BSc Computer Science (Including Placement Year)

"Lords" - An online deck-building game

Tyler Thorn - 1603041

Supervisor: Joseph Walton-Rivers
Second Assessor: Dr. Zilong Liu

Acknowledgements

I'd like to thank my supervisor, Joseph Walton-Rivers for his guidance and support throughout the course of this project.

I would also like to thank Charlie Adams for providing the card artwork for "Bloated Body".

I'd also like to thank Irina Carabella-Kozeni for her help with multiplayer testing, and feedback on this project and report.

Project Abstract

The objective of this project is to create a multi-player deck-building game, similar to real-world trading card games such as *Magic: The Gathering*. The game (hereby referred to by its title, *Lords*) focuses on resource management as a core part of its gameplay.

Lords is a fully implemented online card game. Players can create a deck, using a large variety of cards and put them head-to-head against other players from all over the world. The current card pool includes 26 cards each with a unique ability to help the player execute their strategy.

The game's implementation uses a custom written C# server and the client-side application was created using the Unity Game Engine and C# scripts. In the future, the game has ample room for expansion, including a wider variety of cards being implemented, and ranked matchmaking for more competitive players. The current game also lacks aesthetically and could be improved with artwork, sound effects and background music where missing.

List of Symbols

TCG - A trading card game, otherwise known as Collectible Card Game (CCG), is a type of game that mixes strategic deck-building and play with the collection and trading of cards.

MTG - Magic: The Gathering, a popular TCG.

UI - User Interface

Table of Contents

1. Context 6
1.1. Background Reading 6
1.1.1 Game Developers Conference Talks 6
1.1.2. A Study of Other Card Games 7
1.1.3. Online Unity Tutorials 9
1.2. Sustainability 9
1.3. Legality 10
1.4. Intellectual Property 10
2. Aims and Objectives 11
2.1. Main Objectives 11
2.2. The Game Design Document for *Lords* (Secondary Objectives) 11
3. The Final Product 20
3.1. Product Overview 20
3.2. Technical Achievements 20
3.3. Testing 21
3.3.1 Playtesting 21
3.3.2. Unit Tests (Functional Testing) 21
3.4. Technical Documentation 22
4. Project Planning 24
4.1. Project Management Software 24
4.2. Project Management Methodology 24
4.3. Maintaining Momentum, Adapting To Change and Recognizing Risk 24
4.3.1. Workload in the Autumn Term 25
4.3.2. UNet 25
5. Conclusions 26
5.1. The Deck-Builder and Card Pool 26
5.2. Online Play 26
5.3. Player Feedback 26
5.4. The Game Design Document 27
References 28

1. Context

The purpose for almost any game in society is to entertain and provide fun. For this purpose, *Lords* has to be enjoyable, easy to learn, yet have enough depth to allow players to improve thereby making the game more satisfying to those who choose to invest time into it.

1.1. Background Reading

This section describes the research I've done to obtain a background knowledge of the subject area. The sources of this research come in the form of talks given at industry conferences, successful card games and Unity tutorials.

1.1.1 Game Developers Conference Talks

It was important to take inspiration from other popular card games and their design decisions, and how these decisions impacted their success. I watched several talks given at industry conferences held by the designers behind popular TCGs. In these talks they provided insight into what they believe they did well during the creation of their games. This section covers two talks given at various Game Developers Conferences[1]. Each subsection details a particular talk, and the influence it has had on *Lords*.

Magic: The Gathering: Twenty Years, Twenty Lessons Learned

This subsection describes an industry conference talk[1] given by Mark Rosewater, Lead Designer of *Magic: The Gathering* (MTG), at the 2016 Game Developers Conference. The purpose of the talk was to present to the attendees twenty lessons learned throughout his career as Lead Designer, using examples of certain cards and releases throughout the life of MTG. As the context of this talk is applied to a card game, the lessons could be easily applied to my project. Particular lessons which I found to be most relevant are the following:

- Lesson #4, make use of piggybacking: matching the aesthetic of the game components to its mechanics can help the player learn these mechanics easier. For example, if there was a card type called "flying", where flying cards could only be interacted with by other flying cards, giving these cards the artwork of a flying creature would help the user grasp the mechanic easier.
- Lesson #14, Don't be afraid to be blunt: The player will understand a mechanic easier if it is named. In addition, information presented to the player should be clear. Subtlety should only be used where subtlety is required.
- Lesson #15, Design the component for its intended audience: In this particular context, when designing a card, understand who you're designing it for. For example, consider the following types of players: players who enjoy consistency in results, and players who enjoy successfully completing a difficult, gimmicky or inconsistent combo for an exciting reward. Having an interesting card that will appeal to both types of players can result in it appealing to none. Therefore, when designing cards, one should focus solely on one group of players.

Being an avid player of some TCGs, I found this talk to hold a lot of ground and opened my eyes to some of the design practices used in the games that I enjoy (even if those games are not MTG). These lessons influenced the design of a lot of the cards that I created, and even if those cards are lacking artwork, I have tried to correlate their name with their abilities (for example, "The Combat Medic" undoes the damage a card may have taken from previous battle).

[1] https://gdcvault.com/

Board Game Design Day: Balance Mechanics for Your Card Game's Unique Power Curve
This sub-section describes an industry conference talk[2] given by Dylan Mayo, who talks about the power curves of TCGs including *Magic: The Gathering, Hearthstone,* and *The Pokémon Trading Card Game.* In particular, he talks about how the cost of playing cards in these games correlates to their power level, and how it can be difficult to compare the "power" of cards when so many factors contribute to their usefulness within the context of the game.

The two key points he made when talking about *Hearthstone* in particular that I have taken into account are the following:

- The power of a card should increase exponentially with the investment a player must make into playing it, to make that investment more worth it for the player. To paraphrase, a card that costs 6 mana in *Hearthstone* should be more than 3 times stronger than a card that costs 2 mana. This variance in power usually comes from the card's ability.
- Not all cards need to fit onto the power curve perfectly - there will be cards that are weaker and stronger than others of the same cost.

I used the key points above throughout the process of designing cards for *Lords*. In particular, when designing high-cost cards I ensured the powerful abilities were worth the high investment.

1.1.2. A Study of Other Card Games

This section covers the TCGs which I already had experience with, or gained experience with throughout the course of the project, and how they influenced the design choices of *Lords*. Some of the influences are positive aspects of their respective games that I wanted to incorporate into *Lords*. Others are aspects that I wanted to steer away from because they would not help me achieve my goals with this particular project. In particular, I looked at card games that I have experience with and have shown success.

Yu-Gi-Oh!
Yu-Gi-Oh! has shown much success over the years. It has been running in North America and Europe since 2002[3] and it was the top-selling TCG in 2009 with over 22 billion cards sold[4].
One thing I like about *Yu-Gi-Oh!* is the consistency with which you can execute your strategies if you build your deck well, and wanted to emulate this in *Lords*. Initially, I thought about drawing inspiration from the Extra Deck mechanic in *Yu-Gi-Oh!* (a pool of cards which the player can always access so long as they meet the conditions for each card), but one complaint I see often from new players is the complexity that the Extra Deck can bring due to the wide variety of conditions that have to be met. As I wanted the core rules of *Lords* to be fairly simple, I decided to not include something that may confuse newer players so I searched for a simpler approach.
I considered *Yu-Gi-Oh!'s* approach to designing card effects, but another common complaint from new players is how the amount of text on some cards can be overwhelming (see Figure 1). Consequently, I decided in favor of using minimal text on cards.

