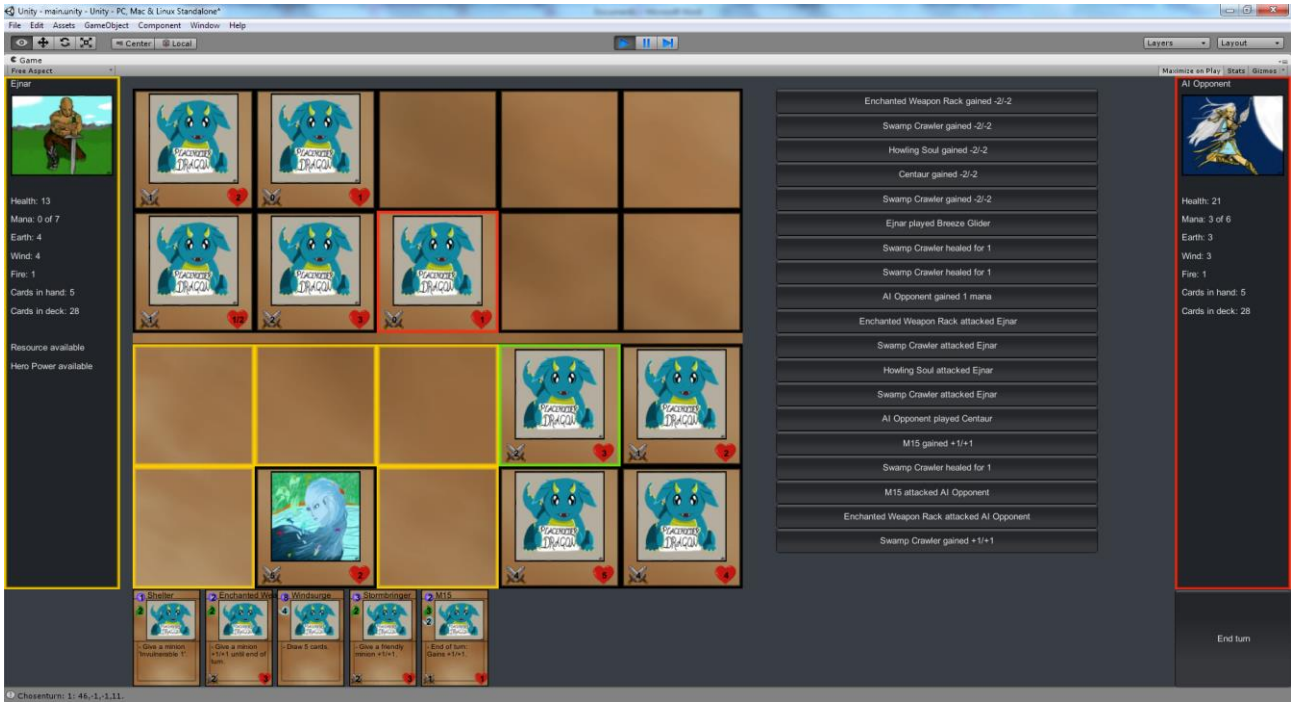
FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

Master's thesis / PhD thesis

Ejnar Håkonsen

System- and AI design of a digital collectible card game



Academic advisor: Jakob Grue Simonsen

Submitted: 06/02/15

Institutnavn:	Datalogisk Institut
Name of department:	Department of Computer Science
Author:	Ejnar Håkonsen
Title:	System- and AI design of a digital collectible card game
Subject description:	Covers the creation of the digital collectible card game ‘Elemental Command’, including game design, implementation and design of an AI opponent.
Academic advisor:	Jakob Grue Simonsen
Submitted:	6. februar 2015
Grade:	

ABSTRACT

This thesis covers the creation of the digital collectible card game Elemental Command.

The game's systems and mechanics are designed based on a set of design objectives that are derived from analysis of academic as well as industry sources. Mainstream games of the genre are analyzed to determine how well their individual design choices support these objectives, and successful solutions are expanded on to arrive at a final design that, while unique, prioritizes quality over innovation. A particular novelty of this design is in the additional options it affords the players, by expanding on the role of creature/minion positioning.

The game is implemented in C# Unity, using ~2500 lines of code. Data formats are developed for defining game content, such that new cards and abilities (including variations in targeting parameters) can be defined entirely in data, as long as the underlying effect is already implemented. 202 cards and their associated abilities are defined using 1565 lines of raw text.

An AI opponent for the game is developed, based on an adversarial search strategy and using an additional ~700 lines of C# code. A number of heuristics are developed to enable competent play and achieve a level of efficiency where 12 turns of lookahead are deemed feasible on a medium tier laptop computer without noticeable wait.

CONTENTS

1. INTRODUCTION 4

1.1 OBJECTIVES AND MOTIVATION 4

1.2 READING GUIDE 5

2 ANATOMY OF A COLLECTIBLE CARD GAME (CCG) 6

2.1 THE BRIEF SUMMARY 6

2.2 MAGIC: THE GATHERING SPECIFICS 8

2.3 HEARTHSTONE SPECIFICS 10

2.4 DUEL OF CHAMPIONS SPECIFICS 12

2.5 ADDITIONAL TERMS AND CONCEPTS 14

3 DESIGN 17

3.1 DESIGN OBJECTIVES 17

3.2 CORE DECISIONS 23

3.3 NAMING THE GAME 39

3.4 GAME RULES 40

3.5 POSSIBLE IMPROVEMENTS 42

4 IMPLEMENTATION 43

4.1 STRUCTURE 43

4.2 IMPLEMENTATION FACTS 44

4.3 IMPLEMENTATION DISCUSSION 49

5 GUI DESIGN 52

5.1 PROVIDING FEEDBACK 52

5.2 ARTWORK 54

5.3 LIMITATIONS 54
6 AI DESIGN 55
6.1 COMMON STRATEGIES 55
6.2 EXISTING AIS 58
6.3 THE AI 60
7 PROCESS 73
8 ACKNOWLEDGEMENTS 75
9 FUTURE WORK 76
9.1 DESIGN 76
9.2 AI 77
10 BIBLIOGRAPHY: 78
10.1 ACADEMIC: 78
10.2 OTHER SOURCES: 80

1. INTRODUCTION

1.1 Objectives and motivation

Trading card games have had a powerful impact since the release of Magic: The Gathering in 1993. A billion cards were sold within the first 18 months and by 1999 the lead designer alone earned more than a hundred million dollars for the sale of the company [0]. Since then the popularity has only grown further, with a Las Vegas tournament in 2013 being watched by more than 140,000 streamers [0] and the MTG Pro Tour alone being responsible for $250,000 of annual prize money [1].

A recent innovation of this genre is the advent of digital collectible card games that use the same compelling game systems in a digital medium. This offers many advantages including offering players a continual availability of opponents, removing all the production costs of physical components and making the game open to patching and rebalancing. Several of these games have already proven immensely successful, including Duel of Champions (released 2012, 1.5 million users [2] within a year before additionally becoming available on Steam), Hearthstone (release date 2014, more than 1 million users before even going into open beta [3], months spent in top 5 of the most streamed games in the world on twitch.tv before game was officially released) and Hex (in 2013 raised $2,300,000 on Kickstarter [4] – an overfunding of 760% in pre-purchases alone) . Video game players have long since proven willing to spend immense amounts on virtual goods, with the numbers from 2010 indicating a worldwide revenue of $7.3 billion and a massive projected increase over the coming years [5]. All of this points towards this genre of products being in an outstanding position to reap the benefit of this growing market as well as having a defining impact on both game streaming and the eSports scene.

The ability to design the complex systems of such games holds obvious value, but it is notable that both Hearthstone and Duel of Champions only offer limited AI opposition with fights that are partly pre-scripted and mainly intended for tutorial purposes. While the emphasis of these games is clearly on battling other players, an effective AI can be valuable for a number of purposes including:

- Widening the possibility of single player games or game modes in the genre.
- Easing the teaching process for new players.
- Providing more consistent and fully configurable training opponents for experienced players.
- The ability to rapidly simulate the outcome of a great number of games could be a valuable in establishing and maintaining the game balance, which is a crucial and challenging aspect of designing these products.

- Likewise, the option of rapid simulation could be valuable for players in simulating how a particular deck design performs.
- Considering the prizes involved in playing these games, a sufficiently skilled AI could be turned into a “bot” and automate both play and income for a player.

1.2 Reading guide

This thesis assumes familiarity with the computer science concepts covered in the ACM core curriculum [6]. Additionally the reader is assumed to have some familiarity with games and their online communities, though not CCGs in particular.

The contents are as follows:

Section 2 describes the genre of collection card games and establishes basic vocabulary for those without this prior knowledge.

Sections 3-6 detail the development of the game: Game design, implementation, GUI design and AI design respectively.

Section 7 covers the work process of the thesis, briefly describing how actual development corresponded to the initial schedule and describing the factors that were not accounted for at the outset.

Section 8 acknowledges those whose help has aided me in this process.

Section 9 considers the applications and wider perspectives of these products.

2 Anatomy of a Collectible Card Game (CCG)

In designing my CCG, I will be building on previous experience in the genre, with my primary sources of inspiration being *Magic: The Gathering* (MTG), *Hearthstone* (HS) and *Duel of Champions* (DoC). This section is intended as a brief summary of how such games *typically* function, as well as some of the design choices that distinguish the individual games.

All talk of CCGs in this section is focused on providing an understanding of the most common form of a CCG, rather than attempting to encompass every phenomenon of the genre. It should be noted that MTG is the physical game that founded the best known form of the genre in 1993. DoC and HS are both digital CCGs that released in 2013 and 2014 respectively.

2.1 The brief summary

A number of players (typically 2) each build a deck (a set of cards) from the pool of all cards they have access to. Each game has a set of deck-building constraints (e.g. MTG's "at least 60 cards total, no more than 4 of a kind").

At the start of the game each player draws a number of cards (his "starting hand") from his deck. Many CCGs include a mechanic that allows each player to shuffle the cards back into the deck and draw a new starting hand (performing a "Mulligan") within certain constraints (e.g. "you can only do so once" or "you get one card less each time"), to reduce the luck factor of the starting hand.

Screenshot from the CCG 'Hearthstone' by Blizzard Entertainment

The game then progresses in turn-based manner: In a turn, a player will typically draw a card and then play a number of cards (subject to whatever constraints the particular game has on playing cards) and use any cards that stay in play after being played (e.g. if the game has minions or creatures that can attack and defend rather than being resolved instantly).

CCGs typically operate with a resource cost for playing cards. In this case, the resource will generally become more plentiful during the course of the game, ensuring that the strongest cards cannot be played until late in the game and that decks need good cards for most stages of the game.

In addition to a resource cost, CCG cards typically have a type as well as statistics relating to that particular type of card. Common examples of types are "spells" which are resolved with an immediate effect (e.g. "draw two cards" or "deal 3 damage to target") and "minions" (also known as "creatures") which stay in play in order to attack and defend.

In CCGs which use the minion type, minions typically have at least the two attributes "attack" and "health". In this case, when a minion attacks another minion, they both deal their attack damage to the other minion's health value and any minion reduced to 0 or less health dies. Common constraints for minion type cards are that they can attack once per turn and not in the turn they were played.

CCGs have an objective in order to win the game. A common form of this is that each player has a health value that can be decreased by attacking the player with e.g. spells or minions, and that the game is won when all opponents are reduced to 0 or less health. An alternate and often supplementary objective is that many games let an opponent be defeated when he has to draw cards after his deck is already empty.

Many CCGs (or their communities) describe the persistent state of the game as the "game board". This typically includes all cards that are effectively on the table but not those in the player's hands. This is mainly important for keeping track of all cards that stay in play (e.g. "minions"). This can be as simple as keeping track of all the cards that are in play, who their owners are and whether their attributes have changed in any way (e.g. they have taken damage or a spell made them stronger). Some games use more complex solutions, such as also making the relative position minions matter (e.g. minions

that make adjacent friendly minions stronger) and/or keeping each player's discard pile on the board (a "graveyard") in order to let spells and effects interact with cards that have already died or been used/discarded.

CCGs typically operate with the rule that anything written on a card takes precedence over what is written in the general rules – e.g. a minion card can have the ability "May attack the same turn it is played" even if the general rules state that minions cannot attack the turn they are played. Additionally, many CCGs are designed with expansions of the original card pool in mind, in order to keep the game fresh while selling more cards. As a result, a general rule of thumb is that the older a CCG is, the greater its card pool and the more exceptions exist to the general rules.

Appendix 2.5 explains a lot of terms and concepts that can be relevant when discussing CCGs. It is optional reading, aimed at those less familiar with discussing the genre, but may be relevant for understanding some of the concepts used in the following sections.

2.2 Magic: The Gathering specifics

MTG was published by Wizards of the Coast in 1993 and is the classic game that the others build on. Compared to the others, it is slightly more complex, has longer games, and the fact that it has to work as a physical game sometimes shows in the mechanic.

Magic: The Gathering - Duel of the Planeswalkers 2015

2.2.1 Combat and board

Placement on the board has no relevance.

When minions are sent to attack, the defender chooses which minions (if any) block where. Each defending minion can only block one enemy, but multiple can be sent against the same. A minion which attacks (or uses certain abilities that likewise "tap" it) is unable to block on the following turn.

Additionally, the current life of each minion is only relevant within a single turn – at the end of each turn they are restored to full health. This all means that the defender has a significant advantage in MTG and contributes a lot to games being longer.

Certain abilities can make a minion harder or even impossible to block. And while a 1/1 minion could normally be sacrificed to block the attack of a 9/9 minion completely, the "trample" ability also aids offense by making sure all excess damage still hits the attacked player.

2.2.2 Resources

Decks in MTG constructed have at least 60 cards with no more than 4 of a kind (except basic land cards). Players start with 20 health and a hand of 7 cards. To balance first-player advantage, the first player does not get to draw an additional card on his first turn. Mulligans work by discarding the entire hand and drawing a new one, with one card less – this can be done until the player is out of cards if desired.

Rather than having automatic resource gain, MTG uses cards of the "land" type: These can be played for free at a rate of no more than 1 per turn and then provide 1 mana of a specific color every turn from then on. This adds some additional complexity to the deck building and the fact that magic has 5 colors of cards (all with various strengths) that are dependent on that particular color of mana also adds some fun balance for those who want to run multiple colors in the same deck. Other consequences are that the progression of mana availability over time is a lot less predictable, and that luck becomes a greater factor (particularly in the starting hands) since a player can end up without useful actions by drawing either way too few or way too many lands. It also means that about 40% of the cards drawn are completely basic cards, there to facilitate other cards.

2.2.3 Card types

MTG has 6 card types:

Minions (called "creatures) and lands have already been mentioned.

Spells are broken up into "sorceries" and "instants". Sorceries can only be played during your own turn when you are not in the process of attacking. Instants can be played as an interruption to anything (e.g. minion targets are selected, they are about to damage each other, you interrupt and play a couple of instants to buff your minions or kill the enemy minions that were trying to kill them). An important aspect of this is that your opponent might play a "counterspell" which blocks the card you were trying to play and immediately discards it instead. These instants add a lot of both depth and complexity to the game, since your opponent can interrupt your plays during your own turn. A side-effect of this is that each play needs a lot more consideration, and that the game duration thus increases.

"Enchantments" are spells that stay in play as powerful permanent effects, unless the opponent has a card that can remove them.

"Artifacts" are a lot like enchantments but serve a variety of purposes. Some stay in play with a permanent effect, others can be attached to minions for various bonuses or use abilities for mana. Additionally, both minions and land can have the artifact type as well, making it susceptible to both the positive and negative effects that target artifacts in particular.

Any card that stays in play (anything but instants and sorceries) is termed a "permanent". In MTG, permanents can have activated abilities – that is, abilities (possibly with a cost) that I can use at any time as long as I have the card with the listed ability in play. An example could be paying one black mana and sacrificing a minion to have target player discard a card. Unless otherwise noted, these abilities can be used anytime an instant could be cast.

2.2.4 Misc.

Based on personal experience, I would estimate playtime to be about 15 minutes on average and 30 minutes in extreme cases.

Phases

Unlike HS and DoC where you can take your actions in any order within a turn, MTG turns are formally segmented in 5 phases:

Upkeep phase: All start of turn effects happen and all your cards untap.

Main phase (1): Where you can play lands, minions, artifacts, enchantments and sorceries.

Combat phase: First combat is declared (in case the opponent wants to cast an instant, knowing that the main phase has ended), then attackers are chosen, the defender chooses which minions block which attacks and finally all attack damage is resolved simultaneously.

Main phase (2): Another main phase, to play non-instants after the combat is resolved.

End of turn: When end of turn is declared, the opponent gets a chance to play things before his own turn starts (and all his lands and cards untap anyway), and finally any end of turn effects are resolved.

Noteworthy oddities

Being a physical game, "tapping" cards (showing that they have used their turn) is done by turning them sideways – and turning them upright to indicate when they are "untapped" at the start of a new turn. Likewise, MTG-sources making mention of "counters" often refer to the physical tokens the game uses to keep track of things.

Since MTG has released more than 13,000 different cards [7] and CCG card abilities inherently take priority over the core rules, it is also by far the game where most exceptions can be found to what I have described here.

The great number of cards also means that a higher number of poorly balanced cards have slipped through over time (particularly in the early sets), and that only a small percentage of new cards are going to be exciting in terms of decks built from the pool of the greatest cards ever released. To handle this, a number of different tournament formats are maintained, some of which only allow cards from the most recent sets, whereas others allow all cards ever released, except a short list of cards that are banned for being too powerful even in that context [8].

2.3 Hearthstone specifics

Hearthstone was released by Blizzard Entertainment in 2014 as a purely digital game.

2.3.1 Combat and board

Hearthstone has a board that allows for at most 7 minions at a time from each player. Each player has their minions on different sides of the board, but the relative placement of your minions matters in the sense that certain effects will affect adjacent minions.

Minions can attack any target including the enemy player (called "hero"). Minions with the "taunt"-ability have to be attacked before any other targets. This makes for a faster game that encourages offense and initiative, but also allows players defensive options to protect their hero or vulnerable minions.

Both attacker and defender deal their damage to each other and minions are not restored to full health between turns. The heroes (player avatars) are also able to get an attack value through spells, abilities or weapons. In that case they fight as minions would when attacking, but still deal no damage when attacked.

Hearthstone

2.3.2 Resources

Decks in HS constructed have 30 cards with no more than 2 of a kind (or just 1 for those of the highest rarity). Players start with 30 health. The starting player draws 3 cards and the second player gets 4 cards plus a special card that is a free spell which gives 1 extra mana on the turn it is played. Mulligans are handled by letting players choose which cards in their starting hand to keep and getting one re-draw of all the others.

The resource system is the simplest of the 3 games: Players start with a "mana crystal" and gain one more every turn. Each turn, the players can spend as much mana as they have mana crystals. This effectively gives a smooth and predictable mana curve with 1 mana on turn 1, 2 mana on turn 2 etc.

Unlike MTG's 5 colors that can be combined as desired, HS has 9 hero classes to select from. All of them come with a unique hero ability that can be activated once per turn at a cost of 2 mana and all of them come with a subset of the card pool (with a specific theme and strengths/weaknesses) that is cards uniquely available to that class – in addition to a great number of neutral minions which all classes can freely include.

2.3.3 Card types

In addition to minions and spells, HS also has the equipment card type, which covers various weapons that give the hero an attack value and may have special abilities besides.

Spells can only be cast in the player's own turn. Some of them are however of a subtype called "secrets": The opponent sees only that a secret has been played but not which (all secrets for a specific

class have the same mana cost) and the secret essentially acts as a trap that takes effect when the opponent performs the specified triggering condition. This serves to add some of the uncertainty and a limited amount of interference during the opponent's turn, but nothing to the scale of MTG where the opponent may interrupt with an instant at any point in your play sequence.

Abilities on minions are relatively standardized in Hearthstone. There are no activated abilities and aside from the combat staples such as the previously mentioned taunt, most of the fancy abilities fall into three categories:

- "Battlecries" that mean the effect happens when the minion is played (e.g. "give a minion +1/1").
- "Deathrattles" that mean the effect happens when the minion dies (e.g. "draw a card").
- Passive abilities that are constantly active while the minion lives (e.g. "minions cost 1 more mana").

Additionally there is the "silence" effect which negates all listed abilities and magic effects on a minion, restoring its attack and max health to original values. This, aside from just killing the target, is one of the game's most potent ways for players to work around troublesome abilities. Since abilities can also be negative (in order to balance out a particularly powerful minion) silence can even be used to remove a downside (or an enemy debuff) from the player's own minions.

2.3.4 Misc.

Based on personal experience, I would estimate playtime to be about 8 minutes on average and 15 minutes in extreme cases.

2.4 Duel of Champions specifics

Might and Magic: Duel of Champions was released in 2012 by Ubisoft as a purely digital game.

Might & Magic: Duel of Champions

2.4.1 Combat and board

DoC features a 4x4 board, with at most one card per space and players having exclusive rights to the two columns nearest themselves (meaning a limit of 8 minions on board). Minions can only attack the nearest target straight ahead (the enemy hero if there is none) and the defender only deals damage if it did not die by the attack. Instead of attacking, a minion can spend its turn moving to another row and there are spells and abilities that move minions around (without consuming a move) or target specific rows, columns or simply all minions adjacent to a target space. Common minion abilities include the ability to attack in any row and the ability to hit the targets immediately above and below the target that gets hit – minion placement is a very important part of DoC and it is a mechanic that is widely supported by the abilities of the game.

There are both "ranged" and "melee" minions. Melee can only be placed in the front column while ranged can only be placed in the back column. When a ranged minion deals damage, the enemy minion will not retaliate unless it too is ranged (and even then, still only if it survives the initial attack).

Minions have separate attack and retaliation values. Each minion deals either physical or magic damage, and that there are minions which are vulnerable or resistant to one or the other.

2.4.2 Resources

Decks in DoC constructed have 1 hero card, 8 event cards at at least 50 other cards (more on card types in next section). Players typically start with 20 or 18 health, depending on the hero they have chosen. Both players draw 6 cards and the mulligan-option is to switch that entire hand for 6 new ones if the first hand seems unlucky.

Much like in Hearthstone, the players have a resource counter that increases each turn. In addition to this, heroes have 3 attributes called "might", "magic" and "fortune" - all cards have a required value (not a cost) of at least one of these attributes to be cast. Every turn, a hero can either increase one of those 3 values by 1 for free or draw a card for 1 resource point (of that turn's budget – not the permanent value). To balance first-player advantage, the first player starts at 1 in the first turn, while the second player starts at 2 – this means that one player is slightly ahead in number of turns and number of increases to his hero attributes, while the other is slightly ahead in resources.

2.4.3 Card types

Decks in DoC are based on one of 6 factions. The faction chosen determines which minions and heroes are available, along with a small pool of neutral minions that seem to generally be slightly weaker.

Each faction has a number of heroes to choose from (unlike HS, these heroes are cards that need to be earned). Each of these has a set of starting attributes for might, magic and fortune (some may even lack one of the attributes entirely), a health value (typically 20 or 18), a number of available spell schools and possibly a special ability that the hero can use, though possibly at a cost.

Unlike minions, spells are not bound to factions but are instead based on one of 7 spell schools. This means that a fire spell can be used in a deck from any faction as long as its hero has the fire school. There are 6 schools heroes can have and the 7th, which contains the most general purpose spells (e.g. "dispel an ongoing effect"), is available to all.

Along with spells and minions there are also "fortune" cards. Aside from using the third hero attribute, I have been unable to find a huge distinction between these and spells: Fortunes seem to contain a greater quantity of cards where chance plays a role and to be a bit more inclined towards affecting resources, hero attributes and other cards (draw/discard effects) with a less emphasis on buffs/debuffs and direct damage, but overall it appears that much of what can be accomplished with one can also be achieved with the other.

The final card type is event cards, of which each player must select 8. These are shuffled together in a common pile, with two drawn and made available to both players. Every turn the oldest if these is replaced with a new draw and as long as they are available on the board, either player can in his turn pay the cost on the card to trigger the effect on it. These can contain a wide range of effects, and often affect the whole board/both players symmetrically (e.g. increasing both heroes' might by 1 or dealing 1 damage to all minions on the board).

