



**Faculty of Electrical Engineering
DEPARTMENT OF COMPUTER SCIENCE**

Bachelor's Thesis

3D Movement shooter with spider mechanics

Dmitrii Zamedianskii

May 2023

Field of Study: Open Informatics

Supervisor: Ing. Tomáš Havlík

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Zamedianskii** Jméno: **Dmitrii** Osobní číslo: **503214**
Fakulta/ústav: **Fakulta elektrotechnická**
Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**
Studijní program: **Otevřená informatika**
Specializace: **Počítačové hry a grafika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Pohybová střílečka s pavoučími mechanikami

Název bakalářské práce anglicky:

Movement shooter with spider mechanics

Pokyny pro vypracování:

1. Analyzujte retro střílečky typu Dusk, Ultrakill a jejich herní mechaniky. Vytvořte příběh a specifikujte návrhové koncepty pro vaši hru.
2. Definujte pohybový systém.
3. Navrhněte bojový systém obsahující mechaniky pro boj na blízko (melee) a na dálku (střelné zbraně).
4. Navrhněte intuitivní uživatelské rozhraní, které umožní vybírání a využívání schopností.
5. Nastiňte tři typy nepřátel, u každého typu popište možnosti pohybu a boje.
6. Implementujte hratelný prototyp obsahující zmíněné mechaniky a omezenou herní oblast.
7. Otestujte prototyp s minimálně třemi osobami. Během vývoje využijte principů user-centered design.
8. Popište možnosti vývoje do budoucna.

Seznam doporučené literatury:

1. SCHELL, Jesse. The Art of Game Design: A Book of Lenses. 1st ed. CRC Press, 2008. ISBN 978-0123694966.
2. SELLERS, Michael. Advanced Game Design: A Systems Approach, 1st ed., Addison-Wesley Professional, 2017. ISBN 978-0-13-466760-7.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Tomáš Havlík katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **13.02.2023**

Termín odevzdání bakalářské práce: **26.05.2023**

Platnost zadání bakalářské práce: **22.09.2024**

Ing. Tomáš Havlík
podpis vedoucí(ho) práce

podpis vedoucí(ho) ústavu/katedry

prof. Mgr. Petr Páta, Ph.D.
podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání

Podpis studenta

Acknowledgement / Declaration

I would like to thank Ing. Tomáš Havlík, for his guidance, willingness to help, and for giving me the opportunity to work on the game. Also, I would like to thank my girlfriend, who provided emotional support and supported me through tough times. Lastly, I would like to thank all of my friends who inspired me and helped me whenever I needed advice.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 20. 05. 2023

.....

Abstrakt / Abstract

Tato studie se zaměřuje na tvorbu 3D pohybové arénové střílečky na základě analýzy retro stříleček jako Dusk a Ultrakill. Zahrnuje vývoj herního příběhu, podrobného pohybového systému, dynamického bojového systému, intuitivního uživatelského rozhraní a jedinečných typů nepřátel. Byl vyvinut, testován a vylepšen hratelný prototyp s využitím principů designu zaměřeného na uživatele. Projekt vyvrcholil určením několika cest pro budoucí rozšíření a vylepšení hry.

Klíčová slova: Unity, 3D střílečka, videohra, herní design.

Překlad titulu: 3D střílečka s pohybem inspirovaným pavoukama mechaniky

This study focuses on the creation of a 3D movement arena shooter video game, informed by the analysis of retro shooters like Dusk and Ultrakill. It covers the development of a game narrative, a detailed movement system, a dynamic combat system, an intuitive user interface, and unique enemy types. A playable prototype was developed, tested, and refined using user-centered design principles. The project culminated in identifying multiple avenues for future expansion and improvement of the game.

Keywords: Unity, 3D shooter, video game, game design.

Contents /

1 Introduction	1		
1.1 Goals of Thesis	1		
1.2 Motivation	1		
2 Analysis	2		
2.1 3D Movement Shooter games	2		
2.1.1 Ultrakill	2		
2.1.2 DUSK	5		
2.1.3 Severed Steel	6		
2.1.4 Other Games	7		
2.1.5 Summary	8		
3 Game Design	9		
3.1 Idea	9		
3.2 Game Elements	10		
3.3 Game Controls	11		
3.4 Enemies	11		
3.5 Combat	14		
3.6 Movement	15		
3.7 Narrative	16		
3.8 User Interface	17		
3.9 Character design	19		
3.10 Level Design	19		
3.11 Sound Design and Music	20		
4 Development of the game	22		
4.1 3D assets	22		
4.2 Scripts	25		
4.2.1 Movement Mechanics	25		
4.2.2 Sliding	27		
4.2.3 WallRunning	27		
4.2.4 LedgeGrabbing	28		
4.2.5 Dashing	28		
4.2.6 Abilities	29		
4.2.7 First Attempt: EnemyAI Script	31		
4.2.8 Reworking the AI System	32		
4.2.9 The Role of ObjectPool System	33		
4.2.10 Procedurally Animated Spider	33		
4.2.11 Projectile Weapons	33		
4.2.12 Raycast Weapons	34		
4.2.13 Additional Weapon Mechanics	35		
4.2.14 Hand-to-Hand Combat	36		
4.2.15 Pie Menu	36		
4.2.16 Pause Menu and Death Screen	37		
4.2.17 Main Menu	38		
5 User Testing	41		
6 Future Development	43		
7 Conclusion	45		
References	47		
A Abbreviations and symbols	51		
A.1 Abbreviations	51		
B Used Assets	52		
C Attachments	53		

Tables /

B.1 Authors of used assets	52
---	----

Chapter 1

Introduction

Games have been a part of human life for as long as we can remember, across all cultures and times. The 1950s brought a big change with the first video game prototypes. By 1971, the first video game for the public came out, starting to take the place of old-school board games. Just 10 to 15 years ago, video games were seen as for a special group of people only. But the quick growth of the video game market has made them a main part of entertainment. It looks like there's no stopping how much more popular video games can get.

1.1 Goals of Thesis

The purpose of this project is to design and develop a 3D movement shooter video game. This project will involve researching and analyzing similar games in the genre to gain an understanding of the conventions and trends in game design. Additionally, the project will involve creating a game design document outlining the concept, mechanics, and features of the game being developed. The project will also research and implement various game development techniques, including 3D modeling, animation, physics, and artificial intelligence. Additionally, the project will explore methods of player locomotion and immersion. The project will also evaluate the gameplay mechanics, balance, and overall player experience. This project will serve as a comprehensive study of the process of creating a 3D-movement shooter video game, from the initial concept to the final product.

1.2 Motivation

My main source of inspiration for this project is my endless love for video games. I wanted to gain a deeper understanding of the game design and development pipeline that any project goes through. Also, I wanted to explore new technologies that I have never used before.

The reason why I chose to develop a 3D arena movement shooter is that I have always been a big fan of classic first-person shooter games such as Doom [1] and Quake [2] since a young age and with the recent revival of this genre with games including games such as Ultrakill and Dusk, I decided to deepen my understanding of this type of games.

Chapter 2

Analysis

One of the most important steps in making a game is the analysis of the market, its thorough examination, and finding references to work with. I found as many similar games in the 3D movement shooter genre as possible to gain an understanding of the conventions and trends in game design. This analysis will include a review of the gameplay mechanics, level design, and other key elements of the selected games.

2.1 3D Movement Shooter games

My goal is to gain a comprehensive understanding of what makes a successful 3D movement shooter game. I will be looking into the gameplay mechanics, level design, graphics and audio, and player experience. By studying a selection of games in the 3D movement shooter genre, I aim to identify recurring patterns and established methods in game design, which I will then incorporate into my own game development process. Through this process, I want to gain a deeper understanding of the genre and create a game that will reach the set bar of expectations of this genre.

2.1.1 Ultrakill



Figure 2.1. Ultrakill (2020). [3]

The first game that caught my eye was Ultrakill. The game’s Steam store page describes the title as “a fast-paced ultraviolent retro FPS¹ combining the skill-based style scoring from character action games with unadulterated carnage inspired by the best shooters of the ’90s. Rip apart your foes with varied destructive weapons and shower in their blood to regain your health.” [3]

¹ First-person shooter (FPS) is a subgenre of shooter video games centered on gun and other weapon-based combat in a first-person perspective, with the player experiencing the action through the eyes of an antagonist or protagonist who is armed, and then controlling the player character in a three-dimensional space. [4]

Since Ultrakill is a *movement shooter*, it would be logical to start with movement mechanics.

First and foremost, *jumping*. The foundation of all other movement mechanics. While the jump ability itself is not that impressive, the game does a good job of extending most of its game mechanics. Here, the jump was expanded using *bunny hopping*², precise *Air Control*, and combinations with other movement mechanics.

The next movement mechanic is the ability for the player to *slide along walls* and perform *wall jumps*. This allows the player to navigate through the game's levels quickly and evade enemies by moving in unexpected ways.

One of the most important mechanics in any movement shooter is *dash* ability. This ability can be used to evade enemy attacks, close in on enemies, or reach distant platforms. The dash ability is also used in combination with the wall sliding and wall jumping mechanics to cover more ground and reach more areas in the level. Overall, the dash ability in Ultrakill is an important aspect of the gameplay, providing an extra layer of mobility and versatility to the player's movement options, allowing them to outmaneuver their enemies and overcome obstacles.

The last movement mechanic is one of the most fun in the game - the *slide* ability allows players to slide along the ground quickly. This ability can be used to evade enemy attacks, reach tight spaces, and cover distances in a short amount of time. This ability also allows the player to slide through narrow spaces and under obstacles, giving them more options to explore the environment and avoid enemies.

One of the key mechanics in the game is fast-paced, intense combat. The player has access to a variety of *weapons*. Each of these weapons can be categorized into a more general weapon type.

Here is the list of weapons included in the game read as follows:

- Revolver
- Shotgun
- Nailgun
- Railcannon
- Rocket Launcher
- Arm

I will discuss these weapons later in the chapter, but let's try to categorize them.

Revolver represents the *Handgun* category - it is a weak weapon that only shoots once per click but serves as a good starter weapon for players to learn the ropes with.

Shotgun is a close-range powerful weapon that deals a lot of damage when the enemy is near. I will categorize this into *Shotgun* category.

Nailgun falls into a category of *Rapid-Fire* weapons. It is a weapon that deals small damage per bullet but compensates for it with high DPS (Damage-Per-Second).

Railcannon is a weapon with precise aiming that shoots once and goes into cooldown for some amount of time. It is really similar to **sniper rifles** in other games.

² Bunny hopping, in video games, is a technique used to increase movement speed and control.

Rocket Launcher - an explosive weapon that deals a lot of damage to all nearby enemies caught within the splash radius. The name seems to be very descriptive by itself, so I will put it into a *Rocket Launcher* category.

Arm is what players use to hit an enemy that is very close. I will categorize it as *Melee* weapon.

What is interesting about most Movement Shooters is that they encourage players to make combo chains out of available weapons. What that means is that the player can shoot an enemy with a handgun, switch to Rocket Launcher, launch the enemy in the air, jump toward them, and finish them with a shotgun. And that is just one out of hundreds of possible action sequences that players can do.

The main differentiating factor that makes weapons in *Ultrakill* stand out from other similar games is that each of them has a unique ability. That expands possible combo chains from hundreds to thousands. To illustrate the possibilities consider the following example. One of the handguns in the game has the ability to throw a coin in the air. If the player hits this coin with a bullet while it flies in the air, the bullet explodes into multiple parts that fly to multiple nearest enemies and hit them right in the head. But what if a player throws a coin in the air, quickly switches to the rail cannon, and shoots the coin? That is one of the many examples where this game encourages players to think outside of the box. Keep in mind that all of the aforementioned meta actions happen during really fast action-packed fights!

Let us have a look at the enemies that the game presents us.

For the sake of this thesis not being only about *Ultrakill* and its game design, I will not list every enemy type in the game. Instead, I will categorize them right away.

- Pursuers - these run after the player trying to hit him.
- Shooters - these try to keep a distance from the player but can hit the player if they get too close.
- Supports - these provide support for other enemies in the form of healing, additional armor or player debuff³.
- Bosses - these possess a lot of armor and unique abilities.

Each of these categories could also be either flying or walking. What the developer achieves with this separation is the establishment of one simple rule - “Never stand still“. While pursuers are running after the player, shooters are trying to shoot them down and supports are making all of the enemies even more powerful.

One important observation: *Ultrakill* does not just spawn every enemy back-to-back, sometimes there is intentional simplicity to it. When it does throw every enemy at you, you are usually given a choice of multiple weapons in close proximity [5].

While I am at it, let us talk a little bit about level design. Levels in *Ultrakill* either have physical hazards that influence player movement, or non-combat objectives that influence the player’s pacing.

Ultrakill’s analysis came out pretty substantive. But I used its case to establish main points and categories for my next examples.

³ An effect that makes a game character weaker.

2.1.2 DUSK



Figure 2.2. Dusk (2018). [6]

Another game I will use as an example is *Dusk (2018)*. I will not go into as much detail as I did with *Ultrakill* since these games share a lot of similarities.

