Thesis

DIGITAL DECK BUILDER

Barnaby Whiteman
43948378

Supervisor
Dr. Joel Fenwick

Submitted for the degree of
Bachelor of Engineering

In the field of
Software Engineering

November 5, 2018

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

5th November 2018

Prof Michael Brünig
Head of School
School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072

Dear Professor Brünig

In accordance with the requirements of the Degree of Bachelor of Engineering (Honours) in the School of Information Technology and Electrical Engineering, I submit the following thesis entitled

"Digital Deck Builder"

The thesis was performed under the supervision of Dr Joel J. Fenwick and tutorship of Adam Stacey. This is a declaration that the work submitted in the thesis is my own, except as acknowledged in the text, and that it has not previously been submitted for a degree at the University of Queensland or any other institution.

Yours sincerely

_____

Barnaby Whiteman

Acknowledgements

I would like to take this opportunity to thank a number of people, without whom this project would not have been possible.

Firstly, I would like to thank Dr Joel Fenwick for allowing me to partake in this project. His passion for the subject is wonderful and has inspired me to play more table top games. I very much appreciate his lending out of personal games for “study” by students.

Adam Stacey has been invaluable in his tutorship throughout this project. His guidance has been of enormous help over the course of the project.

The ITEE Office staff have been very patient in administering Dr Fenwick’s deck building games throughout the year and I thank them.

I would also like to thank my family who have supported me generously throughout this project and my degree. Special thanks to my father David and sister Jemima for assisting in proof reading and giving advice.

Lastly, thank you to Kristen Thomson who has provided support throughout my entire degree. She helped keep me sane through the hard times and celebrated the good.

Abstract

The purpose of this project was to create a software system that allowed users to play *Deck Building* games (a genre of tabletop card game) in an online environment. The project required the system to be flexible and able to play new games with minimum effort. Prior to this project, all *Digital Deck Builder* implementations (deck building games played virtually) allowed users to play only a single deck building game. This project addresses this limitation by creating a generic deck building environment that can play a variety of deck building games. Three games were implemented as part of the project to demonstrate the system's functionality. These games are *Dominion, Ascension: Valley of the Ancients*, and *Caveman*.

The project was made possible by the fact that deck builders share a few commonalities. The structure of players card decks, basic game mechanics (such as purchasing and playing cards), and game setup are all similar among the games within the genre.

As the games are to be played online, a web-based solution was created. Both the front-end and back-end were programmed using the JavaScript language. In order to create the back-end functionality, node.js was used to run JavaScript outside of a browser environment, while socket.io (a web socket implementation) handled communications to and from the clients. The front-end used P5.js, a basic graphics library, to render the games in-browser.

Adding a game to the system is a case of creating two files and adding them to the system's configuration file. Firstly, a JavaScript Object Notation (JSON) configuration file, holding the game's basic information (such as cards, player statistics and starting decks) is required, as is a JavaScript custom functionality file. This holds the end-game condition as well as describing any cards whose functionality is not already built into the system and is too complex to be handled by JSON.

The resulting system has a few limitations; however, it fills the requirements set out by the project. Most notably, the completed system can:

- host multiple games online;
- play different types of games at once; and
- have games added with game configuration and function files.

Future versions of the system could include features such as player log-in and improved visuals, however the system is currently ready to play.

Table of Contents

List of figures

List of tables

1. Introduction

For this thesis project, a *Digital Deck Building* system has been created. The aim of this project was to create a highly flexible system that, with minimal modification, can play a variety of deck building card games.

The system was configured to play three deck building games. These games were *Dominion, Ascension: Valley of the Ancients* as well as a custom deck building game that was designed specifically for this project, called *Caveman*. These games demonstrated not only how to use the system but the functionality it can achieve.

1.1 Purpose

Prior to this project, all existing, public, digital deck builders were only able to play a single game. While some offer expansions, they are still based upon one game. The purpose of this project was to create a digital deck builder system that could play a wide variety of deck building games. Configuring it to play new games should be a trivial task (i.e. with no source code modification required).

In terms of software, a modular system is not a new concept. However, such systems have yet to be applied to digital deck builders.

A secondary purpose was to use this project to display Software Engineering capabilities and understanding of the underlying principles. It was used to demonstrate decision making in a software design context.

1.2 Goals and requirements

There were several goals and requirements outlined at the beginning of the project. These included both general goals – pertaining to the overall system and user-experience – and technical goals, which were specific to the design and implementation of the system.

Note: During the project several goals were modified or withdrawn as the requirements of the system evolved. The goals and requirements listed below are as they stood at the end project. Original goals and requirements can be seen in the *Project Proposal* document.

1.2.1 General requirements

- A working deck building game platform
- Easily accessible online client

- Ability to play games online against real opponents
- Easy for people with little to no coding experience to configure the system to play a deck builder of their choice
- To play at least three deck building games;
    - A free choice (*Dominion*),
    - A custom designed game (*Caveman*), and
    - *Ascension: Valley of the Ancients* (unknown prior to *Semester 2 2018*).

1.2.2 Technical requirements

- Game engine that is highly adaptable
- Minimal (preferably zero) source code modification required to change games
- Configuration file system to change the game being played
- Server-side game code to prevent players cheating by modifying their game states
- A web page to host the client in-browser

1.2.3 Out of scope

As with any project, it is important to define the scope and this project is no exception. There are currently many implementations of digital deck builders that offer a wide variety of functionalities and extra features, a lot of which are beyond the scope of this project. These features are outlined below.

Artificial Intelligence is often a feature in games that allows players to play against computer opponents. This project was not about creating a system to play against other humans or computers; rather, it was about creating a system that can host humans playing against other humans.

Graphical/UI Design was also not the focus of this project. Instead, the focus was placed on the software system. In a commercial setting, a dedicated team of graphical designers and specialists would handle this element of the project. The project was, however, done in such a way that graphics can be added relatively easily. This is discussed further in Section 3.5.2.

Advanced Match Making is seen in several current implementations of digital deck builders (examples given in Section 2.3). In these examples, players can add friends and invite them to play games and other more advanced features. These features were beyond the scope of this project; the matchmaking that was implemented is discussed in Section 3.3.3.

Campaign Modes are offered by some digital deck builders. The system has been made to play a digital representation of physical games and as such, these additional game modes were not required.

Expansion Packs are not explicitly in the scope of the system, though something similar can be achieved by adding a modified version of a games configuration files that includes the differences caused by the expansion.

2. Background

In order to build a flexible deck builder game system, it is important to identify all the common features present in deck building games. This will allow for a system that caters to as many games as possible and minimises the need to re-implement features for each unique game.

The following section lists features found in deck building games and some major differences which need to be kept in mind. Some examples of digital deck builders that already exist are also provided, which give a guide for the user interface and experience design.

2.1 Deck builders

A deck builder is a class of table top card game that focusses on players building up their deck of cards. Generally, each player starts with the same cards and uses them to purchase more cards from central pools. There are several deck building games that are relatively popular, however the genre as a whole is fairly unknown to the general public and is somewhat new. *Dominion* is considered the first deck building game, though its creator Donald X. Vaccarino went through several unpublished prototypes during development [1]. The game was first officially released in October of 2008 making the genre just 10 years old (at the time of writing) [2].

2.2 Common features and differences

The flow of cards is one of the key elements of the deck builder. In every game, at the beginning, each player is given a starting deck of cards. The contents of the starting deck vary widely from game to game. At the start of a round they draw a certain number of cards from their deck. When the cards in their hand have been used, they are placed in the player's discard pile. If the player needs new cards in their hand but their deck is empty, the discard pile is shuffled and is used as the deck.

"Trashing" cards allows players to remove cards from their deck permanently. Usually they are placed in a trash pile and cannot be touched again, however some games utilise them into the gameplay (for example, *Arctic Scavengers*' Junkyard [3], or *Ascension: Valley of the Ancients*' void pile [4]).

All deck builders allow players to purchase cards to add to their decks. As such they all have a form of currency, however there are multiple ways this is implemented in the games. Some

games (such as *Dominion*) have dedicated currency cards [5], while others give value to cards that can perform other actions as well (for example in *Ascension*) [4]. Some games even have multiple forms of currency (like *Arctic Scavengers* with their medicine and food [3]).

The resources available for purchase are located in common card pools in the centre of the table. The way in which these are sorted vary from game to game. *Dominion* has several piles each of which is dedicated to a single type of card. In contrast, *Star Realms* has one pile of a consistent card but also uses a shuffled "trade deck" to feed the "trade row", whereby every time a card is taken from the row it is replaced with the top card of the trade deck [6].

Typically, at the end of each turn, players will place all of their cards (both used or unused) into their discard pile and draw new cards from their deck. This is not always the case. In *Arctic Scavengers*, any unused cards are kept until the end of the round where they are placed into the "skirmish". The player with the highest number of combat points in their unused cards then wins a card from the "contested cards" pile [3]. Others, such as *Star Realms*, have “bases” which, once played, persist until they are destroyed by an opponent [6]. These types of cards do not get placed in the discard at the end of a players turn.

The way in which games end is a point of difference for a lot of games. In *Star Realms* combat points are dealt to opponents until their "Authority" reaches zero [6]. *Dominion* allows for the purchase of "land" of varying sizes that give a different number of victory points. When the largest land card ("Provinces") run out, or three other card piles in the common pool run out, the game is over [5]. Alternatively, *Arctic Scavengers* has a "contested cards" pile from which players fight to get cards. When this pile runs out the game ends [3].

2.3 Digital deck builders

A *Digital Deck Builder* is a deck building game that has been created to allow players to play the game electronically. A number of digital deck builders already exist, although, as previously stated, all of them are designed to play one specific game. Despite the differences in scope of these examples, they provided insight into what the system should offer, as well as the general layout. All of the examples given below use a similar *User Interface* (UI) for gameplay and use similar point and click mechanics for performing moves.

2.3.1 Dominion Online

Dominion Online [7] is a web based, multiplayer implementation that is completely free to play. It allows players to set up matches between random players or friends and supports AI matches as well. The game set-up options can be seen in Figure 1 below.

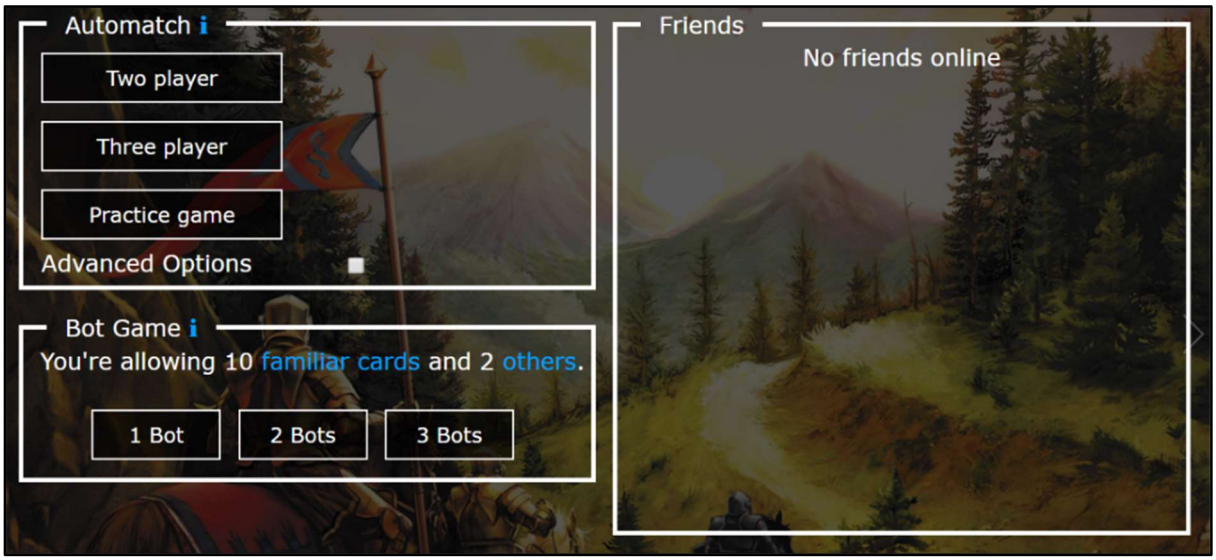


Figure 1: Screen-shot of Match Making screen for Dominion Online [7]

The game screen for *Dominion Online* (shown in Figure 2 below) is set up to resemble the table top version. The pool cards are placed in the centre of the screen, and the players' cards are displayed at the bottom. There are numbers to indicate how many cards are remaining in the common pools as well as on the players' deck. These are good inclusions as it is vital to keep track of these numbers throughout a game.

A log is also kept on the far right which displays a history of cards played during each player's turn. (e.g. "Player1 played 3 gold and bought a Province"). This is also valuable, as viewing what people play in the table top version is an important aspect of gameplay.

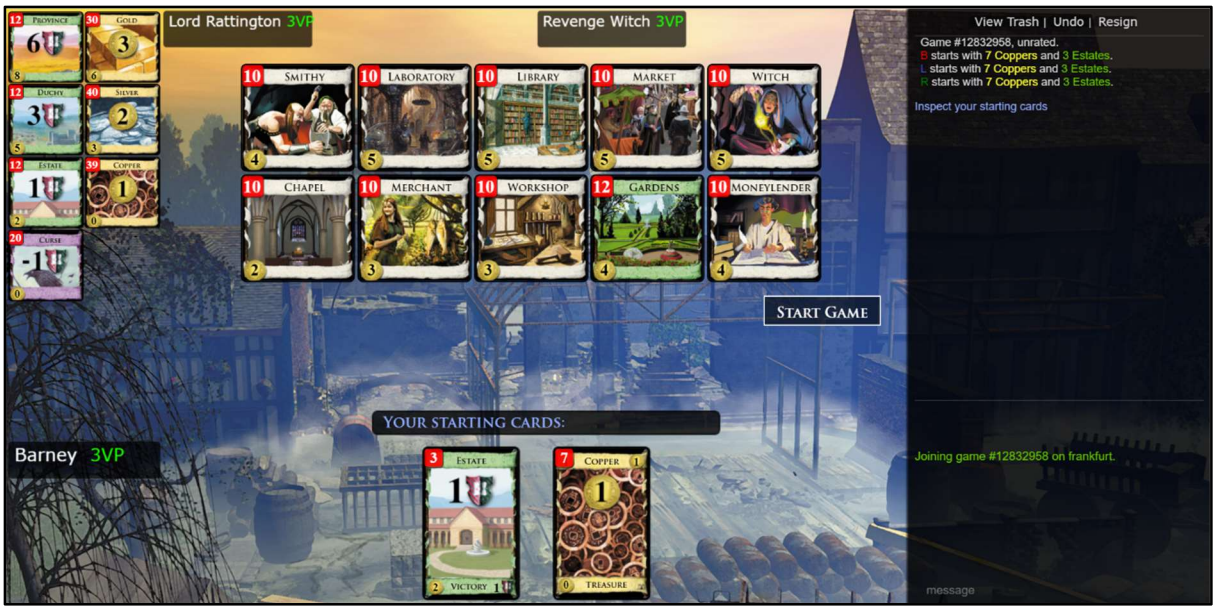


Figure 2: Screen-shot of Game screen for Dominion Online [7]

One element that is lacking from *Dominion Online* is the ability to view a card's information before purchasing it. The cards displayed in the common pools are limited to the graphic and name; this saves screen space but removes vital information about each card. The player is provided no way in which to easily access this information without purchasing the card and waiting for it to appear in their hand where they can view the entire card.

2.3.2 Star Realms

The digital version of *Star Realms* is available for download on Steam, Android and iOS. The free version allows players to play against AI while reserving the rights for online matches to paying customers. It also has a number of purchasable expansions [8]. *Star Realms* offers many of the same features as *Dominion Online* with the addition of a "Campaign" mode.

The layout of Star Realms (seen in Figure 3) is much the same as *Dominion Online* in that it mimics the original table top version.

Figure 3: Screen-shot of Game screen for Star Realms digital version [8]

Unlike *Dominion Online*, the digital version of *Star Realms* allows the player to view a card's information before purchase. When the player clicks on a card, it opens in a full view and gives the player the option to either exit out of the card view or purchase the card. This feature is important for a player's *User Experience* (UX).

2.3.3 Ascension mobile app

Digital deck builders are not confined to the computers. *Ascension* has a mobile application that can be played on phones and tablets [9]. Figure 4 shows that the layout of the application is very similar to that seen in *Dominion Online* and *Star Realms*.

Figure 4: Screen-shot of Game screen for Ascension mobile app [9]

The main points of difference for *Ascension* are that it is a mobile app and that it contains an expansion pack on the base game. As previously stated, expansion packs are not explicitly inside the scope of the project; however, the system can achieve a similar effect using the configuration files.

2.4 Lessons learned

There are several features common to all deck builders, such as a player's card structure and the idea of purchasing cards from pools available to all players. There is also a large variety in the functionalities of cards such that it would be impossible to cater to all cards' functions without the system becoming extremely bloated. To cater to this variety in functionalities, the system must be able to read in extra functions for specific games (see more in Section 3.7).

The digital deck builders currently offered all have a very similar layout. The layout was followed in this project as it is familiar for those that play digital deck builders. It also closely resembles the game layout used when playing physical versions of the games, which should make it easier for players transferring from physical to digital deck builders.

As mentioned, an undesirable feature of *Dominion Online* is that it does not show card information. This is a significant drawback, as new players cannot understand a card simply

from its title. Both *Star Realms* and *Ascension* both show all the card information which is a big improvement on *Dominion Online*.

A beneficial feature of *Dominion Online* is the player log, which shows which cards have been played and purchased by a player during their turn. This is an important feature as this information influences game play decisions.

3. Method

There are several distinct sections of the project that have been developed: the server, the game engine, and the client. A synopsis of each is detailed below, giving insight into the basic operations of each element and how specific events are handled.

3.1 Project plan

As the project development took place over approximately 7 months, a plan was integral to success. In general, design decisions were made as early as possible, maximising time spent in development. While this approach was ultimately successful, more time spent in the initial design phase could have saved redesigns at later stages (see Section 3.9).

Due to the scale of the project, it was first broken into two main categories of deadlines: d*evelopment* and a*ssessment*. The development category was made up of deadlines for stages in the systems creation while the assessment category consisted of all deadlines pertaining to graded items/deliverables. During the initial research and proposal stage of this project (March 2018), a basic outline was created (seen in Figure 5 below) which was generally followed throughout the course of the project.

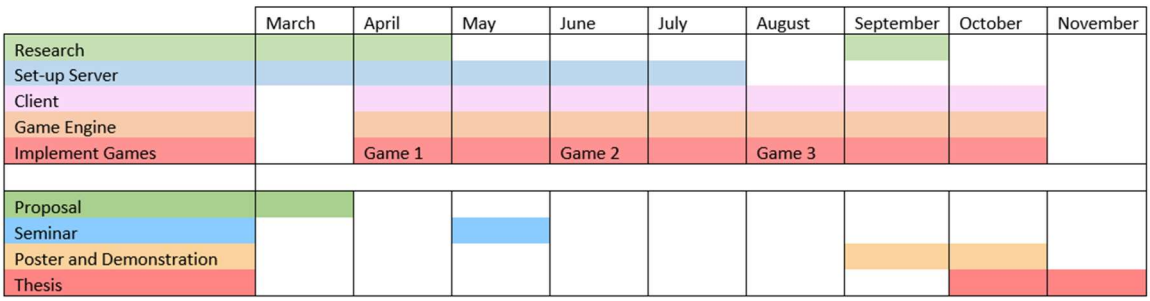


Figure 5: Proposed project timeline

As is evident in Figure 5, the project was divided into several sections. A brief description of each item in the timeline is given below:

3.1.1 Development

Research: In this phase, research was conducted into games within the genre of deck builders. Common features across the games were identified, allowing the system to cater to a wide number of games. This research predominantly took the form of reading game manuals and playing various deck building games. These methods give immediate insight into the structure of the games. Research was targeted towards the games being implemented as every detail of gameplay had to be understood thoroughly to implement these games. General

knowledge of the entire genre was, however, extremely valuable in making the system more flexible and better able to cater to the entire genre rather than three games.

Research was mostly conducted prior to development of the game engine to allow it to target the genre appropriately. A second round of research was undertaken at the release of the last game to be implemented (*Ascension: Valley of the Ancients*).

Server: The server is a key feature of the system as it is what allows the clients to play games online. To do this, communications are set up between the client and server that allow game information to be sent to and from the server.

Client: The client is what the users see and use to interact with the system. It needs to communicate with the server using messages able to be interpreted by the game engine. It also must represent the game state in an intuitive way for users to interpret.

Game engine: The game engine is responsible for running a single game. It keeps track of whose turn it is, what cards each player has, the cards in the common pools, and any other information pertaining to the game state.

Game implementation: Implementation of the games refers to creating the configuration files used for each of the three games.

3.1.2 Assessment

The following sections outline the key milestones from which the body of work was assessed.

Proposal: 22nd March. The goal for the proposal due date was to have the majority of background research completed, this goal was met.

Seminar: 24th May. The plan was to have a working card system (tracking multiple platers cards, shuffling and card pools) and basic client-server communications completed by this date. Communications were completed according to this deadline; however, the card system was incomplete.

Poster and demonstration: 15th October. Project development was planned to be completed by this date, including a demonstration of all three games. Minor changes and fixes were made to the system following this date.

Thesis: 5th November. This document is submitted to meet the requirements for the Honours Thesis.

3.2 Toolchain

JavaScript was considered a suitable language choice for the system (both frontend and backend), as it runs in browsers, can be used to create graphical displays, runs servers and is flexible and modular.

JavaScript also has a native object notation (*JSON*) that allows for data to be stored in a relatively *human-readable* manner. This is ideal for the configuration files as they are easy to create and edit.

Server/Game engine: The server used *node.js* which is a JavaScript run-time environment that allows JavaScript code to be executed outside of a browser [10]. This was used in combination with *Express*, a web framework for node.js that allows for the simplification of creating a server [11].

Game files: Two types of game files exist: configuration files and custom functionality files. The configuration files use JSON to describe game information, while the custom functionality files use JavaScript to describe any extra functions required be cards in the game.

Client: As the system had the requirements of being multiplayer and easily accessible, it was decided that the client would come in the form of a webpage. The client consists of a very basic *HTML* webpage that serves as medium for the client JavaScript to run in. The JavaScript uses *P5.js*, which is a JavaScript library that allows for graphical displays and animations to be created in the browser [12]. This was used to draw cards and user interface (*UI*) elements onto the players screen.

Using JavaScript for both the server/back-end and the client gives the code consistence. As the entire system was developed by an individual, this is extremely valuable.

Client/Server communication: *Socket.io* was used for communications between the client and server. Socket.io is a JavaScript library with both a client-side version to run in browser, and a server-side version that runs in node.js [13]. It is a real-time, bi-directional communication system that predominately uses WebSockets (refer to Section 3.3.2.1 as to why Web Sockets were used).

Version Control: A private, web-based Git repository was hosted on *GitHub* [14]. As this project was an individual undertaking, none of GitHub's collaborative features were used.

Instead it was used to keep track of progress in the project as well as providing remotely hosted back-ups in the event of file lose.

Testing: User/play testing was conducted extensively throughout the development of the system. Newly implemented features on both the server and client side were thoroughly tested prior to committing the code to the repository. As such only known working code was placed under version control.

Note: Unit testing was not implemented for reasons further discussed in Section 3.9.3.

3.3 Server

The role of the server is to not only serve the client HTML but facilitate communications to and from the clients, allowing online gameplay. The server is central to the success of the system as it serves as the connection between the client and the games being played.

3.3.1 Goals

There are several goals with regards to the server, namely:

- Serve HTML
- Communicate in real-time with clients
- Keep track of multiple games
- Keep track of multiple clients (and which game they belong to)

3.3.2 Approach

3.3.2.1 Communications

In order to send and receive messages to and from the clients, a method of communication had to be chosen. There are a number of communication methods used for web-based online games, each having their uses. A brief synopsis of options is given below.

Long-polling is a method that involves the client periodically "polling" the server for information. The server then responds to this request with the required information. This method is largely outdated and is relatively expensive in both CPU and bandwidth. This is due to the need for a new HTTP header for each polling request [15].

WebRTC is a method of communication that relies on the *User Datagram Protocol* (UDP) to transport messages. UDP does not guarantee the order of packages, favouring the most up-to-date information [16].

WebSockets use the *Transmission Control Protocol* (TCP) to send messages. Unlike UDP, TCP "guarantees" messages are sent and that they arrive in the correct order [17].

Note: This guarantee excludes network failures.

UDP is preferable in a lot of online games. This is because it prioritises speed. For high-frequency, non-critical data that needs to arrive quickly (such as player positions or inputs) UDP is the best option. However, in card games the onus is not on speed but accuracy of information. Therefore TCP, and hence WebSockets, is a much more appropriate choice for this system. The library socket.io was used as the WebSockets implementation, allowing the server to communicate with the client in an appropriate manner.

3.3.2.2 Tracking games

There were two main options for how the server/game relationship was handled. The first option is that a single server is responsible for all communications, and relays messages to multiple game instances. The second is for a game instance to be a stand-alone program and each to have its own server.

As playing multiple games is a requirement of the system, the first option was favoured. It more easily allows tracking of multiple games. In addition, it is easier for players to access the system as there is only one server to connect to.

Note: A potential problem for this method is scalability; the server becomes a bottle neck for the system as it must process every request. Several solutions exist such as upgrading server hardware or re-configuring the system to utilise some form of distributed computing, however, this is beyond the scope of this project.

The server used game objects to maintain control over multiple games at a time. These game objects were responsible for tracking and playing a single game (discussed further in Section 3.4).

3.4.2.3 Tracking players

There are two main methods for tracking players; either by creating user accounts or from tracking their connection.