2.4.4 Misc.

Based on personal experience, I would estimate playtime to be about 10 minutes on average and 20 minutes in extreme cases. It should however be noted that of the 3 games this is the one that I have least personal experience with and in which I have the fewest cards available to me.

2.5 Additional terms and concepts

This section describes many of the useful terms and concepts that game communities have adopted in discussing the genre and appropriate mechanics and strategies. I have written the descriptions based on how I have personally observed the terms used in discussion of the genre. The great majority of these concepts can be found in slightly more elaborated forms on community wikis such as wiki.mtgsalvation.com. The terms are sequenced so as to ease chronological reading.

X/Y minion: The abbreviated way to describe a minion with attack X and health Y. May include more slash-separated numbers in games with additional attributes (e.g. DoC).

Discard: The act of losing a card from the hand. This is often used either as a costly side-effect of something the player does or as an offensive effect by the opponent (e.g. "Target player must discard 2 cards"). If a player is forced to discard more cards than he has in hand, he typically just discards all of them (even if that is 0) with no additional consequence. Note the distinction between just having to discard, which means that the victim player chooses the most expendable cards to discard, and having to discard randomly, which is more powerful since it risks hitting the more valuable cards.

Summoning sickness: The term used to describe the fact that a minion cannot attack in the same turn it was summoned.

Keyword: An ability that is common in a game may become "keyworded". This means that instead of having a dozen cards with the ability "May attack the same turn it was played", they just have an ability called "Haste", "Charge" or something similar depending on the game. This requires a slight bit of additional learning on the part of the players, but reduces requirements on card space and allows players to more quickly achieve an overview or more easily search for cards that serve a particular purpose.

Buff: A buff is something that increases the power of something else (the inverse would be called a "debuff"). It can also be used as a verb, e.g. saying that a card buffs or debuffs something else.

Board clear: Clearing the board implies killing or removing all enemy minions. A board clear (used as a noun) is typically a spell or effect that achieves this, e.g. by doing damage to all enemy minions (or perhaps every minion on board with no distinction between friend or foe).

Board control: Players often want to maintain board control. This implies having a non-negligible presence on the board while clearing any significant threats the opponent has on his board. Due to summoning sickness, this almost eliminates the risk that the opponent can perform any significant attack in the following turn. What is involved in maintaining board control can vary a lot between games: In MTG where combat favors the defense, it may not take a lot. In HS, where combat favors the offense and many effects buff minions on board (if any exist), even a lone 1/1 minion might constitute a threat.

Threat: Something that if not dealt with will likely cause the opposing player to lose (e.g. a minion that grows stronger every turn).

Answer: Anything capable of dealing with a specific threat. Note that the responsive nature makes it a lot harder to come up with. A famously aggressive MTG-player is credited for the advice: "*while there are wrong answers, there are no wrong threats.*"

Removal: Cards that serve the purpose of killing/removing a targeted enemy card. These are often treasured as versatile answers – particularly for when an opponent plays a threat while having board control.

Trade: Any situation when a player spends cards to remove enemy cards. I could let my minion kill off your equally strong minion or use a removal spell on one of your cards. "Trade" by default assumes 1-for-1. If I have to use 3 cards to kill your threat; this would be a 3-for-1 trade in your favor.

Card advantage: How far ahead a player is when looking on the number of cards in hand and on the board. When players typically start with a fixed number of cards and only draw one more per turn, there is an inherent value to any card that either does something and draws additional cards or which deals with more than one enemy card. High mana cost cards tend to provide card advantage because it often takes multiple lower mana cost cards to deal with them.

Mana curve: Mana curve is how many cards of each cost your deck contains – often visualized as a bar diagram for each number. This concept is used in deck construction in order build a deck with an ideal chance of always being able to make a useful play in each turn. A "low" mana curve means that the deck tends towards low mana cost cards, which typically means a quick and aggressive deck. A "high" mana curve means that the deck tends towards high mana cost cards, which typically makes for a slow deck that becomes very powerful if the player can survive long enough to utilize the stronger but more expensive cards.

Hearthstone mana curve diagram

Stabilize: If one deck gets off to a superior start, the other deck will typically seek to "stabilize". This means reaching a point where the player is no longer frantically reacting to his opponent's threats and has ideally achieved board control. At this point, the slower deck can typically begin to bring its advantages to bear and stands a far greater chance of survival.

Control: Used ambiguously. A "control deck" typically either implies a slow deck that seeks to maintain enough control that it gets to apply its more efficient cards and win in the long run and/or a deck that seeks to maintain control by shutting down the opponent's options (e.g. making him discard a lot) and always having the right answers available. "Control" is also the role of playing for defense and efficiency, confident that you deck will gain the greater advantage as the game progresses.

Beatdown: Beatdown is the opposite role of control. If one deck will gain the advantage over time, then the other player has to identify this and assume the role of beatdown – that is, being the offensive player who must win before the opponent's advantage spirals out of control. Players may not always realize this if e.g. playing a slow control-oriented deck against an even slower control deck. Likewise, when playing a fast and aggressive deck against an even faster deck, it is a huge advantage to identify that you do not need to outrace the opponent but can instead take on the defensive control role because you win unless your opponent can kill you sufficiently fast.

Aggro deck / Rush deck: A very fast deck that seeks to kill the opposing deck before it can stabilize. Though the two terms are often used interchangeably, rush deck tends to imply simply that the deck is very fast, whereas aggro might also imply that the deck tends to avoid any distractions or attempts for board control and goes directly for the enemy player's health.

Midrange deck: With aggro decks being fast and control decks being slow to get up and running, midrange is the medium speed option that is less vulnerable to aggro and still hopes to kill control decks before they can bring their long-term advantage to bear.

Combo: A combo is any two or more cards in a deck that interact particularly favorably. An example could be a powerful card with the downside that it kills one of your own minions and a minion whose primary power is a strong effect when it dies. A "Combo deck" is when a player finds a particularly strong combo and builds the entire deck around facilitating it and hopefully winning when he succeeds in pulling it off.

Tempo: Tempo is essentially the resource of time and is the reason card advantage alone does not always win matches. A tempo-oriented card could be one that I play for 2 mana to return your 6 mana creature to your hand: This puts me at a clear card disadvantage (since I use a card and you do not lose one), yet gives a "tempo advantage" since you are 4 mana behind when looking solely at creatures played. An alternate (but closely related) use of the word "tempo" is maintaining pressure and board control, so your opponent is pressed to make unfavorable trades or use cards sooner than he would like in order to survive.

"Tempo decks" are not necessarily rush decks, but seek to make the most of all their resources every turn (this can be a challenge for many decks) and play a lot of cards that are likely to trade evenly with enemy cards worth more than their own cost. They try to keep their opponent under extreme pressure, and either force the opponent to continually make unfavorable trades or to establish a dominant advantage (by either control or aggro play) before the enemy ever stabilizes from the pressure or the tempo player runs out of cards.

Counter: A counter is something that does particularly well against certain strategies or card types. A "soft" counter is one that has a clear advantage in those situations but remains good and versatile, whereas a "hard" counter is one that has an overwhelming advantage in those situations but is otherwise almost useless. A 4-mana spell that removes any minion would be an excellent counter against a strong 9 mana minion, yet is still widely applicable in situations that give less of an advantage. A 1-mana card that can remove any minion which costs exactly 9 mana would be a very powerful counter that is utterly useless if the right situation does not arise (e.g. against the many players who may not even have any 9 mana minions in their decks).

Damage from hand: The damage a player can do that is not visible to his opponent. If I sit at 15 life and I know your minions can attack me for 10 damage on your turn, then I know I will survive unless you can play 5 damage from hand. The form of this damage can be any combination of anything applicable to the situation: Direct damage spells, minions that can attack the turn they are played, buffs on the existing minions etc. Damage from hand is hidden information and thus the opponent has to gamble on whether a course of play is sufficiently safe, or he should more of the resources he might want to save for a more important time.

Mill: Milling is the process of going very quickly through your opponent's deck (often through making him draw cards) in the hopes of making him lose by running out of cards. Mill decks are particularly viable in MTG while I haven't seen the strategy attempted much in HS and DoC.

Constructed: The ordinary game mode of constructing a deck and playing it.

Draft / Arena: A special game mode where the players build their decks by repeatedly choosing between narrow sets of random cards offered to them. This makes for less predictable decks and requires players to perform well in constructing and playing a deck on the fly rather than playing a familiar and highly tweaked deck.

3 DESIGN

3.1 Design Objectives

In order to decide the priorities for my design, I have first looked at what my sources describe to be the primary sources of enjoyment in CCGs.

Fullerton does not cover CCGs in particular but lists the following categories of common sources of fun in games in general [9]: Challenge, Social interaction, collection, self-expression, construction/destruction, exploration/discovery, stimulation (flashy graphics and presentation), story and living a fantasy.

Since I'm building on an existing concept, I can decide the form of enjoyment with reasonable certainty: *Collection of cards*, *construction/destruction* of exciting decks, and *competition* between players are virtual certainties. *Social interaction* between players and *exploration/discovery* of the systems and the possibilities afforded by the cards are similarly likely to be strengths of the game almost by default. While Hearthstone and Duel of Champions both do their best to provide flashy *stimulation* within the format and both draw on existing franchises in an attempt to provide *story* and *fantasy*, these particular aspects appear to be of secondary importance for the genre as such and are all less feasible for me to strive for.

The final two, *self-expression* and *challenge*, are both intimately tied to the system mechanics and are both forms of enjoyment that all good games of the genre can be said to possess, yet are also strengths that the format does not virtually provide by default. Going by Fullerton's perspective it would seem crucial that my objectives help me facilitate these.

While Fullerton's categories are general for the medium, the article '*What Makes Online Collectible Card Games Fun to Play?*' [10] is one of the few sources that focus on this genre in particular (online CCGs exclusively in this case). The author's conclusion is that none of the two general models for enjoyment of computer games he has used "are very well suited to cover all levels of these games. The reason is primarily that the games consist of several inter-dependent levels, of which not all of them fit into the classical computer game template."

The poorly fitting enjoyment models and the flaw that the study only spans 5 interviewees, all of a very similar background (top-level players with 2000+ games played, who had just quit and were selling their completed card collections) make it hard to conclude anything with certainty. Yet, in analyzing his limited data, his results mostly echo what Fullerton's model would lead us to expect:

His interviews and analysis indicate that the major sources of enjoyment are matches, deck building and collection, listed from most to least significant. Additionally he again believes that the challenge involved in a particular task is among the most important factors in how enjoyable it is, and the responses from the players indicate that the balance of the game is a massive factor in facilitating both exciting matches and deck building.

With these sources in mind, I have reached the following set of objectives:

- **General game balance:** Making sure a wide range of strategies and choices are viable and competitive, in order to facilitate challenging gameplay and meaningful player decisions.
- **Variation:** Allowing enough random factors in the gameplay to avoid total predictability and constantly face the players with fresh game states.
- **Player Agency:** In an environment with randomness, making sure that the player skill is still an important factor, the player choices are meaningful and that the player feels rewarded for good plays.
- **Depth over complexity:** Providing a wide and satisfying environment for player expression and skill, while keeping both the mechanics and the consequences of each choice transparent.
- **Achieving a rapid pace:** Letting play progress fast enough for both players to remain easily engaged.
- **Mutual fun:** Avoiding mechanics that are known to be fun for one player at the expense of great frustration for the other.

3.1.1 General game balance

Making sure a wide range of strategies and choices are viable and competitive, in order to facilitate challenging gameplay and meaningful player decisions.

In Johansson's study above [10] we see that many dedicated players cite the current state of the game balance at any given time as having one of the most important roles in their fun. I do not consider balance to be something that *adds* fun by itself, so much as the crucial facilitator for all the enjoyment of all the systems that do. In the words of Fullerton [9]: "*A game is fair if it gives all players an equal opportunity to achieve the game goals. If one player has an unfair advantage over another, and that advantage is built into the system, the others will feel cheated and lose interest in the system. In addition, a system can be unbalanced by the availability of dominant strategies or overpowered objects [...]. In these cases, the fact that one strategy or object is better than the others effectively reduces the meaningful choices for the player.*"

There are multiple ways to approach game balance. I will be striving for it on a global, rather than local scale – what is often known as 'perfect imbalance'. I have yet to find any strictly academic sources that have attempted to describe this design strategy – the only source I have seen handle this topic adeptly is Professor and Game Designer James Portnow in his *Extra Credits* show [11], in which he once a week presents and explains a concept of game deisgn.

Portnow points out how game balance in many symmetrical setups (e.g. Chess or Warcraft II) tends to make play more and more predictable as good strategies become widely known. Deep strategic/experimental play is relegated only to the most casual and the most elite players, while everywhere in-between the determining factor is who can execute the known strategies best (whether by best memorizing series of Chess players or most rapidly assigning orders to Warcraft II units).

Perfect imbalance is the concept that, rather than two of our game units being equally balanced in all contexts, I can have one that grossly overpowers yours in a particular context, and this is acceptable so long as there are other options that similarly overpower my unit in a different context. A rock-paper-scissors approach (or "cyclical imbalance" as Portnow would term it) is the simplest form of this. It

creates a "meta-game" - the part of a game that is not in the mechanics but in the evolving choices of the other players – that in the words of Portnow "Creates interesting and evolving problem, rather than letting one playstyle become definitively correct" and "Keeps the players from ever feeling they've mastered the game and there's nothing left to do."

Portnow describes this phenomenon as one of the most difficult forms of balance and presents the requirements for achieving it as such [11]:

"1. You have to create a game where no matter how skilled the player is, their character, deck or avatar can't be great at everything.

2. You need to have a firm knowledge of how your pieces interact, and what beats what. On an intuitive *and* a mathematical level.

3. You need to give your players a wide enough pool of options that they can find an answer to whatever you're going to throw at them. Without it having to be one specific pre-determined answer you have planned."

Even within the concept of perfect imbalance, there are different ways to approach it. MTG, while achieving perfect imbalance, often does so with very hard counters – e.g. cards that have a dramatically powerful effect but only works against one out of the 5 colors. By contrast, Hearthstone has a principle of giving nearly all minion cards special abilities, so you can have a versatile card that is typically slightly subpar but can be great for specific situations. I much prefer to strive for this latter approach, since I find that its counters become less of a binary thing and more a matter of having a series of several cards that possibly interact beneficially and all contribute a slight edge against certain strategies. This in turn makes for what I consider more skill-based strategic play, since there can be wide creativity employed in finding the right set of counters for a situation, and individual parts of that set in a pinch can be played for partial results, whereas binary counters tend to more frequently let it all come down to whether a player has had the luck to draw the right card for the situation.

An interesting side-effect of choosing to go for perfect imbalance is that the meta-game itself becomes an element to nurture and keep "healthy" even if it lies outside the direct control of the developer. Hearthstone Production Direction Jason Chayes and Lead Designer Eric Dodds describe[18] several cases in which they have nerfed (reduced the power of) cards that were not overpowered in isolation, but which were simply having a negative impact on the metagame by e.g. discouraging players from using most of the big fun minions, or encouraging every player to run a particular card, thus reducing meaningful choice in deck building.

In terms of achieving game balance, it is important to note that a mathematical approach can only go so far. Just like Portnow asserts the requirement of intuitive as well as mathematical understanding, Fullerton [9] also describes game balance as "one of the most difficult aspects of design", asserting that "balancing is as much about gut instinct as about numbers". Fullerton's advice echoes the design process of the Hearthstone team [12]: Perfect analytical understanding is considered infeasible, so a far more practical approach is to have a just a rough mathematical outset and a development model that allows for rapid iteration, tweaking the balance based on play experience and data as development goes along.

3.1.2 Variation

Allowing enough random factors in the gameplay to avoid total predictability and constantly face the players with fresh game states.

Randomness may seem like a surprising element to prioritize in a game that emphasizes challenge and player skill. Part of this is simply that randomness is already inherent to the genre, and needs to be managed so as not to completely invalidate the significance of player choices. Another case is that, a carefully measured dose of randomness can actually expand on the significance of player skill by

greatly expanding number of game states players can be expect to encounter and handle, as well as making for a more varied play experience. Salen and Zimmerman [13] describe the phenomenon as such:

“A game that doesn't have any feeling of randomness is likely to feel very dry, and generally more intensely competitive than a game that does have an element of randomness. On the other hand, a game that is completely random can feel chaotic and unstructured. In both cases, the goal is to give players meaningful choices within the larger game system.”

In a lecture on luck in games, MTG lead designer Richard Garfield makes a similar point [14]:

“One of the reasons you might consider adding luck to your games is because it can increase the variety in your game. In a game like chess for example, if you start playing a lot you can get caught in strategic ruts and what will happen then is that the more you play, the more you will have to rely on opening books and rote play to get to the interesting part, which will be 20 moves into the game.

When you have more luck in the game then there will often be stranger situations that will appear, which will make it so that players have to explore a new part of the game space that may otherwise not open up to them.”

This design objective is not about simply adding sheer randomness (which is trivially achieved), but about striking a balance that achieves the benefits of randomness while minimizing the negative impacts.

3.1.3 Player agency

In an environment with randomness, making sure that the player skill is still an important factor, the player choices are meaningful and that the player feels rewarded for good plays.

Quoting Fullerton [9] on the topic of making games fun:

“Game designer Sid Meier once said, “Games are a series of interesting choices.” These choices can range from where to place your blocks in Tetris to how many peons to produce in WarCraft II. If the choices have consequences, then they are interesting. If not, they are merely a distraction. Is your game providing choices with consequence? Or are your players simply micromanaging? Are players aware of the consequences as they make those choices? Creating dilemmas, where players must weigh their choices carefully, is a powerful way to challenge your players.”

The key point in this context is that for games to be fun, players should experience that their choices influence the outcome of the game. Richard Garfield makes a similar point [14], stating that as games mature their communities will typically to desire that the amount of luck be reduced so as to make the more skilled player win more consistently. He brings up mulligans in MTG as a specific example of this that was only introduced after the game had been out for several years. Garfield also mentions how the famed World Chess Champion Bobby Fischer proposed a variant of chess with randomized but symmetrical starting positions to break with rote play [15].

This is in some ways the inverse side of the variation objective. Unlike variation, I believe player agency to be purely a benefit, and thus a factor that I should always strive to maximize. This would be trivial without variation, but with good reasons for including a significant random factor in the game, the challenge lies in how to gain the benefits of randomness while maintaining a high amount of agency. Since the focus is on the play experience, notice that both actual and perceived agency are important: Having the more skilled player win 80% of the time is of limited value if it always feels like just a stroke of luck. Likewise, if agency is less than could be desired, the *impression* that player choices determine the outcome can go a long way towards mitigating this and still making the game fun.

3.1.4 Depth over complexity

Providing a wide and satisfying environment for player expression and skill, while keeping both the mechanics and the consequences of each choice transparent.

Depth is often thrown around as a vague term describing a positive quality in games, yet it is almost never defined (main sources [9][13][16] all use the term without defining it). In the (non-academic) article "Evaluating Game Mechanics For Depth" [17], Game Designer Mark Stout first looks at the dictionary.com definition: "*The amount of knowledge, intelligence, wisdom, insight, feeling... evident either in some product of the mind, as a learned paper, argument, work of art, etc.*"

He then proceeds to explain his own interpretation of the term: "To me, it describes a sweet spot -- that point during a game where the player can repeatedly display his mastery of a game mechanic. Challenges never stay the same long enough to be boring and yet they also don't change so fast that the player can't enjoy his mastery over the game."

That definition has a very wide scope for my purposes. Based on how I have seen the word used by Fullerton [9], I instead use the definition: "*The space of meaningful options, strategies and expression that a game or mechanic affords the player*". I have contacted the author and she confirms that I have interpreted her usage of the word correctly.

While depth and complexity are often correlated, it is important to note that simply increasing complexity does not necessarily add depth. In the words of Fullerton [9]:

"And always keep in mind: A more complex mathematical solution might not offer the most satisfying gameplay result. The goal is always to build a system that is complex enough to delight and surprise your players but not to confound or frustrate them."

Richard Garfield likewise makes the point [14] that agency becomes essentially random if it is hidden behind layers of complexity that are too opaque for the player to understand the consequences of his actions. He makes the example that if a game is for two players to guess the 5,320,034th digit of pi, then you have something that is theoretically a deterministic game with player agency but which more likely just degenerates to random guessing.

Fullerton [9], Garfield [14] and Stout [17] all stress the importance that players understand the consequences of their actions for decisions to be meaningful. It is my perspective that some complexity is a necessity in order to add depth to game mechanics, but that the complexity itself should be seen as an associated expense rather than a virtue in its own accord.

My focus with this objective is to recognize the complexity cost of the design choices I make and attempt to get as much depth as possible for this expense, while making sure the core mechanics are both transparent and provide a wide range of options.

3.1.5 Achieving a rapid pace

Letting play progress fast enough for both players to easily remain engaged. A complexity level that aims for the average turn to take less than 45 seconds, and the average game less than 15 minutes.

This is purely a stylistic choice rather than one of quality. I enjoyed how Duel of Champions and particularly Hearthstone managed to achieve a faster pace than Magic: The Gathering, both in terms of how long a game takes as well as how many individual turns the players go through in that time and how much those turns progress the game state. Either option can be a completely valid choice of pacing for a quality game, but I will value options that help keep the pace high – particularly as it relates to priority #4 of minimizing complexity while still offering a good depth and options.

3.1.6 Mutual fun

Avoiding mechanics that are known to be fun for one player at the expense of great frustration for the other.

One of the innovations that Blizzard's designers made with Hearthstone involved leaving out several staples of the genre, including forcing opponents to discard cards, removing their resources or interrupting their turns to cancel ("counterspell") the card they just played – regardless of what it is. In an interview [12] lead designer Eric Dodds explains their rationale:

In the end, fun proved paramount. "It's fun to play those, but it's no fun to have those played against you," said Dodds, in reference to archetypal counter-spelling or resource-burning CCG decks, which aren't feasible in Hearthstone. "The emotional negativity was so strong that we felt like we couldn't have those in our game, so we cut them."

He elaborates this point in a video interview [18]: "We don't mind you kicking the other player's butt, but there are certain mechanics that just really make you have a strong negative feeling and certainly discard is like that…" He proceeds to mention how "having a strong counterspell mechanic where you feel you can stop anything the opponent does" was avoided for similar reasons.

I agree with their observation that certain traditional mechanics are needlessly unfun to be the victim of – particularly those which are not just powerful but also take away the victim's options. There is a virtually infinite space of mechanics that can add depth and be fun to use, so I find it a worthy design objective to prioritize those that achieve this without causing an undue negative impact on the opposing player's experience.

Another example in the same vein is not letting the game draw on extensively once a player has lost all hope of winning, but rather keeping the game exciting for as long as possible and then making the final victory sweeping and decisive. Fullerton encourages designers to strive for this and, in doing so echoes the motivation for mutual fun [9]:

"Naturally, in the last stages of a game, the scales will tip, and when this happens, let the scales tip dramatically. There is nothing as satisfying as a sweeping victory. This makes the winner feel good and provides for a swift, merciful defeat for the loser."

3.1.7 Notably absent: Innovation

Building on what has been done before, without forcing myself to change or disguise solutions that are already familiar tools for the genre.

While I fully intend to add to what has been done before, I expect this to be an iterative step rather than a revolutionary one. Rather than attempting to redefine the genre, the focus is on creating a quality product with each mechanic being included on its own merits. I will be perfectly content to use or expand on many of the practices and principles that have already proven effective, instead of assigning any intrinsic value to an approach simply being different.

An additional facet of this is that there is actually a major advantage in grounding my mechanics in what feels like familiar territory to most players. A Gamasutra interview [10] with Hearthstone Lead Designer Eric Dodds emphasizes this advantage excellently: *"But make sure you "don't change too much," said Dodds, who suggested that developers should try and adhere to established conventions of the genre they're creating in whenever possible. "Anything you don't have to explain to your player is a godsend.""*

Working from established conventions and standards allows me to add deep mechanics while effectively limiting their complexity cost, as experienced by the player. Similarly to how a designer breaks with the pan and zoom control conventions of a mobile game at their own peril, assigning value

to a mechanic simply being “different”, would necessarily mean a corresponding compromise on quality as an objective.

