Jeremy Choo 1602380

CMPUT 355 Final Assignment 4 Report

1. Name of Group: HexKitties

2. All members: Jeremy Choo, 1602380; Youwei Chen ,1591895; Siyuan Liu 1589879; Yuxin Liu 1582040; Jiaxin Wang 1586532

3. Contribution: I contributed to the visualization, algorithm and GitHub. I mainly created the wiki pages for reports, setup the frame, buttons, board of hex, fixed some bugs of the MC algorithm, and merged most of the code into main. Jiaxin did a good job on implementing the main algorithm of the program, Monte Carlo search tree. All the members are active and we setup meetings frequently to ensure communication.

4. Project Code: The links in readme inspire our codes, but we did not copy any codes. In other words, all of the codes are ours. GitHub repo link: link here. Game video link: link here

5. Game Description: HexKitties is an offline-storage, AI-based learning, python-based local hex game, which combined AI player, visualizer, tutor functionality. Description of the game: link here

6. Summary of Accomplishment: We accomplished most of our goals, designing a player (descent AI), visualizer, tutor program. **Our goals are (1)** Design AI player trained using Monte Carlo tree search, virtual connection algorithm in hex, UCT (Upper Confidence bounds applied to Trees), **(2)** able to export and reuse trained data files, **(3)** implement visualization using Python with UI in Pygame, **(4)** maintains friendly, following mode-view-controller (MVC) design principle and **(5)** Achieve good grades in Olympics contest. The only goal we did not achieve is, we cannot participate in the Olympics contest. The most satisfying part is having an AI player and is able to visualize the result of the simulation of MC search tree, for example the winning percentage printed on each cell. The most disappointing part is we did not solve all board sizes (AI player cannot solve all board cases), and we cannot participate in the contest. In the future, we can develop a better algorithm (e.g., deep learning) for the AI player, make a better performance test (find a better player to compete) to analyse our AI player, and implement a better UI interface that allows more options for tutoring, board size and visualization.

7. Data of Performance's Measurements: We also have 3 figures to virtualize our performance (instruction on the GitHub readme) **(1)** Figure for 1 sec thinking time AI player VS random player (play randomly), average from 100 5x5 hex games, Winning probability: 0.97 **(2)** Figure for 10 sec thinking time AI player VS random player (play randomly), average from 10 6x6 hex games, Winning probability: 1.0 **(3)** Figure for 10 sec thinking time AI player VS 10 sec thinking time AI player, average from 10 5x5 hex games Graphs can be found here: link here Feedback: link here

8. Quality of the project: The quality of this project has exceeded my expectation. The strength of the AI player is not the best it can be, but I am surprised how good it is after more than 10 hours of training. The GUI is somewhat satisfying, especially for the part that the percentages are shown, and how messages are printed, as well as the winning path indication. I would say it has reasonable effectiveness and acceptable level of beauty. It might not be the best version of the program; however, we have done a pretty good job in my opinion.

Summary

We have chosen Hex as the game of the project because we think it is one of the most challenging game in this course. Furthermore, we are highly interested on implementing an AI player, as well as a better visualization instead of using the terminal. It is fun to be able to create our own game program that's working with some extend of intelligence.

Individual Report

Mon Nov 09, 2020: Week 1

Total hours spent: 30 hours, for 5 hours of 3 meetings, 5 hours on GitHub, 20 hours coding. In the meetings, we discussed the topic for this project. We are choosing between hex, nim or tic tac toe. After that, we also discuss if we want to make a player, tutor or visualization. If we want to make a visualization, we will use Pygame as the user interface. Moreover, we setup our GitHub repo, readme, product backlog and meetings report. (Report 1, 2, 3 wiki pages on GitHub)

What I did: Meeting auditor (upload meeting recording to GitHub), wiki page documentation (similar application, resources, meetings), create frame, buttons and board of game hex.

Mon Nov 16, 2020: Week 2

Total hours spent: 25 hours, for 3 hours of 2 meetings, 2 hours on GitHub, 20 hours coding. In the meetings, we mainly deal with bugs of the program and did some work distribution. Also, our main AI player algorithm is implemented, Monte Carlo search tree. The UCT (Upper Confidence bounds applied to Trees) is applied, so we need to discuss a reasonable waiting time for program to generate a decent next move. We also work on the training data, and make sure that it is able to store and retrieve the trained data. (Report 4, 5 wiki pages on GitHub)

What I did: Meeting auditor (upload meeting recording to GitHub), rearrange python file and data files, implement Real player mode (AI player mode), new game, exit functionality, fix mode changing bug, fix update display bug by running different thread so that algorithm and display update are separated, fix next starting player bug.

Mon Nov 23, 2020: Week 3

Total hours spent: 25 hours, for 8 hours of 2 meetings, 2 hours on GitHub, 15 hours coding. In the meetings, we work on our report, made our final decision of some of the feature of the program, finished the documentation (comments, function header), completed readme, reports and wiki pages on GitHub, create video and performance tests of the game, and finally find someone to give us some feedback. (Report 6 wiki pages on GitHub)

What I did: Display winning percentage on each cell, added comments and function header for my code, posted performance test figures to readme, record the meetings and upload meeting reports to GitHub wiki page.