Game AI for a 3D Tennis Game based on Monte-Carlo tree search

Kaito Kimura s1250131 Supervised by Prof. Maxim Mozgovoy

Abstract

AI is now being used in many different areas of video games. In particular, character AI has become an essential part of video game opponents and greatly helps to make games more interesting. In this paper, Monte-Carlo tree search (MCTS) was used to build an AI for a 3D tennis game. By using Monte-Carlo tree search, it is expected that an AI can be built that does not rely on predefined behavioral patterns and can respond to a variety of situations. As a result, I was able to build an AI agent with rational behavior based on MCTS. The MCTS based AI played against the game's built-in AI and won 7 out of 10 matches, with a score of 59-44.

1 Introduction

There are several methods for building Game AI. The most popular one is Rule-Base AI. In Rule-Base AI, rules which are pairs of condition and behavior are predefined, and the AI acts according to these rules. For example, when the opponent's position is in the left corner, AI hits the ball to right corner. Rule-base AI is a simple and very powerful approach, but it has two weaknesses. One is that the person who built the AI needs to deeply understand the patterns of strong and weak acts of the players. The other is that it can be very weak in patterns that the implementer did not anticipate.

These weaknesses can be overcome by using heuristic algorithms to select sequentially plausible actions. In this study, AI for a 3D tennis game is built by using MCTS which is a kind of heuristic algorithm [1]. MCTS is a method that combines tree search and Monte-Carlo methods. The state of the game is represented by the nodes of the game tree, the result of each node's action is simulated, and the AI selects the most appropriate node.

The game being targeted is a complex 3D tennis game, so it needs to be properly modeled and applied.

2 Methodology

The research builds AI based on the World of Tennis (Fig 1). Prior research on game AI for this game includes examples of rule-based AI and case-based reasoning AI applications [2, 3].

Fig 1. World of Tennis

Each player can take the following actions:

- Making a serve
- Returning the shot
- Recovery movement

In serving or returning a shot, the player can choose a target point. There are also multiple choices for the type of shot. Each player can use normal shots, lob shots, slices, drop shots, etc. The recovery movement allows us to move our character after hitting the ball in order to effectively defend or attack in subsequent movements. Basically, the AI can also perform these equivalent actions.

2.1 Monte-Carlo tree search

I used the Monte-Carlo tree search method to build the AI for the tennis game. In MCTS, the state of the game is represented by a game tree. It is an algorithm that finds effective moves by exploring the game tree. MCTS consists of the following four steps:

1. Selection
2. Expansion
3. Simulation
4. Back Propagation

By repeating these steps, the search proceeds like Fig.2.

The root node is the current game state. From there, valid moves are explored. Each node has a record of the number of all attempts and scored attempts, so the

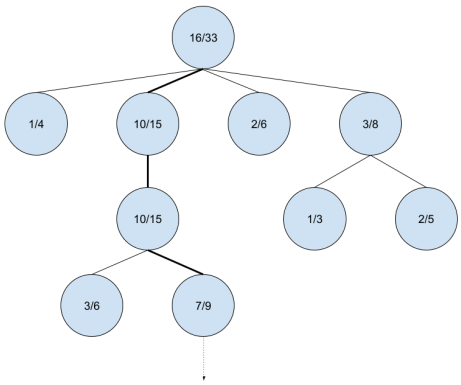

Fig 2. MCTS process.

node with the highest percentage is considered the most effective.

In the selection step, the UCT (Upper Confidence Trees) value is commonly used [1]. AI selects a node that has the highest UCT value from the root node to the leaf node.

$$UCT = X_i + 2C_p\sqrt{\frac{2lnN_i}{n_i}} \quad (1)$$

The first term X_i is reward team.

$$X_i = p_i + (1 - p_i)\frac{w_i}{n_i} \quad (2)$$

The percentage of points scored in the simulation, w_i/n_i, is the sum of the results of the leaf nodes derived from the node. Therefore, it is a measure of how likely it is to score points in the destination from that node. In general, w_i/n_i is used as the reward term. On the other hand, in the game of tennis, every shot has the potential to score directly, and some patterns have very high probability. For example, the probability of scoring on a lob shot when the opponent is at zero distance from the net would be almost 100%. However, since the simulation is done from leaf nodes, the resulting percentage will not be near 100% and will place a gap. For that reason, we adopt Equation (2) as the reward term.

p_i is the probability that the node will score directly, w_i is t is the number of points wins in the simulation and n_i is the number of simulations performed.

The subsequent terms are for biasing the search towards nodes that have not explored enough. A high value is calculated when the number of child nodes n_i is not enough compared to the number of simulations for parent node N_i. C_p is a constant that specifies how strong the bias toward unexplored nodes should be. If there is not enough search for valid nodes, the node with the less search is done, and if it is searched enough, the node with the highest win rate is searched in priority.

