Context-Awareness and Anticipation in a Tennis Video Game AI System

Maxim Mozgovoy
School of Computer Science and Engineering
The University of Aizu
Aizu-Wakamatsu, Japan
mozgovoy@u-aizu.ac.jp

***Abstract*—This paper is dedicated to the problem of using case-based reasoning AI in a commercial mobile game of lawn tennis. We discuss the unavoidable manual game analysis stage, aimed to represent user intentions accurately and supply them to the machine learning procedure. We show how the right combination of machine learning and manual effort helps to construct a solid game AI system, able to play in human-like manner. Our experience shows that the key factor of the successful decision making and reasonable resource consumption in mobile tennis is careful representation of context-aware behavior and anticipation of opponents' actions, exhibited by real players.**

Keywords—case-based reasoning, game AI, mobile tennis

I. INTRODUCTION

Advanced decision making algorithms are rarely used in the world of commercial game development. As Millington and Funge summarize, *"Most games use very simple decision making systems: state machines and decision trees. Rule-based systems are rarer, but important. In recent years a lot of interest has been shown in more sophisticated decision making tools, such as fuzzy logic and neural networks. However, developers haven't been in a rush to embrace these technologies. It can be hard to get them working right."* [1]. Indeed, one needs a special reason for using any complex approach in a commercial game, if a simple system works reasonably well. Few games, however, have such reasons. One of the most cited example is Lionhead's *Black & White* that integrates machine learning into core gameplay [2].

Making AI one of the cornerstones of the basic gameplay process was also an early design decisions in a mobile tennis game *World of Tennis: Roaring '20s*, developed with active participation of the author of this paper. In brief, the principal goal we were trying to achieve was to design a multiplayer mobile tennis game with *as little hindrance to the game process as possible*. Virtually all existing multiplayer tennis games suffer from *connection lags* and *complicated matchmaking*. Tennis is a fast-paced game, so even minor connection drops might cause annoyances during the matches. In addition, people often play mobile games when commuting or in public spaces, where internet connection might be unstable. Next, the players have to compete against the opponents who are currently online, have acceptable data roundtrip time, and possess comparable ranks. In practice, it means that often very few people would satisfy these criteria, and thus even initiating a single match might be a problem.

To remedy the situation, we decided to use machine learning. By playing the game, people *train* their virtual characters (avatars) that can later substitute them in tennis matches. In other words, people in *World of Tennis: Roaring '20s* compete with machine learning-based AI agents rather than with real online opponents. (Similar reasoning lies behind Drivatar AI in the Forza Motorsport series [3]).

This approach can be considered feasible, however, only if the used algorithm is reasonably good. In our case, it means that the virtual characters should: 1) exhibit a wide range of diverse play styles and behavior patterns to appear human-like; and 2) play on par with human opponents to provide sufficient challenge for skillful players.

Our resulting system uses a combination of case-based reasoning decision making with Markov chain-like database of human actions. In a nutshell, agent knowledge is represented as a graph, having individual game situations as vertices, and actions as weighted edges. This way, it represents the fact that a certain action switched the game from situation A to situation B during the learning phase. Decision making algorithm tries to find the best match for the current game situation, and acts accordingly [4, 5]. Unsurprisingly, most efforts were dedicated to the *retrieval* and *reusal* steps of case-based reasoning process [6]. Since the occurrence of identical situations in tennis is unlikely, we had to develop a simplified system of features, reflecting only the most important game parameters, and design an approximate matching algorithm that finds the best-matching situations when the perfect match is not available. Next, we had to decide whether the action we found is still applicable in the current situation, and discard or adapt it if necessary.

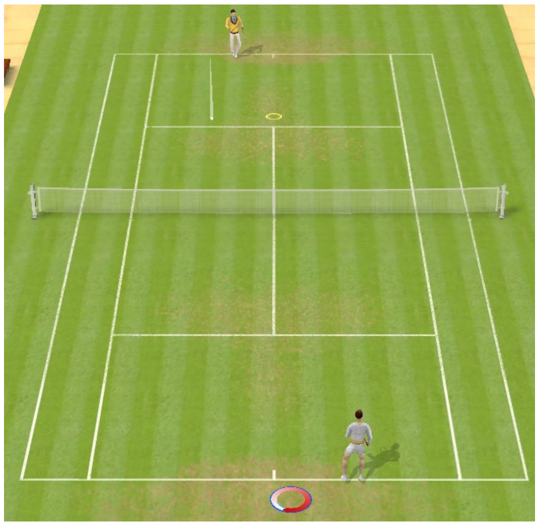
However, it turned out that successful learning was impossible without revision of the basic learning process that had to take into account *context-awareness* and *anticipation*, apparently exhibited by human players. This forced us to rethink what constitutes the *input* of the machine learning algorithm. While this topic can be considered technical and problem-specific, and thus rarely discussed in literature, we believe that our case study can provide insights into the process of learning user actions in a real mobile game. This step is often overlooked as trivial, but we show that even in case of a relatively simple tennis game a certain effort is required to transform user *intentions* into virtual characters' knowledge.

