Logfile Player and Analyzer for RoboCup 3D Simulation

Steffen Planthaber and Ubbo Visser

Center for Computing Technologies
Universität Bremen, Germany
{steffen,visser}@tzi.de

Abstract. In multi agent environments or systems equipped with artificial intelligence it is often difficult to obtain the function or method which led to a particular behavior that is noticeable from outside. However, this information is crucial if not necessary to optimize the agents behavior. In the RoboCup 3D simulation league this dilemma becomes obvious when replaying logfiles of a game that was simulated before. The 3D soccer simulation league monitor (rcssmonitor-lite) is restricted with regards to replaying logfiles.

This paper describes the concept and the implementation of improvements for the logplaying and analyzing abilities of the monitor. The idea is to provide a tool that is able to assist developers to detect problems of their agents both in single and cooperation mode.

1 Motivation

In the past 10 years the simulation league was two dimensional, all players and even the ball moved on the ground. During this time numerous sophisticated tools were created for analyzing the simulated games such as Logalyzer[1] or Team Assistant[2].

The Logalyzer provides information about detected actions like passes and several visualizations for the collected data about the game. The Team Assistant is able to display information provided via agent logfiles along with statistics about detected actions. The Team Assistant is also mentioned in the 2002 league summary[3] as the winner of the presentation tournament.

In 2003, the 3D simulation was introduced including basic tools to view and replay the simulated game. The tools used in 2D can not be used in 3D simulations because of the lack of one dimension and a different format of the logfiles.

The current monitor is capable of showing the current simulated game (at current time) and of replaying monitor logfiles. The replaying mode can be used to watch previously simulated games again. There is also a "single step mode" which provides slow motion replay.

When trying to develop a behavior for an agent or verifying behaviors acquired by machine learning methods it is hard to determine which methods or functions led to the actions observed in the game. This information is crucial when trying to debug or improve the agents in their behavior and collaboration. Especially

G. Lakemeyer et al. (Eds.): RoboCup 2006, LNAI 4434, pp. 426–433, 2007.

in the case of collaboration it is tedious and time-consuming to check what each agents intention is. This is caused by the fact that every logfile has to be searched for the right record of the actual time displayed in the monitor by hand. Additionally, the agent logfiles are not numbered according to the uniform numbers of the agents which complicates finding the desired logfile.

The aim of this work is to create a logfile player and analyzer for the 3D simulation league with new functions allowing to analyze the agent's behavior and collaboration with other agents. The importance of the evaluation of agent teamwork has been addressed in many papers for 2D simulation [4][5] and [6].

2 Related Work

Analyzing tools in 3D simulation are rare, most of the development efforts that have been done in the past years were made in the conception and development of the 3D simulation server. Even in the 2005 soccer simulation development competition just one tool was introduced, which is the "Persian Robotics Analyst"[1].

The Persian Robotics Analyst (PRA) offers information about ball possession, successful and unsuccessful passes and about good and bad actions.

There is also an unpublished "Studienarbeit"[7] (student project) at the Universität Koblenz which has been worked on in parallel to this work. There are some common functions with this work such as the possibility of agents to draw into the displayed scene or the displaying ability of text messages according to the current scene. That tool also provides the detection of (double-) passes, goalshots, dribbling and their outcome. Also ball contacts and tackling are detected. Statistics about these detected events are written into a text file. Commentatory text messages can be displayed according to the current detected situation. These comments were the main aspect of this work.

3 Requirements

In consultation with other agent developers, providing an easy and clear way to gather information about what the agent intends and how the world looks like, according to the agent, is essentially needed.

The following features are judged beneficial and essential to debug handwritten behaviors and to verify the decisions of learned behaviors.

3.1 Features

While analyzing a special situation of the game it is obvious that *forward and backward replay in different speeds* is useful. With this possibility the situation can be analyzed again without starting the game from the beginning until the desired situation is reached.

[1] `http://www.persianrobotics.net`

To gain knowledge of the agents intentions in an situation an *output of agents logfile according to current time* is needed.

In addition to the logfile displaying it is valuable to *enable the agents to draw information directly into the displayed scene*, like a line from the agent to the position it intends to move to.

Also *filtering the logfile output* may be helpful to display only those information needed by the developer. This way only those information provided by the current developed behavior could be displayed.

New *camera positions*, like birdview which resides directly over the agent of interest, may be beneficial.