On the other hand, *Dusk*'s movement mechanics are not as diverse as *Ultrakill*'s. *Dusk* has *jumping* with a possible *bunny hop* and a possibility to do a *flip*.

But let us take a closer look at the arsenal of weapons that *Dusk* offers us:

- Sickles - *melee* weapon that the player has throughout the whole game.
- Sword - rare *melee* weapon that deals a lot of damage.
- Pistol - starter *handgun* that the player use while he is still learning the ropes of the game.
- Shotgun - deals a lot of damage in close range, like *shotgun* does.
- Super shotgun - deals even more damage than normal *shotgun* but has a low amount of ammo per magazine.
- Assault rifle - *rapid-fire* weapon that shoots at a high rate of fire.
- Hunting rifle - *sniper rifle* that shoots only one bullet per click but deals a lot of damage at a big distance.
- Crossbow - another variation of *sniper rifle* that also deals a lot of damage to an enemy.
- Mortar - *rocket launcher* or in this case a grenade launcher that shoots out explosives.
- Riveter - *rocket launcher* that deals the most damage out of almost all weapons in the game.

Developers might use a big arsenal of guns in their games, but they all would fall into these typical categories.

But what is important is that in contrast to other games of the same genre, movement shooters let player chain their usage of weapons. That means a quick change between weapon 1 and weapon 2.

DUSK features a variety of *enemies* that players must defeat throughout the game. Here is a list of the enemies in *Dusk* and how they complement each other:

- Cultist - a basic enemy type, they are weak but can come in large numbers.

- Possessed - possessed enemies are faster and more aggressive than cultists. They can also jump and attack from a distance.
- Outlander - heavily armed and armored enemies that can take a lot of damage. They are particularly effective in long-range combat.
- Hellhound - fast and aggressive enemies. Hellhounds are particularly effective at close-range combat.
- Cacodemon - large, flying enemies that can attack from a distance with projectiles.
- Spider - arachnoid-like enemies that can climb walls and ceilings, they are weak but can attack from unexpected angles.
- Zombie - slow-moving enemies that can take a lot of damage.
- Revenants - a fast-moving type of enemy that can quickly close in on the player and attack with melee weapons.
- Enforcers - an elite type of enemy that is heavily armored and armed with powerful weapons.
- Bosses - a unique type of enemy that the player must defeat to progress.

These enemies complement each other in a way that they offer different challenges and require different strategies to defeat. For example, the Cultists are weak but come in large numbers, requiring the player to use weapons that have a large area of effect or use their magic abilities. Possessed enemies require quick reflexes and a fast-paced approach, while Outlanders require a more strategic approach to take them out from a distance. The bosses require a specific strategy to defeat them and are the climax of each level. The variety of enemies keeps the gameplay fresh and interesting and also encourages players to experiment with different weapons and strategies.

The *level design* philosophy in Dusk is centered around creating a fast-paced and challenging experience for players, while also incorporating elements of exploration and puzzle-solving. The levels are designed to be non-linear, allowing players to explore and find secrets and shortcuts, while also providing enough combat encounters to keep the gameplay engaging.

The levels are also designed to be atmospheric and immersive, using lighting, sound, and level architecture to create a sense of tension and unease. The game's retro-style aesthetic also plays a role in the level design, with levels featuring pixelated graphics and a retro aesthetic reminiscent of classic first-person shooters from the 1990s.

2.1.3 Severed Steel

While Severed Steel is a simpler game than Ultrakill or Dusk from the mechanical and design point of view, it is still a fascinating example of a movement shooter done right.

The secret of Severed Steel is its focus on one core mechanic - *bullet time*⁴. This game mechanic combined with good *level design* and diverse *movement mechanics* creates a unique and fun game experience.

⁴ A mode in which gameplay slows down to allow the player to observe and maneuver around bullets or other fast-moving objects. [8]



Figure 2.3. Severed Steel (2018). [7]

The emphasis in Severed Steel is placed on *speed of movement* and the player trying to beat levels as quickly as possible, beating his previous records (in other words, speedrunning).

That is the main reason why while there is a lot of different *weapons* in the game, most of them are there to add diversity rather than trying to get players to combine them in combos in unique ways.

For the same reason, there are only three types of *enemies* in the game: Soldier, Sniper, and Juggernaut. All of them are attacking types so there is no categorization like in Dusk or Ultrakill.

2.1.4 Other Games

Other examples of similar games include:

- Ghostrunner,
- Neon White,
- Turbo Overkill

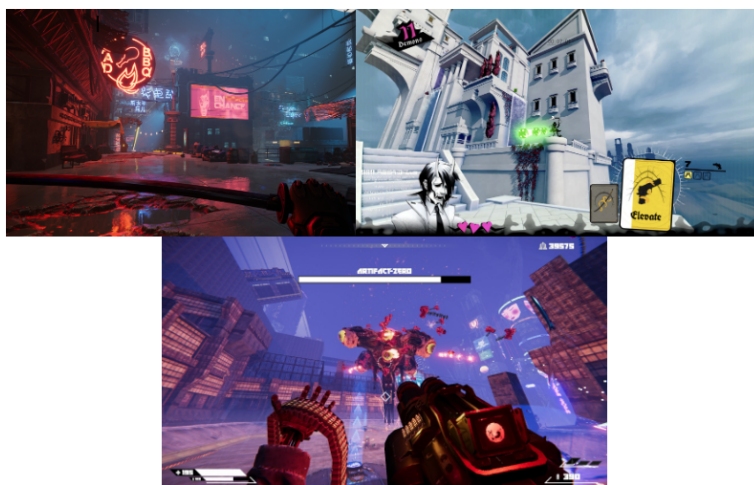


Figure 2.4. Ghostrunner [9], Neon White [10], and Turbo Overkill [11]

■ 2.1.5 Summary

By analyzing various examples, we can identify the key elements that make movement an engaging game mechanic. These include:

- Combining moves to create fluid sequences.
- Utilizing the environment to enhance movement options.
- Timing movements to take advantage of opportunities.
- Building momentum to increase speed and power.
- Understanding the trajectory of projectiles and movements.
- Incorporating physics to add realism and challenge.

From my research on movement-based shooters, I have discovered that the most successful games in this genre are those that minimize the skill required to master the movements, while still providing a challenging test of skill through the incorporation of the above elements.

Chapter 3

Game Design

A game design document is a crucial part of the game development process, serving as a blueprint for the game's design and functionality. It outlines all the key elements that will make up the game, including the game mechanics, the player's journey, and the overall vision for the game.

In this chapter, I will delve into the intricacies of the game design document, focusing primarily on the core concept and inspiration behind the game I am trying to make, the various elements that make up the gameplay, the method of control and navigation, and an in-depth examination of desired level design.

3.1 Idea

As Jesse Schell writes in his book "The Art of Game Design: A Book of Lenses" [12] there are 8 filters that any idea for a game should go through at different stages of development:

- Does this game feel right?
- Will the intended audience like this game enough?
- Is this a well-designed game?
- Is this game novel enough?
- Will this game sell?
- Is it technically possible to build this game?
- Does this game meets social and community goals?
- Do the playtesters enjoy this game enough?

The main idea is that the player takes on the role of a humanoid spider and navigates through a 3D first-person movement shooter game. The core emphasis of the game is centered around its combat system, which features a wide variety of enemies that will constantly challenge the player and keep them on their toes. In addition to this, the design of the levels should take into account the player's ability to move quickly and perform a variety of maneuvers, allowing for a more dynamic and engaging gameplay experience.

So let us run my idea through these filters:

- The first filter is „Does this game feel right?“ as mentioned in Jesse Schell's book. He suggests that this is the most personal of the filters and that the gut feelings of the developer are important. I agree with this perspective and my gut tells me that this game feels right. From my personal experience and confidence in the game, I trust my instincts.
- The second filter is „Will the intended audience like this game enough?“. I believe that the target audience for my game is young adults between the ages of 18-24 who are fans of games such as Doom, Quake, Ultrakill, and Dusk. The game design fits this demographic perfectly.

- The third filter is „Is this a well-designed game?“. As Jesse Schell notes in his book, this filter takes into account everything we know about creating a good experience, including aesthetics, interest curves, resonant theme, and game balancing. From an aesthetic standpoint, I think the game is well-designed and it will stand out with its gothic grotesque style. In terms of overall game design, balancing the game is my main priority as it is crucial for movement shooters. If the game is too easy, players lose interest quickly, and if it is too hard, players get tired and also lose interest.
- The fourth filter is „Is this game novel enough?“. “If you are designing a new game, by definition, there needs to be something new about it, something players haven’t seen before“ [12]. In terms of movement mechanics and aesthetics, I believe this game brings a lot of new elements to the table. The unique abilities that the player has access to are something that I have not seen in other similar games.
- The fifth filter is „Will this game sell?“. From my market analysis, I have found that 3D first-person movement shooters are generally successful as they combine elements of classic first-person shooters with modern graphics and gameplay mechanics. Some games also feature a strong retro aesthetic, which appeals to players who enjoy nostalgia. Players value intense action, challenging difficulty, and creative level design in these types of games.
- The sixth filter is „Is it technically possible to build this game?“ and the answer is yes. There are many examples of similar games on the market.
- The seventh filter is „Does this game meet the social and community goals?“. Jesse Schell suggests that “It is not enough for a game to be fun. Some of the design goals may require a strong social component, a strong viral component, or the formation of a thriving community around your game.“ [12]. My game does not have any intended social goals such as multiplayer, but it does have the potential for players to share different secrets found in the game or new interesting combos, which will help to build a community around the game.
- The last filter is „Do the playtesters enjoy this game enough?“. Unfortunately, I cannot answer that question yet, as I have not conducted any playtests with people who have not seen the game before. It is important to gather feedback from playtesters in order to understand if the game is enjoyable and engaging for the intended audience.

3.2 Game Elements

In this game, the central focus is on the mechanics of movement and the unique abilities of the player. These abilities include the ability to climb walls, run along walls, swing on the web, and quickly grapple to a specific point for faster traversal.

Another crucial aspect of the game is the diverse array of enemies that the player must contend with. These enemies pose a constant threat, requiring the player to remain vigilant and aware of their surroundings during combat. Some enemies may chase after the player, while others may attempt to shoot them from a distance. The player must constantly navigate these dangers while simultaneously eliminating all enemies.

To successfully complete the game, the player has access to a variety of weapons that can be used in combination with one another. For example, the

player may use a rocket launcher to launch an enemy into the air, and then jump up and shoot them while they are still in the air.

3.3 Game Controls

The game was designed to be played using a combination of a mouse and keyboard, providing players with intuitive controls for an immersive experience. The following controls were implemented to enhance player interaction:

1. **W, A, S, D:** These keys control the character's movement, allowing players to navigate the game world with ease.
2. **Space:** Pressing the Space button enables the player to make their character jump, adding verticality to the gameplay.
3. **Left Mouse Button:** The left mouse button is used for shooting, allowing players to unleash their chosen weapon upon enemies.
4. **Right Mouse Button:** By clicking the right mouse button, players can activate their character's unique ability, adding a strategic element to gameplay.
5. **Q:** Pressing the Q key allows players to choose between different abilities, providing flexibility and adaptability in various situations.
6. **W (while looking at the wall):** When facing a wall, pressing the W key enables the character to climb the wall, expanding vertical traversal options.
7. **W (with a wall on either side of the player):** When near a wall on the right or left, pressing the W key triggers a wall run, allowing the character to move swiftly along vertical surfaces.
8. **F (while looking at the wall):** When facing a wall, pressing the F key enables the character to stick to the wall, offering additional navigation possibilities.
9. **1, 2, 3, Mouse Scroll Wheel:** These controls enable players to switch between different weapons, ensuring flexibility in combat situations.
10. **R:** Pressing the R key triggers a reload action, allowing players to replenish their ammunition during intense battles.
11. **G:** The G key is used for a melee attack, providing players with a close-quarters combat option.
12. **Esc:** Pressing the Esc key allows players to pause the game, providing a convenient option to access menus or take a break.

By incorporating these controls, the game offers a range of movement options, combat mechanics, and abilities, granting players a heightened level of control and immersion in the gameplay experience.

3.4 Enemies

The assets employed for the construction of enemy models are documented in the section denoted by B.