Figure 1 - *Endymion, The Mighty Master Of Magic* from *Yu-Gi-Oh!*

Hearthstone

When designing *Lords* I knew that I wanted a source of consistency that players could rely on when building their deck. Part of the fun of a TCG is the luck of the draw, but it's not fun when you can't do what your deck is meant to, due to poor luck. To mitigate that, I wanted there to be a resource that players always had access to.

Ultimately, I turned to *Hearthstone* and its class system. *Hearthstone* is an online CCG available on desktop and mobile devices. It has been very successful since its 2014 release, achieving over 10,000,000 downloads on the Google Play store[5]. In *Hearthstone* when a player creates a deck, they have to choose one class for that deck from a selection of 10. Each class has a unique hero power, which is an ability the player can activate during their turn (for example, the Priest can pay 2 mana to restore 2 health to either hero or a creature on the field). Furthermore, certain cards can only be added to a deck when using a specific class, usually complementing the class' hero power. I drew inspiration from *Hearthstone*'s class system for *Lords*, and used elements of it when creating Lord cards, but made some modifications for the purpose of *Lords*. For one, I increased the power level of Lord cards, to make them more impactful. Also, I did not restrict any cards to a specific Lord, as I wanted to allow players to have as much freedom as possible with their deck-building choices. I also took inspiration from *Hearthstone*'s keywords that are used to shorten the card text (e.g. A card with the keyword *Taunt* must be selected as an attack target). This is also used in *Magic: The Gathering* to help players learn mechanics. To keep the text on cards as simple as possible, I have also used keywords in *Lord.* For example, the keyword *Bounce* in *Lords* refers to returning a card in play to its owner's hand.

Pokémon Trading Card Game

Out of the games on this list, I spent the least amount of time with the *Pokémon Trading Card Game*. However, there were still elements of its design that influenced *Lords* significantly.

For context, to win a game of *Pokémon Trading Card Game*, you have to use your own Pokémon (cards) to defeat your opponent's Pokémon in battle. When you do so, you gain a prize card. When you obtain 6 prize cards, you win the game. Naturally, Pokémon that require a low investment to put into play, yet have a high power level are popular. This is where GX and EX Pokémon come into play. GX and EX Pokémon are easy-to-summon, high power cards that you can put into play immediately. However, when a GX or EX Pokémon you own is defeated your opponent takes 2 prize cards, giving them a high-risk fact or.

I particularly enjoyed the high-risk factor of GX and EX Pokémon, and it influenced the gold system that *Lords* uses. I wanted players to be able to play the cards in their hands immediately, but I did not want them to mindlessly play all of their cards. Thus, I used GX and EX Pokémon as a source of inspiration when designing how gold works in *Lords* - you can use any card whenever you want during your turn, so long as you have the gold to pay for it. The caveat is that when you run out of gold, you lose the game. In doing so, playing high power cards with higher costs pose a greater risk, adding an extra layer of strategy to *Lords*.

1.1.3. Online Unity Tutorials

It was important to research how the implementation of a card game might be possible using the Unity Engine. I looked into online tutorials for multiplayer games on Unity and researched the best way to implement the client-side application. In particular, I used a YouTube channel called "Quill18Creates"[2], who has a series of tutorials on drag-and-drop UIs and a multiplayer game implementation, both of which apply to this project. These series are called *Unity 3D: Multiplayer First-Person Shooter*[6] and *Unity Tutorial - Drag & Drop Tutorial*[7]. The sub-sections below describe these tutorials and explain their relevance to the project.

Unity 3D: Multiplayer First-Person Shooter

This series of tutorials teaches viewers how to create a multiplayer game using the Unity multiplayer framework: UNet. While the example in the tutorial is for a first-person shooter, I still believe the tutorial was useful to teach the concepts of UNet, even if I did not use it in the final product due to certain limitations.

Unity Tutorial - Drag & Drop Tutorial

This series of tutorials teaches viewers how to create a UI that features game objects the user could drag and drop into different segments on the screen. This was relevant for the implementation of the client-side application of *Lords*, as the user puts cards into play by dragging them from their hand to their field.

1.2. Sustainability

In the future, *Lords* has ample room for expansion. Fortunately, this shouldn't be too difficult, as I've made as many efforts as possible to make the code expandable. This is due to modularity - the server code and client code is completely independent, so changing the functionality of a game mechanic requires a change in only one place. Card stats are kept out of the code base and can be modified by changing the "Cards.csv" file in the project's "Server" directory.

Changes could be made to improve the modularity. While adding cards is fairly simple, it does require duplication of information, as card values are stored both client-side and server-side and so data must be added to both. In the future, it may be beneficial to implement a method for the client to retrieve card values from the server.

Each type of card on the client uses a separate prefab, which acts like a template for Unity game objects. This means that if I make a change to the layout of the cards, subsequently I have to manually change every single card. In the future, it would be beneficial to utilize a prefab variant which is a base prefab that each child prefab derives from. Doing this would make it easier to implement changes to all cards and improve the sustainability of *Lords*.

[2] https://www.youtube.com/user/quill18creates

1.3. Legality

This project meets the licensing requirements for all tools used throughout its course. As I am not making any revenue from my use of Unity, I can use the "Personal" and "Student" plan. If I were to begin making a revenue of at least $100,000 (£80,000) per year, then I would need to subscribe to "Unity Plus", a business version of Unity for $40 (£32) a month. If that increases to $200,000 (£160,000) per year then I would need to subscribe to the "Pro" version of Unity for $150 (£120) a month. All conversions to GBP are approximate and accurate to the time of writing this paper.
The software does not store more information than is required, per GDPR laws. The only data stored about players at the time of writing this paper are the decks they create, and that is client-side only.

1.4. Intellectual Property

Lords does not draw on the intellectual property of other products, nor does it break copyright. All resources used in *Lords* are either created specifically for *Lords* or are general-purpose tools.

2. Aims and Objectives

The following section defines both the primary and secondary objectives of this project as they were initially defined. The secondary objectives were defined within a Game Design Document, written early in the Autumn Term, and defines most of the features and rules of the game, as well as the aesthetic. Parts of the game design document are outdated, as design choices and limitations have led to changes being made. As a result, the Game Design Document has been used as a guideline, as opposed to the equivalent of a Software Requirements Specification.

2.1. Main Objectives

The primary objectives to be met by the end of the project were defined when the project was undertaken in the summer of 2019. These are the following:

- Create an in-game deck builder that allows the player to create their own decks from a pool of at least 100 cards.
- Create online functionality, allowing players to play with each other from other locations.
- Introduce TCG players to *Lords* to playtest and provide feedback on the game. This feedback will then be used to polish the game and improve the end result.

2.2. The Game Design Document for *Lords* (Secondary Objectives)

The following is the Game Design Document I wrote for *Lords*. Some of the contents of this document are outdated, either due to the limitations of my artistic abilities, change of priorities during the course of the game's development, or feedback from my second assessor during the Week 11 interim presentation. These changes will be detailed in the "The Final Product" section.