In the expansion step, the game tree is expanded by adding child nodes if a leaf node has been explored beyond a certain threshold. In the simulation step, the result of the game from the leaf nodes is simulated. The result of the match is reflected in all the nodes selected in the backpropagation step.

By repeating these steps, the percentage of scoring is calculated at each node, and the AI judges the node with the highest $\boldsymbol{X_i}$ value of reward as the valid move and acts on it.

2.2 Application to the tennis game

Video game looks like a complex game, but actually it's simple. All a player must do is choose a target point to return a shot, and to select a type of shot. The flow of the game can be represented as a repetition of each player's action. In other words, the flow of the game can be represented by a game tree. In this case, an AI is built such that each node has the following information:

- Shot from: AI or enemy
- AI position
- Opponent position
- Target position

Each node has three coordinates: the AI's coordinates, the opponent coordinates, and the target point's coordinates. All of which are indexed by dividing the court into 48 parts as shown in Fig. 3.

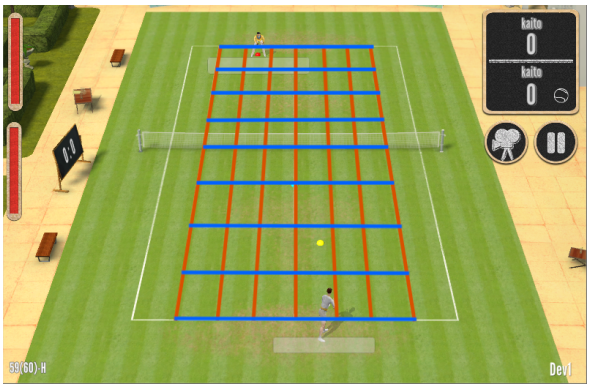

Fig 3. Separated court.

The reason for this is that if nodes were built for each coordinate, there would be too many valid nodes to search properly. This allows the application of MCTS, since it can be represented by a game tree.

Fig. 4 shows an example of a game tree for a tennis game.

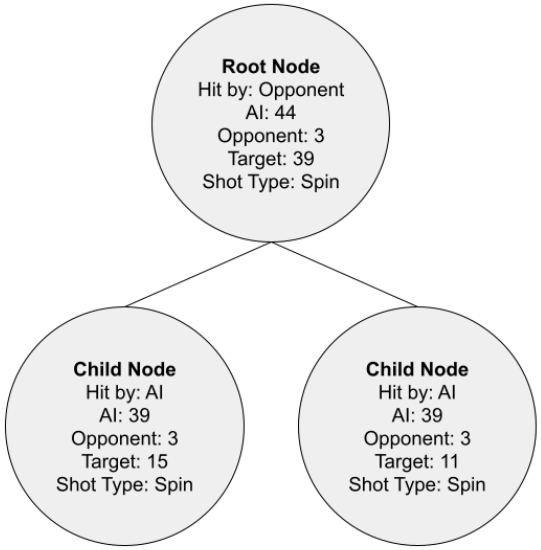

Fig 4. Node state for MCTS.

The opponent is the player at the top of the screen in positions 0 to 23, and the AI is in positions 24 to 47. The root node represents the current state of the game, where the opponent in position 3 is hitting the ball towards position 39. The following child nodes are the effective moves at the next point in time, i.e. when the AI returns the ball. Both AIs are moving to position 39, and there is a separate node for each of the next return positions. Only two nodes are shown in Fig.4, but in actuality, there are 48 nodes for 24 different return positions, one for spin shots and one for lob shots.
In this way, the state of the tennis game is appropriately represented in the game tree.

2.3 Simulation Logic

In MCTS, the AI simulates the game with a random selection from leaf nodes in the simulation step. In each simulation, the AI needs to judge whether the game will eventually result in a score or a loss of points. The accuracy of these simulations is very important in creating a strong AI, because the AI determines which action is effective based on this scoring rate.

The difficulty is that there is no clear scoring/losing rule in a tennis game. From information about each player's position, target point and the shot type in a node, we need to create the logic to properly determine if the shot is a scoring shot or not. In this research, more than 1400 scoring patterns from statistical data of 10,000 actual games were extracted. Each match data is about 2-3 minutes long and contains about 28 ball hitting events. In our game data, the coordinates of each player, the coordinates of the ball, and the ball type are stored as time-series data. From this match data, we extract the following information, which is in the same format as the game tree plus the Boolean value of whether the shot is a scoring shot or not.

- Attacker's position
- Defender's position
- Target position
- Shot type
- Scoring shot or not

Then, for each pattern, the possibility of scoring, the number of scored patterns divided by total patterns is calculated. For example, when the opponent is at position 33 and the ball is hit with a lob shot from position 16 to position 44, there is an 80% chance of scoring. (Fig. 5).