When using the single step mode of the monitor *displaying the ball's and player's movements for some time forward* can be an improvement, the developer could see the next movements without proceeding forward in scene display.

A *Grafical User Interface* would be useful to grant an easy access to all functions of the monitor/ logplayer. This way nobody would have to remember all the keyboard shortcuts.

3.2 Additional Features

Displaying the offside line will become necessary when the 0.4 version of the simulation server becomes official (probably RoboCup 2006, not for qualification), which will include the offside rule.

In some situations it could be difficult to distinguish on which side of the offside line an agent is, so *marking agents that are in an offside position* if the ball would be played to them may be a good feature.

Detecting events and the number of their occurrence may provide essential information about what parts of the agent have to be worked on (i.e. if passes often fail there is some space for improvements). Those events are: pass success/fails, ball dribbling, lost balls, goal shot (success, out, intercepted) and kick out.

By *detecting event sequences* more information can be extracted such as lost balls after dribbling.

Plotting graphs also may provide beneficial knowledge. If the ball is in the own half most of the game, the agents may play too defensive. Or if they move all the time their batteries may run low. Useful graphs could be: goal distance, velocity and ball possession.

4 Implementation

When talking about a logplayer or a monitor this essentially means the same program in two different operation modes, logplayer means that the program replays logfiles of a previously simulated game, montior that a game that is currently simulated is displayed.

4.1 Features

Forward and backward replay in different speeds is achieved by reading and parsing the logfiles at the beginning of the execution of the logplayer into a new data

structure with the result that playback is simply done by iterating through the (previously parsed) gamestates. Playback is much faster now, it has to be slowed down artificially leading to the opportunity of different playback speeds.

Output of agent logfiles according to current time can be done if the agent provides time information for its output in its logfile. The times according to the various agent outputs is also parsed at startup. When displaying, the fitting record of the agent logfile is found using the time currently displayed by the monitor.

The rcssmonitor-lite does not really know what the numbers of the agents (0-21) according to their uniform numbers are, this makes it hard to display the right logfile, only if the agent writes its team name and uniform number into its logfile, the right logfile is displayed.

In order to *Enable the agents to draw information directly into the displayed scene* the agent's logfile record for a given time is separated into "draw commands" and plain text. Draw commands are not displayed as text output. They are parsed according to the kind of the command and the desired action is executed (fig. 1).

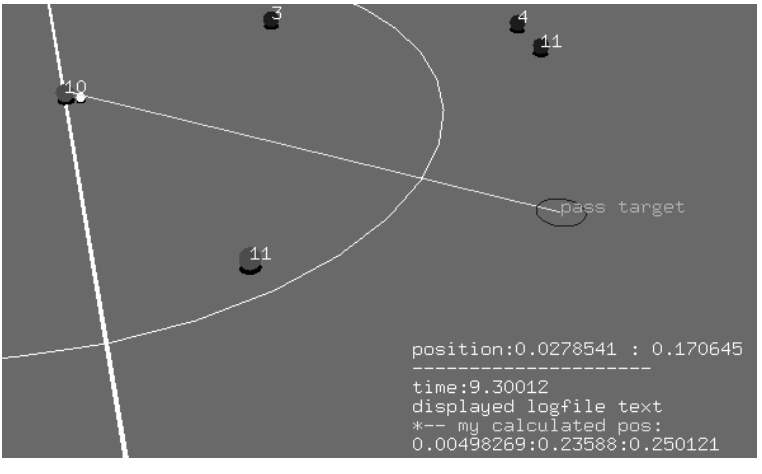

Fig. 1. Output of an agent

Implemented draw commands are: A circle which draws a circle at the given position with a given radius. Lines can be drawn from a starting point to a target point, if a length is given the line starts at the start point into the direction of the target point but stops at the length. Text can be displayed at a given position in the 3D scene. Ground rectangles colorizing the ground in color given though the color command which changes the color of all further items drawn.

In case of *filtering the logfile output* the agent just has to name a filter himself and use it. The filter is detected while parsing the logfile and is made selectable for the developer.

New *camera positions* were implemented, most of them are adjustable in a way (camera height, distance or point to look at).

Displaying the ball's and player's movements for some time forward is solved as lines in the color of the team or white for the ball. The length of the line (how far into the future movements are shown) can be adjusted. The agents track (movements throughout the whole game) can also be displayed, this track can be colored according to the agents movement speed.

