Ethan Chan
246667

AceUP: A game about Blackjack and Cheating

Project Supervisor: Paul Newbury

BSc Games and Multimedia Environments
Department of Informatics
University of Sussex
2024

Statement of Originality

This report is submitted as part requirement for the degree of Games and Multimedia Environments at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years.

Candidate Number: 246667

X 

Ethan Chan

Date Signed: 21/04/2024

Acknowledgements

The idea of a video game with two vastly different genres has stuck in my mind ever since participating in my first game jam. As such, this project has meant a lot to me as it symbolises how far I have come as a student and as a game developer to be able to create such a game. For this reason, I'd like to thank my supervisor and academic advisor – Paul Newbury for guiding me on this project. I'd also like to thank my family for their endless support as well as all of those who have helped playtest this game. You have helped make this game a reality.

Abstract

“AceUP“ is a unique game combining the classic card game, Blackjack, with a first-person shooter. Blackjack is a game typically played in casinos against a dealer with the main goal of achieving twenty-one. At it’s core, Blackjack is a game of probabilities, usually drawing cards from a deck of fifty-two or more. Each round, players are given the option to hit (request an additional card) or to stand (end their turn). This creates the suspense of the game as every card drawn could lead to your loss. Primarily a luck-based game, AceUP puts a twist on this concept by incorporating strategic elements such as cheating to gain an advantage against your opponents at the risk of starting a free-for-all fight where you must eliminate all opponents you’ve had to play against that round.

This paper goes into the design and development of this game and experiments with the concept of combining genres to create a new style of gameplay. The paper begins by discussing my motivations and findings, followed by requirements, and then justifying the design choices made and the implementation of this game. Finally we will conclude with the results of this project and the overall successfulness via user playtesting.

Contents

1. Introduction

1.1. Project overview

This project experiments with the idea of combining two established game genres; a tabletop game and a first-person shooter, in order to create a new unique game. In this project I aim to test if new gameplay can emerge between two completely different categories' of video games and whether it would create a fun and engaging experience for the player.

This project introduces a new concept into the popular card game: Blackjack, where the typical rules are usually to get as close to twenty-one without going over. In AceUP, players are allowed to cheat and swap cards in their hands with the risk of starting a fight if they are caught. This project consists of many complex features such as an item management system, simulated AI in opponents, and AI navigation using Unity's NavMesh system. This will be covered in further details in later chapters of this report.

1.2. Motivations

As a Games and Multimedia Environments student, this project allows me to create a new game independently using skills I have learnt in my course. This includes 3D asset creation learnt from 'Programming for 3D' and '3D Modelling and Animation'. As well as game development skills from modules such as 'Game Design and Development'. The finished game will also be used after my studies to show potential employers alongside my portfolio.

The concept of mixing and combining two genres came about when I first participated in a game jam in sixth form college [1]. Due to lack of experience as a beginner programmer alongside a short two day deadline, I was unable to complete the game. The final-year project has given me a chance to retry what was once a failed project, starting from scratch to document how I have improved over the years of studying game development at university.

I chose to create a tabletop game such as Blackjack as the rules are easy to understand but can get quite complex when trying to weigh the pros and cons of requesting an additional card, adding to the adrenaline rush of a perfect hand '21'. During my childhood, Blackjack was one of the card games I played the most with my family. We enjoyed it's simple premise and recognised that it practised simple arithmetic at a young age. The inclusion of the cheating mechanic was a choice made during my first game jam as it tied together the two genres whilst also challenging myself to add a complex feature such as card management into the game which provided an opportunity to learn new programming skills.

1.3. Findings + Achievements

This project experiments with the merging of two genres, as such, a decent amount of research was done on games which dabble with this idea. Video games such as Dungeons and Degenerate Gamblers, FPSChess, and Portal [2, 3, 4] all combine different genres in different ways, each providing a unique and engaging experience. In these aforementioned games, FPSChess most closely resembled the idea of AceUP, therefore further research on the game was done (See Appendix A).

What makes FPSChess a fun and engaging experience is that it offers a risk and reward system to the game, where players weigh the consequences of capturing a piece which starts a fight. This flips the game from a typical strategy game into one where it requires skill to win therefore nullifying the mundaneness of a standard chess match and adding an unpredictability aspect to the game. FPSChess showcases the uniqueness of merging two genres, and how a new sub-genre can emerge from a classic.

Although tabletop games such as Blackjack and chess incorporate multiplayer gameplay to complement its competitive playstyle, this goes outside the scope of this project and will not be developed, but instead noted for future developments (See chapter 8). The AI opponents programmed serve as a functional prototype where they incorporate semi-realistic decision making and pathfinding.

1.4. Previous experimentations

As mentioned in section 1.2, this game was originally created and submitted as part of the Mix and Game Jam competition in 2020. Despite the game being unplayable, the idea of the game showed promising signs that it'd be fun to play. The Blackjack aspects seemed to work well in a first-person point of view and it blended well when incorporating the shooter elements. However, as stated, this game was not fully playable as many core components had yet to be implemented (such as enemy movement). This meant a lot had to be improved upon to create a functional demo. As such, all code and assets were made afresh for this final-year project with vastly different and improved coding practices kept in mind.

Figure 1.1. Previous prototype of AceUP.

1.5. Overview of chapters

This document serves as a starting point for the development of AceUP.

As documented below, we shall cover:

- The software used.
- Levels in the game.
- Implementation of the game (Blackjack, Fights, Artificial Intelligence, et cetera).
- Results and Playtests.
- Conclusion of the games success.
- Future work.

2. Professional Issues

Attached to the appendix (See Appendix E) is the signed documentation for the User Testing Compliance Form. As user testing will be required for play testing a game, ethical considerations are mandatory and have been noted:

Professional Considerations:

1. The game will contain work that has been fully made by myself. Some code may be from following a tutorial, in which case, it will be referenced in the final report. Similarly, art assets may take inspiration from online. This will also be referenced with images attached. (BCS Code of Conduct 2.a, 2.b)
2. No depictions of drugs, alcohol, sex or other inappropriate acts will be made in the game. (BCS Code of Conduct 1.a, 4.a)
3. No strong language will be mentioned in the game. (BCS Code of Conduct 4.a, 4.c)
4. The game will not discriminate against age, race, sexuality, or any other categories that target a group or person. (BCS Code of Conduct 1.c)

Ethical Considerations:

1. **Participants were not exposed to any risks greater than those encountered in their normal working life.**

   Participants will be told the game is rated Pegi 18. No physical harm will be caused from the game but depictions of mild violence will be included in the game due to the nature of first-person shooters. Gambling will also be in the game (although with in-game currency – poker chips) as Blackjack is a game commonly played in Casinos. (BCS Code of Conduct 1.c, 3.a)

2. **The study materials were paper-based, or comprised software running on standard hardware.**

   Standard equipment will be used throughout testing, here we will demonstrate standard equipment as my Dell Inspiron Laptop and Chichester's lab computers. Feedback will be written on paper, or on a Microsoft Word document on my laptop.

   (BCS Code of Conduct 1.d)

3. **All participants explicitly stated that they agreed to take part, and that their data could be used in the project.**

   Participants will be told prior to playing the game that related data will be collected (anonymously). Participants will be told what the project is about to get a better understanding of what data will be collected. Participants will also be made aware that they can remove their data at any time. (BCS Code of Conduct 3.d, 3.e)

4. **No incentives were offered to the participants.**

   Feedback will be from volunteers who will not be bribed to participate. (BCS Code of Conduct 2.g)

5. **No information about the evaluation or materials was intentionally withheld from the participants.**

   All participants will be told that they would be playing a game similar to Blackjack, and taught that they can cheat during the rounds. Participants would also be made aware that there are consequences to cheating. In this way, participants will understand how the game works and what they will be playing. (BCS Code of Conduct 3.e)

6. **No participant was under the age of 18.**

   Participants will all be over the age of 18. They will also be university students or graduates. (BCS Code of Conduct 1.a)

7. **No Participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.**

   All participants will play the game equally, with no distractions. Testing will be done at a suitable time for participants. (BCS Code of Conduct 1.d)

8. **Neither I nor my supervisor are in a position of authority or influence over any of the participants.**

   All participants will be of equal authority to myself. (likely students or graduates). (BCS Code of Conduct 2.g)

9. **All participants were informed that they could withdraw at any time.**

   Participants will be made aware that they could withdraw at any point. This will be stated in the brief as well as before player testing occurs. (BCS Code of Conduct 3.a)

10. **All participants have been informed of my contact details, and the contact details of my supervisor.**

    A document will be handed out which include my contact details (via university email) as well as my supervisor's. (BCS Code of Conduct 1.a, 4.a)

11. **The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session.**

    A document will be handed out which contains details about the game. Should participants have any questions, they can ask me at any point (as long as it is not

spoiling the game). Before testing, participants will also be told what would be about to happen. (BCS Code of Conduct 1.d, 2.e, 3.e)

12. **All the data collected from the participants is stored securely, and in an anonymous form.**

    All data will be recorded either directly onto my personal laptop or on written paper which will be stored securely in a folder. No personal details will be collected (i.e. name, sexuality, address). (BCS Code of Conduct 1.a)

    Data will only be held for as long as necessary and in line with the Data Protection Act (2018).

3. Game Development – Background Research

3.1. Overview

The main goal of this project is to achieve a functional demo of a game with the intent of combining two popular video game genres for new and unique gameplay to emerge.

I have chosen to create a table-top game (Blackjack) and a first-person shooter as these are two established themes for games which can blend seamlessly together. Despite effectively creating two different features of games, an attempt will be made to fully complete this project. After the development of this game, users will playtest the game, where feedback can be given and opinions stated on whether the combination of two genres enhanced the traditional Blackjack game.

3.2. Game Description

AceUP is a first-person shooter Blackjack game, with the objective of betting poker chips to earn as much as possible. Players can cheat to attempt a better hand at the risk of getting caught and starting a free-for-all fight with others. This incorporates a strategy factor into the typical luck of a standard Blackjack game, with players balancing the pressure of cheating to get a better hand or to stay and risk losing it all.

This section will introduce a normal game of Blackjack and compare it to how our game will differ. We will also look into a similar game and analyse how a slight change in rules can create vastly different gameplay.

3.2.1. The Standard Blackjack Game

Blackjack is a game built on statistics, the probability of a bust or a perfect twenty-one depends on what cards are revealed in your hand. In a casino setting, where Blackjack is often played, it's usually difficult to predict a win unless the person has been counting cards. As the goal of an ordinary Blackjack game is to beat the dealer, the main strategy is to stand once you've reached seventeen or higher, this is because the odds of busting dramatically increases at this point. This differs from the strategy in AceUP as you can avoid a bust in some instances.

According to Official Game Rules [5], The main structure of Blackjack goes as follows:

1. Players place a bet in a betting box in front of them.
2. Once every player has bet, the dealer gives every player two cards face up. The dealer has one card face down and one face up.
3. During the players turn, they can choose to 'stand' (keep their current hand) or 'hit' (request an additional card from the deck). If the player hand goes over twenty-one, they bust and immediately lose.
4. Once all players have had their turn, the dealer reveals their face down card, and plays similarly to how a player would, requesting a card or standing. Unlike players, the dealer must play following the rules of the house, such as requesting cards if under seventeen.
5. If the dealer busts, all players automatically win. Otherwise only players with a higher hand than the dealer would win the bet. In the chance of a tie, the players original bet would be returned.

In Blackjack, Players play against only the dealer in a typical game. This means that they do not play against anyone else currently sat at the table.

Figure 3.1. An example layout of Blackjack

3.2.2. Modifications

Although AceUP is similar to an ordinary game of Blackjack, there are multiple distinctions due to the new mechanics added. The main noticeable difference is that once we allow the players to cheat, the uncertainty of getting a good hand becomes redundant. Players have a general idea and strategy of what hand they could go for every round. Players are also opposing every other AI in the game, instead of it being solely against the dealer. This is a design choice intended to make the player focus on multiple targets rather than one opponent when cheating and fighting. When the player does want to cheat, they have to weigh the pros and cons of switching their cards or when calling other opponents out as it's unlikely or sometimes impossible to win a fight if they don't have enough cards. Due to the random combination of cards each round, each Blackjack game can vary, with the inclusion of a fight mechanic, each game feels distinct from one another which provides a different and fun challenge every time the game is played.

3.2.3. Dungeons and Degenerate Gamblers

Dungeons and Degenerate Gamblers (D&DG) is a roguelike deck builder game in development by Purple Moss Collectors. Although a Blackjack game similar to AceUP, D&DG adds a twist to the game by incorporating unique cards from both real life and other card games. This game differs from Blackjack as you build your own unique deck per session which each provide different abilities.

Figure 3.2 Dungeons and Degenerate Gambler (from Itch.IO)

Instead of a standard game where cards have an numeric value which add to twenty-one, D&DGs provides cards that can reduce your hand value, steal cards from opponents,

duplicate cards, and more. The spin on this game incentivises players to think outside the box, and procure a strategy in every game. However, like Blackjack and the concept of AceUP, the main goal is to beat the opponent by getting as close as possible to twenty-one whether that is by using unique cards to twist the odds in your favour or cheating to swap cards out.

3.3. Game Development – Software Used

By the end of this project, we aim to have a functional demo of a game, therefore it should be created in a game engine which provides a fluid development process. I have opted for Unity [6], despite alternatives such as Unreal and Godot [7, 8] being available. This is due to Unity's extensive documentation and tutorials alongside prior experience of the engine. Additionally, other software have been used such as Blender, Paint.NET and Photoshop [9, 10, 11]. This section will elaborate on these software decisions alongside an explanation on how these software aided in the design process.

3.3.1. Unity

Unity is a popular game engine developed by Unity technologies on June 2005. Since then, it has grown to become a staple game engine for many game developers benefitting from a long and helpful community [12]. As such, the creation of AceUP will be made primarily in this software. Unity's ease of development alongside Microsoft's Visual Studio [13] code suggestion feature for C# lends itself to a massive advantage for game programming, hence this became the best choice.

Unity provides many packages such as post-processing, skyboxes, and particle systems. These were used during development to improve the game scene. Although these took a significant amount of time to implement, this report focuses on the technical programming aspects of this game (See Appendix B).

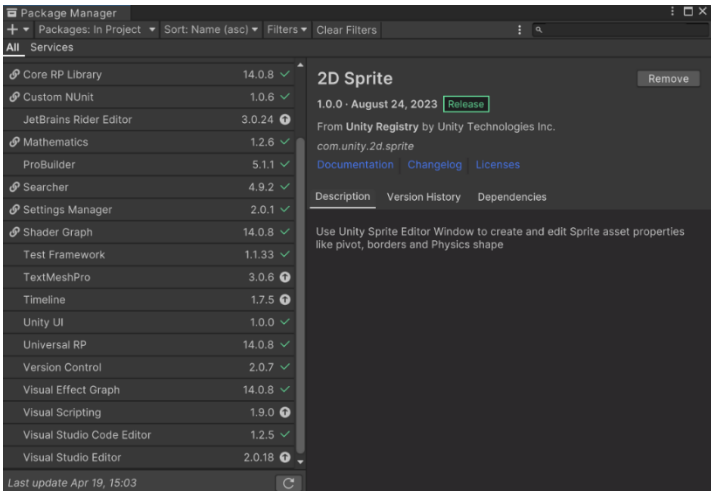


Figure 3.3. Unity's Package Manager

3.3.2. Blender

Blender is a free and open source software (FOSS) that allows the user to create 3D assets ready to be imported into game engines such as Unity. It is capable of creating high-quality models as well as low-poly models (3D models that have a lower polygon count) fit for games. Low-poly models were ideal for this project as they are quick to model for prototyping whilst minimising the GPU bottleneck which optimises the overall performance of the game [14]. The creation of all assets such as buildings and environmental props were all designed with the intention of keeping polygon counts at a minimum (as shown by Figure 3.4).

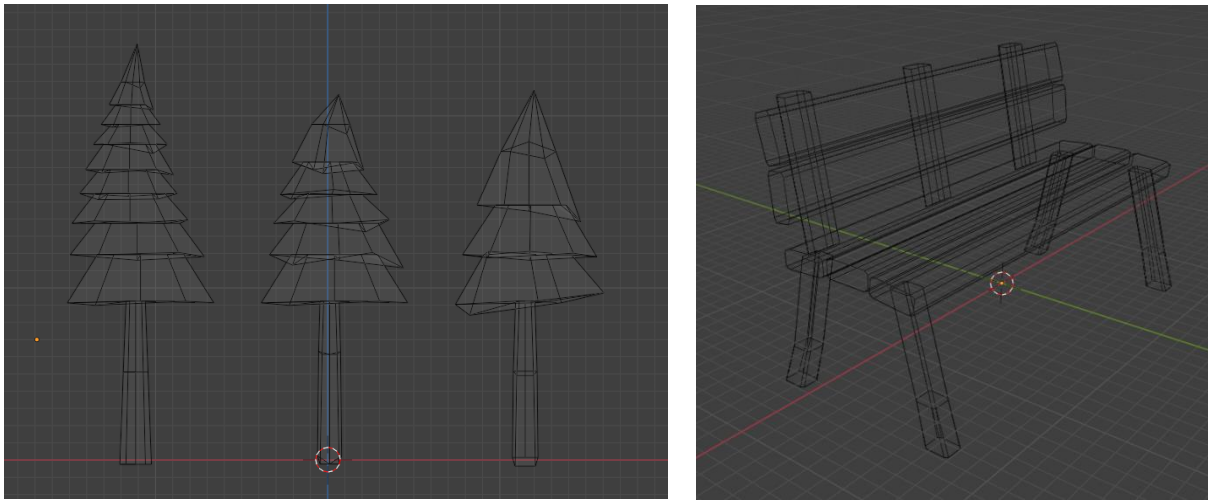


Figure 3.4. 3D assets (Trees and a Bench) made in Blender, which could be imported into Unity

3.3.2.1. Sprytile

As Blender is open source, this allows for addons such as Sprytile to be created. Sprytile is a tile mapping addon for Blender, focused on giving Blender tools similar to 2D tile mapping workflows, but for 3D [15]. The addon allows planes (flat surfaces) to be painted onto the 3D viewport. These connect together to create larger models and were primarily used to create houses in this project.

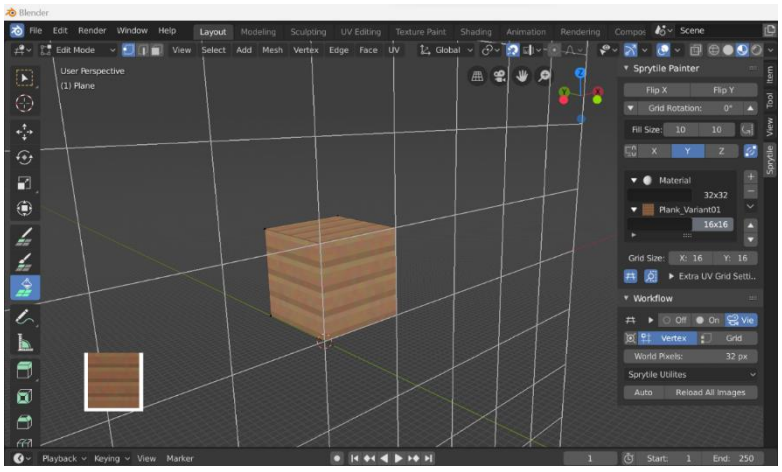


Figure 3.5. Sprytile addon for Blender. Textures can easily be painted onto the scene.

Figure 3.6 shown below are all textures used for the buildings in the game. Sprytile limits models to simple squares therefore heavy inspiration was taken from Minecraft textures and builds. These textures were imported into Unity where materials could then be created and applied to the houses which allowed for each house texture to appear different despite using the same models.