II. Basic Game Mechanics

Before discussing the problems we faced, it is necessary to get acquainted with the basics of the game mechanics of *World of Tennis: Roaring '20s*. We will consider a simplified game description that includes only the elements, relevant for the subsequent discussion. We will use the word *player* to refer to the actual person playing the game, and reserve the word *character* for the in-game player avatar.

World of Tennis: Roaring '20s is a mobile version of a conventional lawn tennis game. The player always controls the bottom character, while the top character is controlled by the AI system (see Fig. 1).

Fig. 1. Gameplay of *World of Tennis: Roaring '20s*



The game process can be separated into four phases (as seen by a human player):

Serve. The serving character is located in one of the serve areas (A or B, shown in Fig. 2). By tapping the screen within the current serve area, the player moves own character to the left or to the right along the baseline. A tap within the corresponding target area (D' for A or E' for B) initiates the serve.

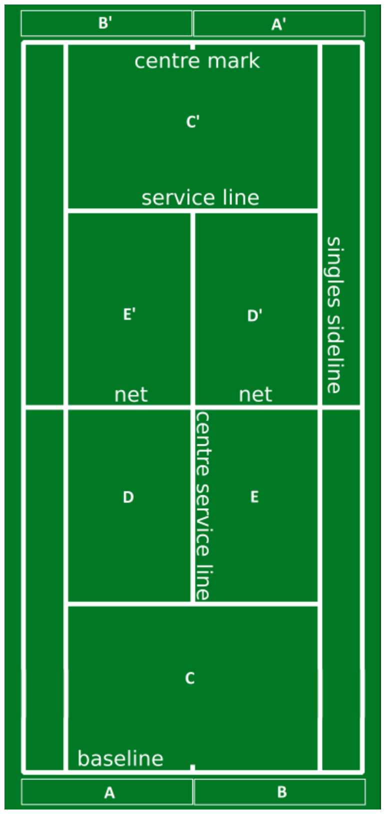
Recovery movement. When the ball is moving away from the player's character, the player can tap anywhere on the own half of the court to initiate a *recovery movement* and occupy a reasonable spot on the court before the opponent's shot.

Returning a serve or a shot. Once the ball is shot towards the player, the system automatically starts steering the character to the optimal ball receive location. At this phase, the player can tap anywhere on the opponent's half of the court to set the target point for the next shot. If the character is able to receive the ball, the shot will be performed automatically. The accuracy of the shot depends on the character's skill values and shot timing: the game encourages the player to set shot target points early, as they have higher accuracy.

Preparing to receive a serve. The receiving character is located in the area across the serving player: A for A' or B for B'. The player can move the character inside this area by tapping the screen (as in the *Serve* phase), until a serve is initiated.

The general motivation of these design decisions is to facilitate tactical rather than pure arcade gameplay. The player is encouraged to select winning shot and recovery target points, while the game engine takes care of the rest. There is even a capability to turn on the "AI control mode" instead of manual control, so players might use this "autopilot" feature when facing particularly weak opponents.

Fig. 2. Tennis Court Diagram



III. Learning and Decision Points

The cornerstone concept of case-based reasoning is a (stored) *case*, i.e., *"a previously experienced situation, which has been captured and learned in such way that it can be reused in the solving of future problems"* [6]. This notion is typically illustrated in literature with real-life examples, where certain static situations are addressed with particular expert decisions, thus forming "situation/decision" pairs.

However, in the world of an action game the logic of pairing decisions with the corresponding situations becomes more complex. Simply speaking, *we do not know* which situation triggers a certain user action. We know *the current onscreen situation* when a certain user input is detected, but obviously, the real trigger has occurred earlier, since user reaction is not instant.

Simply taking into account typical reaction times, known from empirical research, is not sufficient either. In *World of Tennis: Roaring '20s* there is usually no reason to act as quickly as possible. In most cases, the players have a chance to think a bit longer before deciding the next shot or move. It is also often possible to *rethink* the decision, i.e. to tap the screen several times within the same game phase.