In the game, there are five different enemy types: Spider Tank, Spider Turret, Flying Robot, Spider Zombie, and Spider Robot. As explained in Scott Rogers' book, *Level Up! The Guide to Great Video Game Design* [13], a diverse array of enemies, each with a unique behavior and purpose, is crucial in maintaining player engagement and providing a dynamic gameplay experience. In this

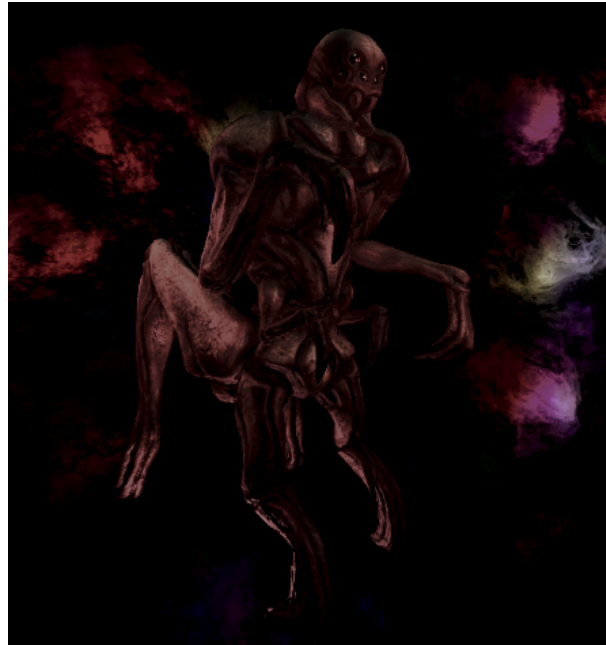


Figure 3.1. Spider Tank.

context, each of these enemies in the game serves a specific purpose and adds unique challenges to the gameplay.

The Spider Tank is a formidable enemy with a large amount of health points. Its main objective is to relentlessly chase the player, creating a sense of urgency and requiring the player to find strategies to avoid its pursuit.



Figure 3.2. Spider Turret.

The Spider Turret is a stationary enemy that poses a long-range threat. It remains in one place and continuously fires projectiles at the player, forcing them to keep moving and find cover to avoid taking damage.

The Flying Robot is a fast and agile enemy. It hovers in the air, swiftly maneuvering and shooting at the player. Its speed and aerial capabilities make it a challenging enemy to evade, pushing the player to be agile and precise in their movements.

The Spider Zombie is an enemy that constantly follows and tries to attack the player. Its relentless pursuit keeps the player on their toes, requiring constant vigilance and the use of evasive actions to avoid being hit.

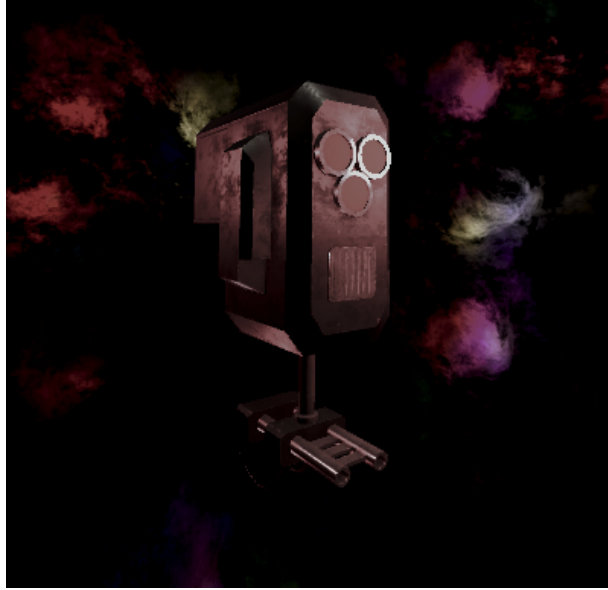


Figure 3.3. Flying Robot.



Figure 3.4. Skelet zombie.

The Spider Robot is a visually impressive enemy with procedural animation. It dynamically follows the player, constantly adjusting its movements to deal damage. Its realistic animation adds depth to the gameplay, creating an immersive and captivating experience.

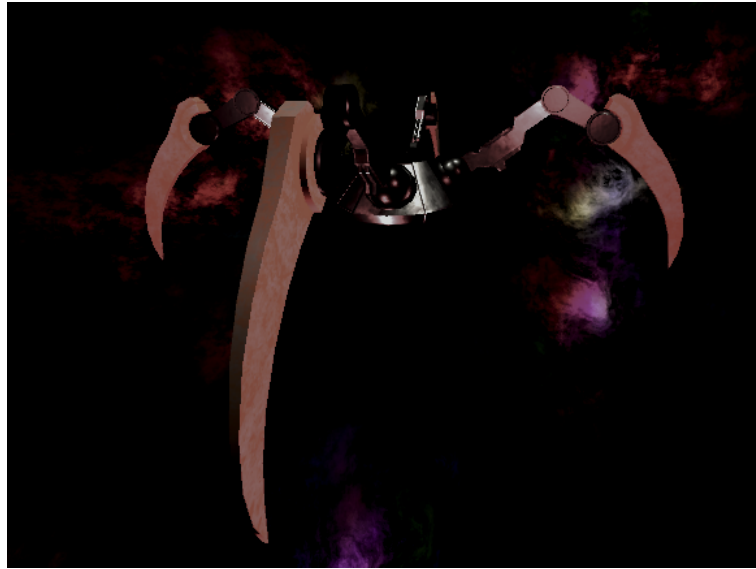


Figure 3.5. Robot Spider.

These different enemy types provide variety and strategic depth to the game. Players must adapt their tactics and gameplay approach to counter each enemy's unique behavior and abilities. By introducing diverse enemies with specific roles, the game provides player engagement as well as a dynamic and challenging experience.

3.5 Combat

The utilized resources for weapon models and visual effects are indexed in the Used Assets section B.

Combat mechanics play a pivotal role in shaping the excitement and intensity of any shooter game. The combat experience in the game is designed to be engaging, dynamic, and immersive, providing players with a diverse range of weapons and combat options to create thrilling and strategic gameplay moments.

In the game, players have access to a variety of weapons, each offering unique characteristics and playstyles. These weapons, ranging from rapid-fire firearms to rocket launchers, cater to different preferences and combat scenarios. To ensure clarity and distinction, future plans include incorporating detailed and visually appealing models for each weapon, allowing players to easily differentiate between them.

Expanding the range of available weapons is a priority for the game's development. Additional options such as handguns, shotguns, sniper rifles, and melee weapons are in the works. This expansion not only increases player choice but also adds depth and variety to the combat experience. Each weapon will present its own advantages and drawbacks, enabling players to adopt different strategies and playstyles based on their preferences.

The game aims to deliver a combat experience that is both accessible and deep. Responsive and intuitive controls are being developed to establish a strong

connection between player actions and in-game combat. Smooth and fluid movement mechanics, precise aiming controls, and impactful feedback from weapon effects contribute to a satisfying and immersive combat experience.

In addition to the diverse selection of weapons, the game introduces a hand-to-hand combat system, adding an exciting layer of strategy and skill to the gameplay. Players will have the opportunity to engage in close-quarters combat, utilizing well-timed attacks and defensive maneuvers to overcome adversaries. This system offers new avenues for tactical decision-making and provides a different type of combat experience within the game, reflecting the playcentric approach discussed by Fullerton in *Game Design Workshop* [14].

The ongoing refinement and expansion of combat mechanics aim to strike a balance between accessibility and depth. The goal is to ensure that players of all skill levels can enjoy the combat experience while providing opportunities for mastery and strategic gameplay. Through careful tuning and iteration, the game aims to deliver thrilling and satisfying combat encounters that keep players engaged and immersed throughout their gameplay journey.

Overall, the focus on combat mechanics in the game is aimed at delivering an exciting and immersive gameplay experience. By offering a diverse array of weapons, strategic combat options, and responsive controls, the game provides players with a thrilling and engaging combat experience that will captivate them as they progress through the game.

3.6 Movement

Influenced by the principles shared in Alex Wiltshire's article *Designer Interview: Getting Titanfall's controls just right* [15], in which Alex interviewed several Titanfall developers, I have made significant refinements to the movement mechanics in my game to ensure a seamless and immersive experience. The mechanics not only provide players the freedom to navigate through the game world but also enable the execution of unique and dynamic movement abilities.

I started by implementing a climbing mechanic that allows players to scale vertical surfaces. This addition not only opens up new pathways and exploration opportunities but also adds verticality to the gameplay, reinforcing the emphasis on fluid and intuitive movement mechanics.

Building on this, I introduced a swinging ability, facilitated by a web-based mechanic. This feature allows players to swiftly move through the environment, leaping from one point to another. This mechanic echoes the focus on a sense of fluidity and speed in movement design, granting players control over their movement and immersing them in the exhilarating traversal experience.

I have also incorporated mechanics such as sliding, crouching, and dashing, enabling players to quickly evade obstacles and enemies, move stealthily, and cover longer distances in shorter times, respectively. Furthermore, a grappling mechanic was introduced to expand upon movement possibilities further. Players can now rapidly traverse the game world, effortlessly reaching new areas and propelling themselves with speed and agility.

These mechanics work in harmony, providing players with a wide range of movement options. Whether it is climbing walls, swinging through the environment, sliding to dodge obstacles, or dashing to cover large distances, these mechanics contribute to an immersive and dynamic gameplay experience.

These mechanics are also integrated into the overall gameplay loop, creating opportunities for strategic decision-making and enhancing gameplay depth. Players can utilize different movement abilities to approach challenges, explore the environment, and engage in combat encounters that suit their playstyle.

Overall, by developing and implementing these refined movement mechanics, I aim to offer players a sense of freedom, immersion, and control. Drawing from the principles of fluid and intuitive movement mechanics, I have sought to ensure smooth and responsive gameplay, delivering an engaging and enjoyable experience for players as they navigate the game world with ease and precision.

3.7 Narrative

As Michael Sellers stated in his book “Advanced Game Design: A Systemic Approach” [16]: “Narrative is important in that it bridges both architecture and theme: it has an inward, developer-focused side in terms of how the story is put together out of the underlying functional elements, and it has a player-facing thematic side in how it sets the stage for the players and informs them of what the game is about.”

In line with this perspective, the game’s narrative holds significant importance as it not only shapes the underlying functional elements but also sets the stage for players, providing them with a sense of purpose and a deeper understanding of the game’s essence. By carefully crafting a compelling narrative, game designers can create a world that captivates players’ imagination and draws them into a rich and immersive experience.

In this particular game, players assume the role of a humanoid spider, immersing themselves in a world where spiders have evolved into a higher form of life and have become the dominant species on Earth. This unique premise offers a distinctive perspective that sets the game apart and presents players with an intriguing scenario to explore.

To ensure consistency and strengthen the narrative immersion, it becomes crucial to align all aspects of the game with the spider theme. This includes designing enemies with spider-like features and abilities. By incorporating these elements into the enemy designs, the game world becomes more cohesive and resonates with the overarching narrative. Players will encounter foes that share similarities with their own character, reinforcing the concept of a spider-dominated world and fostering a deeper connection to the game’s storyline.

Moreover, the narrative aspect of the game can extend beyond the visual elements and permeate the gameplay mechanics and level design. By integrating narrative-driven objectives, challenges, and puzzles, players can further immerse themselves in the world and feel a stronger sense of purpose and progression. For example, players may need to navigate intricate web-like structures, solve spider-themed riddles, or engage in encounters that test their spider-like abilities

and instincts. These gameplay elements not only contribute to the narrative coherence but also add depth and richness to the overall game experience.

Additionally, storytelling techniques such as in-game dialogues, cutscenes, or environmental storytelling can be employed to provide players with contextual information, unveil the game's lore, and deepen their connection to the narrative. By weaving these narrative elements into the fabric of the game, players become active participants in the unfolding story, fostering a greater sense of agency and investment.

Overall, a well-crafted and immersive narrative elevates the game beyond its mechanical components, allowing players to delve into a world where they can connect with the themes, characters, and events on a deeper level. By paying attention to narrative coherence, aligning all game elements with the spider theme, and incorporating narrative-driven gameplay mechanics, the game design aims to create a captivating and memorable experience that resonates with players and leaves a lasting impression.

3.8 User Interface

The assets utilized for the user interface, which include icons and look-up textures, are detailed in the Used Assets section B.

The user interface (UI) in the game was meticulously designed with the principles shared by Jenifer Tidwell in *Designing Interfaces* [17]. The objective was to offer an intuitive and immersive experience for players, combining simplicity, functionality, and customization to enhance gameplay engagement.

One of the key elements influenced by these principles are the crosshair and web indicators. A dot is prominently displayed on the screen, serving as both a crosshair for accurate aiming and an indicator for the location where the web will attach. This feature ensures precise targeting and helps players make accurate shots.

Likewise, to keep players informed about their ammunition status, a text field, as suggested by interface design patterns, is positioned in the bottom right corner of the screen. This displays the remaining bullets in the player's current weapon, empowering players to manage their ammunition effectively.

Interactable objects within the game world are intuitively highlighted when players direct their gaze toward them. This visual feedback provides players with a clear indication of objects they can interact with, encouraging exploration and progression.

In order to provide players with crucial information about their in-game well-being, a health indicator will be added. This dynamic element will offer a real-time representation of the player's current health status, enabling them to assess their situation and make strategic decisions accordingly.

Another significant addition to the UI is a weapon indicator. This feature will allow players to effortlessly switch between the various weapons at their disposal. By providing a quick and accessible way to manage their arsenal, this indicator enhances gameplay fluidity and empowers players to adapt to different combat scenarios.

To further augment the player's situational awareness, an armor indicator will be implemented. This indicator will display the player's current armor level, providing important information about their defensive capabilities. Players can use this information to make informed decisions about their approach to combat and self-preservation.

In the event of the demise of the player's character, a meticulously designed death screen will appear. Managed by the `DeathScreenController` script, this screen freezes the game, displaying relevant information to the player. It offers options such as the ability to restart the current level from the beginning or exiting to the main menu.