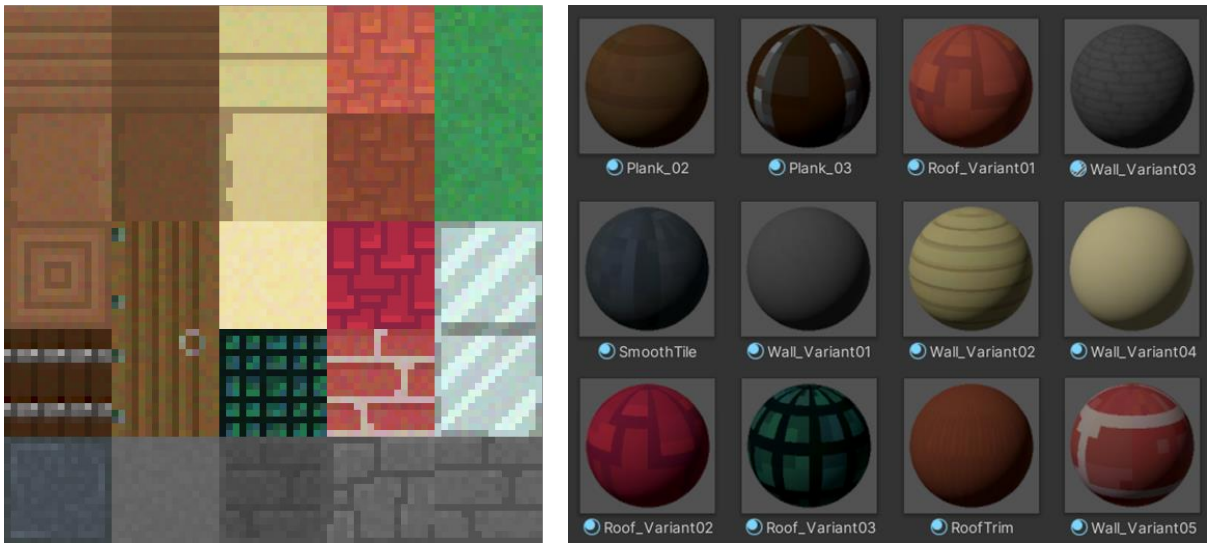


Figure 3.6. Texturesheet made in Paint.Net for Sprytile painting. These textures were then imported as materials for Unity.

3.3.2.2. Minecraft Inspiration

The Minecraft map was downloaded from a YouTube video [16] which was imported into Blender by converting the world into a FBX file using Mineways [17] which allowed houses in the scene to be separated into individual models and rearranged into an ideal layout.

When recreating the models, all unique houses were printed out on paper so that they could be annotated and measured. This helped with the 3D modelling as the dimensions of the houses were all labelled.

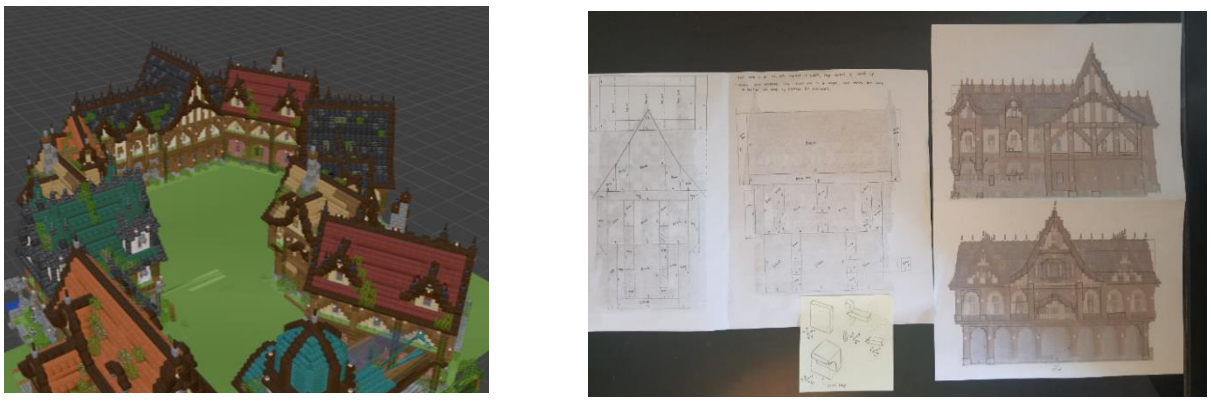


Figure 3.7. Prototype level layout imported into Blender from Mineways, these houses were independently screenshotted and printed out for measuring and labelling.

3.3.3. Paint.NET and Photoshop

As shown by Figure 3.8, textures were created in a pixel art style. This was a stylistic choice lending itself to the low-poly vision of the game. These designs were made in Paint.NET taking inspiration from Minecraft. Textures were made in multiple layers, first starting with a base layer, then a detail layer, and finally a noise layer. The noise layer added detail to the texture so it wouldn't appear too flat on the houses. This added depth to the tileable textures.

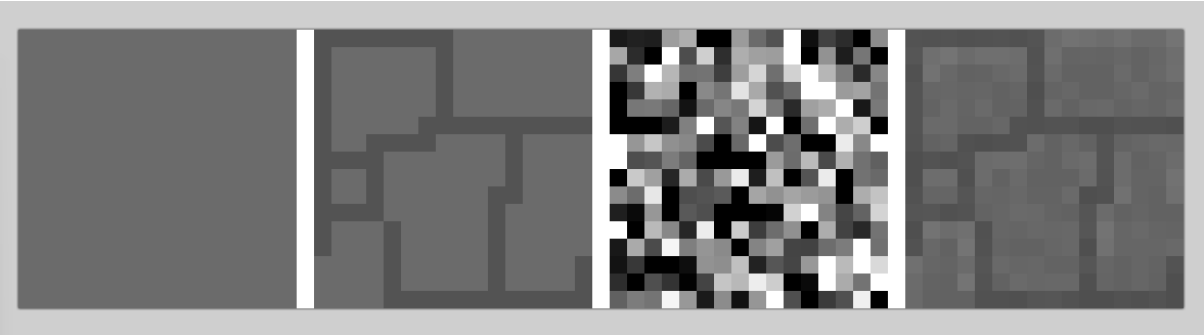


Figure 3.8. Tileable stone texture creation process in Paint.Net (Base Layer, Detail, Noise, Final Image).

Although Paint.NET provided an easier workflow due to its simple layout, Photoshop was used for some asset creation due to its superior capabilities as an image editing tool. Most user interfaces such as the currency (see figure 3.9) were created in Photoshop as sprites could easily be rearranged. The tutorial (See Appendix C) was also made in Photoshop as it consisted of multiple layers and image masks which Paint.NET could not provide.

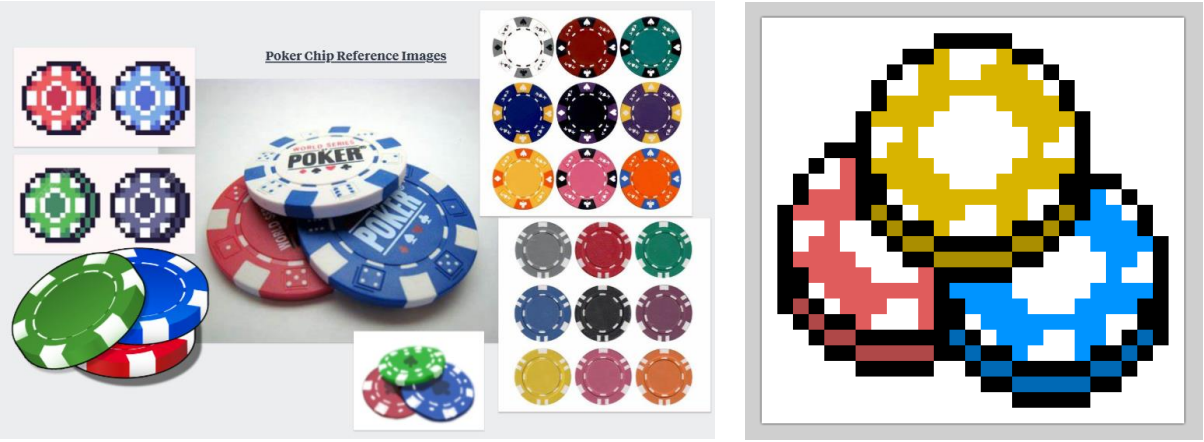


Figure 3.9. An inspiration board for poker chips (used in the game as currency), which gave inspiration to our final design.

4. Requirement Analysis

Before the game was developed, a set of requirements were defined in the interim report (See Appendix F). These could be split into functional and non-functional requirements and would serve as a basis during the whole development cycle. Some requirements and features have been omitted due to the word limitations of this report. This section focuses on these main requirements which were followed to allow for this game to be made.

4.1. Functional requirements

Functional requirements listed as follow:

1. Players will be able to play a game of Blackjack as normally intended. This includes the choice to hit (where the player can gain a card), and to stand (where the player ends their turn).
2. Players will be able to cheat and swap their hand out with cards in their inventory.
3. The player and the AI will be able to start a fight if they catch one another cheating. This fight would include:
    - Shooting: where the player can fire ammunition at other opponents.
    - Taking damage: where getting hit by a projectile causes the player to lose.
    - Winning/ losing fights: where the winning player earns a prize, whereas the loser receives nothing.
4. Cards gained must be random from a set deck of cards (therefore no duplicate cards).
5. The AI should know when to hit, stand, and cheat when playing Blackjack, and to move and shoot when fighting. Therefore the AI must simulate awareness of their environment.

4.2. Non-functional requirements

Non-functional requirements listed as follow:

1. Players should be able to play in an arena where they are limited to a portion of the map.
2. The player should have an indication of how many chips they have remaining to bet.
3. The player should know when other players are cheating and when they are cheating.

5. Game Design – Level Scenes

5.1. Overview

In the following chapter, I will discuss the two main levels used for the game. Due to the large quantity of 3D models alongside numerous user interfaces and script elements, multiple Unity scenes were used to minimise the number of components at any given instance which allowed memory allocation to be reduced at any given point during playtime. These included the main menu, card shop, blackjack game scene, and the fight scene which were all chosen to be independent levels as they all handled different aspects of the game. This however came with the challenge of utilising multiple Unity scenes and linking them together. We will cover how we resolved this issue in section 5.4.

Although one scene could have been used for the entire game, multiple levels allowed for a better workflow when managing the games hierarchy and helped split the game into multiple parts. The choice to use multiple game scenes was made early on during development and helped massively when making the game.

5.2. Blackjack Scene – Tabletop Genre

The Blackjack scene covers all aspects of the tabletop genre for the first portion of the game. This level allows the player to place a bet with other opponents, and then play a normal hand of Blackjack where they can hit or stand. Optionally, when hovering over the cards held in the players hand, the player can choose to cheat with a left mouse click. This highlights the card in red and displays up to three cards which can be swapped around. When the round starts, each opponent is capable of cheating if they believe they have a bad hand and this can be called out by the player, similarly, the AI can also call the player out for cheating which is done by a coroutine which runs recursively to check if the player is cheating. This scene is where the player will spend most of their time playing the game as they will be betting and playing multiple rounds of Blackjack if they do not cheat, however when required, the level links to other Unity scenes such as the card shop and the fight scene.

5.3. Fight Scene – First-Person Shooter Genre

The fight scene is where the first-person shooter elements come into the game. This level is only loaded up when the player is caught cheating or catches another opponent cheating which means that it is directly linked to the main Blackjack scene. Although the two levels look alike, the fight scene has a few additional components, such as a level box collider and NavMesh floor, which serve as barriers and AI traversable areas for the level. When the player loads into this level, they must fight off against the three opponents that they played Blackjack with. This is done by throwing cards using the left mouse click. If the card hits an opponent, the opponent hit will be slain in one shot (removed from the scene). However, the player can also be shot at by the other opponents, which means that they must be careful. This game mode is a free-for-all fight where opponents and players will attack one another and the winner is dictated by the last remaining player standing. After this, the player is met with an option to continue playing, or to quit to the menu. Ideally, the player would continue playing which is the most likely choice however, the player can choose to stop playing at any time.

5.4. Game Scenes – Issues and Fixes

Changing one aspect of the game, such as the location of the main table or an object as simple as a tree, came with the complication of having to adjust every scene that had the same layout. Thankfully, only the Blackjack scene and the Fight scene had to be changed in most cases as the other scenes would not be affected which meant that it was not too big of an issue. Had there been multiple scenes with the same layout, a common fix would be to have multiple scenes open and deactivate unused scenes whilst keeping the common gameobjects activated. This would have replaced the need for loading scenes but would have increased the file size. Similarly, multiple scenes increase the file size during build time but these files are split between every scene which means that the game will play the same regardless.

A much larger issue that had to be tackled was a way to handle state persistence between scenes. As each game played is random (i.e. cards you receive/ buy) this meant that we couldn't predict what cards the player would obtain as each combination was unique. Therefore the game needed a method to save information such as the players total currency, cards they have stored, and the total bet amount. This was done through a script called 'playerInfo' that does not get deleted when loading between scenes. Unity provides a method called 'DontDestroyOnLoad' which preserves an object during scene loading [18]. As the object is not removed, this also means that any child components it contains would also persist. This allowed for the playerInfo script to hold all necessary

information that the game required where each variable was made public which allowed for every other script to be able to access this information when requested.

Figure 5.1. Class diagram of the playerInfo script and the PlayerInfo GameObject in the hierarchy.

6. Game Development – Implementation

6.1. Overview

As can be seen from Figure 6.1, most scripts are referenced by the ‘cardManager’ or the ‘fightManager’. These scripts manage the two aspects of the game and directly control what happens in each aforementioned game scenes. A total of fifteen scripts were used whilst developing AceUP, this chapter documents how each script was implemented in order to achieve our finished demo of the game.

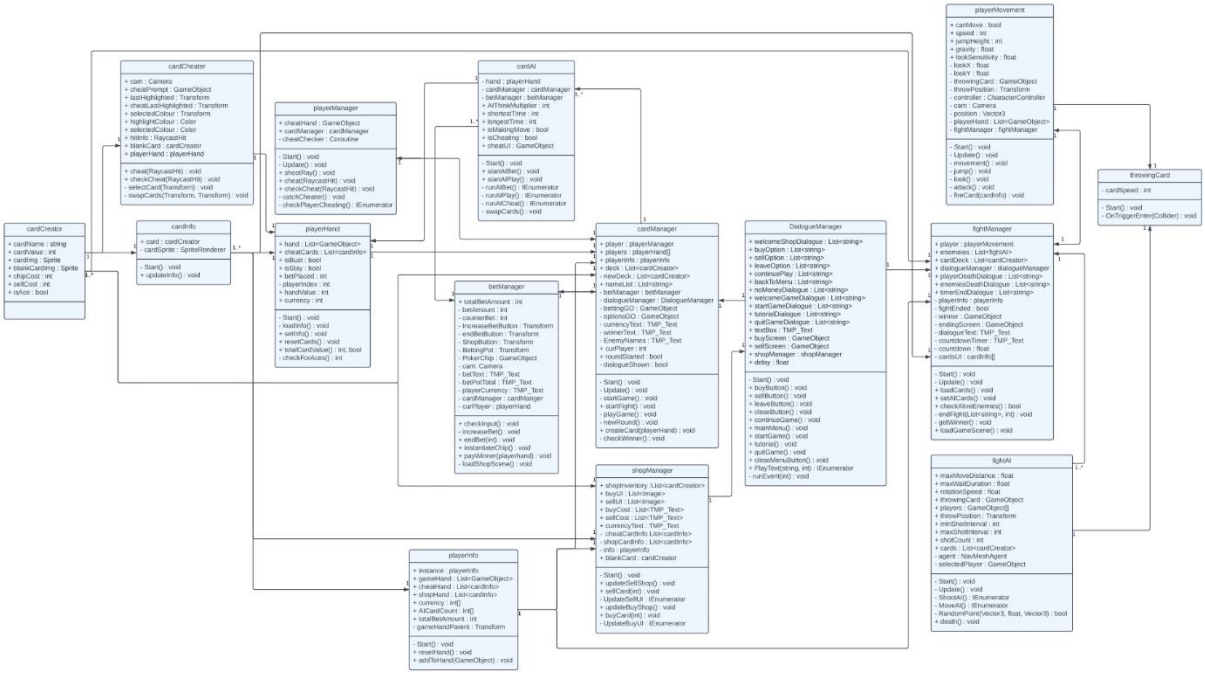


Figure 6.1. A high level class diagram showing all links between scripts.

6.2. Blackjack

6.2.1. Card Manager

In the Blackjack portion of the game, the round can be split into two parts: the betting phase and the card phase. This is controlled by one main script called the ‘cardManager’ which handles all logic for the game to run. When the Blackjack scene first loads, the cardManager will prompt the player to place a bet before incrementing to the next player in the game order. This is done before every round of the game to ensure that every player puts forth a certain amount of money before the cards get distributed and the round begins.

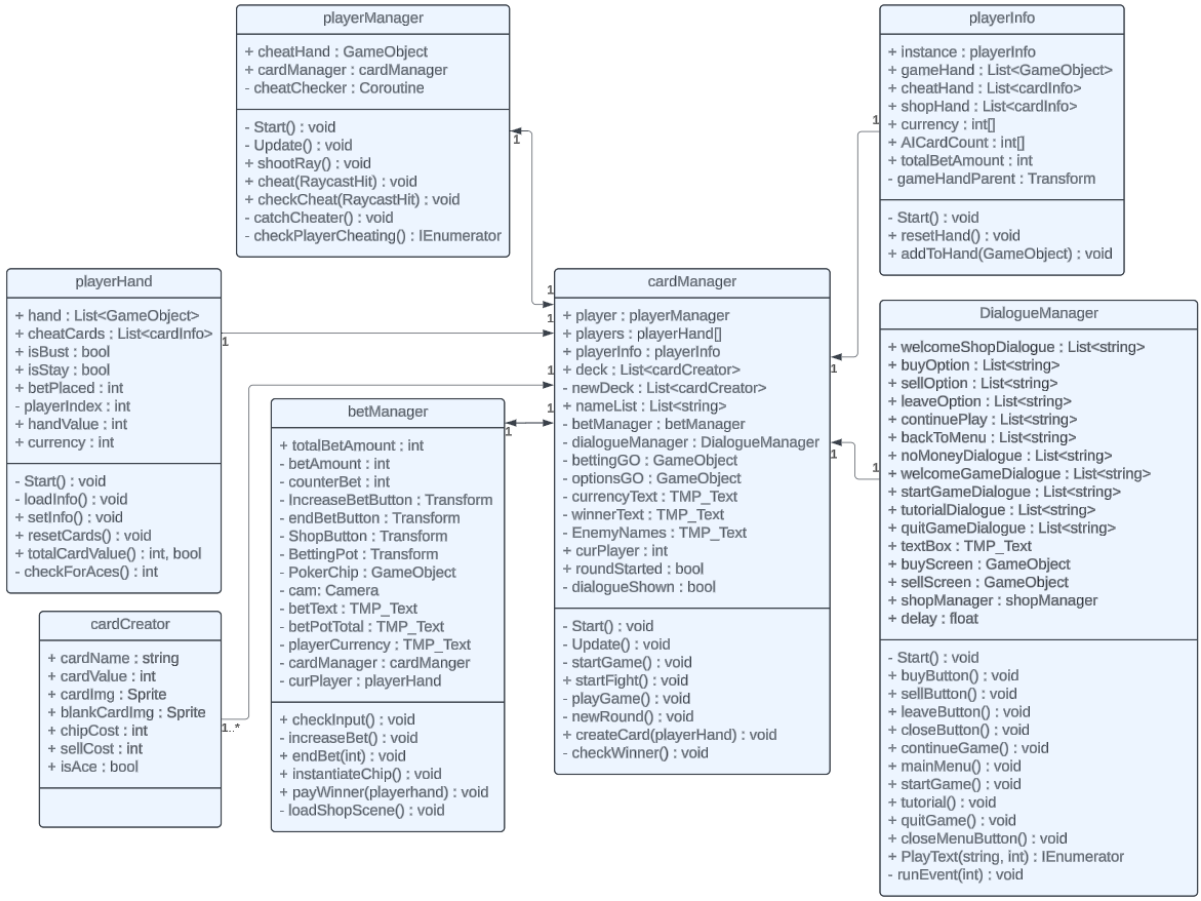


Figure 6.2. A class diagram of the cardManager and all scripts it directly interacts with.

