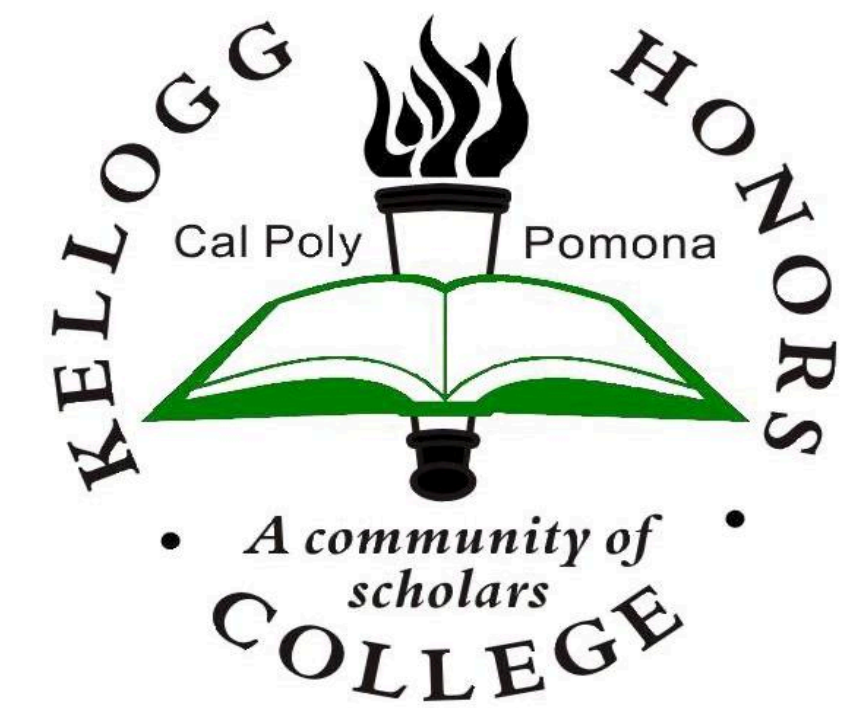# Isolation Game Using Monte Carlos Tree Search

Jordyn Sato, Computer Science

Mentor: Dr. Adam Summerville

Kellogg Honors College Capstone Project

Introduction

Isolation Game is an open-source, two-player video game developed in Godot with an artificially intelligent computer opponent. The purpose of this capstone is to improve my skills in software engineering and to explore an artificial intelligence algorithm, Monte Carlos Tree Search.

Development

Isolation Game was developed over 9 months with over 400 lines of code. Godot uses GDScript, a Python-lite backend scripting language, and nodes arranged in a tree-like manner for development. Games in Godot can be developed through either the node editor by adding nodes manually or through the script. Isolation game is made up of 45 manual nodes, but also creates nodes through the script when initializing the game board and performing Monte Carlos Tree Search.

Game Features

Isolation Game features a full game loop and the following gameplay mechanics:
- The player can choose one of three difficulties for the computer
- Highlighted squares show the player what moves are available to them
- Click to move the player's queen into an available spot
- Taken squares are signified by red x's

