Dominion

Analysis, Design and Software Architecture

Jakob Melnyk, jmel@itu.dk
Christian Jensen, chrj@itu.dk
Frederik Lysgaard, frly@itu.dk

December 14th, 2011

Abstract

This project is about a virtual representation of the card game Dominion in C#. Dominion is a turn-based, deck-building game, where the objective is to gather more points than the other players. The game is played by 2 - 4 players.

Contents

1 Requirements

1.1 Mandatory

Must be able to play a full game of Dominion

- Must support 2 players in Hot-Seat configuration
- At least 10 Kingdom cards must work
- The game must be playable in a Picture-based GUI

1.2 Secondary

High priority

- Be able to play the game with 3 or more players
- Be able to use at least 20 Kingdom cards
- Be able to select Game Mode
  - Be able to play 'First Game' Card-set
  - Be able to play with 10 randomly select Kingdom cards
- Be able to see all Available Kingdom cards without scrolling

Medium priority

- Be able to view a Tooltip when mousing over any Card in the game
- Be able to play the game over LAN
- Be able to use all Kingdom cards (from the original version of the game)
- Be able to play all the Card-sets defined in the original rules

Low priority

- Be able to Draft Kingdom cards
- Be able to play the game over the Internet
- Be able to select different screensizes
- Be able to play in fullscreen

- Be able to create a User, that is saved across multiple games, with the following information:
  - Statistics
  - Options (if any)
  - Achievements (if implemented)
- Be able to support Extensions of the basic game
- Implement Achievements for funny and/or hard accomplishments

2 Overview

This project is about our virtual representation of the card game Dominion. Dominion is a turn-based, deck-building game. The objective of the game is to use Action cards to improve your chances or damage the opponent players and using Treasure cards to buy more powerful Action/Treasure/Victory cards to gain the upper hand.

We are planning on using a Model-View-Controller architecture. We want to separate our GUI from both the game rules and the state of the game via the controller. In essence we are likely to have a somewhat static model of the rules and a more dynamic and changing model of the state of the current game.

Frederik Lysgaard is the guy responsible for the design of our graphical interface. He is also the best Dominion player in our group. Because of this, he knows a lot of the usual strategies and is our general "go-to" guy when it comes to the tactics of the game.

Christian Jensen is responsible for implementing the way the different cards interact with the state of the game when used. Christian is also the guy who will be looking into the networking portion of the project if/when it becomes relevant.

Jakob Melnyk is responsible for modeling the state of the game and the communication between the GUI and the model (in our model-view-controller architecture). Jakob Melnyk is also the "version-control-guy", the person with the final word in discussions and the general log-keeper for the group.

3 Dictionary

3.1 General terms

This section describes the general "out-of-game" terms.

Achievements An achievement is token rewarded for funny and/or hard accomplishments within the game.

Card-set A card-set is 10 different Kingdom cards. Card-sets are used to create a different play experience every time you play.

Dominion The card-game we are making a virtual representation of. A link to the full rules can be found at Rio Grande Games [2].

Draft Drafting is done by player 1 selecting one Kingdom card to be used in the game, then player 2 selects a Kingdom card, player 3 selects a Kingdom card, player 4 selects a Kingdom card, then back to player 1. This cycle repeats until a set number of Kingdom cards have been selected.

Extensions Expansion packs add additional types of cards to the pool of cards.

Game Mode There are different possible game modes: draft, random card selection and predefined card-sets. These are selected before the game starts.

Hot-Seat Hot-Seat is the act of having 2 or more players play on the same computer. The active player "sits" in the hot-seat while playing, then passing the spot to the next player when his turn ends.

Message Type Messages of different types can be passed around in our server-client network.

Model-View-Control Often abbreviated MVC, Model-View-Control is often used to seperate something "showing" data and the actual representation of the data on the disk. Control is usually the middle-link that takes care of the communication between the two.

Picture-based GUI A pictured-based GUI is a visual representation of the state of the game. The different cards are shown as pictures in the GUI.

Server-Client In a client-server design, the clients communicate with the server and the server then relays the information it was given by the client to the other clients.

Statistics Statistics such as number of games played, numbers of games won/lost, and other similar data about gameplay.

Tooltip A box with text describing something in the GUI in detail.

User A user is an entity storing statistics and achievements over the course of different games.

3.2 In-Game terms

This section describes the types of cards, supply and other "in-game" terms.

Available Available Cards are the Cards that can be bought from the Supply.

