# The magmaOffenburg 2017 RoboCup 3D Simulation Team

Martin Baur, Klaus Dorer, Jens Fischer, Duy Nguyen, Carmen Schmider, David Weiler[1]

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany

**Abstract.** In this TDP we describe a new tool created for testing the strategy layer of our soccer playing agents. It is a complete 2D simulator that simulates the games based on the decisions of 22 agents. With this tool, debugging the decision and strategy layer of our agents is much more efficient than before due to various interaction methods and complete control over the simulation.
In the future, the tool could also serve as a measure to run simulations of game series much faster than with the 3D simulator. This way, the impact of different play strategies could be evaluated much faster than before.

## 1 Introduction

In the first years after the 3D simulation league was introduced, the main focus rested on implementing and optimizing low-level behaviors such as walk or kick behaviors. Over the years, as these basic behaviors advanced to a fairly robust state, teams started implementing high-level behaviors like positioning and team play.

As the focus of the league shifts slowly but surely from successfully executing low-level behaviors to strategies and decision-making, tools for the development of the latter become more important. One such tool is our newly created "2D strategy simulator".

Success in RoboCup always has been closely related to the value of tools that help to define and understand what the agents are or should be doing. The success of CMUnited 1999 in 2D simulation came along with a new debugging tool providing layerd disclosure of agents [1]. A more closely related tool to the one we present here is PlayMaker of FC Portugal [2] that can be used to define strategies and setplays of agents. It is however, to our knowledge, not containing a simulator that runs the decision loop of agents. UT Austin Villa uses RoboViz [4] to visualize the role assignment and marking in their team [3].

## 2 Motivation

Why is a new simulator needed as a tool for developing strategies? What benefits does it have over using the already exisiting 3D simulator?

There are several reasons for this, the two main ones being:

– Simulations in the 3D simulator are expensive, a modern CPU can barely run it in real-time with 22 players. Running it faster than real-time would be desirable in some scenarios, but even if performance allowed it, the results would not be consistent due to the architecture of the 3D simulator.
– The 3D simulator is inherently non-deterministic. This can be painful when trying to reproduce certain bugs during development, where determinism is desired.

For these reasons, we have had a "Simulator View" in our development tool called *magmaDeveloper* for several years (Fig. 1). It presents a soccer field with 22 players and a ball, allowing the user to manually arrange certain situations with drag-and-drop. While he does so, our decision making logic is automatically executed in the background, so changes in decision-making are displayed to the user instantly via various renderers. This includes information like:

– desired positon
– desired behavior
– current role in our strategy
– best kick options
– etc.

Since all agent instances are running in the same Java process together with magmaDeveloper, it's easy to set a breakpoint in the decision-making logic of a particular player. Such a breakpoint will be hit after any interaction occurs, because this triggers another execution of the decision-making logic.

While this has been an incredibly valuable tool so far, it does have one major limitation: due to all interaction happening manually, it was very static. Because of this, it was sometimes difficult to imagine what a strategy would look like in a real game, where players can "move on their own".

This is why we decided to expand upon this idea and implement a real simulator on top of this existing view. The simulator component was to be heavily simplified compared to the 3D Simulator, turning it into a 2D simulation with only a few basic commands for walking, kicking and beaming to certain x and y coordinates.

## 3 Architecture

The new strategy simulator consists of three main components as can be observed in Fig. 2.

Here is a quick overview of what each component is responsible for:

– **Simulator** - The actual simulator that holds all the state (player and ball positions, playmode, fouls, etc...). It is updated 50 times per second to execute the behaviors that the individual players want to perform this cycle and enforces the rules of the simulation.