3.2 Core decisions

With the objectives in place, this is the section where I reach the actual decisions that shape the game. As previously mentioned, designing with perfect mathematical understanding is infeasible as well as impractical. Following the recommendations of Fullerton [9] and Blizzard designers [12], my focus will instead be on establishing a clear direction and solid framework within which I can prototype and iterate on the game experience – initially based on my own impressions and later by drawing on playtesters for additional feedback.

As such, this section will be focused on the concept of each design decision and the role it is meant to fill, whereas the specific values of them will typically be either irrelevant or simply general outsets to be iterated upon.

The order of decisions is structured so as to first handle the decisions that interlock with the greatest number of other components.

3.2.1 Mana and resources

3.2.1.1 Background

A digital CCG reviewer going by the ID MugenMusou at the site digitaltcg.com has written an analysis [19] of the types of resource systems used in the CCGs. While it does not carry the weight of an academic article, I find his conclusions sound and his observations to provide helpful overview. He describes 4 types of resource systems:

No resource system: This is an uncommon model, since it takes away resource cost as a tool in balancing cards. While I have little personal experience with the games that have taken this road, it would appear that they have generally either established an alternate mechanic entirely (e.g. making certain cards pre-requisites to playing certain other cards) or simply accepted that the rarest and most valuable cards blatantly overpower most others, making for an experience where spending the most money on good cards becomes far more crucial for success at the game.

Card-based resource system: This is the classic model of MTG. It comes with both the depth and complexity involved in having an additional card type (making it easier to have different types of resources) and in having to find a suitable balance of resource cards in deck building, as well as the luck and variation caused by having the player’s ability to play his cards be dependent on not drawing too few or too many prerequisite cards. Aside from the increased luck-factor, counter-arguments include that the depth of an additional card type can easily be added in other ways, and that this system lowers the density of “important” cards. Indeed, MTG Lead Designer Richard Garfield has lamented that MTG “had about 40% boring resource cards in the deck” [20].

Sacrifice-based resource system: This iteration on the card-based resource system mitigates the luck factor by dispensing with dedicated resource cards and instead allowing players to let any card take that role. This gives a more predictable rate of resource growth and adds the depth of forcing players to consider which cards they can best afford to sacrifice - a level of depth that is not as easily duplicated by other mechanics. An additional benefit is that the ability to make irrelevant cards useful in other manners allows players to employ a wider range of niche cards while remaining competitive in most contexts. Counterarguments include that resource gain is often so important as to make sacrificing virtually mandatory whenever allowed, and make it quite similar to automatic resource gain. Meanwhile, forcing the players to make these choices every turn also slows the pace of the game,

and, while all the cards are potentially interesting, there is still the scenario where the player does not get to feel that the significant part of them devoted to resources actually do anything for him.

Automatic resource system: Resources increase automatically as the game progresses. Aside from dispensing with a resource system entirely (and not putting something equally complex in its place), this is in many ways the simpler option. It eliminates a massive luck factor in the game and lets every card be important. While HS chooses the basic approach of simply having mana per turn increase by one every turn, DoC employs an automated resource system that still manages to involve multiple relevant resources and player choice in terms of which one to advance. The counterarguments mostly come down to missing out on what the other systems have to offer, whether in terms of greater variation for the games or missing out on the interesting decisions they offer (e.g. in the case of a sacrifice mechanic).

Of these, I consider automatic and sacrificed-based systems to be the superior options:

Going without a resource system, would mean giving up a great tool for both balancing cards and adding depth in a number of ways, or replacing it with something equally complex that lacks the advantage of being easily accessible for fans of the genre. Relying on inter-dependency of specific cards would also be prone to greatly increasing the luck-factor.

A card-based resource system entails a greater luck factor and many less interesting cards. In a 2007-article [21] Mark Rosewater (MTG Head Designer from 2003 to present; Richard Garfield's Lead Designer role was in the initial years of the game) stated that he believed nearly the entire community would consider "mana screw" (getting too many or too few resource cards to play effectively, as a factor of luck) to be a negative feature of the game. He expressed a personal preference for the mechanic and listed 5 benefits that he believed the mechanic contributed: Luck in the resource mechanic gives less skilled players a chance to win, acts as a scapegoat players can blame when they lose, provides a greater variety of game states, allows for more dramatic comebacks and adds more skill to deck construction. Having the vantage point of time however, I can say that the recent automatic resource systems of HS and DoC definitely still see players win by luck, players blaming bad luck for their losses, games playing out in a very varied manner depending on draws, and making dramatic comebacks based on lucky draws – and skill in deck building remains an important factor for other reasons. Based on my observations playing these games, it would seem that many of the benefits that Mark Rosewater credited the luck of resource cards with achieving are actually quite intrinsic in the genre itself due to the random factor involved in drawing cards – even when resource gain over time is made to be a predictable factor that is not in itself subject to luck.

Hex, a digital CCG in the making that was crowdfunded on Kickstarter in June 2013 (collecting $2,278,255 against an initial funding goal of $300,000) also uses a card-based resource system [22]. This has clearly not prevented them from being popular with fans of the genre (based on a presentation of mechanics rather than playing the game). Even so, in order to make this a viable path, they have seen fit to both add a "champion"-mechanic that gives a way to spend resources for useful results when too many resource cards are being drawn, and they are also designing an AI assistant to aid their players in building decks that are less prone to mana screw. Both of these are remedial measures that it would be impractical for me to commit to in selecting my resource system.

Sacrifice-based systems have the weakness of taking many cards out of action, in the same manner as the card-based system. It is also a more complex solution, and adds a tough and time-consuming choice at the start of many turns. It compensates for this by having a very strong depth, but it matches my goal of a rapid pace poorly with the combination of actual time spent making the initial choice in the turn and the slower *experienced* pace of a game where many of the cards are not themselves played to progress the game but instead taken out of the game to facilitate other plays in the future.

3.2.1.2 Decisions

I have chosen a system inspired by DoC's solution: An automatic system that increases the resource budget by 1 every turn, but which achieves added depth by also having 3 requirement attributes (in DoC's case "Might", "Magic" and "Fortune") of which the player can only increase one of them each turn. Note the distinction of pace between asking a player "Which of your current cards (if any) would you like to give up for the rest of the game?" and "Which of your resources would you like to permanently increase first?". The latter is a less complex choice, a gain instead of a sacrifice, and with less permanent consequences, all adding up to facilitating faster decisions.

In iterating upon DoC's solution, I find it to be a weakness that DoCs attributes are essentially "strong physical damage creatures", "spells and strong magical damage creatures" and "utility cards that serve much the same purpose as spells". I find these resources to be an excellent opportunity to add some general themes for the game and let each of them have minions, spells and potentially other card types (if I choose to include any) to play with.

In choosing an overarching theme for this, I have gone with *elements*. This can provide a style to distinguish the game while also letting the individual elements be sufficiently recognizable in function to benefit from existing user associations. Leveraging what people already know to smoothen learning and understanding of mechanics is something that Plants vs. Zombies creator George Fan strongly recommended [23] in a 2012 Game Developers Conference talk that Hearthstone designers have cited as being a "key influence" in achieving easy accessibility for their own game to a broad audience [12].

Going by such existing player associations, I will be choosing the themes/attributes of *fire* (destructive and focused on offensive power), *earth* (tough and focused on defensive power) and wind (unpredictable/in motion, focused on mobility and utility powers).

Design:

- Players will start at 1 mana per turn and have this number increase by one per turn up to a cap of 10.
- Players can each turn choose to invest a point in increasing their Earth, Wind or Fire attribute.
- Cards will have requirements of these attributes, based on their power and how they fit within the themes – e.g. a spell that deals 3 damage to a target may require a fire value of 2 to be able to play.

Advantages of this design:

- It can be used for solid game balance (any of these resource models can).
- It greatly reduces the luck factor, while maintaining variation of gameplay as well as many of the scenarios cited as being advantages of the luck factor to the higher-variation models. I consider this model very strong in the balance between variation and player agency.
- While other models arguably offer slightly greater depth (particularly card-based and sacrifice-based systems), this design offers an outstanding depth to complexity ratio. Compared to the basic automatic resource model, this design is also expanded in order to enhance its depth, with a minimum of added complexity.
- This design emphasizes the experience of a rapid game pace. It avoids slow choices at the start of every turn and demands that all cards serve to progress the game.
- While losing due to bad luck is still completely possible, it avoids two such ways ("mana screw" and "mana flood" that are known to be particularly frustrating for the losing player, because they effectively take away that player's options.

3.2.2 Board structure

3.2.2.1 Background

While MTG and HS use an almost non-existent board structure for minions, DoC uses a 4x4 grid with 1 card being allowed in each space and each player only being allowed to place minions in the two columns closest to himself. This structure only allows minions to attack directly across from themselves (within the same row) at the opponent minion in front, effectively allowing the players to shield their minions behind other minions. This DoC design involves constraints on which columns allow which minions, as well as a hard limit of 8 minions on the board for a player (because he only has that many slots) and a number of abilities which either target specific rows/columns or allow a minion to affect adjacent rows as well as its own, e.g. letting the attack hit up to 3 targets at once, if they are perfectly lined up within the same column.

Another consideration in regards to board structure is whether to maintain "graveyards" (keeping track of each player's discard pile and letting cards affect them to e.g. resurrect dead minions). Used and discarded cards are permanently removed from the game session in HS. MTG and DoC instead maintain a deck of used cards for each player and allow spells and effects to interact with it to e.g. resurrect dead minions. This mechanic can add both complexity and depth, as well as adding mechanic support for the theme of decks that focus on resurrecting and utilizing the dead.

3.2.2.2 Decisions

I will use a structure similar to DoC's 4x4 board, as I think it can allow for some interesting and deep mechanics (covered in the following decision-sections), while also providing interesting challenges in terms of the AI design. The board structure and attack mechanics are also areas where I find it interesting to innovate a bit, as simple changes in these areas can have a huge impact in terms of providing a unique game experience, and I believe such changes to be very viable without impacting the quality of the game in any negative way.

I will expand the model to a 4x5 board (5 rows). DoC's 4x4 board size enforced a hard limit of max 8 minions per side, as well as great restrictions in minion movement as that limit was approached. Additionally, while the game's mechanics for affecting adjacent rows were both innovative and interesting, I found the size a bit too constraining for those purposes.

I will not be maintaining graveyards. I find this added complexity to be unnecessary: While interesting mechanics can no doubt be made in that direction, I already have a plentiful possibility space for mechanics of similar depth and quality.

Design:

- Board of 5 rows and 4 columns. Players can only place minions in the two columns closest to themselves. [During prototyping, this design was rotated 90 degrees for considerations of screen size – this does not change the gameplay implications]
- No "graveyards" will be maintained.

Advantages of this design:

- Wide opportunities for adding depth (in conjunction with other mechanics), with relatively low complexity cost.
- From a personal perspective: Offers interesting AI-challenges
- Without compromising on quality, offers a design space with more room for innovation.

3.2.3 Minion and attack mechanics

3.2.3.1 Background

In MTG the defensive advantage is threefold: The defender chooses which minions (if any) fight the attacking minions (and can even send more against one attacker if desired) in order to select the most favorable match-ups for himself. In addition to this, all minions heal to full health at the end of each turn, magnifying the impact of this advantage, and attacking minions are also rendered unable to defend and use certain special abilities in the following turn. It is not uncommon for two players in MTG to temporarily be in a stalemate, both waiting to draw a card with a decisive impact while it is clear that the first player to attack will find himself at a disadvantage. Other notable consequences include making stronger minions significantly more valuable (since a 6/6 takes no permanent damage by defeating a 5/5) and making minions with low health totals far easier to keep alive (since they cannot be targeted with attacks but have to be killed with spells).

HS and DoC have instead placed the advantage on the offensive player, making minion attacks something that the player will nearly always want to perform, and keeping both the literal and the experienced pace high because hardly a turn goes by without something attacking and minions dying or a player's health total decreasing; beyond simply making the game progress faster, in my experience it also ensures that the individual turns feel more meaningful while the game progresses.

HS achieves this by letting any minion attack any other minion, except for minions with the "taunt"-ability which have to be killed before that player or any of his non-taunt minions can be attacked. This creates a very high pace and a clear offensive advantage of being the player who has minions ready to attack whenever the other player plays a minion and tries to build some board presence. It has a clear positive feedback loop of always letting the player with the strongest board presence and initiative make the most favorable trades, increasing the advantage – this is in many ways a healthy incentive, but I do find that the offensive advantage might be slightly too high and that the defense is somewhat limited in its options, often being in the situation of certain loss unless the right card (e.g. deal x damage to all enemy minions) can be drawn to come back from such a situation.

DoC lets its minions attack directly in front of them, hitting the front enemy minion in the same row, or the hero if no minion is there to block the attack. They have the additional mechanic of letting offense deal its damage first, meaning that any attack strong enough to outright kill the defending minion incurs no retaliation and thus also creating a positive feedback loop. On the other hand, it is often quite easy to protect weak minions with utility abilities (as opposed to those focused on strong combat attributes) since minions can only attack directly across and any minion with no interest in attacking can switch row every turn. As a player, targeting can also sometimes feel like a bit of an automatic affair with little option for choosing the best targets, since minions either have the choice of attacking directly across or switching row (which means the inconvenient defender can just spend its turn moving similarly, unless that space is blocked).

DoC additionally works with a distinction between ranged/melee minions (in terms of what can retaliate against each other) and physical/magic damage. The ranged/melee distinction also involves restrictions on where minions can be placed, though it appears to me that this mechanic primarily serves a thematic purpose, as it can be argued that it would make less sense for melee minions to be allowed to attack from the rear column.

3.2.3.2 Decisions

I wish to experiment with letting minions attack the front minion their own or any adjacent row, and the enemy hero if a clear direct line of attack exists (no enemy minions in the attacker's own row). This makes targeting a far less automatic affair and opens for many interesting decisions, as well as making

elusive minions a bit harder to hide somewhere they can't be reached, while still leaving limited defensive options when no taunt is in hand.

All of that adds up to a significant offensive advantage. Part of this can be restrained by making sure minions don't automatically heal every turn and that both attacker and defender always take damage when an attack is resolved. To further balance this, and since protecting the hero and vulnerable minions is made harder, I will also be including a taunt ability for minions, that means the minion can be attacked from anywhere and no other minions or the hero can be attacked while taunt minions remain alive for that player. One special case of this is that since only the front minion in a row is ordinarily targetable, I think it will both be more intuitive and add depth to the game if the taunt ability is inactive as long as a friendly minion is placed in front of that minion.

With a board where minion positioning is of significant importance, I will also be following DoC's lead and allow minions to move position. I think that DoC has minions jumping a bit too much all over the place (particularly those with low interest in attacking), and since I have chosen a larger board there will be even more freedom for such behavior, but I believe the wider freedom of targeting may counterbalance this sufficiently. I will set out with DoC's model of allowing a minion to move to any other valid location instead of attacking and based on those experiences consider whether to add further constraints – e.g. a resource cost or a distance limit for a single movement.

For minion attributes, I think that attack and health are perfectly functional values that I'm content to work with. I can already differentiate minions vastly by their abilities, costs and resource requirements, so I don't see any particular need for adding further attributes. One very valid possibility to consider, is having a separate value for damage done by retaliation, as in DoC, but the wider freedom of attack targeting makes this value a lot less valuable (since minions are better able to attack around the obstacles an opponent places in front of them) and I frankly think it is a level of complexity that I do not need.

I will not use an inherent distinction between ranged/melee minions or physical/magical damage. I think having these aspects as basic elements of combat adds a needless level of complexity – particularly since a basic approach without concern for these elements still allows for making minions with the listed ability "Avoids retaliation" or even adding a "Ranged" keyword-ability with the effect that it can only be retaliated against by other minions with the same ability. Should I even wish such an ability down the road, having it as a clearly listed ability on a minority of minions is still superior to making the type of attack a basic consideration (and taking up card space) for every single minion in the game.

Likewise, I will dispense with DoC's restrictions on where certain types of minions can be placed. I consider this restriction a needless layer of complexity and think that removing it might actually prove very beneficial by widening the range of tactical options available to the players.

Design:

- Minions can attack once per turn except not in the turn they are first played.
- A minion can attack the frontmost minion, in either its own, or any adjacent row.
- A minion can attack the enemy hero when no enemy minions remain in its own row.
- Minions with the "Taunt"-ability can be attacked from anywhere. The owner's hero and other minions cannot be attacked so long as a taunt-minion is alive. Taunt is negated while placed behind a friendly minion.
- Minions have the attributes "Attack" and "Health" (aside from abilities and cost/requirements).
- Both attacker and defender deal their attack damage to each other.
- Minions do not heal automatically.
- There will be no constraints on minion placement (within the player's two columns).
- Instead of attacking, a minion can spend its turn moving to any other valid space.

- Minion actions are resolved one at a time in any order, rather than all at once.

Advantages of this design:

- Offers a wide range of meaningful player options, while dispensing with many elements of complexity from previous games.
- Rewards high pace and offensive play, while still slightly limiting offensive advantage and how it takes away the defensive player's options, as compared to Hearthstone.
- Maintains tools for keeping defensive play competitive and balanced.

3.2.4 Card types

3.2.4.1 Background

In the previous decisions I have already worked from the assumption that the game would have some form of minions and some form of spells:

In an MTG design internship competition, MTG R&D posed the question of which of the 6 card types they would remove if forced to choose one [21]. In evaluating the responses, Head Designer Mark Rosewater said that he thought a reasonable argument could be made for any of the 6 except creatures (their minion type) and that they had internally joked about instantly throwing any who might make that choice into the reject pile instantly. In this it should be noted that "spells" was not an option to choose, since MTG effectively has 3-4 different spell types and reasonable arguments could be made for eliminating certain types of spells or rolling them into other card types.

While it is theoretically possible to dispense with the minion and/or spell type in a CCG, either of these choices would create something wildly distinct from what is commonly expected within the genre – I personally cannot name a single CCG that I'm aware of having done away with either. Since I have chosen to work within this genre and with an emphasis on quality over innovation, I am perfectly content to accept both of these types.

Across all the example games, these two types form the basics. Hearthstone has an additional type of card (weapons) but 5 out of 9 of its heroes have no cards of that type and make do just fine. DoC has the fortune cards which don't do anything I couldn't do with the spell type anyway. It also has event cards but these are not so much a card type in the deck as a an entirely separate mechanic layered on top – although the basic tradeoff of increased variation in return for added complexity does apply for this as with other additions of card types.

Assuming a spell type is used, another consideration is how broad to make its functionality. MTG has essentially split their spell type into 3-4 different types of cards to categorize different types of functionality. The two most important decisions here are whether to allow spells to be cast during an opponent's turn and whether all spells are resolved instantly or some are allowed to stay in effect for a number of turns or until the opponent dispels them somehow.

3.2.4.2 Decisions

I do not see a need for more than the two basic types, as these have proven sufficient avenues for providing nearly endless variation and depth. Additional card types (when done well!) can also be avenues of providing those benefits (possibly weighed slightly more towards variation than depth) with a complexity cost that is no less reasonable than that of expanding what is possible within the existing classes, but that still requires both development resources to add the feature and the design competence and playtesting/iteration to polish it well. While both choices can be very appealing in the right context, I find that the scope of this project clearly favors the simpler approach.

For casting spells in an opponent's turn, I have already touched upon how I agree with the HS designers' rationale that counterspells (cards whose effect is essentially "negate the card your

opponent just tried to play") cause a needlessly negative experience for the victim. That in itself does not rule out allowing other spells in an opponent's turn (e.g. "deal 3 damage to target" rather than "negate whatever your opponent just played"), but I find that part of what allows HS and DoC to play out at a faster pace is that players can focus on how to play their own cards in their turns, without concerning themselves with backup plans in case they get sabotaged along the way. It makes for fun and flashy combos, without taking away the need to consider what the opponent might have to counter it in his own turn. In all fairness, disallowing spells outside turn is going to cause a significant loss of depth. When I still make this choice it is because I feel that loss is outweighed by an even greater reduction of complexity, and a significant increase in the pace of the game.

As for spells staying in effect, I feel this is something that adds a lot to MTG, both in terms of depth and complexity. However, I've experienced a tendency in MTG and DoC that it can be hard to include adequate counters to such effects because it's hard for a deck to be competitive with more than a few of such situational cards, which often makes such situations come down to a binary result of whether the opponent has had the luck to draw the answer or not. By contrast, I feel that Hearthstone's solution is quite elegant, in that they moved such lasting effects over to be passive abilities on minions, which both means that an entire layer of complexity is removed as thee effects are countered by the basic anti-creature tools (including ordinary attacks), and also creature an additional tool for depth and balance in that not all effects are equally vulnerable. Whereas an MTG enchantment is binary in that the opponent either has something to dispel it or not, HS minions can have vastly different health totals and can even have special abilities or buffs that help them stay alive or make them more exposed. I will be using this approach, rather than having to establish lasting spell effects.

Design:

- The card types are "minions" and "spells".
- Cards cannot be played outside of one's own turn.
- Effects are by default either immediate, permanent (until actively removed) or lasting only until the end of the current turn. Temporary impairments for enemy minions (e.g. "skip a minion's next turn") are an exception and may last a specified number of turns.
- Spells do not stay in play. Any lasting effect will have to be attached to a minion and disappears when that minion dies.

Advantages of this design:

- Very low complexity while maintaining a wide range of options for providing depth and variation.
- Favors a fast progression of the game, while avoiding options that are known to impair mutual fun.
- Well-suited for the scope of this project. The additional classes that other games have introduced generally (read: in the better cases) represent a balanced cost of complexity for the additional depth and variation achieved. In light of this project's limited development resources, that trade simply is not of significant priority.

3.2.5 Balance scheme

3.2.5.1 Background

While solid balancing includes a significant dose of gut feeling and prototyping, the process can be eased a great deal by developing a mathematical model for the strength of a card (relative to its cost and requirements) and using it as a guideline in assigning the initial values.

Beyond easing and speeding up the balancing process, using such a model from the outset is also valuable for the long-term development of a CCG. As new content is released there are two general models for encouraging purchases:

The easy way is to embrace "power creep" – that is, letting stronger and stronger options become available over time. If a player wants to remain competitive, he will have to buy the new content. The downside of this model is that it allows game content to degenerate into worthlessness over time, harming the initial version of the game as well as taking away the value of a player's collection and continually giving a fresh excuse to quit.

The far more challenging model is that of attempting to keep new content on the same power level and expanding the game in the width of options rather than the strength of them. Part of what makes this challenging is that players can choose the best of the options available, such that every new release of content risks including a few cards that are almost universally better than some existing option. Likewise, some extent of power creep is still going to happen simply because better options become available, but while specific decks might go slightly obsolete without including the full range of cards that would be best for them, the individual cards of the decks should generally remain competitive. MTG, DoC and HS all strive for this second option, although the latter two have a far easier time of it with the move into the digital space where content can be balanced after release – even so, it is something done with great care since it is a priority to let players feel ownership of their card collections to the same extent as their physical competitors [24].

Though I have the luxury of not having to support my game after release, this should not be an excuse for designing a game that cannot maintain its quality in the long term. I will design and use a balance scheme, both for improving the initial balancing process and for the theoretical long term sustainability of my game balance.

In designing my own model, I have looked into how Hearthstone has approached the balance of its cards (minions in particular) and certain trends quickly became clear. I have later found that my reverse-engineering of Blizzard's balancing process is consistent with observations that a major community site made in October 2013 [25]:

- A minion has a budget of 1 + 2*mana cost points of attributes.
- Class-specific cards and certain high-rarity cards have an extra point of value.
- Each point of attack or health costs 1 point of the card's budget.
- Nearly all minions have an ability. Each ability has a (positive or negative) cost.
- The cheapest minions (mana cost <= 2) seem to have 2 extra points of value that can only be spent on abilities.

This process is by no means perfect, but provides a strong guideline for how strong attributes and abilities to assign a minion of a specific mana cost, or how high a mana cost to set for a particular minion concept. The greatest source of uncertainty is in pricing non-standardized abilities: Whereas keyword abilities (such as 'taunt') have a set cost, a unique ability such as "Swap the Attack and Health of a minion" may be hard to judge the quality of. Additionally, it is worth noting that successful games often very deliberately allow certain elements to be slightly better or worse than the model should ordinarily allow – part of the skill is in choosing the best cards and even in identifying when a card is overplayed to the extent that it might be even more valuable to select an otherwise weaker card that acts as a useful counter to it. This all comes back to the principles of perfect imbalance [11].