User accounts require a much more complex back-end. Users need to be able to create accounts and all their data needs to be stored (to see more about user accounts, refer to Section 4.5.3). There is not much to be gained by user accounts in this system.

Connection tracking will only maintain user data while they are using the system. This functionality is already built into the socket.io library. This method imposes fewer obligations for users as they do not need to sign up for anything, they can simply start playing.

Tracking the user's connection was a more appropriate choice for this system as it is a much simpler solution to the problem and is already a part of the system via the socket.io library. To take advantage of this, when a player joins a game, a *tuple* (pair of values) is made of their socket ID and game ID and this is added to the list of players in games. This allows the server to know which players are involved in which game.

3.3.3 Execution

The server listens to a specified port on the local host for connections. When connections are made, they are served the HTML of the client. This is achieved using node.js in accompaniment with Express.

The server keeps track of current games (using game objects) as well as players (using player objects). As far as the server is concerned, players can exist in two states; either pending or playing.

3.3.3.1 Joining games

A socket connection is established with the client and the *socket ID* is used to track a given client.

Note: there is a unique socket ID for each instance of the webpage, allowing users to have multiple independent clients open in one browser.

When a client initially connects, their ID is added to a *pending* list and they are asked to choose which game they would like to start, or if they would like to join an open game.

Note: "open" games (i.e. already existing games) are games created by other users that have less than the maximum players joined and have not yet started.

When a response is made, the ID is removed from the pending list and either added to a game or a new game is created, and the client added to it. The client's ID is then stored in a separate list for clients in games along with the game ID of the game they are playing. When a client joins a game, the game selection screen is updated to reflect the changed state of the games.

3.3.3.2 Making moves

As the server handles all communications, messages intended for the game in which a player is playing are forwarded onto it by the server (by looking up the game ID associated with the player).

3.3.3.3 Disconnections

When a client disconnects, a socket *disconnect* message is sent notifying the server. If the client was in the pending state, it is simply removed from the list.

If the client was playing a game, the game is told to remove the player. If the game does not meet the minimum player value after a player is removed, the game ends and is removed from the games list. All players that were playing the game are notified that the game has unexpectedly ended due to players leaving the game. All remaining players are then sent back to the game selection screen.

Note: Disconnect messages are only sent when the page is closed. If a client loses internet connection, socket.io will automatically attempt to re-establish the connection. When the connection is re-established, the game will continue as normal.

3.3.3.4 Hosting

For demonstration purposes, the system was hosted on a personal desktop computer. The network router redirected requests to the computer using the port the server was listening to. While this is not a permeant or optimal solution, it was adequate for the purposes of demonstrating the system.

Hosting with a service is a superior long-term solution and is not difficult to achieve. Once the files have been pulled onto the server and node.js installed, a web server such as *Apache* can be used to redirect requests on a certain port to be forwarded to the port on which the server is listening.

3.4 Game engine

3.4.1 Goals

The game engine is an integral part of the system. Its features must include:

- Be able to read in external files to modify gameplay
- Track overall game states

- Track players' game stats

3.4.2 Approach

3.4.2.1 Game logic location

There are two main options for keeping track on the game state: either the client can manage its own game state, or a game object (on the server side) can manage it for everyone in a game.

Client-side: While running the game code on the client side would reduce the load on the server there are some other issues inherent with this option.

Firstly, all the clients for one game would have to be kept up-to-date with each other. This would require some sort of system to be running on the server to coordinate the game.

Secondly, it is more easily cheated as players have direct access to their own game states. This raises the question of whose game state is to be trusted if there is a discrepancy.

Thirdly, this would drastically increase the amount of data being sent to the client. The clients would have to receive the configuration files required to play specific games.

Server-side: Running the game code not only simplifies the solution, but also fixes the issues present for the client-sided alternative. As the game is being conducted in a single location, there is no need for coordination. And as there is only one copy, no discrepancies exist.

With regards to the load placed on the server; the game engine only needs to process information when a move is made. As this is a card game, not only are moves infrequent (when compared with other online games), but only one person per game cane make moves at a given time (this is due to the turn-based structure inherent in deck building games).

3.4.2.2 Game and player states

In order to keep track of the games, an object-oriented approach was taken. Both game and player objects were created to simplify tracking multiple of each.

This object-oriented approach could have been taken further by creating objects for cards, pools, decks and so on. This was deemed unnecessary; the card data are described using JSON, essentially making it an object already (refer to Section 3.6.3.3). Pools and decks are, at their core, just arrays holding references to card values (refer to Section 3.9.5).

3.4.2.3 Configuration file input

A few options exist for adding configuration files into a game. These options are discussed below:

Pre-load: This method entails the server loading the game files, contained within the server configuration file (see Section 3.6.3), when it is started. When a game is created, a copy of the loaded game file would be given to the game object.

Load-on-demand: In contrast, loading on-demand would see the game object load the files itself when the object is created.

The pre-load method would reduce stress on the system as the files only need to be loaded once when the server has start. The load-on-demand method has to load the files every time a new game is created. The benefit to loading the files every time is that any changes made to the configuration files will be included into the new game without having to restart the server. This means patches can be made to the games without the system going offline.

Both methods are valid, but due to the nature of this project (with a lot of configuration file prototyping), the load-on-demand method was used as it drastically reduced the amount of time for testing features in the configuration file.

3.4.3 Execution

3.4.3.1 Game objects

Game objects are used by the server to handle entire games. A single game object stores all the pools, player data and game states for a single game and is what the clients interact with (via the server) when they play.

Creation: When created, a game object is given the path to the JSON configuration file that relates to the type of game being created (see Section 3.6). The JSON file contains the path to the custom functions JavaScript file. Cached versions of both files are removed and then re-loaded and stored by the game object. This is done so that changes in configuration files will manifest themselves in-game without the need to restart the server. Once the configuration file is loaded, the pools can be initiated, and players added.
Note: Games cannot be added without restarting the server; only modifications to existing games via their JSON configuration files or their JavaScript custom functions file. Further, a running game will not be affected by these changes; only a newly created one.

Adding players: Once players have chosen the game they would like to play, they are added to the game (whether it is a newly created game or an already existing one). The game keeps track of the multiple players using player objects that are stored in an array.
The first player to join a game (i.e. the player that started the game) is given the chance, should the game configuration allow for it, to choose the pools with which they would like to play the game (see Section 4.2.2). Once the choosing process is complete, the pools are initiated. If the game configuration does not permit players to choose starting pools, the pool is initiated immediately.

When initially joining a game, a player is again put into a pending state. Once all the players in a game have clicked a "ready" button and the pools have been selected, the game can begin. This causes it to no longer be "open" for other players to join, regardless of whether it is at maximum player capacity.

Removing players: If a client disconnects (as described in the Section 3.3.3), then their associated player will be removed from the game in which they are participating, and players will be notified via the player log. Once the player is removed, if the game has fewer than the minimum number of players, the game will prematurely end. If that is not the case, then the game will proceed as normal, without the disconnected player. If it was the disconnected player's turn, the next player in the order will begin their turn.

Making moves: Once a move message has been based onto the relevant game, several steps are undertaken. Firstly, the move is given a basic validation check to ensure that the player that is making the move should be doing so.
Then the type of move is determined (i.e. playing from the hand, purchasing from a pool, or completing a choice), and the appropriate action is then taken for the cards within that move.

3.4.3.2 Player objects

Player objects are used by the game objects to keep track of each player within a game. As previously stated, when a player joins a game, a new player object is created in relation to that player.

The player object is responsible for maintaining the state of a player, including their deck, hand, and discard piles. It also contains functions for manipulating the cards owned by the player in all the ways commonly seen in deck builders (e.g. adding cards to the player's deck, discard pile, and hand).

3.5 Client

3.5.1 Goals

The aim of the client is to provide a method of interaction to the users. It is the visual manifestation of a player's game state within the game object of the game they are playing. The main aims for the client are:

- that it is intuitive and easy to use, and
- it is easily accessible.

3.5.2 Approach

A large factor in making the system intuitive is making it familiar to users. By researching similar products on the market, insight can be gained into what the users are used to interacting with (See more in Section 2.3).

As previously stated, graphics were not the primary concern for this project. As such, all *Graphical User Interface* (GUI) elements are extremely basic, to the extent that using a graphical library (i.e. P5.js) is not necessary to achieve the look seen in the completed client. Using P5.js also puts more load on the client as rendering occurs numerous times per second. Doing it in such a way, however, lends itself to easier enhancements to graphics in the future. P5.js allows for more complex graphical elements than can be achieved using the HTML elements and the groundwork for it has already been laid.

A number of approaches can be taken to with regards to the client implementation. Most notably, the system could be a desktop application or a web-based solution. Both have merits; however, a web-based solution was taken. The main reasons for this were:

- Easily accessible (no download for players)
- Browser environment lends itself to both networking and graphics

There are disadvantages to a web-based solution; the main one being performance. A desktop environment is more powerful than that of the browser. However, it was deemed the advantages outweighed the disadvantages.

3.5.3 Execution

All visuals (excluding the player log) were made using P5.js by utilising its in-built "draw" loop which is used to continually draw elements onto the canvas. The client is put into

different states by socket messages initiated by the game. Based on these states, the relevant screens are rendered onto the screen.

The player log was a simple *textarea* HTML element that had its *innerText* updated as player log messages are received from the server. There was a problem encountered when users would rapidly click on the game screen causing the text in the player log to be selected. This was fixed by disabling input into the text area and making it un-selectable.

The client is not responsible for any game logic. The closest it gets is basic checking when a player is asked to make a choice (e.g. “Select a card to be discarded from your hand”). This checking is backed-up by validation on the server-side and serves more as a pre-screening. The game logic is entirely server-sided to limit the potential for cheating by players modifying their client-side game state. This has the adverse effect of putting the majority of the strain on the server as it has to keep track of every player and every game.

3.6 Game configuration JSON files

3.6.1 Goals

The purpose of a *game configuration file* is to provide the necessary information to the game objects for them to play unique games. The information contained within the configuration file should be specific to a single game. This file must be human-readable to allow users to create their own games with little prior knowledge.

3.6.2 Approach

Originally *XML* was going to be used instead of JSON (as per the *Project Proposal* document); however, this was changed during development for the following reasons:

- JSON is native to JavaScript
- JSON is parsed into a JavaScript object making it easier to work with
- JSON supports arrays (these were used extensively, see Section 3.6.3.2 on Pool data)

3.6.3 Execution

In order to add a game to the system, a configuration file must be created that describes the game to be added. This configuration file holds the basic game information as well as definitions for all of the cards and pools. A single “master” JSON configuration file (named *config.json*) holds the paths to the configuration JSON files of each game. The master

configuration file can be seen below in Figure 6. This config file also allows users to change the port number of their server.

```
{
    "port": 8000,

    "games": {
        "Dominion": "./GameFiles/Dominion/dominion.json",
        "Caveman" : "./GameFiles/Caveman/caveman.json",
        "Ascension:VotA": "./GameFiles/Ascension/ascension.json"
    }
}
```

Figure 6: Server configuration file JSON

Any added games must have paths to their JSON configuration files specified in *config.json* within the *games* tag. A game's individual configuration file can be broken into several sections as seen below.

3.6.3.1 Basic game information

There are several elements that need to be described within the configuration file that do not pertain to card data. These elements can be seen in Figure 7 below.

```
"title": "Dominion",
"require": "./GameFiles/Dominion/dominion.js",
"max_players": 4,
"min_players": 2,
"draw_size": 5,
"victory_type": "points",
"default_round_stats": {
    "action": 1,
    "buy": 1,
    "treasure": 0
},
"stat_types": ["action", "buy", "treasure"],
"turn_structure": ["action", "buy", "clear"],
"starter_cards": [
    {
        "name": "copper",
        "count": 7
    },
    {
        "name": "estate",
        "count": 3
    }
],
```

Figure 7: Basic game information as described in the Dominion JSON

Title: Is the title of the game being played.

Require: This holds the path (relative to the Server files) of the additional functions for the game. These additional functions are explained in Section 3.7.

Max/Min players: Describe the number of players required to play the game.

Draw size: Refers to the number of cards drawn by a player into their hand at the beginning of their turn.

Victory type: Describes how the winner is decided. This can be either "points" or "health". Using points will look for cards that award points to calculate the players score, this is the system used for both *Dominion* and *Ascension: Valley of the Ancients*. The health system will decide winners based on who has the most health once the game has ended. This is used in the custom game *Caveman*.

Default round stats: These are the statistics given to a player when they start a new round.

Stat types: Are the types of statistics present in the game. Points and health are excluded from these as they are handled separately.

Turn structure: Describes the order of operations within a player's turn. Within the turn structure, "action", "buy", and "clear" phases all possess specific functions. You cannot purchase cards during the action phase and you cannot play action cards during the buy phase. The clear phase indicates that the player's hand should be "cleaned", and a fresh hand given to them. Other phases can be used, such as in *Ascension: Valley of the Ancients* wherein the turn structure is "turn", "clear". This is due to the fact that players are allowed to play whatever cards in any order regardless of type. Using "turn" circumvents the restrictions on "action" and "buy", thereby allowing the free-flowing nature of the game.

Starter cards: Holds the information that describes the initial deck with which players start the game. For each type of card in the hand, the name of the card is given as well as the count (i.e. the number of cards of that type). A player's deck is always shuffled at the beginning of the game.

Colours: Different cards can be assigned colours based on their faction or type. The colours are described in an array with three values, corresponding to the *red, green* and *blue* values of the colour respectively. An example of the colours can be seen in Figure 8 below.

```
"colours": {
    "hero": [150, 150, 150],
    "monster": [250, 0, 0],
    "temple": [160, 100, 0],
    "enlightened": [0, 140, 200],
    "void": [160, 0, 200],
    "mechana": [150, 75, 50],
    "lifebound": [50, 200, 0],
    "default": [250, 250, 250]
}
```

Figure 8:Colour information from the Ascension JSON

Faction is given precedence over type; for example, the *Ancient Stag* card from *Ascension: Valley of the Ancients* is a *hero* card from the *lifebound* faction. It will therefore be displayed using the *lifebound* colours. When a faction or type is not specified, the default colour will be used.

3.6.3.2 Pool data

There are several types of pools that can be employed by a deck builder, each of which is seen in the three games implemented. Figure 9 below shows the overall structure of the pool object as seen in the configuration file. This Figure shows 2 examples of pools. Each type of pool seen is explained in further detail below.

```
"pools": {
    "const": [···
    ],
    "num_random": 10,
    "random": [···
    ]
},
```

```
"pools": {
    "const": [···
    ],
    "deck": {···
    },
    "dealt": {···
    }
},
```

Figure 9: Pool structures from the Dominion and Ascension: Valley of the Ancients JSON

Constant Pools: Are always present in every game that is played. For example, the treasure cards from *Dominion* would be considered constant pools as they are always available regardless of the setup of the other pools.

```
"const": [
    {
        "name": "Copper", "shuffled": false,
        "cards": [{
            "name": "copper", "count": null, "values": {"2": 46, "3": 39, "4": 32}
        }]
    },
    {
        "name": "Silver", "shuffled": false,
        "cards": [{
            "name": "silver", "count": 40
        }]
    },
```

Figure 10: Example of constant pools from the Dominion JSON

As seen in Figure 10, pools can either have a fixed *count* of cards (like the *Silver* pool) or can be varied based on the number of people playing (as seen in the *Copper* pool). The card's key holds reference to an array of objects, each referring to one type of card, more than one card type can be used if desired (see *Consistent* and *Shuffled* pools later in this section).

Random Pools: Are pools that can be selected by the players. An example of this can be seen in *Dominion*, in which players are able to select 10 pools of cards with which they would like to use in that game (or use a random selection).

```
"num_random": 10,
"random": [
    {
        "name": "Cellar", "shuffled": false,
        "cards": [{
            "name": "cellar", "count": 10
        }]
    },
    {
        "name": "Chapel", "shuffled": false,
        "cards": [{
            "name": "chapel", "count": 10
        }]
    },
```

Figure 11: Example of random pools from the Dominion JSON

Figure 11 shows an example of the random pools. The "num_random" refers to the number of pools to be selected from the pools held in the "random" array. If there are more elements in the array than the number of pools to be selected, the user gets to choose which ones are used (see more in Section 4.2.2). Otherwise, all the random pools are used.

Deck and Dealt Pools: The deck pool emulates a shuffled deck of cards that is used to deal into dealt pools. When a card is purchased from a dealt pool, the top card from the deck is

then dealt into the pool where the card was purchased from. This system is used in *Ascension: Valley of the Ancients* for the "centre row" as seen in Figure 12 below.

```
"dealt": {
    "num": 6,
    "size": 1
},
"deck": {
    "shuffled": true, "permanent": true,
    "cards": [
        {"name": "hellfrostimps","count": 6},
        {"name": "kanzirtheravager","count": 1},
        {"name": "starvedabomination","count": 3},
        {"name": "hurrasseasfury","count": 1},
        {"name": "cryptlurker","count": 3},
        {"name": "mutatedscavenger","count": 2},
        {"name": "ikuvalleytyrant","count": 1},
        {"name": "cavernhorror","count": 3},
```


Figure 12: Example of dealt and deck pools from the Ascension JSON

Dealt pools are simply described by how many there are ("num") and how many cards they possess ("size"). The deck is given all the cards in the configuration file.

When using a Deck/Dealt pool it should be noted that the deck itself is not displayed in the GUI to save screen real-estate, since these cards are not to be seen by the players.

Consistent Pools: Constant, random and deck pools can all be consistent as this simply refers to the fact that they only contain one type of card. All *Dominion* pools are consistent in this respect. An example can be seen in Figure 13 below.

```
{
    "name": "Gold", "shuffled": false,
    "cards": [{
        "name": "gold", "count": 30
    }]
},
```


Figure 13: Example of consistent pool from the Dominion JSON

Shuffled Pools: In contrast to consistent pools, shuffled pools are given a randomised order at the beginning of the game. While a pool with only one type of card can theoretically be shuffled, it is only practical to shuffle a pool if there is more than one type of card in it. *Caveman* uses shuffled pools for its action cards (as seen in Figure 14), as do a number of other deck building games.

```
{
    "name": "Pile 1", "shuffled": true,
    "cards": [
        {"name": "hunter", "count": 10},
        {"name": "gatherer", "count": 10},
        {"name": "farmer", "count": 10}
    ]
},
```

Figure 14: Example of shuffled pool from the Caveman JSON

To use shuffled pools, the "shuffled" tag must be set to true. If a specific order is required, the shuffled tag can be set to false and cards can be put into the cards array in the desired order.

Permanent Pools: A permanent pool is one in which the player does not take a card when purchasing from it. This effect can be seen in the *Cultist* card from *Ascension: Valley of the Ancients* (as seen in Figure 15 below).

```
{
    "name": "Cultist", "shuffled": false, "permanent": true,
    "cards": [{
        "name": "cultist","count": 1
    }]
},
```

Figure 15: Example of permanent pool from the Ascension JSON

In a permanent pool, only the first card in the cards array will ever be seen unless the pools are manipulated by the custom game functions (discussed in Section 3.7).

3.6.3.3 Card data

Deck builders are essentially card games, and it is the properties and functions of the cards are that make each game distinct. Despite this, there are some fields that are common throughout the genre. In the configuration JSON file, there is a field named "all_cards" that holds data for all of the cards in a given game. In Figures 16 and 17 below are two examples of card data.

```
"cavernhorror": {
    "name": "Cavern Horror", "type": "monster",
    "cost": 2, "currency": "power", "stats": {"add_points": 1},
    "discard": true, "function": "cavernhorror",
    "desc": "Reward: 1 Honour.\n You may shuffle your discard into your deck"
},
```

Figure 16: Cavern Horror card data from the Ascension JSON

```
"workshop": {
    "name": "Workshop", "type": "action",
    "cost": 3, "currency": "treasure", "stats": {"add_card": [4, "treasure"]},
    "desc": "Gain a card costing up to 4 Treasure"
},
```

Figure 17: Workshop card data from the Dominion JSON

Key: The key holds the reference to all the data for a given card (in Figure 16, the key is "cavernhorror", not to be confused with the card name "Cavern Horror"). To reduce memory usage, this key is all that is used in every card storage within the system (i.e. pools, player's hands, discards etc.). When a card's data is needed, the "all_cards" object is queried and returns the object with all of the data for a given card.

Name/Type: The name refers to the title of the card, which is used for displaying the card. The type of the card is also displayed but has bearing on how it can be played, for example *action* cards can only be played during the *action* phase.

Cost/Currency: Dictates how much to deduct from a player when purchasing a card. Some games have multiple currencies, such as *Ascension: Valley of the Ancients* using *Runes* and *Power* as currency for different types of cards.

Stats: The stats object can hold various key words that trigger specific actions along with values associated with these actions. Valid key words are listed below.

Points: Determines the number of points a card awards a player. These points are tied to the card. If the card is trashed, the player forfeits the points associated with it. The associated value is the number of points the card awards.

Add Points: Grants additional points to a player. These points are not tied to a card but rather to the player. The associated value is the number of points to be added.

Cards: Allows the player to add new cards to their hand from their deck. The associated value is the number of cards the player draws.

Add Card: They player is allowed to add a card from one of the pools. The associated value is an array: the first element is the cost they are allowed up to and the second value is the currency.

Trash: Allows the player to trash cards from their hand. The associated value is an array with two elements, being the minimum and maximum number of cards respectively.

Discard: Allows the player to discard cards from their hand. The associated value is an array with two elements, being the minimum and maximum number of cards respectively.

Damage: Deals damage to all opposition players. The associated value is the amount of damage to be dealt.

Player Stats: Alternative to the above, a card can also add value to a player's statistics. This is how cards grant extra purchasing power, actions, buys etc. Only stats specified in the "stat_types" variable (see Section 3.3.1 on *Basic Game Information*) will be added. The associated value is the value to be added (or removed if the number is negative) to the player's statistics.

Faction: Along with a cards *type*, some games (such as *Star Realms* and *Ascension: Valley of the Ancients*) give cards a faction. Factions can be used in a variety of ways by different games. An example of a card with a faction can be seen in Figure 18 below.

```
"burialguardian": {
    "name": "Burial Guardian", "type": "hero",
    "cost": 4, "currency": "runes", "stats": {"points": 2, "runes": 2},
    "faction": "lifebound", "function": "burialguardian",
    "desc": "+2 Runes.\nThe next time you acquire a lifebound card this turn: Death Keystone"
},
```

Figure 18: Burial Guardian card from Ascension JSON

Discard: If a card's *discard* attribute is set to true, then upon purchase the card's statistics will be applied, and the card will not be added to the players deck. This feature can be seen in use with the *monster* cards in *Ascension: Valley of the Ancients*.

Persistent: Persistent cards are not cleared at the end of a player's turn if they have been played. This was used for *Ascension: Valley of the Ancient's Construct* cards and could also be used for *Star Realm's Bases*. At the beginning of a player's turn, any persistent cards statistics are applied and if they have special functionality, call-back functions are kept and checked when the player's cards update in any way.

Function: Gives the name of the function associated with a card. If a function name is given then the game functions file (specified in the "require" tag, see Section 3.6.3.1) should have a function with an identical name. (See Section 3.7 regarding C*ustom functionality JavaScript files*).

Description: Holds a human readable description of the card's statistics and functionality. This is displayed on the face of a card in the client and this is how players know what a card

does. The description should reflect any custom functionality associated with the card. Name, cost, type, and so on do not need to be included.

3.7 Custom functionality JavaScript files

3.7.1 Goals

The aim of the custom functionality JavaScript files is to allow the complex actions performed by some cards to be described conveniently. This file should be included into the game so that the functions contained within can be executed with ease.

3.7.2 Approach

Originally, it was hoped that the entire game configuration, including all card functionality, could be contained within a single file. However, this was abandoned as there was no convenient way in which to do so. This is due to the fact that some cards are far too complex and cannot be described within the confines of either XML or JSON. This made it necessary to include another method of adding functionality to cards.

The best way to do so was to create a supplementary *custom functionality file* system. These files were written in JavaScript as this allows them to be executed by the game objects and keeps languages consistent across the system. The custom functionality files are included into the game object and the functions contained within can be freely called.

3.7.3 Execution

As previously stated, many cards in each game possess unique attributes and functionalities that are not seen in any other card or game. Creating a system that can inherently perform all of these actions would quickly become bloated. This problem was solved by implementing a system whereby a card can have a custom function associated with it if necessary.

To achieve this, functions are written in a JavaScript file which are then included into a *Game* object when it is created. The path to the file is specified in the games JSON configuration file (refer to Section 3.6.3.1 on *Basic Game Information*, specifically regarding the *require* tag).

3.7.3.1 Game over function

Every game function file must have at least one function present. This function checks if the end game condition is met as the condition varies largely between games and therefore

cannot be described in the JSON configuration file. This function must be named "game_over" and must return a Boolean. The function is given a reference to the game as a parameter. *Dominion's* end game function can be seen in Figure 19 below. It should be noted that this function is an export, as this allows it to be accessed from external files.

```
exports.game_over = function (game) {
    //Checks for game over
    if(game.pool_empty("Province") || game.num_pool_empty() >= 3) return true;
    return false;
}
```

Figure 19: Game Over function for Dominion

3.7.3.2 Card functions

Any functions specific to a card need to be named exactly as they appear in the JSON configuration file (see Section 3.6.3.3 on *Card Data*, specifically regarding the *function* tag). They also must be exports, such that they are accessible from an external file (as demonstrated in Figure 20 below).

```
exports.cellar = function(game, player, cards) {
    if(cards == null) { //first time being called
        var message = "You can discard as many cards as you like, they will be replaced by new cards from your deck."
        game.player_choose(player, "hand", 0, game.players[player].hand.length, "discard", "cellar", message);
    } else { //cards have been selected
        game.players[player].draw_cards(cards.length);
    }
}
```

Figure 20: Example of a card function from Dominion

As seen in Figure 20 above, every card function is given a reference to the game as well as a reference to the player using the card as parameters. The *cards* parameter is used after the player has made a choice and the function has been set as the call-back; however, *cards* is *null* when called from applying the cards statistics.