The *Grafical User Interface* (GUI) has different layouts according to the current usage. When the program is used as monitor, game control buttons are displayed (drop ball, kickoff, kickoff side). The controls for logfile displaying are shown when used as logplayer.

4.2 Additional Features

For *plotting graphs* the desired informations are piped into the gnuplot[2] program. Information that can be displayed this way are ball position, player position, distance to opponent goal and player speed.

4.3 Other Features Derived from the Implementation

The new data structure of the program allows the user to use single step mode, for- and backward even when watching the game "live". The server is continuing the simulation in background, even if the monitor is in single step mode. Functions that are independent from agent logfiles, such as movement display, can be shown in this mode.

5 Results

The logfile output is useful to understand the agents behavior. This program has replaced the lite monitor that comes with the simulation server within the Virtual Werder team, which started the development of this program. The agent may communicate its decisions to the developer. (i.e. which role it plays, which behavior of the role it has chosen and what action it selected (fig. 1, 2)).

In occurrence of an error in the ball movement prediction the agent was changed to display its data about the ball (its position and movement vector according to the agents world model). This way the error became visible. The line representing the movement vector pointed straight upwards, no matter in which direction the ball was moving. This way the error was resolved very quickly, the x and y component of the movement vector were not written properly into the world model. Without displaying that data, localizing the error would have taken much longer.

When creating a behavior that covers an opponent, it is mostly wanted that only one player covers an opponent at a time. To verify the opponent selection

[2] `http://www.gnuplot.info`

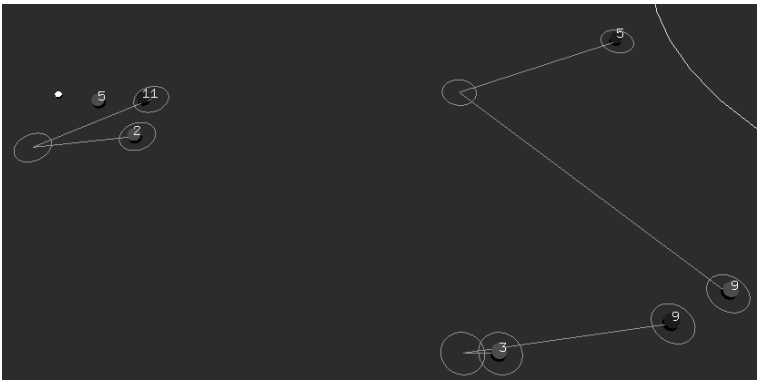

Fig. 2. Covering output

the agent can simply draw a line from himself to the opponent it intends to cover or include the chosen position into its drawings. When displaying the draw commands of all agents it becomes visible if something is wrong with the selection or the position the agent has chosen to move to (lines connecting the agent with the one he has chosen to cover, the circle in middle of the line is the cover position; fig.2). Especially in this case it is useful to display all drawings of the agents to recognize errors in team collaboration.

The new camera features are also useful for verifying behaviors. For example in case of optimizing the ball approaching, a good position for the camera is directly over the agent looking down and moving with it. When checking the goalie behavior a view from behind the goal looking to the ball is a good position for the camera. The view from the side moving left and right with the selected object will be useful to develop offside rule handling.

In case of the ball approach the display of the future movements in addition to the camera view becomes valuable, the developer may directly see possible failures i.e. the agent takes a way that is too long or is moving around the ball too close or far away (fig. 3).

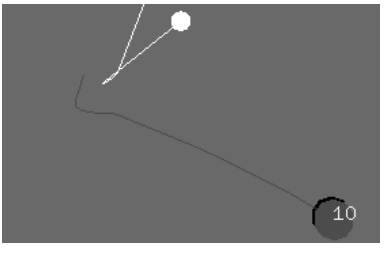

Fig. 3. Approaching the ball

The ability of reverse playback of the game gives the opportunity of analyzing the same scene again without watching the game from the beginning. Also slight transitions in action selection may be analyzed without much interference. This feature in addition to the display of agentlogs (in both ways, text and drawings) gives the opportunity to understand the agents decisions according to its own world model, and not only the real simulated world displayed in the monitor.