3.2.5.2 Decisions

In designing my own balance scheme, Hearthstone's model provides a useful outset. The primary difference I need to account for is that my minions will have requirements of the earth, wind and fire hero attributes. This means that I both need to consider which requirement values to use as baseline for a given power level and that I need to consider how it should be budgeted in terms of ability cost

when certain minions have requirements higher or lower than usual or possess abilities that lie outside what the respective element is normally capable of. As an example, a fire spell or minion might be distinguished by having a slightly lower requirement for a certain type of effect (making this effect accessible to strategies that invest very little in fire), but since this reduction of requirements incurs an ability cost, it means that the effect will either be less potent or have a higher mana cost than similar fire effects with no such reduction in requirements.

Design:

Effort will be made to balance the game in a way that can be maintained over time and hypothetical content expansions. A modified version of Hearthstone's balance scheme is designed:

Cost of attributes:

- Damage and health each cost 1 point.
- Each ability is assigned a cost (positive or negative) – most of them constant values, but a few minion passives are scaled with how efficiently the minion is able to apply them.

Assigning budget:

- Cards default to a budget of 2*[mana cost]+1.
- Main attribute requirement defaults to 1+floor([mana cost]/2).
- Budget is increased by 2 for each 1 point increase in main attribute requirement and by 1 for each increase (above 1) in the requirement of other attributes.
- Budget decreases at same rate for cards with lower than typical requirements. Multiple attributes can have the same requirement value – one is calculated as main and all others secondary.

Restrictions:

- Virtually all abilities are aligned with a specific element (e.g. haste and damage for fire, healing and taunt for earth, movement abilities and card draw for wind). For a card to have an ability at a certain strength, implies a requirement of the aligned element proportional to the ability cost.
- A card which has a lower than expected resource requirement for an ability (e.g. an earth-based damage spell, with low fire requirement) incurs a greater mana cost for the same effect. Certain effects are entirely unavailable without some minimum value in the corresponding resource (e.g. that earth-based damage spell would likely still require some amount of fire – just less than what the amount of damage would normally imply).

Calculation tools for this balance scheme are set up and used as a guideline during content creation.

Advantages of this design:

- Enhances awareness and consistency of balance decisions during content creation.
- Promotes solid game balance and long-term maintainability. These gains come without a cost in the other objectives.
- While the balance scheme requires an initial investment of development time, more time is saved down the line (quicker content creation and fewer balancing iterations) for any project that aspires to a good game balance.
- Restricting abilities based on elements enforces the theme, by actually giving the choice of elements depth and meaning. Realizing the elements as distinct playstyles (which can still be combined in hybrid decks) also adds variation to the game. These are all great gains compared to the moderate complexity cost, but it should be noted that the potential was created and the

gains anticipated from the moment theme and resource system were decided – this is just the place where we see the potential of the isolated mechanics realized in a meaningful whole.

3.2.6 Deck and card constraints

3.2.6.1 Background

CCGs generally come with a number of constraints: The number of cards in a deck, how many copies of each card is allowed, and how many cards are allowed in hand.

Number of cards in a deck

The ordinary (non-draft) format of MTG requires decks to be at least 60 cards. DoC requires at least 50 (not counting special card choices for mechanics that are not a factor in this game) and HS requires exactly 30. CCGs are virtually always played with as few cards as the game allows, since removing the least important card from a deck will generally improve both the deck's average card quality, the predictability of future options and the odds of drawing any particularly important cards that are already included in the maximum amount allowed.

Copies of a particular card allowed

MTG and DoC decks allow up to 4 copies of each card, although DoC also features "unique" cards which only allow 1 copy. HS allows 2 copies of each card, except those of the highest rarity (legendary) which are only allowed 1.

A game's ratio, between number of copies allowed and total number of cards in the deck, plays a large role in determining how reliably decks can draw the cards they depend on, and thus how random the game plays overall. On the other end of the spectrum, this limit of copies forces diverse and less predictable play.

Hand limit

MTG has a hand limit of 7 cards at the end of a player's turn, after which he is forced to discard cards of his choice until he is down to 7 again. This can both lead to negative feedback loops, for players with a sufficiently unlucky draw that they are unable to play their cards, and enable strategies that rely on being able to discard certain cards (because that game operates with a graveyard, where dead cards can be affected).

HS operates with a limit of 10 cards in hand at any given time – after this, any card that would be added to the hand is instead shown to both players and discarded. This number is large enough to not be a hindrance most of the time, but still enables strategic usage by filling up an opponent's hand to cause lost cards (which are then known to be eliminated – potentially a huge advantage) and/or eliminate valuable minions by returning them to hand.

DoC appears to effectively have no hand limit (I have found no mention of it, and was able to reach 19 cards before the AI killed me).

3.2.6.2 Decisions

Deck size and copies

The game will require 40 cards in a deck and maximum of 3 copies of a particular card.

I find that HS achieves a really strong sense of having each card matter: The lower deck size, combined with a pace that allows a greater number of turns during a game, means that it's a relatively frequent occurrence for games to go on until both players have run through most of their decks, and that skilled players generally maintain a good sense of what they have left in their decks. These are qualities of pace, depth and agency that I want to seize on. That said, I think it will be worth it to slightly increase

deck size (thus correspondingly decreasing the importance of each particular choice) in order to realize other gains:

I find that HS can at times feel so constrained on deck size that players feel their entire deck consists of must-haves for their strategy, with very little room for including some nice-to-haves of personal preference. I wish to expand on these options and also think that it would be beneficial for customization with a middle ground between "as many copies as possible" and "a single one in the deck". There is a genuine risk that players may instead feel forced to include maximum copies of nearly every card and thus feel even more constrained, but being able to take the risk and test for such things is what early design iterations are for.

Additionally, both MTG and HS commonly feature the frustrating scenario where one has deliberately built a deck with a maximum number of counters for exactly the situation at hand, yet that card simply has not been drawn. Being able to further increase the ratio of a particular card in the deck adds more agency to such situations.

Hand limit

The game will use Hearthstone's solution of having a hand limit of 10 cards, with additional draws being shown to both players and discarded.

Having no hand limit is a perfectly viable solution, in general, so long as the GUI is implemented to handle it. Having a hand limit can however add certain benefits: It limits how far one player can get ahead and thus better keeps the game open for most of the game (a factor in mutual fun) and also increases pace by forcing players to actually apply their resources in order to gain an advantage. Additionally, both MTG and HS show that a hand limit can also add depth to the game by enabling strategic usage. Of these, I favor Hearthstone's approach with a limit set high enough to not get in the way of normal play, or further punish players who get an unlucky start, and a consequence strong enough to occasionally use the mechanic as a weapon against the enemy.

Design:

- A deck is 40 cards.
- A deck can have a maximum of 3 copies of each card.
- Hand limit is 10. Any additional cards that would be added, are displayed to both players and discarded.

Advantages of this design:

- Large impact of individual card choice in deck building.
- Significant freedom of customization / room to add personal touches to decks.
- Ability to specialize better and minimize luck factor in specific areas, if preferred.
- Hand limit achieves minor, but cheap, gains in pacing, depth and mutual fun.

3.2.7 Starting hand, mulligans and counterbalancing first-player advantage

3.2.7.1 Background

The size of the starting hand as well as a game's mulligan mechanic are significant elements in managing both randomness and the amount of options each player has available. Additionally, these are typically the same parameters used for counterbalancing first-player advantage.

First-player advantage is a traditional challenge in balancing turn-based games. Games such as Connect-4, Go and Chess tend to intuitively appear balanced by virtue of the complete symmetry but when you dig into specific data of high-level play (e.g. chess [26]), tendencies favoring the starting

player will generally become apparent. Hearthstone Designer Ben Brode does an excellent at job explaining the challenge from a CCG perspective [27]:

"Throughout the development of Hearthstone, going first had been a tremendous advantage. Winning the coin flip during our alpha testing meant you were 20% more likely to win the game. Losing the flip 3 games in a row was a table-flipping experience.

Being Player 1 has two advantages: (1) You get to spend 3 mana before you opponent does. (And 4 mana. And 5 mana. Etc.) This means that you can respond to your opponent's 2-mana play with a 3-mana card, but your opponent only had 2-mana to respond to your 2-mana play. It also means that you are usually the one choosing who you want to attack, which ends up being a huge advantage.(2) You get more total mana than your opponent. Imagine a game that ends on turn 7. If you went first, that means you had access to 1+2+3+4+5+6+7 total mana (that's 28!), but your opponent has only had access to 21 mana up that point, since they haven't had a 7-mana turn yet. If player 2 wins, both players have had access to the same amount of mana throughout the game."

A quick summary on how the other games deal handle these parameters:

MTG has 7 cards in a starting hand (out of a 60 card deck). Mulligans work by letting the player draw his entire hand over with 1 card less than previously and repeating this process as desired. A popular house-rule to this (also applied in the MTG: Duel of the Planeswalkers digital games) that the first mulligan does not reduce the hand by a card, thus reducing the luck factor further. First-player advantage is handled by not letting the starting player draw a card at the beginning of his first turn.

DoC has 6 cards in a starting hand (out of a 50 card deck). Mulligans work by giving the player a single opportunity to draw his hand over. First-player advantage is balanced by letting the second player start his first turn at 2 resources, thus letting him have more resources at the start and throughout the game (similarly to the HS example above), while the opponent instead gets more turns total and gets to increase his secondary resources (equivalent to this game's elements) first.

HS has 3 cards in a starting hand (out of a 30 card deck). Mulligans work by letting the player choose which individual cards to shuffle back into the deck and draw over. First-player advantage is counterbalanced by giving the second player a 4th card (which can also be mulliganed, thus leading to a more consistent start) and a spell card which can be played to gain 1 extra mana for a single turn.

3.2.7.2 Decisions

A common theme for all 3 of these decisions is that they take supportive roles; with the more fundamental mechanics in place, these are relatively easily tweaked elements that for the most part "just" have to align in the sweet spots that enables everything else to work at its best. This is not necessarily easy but, going back to the experience of Fullerton [9], Portnow [11] and the Hearthstone designers [12] (see section 3.1.1 on game balance), mathematical understanding alone is no replacement for also playtesting the game and experiencing the gameplay first-hand. This is one of those instances where the practical solution is simply just to iterate on the initial values until the gameplay feels right. As such, the specific solutions are not as important here as just being aware what I want to achieve and selecting values/models for the first iteration that are ballpark sensible for those purposes.

Mulligan

Mulligans evolved in response to an increasingly skilled CCG audience requesting measures to reduce the luck factor and make players have more agency in determining the outcome of a match [14]. To this end, Hearthstone's system stands out by placing a greater number of significant decisions in the hands of the player, while also providing the most agency (mulliganing a full hand in MTG or DoC is essentially just a subset of the options a player has in Hearthstone). While I will start with this system, there is some risk of creating too predictable play (particularly when hand size increases, as compared

to number of cards and copies in decks). An eye should be kept out for that during iteration, but if it doesn't become a problem, this is otherwise the best solution of the three.

Starting hand

One thing that's struck me about Hearthstone's 3-4 card starting hands is that it is nearly always ideal to mulligan solely based on early-game priorities and just hope that the mid-late game draws fall into place in the turns it takes to get there. I would like to see how the gameplay turns out with a larger hand, that generally provides the players a wider range of options, gives them more freedom and agency during mulligans and even allow them to keep one or two valuable late-game cards in their initial hand, and still have a viable early-game. I think 6 would be a good number to start at in trying for these goals.

First-player advantage

The Ben Brode quote above does a good job of summing up how to approach this problem, namely that I would want to observe the magnitude of it in high-level play before deciding on the extent of counterbalancing needed. For the initial iteration I have chosen to let the first player be unable to gain a resource increase on his first turn, meaning that the second player can use his greater resource values to play higher-requirement cards sooner, have a wider range of options open to him or have more freedom to skip an element increase or two for a temporary boost in mana at the right times. In addition, the starting player does not draw a card at the start of his first turn.

These are both simple and non-intrusive adjustments that should help in achieving a better approximation of balance for the first iteration, without any significant implementation cost for this first-iteration solution. By non-intrusive, I mean that it is based in core mechanics (tweaking the availability of existing resources) and therefore unlikely to obscure any underlying problems, which could make future iterations harder.

Design:

- A starting hand has 6 cards
- Each card can be mulliganed separately. They are all redrawn at the same time. This can be done only once.
- On the starting player's first turn, he does not get to draw a card or increase a resource.

Advantages of this design:

- Prioritizes solutions that decrease luck factor, emphasizing player agency.
- Provides a wide range of options for the players, again increasing agency.
- Counterbalances first-player advantage to some (unknown until measured) extent.
- None of these first-iteration solutions have huge implementation cost or significantly complicate the task of analyzing results and iterating further.

3.2.8 Hero interaction

3.2.8.1 Background

Both HS and DoC operate with giving heroes abilities to directly affect the game each turn (or when certain conditions are met) – typically at the cost of a few mana.

DoC has heroes as cards that players can collect and use for their decks, meaning that there is a selection between a large number of heroes (for each of the 6 "factions that determine which cards the player is able to include in his deck). These can stand out by their starting resources, which spell cards they have access to, how much health they have and which ability they have. This effectively means

that there's a wide range of hero abilities one can run into when faced with random opponents, that having an active hero ability generally comes at a cost in other areas (e.g. lower starting resources), and that there's a lot of opportunity for customization, by finding a hero ability that synergizes with a particular deck. This adds some complexity but also a significant amount of depth.

By contrast, HS has chosen a far simpler approach and only has 1 hero of each "class" (equivalent to the DoC factions). Each of these heroes comes with a special ability, that very much defines what the class is about and which is typically supported by a great number of class-specific cards. While this limited number of hero abilities adds less depth, it does so at a lower complexity cost and also gives the developers a lot more freedom in balancing the game, because they can mostly avoid the concern that a new card, which would be balanced and useful in the hands of most heroes, becomes overpowered to one or two that are specialized in that direction.

Another interesting factor is that the opponent's strategy becomes more transparent in DoC, because the hero choice says a lot about which cards are likely to fit in, whereas HS is built such that most classes have several very different strategies – though admittedly it took HS several expansions to get to that point and, even 2 expansions into the game, the classes with 2-3 vastly different competitive archetypes (e.g. one fast aggro and one slow control) gain significant advantage in mulligans and early-game predictions as compared to classes which only have 1 competitive archetype or whose archetypes don't differ a lot in tempo.

3.2.8.2 Decisions

While I admire the elegance of HS' solution and the gains they achieve at very marginal cost, I believe that having a wider range of heroes (essentially choosing from a card pool of heroes as in DoC) would actually be the better option for this game: With only 3 different elements (as opposed to having factions or classes), a greater range of heroes would be a way to add to the variation and depth (e.g. heroes that support certain hybrid strategies). A significant part of the remarkable balance and depth that HS achieves with their simpler solution is gained by balancing both the classes and the overall card pool with those abilities in mind. It is unlikely that I would be able to realize those gains to quite the same extent, since balancing around just 3 hero abilities is likely to have a higher cost in the variation, and since I won't be able to produce the same volume of content around them.

All that said, I have still chosen an outset in the HS solution. While I consider a wider selection of heroes slightly better suited for this game, both solutions perform very well, and the relatively small advantages do not appear to justify the increased cost in development resources for both the additional content and GUI requirements – at least considering the scope of this project and how scarce I can expect those development resources to be.

The selected abilities are:

- Earth: Give a minion +1/+1 until end of turn.
- Wind: Move a minion to an adjacent space.
- Fire: Deal 1 damage. [to any target]

In designing the specific abilities, I have striven to make them as diverse and flexible in their uses as possible. An example would be that even offensive abilities (e.g. "deal 1 damage") do not require that the target be an enemy. This may initially sound counterproductive, but these are abilities that are always going to be at hand and so it adds a disproportionate amount of depth if they are not just applicable for the obvious scenarios but also to e.g. take a friendly minion with a powerful effect upon dying and using the hero ability for that final point of damage at just the right moment. Similarly, the wind ability could wreak havoc on an opponent's plans if it gets away with moving an enemy taunt minion behind one of the other minions (thus making the taunt-ability inactive).

I considered variations, such as letting the hero ability not cost mana and instead be an alternative to the resource increase for the turn. That particular idea was discarded because the hero ability should

let players feel good for puzzling out a way to utilize it to counter an opponent's play and thus gaining a card advantage – not feeling disappointed for giving up a permanent resource increase (which would also greatly decrease the frequency of its use). Another idea, getting an improved version of a hero ability when reaching 6 or 7 in that element, was discarded for discouraging diverse strategies that utilize multiple elements well. In the end, the basic solution was deemed superior.

Design:

- Each element has one hero
- Each hero has a unique ability that can be used once per turn for 2 mana
- The abilities should ideally be simple, flexible in their use (e.g. being able to affect both friendly and enemy minions) and fit the theme of the element.

Advantages of this design:

- Significantly increases depth.
- Relatively low costs in both complexity and development time.

3.2.9 Card acquisition, rarity and monetization

Collection is a classic element of CCGs, highlighted by multiple sources as an important source of fun in games and CCGs in particular (see section 3.1: Design Objectives). The only reason this is listed not a general design objective, is that the collection and card acquisition mechanics tend to be a whole parallel system to the primary game rules and execution; while it's obvious that one would want both systems, they may well never actually interact.

I have chosen to scope off the collection element of this CCG and focus on the core game. Implementing a meaningful solution would have required a lot of time-consuming, yet unchallenging, GUI work as well as a sufficiently large card base to still have some freedom in deck construction before unlocking other cards. For these reasons, I have chosen to prioritize other aspects of design and implementation. Both design and implementation are designed to interface with a card acquisition system, but no specific design for it has been chosen. I will however briefly cover the considerations that would be relevant in doing so:

Rarities

MTG originally launched with the rarities "common", "uncommon" and "rare". They have since had to expand this to the additional rarity "mythic rare". Lead Designer Mark Rosewater explains [28]:

"The trading card game genre has created some standards that evolved from decisions made after Magic's creation, rarity being one of the best examples. The idea of a TCG with only three rarities is antiquated. Magic is the only major trading card game currently printed with only three rarities. If we want to stay competitive in attracting new players we have to keep up with the industry standards."

"…there's a reason things become an industry standard. They work. Knowing that you have the potential to open something you can show off to all your friends is very compelling and helps draw new players into the game."

Coming from an industry leader with 15 years on the market at the time, this speaks highly in favor of using 4 tiers of rarity.

For an example of how this is distributed, 63 EUR worth of Hearthstone card packs gives 300 cards with an average distribution [29] of:

- *214 tier 1 "Common" cards*
- *69 tier 2 "Rare" cards*
- *13 tier 3 "Epic" cards*
- *4 tier 4 "Legendary" cards*

It should be noted that both Hearthstone and Duel of Champions facilitate such high rarities by having a backup system that allows purchase of selected cards (priced based on rarity) with a currency that is also gained from acquiring cards.

Additional innovations on the topic of rarities include that HS and DoC both, as free to play games, provide a certain amount of starting cards that don't have to be unlocked at all, and that HS uses the highest rarity for the cards with the most distinctive (not necessarily best) abilities and only allow one copy of those cards in a deck. Many of those "legendary" cards effectively add both new depth and variation to the player's play experience when used, because they are capable of changing the game so distinctively.

Monetization

Card acquisition is highly dependent on the form of monetization desired. I see two overarching strategies:

A free to play method could be pursued, similarly to both DoC and HS: A number of cards are made available from the beginning, and new cards can be acquired slowly through play or quickly through purchase. The games I have used as examples have all disallowed trading between players, although I have seen at least one free to play digital CCG allow it (*Infinity Wars 2014* by Lightmare Studios)– I can see either choice working as long as it's taken with awareness of the pros and cons. If card acquisition is used for monetization, it's important to note that one cannot get a true sense of the game without real payments enabled – Blizzard developers have elaborated on this, in explaining why players had to pay for cards in even the beta stages of Hearthstone [18].

The second strategy I'd consider viable is to go entirely for a singleplayer experience, with a one-time purchase: This is less ambitious in terms of income potential but also sidesteps the intense competition for that market. An advantage of this approach is that players can be granted a faster and more satisfying rate of progression through ordinary play. The MTG Duel of the Champions series has mostly relied on this approach (though the 2015 approach has made a partial shift to paying for cards) and this genre has also seen success in the past with games such as Nival Interactive's Etherlords (2001) and Etherlords 2 (2004).

Some sort of middle-ground (e.g. free-to-play) single player can obviously also exist. When I have highlighted these two approaches as the ones I would have been most likely to choose from, it is based on personal estimations of players' willingness to pay up-front for multiplayer products (when "free" high-quality alternates exist) and to pay for performance-enhancing content in single player products for PC.

3.3 Naming the game

Having a name for the game is mostly a matter of convenience for the following sections of the report, and I won't claim any deep scientific background for the choice. While a focus group could be used, given the resources, and some marketing research may be relevant to naming a product, both of those areas are outside the scope of this project and a simple subjective decision will suffice.

I have chosen the name "*Elemental Command*" (EC). It is a name that I find memorable and to the point, while conveying both the elemental theme and the strategic genre. It borders on the generic, but in a

genre populated with titles such as *Duel of Champions* and *Infinity Wars*, I don't find that a cause for concern.

A brief search of Google, Google Play and the US trademark [30] register doesn't indicate any conflicts. Numerous role-playing games (e.g. *Dungeons and Dragons* and *Everquest 2*) have items "of Elemental Command". For a medium-large publisher, this is mostly just a convenient nostalgic association. For a small indie developer, I would steer clear and choose a name with a better signal-to-noise ratio on web searches.

3.4 Game rules

This sums up the rules of the game and is written so as to be a starting reference for new players. The intended audience is assumed to have at least a rough concept of how a CCG works (e.g. no "This is a turn-based game" or explaining what a deck is).

3.4.1 General rules

Deck building:

Decks consist of 40 cards total, max 3 of any particular card.

One of three heroes can be chosen (icons next to deck name). These each have different starting resources (elements) and a hero ability that is available once per turn for 2 mana.

Objectives and setup:

Each player has a hero with 30 health. The objective is to reduce the enemy hero to 0 or less health.

Players start with a hand of 6 cards. There is initially a "mulligan"-phase where each player can select any cards from the first hand given to shuffle back into the deck and get new cards.

Resources:

The resources are mana and the three elements: Earth, wind and fire.

Cards have a mana cost (top left) and a set of element requirements (below mana cost). When playing a card, the cost is subtracted from one's mana that turn. Elements do not decrease from playing cards – the requirements just need to be met.

Players start with 1 mana available per turn. This increases by 1 each subsequent turn to a maximum of 10.

Players start with element values matching the hero they have chosen.

Once per turn, a player can increase one resource by 1. Mana is a temporary gain for that turn only; element increases are permanent.

Since the starting player gains access to higher mana values quicker, only the second player is allowed a resource increase on his first turn.

Cards:

Cards are either spells or minions. Spells have immediate effect, whereas minions also stay on the board. Any card that has damage and health values is a minion.

Players draw a card at the start of each turn. The starting player does not get to draw a card on his first turn.

Players can have a maximum of 10 cards in hand – any card beyond that is lost. Both players will know which card it was.

If a player has run out of cards but still has to draw, he will take 1 damage. Every time this happens, the damage taken increases by 1.

Minions:

Minions have damage and health. Attacking another minion means they both deal damage to each other. Any minion reduced to 0 health dies.

Minions can either move or attack once per turn, but not on the turn they were played.

The board:

The board is 5x4 spaces. Each player has half of the board (2 rows of 5 spaces each) reserved for his own minions. Minions cannot be played or moved into the opponent's 10 spaces.

Minions can attack enemy minions in their own column or in either of the neighbor columns. Only the closest enemy minion in each column can be attacked (i.e. minions in back row can be shielded by minions in front row).

Minions can attack the enemy hero if no enemy minions exist in their own lane. Heroes don't deal any damage to minions that attack them.

3.4.2 Keyword abilities:

While many cards explain exactly what they do, these common minion abilities are shortened to a single keyword for convenience.