Overview

Lords is a 1 on 1 card game set during a war between neighbouring kingdoms. There is an emphasis on the financial management aspect of war. Players take decks of their own making head-to-head against each other, made up of 3 types of card: Units, Utilities and a single Lord per deck. Units are creatures that stay on the battlefield and can enter combat with the opponent's units, or raid the opponent's gold supply should their battlefield be empty. Utilities are one time abilities with powerful effects, ranging from drawing extra cards, to gaining gold, to killing an opponent's unit. Each player also plays as a Lord, which is a powerful card with a unique ability that supports different strategies, but the Lord cannot attack units. The Lords' abilities will help you win the game through amassing gold, maintaining/gaining card advantage, etc. Building your deck to synergize with your Lord's ability is a key aspect to winning the game.
The card game is a fast-paced one, closer to *Yu-Gi-Oh!* than *Hearthstone* or *Magic: The Gathering*, to reflect the hectic nature of war. Players use Gold to do everything, including hiring units, using utilities, using unit abilities, and paying their unit's wages. If a player runs out of gold they go bankrupt. They are unable to pay their units wages and as a result, they begin to revolt and that player will lose the war should they not recover. As a result, gold management is a key feature of the game.

Players

Lords is a game for two players who go head-to-head with decks that they've created. Each card in the deck represents either a unit in their army (for example, a plague doctor) or a utility that they can use to get ahead (for example, a bank withdrawal). The exception to this is the Lord.
Each player can select one of several Lord's, which acts as not only their character with a unique ability but also as their avatar. This would be similar to the class a player selects in *Hearthstone*, or

their Commander in the Commander format of *Magic: The Gathering* (but without the limitation to deck-building present in both of these cases).
Players play together over a network on two separate machines. However, the game is designed in such a way that it would be easily replicable in real life with physical cards. To specify, in-game the player can only do with the cards what they can do with a card in real life, for example, change its orientation, flip it over, or put counters on it.
Players also control Units. A unit that is hired by the player can be ordered to attack other units, the opponent's Lord, and also use own ability (should it have one). Within the context of the game, this is the player (The Lord) giving orders, as opposed to playing as the Unit itself.

Story

Lords does not have an in-depth story – the primary focus is on gameplay and having a too in-depth lore can, in my opinion, limit card design. However, as a context for the game, the Players play as Lords, who are the leaders of various neighbouring kingdoms and clans fighting over a barren land they all wish to claim to expand their territory. Each Lord commands an army consisting of hired units, and making use of hired services known as utilities. The victor of a match would be the ruler who claims this barren land.

Look & Feel

During battle, the player's perspective in *Lords* is a fixed-perspective, top-down view of an initially empty battlefield, which is populated with cards as the game progresses. Cards and any counters the game needs are represented by 2D sprites. The player's side of the field is on the bottom half of the field, while the opponent's field is rotated onto the opposite side. Cards are always displayed upright for each player so they can read any cards in play. Cards in the opponent's hand cannot be viewed. For more detail on the orientation of the playing field see the Interface section.

The feel is an extension goal – functionality is more important.

The feel is that of a brutal battlefield – cards are destroyed, burnt, etc. Combat is accompanied by the sounds of clashing swords, and dying units will roar in pain, all accompanied by the sound of war drums. Actions should feel impactful, and so will be accompanied with appropriate visual and sound effects. Card artwork should have a consistent medium, but the theme should differ depending on the kind of card it is. For example, some cards may depict creatures such as frogs, yet others may depict rotting zombies. The latter would be a darker themed card, yet drawn in a similar style, as opposed to the former which may be a bit more colourful.

Tokens

The following is a list of tokens in *Lords:*

- Cards (see *Cards.xlsx* or *Cards.csv* for specific details on each card) (in-game and in deck-builder):
    - Units
    - Utilities
    - Lords
- Gold
- Card Cost
- Card Health
- Card Strength
- Poison Counters
- Decks (in-game and in deck-builder)

- Battlefields
- Hands
- Discard piles
- Players
- Dice
- Card backs (backside of card)
- Debt counters
- Multiplayer score (based on wins and losses)

Rules

The following are a list of the rules that make up *Lords*:
[R1] Each player starts the game with 25 gold.
[R2] When a player starts their turn with 0 gold, they gain 15 gold and one debt counter.
[R3] If a player starts their turn with 0 gold and with a debt counter, they lose the game.
[R4] Each player's deck must be exactly 25 cards, PLUS one Lord
[R5] Each player starts the game with their Lord outside of the deck, to the left of their hand/field.
[R6] Every card has a cost in gold to put it into play. Units also have health and strength.
[R7] When played, Units will stay on the field but Utilities will go to the Discard pile once their effect has been resolved.
[R8] Units are killed either when their health reaches 0 or when killed by a card effect. Killed cards go to the Discard pile unless specified otherwise.
[R9] Card effects overrule game rules
[R10] The turn player can enter combat with each of their Units once per turn
[R11] When entering combat, the attacking player selects an opponent's unit as a target. Both units then lose health equal to the other's strength.
[R12] If the opponent has no units on the field the turn player can "raid" the opponent's supplies instead. If they do, the opponent loses gold equal to half the strength of the attacking unit.
[R13] A unit with a poison counter loses 1 health at the end of each turn per counter.
[R14] Each player gains 5 gold at the beginning of every one of their turns (after checking whether they need a debt counter) EXCEPT on their first turn.
[R15] Each player draws a card at the beginning of their turn.
[R16] The player who goes first cannot attack during their first turn.
[R17] Units can be "buried" through the effects of certain cards (including themselves!). Buried cards are be flipped face-down.
[R18] Buried cards cannot attack, be attacked or use their effects.
[R19] Cards cannot effect buried cards with their effects unless they specify that they affect a buried card(s).
[R20] Some cards can "bounce" themselves or other cards using effects. This means to return the card to its owner's hand.
[R21] At the beginning of each player's turn (after checking for debt) they must pay their unit's wages (1 gold per unit).
[R22] Anything that occurs at the beginning of a player's turn (checking for debt counters, effects, etc) happens *before* that player draws.
[R23] Any card effects that happen at the end of a player's turn happens immediately before the start of the next player's turn. This means the turn player cannot do anything (hire units, use utilities, etc) after these effects go off until the beginning of their turn.
[R24] Should multiple effects trigger simultaneously, the turn player gets priority on their effects. They may choose any ONE effect that has triggered, resolve it, then the opponent may choose any other ONE of their effects that has triggered and resolve it. This continues until there are no more effects to resolve.

[R25] Lord effects must always resolve immediately when triggered. This means they occur before the priority system detailed in **[R24]**.
[R26] The "Beginning" of the turn begins with checking for Debt Counters, THEN effects go off.
[R27] Some effects have "triggers". For example, one card might have its effect trigger when a Unit (possibly itself) is killed.
[R28] Every Lord card has a powerful effect that triggers for FREE (no gold cost), but usually only under certain conditions.
[R29] Wages do not have to be paid for buried units.
[R30] Player's can freely view the contents of either discard pile.
[R31] Neither player can view the contents of the deck in a game unless a card effect requires it.

Features

[F1] Players must weigh the benefits of using all their cards quickly, as overextending and using all their resources may lead them particularly vulnerable, with an empty field, few cards in hand and low gold resources, should all their Unit's be killed. **[R2], [R3], [R6] & [R12]**.
[F2] Players must use their Units efficiently to make sure they're getting as much value out of them as possible. This means knowing when to use their abilities, what to use those abilities on, which of the opponent's Units to attack with each Unit, etc. Essentially, getting the most out of their money. **[R1], [R2], [R6] & [R8]**
[F3] Some cards have effects that could lead into other effects, creating a combination of cards that would greatly benefit the player who used them (in either gold/card advantage or field presence). When deckbuilding players must try to play cards that could combo off together. **[R4]**
[F4] When deckbuilding, players must build their deck so that it has a consistent strategy/win condition that ideally complements their Lord's ability. **[R4] & [R5]**
[F5] Players can tactically kill their own Unit's to benefit from their own, or other cards effects **[R27]**

Gameplay

The players in this game will create decks of their choosing by selecting for a wide range of cards with different costs, stats and effects, and also select a Lord card for each deck they play. As there's limited space in their deck, they'll have to weigh the benefits of putting one card in their deck over another and how many copies to use (more copies increases consistency of the deck but reduces options later in the game). They will then take these decks and play them one on one against another player, also using a deck of their creation.