An example of this call-back functionality can be seen in the cellar function from *Dominion* shown above in Figure 20. The cellar card allows a player to discard as many cards from their hand as they wish. Once they have discarded the cards, they may draw the same number of cards as they discarded.

This is achieved by asking the player to choose the cards they would like to discard when *cards* is *null* (i.e. first time the function is being called). The call-back is also set to be the cellar function.

Once the cards have been selected by the player, the cellar function is then re-called (as it was set as the call-back) with *cards* being an array containing the cards that were selected to be

discarded. As *cards* is no longer *null*, the second half of the function is executed which draws cards for the player, with the number being equivalent to the number of cards discarded.

Another type of call-back exists, used for cards with functions triggered by external events. An example of this can be seen in the Burial Guardian card whose function is seen in Figure 21 below. The card is triggered the next time the player acquires a *hero* type card from the *lifebound* faction.

```
exports.burialguardian = function (game, player, cards) {
    //next time you acquire a lifebound hero this turn: Death Keystone
    if(cards == null) {//first time through
        //add callback
        game.players[player].add_callback("burialguardian");
    } else if(cards[1] == "acquired" && game.cards[cards[0]].faction == "lifebound"
     && game.cards[cards[0]].type == "hero") {
        //card has been acquired
        death_keystone(game, player);
        return true;
    }
}
```

Figure 21: Burial Guardian call-back from Ascension

When the function is first called, the player who played the card gets the *"burialguardian"* function added to its call-backs. Upon certain events (i.e. "played" and "acquired"), the call-backs are run and removed if they return true.

Note: Both types of call-backs can be used (player choosing, and event call-backs) by testing the *cards* parameter; on event call-backs use a tuple for the cards object with the second item being the event type ("played" or "acquired").

3.7.3.3 Other functions

The game functions file can of course hold functions that are not tied to a card but are useful to other functions within the file. A basic example of this can be seen in Figure 22 below.

```
function echo(game, player, faction) {
    //players discard must contain a card of the same faction
    for(var c in game.players[player].discard) {
        if(game.cards[game.players[player].discard[c]].faction == faction) return true;
    }
    return false;
}
```

Figure 22: Example of extra function from Ascension

In *Ascension: Valley of the Ancients*, some cards are granted extra functionality if certain conditions are met. Figure 22 shows one such condition called *echo*.

Note: This function is not an export as the other function types must be. This is because it does not need to be accessed from outside the scope of the file in which it is written.

3.8 Communications

3.8.1 Goals

The main goal with the communication was to keep it as light as possible so that bandwidth usage is minimised. The messages should also be logical and targeted, containing only the required information.

3.8.2 Approach

As the client is embedded in a webpage, the *Hyper Text Transfer Protocol* (HTTP) is used for serving the initial web-page. HTTP is the standard using in web-browsers.

Web sockets are then used to communicate all game information. This is achieved using socket.io (refer to Section 3.3.2.1 regarding why sockets are used).

Socket messages are structured such that they have both a *topic* and *message*. The topic was used to send targeted data, while the message was used to carry the data itself. When the data was sufficiently complex, JSON objects were used, otherwise variables were sent directly.

3.8.3 Execution

The communications can be broken into two main sections: messages sent by the server, and messages sent by the client. The following section details the messages, data, and its purpose.

3.8.3.1 Server messages

Listed in Table 1 below are messages *sent* by the server with an explanation of what they do.

Topic	*Data*	*Description*
alert	message	Displays the message in an alert.
choose_game	games	Sets and displays all the currently open games so the player can choose which game to join (or start a new one).
choose_pool	pools_to_select	Sets and displays the pools the player can choose from at the beginning of a new game (if permitted by the game).
disable		Disables the *end phase* button and changes the button text to "Not your turn".
hand	cards	Sets the cards in a player's hand for display.
log	message	Adds the message to the player log.
new		Sets the client up for playing a new game (once a game has been joined).
over		Lets the client know the current game is over.
phase	phase	Sets the *end phase* button text to reflect the current phase ("End" + phase).
played	cards	Sets the cards a player has played for display.
pool	cards	Sets the cards one top of each central pool for display.
reset		Resets the client to its initial state.
revealed	cards	Displays the revealed cards.
started		Lets the client know the current game has started.
stats		Sets the players statistics for display.
unexpected		Lets the client know that the current game has ended because there are no longer enough players.

Table 1: Socket messages sent by the server

3.8.3.2 Client messages

Listed in Table 2 below are messages *sent* by the client with an explanation of what they do.

Topic	*Data*	*Description*
disconnect		Lets the server know a player has disconnected
end_phase		Passed to game. Lets the game know that a player has ended the current phase of their turn.
game_selected	game	Lets the server know which game a player wants to play. Player either joins the selected game or creates a new one (decided by the player).
move	move	Passed to game. Lets the game know a move made by a player
new		Resets the player and joins them back to the game selection screen
pools_selected	pools	Passed to game. Lets the game know which pools the player would like to play with (if permitted by the game)

Table 2: Socket messages sent by the client

3.9 Problems encountered and lessons

3.9.1 Communications

Originally, communications over sockets did not use JSON or any such standardised data structure. This was due to the fact that, in the beginning of the project, messages that were being sent were extremely basic and did not warrant such a system. However, the communications rapidly became more complicated, but the method of communication was not changed.

This led to a re-design of communication data having to occur towards the end of the project. This was done to ensure that communications were kept serviceable.

3.9.2 Game selection

As stated in Section 1.2.1, of the three games to be implemented, one was a free choice of deck building game. Originally this game was to be *Arctic Scavengers*. This was due to the fact that it contained a lot of unique functionality and, in theory, display the capabilities of the system. In reality, it caused a narrowing of vision with the system focussing on the needs of the one specific game rather than the genre as a whole. For this reason, the demonstration game was changed to be *Dominion* as this was the first ever deck building game and contains features found in most deck builders. This made it a much more suitable choice of game to base implementation around.

3.9.3 Testing

In the *Project Proposal* document, it was stated that Unit Testing would be implemented to ensure that new code would not impact systems that had been implemented. This plan was not followed through. This is due to the fact that rigorous play testing was conducted for each new feature added. It is now apparent that Unit Testing could have been a valuable tool for quickly identifying faults in the system. Time taken to write the tests would most likely have been less than the time saved by doing so. Originally this was not thought to be the case, hence the decision to not implement them.

3.9.4 Input method

The client currently has the players select cards individually to be played, closely resembling the sequence of events that occurs in the table top equivalent. A different method if input was trialled however; players would select all the cards they wanted for a phase of the turn and

press a “send move” button. This would send all the selected cards to the server as one message to be processed. Doing it in such a manner caused issues not only with respect to user experience, but also was unnecessarily complex. As such, the player input method was changed to the final, individual card method.

3.9.5 Pool data structure

For the majority of development, the pool data structure remained as a two-dimensional array. These arrays held all the cards possessed within a given pool. This method made it difficult to do more complex, functions with them. For example, checking if a certain pool is empty would rely on either knowing the cards within it or knowing its index within the array. Another problem came with the introduction of “permanent” pools (seen in *Ascension: Valley of the Ancients* with the *cultist* card) as there was no easy way to track the properties of a pool.

This led to the pool structure being refactored to be an array of objects. Each pool object has the fields: *name, cards, type, permanent*. This extra information simplified a number of operations with the pools, including the ones mentioned above.

3.9.6 Communication data

The client-server communication was one of the first elements to be implemented into the system. At the time of implementation, the messages being sent back and forth were very basic (e.g. a test message or a single number). As the system progressed, the messages got more complex, with more variables to be sent. To send them, all the values were packed into an array. This method quickly started to break down and become messy.

To amend this, the message formats were changed to be JSON. This greatly reduced the complexity and improved the readability of code. Some messages did not require this change and were left as they were; however the messages that did need changing benefitted greatly from it.

4. Deliverables

The outcome of this project was a modular digital deck building system that fulfilled the requirements. Most notably it can:

- host multiple games online;
- play different types of games at once; and
- have games added with game configuration and function files.

Note: As both *Dominion* and *Ascension: Valley of the Ancients* have a large number of distinct cards, some were omitted from implementation as they are not necessary to demonstrate the capabilities of the system. While there are some limitations to what can be handled by the system (See Section 4.5 regarding the limitations), the cards chosen display a range of, if not all, functionality present in each game (the cards implemented in this demonstration system are described in Appendix E and Appendix F)

4.1 System overview

Figure 23 below shows an overall view of the system. On the left of the image is the *front end* of the system (i.e. the client, what users interact with), The right shows the *back end* (i.e. the server and game engine).

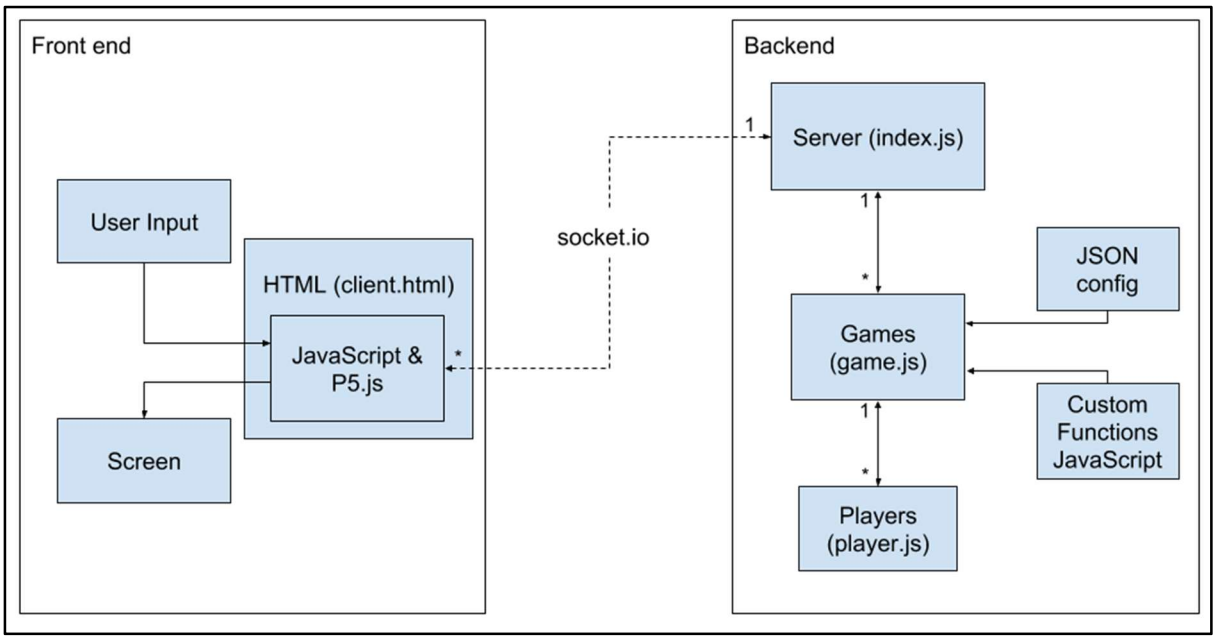


Figure 23: Final system diagram

Note: The server has a one-to-many relationship with both clients (via socket messages) and game objects. The game object has a one-to-many relationship with player objects.

4.2 Client

The images shown in this section are screenshots taken directly from the final system.

4.2.1 Selecting games

The game selection screen is used to choose games to play. If there are no "open" games (defined in Section 3.3.3 under *joining games*), the screen will look like Figure 24, however Figure 25 shows an example of the game screen with "open" games.

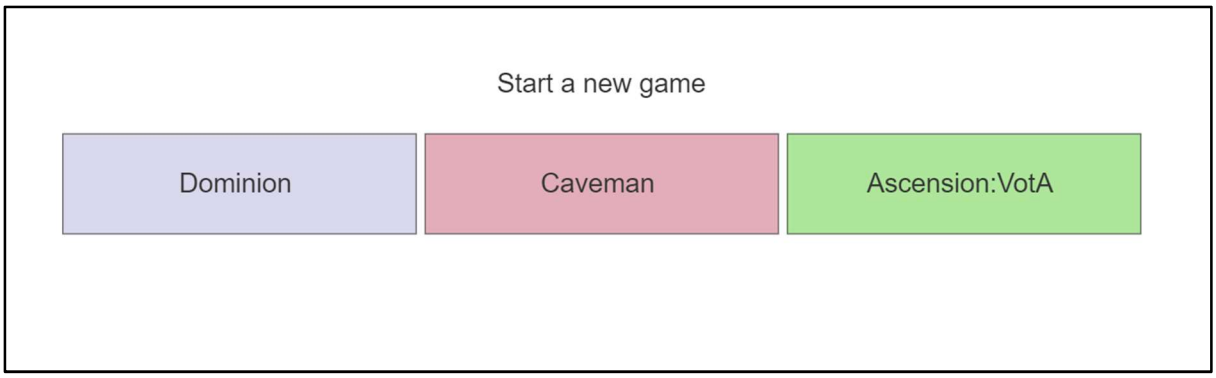


Figure 24: Game selection screen (no open games)

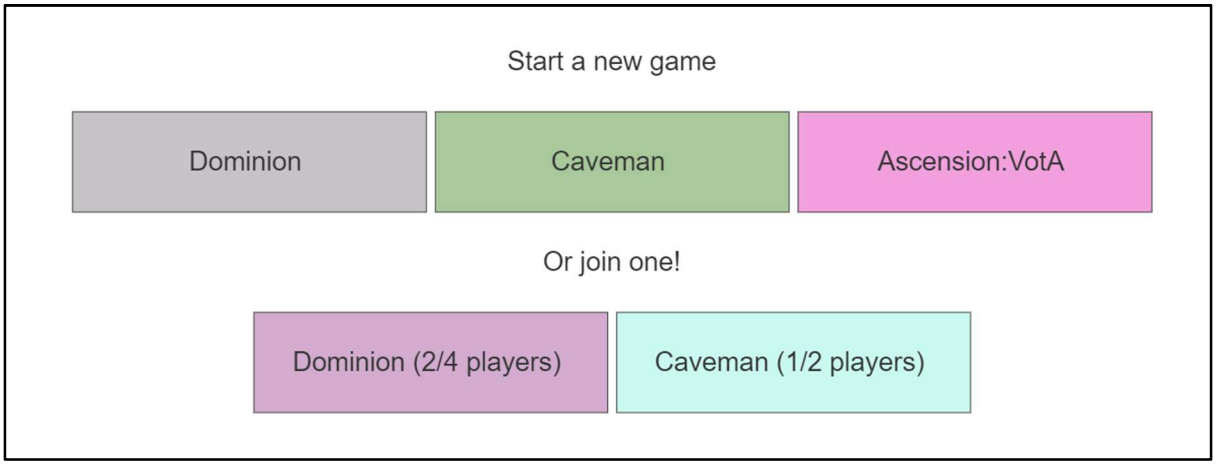


Figure 25: Game selection screen (one open game)

4.2.2 Selecting pools

When given the option to select pools, players will see the screen seen in Figure 26. An example of what the screen looks like after selecting pools can be seen in Figure 27. The number of pools to select and pool names are all based on the configuration files of a given game. The option for selecting pools is also only given under certain circumstances (see Section 3.6.3.2 about *random pools*).

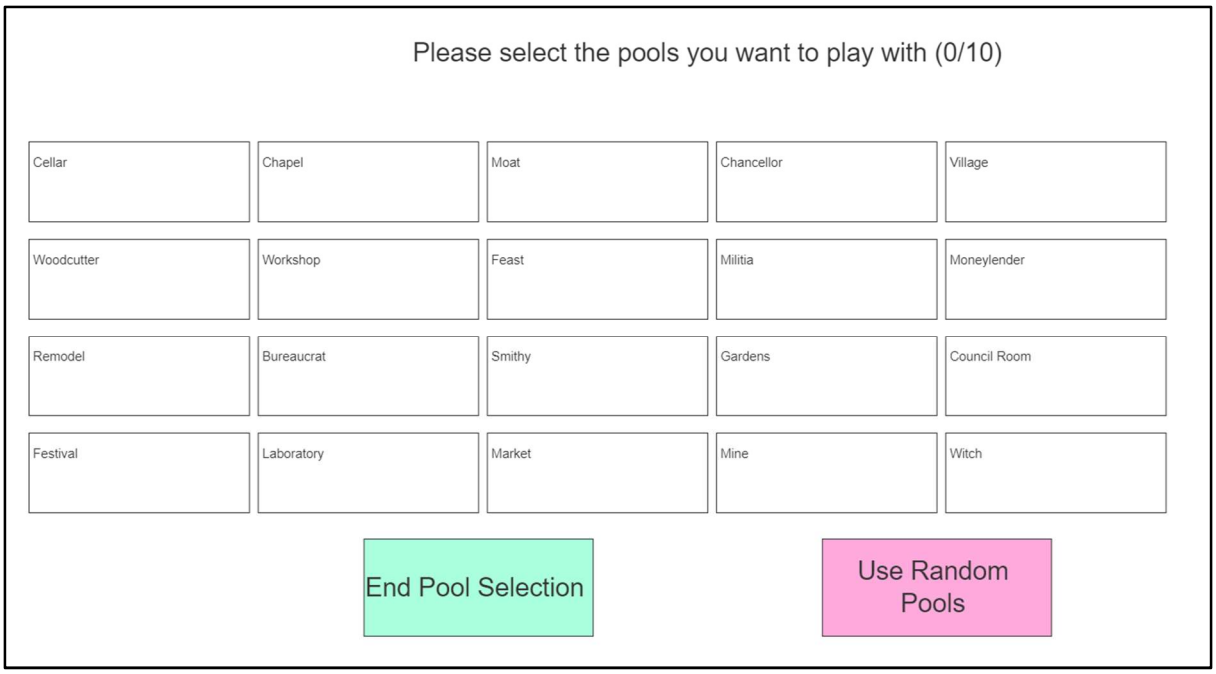


Figure 26: Pool select screen (none selected)

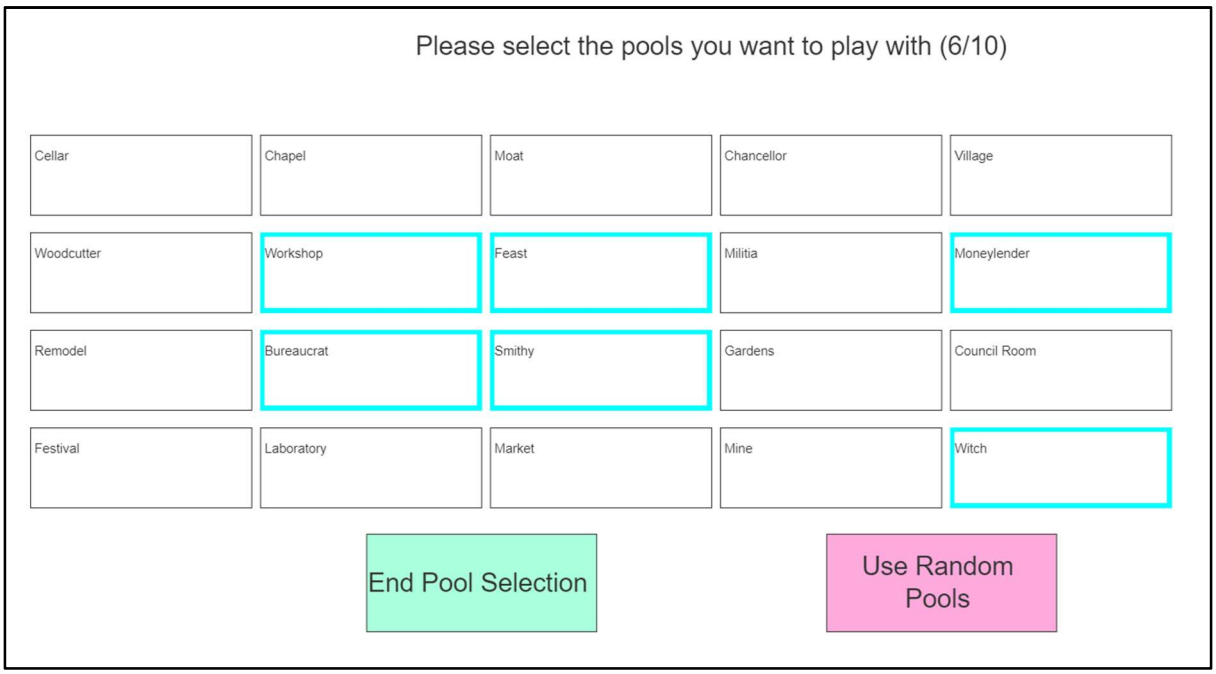


Figure 27: Pool select screen (some selected)

4.2.3 Playing games

Three games were implemented to demonstrate the functionality of the system:

- *Ascension: Valley of the Ancients*
- *Caveman* (custom game)
- *Dominion*

A screenshot from each game can be seen in Figures 28, 29, and 30 respectively.

Figure 28: Game of Ascension: Valley of the Ancients

Figure 29: Game of Caveman

Figure 30: Game of Dominion

4.2.3.1 Alerts

Alerts are used extensively throughout the lifetime of a game. Alerts are used to notify players of extra information. Alerts are used at the following times:

- Start of the game
- Start/end of a player's turn
- Specialised card actions
    - When a player is asked to choose a card
    - When cards are being revealed to a player
- At the end of the game to notify players who won

Examples of alerts can be seen in Figures 31 and 32 below.

Figure 31: Alert for a player choice

Figure 32: Alert for game over

4.2.3.2 Unexpected game over

If a player leaves a game and reduces the number of players to be below the minimum number of players for the game, the game will end. When a game ends in this manner, all players remaining in the game will be shown the screen seen in Figure 33.

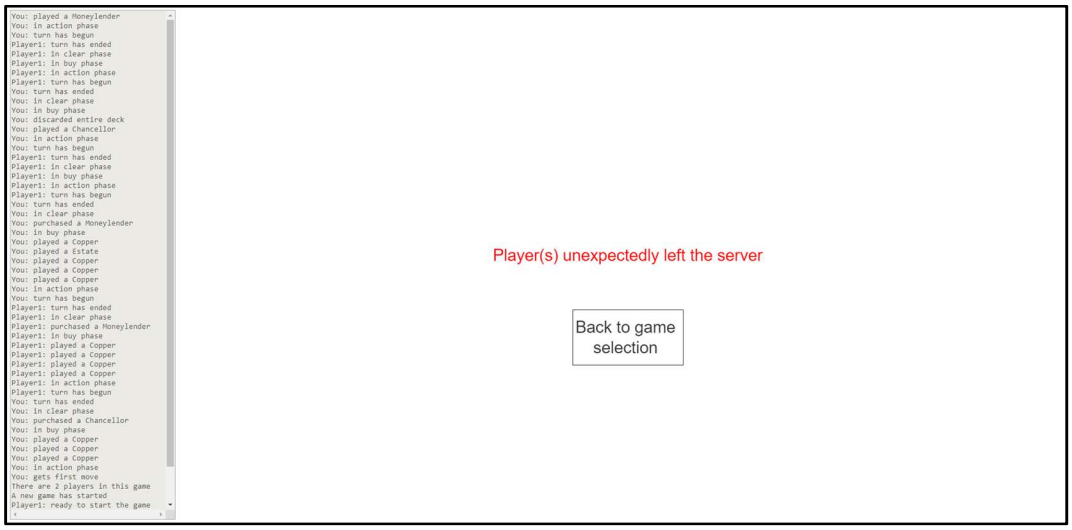


Figure 33: Unexpected game over screen

4.3 Server

4.3.1 Set-up and operation

In order to run a copy of the server, node.js must be installed (from https://nodejs.org/en/). The required packages can be installed using the command "`npm install`" in the server's directory.

Note: *npm* is a software registry for sharing JavaScript packages. It is included in the node.js download [18].

To run the server, use the command "`node index.js`" while in the server's directory. This will start the server, listening to the localhost on the port specified in the *config.json* file.

4.3.2 Adding games

Adding games to the server is a relatively straight-forward process as there are only three steps:

1. Create game's JSON configuration file
2. Create game's JavaScript extra functionality file
3. Add the path to the JSON configuration file to the server's *config.json* file

The three games that have already been added can be used as an example as to what these files should contain. A detailed break-down of what is contained within these files can be seen in Sections 3.7 and 3.8.

4.4 Custom game

As a requirement of the project, a custom deck building game was developed. This not only displays an understanding of what makes a deck builder, but also shows functionality within the system. As such, the game *Caveman* was created. *Caveman* is played between two people and uses some features not seen in *Dominion* and *Ascension: Valley of the Ancients.* These features will be highlighted in the following sections.

4.4.1 Caveman cards

As with any deck building game, there are several different types of card. Below is a list of categorised cards created and implemented for *Caveman*.

4.4.1.1 Treasure Cards

The treasure cards give the players *treasure* that can be used to purchase any other card in the game. *Treasure* is the only currency in *Caveman*. Table 3 below lists the treasure cards along with their cost and how much treasure they grant the player when played. The count shows how many of a given card are present in the deck.

Card	***Cost (Treasure)***	***Treasure***	***Count***
Wood	0	1	40
Bone	3	2	30
Stone	6	4	15

Table 3: Caveman treasure cards

4.4.1.2 Weapon Cards

Weapon cards are used to attack the opposing player. These are the only way the game can end or be won as the game only ends once a player's health reaches zero. Table 4 below lists the weapon cards along with their cost and how much damage they will deal to the opposing player when played. The count shows how many of a given card are present in the deck.

