DEVELOPING TRAINABLE BOTS FOR A MOBILE GAME OF TENNIS

Maxim Mozgovoy[1], Akane Yamada[1], and Iskander Umarov[2]

[1]The University of Aizu
Tsuruga, Ikki-machi, Aizuwakamatsu
Fukushima, 965-8580 Japan
{mozgovoy, m5191102}@u-aizu.ac.jp

[2]TruSoft Int'l, Inc.
204 37th Ave. N #133
St. Petersburg, FL 33704 USA
umarov@trusoft.com

KEYWORDS
Case-based reasoning, learning by observation, behavior capture.

ABSTRACT

While behavior patterns of AI-controlled videogame characters are typically designed manually, self-learning AI systems possess unique attractive features. They can be used to create distinct character personalities of different skill levels, and "train your own character" can be a major user-end feature of a game. In this paper we present our attempt to develop a self-learning AI system for a mobile game of tennis. We show that our AI agents can learn behavior patterns from user actions, and play accordingly in similar game situations, exhibiting playing skills comparable to skills of a trainer.

INTRODUCTION

According to the report by International Data Corporation, online multiplayer games already surpassed single-player games in terms of consumer spending and player commitment (Ward 2015). The competition among such games is high, so game designers have to introduce innovative gameplay elements to stay on the market.

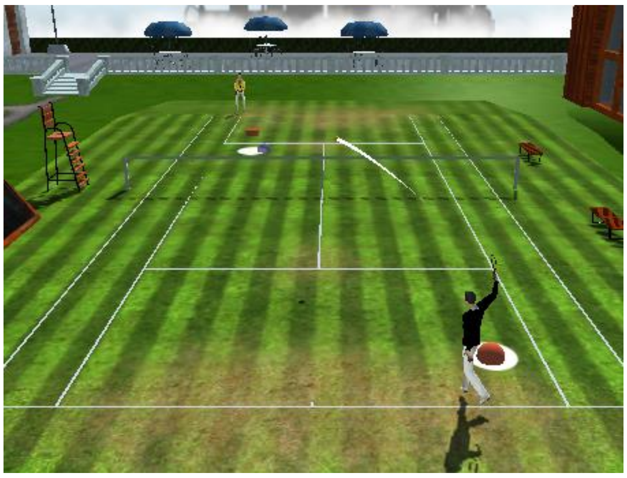
Typically, in online multiplayer games people compete both with other people, and with AI-controlled bots, so the quality of AI system has a significant impact on the overall success of a product. For our ongoing project of an online mobile tennis game, we decided to concentrate the effors on the AI system that implements the elements, suggested in (Umarov and Mozgovoy 2014):

1. Complex, non-repetitive behavior of AI bots.
2. Distinct personalities of AI bots, exhibiting a variety of skill levels and playing styles.
3. "Train your own character" mode as an element of gameplay.

To achieve these goals, we employ a learning by observation-based approach to AI design, outlined in (Mozgovoy and Umarov 2011). As a result, we are planning to obtain AI agents that can learn from human players, and exhibit human-like behavior, comparable in terms of style and skill level to behavior of their trainers. Our current system shows promising results, and is already able to learn from human actions and play accordingly.

TENNIS GAME ENGINE

The game of tennis relies on a custom-designed game engine, developed with Unity3D (see Figure 1).



Figures 1: Tennis Game Engine

Each player in the game can perform actions of two types: run to the specified location and shoot the ball into the specified point on the court. The game physics is accurate (Lopukhov 2015), so, for example, if a player tries to send the ball coming at a high speed to a nearby point on the opponent's side of a court, a backspin shot will be performed, so the ball will fly high over the net. Since we want to favor tactical gameplay rather than arcade, the game engine will automatically steer the players towards optimal ball receiving points. Thus, a player only has to care about own character location while the ball is moving towards the opponent, and about the best target for the next shot.

AI DESIGN PRINCIPLES

The AI subsystem can operate in two distinct modes. In learning mode, it observes the actions of the specified player, and stores them in its knowledgebase. In acting mode, the AI subsystem uses its knowledgebase to retrieve the most suitable action for a given game world state upon request from the game engine.

AI knowledgebase is represented with a set of three graphs, where individual vertices correspond to different game situations, and edges correspond to player actions. The difference between the graphs is in the list of attributes used to identify a single game situation. In other words, each graph describes desired agent behavior in form of a finite state machine at more or less accurate levels of abstraction. Our current system setup is described in Table 1.

In learning mode, each player action triggers insertion of a new (Game situation, action) pair into each of three graphs. If a game situation with the same attributes already exists in a certain graph, the action will be connected to the existing game situation node.

