

Technical Design Document

**Dominion AI Using Monte Carlo Tree Search (MCTS)**

Version 1.0

|  |  |  |
| --- | --- | --- |
| **Position Title** | **Name** | **Signature** |
| Programmer | *Jonathan Sarasua* |  |
| Advisor | *Matt Butler* |  |

* ***Delete all instructional information before turning in final version.***
* *Replace the image above with an appropriate image.*
* *Replace names above.*
* *Update the Header and Footer with the correct information.* 
  + *Lock the date to when the version is completed, not the date of its opening.*

Table of Contents

[Table of Figures 5](#_Toc64922132)

[Introduction 6](#_Toc64922133)

[Overview 7](#_Toc64922134)

[Scope 7](#_Toc64922135)

[End Product 7](#_Toc64922136)

[Deliverables 8](#_Toc64922137)

[Table of deliverables 8](#_Toc64922138)

[Technology Sources 9](#_Toc64922139)

[Background and Previous Work 10](#_Toc64922140)

[MCTS 10](#_Toc64922141)

[Dominion 10](#_Toc64922143)

[Theory 12](#_Toc64922144)

[Monte Carlo Tree Search 12](#_Toc64922145)

[Selection 12](#_Toc64922147)

[Expansion 13](#_Toc64922149)

[Simulation 13](#_Toc64922150)

[Backpropagation 13](#_Toc64922151)

[Choosing the best move 13](#_Toc64922152)

[Implementation 14](#_Toc64922153)

[Chance Nodes 14](#_Toc64922154)

[Selection 14](#_Toc64922155)

[Expansion 14](#_Toc64922156)

[Simulation 14](#_Toc64922157)

[Backpropagation 14](#_Toc64922158)

[Architecture 14](#_Toc64922159)

[Results 15](#_Toc64922160)

[Selection 15](#_Toc64922161)

[Expansion 15](#_Toc64922162)

[Simulation 15](#_Toc64922163)

[Bibliography 16](#_Toc64922164)

# Introduction

*Add your usual thesis introduction. Should be a short paragraph between 3 – 5 sentences in length, use common industry terms, and reference important existing algorithms or games.*

Dominion is a deckbuilding game where every turn you try to play and buy cards to make your deck better with the eventual goal of buying more victory point cards than your opponents. The artifact is both a playable game of Dominion and an AI using MCTS as its algorithm. The AI is implemented with chance nodes to handle the hidden information and uses Upper Confidence Bounds Applied to Trees (UCT) as the selection method. The AI is tweakable to show how changes to the AI change the result. Simple state machine strategies for Dominion like “Big Money” and “Single Witch” are used to test the strength of the AI.

# Overview

Deckbuilding games like Dominion have built in complexity with handling hidden information like not knowing what cards will be drawn or what cards the opponents have. Every card added to the player’s deck changes the way the deck plays. MCTS is an exciting AI strategy because of not relying on domain knowledge. The focus can then be on making minor changes instead of trying to solve the entire game.

## Scope

*Give a brief statement of the scope of the project from a technical point and view. Tie this to the design vision and introduce the overarching development plan for this project. Insert as much detail as necessary.*

Scope limitations such as not implementing all cards and not handling different sets of cards.

## End Product

*Describe the pieces that are going to be built in order to develop the final game as defined in the Game Design Document. This sets expectations for amount of work necessary.*

*Gameplay*

* *List separately all game mechanics that need to be implemented*

*Game Objects*

* *List separately all game objects that need to be implemented*
* *For example:*
* *Player Character(s)*
* *Enemies (list with description)*
* *Pick-ups*
* *Destructible objects*
* *Vehicles*
* *Etc.*

*HUD & UI*

* *All GUI attributes implemented*
  + *List each aspect of the UI separately*

*Menu Systems*

* *All systems implemented*
* *List menus separately, with purpose and placement*

*Etc.*

# Deliverables

## Table of deliverables

Table of all of the features in a similar format to what Danny did

# Technology Sources

I will say I used Visual Studio with TinyXml2, STBI, and potentially ImGui

# Background and Previous Work

### MCTS

MCTS has been used with many different games like Alpha Zero with chess. [1] It is commonly mixed with other AI strategies with thousands of papers covering a wide range of use cases. MCTS with Dominion though has been researched very little. Only one paper was found on the topic by Jansen and Tollisen. [1] They changed the simulation step to use a greedy heuristic to solve issues random has with the game of Dominion. They also tested using UCB versus UCT for selection.

### 

Figure : Dominion setup [2]

### Dominion

Dominion is a deckbuilding card game for two to four players where each tries to get the highest victory points. The game was published by Rio Grande Games. [3] During setup piles of treasure, victory, and action cards are laid out in easy reach of the players. The game ends when any three piles are empty, or the Province victory card pile is empty.

At the beginning of the game each player starts with a deck of ten cards, seven coppers and three estates. The decks are shuffled then each player takes turns playing one action and buying one card. At the end of each turn all cards played and still currently in their hand are discarded and five new cards are drawn. If the deck is empty, the discard pile is shuffled and that becomes the new deck. Action cards when played allow the basic rules to be modified. For example, the Village card allows the player to play two more actions and draw one more card. Any number of treasure cards can be played which give the player money to buy other cards.