Card	Cost (Treasure)	Damage	Count
Club	4	1	15
Sling	6	2	10
Spear	10	4	5

Table 4: Caveman weapon cards

4.4.1.3 Action Cards

Action cards provide supplementary actions to the base gameplay and are integral for making the game enjoyable. Table 5 below lists the action cards along with their cost and other effects granted when played. There are 10 of each of the action cards present in the deck.

Card	Cost (Treasure)	Effects	Description
Hunter	3	+2 Treasure	
Gatherer	3	+1 Action	
Farmer	3	+1 Buy +1 Treasure	
Toolmaker	4		Trade a treasure for the next treasure up.
Artist	4	+2 Cards	
Priest	4		You may trash up to 2 cards. Gain +1 Health for each card trashed.
Prophet	5	+3 Cards	The top card from every player's deck gets shuffled and dealt into each player's discard. You do not get to see the card you are given.
Warrior	5	+2 Actions +1 Card	You may trash a card.
Elder	7	+1 Card +1 Buy +1 Action +2 Treasure	
Sacrifice	7	+1 Action +1 Card	Gain a card costing up to 4 Treasure. Trash this card once used.

Table 5: Caveman action cards

4.4.2 Caveman pool structure

Caveman uses a combination of both consistent and shuffled pools.

The consistent pools contain one pile for each treasure and weapon card, meaning a total of 6 consistent pools.

There are 4 shuffled pools in *Caveman*; one for each price bracket of action cards (e.g. a "3 Treasure" pool, "4 Treasure" pool, etc). These pools are shuffled so that each pile has a random distribution of cards within, meaning that when a card from a shuffled pile is purchased, the next card on the pile is not guaranteed to be the same as the card that was just purchased. Of the three games implemented, *Caveman* is the only one to contain shuffled pools; however, this is not a unique mechanic in the realm of deck builders.

4.4.3 Rules

Similar to *Star Realms*, the objective in *Caveman* is to conquer the opposition player. The game ends when a player has no health remaining (each player starts with 20 health points). Damaging a player is achieved using the weapon cards described in Section 4.4.1.2. Neither *Dominion* nor *Ascension: Valley of the Ancients* use a health-based system; however, it can be seen in other games such as *Star Realms* [6].

4.4.3.1 Starting Cards

The game begins with each player possessing a starting deck of cards. This starting deck consists of 4 *Wood* and 2 *Bone* cards.

4.4.3.2 Turn Structure

At the beginning of each turn, the player draws 3 cards from their deck. If their deck does not contain enough cards, the player should deal the remaining cards first then shuffle their discard and use this as their deck. They then deal from the refreshed deck until their hand contains 3 cards.

Like many deck builders, each turn has an *action* and *buy* phase in that order. At the start of each turn, the player gets 2 actions and 1 buy. This means that in one turn they may play 2 action cards during their action phase, and purchase 1 card during their buy phase. Players can increase the number of actions and buys they have by use of cards with *+Action* or *+Buys* respectively.

4.5 Limitations

Regrettably, there are a few limitations within the system. These range from limited in-game functionality to User Experience flaws. Outlined below are the more significant limitations posed by the system.

4.5.1 Variable draw sizes

The player's *draw size* (i.e. how many cards the player draws into their hand) is set as a fixed value in a game's configuration file, however some games make use of variable draw sizes throughout the game. For example, in the first round of *Star Realms*, the player going first draws 3 cards while the others draw 5, then in all successive rounds every player draws 5 cards. This dynamic draw size is not currently supported by the configuration files or the system itself but could be added in future releases.

4.5.2 Actions on discard piles

There are a number of systems that allow players to choose cards from their hands for specific actions (such as *discarding* and *trashing* cards) that are well-supported by the system. Some games, however, extend this functionality to a player's discard pile. This feature is not currently supported by the system.

4.5.3 User accounts

Players are tracked by their *socket ID* which is a unique identifier granted to each connection made to the server. This means that a player's data is tied to their session and re-opening the webpage will cause them to lose any previous data. Having user accounts would allow players to have persistent data over multiple sessions. User accounts were not implemented mainly because it was not a requirement or priority for the system and would add unnecessary complexity.

4.5.4 Page navigation

The client is a single webpage and all "screens" are simply different renderings onto the HTML canvas. As such, pressing the "back" button within the browser will cause users to leave the page. More importantly, on some screens there is no method of returning to a previous screen. For example, if a player no longer wants to join the game they selected (before it has begun), the only way to return to the game selection screen is to refresh the webpage, giving players a poor experience.

5. Conclusions and future work

Deck building card games are only getting more popular as more people discover them and more games are made. With this rise in demand comes an increase in the number of digital deck builders – a way to play the games without a physical copy.

The digital deck builders that existed before this project could all only play a single game. The system created for this thesis project addresses this problem. It offers a single platform to play a wide range of deck building games, with the ability to add more by simply including configuration files that describe the games features.

There is the potential that the lack of multi-game digital deck building platforms is due to copy-right and trademarking issues. Such legal issues could dissuade commercialisation and hence decrease incentive; however, from the position of a deck builder enthusiast, such disincentives do not exist.

There is room for work on the system to continue into the future. A number of improvements are listed below:

- The graphics of the system could be updated;
- Limitations within the system could be addressed;
- User accounts could be added, allowing players to track their win rate and other statistics; and
- Some form of load-balancing to enhance the systems scalability could be introduced.

However, despite the potential for improvement, the system fulfils all of the requirements set out by the project.

The timeline, laid out in Section 3.1, was followed closely throughout the duration of the project, only minor milestones were not met. A more granular task break-down would have been beneficial however, as tasks were often completed very close to deadlines. A more rigorous design process could also have prevented several systems' reconstructions.

The final outcome from this project was a functional digital deck building system that achieves everything that was set out for it.

6. Appendix

Appendix A: Project proposal

Project Proposal

Digital Deck Builder

Barnaby Whiteman
43948378

School of Information Technology and Electrical Engineering,
University of Queensland

Supervisor
Joel Fenwick

March 22, 2018

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

1 | Introduction

For this thesis project a Digital Deck Building system will be created. The aim of this system is to create a highly flexible system that, with minimal modification, can play a variety of deck building card games.
The system will have a client and server component allowing players to connect with their clients via the internet to play games against each other. The server will be able to host multiple games at a time.

The system will be initially configured to play a game named *Arctic Scavengers* as well as a custom deck building game that will be designed specifically for this project. In Semester 2 of the University of Queensland's academic year (commencing 23rd of October 2018) another game (presently unknown) will be implemented. This will test the flexibility of the system.

1.1 Goals

1.1.1 General Goals

- A working deck building game platform
- Easily accessible online client
- Be able to play games online against real opponents
- Easy for people with little/no coding experience to configure the system to play a deck builder of their choice
- Simple to start playing a game
- UX should be fun and intuitive. (Despite graphics and UI not being the focus of this project, Most visuals will be place-holder art as making compelling visuals is out of the scope of this project)

1.1.2 Technical Goals

- Game engine that is highly adaptable
- Minimal (preferable zero) source code modification required to change games
- A configuration file system to change the game being played

- Secure server communications with client
- Server side game code to prevent players from being able to hack their game states.
- A web page to server the client in-browser.
- Unit testing to ensure a working system

1.2 Purpose

The purpose of this project is two-fold. Firstly to design and implement a working digital deck building system. Secondly to provide proof of Software Engineering capabilities and underlying principles. It provides an opportunity to display good decision making in a software design context.

2 | Background

In order to build a flexible deck builder system it is important to identify all the common features present in deck building games. This will allow for a system that caters to as many games as possible and minimises the re-implementation of features for each unique game.
The following section lists features found in a deck building games and some major differences which need to be kept in mind. Some examples of digital deck builders that already exist are also provided, these give a guide for the user interface and experience design.

2.1 Common Features and Differences

The flow of cards is one of the key elements of the deck builder. Each player has a deck of cards. At the start of a round they draw a certain number of cards from their deck, when the cards in their hand have been used they are placed in the players discard pile. If the player needs new cards in their hand but their deck is empty, the discard pile is shuffled and is used as the deck.

Trashing cards allows players to remove cards from their deck permanently. Usually they are placed in a trash pile and cannot be touched again, however some games utilise them into the gameplay. (For example *Arctic Scavengers*' Junkyard[1])

All deck builders allow players to purchase cards to add to their decks. As such they all have a form of currency, however there a multiple ways this is implemented in the games.

Some games (such as *Dominion*) have dedicated currency cards[2], others give value to cards that can perform other actions as well (for example *Ascension*)[3]. Some games even have multiple forms of currency (like *Arctic Scavengers* with their medicine and food[1]).

The resources available for purchase are located in common card pools in the centre of the table. The way in which these are sorted vary from game to game. *Dominion* has several piles each of which is dedicated to a single type of card where as *Star Realms* has one pile of a consistent card but also uses a shuffled "trade deck" to feed the "trade row", every time a card is taken from the row it is replaced with the top card of the trade deck[4].

Typically at the end of each turn, players will place all of their cards (used or unused) into their discard pile and draw new cards from their deck. This is not always the case. In *Arctic Scavengers* any unused cards are kept until the end of the round where they are placed into the "skirmish", the player with the highest number of combat points in their unused cards wins a card from the "contested cards" pile[1].

The way in which games end is a point of difference for a lot of games. In *Star Realms* you must deal combat points to your opponent until their "Authority" reaches zero. *Dominion* allows you to purchase "land" of varying sizes that give a different number of victory points, when the largest land card ("Provinces") run out or three other card piles in the common pool run out, the game is over. Arctic Scavengers has a the "contested cards" pile that players fight to get cards from, when this runs out the game ends.

While most of the underlying mechanics are the same for every game, ensure that the system built can handle all the unique aspects of each game will be a challenge.
The main features necessary will be the handling of multiple cards decks (with the ability to shuffle), purchasing cards and adding and removing cards from decks. Array implementations in most languages allow for all of these features.

2.2 Digital Deck Builders

A number of digital deck builders already exist, although all of them are designed to play one specific game. The match-making features present in these examples are more advanced then the scope of the proposed system (read more about the match making in the *Design* section in the *Project Plan*). Despite these differences, they provide some insight into what they system should offer and how the UI should be laid out. Both of the examples below use a similar UI during the game and use similar point and click mechanics

for making moves/performing actions.

2.2.1 Dominion Online

Dominion Online[5] is a web based, multiplayer implementation that is completely free to play. It allows players to set-up matches between random players or friends and supports AI matches as well.

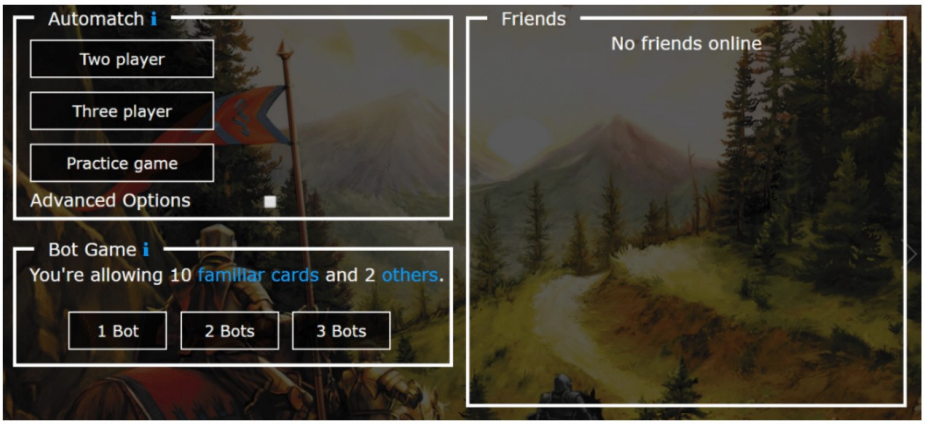


Figure 2.1: Screen-shot of Match Making screen for Dominion Online from *https://dominion.games/*.

The game board for *Dominion Online* is set up to resemble the table top version. The common cards are placed in the centre of the screen and the players cards are displayed at the bottom. There are numbers to indicate how many cards are remaining in the common piles as well as on the players deck, these are good additions as it is vital to keep track of these numbers through out the game.
A log is also kept on the far right which displays history of cards played in each players turn. (ie "Player1 played 3 gold and bought a Province"). This is also helpful as viewing what people play in the table top version is an important aspect of gameplay.

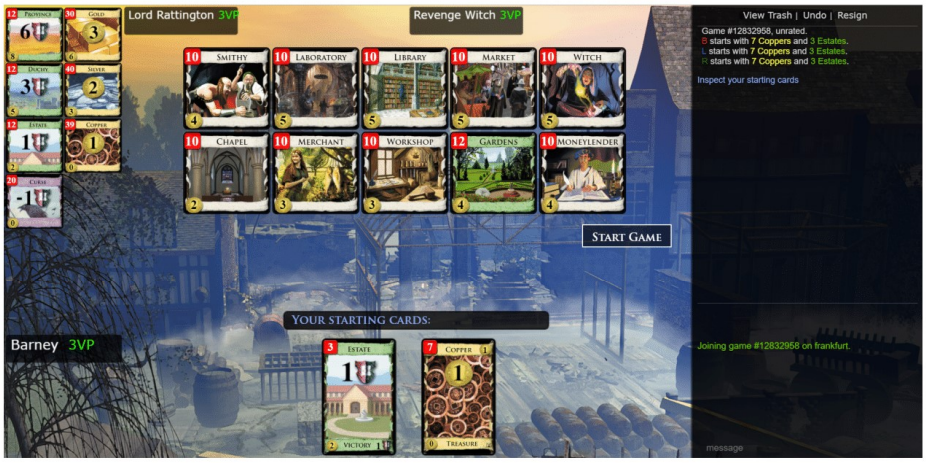


Figure 2.2: Screen-shot of Game screen for Dominion Online from *https://dominion.games/*.

One element that is lacking from *Dominion Online* is the ability to view a card's information before purchasing it. The cards displayed in the common pools are limited to the icon and name to save on screen space, but this cuts out vital information about each card. The player is given no way to easily access this information without purchasing the card and waiting for it to appear in their hand where they can view the entire card.

2.2.2 Star Realms

The digital version of *Star Realms* is available for download on Steam, Android and iOS. The free version allows players to play against AI while it reserves the rights for online matches to paying costumers. It also has a number of purchasable expansions[6].
It offers many of the same features as *Dominion Online* with the addition of a "Campaign" mode(This is out of the scope of the project).
The layout of Star Realms is much the same as *Dominion Online* where it mimics the original table top version.

Figure 2.3: Screen-shot of Game screen for Star Realms digital version.

Unlike *Dominion Online*, the digital version of Star Realms allows the player to view a cards information before purchase. When the player clicks on a card it opens it in a full view and gives the player the option to either exit out of the card view or purchase the card. This is an important User Experience element that will be implemented into the Digital Deck Builder system.

3 | Project Plan

3.1 Plan

	March	April	May	June	July	August	September	October	November
Research									
Set-up Server									
Game Engine									
Implement Games		Game 1		Game 2		Game 3			
Proposal									
Seminar									
Poster and Demonstration									
Thesis									

Figure 3.1: Proposed Project Outline. More details for each task are given below in the *Milestone Breakdown* section.

The above plan is split into two sections. The top section shows the Milestones that need to be completed for the project while the bottom shows deadlines for the various deliverables of the project.

3.2 Milestone Breakdown

3.2.1 Research

In this phase, research will be conducted in to the games within the genre. Common features will be identified so that implementation of the base system can cater to a wide number of games. The research will come mainly in the form of game manuals as these reveal the nuances of each game. These can then be compared between games and the common factors are highlighted.
This research has been completed to a satisfactory level for the compilation of this document, however more shall be done before development begins to ensure that the system is designed in such a way as to assist with completing the project rather than hinder it.

When the mystery game is announced in Semester 2 more research will be conducted into the game in order to have a solid understanding of it before implementation.

3.2.2 Set-up Server

One of the key features for this project is the client-server communication. The server will be run on a Linux machine (more details can be found in the *Tool Chain* section). It must be able to handle playing multiple games at a time.

3.2.2.1 Client-Server Communication

The main aspect of running the server is setting up client-server communication. This will involve configuring both the client and server to seamlessly communicate and design packets to efficiently communicate the game data.

3.2.2.2 Multi-Client Server handling

Once simple communications have been established the server will have to be adapted to handle multiple clients at once. It will have to be able to receive information from all connected clients and keep track of which game they are playing.

3.2.3 Game Engine

The basic engine includes keeping track of the common card pools, players decks, hands and discard piles, shuffling, keeping track of players turns etc.
The main game logic will be run from the server side and all the information necessary will be sent to the clients. This will help to combat cheating as the server will be able to validate the moves being made against its stored game state.

3.2.3.1 Game Configuration

To play each different deck building game, the system will have to be configured accordingly. The system will have to be designed in such a way that all of the cards specific to the game can be loaded in from an external file. There will also have to be a method for adding functions for the more complex cards whose actions are unique and cannot be described from those more common in deck building games (ie not implemented in the system due to it only being used by one card in one game).
Currently it is planned that card info (ie name, cost, description) will be stored in an XML file and loaded externally. An external file will also be used for functions needed by cards with non-standard functionality or play elements specific to that game.

3.2.4 Implement Games

Each game will have to have its own external files written (mentioned above) describing all the cards and non-standard functions.
Three games will be implemented over the course of the project, these being:

- Arctic Scavengers
- Custom designed game
- Mystery game

The custom game will be a deck builder designed specifically for this project by the author.

The mystery game is currently unknown and will be revealed in Semester 2. This is done to test the flexibility of the system and ensure that the project goals are met.

3.2.5 Proposal

22nd March
The majority of background research must be completed by this date.

3.2.6 Seminar

14th - 18th May
Basic system should be in place by this date. This should include a working card system (keeping track of multiple players cards, shuffling and common card pools) and basic client-server communication.

3.2.7 Poster and Demonstration

No later than 19th October
Project development must be completed by this date. The system must be configurable to play all 3 games developed over the course of the project.

3.2.8 Thesis

4pm 5th November
Project report must be finished along with a working system.

3.3 Tool Chain

The deck building system will be made as a web-app as this lends itself to easy server communications as well as simple graphical integration using the browser. Using a scripting language for the system will allow for functions to be passed as variables, this will be useful for adding functionality specific to certain games or cards. JavaScript will be used for the client and server as it can be effortlessly integrated with both the front and back end of a web-app. There are also a wide number of libraries that can be used with JavaScript (the ones planning to be used are listed below. This list may be subject to change over the project's lifetime if requirements change).

3.3.1 Server

The server will use *Node.js*[7] as its back-end since it runs on the same language as the client and can perform all the required server actions. Some examples of these actions are listed bellow:

- Keep track of game state (ie whose turn, round number, etc)
- Send and receive game information
- Host multiple clients at a time
- Host multiple games at a time

3.3.2 Client

For the client, *P5.js*[8] will be used on top of JavaScript as it allows for very simple graphical displays. The majority of the code will be written in base JavaScript with P5.js mainly being used for graphical representations of the game data.
A basic web-page will be made in HTML with CSS styles added, this will serve as a platform for the JavaScript to be viewed and executed.

3.3.3 Unit Testing

For unit testing *Jest*[9] will be used. Jest is built specifically for JavaScript and is simple to set up. Unit testing will be utilised in this project to catch any errors caused by new code in older systems.

3.4 Design

The system will be designed in a modular, object oriented manner. Cards will be created as objects to simplify deck management.
Ideally, a configuration file will store all the data necessary to set up the system to play any given game, however additional functions will have to be supplied for some of the more unique aspects of games. This is easy to do in JavaScript as functions can be passed as variables. This allows for the custom functions (specific to the currently configured game) to be given as a parameter into the system's existing function.

The configuration file will hold data such as set up information (how many cards to start with, what cards/how many go in the common pools etc). Configuration files will also detail how many players can play, what the game-ending conditions are and the list of cards available.

Cards will be stored as XML objects. They will all have basic information such as title, cost and description and can be given other attributes easily due to the nature of XML. This additional information will contain data such as custom functions to be used (as mentioned earlier) and bonuses (eg +1 Action).

For the multiplayer aspect of the system match making will be simplified as it is not the focus of this system. The server will allow plays to join a game until it has started or until the player limit is met, at such a time it will create a new game instance and any new players will be connected to it instead.

3.5 OHS

The software laboratory being used for this project (Software Thesis Lab, Level 1, Building 87 General Purpose South) is considered a *low risk laboratory*, as such it is covered by general OHS laboratory rules.

3.6 Risk Assessment

There are several risks to take into account while working on this project. Each risk will be rated on its *severity* (how big the impact of a risk is to the project), and its *probability* (how likely it is to occur during the project). Each factor will be rated as either LOW, MODERATE, or HIGH.

For severity:

- LOW - Set back of less than a day
- MODERATE - Set back of more than a day
- HIGH - Set back of more than 2 days

For probability:

- LOW - Unlikely to occur during project
- MODERATE - About 50-50 chance of occurring during project
- HIGH - Likely to occur during project

3.6.1 Lose of work

Severity: HIGH
Probability: MODERATE

A number of measures have been put in place in order to combat a loss of work. Firstly, the entire project will be kept under version control hosted on a private GitHub repository. All meaningful changes, upon completion, will be committed to the repository effectively "backing up" the project on the internet.
If for some reason the repository is corrupted or lost the files will remain on the hard drive on the machine used for development.
A local hourly backup will also be automatically made onto an external hard drive should both GitHub and the main development machine should fail together. This means at most an hour of work will be lost.

The repository will also be semi-regularly pulled onto a laptop for another safe copy of the project to be stored as a last resort.

3.6.2 Server crashes/issues

Severity: MODERATE
Probability: MODERATE

In order to combat server crashes or other server issues from seriously damaging the progress of this project, the server will be run from a local Linux machine, allowing for easy access in the event that anything should go wrong.
The server has a battery backup in the event of a power outage.

3.6.3 Breaking Code

Severity: LOW
Probability: HIGH

To prevent unknowingly breaking already implemented systems, unit testing will be added to the code base. This will highlight any old code that gets broken while developing new sections of the system.

References

[1] R. G. Games, *Arctic scavengers rulebook*, 2012. [Online]. Available: http://riograndegames.com/uploads/Game/Game_384_gameRules.pdf.

[2] D. X. Vaccarino, *Dominion rulebook*, 2008. [Online]. Available: http://riograndegames.com/uploads/Game/Game_278_gameRules.pdf.

[3] S. Entertainment, *Ascension rulebook*, 2018. [Online]. Available: http://ascensiongame.com/game/how-to-play/.

[4] W. W. Games, *Star realms learn to play*.

[5] S. iT, *Dominion online*, 2018. [Online]. Available: https://dominion.games/.

[6] W. W. Games, *Star realms digital deckbuilder*, 2018. [Online]. Available: https://www.starrealms.com/digital-game/.

[7] N. Foundation, *Node.js*, 2018. [Online]. Available: https://nodejs.org/en/.

[8] P. Foundation, *P5.js*, 2018. [Online]. Available: https://p5js.org/.

[9] O. S. Project, *Jest - delightful javascript testing*, 2018. [Online]. Available: https://github.com/facebook/jest.

[10] D. Nakamura, *So what exactly is a deck-building game anyway?*, 2014. [Online]. Available: https://www.destructoid.com/so-what-exactly-is-a-deck-building-game-anyway--283188.phtml.

[11] D. X. Vaccarino, *The secret history of dominion*, 2013. [Online]. Available: https://www.boardgamegeek.com/thread/996671/secret-history-dominion.

[12] Alderac, *Thunderstone rulebook*, 2009. [Online]. Available: https://www.alderac.com/thunderstonerules.pdf.

[13] C. G. Labs, *Shadowrun:crossfire rulebook*, 2013. [Online]. Available: http://www.shadowruntabletop.com/wp-content/uploads/CAT27700_Crossfire_Full%20Rules.pdf.

Appendix B: Dominion rulebook [5]

Donald X. Vaccarino

You are a monarch, like your parents before you, a ruler of a small pleasant kingdom of rivers and evergreens. Unlike your parents, however, you have hopes and dreams! You want a bigger and more pleasant kingdom, with more rivers and a wider variety of trees. You want a Dominion! In all directions lie fiefs, freeholds, and feodums. All are small bits of land, controlled by petty lords and verging on anarchy. You will bring civilization to these people, uniting them under your banner.

But wait! It must be something in the air; several other monarchs have had the exact same idea. You must race to get as much of the unclaimed land as possible, fending them off along the way. To do this you will hire minions, construct buildings, spruce up your castle, and fill your treasury. Your parents wouldn't be proud, but your grandparents, on your mother's side, would be delighted.

Goal

This is a game of building a deck of cards. The deck is your Dominion. It contains your resources, victory points, and the things you can do. It starts out a small sad collection of Estates and Coppers, but you hope by the end of the game it will be brimming with Gold, Provinces, and the inhabitants and structures of your castle and kingdom.

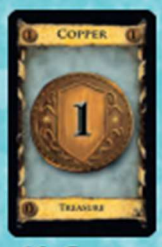
The player with the most victory points in his Deck at game end wins.

Contents

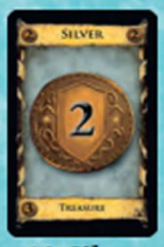
Before the first game, remove the five sets of cards from their wrappings and place them in the card tray. One side of the included inlay suggests a way to organize the cards. The opposite side allows players to create an organization that fits their needs.