Taunt: While this minion lives, none of the owner's minions without taunt can be attacked. This minion can be attacked from any distance. Taunt only has effect when this minion is not hidden behind another friendly minion.

Haste: This minion can act the same turn it was played.

Frenzy: This minion may attack twice every turn.

Sweep: When this minion attacks an enemy minion, any minions directly to the left or right of the target are also hit by the attack and do not get to retaliate.

Berserk: Whenever this minion is able to attack, it immediately performs its attack straight ahead.

Deathtouch: Any minion that takes damage from this minion immediately dies. Effects that prevent the loss of health also prevents this ability from working.

Free targeting: Can attack any enemy minion, regardless of position or taunts. Can still only attack the enemy hero if there is a direct line of attack. [Bugged: Cannot currently attack heroes.]

Reflect damage: Any minion that attacks this minion also suffers the effects of an attack upon itself, including special effects (i.e. deathtouch). The minion with this ability still takes damage itself.

Invulnerable X: The next X times this minion were to take damage, no damage is received.

Swift: This minion can move and attack in the same turn.

Root/Rooted: A rooted minion cannot move on its own but can still attack and be moved by spells.

Stun/Stunned: A stunned minion loses its turn.

3.5 Possible improvements

Iterating and expanding

I would have liked to regularly play the game with others and continually iterate on the design based on those experiences. While I'm quite satisfied with how the design and gameplay turned out, iterative design would have been a safe and relatively low-effort path to further improvement.

More options

The 202 cards currently in the game are just barely enough to provide a sufficient range of options. Hearthstone had 382 cards at launch and saw a vastly increased range of competitive play styles when this was increased further. More cards are obviously better (learning curve considerations aside and assuming the quality doesn't suffer) but particularly the next ~100 cards would have a significant impact in diversifying options.

Distinguishable cards

Quite simply, having names and artwork for all cards would both make the game prettier and the cards far easier to distinguish and remember. For further effect, they could also be given voices as is the case in Hearthstone.

4 IMPLEMENTATION

The project is available at

https://www.dropbox.com/s/gvkngxg9gcod0mi/Ejnar%20thesis%20code.rar?dl=0

(Non-digital readers can go to kortlink.dk/fu8f)

The file contains a project to be run with Unity, which is available for free download for non-commercial use. The code was developed and tested using Unity version 4.3.3f1.

Code and content declarations can be found in the Assets-subfolder. All graphical resources are in Assets/Resources. The rar file also contains a standalone build, which can be relevant for exploring multiplayer.

4.1 Structure

My implementation is comprised of the 5 main components: Controller, Menu, Render, Client and AI. This structure is essentially a model-view-controller pattern, with the model (Menu, Client and AI) being large enough that it has been practical to split it into 3 primary areas of responsibility.

Controller takes care of global coordination and variables ("Which deck is selected?", "Am I host?", "How many cards go in a deck?"), as well as loading in profiles and cards for the other components to access.

Menu handles all GUI-navigation and -display outside of an on-going game. This includes menu navigation, deck building/selection, establishing connection with another player and starting a game. Once a game has begun, responsibility transfers to *Client*.

Render is largely a set of assistant functionality that the other components can rely upon. It contains functionality for easily drawing the more complex GUI elements, such as cards and minions which are really composite GUI items with 10+ simple items (both text and graphic assets) layered on top of each other in correct order. These processes are reduced to simple functions, which *Render* also takes responsibility of running in the GUI. *Menu* and *Client* each construct their own simpler GUI objects, but the complex ones – Cards, Minions, LogEntries and all mouseover effects – all just have to be defined as data for *Render* to loop through and draw the objects correctly.

Client handles the execution of an actual game (read: a specific match of the overall game). This is the largest component as it spans GUI, game state, targeting rules, AI interaction and execution logic of the actual game.

AI contains all the functionality specifically associated with the AI. It relies on Client functionality to simulate the consequences of player actions and it is called by the Client to return a sequence of actions for the AI to perform, based on a given game state.

In the Unity editor, these 5 scripts are all attached to the main camera object, which is the only Unity GameObject I use – the rest of the game is run entirely in the code of these scripts.

4.2 Implementation Facts

This section is dedicated to explaining how the game is implemented. Any notes on why these choices were made or any potential alternate options have been relegated to the 'Implementation Discussion' section.

4.2.1 Data formats for defining content

I have designed a data format for defining cards and abilities in the game. These are written in txt files which are parsed and converted to meaningful game data by the Controller.

Cards:

Warlord of Terror

8,5,5,5

M,2,3

0,0

12_1_1,40_-2_-2

The first line is simply the name of the card.

Second line first describes the card's mana cost and then its requirements in Earth, Wind and Fire respectively.

Third line first lists the card's type (H = Hero, M = Minion, S = Spell) and then its damage and health if applicable.

Fourth line has data for card availability. The first value is whether the card is only available to a certain hero and the second is the rarity of the card in terms of unlocks. Neither of these are particularly relevant in my implementation at this stage, but I have made my implementation with the potential for both of them in mind, and just let Controller have a toggle to disable those restrictions.

The fifth line is a list of comma-separated abilities, with up to two underscores indicating the card's specific input values for an ability – e.g. whether an ability is to do 3 or 5 damage without creating two separate abilities.

The image of a card is stored as a graphical asset with a name matching the index of the card. In a hypothetical future version, with card names and images more consistently in place, it would be superior to let the images match a card's name rather than index, so as to make the card data file more easily maintainable.

Abilities:

19

13

Draw X cardZ when damaged

6,3,1,2

The first line is simply the ability number, used for identification. This information is redundant from a technical point of view, as the abilities are loaded and stored in order. Its function is instead to make the ability numbers easily visible to designers as they add content and to let *Controller* assert that abilities are being loaded in the correct order.

The second line is the basic effect of the ability, e.g. drawing cards or taking damage.

The third line is the description of the ability in text. Note the use of capital X/Y/Z: X and Y to indicate where an input value should be inserted in the final display of the string, and Z to indicate that a plural s should exist if X is greater than 1. These 3 capital letters are not used by any of the ability descriptions.

The fourth line contains the targeting of the ability – this along with the description is really the only thing that separates a concrete ability from a basic effect:

The first value shows the type of ability; that is, the circumstance in which it is even relevant. 6 here indicates 'upon receiving damage'. 0 is a passive trait (such as taunt), 2 indicates upon death, 3 at end of turn, 4 an immediate ability (either a spell or upon a minion being summoned), 5 when dealing damage, 6 upon receiving damage and 7 upon a friendly minion dying. 1 is obsolete; it was designed to be upon a minion being summoned, but that case is now included in 4.

With the circumstance in place, the next 3 values indicate the targets of the ability:

The first of these is the target group affected: 0 indicates just the minion itself, 1 a single specified target, 2 all adjacent minions, and 3 all valid targets.

The second is target alignment: 1 indicates that only friendly targets are valid, 2 that only enemy targets are valid, and 0 that the ability doesn't discriminate and all targets are valid.

The final target parameter is target type: 1 indicates minions only, 2 heroes only and 0 that both are viable targets.

Thus a line of 4,1,2,0 means that upon playing the card the effect is immediately applied upon a target enemy, with both minions or the hero being valid targets. Likewise in the example ability above (6,3,1,2): Upon receiving damage, the draw card effect is applied to all targets of type friendly hero.

The code is resolved such that players can only select 1 set of targets for a card. Any card with multiple single target abilities (e.g. "Give a minion +4/+0. Give a minion Berserk.") only faces the player with a single targeting decision and uses that same target for all of the abilities.

Abilities that target a player (e.g. "draw 2 cards" or "increase fire by 1") are modeled as targeting that player's hero. When such an ability allows no choice in which player to target (e.g. "Opponent draws 2 cards") it is set to target group 3 (all valid targets). This skips redundant targeting decisions for the player and frees up card design to feature player effects along with targeted abilities (e.g. "Destroy a minion. Opponent draws 1 card.").

Effects:

Effects is the final layer in defining content: All abilities in the game boil down to some variation of these 31 effects (e.g. taunt, damage, draw cards), which are all handled in *Client*.

New cards with any variation of targeting, execution circumstances, specific values and description for each of these effects can be defined through the txt files. Testing is largely reduced to just the effects and the methods of application and targeting, rather than the entire scope of permutations.

4.2.2 Save/load (player profiles, decks and unlocks)

The main menu has a field for the player to input a profile name.

A "[name].sav" file is generated with a list of cards that player has unlocked. The format is a sequence of lines of "[cardID], [amount unlocked]" ordered by cardID. Note that unlocking of cards is not an implemented feature although the Controller and data formats support it. Any cards that would otherwise be restricted are currently made available by the *Controller.cardsUnlocked* toggle.

Decks are saved in "[deckName].deck" files inside an "Assets\Decks\[profileName]\"-folder. The format is a single line with the cardID of the hero chosen, followed by lines of "[cardID], [amount]", ordered by cardID.

"lastSession.sav" stores the latest values used between sessions. Upon starting up a session, profile name, deck selection, AI deck selection and multiplayer connection IP all default to values from the previous session (if any exist). The file is updated whenever the player navigates from one menu to another.

4.2.3 Main menu

The main menu is a state machine, showing a different composition of GUI-elements based on an integer in the Controller.

Transitions between menus are not done by setting that variable directly but by calling the Menu.MenuSelection function, which performs the change while also saving changes and performing any necessary variable cleanup, based on the destination menu.

4.2.4 Network connection

The main camera is equipped with a Network View component. The controller saves a reference to this at startup and other classes perform their network operations through the controller.

When hosting or starting a single player game, a server is initialized on the Network View. All open connections (including a locally initialized server) are shut down when either returning to main menu or to the multiplayer choice of hosting or joining.

Games are joined by IP address. When a connection is established, both players immediately progress to a newly started game with the last selected decks.

Communication in-game is handled by transmitting calls to RPC functions over the Network View.

4.2.5 Game setup

The Client is built around network play from the ground up.

Upon entering a new game, or after one has been concluded, the NewGame function is called:

- All local variables are reset to the correct starting values
- The host randomizes the player numbers and transmits them to each player by the SetSelf RPC function.

- As a client receives its player number, it transmits player name, hero choice and deck (in pieces of cardIDs and amounts) to all players (itself included). This information is used to set up the basic starting values for each player.
- When the host has received a required number of cards from each client, these cards are shuffled server-side and the host sends each client a number of card draws matching their starting hands.
- Cards are sent by the host informing the client what has been drawn and the client removing that card from their deck locally.

4.2.6 Game state

The Client maintains a gameState. This is a class which contains all the information needed to describe a certain configuration of the game. It contains:

- A *board* of 20 minions. These are numbered left-to-right and bottom-to-top so that the bottom row is indices 0-4 and the top row is 15-19. Empty slots are represented by dummy-minions with cardID -1.
- Hand and deck of remaining cards for each player. These are lists of card numbers (integers) – looking up the IDs in the Controller allows the client to fill in the values and display the information.
- Health and resources for each player.
- How much mana each player is set to when their turn starts.
- Whose turn it is. Players are numbered 0 and 1. Turn -1 indicates mulligan phase.
- Whether each player has used their resource increase and hero power for the current turn.
- A list of minions which are dying and need to be cleaned up when the current action finishes resolving.

4.2.7 Player actions

The game is progressed and kept synchronized by the RPC function Client.Act, which takes a state and updates it with the results of a player or host action. In addition to state, it takes an action type (int), player number (of the player performing the action), and up to 4 arguments for the action. These are the possible actions:

- **DrawCard:** Host sends a number (arg1) of card draws to target player. A host action used to synchronize card draws as a result of abilities.
- **PlayCard**: Plays a card of specified ID (arg1). It's abilities are automatically resolved, with arg2 and arg3 defining targeting if necessary. Minions use arg4 for their own position.
- **Attack:** Minion A (arg1) attacks a target (arg2).
- **Move**: Moves a minion from position A (arg1) to position B (arg2). This is used both for the player action and as a host action in resolving movement abilities. arg3 indicates whether it is a player action, so those can appropriately deplete the minion's movement for the turn and be featured in the event log.
- **EndTurn:** A signal from a player to end his turn. If this matches the player whose turn it is, the match advances to the next turn and the appropriate logic is called to draw a card and refresh minions+mana.
- **HeroPower**: Uses hero power. arg1 specifies the ID of the hero and arg2+arg3 define any targeting, as when playing a card.
- **LoseCard**: Host action to remove a card from a player's hand. Used when a received card surpasses the hand limit.
- **Mulligan:** Player sends a list of cards to be mulliganed (arg1). Host processes this (including adding them back into the deck and sending out new draws. If both players have had their

mulligans processed, game is advanced to first turn. In order to pass the list of cards to be mulliganed as argument to a network function, it is encoded as a an integer with each digit representing a card and the values 1 and 0 indicating to mulligan or not respectively. As an example, the integer "100101" would mulligan cards 0, 3 and 5 in the hand.

- **HeroAction:** Player action to increase a resource type (arg1).
- **AddCard:** Host action to add a specific card to a player's hand. Used to resolve abilities that return a minion to hand.

This function and most others with any relation to targeting expect that any player numbers in the input have had 100 added to the value. Any type of target can be passed as an integer and the recipient functions use the added value to distinguish whether to interpret the target as a minion or player index.

4.2.8 GUI

Everything displayed to the player is drawn by code in theGUI layer instead of using game objects.

The Unity-inherent GUI.depth functionality appears to be utterly without effect. As a result, the ordering of GUI drawing matters: First drawn button is the one that player clicks interact with in that location, whereas the last drawn element in a location is the one that is visual to the player. As a workaround to this, the hero menu (the pop-up buttons than may overlap with the game board) has been moved from Client to Render and is being drawn (identically) both before and after GUI objects in its area.

Rendering cards

Render uses two classes for defining cards to be displayed: renderCard and mouseoverCard.

renderCards are defined by a cardID, a set of coordinates, a highlight value and occasionally a scale. This is the normal way of displaying the generic version of a card at a desired location, with appropriate highlight. Examples of usage include displaying a player's hand and showing the available cards in the deck builder.

mouseoverCards differ by only taking a cardID and possibly a minion. The position is always near the cursor (slightly translated so as not to block mouse clicks). They support both basic card representation by ID and representation of the updated attributes and abilities of a particular minion. When this feature is used for log events, the underlying minion data is cloned so that it persists even after the minions die and so the log will always display the minion as it was at the time of the particular event.

Log

Client defines the logEntry class, which contains information pertaining to an event, including copies of up to two involved cards (typically attacker/caster and target) and the source/target locations for highlighting when mousing over an entry in the log. The type of a logEvent is a string which Render case matches on in order to construct the correct line of text for the log and interpret the supplementing data correctly.

Client maintains a list of logEntries which Client.Act writes to whenever it is run on the active game state. Render continually reads this list in Client and draws the contents.

4.2.9 Targeting

Client contains several function whose primary purpose is to highlight valid targets and selections:

- In a neutral state, all cards which are affordable and possess a target (if required to play them) are highlighted.

- Selecting a minion highlights the possible targets for both attacks and movement.
- When a card is clicked, valid targets are highlighted if it is a spell and valid placement positions if it is a minion. Minions then proceed to ability targeting afterwards if relevant. Hero powers use the same targeting logic.
- Movement abilities require a secondary target before the action is resolved. This means that minions with movement abilities go through 3 targeting states before resolving.

Client has 5 variables that facilitate finding the right targeting state:

- *"selected"* indicates which card# in hand (if any) is currently selected and as basis for targeting.
- *"selectedM"* indicates which minion (if any) is currently selected and as basis for targeting.
- *"selectedT"* indicates that a movement ability is being targeted. It contains the ID of the first target.
- *"entryMinion"* indicates if the current ability is the result of a minion being placed and, if so, which position has been selected for that minion. This also facilitates removing the minion again if the play is canceled before all targets have been selected.
- "*heroPower*" indicates which hero power (if any) is being used as basis for targeting.

This is supplemented by the ClearSelection function, which cleans up variable data (returning to a neutral state) when the targeting sequence is completed or right-click is used to cancel.

There is no separate layer of verification when a target is selected – if the targeting algorithm has it highlighted as valid in the GUI, that evaluation is also trusted for executing the action.

4.3 Implementation Discussion

4.3.1 Limitations

- There is no cheat prevention in place. By editing the save files or transmitting the correct data over the network it would be possible to e.g. possess a deck that does not meet normal requirements or play cards that do not exist in one's hand.
- The design of the content definition format is optimized for easy card and ability creation. As a consequence of this prioritization is that there is no way to condense tooltips of multiple abilities on the same card – e.g. changing "Give a minion +4/+0. Give a minion Berserk." to "Give a minion +4/+0 and Berserk."
- The game is currently only designed and tested for one resolution (1920*1080). While the code is such that basic GUI dimensions generally scale to by screen width and height to any resolution, this may lead to visual artifacts such as GUI elements overlapping, pictures becoming pixelated and text overflowing or becoming too small to read. Other resolutions are not generally unplayable – just untested.

4.3.2 Known bugs

Gameplay bugs:

- There is a rare bug, mainly observed in multiplayer, where one player does not receive any deck information (including heroes). If, at the start of the game, a player sees either hero having 0 in all resources, the game should be restarted.
- The "Draw cards when damaged" ability does not yet work.
- The "Draw cards when a friendly minion dies" ability does not yet work.
- Minions with the "Free targeting" ability do not get the option to attack the enemy hero when there are no minions blocking. This is a GUI-bug and so does not hinder the AI opponent.
- The "Berserk" ability (automatic attack) is only checked at start of turn. It should also take effect if an effect gives the minion the ability while it is still able to perform an attack.

- The case of two minions with “Reflect damage” attacking each other is currently unhandled. The game may freeze if this were to happen.

Other bugs:

- The death of a minion will frequently cause an error message (standalone builds typically have these disabled) as Render appears to draw the non-existent minion. This has no practical consequence, as the GUI runs correctly again from the next frame onwards.
- When playing a standalone build, GUI components (particularly the log) are sometimes displaced, causing partial overlaps. This appears to be a Unity bug, as the behavior is in direct contradiction with the specified coordinates and the bug only occurs when making a standalone build, based on code that runs correctly in the editor.
- Switching to a new profile still allows play with the previous deck selections by hitting confirm without making a new selection.
- Current card layout does not allow sufficient space for card names.
- There is no handling of minions acquiring more abilities than can be displayed on the card. Some of this information will not be (easily) visible.

4.3.3 Potential improvements

4.3.3.1 Functionality and optimizations

Bug fixing

Many of the existing bugs, particularly those with a direct impact on gameplay, are quickly fixed as soon as feature completeness becomes a priority. The only insidious gameplay bug is the one of not initializing the game correctly. This is the most critical bug, once the game progresses from two players deliberately connecting to each other, towards a community with automatic match-making, but also appears to be quite challenging to fix. A faster/cheaper workaround would be to simply detect when it occurs and abort those games without result. Hearthstone has had success with something similar and its players will occasionally encounter games that are aborted immediately after loading, because the starting conditions are no longer met.

The GUI bugs require more effort to fix and are generally of lower impact. I would redesign the card layout to properly fit the names, and identify the bug that's causing the GUI-displacement in standalone builds. Beyond those two, the rest are non-critical and could even stand waiting until sometime well into an open beta.

More concerning than these specific bugs is that I do not know to which extent the lists are exhaustive. Timewise, the lack of dedicated QA resources has been the single biggest challenge in implementing a project of this scope within the allotted time. The code is generally simple enough that the actual fixes can be done quite swiftly when prioritized. The greater lack, which increased resources would make up for, is in systematic testing to identify where outstanding issues remain in the first place.

Robustness and cheat prevention

The robustness of this implementation is generally such that the user can reasonably arrive at the desired outcome rather than such that the user cannot reasonably arrive at an undesired outcome. The robustness of the implementation presents ample opportunity for improvement.

In the context of robustness for a wider release, cheat prevention measures would also be important. This would include:

- Profile information (e.g. cards unlocked) stored on a central server or at least encrypted and tied to a specific username.

- When players send their actions to the host, verify that they are allowed in the current game state.
- Change the information transfer such that no hidden information (opponent's hand or deck contents, own deck ordering etc.) can be gleaned from sniffing packets or memory. This would ideally involve transferring host responsibilities to a third party, such as a central server.

Further condensing effects and abilities

The data structures for resources and costs were simplified late in the project. As a consequence, it would be possible to reduce the number of ability effects from 32 to 29 (condensing the 5 resource abilities to just "Modify resource X by Y" "Modify all resources by X") as well as reducing the number of abilities from 70 to 67 (condensing the 6 resource abilities to "Give yourself Y resource X", "Give opponent Y resource X" and "Reduce all opponent's resources by X").

This has not been done because the plaintext data format defines abilities in a fixed ordering that is not easily refactored. The system works fine for adding content, but during early development it is poor at supporting the removal of abilities.

In considering the perspectives for practical handling of this issue, a large team would probably have developed a toolset for editing the raw data and be able to more easily perform refactorings in that manner. The simpler solution, for those without such resources, would simply be to temporarily tolerate blank dummy content at the freed indexes of Controller.abilities and in future development use those freed indexes for new abilities.

4.3.3.2 Feature additions

Card rarities and unlocks

When card acquisition has been declared out of scope (see 3.2.9), it is largely due to the implementation burden involved. Much of the existing code and design (data formats, profile saves, card display in deck builder) is already created with full regard for card acquisition. As soon as the resources exist for the GUI work involved, this would be a relatively easy and very impactful improvement on the game.

Deck sorting, filters and searches

Currently, the only sorting of the cards in the deck builder is by the order they are defined as data. An obvious improvement would be to automatically sort them by mana cost and add other sorting parameters, as well as filters and search options. This would both ease navigation and overview in the deck builder as well as placing fewer constraints on the way additional content is added.

Draft functionality

I would have liked to add a "draft" game mode, where decks are built from a constrained pool of cards and players are tester on their skill at making the best of what they have, as well as improvising with a new and unfamiliar deck against other decks which similarly cannot be expected to fit the habitual strategies. This game mode was a success in MTG and variations have also been adopted by HS and DoC when they were created.

5 GUI DESIGN

HCI was never an intended emphasis of this project. Yet a big flaw in that initial plan was that all the games I used for inspiration have used animations to communicate much of the meaning to the player, while I had no such luxury. Designing some simple yet functional solutions in this area therefore became a matter of core functionality rather than just expendable polish.

5.1 Providing feedback

The primary challenges were to highlight possibilities and display the consequences of actions.

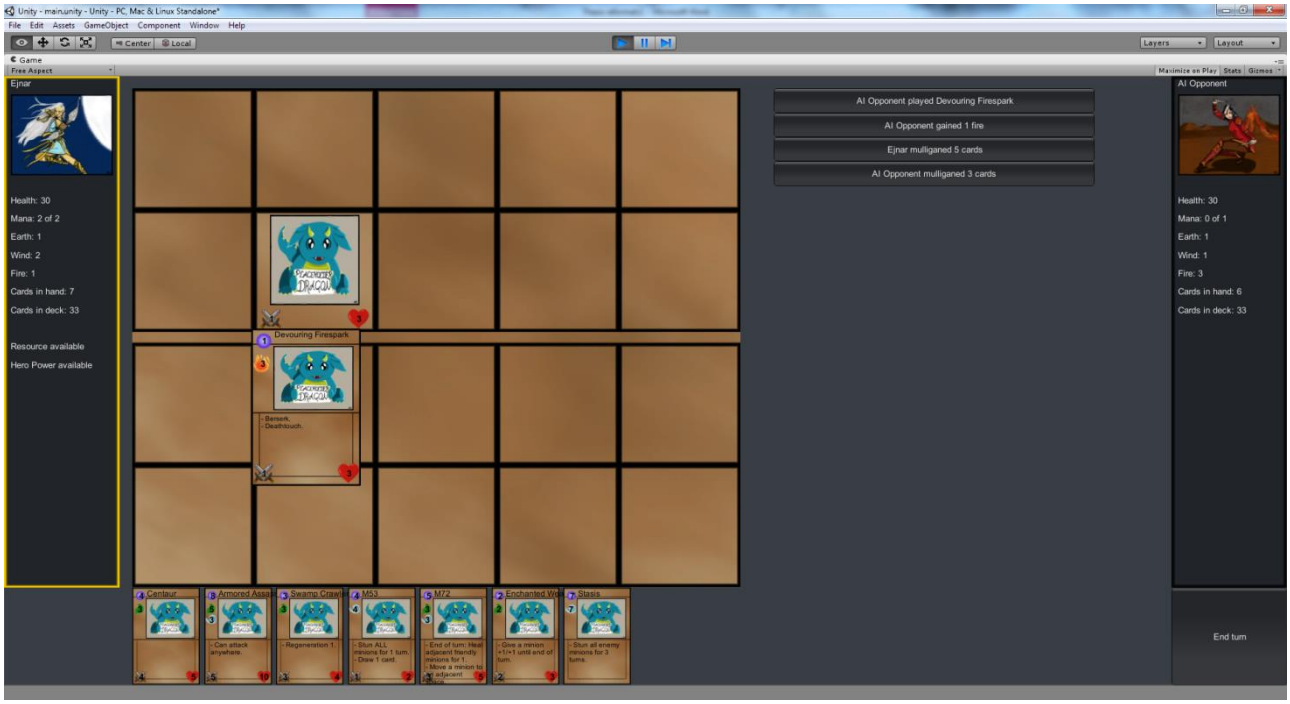
5.1.1 Highlights

The background elements for cards, minions and player menus are all produced with neutral (black), red, yellow and green borders. This allows cards, minions and players to all possess a highlight value, which can be set in the Client in order to automatically have Render draw the object so as to appear highlighted to the player. This functionality is used whenever a set of valid options/targets have to be communicated to the player; the yellow color is the default highlight, whereas red and green are occasionally used when it has been important to specifically indicate a hostile action or a selected/friendly target.