How to Play

- Click "Start" on the main menu
- Read the instructions and click "Ok"
- Click the corresponding button to choose the difficulty of the computer
- Highlighted squares will appear for the player to click on a valid move
- Each time a player moves between squares on the board, that square is marked as "taken" as indicated by the red "X"
- The game ends when either the player or computer is unable to make a valid move and will show a corresponding win/lose screen
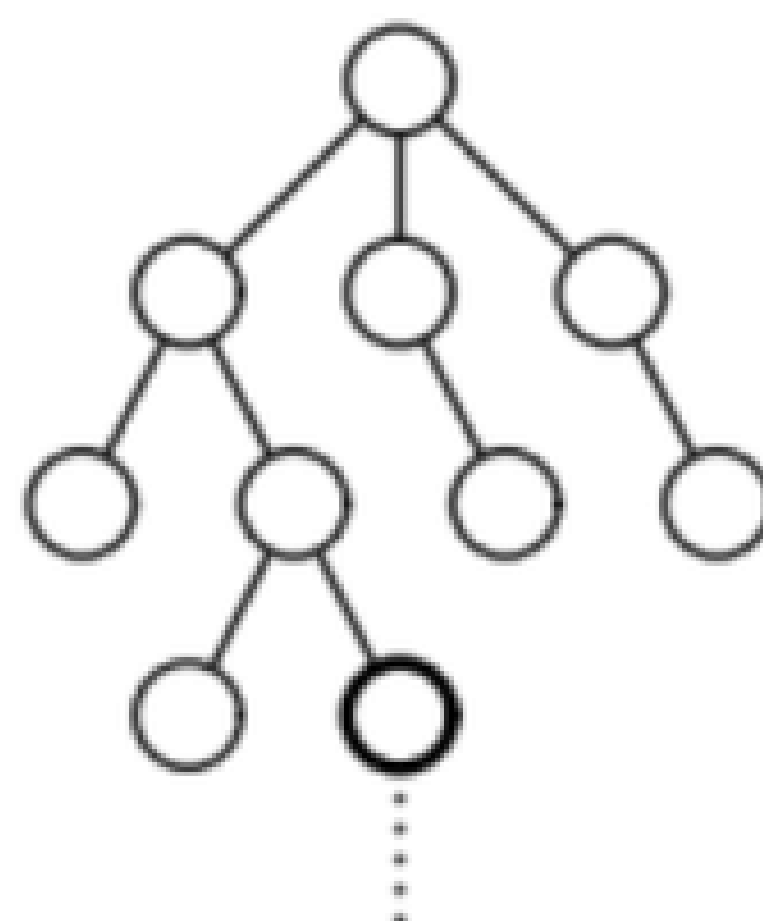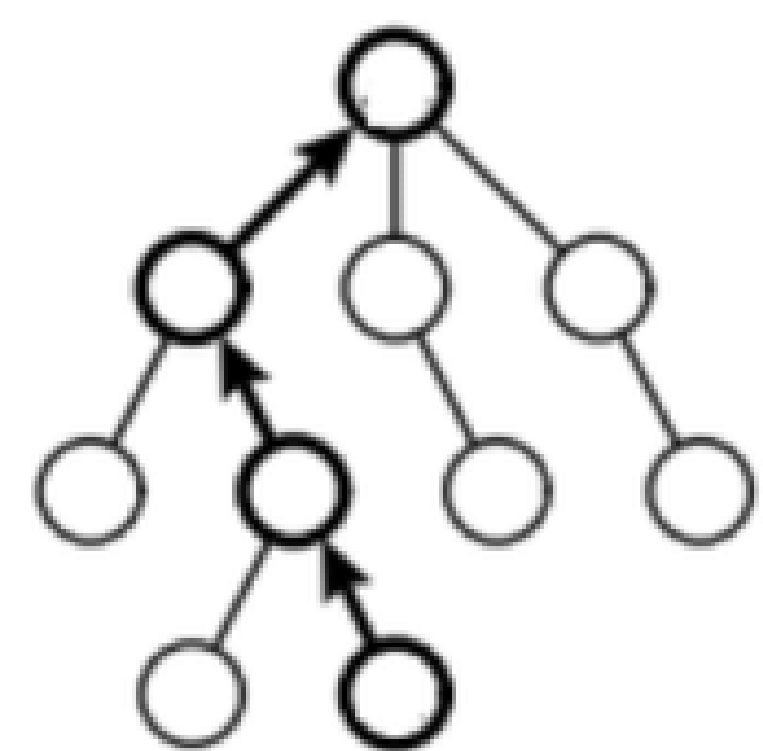- Click "Go back to Main Menu" to start over