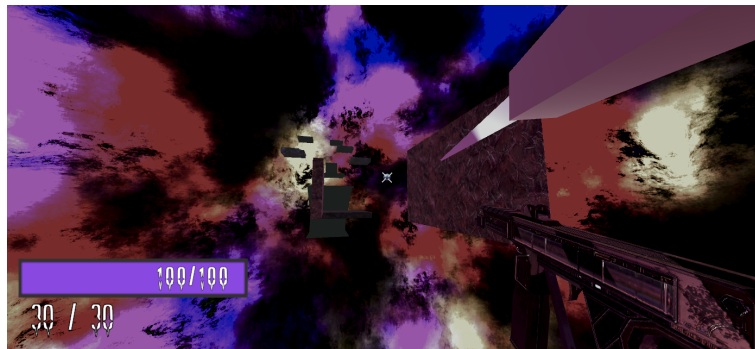


Figure 3.6. UI that can be seen in the game.

Sound settings are customizable through the `AudioManager` script, which enables players to adjust the Master, Music, and SFX volumes. These settings persist across game sessions, ensuring players can consistently enjoy their preferred audio experience consistently.

The game's options menu, driven by the `OptionsScreen` script, provides a comprehensive interface for players to personalize their game settings. From adjusting screen resolution to toggling full-screen mode and fine-tuning volume controls for Master, Music, and VFX channels, players have the ability to customize the gameplay environment to their preferences.

The main menu, controlled by the `MainMenu` script, serves as a central hub for players to navigate different aspects of the game. It offers options to start the game, access the sandbox mode, open the options menu, and quit the game, providing a seamless and convenient experience.

To enrich the game's audio environment, the `SoundtrackPlayer` script includes a selection of soundtracks and ambient sounds. From the get-go, a random soundtrack is played along with accompanying ambient sounds, immersing players in the game's audio atmosphere.

These carefully crafted UI elements, along with their associated scripts and features, work harmoniously to create an immersive, user-friendly, and polished gaming experience. By seamlessly integrating functionality, customization, and informative feedback, the UI enhances gameplay engagement and ensures players have an enjoyable and memorable time playing the game.

3.9 Character design

Given that the game centers around the theme of spiders, it is essential that the enemies within the game possess features and abilities that are inspired by real-world spiders. This may include incorporating arachnid-inspired body shapes and appendages, such as multiple limbs and eyes, into the design of the enemies. Additionally, the enemies should possess abilities that mimic those of real-life spiders, making them formidable foes that challenge the player to use strategy and skill in order to defeat them. As the overall aesthetic of the game is grotesque, it is also important to ensure that the design of the enemies aligns with this style. This will help to further immerse the player in the game world and enhance the overall experience.

3.10 Level Design

The assets employed for level design, such as skyboxes and materials, are located in the Used Assets section B.

Verticality in level design, as discussed by Scott Rogers in *Level Up! The Guide to Great Video Game Design* [13], is a key feature that aligns with my game's design philosophy. By incorporating high platforms, walls, and ceilings accessible to players, I aim to add depth and excitement to the gameplay experience. Inspired by classics such as *Dusk*, *Ultrakill*, and the *Quake* series, levels will feature vertical elements that not only provide opportunities for exploration but also incorporate enemy encounters, secrets, and alternative routes. This emphasis on verticality will engage players, stimulate their senses, and enhance the overall player experience, keeping them coming back for more.

- **Non-Linearity:** Levels should encourage exploration and provide multiple paths to reach the end goal, a feature common to games such as *Dusk* and *Quake*. This non-linearity encourages players to replay levels, enhancing the game's longevity.
- **Verticality:** Given that the protagonist possesses spider-themed abilities, verticality should be a significant aspect of the level design. High platforms, walls, and ceilings should be accessible and incorporate enemy encounters, secrets, or alternative routes.
- **Environmental Storytelling:** Each level should subtly convey its narrative through the environment. This includes the layout, the aesthetic, and the objects placed within the level.
- **Difficulty Progression:** Each successive level should be incrementally more challenging, introducing more complex enemies and environmental hazards to keep players engaged and challenged.

Existing Levels

- **The Sandbox Level:** This experimental playground is designed to test various game mechanics. It is a buffet of enemy encounters, mobility challenges, and ability showcases that inform the design of the rest of the game. Sandbox serves as the crucible, allowing for the testing, refining, and polishing of mechanics before their implementation into more structured levels.
- **The Tutorial Level:** The tutorial level is designed to ease the player into the game. It introduces basic controls, movement mechanics, and spider-themed

abilities. The level has been designed to be low-stakes but engaging, offering ample opportunities for players to practice their newfound skills without the stress of high-difficulty encounters.

Future Levels

- **The Spiders' Den:** Levels should encourage exploration and provide multiple paths to reach the end goal, a feature common to games such as *Dusk* and *Quake*. This non-linearity encourages players to replay levels, enhancing the game's longevity.
- **The Hunt:** Given that the protagonist possesses spider-themed abilities, verticality should be a significant aspect of the level design. High platforms, walls, and ceilings should be accessible and incorporate enemy encounters, secrets, or alternative routes.
- **The Web of Time:** A high-stakes level where the player is racing against time to reach the end, all while battling enemies and avoiding obstacles. This level could provide a thrilling test of the player's skills and decision-making under pressure.
- **Boss Level - The Black Widow's Lair:** The final confrontation takes place in a sinister, labyrinthine lair. The player must navigate through the dark, twisting tunnels, solving puzzles, and overcoming obstacles, all while battling the deadly Black Widow. This ultimate test of the player's skills and tenacity would provide a fitting climax for the game.

These future levels reflect the core aspects of the game – exhilarating movement, strategic combat, and thematic consistency with the spider-inspired protagonist. Each level presents its unique challenges and play styles, engaging players in different ways and enhancing the overall replayability of the game.

3.11 Sound Design and Music

The sound design assets, including soundtracks and sound effects, are listed in the referenced section B.

The incorporation of a well-crafted soundtrack and sound design is crucial for creating an immersive and engaging video game experience. In this particular game, careful attention has been given to the audio elements to enhance the overall atmosphere and gameplay dynamics. The chosen musical style for this game is heavily influenced by rock, designed to evoke feelings of energy and aggression, ultimately encouraging a more serious and assertive gameplay experience.

Soundtrack Composition: The process of composing the game's soundtrack began approximately one year ago. The initial compositions were created to match the overall theme and tone of the game, ensuring that the music aligns with the intended player experience. The rock-influenced style of the soundtrack was selected to complement the fast-paced action and intense combat scenarios.

Main Menu Soundtrack: A specially crafted soundtrack has been incorporated into the main menu to captivate players and set the tone for the game. The music in this section is designed to create anticipation and excitement, drawing players into the game world from the start.

Level Soundtrack: Each level within the game features its own unique soundtrack, carefully composed to enhance the atmosphere and complement the gameplay. The music evolves and adapts to the challenges and progression of the level, intensifying during combat encounters or building suspense during exploration segments. This approach ensures that the music remains engaging and responsive to the player's actions, further immersing them in the game world.

Sound Effects: In addition to the soundtrack, various sound effects have been implemented throughout the game to provide audio feedback and enhance the player's interaction with the environment. These include:

- **Footstep Sounds:** Realistic footstep sounds have been created to match the movement of the player character. These sounds vary depending on the type of surface the player is traversing, such as metal, concrete, or dirt, enhancing the realism and immersion.
- **Weapon Sounds:** Each weapon in the game has its own distinct sound, creating a sense of uniqueness and individuality. Different variations of shooting sounds have been designed to differentiate between firearms, ensuring that players can audibly identify the weapon they are using.
- **Enemy Footstep Sounds:** To heighten the player's awareness and provide audio cues about nearby enemies, sound effects have been added to simulate the footstep sounds of hostile entities. This allows players to anticipate enemy encounters and react strategically.
- **Interactive Sound Elements:** Various interactive elements within the game world have specific sound effects associated with them. These sound effects are triggered when the player interacts with objects or activates switches, providing audio feedback that reinforces the player's actions and increases the sense of interactivity.

Sound Mixing and Balancing: Attention has been given to sound mixing and balancing to ensure that all audio elements blend harmoniously. The volume levels of different sound effects, music, and any potential voice-over have been carefully adjusted.

Chapter 4

Development of the game

The process of converting the game design document into a playable prototype involves a number of tasks. These include creating models, implementing functional movement mechanics, developing a flexible and easily modifiable weapon system, designing artificial intelligence, and perhaps most importantly, incorporating spider-inspired abilities. All of these elements must work together seamlessly to create a cohesive and engaging gameplay experience.

4.1 3D assets

At the start of the semester, I began work on the initial model for the game, specifically the model of the first monster. Throughout the process of 3D modeling, I utilized Blender as my tool of choice. The decision to use Blender [18] was based on my prior experience with the software, the abundance of tutorials available, and the fact that it is open-source and free to use. I used the following image as a reference for my model.

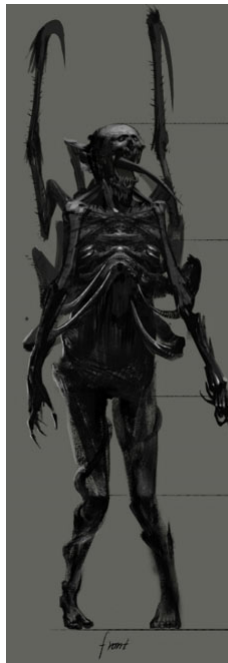


Figure 4.1. Reference used for the enemy model.¹

I began creating the model of my character by constructing the basic geometry using a simple cube. I scaled the cube to match the height of my reference image. I then made cuts into the cube and added a mirror modifier as the reference

image is symmetrical along the up-axis. To ensure that the mirrored parts do not interfere with each other, I used the clipping option. Next, I added a head to the model and applied a subdivision surface and mirror modifiers to refine the details. I then positioned the legs and arms differently and made adjustments to the face and stomach of the model.

After completing the initial character model, I began modeling the spider. The reference for this model is shown below.



Figure 4.2. Reference used for the spider model.²

I created a new cube and applied a subdivision surface to it, transforming it into a low-poly sphere. I then used the subdivision modifier again to precisely sculpt the spider's legs. Afterwards, I added eyes as separate objects, as they will require different materials in the future. Once that was completed, I made adjustments to the shape of the legs and the overall form of the spider, to achieve the desired result.

Once the spider model had been completed, I positioned it on the back of the zombie model, merged the two models together, and applied all necessary modifiers. I then applied textures to the body and made adjustments to address any defects that were visible on the model. The final result can be seen in the following screenshot.

My final modeling task revolved around applying textures to the model. The process of texturing the spider model is similar to the process used for texturing

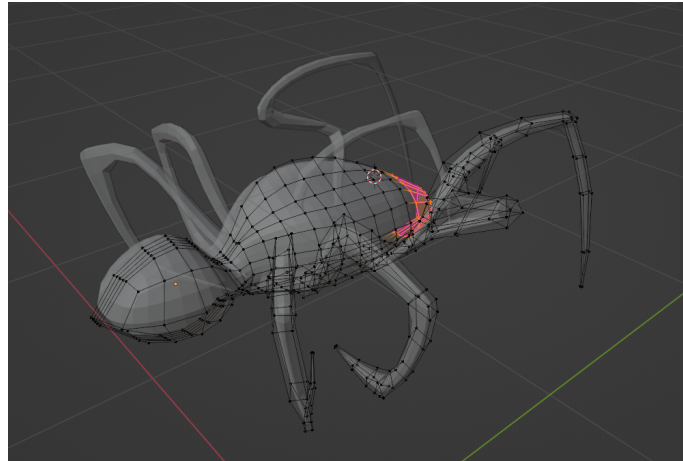


Figure 4.3. Finished spider model.

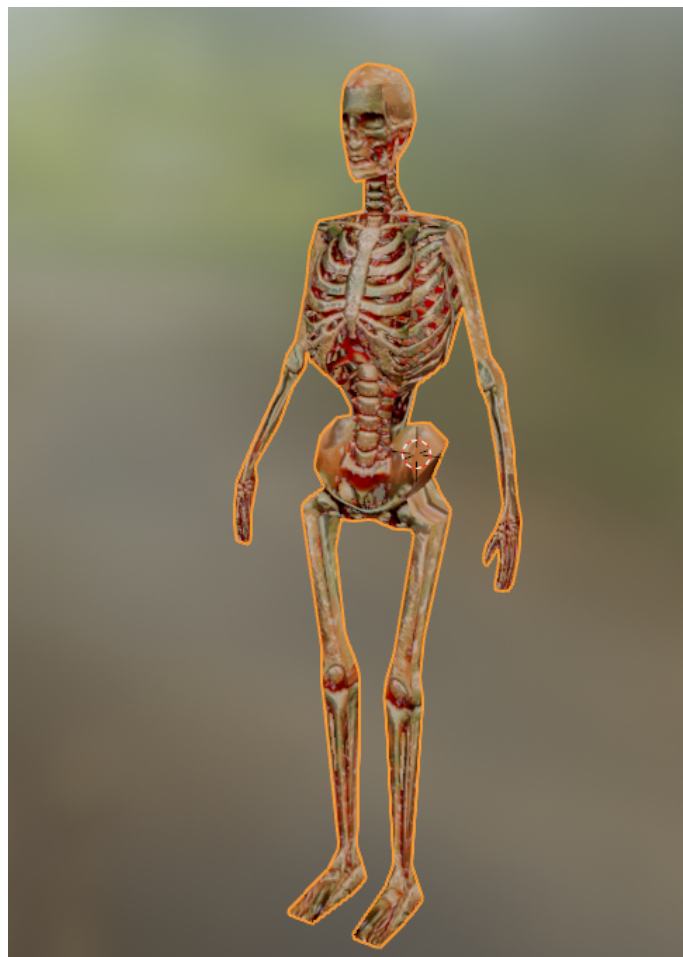


Figure 4.4. Textured body model.

the other models. I first applied the base texture and then went on to create and apply specular, roughness, bump, and normal maps. The final result can be seen below.