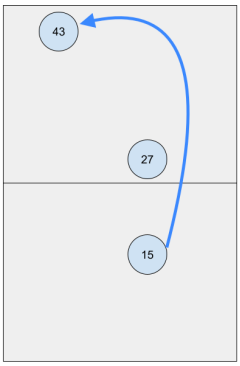

Fig 5. Example of shot pattern.

In the simulation step of MCTS, the decision that the shot is scoring or not is made with the same scoring probability as this extracted shot pattern.
In addition, this scoring probability per pattern is also used in p_i in Equation (2) which represents the value of the reward.

3 Result

The game system has its own AI called “Coach AI”. The performance of the AI can be determined by playing multiple matches against the game's AI and recording the winning percentage.

Table 1 shows the results of the game against the coach AI. 10 games were played and 7 games were won. The total score 59-44. The scores of the matches show that the AI is able to take rational actions.

On the other hand, it cannot be said to be a strong AI because the coach AI is a simple AI.

I consider the following three points as the cause of this.

(1) The influence of nodes is not deep enough.
(2) The problem of AI movement
(3) Simulation accuracy

(1)
The first is that the influence of nodes is not deep enough.

MCTS is an algorithm that acts on the indicator of how much AI will win in the end if AI act on that node, but in a tennis game, the relationship between choosing that node and eventually scoring is not that strong.

In basic rallies, there was not much relationship, and the win rate in the simulations for most of the nodes converged around 50%. Only MCTS is strongly effective is when the act of the node is related to two or three moves ahead.

(2)
Second, there is the issue of the AI's movement.
Connected to the first problem, if MCTS works effectively, it is in patterns where the choice of that node results in a score/failure two or three moves ahead.

Since the opponent will come to prevent a clear scoring pattern from the AI's current position, the strength of MCTS-based AI should be to score points by taking actions with a high scoring probability pattern found using simulation from among many options including the range of movement.

However, our AI did not generate valid child nodes for the AI to move to and act from the parent node. For example, if the opponent hits the ball at position 10, a valid child node would be for the agent to move to position 10 and hit any position with any shot.

Thus, the agent can act to prevent a scoring pattern a few moves ahead, but it is not able to move to a scoring pattern that the opponent has not identified and go for the score. The fact that MCTS-based AI scores fewer points from netplay in Table. 1 also shows this.

This is a problem with game tree modeling and applying MCTS to AI behavior, so there is room for improvement.

(3)
Third, there is the issue of simulation accuracy. In this AI, AI determined whether a shot was a scoring shot or not by extracting the scoring patterns from 10000 match data. As mentioned in the section on simulation logic, this accuracy is very important.

However, since the number of patterns is 27648 = 24 * 24 * 24 * 2, the number of data for each pattern is about 3 in both average and median. The number of data for each pattern is about 3 for both the mean and median. For some frequent patterns, the number of data is in the tens of patterns, so they are reliable, but for many patterns, the accuracy of the simulation is low.

4 Conclusion

In this paper, we have created an AI for a 3D tennis game using a heuristic algorithm, Monte Carlo tree search. By properly modeling the complex game state, the AI can represent the game state in a game tree and perform the search. MCTS learns whether a move at a node is valid or not by repeatedly simulating the game state from the node. Therefore, the accuracy of the simulation is important. In this study, we conducted simulations by extracting patterns that are likely to be scored from actual game data. As a result, we were able to simulate with a certain level of accuracy and construct an effective agent.

As for future works, there are multiple areas for improvement the AI.

One is the modeling of the game state. We believe that more detailed modeling, including the player's movement, ball type, etc., will allow us to build a more flexible AI.

The second is to improve the accuracy of the simulation logic. In this paper, we extracted probabilities from scoring patterns, but there was a problem that the simulation accuracy was not sufficient except for highly frequent patterns. We believe that better accuracy can be expected by increasing the number of match data from which scoring patterns are extracted about ten times.

Finally, there is room to improve the efficiency of the search time. We need to iterate through at least 2000 simulations to search for valid nodes, which takes 2-3 seconds of time to search. Since this is a real-time game, we will need to improve the efficiency and drop the search time to at least 500 microseconds, and also improve the search to be asynchronous.

References

[1] Browne, Cameron B., et al. "A survey of monte carlo tree search methods." *IEEE Transactions on Computational Intelligence and AI in games* 4.1 (2012): 1-43.

[2] M. Mozgovoy. Context-Awareness and Anticipation in a Tennis Video Game AI System. Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics, Miyazaki, Japan, 2018.

[3] Shinsei Kinouchi, “Identifying Key Elements of Successful Behavior in a Video Game of Tennis”

Table 1. Result of match vs Coach AI

Match Number	1	2	3	4	5	6	7	8	9	10
Score MCTS AI-Coach AI	7-3	7-3	3-7	7-4	7-5	7-3	7-2	7-3	2-7	5-7
Net Point	1	0	1	0	0	0	1	0	0	0
MCTS AI’s Miss Shot	0	0	2	0	1	1	1	0	2	2