5.1.2 Mouseover info

In a number of situations, it has been important to represent an object in a space-limited manner while still making an expanded level of information available to the player when needed. The solution to this has been to implement a mouseover-functionality that lets Render display the full card info when pointing at the smaller object.

This is relevant both in the deck building and the log (where entries are just a single line) as well as for minions in-game. The latter have to be a smaller version of cards with just the most necessary overview-information, yet it is essential for a player to be able to point at it and both see the updated list of effects/abilities (including any buffs that may have been played on it etc.) as well as basic card information such as costs, that may be relevant if the player is considering returning it to his opponent's hand.



Elemental Command: Mousing over a minion

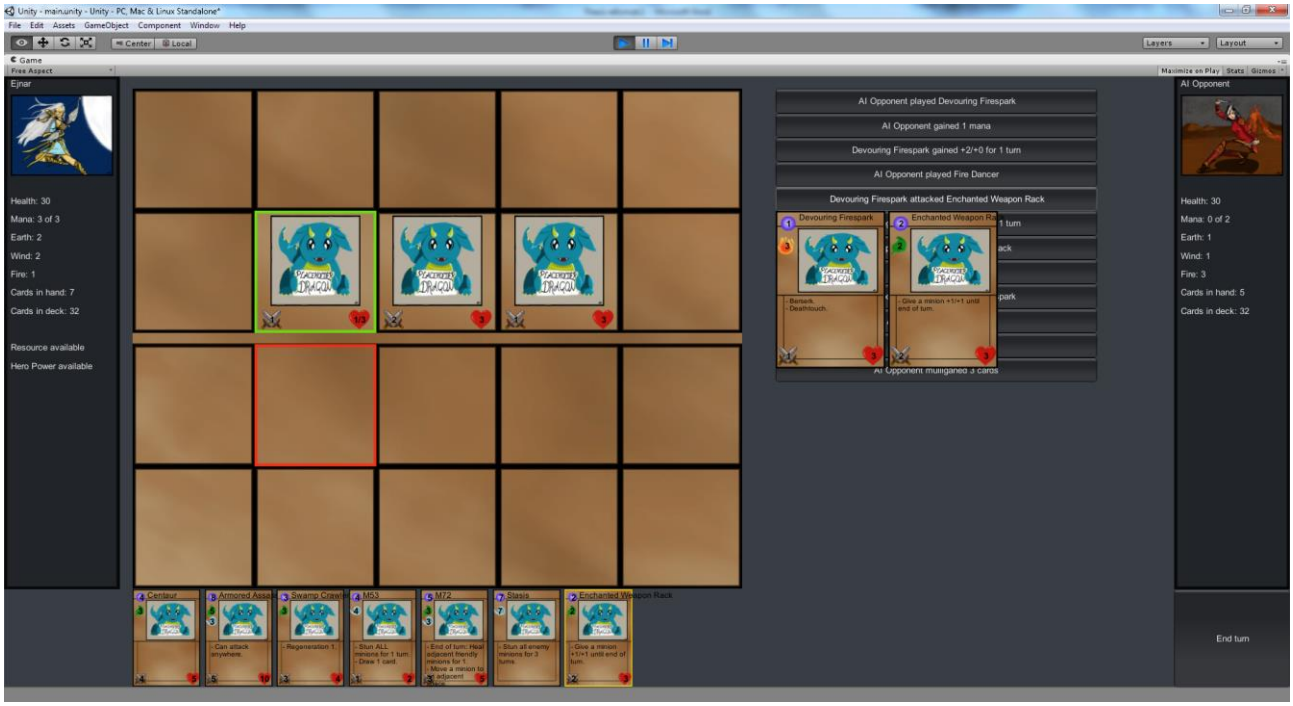
One current limitation is that the mouseover functionality, in the case of minions, only displays current information. This is what the player typically needs to know but, as previously mentioned, it can also be important to be able to read the card's basic values/abilities and these are not readily apparent (for either player) when that minion has been heavily affected by external effects (dispel, buffs etc.).

5.1.3 Event Log

The right side of the play area features a log which registers player actions ("A attacked B", "Player A played card X" etc.) and events ("Player A drew 1 card", "minion A took 1 damage" [from mass target effect] etc.). Since minions may very well have died and disappeared by the time a log entry appears, pointing at most log entries highlights the source (location) in green, the target (if any) in red and provides mouseover displays of the attributes of both involved minions as they were when the log entry was created.

Aside from just logging player actions, I felt a need to register certain specifics, such as damage taken and cards drawn, whenever those effects are caused by abilities (as opposed to basic events such as 2 minions attacking each other). This means that those effects are not easily overlooked and also provides a far better overview of the consequences when AoE spells are played.

Beyond listing all player actions in the log, the AI is also implemented with a 1 second delay between the execution of each action. The intention is to give experienced players a chance to keep track with its moves as they happen, rather than having to catch everything up from the log afterwards.



Elemental Command: Mousing over an attack event in the log

5.2 Artwork

I am no artist and have not personally made anything more complex than rectangular placeholders. The limited amount of artwork that the game does contain is all volunteer contributions by amateur artist Mia Louw Værndal, made to my specifications and design. Her contributions have particularly provided me with non-copyrighted icons as well as the card backs and frames to facilitate the previously mentioned highlighting.

It has been a deliberate decision to model cards as having images, since that ordinarily provides experienced players of a CCG with faster and more intuitive overviews. With a highly limited amount of images, most of the cards in the current version default to using a picture of the "placeholder dragon".

5.3 Limitations

These would have been nice to have but have deemed non-essential for a working product.

- The Deck builder lacks a visual indication that the mouse wheel is used to scroll up and down. This applies both in available and selected cards, depending on which side of the border the cursor is on.
- The in-game log lacks a visual indication that the mouse wheel is used to scroll up and down.
- Active games lack a quit button (and appropriate confirmation menu). Games are currently quit by rebooting the game.
- Multiplayer does not have its own deck selection menu. Instead it uses whichever deck was last selected in single player or deck builder.
- Few cards have any artwork and many minions have yet to be given a name. This work would aid in making them better memorable and distinguishable to the player.

6 AI DESIGN

6.1 Common strategies

Artificial Intelligence encompasses logic, probability, continuous mathematics, perception, reasoning, learning and action [31]. This field is what author of AI Game Programming Wisdom, Steve Rabin, terms "mainstream AI" or "academic AI" [32]. The focus of this section being to develop an effective and entertaining opponent in a particular video game context, the focus will instead be on "game AI".

Steve Rabin describes the contrast as such: "*Much of the disparity between game AI and mainstream AI stems from a difference in goals. Academic AI pursues extraordinarily difficult problems such as imitating human cognition and understanding natural language. But game AI is all about fun.* […] *In a sense, it's a shame that game AI is called 'AI' at all. The term does as much to obscure the nature of our endeavors as it does to illuminate it. If it weren't too late to change it, a better name might be 'agent design' or 'behavioral modeling'.*"[32]

Since I am deviating from academic AI into the territory of commercial games, many of the techniques used are either unspecified or proprietary secrets – as such, my sources on what has been done before are largely limited to media articles and interviews with the developers.

In considering which strategy is best suited, I will initially consider the merits of a range of machine learning approaches. These are some of the general strategies I have found most commonly recommended in literature or applied in games:

6.1.1 Reinforcement learning

Concept: A form of unsupervised learning, an agent has to choose from a set of available actions in a given state. Initially this choice is made randomly, but as a reward function rewards good outcomes and punishes bad ones, the agent should eventually begin selecting the best actions in each state in order to maximize expected reward.

Examples of usage in games: Research results in Checkers [33], Civilization IV [34], and a real-time strategy game [35]. The award-winning Berkeley Overmind (AI for the RTS Starcraft) used reinforcement learning for tuning some of its primary techniques [36]. In terms of commercial usage, a Monte-Carlo Tree Search was used for turn-based parts of the AI in the (hybrid real-time/turn-based)

strategy game Total War: Rome II [37] and reinforcement learning was also used for certain characters in Hitman: Absolution [38].

Suitability for this game: Poor. The most basic parts of a reinforcement learning model would be the set of possible states and the set of possible actions, but each of these is virtually infinite in the context of this game. I expect that achieving some degree of functionality would be possible by making sufficiently large generalizations on the equivalence of different states and actions, but this would from the very outset sacrifice the possibility for significant distinctions that human players are capable of making.

6.1.2 Bayesian networks

Concept: A form of supervised learning, Bayesian networks (of the type described by Steve Rabin in Game Programming Wisdom [32]), rely on causal inference: Given what it seen of an opponent's actions, it can form reasonable estimates for what to expect of the parts it has not yet seen, including a human-like vulnerability to feints by sudden shifts in strategy. Its awareness of causal relationships additionally allows it to maintain dependency graphs for desired outcomes and use those as a basis for choosing actions.

Examples of usage in games: I have found almost nothing on this front. Anders Walther, in a master's thesis on making real-time strategy AI better capable of adapting to human strategies, has chosen to use a Bayesian network for resource management [39]. It can be relevant in games with random elements (e.g. Monopoly) or hidden information (e.g. Stratego) for realistically guessing outcomes. Developers technically have perfect information though, so it would appear many choose to go for the simpler implementation of just faking a certain level of ignorance.

Suitability for this game: Very situational. It appears that the strength of this technique is in making a convincing and adaptive AI, more than in actually making it effective in the first place. If the scope of my AI was such that I had the time for a composite project, using elements of many different techniques, I believe Bayesian techniques, while a poor approach to general choices and strategy, would be very useful in convincingly predicting the opposing player's hand and strategy. As it is, slightly subpar estimates on these fronts can be made with a vastly simpler implementation.

6.1.3 Adversarial Search (Minimax, alpha-beta pruning etc.)

Concept: Minimax is a search algorithm which takes a utility function for calculating the value of its current position and then assumes that each player will at every turn make the choice that maximizes their own score. Alpha-beta pruning improves on this basic concept by cutting off the branches which don't represent an optimal choice for the active player, thereby greatly reducing the branching factor and increasing the depth of the search. Expectiminimax is yet another modification which handles elements of chance by weighing branches by their probability and evaluating the branches by their average weight.

Examples of usage in games: It sees application for particularly most turn-based two-player games – chess and tic-tac-toe being common examples [31]. Other games can also be modeled so as to fit this strategy – e.g. using the strategy in the single-player game 2048 by representing the random spawning as an adversary [40].

Suitability for this game: Good. Minimax is a tried and true approach that does not require training and which can be easily modified to my needs. My inability to achieve a single objective value for a game state would be a limitation: While the algorithm is pretty resistant to me using estimated values, it does mean that my pruning would likely be a bit more lenient. The alpha-beta pruning and expectiminimax variations both provide inspiration for handling challenges for my AI (branching factor and random chance), even if I should choose not to use them in their standard forms. The

strategy is at its core simple enough that it could easily form the back-bone of an AI without shutting off the possibility of using different strategies for specialized purposes along the way.

6.1.4 Metaheuristic optimization (e.g. Hill Climbing, Simulated Annealing, Tabu Search etc.)

Concept: In a search space that is too large to explore by brute force, these algorithms offer local search strategies for achieving results that often approach optimal with far less computation effort.

Examples of usage in games: I have found no games that rely primarily on this for their AI. Rabin [32] advises great care relying primarily on these techniques as they can be vulnerable to local optima in the search space as well as noise in the performance assessments.

Suitability for this game: Situational. While these techniques can be extremely powerful supplements when used within a limited scope, the difficulty in providing an exact and objective valuation of a state is a weakness. In using them as the backbone of my full AI, it would also be far from trivial to represent the search space in a way that can be easily traversed. Even given such a representation, I expect the great variation in results, based on which card/target combination or sequencing of events is chosen, would make it riddled with local maxima.

6.1.5 Neural networks

Concept: A form of supervised learning, artificial neural networks (ANNs) attempt to model neurological processes in the mind: "Neurons" are placed in succeeding layers, each neuron receiving inputs from neurons in preceding layers and/or the outside world and passing on an output to the next layer, eventually reaching the output layer and returning a result. Learning is achieved by tweaking the weighting of each input until the collective network produces meaningful results for a given context.

Examples of usage in games: An article from 2010 found "that the use of neural networks is still quite rare in mainstream commercial games and that the range of neural networks used is very limited" [41]. Examples of usage include the non-player drivers in Colin McRae Rally 2 and the trainable creatures that formed a central feature of Black & White. [41]

Suitability for this game: Poor. The wide range of input states coupled with the lack of a clear correct response makes for difficult training and the complex model makes for greater overhead costs than the other techniques listed here. ANN is also very much a black box solution, which leaves little opportunity to derive benefit from a deep comprehension of the problems at hand.

6.2 Existing AIs

Moving beyond conceptual AI techniques, I will now look into which practical solutions have been applied in the field of digital CCGs – specifically concerning the 3 games that have also served as my primary sources of inspiration in designing the game mechanics of Elemental Command.

6.2.1 Duel of Champions

I have found very little description of the AI processes used in DoC. Live Producer Samuel Jobin describes [42] that the AI is one of the main challenges that has prevented them from expanding their single player mode, that making it able to play all cards would be an incredibly complex process, and that their AI requires heavy retuning with each new expansion they release, in order to be able to adapt to those cards. It appears that they do have a challenging single player campaign, but achieve this mainly through changing the premise in favor of the AI (e.g. letting it start a match with minions on board) and only letting it play pre-defined decks that are simple enough for it to handle.

6.2.2 Magic: The Gathering

In the PC and Xbox 360 game MTG: Duels of the Planeswalkers, the AI is fundamentally based on lookahead to explore all possibilities [43]. Since, in the words of game studio CEO, "*if you approached the implementation of this system in a simple brute-force fashion, you'd need an orbiting spaceship filled with a multi-dimensional array of massively parallel processors built from alien technology sent through a worm-hole from The Future™*" [43], the main emphasis has then been on optimizations, simplifications and limitations of scope. This appears to largely have been very successful, based on both my own experiences with the AI and the instances the AI designer mention of how it seized on possibilities he himself had not considered before facing it. Despite this, it seems that these relatively good results are actually achieved with very limited lookahead: The designer explains that it is incapable combos that require more than 2 cards in sequence, and that, even with all their effort on optimizations, it still (as of the 2009 interview at least) cannot look as far as the opponent's next turn and consider that it might be beneficial to leave some creatures behind for defense.

Since its lookahead is implemented with some form of minimax (expecting optimal play from the opponent) my personal experience is that it can grow too predictable: Its cleverness to pull out cards in unexpected situations can be devastating as a beginner. It is a game where bluffing can play a large role however and it eventually becomes a significant weakness for the AI, that moves which are clearly to its disadvantage reliably inform the player that the AI holds a card to turn that particular situation around if he steps into it. The AI only plays a set of pre-designed decks and, while I have no confirmation that it cheats, I did have the impression that probabilities were manipulated to its advantage, and community members claim to have found reproducible steps to prove that it is aware of which cards the player is going to draw [44].

6.2.3 Hearthstone

The AI in Hearthstone was mainly designed to serve a tutorial function [45]. It is deliberately designed to represent an intermediate player and it can only play a narrow selection of pre-made decks that have been tuned to provide an appropriate curve of entry [45]. The AI does appear to serve that function very well and to do so without any sort of cheating, hidden knowledge or manipulating random outcomes. An interesting aspect of their choice of AI design is that they achieve very quick AI turns and are able to run the AI server side with a minimum of resources.

The simple AI did become a challenge when a single player expansion was later released. While it featured immensely challenging encounters, this was done almost entirely through changing the terms

of the game to favor the computer player, rather than through the skill of the AI. An Ars Technica review highlights the AI as one of the expansion's flaws, mentioning multiple examples of blatantly poor play decisions [46] and the community has discovered a hole in the AI that lets the player defeat some of the toughest fights by taking a minion, whose function is normally to destroy itself and everything on the board after one turn, and removing its ability, turning it into a harmless 0/7 minion that keeps the uncomprehending AI from ever playing a card [47]. The latter indicates that the AI is set up as a state machine with pre-programmed conditions, rather than actually testing what the outcome of an action will be.

6.2.4 Hearthstone "bot" (third party AI)

Over the summer of 2014, the Hearthstone community suffered significant problems with players using third party programs ("bots") to play for them automatically. A qualified estimate [48] suggests that a significant number of bots made it even to the highest rank of the game (for the top fraction of a percent of players). This gives a deceptively high estimate of their skill: The top bots would generally run the same 1-2 decks which the AI managed relatively well – both of them also being so fast and aggressive so as to end games one way or the other before the more complex options opened up to their opponents. Additionally, the path to the highest rank requires a certain net gain of wins – this can be difficult time-wise as a player (the luck factor of the game means that even 60% winrate against peers is dramatic), but the requirements are in practice much lower if one can play 24 hours a day, always stabilizing at a reasonably high rank when things go bad and just waiting for the right streak of >50% winrate from there to the highest rank. The author makes the estimate that the bots who reached the highest rank represent the luckiest ~5% of the fully active bot population.

Googling for communities devoted to Hearthstone bots and searching through their forums, I managed to locate a public GitHub repository with the source code of one of the popular ones (named Silverfish) [49]. Running it would have required a paid subscription to the bot and would also have violated the Hearthstone terms of use (and eventually gotten my account banned, when they started detecting and punishing the abusers). This was my best inspiration for a truly competitive AI. Skimming through the first 20,000 lines of the code, these were my impressions:

- Extremely verbose code: 2 C#-files (representing two different strategies), each of more than 30,000 lines, as well as a txt-file of ~45,000 lines for defining the data of roughly 850 cards. It seems the two main files are maintained in a nearly identical state, so that relatively few differences exist between them.
- Code seems to consider many extremely specific cases rather than generalizing them - e.g. going through a long list of cards, spending 25+ lines considering the impact of each in a particular situation. It's also largely uncommented and the language of its variables occasionally switches to German.
- It expects a large amount of user configuration for some rather specific choices (e.g. how low health should the enemy hero be at for me to attack him directly with a weapon rather than focusing on his minions) rather than deciding those automatically in the situation.
- Part of the reason for its length is that it cannot hook directly up to the game logic to evaluate outcomes, but has to possess a lot of that logic on its own to be able to make valid decisions.
- It appears to largely be a heavily tweaked state-machine which evaluates the impact of a lot of specific opposing plays but does not truly simulate outcomes in its attempts to predict opposing moves. Long-term maintainability seems to be a huge issue, as I do not believe it adapts very well to new cards being added until the developers can add a similar level of tweaking for each new card in each relevant case. The code contains clear TODOs about adding support for understanding the logic of certain (common and very relevant) cards that were added a few months prior to me downloading the code

- Later code (after the first ~5,000 lines) is better structured and seems to extend the initial design with a selection of lookahead options of various cost for the situation, as well as some logic for depth, breadth and cutoff.

In summary, it seems like a verbose mess that is painful maintain - yet it also seems to speak to the power of implementing rudimentary lookahead and having a state machine that actually considers the full scope of possibilities the game offers for the cards the AI is playing. Additionally it appears that there is a lot to be gained by tweaking all sorts of minor variables optimally for the particular deck being played, and that this AI leaves that entire iteration process to the player.

6.3 The AI

6.3.1 Strategy

My AI is mainly based on adversarial search. It searches through the possibility space of actions and perform lookahead with the assumption that the opponent will make optimal moves (Minimax). A utility function is designed to make a swift and rough estimate of how beneficial a particular state is, and in order to achieve good depth in my search it make certain simplifications and prunes all card play actions that do not immediately produce a net value gain. It also narrows the number of observed branches at deeper levels, from the reasoning that it is more important to consider a wider range of possible paths for the immediate future, than to slightly more accurately estimate how that play is going to perform many turns down the line.

The sub-problems (e.g. evaluating a state to a value) generally use simple heuristics that are designed from my comprehension of the game mechanics. These typically make significant tradeoffs in precision and optimality to prioritize efficiency. These values (e.g. how good is a specific minion) are often so subjective and situational that they are hard to predict without the possibility branches of the full game, and so my strategy is that a relatively gross assessment will be preferable if the gains in efficiency instead allow for greater depth of simulation.

6.3.2 Structure

I define two simple classes: The *Action* class contains a set of input parameters for Client.Act, which can be easily used by Client to replicate the specified player action. (Client.Act is the function that takes a player action and updates all players with the resulting game state.) The *Branc*h class stores a game state and the list of actions that lead to it.

The Client sets up an initial branch to be a copy of the current game state and passes it to the AI script's MakeDecision function, which either goes into mulligan or normal turn logic as appropriate. The mulligan decision process will be elaborated in section 6.3.3. The normal turn logic has the following main steps:

- Play cards
- Perform attacks
- Increase a resource (if not done yet)
- Perform lookahead on the (seemingly) best branches by calling MakeDecision recursively
- Pick the highest valued branch and return

Playing cards:

MakeDecision takes a list of branches as input (though this list is of size 1 when initially called by Client). For each branch, every card in the active player's hand is tried – and on every valid target for those that require single target selection. "Trying" a card/target combination involves simulating the

outcome of the action on a clone of the existing branch and evaluating the resulting state. As long as more cards can be played from the resulting branch, this process is iterated. Every branch encountered along this iteration (including the originals where no cards have been played) is passed on to the next step.

The auxiliary functions CostCheck and CostPrep respectively verify that a card is affordable (accounting for a possible resource increase, if still available this turn) and perform any required resource increase action before trying that card.

The AI uses a simplification in the placement of minions: Since the combinatorics of being able to play every minion on every open space, with every possible target, are extremely nasty (potentially 100+ ways to play a single card) and the positioning makes no difference in state value until several turns down the line, I use the EvalPosition heuristic (elaborated in section 6.3.3.6) and just place minions on the first of the positions that score best, instead of trying all the (typically ~10) positions.

When trying plays, the AI immediately prunes every play that does not lead to an increased value compared to the prior state. Since cards themselves have a slight value, this also means that there must be some net gain to a play. That this pruning happens immediately, means that not only is it not passed on to the later stages, it is not iterated on with more plays either. This increases efficiency further, though at the cost of not perceiving the occasional ingenious combo that can only be pulled off by shooting oneself in the foot along the way.

Attacking targets:

For reasons of efficiency, attacking is done on a per target basis: Rather than exploring every ordering of attacks (an O(n!) problem), each of the possible targets is considered and the AttackMinion heuristic (section 6.3.3.4) estimates the most efficient order of attacks for eliminating it. This strategy reduces the complexity of the problem massively but also at significant cost, since optimizing for each target in isolation is almost guaranteed to provide suboptimal results in the full context.