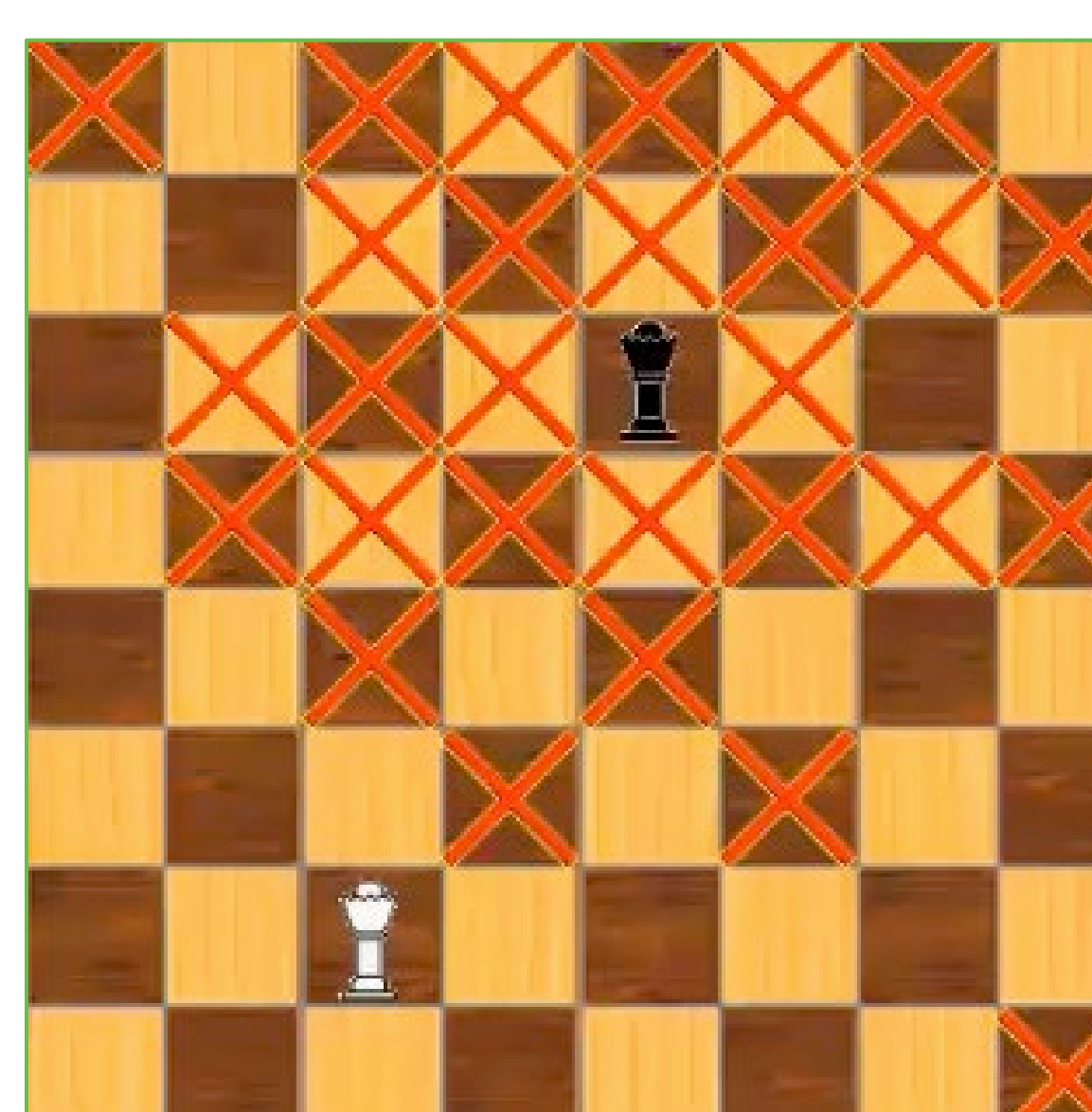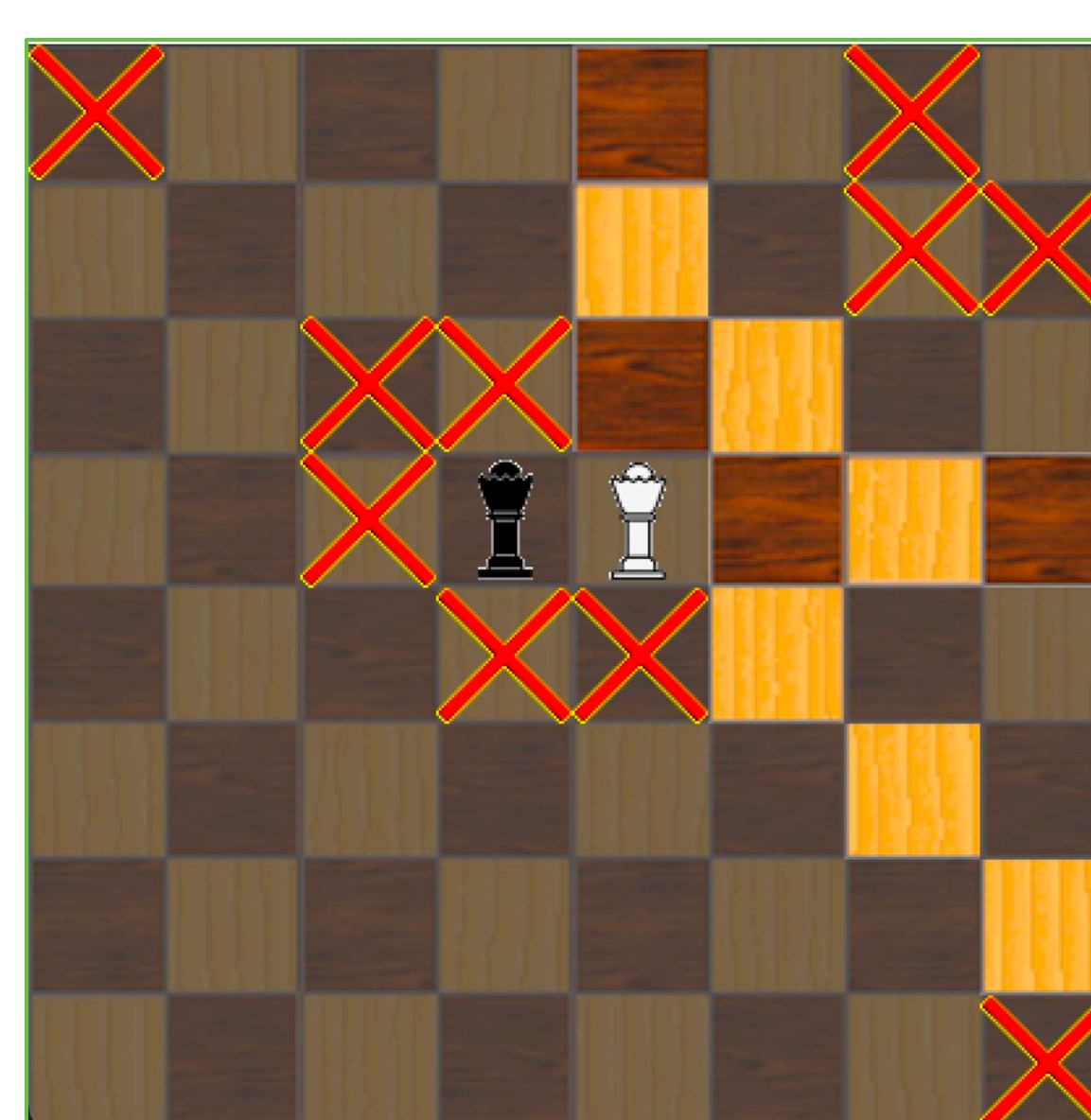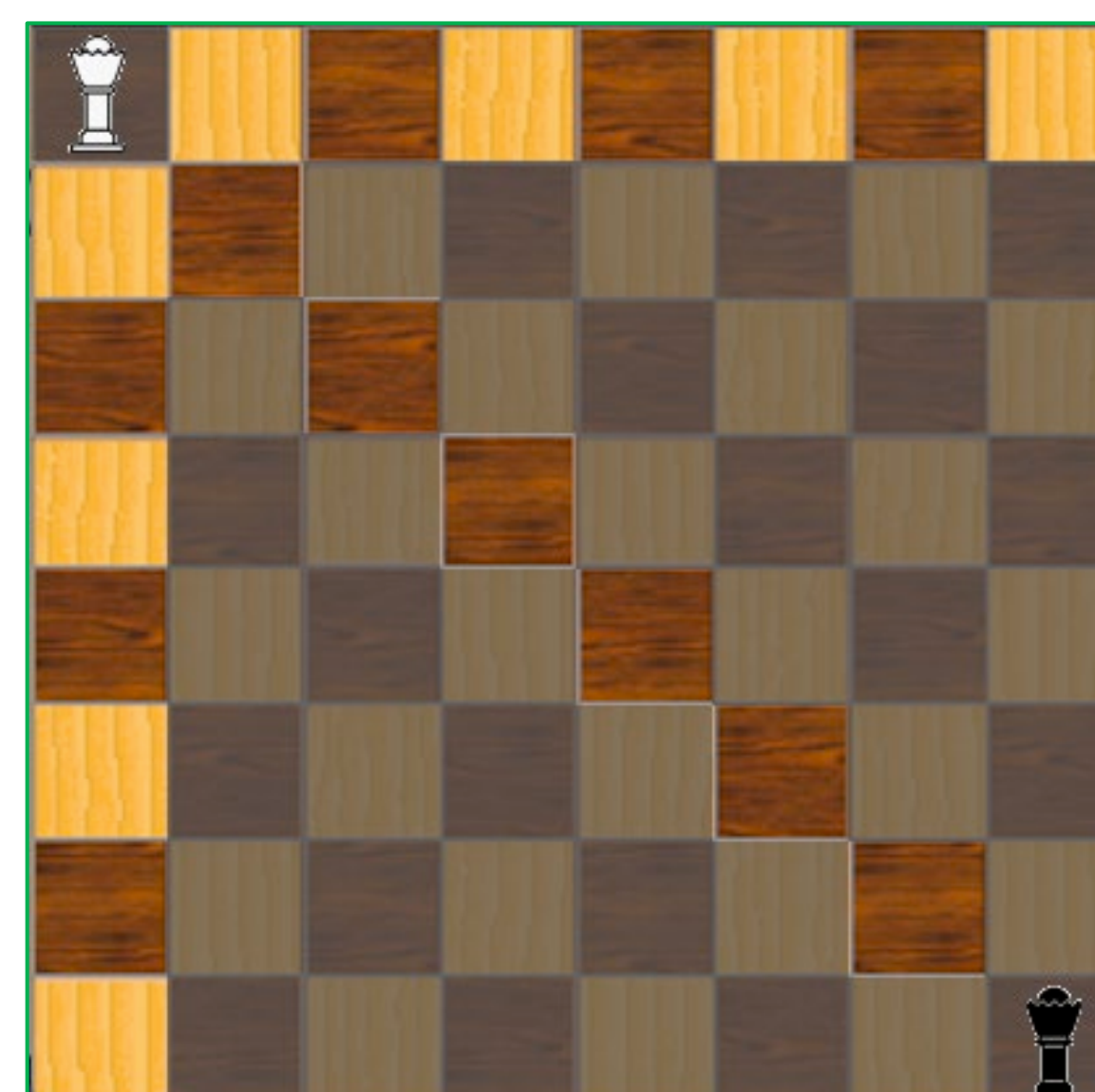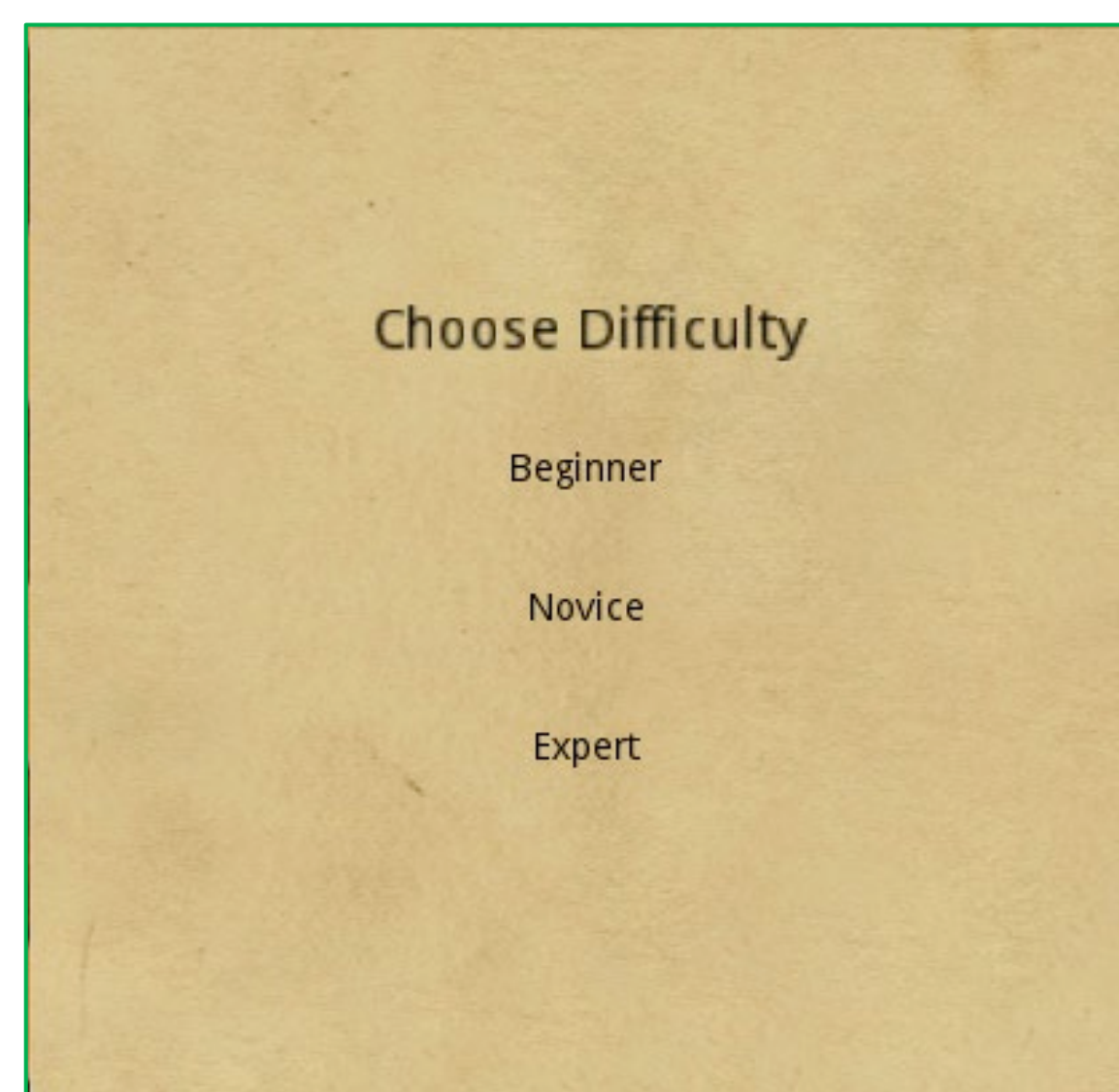