Action Phase In an action phase, a player have one Action, which he or she may use to play an Action Card. Playing an action card this way always costs one Action. Cards played may allow a player to receive additional actions. The Action Phase ends when a player has no more Actions left or chooses not to use his or her remaining Actions.

Buy Phase When a player's Action Phase ends, the Buy Phase begins. In this, the player receives a "Coin" amount, which is the combined value of all Treasure Cards in his or her hand and any Action Cards, that add "Coins". The player can then use a Buy to buy any Card they want from the Supply. Played Action Cards can allow more Buys. Bought Cards are added to the Discard Stack. After the Buy Phase, the Clean-Up Phase begins

Card A Card is the basic playing unit in Dominion. Everything you 'own' is represented by a Card in your deck. Cards are primarily added to the deck through the Buy phase. Each Card has a value, which represents what it costs to get during the Buy Phase.

- **Curse Card** A Curse Card is a special type of Victory Card, which gives a negative amount of Victory Points. While these cards can technically be bought by any player and added to his or her deck, they are usually given to other players by using Attack Cards against them.
- **Kingdom Card** Kingdom Cards are what make each game of Dominion unique. With one exception all Cards here are Action Cards (one is a special Victory Card) and there are no Action Cards which are not Kingdom Cards. Each game requires selecting 10 of the 25 Kingdom Cards to use.
  - **Action Card** An Action Card is used during the Action Phase.
    - **Attack Card** An Attack Card is a type of Card which affects other players negatively. All Attack Cards are Action Cards and the "Attack" actives when the Card is used as an Action.]
    - **Action-Reaction Card** A Reaction card is used to respond to an Attack by another player. When an Attack Card is used against a player, that player may reveal a Reaction Card from his or her hand and do what the Reaction allows. Only one Reaction Card is in this game, 'Moat', which allows the player to negate the attack used against them.
  - **Kingdom Victory Card** A Kingdom Victory Card is a card that does generally not behave like usual Victory Card, but instead have special effects granting the player Victory Points.

Treasure Card A Treasure Card adds a number of "Coins" to spend in the Buy Phase. Note that a Treasure Cards value (the price to buy it) are usually different from what they cost to buy.

Victory Card A Victory Card gives a number of Victory Points at the end of the game. The player with the most Victory Points win the game.

Clean-up Phase The Clean-up Phase consists of putting all bought Cards, played Cards and Cards remaining in the Hand into the Discard Stack.

Deck A players Deck is his or her representation in the game. It consists of all the Cards that player started with and have bought during the game. A player's Draw Stack, Discard Stack and Hand is that player's Deck.

Discard Stack This contains previously played cards and any newly bought cards.

Draw Stack This contains face-down Cards for a player to draw. When there are no more cards available for a player to draw, the Discard Stack is shuffled and used as a new Draw Stack. Each player have their own Draw Stack and Discard Stack.

Hand The Hand represents a players current options in the following turn. These are drawn at the start of the game and each player draws a new hand after a turn has finished. When drawing a new hand, it always consists of 5 Cards.

Supply The Supply consists of 10 types of Kingdom Cards, 3 types of Treasure Cards, 3 types of Victory Cards and Curse Cards.

Round A game of Dominion consists of a number of rounds. Each Round is divided in to Turns, one for each player.

Trash Stack Sometimes a Card calls for itself or some other card to be Trashed. This means that it should be completely removed from the game and the Trashed Card is put on to the Trash Stack. All players share the Trash Stack.

Turn The player usually take turns in clockwise order. A players next Turn will be in the following Round.

4 Example

5 Revision History

Some of our commits to our Github (https://github.com/esfdk/BDSADominion) with commit ID. The commits are sorted by time of commit, descending. Full commit log can be found at: https://github.com/esfdk/BDSADominion/commits

———————————————————— Code freeze

c2dcc6786210ecf5ac74a8d34bcf07e72cf8360c : pex snapshot

a397b2b79291e8349e106ef1773406e3c3a57e55 : made a dotcover solitaire run

cae05ae98caefbeaf2cf31fa4eb9473507486ce1 : contract

00782159d83baf89bf3a41acea7f87d6484e263f : network design done.

28cb2b5c8f343793662fa0cec0d8dae0e8476564 : Fixed bug in shuffling of cards. Not as random as it was before.