This observation leads to a related problem of decision making: what are the appropriate moments of time for an AI-controlled character to decide its actions? In theory, the AI system can override its previous decision on each subsequent frame, though this approach will most probably cause jittery behavior and will greatly affect game performance (on most modern smartphones the game runs at 60 frames per second). The extreme opposite choice, i.e. making decisions early and never overriding them will make AI unresponsive to the actions of human players.

In the subsequent discussion, we will use the term *learning point* to refer to a moment of time when the AI system is given an observed (situation, action) pair to be recorded. The total amount of learning points corresponds to the number of cases processed by the machine learning algorithm. We will use the term *decision point* to refer to a moment of time when the AI system has to perform an individual act of decision making.

From our experience, the task of choosing appropriate learning and decision points is not easy, and requires good understanding of a particular game and its specifics. For relatively simple games, such as classic Atari arcades, one may adopt the "act on each *k*-th frame" approach [7]. However, this method is hardly applicable in modern commercial game projects (at least, due to performance considerations).

IV. Learning and Acting in a Mobile Tennis Game

The present design of the learning and acting algorithm is a result of the whole series of iterations, reflecting our understanding of the game process. Learning and acting is handled differently in different game phases (the algorithm used to switch the phases is shown in Fig. 3).

A. *Serve phase*

During serve, the player first moves to the chosen serve position, and then shots the ball to the target location on the opponent's side of the court. To reduce possible pre-serve walking, we impose a time limit on serve actions, which is consistent with ATP rules of real tennis. When the player initiates a serve, the AI learns this action with *no associated game situation*. One may argue that the serving player must take into account the position of the receiver. However, this logic leads to undesired behavior: the receiver reacts to the change of the position of the serving player, and the serving player adjusts own position in response to the receiver's move. We decided that the AI system should presume that the receiver can always adapt to the serving player's position, and thus the best strategy is to serve without looking at the opponent. In addition, we discard all player movements made before a serve. In other words, we do not treat pre-serve walking as an essential part of the game, and learn only the last movement to the final serve location.

Learning point: as soon as the serve target is known.

Situation: none

Action: move to a specified serve position; serve to a specified target point on the opponent side of the court.

Decision point: in the beginning of the Serve phase.

B. *Recovery phase*

During this phase, a player can freely move own character on the bottom half of the court, so in theory this trajectory can be quite complex. In practice, however, the task of the player is to occupy a certain winning spot on the court to maximize own chances of returning the opponent's shot. At the same time, the opponent is running towards the ball landing area, known to the player, so the player can predict the opponent's movements already in the beginning of Recovery phase. Therefore, we decided to keep only the *last* movement of the player, discarding all earlier actions. This decision is still being debated, and might be revised in the future. While erratic recovery movements are probably not a part of the normal game process, in theory they can be used as a stratagem to confuse the opponent. In addition, their absence can be treated by the players as a sign of artificial, robot-like behavior.

While working on this phase, we also had to discuss a simple but important case of learning user *intentions* rather than actions. When a player skips recovery (does not tap anywhere), should we treat it as "do not move anywhere" or "move to the point where the character is currently located"? The answer is crucial for decision making. If a situation A is considered similar to another situation B, then we should be able to apply in B the same action as in A. If the player did nothing during Recovery phase in A, what should we do if the player character in B is located not in the same spot as in A? One option would be to keep the character still (and thus treat the absence of action as "do nothing"), but we can also move the character to the location recorded in A. (The third option would be to treat the "no action" situation as a player error and do not learn it at all).

Our experiments with the game and the analysis of actual play styles of our beta testers showed that the in most cases the desired action was to move the character rather than to keep it still. The cases of user errors were also rare: if a player does not specify the next recovery point, it usually means that the current location should be treated as a desirable target for the recovery movement.

Learning point: as soon as the opponent hits the ball (end of the Recovery phase).

Situation: includes the coordinates of both characters and the ball target, taken at the beginning of the Recovery phase.

Action: move to the last specified position.

Decision point: at the beginning of the Recovery phase.

C. *Return a serve/shot phase*

This phase is perhaps the most important in the game, and should be examined carefully, since *making the right shots* is the essence of any tennis game. In our case, players face a choice: on one hand, the game engine encourages early shot actions, as they provide more accuracy; on the other hand, one might prefer to wait a bit longer and to react to the opponent's recovery movement. In the present version of the system, we decided to learn two actions during this phase. The first action corresponds to an early decision, and is paired with the game situation that lacks information about the opponent movement direction. The second action corresponds to a decision made when the ball is crossing the net, and includes opponent direction value ("towards the ball target", "from the ball target", "no movement relative to the ball target").