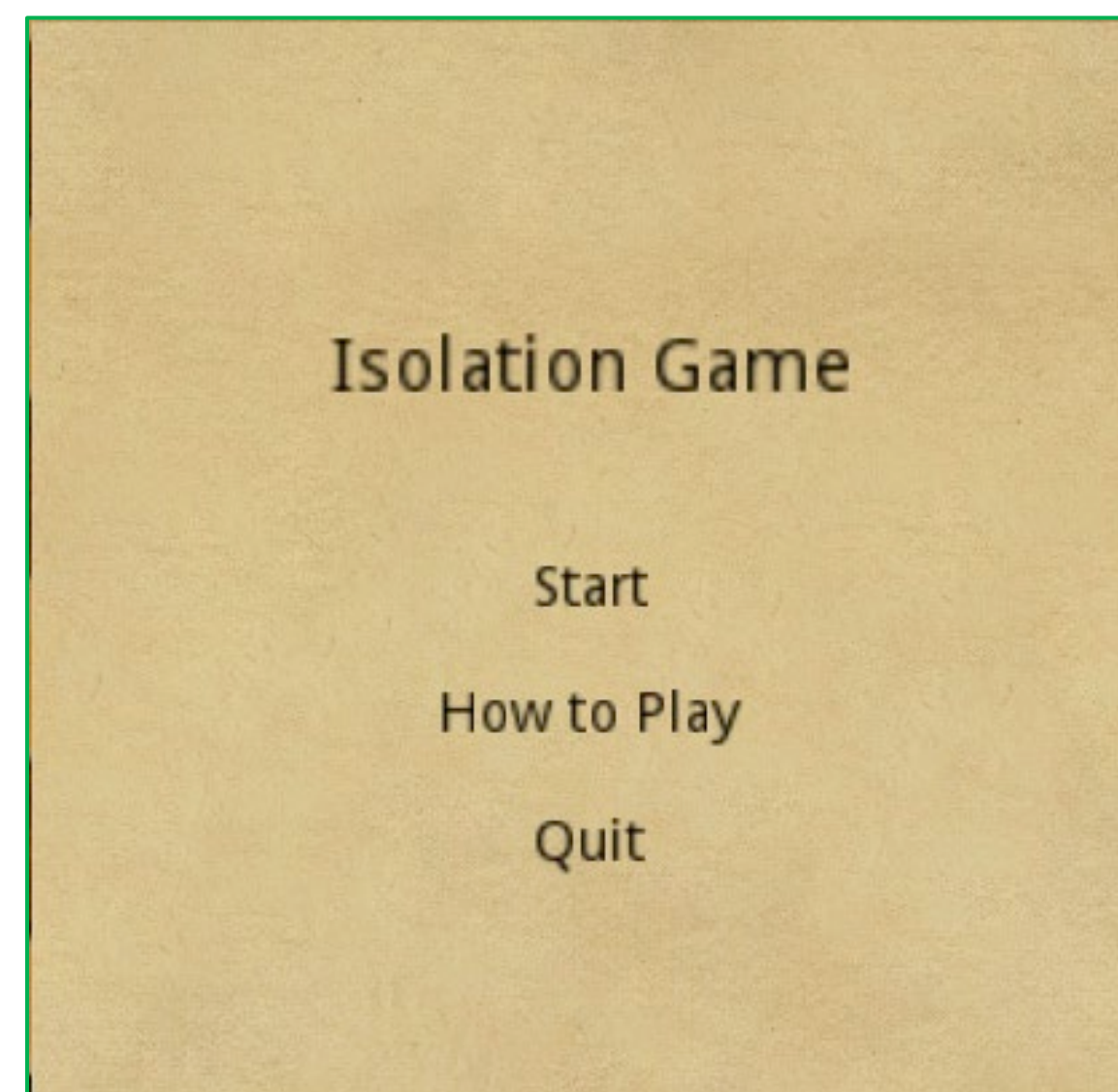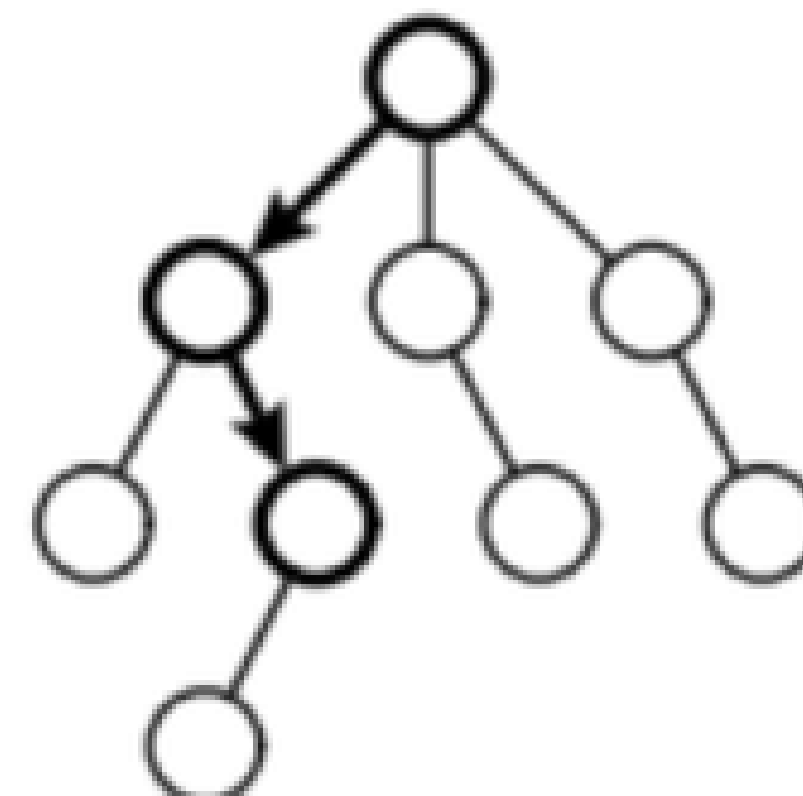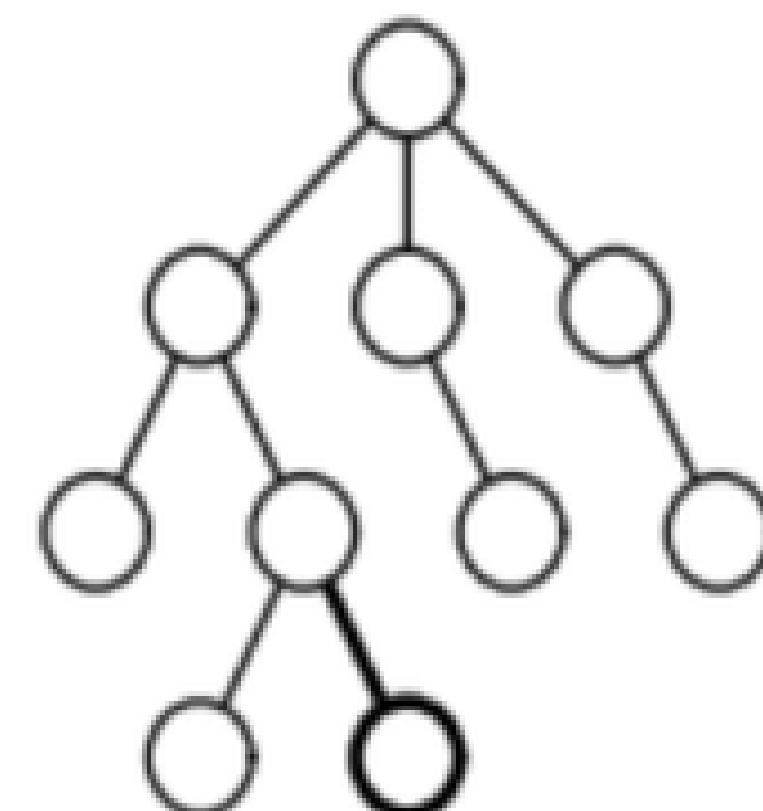
Monte Carlos Tree Search

