Dominion in Python

W200 Project 1

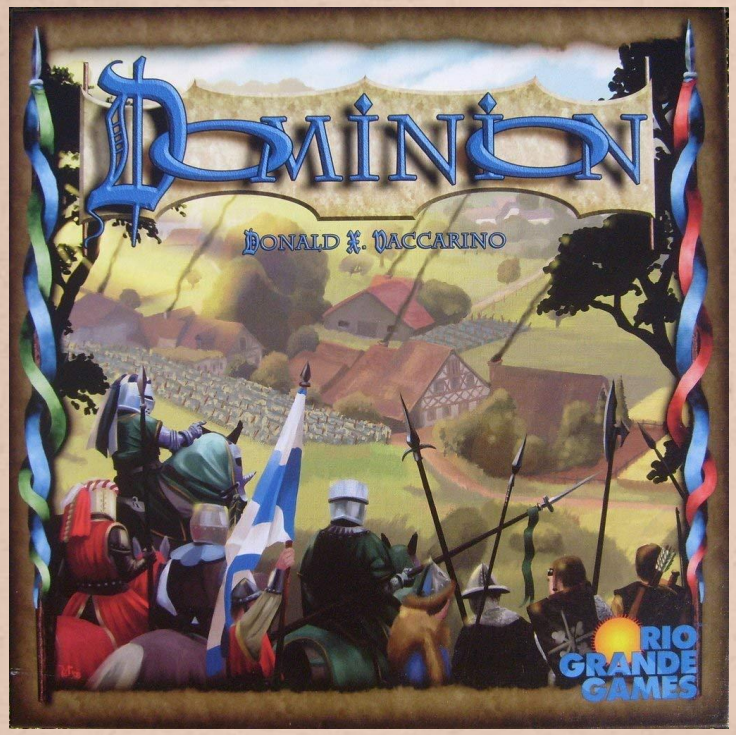
Matt DeGostin

March 11th, 2019

Project Overview

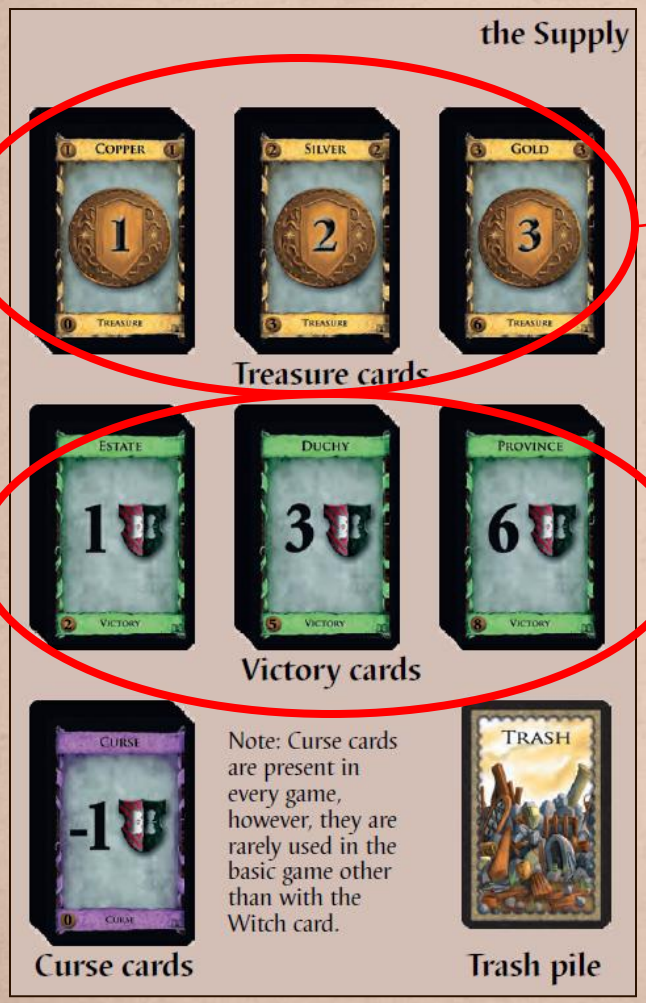
- A Python OOP implementation of Dominion card game
- 1-4 human players
- Alternate turns via command line
- Capability for additional features (like CPU players!)
  - Polymorphism
  - Modularity

What is Dominion?