234ca16500ee52cbf8f37dff117de83c07febebe : informal is now legit

e22f402855a48cc2ce93ce651305bb801e121d75 : Key and control update

fe7e9ba83b41594a5ef91560af0fb505a55da627 : Some pex tests, better shuffle method, fixed adventurer and more.

3074885da9f22646bac0becbfe8bc45ae975723a : Made some changes to GUI it looks good now

e083766fe15b127f0a9ae03528e1eea956a6702e : dotCover solo run added

1951adb1e07b4d9735847e0bf3e622464380aa2f : Trying to get these weird bugs out of the game.

7c53affdf9aba583efc71dedab9476fe1441eeea : Done with some of the docs for the GUI

1173a01763032deb6b033d01471039be84a61f0f : Wrote some of system design and started on system production

a35c4939236770661fb751f378be4c7e5259d0fe : Formal bon 99% done for Melnyk.

97ae13e744b2283f0a0aecafae954b357d0e2c04 : Minor fix in Control to possibly prevent crashes. Also added some more Contracts to the formal BON.

aa1c0af808bc536cc50e5e988e185834ca34bd13 : GUI start update working now

9d4d9b2bcf0b8acda2544fd3c93ad926122237c4 : added the listeners

44f6bb367f5b56014069c2bfd8b1e8fd1a61da7c : Made files to be used in the hand-in.

911af507af6ec9e64a36d470280d1b4d7e9707d7 : Network starts the game

b7210739f2ed0b37875ab7d4a858559602e029b2 : Server PreGame messaging working. Still lots of WriteLines.

9a3c7bfc2334e04a0709a09eeb0d1b4a532327a7 : Much of Melnyk's part of Control is done! Mainly need communication with server!

3417bf70282d5b3bdfd39c8855f258eeb05e57a2 : Network seems to be working alright.

c710a8d4a93dfdc6623be4022bcb8da2a0259d4d : GUIInterface done. Network testing

bb1b5dbabccfc6c7d3c1d25ffa97fcf5e8d37b64 : Starting tests on network interface

537f2f25db57b263d63cac1ab298a69de5e80bf3 : GUI-Interface almost done, only need endPhase button in gamestate.

9f2a6f20d46e0037a2dfa2d917b3030b720551c3 : GUI Interface getting up and running. Console added :D

a883a6fd4d552fe71cb3aae641a7ec0af4037e06 : GUI now draws what needs to be drawn

7c368444740ed2bc8d49027e7a7d666674600ec2 : Finished gamestate and player for now

24454e804bde99435b2022244e6307d89a9c323a : GUI now updated to work with CardName

1ddb9247acd2f32b0e979f7dd561bd98edf4d570: Alfa GUI is done. LOCK AND LOAD!

5c28c48e556fea810b4dec2e3db4108f6c455ae4 : Diary and some gamestate stuff done.

857d25376348dd601579e20899eb76efc074a30d : Networking added. More or less direct copy of work done outside of project. Interfacing with Control added, but not completed

bec4f5f40ae4aca6189ab41078fbd9c0475ca8e5 : Started coding gamestate

523bd81b3a6e0e20185ff5e2b3aa5fce13ccbc25 : Arranged classes into folders for better overview.

ff7248e0612f9f2e53525fde41de536b22d624b7 : Started creating gamestate in code

5e858aff8cab0331b3a504f4d4280a4dbf774cc7 : Did lots of BON

1d7eb386c7839a9c97ac1b8efb286b12fdaf7d97 : Added all the necessary classes for the GUI part of the project

6af320925f540af6842e9b6156355738d3cc94b1 : Did a lot of informal BON

25ff80260b970ec14e36728249e3f9cb324052b0 : Did some BON and added some diary entries

e0ce1ea7ec8bd8345bfb6f66c77c3d908d40af03 : Put in BON files

86029d88fde9a403f777855e433cd3fd8775ff7e : Added diary files

533e909a8d36dbffb1c6953cf99639a5baa90fbc : Added GUI classes

905a2d3a02218dec666bff65c4738306e9e7877c : Starting project

6 Milestones

6.1 System analysis

6.2 System design

6.2.1 General

We decided to use an MVC pattern for our overall structure, because we felt we could seperate the model (the Gamestate), the View (GUI) and Control (Game Logic and more). We also use a Client-Server architecture to do the network communcation.