During matches, players will summon Units and use Utilities during their turn in an attempt to bring their opponent to bankruptcy. Hiring more Units means they can do this quickly as they can establish a board presence and raid their opponent more. However, hiring many Units will consume their gold, increase maintenance/wages, and leave them in a situation where it's harder to recover should they lose their field presence. Depending on the player's deck's strategy, they should try to build a field presence at a pace that gives them the best of both worlds. Also, players should save their Utilities to make them as impactful as possible. Hypothetically, should the opponent have a single Unit on the field and the turn player has a more established board, the turn player could use a Utility card in their hand to kill the last Unit the opponent has, but depending on how many cards in the opponent's hand and how much gold they have, it may be better to save it for a stronger Unit in a later turn, in case the opponent recovers.

60 Second of Gameplay

The following 60 seconds is from the perspective of the player going second after their opponent has already played their turn.

- My turn begins so I draw a card.
- My opponent has 3 cards left in their hand, and 18 gold left after they finished their turn, so it's quite likely they can recover should I kill their Units.
- They have 2 units on the field, I have 25 gold to spend. One of their Units is a Combat Medic (2hp, 2 strength). I have a Plaguespreader (5hp, 2 strength) in my hand, so I'm going to hire that, paying 4 gold, and attack the Combat Medic.
- My Plaguespreader takes 2 damage, leaving it on 5hp, and the combat medic takes 2 damage, leaving it on 0hp.
- My Plaguespreader kills the Combat Medic, so I can use my Plaguespreader's ability, paying 1 coin to bury the Combat Medic on my side of the field. I do this because my Lord card, the Necromancer, will resurrect the Combat Medic at the beginning of my next turn.
- The other card on my opponent's battlefield is a Negotiator. I have two other units I'm considering using in my hand, a Mad Scientist and a Burly Zombie. While attacking with either would kill the Negotiator, my Mad Scientist would die at the end of the turn due to the opponent's Lord's ability. As they're using The Executioner, my Scientist would die due to having 1hp remaining.
- I decide to hire the Burly Zombie, paying 6 gold. As he's a bit bigger than the Scientist in terms of hp, he'll survive the turn. Not to mention that should he be killed he'll bury himself and come back stronger due to his ability.
- I attack my opponent's Negotiator with my Burly Zombie. The Negotiator takes a whopping 5 damage, killing it. My Zombie takes only 1hp, leaving it on 4hp.
- I decide that I'd like to raid my opponent's gold to bring them closer to gaining a debt counter.
- I hire my Negotiator, paying 2 gold, and its ability prevents me from having to pay wages for my Units while he's alive. I decide not to summon the Scientist so that I can resurrect a minion from my discard pile later.
- I attack with my negotiator, and my opponent loses 2 gold as a result, leaving them on 16 gold. I am on 13 gold. I pass the turn to my opponent.
- As all of my cards are above 1hp, his lord effect does not kill any of them at the end of the turn.
- He proceeds to draw, gains 5 gold as it's the beginning of their second turn, and hires a Bloated Body! He uses it to attack my Burly Zombie. His Body takes 5hp of damage, killing it, while my Burly Zombie takes 1hp.
- As the Bloated Body was killed, it's effect triggers, dealing 3hp worth of damage to all my Units. This kills ALL of my Units. As a result, my Burly Zombie is buried, but my other 2 cards go to my Discard pile. My Combat medic remains buried, as you can't kill buried units.
- They then proceed to pay 4 gold to hire their Mad Scientist and pay an additional 2 gold, resurrecting their Negotiator.
- They raid me with both (as my buried Units can't protect me), and I lose 6 gold. They then use a "Concrete Floor" utility to send my buried Burly Zombie to the Discard Pile and end their turn.
- At the beginning of my turn, I draw and gain 5 gold, putting me to 12 gold with 3 cards in my hand. Then, my Lord's ability triggers, resurrecting the Combat Medic I stole from my opponent!

Game continues from here…

Interface

In-match interface

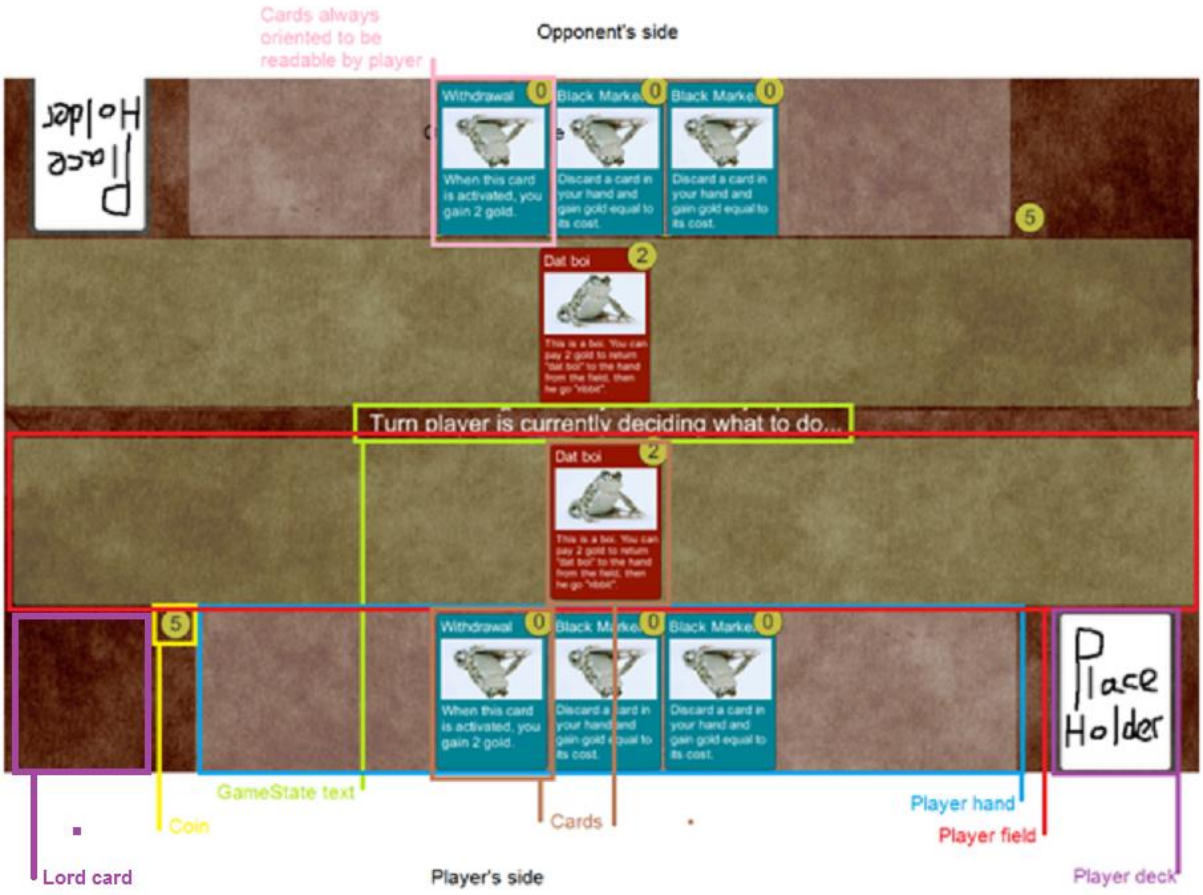

Figure 2 - Diagram of the in-match interface

Player's drag cards from their hand to their field to play them using the mouse. Cards on the field can be ordered to attack or use their effects by hovering the mouse over them, at which point buttons will appear to attack or use effect. If the player needs to target a card for attack or effect, that can be done by clicking the targeted card when required (indicated by the game state text).