Monte Carlos Tree Search (MCTS) is the tree search algorithm used in Isolation Game so the computer can choose its next move. There are four main steps when performing Monte Carlos Tree Search:

Selection

The artificial intelligence recursively selects a child node that is non-terminal and has unvisited children based on the upper confidence tree bound algorithm (UCT):

$$UCT = \frac{w}{n} + c\sqrt{\frac{\ln N}{n}}$$

- w: number of wins
- n: number of times the artificial intelligence has already visited the child node
- N: number of times the artificial intelligence has visited the parent node
- c: constant equal to $\sqrt{2}$

Expansion

The artificial intelligence takes the selected node and determines all the possible next game states and saves it as its children nodes.
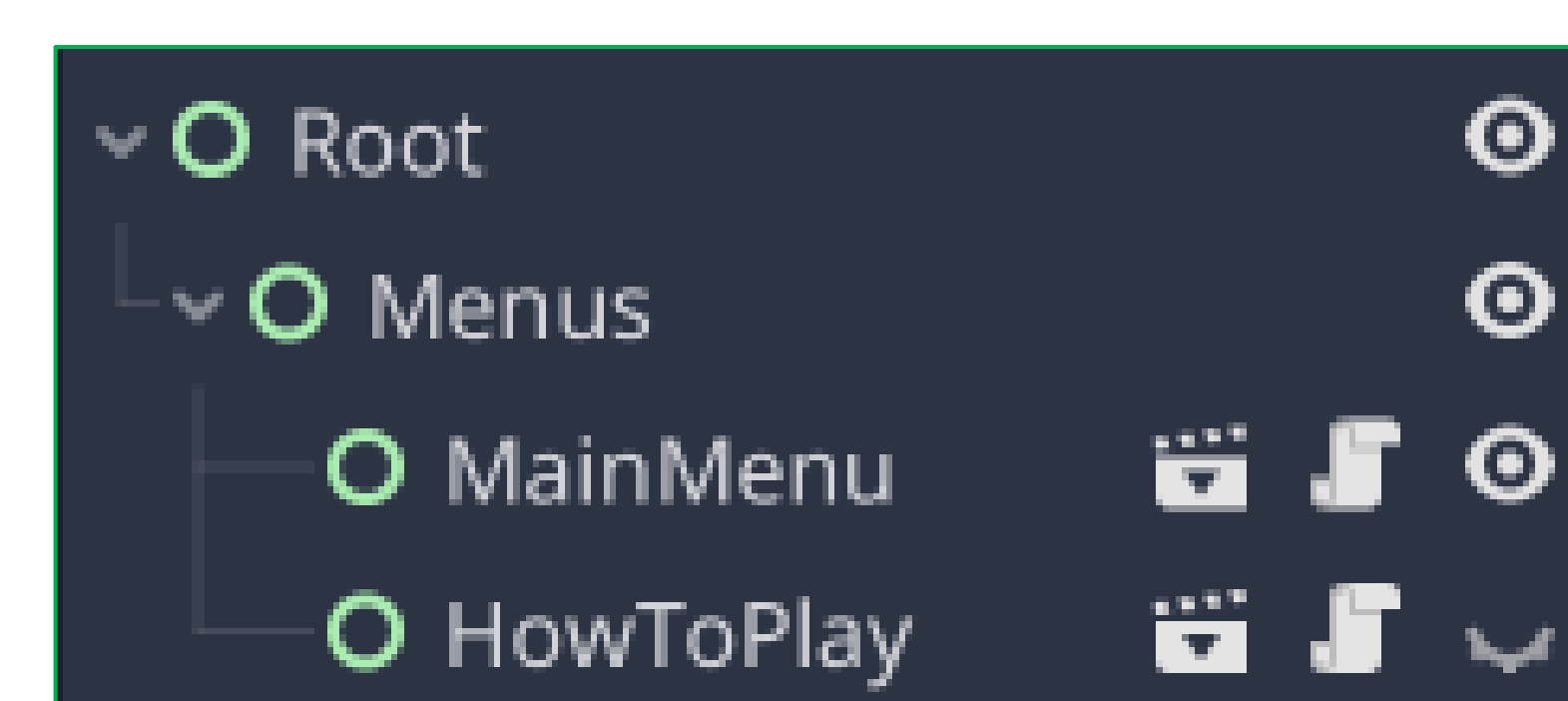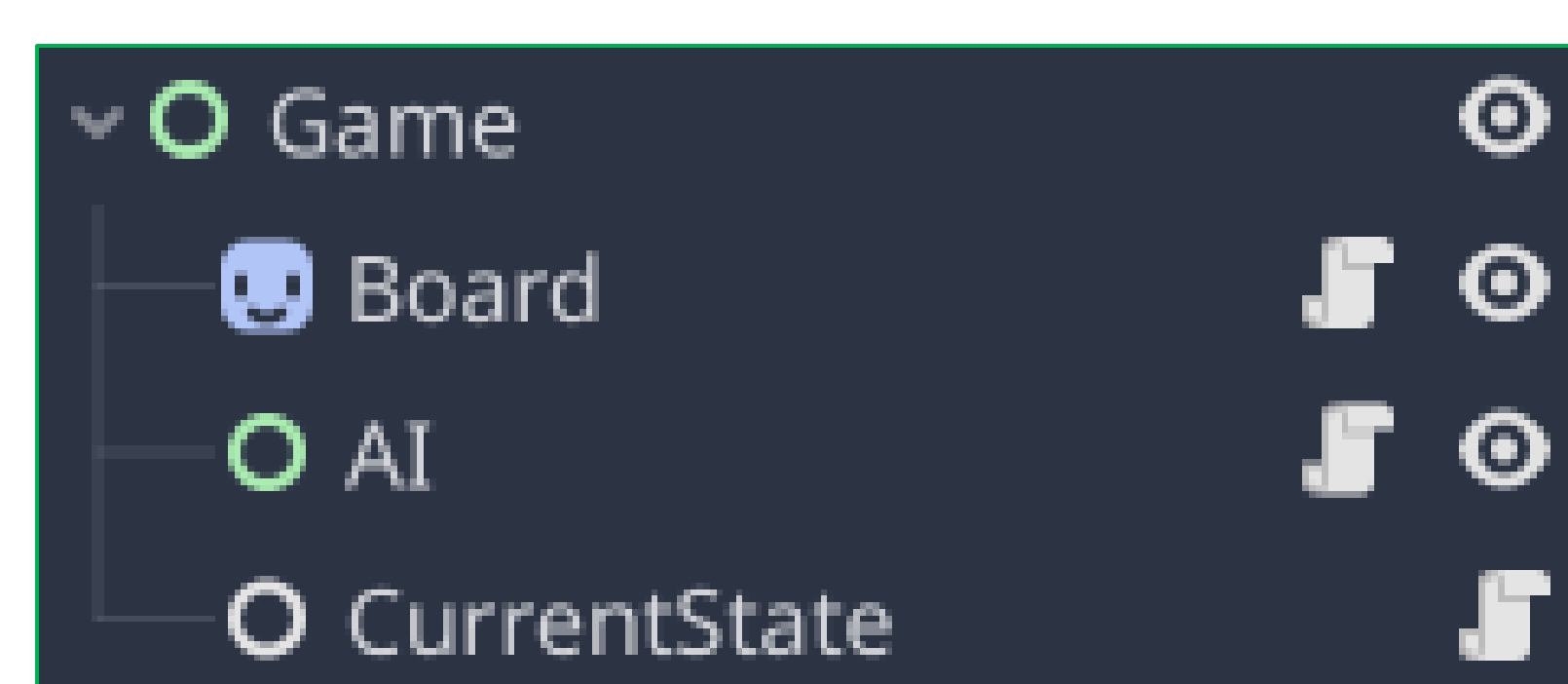
Simulation

The artificial intelligence will simulate random moves for each player for each child node. The number of random moves simulated is determined by the difficulty of the artificial intelligence:
- Beginner: 1 move
- Novice: 15 moves
- Expert: 30 moves

Backtracking

The artificial intelligence goes back up the game tree to update the statistics of the nodes it simulated through and chooses the child node that has the best value.

Release