com/#/ [Accessed 22 Feb. 2025].

Ansley, M. (2021). *Enums! (an introduction to enumeration types with Unity)*. [online] Medium. Available at: https://m-ansley.medium.com/enums-an-introduction-to-enumeration-types-with-unity-e4340c883da8 [Accessed 10 Feb. 2025].

Gamedev Inspire (2023). *first person head bob in unity3d | very easy fps head bob system*. [online] YouTube. Available at: https://www.youtube.com/watch?v=2ysd9uWmUfo [Accessed 24 Jan. 2025].

Hero 3D (2021). *Unity Quick Guide - FPS Head Bob + Stabilization*. [online] YouTube. Available at: https://www.youtube.com/watch?v=5MbR2qJK8Tc [Accessed 24 Jan. 2025].

Hetherington, A. (2024). *Wall Running in Unity*. [online] Footnotesforthefuture.com. Available at: http://www.footnotesforthefuture.com/words/wall-running-1/ [Accessed 18 Jan. 2025].

Max O'Didily (2023). *How to Import Mixamo Animations Into Unity (Quick and Simple)*. [online] YouTube. Available at: https://www.youtube.com/watch?v=2T0hDIoQKNc [Accessed 17 Jan. 2025].

Technologies, U. (n.d.). *Unity - Scripting API: Array*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/2020.1/Documentation/ScriptReference/Array.html [Accessed 15 Dec. 2024].

Technologies, U. (n.d.). *Unity - Scripting API: Input.GetAxisRaw*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Input.GetAxisRaw.html [Accessed 2 Dec. 2024].

Technologies, U. (n.d.). *Unity - Scripting API: Mathf.Abs*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Mathf.Abs.html [Accessed 6 Dec. 2024].

Technologies, U. (n.d.). *Unity - Scripting API: MonoBehaviour.StartCoroutine*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.StartCoroutine.html [Accessed 2 Dec. 2024].

Unity (n.d.). *Quick start guide | Input System | 1.0.2*. [online] docs.unity3d.com. Available at: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/QuickStartGuide.html [Accessed 2 Dec. 2024].

Unity Technologies (2014). *get the collision angle*. [online] Unity Discussions. Available at: https://discussions.unity.com/t/get-the-collision-angle/540075 [Accessed 20 Jan. 2025].

Unity Technologies (2015). *Rotate Player with Camera rotation*. [online] Unity Discussions. Available at: https://discussions.unity.com/t/rotate-player-with-camera-rotation/131656/3 [Accessed 17 Jan. 2025].

Unity Technologies (2019a). *Switch Statements - Unity Learn*. [online] Unity Learn. Available at: https://learn.unity.com/tutorial/switch-statements#5c8a6f91edbc2a067d4753d4 [Accessed 20 Jan. 2025].

Unity Technologies (2019b). *Unity - Scripting API: Physics.Raycast*. [online] Unity3d.com. Available at: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html [Accessed 5 Dec. 2024].

Unity Technologies (2020). *How to detect if the player is falling?* [online] Unity Discussions. Available at: https://discussions.unity.com/t/how-to-detect-if-the-player-is-falling/787443 [Accessed 9 Feb. 2025].

Unity Technologies (2024). *Unity - Scripting API: LayerMask*. [online] Unity3d.com. Available at: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/LayerMask.html [Accessed 5 Dec. 2024].

Unity Technologies (2025). *Unity - Scripting API: Physics.SphereCast*. [online] Unity3d.com. Available at: https://docs.unity3d.com/6000.0/Documentation/ScriptReference/Physics.SphereCast.html [Accessed 20 Jan. 2025].

Youtube and Dave / Game Development (2022). *FIRST PERSON MOVEMENT in 10 MINUTES - Unity Tutorial*. [online] www.youtube.com. Available at: https://www.youtube.com/watch?v=f473C43s8nE [Accessed 2 Dec. 2024].