Discard pile/ deck searching interface

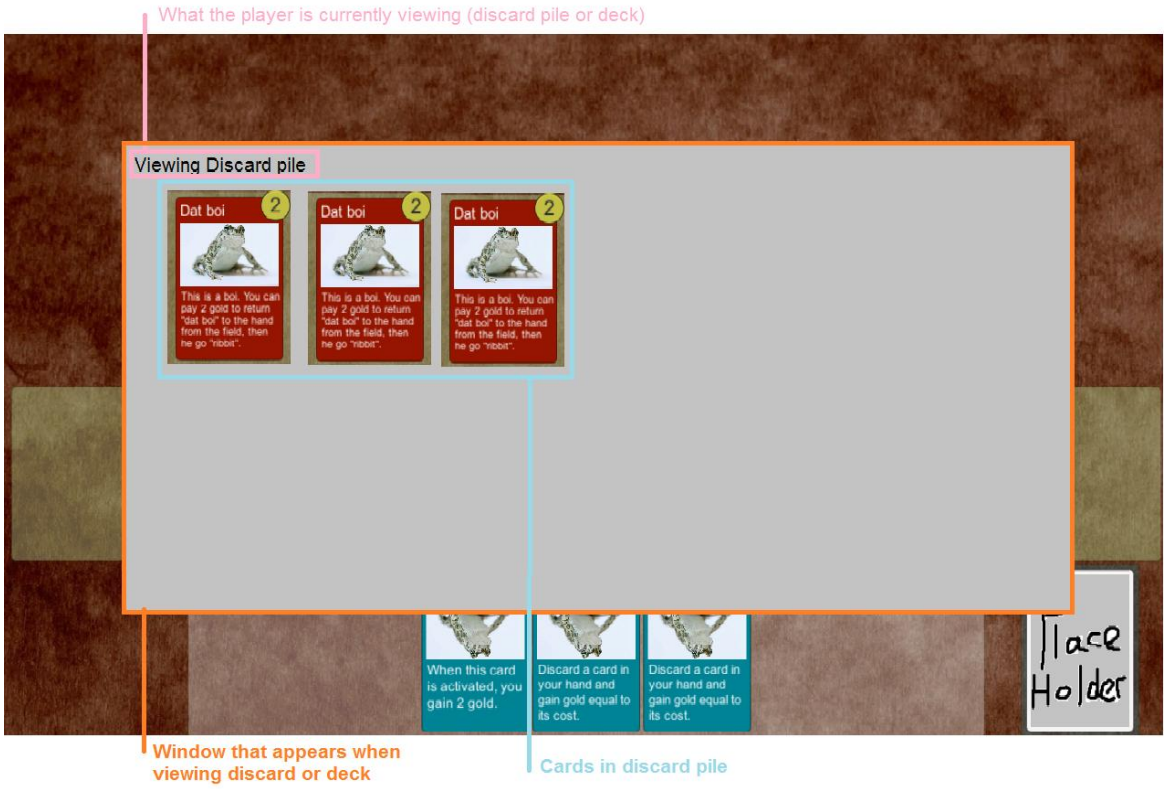

Figure 3 - Diagram of the searching interface

When viewing the discard pile or deck a window like the above appears. If viewing for an effect the player may need to select a card in the discard or deck. In this case, it can be selected by clicking on it.

Card diagram

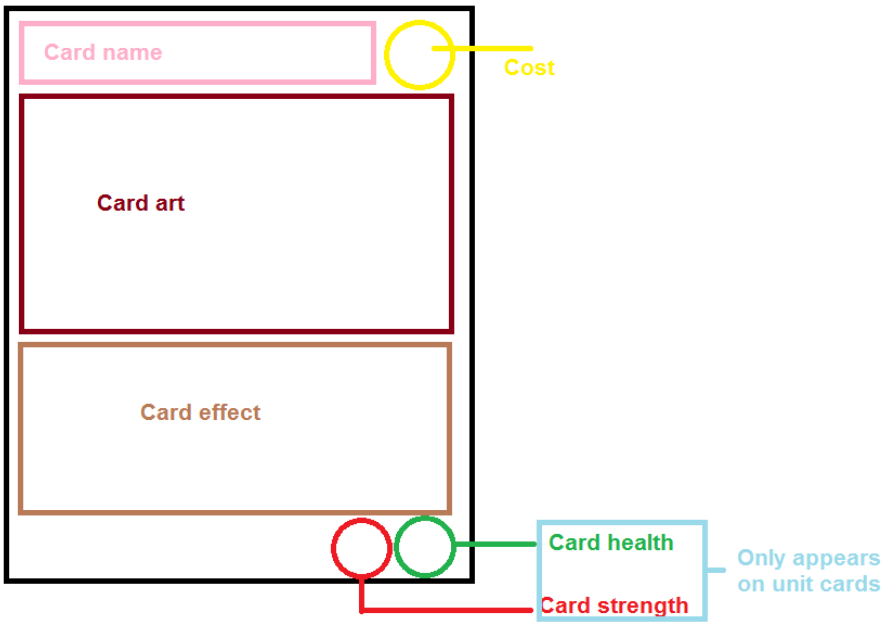

Figure 4 - Diagram of the card layout

The above shows the layout of a card in *Lords,* showing where each component of a card is located