The ability of delayed playback and direct analyzing of a currently simulated game provides a real speedup in development in comparison to other tools. Simulating a 3D game may take up to 10 or 20 min which is a long time the developer has to wait in order to in example analyze the previously mentioned ball approach (fig. 3) in slow motion. With the delayed playback that situation may be analyzed while the game is still simulated in background. After the analyzing was finished the game may be watched time shifted in normal speed or the replay may be fastened to reach the most recent scene.

The track display in the 3D scene along with the plotting opportunities were giving the developer essential information about the agents or ball movements throughout the game. This way the developer can find agents which are not moving much or fast, which may point to a suboptimal formation or behavior.

By viewing the ball movement plot (fig. 4) of the 2005 final between Brainstormers3D (left) and Aria2005 (right) it is obvious that the ball was mostly located on the left side of the field and though that Aria2005 dominated the game. It also looks like Aria2005 is able to cross the ball and the Brainstormers3D are not. Aria2005 won this match 2:0.

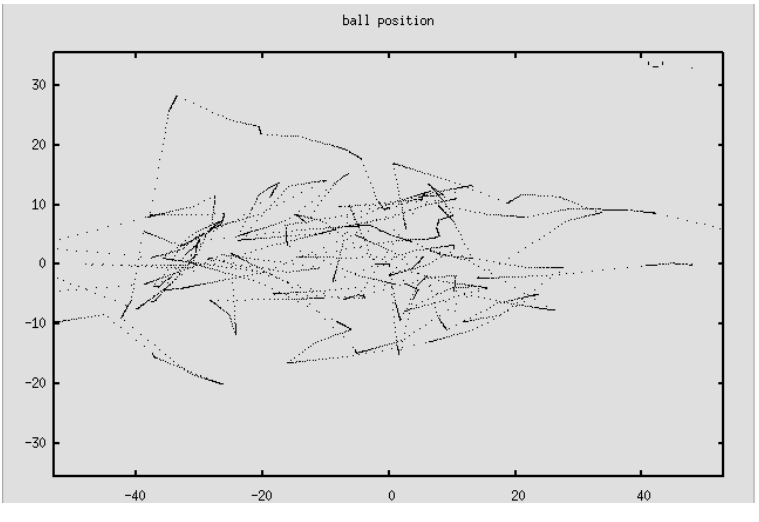

Fig. 4. Plotted ball position

6 Conclusion

The logplayer can be a useful tool when trying to resolve strange behavior of an agent or verifying the proper work of the agent. The Virtual Werder team resolved some problems within their code using the logplayer. These resolved problems are namely the examples given in section 5. The resolving of these problems would have taken longer without the possibility of drawing or text output.

Due to the parsing at the beginning the startup of the logplayer takes longer in comparison to the lite monitor/ logplayer, but the playback is much faster and the monitor does not have to be restarted to watch the game again.

On the other hand the agent logfiles are growing rapidly in size if a lot of draw commands are written to them which also leads to high memory usage after startup.

References

1. Bezek, A.: Modeling multiagent games using action graphs. [Online]. Available: `http://dis.ijs.si/andraz`
2. Nazemi, E., Zareian, A., Samimi, R., Shiva, F.A.: Team assistant team description paper. In: Proc. of Robocup (2002) [Online]. Available `http://www.sbcee.net/files/sbce.pdf`
3. Obst, O.: Simulation league - league summary. In: Kaminka, G.A., Lima, P.U., Rojas, R. (eds.) RoboCup 2002. LNCS (LNAI), vol. 2752, pp. 443–452. Springer, Heidelberg (2003)
4. Raines, T., Tambe, M., Marsella, S.: Automated assistants to aid humans in understanding team behaviors. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup-99: Robot Soccer World Cup III. LNCS (LNAI), vol. 1856, pp. 85–104. Springer, Heidelberg (2000)
5. Takahashi, T.: Logmonitor: From player's action analysis to collaboration analysis and advice on formation. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup-99: Robot Soccer World Cup III. LNCS (LNAI), vol. 1856, pp. 103–113. Springer, Heidelberg (2000)
6. Kaminka, G.A.: The robocup-98 teamwork evaluation session: A preliminary report. In: Veloso, M.M., Pagello, E., Kitano, H. (eds.) RoboCup-99: Robot Soccer World Cup III. LNCS (LNAI), vol. 1856, pp. 345–356. Springer, Heidelberg (2000)
7. Werres, A.: Erweiterung eines 3d-simulators für den robocup (2006) (unpublished)