- Deck-building card game.
- Players alternate turns, playing Action cards, and buying cards to strengthen their deck.
- Player with the most victory points at the end wins.

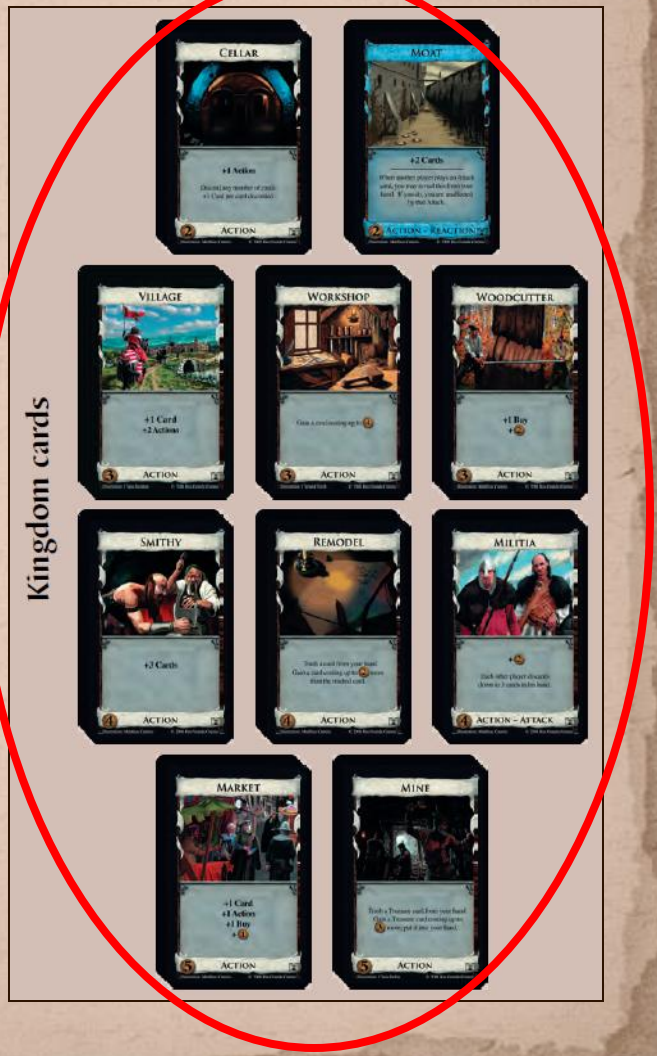
How do you play?



The game's currency

Territory – collect these to win

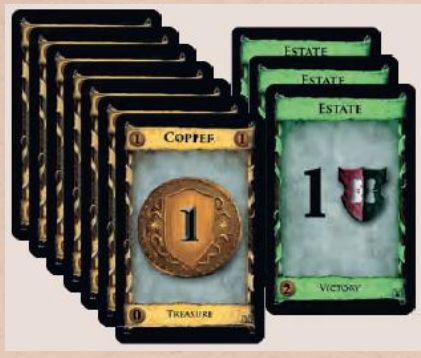
Action cards – each of these has a special effect when played



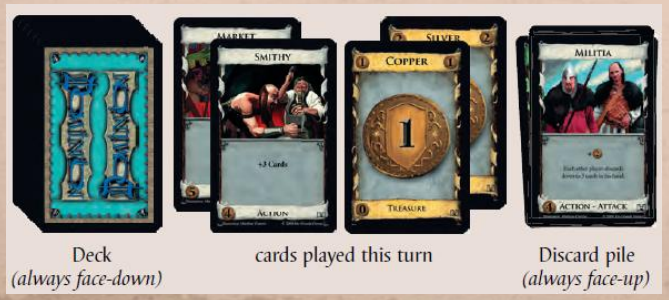
How do you play?

