AI Snail Game

Group Members:

1. Sardar Moazzam Sajid (BSCS-2019-17)
2. Muhammad Asad Ayyub (BSCS-2019-25)
3. Azeem Akhtar (BSCS-2019-20)
4. Ms. Safia Sultana (BSCS-2019-)

Abstract:

In the modern world game industry has grown rapidly and is now playing a significant role in empowering the economy of countries. With the progress of time people don’t have time for each other and they can’t play game with each other. In solution to this problem computer scientist and game developers have formulated a solution that a person can play game with computers in which the user takes its turn as a result one turn is taken by the computer. So, this is what all the AI game is about.

Acknowledgement:

Thanks to all of my fellows who played their part in making this game. Special thanks to our lab instructor Sir Faisal Zeeshan in helping us developing this game. In addition thanks to our senior Mr. Noor Nabi who helped us in understanding the concepts of the game.

Contents

Introduction:

Now going on the part what is game all about and how it works and how it is being developed.

a. **Old games without AI**:
   In old games without AI there were two situations either two players can play with each other or if one player is playing against computer agent then computer will not try to win it will just take random turn.
b. **Games with AI**:
   Games with AI (artificial intelligence) means that player will play against the computer agent but now it will not take just random chances but it will think just like humans and will take its turn in such a way that it can maximize its winning chances at the same time can reduce the winning chances of the human player. So, in AI game we will make the computer think and after that make a decision and can take a turn.

➢ The Snails Game

Our game is a two-player game in which one turn is of player(human) and other is of the AI agent (computer player). Just like other games e.g., chess, Ludo, tic-tac-toe our game is also board based. Name of game is Snail game. Our game will have two snails; one is of human player and other for AI agent. This game is all about occupying the more blocks on board. One who achieves more blocks at the end is winner. Our game contains the grid which is 10*10 in size means 100 total blocks, and one who occupies more blocks will win in the end.

➢ How to play the game?

- This is a turn based game. There are two players. One Player is Human while the other is computer(AI-AGENT).
- Each player alternates his or her turn.
- To make a turn, player has to click the box of its own choice.
- A player can only make choice of block, which is adjacent to the location of sprite, whether it is empty or contains splash of its own.
- When a player moves to next block, if next block is empty, its score is increased by 1 and the previous location is filled with its splash, which means the area now belongs to this player. If it moves to its previously occupied block, then score will not be increased.
- A player cannot move to the block which is occupied by the opponent.
- Player can only move horizontal and vertical adjacent blocks; it cannot move directly.
- If a player clicks the box outside of board, its turn is wasted and no score is increased.
- When the human makes his choice, the AI-AGENT makes his choice intelligently, and then pass the turn to human, this phenomenon continues till the whole board is filled.

➢ End Result:

- When the board is filled, the scores are counted and the player which occupied more blocks than others, will win in the end.

➢ Literature Review:

Many of the articles and research papers we read aided us in developing efficient algorithms. There are several approaches that have been explored to make the computer think. For example, the minimax algorithm. All conceivable movements were considered, and the finest ones were chosen to make AI-AGENT capable of thinking and difficult to defeat. The algorithm aided the AI-AGENT in making sound judgments. Rather than making random moves, AI-AGENT is taught to think rationally via the use of several strategies.

➢ Methodology:

There are some fundamental functions used to build this game, which will be discussed in detail below.

❖ **Game_Loop:**

The GAME starts with an initialize grid which makes a backend grid and then a frontend board is made using Arcade. Evaluate board will evaluate grid after every move, if it is fully occupied then the game will over, otherwise continue. Human and AI-AGENT play alternatively until final state is reached. Then the winner is decided on the basis of number of boxes occupied.

▪ **Initialize Board ():**

First of all, a front-end board is developed using Python's Arcade library. It is built to facilitate the players and provide an easy to use game board. A 10*10 board, which contains 100 boxes is built using Python Arcade library.

- **Setting up Sprites:**
  Then the sprites for both players are set on the opposite corners of the board.

- **Setting a background grid:**
  A 10*10 2d array background grid is established at back-end which connected the frontend with backend of the game. All the scores are stored and retrieved using this backend grid.

- **Is Legal Move ():**
  This is a function which is called when a player makes a turn. This function is a Boolean function which tells about a move whether a move is legal or not. If it is legal then we will consider the turn to be legal and update the grid and scores otherwise consider it as illegal and pass the turn to the other player.

- **Evaluate Board ():**
  This is a function which inspects the board at the given situation. It returns the current situation of the grid, whether a player is winning or losing or game is drawn. It returns the state of the board. If human is winning then it returns 10, if AI-AGENT is winning it will return 20, if match is drawn it returns 1020 and if the match is in continuous state then it will return 0.

- **Splashed ():**
  This function is used when a player clicks on its own trail/splash. It will check the next empty box in that direction of trail and place the sprite before it. The function is used to make slippery surface.

- **Minimax Algorithm:**
  The main algorithm used to make AI-AGENT think is minimax algorithm. It is a recursive algorithm which uses backtracking technique to make a decision. In this technique, both the players strive to maximize their winning chances and minimize opponent's w inning chances. Max player will try to maximize its winning chances and the Min player will minimize

the Max’s winning chances. So, it is an excellent strategy based decision making algorithm. It uses breadth first search to explore all the leaf nodes and calculates winning chances of each path using heuristic function.

We pass a grid, the current depth and the turn of player to minimax function.

- **Heuristic Function:**
  This function calculates the winning chances of a player at a current location. This function is used within minimax algorithm to find optimal move.

- **Find Best Move ():**
  This function returns the best move for AI-AGENT after analyzing all the possible turns in depth 7. By using minimax and heuristic function, it returns the best optimal solution for the AI-AGENT.

- **Game User Interface**
  The Arcade Drawing Library and Python's OOP classes are combined to create a user-friendly gaming interface. There are multiple screens which appear in sequentially,
  - Main View
  - Instruction Screen
  - Game Play
    - Grid view:

      A board which consists of boxes.

        - Scores:

          The current scores of both the players are shown on game view screen.

        - Sprites/splashes:

          The sprites and respective splashes of both the players are shown within a board in game view screen.

    - Game Over

- **Mouse-Touchpad Click for turn:**

  To make a turn, a player simply has to click in a desired legal box using mouse or touchpad.

- **Testing and Analysis:**

- **Conclusion:**

It is an excellent method for creating artificially intelligent games. The numerous strategies assisted the AI-AGENT in thinking like a human, taking into account all of the possible outcomes of the current state. The AI-AGENT is as capable as a human gamer. One limitation which we observed is the slowing down of game when we use greater depth. As a result, we must make advantage of the restricted depth, else the procedure would be extremely sluggish. As a result, in larger games such as chess, this technique will be ineffective.

References

1. https://api.arcade.academy/en/latest/examples/view_instructions_and_game_over.html#view-instructions-and-game-over
2. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
3. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-2-evaluation-function/
4. https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-3-tic-tac-toe-ai-finding-optimal-move/