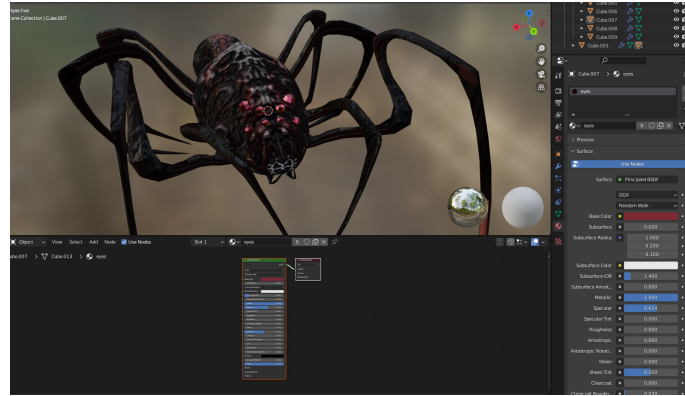


Figure 4.5. Textured spider model.

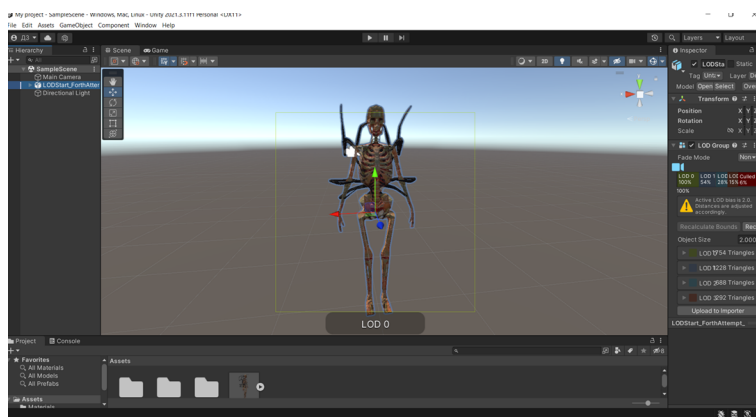


Figure 4.6. Model in Unity Engine.

Upon creating the texture, I created 4 levels of LOD³ and imported the finished model into Unity.

4.2 Scripts

The game's mechanics, including movement, object interactions, and artificial intelligence are all implemented through scripts. Scripts are text documents that contain code that represents the logic and functionality of the game. The programming language used for creating these scripts in Unity [19] is C#. In the following chapter, I will elaborate on the more complex logic and explain how it was implemented in the game.

4.2.1 Movement Mechanics

During the initial phases of the development, I focused on implementing the player's movement mechanics. Given that the game is a first-person shooter, ensuring smooth and responsive movement was paramount. To accomplish this, a collection of scripts was created to manage the character's and the camera's movements. The `MainCamera` and `MoveCamera` scripts were written to handle the rotation and field of view of the camera. The `MainCamera` script controls the

³ Level of Detail (LOD) is a technique that changes the mesh of the model depending on the distance from the viewer's camera.

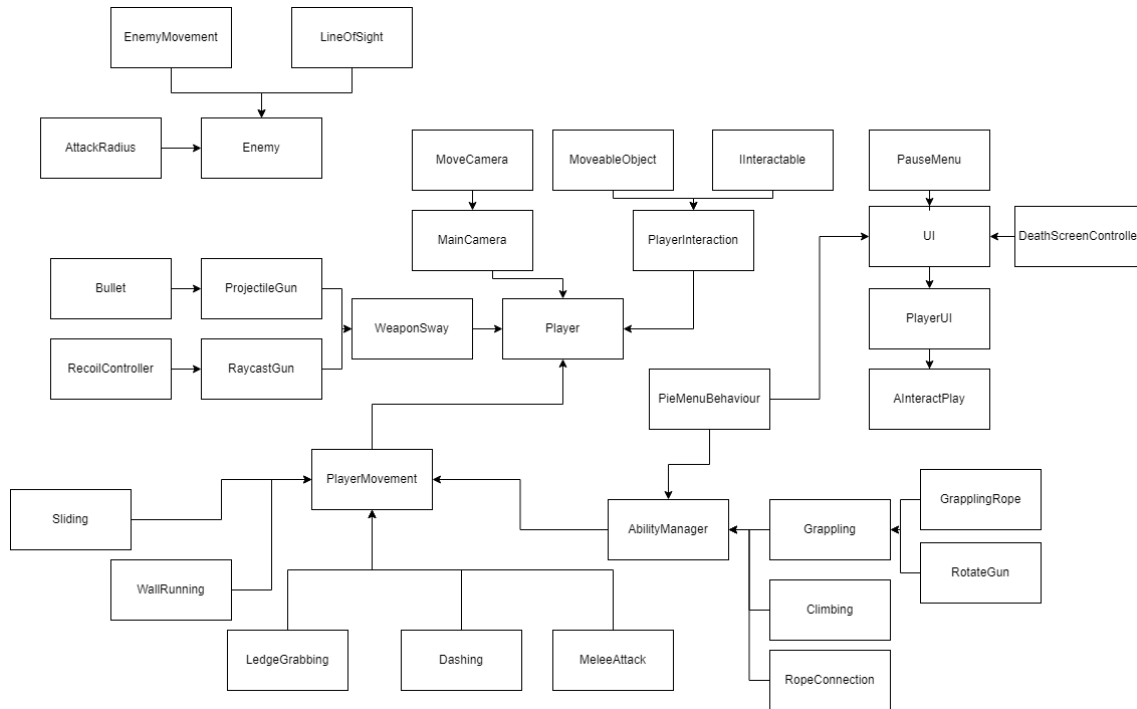


Figure 4.7. Class Diagram.

rotation and field of view of the camera, while the `MoveCamera` script tracks the player's world position and adjusts the camera's position accordingly. This helps to prevent jittering and other potential issues associated with directly attaching the camera to the player's body.

The `PlayerMovement` script, a critical component of the movement mechanics, was developed next. This script processes the player's keyboard input and uses it to navigate the player's avatar in the game. As Adams and Dormans (2012) [20] have emphasized in their book, 'Game Mechanics: Advanced Game Design', a variety of movement states such as Dashing, Climbing, Wall Running, Sliding, Crouching, Swinging, Sprinting, Grappling, and Swinging, were implemented, each managed by separate components to provide unique behaviors:

- The `Sliding` 4.2.2 component, for instance, checks if other movement states are active, adjusts the player's (and camera's) height when needed, and if the player is on a slope, it maintains sliding even after the normal cooldown has elapsed, thereby increasing the player's speed and momentum.
- The `WallRunning` 4.2.3 component enables the player to run alongside walls when not grounded and allows the player to crawl on walls when they fixate on them for a certain duration. This state can be exited at any time by pressing the Jump key.
- The `LedgeGrabbing` 4.2.4 component helps the player attach to a wall for a certain period, after which the player will drop off the wall.
- The `Dashing` 4.2.5 component propels the player's avatar in the direction the camera is pointing by applying force to the model.

Additionally, the `PlayerMovement` script is equipped to interact with various terrains and obstacles, including slopes and moving platforms, thanks to a built-in detection mechanism that responds appropriately when the character encounters different objects in the game world.

4.2.2 Sliding

The `Sliding` script is a component attached to the game character that enables sliding mechanics in the game. The script can recognize and process player input for initiating a slide, typically assigned to the 'left CTRL' key. The script checks if the player has pressed the slide key and whether the character is currently moving either horizontally or vertically. If these conditions are met, the script prepares the character for a slide.

Once the slide has started, the script modifies the scale of the character along the up-axis, reducing its height to simulate a sliding posture. Then, it adds a downward force to the character to mimic the effect of gravity during a slide.

While the character is in the sliding state, the script manages how the character moves. Depending on whether the character is sliding on a slope or not, it calculates the character's direction and applies a force in that direction. The script continuously reduces a timer that determines the maximum duration of a slide. If the character is no longer grounded (perhaps due to falling off a platform), the sliding movement stops.

When the sliding movement concludes, either due to the timer reaching zero or the player releasing the slide key, the script restores the character's original height and initiates a cooldown period, after which the character can slide again. During this cooldown period, the character cannot initiate another slide.

4.2.3 WallRunning

The `WallRunning` script is an essential component of a player character that enables the ability to run on walls.

The script allows the player character to interact with walls in a unique and exciting way. It detects and responds to player input, enabling the character to initiate wall runs, wall jumps, and climb walls.

When the player character approaches a wall, the script detects the presence of a wall and determines if the character can initiate a wall run. It checks for various conditions, such as the availability of a wall within range and the absence of obstacles, to ensure a successful wall run. During a wall run, the script applies forces to the character, allowing them to move along the wall's surface. It considers player input, such as pressing specific keys to control the character's movement, including ascending or descending on the wall.

The script also handles wall jumps, enabling the character to propel themselves off the wall to reach higher areas or overcome obstacles. It calculates and applies the necessary forces to execute the wall jump, considering factors like the direction and angle of the jump.

In addition to wall running and wall jumps, the script provides the capability to climb walls. It recognizes when the character is in proximity to a climbable surface and allows them to scale the wall by initiating the climbing mechanics.

Throughout these actions, the script manages the character's state, determining whether they are wall running, climbing, or in other states. It transitions between states based on input, wall detection, and specific conditions.

By incorporating this wall-running script into the game, players can engage in thrilling wall-running sequences, adding an element of agility and dynamism to their gameplay experience.

4.2.4 LedgeGrabbing

The `LedgeGrabbing` script is a component that empowers a game character with the capability to grab and hold onto ledges, as well as the ability to spawn ledges within the game world. The script allows the character to detect and interact with ledges, enabling them to grab and maintain a firm grip. It incorporates parameters and references to facilitate a seamless ledge-grabbing experience.

By pressing an 'F' key, the player can instruct the character to create a ledge. The script casts a ray forward from the character's position and checks for a suitable wall surface. If a valid wall surface is found within a specified distance, a ledge object is instantiated at the detected location, simulating the creation of a ledge for the character to grab onto.

Once a ledge is spawned or a ledge is detected within the character's proximity, the script enables the character to grab and hold onto the ledge. The player can initiate the ledge grab by pressing the designated jump key.

During the ledge grab, the script adjusts the character's position and movement to ensure a secure hold. It temporarily disables the character's gravity, preventing them from falling, and smoothly moves the character toward the ledge's position. This ensures accurate alignment with the ledge for a stable grip.

While holding onto the ledge, the character can perform a ledge jump by pressing the jump key. This action propels the character forward and upward, allowing them to release their grip on the ledge and execute a jump from the ledge's position.

Timing considerations are also implemented in the script. It tracks the duration that the character remains on the ledge, and if certain conditions are met, such as surpassing a minimum required time and providing input for movement, the character releases the ledge hold.

Additionally, a cooldown period is enforced after releasing the ledge grip. During this period, the character cannot immediately grab another ledge. Once the cooldown expires, the character regains the ability to grab ledges.

The `LedgeGrabbing` script enhances the gameplay experience by introducing the ability for the character to grab and interact with ledges. It provides players with opportunities for strategic navigation, dynamic movement, and creative exploration within the game world. Furthermore, the ability to spawn ledges adds an additional layer of interactivity and level design possibilities.

4.2.5 Dashing

The `Dashing` script is a component that adds the ability for a game character to perform a quick dash movement. It enhances the character's mobility and allows for dynamic and fast-paced gameplay. The script handles the character's dashing action, which can be triggered by pressing a designated 'Left Shift' key.

Upon pressing the dash key, the character executes a dash maneuver with the defined force and duration.

During the dash, various settings and effects come into play. The character is propelled in a specific direction, determined by the orientation of the character or the camera. The dash force and upward force contribute to the character's movement, allowing them to quickly cover a significant distance. Additionally, the script provides options to disable gravity during the dash, which can be useful for certain gameplay scenarios.

To ensure a smooth and controlled dash experience, the script includes cooldown functionality. After a dash is performed, a cooldown period is initiated, preventing the character from executing another dash immediately. The duration of the cooldown can be configured to suit the gameplay balance.