After all players place a bet, two cards get distributed for every player from the standard card deck of fifty-two clearing any cards the player may have from previous rounds. This is handled by the newRound() function which starts the Blackjack phase of the game.

When all players have ended their turn, the round will finish off by detecting the winner by calling the totalCardValue() function (found in the 'playerHand' script) and comparing them with one another. In order to optimise the game, this comparison is only done if neither of the two players are over twenty-one (aka a confirmed loss). Any Aces are also accounted for as they may have a value of one or eleven. This is done by calling the checkForAces() function. Upon confirming the winner, the game will automatically pay an amount based on the total bet amount from all players, plus a bonus of the winners bet acquired from the 'betManager' script.

In order to balance the game in favour of the player, the opponents will always have a currency that falls within 35% to 75% of the players maximum currency. At the end of every round, the cardManager script ensures this condition so that the AI will always be able to bet somewhat accordingly to the value the player would bet.

6.2.2. Betting

The betManager script component is attached to the bet device and is directly controlled by the cardManager. This component will detect if the player clicks on any of the devices buttons when it is their turn by using Unity's Raycast physics to project a line from the camera position. When the Raycast detect a valid object collider, an appropriate function will run depending on which button was pressed. This allows the player to add to their current bet, confirm their total bet, and to go to the shop to purchase or sell cards they've acquired.

Figure 6.3. Betting Device 3D model, the green outlines indicate where the clickable box colliders are.

The betManager also stores an integer variable for the total bet placed during the round. Although each player has information on how much they bet themselves, the betManager stores the total value of all players bets. This is done to easily manage payments to the winning player which is called by the cardManagers checkWinner() function.

As the betManager is programmed from a seperate script file than the cardManager, opponents are able to directly access this file easily. This means that the same script can be used by the player and opponents when placing bets. Although they cannot access the shop, the opponents are able to end their bet which allows for the game to know when they have ended their turn.

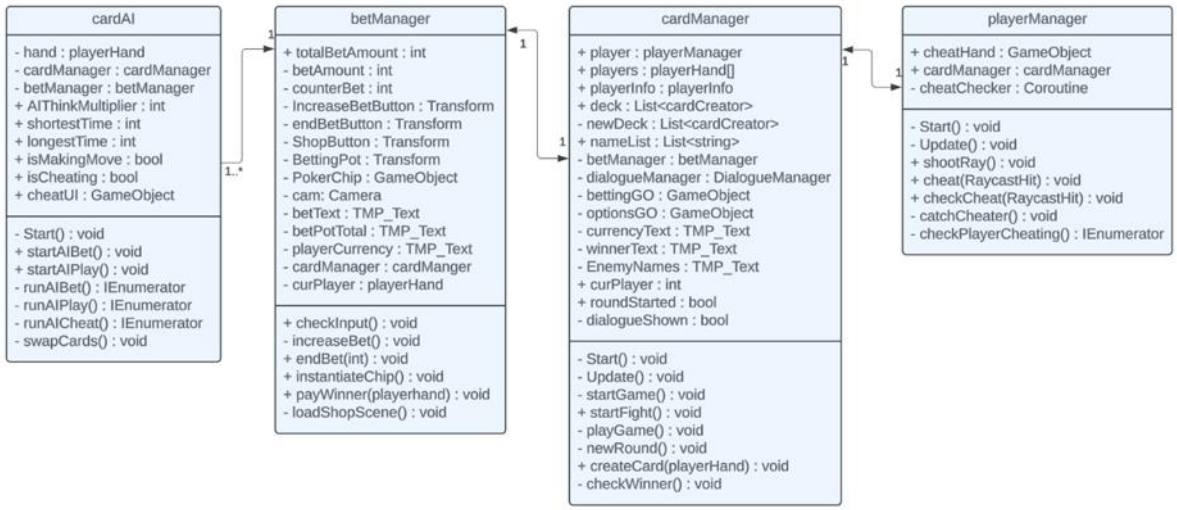


Figure 6.4. The betManagers functions are handled by the cardManager which the playerManager and cardAI have access to.

6.2.3. Blackjack Game

As mentioned previously, a standard deck of cards consist of 52 cards in four suites: Spades, Clubs, Diamonds, and Hearts. This meant that multiple card assets with independent values were required. This was done through scriptable objects as it allows for modularity and encourages data driven design [19].

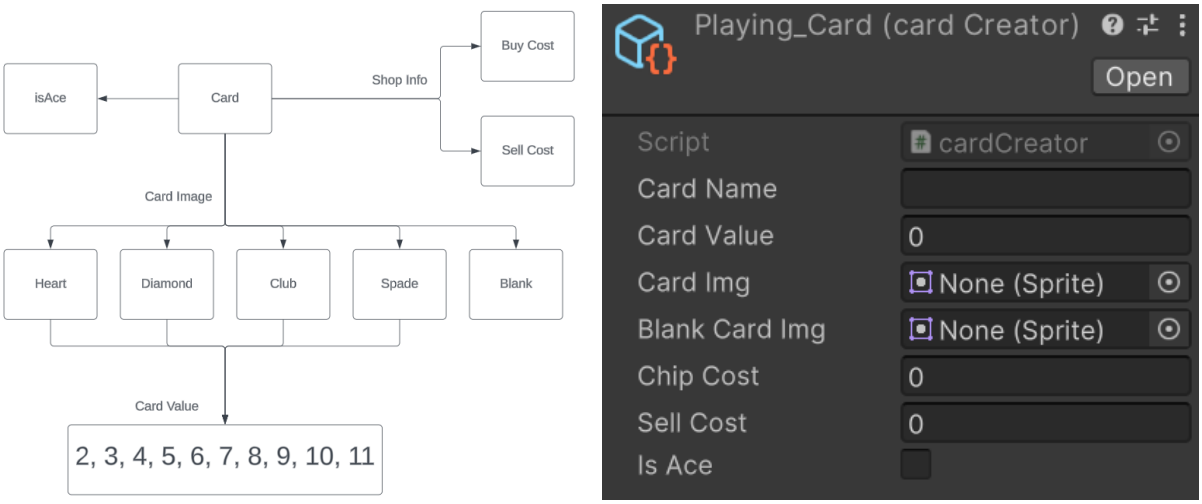


Figure 6.5. A structure diagram of the card scriptable object and its corresponding variables in the Inspector.

Every card has a unique name based on its card value and suite. Additionally a cost for buying and selling the card was included in the scriptable object so that it could be assigned to in the card shop alongside other variables which determined what kind of card it was. These scriptable objects would be stored in a 'cardInfo' script which is attached to a prefab that is instantiated by the cardManager. Rather than assigning the scriptable object directly to the card, a separate script was needed due to the cheating mechanics forcing the scriptable object to have to be changed. cardInfo ensures that all

cards are rendered with their corresponding texture in the game. This is accomplished in a public function called updateInfo() which gets referenced multiple times by different scripts.

During the round, the player and all opponents have a ‘playerHand’ component which holds all the cards in the players hand. The cardManager has access to every playerHand and uses this script to access the totalCardValue() and resetCards() functions. The playerHand is also used to load and set information to the playerInfo script so that the players hand and currency is saved between scenes.

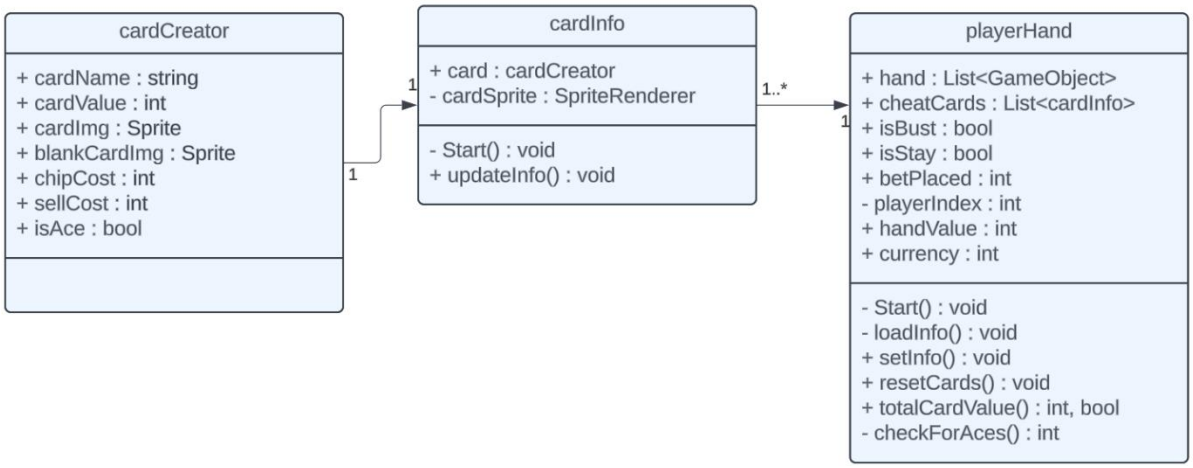


Figure 6.6. Class Diagram for cards in the player and opponents hand.

6.2.4. Cheating

At any given point during the round, the player can cheat and swap their cards. Similarly, the opponents will also cheat during the round when they have had their turn. Although both the player and the opponents can cheat in the same way, this is handled by two separate scripts: the ‘playerManager’ and the ‘cardAI’. This section will focus on how the player cheats as section 6.5 will go more in-depth for the AI.

Figure 6.7. cardCheater has access to the players hand and is inherited by the playerManager. The playerManager runs certain functions when it is the players turn.

During the game, every frame will check to see if the player is hovering over a card by the shootRay() and checkCheat() function. If this condition is met (highlighted in orange) and the player left clicks, the cheat() function will run, causing the card to become selected and a set of 3 cards to appear (which we will call the 'cheat hand') which means that the player has begun cheating. There are two options for the player when they have started to cheat. The player can either choose to stop cheating (by reselecting the same card), or they can continue cheating and swap their currently selected card with any other card in their cheat hand.

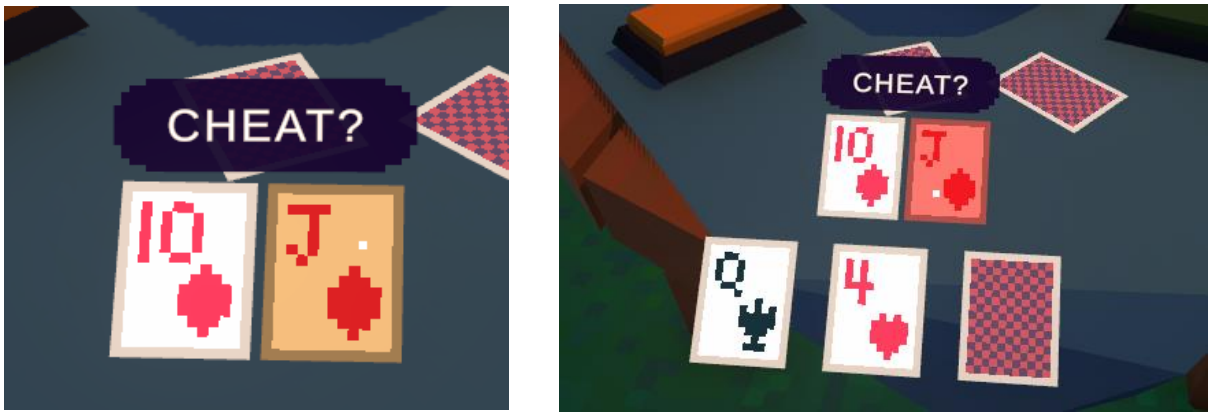


Figure 6.8. Hovering over a card highlights itself in orange. When clicked on, it will become red, and three sets of cards will appear. The player can only swap with non-blank cards.

6.3. Fighting

The fight scene fulfils the second genre of the game and is handled by the fightManager. This scene is loaded directly by the cardManager when an opponent is detected cheating by the player, or when the player themselves are caught cheating. Before the scene is loaded, all information is saved onto the playerInfo script to keep all important data. This is important as all players currency must be saved alongside how many cards they were holding before the fight started.

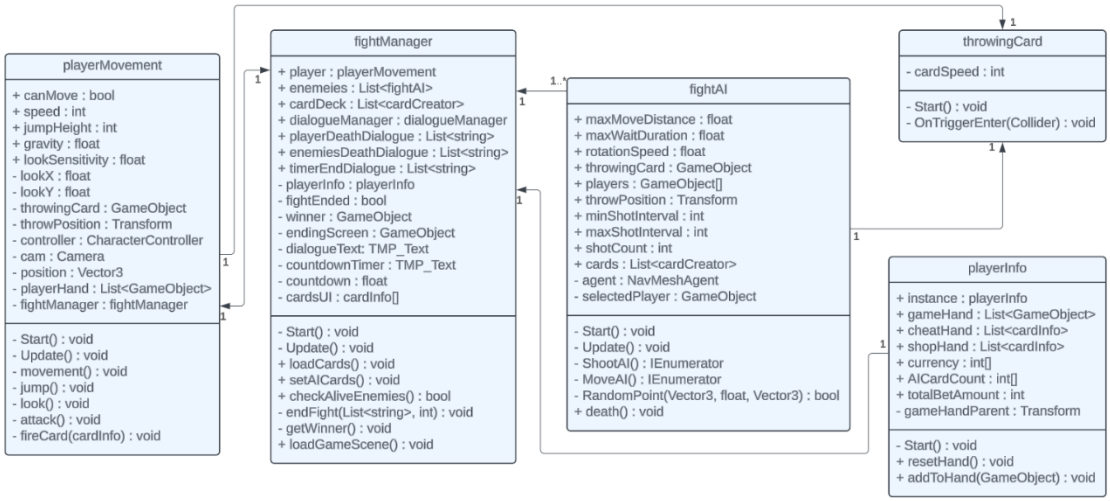


Figure 6.9. A class diagram of the fightManager and all scripts it directly interacts with.

6.3.1. Player Movement

Although the player can look around their movement is locked in the Blackjack game scene. This movement is unlocked when in the fight scene, allowing the player free roam of the entire level confines by using the WASD keys. The player is also allowed to jump using the spacebar key. Movement is allowed for the player in the fight scene as they have to avoid oncoming projectiles and move to shoot at opponents. On the contrary, movement is restricted in the slower-paced Blackjack level as we want to limit the players field of view and prevent them from standing next to or too close to enemies. The players movement is controlled by the 'playerMovement' script. This script also controls looking around and attacking by calling the movement(), jump(), and look() functions every frame.

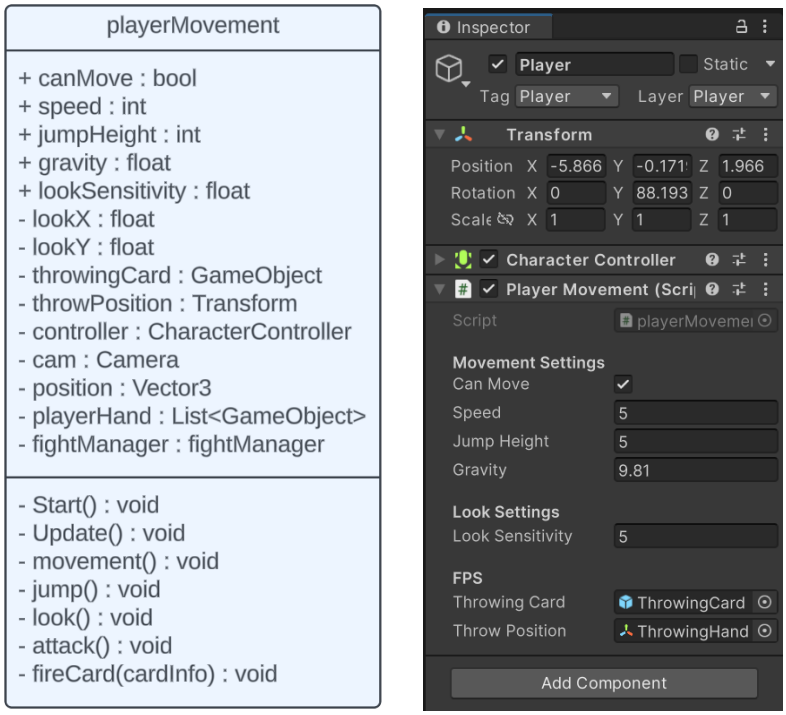


Figure 6.10. The playerMovement class diagram, and the Player GameObject with the script attached. playerMovement uses Unity's Character Controller to move the player.

6.3.2. Card Ammunition

The players ammunition is set when the fight scene first loads in. This data is retrieved from the playerInfo script and is used in playerMovement as this script also handles attacking. Ammunition is set based on the players total cards and what cards they had whilst they were playing Blackjack. This creates a new strategy of hoarding multiple cards in order to receive more cards to shoot, at the cost of the player likely going over twenty-one. The design choice of using cards as ammunition allows for the player to choose to play riskier as they have a higher chance of winning a fight with more cards to shoot.

Controlled by the attack() and fireCards() function, cards are shot by using the left mouse button. This removes the card from the players total card count and a card projectile game object is created in the scene with a forward velocity destroying any opponents it comes across. Cards may also bounce off environmental colliders such as the buildings or trees which adds to the games difficulty as the player must be careful not to hit themselves.

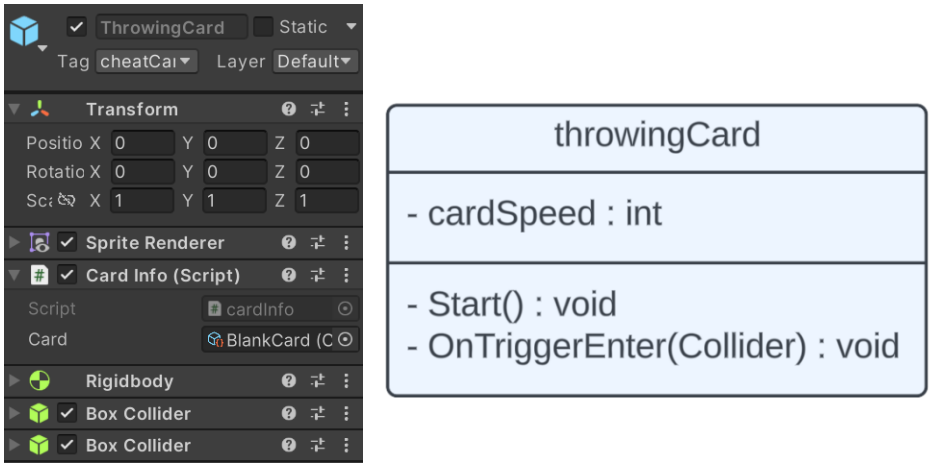


Figure 6.11. throwingCard script attached to the ThrowingCard prefab and its corresponding class diagram.

Although it would have been ideal for the different values of Blackjack cards to deal different amounts of damage to opponents, this falls outside our requirements established and therefore outside the scope of this game demo. Future developments of this project will likely involve different damage values for cards and a healthbar for players. This would add to the Blackjack elements of the game as not only does your hand need to equal twenty-one, it must also deal enough damage to win a fight leading to higher value cards becoming more beneficial.

