The first version of the game is cleaner, and easy to interpret than the second, suggesting it is designed for adults and teenagers. I will be aiming to make my game clean in this way, since it removes distractions and appears more user friendly for adults than version 2.

Both Ultimate TTT and Strategic TTT allow the user to play with AI and co-op multiplayer, however Ultimate TTT allows the user to also play online using a sharable web address. I will not be using web address since my game is not online.

The AI in Strategic TTT has no modes of difficulty and is far easier to beat than Ultimate TTT, since it is designed for adults. Ultimate TTT has 8 modes of difficulty, my game will have at least 3: Easy, Medium, Hard, with the addition of an Impossible mode if time allows.

Both games highlight the local grid for the player to play in next. This avoids confusion and gives clear directions as to what is fixed and what is optional in the game. I will be including a form of highlighting similar to this in my game.

Ultimate TTT also has a tab containing the history of all the moves that have taken place, which each player is able to see. I think this is unnecessary since all the moves are present on the global grid as symbols very clearly. I think the confusion occurs when the current player is unsure where the opponent placed his last symbol, and to resolve this, I will highlight both the last move made, as well as the local grid/s the player is able to make moves in.

Both existing systems incorporate some form of clear menu, to allow players to choose between the types of modes they would like to play.

Both systems include a tutorial in the main menu, which gives clarity as to the rules of the game and how to play it.

In the online games of TTT which I have researched, there is one particular variation with the rules addressing what happens when a player is directed to a local grid that has already been won, or one that is already full of player symbols.

In these 2 games, when a local grid is won, it is replaced by the symbol of the winning player, and whenever a player is directed to this local grid, they can play in any local grid.

While in other instances, if there are any free spaces within the local grid, the player must play in the directed local grid, regardless if it has been won or not.

The second rule is susceptible to a winning strategy in which second player is directed to the centre local grid, even after it has been won, allowing the first player to claim the majority of the rest of the grids:

Ultimate Tic Tac Toe is significantly more complex than other variations of Tic Tac Toe.

This is largely because the moves the player makes must also take into account the global board in several ways:

Anticipating opponent’s move/blocking:

It is difficult to choose preference of the smaller grids or the larger tables through the course of the game, in some circumstances the smaller grid is more significant whereas in others the larger table is far more significant.

Planning:

It is far more difficult to visualize the opponent’s responsive moves due to this second dimension that must be considered, making opponents more difficult to predict.

Winning:

It is difficult to lose focus of the global game due to the appeal of a particular local board. In actuality the win or lose of a local board is not valuable if it is not used to win the global board.

Overall, the balancing of the local and global boards and the difficulty by which the opponents move can be predicted makes this game very complicated and difficult for humans to play.

I have found a few different implementations of AI for Ultimate Tic Tac Toe.

In my research on what algorithm to use to implement the AI in Ultimate Tic Tac Toe, I found that the two most suited to my project are the minimax algorithm and the Monte Carlo Tree Search Algorithm.

Minimax: (+alpha-beta pruning)

The minimax algorithm makes use of a heuristic evaluation function to determine how good a game state is for a player.

The current game state is evaluated for all possible moves, generating a tree of possible moves.

Each leaf in this new tree is then evaluated in the same way, recursively, until the maximum depth is reached. Once maximum depth is reached, the leaf nodes are evaluated using the heuristic evaluation function to assign them a value.

For each row of nodes in the tree, on the AI player’s turn, the maximum value is chosen, while on the human player’s turn the minimum value is chosen. Maximum = good for AI, minimum = bad for AI.

Game Tree

Use of AI algorithms:

Minimax and Monte Carlo

Which is better?

How to implement

Include other AI examples

Objectives

All of them

Note:

Can use non- source code projects to analyse features

Use pure AI and an example to analyse algorithm, etc…

Include both Minimax and Monte Carlo Tree Search, compare both and analyse the advantages and disadvantages of both.

“I will decide on the design stage which to use”.

Websites:

<https://www.youtube.com/watch?v=l-hh51ncgDI>

<https://www.google.com/search?q=minimax+with+alpha+beta+pruning&rlz=1C1GCEA_enGB816GB843&oq=minimax+with+alp&aqs=chrome.0.0j69i57j0l4.3302j0j7&sourceid=chrome&ie=UTF-8>

<https://github.com/pinkeshbadjatiya/ultimate-tic-tac-toe>

<https://github.com/noahnu/ai-ultimate-tic-tac-toe>

<https://github.com/mck-/T3>

<https://www.google.com/search?hl=en&ei=aW6sXJO_JvuE1fAPoMKD0AQ&q=static%2Fheuristic+evaluation+function+tic+tac+toe&oq=static%2Fheuristic+evaluation+function+tic+tac+toe&gs_l=psy-ab.3...14050.29689..31724...7.0..0.110.1614.21j2......0....1j2..gws-wiz.......0i71j0i7i30j0i30j0i13i30j33i10.zM9ZX1GyX5A>

Objectives must be numbered not bullet pointed.

Introduction

Problem

End User

Analysis of existing systems and features/ algorithmic features use or not use