Main Menu

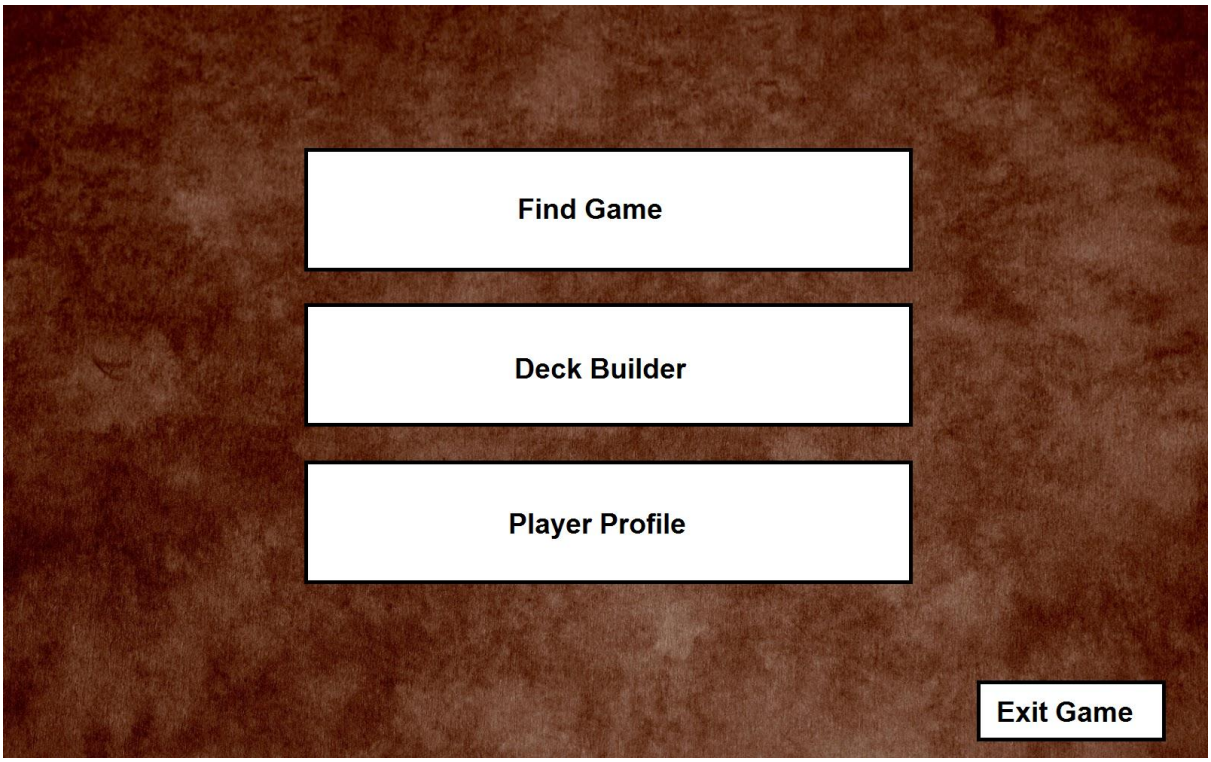

Figure 5 - Diagram of the Main Menu

The Main Menu would have a layout similar to the above. Clicking the "Find Game" button will bring up a list of people in lobbies who are looking for someone to play with. On this page will also be a "Find Match" button which will auto-match you with an available player with as close a Multiplayer score to you as possible.
The Deck Builder page will bring you into a page with a button for each of the decks you've built. Clicking one of those buttons will open up the Deck Editor for that deck, detailed in the "Deck Editor" section.
The player profile will open up your player profile page, which will contain stats such as total wins, losses, multiplayer score, and potentially stats such as Favorite Leader/Unit/Utility based on usage.
Exit game would exit the game.

Deck Editor

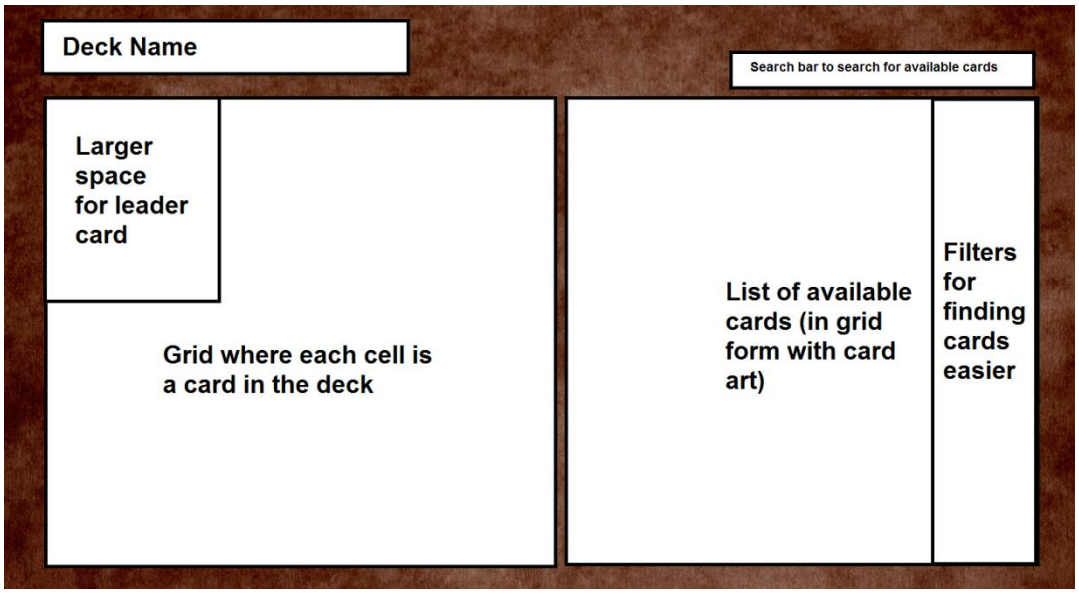

Figure 6 - Diagram of the in-game deck editor

Objectives

The objective of the game for both players is to drive their opponent to bankruptcy through raiding their gold stock, while also preventing the opponent from doing the same. Players achieve this through proper use of their cards, prioritising threats, maintaining card advantage, correct gold management and clever deck-building.

A player loses if they begin two turns with 0 gold. To keep track of how many turns they've begun in this state, they gain a debt counter at the beginning of their turn if they have no gold. This means a player can win when their opponent has 1 debt counter, and 0 gold at the beginning of their turn.

3. The Final Product

The following section gives an overview of the Final Product. It also provides a description of technical achievements, testing, and technical documentation that has been produced for this project.

3.1. Product Overview

Lords is a multiplayer deck-building game created in Unity that draws inspiration from games such as *Hearthstone, Yu-Gi-Oh!* and *Magic: The Gathering.* It allows players to create decks from a selection of 26 cards and take these decks to play in 1-on-1 matches online against other players from different locations. A C# server currently hosted on my PC matches players against each other on a first-come-first-served basis, and maintains the game state of each match.

Cards are divided into 3 types: Units, Utilities, and Lords. In-game, players take it in turns to hire units, use utilities, use their units' abilities and declare attacks. Each card has a unique ability, which can range from dealing damage to units, drawing cards, searching for cards from the deck, and more. Some abilities are triggered manually by the player, yet others will trigger automatically when certain events happen in-game (for example, when a card returns to the hand from the field). All of these actions are validated and processed by the server. A more detailed guide of the rules can be found in-game in the "How To Play" section of the main menu. This guide is also available while the player is in a match by clicking the help button, labelled with a question mark.

3.2. Technical Achievements

Before listing the project's achievements it's important to list what was built upon, modified or adopted.

First and foremost, the client-side application was built using the Unity Engine, which provides a functionality for a UI, scenes - which are a means of separating content into a room-like organization, for example, each menu is a scene - rendering, and other tools which are helpful for building a game. This reduced the amount of work I had to do significantly, as I did not have to implement code for a game loop, rendering, etc.

Secondly, the initial code of the in-game interface was created using the Quill18Creates' tutorial series, *Unity Tutorial - Drag & Drop Tutorial*[4]. This assisted in the creation of the initial code that allows cards to be dragged around the screen by the player.

Finally, the inspiration for several game rules and features was taken from other card games and from the expertise of these game's designers. In particular, two games that helped shape the design of the game are *Pokémon Trading Card Game* and *Hearthstone*. The insight shared in the talks given by Mark Rosewater[1] and Dylan Mayo[2], designers of *Magic: The Gathering* and *Pokémon Trading Card Game* respectively, was drawn upon for the design of the game.

In regards to the technical achievements, the remaining code was written from scratch. Aside from the Unity Engine, no external sources were used for code. In particular, I'm quite proud of the robustness of the server. While the server is not constantly running, when it is running, it handles disconnections gracefully. Players will rarely find themselves in a situation where they're locked out of the game due to connection issues - the server handles all of that for them. Also, if the server closes or the player loses connection to the server, the client will handle the situation gracefully and allow the player to return to the main menu.