When no taunts exist, the player is also a valid target, using the AttackPlayer function. This is rather more simple than AttackMinion, since there is no retaliation cost and no cap on how much damage is usefully spent in that direction. Choosing the player as target essentially just attacks with all remaining resources that possess a direct line of attack.

If more minions are able to attack once a target has been eliminated, the branch is continued so that additional targets can be eliminated in the same turn. As with playing cards, all intermediate branches (e.g. those that do not make the attack) are also continued.

Lookahead:

MakeDecision, aside from the list of branches, also takes a maximum depth value as input. While this has not yet been reached, and no player has yet won, the highest scoring N branches are found and given as input to the next depth of MakeDecision.

N is determined by the maximum depth and another parameter which defines the extent of branching at each level. This extent of branching is not independent of depth: The algorithm prioritizes greater width of possible actions in the short term, and performs more aggressive pruning the further down the line to facilitate exploring their consequences deeper

6.3.3 Heuristic design

Whereas the previous section mainly considered how to efficiently explore the possibility space, with little consideration for insight into the underlying game mechanics, these underlying heuristics have to rate what makes for a good decision in the game and do so in simple enough terms that it does not impair overall efficiency. As such, what follows is a combination of efficiency considerations and game systems design.

All constants in this section are designed to be easily configured in order to enable further tweaking of the AI. I have chosen initial values based on my comprehension of the game mechanics. These should provide a functional outset but can almost certainly be improved upon if a coordinated effort is made to iterate on them.

6.3.3.1 Evaluating a state

Solution:

Code location: AI.EvalState

A state is evaluated to a score, by considering the following parameters:

- Hero health
- Resources
- Value of individual minions (see EvalMinion, section 6.3.3.2)
- Cards in hand
- Remaining cards in deck
- Whether the player has a board presence (minion on the board)
- Whether the player has denied his opponent a board presence

For most parameters, values are generated for each player and the opponent's score is subtracted from the active player's to arrive at a final value.

- Hero health is evaluated at 2.0*sqrt(n) or infinitely low when <= 0.
- Resources are evaluated at 1.0.
- Individual minion values are covered in section 6.3.3.2.
- Having no minions on the board subtracts 2.0 + 1.0 * [turn count to a max of 10].
- Having cleared the enemy board adds 2.0 + 1.0 * [turn count to a max of 10] (These two are not calculated for the opponent. Values are currently identical to the line above, but the two scenarios deliberately allow separate tweaking).
- Cards in hand are evaluated at 3.0 for the first 3 cards and 2.0 for every card beyond that.
- At turns 10+: Remaining cards in deck are valued at (sqrt(n) – draw damage) * 1.0.

Background:

The challenge here is to identify which parameters play into measuring one player's advantage over the other, and to which extent the AI can afford to compromise on them for efficiency.

In my initial planning, I envisioned a far more complex solution, which was less of a snapshot and had greater consideration for how the current position was prone to change – e.g. if the immediately available attack opportunities would favor one player significantly. However with the choice of an adversarial search strategy, I decided that sufficiently deep lookahead would account for such factors better than any rough estimate, and the emphasis shifted almost entirely to speed, in order to enable such deeper lookahead.

The parameters and their evaluation functions are relatively simple:

- Resources and minion attributes are simply lumped into a weighted sum.
- Hero health and remaining cards in deck are both extremely expendable resources until they suddenly get low enough to be dangerous. A weighted root function ensures that having more is always slightly better and that the weighting of each point/card increases as the overall value gets lower.
- Cards in hand are generalized to a constant value per card. Since a card advantage is no stronger than the difference in cards that can actually be played before the game is brought to an end, cards beyond a certain number are weighted lower. This was found to prevent the AI

from making too great sacrifices in tempo in order to make the most from particularly its cheaper early-game cards.
- Having a board presence or denying the opponent from having one are both given a value. This includes both a constant value and the maximum mana per turn, in order to let these bonuses scale up as the game is more progressed.

The AI runs through cards in hand, cards in deck, minions and the total number of minion abilities all in linear time. Hero health and resources are handled in constant time. The design sacrifices consideration of minion positioning and attack opportunities as well as assigning an individual value to each card in hand, based on overall strength and situational applicability. All of these, while worthwhile in an AI more akin to a state machine, were here deemed to be factors that lookahead would account for sufficiently well, that they would not be worth the additional cost.

6.3.3.2 Evaluating a minion

Solution:

Code location: AI.EvalMinion + AI.EvalMinionAbilities

- Damage is evaluated at 1.0 per point.
- Health (current) is evaluated at 1.0 per point.
- Passive abilities are typically evaluated at roughly their budget value (from card balancing).
- Non-passive abilities are not considered individually and are valued at 3.0.

Background:

As with the general state evaluation, this heuristic is all about speed. Damage and health are quickly lumped together into a value and any special traits that the minion possesses (passive abilities, e.g. taunt) add additional bonuses.

These bonuses are mostly based on each trait's cost in a card's budget, but some are weighted by how well a minion's other traits support them – e.g. invulnerability is more valuable, the higher the damage of the minion and reflect damage is better the higher the health. Anything outside the common passive abilities (e.g. deathrattles or "draw a card each time another friendly minion dies") is not considered individually but generalized to a value of 3.0. This generalization is both for reasons of efficiency (since many such abilities' values depend on the current board situation) and also because I lacked the development time for any additional granularity in this area.

6.3.3.3 Mulligans

Solution:

Code location: AI.MakeDecision - search "//Mulligan logic"

- The AI checks for which turns it (currently) lacks a minion or damage spell with matching mana cost.
- Mulligan each card with higher mana cost than the second such turn (of no appropriate play), unless the card is both the cheapest on hand and is cheaper than the deck average.
- Mulligan each card with high enough resource cost that it cannot be played on the turn matching its mana cost.

Background:

A simple yet effective heuristic would have been to simply mulligan everything above a certain mana cost (likely 3 or 4). This was a suitable baseline to find cheap measures to improve upon.

The objective of a mulligan decision is generally to ensure that the player has what he needs to start the game at strong pace and be able to utilize his resources every turn. Based on this, the baseline has

the obvious limitations that it neither observes how *many* of the upcoming turns the player has something useful to play (The entire hand could be mana cost 3), nor that particular deck's odds of actually giving the player something better than what he already has.

My heuristic operates with the concept of counting "gaps" - turns in which an initial hand has no plays with an immediate impact on the board control (that is, either a minion or a damage spell). My experience from high-level play in Hearthstone (which was the main inspiration for my balancing scheme) is that it is quite viable to skip a turn 1 play, but that skipping both the first turns can lead to a very rough start. With EC having twice the number of cards in the starting hand, tolerating no more than 1 gap seems a reasonable goal. Another useful concern is in optimizing the worst-case outcome by holding on to the cheapest minion in the hand, even if it does exceed the desired cost.

The algorithm runs through the full deck, calculating the average cost, then through the hand to determine what the cheapest minion is, and when the second gap will occur (assuming the player doesn't sacrifice any permanent resource increases to boost a particular turn's mana). In the event that a card in hand has a mana cost higher than the second gap, it is mulliganed unless it both has the cheapest mana cost in hand and that cost is lower than the deck's average mana cost. Since EC features cards that are exceedingly strong for their mana cost, at the price of increased resource requirements, the algorithm additionally mulligans any card with a resource cost greater than what can be available at the turn corresponding to the mana cost.

The algorithm establishes how many cards exist at each mana cost, and what distribution of elements the deck favors. These metrics are not currently used but, given more development time, the intention was to improve estimates on the probability of getting a cheaper card than the one mulliganed away, and on how accessible a particular requirement is (a higher element requirement being more accessible if it's of the sole element of focus in a deck).

With a large starting hand it is also a viable strategy to hold on to specific particularly valuable cards, regardless of their cost. Without a predetermined deck for the AI, I have no easy way to identify such cards and have opted not to do so. This is mainly a loss for combo decks, which have been built with very specific intentions for card interactions and typically have one or two cards that are essential pieces in facilitating the winning combo. This particular type of deck was however also one that I from the outset deemed it computationally infeasible for my AI to play competently (as they typically require awareness of the full possibility space) and so it's a limited loss.

The heuristic runs through the hand a constant number of times and through the deck once. Its runtime is linear in the size of the deck.

6.3.3.4 Predicting the human player's hand

Solution:

Code location: AI.InitPlayerHand + AI.PrunePlayerHand

- The opponent's hand is represented as his entire deck.
- Before the opponent plays cards, his hand is temporarily pruned to the number of each card that he can afford to play.

Background:

A crucial part of what makes a good player, is anticipating the opponent's moves, limiting their options and not overextending into a position where the right card in the opponent's hand will lose you the game. Replicating this awareness is essentially a question of performing lookahead on the opponent's hand, but it was a major challenge to model a qualified guess for the contents of this hand, rather than letting the AI blatantly cheat by knowing the contents of its opponent's hand at all times.

The naïve solution would be to consider all cards that the player can afford at the time. While this might have been technically feasible with the current set of cards (~200), it is an expensive solution and I really did not like making the algorithm scale to the size of an ever expanding card base (MTG has grown to ~12000 unique cards by now).

Another strategy would be to choose a number of dummy or baseline cards to present generic plays (4 mana minion, best AoE available at 6 mana etc.). While this could be done a lot cheaper, it would both require a great deal of development time and it would make the AI very primitive in its considerations, as well as completely unable to adapt to many strategies that deviate too far from these baselines.

The strategy I ended up choosing was to instead let the AI cheat by having secret knowledge of its opponent's deck, rather than hand. This has the advantages of the naïve solution, while bounding the number of cards by the deck size. Whereas having direct knowledge of the player's hand would both be a significant unfair advantage to the AI as well as being a rather blatant and visible way of cheating, knowledge of the deck has fewer such consequences: Without direct knowledge of the hand, both the visibility of the cheat as well as the advantage gained will be greatly reduced. Additionally, part of being a good player is to quickly identify the kind of deck you are up against and from that deduce what other cards and strategies you are likely to encounter from your opponent. This is possible both since certain cards indicate that other cards are more or less likely to be a valuable inclusion for a deck, and since the core strategy of any new strong deck quickly becomes widely known, and the variations of that deck typically only deviate with a few cards from player to player. As such, this cheat simulates this deductiveness for the AI to only a slightly more accurate extent than what is possible for an expert player. In terms of the play experience, this could well turn out to be a net gain, as the AI actually adapts far more competently to the player's strategy. The main weakness of this choice is that it is impossible to fully surprise the AI with a peculiar card choice in the deck, even if it's the player's first game after including that card.

With this strategy chosen, the AI essentially represents the opponent's hand as his entire deck and predicts from there. In order to reduce branching, the hand is temporarily reduced to only the cards that the player can afford to play (and the number of each particular card that he can afford to play), while cards are played. Since resources are most limited early in the game, this reduction is most effective exactly at the stages when the deck is at its largest.

Presently, this representation distorts the results of EvalState by placing an incorrect number of cards in one player's hand. This is generally not a huge issue since EvalState is mainly used for relative comparisons but it might distort a few results – particularly around the moments when the temporary pruning happens or is undone. Given more time, it would be preferable to iron out this kink by also keeping track of the true number of cards in the player's hand.

A potential weakness of this strategy is that it may lead to overly careful play, because the opponent is always believed to be holding all the possible counters. This is particularly relevant, since an expert player trait is knowing when to hold on to an advantage by playing it safe, as well as knowing when one is so disadvantaged that gambling, on the opponent missing a particular counter, represents the best remaining chance of winning. Given more time, it would make sense to add a system for considering how improbable an exceptionally effective enemy play is and vary the AI's willingness to gamble, based on how favorably it evaluates its current game state. In that same vein, it would make sense to maintain an awareness of cards that are more or less likely to be in the opponent's hand – both cards that are known for certain due to a minion being returned to the hand, as well as reducing likelihood of a player possessing a specific card if he failed to play it at a time when it represented a far superior course of action to what was instead chosen.

6.3.3.5 AttackMinion

Solution:

Code location: AI.AttackMinion

- Finds all minions which are able to attack the chosen target.
- For each possible attacker, calculate effective damage (capped by target's health) and the value cost of attacking the target.
- Sort attackers by damage/cost.
- If target has invulnerability, attack with lowest cost attackers until it is all gone.
- Pick most efficient attackers until combined damage is enough to kill target.
- Rather than account for all possible special abilities (e.g. deathrattles that do damage to all other minions), all attackers with unknown abilities are flagged as "special".
- For each "special" minion, a candidate branch is generated with it as the first attacker before an efficiency-ordered attack sequence is applied to the rest of the target's health total.
- An attack sequence is considered valid if it can be simulated, without the target or any attackers being prematurely dead, and the target is dead at the end of it.
- AttackMinion takes two branch lists as arguments. If the player has no more minions who are able to attack, the result of an attack sequence is added to branch list A, otherwise the result is added to list B.
- The main AI keeps looping through attack procedures until list A is empty, and then moves on to the next stage with list B.

Background:

Though one of the longest algorithms, it is conceptually pretty straightforward: The most efficient attackers are sent to kill the target and, if needed, the cheapest attackers are sent in first to remove any invulnerability. Achieving this level of simplicity comes with two significant tradeoffs:

Firstly, abilities are not considered in full detail. Accounting for more detail isn't so much a matter of computational expense as of having the development resources to build a sufficiently large state machine into the heuristic. The solution I came up with is for the algorithm to simply try combinations involving all minions with unrecognized special abilities and then measuring if the resulting state value turns out superior. Neglecting to account for those abilities in full detail also requires the algorithm to verify that nothing unexpected happened during an attack sequence and discard those sequences for which it did.

The second disadvantage is that the algorithm only accounts for the direct state value loss of attacking with a minion, and not the opportunity cost of e.g. using a minion with way more damage than is necessary. This especially applies when using the "cheapest" minions to strike off the layers of invulnerability. In cases when the most efficient minion does not deal enough damage to kill the target by itself, the greedy approach of continually adding the next most efficient attacker is also prone to suboptimal results because the final attacker to be added may provide more damage than is needed and thus have a lower effective efficiency than another minion further down the priority queue.

6.3.3.6 Eval position

Solution:

Code location: AI.EvalPosition

- For each attack target from a position, calculate the net value of the attack being made.
- If minion has the initiative (e.g. it has haste) pick the most valuable attack, if not pick the worst as the value of the position.
- Ties are broken in favor of the positions with wider reach (those away from the edge lanes).

Background:

Being able to estimate the value of a given position will enables the AI to greatly reduce branching by not having to try all of them. Performing this estimate represented a significant challenge for three reasons:

- The consequences of choosing a particular position mainly manifest typically don't manifest until the opponent's next turn or even later.
- The heuristic had to be fast enough to run up to 10 (the number of positions on each player's side) times for each card play considered, meaning that local lookahead would very likely be too costly.
- I have no personal high-level play experience in any game where this type of positioning exists. Additionally, the design has been changed sufficiently from my DoC inspiration, so as to render my intermediate experience with that game, and any strategy articles I may find from it, mostly useless in this context.

I have found no easy way around those challenges and have instead accepted that a rough and potentially flawed estimate is better than no estimate.

My approach is a form of single-turn lookahead. It considers the targets that a minion in that position would be able to attack, and which minions would be able to attack it. If the minion played has the initiative (e.g. due to haste) then the value of the position is the net outcome of the best attack, if not then it is the value of the worst. As with AttackMinion, calculating the net value change of an attack does account for deathtouch, reflect damage, invulnerability etc.

The main limitation is that the AI performs card plays before attacks and so the position evaluation is usually going to be partially obsolete by the time the opponent has his turn and his opportunity to attack. Beyond that, there are many additional factors that are not fully accounted for, due to a combination of speed considerations and prioritizing development time:

- The algorithm only accounts for the single most costly attack - not a sequence of attacks.
- Special targeting effects, such as sweep, berserk, free targeting, stun and swift minions with a free spot to move to are not accounted for.
- Taunt is accounted for when the placed minion has the initiative, but on defense it is not considered whether a friendly taunted exists and whether it is going to be killed with sufficiently few resources that the worst attacker is still available.
- I add a slight bias towards middle positions to encourage the AI to seize the lanes which provide wider choice for attacks. This serves its basic purpose, but a more ambitious approach would distinguish between those minions who are served by that wider range of options and those which are better off hidden away in a corner, with as little exposure to attacks as possible. Additional worthwhile considerations would be positions that allow hiding behind friendly minions (again difficult to consider because the enemy might be able to kill the shielding minion and still have the worst attacker unused), placing minions that are valuable for their abilities in the back row by default (to facilitate later shielding them with other minions) and to account for the minions with abilities that affect adjacent friendly targets.

This heuristic serves its basic purpose and does so very efficiently but, given additional development time to go into a greater level of detail with the AI, this is one of the areas that could be improved the most. Particularly if the general structure of the AI could also be made more flexible, so as to tolerate attacks and card plays in any order, this heuristic could come a lot closer to making optimal rather than merely adequate choices, without sacrificing too much efficiency.

6.3.3.7 Resource choice

Solution:

Code location: AI.MakeDecision - search "//Pick resource"

- For each card in hand, for each resource, sums any lack in that resource to meet the card's requirements. The resource most lacking in that manner is chosen.
- If no resource is lacking for the cards in hand, the process is repeated for the remaining deck.
- Any further ties are broken in favor of the element most prone to reduction by card effects.

Background:

This heuristic is for the cases when no resource increase was needed for the plays performed in a particular branch. Rather than splitting each such branch in 3, it performs a simple approximation on which of the resources is going to be most useful down the road and increases that one.

The heuristic defaults to fire if the choice ends up in a tie (e.g. if the entire deck can be afforded). Since multiple fire cards come with the cost of reducing one's own fire resource, it is the element most prone to reduction over the course of the game.

6.3.4 Evaluation

6.3.4.1 Limitations

Section 6.3.3 already details some areas with potential for improvement. While all of these are functional, there are currently three gaps in the AI that I have had to declare outside the scope of this project in order to arrive at a finished product in time:

- The AI does not perform any movement actions with its minions. In the great majority of cases, it is indeed optimal to prioritize attacking, but it does limit it in certain since it neither moves around to prioritize more valuable targets or to preserve its own valuable minions when they are exposed. In terms of both developer time and processing power, this is by far the most expensive of these 3 gap issues to handle but, given the time, I believe a useful approach could be designed by generalizing to the 4 cases: "movement to avoid threats", "movement to shield another minion", "movement to dodge defenders" and "movement to threaten enemy minions". Utilizing the EvalPosition function (similarly to when placing minions), it seems very feasible to achieve effective behavior in this area as well, without increasing branching too heavily.
- The AI currently does not play cards with movement abilities. Expanding it to handle these is a relatively simple edition – particularly if either movement logic has been implemented or the change of placement is simply performed in a vein similar to when placing minions. I estimate that this would be less than a day's work, but this edge case was deemed non-essential to demonstrating the conceptual functionality of the AI.
- The AI is currently completely unaware of both players' hero powers (those that can be activated for 2 mana once per turn). I expect an easy strategy for implementing these would be to add the heroes as cards to the deck (since the abilities exist on those cards) and expanding the card play logic to match on cards of type "H" for a different type of RPC call, as well as tweaking the places in the code that depend on number of cards in a deck (e.g. EvalState). Again, this is not a difficult expansion, but was not considered essential to creating a working AI.

6.3.4.2 Testing

Formalized user testing was deemed a low priority for this project. What follows is my personal observations about the AI, as well as limited testing with the intent of providing rough sanity checks as to its functionality and performance, using a minimum of resources.

I have built two decks – one aggressive and one control-oriented. I have had regular games against the AI during development and tweaking as well as ~5 additional games at the end. My personal impression is that both the heuristics and overall AI structure/strategy are working as intended – I was occasionally surprised by nasty plays I hadn't seen coming and was impressed to find that it scored the occasional victory against me – particularly when it draws a good hand with the aggressive deck. I am still winning far more than 50% but, given that I have played Hearthstone at the highest levels (the top league being < 0.5% of a playerbase of millions of human players), I did not take it for granted that it would score any victories at all without me holding back.

Going for a more objective sanity check of its performance, I let the AI run 10 games with the aggressive deck against an opponent who would just pass his turn and benchmarked how fast it won those games (see table to the right):

Game#	AI victory in turn	Opponent turns passed
1	5	4
2	5	4
3	4	4
4	5	4
5	5	4
6	5	5
7	5	4
8	5	4
9	5	5
10	4	4

AI turns till victory in 10 games with no opposition

It appears that an opponent should expect to lose unless meaningful defense is played within the first 4 turns. These numbers only have any meaning in the context of the mechanics so, while I as the designer find them to indicate a cruel and highly competent pace of attack, a more accurate impression of their meaning is found by the reader booting up two instances of the client and seeing how they are able to perform with manual control.

Note that the AI did not have the advantage of knowing that it would face no opposition, and that it was still observed to make plays that prioritized board control over plain damage (e.g. low-damage deathtouch minions) during these sessions. For the purpose of replicating this experiment, the idle opponent was set to the identical deck ("Aggro" in the profile "Ejnar"). This can influence the experiment in terms of which defenses the AI is expecting to play around.

In testing the efficiency, I have had the AI output the number of branches processed and I then cranked up the depth and branching limits to see swiftly they were processed and how deep lookahead was feasible. This was done on an Intel Core i-5 4670 4*3.40 GHz processor, without utilizing more than 1 core.

The numbers vary a lot, but my high-end desktop PC appeared to be processing to the order of 10,000 branches per second. I had been concerned that my calculations would be too expensive to facilitate even a few turns of lookahead, with heavy pruning from turn to turn, but these fears proved utterly unfounded: I have set the game to look up to 12 turns ahead, carrying hundreds of branches over from most turns along the way. I have run depth 14 and more than double the branching limits with only seconds of processing time for the AI to decide on a turn. The reduction to depth 12 and hundreds of branches (both of which should be more than enough for virtually all purposes) is an estimate of which settings can run on older laptops without any noticeable processing time.

In running the game, be aware that the AI process has finished by the time the first move is taken, and that the 1 second delays between moves are at that point just added for usability purposes.

6.3.5 Potential improvements

6.3.5.1 Quality assessment

Meaningful quality testing

This desire applies to both game systems (e.g. balance) and the AI. The challenge to doing this hasn't so much been a lack of time, as a lack of applicable test users. For perspective, in Hearthstone, visual cues that an opponent may only have a few months of experience are considered hints of an easy match – and that sheer number of hours played doesn't nearly convey the full magnitude of stress testing that those mechanics and AIs receive: The millions of players press each other to new heights during competition, guides are published to assist learning and take credit for inventing the best strategies, the decklists of the top performers in the world are continually available for all the others in the community to try and improve on, and anyone can go and watch their favorite international superstar player stream live while discussing all the decision processes he or she goes through during they play. It should be noted that, even in those environments, it routinely happens that new and superior strategies are developed from component pieces that have all been available for everyone to see and apply for several months.

Dragging in test users from the street and attempting to make AI or game mechanic results relevant for current commercial games (even with the resources to give testers 2-20 hours of experience before experiments) would be an exercise in futility unless applied to very specific situations (e.g. first impressions, usability testing or the AI as a learning tool) that are not really core to my goals – and even then I would still lack a baseline for what skilled play in Elemental Command actually looks like.

The way I would instead have liked to have been able to perform my testing is to have a development team of several people who, over the course of development, all accumulate experience and try to surpass the others. If the resources exist, this can later be expanded to also inviting great numbers of players for one or several stages of beta testing (as Hearthstone, MTG and Duel of Champions have all done in tweaking their games). Whether 6 people during office breaks or 2 million people online, what I perceive as the essential component, to facilitating meaningful tests, is the shaping of a competitive environment where skills and experience can be nurtured over a period of time. Given any such environment, running meaningful tests (e.g. game balance for the range of competitive strategies or AI for its performance against skilled players) mainly becomes a matter of giving the (by no means unbiased) participants an incentive to play at their best – e.g. by having an office competition of who can manage the best winrate in 10 games against the AI. This setup can easily be repeated and constrained as necessary, to test specific aspects of the AI. This environment additionally provides documentably skilled players to use as a baseline for comparing certain AI decision processes (e.g. mulligans) or a new player's rate of learning.

Once that requirement of a competitive environment is met, there are of course still different qualities of tests: Cheap internal testing on by a motivated development team should be sufficient for the early iterations but, given the resources, a large-scale beta test using comparatively unbiased players would clearly be a better solution for particularly either research or late-stage iteration purposes.

