User Guide for SE 4G06
An AI-based Approach to Designing Board Games

Team #6, Board Gamers
Ilao Michael, ilaom
Bedi Hargun, bedih
Dang Jeffrey, dangj12
Ada Jonah, karaatan
Mai Tianzheng, mait6

April 6, 2023

1 Game Engine

1.1 Installation

It is required that the user has Python 3.9 installed. Navigate to the src folder and install the Python requirements

```
cd src
```

```
pip install -r requirements.txt
```

1.2 Run

The game engines are in the following folders for An Age Contrived and tic-tac-toe respectively:

```
src/AnAgeContrived/
```

```
src/tictactoe/
```

Navigate to the game you would like to simulate and run the command

```
python main.py
```

The following arguments can be passed to change the simulation configurations

```
--training-num 10
// Will train the AI for 10 simulations

--test-num 5
```

```
// Will test the AI for 5 simulations

--seed 513464
// Seed value for the random functions

--epoch 5
// The number of iterations per simulation

--resume-path log/tictactoe/dqn/policy.pth
// To resume training of a pre-trained AI

--opponent-path log/tictactoe/dqn/policy.pth
// For the opponent players to have a pre-trained decision making
    process instead of a random policy
```

The output after a simulation will be a log file of simulations and a new policy that the AI has learned. The log file will be under:

```
src/AnAgeContrived/ai_history/simulation_history_time_stamp.json
```

It will be named after a timestamp of when the simulation took place.
The Polciy generated will be under

```
log/AnAgeContrived/dqn/policy.pth
```

2 Data Visualizer

2.1 Installation

The data visualizer is in the following folder:

```
src/visualization_v1
```

It is required that the user has the newest Node.JS version installed and npm (node package manager).

Run the following command:

```
npm install
```

2.2 Run

After installation, run the following command to start the visualizer:

```
npm start
```

2.3 Configuration

The visualizer expects two files to be in the following folder:

```
visualization_v1/public/files
```

The files are:

- JSON file containing the simulation data
- JSON file containing all the possible actions

Once the files are in the folder, the following file needs to be updated:

```
visualization_v1/src/data/getFiles.js
```

This file contains an array of objects that has the following structure:

```
{
    game: "An Age Contrived",
    name: "1000_simulations",
    filename: "../files/1000_simulations.json",
    actionFile: "../files/allActions.json",
}
```

Update the game key to name of the game, name key to the name that will be displayed in the dropdown menus in the visualizer, filename key to the relative path to the JSON file containing the simulation data and actionFile key to the relative path to the JSON file containing all the possible actions. The path is relative to the *visualization_v1/public* folder.

In order to display different category types in the Tree Graph - Common Paths component, you must update the following file:

```
visualization_v1/src/components/SimpleTreeGraph.js
```

In this file, *getCategories* function must be updated. This function return an array of objects of the following structure based on the name of the game. Example:

```
if (name === "Tic Tac Toe") {
    return [
      { name: "Start", itemStyle: { color: "green" } },
      { name: "End Game", itemStyle: { color: "red" } },
      { name: "0", itemStyle: { color: "lightgreen" } },
      { name: "1", itemStyle: { color: "aqua" } },
      { name: "2", itemStyle: { color: "gold" } },
      { name: "3", itemStyle: { color: "plum" } },
      { name: "4", itemStyle: { color: "purple" } },
      { name: "5", itemStyle: { color: "lightblue" } },
      { name: "6", itemStyle: { color: "orange" } },
      { name: "7", itemStyle: { color: "lightsteelblue" } },
      { name: "8", itemStyle: { color: "gray" } },
      { name: "9", itemStyle: { color: "paleturquoise" } },
    ];
}
```

The name key corresponds to the name of the category and the itemstyle key contains an object containing the color key that expects an HTML compatible color string. The color names can be found here.

After configuration, simply refresh the app.