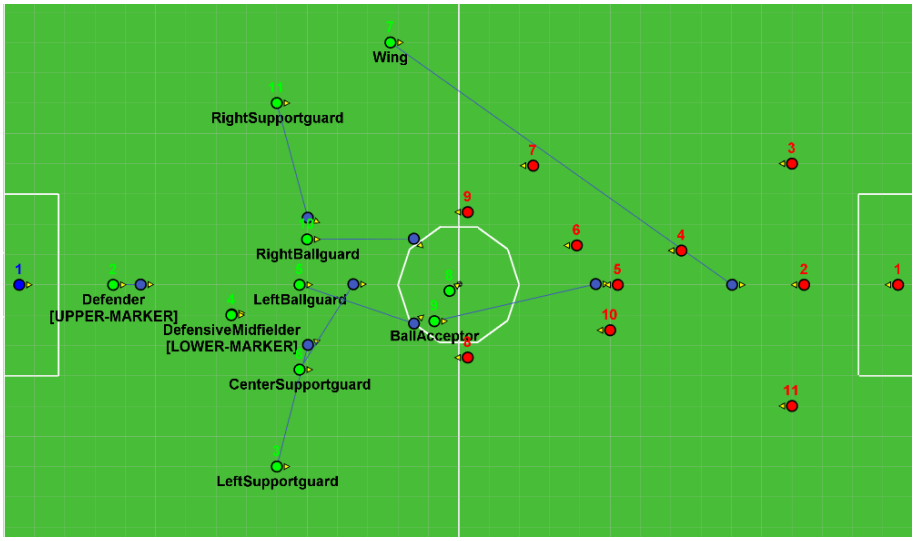


**Fig. 1.** The old simulator view in magmaDeveloper with the roles renderer active.

- **MagmaSimulator** - Not an actual "simulator" as the name might suggest, but rather the "glue code" between agent runtimes representing a single agent and the simulator. This means it extracts information about what behavior to execute and informs the simulator about it, as well as updating the internal state of the agent runtime to reflect the new player and ball positions after a cycle.
- **Simulator View** - The simulator view is the Java Swing GUI that renders various information from agent runtimes, MagmaSimulator and Simulator. It already existed before, but has been updated to be compatible with the changed architecture and to support the new features.

## 4 Features

Fig. 3 shows the new simulator view, the biggest difference to the old one being a "Play"-button that can be used to start or pause/resume the simulation. A spinner component can be used to control the desired speed of the simulation, which is a big improvement compared to the 3D simulator.

Each cycle, the complete state of the simulator is copied into a history, which enables *rewinding* the simulation to an arbitrary previous state, and picking up again from there. This is invaluable for debugging purposes, as the same situation can be observed again and again almost as if looking at a log file. In an even more advanced use case, the user might decide to set a breakpoint in the decision-making logic of the agent, make some changes and let the IDE hot-swap that code into JVM - all at runtime without a need to restart the tool. Like this,

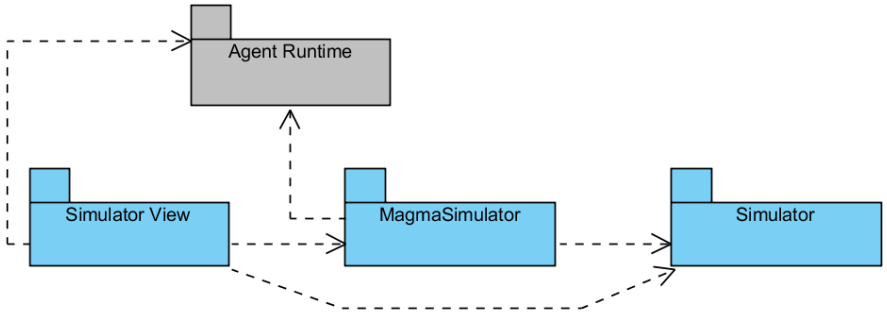


**Fig. 2.** The main components of the strategy simulator.

the changes in behavior can be observed right away, allowing for rapid iteration times.

The history can even be saved to and loaded from a file, in case one of our team members has found a bug and wants to give another team member an easy means of reproducing it.

The simulation itself is designed to be as similar as possible to the high-level behavior of games in the 3D simulator. One of the measures we took to achieve this was to port relevant playmodes and fouls directly from the 3D simulator's `soccerruleaspect.cpp`. Some playmodes like free kicks were left out, since they are only used by a human referee. The only fouls that were ported are those related to player-positioning fouls, namely "crowding" and "illegal defence".