Actions and treasure cards purpose is to give you enough money to buy victory cards which are the only way to gain victory points. Victory cards do not help your deck otherwise, so its important to buy them only when you plan on trying to end the game as quickly as possible. [4]

Figure : Treasure cards [4]



Figure : Curse and Victory cards [4]



Figure : Examples of Action cards [4]

# Theory

*Include as many sections as necessary to cover all theoretical background information that is considered a prerequisite to the thesis. Describe important math and algorithms.*

## Monte Carlo Tree Search

MCTS is an AI strategy based on the idea that if a perfect tree of all game states existed for a game, then the best move would be based on which move maximizes your chance of winning while minimizing your opponent’s chance. Since a perfect tree is not feasible for many games, MCTS tries to build a tree strategically to look for the most promising paths while still exploring other moves. The method for choosing the best area of the tree to expand uses Upper Confidence Bounds Applied to Trees (UCT). [1] To build the tree four steps are done repeatedly until a move is required. The four steps are selection, expansion, simulation, and backpropagation.

### 

Figure : MCTS Steps [2]

### Selection

The first step in MCTS is selection. At every node, two questions are asked:

Have all moves been expanded?

If not, what child node is the best to continue down the tree?

The first question is yes only if the number of child nodes is equal to the number of possible moves. If not all moves have been expanded, then a random move of the remaining choices is chosen for the expansion step. [1]

##### Upper Confidence Bounds Applied To Trees (UCT)

### 

Figure : UCT formula [5]

The UCT formula is the idea that exploitation should be balanced with exploration. [6] wi/ni is the number of wins at the current node divided by the number of simulations. This number will be between zero and one. The second half tries to nudge selection towards the least commonly visited children. Ni is the number of simulations the current node’s parent has done and ni is the number of simulations the current node has done. Note that the parent’s number of simulations is the sum of its children plus one where the one is the first visit to the parent. C is the exploration constant most often set to square root of two. [1]

### Expansion

Expansion is where the move selected is added to the tree. Only in expansion does the tree grow.

### Simulation

During simulation the node expanded during expansion step is used as a starting point for a game. In pure MCTS this step is where a game of random moves for both players is played. The result of a win, loss, or tie is passed to the final step of backpropagation. The simulation step can be easily modified to use a different AI strategy instead of random to speed up the accuracy of results.

### Backpropagation

Backpropagation takes the result from the simulation and saves it in the current node. Each node has the number of wins and simulations saved. The result is then sent up the tree where each parent also saves the result. Note that the wins is relative to each node, so as the result gets sent up it will flip between a win or loss depending on the current node.

### Choosing the best move

One of the benefits of MCTS is the ability to ask the tree at any point what it believes the best move is. The two most common methods choosing which move leads to the highest win rate or which move has been simulated the most. The highest simulations method puts the faith in the selection method. Using win rate can sometimes be inaccurate if the number of simulations is low.

# Implementation

Here I will talk about my implementation of MCTS and include where I differed from pure MCTS. Using heuristics as the simulation method, not allowing all expansion paths, etc.

### Chance Nodes

### Selection

### Expansion

### Simulation

### Backpropagation

# Architecture

Here I will talk about how the code is laid out. MCTS is separate from the Dominion game and must ask the game for what is playable and what isn’t. Also all moves still go through the game. The basic methods for MCTS match up to the four phases

#### *Engine*

#### *Game*

# Results

Here I will talk about lessons learned and what worked the best. I will give data on how much better with tables.

### Selection

### Expansion

### Simulation

# Bibliography

|  |  |
| --- | --- |
| [1] | J. V. Jansen and R. Tollisen, "An AI for Dominion Based on Monte-Carlo Methods," 2 June 2014. [Online]. Available: https://pdfs.semanticscholar.org/28b6/ada13e948cfaee4af5138ee667d404eb01ac.pdf. [Accessed 9 July 2020]. |
| [2] | "Monte Carlo Tree Search: Implementation for Tic-Tac-Toe," Stack Overflow, 22 April 2014. [Online]. Available: https://stackoverflow.com/questions/23803186/monte-carlo-tree-search-implementation-for-tic-tac-toe. [Accessed 9 February 2021]. |
| [3] | "Dominion\_(card\_game)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Dominion\_(card\_game). [Accessed 9 February 2021]. |
| [4] | "Dominion Beginners Guide | Tutorial," UltraBoardGames, [Online]. Available: https://www.ultraboardgames.com/dominion/beginners-guide.php. [Accessed 9 February 2021]. |
| [5] | "Monte Carlo Tree Search," Wikpedia, [Online]. Available: https://en.wikipedia.org/wiki/Monte\_Carlo\_tree\_search. [Accessed 9 February 2021]. |
| [6] | L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Machine Learning: ECML*, 2006. |
| [7] | "How Does Monte Carlo Search Tree work?," Stack Overflow, June 2017. [Online]. Available: https://stackoverflow.com/questions/44230911/how-does-monte-carlo-search-tree-work. [Accessed July 2020]. |
| [8] | "Big Money," 20 October 2018. [Online]. Available: http://wiki.dominionstrategy.com/index.php/Big\_money. [Accessed 9 July 2020]. |