Visual effects are also incorporated to enhance the dash experience. The script adjusts the field of view (FOV) of the camera to provide a dynamic and immersive visual effect during the dash maneuver.

To handle the character's movement direction, the script considers input from the player. By analyzing the horizontal and vertical input axes, the script determines the desired movement direction for the dash. The character can dash forward, backward, or sideways based on the allowed movement directions configured in the script. Throughout the dash process, the script interacts with other components such as `PlayerMovement` and `MainCamera` to synchronize the character's movement restrictions, maximum vertical speed, and camera effects.

Once the dash is initiated, the script applies the calculated forces to the character's rigid body using impulse force mode. This imparts the necessary acceleration to the character, resulting in a rapid dash movement.

After the dash duration elapses, the character's dashing state is reset, allowing normal movement and physics to resume. Gravity is re-enabled if it was temporarily disabled during the dash. The camera's FOV is also reset to its default value.

The `Dashing` script contributes to the gameplay experience by introducing a burst of speed and agility to the character's movements. It enables players to perform swift evasive maneuvers, engage in intense combat scenarios, and navigate through challenging environments with speed and precision.

4.2.6 Abilities

The `Grappling` and `GrapplingRope` scripts work together to form a grappling hook feature that significantly enhances the player's gaming experience.

The `Grappling` script is essentially the **engine** that drives the grappling hook. It initiates two core actions: swinging and grappling toward a target point in the game world. This is triggered through specific commands by the player. To ensure fair gameplay and prevent immediate reuse, there's a built-in break between successive uses of the grappling ability. Moreover, the script offers extra controls for better precision during the swing action.

The `GrapplingRope` script, on the other hand, is responsible for the visual presentation of the grappling hook. It essentially generates a realistic, dynamic

representation of a rope that reacts to player's actions. It creates a visually pleasing effect of a rope swinging back and forth as the player moves, adding an extra layer of realism and immersion to the game.

In summary, these scripts work together to incorporate an exciting, fun, and visually appealing grappling hook feature into the game. They balance technical mechanics with player-friendly controls and realistic visuals to offer an engaging, immersive gaming experience.



Figure 4.8. Grappling Ability.

The `Spring` and `RotateGun` scripts are crucial additions that refine the grappling mechanism in the game, allowing for a more realistic and immersive gameplay experience.

The `Spring` script represents a physics-based model of a spring, incorporating factors such as strength and dampening. This script is designed to simulate natural spring-like motions in the game, which can include the recoil of a gun or a bouncing effect.

The `MyUpdate` function is a key component of the `Spring` script. It calculates and updates the state of the spring in real time, which helps create realistic movements. The `Reset` function, on the other hand, allows the spring to be brought back to its original state when necessary.

Meanwhile, the `RotateGun` script handles how the grappling gun moves in the game. It consistently checks the status of the grappling action and adjusts the direction of the gun accordingly. When the grappling action is not being used, the gun stays in line with the player's direction. But when grappling is initiated, the gun smoothly transitions to aim at the grappling point.



Figure 4.9. Swinging Ability.

Together, the `Spring` and `RotateGun` scripts provide enhanced depth to the game's grappling mechanism, significantly contributing to a more immersive and realistic gaming experience. The `Spring` script models lifelike movements of in-game elements, adding authenticity to the physics-based actions in the game. Simultaneously, the `RotateGun` script assures that the visual feedback of the grappling gun's orientation synchronizes seamlessly with the player's actions. The balance between realistic movements and accurate visual representation created by these scripts ensures a nuanced and engaging player experience. These scripts, thus, function as key technical enhancements that enrich the overall enjoyment and authenticity of the gameplay.

The scripts `PlayerInteractions`, `MoveableObject`, and `IInteractable` combine to create an innovative player interaction system in the game. This mechanism lets players link two objects in the game environment, causing these objects to move toward each other.

The `PlayerInteractions` script serves as the bridge between the player and the objects within the game. It identifies which objects around the player can be interacted with and enables the player to select these objects. When a player selects another object while maintaining the connection with the first one, the two objects are linked and begin to move toward each other. The script also changes how the object looks when it is chosen or released, giving the player a clear signal of their action.

The `MoveableObject` script is tied to game objects that can interact with the player. When an object is selected to connect with another, this script helps the objects move or 'gravitate' towards each other. It also manages how the object appears, returning it to its original look when it is no longer selected.

The `IInteractable` interface is crucial in changing how objects appear during player interaction, providing a visual response to the player's actions.

In summary, these scripts in combination create a unique interaction mechanism, enabling the player to select two objects, connect them, and cause them to move toward each other. This feature, achieved through a mix of physics, player inputs, and visual responses, creates an engaging and interactive gaming experience.

4.2.7 First Attempt: EnemyAI Script

I did not have much experience on the subject when I started work on the AI component, so I opted to first create a simple generalized component called `EnemyAI`. The script uses Unity's `NavMeshAgent` [21] component and the `Rigidbody` component to control enemy movement and physical interactions with the game world. It also uses a number of serialized fields, including a reference to the player character, layer masks for determining what the enemy can interact with, and various ranges and variables for determining enemy behavior.

The script includes several methods for controlling enemy behavior, including the `Update` method, which checks the distance between the enemy and the player and determines what action the enemy should take. It also includes methods for patrolling, chasing the player, and attacking the player. Additionally, the

script includes a method for taking damage, and if the enemy's health drops to zero, it will invoke a method to destroy the enemy game object.

4.2.8 Reworking the AI System

After some time and thought given to it, I began completely reworking the AI system. I started by making a script that controls the fundamental behavior of each enemy. I made it as an extension to a `PoolableObject` class to enable more efficient instantiation and destruction of objects in the game, essentially creating a pooling system. This system significantly reduces the performance cost of frequently instantiating and destroying objects, particularly relevant for enemies and other entities that frequently enter and exit the game. The `OnAttack` method is triggered when the enemy attacks, prompting the enemy to look at the target and animate the attack. When the enemy takes damage, it is managed by the `TakeDamage` method which decrements the enemy's health and plays a damage sound. If the health drops to zero or less, the enemy game object is deactivated, contributing to the object pooling system's efficiency.

The `EnemyMovement` script, an integral part of the game, effectively manages the movement and behavior of enemy characters. It efficiently employs Unity's `NavMeshAgent` [21], a powerful tool for handling complex navigation tasks. This script contains multiple enemy states, namely 'Idle', 'Patrol', and 'Chase', each defining a specific behavior pattern. In the 'Idle' state, the enemy roams aimlessly within a particular area. In the 'Patrol' state, it systematically follows a preset path, while in the 'Chase' state, the enemy becomes more aggressive, chasing the player relentlessly.

The transition between these diverse states is deftly handled by the script. It uses a sophisticated method, `HandleStateChanged`, to effectively switch between states based on the enemy's current situation and what is happening around them. Additionally, this script is intelligent enough to flip between the 'Patrol' and 'Chase' states, relying on whether the enemy has sight of the player or not.

The `EnemyLineOfSightChecker` script and the `EnemyMovement` script collaborate to create an essential component of the enemy AI's detection system. Through clever utilization of a `SphereCollider` component, the `EnemyLineOfSightChecker` script defines the enemy's field of vision. By employing raycasting techniques, it determines if the player is within this designated range, enabling the enemy to detect the player's presence.

Enhancing the capabilities of enemy AI, the `AttackRadius` script is responsible for defining the enemy's attack behavior. The `RangedAttackRadius` class further extends these capabilities, adding the ability for enemies to launch attacks from a distance.

Altogether, this complex interplay of scripts provides a robust and sophisticated framework for enemy AI. They work together to grant the enemy the ability to detect the player and respond accordingly. This intricate system amplifies the game's enjoyability, ensuring an immersive and engaging experience for all players.

4.2.9 The Role of ObjectPool System

The game employs an `ObjectPool` system for efficient management of bullet and enemy objects. These objects are stored in a pool for reuse, reducing the overhead of frequent instantiation and disposal. Objects are taken from the pool when needed and returned when not, contributing to smoother gameplay by ensuring efficient resource usage.

4.2.10 Procedurally Animated Spider

One of the most prominent features is the animation script powering the game's spider enemy archetype. This script brings to life the intricate movements of a spider through procedural animation and skilfully navigates the enemy around obstacles in the game.

Various elements come together to drive the animation of each leg of the spider. The script maintains arrays that hold important information about the position of each leg and its movements. The game constantly calculates and updates the position of each leg based on the spider's speed. When a leg needs to move a significant distance, the script initiates a stepping motion, resulting in a smooth, realistic movement that mimics a real spider's stride. The game also plays a footstep sound at the end of each step, adding another layer of realism.

In addition to the legs, the script also adjusts the orientation of the spider's body based on the positioning of the first pair of legs. This adjustment aids in enabling the dynamic and lifelike animation of the spider.

In essence, the `SpiderProceduralAnimation` script is a key component that heightens the authenticity of the spider-like enemy. It adds depth to the game by enhancing the movements, body orientation, and sound effects of the creature, resulting in a more immersive gaming experience.

4.2.11 Projectile Weapons

In any shooter game, the weapon system is a cornerstone feature. It is responsible for creating the 'action' component of gameplay and driving the core combat mechanics. The system can handle various types of weapons, from simple melee items to complex firearm mechanics, including reloading, shooting, spread, recoil, ammunition count, and even detailed bullet behavior. In this section, we will describe two scripts that play a key role in realizing a rich firearm system in my game.



Figure 4.10. Weapons used in my game.

I began by designing a weapon that fires projectiles, like bullets or other types of ammunition. This decision was made keeping in mind that I might want to add explosive weapons like RPGs or grenade launchers later on.

The script managing this weapon allows for various customizable features like shooting power, reload time, magazine size, and the number of bullets that can be fired at once. It also has a special feature that allows the weapon to keep firing as long as the fire button is held down. The central part of this script is the method that handles the firing of the weapon. This method calculates the direction of the bullet, taking into account factors like the spread of the gun, creates the bullet, and sets it in motion. It also manages the recoil of the gun each time it is fired.

Another important part of this script is the method that handles the reloading of the weapon. It ensures there is a delay before the weapon is ready to fire again, simulating the time it takes to reload in real life.

Each bullet that is fired has its own script that manages its behavior. This includes the physics properties of the bullet like how it bounces off surfaces, how it is affected by gravity, and how long it lasts before it explodes.

The method of managing the explosion of the bullet is triggered when the bullet hits a target or when it is time for the bullet to explode. This explosion affects nearby enemies and causes damage to them. It also keeps a count of how many times the bullet has hit non-enemy targets and triggers an explosion if it hits too many times or lasts too long.

In summary, the scripts for the projectile gun and the bullets work together to implement a versatile and engaging weapon system in the game. This system allows for a wide variety of firearm mechanics, making the gameplay more interesting and diverse.

4.2.12 Raycast Weapons

After successfully implementing and refining a physics-based projectile gun system in the game, I turned my attention toward raycast guns. Unlike projectile guns, which fire physical objects that move through the game world and interact with other objects, raycast guns utilize a concept called raycasting, which, as the name suggests, involves casting a ray, out from a point in a specific direction.

In my `RaycastGun` script, I designed a weapon system that responds instantly when the trigger is pulled. This immediate response is achieved using a method that casts a virtual ray straight from the gun and notes any objects in its path as 'hits'. I included options to customize the gun's features like its damage power, shooting range, impact force, fire rate, and recoil.

I am keeping track of the ammunition count using a counter that starts off at `magazineSize` and decreases with each shot. I also added a feature to reload the gun, which can be manually initiated by pressing the 'R' key and is also triggered automatically when the ammo runs out. During the reload time, the gun cannot be fired, making the game feel more real.

When the left mouse button is pressed, the shooting action is initiated. This action triggers the `muzzleFlash` effect, reduces the ammo count, and causes the

gun to recoil. At the same time, a virtual ray is sent out from the camera's position, scanning for any object that falls within the defined `canBeShot` layer mask.

If a target is hit, the script checks if the object has an `Enemy` component and applies damage if it does. The script also looks for a `Rigidbody` component on the target, and if it finds one, applies a force in the opposite direction to simulate the impact of a bullet.

For hits on non-enemy targets, the script creates a `bulletHole` graphic at the point of impact. Regardless of what kind of target is hit, an impact effect is created at the point of hit, making the weapon feel more realistic with an immediate effect.

4.2.13 Additional Weapon Mechanics

To make the gameplay even more exciting and engaging, I have been working on three additional scripts that enhance the weapon mechanics: `RecoilController`, `WeaponSway`, and `WeaponSwitching`. These scripts add more dynamics to the gameplay and make the gaming experience more enjoyable for players.



Figure 4.11. Recoil Demonstration.

The purpose of the `RecoilController` script is to make the weapons feel real by simulating recoil. It adds a bit of a challenge to the shooting experience, making each shot have more impact. When a shot is fired, the `ApplyRecoil` method is called, which creates a recoil effect by tweaking the `currentRecoil` based on a randomly defined range. With every game update, the script makes the camera jolt with each shot fired and then eases it back to its original position, just like in real life when someone fires a weapon and then steadies it.