Table 1: Attributes of a Game Situation

Graph	Attributes
most accurate (level 0)	Game state (serve / hit / receive / etc.)
	Player position*
	Ball target point*
	Player's intended shot target*
	Player movement destination point*
	Opponent position*
	Opponent intended shot target*
	Ball position*
average accuracy (level 1)	Game state (serve / hit / receive / etc.)
	Player position*
	Ball target point*
	Player movement destination point*
	Opponent position*
	Opponent intended shot target*
	Ball position**
least accurate (level 2)	Game state (serve / hit / receive / etc.)
	Player position**
	Ball target point**
	Opponent position**
*	*(x, y); inside a 10×10 grid*
**	*(x, y); inside a 5×5 grid*

Table 2: Sequence of Knowledgebase Queries

Query	Graph level	Require action adjacency	Extended range search on attributes
1	0	Yes	—
2	0	No	—
3	1	Yes	—
4	1	No	—
6	2	Yes	—
7	2	Yes	Player, opponent, and ball coordinates
8	2	No	—
9	2	No	Player, opponent, and ball coordinates

In acting mode, the AI subsystem performs a number of queries to the graphs in order to find the best possible action in the given game situation. We start with the most strict query to find a perfect match for the given game situation in the most accurate graph, and if no relevant actions are found, we relax search conditions.

Currently there are three ways to relax a query: a) search in a graph of a higher abstraction level; b) instead of a perfect match require a match within a given value range — this is helpful for numerical attributes, such as player coordinates; c) do not require that two subsequent actions are also adjacent in the game graph (and thus do not follow the same game strategy). We cannot rely on assumption that AI always finds a perfect match in the knowledgebase for any possible game situation, so certain capabilities for approximate search are necessary.

A fragment of an actual level 0 graph of a trained agent is shown in Figure 2. The sequence of search actions we currently use in acting mode is provided in Table 2.

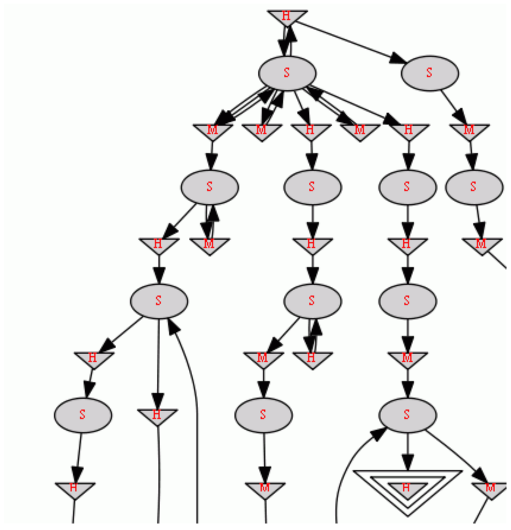


Figure 2: A Fragment of a Game Graph (Visualized with AT&T GraphViz)

EXPERIMENTAL SETUP

One of the goals of fine-tuning graph attributes and query conditions is to improve the overall playing experience for the users. In particular, we believe that the AI system should exhibit satisfactory behavior after 6-10 minutes of observations. Our experiments show that in average a player performs approximately 100 actions per minute (auto-steering movements, performed by the game engine, are also classified as actions), so a typical training session includes 600-1000 actions.

As a preliminary test of our system, we ran several game sessions between three players, as described in Table 3. An individual match ends when seven points are scored by either party. Obviously, the most skillful player is C, and the key component of his playing style is a winning strategic placement of a game character.

Next, we tested how the obtained AI agents play against each other by playing three matches for each pair of opponents. In all experiments the agents showed performance comparable to their trainers: A beats B with a score ranging from 7:0 to 7:3, and C beats A with the same outcome. Furthermore, C was able to beat B with a score 7:1 in all three test matches. This outcome is expected, since A is more skilled than B, but our training data did not contain games between B and C. While these results cannot yet prove that the obtained agents actually preserve acting style of human players, they show that the relative skill level of the opponents was preserved.

Table 3: Game Sessions

Session	Duration, sec	Players	Outcome
1	56	A vs. B	7:0
2	66	A vs. B	7:1
3	80	A vs. B	7:2
4	81	A vs. B	7:2
5	83	A vs. B	7:2
6	57	A vs. C	2:7
7	53	A vs. C	0:7
8	47	A vs. C	0:7

CONCLUSION

Game AI is a special topic for academic research, since computer games set such unusual requirements for AI systems as human-likeness, unpredictability, skill level adjustments, and *fun*, a core of any entertainment. Self-learning AI systems have a potential to address these challenges, since they can directly learn from human opponents, and thus exhibit distinct human-like behaviors of different skill levels. Surveys show that gamers actually prefer such AI opponents (Soni and Hingston 2008).

In this paper, we briefly discussed our work-in-progress AI systems for the game of tennis. While our AI did not reach a production-quality stage yet, it demonstrated the ability to learn and repeat behavioral patterns of human players

REFERENCES

Lopukhov, A. 2015. "Realistic Ball Motion Model for a Tennis Videogame". *Proceedings of International Workshop on Applications in Information Technology (IWAIT), pp. 81-83*.

Mozgovoy, M. and Umarov, I. 2011. "Behavior Capture with Acting Graph: a Knowledgebase for a Game AI System." *Lecture Notes in Computer Science*, vol. 7108, pp. 68-77.

Soni, B. and Hingston, P. 2008. "Bots Trained to Play Like a Human are More Fun". *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 363-369.

Umarov, I. and Mozgovoy, M. 2014. "Creating Believable and Effective AI Agents for Games and Simulations: Reviews and Case Study". *Contemporary Advancements in Information Technology Development in Dynamic Environments*, pp. 33-57.

Ward, L. 2015. "Gaming Spotlight, 1H15: How Online Multiplayer Is a Game Changer." Document #256831, International Data Corporation.