I'm also happy with how the implementation of the game allows for the easy addition of new cards. The only changes on the client-side application that would have to be made to introduce a new card is the creation of the card's prefab (which could be automated in the future), and the card's deck builder prefab. No additional client-side code is required for new cards unless an entirely new feature is being added. Server-side, I would have to create code to handle the card's specific ability, map its ability to its name, and add an entry of its values to *CardData.csv*. This process doesn't take too long, unless entirely new features are being added. This allows for fast expansion of the card pool and allows for card concepts to be playtested quickly and easily.

I take pride in the design of the game as I believe the rules I've created bring strategic depth to the gameplay of *Lords*. Once *Lords* has been expanded upon and polished, I think it has the potential to be an enjoyable game.

3.3. Testing

The testing for this project was conducted through playtesting and unit testing. The following subsections provide more details on each type of testing.

3.3.1 Playtesting

A portion of testing for *Lords* was executed by playing the game with an end-user who would provide feedback based on their experience. Unfortunately, this method of testing was not conducted as much as I would have liked, as the game reached a playable state too late to be able to take full advantage of playtesting. The feedback I did manage to receive was put into a file called "feedback.txt" on the project's GitLab repository. Some feedback was worked upon and corrected, and where that is applicable the feedback has been removed from the file.

3.3.2. Unit Tests (Functional Testing)

Much of the functional testing for Lords was done through C#'s Unit Tests. I wrote a test suite on the server that would test the functionality of the game. Most of the major functionality and logic of the game has an associated set of tests. Each card has a set of tests allocated to them as well, each in a file called "<card_name>Tests.cs". The number of tests each card has is proportional to the complexity of it. Some niche bugs I found during playtesting also have tests associated with them to ensure that they don't arise in the future. After a change is made to the server-code, these tests are re-run to ensure that the project has retained its functionality. Any future functionality will also have a set of associated test suites.

Figure 7 shows an example of the results after a test run. The green bar on the top of the image indicates the ratio of passed tests (green) to failed tests (red).

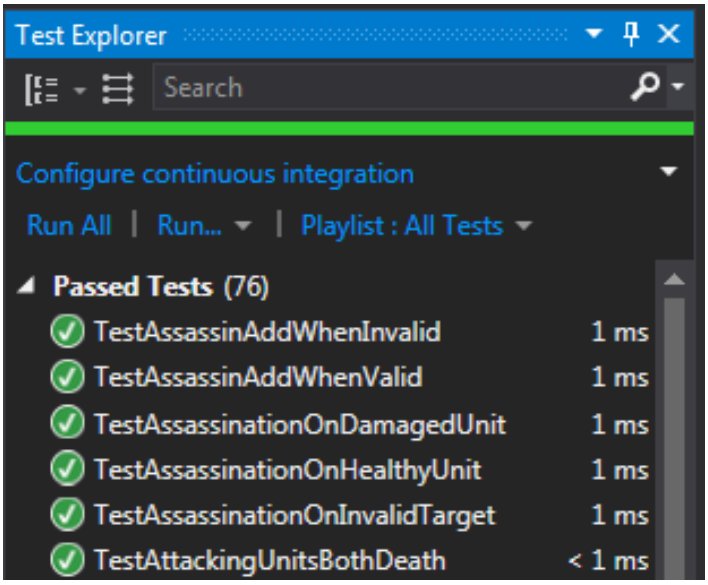

Figure 7 - An example of the test suite after it's been run

3.4. Technical Documentation

The below technical documentation can also be found in the README.md file on Gitlab.

Running the Game

To run the game executable, simply download the Executable/Lords folder from the repository. From here you can run the Lords.exe to run the most recent build of the game.

Requirements for running the game

System Requirements

The following system requirements are according to the recommendations on the Unity Website here

- Windows 7 SP1+.
- Graphics card with DX10 (shader model 4.0) capabilities.
- CPU: SSE2 instruction set support.

Opening the Game in the Unity Editor

To open the game in the Unity Editor, you must have Unity installed. Download this repository, then when you open Unity, select "open project", and select the "ce301_thorn_t" folder.

Requirements for opening the game in the Unity Editor

The Unity version used to create this game is 2018.4.10f1. It is therefore recommended that you use the same version to open the game yourself. You can download this version of the Unity Engine here.

System Requirements

The system requirements for opening the game in the Unity Editor are identical to those for running the game executable.

Running the Server Locally

By default, Lords will attempt to connect to a server hosted on my home computer. In the future this will be a different machine constantly running the server. To host a server locally on your machine follow these steps:

1. Download the Server_Executable folder in the Git repository.
2. In this folder, open Config.settings.
3. Change the value of test_local_server to true.
4. Run the server by double-clicking on Server.exe.

Note that you may need to enable Server.exe on your anti-virus/firewall. Also, this has only been tested to work on Windows.
To then connect to this server, open your instance of Lords and follow these steps:

1. Click "Enter Game".
2. Press the 'L' key to toggle between local-server on and off (when it is on some text will appear to indicate as such).
3. Choose a deck and click "Enter Game".

Note that this only supports instances of Lords running on the same machine as the server. To start an actual game, you'll need to start a second instance of Lords and repeat these steps.

Running the Server Unit Tests

To run the server Unit Tests follow these steps:

1. Import the server project, found in the /Server directory, to Visual Studio.
2. On the toolbar at the top of the page, click Test -> Run -> Run All Tests
3. The "Test Explorer" window should appear on the screen, presenting the results of the tests including which tests passed and which tests failed.

4. Project Planning

4.1. Project Management Software

Project planning throughout the year was done primarily using the university's Jira server. On Jira, issues represent work items for the project. These issues are separated into different categories dependant on the type of work that needed to be done, and these categories were used to help organize work items. The main categories I used and the way I used them, are as follows:

- Stories - A ticket that introduces new functionality for the product.
- Bugs - An issue used for highlighting unintended behaviour on the product.
- Task - While traditionally used for minor work items, I used tasks as a means to represent "admin" tasks. This included work that did not directly impact the product but still had to be completed, for example, a literature review.
- Epics - A large body of work, usually representing features, that is separated into multiple stories and tasks. For example, creating the in-game deck-builder.

The University's Git server was used as a source control software. When a work item was completed, its associated Git commit was linked in the work item's comments as a means to connect the work item to its associated changeset.

In the summer preceding the start of the Autumn term, I was using a repository on Github for source control, and a browser extension called Zenhub for the creation and management of work items. I decided to use Zenhub during the summer term because I had experience with it, using it for the majority of my placement year at IBM. As Zenhub is very similar to Jira, the transition to Jira was not difficult.

4.2. Project Management Methodology

The objectives of the project were set early in the academic year as they were defined in early October. I produced a game design document soon after, which defined the specifics of how the end product would work, at least in terms of rules and gameplay.

Throughout the course of the project, an Agile methodology was adopted. This is my preferred methodology, as I have used it in a professional environment, giving me prior experience. I feel it gave me more flexibility with the project, as I could decide which work items to address at the beginning of the week. I chose work items based on how much time I could allocate to the project and based on the progress from the previous week. Furthermore, I incorporated the teachings from the lectures and labs from my other modules into this project, especially "High-Level Games Development", as it was relevant to my project. For these reasons, I believe an Agile approach was the correct choice for this project.