To make the game feel more real, I have added the `WeaponSway` script. This script makes the weapons bob and sway as the player moves around and looks in different directions, creating a more immersive first-person experience. The script calculates the sway and bob based on the player's actions. Sway is calculated based on where the player is looking while bobbing is tied to the player's movement. These calculations are then used to adjust the weapon's position and rotation, creating a fluid motion that follows the player's movements. There are

also additional settings to customize how the weapon moves during specific actions, like when the player is jumping.

I have also added the `WeaponSwitching` script to give players a chance to choose their weapons strategically. This script allows players to switch between different weapons using the mouse scroll wheel or the number keys. This is done by making the selected weapon active and the others inactive. With this system, players can quickly adjust to changing combat situations in the game, giving them more control and making the game more fun to play.

By using these scripts along with the `RaycastGun` script, I have managed to create a complete and engaging shooting experience in my game.

4.2.14 Hand-to-Hand Combat

I have also added a feature that allows players to engage in hand-to-hand combat, introducing a whole new strategic layer to the game.

The hand-to-hand combat feature is activated with a simple keypress, which in the current setup is the 'G' key. Players can personalize the feature, adjusting how far they can reach with an attack, how much delay there is before an attack lands, how quickly the attack happens, and how much damage each hit does.

A standout part of this feature is the ability to perform targeted attacks. When a player starts an attack, the game looks at where the main camera is pointing. It then sends out a signal in that direction and checks if it hits an enemy. If it does, it creates a visual effect at the point of impact and reduces the enemy's health by the set attack damage.

To make combat feel more realistic, the feature also includes a delay between attacks. This prevents players from launching a flurry of instant attacks, instead making them wait a short period between each attack.

The combat feature also includes placeholders for attack sound and animation. These can be added later to make hand-to-hand combat even more immersive and thrilling.

By combining this hand-to-hand combat feature with the existing weapon mechanics, the game now offers a wide range of combat options. This variety lets players make tactical decisions based on what is happening in the game, creating a richer and more engaging gameplay experience.

4.2.15 Pie Menu

To enhance the user interface, I created a `Pie Menu`, also known as a `Radial Menu`, which is a widely utilized feature in many video games. This menu provides players with a convenient method for quickly selecting abilities, items, or actions. The `Pie Menu` operates with the assistance of three scripts that effectively manage and control its behavior.

The implemented circular menu allows players to effortlessly switch between various abilities such as swinging, connecting objects, or employing a grappling hook. By tracking the mouse's position, the menu determines the selected ability based on its angle relative to the center of the screen. Consequently, any change in the player's selection prompts the menu to update accordingly, effectively highlighting the newly chosen ability.

To activate or deactivate the pie menu, players can utilize the 'Q' key. Upon pressing this key, the game enters a slow-motion state, creating a visually captivating effect. Simultaneously, the mouse cursor is unlocked, and the pie menu is displayed for easy access. Releasing the 'Q' key promptly resumes the game and conceals the menu once again.

Furthermore, each ability within the pie menu undergoes visual transformations when selected or deselected, ensuring clear and immediate feedback for the player. This visual feedback significantly facilitates swift and efficient ability selection, relying solely on the mouse and a single key.

In conclusion, the cohesive system comprised of these scripts allows for a seamless Pie Menu experience. Users can effortlessly choose abilities using their mouse and a simple key press, all while receiving visual feedback that indicates the currently selected ability.

4.2.16 Pause Menu and Death Screen

After finalizing the Pie Menu 4.2.15, I also added a Pause Menu and a Death Screen. Both of these are common features in many games. The Pause Menu is activated by pressing the 'Escape' or 'P' key. When activated, the game is frozen, the pause menu appears, the mouse cursor becomes visible and mobile, and the game's audio is either silenced or stops. The pause menu offers several options like restarting the game, adjusting settings, or quitting to the main menu.

Once the game is unpaused, the Pause Menu disappears, the game resumes its normal speed, the mouse cursor is hidden and position-locked, and the game's audio resumes. This pause menu offers players a way to interact with the game's settings and options without exiting the game entirely.

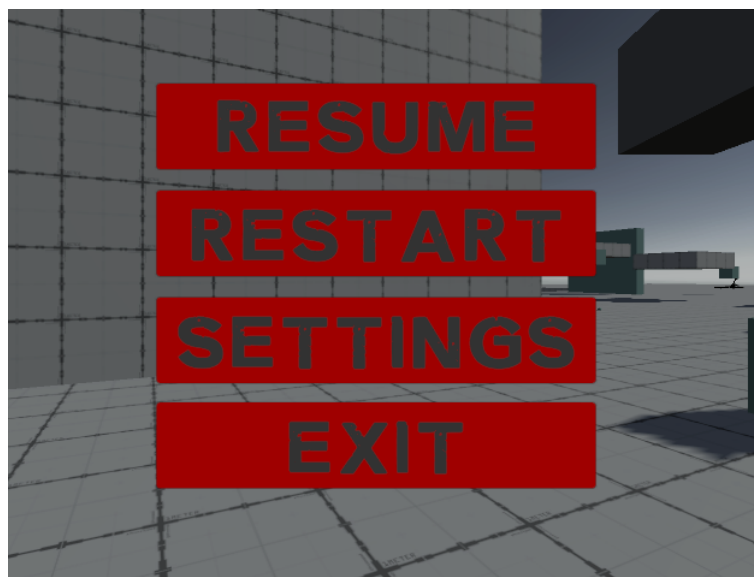


Figure 4.12. Pause Menu.

The `DeathScreenController` script manages the appearance and behavior of a death screen when the player dies. The `ShowDeathScreen` method activates



Figure 4.13. Death Screen.

the death screen, freezes the game by setting the timescale to 0, and allows the mouse cursor to move freely and be visible. The `HideDeathScreen` method does the opposite – it deactivates the death screen, sets the timescale back to normal, and hides and locks the mouse cursor.

The script for the game over screen manages when and how this screen shows up after the player’s character loses. It stops the game, makes the mouse cursor visible, and lets the player use it freely. The game over screen is then hidden, the game returns to normal, and the mouse cursor disappears when we are ready to continue.

Furthermore, the script manages the volume controls for different audio aspects of the game. It communicates with the `AudioMixer` to manipulate the Master, Music, and SFX volumes via corresponding sliders. The respective volume levels are stored in `PlayerPrefs`, ensuring their persistence between game sessions. Each slider’s value is also displayed as a label, updating dynamically as the sliders are adjusted.

By providing a comprehensive options menu, the `OptionsScreen` script allows the player to customize the game environment to their preference, optimizing their overall gameplay experience.

All of these features, from the in-game ability selection tools like the `PieMenu`, `PieMenuBehaviour`, and `MenuItem` scripts, to game management tools like the `PauseMenu` and `DeathScreenController`, to game customization tools like the `OptionsScreen`, work together to create a well-rounded gaming experience. This combination of tools helps make the game more enjoyable and user-friendly. It also moves the game closer to being a complete, polished product.

4.2.17 Main Menu

The `MainMenu` script controls the main menu of the game. It includes a place to set the names of the first level and the sandbox scene. This script also connects to the options menu, so the options can be shown when needed.

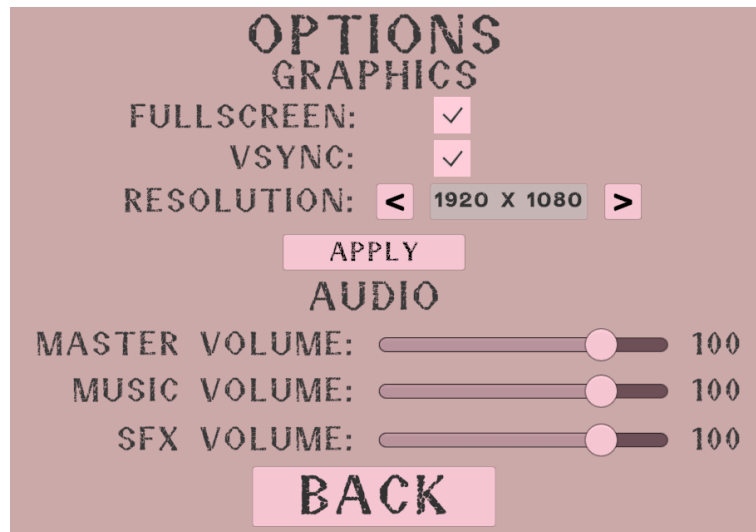


Figure 4.14. Options Screen.



Figure 4.15. Main Menu.

When the `PLAY` button is clicked, the `StartGame` method opens the first tutorial level. When the `SANDBOX` button is clicked, the `Sandbox` method opens the sandbox level. The `OpenOptions` and `CloseOptions` methods show and hide the options menu when the `OPTIONS` button is clicked. The `QuitGame` method closes the game when the `QUIT` button is clicked.

The `AudioManager` script handles the game's sound settings. When the game is launched, it finds the saved audio settings from previous play sessions and uses these settings to set the sound levels. These settings control the Master, Music, and SFX volumes, allowing players to start their game with their chosen sound levels.

The `OptionsScreen` script offers a detailed interface for players to adjust their game settings. Players can change the screen resolution and switch to full-screen mode.

The `SoundtrackPlayer` script adds to the game's sound environment. It includes a selection of soundtracks and an ambient sound. It uses two `AudioSource` components, one for the soundtrack and one for ambient sounds. When the game



Figure 4.16. Main Menu Options.

starts, it picks a random soundtrack and starts playing it alongside the ambient sound.

Chapter 5

User Testing

In order to gather valuable insights and improve the quality of my game project, I conducted a thorough testing phase with a focus group. The objective was to assess the game's mechanics, controls, technical aspects, and elements of fun, and gather suggestions for further development. This chapter presents a summary of the testing process and the feedback received from the participants, shedding light on both the strengths and areas for improvement in the game.

To gather feedback, I distributed the game build to a diverse group of individuals and posed several key questions. The focus group participants were asked to provide their opinions on the mechanics, controls, technical issues, elements of fun, sound and music, weapon functionality, and overall potential of the game. Their responses were analyzed, and common themes and patterns were identified to guide future development and refinement.

Participants: The focus group consisted of players with varied gaming backgrounds, all of whom have experience with first-person shooter (FPS) games. As these participants were familiar with similar games, their feedback was expected to be insightful and relevant to the development of this game. Some of the participants had casual experience with FPS games, while some had prior involvement in the competitive scene. Some of the participants had nostalgic connections to classic games like Unreal Tournament and Quake, providing valuable perspectives on how the game could evoke similar enjoyable experiences.

The questions posed to the focus group participants were as follows:

1. What mechanics would you change in the game?
2. What mechanics would you add to the game?
3. Did you find the control scheme intuitive and user-friendly?
4. Did you experience any technical issues or bugs during gameplay?
5. What elements of the game made it fun for you?
6. How engaging did you find the sound and music in the game?
7. Were there any weapons or tools you found particularly useful or enjoyable?
8. Were there any weapons or tools you found underpowered or underwhelming?
9. If there was one thing you could change about the game, what would it be?
10. Do you see potential in the game if the development continues?
11. Are there any other comments or suggestions you would like to share about the game?

Finding:

1. **Mechanics.** The game's mechanics were generally well-received by the focus group. Participants appreciated the fast learning curve and the unique incorporation of spider-like abilities, describing them as fun and engaging. However, one participant pointed out the need for better explanations regarding grapple mechanics.

2. **Controls.** The control scheme was generally deemed intuitive and user-friendly. However, some participants expressed a desire for additional feedback cues, such as cooldown, or reload notifications, to enhance gameplay clarity. One of the participants also mentioned a slight lack of precision and responsiveness in the controls, comparing it to a feeling of roller-skating in space.
3. **Technical Issues.** Several technical issues and bugs were identified during gameplay. These included an unclickable ESC menu UI in the tutorial, improper scaling of the damage vignette, instances of falling through the game world, and occasional issues with character movement and collisions. Addressing these technical issues would greatly enhance the overall experience of the game. The majority of technical issues raised during testing have been successfully addressed and resolved, ensuring a smoother and more enjoyable gameplay experience.
4. **Elements of Fun.** The focus group found the game's unique mix of spider mechanics, gunplay, and horror elements to be enjoyable. The fast-paced nature and consistent feel of the game, coupled with the well-composed music, evoked nostalgic memories of classic games like Unreal Tournament and Quake.
5. **Sound and Music.** The music received high praise, with participants considering it to be of exceptional quality. However, some participants mentioned that other sound effects felt somewhat subdued, suggesting the need for a better balance between sound elements. Additionally, softer or more subtle footstep sounds were suggested to enhance immersion.
6. **Weapons and Tools.** The grapple mechanics and gun recoil were identified as particularly enjoyable and well-implemented. However, participants expressed a desire for the other weapons to be more powerful and suggested improvements to enemy attacks to balance gameplay. Lack of clarity regarding cooldowns and reloads for certain tools were also mentioned as an area for improvement.
7. **Potential and Genre Interest.** Participants expressed interest in the game's genre and believed it had the potential for further development. They appreciated the unique mix of gameplay mechanics and the opportunity for an engaging storyline. One participant suggested the inclusion of more narrative elements to enhance understanding of the game's context.