The walk speeds are provided by the so-called `WalkEstimator`-component of our agent runtime, which contains walk speeds measured in the 3D simulation. There are no additional checks performed by the strategy simulator to see if the walk speeds are viable or not. Similarly, we measured probability distributions for all the kicks we use (and separately for Nao and NaoToe) to achieve realistic kick distances and angles.

A kick can be considered "unstable", meaning that the player falls down after performing it and becomes inactive (greyed out in the view) for the time it normally takes him to get up (around 2 seconds).

Kicks are the only thing that has any noise associated with it. Each player has a global view and knows the exact position of all players and the ball at all times. Despite kick noise, the simulation is still completely deterministic, and the seed passed to the random number generator (RNG) can be changed. When copying the simulator state for the history, the initial RNG seed is saved along with the number of invocations so far, making the randomness reproducible.

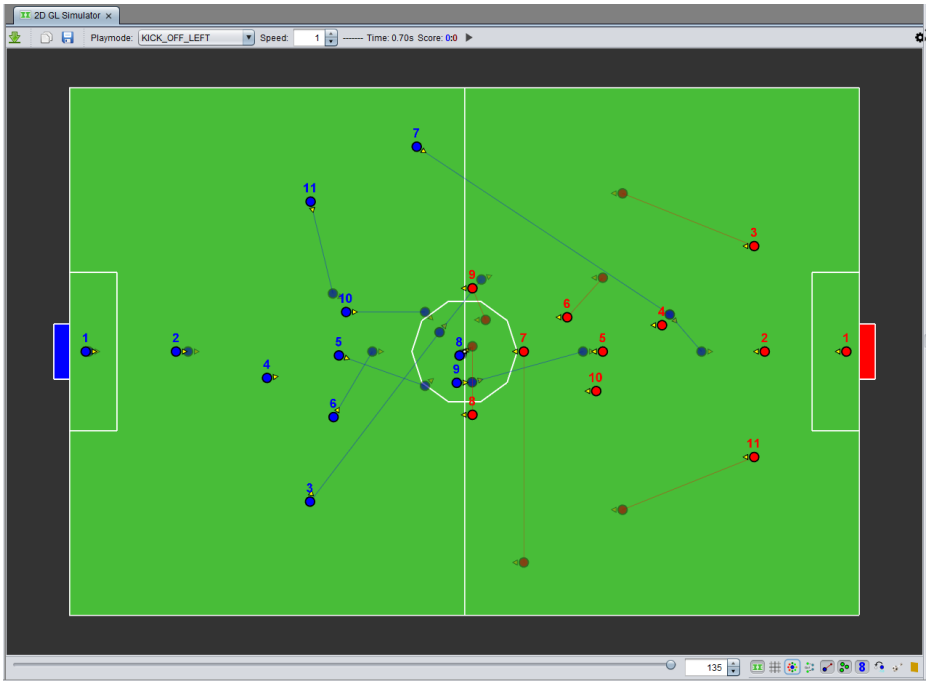


**Fig. 3.** The new simulator view with the desired positions renderer active.

## 5 Results

With the new strategy simulator, we have a very promising tool which we hope can help us develop better strategies in the future. Already, its development led to several interesting observations that revealed optimization possibilites in our agent runtime. One example for this is the kick distance probability distribution we measured for our 15m kick (Fig. 4). There is a distinct spike around 0m for NaoToe, which suggests NaoToe's 15m kick is significantly less reliable than its counterpart. The curve also shows a slight dent at around 13m.

From the start, one of the objectives for the strategy simulator was to provide a faster way to play game series. Game series are a valabule tool for evaluating changes to agent behavior, but are very costly when it comes to time. A game series with 100 games typically takes around 22 hours in the 3D simulator in case of serial game runs.