500 cards

130 Treasure cards



60 Copper



40 Silver

30 Gold

name
value (3 coins)
card type (treasure, yellow)
cost (6 coins)

48 Victory cards

24 Estates

12 Duchies

12 Provinces

name
value (6 victory points)
card type (victory, green)
cost (8 coins)

30 Curse cards

name
value (-1 victory points)
card type (curse, purple)
cost (0 coins)

7 blank cards

32 Randomizer cards

1 Trash card

2


252 Kingdom cards
24 Action cards (10 of each)
CELLAR
CHAPEL
CHANCELLOR
VILLAGE
WOODCUTTER
FEAST
MILITIA
MONEYLENDER
REMODEL
SMITHY
SPY
THIEF
THRONE ROOM
COUNCIL ROOM
FESTIVAL
LABORATORY
LIBRARY
MARKET
MINE
WITCH
ADVENTURER
WORKSHOP
name
card ability
card type (action, white)
cost (3 coins)
Most kingdom cards are action cards, but there are other kinds.
action-attack
BUREAUCRAT
name
card ability
card type (action-attack, , white)
cost (4 coins)
action-reaction
MOAT
name
card abilities
card type (action-reaction, , blue)
cost (2 coins)
1 Victory card (12)
GARDENS
name
card ability
card type (victory, green)
cost (4 coins)
3

Preparation

Place the Treasure cards, Victory cards, Curse cards, and the Trash card in every game (players may place them as shown or in any other arrangement that is convenient for them.

TREASURE CARDS

Copper, Silver, and Gold cards are the basic Treasure cards, and they are available in every game. After each player takes 7 Copper cards, place the remaining Copper cards and all of the Silver cards and Gold cards in face-up piles in the Supply.

VICTORY CARDS

Estate, Duchy, and Province cards are the basic Victory cards, and they are available in every game. After each player takes 3 Estate cards, place 12 each of the Estate, Duchy, and Province cards in face-up piles in the Supply **in a 3 or 4 player game**. **In a 2 player game**, place only 8 of each of these Victory cards in the Supply. Place unused Victory cards back in the box.

CURSE CARDS & TRASH CARD

Place 10 Curse cards in the Supply for a 2 player game, 20 Curse cards for 3 players, and 30 Curse cards for 4 players. Return unused Curse cards to the box. Curse cards are used most often with specific Action cards (e.g. Witch). If a player buys a Curse card (0 cost), it goes in his own discard pile, like any other gained card. This, of course, will not often occur. The Trash card marks the place where players place cards trashed in the game.

the Supply

Treasure cards

Victory cards

Curse cards

Note: Curse cards are present in every game, however, they are rarely used in the basic game other than with the Witch card.

Trash pile

each player starts with 10 cards:

Each player starts the game with the same cards:
7 coppers
&
3 estates.

Each player shuffles these cards and places them (his Deck) face-down in his play area (the area near him on the table).

Now, each player draws 5 cards from his Deck. These cards are the player's hand.

4

Place 10 sets of the 25 different Kingdom cards next to the Treasure, Victory, Curse, and Trash cards to start each game. Thus, each game can have a different set of 10 cards.

KINGDOM CARDS

In addition to the Trash, Treasure, Victory, and Curse cards that are used in every game, the players also select 10 Kingdom cards and place 10 of each in face-up piles on the table.

Exception: Kingdom Victory card piles (e.g. Gardens) have the same number as the Victory card piles (12 for a 3 or 4 player game and 8 for a 2 player game).

For the first game, we recommend using the following 10 Kingdom cards: Cellar, Market, Militia, Mine, Moat, Remodel, Smithy, Village, Woodcutter, and Workshop. At the end of the rules, we list more suggestions for sets of 10 Kingdom cards. Return Kingdom cards not chosen for the game to the box.

In later games, players can choose the 10 Kingdom cards using any method they agree on.
For example, the players can shuffle the Randomizer cards for all Kingdom cards and draw 10 to select the cards for the game. Or, players can take turns selecting cards.

Players may also use the Randomizer cards as Placeholders to mark the card piles so empty piles are easily seen.

example of a player's play area during the game:

Deck *(always face-down)* | cards played this turn | Discard pile *(always face-up)*

Playing the Game

STARTING PLAYER

Randomly determine the starting player. When playing multiple games, the starting player is the player to the left of the winner of the last game. If there was a tie in the previous game, randomly choose the starting player from the players that didn't win. Players take turns in clockwise order.

Players choose starting player randomly or based on previous game. Players take turns in clockwise order.

TURN OVERVIEW

Each turn has three phases (A, B, and C) in the order shown:
A) Action phase - the player may play an Action.
B) Buy phase - the player may buy a card.
C) Clean-up phase - the player must discard both played and unplayed cards and draws five new cards.
After a player completes all three phases, his turn ends.

Each turn, the player does the A, B, and C phases in order:
A) Action phase
B) Buy phase
C) Clean-up phase

ACTION PHASE

In the Action phase, the player may play one Action card. Action cards are the Kingdom cards that say "Action" at the bottom of the card. Since players do not start the game with any Action cards in their initial Decks of 10 cards, a player will not have any Actions to play during his first 2 turns. Normally, a player may play only one Action card, but this number may be modified by the Action cards that the player plays.

The player may play one action card if he has one. This is optional, even if the player has an action card, he need not play it. Action cards will allow players to do extra things during their turns.

To play an Action, the player takes an Action card from his hand and lays it face-up in his play area. He announces which card he is playing and follows the instructions written on that card from top to bottom. The player may still play an Action card even if he is not able to do everything the Action card tells him to do; but the player must do as much as he can. Furthermore, the player must fully resolve an Action card before playing another one (if he is able to play another Action card). Detailed information about card abilities can be found in the card descriptions at the end of these rules. Any Action cards played remain in the player's play area until the Clean-up phase of the turn unless otherwise indicated on the card.

As some action cards offer a player additional actions, a player may be able and choose to play several action cards in a turn. Players can play their action cards left to right in their play areas. In this way, they can easily keep track of what and how many extra things they may do. The player will discard these cards in the clean-up phase (see below), and should not be discarded prior to this.
Note: *as the players begin the game with no action cards, they will be unable to play action cards for at least the first two turns.*

The Action phase ends when the player cannot or chooses not to play any more Action cards. Generally, a player can only play Action cards during the Action phase of his turn. However, Reaction cards are an exception to this rule as they can be used at other times.

Common terms used on the Action cards:
"+X Card(s)" – the player immediately draws X number of cards from his Deck. If there are not enough cards in his Deck, he draws as many as he can, shuffles the Discard pile to form a new Deck, and then draws the rest. If he still does not have enough cards left after forming a new Deck, he just draws as many as he can.

+ X Card(s): must draw X more Cards immediately

6

"+X Action(s)" – the player may play X number of additional Actions this turn. +X Action(s) adds to the number of Actions that can be played in the Action phase. It does not mean play another Action immediately. The instructions on the current Action card must be completed before playing any additional Actions. The player must complete all of his Actions before he moves on to the Buy phase of his turn. If a card gives the player more than one additional Action, he may keep track of the number of Actions he has remaining out loud.

+ X Action(s): can play X more Actions in Action phase
If a card gives the player more than one additional Action, it is helpful to keep track of the number of Actions he has remaining out loud.

"+ X " – the player has X number of additional coins to spend in the Buy phase. The player does not take additional Treasure cards for these coins.

+ X : can spend X more coins this turn

"+1 Buy" – the player may buy an additional card from the Supply during the Buy phase of his turn. +1 Buy adds to a player's potential Buys, it does not allow the player to buy a card during the Action phase.

+1 Buy: can buy 1 more card in Buy phase

"Discard" – unless otherwise specified, discarded cards are from the player's hand. When a player discards a card, he places the discarded card face-up onto his Discard pile. When discarding several cards at once, the player need not show all cards he is discarding to his opponents, but player may need to show how many cards he is discarding (for example, when playing the Cellar). The top card of a player's Discard pile is always visible.

Discard: put cards face-up in your Discard pile

"Trash" – when a player trashes a card, he places it in the Trash pile, not his Discard pile. Trashed cards are not returned to the Supply and are not available for purchase.

Trash: put card(s) in the Trash pile

"Gain" – when a player gains a card, he takes the gained card (usually from the Supply) and puts it onto his Discard pile (unless the card says to put it elsewhere). The player does not get to use the card when he gains it.

Gain: take a card and put it in your Discard pile

"Reveal" – when a player reveals a card, he shows a card to all players and then returns it to wherever it came from (unless instructed specifically to put it elsewhere). If the player is required to reveal cards from the top of his Deck, and he does not have enough cards, he shuffles in order to reveal the required number of cards.

Reveal: show card(s) and return them to where they came from

"Set Aside" – when a player sets aside a card, he places it face-up on the table (unless otherwise indicated) without following any instructions on the card. An Action that requires a player to set aside cards will instruct him on what to do with these cards.

Set Aside: put cards aside until the instructions indicate where they go

BUY PHASE

In the Buy phase, the player can gain one card from the Supply by paying its cost. Any card that is in the Supply may be purchased (Treasure, Victory, Kingdom, and even Curse cards). The player may not purchase cards from the Trash pile. Normally, a player may buy only one card, but he may buy more if he played certain cards earlier in his Action phase.

The player can gain one card from the Supply by buying it - paying the cost shown on the card. The player pays in coins from Treasure cards (the number on the coin) and from previously paid Action cards.

The cost of a card is in its lower left corner. The player may play some or all of the Treasure cards from his hand to his play area and add to their value the coins provided by Action cards played this turn. The player may then gain any card in the Supply of equal or lesser value. He takes the purchased card from its Supply pile and places it face-up on his Discard pile. He my not use the ability of the card when it is gained.

The player may use any combination of Treasure cards from his hand and coins shown on Action cards played this turn.

7

If the player has multiple Buys, he combines Treasure cards and any coins available from Action cards to pay for all of the purchases. For example, if Tyler has +1 Buy and 6 coins provided by two Gold cards, he can buy a Cellar costing 2, placing it face-up in his Discard pile. Then, he can buy a Smithy with the remaining 4 coins and place that face-up in his Discard pile. If he wants to use all 6 coins to buy one card, he can buy a Copper (for free) with his second Buy or not buy a second card. Players do not have to use any or all of their Buys.

Any Treasure cards played can be placed in his player area from left to right, adding them to any cards previously played this turn.

The Treasure cards remain in the play area until the Clean-up phase. Treasure cards will be used multiple times during the game. Although they are discarded during the Clean-up phase, the player will draw them again as his Discard pile is shuffled into a new Deck. Thus, Treasure cards are a source of income, not a resource that is used up when played. When played, Coppers are worth 1 coin, Silvers are worth 2 coins, and Golds are worth 3 coins.

All these cards will be discarded at the end of the turn, and should not be discarded prior to the Clean-up phase (see below).

CLEAN-UP PHASE

All cards gained this turn should already be in the player's Discard pile. The player places any cards that are in his play area (Action cards that have been played in the Action phase as well as Treasure cards that have been played in the Buy phase) and any cards remaining in his hand onto his Discard pile. Although the player need not show the cards remaining in his hand to his opponents, since he places the cards in the Discard pile face-up, his opponents will always be able to see the top-most card of his Discard pile.

The player places all cards in his play area onto his Discard pile. This will include all Action cards and Treasure cards he played during this turn. He also places all cards left in his hand onto his Discard pile.

Then, the player draws a new hand of 5 cards from his Deck. If there are not enough cards in his Deck, he draws as many as he can, shuffles his Discard pile to form a new face-down Deck, and then draws the rest of his new hand.

Draw 5 cards from his Deck.

Once the player has drawn a new hand of 5 cards, the next player starts his turn. To speed play, players may begin their turns while previous players are completing their Clean-up phases. When someone plays an Attack card, the players must complete their Clean-up phases in order to properly resolve the Attack.

The player's turn is over. Play passes clockwise.

GAME END

Game end

The game ends at the end of any player's turn when **either**:
1) the Supply pile of Province cards is empty **or**
2) any 3 Supply piles are empty.

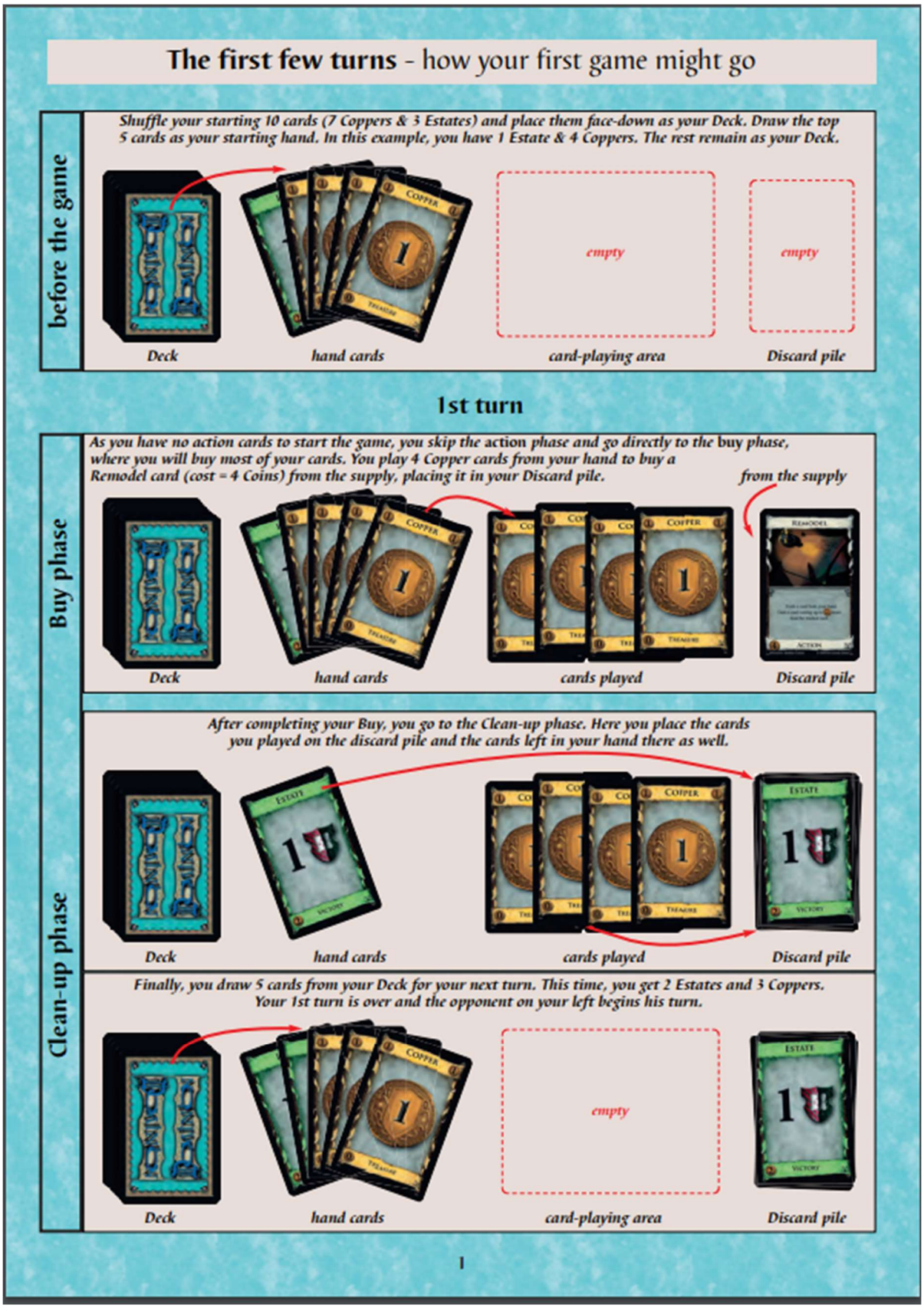
1) Province card pile is empty
or
2) any 3 Supply piles are empty

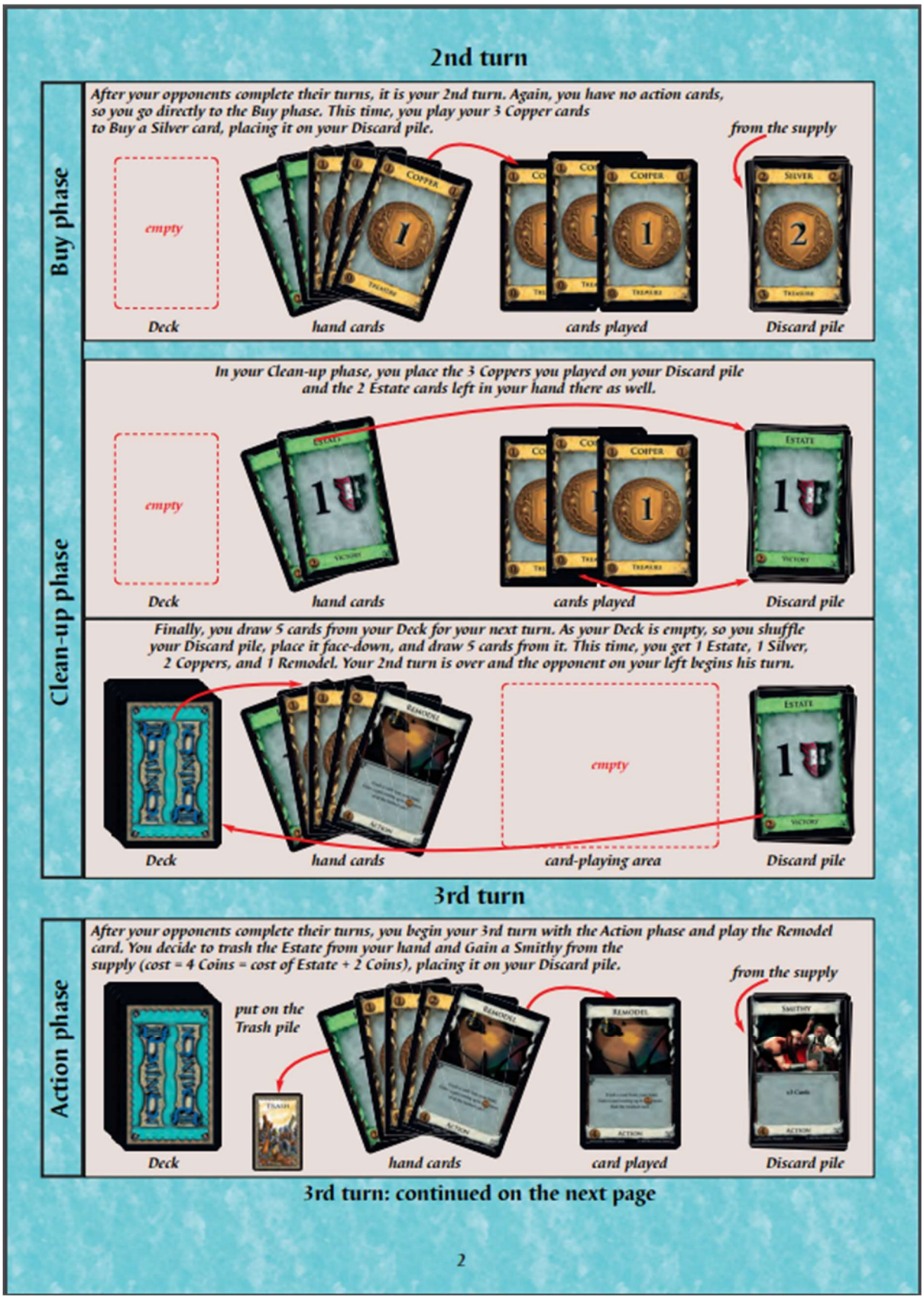
Each player puts all of his cards into his Deck and counts the victory points on all the cards he has.

Players count their victory points.

The player with the most victory points wins. If the highest scores are tied at the end of the game, the tied player who has had the fewest turns wins the game. If the tied players have had the same number of turns, they rejoice in their shared victory.

Most victory points wins.
Ties go to the player with the fewest turns


The first few turns - how your first game might go
before the game
Shuffle your starting 10 cards (7 Coppers & 3 Estates) and place them face-down as your Deck. Draw the top 5 cards as your starting hand. In this example, you have 1 Estate & 4 Coppers. The rest remain as your Deck.
empty
empty
Deck
hand cards
card-playing area
Discard pile
1st turn
Buy phase
As you have no action cards to start the game, you skip the action phase and go directly to the buy phase, where you will buy most of your cards. You play 4 Copper cards from your hand to buy a Remodel card (cost = 4 Coins) from the supply, placing it in your Discard pile.
from the supply
Deck
hand cards
cards played
Discard pile
Clean-up phase
After completing your Buy, you go to the Clean-up phase. Here you place the cards you played on the discard pile and the cards left in your hand there as well.
Deck
hand cards
cards played
Discard pile
Finally, you draw 5 cards from your Deck for your next turn. This time, you get 2 Estates and 3 Coppers. Your 1st turn is over and the opponent on your left begins his turn.
empty
Deck
hand cards
card-playing area
Discard pile
1


2nd turn
Buy phase
After your opponents complete their turns, it is your 2nd turn. Again, you have no action cards, so you go directly to the Buy phase. This time, you play your 3 Copper cards to Buy a Silver card, placing it on your Discard pile.
from the supply
empty
Deck
hand cards
cards played
Discard pile
Clean-up phase
In your Clean-up phase, you place the 3 Coppers you played on your Discard pile and the 2 Estate cards left in your hand there as well.
empty
Deck
hand cards
cards played
Discard pile
Finally, you draw 5 cards from your Deck for your next turn. As your Deck is empty, so you shuffle your Discard pile, place it face-down, and draw 5 cards from it. This time, you get 1 Estate, 1 Silver, 2 Coppers, and 1 Remodel. Your 2nd turn is over and the opponent on your left begins his turn.
empty
Deck
hand cards
card-playing area
Discard pile
3rd turn
Action phase
After your opponents complete their turns, you begin your 3rd turn with the Action phase and play the Remodel card. You decide to trash the Estate from your hand and Gain a Smithy from the supply (cost = 4 Coins = cost of Estate + 2 Coins), placing it on your Discard pile.
from the supply
put on the Trash pile
Deck
hand cards
card played
Discard pile
3rd turn: continued on the next page
2


3rd turn (continued)
Buy phase
You have 4 Coins in your hand for the Buy phase. You play the 4 Coins (2 Copper cards and 1 Silver card) and decide to Buy a Militia card from the Supply, placing it in your Discard pile.
from the supply
Deck
hand cards
cards played
Discard pile
Clean-up phase
After completing your Buy, you go to the Clean-up phase. Here you place the cards you played on the discard pile.
Deck
empty
hand cards
cards played
Discard pile
Finally, you draw 5 cards from your Deck for your next turn. This time, you get 2 Estates and 3 Coppers. Your 3rd turn is over and the opponent on your left begins his turn.
Deck
hand cards
empty
card-playing area
Discard pile
SAMPLE ACTION CARD
MARKET
+1 Card
+1 Action
+1 Buy
+1
5
ACTION
Name of card
Draw another card immediately
Lets you play another Action during your Action phase
Lets you buy an additional card in the Buy Phase
Gives you an additional coin to spend in the Buy phase.
Cost to buy the card.
Card type (both name and color of banner)
3

Additional Rules

Each player has his own Dominion, which he builds from cards in the supply. During the game, a player's cards are usually in three parts: his Deck (which he draws cards from), his hand, and his Discard pile. The player draws cards from his own Deck and discards cards to his own Discard pile. When his Deck is exhausted **and** the player needs to draw or reveal cards from his Deck, he shuffles his Discard pile to reform his Deck. He does not shuffle his Discard pile until he needs to reveal or draw a card from his Deck and cannot. At any point in the game, if a player has to draw or reveal more cards than are remaining in his Deck, he must draw or reveal as many as he can and then shuffle his face-up Discard pile to form a new face-down Deck. Then, he draws or reveals the remaining number of cards from his newly shuffled Deck.

Each player has his own Dominion, built from the cards in the supply.

Shuffle **only** when new cards needed.

A player does not shuffle his discard pile until **all** cards in his Deck have been drawn or revealed.

A player places cards he Buys or otherwise acquires during the game on his Discard pile unless he is specifically directed to place them elsewhere.

Gained cards go on the player's discard pile.

At the end of a player's turn, he places all the cards he played and those still in his hand on his Discard pile.

All hand and played cards are discarded at the end of a turn.

A player is allowed to count how many cards are left in his Deck, but not in his Discard pile. A player may not look through his Deck or his Discard pile. A player may look through the Trash pile, and players may count the number of cards left in any pile in the Supply.

Players may count cards in Decks and Supply piles and may look at cards in Trash.

If an ability of a card affects multiple players, and the order matters, resolve that ability for each affected player in turn order, starting with the player whose turn it is.

Use turn order to resolve card affects.

During each turn, a player is allowed 1 Action and 1 Buy, but may be entitled to more based on Action cards played. The instructions written on all the action cards alter the rules of the game by, for example, allowing the player to draw more cards from his Deck, play more Action cards in the Action phase, use more coins for the Buy phase, Buy extra cards in the Buy phase, and so on.

Player allowed at least 1 Action and 1 Buy.

All Action cards alter the rules in some way.

When an Action card allows a player to gain a card costing up to a certain value, he may not add coins from his hand or other action cards to gain a higher-valued card.

Players may not add coins to increase the affect of an Action beyond the instructions.

4

Kingdom Card Description

Adventurer – If you have to shuffle in the middle, shuffle. Don't shuffle in the revealed cards as these cards do not go to the Discard pile until you have finished revealing cards. If you run out of cards after shuffling and still only have one Treasure, you get just that one Treasure.

Bureaucrat – If you have no cards left in your Deck when you play this card, the Silver you gain will become the only card in your Deck. Similarly, if another player has no cards in his Deck, the Victory card he puts on top will become the only card in his Deck.

Cellar – You can't discard Cellar to itself, since it isn't in your hand any longer when you resolve it. You choose what cards to discard and discard them all at once. You only draw cards after you have discarded. If you have to shuffle to do the drawing, the discarded cards will end up shuffled into your new Deck.