6.3.3. Winning Fights

While a normal payment for Blackjack is paid towards the player whose hand is closest to twenty-one without busting, payments for winning a fight goes towards the last remaining player. This means that in order to receive payment of the total bets made, the player must first eliminate all three other opponents first. Similarly, the opponents must eliminate the player before one of them can receive the winning prize.

In order to incentivize the player to start a fight and risk cheating, the player is offered the total bets made multiplied by 1.5 times. This would likely be more than the amount they'd win if they had played a default game of Blackjack, especially if the player hadn't bet much originally. This means that fighting is a good means to earn currency in the game, however winning the fight may be difficult if the player has a lack of cards in the first place.

In some instances, it may be impossible to win a fight at all. This may occur if the player and opponents all miss their limited card shots. This problem is resolved by adding a thirty second countdown to the game which prevents the game from being stuck on the fight level due to multiple players surviving. Thirty seconds was chosen as an ideal sweet spot as any less would means that the player would have to frantically shoot (meaning a very likely chance of a loss), and any more would lead to opponents using up all their shots and allowing the player to move near opponents and guarantee a victory by never missing. Although a couple seconds difference may not seem to affect the fights too drastically, a time of fifteen or even forty-five seconds was found to negatively affect enjoyment in this portion of the game.

Figure 6.12. a thirty second timer that counts down on the top right of the players user interface.

6.4. Card Shop

The card shop was originally a desirable requirement whilst writing the interim report. Despite this, it became crucial for allowing the player to start cheating in the first place. This is because it'd be unrealistic for the game to allow the player to remove a card and store it in their cheat hand. If the game were to be played in real life, it'd be obvious that a card was removed or added when the player has yet to request a card. Therefore to maintain immersion, the player can only cheat when they finally buy cards to cheat with. This locks the player from cheating during the first few rounds and forces the player to play Blackjack as intended until they earn enough chips to spend. Although the player starts the game off with one hundred chips, this would only guarantee one high value card, or multiple lower value ones which means that it'd be unlikely to win straight away unless the player puts some thought into their gameplay strategy.

The card shop is only accessible before cards are distributed to players in the game. This is to prevent mid-game purchases where the player knows what cards they need to buy to guarantee a win.

The shopManager script controls actions made in the shop scene and led to the creation of the DialogueManager (See Appendix D).

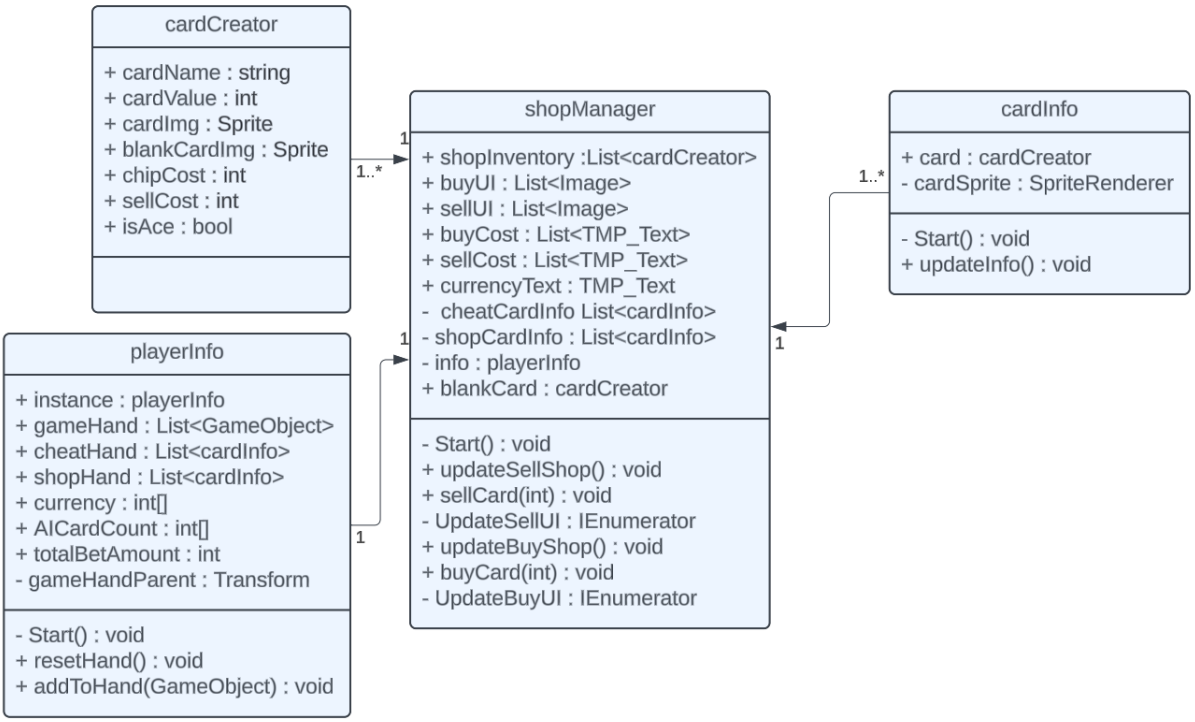


Figure 6.13. a class diagram of the shopManager and all scripts it directly interacts with.

6.4.1. Buying Cards

When buying cards, the worth of each card is based on ten times their card value. For example, this means that a two of diamonds would cost twenty or a five of spades costs fifty. As Aces are special cards with multiple values (a one or eleven), the cost of Aces should be higher hence why they cost one hundred and fifty chips.

In the shop, the cards the player can buy are limited to three at a time. This means that the player must sell some cards if they want to buy a new card but are already holding the maximum amount in their cheat hand inventory. This also means only three cards will be shown to the player at a time which persist until the player eventually buys the card even if the scene was unloaded. Displaying only three cards forces the player to have to reroll cards in order to get a desired hand, therefore making the player use their currency earned which sways the player to bet more money to earn more in the game. This feeds into our core gameplay loop of playing Blackjack to earn and then spend.

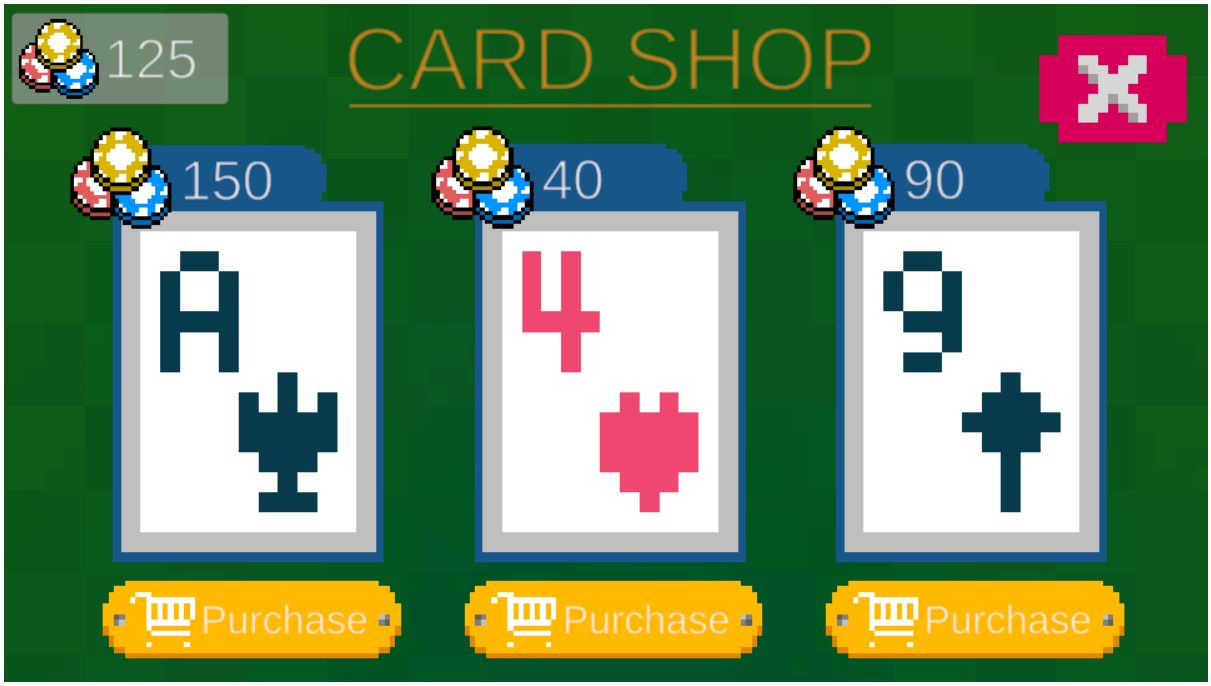


Figure 6.14. The shop user interface when purchasing cards.

Although the cards are naturally set at a constant value, card cost does become redundant quickly as the value of chips deflate over time due to money constantly being earned in the game. This was found during player testing with testers who played riskier becoming too wealthy easily meaning that they could then buy any card they wanted. When developing the game further, it may be ideal to add a multiplier to the cost of cards so that their value goes up as the player has more money in order to keep the rate of cards to be relatively the same price.

6.4.2. Selling Cards

Similarly to buying cards, when selling, the player can only sell three cards at a time. This is because they can only hold three in their inventory so any more cards on the user interface would become redundant. When selling cards, the value they sell for is worth half the amount it cost to buy. This forces the player to spend their chips wisely if they are on a limited budget.

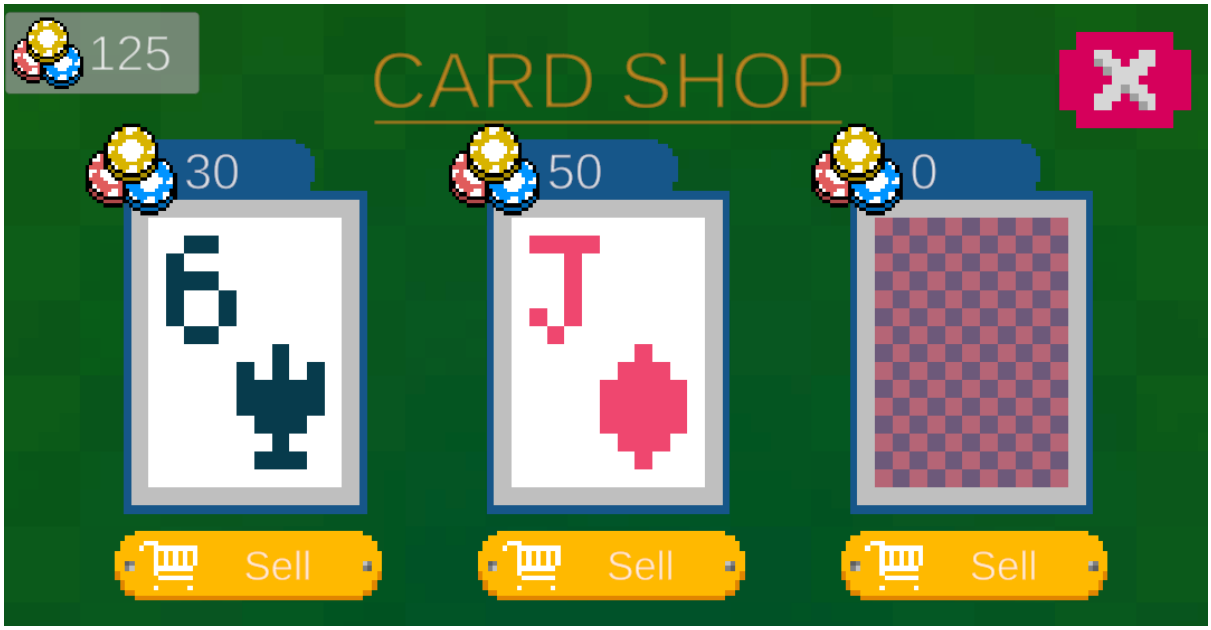


Figure 6.15. The shop user interface when selling cards.

The game could as likely pay the player no chips when selling, however, adding a sell cost to each card was an improved design choice as it causes desperate player to have to cheat and bring back higher value cards to the shop at the risk of losing the Blackjack round. For example, if the player buys a low value card (two of diamonds bought for 20 chips) and then swaps it out for a high value card (such as a ten of hearts sold for 50 chips), the player would earn a slight profit (in this case 30 chips). While this would not lead to a significant change in the players currency, it may prevent the player from immediately losing the game early on.

6.5. Opponents – Artificial Intelligence

The ‘cardAI’ script and the ‘fightAI’ script is in charge of every opponent game objects logic. These two scripts are used in both the Blackjack portion of the game and the fight portion and are one of the many fundamental components used for the game to operate. The following sections will explain how each script was created alongside how they are used in the game.

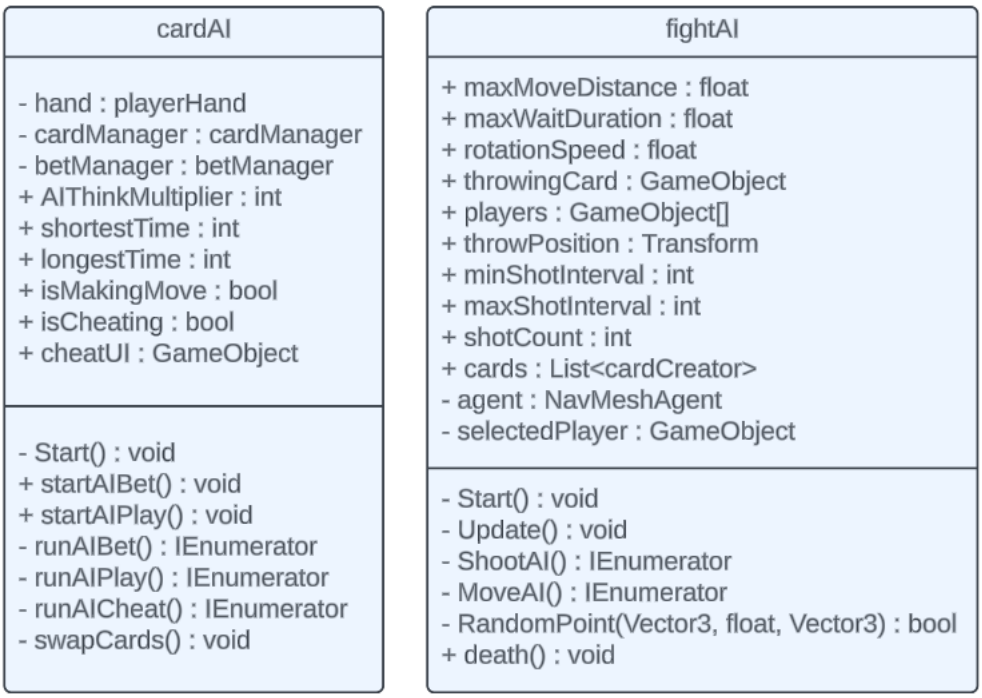


Figure 6.16. The cardAI and fightAI scripts.

6.5.1. cardAI Script

The cardAI script only runs when it is that opponents turn. This logic is controlled by the cardManager calling the functions inside of the script. Before the round starts, the AI must place a bet, this function is only called once where the current opponent will proceed to bet between 10% to 70% of their entire currency. As stated in section 6.2, this ensures the opponent bets inside of a random range of the players currency. As the bets for the AIs are set in a certain percentage range, this means that the more money the player has, the higher the stakes from betting.

After all opponents bet and the player has chosen to stand (pass their turn), the next AI will start playing Blackjack. Each AI will calculate their total hand value and either hit or stand. This action mimics the player although the AI can be predicted somewhat due to the fixed conditions. When playing the round, if the opponents hand is currently under seventeen, they will always request an additional card until they go over this threshold. However, if the opponents hand has busted, they will attempt to cheat by calling the ‘runAICheat’ function. This function will firstly display an indicator that the AI is cheating,

then make the AI swap their cards, before stopping the AI from cheating and thus removing the indication. Knowing this, it can become easy to predict when the AI will cheat and at any point during this coroutine function, the player can choose to call the opponent out.

Figure 6.17. The opponent cheating (indicated by the "Call Out?" popup).

When swapping cards out, it would require significant memory and computational power to find the best combination of hands to sum up to twenty-one. Instead of this, we attempt a faster method of taking the last card in the opponent's hand (the card that made the opponent go over twenty-one), and then search for a card with a lowest value out of their three available cards. This swaps the opponents high value card with a much lower one. Although this method means that the AI will most likely be unable to select the best combination of card to win, efficiency was priority here which meant that the AIs intelligence had to be sacrificed for speed.

6.5.2. fightAI Script

A small portion of this script was obtained from a public GitHub repository written by Jon Dev Tutorials [20], as such, the RandomPoint() function was not created by myself although it has been edited slightly and is still worth mentioning as it directly controls where the opponent chooses to move towards. RandomPoint is called by a recursive coroutine with a slight delay to select a position in the environment within a certain radius of the game objects current position. When the scene loads, every opponent selects a random point within the bounds of the NavMesh floor. The Unity's built-in NavMesh system allows for AI pathfinding using the A* algorithm. This meant that we did not have to develop our own system, but instead could select a location within the confines of the map. Opponents will constantly select a position on the map and attempt to go to that location, however a new position is set every few seconds which forces them to change directions. This causes the AIs to act as if they are frantically moving around the scene similarly to how a player would. Likewise, the AIs will also constantly rotate themselves to look at a random player when moving around, changing to look at a different player every so often or if the previous player had been removed from the scene. This makes it look like the opponents are looking around and scanning their environment before shooting. The combination of random movements and looking at others simulates how a player would act in the scene. This script runs for each individual opponent leading to each of them running in a different direction each round.

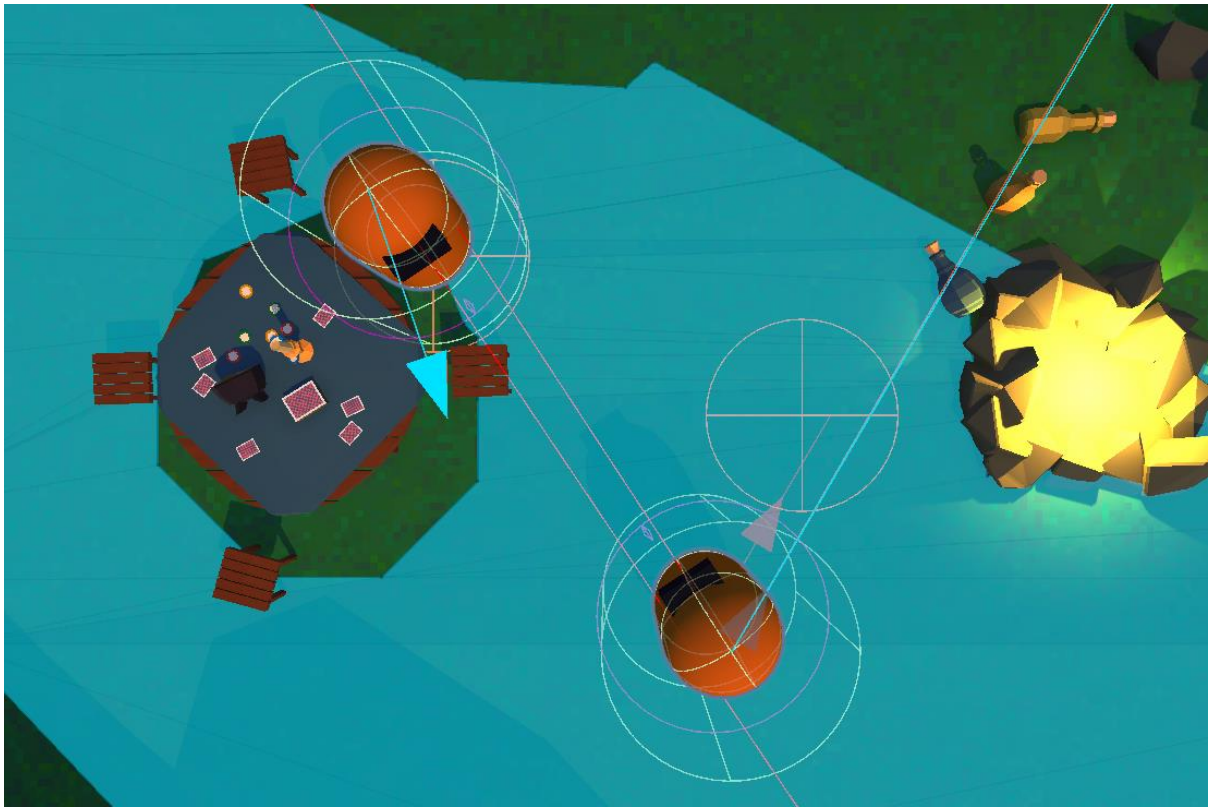


Figure 6.18. The opponents from a top-down view. Each opponent will move to a random location (indicated by the pink circle) within the NavMesh area (blue floor). The red Raycast shows where the opponents are looking.