In the decision making mode, the AI always sets the ball target early in the beginning of the phase. When the ball crosses the net, it requests the second decision and overrides the first decision if the new decision is ranked higher according to the situation similarity value, i.e. when the new decision is considered closer to the play style of the human player.

Note that in this phase the user can also skip a turn, giving no commands to own character. Unlike "implicit recovery movements", however, there are no "implicit shots", so we treat this situation as a user error, and do not learn anything in these cases.

Learning point: as soon as the player hits the ball.

Situation 1: includes the coordinates of both characters, and the ball target point, taken at the beginning of the phase.

Situation 2: includes all the features of Situation 1, and the direction of the opponent character, taken when the ball was crossing the net.

Action: shot the ball to the specified target point.

Decision point 1: at the very beginning of the phase.

Decision point 1: when the ball is crossing the net.

D. Prepare to receive a serve phase

While the serving player is moving to the desired serve location, the receiver should occupy a convenient spot on the court to receive a serve. There are three cases when we allow the receiver to change its location: 1) at the beginning of the phase; 2) when the serving character starts moving; 3) at the end of the phase, when a serve is initiated. In practice, we impose an additional minimum move distance in (2) to prevent quick response of the receiver to every slight motion of the serving player, which looks unnatural. We also decided to let the learning algorithm know the target location of the serving player's movement. While this location is not visible to human players (they merely see the direction of the move), we treated it as an acceptable level of cheating for the AI: it merely reduces the number of decision points in this phase, without providing any real advantages.

Learning point: when the opponent's serve is initiated.

Situation: includes the coordinates of both characters, and the target point of the opponent's movement, taken during the last opponent's movement before the serve.

Action: move to the last specified position.

Decision point 1: at the beginning of the phase.

Decision point 2: when the serving player starts moving.

Decision point 3: when a serve is initiated.

V. Discussion

The previous section clearly shows that adopting machine learning-based AI even for a relatively simple game of one-versus-one lawn tennis is not a straightforward task. Here we skip the discussion of the algorithm itself and concentrate only on defining input and output data, and the principles of choosing learning and decision points. This seemingly technical procedure, often overlooked in literature, has grown into a significant module, fundamental to proper AI functioning. This means that there is a notable entry threshold for using machine learning in real games, which reduces its attractiveness as a method of minimizing manual work.

On the positive side, the right combination of manual game analysis and machine learning technology, and a careful choice of learning and decision points is able to produce very satisfactory results. Our main goal was to implement a human-like AI system, able to substitute real online opponents *transparently*, and we believe that some users do not even realize that they play against AI. The game is currently in the soft-launch phase, and is being played by around one thousand users daily. We regularly get players' feedback, and there are virtually no complaints about the AI.

It is important to note that one of the strong points of our AI system is its implicit ability to *anticipate* opponents' actions. This is achieved by learning human actions in pairs with *earlier* game situations (rather than with the situations, perfectly corresponding to user input). In general, we use the earliest situations that have enough data to react to the opponent, and ignore actual user input delay. This makes AI-controlled characters somewhat stronger than people, who cannot react immediately. On the other hand, people are more flexible in their strategy, and our current experiments show no clear advantage of AI over people and vice versa.

However, as game designers we see at least, two directions for future improvements. We noticed that people often predict actions of *familiar* opponents, i.e., adjust their game strategy when facing a particular player. Currently, our AI does not distinguish opponents. Another important human trait is the ability to learn quickly on the go. If some action leads to a quick victory or a defeat against a particular opponent, people notice it and adjust their actions accordingly. Implementing this ability would require some variation of reinforcement learning, which we have in plans.

VI. Conclusion

Machine learning and case-based reasoning methods are often described and used in discrete environments with clearly defined input and output data, and streamlined decision-making process. In the domain of computer games, these assumptions often do not hold, so the game designers have to perform a thorough analysis to entwine machine learning with game logic. Much of this work is dedicated to transforming user intentions into actual (situation, action) pairs that make sense in the given game world, and agree with its rules. Even in relatively simple games we see that user actions are largely dictated by context-dependent tactics and anticipation of opponents' actions. We believe that automating this work is hardly feasible with current state-of-the-art methods, especially in commercial computer games that require reasonable decision making as well as low consumption of processor and memory resources. It means that "automatic" methods of constructing behavior of non-player characters (such as a combination of learning by observation with case-based reasoning) in practice still come with a large amount of manual effort, and thus their use should be well justified. In case of World of Tennis: Roaring '20s we had a special reason to use these methods, and we are satisfied with the obtained results.

VII. References

[1] I. Millington and J. D. Funge, *Artificial Intelligence for Games,* 2nd ed. Burlington: Morgan Kaufmann Publishers, 2009.