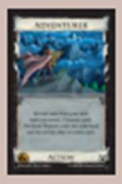
Chancellor – You must resolve the Chancellor (decide whether or not to discard your Deck by flipping it into your Discard pile) before doing other things on your turn, like deciding what to buy or playing another Action card. You may not look through your Deck as you discard it.

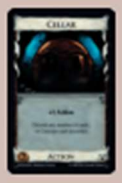
Chapel – You can't trash the Chapel itself since it isn't in your hand when you resolve it. You could trash a different Chapel card if that card were in your hand.

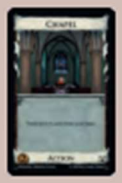
Council Room – The other players must draw a card whether they want to or not. All players should shuffle as necessary.

Feast – The gained card goes into your Discard pile. It has to be a card from the Supply. You cannot use coins from Treasures or previous Actions (like the Market) to increase the cost of the card that you gain. If you use Throne Room on Feast, you will gain two cards, even though you can only trash Feast once. Gaining the card isn't contingent on trashing Feast; they're just two things that the card tries to make you do.

Festival – If you are playing multiple Festivals, keep a careful count of your Actions. Say how many you have left out loud; this trick works every time (i.e. "I'm playing the Festival and now have two Actions remaining. I play a Market and have two Actions remaining. I play another Festival and now have three Actions remaining....").








Gardens – This Kingdom card is a Victory card, not an Action card. It does nothing until the end of the game, when it is worth 1 victory point per 10 cards in your Deck (counting all of your cards – your Discard pile and hand are combined with your Deck before scoring). Round down; if you have 39 cards, Gardens is worth 3 victory points. During set-up, place 12 Gardens in the Supply for a 3 or 4 player game and 8 in the Supply for a 2 player game.

Laboratory – Draw two cards. You may play another Action card during your Action phase.

Library – If you have to shuffle in the middle, the set-aside cards are not shuffled into the new Deck. They will be discarded when you have finished drawing cards. If you run out of cards even after shuffling, you just get however many there were. You are not obligated to set aside Actions – you just have the option to do so. If you have 7 or more cards in hand after you play the Library, you don't draw any cards.

Market – Draw a card. You may play another Action card during your Action phase. During your Buy phase, you may buy an additional card from the supply, and add one coin to the total value of the Treasure cards played.

Militia – The attacked players discard cards until they have only 3 cards in hand. Players who had 3 or fewer cards in hand when Militia was played do not discard any cards.

Mine – Generally, you can trash a Copper card and gain a Silver, or trash a Silver card and gain a Gold. However, you could also trash a Treasure to gain the same Treasure or a cheaper one. The gained card goes in your hand; thus, you can spend it the same turn. If you don't have a Treasure card in your hand to trash, you can't gain anything.

Moat – An Attack card is one that says "Attack" on the bottom line (usually, "Action - Attack"). When someone else plays an Attack card, you may reveal the Moat by showing it from your hand to the other players and then returning it to your hand (before the Attack card resolves). You are then unaffected by that Attack card. You won't gain a Curse because of a Witch or reveal a card to a Spy, and so on. It's just like you aren't in the game for purposes of resolving that Attack. Moat doesn't stop anything an Attack does to other players or to the player of the Attack; for example, if everyone else Moats a Witch, the person who played it still gets to draw 2 cards. Moat can also be played on your turn as an Action to draw 2 cards.

6

Moneylender – If you do not have a Copper in your hand to trash, you don't get the +3 coins to spend in the Buy phase.

Remodel – You cannot trash the Remodel as it isn't in your hand when you resolve it (you can trash a different Remodel card from your hand). If you do not have a card to trash, you cannot gain a card from the Remodel. The gained card goes in your Discard pile. You can only gain cards from the Supply. The gained card need not cost exactly 2 coins more than the trashed card; it can cost that much or any amount less. You cannot use coins from Treasures or previous Actions (like the Market) to increase the cost of the card you gain. You can trash a card to gain a copy of the same card.

Smithy – Draw three cards.

Spy – Spy causes all players, including the one who played it, to reveal the top card of their Deck. Note that you draw your card for playing Spy before any cards are revealed. Anyone who does not have any cards left in their Deck shuffles in order to have something to reveal. Anyone who still has no cards to reveal doesn't reveal one. If players care about the order in which things happen for this, you do yourself first, then each other player in turn order. Revealed cards that aren't discarded are returned to the top of their players' Decks.

Thief – A player with just one card left reveals that last card and then shuffles to get the other card to reveal (without including the revealed card); a player with no cards left shuffles to get both of them. A player who still doesn't have two cards to reveal after shuffling just reveals what he can. Each player trashes one Treasure card at most, of the attacker's choice from the two revealed cards, and then you gain any of the trashed cards that you want. You can only take Treasures just trashed—not ones trashed on previous turns. You can take none of them, all of them, or anything in between. Put the Treasures you decided to gain into your Discard pile. The ones you choose not to gain stay in the Trash pile.

Throne Room – You pick another Action card in your hand, play it, and play it again. The second use of the Action card doesn't use up any extra Actions you have. You completely resolve playing the Action the first time before playing it the second time. If you Throne Room a Throne Room, you play an Action, doing it twice, and then play another Action and do it twice; you do not resolve an Action four times. If you Throne Room a card that gives you +1 Action, such as Market, you will end up with 2






Actions left afterwards, which is tricky, because if you'd just played Market twice you'd only have 1 Action left afterwards. Remember to count the number of Actions you have remaining out loud to keep from getting confused! You cannot play any other Actions in between playing the Throne Roomed Action twice.

Village – If you're playing multiple Villages, keep a careful count of your Actions. Say how many you have left out loud; this trick works every time.

Witch – If there aren't enough Curses left to go around when you play the Witch, you deal them out in turn order – starting with the player after you. If you play Witch with no Curses remaining, you will still draw 2 cards. A player gaining a Curse puts it face-up into his Discard pile.

Woodcutter – During your Buy phase, you add two coins to the total value of the Treasure cards played, and you may buy an additional card from the Supply.

Workshop – The card you gain is put into your Discard pile. It has to be a card from the Supply. You cannot use coins from Treasures or previous Actions (like the Market) to increase the cost of the card you may gain.

Recommended sets of 10

You can play Dominion with any set of 10 Kingdom cards, but these sets are intended to highlight some interesting card interactions and game strategies.

First Game: Cellar, Market, Militia, Mine, Moat, Remodel, Smithy, Village, Woodcutter, Workshop.

Big Money: Adventurer, Bureaucrat, Chancellor, Chapel, Feast, Laboratory, Market, Mine, Moneylender, Throne Room

Interaction: Bureaucrat, Chancellor, Council Room, Festival, Library, Militia, Moat, Spy, Thief, Village

Size Distortion: Cellar, Chapel, Feast, Gardens, Laboratory, Thief, Village, Witch, Woodcutter, Workshop

Village Square: Bureaucrat, Cellar, Festival, Library, Market, Remodel, Smithy, Throne Room, Village, Woodcutter

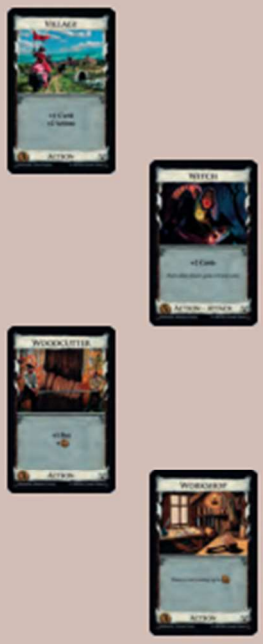


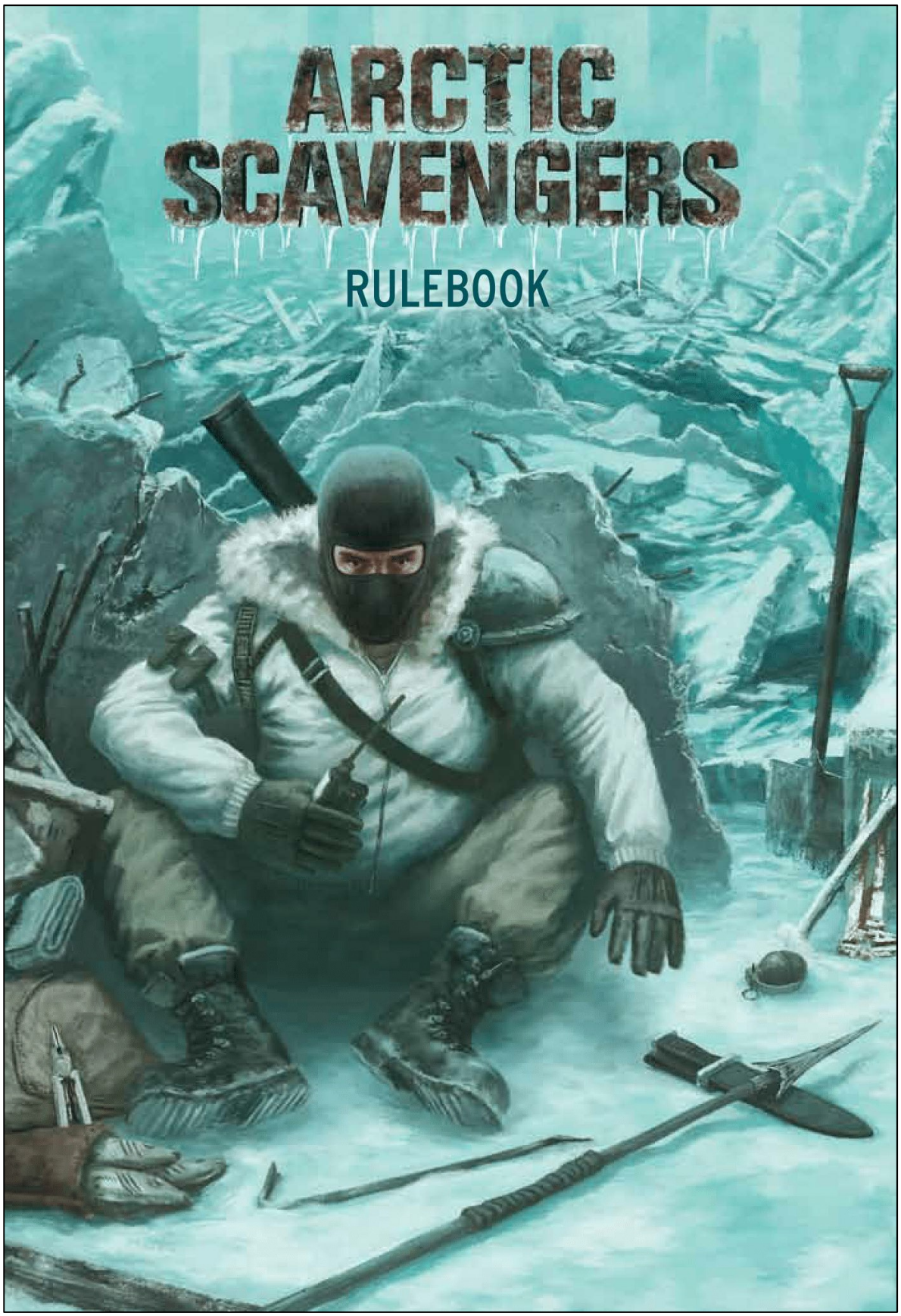
Illustration: Matthias Catrein

Developers: Valerie Putman & Dale Yu

Our thanks to our playtesters: Kelly Bailey, Dan Brees, Josephine Burns, Max Crowe, Ray Dennis, David Fair, Lucas Hedgren, Michael M. Landers, W. Eric Martin, Destry Miller, Miikka Notkola, Molly Sherwin, Sir Shufflesalot, P. Colin Street, Chris West, the 6am Gamers, the Cincygamers and the Columbus Area Boardgaming Society.

RIO GRANDE GAMES

Appendix C: Arctic Scavengers rulebook [3]



ARCTIC SCAVENGERS

players:	2 – 5	learning:	5 – 10 minutes
preparation:	3 minutes	game length:	45 – 60 minutes
age:	13+		

Contents – Basic Game



1 rule booklet
1 rule summary
1 junkyard mat
1 contested resources mat
1 initiator card
149 playing cards
- 20 refugees
- 69 mercenaries *(10 brawlers, 8 hunters, 8 saboteurs, 8 scouts, 5 group leaders, 5 sniper teams, 5 thugs, 20 scavengers)*
- 46 junkyard cards *(7 junk, 4 multitools, 4 nets, 6 spears, 4 pickaxes, 6 shovels, 6 medkits, 9 pills)*
- 14 contested resources *(2 wolf packs, 2 grenades, 2 sled teams, 2 field crews, 6 tribe families)*

Contents – HQ Expansion



1 expansion rules summary
1 engineering schematics mat
1 storage cover
49 playing cards
- 12 buildings *(3 each of bunkers, armories, pharmacies, hydroponic gardens)*
- 10 tribal leaders
- 8 junkyard cards *(4 rifles, 4 toolkits)*
- 8 medics
- 8 engineers
- 3 gangs *(1 each of the Gearheads, the Pharmers, the Masons)*

The Story

In the year 2097, the entire earth was enveloped in a cataclysmic climate shift plunging the globe into another ice age. Over 90% of the world's population was eliminated, driving the survivors to band together into loose communities and tribes.

Each player is the leader of a small tribe of survivors. Resources, tools, medicine, and mercenaries are all in scarce supply. Each tribe is pitted against other tribes in a fight for survival. The players build up their tribes, skirmish against other players' tribes, and can even bluff on the way to victory.

The leader who gathers the largest tribe will win the game!

Game Play Overview

Players work to build their tribes as large as possible by hiring mercenaries, scavenging junk piles and winning skirmishes against other players' tribes. Each tribe member card in a tribe represents the number of people shown on the card. The player with the largest tribe (most people) at the end of the game is the winner.

2

Anatomy of an Arctic Scavengers Card

The cards are designed to make it easy to identify the different types of cards, make sorting and game setup quicker, and facilitate faster decision-making during the game.

Understanding the above card anatomy

Green labels in the above graphic indicate something that is always present on a card. Red labels indicate something that may or may not be present, depending on the card. Gray labels are used to describe the sub-types of card actions.

Every card has a type (person, tool, medicine), title, image, and actions. Standard actions can be performed by this card alone, action modifiers must be combined with another card, and disabled actions cannot be used. Some cards will be sorted based upon supply pile icon, others by card title. Some cards have other identifiers (instructions, cost, and number of tribe members represented by the card).

Game Preparation

Arctic Scavengers can be played by 2 to 5 players. Each player takes an identical starter deck representing her tribe. That deck consists of 10 cards:

- 4 *refugee* cards
- 3 *scavenger* cards
- 1 *brawler* card
- 1 *spear* card
- 1 *shovel* card

3

Remove any additional *refugee* cards from the game. Sort the remaining cards into three categories: *contested resources* (✊), *junkyard* (⬢), and mercenaries. Shuffle the *contested resources* cards ✊ and place them face-down on the ✊ mat to form the *contested resources* stack. Shuffle *junkyard* cards ⬢ to form the *junkyard* pile and place them face-down on the ⬢ mat. The remaining cards (the ones with character portraits and a cost listed) are the mercenaries. Sort them into 8 separate stacks, by mercenary type. Place them face-up on the playing surface in separate stacks.
Each player shuffles her deck of 10 cards and places it face-down in her play area. Randomly select a player and hand her the initiator card. This player will serve as the *initiator* for the first round.

When Playing with only Two Players

- Remove two cards of each kind from the *junkyard* (i.e. 2 *junk*, 2 *medkit*, 2 *pickaxe*, etc.)
- Eliminate all *contested resources* peeking
- The number of tribe members a player brings to a skirmish is no longer relevant
- Skirmishes must be won by 2. If not, then it is a tie and the players place the *contested resource* on top of the *junkyard* and then shuffle the *junkyard*

PLAYING THE GAME

Each round of Arctic Scavengers has three phases, played in the order shown below:

Drawing Phase

1) Discard all cards from the previous round.
2) Each player draws 5 cards from the top of her deck, shuffling the discard pile and adding it to the deck when her deck becomes exhausted.
3) Beginning with the third round, and continuing for the remainder of the game, the initiator peeks at the top card on the *contested resources* stack and then returns it to the stack.

Resource Gathering Phase

4) The initiator takes the first action and uses any number of cards to gather resources.
5) The initiator announces how many of her cards will remain for the skirmish.
6) Play continues with the other players, in clock-wise order, and they execute steps 4 and 5.

Skirmish Phase (Round 3 and following)

7) Once all players have taken their resource gathering actions, players reveal their remaining cards for the skirmish and a winner is determined.
8) The winner of the skirmish takes the top card from the *contested resources* stack and adds it to her discard pile.
9) The initiator passes the initiator card to her left-hand opponent, starting a new round.

The game continues like this until the last *contested resource* card is won.

No Skirmish until the Third Round

The game begins with two rounds of resource gathering (*digging* in the *junkyard*, *hunting* for *food*, and *hiring* mercenaries). In the first two rounds, there is no skirmishing. This allows each player to get a solid foundation prior to skirmishing over the *contested resources*.

Peeking at the Contested Resources

Beginning with the third round, the initiator looks at the top card of the *contested resources* stack. The player does not reveal this card to the other players. After looking, the player returns the card, face-down, to the top of the stack. This provides the first player of each round with special insight regarding what resource will be contested over during the skirmish phase.

Gathering Resources

The first phase of each round involves gathering resources. There are three primary resource gathering actions: *dig, hunt,* and *hire*. *Dig* allows the player to retrieve cards from the *junkyard*, *hunt* enables the player to get *food* (used as currency during that round), and *hire* uses a combination of *Food* from *hunting* and *medicine* from the player's hand to serve as currency for hiring mercenaries to join the player's tribe.

Announcing Cards

Players save cards not used for resource gathering for the skirmish. After a player has performed her resource gathering, she puts all cards already used in her discard pile to clear her play area. She then places her remaining cards in her play area, face down, and announces the number of cards she is taking to the skirmish.

Bluffing

Sometimes a player has additional tools that cannot be used, or *refugee / tribe family* cards that have no tools. The player should carry these into the skirmish anyway. This presents an apparently stronger hand to her opponents and may impact the decisions that they make.

Winning a Skirmish

When resolving a skirmish, the player with the highest *fight* score wins. Several factors affect a player's *fight* score: each player totals the attack abilities of each of her tribe members, adding the modifiers, and factoring in special actions such as *disarm* and *snipe* (explained later).

Winning the Game

When the final *contested resource* card has been won, the game is over. The winner is the player who has built the largest tribe. Players determine the size of their tribes by counting the number of tribe member icons on the lower right corner of every card in their decks. If there is a tie, then the player among those tied with the most *contested resource* cards is the winner. If there is still a tie, the player among those tied with the largest number of cards in her deck is the winner.

If there is still a tie, the tied players rejoice in their shared victory.

Key Concepts

Arctic Scavengers uses a handful of basic rules. Detailed rules will follow, but these form the core rule set.

- A player may take multiple actions in a round
- Multiple cards used in a single action increase the potency of that action
- A player may only perform a given action (*dig*/*hunt*/*draw*/*hire*/etc.) once per round
- Once a card has been used to perform an action, it may not perform another action that round.
- To be useful, tools, *group leaders*, and any card with a modifier (i.e. +1, +2, +4) must be combined with another card that has the base ability.
- Only one tool card may be used by a tribe member card, even if that card represents multiple people.
- When a player needs to *draw* more cards than are available in her deck, she first draws all cards from her deck, then shuffles her discard pile to create a new deck and continues.

GAME DETAILS

Now that you understand the game flow, this section describes in detail the various actions that are available and the game scenarios that players are likely to encounter.

Resource Gathering Phase

During this phase, each player plays any number of cards from her hand in order to add more resources (cards) to her deck. Gathering starts with the player that has the initiator card (lying in her play area) and play continues in clockwise order.
There is no limit to the number of actions that a player can take during the resource gathering phase, although **no action may be taken more than once.**

Dig

The player draws one or more cards from the top of the *junkyard* stack. The player may choose one to place in her discard pile and returns any other cards to the bottom of the *junkyard* pile. The number of cards is determined by the sum of all the *dig* numbers displayed on the card(s) played from the player's hand. As with other actions, *draw* may only be used once per round.

For example: Natalie plays a brawler card in her play area and declares a dig action. Since a brawler has a dig value of 1, Natalie draws one card from the junkyard and places it in her discard pile. If Natalie were to also play a shovel, then the combined dig score would be 3, as shovel adds 2 to dig. Thus, Natalie would draw 3 cards from the junkyard, select one to keep (place on her discard pile) and return the other 2 to the bottom of the junkyard.

Draw

The player *draws* one or more cards from her deck, adding them to her hand. Multiple cards may be combined to *draw* several cards. As with other actions, *draw* may only be used once per round.

For example: Anna plays a scout with a draw of 2 and a sled team with a draw of 2 in her play area. This combination enables Anna to draw 4 cards from her deck and add them to her hand. Anna could also play two scavengers and a scout to yield the same result.

6

Hire

The *hire* ability is a free action (no card is needed to grant this action) in which a player takes a mercenary card from the face-up piles of cards. The player must be able to pay the cost displayed on the card. There are two currencies: *food* and *medicine* (*meds*). *Food* is created in real-time using the Hunt action (described below). *Meds* are provided in the form of *medicine* cards (*pills* and *medkit*) that must be played from the players hand in order to complete the purchase of a mercenary. *Medicine* cards do not require an associated person to spend. **Each player may only *hire* one mercenary per round, and the player immediately adds the card to her discard pile.**

Hunt

The hunt ability generates *food* during the round that can then be used as currency for purchasing a single mercenary card. The amount of *food* generated is determined by summing all hunt abilities played from the player's hand. The player can assign tools to the mercenaries to improve their hunt abilities.

> *For example: Natalie plays two scavenger cards and a spear card. Each scavenger has a hunt of 1 and the spear grants one of those scavengers an additional hunt of 1. Her combined hunt score is 3. Natalie can now spend 3 food this round to hire a mercenary card.*

Trash

The trash ability is a free action (no card is needed to grant this action) that allows players to take any number of cards from their hands and add them to the *junkyard* supply pile. The player **does not** *draw* cards from her deck to replace these cards.

Special Actions

Some cards may grant special actions that can be used during the resource gathering phase. These cards are *saboteur* (*disarm* another player's tool) and *sniper team* (*snipe* another player's tribe member).

Shuffling the Junkyard

Throughout the course of the game, players may add cards to the *junkyard*. This can occur when players use the Trash action (see above) or when skirmishing for a *contested resource* (see below) ends in a tie. When cards are added to the *junkyard*, players place them face-down on top of the *junkyard* and then shuffle the entire *junkyard* deck.

Using Tools

Each player starts with two tools – a *spear* and a *shovel*. Players can find additional tools by *digging* in the *junkyard*. Tools cannot perform actions on their own (these are inanimate, objects after all). A tribe member can use a single tool that enhances the ability of the action the tribe member is performing. Using a tool does not grant a tribe member an ability that she cannot inherently perform.

> *For example: Anna could combine her scavenger (dig 1 / draw 1 / fight 1/ hunt 1) with a shovel (fight +1 / dig +2) to achieve a dig of 3. If Anna would prefer to use her scavenger to hunt, the shovel could not be used to improve the scavenger's hunt value, since the shovel does not have the hunt ability. Thus, Anna could not use the shovel this round to dig unless she has another tribe member card (perhaps a refugee) capable of digging that could use the shovel.*

7

Playing Multiple Cards

Playing multiple cards for an action improves the results. This could involve playing multiple people or people with tools. The total of all ability values played is used to perform that action (provided that the total is 1 or greater).

Taking Multiple Actions

Although many cards support multiple actions, once a card has been played for a given action it cannot be used for another action during that round.

For example: if Natalie plays a refugee card, this can be used to either dig in the junkyard or hunt for food. Under no circumstances could she use the card for both actions.

Skirmish Phase

Once all the players have completed their resource gathering actions, the skirmish phase begins. All players simultaneously reveal the cards that they have saved for the skirmish. The sum of the *fight* values from all of a player's cards represents the strength of that player's attack. All rules related to tool usage also apply to tools used as weapons during the skirmish phase (see 'Using Tools' above).

Skirmishes are resolved by starting with the Initiator and having her declare any actions performed by her units (including saboteurs and sniper teams). Play then passes to the next player in clockwise order to declare actions for her units (provided that they were not sniped or disarmed by the previous player). This continues until all skirmish actions have been resolved. Then the total remaining *fight* values are summed and the player with the highest *fight* value wins. The winning player draws the top card from the *contested resources* supply pile and adds it to her discard pile without revealing it to the other players.

Resolving Ties During the Skirmish

In the event of a tie, each player sums **the number of people** involved in the skirmish (this includes refugees and tribe families that may not be directly contributing to the *fight* score) and the player with the most people involved wins the skirmish. **If players are still tied, then the skirmish is considered a deadlock.** The players place the top card from the *contested resources* pile into the *junkyard* pile without looking at it. The players shuffle the *junkyard* and the round ends. The player with the initiator card passes it to the left to begin a new round.