The fightAI script also controls when the opponent will shoot. The opponent will only attempt to shoot after a certain delay, and only if they still have cards remaining to throw. When shooting, the AI will be looking at a randomly selected player, however they are also likely to miss their shots as they may be midway through a turn as they shoot. This makes the AI more realistic as it'd be unlikely that they would be perfect in every fight and would be frustrating for the player if they were too accurate when aiming. This inaccuracy gives the player a chance to orient themselves and line up their own shots as they won't be targeted and lose immediately.

6.6. Challenges

This section will mention challenges that arose during the development of AceUP and what was done to resolve these issues. These challenges are important to note as a significant amount of time was spent on these tasks which took up a decent amount of the development process.

6.6.1. 3D Modelling Buildings and Environment Design

During development, a large amount of time was spent on creating 3D assets to fill the scene. The game scene went through multiple iterations before the large open space that we have today. Although it was only planned to take four weeks in the project plan, this timeframe massively undermines the actual time used to create the assets. Although AceUP is a working demo of a game, I feel as if more time could have been spent polishing code and refining different features of the game.

The focus on environmental design also meant that there was no time to create additional assets such as player models. This has left all opponents models as a simple capsule with a cube on their face to signify eyes. Although adding player models would've been ideal, the need to rig and animate the bones meant that it could not be implemented during development.

Whilst I am satisfied with the overall look of the game, I feel as if the feel of the game could have been improved had I not spent so long in Blender creating models.

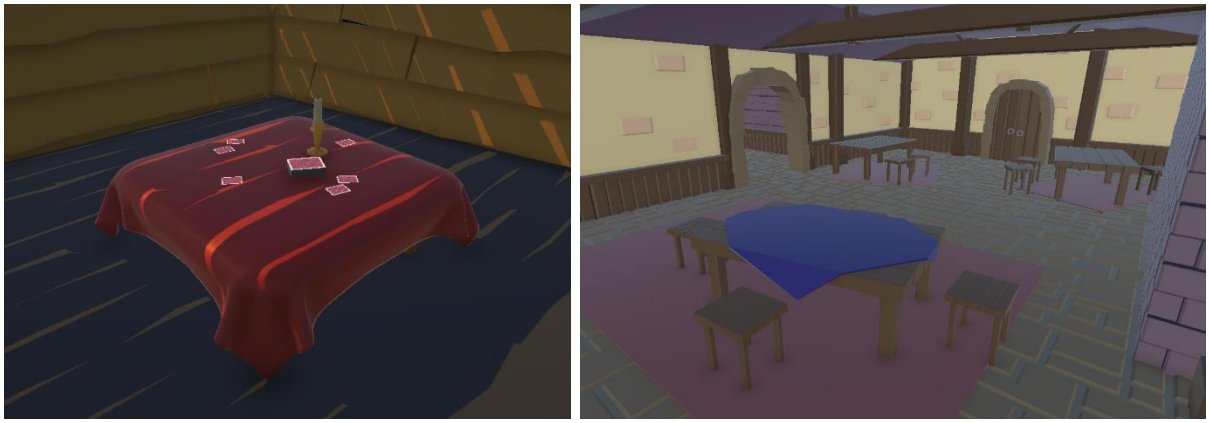


Figure 6.19. Previous iterations for the level design.

6.6.2. Ace Values and Multi-Valued Cards

Working with Aces was difficult as they can have two different values in Blackjack. This meant that the standard method of adding up values did not work and an additional function was needed. Although in hindsight, this was a simple fix of subtracting the difference to get the lower value, this only allowed for Aces to have a different value. When first managing multi-valued cards, it would have been much preferable to have unique cards, such as negative cards, multiplier cards, et cetera. However this was not added as it required the game to search for the best combination of card values to achieve a twenty-one. A fix to this issue could have been to manually have the player to select what card value they would want, however this would've added too much complexity to the game and would've likely overwhelmed the player of having to cheat and attempt mental maths for each card combination.

Automatically searching for the best combination of cards would be a time consuming process and would've caused to game to run much slower despite any optimisations we made. The player can have a maximum of six cards in their hand alongside an additional three cards to cheat with, each with a chance to have a different value. On top of this, had there been additional cards which had multiple values, the opponents AIs would also run much slower as they'd have to attempt every combination of card to get the best value they could. This would drastically reduce the pace of the game as it would have to calculate what combinations could be made.

6.6.3. Card Highlighting and Swapping

Although not inherently a difficult process in theory, the handling of card swapping and selecting cards was quite difficult due to the unpredictability of the players actions. The player could perform multiple actions when they start cheating.

Such as:

1. Hovering over a card
2. Clicking on a card
3. Hovering over a different card
4. Selecting a different card
5. Deselecting their current card
6. Selecting an invalid blank cheat card
7. Hovering over a valid cheat card
5. Selecting a valid cheat card

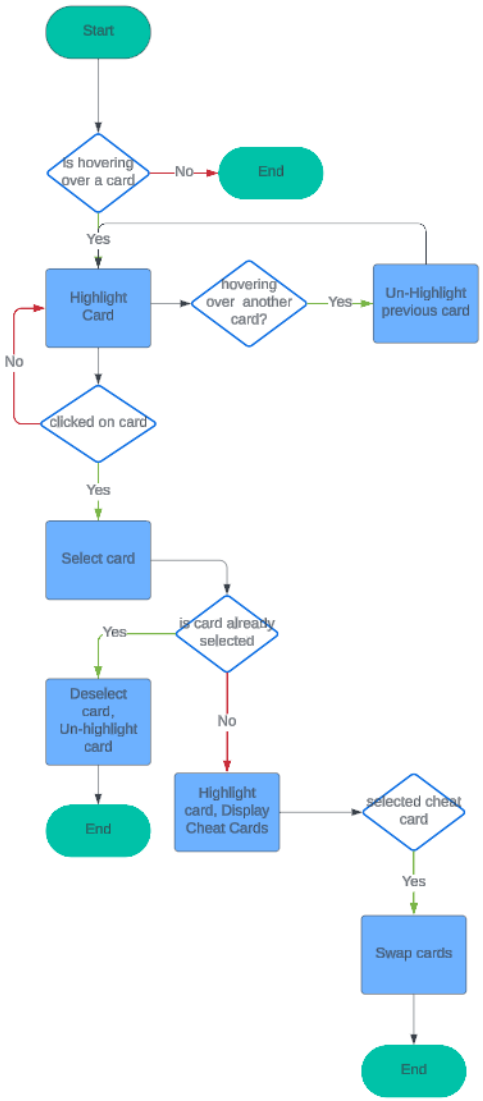


Figure 6.20. A flowchart for possible actions for card swapping and highlighting.

Each of these actions all had to be accounted for in the playerManager and a swap of cards can only occur when the player has selected two cards to exchange. These actions could also happen at any order during the entire round so frequent checks must be made to ensure that the player can cheat properly. This meant that the program had to be rigorously tested to make sure there were no bugs with the code.

6.7. Conclusions

To conclude, although functional and playable, there are still many feature of AceUP that could be developed upon. At present, besides the gameplay loop of earning chips to purchase cards to get higher scores, there's currently no other incentive to play the game. This causes the game to be fun at first, but to become boring later on when the player has an abundance of money. The AI is also currently far less capable than what I'd hope for which means that the player is not exactly challenged in the game and it becomes a slow grind to play. Despite this, the AIs do have a high chance to win against the player and offers some challenge if the player does not play strategically enough. Therefore I feel as if the game achieves the original goal of combining two genres due to the working Blackjack game and the First-Person shooter elements incorporated. The player can play Blackjack fairly, or cheat and risk starting a fight which is the core goals we had set out to fulfil this project.

In chapter 8, a list on some future developments will be made for this game and how they could offer a better gameplay experience. This will cover some challenges introduced in section 6.6 as well as additional features I'd like to implement but was unable to within the time frame given.

7. User Testing and Playtesting

During development, playtesting was mostly done by myself until in-game feedback was given to players rather than in console logs. Once the game provided responsive feedback, I started to gather people to play AceUP. This led to some reviews about the game and gave a good idea of the general gameplay experience.

7.1. User Testing

User testing was done throughout the development cycle by using console logs. This was used to determine what occuring in the game and provided feedback to the players input. Figure 7.1 shows console logs of a typical game.

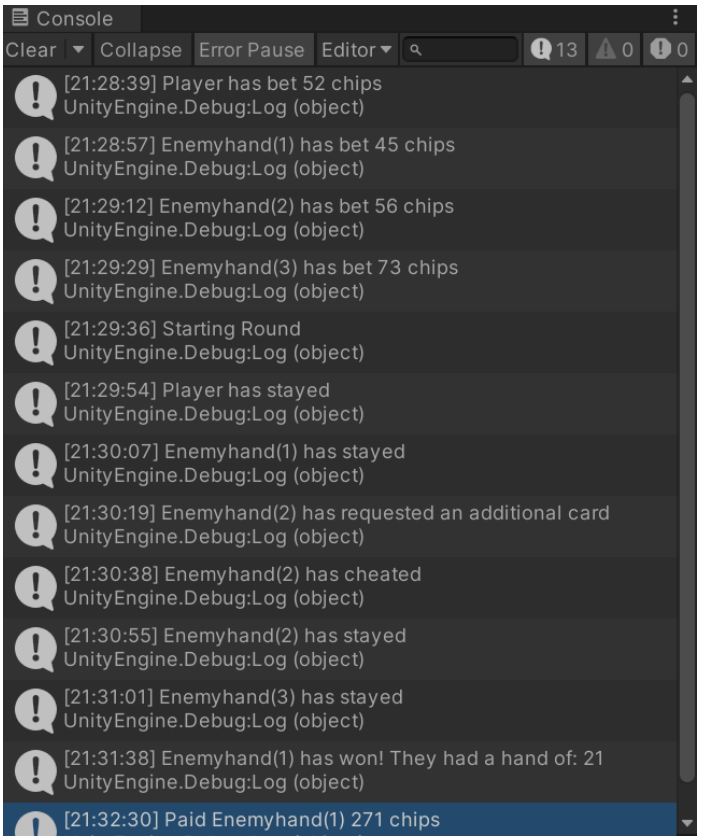


Figure 7.1. Console logs printed during a typical game

By referring to these debug logs, a general sense of how the game would play was given. This helped during development to find any errors that may occur. In the code, the debug.log method also provided itself as a placeholder and reminder of where in-game feedback needed to be given. These methods were eventually replaced to allow the player to visually see what would occur in real-time games.

7.2. Player Testing

This section will document the feedback given by players and how these issues were or could be resolved.

Feedback	Fix
The rules of the game are a bit confusing at first. It was a bit confusing when I clicked on a card and it highlighted red.	Add a cheat prompt above the playing cards to let players know that they are cheating when they click on a card.
Enemies immediately start shooting at the player which makes it difficult to orient myself when I first load in the fight.	Add a slight delay before enemies can shoot when the fight scene loads in.
The betting phase can take a long time, especially as I continued playing for longer and everyone else had a surplus of money to bet.	Decrease the time it takes for enemies to place bets. Currently the duration is based on how many chips they bet, potentially include a multiplier that decreases the delay as more chips are betted.
Money can become redundant quickly, resulting in cards being easily bought from the shop as the rate of earnings is very high.	Reduced the starting currency (from 300 to 100) so that the player doesn't immediately buy cards to win. This is a temporary fix implemented into the game. It does not fix the overall issue of players earning a lot of money in the long term. This issue arose in discussion with my project supervisor, who recommended to implement a card cost multiplier based on the players total currency. This would make the difficulty maintain as the game progresses and incentivises the player to place bigger bets.
The card texture doesn't really look like a real life deck of cards. When prompted further, they stated that in particular, the spades and clubs were confusing to look at	Although the suite of the card does not matter in Blackjack, a potential fix would be to increase the pixel arts' resolution so that the card texture could be drawn clearer and more akin to a real deck of cards.
The fights last too short, I barely had time to realize what's happened. Note: The opposite was true when the time was too long.	Adjust the time until the best average time is found for players.
I would've had more fun playing the game had it been multiplayer.	Although ideal for the game to be multiplayer, this would have been incredibly difficult to implement into the game within the timeframe and also falls

(continued)

	outside the scope of this project. Despite this, some research has been made on Photon and how it could be implemented into the game in the future.
It's a bit confusing when first cheating and seeing three blank cards and clicking on them do nothing.	Add responsive feedback whenever the player clicks on the blank card stating “You can’t select a blank card”. An alternative approach would be to hide the card game object if it currently has no value (blank).
It can be a bit obvious when the AIs cheat as you can easily constantly watch them all in the peripherals and catch when the text prompt appears.	Force the player to have to look directly at the opponent in order to reveal if they are cheating.

8. Conclusion and Future Developments

8.1. Results

This report concludes with the success of the project as we have created a working demo of a game featuring two genres with multiple complex features implemented such as a card inventory system, scene state persistence, a dialogue system, and opponent artificial intelligence as well as an environment that fits the general aesthetic of the game. Due to the two completely different mechanics of this game, with the tabletop and the first-person shooter portion. We have essentially made two games in one. This challenging task should not be undermined as each genre could have been split into its own game.

Despite this, we should also mention how this project could have been improved. Whilst at the start of the project, consistent and cleanly commented code was maintained, As the deadline drew closer, the code slowly became messier which can be noticeable in certain scripts such as the DialogueManager. I also feel as if less time should've been spent on 3D asset creation which would've allowed for more refinement in the game and this report. It should also be stated that this game does not prove as a good example on whether combining genres can lead to fun gameplay. This decision would come down to the individual player. Although I find this game fun, others may find it monotonous. As such, further research and the creation of multiple games following this idea should be produced. Nevertheless, I believe AceUP is a unique concept and provides a different twist on the standard game of Blackjack.

8.2. Future Development

This section outlines what could be added to the game and how it adds to the gameplay experience of this project.

8.2.1. Player Models

One of the most noticeable aspect of the game is how the player and enemies are all generic Unity capsules. Although this was only a placeholder meant to be used temporarily, this became the final design of all people of the game. Due to a lack of development time, and slight inefficient use of my time, I was unable to model, rig and animate a humanoid model into the game. This is disappointing as interactions with more emotive/ lifelike opponents would have greatly improved the feel of the game.

8.2.2. Audio and Music

During development, music and any additional audio was neglected. Whilst developing the game, I had gotten accustomed to the lack of audio. This was a mistake on my part and sound effects such as opponents talking, chips hitting the pot, cards throwing, et cetera would have greatly improved the feel of the gameplay. Although I lack the experience in creating music, it would have added to the ambience had a soundtrack been incorporated into the game.

8.2.3. Unique Blackjack Cards

As mentioned in section 6.6, cards with different values would have created a more unique Blackjack experience. This feature was unfortunately scrapped during development due to the challenges I faced when incorporating cards with multiple different values. Despite this, in the future, I'd like to attempt to add this feature once again as I feel as if it would have made the game more fun overall and I feel quite disappointed that I was unable to program this in.

8.2.4. Card Damage Values

During one of the meetings with my supervisor, I was asked if cards instantly eliminated opponents. This sparked the idea of cards dealing damage varying damage to others rather than all cards being a one shot kill. This would mean that the player had to account for what card value they have in their hand before the fight, rather than how many cards they have in total to throw. This would also fit well with the theme of the game where players must constantly be adding their hand value and remembering this mentally. By having different damaging card values, I feel like this would allow for fights to be more challenging as it's no longer about who can shoot the fastest and most accurate.

8.2.5. Smarter Enemy AI

Although the opponents are capable of quickly navigating around the scene and move around frantically similarly to how a player would, one noticeable difference is how none of the enemies are capable of jumping in the game. Whereas it would be less obvious if we also disable player jumping, this felt like the wrong approach to handle things. Instead, it would be ideal if the enemies could jump around in the scene similar to how the player is able to. I was unable to implement this into the game as the NavMeshAgent component attached to the enemies meant that they'd have to be stuck to the floor or else they could not navigate the environment. Although an attempt was made to have the opponents be able to move around and jump, I could not discover a way to do this.

8.2.6. Options and Settings

Although the game features a main menu, the player can only choose to start or quit the game, or access the tutorial. Ideally there would be an options and setting interface that allows the player to adjust the game to their needs. This feature would likely allow the player to deactivate post-processing such as bloom and lighting which could affect those who are sensitive to such effects.

8.2.7. Multiplayer Gameplay

As mentioned previously, a multiplayer mode would have been ideal to create a fun game, adding a larger competition aspect to the game. This could be added using the Photon package provided by Unity to create a local server. I believe a multiplayer mode would enhance the overall feel of the game and could provide more challenging gameplay.