[2] A. Champandard, *Top 10 Most Influental AI Games: AIGameDev.com.* Available: http://aigamedev.com/open/highlights/top-ai-games.

[3] D. Takahashi, *How Microsoft's Turn 10 fashioned the A.I. for cars in Forza Motorsport 5.* Available: https://venturebeat.com/2013/11/06/how-microsofts-turn-10-fashioned-the-ai-for-cars-in-forza-motorsport-5-interview.

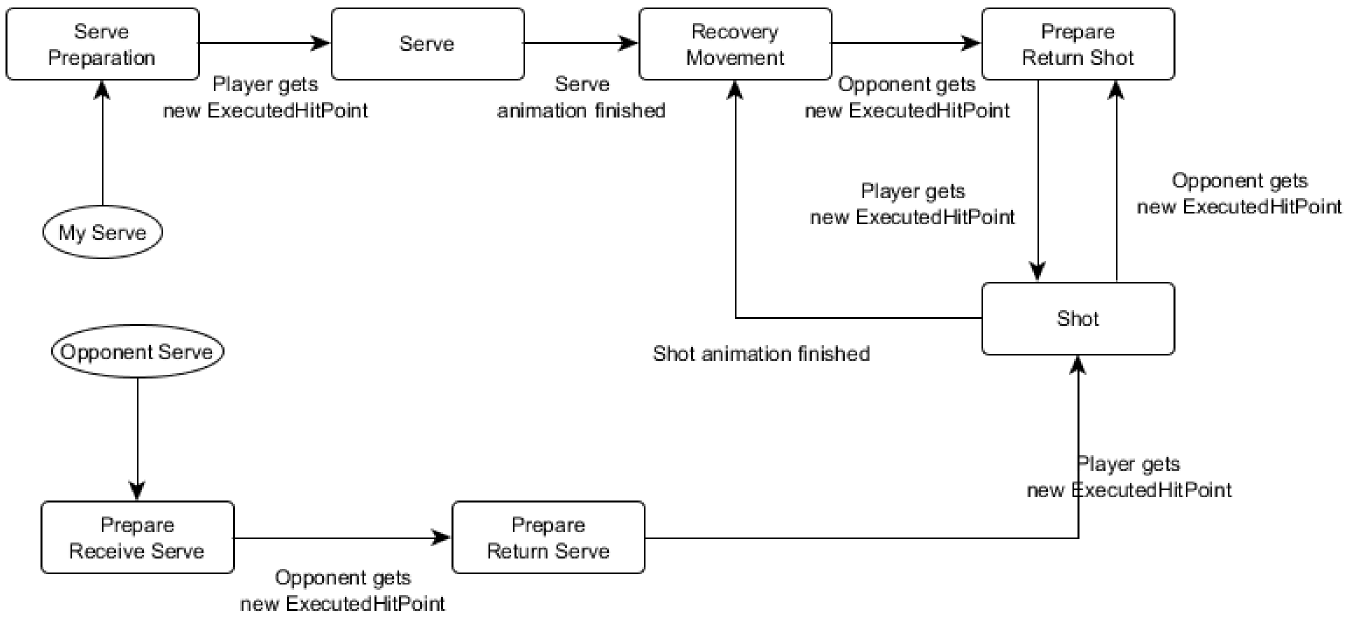
[4] M. Mozgovoy and I. Umarov, "Behavior Capture with Acting Graph: A Knowledgebase for a Game AI System," in *Databases in Networked Information Systems (DNIS): 7th International Workshop*, S. Kikuchi, A. Madaan, S. Sachdeva, and S. Bhalla, Eds, Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 68–77.

[5] M. Mozgovoy, M. Purgina, and I. Umarov, "Believable Self-Learning AI for World of Tennis," in *2016 IEEE Conference on Computational Intelligence and Games*, 2016, pp. 1–7.

[6] A. Aamodt and E. Plaza, "Case-based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," *AI communications*, vol. 7, no. 1, pp. 39–59, 1994.

[7] V. Mnih *et al,* "Playing Atari with Deep Reinforcement Learning," in *NIPS Deep Learning Workshop*, 2013.

Fig. 3. Game Phases Diagram



Serve Preparation
Serve
Recovery Movement
Prepare Return Shot
Player gets new ExecutedHitPoint
Serve animation finished
Opponent gets new ExecutedHitPoint
My Serve
Player gets new ExecutedHitPoint
Opponent gets new ExecutedHitPoint
Shot
Shot animation finished
Opponent Serve
Player gets new ExecutedHitPoint
Prepare Receive Serve
Prepare Return Serve
Opponent gets new ExecutedHitPoint