CARD DETAILS

= contested resource = junkyard = starting card

Mercenary Cards

brawler		A flexible unit with an emphasis upon *fighting*.
hunter		A unit that can *hunt* or *fight*.
group leaders		A multi-purpose unit that can increase the effectiveness of one other unit by modifying an action.

saboteur	A specialized attack unit that renders one opponent's tool (the *shovel*, *spear*, *wolf pack*, *grenade*, etc. is discarded) as ineffective for that round. **This card may be used during the resource gathering phase OR the skirmish phase.** The *saboteur* is used to attack one other player and prevent her from using a single tool (the attacking player must declare the attack). The player plays the card from her hand or from the cards committed to the skirmish. If this skirmish has not started, the *saboteur* card is discarded after use.
scavenger	A weak, but flexible unit.
scout	A flexible unit that can *draw* or *fight*.
sniper team	A specialized attack unit that renders one opponent's tribe member (the mercenary, *refugee*, or *tribe family* card is discarded) ineffective for that round. If multiple players are involved in a skirmish, the sniper can target only one single opponent's card (rather than one card per opponent). The *sniper team* **may be used during the resource gathering phase OR the skirmish phase to *snipe* an opponent's card.** This action is played from the player's hand or from the cards committed to the skirmish. If the skirmish has not started, the *sniper team* card is discarded after use.
thugs	A flexible unit that is highly proficient at *fighting*.

Other Tribe Member Cards

field crew	A flexible, multi-purpose unit.
sled team	Enables a player to draw more cards from the deck or contributes to *fighting*.
tribe family	Represents 3-5 people (very important for winning the game). Can equip tools to *fight* or *hunt*, or can be used during the skirmish to break a tie.
refugee	A weak unit that can *dig* or hunt if equipped with a tool. May also help break a skirmish tie.

Tool Cards

These cards modify actions, they cannot grant actions.

grenades	An extremely potent weapon
multitool	A flexible tool that can modify actions
net	Hunting tool with limited *fight* modifier
pickaxe	Fighting tool with limited *digging* modifier
shovel	Digging tool with limited *fighting* modifier

9

spear / Fighting tool with limited *hunting* modifier

wolf pack Powerful *hunting* and *fighting* tool.

Medicine Cards

Medicine is played from the player's hand as currency to complete a *hire* action.

medkit Provides two *meds* when *hiring* mercenaries

pills Provides one *med* when *hiring* mercenaries

FREQUENTLY ASKED QUESTIONS

How much does it cost to hire thugs?
Thugs can be hired using any combination of *food* and *meds* totaling 6 units.

What are you supposed to do with junk cards?
Sometimes players *dig junk* out of the *junkyard*. Players do not have to hold onto these cards. They may immediately discard them at the bottom of the *junkyard* (they do not get another *dig*).

Can group leaders, or snipers carry tools?
No. A card must have a base ability in order to improve it with a tool.

Do group leaders count as a tool when counting equipment limits?
No. A tribe member card could use a tool AND have a leader increase their action.

Can group leaders enhance a tribe family or refugee?
Yes. A *refugee* combined with a *group leaders* card becomes capable of *dig* or *hunt*. A *tribe family* enhanced by *group leaders* can attack or hunt.

What can a saboteur disarm?
Any card with a tool icon .

What can a sniper team attack?
Sniper teams can only attack people (i.e. card with a person icon in the upper-left).

What happens to the tools held by sniped people?
After a *snipe* action is performed, the sniped player may rearrange the available tools as desired.

> *For example: Player 1's brawler is holding a grenade (+3 fight) and gets sniped by Player 2. Player 1 also has a scavenger in play holding a spear (+2 fight). The scavenger drops the spear in favor of the grenade.*

How is a skirmish resolved when multiple saboteurs and/or sniper teams are in play?
Skirmishes are resolved by starting with the Initiator and having her declare any actions performed by her units (including saboteurs and sniper teams). Play then passes to the next player to declare actions for her units. If a player with a earlier turn order snipes another sniper or snipes a *saboteur* of a later turn order, then those units will not have an opportunity to exercise their special abilities. Sniping a *sniper team* or *saboteur* of a earlier turn order will only impact the number of people for the purposes of breaking a tie.

10

For example: The initiator used a saboteur to disarm Player 3's grenade Player 2 has no special actions. Player 3 uses a sniper team to snipe the initiator's saboteur. Doing so reduces the number of people that the initiator has, but does not prevent the disarm action from occurring. Player 4 then uses her sniper team to snipe Player 5's sniper team. Player 5 will not be able to use her sniper team this round.

Can wolf packs operate independently?
The *wolf pack* cannot since it is a modifier card (notice the *+fight* / *+hunt*). Also note the tool icon in the upper-left corner.

SAMPLE ROUND WALKTHROUGH

Player 1 draws: *brawler* (*fight* 2 / *dig* 1), *shovel* (*fight* +1 / *dig* +2), 2 *refugees* (*dig* 0 / *hunt* 0), *pills* (*meds* 1)

Player 1 performs a *dig* action, using a *refugee* and a *shovel* for a *dig* of 2. Two cards are drawn from the *junkyard* – a *net* (*fight* 1 / *hunt* 2) and *junk* (nothing). The player takes the *net* card, places it in her discard pile, and places the other card on the bottom of the *junkyard*. The player clears the playing surface, taking the *refugee* and *shovel* cards and placing them in her discard pile. The player then announces that three cards are being reserved for the skirmish (*brawler*, *refugee*, and *pills*) and places these cards face-down on the table.

Player 2 draws: *scavenger* (*dig* 1 / *draw* 1 / *fight* 1/ *hunt* 1), *spear* (*fight* +2 / *hunt* +1), 2 *refugees* (*dig* 0 / *hunt* 0), *tribe family* (*fight* 0 / *hunt* 0)

Player 2 performs a *dig* action, using a *scavenger* for a *dig* of 1. The player also performs a *hunt* action, using a *refugee* and a *spear* for a *hunt* of 1. The player draws 1 card from the *junkyard* (*dig* of 1) and places that card face-down in her discard pile. The player then uses the 1 *food* generated by the *hunt* action to *hire* a *scavenger* (cost is 1 *food*) and places this card in her discard pile. This leaves a *tribe family* and *refugee* for the skirmish phase. These cards cannot fight (no attack can be made unless the *fight* total is at least 1). The player does not communicate this however. Instead, the player bluffs and declares that two cards are going to the skirmish.

Player 3 draws: 2 *scavengers* (*dig* 1 / *draw* 1 / *fight* 1 / *hunt* 1), *spear* (*fight* +2 / *hunt* +1), *shovel* (*fight* +1 / *dig* +2), *medkit* (*meds* 2)

Player 3 performs a *hunt* action, using a *scavenger* and a *spear* for a *hunt* of 2. This generates two *food* to purchase a mercenary card. The player also plays a *medkit* card, representing two *meds*. The player then initiates a *hire* action, with two *food* and two *meds* to spend on a single purchase. In this case, the player chooses to purchase a *scout* (*fight* 2 / *draw* 2). The player places the *scout* card and all cards played on her discard pile. The player then announces that two cards are being reserved for the skirmish (a *scavenger* and a *shovel*) and places these cards face-down on the table.

The skirmish phase begins and each player reveals their cards and combines their *Fight* values. Player 1 has a *Fight* of 2 (*Brawler* contributes 2 and *Refugee* does not have a fight score). Player 2 has a Fight of 0 (*Refugee* and *Tribe Family* have a fight score of zero). Player 3 has a *Fight* of 2 (provided by the *Scavenger* wielding a *Shovel*). Players 1 and 3 are tied with a *Fight* of 2. The tie is broken based upon the number of people involved in the skirmish. This tips the scale in favor of Player 1 who brought 2 people to the fight (*Brawler* and *Refugee*).

11

ARCTIC SCAVENGERS - HQ

Basic Premise

The first expansion for Arctic Scavengers introduces the notion of a base camp or headquarters for each tribe. This base camp consists of a Tribal Leader (complete with special abilities) and the potential to construct buildings that can be used strategically during game play. Additionally, the game introduces alternative victory paths, new mercenaries, new tools, and the addition of the *"engineering schematics"* deck.

Game Play Overview

This expansion does not dramatically alter the core game mechanics or objectives. All cards and mechanics from the original game are involved, the *contested resources* still represent the game timer, and the winner is still determined based upon amassing the largest tribe.

Modules

The expansion is organized into modules. Players may want to play the expansion progressively in modules in order to ease into the various new elements the expansion introduces.

Module #1 - Medics, Tools, and Gangs

1. Set up the game in the standard fashion.
2. Add the stack of *medic* cards alongside the other mercenary cards.
3. Shuffle the 8 new *junkyard* cards into the *junkyard* deck. For now, ignore the special ability of the *toolkit*.
4. Locate the *gearheads* and *pharmers*. Place them on the table next to the mercenary stacks.

Medics (new Mercenary)

Medics are quite versatile. They have a *draw* of 1 (like a scavenger), can be played from the hand during a *hire* action to represent 1 *med* (like *pills*), and they can also be played from the hand to save another tribe member card from a sniper attack (they could even be used to save another player's tribe member from sniper attack).

Save (new Action)

Save – protect a tribe member or leader from a *snipe* attack.

Rifle (new Tool)

A versatile tool that is good at *hunting* and *fighting*. This is the first tool from the *junkyard* to grant a +2 in two categories.

Toolkit (new Tool)

A *toolkit* can be used to enhance *digging* (either *junkyard* or *engineering schematics*) or can be handed to any tribe member that is used to speed up the *building* process on a building. The *toolkit* enables additional cards to be removed from one building that is under construction.

12

Gangs (new Game Concept)

Three gangs are introduced in this expansion. Each gang is watching the struggle for survival amongst these various tribes from a safe distance. Once all of the *contested resources* have been gathered, then these tribes will choose to form an alliance with whichever tribe best meets their needs. Each gang is motivated by different things (tools, medicine, buildings). Winning a gang could easily make the difference in a closely contested game.

The Gearheads "Most tools" is determined by adding up all the cards with a tool icon from each player's deck. Ties are broken by counting the total number of tools that are *contested resources*.

The Pharmers "Most Meds" is determined by adding the total '*meds*' value of all *pills* and *medkits* in a player's deck. Ties are broken by counting the total number of medics.

The Masons "Most buildings" is determined by adding the total number of completed and enabled buildings each player has in their headquarters. Ties are broken by adding up the total number of *engineers*.

Module #2 – Engineers and Buildings

1. Set up the game just as you did for Module #1.
2. Add the stack of *engineer* cards alongside the other mercenary cards.
3. Shuffle the 12 building cards together (indicated by the building icon in the upper-left corner) and place then face-down next to the *junkyard* deck (they form a 3rd pile of cards next to the *junkyard* and *contested resources* piles)
4. Add the 3rd gang card (*masons*) next to the other 2 gangs.
5. The 'special' ability from the *toolkit* can now be used (since it works with buildings)

Engineers (new Mercenary)

Engineers can *dig* in the *junkyard* for resources, but the primary use of their *dig* action is to *dig* in the *engineering schematics* pile (in fact, they are the only mercenary capable of such *digging*). Other cards with a *dig* ability can be combined with the *engineer* to improve *engineering schematics digging*. Much like the *junkyard*, only 1 card may be selected as a result of a *dig* in the *engineering schematics* pile. The player places this card face-up in her play area to represent a construction project for a building that is currently underway. A number of cards equal to the building's *'build time'* are drawn from the top of the player's deck and placed face down on top of the building which is under construction. This represents the *build* timer. Finally, the *engineer* card is placed in the player's discard pile.
To learn more about buildings, see 'Buildings' below.

Buildings (new Game Concept)

Players can use *engineers* (see above) to construct buildings at the player's headquarters. These buildings take time to *build*, but once built offer the player strategic advantages in certain areas.

Constructing Buildings

The process of constructing a building is as follows:

1) Play an *engineer* card
2) Use the *engineer's dig* (combined with any modifiers) to *draw* cards from the *engineering schematics* pile equal to the *dig* value
3) Select a maximum of 1 card to *build* and return the others to the bottom of the *engineering schematics* pile
4) Place the new building card face-up in your play area
5) *Draw* cards from the player's deck equal to the building's *build time* and place those face-down on top of the new building

13

6) Discard the *engineer* card
7) At the start of the player's next round, remove 1 card from each building that is under construction
8) Starting with the round following the initial building placement, additional tribe members (and *toolkits*) may be played from the player's hand to accelerate construction on one building. Each tribe member played can remove 1 card from a building that is under construction. If wielding a *toolkit*, then 2 additional cards may be removed. Tribe members that work on construction projects may not take any other actions that round and are immediately discarded.
9) Once a building has zero cards stacked on it, it is complete and immediately goes into effect.

Building Types

Armory	Enables up to two tools to be *stored* under this card to be *retrieved* at any time (except during the skirmish). One or two cards may be placed in this building at a time.
Bunker	Enables up to three tribe member cards to be *stored* under this card to be *retrieved* at any time (except during the skirmish). Up to three cards may be placed in this building at a time.
Hydroponic Gardens	Generates 1 *food* each round to be used as part of a *hire* action. *Food* does not accumulate from round to round.
Pharmacy	Enables up to two *medicine* cards (*pills* or *medkit*, but not tribe members such as *medic*) to be *stored* under this card to be *retrieved* at any time (except during the skirmish). One or two cards may be placed in this building at a time.

Using buildings

Typically, buildings may only be used during a player's turn. Buildings may never be used during the skirmish. During another player's turn, the only building which a player may access is the bunker.

> *For example: a sniper team could be retrieved from a player's bunker and used during another player's turn in order to interrupt that player's resource gathering action.*

During a player's turn, that player may place cards into the bunker, armory, or pharmacy. Likewise, cards may be retrieved from these buildings. Once a player has committed cards to the skirmish, no cards may be placed into a building by that player nor can cards that come out of a building be added to the skirmish set. Once a player has committed cards to the skirmish, no change may take place in those cards except for the case of using a *saboteur* or *sniper team* from the committed cards to interrupt another player's resource gathering.

The hydroponic garden comes into effect each round and generates *food* in real time in much the same way that a hunt of 1 generates *food* (although it does not involve *hunting* and thus does not benefit or suffer from *hunting*-related modifiers).

Store (new Action)

Take one or more cards of a given type from your hand and place them, face-down, under a completed building. Do not exceed building capacity or violate type.

14

Retrieve (new Action)
Take one or more cards from an active building and place them in your hand. This action can occur any time during a round so long as the skirmish has not been initiated.

Module #3 – Tribal Leaders

1. Set up the game just as you did for Module #2.
2. Shuffle the tribal leader cards together, and deal two, face-down, to each player. From those two, each player keeps only 1 to represent her tribal leader (players may look at them to decide). The card that is kept is placed face-up in the player's play area for everyone to see. The other card is set-aside as it will not be used this game.

Tribal Leaders (new Game Concept)
The introduction of tribal leaders provides players with a unique capability that no other player in the game will have. A leader might grant special abilities to a tribe's refugees, provide unique protection against certain kinds of attacks, or grant the player other advantages. At the start of the game, each player chooses one tribal leader (out of two provided to him/her) to lead the tribe to victory. The selection of a tribal leader may have a profound impact upon strategy and game play.

Butcher / Cannibal / Fanatic
Cards that are removed from play cannot return to the game and do not count toward final scoring.

Butcher / Cannibal / Sergeant at Arms
These leaders can use their special abilities without the need for playing a *refugee*.

Excavator / Fanatic / Gangster / Mentor / Organizer / Peacemaker / Ranger
Requires the use of a *refugee* to utilize the leader's special ability.

Butcher / Cannibal / Fanatic / Mentor / Organizer / Peacemaker
These leaders grant abilities that can only be utilized once per round.

Excavator / Gangster / Ranger / Sergeant of Arms
These leaders grant abilities that are always in effect and could impact multiple cards in a given round.

Module #4 – Dirty Deeds

1. Set up the game just as you did for Module #3.
2. Add two new rules.

NEW SABOTEUR ABILITY:
A *saboteur* may attack a building that is completed and disable it, rendering it unusable (flip the building's card over to reveal its back, indicating that the building is disabled). If a building is disabled, its abilities cannot be used and cards cannot be added or removed from the building. To repair the building, a person card must be played from the player's hand to work on the building and then immediately discarded.

NEW SNIPER TEAM ABILITY:
A *sniper team* may attack a tribe leader, wounding him (turn the leader's card over to indicate he or she is injured). This removes the leader's advantage until a *med* is applied against the leader to heal him.

15

Credits and Acknowledgements

This game could not be been made possible if it were not for the time, energy, and commitment of friends, family, consultants, and the fantastic array of playtesters that were willing to play the game over and over and provide feedback and fresh insight.

Special thanks go out to the following people that helped make this game possible:

Chris White
Dylan Gould
Elizabeth Gabhart
Ingrid Norsic
Jon Rosen
Lance Bailey
Mike Cooper
Quinn Munnerlyn
Robert Virata
Shea Parkes
Todd and Crystal Dunlap
David Turner
Everyone at the South Arlington Gamers Guild
The regulars at my Friday game nights

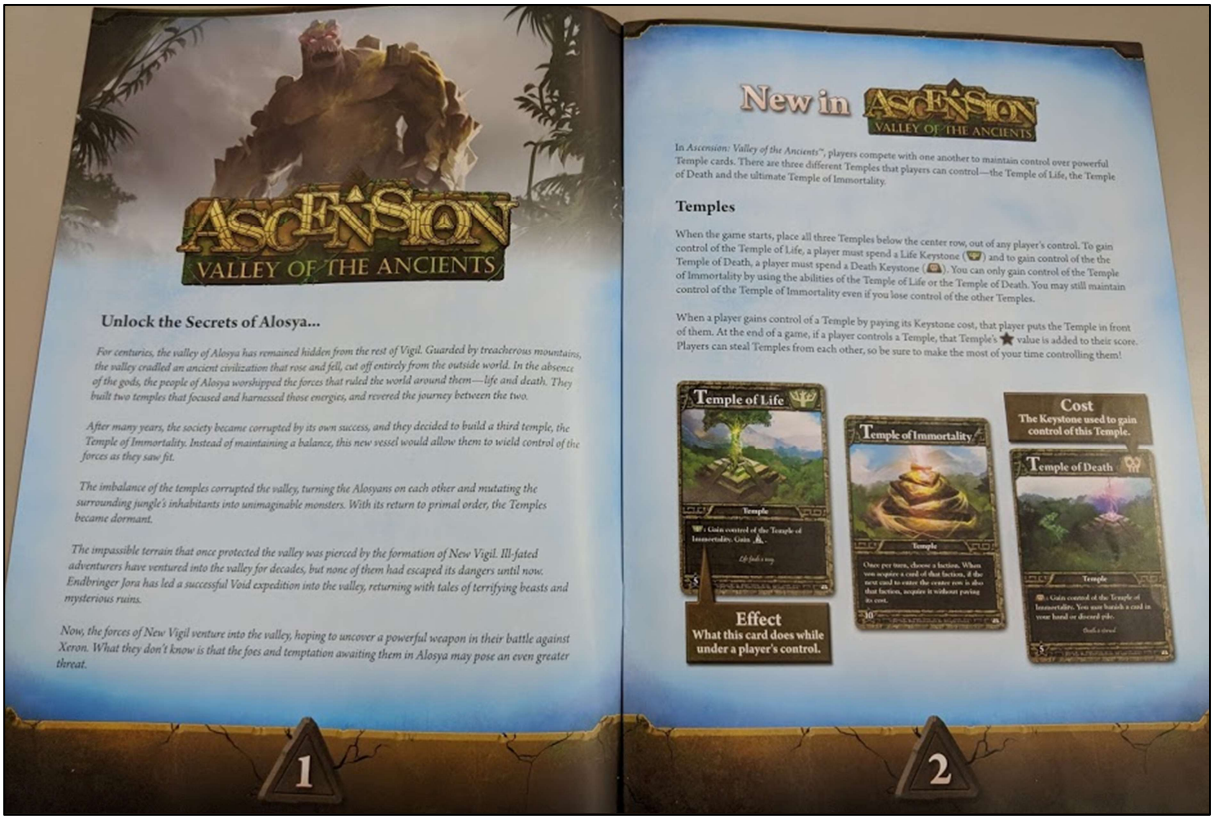
Game Designer – Robert Kyle Gabhart
Art and Graphic Design – Martin Hoffmann

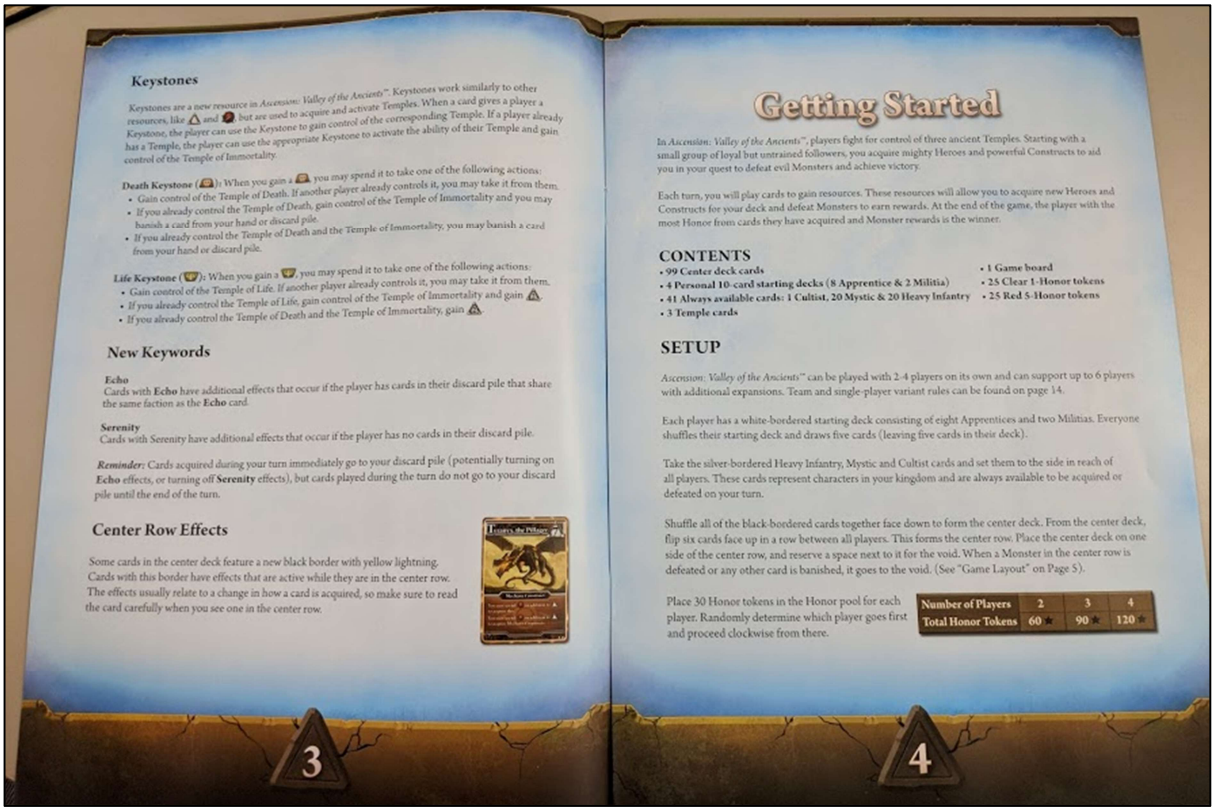
For resources, updated rules, and information regarding expansions, visit our website:
www.riograndegames.com

Rio Grande Games
PO Box 1033, Placitas, NM 87043, USA

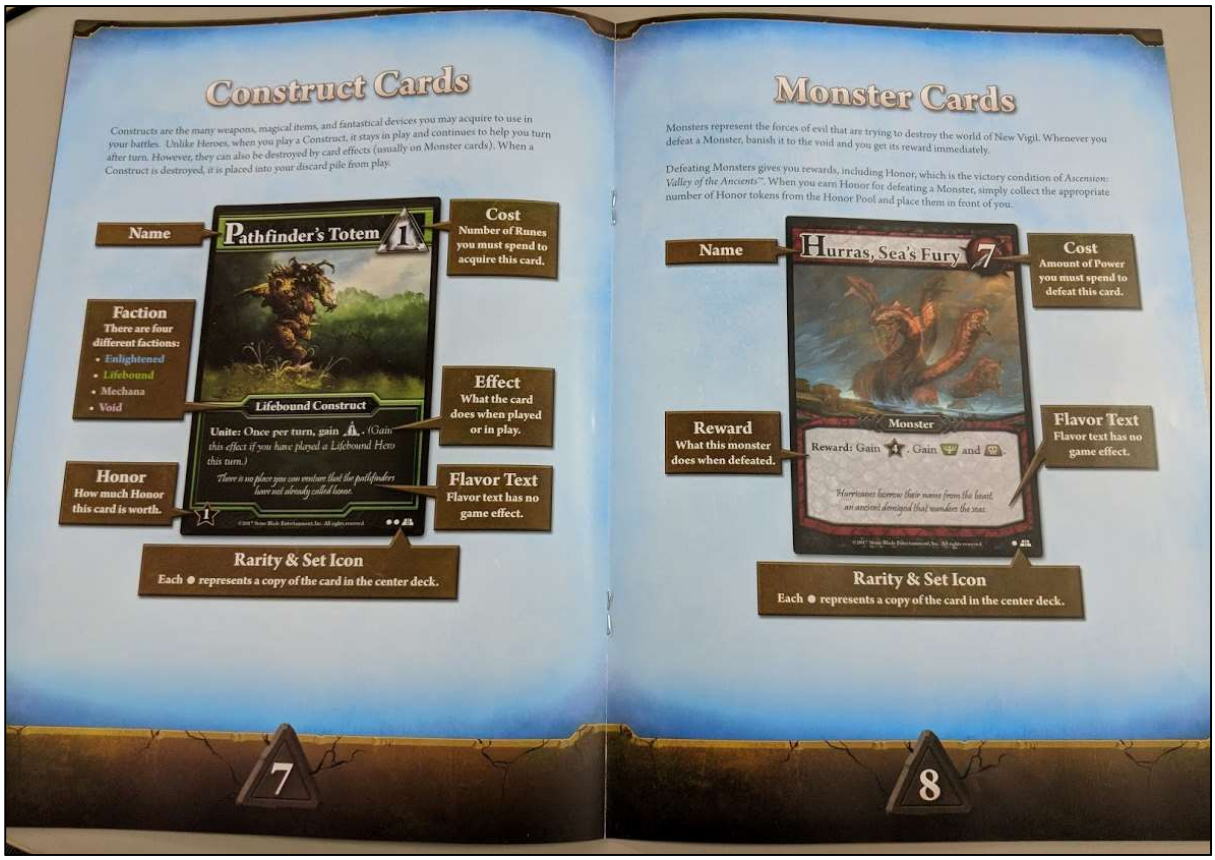
16

Appendix D: Ascension: Valley of the Ancients rulebook [19]






Game Layout
Always Available Cards
(Mystic, Heavy Infantry & Cultist)
Honor Pool
Center Row
Starting Hand
(5 cards)
Starting Deck
(remaining 5 cards)
Personal Deck
(10 cards: 8 Apprentice & 2 Militia)
Temples
Center Deck
Void
5
Hero Cards
Heroes are the allies you acquire to aid you in your quest to protect the realm of New Vigil. You can play any number of Hero cards from your hand each turn. When you play a Hero from your hand, gain the effect listed on the card. It stays in front of you until the end of your turn, and then it goes to your discard pile.
Name
Resourceful Trader
2
Cost
Number of Runes you must spend to acquire this card.
Faction
There are four different factions:
• Enlightened
• Lifebound
• Mechana
• Void
Mechana Hero
Draw a card.
The next time you acquire a Construct this turn, gain .
"You need a keystone? I can get you a keystone."
Effect
What the card does when played or in play.
Honor
How much Honor this card is worth.
1
Flavor Text
Flavor text has no game effect.
Rarity & Set Icon
Each ● represents a copy of the card in the center deck.
6


Construct Cards
Constructs are the many weapons, magical items, and fantastical devices you may acquire to use in your battles. Unlike Heroes, when you play a Construct, it stays in play and continues to help you turn after turn. However, they can also be destroyed by card effects (usually on Monster cards). When a Construct is destroyed, it is placed into your discard pile from play.
Name
Pathfinder's Totem
1
Cost
Number of Runes you must spend to acquire this card.
Faction
There are four different factions:
• Enlightened
• Lifebound
• Mechana
• Void
Lifebound Construct
Unite: Once per turn, gain . (Gain this effect if you have played a Lifebound Hero this turn.)
There is no place you can venture that the pathfinders have not already called home.
Effect
What the card does when played or in play.
Honor
How much Honor this card is worth.
1
Flavor Text
Flavor text has no game effect.
Rarity & Set Icon
Each ● represents a copy of the card in the center deck.
7
Monster Cards
Monsters represent the forces of evil that are trying to destroy the world of New Vigil. Whenever you defeat a Monster, banish it to the void and you get its reward immediately.
Defeating Monsters gives you rewards, including Honor, which is the victory condition of Ascension: Valley of the Ancients™. When you earn Honor for defeating a Monster, simply collect the appropriate number of Honor tokens from the Honor Pool and place them in front of you.
Name
Hurras, Sea's Fury
7
Cost
Amount of Power you must spend to defeat this card.
Monster
Reward: Gain . Gain and .
"Hurricanes borrow their names from the least, an ancient demigod that wanders the seas."
Reward
What this monster does when defeated.
Flavor Text
Flavor text has no game effect.
Rarity & Set Icon
Each ● represents a copy of the card in the center deck.
8

Gameplay

General Rules

At any time, if your personal deck is empty and you need to draw a card or reveal a card from the top of your deck, reshuffle your discard pile to replenish your deck. If the center deck runs out, reshuffle the void to replenish it.