4.3. Maintaining Momentum, Adapting To Change and Recognizing Risk

Maintaining momentum throughout the project was a challenge, and I believe that my momentum was inconsistent, partially due to factors out of my control that I should have been prepared to deal with. I either underestimated the impact that these factors would have on my work or did not have the foresight to do adequate research to prepare for them. I will address the two which I believe had the largest impact on the project.

4.3.1. Workload in the Autumn Term

When I began the project, I planned to have a completed single-player prototype by the end of the Autumn term. While I was initially on the right track to achieve this, I had completely underestimated the amount of coursework I would have to complete in Autumn. This was due to 80% of my other modules being in the Autumn Term. This kept me occupied with assignments other than this project throughout the latter half of the Autumn Term, resulting in a couple of weeks where progress was nearly at a standstill. I acknowledge that the responsibility lies on me, and I should have been better prepared to overcome this, and adjusted my initial schedule accordingly. As a result, my project was behind schedule by the end of December. Even if this allowed me to spend more time on the project in the Spring Term, I had less time to collect feedback on the game in its playable state. This may have affected the quality of the end product, at least in terms of enjoyment and game-balance.

In hindsight, I should have expected the workload and spent more time in the Summer of 2019 working on a single-player prototype to offset this. Doing so would have hopefully allowed for more progress to be made by the Christmas break, and I could have spent more time in the Spring term play-testing the game with other people and producing a higher quality end-product.

4.3.2. UNet

Another factor that impacted the momentum of the project was the use of UNet, which is Unity's inbuilt multiplayer framework. My attempts to use UNet cost me at least a week of January, as well as a large portion of time in the 2019 summer too. While I don't believe trying to incorporate UNet was initially a bad idea, I do believe the amount of time I invested into it was a mistake. I finally concluded early in the spring term, that UNet would not be appropriate for this project after investing multiple weeks between the 2019 summer break and 2020 spring term and obtaining no results. I eventually decided to create my server from scratch in C#, instead of using UNet, which took approximately 2 weeks to write.

The problems I faced with UNet which helped me make this decision was a lack of documentation on its uses, and its deprecation going forward. Specifically, UNet is not appropriate for multiplayer games that rely heavily on a UI to function, such as a card game. I reached this conclusion after weeks of attempting to synchronize in-game events, such as drawing a card, in a multiplayer game. The lack of documentation regarding the use of UNet with UI elements, such as canvases, led me to spend too much time making too little progress.

Another problem with UNet is that it will become deprecated over the next few years in favour of a new multiplayer framework. Changes in this fashion would lead to additional work for me in the future if I choose to maintain this project, and could hurt the longevity of the game. Ultimately, this influenced me to write a server from scratch, which I have found much easier to implement and has given me more control over the end-product.

If I had done more research on the creation of multiplayer games in Unity, I may have realized earlier that I had other options that suited my project better. To avoid a similar problem in the future, I will do more research on all aspects of a project before I start it. Most of the research I conducted for the project was on the design of card games and I do not believe I carried out enough research on their implementation.

5. Conclusions

This section presents the progress made towards the main objectives and the game design document. It also describes future work to be undertaken on each of these aspects to produce a higher quality product.

5.1. The Deck-Builder and Card Pool

This subsection covers the deck-builder and the final card pool. This objective was written as:

- Create an in-game deck builder that allows the player to create their own decks from a pool of at least 100 cards.

The final version of the game does include a functional deck-builder. The player can create as many decks as they like out of the available pool of cards, and the deck-builder allows the user to search for cards in the available card pool by name, card type and card cost. However, the card pool is very shy of 100 cards, currently sitting at 26 cards. Throughout the development of *Lords* before submission, I prioritised other aspects of the game such as reliable online functionality, which I considered more important than creating a diverse card pool.

In the future, the card pool could easily be expanded to include more cards. Future cards could include features which were listed on the initial Game Design Document but didn't make it into the final submission. The final deck-builder does not include functionality to allow the user to modify or delete existing decks but this could be added in the future.

5.2. Online Play

This subsection covers online multiplayer. This objective was written as:

- Create online functionality, allowing players to play with each other from other locations.

Players can play online with each other when the server is running. Currently, the server does not run at all times due to limited computational resources. In the future, I would either move this to a spare computer I own or invest in a hosting service. This would prove essential if *Lords* was ever released to a wider audience.

The server currently runs on a first-come-first-serve basis. At the moment, the user has no control over who they are playing against, but the user should have the option to host private games where they can play with their friends. Also, a ranked system could be implemented to calculate a player's skill level (for example, based on a win-loss ratio) and then pair players against others with a similar skill level. These features could be implemented to improve the multiplayer experience.

5.3. Player Feedback

This subsection covers the following objective:

- Introduce TCG players to *Lords* to playtest and provide feedback on the game. This feedback will then be used to polish the game and improve the end result.

While I was able to obtain feedback from a few players, it wasn't to the scale I was hoping for, and as a result, I don't have much feedback to make improvements on. This was due to the factors detailed

in section 4.3. In the future, I would like to receive feedback from a wider variety of players with different levels of experience with TCGs which I would draw from to improve *Lords*.

5.4. The Game Design Document

This subsection covers how the final product compares to the Game Design Document written in October 2019.

The rules section of the Game Design Document lists functionality that I would have liked the final product to include. I decided to disregard or change some of these rules due to feedback received. For example, rule 21 was changed so that wages are paid at the end of the turn instead of the beginning. Some mechanics detailed in the rules which were not implemented but I would have liked to, are the poison counters (Rule 13) and buried cards (Rules 17-19 and Rule 29). These rules can be easily implemented when the card pool is expanded, as new cards will pave the way for new game mechanics.

Certain other sections of the Game Design Document are outdated, due to changes in the game design, or were a low priority (such as Look & Feel and Story), and as such, these are very much aspects of the game I would like to work on in the future to improve the quality of *Lords*. Creating artwork for each card, as well as providing animation and sound effects will not only improve the aesthetic of the game but help illustrate the game state to the player.

References

1. GDC (2016) *Magic: The Gathering: Twenty Years, Twenty Lessons Learned.* [online video] Available at: https://www.gdcvault.com/play/1023186/Twenty-Years-Twenty (Accessed: 12/Aug/19).

2. GDC (2018) *Board Game Design Day: Balancing Mechanics for Your Card Game's Unique "Power Curve".* [online video] Available at: https://www.gdcvault.com/play/1024913/Board-Game-Design-Day-Balancing (Accessed: 23/Aug/19).

3. Miller, John Jackson (2003), *Scrye Collectible Card Game Checklist & Price Guide, Second Edition*, pp. 667–671.

4. *Yu-Gi-Oh! Card Sales Set New World Record* (2009), Konami.jp. August 7 2019, https://web.archive.org/web/20090810144853/www.konami.jp/topics/2009/0807/index-e.html

5. *Hearthstone,* Google Play Store https://play.google.com/store/apps/details?id=com.blizzard.wtcg.hearthstone&hl=en_GB

6. Quill18Creates (2014) *Unity 3d: Multiplayer First-Person Shooter.* [online video playlist] Available at: https://www.youtube.com/playlist?list=PLbghT7MmckI7BDIGqNl_TgizCpJiXy0n9 (Accessed: 14/Sep/19).

7. Quill18Creates (2015) *Unity Tutorial - Drag & Drop Tutorial.* [online video playlist] Available at: https://www.youtube.com/playlist?list=PLbghT7MmckI42Gkp2cILkO2nRxK2M4NLo (Accessed: 03/July/19)