9. References

[1] Mix and Game Jam. Mix and Jam. https://itch.io/jam/mix-and-game-jam-2020. (Accessed 01/05/2025)

[2] Dungeons and Degenerate Gamblers. Purple Moss Collectors. Purple Moss Collectors. Due to release 2024

[3] FPSChess. Meta Fork. DigiPen Institute of Technology. 2022

[4] Portal. Valve. Valve. 2007

[5] OfficialGameRules. Beginners Guide to Blackjack. https://www.officialgamerules.org/blackjack. (Accessed 30/04/2024)

[6] Unity Technologies. Unity 3D. https://unity.com/. (Accessed 23/04/2024)

[7] Unreal Engine. Epic Games. https://www.unrealengine.com/en-US/. (Accessed 23/04/2024)

[8] Godot Engine. Godot. https://godotengine.org/. (Accessed 23/04/2024)

[9] Blender. https://blender.org/. (Accessed 23/04/2024)

[10] Paint.NET. https://www.getpaint.net/index.html . (Accessed 24/04/2024)

[11] Adobe. Photoshop. https://www.adobe.com/uk/products/photoshop.html/. (Accessed 24/04/2024)

[12] Marie Dealessandri, Alex Calvin. What is the best game engine: is Unity right for you? https://www.gamesindustry.biz/what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you. (Accessed 30/04/2024)

[13] Visual Studio. https://visualstudio.microsoft.com/. (Accessed 23/04/2024)

[14] Sebastian Kozlowski. CPU and GPU bottleneck: How to fix a bottleneck in your PC. https://www.wepc.com/tips/cpu-gpu-bottleneck/. (Accessed 29/04/2024)

[15] Jeiel Aranal. Sprytile Documentation. https://docs.sprytile.xyz/. (Accessed 24/04/2024)

[16] NeatCraft. Minecraft Medieval Town – FREE DOWNLOAD. https://www.youtube.com/watch?v=tBMeN-FgM-A&ab_channel=NeatCraft (Accessed 02/04/2024)

[17] Eric Haines. Mineways. https://www.realtimerendering.com/erich/minecraft/public/mineways/ . (Accessed 02/04/2024)

[18] Unity Technologies. Unity Documentation. https://docs.unity3d.com/ScriptReference/Object.DontDestroyOnLoad.html . (Accessed 28/04/2024)

[19] Lauren Bennett, Alyssa Syharath. Scriptable objects: Supercharging game development in Unity. https://embrace.io/blog/scriptable-objects-in-unity/. (Accessed 29/04/2024)

[20] JonDevTutorial. RandomNavMeshMovement. https://github.com/JonDevTutorial/RandomNavMeshMovement. (Accessed 27/03/2024)

[21] BOXOPHOBIC. FREE Skybox Extended Shader. https://assetstore.unity.com/packages/vfx/shaders/free-skybox-extended-shader-107400. (Accessed 15/04/2024)

10. Appendices

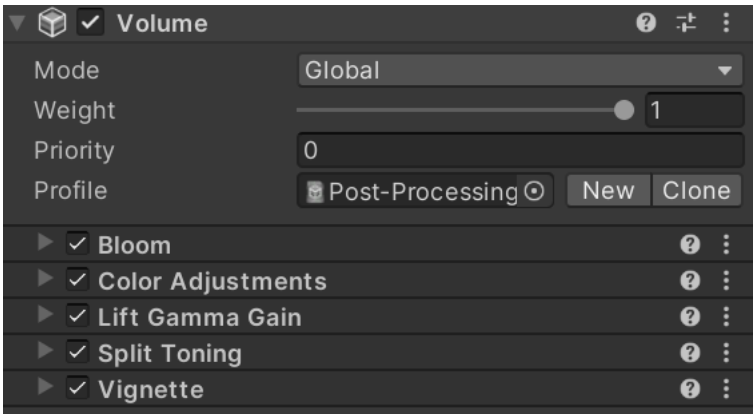
A. FPSChess

FPSChess takes the board game and merges it with a shooter, this causes the game to not only require strategy, but also skill. The twist causes pieces to be more difficult to take over. A king can no longer be checkmated as it will fight back. A queen is no longer stronger than pawns. Although each chess piece has a unique set of abilities, any piece can win in a fight with enough skill. This nullifies the mundaneness of a standard chess match and provides a different strategy on whether to take a piece or to avoid it in general consequently creating a risk and reward type system. The risk being losing a valuable piece and the reward being less enemy pieces on the board. This proves that by merging two genres, a new sub-genre may emerge causing a unique playstyle on an already established classic.

Appendix A: A fight in FPSChess of a pawn going against another pawn.

B. Post-Processing, Skybox, and Particles

The game was created in Unity's Universal Render Pipeline. This meant that post-processing/ volumes could be added to the scene to create a more atmospheric ambience which fit the surrounding environments. As can be seen by the image below, a combination of bloom, colour adjustments, vignettes, and other filters were used. These were all manually adjusted to suit the feel of the game.

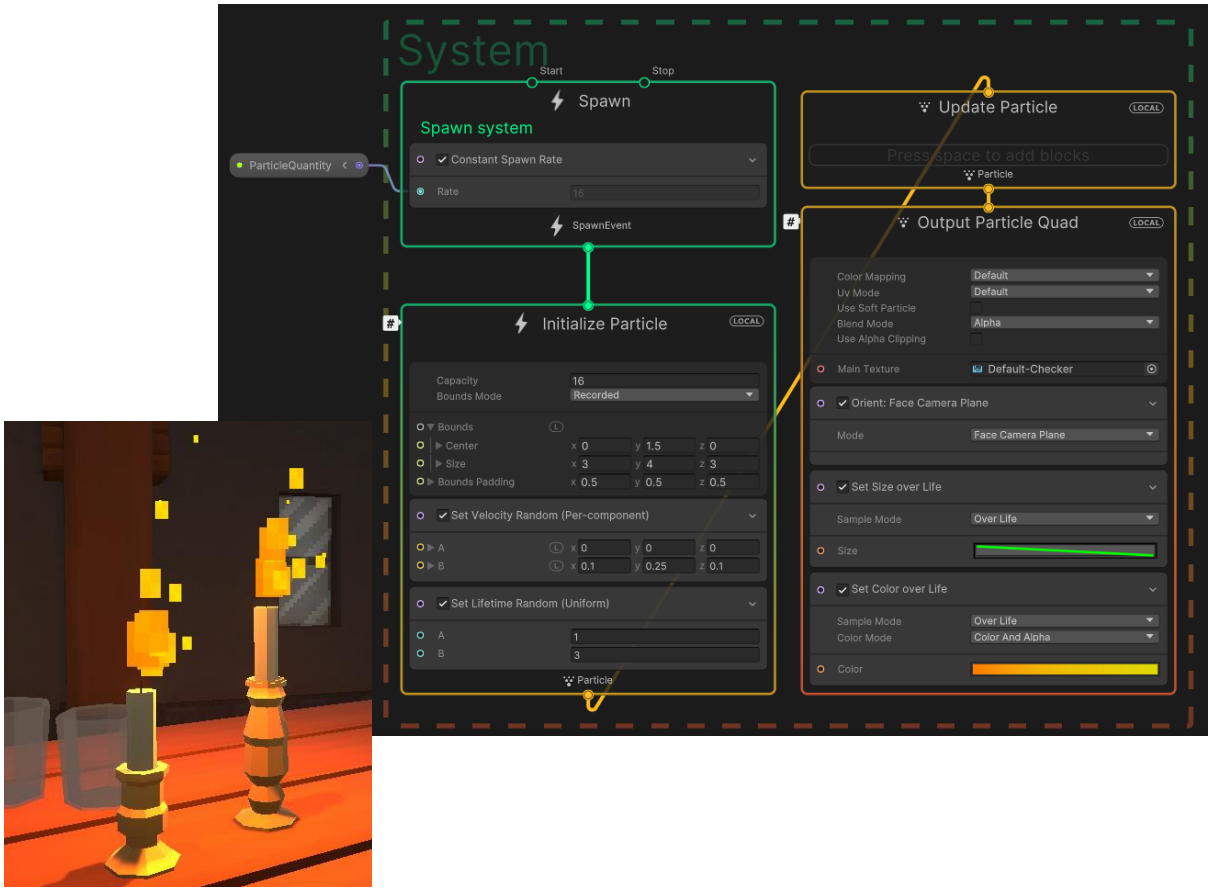


Appendix B.1: Post-Prossesing Volume used in the game.

The custom skybox [21] was downloaded on the Unity asset store. This skybox was dynamic which meant that the stars would move gradually over time. Although not noticeable unless focused on, I feel as if this added to the game.

Appendix B.2: A screenshot of the skybox used in the game.

Unity’s particle system was used to add particle effects to the scene. For the campfire and candles flame, a gradual gradient was added so that the flames colour changed over time as the size grew smaller. The use of particle systems greatly added to the atmosphere of the scene as it made the game feel more alive by adding a form of movement into the level.

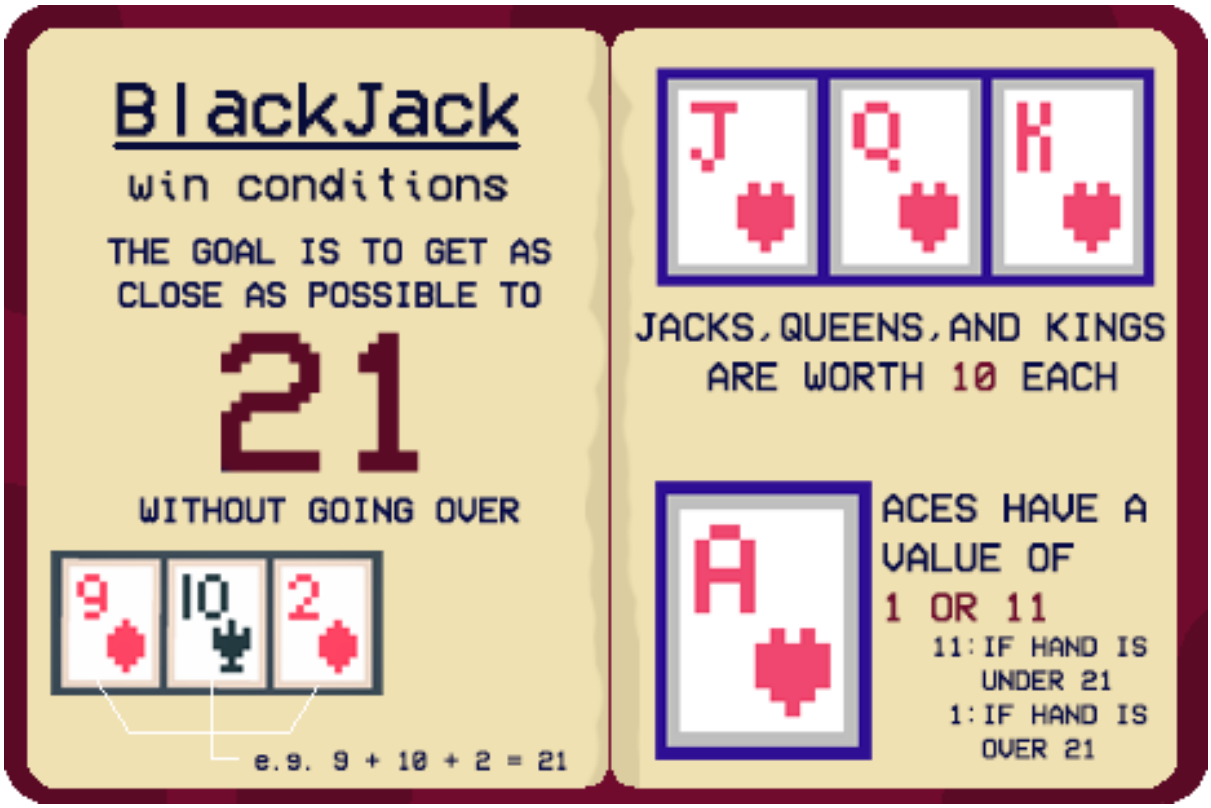


Appendix B.3: The particle system of the candle flame and the candle GameObject. The flames of the candle flicker in the scene.

C. In-Game Tutorial

The prototype of the tutorial was originally created using pencil and paper as simple illustrations, however, the official in-game tutorial was created in Photoshop using multiple layers. The image shown below is just one page out of fifteen created. These user interfaces are accessible on the main menu of the game, where the player can read and learn how to play the game.

The in-game tutorial was required in the game as many players did not understand how to play until prompted on what to do. This meant that the guide had to be clear and concise, having to cover all aspects of the game.



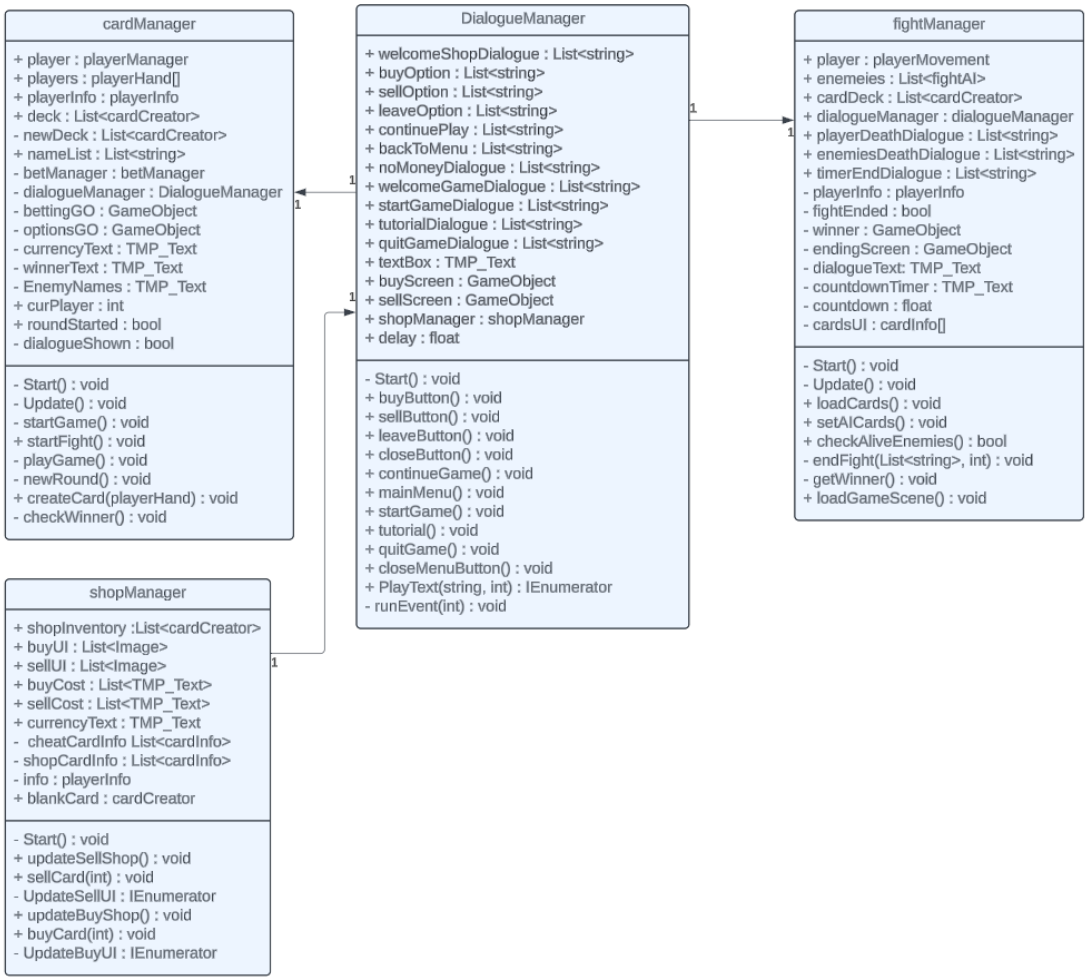
Appendix C: A page of the tutorial screen

D. Dialogue Manager

The dialogue manager is responsible for all dialogue that occurs in the game. It holds multiple lists of strings which randomly get selected when the game displays the text on the screen. The dialogue is manipulated by the players inputs via the button components which can be accessed from the user interface. These buttons are all linked to a corresponding public function.

The dialogue manager automatically starts when the other managers enable the dialogue GameObject. The PlayText() function takes in a string variable and an integer variable which stores the dialogue to display, and its corresponding event to run by calling the runEvent() function. This event varies from a scene load, quitting the game, and multiple other actions.

The DialogueManager is one of the last scripts modified for the game. Due to a lack of time towards the end of the project and the multitude of dialogue options, this script is less organized than I'd like it to be. Despite this, the code is readable and works as intended.



Appendix D: A class diagram of the DialogueManager and all scripts that it interacts with.

E. User Testing Compliance Form

User Testing Compliance Form for UG and PGT Projects*
School of Engineering and Informatics University of Sussex

This form should be used in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects".

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with **all** of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation.

If this is not the case, you should refer back to the "Research Ethics Guidance for UG and PGT Projects" document for further guidance.

---

1. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

2. The study materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of nonstandard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information are deemed to be high risk and require ethical approval through the relevant CREC.

---

* This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

4. No incentives were offered to the participants.
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

6. No participant was under the age of 18.
   Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

7. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.
   Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

8. Neither I nor my supervisor are in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

10. All participants have been informed of my contact details, and the contact details of my supervisor.
    All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.

11. The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session.

Participants must be provided with sufficient information prior to starting the session, and in the debriefing, to enable them to understand the nature of the investigation.

12. All the data collected from the participants is stored securely, and in an anonymous form.

All participant data (hard-copy and soft-copy) should be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymised form.

Project title: <u>Ace Up – A Card Based FPS Game</u>

Student's Name: Ethan Chan

Student's Registration Number: 22101741

Student's Signature: 

Date: 02/11/2023

Supervisor's Name: Paul Newbury

Supervisor's Signature: 

Date: 03.10.2023

F. Interim Report

Interim Report 3D Unity Game

Figure 1: Logo of the game

Contents

1. Introduction

1.1. Blackjack

In “The Book of Card Games”, Arnold describes Blackjack as a gambling game originating from France. [1, p. 28] The game, also known as twenty-one (or Vingt-et-un in French) is a relatively simple game. As the former name suggests, the goal is to achieve a hand as close as possible to twenty-one with an ace and a jack becoming the highest hand, a Blackjack. In the game, a person can choose to hit or stand. Hitting is when the dealer gives you an additional card. This can be done multiple times, but if your hand goes over twenty-one, you are out (also known as busting). On the other hand, if the person is satisfied with their hand, they can opt to stand. This is when your hand gets locked and play moves onto the next person until all players are satisfied with their hand. These simple rules have made Blackjack exceptionally popular in casinos however, it can become a bit stale and somewhat repetitive in longer games.

Blackjack has been implemented multiple times on computer games, however there is not a game where a FPS genre is mixed in with the game henceforth creating a unique experience for players who enjoy the game but would like a twist. Blackjack players can experience the new game mode as they choose to cautiously play as normal, or choose to gain an advantage on other players which consequently causes themselves to focus more on their actions and engage more in the game.

1.2. Aims

This project aims to design and develop a 3D video game based on the classic board game Blackjack merged with the genre of a first-person shooter. This project will be developed in a 3D Unity [2] environment with the aid of software such as Blender [3] and Visual Studio [4] (for 3D modelling and programming respectively).

The game features players surrounding a table in a first person perspective (as shown in figure 1.1 below). Every round, the player will receive two cards, where they can ‘Hit’ or ‘Stand similarly to a standard game of Blackjack. However, the player can also make the additional move of cheating which puts a twist on the game. The goal is to not get caught cheating in the game, because if caught, the player must eliminate all opponents they have played against. Ideally, there will be a shop using poker chips as a form of currency. This will provide incentive as a reward system for the game.

Once the game has been completed, user studies will start in order to provide an overview of the game in its finished state.

Figure 1.1: Concept model for how the finished game will look. The player has two cards that give a value of twenty-one but has no idea what other opponents have.

1.3. Motivations

This project provides experience on indie game development and covers mostly all aspects needed to work independently on a game, such as sound, graphics, programming, et cetera. By the end of this project, I also aim to have a finished project to show to potential employers alongside my portfolio.

As a Games and Multimedia Environments student, this project allows me to create a new game independently using skills I have learnt in my course. This includes 3D game design from 'Programming for 3D' and '3D Modelling and Animation'. As well as in game development from modules such as 'Game Design and Development'.

I have chosen to use the Unity game engine with C# as my programming language as it has been the primary language used throughout my course. I also have prior experiences with Unity where I have participated in game jams and other competitions which utilised this software.

1.4. Potential Issues

In making this game, a typical problem that may arise is scope creep. This is where a project expands over time which can cause unnecessary features to be added which could in turn ruin the overall feel of the game. One example of this is in the game Fallout 76. Bethesda had failed to create a cohesive game due to poor scope management. Too many features were added which overall took away from the feel from the original game. This included forced multiplayer gameplay, lack of a story, and promises that did not get added to the game which has caused Fallout 76 to be one of the worst released games on first launch. [5]

To avoid this problem, the requirements (noted later on in this report) will serve as a cohesive plan which allows for all mandatory features to be added before extra features. An agile workflow will also be used throughout development which will allow for a more dynamic approach in making this game.

1.5. Overview of chapters

This document serves as a starting report for the development of Ace Up.

As documented below, we shall cover:

- Background Research – Similar games, choice of game engine, and other design reasonings.
- Evaluation – What will be done during and after development of the game.
- Ethical Considerations – How the game will follow user testing guidelines.
- Requirements – What is needed of the game to function, as well as desirable features for the game.