Select # Players

Deal Starting Decks

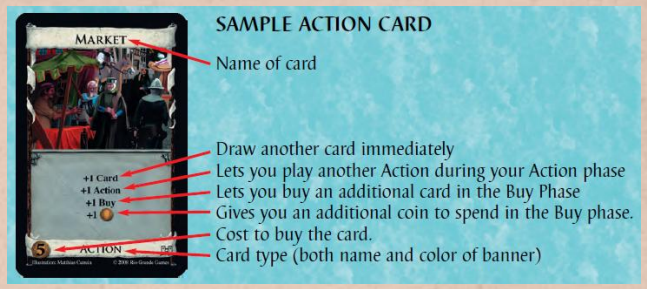


Setup Play Area

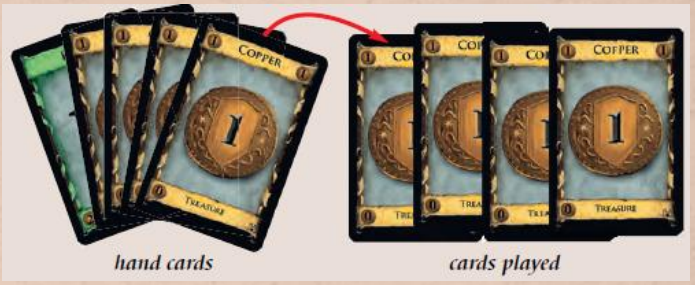


During Each Turn

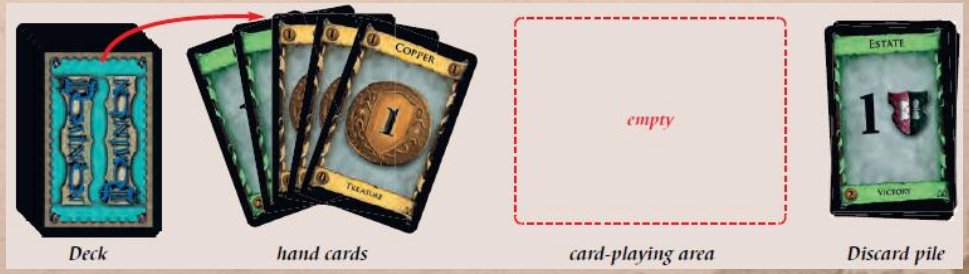
1. Play Action Card(s)



2. Buy Card(s) from Supply



3. Clean-up (discard then draw)



Object Oriented Design

Card Class

- Name, cost, etc.

Treasure Class

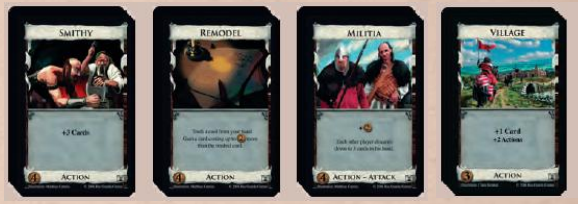
- # coins

Victory Class

- # victory points

Action Class

- action method



Container Class

- Generic group of cards
- Methods to add/remove
- Used for hand, deck, discard pile, etc.

Supply Class

Object Oriented Design

Player Class

- Name
- Number of points
- Deck, hand, play mat

Human Class

- Methods for action, buy, discard, etc.
- Uses input()

CPU Class

- Difficulty, strategy
- Methods for action, buy, discard, etc.

Config Class

- Holds all of the data for the game
- Players, decks, supply, etc.

Game Class

- Holds the high level logic for game play
- Whose turn is it?
- Who won?

Challenges

- User Interface
  - Needed to be able to represent the supply piles, player's hand, available actions, etc. all on in one terminal.
- Game Flow Control
  - Control of the game passes between the game itself, individual players, action cards, and affected players and back up the chain.
  - How does the lowest level entity know about what's going on at the highest level?
- Shallow and Deep Copies
  - Needed to make sure every instantiated card was its own object (deep copying).
  - Needed to make sure each player was unique but ran into issues with excessive deep copying, where a copy of the current player was affected but not the player itself.