In the strategy simulator, the bottleneck is no longer the simulator itself, but updating the agent runtimes. We were able to easily parallelize this step, since the updates are not dependent on each other. In theory, we could benefit from up to 22 CPU cores this way, but with only 4, a single game already only takes around 20 seconds. This allows playing 1000 games in 5 to 6 hours, making it roughly 40 times as fast as game series in the 3D simulator. Note that we created

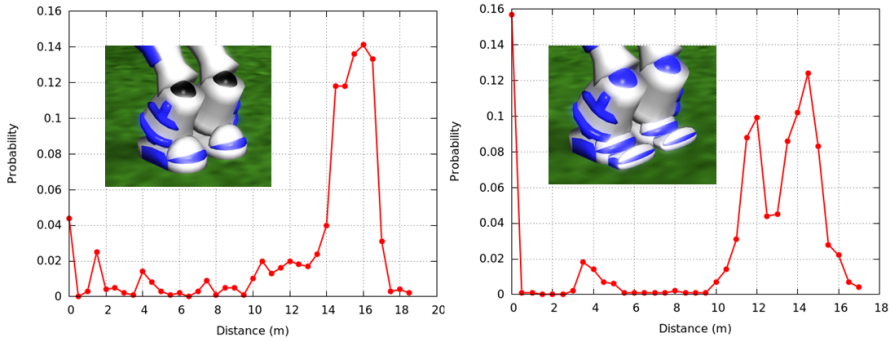


**Fig. 4.** Kick distance probability distribution for Nao (left) and NaoToe (right).

a separate command line interface for the strategy simulator, which is slightly more optimized than the version with a GUI running in magmaDeveloper.

Unfortunately, it seems like results from strategy simulator game series are currently not indicative of a similar performance in the 3D simulator. To evaluate this, we played game series of our "ManToManMarker" strategy against all other defined strategies, both in the 3D and strategy simulator. The results can be seen in Table 1. Series in rcsserver3d consisted of 100 games, series in the strategy simulator of 1000.

| **ManToManMarker vs ...** | **rcssserver3d** | **Stategy Simulator** |
|---|---|---|
| NoWingJustAcceptor | 0.440 - 0.240 | 0.838 - 0.736 |
| Acceptor | 0.490 - 0.290 | 0.820 - 0.845 |
| BallReceiver | 0.600 - 0.250 | 0.913 - 0.878 |

**Table 1.** Game series in rcssserver3d and strategy simulator

Two things can be observed here: the results of the "ManToManMarker vs Acceptor" game are contradictory, and in general, the goal difference seems to be much smaller in the strategy simulator.

In the future, we hope to eliminate as many discrepancies as possible between the behavior of strategies in the two simulators. If we succeed in doing so, it should give us a way to significantly speed up iteration times when it comes to testing new strategies or changes to existing ones.

Layered Disclosure: Revealing Agents' Internals

## References


1. Patrick Riley, Peter Stone, and Manuela Veloso (2000) *Layered Disclosure: Revealing Agents' Internals.* In C. Castelfranchi and Y. Lesprance, editors, Intelligent Agents VII. Agent Theories, Architectures, and Languages — 7th. International Workshop, ATAL-2000, Boston, MA, USA, July 7–9, 2000, Proceedings, Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Berlin, 2001.
2. R. Lopes, L. Mota, L. P. Reis, and N. Lau (2010) *Playmaker: Graphical Definition of Formations and Setplays.* In: Information Systems and Technologies (CISTI), pp. 1–6.
3. Patrick MacAlpine and Peter Stone (2016) *Prioritized Role Assignment for Marking.* In Sven Behnke, Daniel D. Lee, Sanem Sariel, and Raymond Sheh, editors, RoboCup 2016: Robot Soccer World Cup XX, Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin.
4. Justin Stoecker, and Ubbo Visser (2012) *RoboViz: Programmable Visualization for Simulated Soccer.* In Röfer, Thomas and Mayer, N. Michael and Savage, Jesus and Saranlı, Uluc (eds.) RoboCup 2011: Robot Soccer World Cup XV, Springer Berlin Heidelberg, pp. 282–293.
5. https://github.com/magmaOffenburg