This will serve as a guideline to what goals will be achieved for this final year project

2. Background Research

2.1. Blackjack and similar games

Blackjack is a popular game, with many replicas that can be found online. During research, A notable game stood out. Whilst still in development as of writing this report, Dungeons and Degenerate Gamblers (shown in figure 2.1 below) has a mechanic where players can buy special cards in the game.

This creates a new dynamic where it becomes closer to that of a deck builder than a standard card game. This spin on the game incentivises players to think outside the box and procure a strategy in the game. Each game is unique with cards being random from the shop. This makes the game repeatable as players try to come up with the best deck to win the game. The game however does not stray away from the core mechanics of Blackjack as the goal is to still beat the opponent by getting as close as possible to twenty-one.

Figure 2.1: A typical game in Dungeon and Degenerate Gamblers, special cards are shown being played.

2.2. Art

Indie game designers commonly use low poly renders for 3D games. This approach is efficient as the design process can be much faster than it would be if a higher vertices count were used [6]. Using lower polygons also boost performance on computers and can still look visually appealing. This form of art style can be seen in many games including Superhot and Journey, a FPS action game and a narrative adventure game (as shown in figure 2.2 below). The low poly assets lend itself in favour for these games to create a certain aesthetic. Simple models can be modelled and textured within a few hours which can be reiterated on multiple times without becoming too much of a sunk in time. This can be done in Blender with additional software such as Photoshop [7] for texturing materials for added detail. From the game Journey, we can see the game uses lighting to enhance the textures on materials. The use of shadows and highlights on low poly models completes its visuals and lends itself to an excellent storyline, all done with a low poly art style.

Figure 2.2: Low Poly art style from the game Journey, the lighting of the scene and texturing of the materials counteract the low poly art style lending itself to a new aesthetic.

2.3. Game Engine

I have opted to use Unity for the development of this project as opposed to other game engines such as Unreal [8] and Godot [9]. This decision was made largely due to Unity being primarily used throughout my course (such as in Game Design and Development, Programming for 3D, and Software Engineering) this therefore means it is the engine of which I have the most experience with.

Unity is also provided with a lot of extensive documentation online which will aid in research during debugging. Although this could be said for Unreal and Godot, due to the familiarity with the documentation in Unity, alongside the active community, this has become the ideal choice. In addition, Unity is an object orient programming environment tailored for both 3D and 2D games. This means that code can easily be organised and reused for multiple interactions in both a 3d and 2D scene. Godot is mainly used to create 2D games with little support for 3D game development which means that it will not be capable of creating a game fitting the scope of the project plan. On the other hand, Unreal Engine has excellent support for 3D development. Despite this, the game won't have any graphics that will be too high-end. This leads to the conclusion that Unity will be the best option for the game that I will be making.

2.4. Sound Design

Background audio is used in videos game to retain the players attention and to provide a environment befitting to the games themes. When looking into music typically found in casinos, a pattern emerged where music is played to create a sense of anticipation and urgency which aids in the enjoyment of card games such as Blackjack [10].

Opposing this, slower music can also be found in casinos. Slower tempos are used to ease those in casinos so that they feel less tense about the location. The music feels more relaxing consequently causing the atmosphere to become more addictive. This can be reflected in games where music can be used to keep the players retention whilst dynamically switching to a faster tempo to keep the game intense at a moment's notice. Utilising software such as Bosca Ceoil [11] can produce high quality 8-bit music which simulate the sounds found commonly emanating in slot machines.

2.5. Merging genres

A big issue when marketing a video game is to stand out and appeal to a user demographic. One method to stand out is to combine two already established genres. Mark Brown highlights that 'the advantage to this method is the creation of entirely new games' [12]. An example used in his video was Portal in which the game utilises the standard first person aspect of FPS games with that of a puzzler. This has created a new sub-genre for puzzles which has since sparked games such as The Witness and Superliminal. Portal uses a first person narrative to build on its environment and engage the player in a unique experience.

FPSChess (shown in figure 2.3) takes the board game and merges it with a shooter, this causes the game to not only require strategy, but also skill. The twist causes pieces to be more difficult to take over. A king can no longer be checkmated as it will fight back. A queen is no longer stronger than pawns. Although each chess piece has a unique set of abilities, any piece can win in a fight with enough skill. This nullifies the mundaneness of a standard chess match and provides a different strategy on whether to take a piece or to avoid it in general consequently creating a risk and reward type system. The risk being losing a valuable piece and the reward being less enemy pieces on the board. This proves that by merging two genres, a new sub-genre may emerge causing a unique playstyle on an already established classic.

Figure 2.3: A fight in FPSChess of a pawn going against another pawn.

2.6. First person shooters

In "Level Up! The guide to great video game design", Rogers emphasises the importance of the players point of view by utilising the camera to form one of the Three C's. [13, p. 121] As game designers aim to retain the attention of their player base, we must first understand what attracts a player to a game. Ever since the release of Maze War. The game became well known as the earliest form of the FPS genre. The use of a first person camera created a sense of immersion in game which drew in players at the time. This pioneered future games such as Doom, Wolfstein 3D and eventual triple-A titles.

In his book, Rogers mentions both advantages and disadvantages with a first person camera. While a first person point of view can create a more immersive and atmospheric situation, the player can no longer see their character, this formed a loss of an emotional engagement. Some games tackle this issue by allowing the user to toggle between multiple perspective. The game Minecraft (shown in figure 2.4) does this to provide players the freedom to choose how they wish to play. Minecraft is an open world game, this provides the added benefit of allowing players to choose what to focus on whether it be particular areas or to zoom out and view their surroundings more easily from a different perspective. Multiple perspectives becomes rewarding to the player who can now notice what that they could not have before whilst still immersing themselves into the game. However, we must ask whether the integration of multiple viewpoints works for all games or if it is just another form of an unnecessary feature.

Figure 2.4: Minecraft in first person and third person. The different perspectives gives the player freedom of what they want to see.

3. Evaluation

Evaluation will be completed in stages throughout the creation of the game. This will be dictated by how closely the game is to the original project plan, gameplay experience, how fun the game is, and feedback on the game by other people.

In order to accomplish this, throughout the project, playtesting will constantly be done by myself. At the end of the development process, final user testing will commence where users will play the game and leave their opinion on the game with some questions asked in an interview-like setting.

3.1. Playtesting

Playtesting will be done by myself throughout the development of the game. This will be done to ensure there is a minimum amount of bugs when participants test the game. Unity allows for quick runtime during the game in play mode. Occasional builds of the game will also be created to test the game outside of the Unity environment. This also serves as a form of backup in case some files get corrupted despite weekly backups of the game being made. Playtesting will be important as it gives an idea of what has been implemented on, and what needs to be done and improved.

3.2. User Testing

User testing will be carried out after the game is in a playable state. User testing will be important as it provides a real-word scenario of what an average person might do in the game. It will help document whether the game is too simple, or if it has been made too difficult. This will determine whether there are any flaws with the design which can be changed earlier on. In the final weeks of the project, more thorough user testing will be held, with multiple volunteers playing the game. All volunteers will be asked a series of questions in relation to how they found the game alongside a overall score of satisfaction for the game.

4. Professional and Ethical Considerations

Attached to the appendix is the signed documentation for the User Testing Compliance Form. As user testing will be required for play testing a game, ethical considerations are mandatory and have been noted:

Professional Considerations:

1. The game will contain work that has been fully made by myself. Some code may be from following a tutorial, in which case, it will be referenced in the final report. Similarly, art assets may take inspiration from online. This will also be referenced with images attached.
2. No depictions of drugs, alcohol, sex or other inappropriate acts will be made in the game.
3. No strong language will be mentioned in the game.
4. The game will not discriminate against age, race, sexuality, or any other categories that target a group or person.

Ethical Considerations:

1. **Participants were not exposed to any risks greater than those encountered in their normal working life.**
   Participants will be told the game is rated Pegi 18. No physical harm will be caused from the game but depictions of mild violence will be included in the game due to the nature of first-person shooters. Gambling will also be in the game (although with in-game currency – poker chips) as Blackjack is a game commonly played in Casinos.

2. **The study materials were paper-based, or comprised software running on standard hardware.**
   Standard equipment will be used throughout testing, here we will demonstrate standard equipment as my Dell Inspiron Laptop and Chichester's lab computers. Feedback will be written on paper, or on a Microsoft Word document on my laptop.

3. **All participants explicitly stated that they agreed to take part, and that their data could be used in the project.**
   Participants will be told prior to playing the game that related data will be collected (anonymously). Participants will be told what the project is about to get a better understanding of what data will be collected.

4. **No incentives were offered to the participants.**
   Feedback will be from volunteers who will not be bribed to participate.

5. **No information about the evaluation or materials was intentionally withheld from the participants.**
   All participants will be told that they would be playing a game similar to Blackjack, and taught that they can cheat during the rounds. Participants would also be made aware that there are consequences to cheating. In this way, participants will understand how the game works and what they will be playing.

6. **No participant was under the age of 18.**
   Participants will all be over the age of 18. They will also be university students or graduates.

7. **No Participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.**
   All participants will play the game equally, with no distractions. Testing will be done at a suitable time for participants.

8. **Neither I nor my supervisor are in a position of authority or influence over any of the participants.**
   All participants will be of equal authority to myself. (likely students or graduates).

9. **All participants were informed that they could withdraw at any time.**
   Participants will be made aware that they could withdraw at any point. This will be stated in the brief as well as before player testing occurs.

10. **All participants have been informed of my contact details, and the contact details of my supervisor.**
    A document will be handed out which include my contact details (via university email) as well as my supervisor's.

11. **The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session.**
    A document will be handed out which contains details about the game. Should participants have any questions, they can ask me at any point (as long as it is not spoiling the game). Before testing, participants will also be told what would be about to happen.

12. **All the data collected from the participants is stored securely, and in an anonymous form.**
    All data will be recorded either directly onto my personal laptop or on written paper which will be stored securely in a folder. No personal details will be collected (i.e. name, sexuality, address). Data will only be held for as long as necessary and in line with the Data Protection Act (2018).

5. Requirement Analysis

Despite Blackjack being a card game found commonly in casinos, the aim of this project is to create a game that anyone can pick up and play. This section aims to demonstrate what mechanics the game will have alongside some minimum features.

5.1. Usability

Creation of the game will be made on my laptop. This means that it will run on a Windows 11 primarily. The game should not have any noticeable frame drop nor any game breaking code.

At no point should players feel intimidated to start playing the game. This means that the instructions to play should not be too difficult to follow. Hence intuitiveness should be prioritized during the making of the game. To accomplish this, any instructions should be placed in a suitable user interface where accessibility is accounted for (such as clear menu indicators, straightforward gameplay loop, and legible text formatting).

5.2. Gameplay

As players (user testers in particular) only have a set amount of time to play the game, it should be intuitive and engaging from the start. Blackjack was chosen as the rules of the game are simple (aim as close as possible to twenty-one without going over). FPS games are common, with popular releases such as Call of Duty and Valorant. Players picking up the game for the first time should understand that they have to eliminate the opponents when they start attacking you. In order to stop players from cheating and looking at other players cards, it would be ideal to lock the movement in place so that they cannot walk around the table during play. This would later be unlocked when the player gets up and has to fight.

The game should have a suitable gameplay loop that is not repetitive. We can engage the player by providing rewards in game. This would provide reason to keep playing even after a round of Blackjack is played.

5.3. Requirements

To aid with development, requirements for the game has been split into two categories: mandatory and desirable. This will serve as a basis during the creation of Ace Up and will be constantly referred back to as a plan.

5.3.1. Mandatory

1. The game must run on the Windows 11 operating system.
2. The game must accept user input from keyboard and mouse.
3. The game must be played in first person.
4. The game must automatically detect when the player or enemy has busted (gone over 21).
5. The game must have sound effects and a simple music track.
6. The player must be able to play an ordinary game of Blackjack.
7. The player must be able to hit to add another card to their hand.
8. The player must be able to stand if they are satisfied with the value of their hand.
9. The player must be able to cheat whilst playing Blackjack. Cheating is when a player chooses to swap any card in hand with another card that they have in their inventory.
10. The player must be able to get caught cheating if not careful. Enemies catching the player cheat will call them out on it. This will start a fight with everyone.
11. The player must be able to take and deal damage to enemies. This will be done by attaching a health bar to the player and enemies.
12. The player must play against at least 3 other enemies.
13. Enemies must attack the player if they catch them cheating. This means that they will have simple AI and animations indicating an attacking motion towards the player.
14. Cards gained during Blackjack must be random from a set deck of cards.
15.

5.3.2. Desirable

15. The game should accept other forms of user input (such as a controller).
16. The game should include a player shop using poker chips as currency.
17. The game should include the option to play against different numbers of opponents.
18. The player should be able to change their perspective in game.
19. The player should be offered reasoning on why they should cheat (This should incentivise the player to play at a higher risk for more rewards).
20. The player should have a way to gain poker chips whilst playing Blackjack (these chips will be used in the shop).
21. There should be special cards not typically found in Blackjack to offer a further twist in the game.
22. There should be unique sound effects to provide a more immersive experience.

23. There should be a pause and options menu so that the player can take a break or adjust their settings at any time.
24. There should be support for those with disabilities so that anyone can play the game. (This may include changeable controls, high contrast setting, audio sliders, et cetera).

6. Project Plan

6.1. Gantt Chart

TASK	T1W1	T1W2	T1W3	T1W4	T1W5	T1W6	T1W7	T1W8	T1W9	T1W10	T2W1	T2W2	T2W3	T2W4	T2W5	T2W6	T2W7	T2W8	T2W9	T2W10	T2W11
Project Proposal																					
Project Research																					
Programming																					
2D Art Production																					
3D Art Production																					
Interim Report																					
Ethics Review																					
SFX																					
Game Polishing																					
User Testing																					
Final Report																					

Figure 6.1: A layout that will be used as a guide to what will be done each week during development and documentation.

6.2. Trello Board

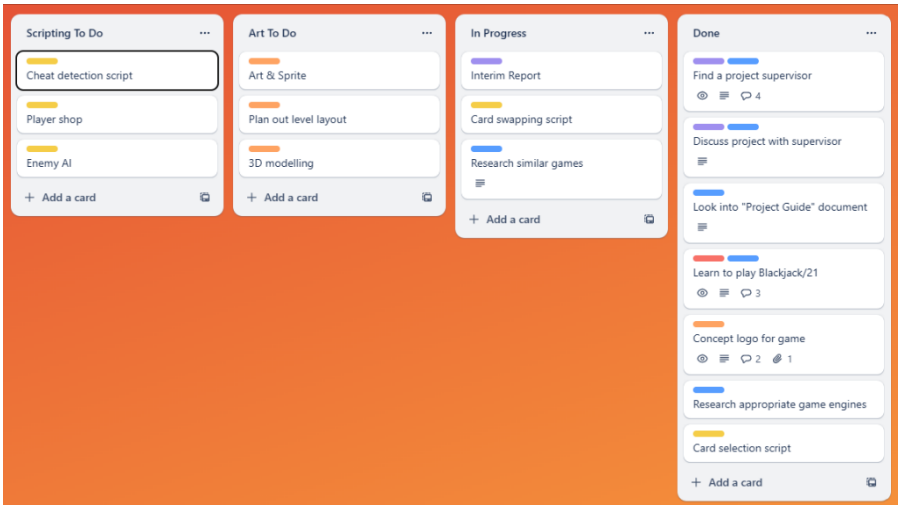


Figure 6.2: A Trello board with some tasks written in. (more tasks may be added on to expand on the games requirements)

6.3. Progress so far

Relevant research into similar games have been mostly completed alongside learning the rules of Blackjack including optimal strategies [1, p. 31]. This was done early on in this project. Which led to development starting in week 4. Further research may be needed later on in development to aid in the first person shooter portion of the game. The project proposal document (attached to the appendix) was sent to my supervisor in week 5 which has aided the interim report. This was when the Gantt chart and Trello board was created alongside early documentation on what the project will be about. During this time, work was constantly being done on the game with placeholder models placed into the game. Following the prototype model in Figure 1.1 as a guide.

Figure 6.3: Prototype models used for the game with random cards entering the player's hand. On the right is a button to 'Hit' and on the right a button to 'Stand'.

Multiple scripts have been written in Visual Studio. These include:

- **Player Manager:** This script determines the players actions during card gameplay. The player can choose to hit a button to gain an additional card, to stand to move on to the next player, or to cheat (which has yet to be fully implemented).
- **Player Movement:** This script allows for player movement, camera rotation, and jumping. The movement can be locked in place by a Boolean.
- **Player Hand:** This holds the players cards in a list, it also contains 2 Booleans (to determine if the player has busted or stayed). It also holds the function to reset the hand and a function to total the hands value.
- **Enemy Card AI:** Currently this script is a simple AI which dictates the enemies moves. If their hand is under 17, the enemy will ‘hit’ to gain an additional card, otherwise they will choose to stay.
- **Card Creator:** This is a scriptable object script which allows easy creation of new cards.
- **Card Info:** This assigns a card from a list of scriptable objects and displays itself for the player.
- **Card Manager:** This starts the round of Blackjack (by giving two cards to each player), determines player turn, and checks the winner at the end of the round.

```
Unity Script (4 asset references) | 7 references
public class playerHand : MonoBehaviour
{
    public List<GameObject> hand = new List<GameObject>();
    public List<GameObject> cheatCards = new List<GameObject>();

    public bool isBust;
    public bool isStay;

    public int handValue;

    //Adds up all the cards currently in your hand
    8 references
    public int totalCardValue()
    {
        handValue = 0;
        foreach (GameObject card in hand)
        {
            handValue = handValue + card.GetComponent<cardInfo>().cards.cardValue;
        }
        if (handValue > 21) { isBust = true; }
        return handValue;
    }

    //Reset player hand
    1 reference
    public void resetCards() {
        foreach (GameObject card in hand) { Destroy(card); }
        isBust = false;
        isStay = false;
        hand = new List<GameObject>();
    }
}
```

```
//Resets all players cards, and gives 2 new cards to each players
3 references
void newRound()
{
    newDeck = new List<cardCreator>(deck);
    curPlayer = 0;
    for (int index = 0; index < players.Length; index++)
    {
        players[index].resetCards();
        for (int j = 0; j < 2; j++) { createCard(players[index]); }
    }
}

//Creates a new card from available cards in deck, Gives the card to 'player'
3 references
public void createCard(playerHand player)
{
    if (newDeck.Count > 0) //If there are cards in the deck
    {
        GameObject card = new GameObject();
        cardInfo cardInfo = card.AddComponent<cardInfo>();

        int randomIndex = Random.Range(0, newDeck.Count);
        cardInfo.cards = newDeck[randomIndex];     //Give a random card value
        newDeck.RemoveAt(randomIndex);     //remove card from deck

        RectTransform rect = card.AddComponent<RectTransform>();
        rect.sizeDelta = new Vector2(.14f, .18f);

        card.transform.SetParent(player.transform, false);
        player.hand.Add(card);
    }
}
```

Figure 6.4: Some code currently in the game. On the left is the playerHand script, and on the right is part of the cardManager script

A logo has been illustrated which would become the games title screen. However the logo needs to be vectorised in order for animation to take place which has yet to be completed.