6.2.2 GUI

Frederik Lysgaard In terms of system design, there isn't really much to say to the GUI part since none of us knew the XNA framework, and therefore didn't have any knowledge of the limitations and benefits. This led to a very ad hoc way of designing the GUI system, for the first draft I used my exstensive knowledge of the game to try and get a overview of the components needed to properly display and play a game of Dominion. This resulted in the basic design for the GUI, but since I was still a newbeginner to XNA, this led to a design with almost all the right classes but with some strange inheritance, all in all it felt weird which made me study XNA some more, and after having used some days getting familiar with the framework I came up with the second and final draft.

Even though this was the final draft, it was far from perfect, I would for example if had, had more time with XNA before the project suggested some inheritance between the zones since they all is made from the same sprite "template". So to summarize there wasn't really any general battle plan for the GUI system design at the start of the project, which was a challenge, that taught me one thing, if you know you're going to work in a new framework, learn it beforehand.

6.2.3 Client & Server and Control (Server and start-up parts)

Christian Jensen

6.2.4 Gamestate and Control (Game Logic)

Jakob Melnyk

6.2.5 BON specification

In this sub section, we have included all of our BON. If we had had more time, we would have focused on doing scenarioes - especially in a context that could give us more code coverage.

```
system_chart BDSADominion
indexing
        author: "Frederik Lysgaard (frly@itu.dk)";
        author: "Christian 'Troy' Jensen";
        author: "Jakob Melnyk (jmel@itu.dk)";
        supervisor: "Joe Kiniry";
        course: "BDSA-E2011";
        created: "28th November 2011";
        lastModified: "14th December 2011";
explanation
        "System chart for the BDSADominion project in the Analysis, Design and Software Architecture
cluster DOMINION_SYSTEM
        description "The Dominion game system."
end

cluster_chart DOMINION_SYSTEM
class CONTROL
        description "The man in the middle between the three parts of the architecture. Contains gam
cluster GUI
        description "Used to display the current state of the model and to interact with the user."
cluster GAMESTATE_CLUSTER
        description "The 'model' of the project. Remembers information about most of the states and
cluster NETWORK_CLUSTER
        description "Communicates between different instances of the application across LAN."
end
```

```
system_chart BDSADominion
indexing
        author: "Frederik Lysgaard (frly@itu.dk)";
        author: "Christian 'Troy' Jensen";
        author: "Jakob Melnyk (jmel@itu.dk)";
        supervisor: "Joe Kiniry";
        course: "BDSA-E2011";
        created: "28th November 2011";
        lastModified: "14th December 2011";
explanation
        "System chart for the BDSADominion project in the Analysis, Design
cluster DOMINION_SYSTEM
        description "The Dominion game system."
end

cluster_chart DOMINION_SYSTEM
class CONTROL
        description "The man in the middle between the three parts of the a
cluster GUI
        description "Used to display the current state of the model and to
cluster GAMESTATE_CLUSTER
        description "The 'model' of the project. Remembers information abou
cluster NETWORK_CLUSTER
        description "Communicates between different instances of the applic
end
```

6.3 System production

6.3.1 General

Our split into the different, very seperate parts of the code, made it somewhat cumbersome to combine at the end, but once it actually combined, it was quite an easy ride home in terms of getting the game to play. The different parts of the architecture should be quite replaceable, especially considering the GUIInterface and NetworkingInterface concepts and the way Gamestate works.

6.3.2 GUI

Frederik Lysgaard The production of the GUI can be split into three parts:

- The initial idea.
- The attempt to write it.
- And at last the rewrite of it all.

So let's start at the begining. The initial idea of how to produce the gui was that all drawn classes should inherit from a super Sprite class but as I began coding I realized that the idea wouldn't be so optimal, since we had different objects with different positions which at that point, in my XNA traning, semmed to make it all very hard to draw, atleast with different positions.

So after realizing that my first attempt of code was not going to work, I set to rewriting what I already had and try and reform it with my new knowledge of XNA. I then ended up with what is our end GUI which consist of a lot of zones where you can either draw buttons or cards sprites to, this seemed like a extremly easy straight forward solution, even though if I had had more time, I would have loved to code in some inheritance, espcially a super zoneclass that would act as template for the other zoneclasses.

6.3.3 Server and Control (Server and start-up parts)

Christian Jensen

6.4 Gamestate and Control (Game Logic)

Jakob Melnyk

References

[1] http://www.riograndegames.com/games.html?id=278

[2] http://www.riograndegames.com/uploads/Game/Game_278_gameRules.pdf

[3] Simon Henriksen shen@itu.dk