The testing phase provided valuable insights into the game's strengths as well as areas that could be improved. The feedback highlighted the engaging mechanics, intuitive controls with room for refinement, technical issues that require attention, elements of fun, and the potential for further development. By addressing the identified issues and incorporating suggested improvements, the game can evolve into a more polished and captivating experience. The positive reception from the focus group underscores the interest in the genre and supports the potential for success if the development proceeds.

By conducting user testing, I gained valuable insights that helped guide me in refining and enhancing the game, paving the way for a successful conclusion to my bachelor thesis project.



Chapter 6

Future Development

As the development of my 3D Unity shooter game progresses beyond the scope of my bachelor thesis, there are several exciting avenues for future development and expansion. This chapter outlines my plans to address existing issues, introduce new gameplay elements, enhance the narrative, and further polish the overall experience.

First and foremost, I aim to address any remaining bugs and technical issues identified during testing. This will ensure a smooth and seamless gameplay experience for players. Additionally, I plan to create new levels that fully utilize the game mechanics, offering diverse and engaging challenges to players.

To enrich the gameplay experience, I have envisioned several new mechanics to be implemented. These include introducing bunnyhopping as a movement technique, allowing players to traverse the game world with speed and agility. Furthermore, I plan to enhance the player's ability to manipulate objects by incorporating a 'stun' ability, where enemies can be immobilized and connected to other objects or enemies for increased damage or explosive effects.

Adding a boss encounter will provide players with an exciting and challenging experience. I intend to design a formidable boss character that requires strategic thinking and skillful execution to defeat. Additionally, incorporating compelling lore and plot will add depth to the game, immersing players in a spider-themed world and providing motivation to explore and progress further.


To strengthen the thematic cohesion of the game, I plan to create original 3D assets that align with the spider world narrative. This includes redesigning enemies to be spider-like, adapting weapons to have a more spider-themed appearance, and incorporating sounds that emulate the noises spiders make. Furthermore, refining animations and adding visual effects will contribute to a more immersive and visually appealing experience.

To enhance user experience, I will focus on making the user interface (UI) more intuitive and visually pleasing. This involves streamlining the UI elements, improving readability, and providing clear feedback on cooldowns, reloads, and other important game information. By creating a smooth and seamless UI, players will be able to navigate the game more efficiently.

As the development of the game continues, I am open to suggestions and ideas from players and the community. Collaboration and feedback are essential in creating an exceptional gaming experience. Therefore, I welcome suggestions for new mechanics, level designs, visual enhancements, or any other aspects that can further enrich the game and captivate the players.

The future development of my 3D Unity shooter game holds great potential for growth and improvement. Through bug fixes, expanded gameplay mechanics,

boss battles, enriched narrative, enhanced visuals and audio, improved UI, and an open-minded approach to suggestions, I aim to create a captivating and immersive experience for players. The journey to refining and expanding the game beyond my bachelor thesis is an exciting one, and I look forward to the challenges and opportunities that lie ahead.



Chapter 7

Conclusion

The objective of this bachelor thesis project was to develop a 3D first-person movement shooter game inspired by retro shooters such as Dusk and Ultrakill. The project involved analyzing the mechanics of these games, creating a story, and specifying design concepts. This chapter serves as a conclusion, summarizing the completion of each task from the project's guidelines and outlining future development possibilities.

1. **Analysis of Retro Shooters and Design Concepts.** Extensive analysis was conducted on retro shooters like Dusk and Ultrakill, examining their gameplay mechanics and design elements. Based on this research, a story was created, and design concepts were specified, setting the foundation for the game's development.
2. **Movement System.** A comprehensive movement system was defined and implemented. This system allows players to navigate the game world with agility and precision, providing a fluid and responsive gameplay experience.
3. **Combat System.** A combat system was designed, incorporating both melee and ranged mechanics. Players can engage in intense close-quarters combat as well as utilize various firearms to engage enemies from a distance. These mechanics were implemented to provide a dynamic and satisfying combat experience.
4. **Intuitive User Interface.** An intuitive user interface (UI) was developed, enabling players to select and utilize abilities seamlessly. The UI design focused on providing clear and accessible options for ability selection and activation, enhancing the player's control over their character's capabilities.
5. **Enemy Types and Behavior.** Three distinct enemy types were outlined, each with unique movement and combat capabilities. The design considerations for these enemies encompassed strategic movement patterns and diverse attack behaviors, creating engaging and challenging encounters for players.
6. **Playable Prototype.** A playable prototype was implemented, incorporating the defined mechanics and a limited gameplay area. This prototype served as a proof of concept, allowing for testing and iteration to refine the gameplay experience.
7. **User Testing and User-Centered Design.** The prototype was tested with a minimum of three participants, employing user-centered design principles throughout the development process. User feedback was gathered and utilized to identify areas for improvement, refine mechanics, and address any usability issues.
8. **Future Development Opportunities.** Having accomplished the outlined tasks and successfully completed the bachelor thesis project, numerous opportunities for future development and expansion of the game emerge. **These include:**
 - Expanding the game world with additional levels and environments, providing players with a diverse and immersive experience.

- Further refining and balancing the combat system, ensuring engaging and satisfying encounters with enemies.
- Enhancing the visual and audio elements of the game, including refining 3D models, textures, sound effects, and music, to create a cohesive and immersive atmosphere.
- Deepening the narrative and incorporating more lore and plot elements to enhance player engagement and provide a compelling context for their actions.
- Continued bug fixing and polishing to ensure a seamless and enjoyable game-play experience for players.

This bachelor thesis project has successfully achieved the defined tasks and objectives outlined in the project guidelines. Through the analysis of retro shooters, the development of movement and combat systems, the implementation of an intuitive user interface, the design of enemy types and behaviors, the creation of a playable prototype, and user testing, a solid foundation has been established for the future development of the game.

Moving forward, the project presents numerous opportunities for expansion and improvement. By focusing on refining gameplay mechanics, enhancing the visual and audio elements, and incorporating a captivating narrative, the game has the potential to become a compelling and enjoyable experience for players. The completion of this bachelor thesis project marks a significant milestone in the game's development journey, and I am excited to continue working on the game and bringing it closer to its full potential.



References

- [1] *Doom (1993) on Steam*.
https://store.steampowered.com/app/2280/DOOM_1993/. Accessed on 06.07.2020.
- [2] *Quake on Steam*.
<https://store.steampowered.com/app/2310/Quake/>. Accessed on 15.07.2020.
- [3] *Ultrakill on Steam*.
<https://store.steampowered.com/app/1229490/ULTRAKILL/>. Accessed on 15.07.2022.
- [4] Gerald Voorhees. *Shooting*. In: Bernard Perron, eds. *The Routledge Companion to Video Game Studies*. Taylor & Francis, 2014. 251–258. ISBN 9781136290503.
- [5] FUNKe. *More movement FPS - Funke Study*. 2021. Available On YouTube:
<https://www.youtube.com/watch?v=oRTh2aMgPY4>. Accessed on 13.10.2022.
- [6] *Dusk on Steam*.
<https://store.steampowered.com/app/519860/DUSK/>. Accessed on 02.07.2022.
- [7] *Severed Steel on Steam*.
https://store.steampowered.com/app/1227690/Severed_Steel/. Accessed on 20.07.2022.
- [8] Wikipedia. *Bullet time* — *Wikipedia, The Free Encyclopedia*. 2023.
https://en.wikipedia.org/wiki/Bullet_time. Online; accessed 24-May-2023.
- [9] *Ghostrunner on Steam*.
<https://store.steampowered.com/app/1139900/Ghostrunner/>. Accessed on 29.06.2022.
- [10] *Neon White on Steam*.
https://store.steampowered.com/app/1533420/Neon_White/. Accessed on 23.04.2023.
- [11] *Turbo Overkill on Steam*.
https://store.steampowered.com/app/1328350/Turbo_Overkill/. Accessed on 03.05.2023.
- [12] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC Press, Taylor; Francis Group, 2020. ISBN 1138632058. Accessed on 12.07.2020.
- [13] Scott Rogers. *Level Up! The Guide to Great Video Game Design*. Hoboken, NJ: Wiley, 2010. ISBN 9780470688670. Accessed on 14.03.2023.
- [14] T. Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Morgan Kaufmann, 2008. ISBN 0240809742. Accessed on 28.03.2023.

- [15] Alex Wiltshire. *Designer Interview: Getting Titanfall's controls just right*. 2017. Available:
<https://www.gamedeveloper.com/design/designer-interview-getting-i-titanfall-i-s-controls-just-right>. Accessed on 20.10.2022.
- [16] Michael Sellers. *Advanced game design: A systems approach*. Addison-Wesley, 2018. ISBN 0134667603. Accessed on 20.11.2022.
- [17] Jenifer Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Media, Inc., 2010. ISBN 1492051969. Accessed on 03.02.2023.
- [18] Blender Foundation. *Blender documentation*.
<https://docs.blender.org/>.
- [19] Unity Technologies. *Unity user manual 2021.3 (LTS)*.
<https://docs.unity3d.com/Manual/index.html>.
- [20] Joris Dormans Ernest Adams. *Game Mechanics: Advanced Game Design*. Berkeley, CA: New Riders, 2012. ISBN 9780132946681. Accessed on 08.02.2023.
- [21] Unity Technologies. *Navmeshagent*.
<https://docs.unity3d.com/ScriptReference/AI.NavMeshAgent.html>.
- [22] *Animated Cosmos Shader*.
<https://assetstore.unity.com/packages/vfx/shaders/animated-cosmos-shader-155006>. Accessed on 15.05.2022.
- [23] *Sci fi Drones*.
<https://assetstore.unity.com/packages/3d/characters/robots/sci-fi-drones-90326>. Accessed on 23.04.2023.
- [24] *Human Mutant*.
<https://assetstore.unity.com/packages/3d/characters/creatures/human-mutant-140704>. Accessed on 20.04.2023.
- [25] *Fantastic Creature 1*.
<https://assetstore.unity.com/packages/3d/characters/creatures/fantastic-creature-1-103074>. Accessed on 20.12.2022.
- [26] *Particle Pack*.
<https://assetstore.unity.com/packages/vfx/particles/particle-pack-127325>. Accessed on 14.01.2023.
- [27] *Simple Particles FX: Toon Effects*.
<https://assetstore.unity.com/packages/vfx/particles/simple-particles-fx-toon-effects-244171>. Accessed on 28.05.2023.
- [28] *Spider Orange*.
<https://assetstore.unity.com/packages/3d/characters/robots/spider-orange-181154>. Accessed on 20.12.2022.
- [29] *Weapon Master-SciFi Weapon.1*.
<https://assetstore.unity.com/packages/3d/props/weapons/weapon-master-scifi-weapon-1-126330>. Accessed on 12.05.2023.
- [30] *Stylized Materials Pack*.
<https://eldamar-studio.com/product/stylized-materials-pack/>. Accessed on 10.05.2023.
- [31] *100 Horror LUTs Pack*.
<https://eldamar-studio.com/product/100-horror-luts-pack/>. Accessed on 10.05.2023.

-
- [32] *Horror Fonts*.
<https://eldamar-studio.com/product/100-game-fonts-pack/>. Accessed on 10.05.2023.
- [33] *Ambient Soundscapes*.
<https://eldamar-studio.com/product/ambient-soundscapes-pack/>. Accessed on 10.05.2023.
- [34] *Ambient Soundscapes*.
<https://eldamar-studio.com/product/100-horror-icons-pack/>. Accessed on 10.05.2023.
- [35] *ProBuilder*.
<https://docs.unity3d.com/Packages/com.unity.probuilder@5.0/manual/index.html>. Accessed on 14.01.2023.



Appendix **A**

Abbreviations and symbols



A.1 Abbreviations

3D	three-dimensional
(G)UI	Graphical User Interface
AI	Artificial Intelligence
LOD	Level of Detail
DPS	Damage per Second
RPG	Rocket-Propelled Grenade
FPS	First-person Shooter

Appendix B

Used Assets

Asset name	Author
Animated Cosmos Shader [22]	Specoolar
Sci fi Drones [23]	Lukas Bobor
Human_Mutant [24]	Vadim Ziambetov
Fantastic Creature 1 [25]	GrigoriyArx
Particle Pack [26]	Unity Technologies
Simple Particles FX : Toon Effects [27]	Indian Ocean Assets
Spider orange. [28]	Dmytro
Weapon Master-SciFi Weapon.1 [29]	xiaolianhuastudio
Stylized Materials Pack [30]	Eldamar Studio
100 Horror LUTs Pack [31]	Eldamar Studio
Horror Fonts [32]	Eldamar Studio
Ambient Soundscapes [33]	Eldamar Studio
100 Horror Icons Pack [34]	Eldamar Studio
ProBuilder [35]	Unity Technologies

Table B.1. Used assets in the game.



Appendix C

Attachments

`SpithreEngine.zip` Unity project files for the game, featuring all the assets, scripts, scenes, and prefabs required to understand, modify, and build the game.

`SpithreBuild.zip` compiled build of the game, providing an executable version ready for direct gameplay on compatible systems without the need for Unity or any additional development software.