Figure 6.5: The Ace Up logo. The title screen will have the cards animated as if they are being thrown across the screen

7. Referenced Games

Fallout 76, Bethesda Game Studios, Bethesda Softworks, 2018

Dungeons and Degenerate Gamblers, Purple Moss Collectors, Purple Moss Collectors, Due to release 2024

Superhot, Superhot Team, Superhot Team, 2016

Journey, Thatgamecompany, Sony Computer Entertainment, 2012

Portal, Valve, Valve, 2007

The Witness, Thekla Inc, Thekla Inc, 2016

Superliminal, Pillow Castle, Pillow Castle, 2020

FPSChess, Meta Fork, DigiPen Institute of Technology, 2022

Maze War, (Steve Colley, Greg Thompson, Howard Palmer), N/A, 1973

Doom, id Software, id Software, 1993

Wolfstein 3D, id Software, Apogee Software, 1992

Minecraft, Mojang Studios, Mojang Studios, 2011

Call of Duty, Infinity Ward, Activision, 2003

Valorant, Riot Games, Riot Games, 2020

8. References

[1] Peter Arnold. The Book of Card Games. 1988

[2] Unity Technologies. Unity 3D. https://unity.com/. (Accessed 14/11/2023)

[3] Blender. https://blender.org/. (Accessed 14/11/2023)

[4] Visual Studio. https://visualstudio.microsoft.com/. (Accessed 14/11/2023)

[5]Grayce Slobodian. Medium. https://slobodiang.medium.com/the-downfall-of-fallout-76-1faedd807d08. (accessed 11/11/2023)

[6] Jolma, Valtteri, Animated Low Poly Characters. https://www.theseus.fi/bitstream/handle/10024/72726/Jolma_Valtteri.pdf. (Accessed 12/11/2023)

[7] Adobe. Photoshop. https://www.adobe.com/uk/products/photoshop.html/. (Accessed 14/11/2023)

[8] Unreal Engine. Epic Games. https://www.unrealengine.com/en-US/. (Accessed 14/11/2023)

[9] Godot Engine. Godot. https://godotengine.org/. (Accessed 14/11/2023)

[10]Auralcrave. Auralcrave. https://auralcrave.com/en/2022/08/18/the-impact-of-music-accompaniment-in-casino-venues/?expand_article=1. (accessed 12/11/2023)

[11] Bosca Ceoil. https://boscaceoil.net/. (Accessed 14/11/2023)

[12] Game Maker's Toolkit. How To Combine Video Game Genres. https://www.youtube.com/watch?v=H63Bqex1Urs&ab_channel=GameMaker%27sToolkit. (accessed 10/11/2023)

[13] Scott Rogers. Level Up! The Guide to Great Video Game Design. 2010

9. Appendices

9.1. Project proposal

Contents

Name: Ethan Chan **Candidate Number: 246667**

Supervisor: Paul Newbury **Contact: p.newbury@sussex.ac.uk**

Brief Description:

'Ace Up' is a FPS deck builder where you compete against opponents in rounds of Blackjack with the twist of being able to cheat. The goal is to win each round, but you can be caught cheating which starts a fight!

Aims:

'Ace Up' will be a new style of game which merges first-person shooter elements with that of a ordinary card game (Blackjack). There are similar games in this genre (e.g. Inscription, Dungeons & Degenerate Gamblers, FPS Chess) but none quite fitting the ideas of 'Ace Up' entirely.

As a 'Games and Multimedia Environments' student, I feel as if putting a unique twist on two beloved genres would be quite unique and stand out once completed. I hope to achieve a core game loop with challenging yet enjoyable gameplay.

Objectives:

Primary Objectives:

1. Card Based Gameplay: Functional replica of Blackjack where all players goes against each other (instead of just the dealer). At the start of the round, every player should get two cards. The player should be able to 'Hit' or 'Stay'. Hitting increases the players card by 1 from the deck. Staying means that play goes onto the next player where the cycle repeats itself. The player should not be able to move during this duration

2. FPS Gameplay: When any player is caught cheating, they are marked and must be defeated in a FPS style mode. All players can move around and target the cheating player. The player must have a weapon, either a gun or throwing cards. The cheating player must eliminate everyone that they played with in order to win. Once either all enemies, or the cheating player alone lives, more player AI's join the table to keep a consistent number of players.

3. Cheating Mechanic: Every player has a figurative and literal 'ace up their sleeve' in which they can secretly cheat and swap out their cards if they do not like it. These smuggled cards would either be random or bought from a shop (see extensions). In order to make the game balanced, the player can only hold three extra cards so that they cannot get exactly 21 each time and win too easily. During the duration of cheating, any player can catch you and a fight ensues.

4. Artificial Intelligence: To add meaningful gameplay, there should be opponents to play against. They should be able to toggle to and from the usual card game AI and the FPS AI. The AI must know when to hit, stand and when they have busted. The AI can also cheat at

any point but must know when they are being watched so that they can stop cheating to facilitate a natural response.

5. Level Layout: As the card game is Blackjack, it seems fitting for the game to be played in a casino, however alternatives are available such as in a pub to tailor towards a bar brawl theme. Regardless, the level should look decent as this will be where the game takes place. The scene should have a table with at least 4 opponents at any given time. When looking at your own cards, the player should not be allowed to see what their opponent is doing or else it would be too easy to catch someone cheating.

Extensions:

1. Multiplayer Gameplay: As an alternative to using AI, the game could be run on a local server so that you can play with friends. This would make the game more competitive as the player can go against people they know, however may prove difficult to develop during implementation.

2. Special Cards: The usual cards of Blackjack go from 2-10, Jack, Queen, King and Ace. As a bonus, there should be a rarer deck which can be used such as negative cards, multiplier cards, a card with no value, and cards with a value of 21. This makes the player think more about their moves and adds synergies to the cards that they play and cheat with.

3. Item Shop: After each game, the player should be able to upgrade themselves. This would be done in the item shop. The player should be able to upgrade their stats (e.g. Health, Speed, Damage) as well as improve the cards that they can cheat with. This adds some variety of gameplay and allows incentives for the player to keep playing the game.

4. Chip Currency: In Blackjack, chips are used as currency for betting. This can be used similarly to buy upgrades in the item shop. The chip currency should be gained from winning against other opponents. This provides some reasoning as to why the player might want to cheat as cheating means winning which equates to more chips to upgrade which improves our core game loop.

Relevance:

This project involves making a game in Unity. This has been the primary software used during the ‘Games and Multimedia Environments’ course (such as in ‘Game Design & Development’ and ‘Software Engineering’). The use of 3D graphics will also be used in this project which utilises skills learnt in ‘3D Modelling and Animation’ as well as ‘Programming for 3D’. In addition, this project goes beyond technical programming skills, as the game will have to be play tested, and researched into whether it is enjoyable. This goes beyond the skills learnt during my course.

I frequently play tabletop games such as ‘Exploding Kittens’, ’Monopoly’, and ‘Risk’ with my friends which makes you think about your moves. This project particularly interests me as it puts more strategy into a mostly luck based card game. A similar game to my project is ‘FPS Chess’ which can be downloaded from ‘Steam’. I particularly enjoyed this game as it puts a twist on the original board game and makes the game more uncertain each round. This project gives me a opportunity to create something similar that could be equally as popular.

Resources Required:

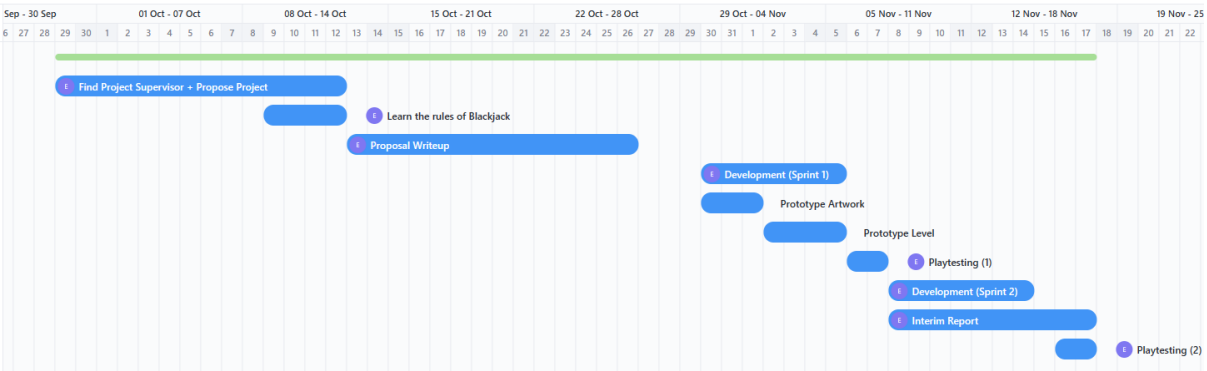
This project will be made in Unity, with 3D software such as Blender, Cinema4D, and MagicaVoxel. A Cinema4D licence may be useful but is not necessary as there are alternatives. Occasionally, a study room would be required for play testing when the game is completed.

Weekly Timetable:

This is the timetable from Monday 6th November onwards. I plan to use my free days on Wednesday and Friday as well as the rest of the afternoon on Tuesday for the project or other coursework's.

<table>
<tr><th></th><th>Monday</th><th>Tuesday</th><th>Wednesday</th><th>Thursday</th><th>Friday</th></tr>
<tr><td>11:00</td><td rowspan="4">Human-Computer Interaction Lecture</td><td rowspan="4">Programming for 3D Lab</td><td></td><td rowspan="4">Programming for 3D Lab</td><td></td></tr>
<tr><td>11:30</td><td></td><td></td></tr>
<tr><td>12:00</td><td rowspan="8">Project / Coursework</td><td rowspan="8">Project / Coursework</td></tr>
<tr><td>12:30</td></tr>
<tr><td>13:00</td><td></td><td></td><td rowspan="7">Visual Effects Lab</td></tr>
<tr><td>13:30</td><td></td><td></td></tr>
<tr><td>14:00</td><td></td><td rowspan="5">Project/Coursework</td></tr>
<tr><td>14:30</td><td></td></tr>
<tr><td>15:00</td><td rowspan="2">Human-Computer Interaction Seminar</td></tr>
<tr><td>15:30</td></tr>
<tr><td>Later</td><td></td><td></td><td></td></tr>
</table>

Project Timescale:



Bibliography:

Steam. (2021). Inscryption on Steam. [online]. Available at https://store.steampowered.com/app/1092790/Inscryption/ [Accessed 26 Oct 2023].

Steam. (2023). Dungeons & Degenerate Gamblers on Steam. [online]. Available at https://store.steampowered.com/app/2400510/Dungeons__Degenerate_Gamblers/ [Accessed 26 Oct 2023].

Steam. (2022). FPS Chess on Steam. [online]. Available at https://store.steampowered.com/app/2021910/FPS_Chess/ [Accessed 26 Oct 2023].

OfficialGameRules.org. (n.d.). Official Game Rules. [online]. Available at https://www.officialgamerules.org/blackjack [Accessed 26 Oct 2023].

Interim Log:

Group meeting with Paul Newbury (12th October 2023)

Individual meeting with Paul Newbury (27th October 2023)

9.2. Project Log

Term 1 Week 1

- Found a project supervisor
- Proposed Project to supervisor
- Started a draft of the project proposal write-up

Term 1 Week 2

- Met up with project supervisor (as a group)
- Discussed project proposal writeup as well as how to start Interim report
- Looked into project guide document

Term 1 Week 3

- Researched on how to play Blackjack
- Attended Ethics & User Testing Lecture
- Designed concept logo for game
- Planned out Gantt Chart
- Created a Trello board
- Finished project proposal writeup

Term 1 Week 4

- Met with supervisor (individually)
- Started Unity Project
- Created placeholder environment
- Scripted moveable player
- Created playing cards (as a Scriptable Object)

Term 1 Week 5

- Created simple 2D pixel art to playing cards using Paint.Net
- Modelled a 'Hit' and 'Stay' button using ProBuilder
- Scripted Simple AI for enemies to play Blackjack
- Assigned hand value to all players and automatically 'busting' if hand exceeds 21
- Scripted interactable actions with button to Hit and Stay

Term 1 Week 6

- Started on cheat mechanic
- Players can now select cards
- Hovering over cards highlights them
- Simple function for cheat implemented (with a Debug.Log for temporary testing)

9.3. Supervisor Log

Meeting 1: 12/10/2023 (14:00)

- Discussed project ideas
- Mentioned Pegi Rating and preference of game engine
- Discussed Project Proposal

Meeting 2: 27/10/2023 (12:30)

- Showed progress of game
- Showed project proposal
- Discussed about Gantt chart as well as Trello

Meeting 3: 9/10/2023 (13:00)

- Quick meeting where progress of game development was shown
- Discussed about Interim report due date

9.4. Signed Ethics Form

User Testing Compliance Form for UG and PGT Projects*
School of Engineering and Informatics University of Sussex

This form should be used in conjunction with the document entitled "Research Ethics Guidance for UG and PGT Projects".

Prior to conducting your project, you and your supervisor will have discussed the ethical implications of your research. If it was determined that your proposed project would comply with **all** of the points in this form, then both you and your supervisor should complete and sign the form on page 3, and submit the signed copy with your final project report/dissertation.

If this is not the case, you should refer back to the "Research Ethics Guidance for UG and PGT Projects" document for further guidance.

---

4. Participants were not exposed to any risks greater than those encountered in their normal working life.

   Investigators have a responsibility to protect participants from physical, mental and emotional harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, physical hazards or discomfort, emotional distress, use of sensory deprivation (e.g. ear plugs or blindfolds), sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity) or those which might induce discomfort, stress or anxiety (e.g. violent video games), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback.

5. The study materials were paper-based, or comprised software running on standard hardware.

   Participants should not be exposed to any risks associated with the use of nonstandard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and tablet computers is considered non-standard.

6. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.

   Participants cannot take part in the study without their knowledge or consent (i.e. no covert observation). Covert observation, deception or withholding information are deemed to be high risk and require ethical approval through the relevant CREC.

---

* This checklist was originally developed by Professor Steven Brewster at the University of Glasgow, and modified by Dr Judith Good for use at the University of Sussex with his permission.

If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, the data is to be published or there are future secondary uses of the data), then it will be necessary to obtain signed consent from each participant. Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script (see Appendix 1).

13. No incentives were offered to the participants.

    The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle. People volunteering to participate in research may be compensated financially e.g. for reasonable travel expenses. Payments made to individuals must not be so large as to induce individuals to risk harm beyond that which they would usually undertake.

14. No information about the evaluation or materials was intentionally withheld from the participants.

    Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants of the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require approval from the relevant C-REC.

15. No participant was under the age of 18.

    Any studies involving children or young people are deemed to be high risk and require ethical approval through the relevant C-REC.

16. No participant had a disability or impairment that may have limited their understanding or communication or capacity to consent.

    Projects involving participants with disabilities are deemed to be high risk and require ethical approval from the relevant C-REC.

17. Neither I nor my supervisor are in a position of authority or influence over any of the participants.

    A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any study.

18. All participants were informed that they could withdraw at any time.

    All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script (see Appendix 1).

19. All participants have been informed of my contact details, and the contact details of my supervisor.

    All participants must be able to contact the investigator and/or the supervisor after the investigation. They should be given contact details for both student and supervisor as part of the debriefing.

20. The evaluation was described in detail with all of the participants at the beginning of the session, and participants were fully debriefed at the end of the session. All participants were given the opportunity to ask questions at both the beginning and end of the session.

Participants must be provided with sufficient information prior to starting the session, and in the debriefing, to enable them to understand the nature of the investigation.

21. All the data collected from the participants is stored securely, and in an anonymous form.

    All participant data (hard-copy and soft-copy) should be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymised form.

Project title: Ace Up – A Card Based FPS Game

Student's Name: Ethan Chan

Student's Registration Number: 22101741

Student's Signature: 

Date: 02/11/2023

Supervisor's Name: Paul Newbury

Supervisor's Signature: 

Date: 03.10.2023

G. Project Log

Semester 1	Week 2	Meeting with Paul Newbury - Group discussion & Intro
	Week 4	Started Project Created temporary Environment Created a moveable player Created cards (with 2 cards going into each players hand) Met up with Paul - Showed Project and discussed what to put in interim report
	Week 5	Cards now have a design Simple 3D Models added (Hit & Stay Buttons) Player & Enemies can Hit, Stay, and Bust Players hand adds up all card values Enemies have basic AI to hit or stand
	Week 6	Started on Cheat mechanic Now able to select cards Hovering over cards highlights them Clicking on card prints "Cheat" – Console Line Met up with Paul - Quick meeting, showed project, checked if I had any questions
	Week 7	Finished Interim Report write up
	Week 8	Modelled Walls & Floor Modelled Table Added UI to Hit and Stay Added delay to Enemy AI Adjusted scene lighting Looking away from cards now stop you from cheating
	Week 9	3D modelling – Prototype Level
	Week 10	3D Modelling – 3D Environment props Full scene layout imported into Unity with materials
Semester 2	Week 2	Showed progress of work done over holidays Cheat Management Cheat Highlighting (Orange on Hover, Red on Select - click) Card Swapping (When Cheating) Shop Scene with dialogue system Card Purchasing (Chip Currency Cost) Card Selling (Sell Cost)
	Week 4	Showed betting of chips Asked for advice on how to deal with Aces (multi valued cards) Stopped players from gaining too many cards Updated chip betting
	Week 6	Updated Chip Betting - Bets now get taken from players, bets can range from 1 -> max player currency (cannot bet lower or higher) Auto selecting Ace value (1 or 11) with auto checking to make sure to get as close to 21 as possible

		Stopped player from requesting too many cards (limited to 6) Start of round now requires betting, instead of instantly gaining 2 cards for Blackjack Fixed a bug where AI's would not be able to finish their final turn whilst making a move (bool added: isMakingMove) Removed Stay & Bet button when it is no longer your turn Add Debug.Log() to show which AI is thinking, and what they do (e.g. create card & stay) AI can now cheat when they have gone over 21, they will swap the last card with a card in their cheat hand with the smallest value
	Week 8	Chip betting <- Gaining currency on win (normal games) Altered AI's Betting (Range from 10% to 70% of their currency) Check to see if player can still play (if they still have money) Check to see if AI can still play (if they still have money) Started implementation on fighting (shooting and throwing cards) Shooting requires cards from playing a game of Blackjack
	Week 10	Added cheat detection for player Created new scene specifically for fighting Added AI Movement following -> Enemy Patrol AI: https://www.youtube.com/watch?v=dYs0WRzzoRc&t=90s&ab_channel=JonDevTutorials Players cards get loaded between scenes Added UI to see how many cards the player has Fixed a bug where the player could no longer shoot in the new scene Decreased amount of cards the player has once shot Added AI shooting (AI's card count based on how many received during Blackjack)
	Week 12	Modelled 4 different houses Added houses to scene Created more textures for Unity materials (21 in total)
	Week 14	Adjusted Scene Lighting Added Post-Processing Added skybox: https://assetstore.unity.com/packages/2d/textures-materials/sky/fantasy-skybox-free-18353 Added and created more 3D assets to use in scene (trees, bar, rocks, grass, new table, lights, wooden bench, paintings, betting device & pot) Updated Fight Scene (with new environment & improved NavMesh Floor) Updated Shop Scene (UI and new environment) Updated Blackjack Scene (new Environment) Chip Pot shows total bet amount of the round

		UI added to show players currency and previous winner Added a button to go to the shop Fight scene determines winner, and gives money on last remaining player, if there is multiple, no one gets the money Timer added (countdown) in fight scene so it does not last forever (e.g. if everyone misses their shots) Enemies now display blank cards so the player cannot see what they got PlayerInfo contains more variables which persists between scenes (e.g. currency and total bet amount) Enemies now have a random name (which gets displayed in game), if they run out of money (less than 35% of players, they get replaced (new enemy has 75% of players currency)) Player can now be caught out for cheating (although there is no indication of when they will be caught) Main menu scene created Player info is destroyed at main menu so that the whole game resets (needed in cases where the player runs out of money (aka currency = 0)) Fight Scene linked to main scene and main menu Dialogue set up for fight scene and main scene Player is sent back to main menu if they have no money Made scene transitions nicer by adding a fade to scene animation when a level loads Added an animation to the main menu Built the game with adjusted player settings, app logo, title screen Started Playtesting and receiving feedback Planned out and started on report
	Week 15	Report writeup