Hero cards played during your turn go into the discard pile at the end of your turn. (In other words, don't reshuffle Hero cards you've played this turn if you need to reshuffle your discard pile in the middle of your turn.)

Cards you acquire during your turn are placed in your discard pile.

Whenever a card leaves the center row, replace it immediately from the center deck. This happens before anything else occurs. Any time a player is making a decision, the center row should be full.

Many cards will have text that contradicts these rules. If card text ever contradicts these rules, follow the card text instead.

Turn Breakdown

1. Play cards from your hand to gain Runes (△), Power (●), Honor (★), or a Keystone (▭ or ▽). Acquire Heroes and Constructs, and put them in your discard pile for future turns. Defeat Monsters for rewards and Honor. Use Keystones to gain control of Temples or activate them.
2. Place played Hero cards and unplayed cards in your hand in your discard pile at the end of your turn.
3. Draw five cards from your personal deck.

To play a Hero card from your hand, simply place the card face up in front of you, announce its effect, and follow the instructions on the card. Any choices required by the card must be made at that time and any conditionals must be met for additional effects, though any △, ●, ▭, or ▽ generated can be spent until the end of your turn. You do not have to pay any costs to play Heroes from your hand.

Example: *Qadim Scout says, "Gain △ and ●." When you play Qadim Scout, you gain the △ and ● immediately, but you do not have to spend them immediately.*

Constructs are played just like Heroes, except Constructs remain in play and can be used every turn. If a Construct has multiple abilities, each one may be used.

Example: *Deranged Dirge says, "Once per turn, gain 1. You may destroy this to banish a card in your hand or discard pile." You may choose to gain 1 at the beginning of your turn, and still use the second ability later in the same turn.*

Acquiring Heroes and Constructs

You need Runes (△) to acquire Heroes and Constructs. Runes come from Heroes played from your hand or from Constructs you have in play. You may acquire any number of cards as long as you have enough Runes. To find out how many Runes you get from a card, look in the text box of the card. If a card provides Runes, there will be a Rune icon with a number.

You may acquire any Hero or Construct in the center row as well as Mystics and Heavy Infantry, which are always available. When you acquire a card, place it in your discard pile and replace the card in the center row with the top card of the center deck.

Defeating Monsters

To defeat a Monster, a player must have enough Power (generated by cards with ● in their text box—e.g., Heavy Infantry). The amount of Power a card generates is found in its game text box represented by the Power icon with a number.

The Defeat Cost of a Monster is listed in the upper-right corner of the card. This cost has a Power icon behind it.

When you defeat a Monster, you gain a Reward. Most rewards include Honor, and how much Honor you gain is equal to the number indicated on the Honor symbol (★). If the Monster has another effect as part of its reward, that effect occurs immediately after the card is replaced in the center row.

After defeating a Monster, place it in the void and replace it in the center row. You may defeat any numb of Monsters as long as you have enough Power.

The Cultist is always available to defeat and is never banished to the void. There is no limit to how ma times you can defeat the Cultist in one turn.

Banishing Cards

Some cards allow you to banish cards in either the center row or your own deck. When a card tells you to banish from the center row, put the card into the void, then place the top card of the center deck into the center row. If a card instructs you to banish multiple cards from the center row, place all of those cards into the void, then put that many cards from the center deck into the center row.

When a card tells you to banish a card in your hand and/or your discard pile, you may put a card from that zone into the void. If a card allows you to banish a card in your hand, you may not banish a card that you've already played that turn, as that card is no longer in your hand. Banishing your weaker cards allows you to draw your more powerful cards more frequently, so make sure to banish your Apprentice and Militia cards whenever you get the opportunity.

Ending Your Turn

After you are done playing cards from your hand, acquiring Heroes and Constructs, and defeating Monsters, place any remaining cards in your hand and any Heroes played during your turn into your discard pile. You lose all unspent △, ●, ▭, and ▽.

Draw five cards to replenish your hand. If you run out of cards in your deck and need to draw more, simply reshuffle your discard pile into your deck. When you end your turn, play passes clockwise around the table.

Game End

The game ends when a certain amount of Honor has been earned, based on the number of players.

Number of Players	2	3	4
Total Honor Tokens	60 ★	90 ★	120 ★

Set this much Honor aside in Honor tokens before starting the game. When the final Honor token is earned, the game ends at the end of the current round (after the last player to start the game takes a turn). Thus, each player will play the same number of turns during the course of the game. Players can still gain Honor even when the set number of Honor tokens is gone. Use additional Honor tokens left over to track any Honor earned by players before the game ends.

Cards in each player's deck are also worth Honor, indicated by the number in the Honor symbol (★) on the bottom left corner of the card. When the game is over, count the Honor from Heroes and Constructs (this includes your hand, deck, discard pile, and played Constructs), Honor tokens acquired during the game, and Temples you control when the game ends. The player with the most total Honor is the winner!

If multiple players have the same number of Honor Points, the last player to start wins (i.e., the starting player loses all ties, the second player loses to third and fourth, etc.).

F.A.Q.

Q: What do I do with cards like Jubilant Monk, Grave Golem, Blind Seer, Excavation Sentry, and Silenced Prophet when there are no cards left in my deck?
A: First shuffle your deck and then continue with the card's effect.

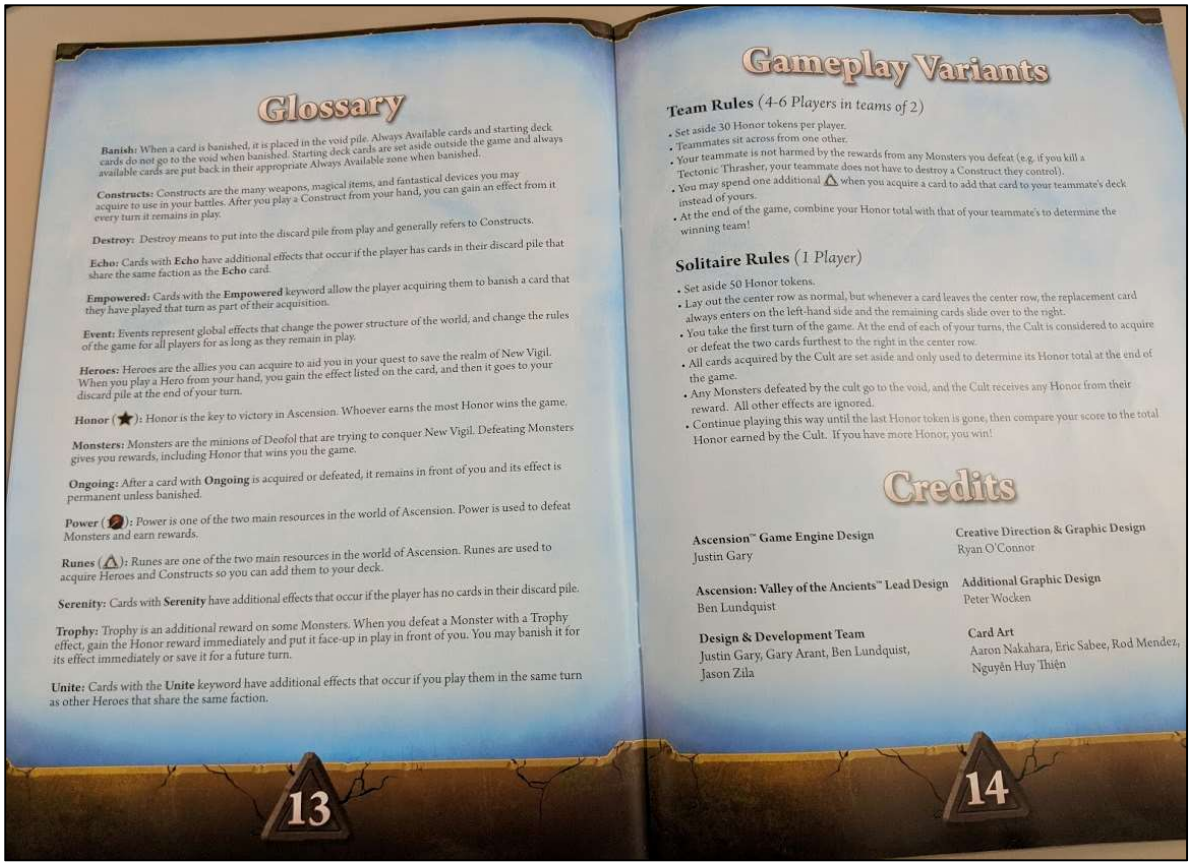
Q: If I defeat Hellfrost Imps and another Hellfrost Imps enters the center row in its place, can I defeat it using the first Hellfrost Imps' reward?
A: Yes. All cards in the center row must be replaced before the effects from acquiring and defeating cards in the center row are gained.

Q: Can I copy the effect of Isla, Serene Prodigy with a card like Twofold Askara even though it is no longer in play?
A: Yes. Isla, Serene Prodigy is still an effect that can be copied, as it is a Hero you have played that turn.

Q: Do the Heroes I play during my turn go to my discard pile when I play them?
A: No. Heroes you play during your turn do not go to your discard pile until the end of the turn.

Q: What happens when I defeat Grave Golem and my opponent reveals Isla, Serene Prodigy. Do I get to shuffle his Isla, Serene Prodigy into my deck?
A: No. Isla, Serene Prodigy will stay in the opponent's discard pile, but you still gain the rest of the effect.

Q: If I play a card with Echo and don't have a Void card in my discard pile, but did previously that turn, do I get the Echo effect?
A: No. Echo only checks at the moment you play the card.



Glossary

Banish: When a card is banished, it is placed in the void pile. Always Available cards and starting deck cards do not go to the void when banished. Starting deck cards are set aside outside the game and always available cards are put back in their appropriate Always Available zone when banished.

Constructs: Constructs are the many weapons, magical items, and fantastical devices you may acquire to use in your battles. After you play a Construct from your hand, you can gain an effect from it every turn it remains in play.

Destroy: Destroy means to put into the discard pile from play and generally refers to Constructs.

Echo: Cards with **Echo** have additional effects that occur if the player has cards in their discard pile that share the same faction as the **Echo** card.

Empowered: Cards with the **Empowered** keyword allow the player acquiring them to banish a card that they have played that turn as part of their acquisition.

Event: Events represent global effects that change the power structure of the world, and change the rules of the game for all players for as long as they remain in play.

Heroes: Heroes are the allies you can acquire to aid you in your quest to save the realm of New Vigil. When you play a Hero from your hand, you gain the effect listed on the card, and then it goes to your discard pile at the end of your turn.

Honor (★): Honor is the key to victory in Ascension. Whoever earns the most Honor wins the game.

Monsters: Monsters are the minions of Deofol that are trying to conquer New Vigil. Defeating Monsters gives you rewards, including Honor that wins you the game.

Ongoing: After a card with **Ongoing** is acquired or defeated, it remains in front of you and its effect is permanent unless banished.

Power (●): Power is one of the two main resources in the world of Ascension. Power is used to defeat Monsters and earn rewards.

Runes (△): Runes are one of the two main resources in the world of Ascension. Runes are used to acquire Heroes and Constructs so you can add them to your deck.

Serenity: Cards with **Serenity** have additional effects that occur if the player has no cards in their discard pile.

Trophy: Trophy is an additional reward on some Monsters. When you defeat a Monster with a Trophy effect, gain the Honor reward immediately and put it face-up in play in front of you. You may banish it for its effect immediately or save it for a future turn.

Unite: Cards with the **Unite** keyword have additional effects that occur if you play them in the same turn as other Heroes that share the same faction.

13

Gameplay Variants

Team Rules *(4-6 Players in teams of 2)*

- Set aside 30 Honor tokens per player.
- Teammates sit across from one other.
- Your teammate is not harmed by the rewards from any Monsters you defeat (e.g. if you kill a Tectonic Thrasher, your teammate does not have to destroy a Construct they control).
- You may spend one additional △ when you acquire a card to add that card to your teammate's deck instead of yours.
- At the end of the game, combine your Honor total with that of your teammate's to determine the winning team!

Solitaire Rules *(1 Player)*

- Set aside 50 Honor tokens.
- Lay out the center row as normal, but whenever a card leaves the center row, the replacement card always enters on the left-hand side and the remaining cards slide over to the right.
- You take the first turn of the game. At the end of each of your turns, the Cult is considered to acquire or defeat the two cards furthest to the right in the center row.
- All cards acquired by the Cult are set aside and only used to determine its Honor total at the end of the game.
- Any Monsters defeated by the cult go to the void, and the Cult receives any Honor from their reward. All other effects are ignored.
- Continue playing this way until the last Honor token is gone, then compare your score to the total Honor earned by the Cult. If you have more Honor, you win!

Credits

Ascension™ Game Engine Design
Justin Gary

Ascension: Valley of the Ancients™ Lead Design
Ben Lundquist

Design & Development Team
Justin Gary, Gary Arant, Ben Lundquist, Jason Zila

Creative Direction & Graphic Design
Ryan O'Connor

Additional Graphic Design
Peter Wocken

Card Art
Aaron Nakahara, Eric Sabee, Rod Mendez, Nguyên Huy Thiện

14

Appendix E: Dominion cards implemented

E.1 Base Cards

Card	*Cost (Treasure)*	*Description*
Copper	0	+1 Treasure
Silver	3	+2 Treasure
Gold	6	+3 Treasure
Estate	2	+1 Victory Point
Duchy	5	+3 Victory Points
Province	8	+6 Victory Points
Curse	0	-1 Victory Point

Table 6: Dominion base cards

E.2 Kingdom Cards

Note: cards with an (*) are from the first edition of *Dominion* and have been removed in the second edition of the game.

Card	*Cost (Treasure)*	*Effects*	*Description*
Cellar	2	+1 Action	Discard any number of cards, +1 Card per card discarded
Chapel	2		Trash up to 4 cards from your hand
Moat (Reaction)	2	+2 Cards	When another player plays an Attack card, you may reveal this from your hand. If you do, you are unaffected by that Attack
Workshop	3		Gain a card costing up to 4 Treasure
Chancellor*	3	+2 Treasure	You may immediately put your deck in your discard pile
Village	3	+1 Card +2 Actions	
Woodcutter*	3	+1 Buy +2 Treasure	
Feast*	4		Thrash this card. Gain a card costing up to 5 Treasure

Militia (Attack)	4	+2 Treasure	Each other player discards down to 3 cards in their hand
Moneylender	4		Trash a copper from your hand. If you do +3 Treasure
Remodel	4		Trash a card from your hand. Gain a card costing up to 2 Treasure more than the trashed card
Bureaucrat (Attack)	4		Gain a Silver card, put it on top of your deck. Each other player reveals a Victory card from their hand and puts it on their deck (or reveals a hand with no Victory cards)
Smithy	4	+3 Cards	
Gardens (Victory)	4		Worth 1 Victory Point for every 10 cards in your deck (rounded down)
Council Room	5	+4 Cards +1 Buy	Each other player draws a card
Festival	5	+2 Actions +1 Buy +2 Treasure	
Laboratory	5	+2 Cards +1 Action	
Market	5	+1 Card +1 Action +1 Buy +1 Treasure	
Mine	5		Trash a Treasure card from your hand. Gain a Treasure card costing up to 3 more; put it in your hand.
Witch (Attack)	5	+2 Cards	Each other player gains a Curse card

Table 7: Dominion kingdom cards

Appendix F: Ascension Cards Implemented

F.1 Base Cards

Card	*Cost*	*Effects*	*Description*
Cultist	2 Power		Reward: 1 Honour
Temple of Life	1 Life Keystone	+2 Runes	Life Keystone: Gain the Temple of Immortality
Temple of Death	1 Death Keystone		Death Keystone: Gain the Temple of Immortality. Banish a card from your hand
Temple of Immortality	N.A.		
Apprentice	N.A.		
Militia	N.A		
Mystic	3 Runes	+2 Runes	
Heavy Infantry	2 Runes	+2 Power	

Table 8: Ascension: Valley of the Ancients base cards

F.2 Monster Cards

Card	*Cost (Power)*	*Reward*
Hellfrost Imps	3	2 Honour. You may defeat a monster in the centre row named Hellfrost Imps without paying the cost
Kan'zir, the Ravager	6	5 Honour. Each opponent loses control of all Temples and Destroys all Constructs they control
Starved Abomination	4	3 Honour, Life Keystone
Hurras, Sea's Fury	7	4 Honour, Life Keystone, Death Keystone
Crypt Lurker	4	3 Honour, Death Keystone
Mutated Scavenger	6	4 Honour, 5 Runes

Iku, Valley Tyrant	10	This card costs 2 Power less for each temple you control. 8 Honour
Cavern Horror	2	1 Honour. You may shuffle your discard into your deck

Table 9: Ascension: Valley of the Ancients monster cards

F.3 Hero Cards

Card	*Cost (Runes)*	*Faction*	*Effects*	*Description*
Soulsnare Hunter	4	Void	+2 Power	The next time you defeat a monster this turn: Life Keystone
Shadowridge Scout	2	Void	+2 Power	Echo: Death Keystone
Spiteful Gladiator	3	Void	+2 Treasure	You may immediately put your deck in your discard pile
Burial Guardian	4	Lifebound	+2 Runes	The next time you acquire a lifebound card this turn: Death Keystone
Ancient Stag	2	Lifebound	+3 Honour	Unite: Life Keystone
Alosyan Guide	3	Lifebound	+3 Honour	When you acquire this card, if you have played a Lifebound Hero this turn, put this directly into your hand
Blind Seer	2	Enlightened	+1 Card	Serenity: Life Keystone
Isla, Serene Prodigy	8	Enlightened		Acquire a hero or construct from the centre row for free. Shuffle the card and discard into your deck

Table 10: Ascension: Valley of the Ancients hero cards

F.4 Construct Cards

Card	*Cost (Runes)*	*Faction*	*Effects*	*Description*
Beacon of the Lost	1	Void	+1 Rune	Echo: Once per turn +1 Power
Pathfinder's Totem	4	Lifebound	+1 Rune	Unite: Once per turn +1 Rune
Sacred Pot	1	Enlightened		Serenity: Once per turn draw a card
Templar Outpost	6	Enlightened		Once per turn defeat a monster costing 4 power or less for free

Table 11: Ascension: Valley of the Ancients construct cards

7. References

[1] D. X. Vaccarino, “The Secret History of Dominion,” 24 June 2013. [Online]. Available: https://www.boardgamegeek.com/thread/996671/secret-history-dominion. [Accessed 21 March 2018].

[2] W. E. Martin, “Game Preview/Review: Dominion,” 17 October 2008. [Online]. Available: https://web.archive.org/web/20081020023147/http://www.boardgamenews.com/index.php/boardgamenews/comments/game_preview_review_dominion/. [Accessed 21 October 2018].

[3] Rio Grande Games, “Arctic Scavengers Rulebook,” 2012. [Online]. Available: http://riograndegames.com/uploads/Game/Game_384_gameRules.pdf. [Accessed 21 March 2018].

[4] StoneBlade Entertainment, “Ascension How to Play,” 2018. [Online]. Available: http://ascensiongame.com/game/how-to-play/. [Accessed 21 March 2018].

[5] D. X. Vaccarion, “Dominion Rulebook,” 2008. [Online]. Available: http://riograndegames.com/uploads/Game/Game_278_gameRules.pdf. [Accessed 21 March 2018].

[6] White Wizard Games, “Star Realms Learn to Play,” 2017. [Online]. Available: https://www.starrealms.com/learn-to-play/. [Accessed 21 March 2018].

[7] Shuffle iT, “Dominion Online,” 2018. [Online]. Available: https://dominion.games/. [Accessed 21 March 2018].

[8] White Wizard Games, “Star Realms Digitial Deckbuilder,” 2018. [Online]. Available: https://www.starrealms.com/digital-game/. [Accessed 21 March 2018].

[9] Asmodee Digital, “Ascension,” 8 August 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.playdekgames.android.Ascension. [Accessed 20 October 2018].

[10] Node.js Foundation, “Node.js,” March 2018. [Online]. Available: https://nodejs.org/en/. [Accessed 18 March 2018].

[11] Express, “Express,” 2017. [Online]. Available: http://expressjs.com/. [Accessed 27 October 2018].

[12] Processing Foundation, “P5.js,” March 2018. [Online]. Available: https://p5js.org/. [Accessed 18 March 2018].

[13] socket.io, “Socket.io Documentation,” 2018. [Online]. Available: https://socket.io/docs/. [Accessed 24 October 2018].

[14] GitHub, “GitHub,” 2018. [Online]. Available: https://github.com. [Accessed 27 October 2018].

[15] J. Hanson, “What is HTTP Long Polling?,” 1 December 2014. [Online]. Available: https://www.pubnub.com/blog/2014-12-01-http-long-polling/. [Accessed 31 October 2018].

[16] WebRTC, “What is WebRTC?,” 2018. [Online]. Available: https://webrtc.org/. [Accessed 31 October 2018].

[17] Pusher, “What are WebSockets?,” 2018. [Online]. Available: https://pusher.com/websockets. [Accessed 31 October 2018].

[18] npm, “What is npm?,” 2018. [Online]. Available: https://docs.npmjs.com/getting-started/what-is-npm. [Accessed 1 November 2018].

[19] Stone Blade Entertainment, Ascension: Valley of the Ancients Game Rules, Stone Blade Entertainment, 2017.

[20] Open Source Project, “Jest - Delightful JavaScript Testing,” 18 March 2018. [Online]. Available: https://github.com/facebook/jest. [Accessed 18 March 2018].

[21] D. Nakamura, “So what exactly is a deck-building game anyway?,” 29 October 2014. [Online]. Available: https://www.destructoid.com/so-what-exactly-is-a-deck-building-game-anyway--283188.phtml. [Accessed 21 March 2018].

[22] Alderac, “Thunderstone Rulebook,” 2009. [Online]. Available: https://www.alderac.com/thunderstonerules.pdf. [Accessed 21 March 2018].

[23] Catalyst Game Labs, “Shadowrun:Crossfire Rulebook,” 2013. [Online]. Available: http://www.shadowruntabletop.com/wp-content/uploads/CAT27700_Crossfire_Full%20Rules.pdf. [Accessed 21 March 2018].