Kevin Li (kl624), Mike Kuan (mmk248), John Oderjar (jao57), Matthew Klinger (mzk7)

November 12, 2015

CS 3110

Project 6 Design

**System Description:**

The core vision of the system is to implement a card game with mechanics similar to those of Hearthstone and Magic: The Gathering.

* Our game will feature an AI to play against.
* Our game will allow for a player to either play with a predefined deck, or through a draft mode in which a player creates a deck by choosing one of 3 cards until his/her deck is full.
* Our game will include a text based player interface that provides information on the current board and player status.
* Our game will be packaged with a small rule set describing how the mechanics work so that beginners can easily learn.

The game design will be similar to that of the popular online trading card game Hearthstone. A generic set of cards will be created for the game, featuring two types of cards: creatures, which various creature types as well, and spells. Creature cards will have three mandatory stats and a variety of optional other effects. The three mandatory stats are a mana cost, attack, and health. Other optional effects will be determined based on what can be implemented and will be listed in a card description. Spells would have a mana cost and an effect, which will also be listed in the card description. A deck of cards would consist of 30 cards. A deck can be created or altered through a CSV file which will be passed into the game as input.

A game state would consist of a battlefield, life total, hand of cards, and remaining deck for each player. The game is turn based, so only one player will be performing actions on his turn. On a player’s turn, he will first gain one mana crystal, which will be refilled along with his preexisting mana crystals. These mana crystals can be utilized to cast cards from your hand or use other effects, and they will be refilled at the start of your next turn. Then, the player will draw a card, taking a card from the remaining deck and placing it into his hand. Then, there will be a phase during which the player can input a set of commands, including playing cards from his hand, attacking with creatures on his battlefield, and utilizing his Hero Power, which will be a unique ability that has a particular cost and effect and that can be utilized once a turn. Hero Powers can be selected when a deck is constructed. The player’s turn would end with his input of the “end turn” command. Then the game would continue by beginning the other player’s turn in the same fashion.

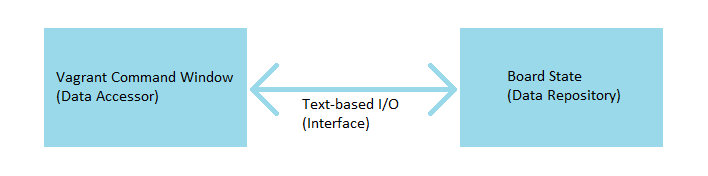
Game state would be printed out to the player excluding specific information about the other player’s hand. The players would respond using text based input commands. These commands would be parsed to determine the action and then applied to the game state. Other features of the game may be implemented if time permits.

**Architecture:**

The architecture of our game is similar to that of a shared data architecture. In a shared data architecture, there are two types of components and one kind of connector. One component of this architecture is the **data repository**, which is the component that stores data. This component is the equivalent of the **board state** in our game model, which would contain all the relevant information about the state of the current ongoing game.

The other type of component in the shared data architecture is the **data accessor**, which is the component that does computation with the data or determines what should be done with the given data. This component is represented in our game model as the user input to the **vagrant command window**. The user will determine what he or she would like to do given the current board state and can communicate this through the vagrant command window input. In order to piece these two components together, a connector is required.

The connector of a shared data architecture is an **interface**, which give read and write access to the data repository. In the case of our game, this connector can be represented by the **text-based input/output**. The user can input text commands into the vagrant command window, and these commands will be parsed and interpreted in order to manipulate the game state, which is the data repository. Then the data repository will print out its current state in a text format readable by the user in the vagrant command window. The components and connectors diagram is shown in the following image:



**System Design:**

**board.ml** – This module will be the one to hold all the information about the game states as a game progresses. The information is updated as inputs are taken and the game state changes. Information that is recorded includes: game mode (player vs player or player vs AI), each player’s hero, each player’s remaining HP, each player’s hand, the creatures on each player’s battlefield, each player’s remaining deck, whether a player’s Hero Power was utilized this turn, each player’s mana pool, and the turn number of the game. All calculations and changes to board state will be performed in this module.

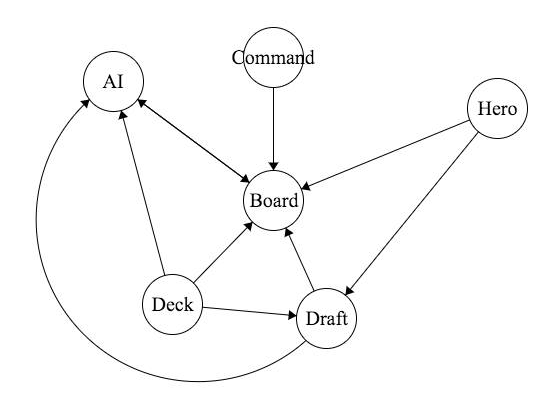
**deck.ml** – In this module, the outline of a card is defined. A card will have a name, an HP stat, an attack stat, an optional effect, a bool showing if it has stealth, a bool showing if it has taunt, a card type (one of four types: Protass, Zerg, Terran, or Spell), and a mana cost. In addition to the definition of card, this module contains a library of all cards imported to the game, a definition of a deck (a list of cards), and functions to draw or shuffle a deck.

**hero.ml** – This module provides a definition for hero, having a name and a power for each hero that can be defined. Additionally, a list of all heroes is defined here.

**command.ml** – This module contains the possible commands that can be called by the players from either the menu or from within the game. It also contains the parser that will parse the input of the players and convert them into the proper commands.

**draft.ml** – In this module, all the procedures for the draft mode of the game are defined. This includes the selection of one out of three randomly chosen heroes, 30 selections of one out of three randomly chosen cards, and the construction of a deck out of these 30 selected cards.

**AI.ml** – This module contains the algorithms required for the AI to play the game. This includes an algorithm for making a move given a board state and an algorithm for the AI to choose cards during the drafting process.



**Module Design:**

The design of the modules is documented in the .mli files.

**Data:**

The data maintained by our system will be for the most part the board states during a game, which will be held as references in the board module. Other than that, there will be the library of imported cards, which will be contained in the deck module as an array. Additionally, a library of possible heroes will be held in the hero module also in an array. The board state will be communicated to the players through printing of the game state as text.

**External Dependencies:**

This system will require the use of the CSV library.

**Testing Plan:**

In order to test our design as we implement it, we will utilize unit tests to check if the functions that we write in the modules are working as intended. In addition to these unit tests, we will test the modules individually by testing specific inputs to the modules. Then to test the overall system, we will attempt to play the game, testing each of the commands and attempting to find any bugs.