QA work

As with the implementation in general, the AI could benefit from having the QA resources to more systematically search through it for potential bugs.

6.3.5.2 Performance improvements

Heuristic tweaking and expansion

The heuristics in section 6.3.3 detail several areas where the logic could be further improved.

Multi-threading

Given significantly more development time, it would make sense to parallelize the AI to utilize multiple cores and enable either greater depth or greater detail in the heuristics.

Scaling to system performance

Rather than scaling the AI to a fixed depth, a better long term solution would be to limit it by wall clock time. This ensures better utilization of the time available, regardless of the system it is being run on.

Training the AI

Many evaluation functions rely on constants (even if that constant is 1.0) to judge the value of specific elements, such as a minion's health points or the value of a card in hand. While I have used my comprehension of the game systems to set initial values I found reasonable, chances are that AI performance can be improved significantly by tweaking these values and perhaps even making them contextual based on the style of the deck that is being played. These constants are all made available for easy tweaking at the top of the script – this would mainly be relevant in the context of machine learning by matching the AI against versions of itself with slightly mutated constants, to arrive at stronger values.

Expanding mulligan logic

The mulligan logic currently does not perform any lookahead and thus only uses a tiny fraction of the available resources. There is definite potential for improvement here by letting it consider multiple branches.

Loosening restrictions on order of play

Given that the efficiency measures on the AI go far beyond what is necessary, it may be beneficial to loosen some restrictions in order to perceive a greater part of the possibility space. I particularly believe great improvements could be made if actions could be generalized such that playing cards, using hero powers, attacking with an available minion or moving a minion (according to a relevant movement strategy) are all possible actions in any order. This would likely require some additional pruning measures within each depth level, but would enable far more complex play sequences and eliminate many of the remaining blind spots of the AI.

Trying multiple resource choices

Again given the excess of computational resources, it would make sense not just to make the resource choice at the end of a turn a greedy choice, but to allow it to perform lookahead based on multiple resource choices – particularly since the great majority of decks/situations will only have 1-2 resources as useful candidates for increase.

6.3.5.3 Feature expansion

Scaling difficulty

Scalable difficulty could be implemented by forcing the AI to pick suboptimal plays when one exists within an appropriate margin of the best found action sequence – e.g. picking an action sequence that leads to a 10-25% lower value increase than the best known solution. The percentage intervals would be based on the difficulty setting. For the best gameplay experience, it should also be less willing to scale down its performance as soon as the player takes the lead.

Such a technique would have useful commercial applications and, as a side benefit, this would have provided a more easily testable parameter for the AI, by gathering data on how well the players progress in AI difficulty and how enjoyable they find that process and the resistance they are facing along the way.

Building and drafting decks

While I believe I got halfway to this point and achieved an AI which does a decent job with most any deck you throw at it (combo decks being a predictable exception), it would have been fun to also add a mechanic for letting it build its own decks within certain defined archetypes (e.g. earth-based control deck) or with a random archetype for fresh challenges. I don't think it would be particularly difficult or time-consuming to make a heuristic which does an acceptable job at this, but the problem would be a lot more interesting if the available card pool was expanded to twice its current size, so there would be a wider freedom of expression and possibility.

An interesting sub-problem is that this could also enable the AI to draft its own decks, under the same constraints that the players would face in a that game mode. Perhaps more interestingly, this could make for an engaging single player draft mode with a progression of difficulty, by making the AI start equally or more constrained than the player and gradually improving the resources it has to choose from, for each deck the player defeats. Almost regardless of how challenging the AI itself is (assuming no complete holes in its logic), it could be a challenging whole to see how many decks of escalating difficulty that a player is able to defeat before suffering e.g. 3 losses – the drafting premise would even offer great replayability and constant hope of circumstances aligning to enable a new personal best. It would however be necessary to present an incentive for performing well with every deck drafted (e.g. a cost for a session or rewards based on average performance) to prevent players from just giving up and restarting every time they do not get an outrageously lucky draft.

It's worth noting that the problem of making good deck selections in a draft context appears to have been solved relatively well for Hearthstone by heartharena.com, which released December 23rd 2014. Personal experience, as well as many testimonies (e.g. [50]), suggests that it performs at a level where even the best players in the world finds its considerations valuable, even if they typically choose to make a few choices different than advised. I have not been able to find a description of how it is implemented but my hypothesis is a basic value for all cards and a massive state machine for modifying those values to fit a specific context, such as having many cards that would synergize well with a particular pick or lacking cards that match that mana cost and/or function.

In considering the results reported (e.g. 8 or 12 wins), it's worth pointing out that the context is players who are paying in real or in-game currency for an opportunity to build a deck and play it until they have lost 3 games or won 12 and their run is done. They receive significant rewards based on the number of wins they end at and it is thus an extremely competitive environment, facing human opponents – many of which use guides written by professional players in determining which cards are generally more valuable to pick. Heartharena relies on similar lists, but weighs a lot more context and synergy into its recommendations.

7 PROCESS

From the outset, I knew that the scope of this thesis would be very ambitious – particularly with only 2-3 full-time months budgeted to produce an implementation of something that full professional teams have typically spent a year or more producing and testing (Hearthstone was 14 developers for 18 months [45]). A benefit of this volume and relative modularity (design, implementation and AI) of work was that I knew from the outset that I might have to reduce ambitions in one area but would still be left with a project of significant scope and content.

In comparing my planning with the actual process, these are some of the factors that I failed to initially account for:

Design

- Shortly before signing the thesis contract, I was fortunate enough to find a different supervisor with greater regard for game design as a field of computer science. This enabled me to place greater emphasis on this side of the thesis, but also increased the workload in this area by about 3 weeks.
- The stronger emphasis on design wound up moving a greater part of design (including source gathering and report writing) to the start of the project – postponing start on the implementation by ~6 weeks compared to the plan but also freeing up significant time at the other end.
- My supervisor and I came to the decision that we could not take knowledge of CCGs for granted in a reader and so I invested the additional time to produce the 'Anatomy of a CCG'-section, to ensure a mutual understanding of concepts and terms for my design section.

Implementation

- I had accounted for the time saved in having a simple visual side of the implementation, but had failed to account for the lack of user feedback caused by forsaking animations. Several weeks of additional GUI-work had to be invested in the implementation, in order to ensure that players would get enough feedback to understand what was going on.
- My ability design and structure surpassed expectations: Aside from its advantages during content production, I wound up spending a single week on implementing an amount of functionality that I had feared would take closer to 4.

- Working alongside other commitments was not generally a huge problem, as it allowed me more time to consider my solutions between work. I had however failed to account for the human factors of this: There was of course some efficiency cost in having to re-establish mental context after periods of intense coursework stealing my full attention. The more significant cost though, was that I occasionally allowed myself to be intimidated at the prospect of digging back into a complex process when initial overview and momentum had been lost - I effectively wasted several weeks with very little useful productivity in this manner. I became better at identifying and powering through this perceived challenge in the later stages of the project.
- I made the mistake of not from the outset designing around game states and the ability to update them. This would have been a safe abstraction to use in almost any scenario and would have spared me an extensive redesign in order to facilitate the ability for the AI to simulate outcomes of actions.

AI

- In writing the AI, it was initially extremely challenging to maintain an overview and get started. Inspired by the Hearthstone bot, my initial work was way too specific and did not fully grasp how much work was being done for me, by virtue of simply trying out possibilities. The first 2 significant segments of it were 150 lines of elaborate state machine approach that proved completely redundant in the final result (although serving to get me started and having a context to write better things). To be fair, a major part of this challenge was also me lacking a grasp of how many options I could expect to be able to try. I have now twice in the context of AIs made the experience that my careful estimates for what is computationally feasible tend to be surpassed by several orders of magnitude by present day CPUs.

Misc.

- I experienced a massive unforeseen distraction as what was supposed to be a high-quality ergonomic chair turned out to have been a major source of continually building tension and pain for more than a year. I identified/eliminated the source of the tension by early August and a determined rehabilitation effort gradually reduced the problem and restored my work capacity over the following months.

8 ACKNOWLEDGEMENTS

- Mia Louw Værndal for providing the artwork for this game, as well as her contributions to coming up with names for the cards.

- Kayla Friedman and Malcolm Morgan of the Centre for Sustainable Development, University of Cambridge, UK for producing the Microsoft Word thesis template used to produce this document.

9 FUTURE WORK

The sections on design, implementation and AI already cover the solutions that have been reached and their possibilities for additional improvement. This section seeks to explore these solutions' perspectives for practical impact and application:

9.1 Design

Design objectives

Based on academic as well as industry sources I have reached a set of design objectives for my work in this area. While there are typically significant elements of personal experience and gut feeling involved in design decisions, I believe that these design objectives can serve as a valuable outset for anyone wishing to make their own product within this genre. As the rationale behind each objective is readily apparent, the objectives should also be able to serve as valuable inspiration, even in cases where modification to fit other priorities is desired.

While the objectives are focused on the particular genre of digital CCGs, it should be noted that most objectives and sources are so widely applicable that it does not take a lot of modification for them to serve as inspiration for the priorities of a new project in a wider context – e.g. "turn-based multiplayer games".

CCG inspiration and decision template

The specific design I have reached can of course in itself either be realized in a more ambitious production and/or serve as inspiration for other projects but, perhaps more importantly, the decision process in each of these areas is readily apparent: In many of the most common areas of the genre, I have evaluated typical solutions, considered useful objectives and arrived at one of many possible paths for seizing the benefits while minimizes undesired aspects. Aside from the inspiration gained from one specific game, as well as the act of highlighting many of the most important decisions when making a game in the genre (and the impacts of those decisions), I believe my design process can serve as a valuable template for arriving at good design decisions in this genre, even if these end up wholly different from my own choices with this particular production.

9.2 AI

Better and cheaper AI for mainstream use

While the performance of this AI has not been fully evaluated, it appears that the lookahead model is more adaptive and robust to exceptions than the models either Hearthstone or Duel of Champions use. The MTG: Duels of the Planeswalkers AI appears to initially have been founded on similar concepts to what I use, but later versions have seen users complaining about what they perceive to be blatant cheating.

Whether one AI-method is superior is not a clear-cut decision, since other AIs may have been chosen based on different priorities. An example of this would be how the Hearthstone AI was in part chosen for its low computational costs [45], though it should be noted that my AI can be reduced to less than one percent of its current resource usage and still retain its 2-turn lookahead - or 4+ turns if a narrower model of branching is used.

One area where I do believe this system to be superior is in terms of development costs: This method appears to require very little card-specific code, since the AI just tries plays by simulating their outcome. Whereas the models used by both Hearthstone and Duel of Champions appear to have to adapt to ever expanding card pools, the adversarial search model (in combination with my hand prediction method) essentially means that neither computational, nor development costs scale with new cards being added.

Single player modes

Duel of Champions and MTG: Duel of the Planeswalkers launched with some form of single player campaign, while Hearthstone is adding single player campaigns through dedicated expansions (one of which is released so far). Advances in abilities and/or development costs of AI could provide developers increased freedom in terms of both quality and quantity of content provided. A sufficiently generic and adaptable AI approach can also make the barrier of entry lower for smaller developers wishing to provide this form of content.

Teaching and training tool

A variation on single player content is that AIs already see widespread use in tutorial modes. A better performing AI would increase the scope of possible teaching modes, whereas an AI that is capable to competently play most decks it is handed could be useful for training/decktesting against certain strategies.

Bot development

Aside from usage in the actual game, these methods could be applied for either producing a monetized bot (which is generally against the terms of use for multiplayer games, including all those used as examples in this thesis) or achieving insight into which mechanics and player options significantly complicate the act of making a competitive bot for a game in this genre.

Rapid simulation

The ability to rapidly simulate games with some extent of competent play could be extremely value for a number of purposes:

- Letting decks mutate and seeing which winning combinations turn up can be used as a cheap but effective first pass on the game balance of current and added content.
- It allows testing theoretical decks against both specific and general opposition.
- It provides an effective tool for testing and improving AIs – e.g. for creating new and improved iterations of an AI, whether by trying the effect of just changing some values, by automating that and letting the winning value sets mutate, or by testing the concrete effect of switching to a new heuristic for a particular problem.

10 BIBLIOGRAPHY:

10.1 Academic:

[6] Association for Computing Mastery
"Curriculum Guidelines for Undergraduate Degree Programs in Computer Science", Dec 20th 2013
http://www.acm.org/education/CS2013-final-report.pdf
Accessed Jan 30th, 2015

[9] Tracy Fullerton, 2008
"Game Design Workshop", 2nd edition
Morgan Kaufmann Publishers, Massachusetts USA

[10] Stefan J. Johansson phD, 2009
"What Makes Online Collectible Card Games Fun to Play"
Digital Games Research Association [online]
http://www.digra.org/wp-content/uploads/digital-library/09287.37268.pdf
Accessed Jan 28th, 2015

[13] Katie Salen and Eric Zimmerman, 2004
"Rules of Play: Game Design Fundamentals"
The MIT Press, Massachusetts USA

[16] Jesse Schell, 2008
"The Art of Game Design: A Book of Lenses"
Morgan Kaufmann Publishers, Massachusetts USA

[31] Stuart Russell and Peter Norvig, 2010
"Artificial Intelligence: A Modern Approach", 3rd edition
Pearson Education Inc., New Jersey USA

[32] Steve Rabin, 2002
"AI Game Programming Wisdom"
Charles River Media, Massachusetts USA

[33] Richard S. Sutton and Andrew G. Barto, 2005
"Reinforcement Learning: An Introduction" [Online]
The MIT Press, Massachusetts USA
http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html
Accessed Jan 28th, 2015

[34] Stefan Wender and Ian Watson, 2008
"Using reinforcement learning for city site selection in the turn-based strategy game civilization iv"
CIG'08: IEEE Symposium on Computational Intelligence and Games
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.205.6462

[35] Manu Sharma, Michael Holmes, Juan Santamaria et. al. , 2007
"Transfer Learning in Real-Time Strategy Games Using Hybrid CBR/RL"
International Joint Conference on Artificial Intelligence [online]
http://www.ijcai.org/papers07/Papers/IJCAI07-168.pdf

[36] Computer Science Division, UC Berkeley
"The Berkeley Overmind Project"
http://overmind.cs.berkeley.edu/
Accessed Jan 28th, 2015

[39] Anders Walther, 2007

“AI for real-time strategy games”

Master thesis at the IT University of Copenhagen, 2007

http://www.itu.dk/image/edu/theses/pdf/AndersWalther06.pdf

Accessed Jan 28th, 2015

[41] Darryl Keith Charles, Stephen Mcglinchey, 2004

“The past, present and future of artificial neural networks in digital games”

Watkins, Al-Dabass D (eds) Proceedings of the 5th international conference on computer games: artificial intelligence, design and education

https://www.researchgate.net/publication/229034750_The_past_present_and_future_of_artificial_neural_networks_in_digital_games

Accessed Feb 2nd, 2015

10.2 Other sources:

[0] Sabotage Times

“20 Years Of Magic: The Gathering, A Game That Changed The World”, Titus Chalk - July 31st 2013

http://sabotagetimes.com/life/20-years-of-magic-the-gathering-a-game-that-changed-the-world/

Accessed Jan 27th, 2015

[1] Wizards of the Coast

“Pro Tour Gatecrash event information”

http://www.wizards.com/magic/tcg/events.aspx?x=mtg/event/protour/gatecrash13

Accessed Jan 28th, 2015

[2] Ubisoft

“Might & Magic Duel of Champions available now on Steam”, Anne Lewis - Nov 7th 2013

http://blog.ubi.com/might-magic-duel-of-champions-steam/

Accessed Jan 28th, 2015

[3] Blizzard Entertainment

“Hearthstone at BlizzCon – Fireside Chat Panel Highlights”, Christina Sims – Nov 8th 2013

http://eu.battle.net/hearthstone/en/blog/11524460/hearthstone-at-blizzcon-fireside-chat-panel-highlights-08-11-2013

Accessed Jan 28th, 2015

[4]Kickstarter.com

“HEX MMO Trading Card Game”, Cryptozoic Entertainment – funded June 7th 2013

https://www.kickstarter.com/projects/cze/hex-mmo-trading-card-game

Accessed Jan 28th, 2015

[5] CNET

“Virtual goods revenue to hit $7.3 billion this year”, Don Reisinger – Nov 15th 2010

http://news.cnet.com/8301-13506_3-20022780-17.html

Accessed Jan 28th, 2015

[7] Wizards of the Coast

Magic: The Gathring - Gatherer

http://gatherer.wizards.com/Pages/Search/Default.aspx?action=advanced&format=+%5B%22Vintage%22%5D

Accessed Jan 28th, 2015

[8] Wizards of the Coast

Magic: The Gathering – Banned and Restricted Lists, revised Jan 19th 2015

http://www.wizards.com/Magic/Magazine/Article.aspx?x=judge/resources/banned

Accessed Jan 28th, 2015

[11] Extra Credits

“Perfect Imbalance - Why Unbalanced Design Creates Balanced Play”, James Portnow – Aug 1st 2012

http://www.youtube.com/watch?v=e31OSVZF77w

Accessed Jan 28th, 2015

[12] Gamasutra

“'Iterate fast' and other design lessons learned from Hearthstone”, Alex Wavro – March 21st 2014

http://www.gamasutra.com/view/news/213701/Iterate_fast_and_other_design_lessons_learned_from_Hearthstone.php

Accessed Jan 28th, 2015

[14] Richard Garfield
"'Luck in Games' talk at ITU Copenhagen", Sep 13th 2013
https://www.youtube.com/watch?v=av5Hf7uOu-o
Accessed Jan 28th, 2015

[15] Svetozar Gligorić, 2002
"Shall We Play Fischerandom Chess?"
B.T. Batsford Ltd., London, England

[17] Gamasutra
"Evaluating Game Mechanics For Depth", Mike Stout - July 21st 2010
http://www.gamasutra.com/view/feature/134273/evaluating_game_mechanics_for_depth.php?print=1
Accessed Jan 28th, 2015

[18] Blizzard Entertainment
"Hearthstone Day QA", March 14th 2014.
https://www.youtube.com/watch?v=h7WuRAqWAbw
Accessed Jan 28th, 2015

[19] digitaltcg.com
"Deeper into the mechanic: Resource System", June 16th 2013
http://www.digitaltcg.com/dtcg-comparison-%EF%BD%9Eresource-system%EF%BD%9E/
Accessed May 1st, 2014

[20] Robert Goudie
"Vtes in Los Angeles: Garfield Reminisces on the Jyhad", July 2001
http://archive.today/PqhIs
Accessed May 2nd, 2014

[21] Wizards of the Coast
"Tests of Endurance", Mark Rosewater - Oct 9th 2006
http://www.wizards.com/magic/magazine/Article.aspx?x=mtgcom/daily/mr249
Accessed May 2nd, 2014

[22] Hex Entertainment

“Hex TCG How to Play”

http://hextcg.com/howtoplay/

Accessed Jan 28th, 2015

[23] Gamasutra

“GDC 2012: 10 tutorial tips from Plants vs. Zombies creator George Fan”, Tom Curtis - March 9th 2012

http://gamasutra.com/view/news/165359/GDC_2012_10_tutorial_tips_from_Plants_vs_Zombies_creator_George_Fan.php

Accessed May 2nd, 2014

[24] Blizzard Entertainment

“Hearthstone's Card Balance Philosophy”, Eric Dodds - Jan 16th 2014

http://us.battle.net/hearthstone/en/blog/12383909/hearthstones-card-balance-philosophy-1-16-2014

Accessed May 6th, 2014

[25] LiquidHearth

“Creature Valuation: How Blizzard Assigns Costs to Cards”, user “curi” - October 18th 2013

http://www.liquidhearth.com/forum/hearthstone/360-creature-valuation-how-blizzard-assigns-costs-to-cards

Accessed May 6th, 2014

[26] Rob Weir

“First Move Advantage in Chess”, Jan 27th 2014

http://www.robweir.com/blog/2014/01/first-move-advantage-in-chess.html

Accessed Feb 4th, 2015

[27] Blizzard Entertainment (reposted on blizzpro.com)

“Ben Brode Talks 'The Coin'”, Ben Brode – Sep 12th 2013

http://hearthstone.blizzpro.com/2013/09/12/ben-brode-talks-the-coin/

Accessed Feb 4th, 2015

[28] Wizards of the Coast

“The Year of Living Changerously”, Mark Rosewater – June 2nd 2008

http://archive.wizards.com/Magic/magazine/article.aspx?x=mtgcom/daily/mr334

Accessed Feb 2nd, 2015

[29] Kotaku

“Here's How Rare Hearthstone Cards Actually Are”, Steve Marinconz – June 5th 2014

http://kotaku.com/here-are-the-actual-hearthstone-legendary-card-drop-rat-1586156518

Accessed Feb 2nd, 2015

[30] United States Patent and Trademark Office

Trademark Electronic Search System

http://tmsearch.uspto.gov/

Accessed Jan 22nd, 2015

[37] AiGameDev.com

“Monte-Carlo Tree Search in TOTAL WAR: ROME II’s Campaign AI “, Alex J. Champandard – Aug 12th 2014

http://aigamedev.com/open/coverage/mcts-rome-ii/

Accessed Jan 28th, 2015

[38] AiGameDev.com

“Reinforcement Learning-based Character Locomotion in Hitman: Absolution”, Alex J. Champandard – Jan 14th 2013

http://aigamedev.com/ultimate/presentation/hitman-reinforcement/

Accessed Jan 28th, 2015

[40] stackoverflow.com answers

“What is the optimal algorithm for the game, 2048?”, user “nneonneo” – March 19th 2014

http://stackoverflow.com/a/22498940/3746354

Accessed Jan 28th, 2015

[42] Reddit.com

"My name is Samuel Jobin and I am the live producer for Duel of Champions at Ubisoft. Ask me anything about DoC…", Samuel Jobin – Jan 8th 2014

http://www.reddit.com/r/duelofchampions/comments/1upy2n/my_name_is_samuel_jobin_and_i_am_the_live/cekisma

Accessed Jan 28th, 2015

[43] Wizards of the Coast

"Duel of the Planeswalkers: All about AI", Patrick Buckland – June 22nd 2009

http://archive.wizards.com/Magic/magazine/article.aspx?x=mtg/daily/feature/44

Accessed Jan 28th, 2015

[44] GameFAQs forums

"CONFIRMED: This game CHEATS LIKE HELL!!! (PROOF INSIDE!)", user "Danteh" – July 11th 2012

http://www.gamefaqs.com/boards/671863-magic-the-gathering-duels-of-the-planeswalkers/63379500

Accessed Jan 28th, 2015

[45] Brian Schwab

"AI Post Mortem: Hearthstone – Heroes of Warcraft", presentation at Game Developers Conference 2014

http://www.gamasutra.com/view/news/224101/Video_Building_the_AI_for_Hearthstone.php

Accessed Jan 28th, 2015

[46] Ars Technica

"Curse of Naxxramas is an uninspiring Hearthstone expansion", Kyle Orland – July 27th 2014

http://arstechnica.com/gaming/2014/07/impressions-curse-of-naxxramas-is-an-uninspiring-hearthstone-expansion/

Accessed Jan 28th, 2015

[47] Youtube channel "BuB Cha Cha"

"Noth just a little disappointed God", Jul 31st 2014

www.youtube.com/watch?v=phPj815Iig4

Accessed Jan 28th, 2015

[48] hearthpwn.com forum

"Bots - An estimate of their skill level, prevalency and what that means for Hearthstone", user "justsa1yan" – Aug 5th 2014

http://www.hearthpwn.com/forums/hearthstone-general/general-discussion/16914-bots-an-estimate-of-their-skill-level-prevalency

Accessed Jan 28th, 2015

[49] github.com repository of the Silverfish bot

github.com/noHero123/silverfish/, unidentified developers

Accessed Oct 7th, 2015

[50] Reddit.com

"My first ever 12-0 - Priest (all picks from Heartharena)", user "Morgaith" – Feb 3rd 2015

https://www.reddit.com/r/HearthArena/comments/2unw4o/my_first_ever_120_priest_all_picks_from/

Accessed Feb 5th 2015