RoboLog 3D Development Team Description

Heni Ben Amor[1] and Oliver Obst[2]

[1] VR and Multimedia Group, TU Bergakademie Freiberg, Freiberg, Germany, amor@informatik.tu-freiberg.de

[2] School of Electrical Engineering and Computer Science, The University of Newcastle, Callaghan, NSW 2308, Australia, oliver.obst@newcastle.edu.au

Abstract. The Spark physical multiagent simulation system (Simspark) provides a high degree of flexibility for creating new types of simulations. One of the drawbacks, however, is the somewhat limited quality of the visualisation for specific simulations. With the visualisation we aim to present at RoboCup 2007, we hope to address some of the current issues by implementing a new visualisation that (a) is able to render bodies simulated in Simspark simulation in a generic way, and (b) provides some functionality specific to soccer simulations such as the visualisation of tactical information. For the current development, we are using the Mac OS X platform, so that one part of the project is the extension from the current Linux/Windows compatibility to Mac OS X (without giving up the other platforms). To be more platform independent with respect to the visualisation, we are using the open-source graphics engine Ogre3D.

1 Introduction

Originally, the very first parts of Simspark have been implemented on MS Windows, with a high quality visualisation specific to a restricted set of graphic adapters [1]. At that time, the visualisation was directly coupled with the simulation engine, so that it was not possible to connect visualisations from other machines. Later on, when the system was ported to Linux and turned into a more generic simulation system [2], the restrictions of the original graphics engine caused some problems on many machines, so that for the first soccer simulation (using spheres as robots) a simple, simulation specific monitor system was created (*rcssmonitor3D*). This monitor system has the advantage of being simple enough to be compiled and run on virtually any machine, and gets its updates via S-expressions encoded in strings, which can be transferred over the network. With the current attempt to use more complex types of robots for the soccer simulation, this monitor system becomes virtually unusable. Meanwhile, the generic type of visualisation (*monitorspark*) is running on most Linux-boxes (through the effort of some developers of the Simspark development team), but does not provide any soccer specific information such as scores or team names. In the current graphics engine of Simspark, *Kerosin*, some of the interesting features for a visualisation, like 2-D overlays, sky boxes, or different kinds of shadows and lights, have to be programmed manually or are difficult to access.

One additional problem of a networked monitor system we would also like to address with our approach, is the significant amount of traffic the transfer of an entire scene creates.

2 Our Approach

The creation of a simulation system like Simspark, takes a significant effort in programming not only for creating new desired features, but also for maintenance, for instance just to keep up with the ongoing development of operating systems and external libraries. Because of this, we do not believe that it is feasible for the Soccer Simulation community to maintain a high quality graphics engine together with the ongoing development of the soccer simulator (in fact, even the development of the pure simulator requires a high amount of work and seems to be sometimes difficult enough for the community). Instead of extending the current generic graphics engine *Kerosin* of Simspark, we have chosen to use *Ogre3D*, an existing, open-source graphics engine which has been actively developed for six years now and is available for MS Windows, Linux and Mac OS X [3]. The following subsection describes a kind of roadmap for our project, while the first important goals of this roadmap are then described in the subsequent subsections. Until RoboCup 2007, we hope to implement most of the basic steps of our roadmap. Because we are planning to release our visualisation under an open-source license, and extend the platform-independence of Simspark along with the development, we believe that the steps we do not finish until the first release might also be taken up by other members of the Simulation League community, possibly with some features that are not currently on our roadmap. The current development takes place on the Mac OS X platform, but is carried out in a platform independent fashion. However, because of the limited time for the qualification, the current version of of our code is tested on Mac OS X only.

2.1 Roadmap

This subsection contains a brief listing of the roadmap of our project.

1. [**Sect. 2.2**] Simspark Mac OS X port
2. [**Sect. 2.3**] Simple plugin to connect Simspark and Ogre3D
3. [**Sect. 2.4**] Basic 2-D overlay with team name and score information
4. [**Sect. 2.5**] Autonomous camera-agents
5. [**Sect. 2.6**] Tactical soccer information
6. [**Sect. 2.7**] Generic graphics engine bridge
7. [**Sect. 2.8**] Quicktime movies
8. [**Sect. 2.9**] Networking support

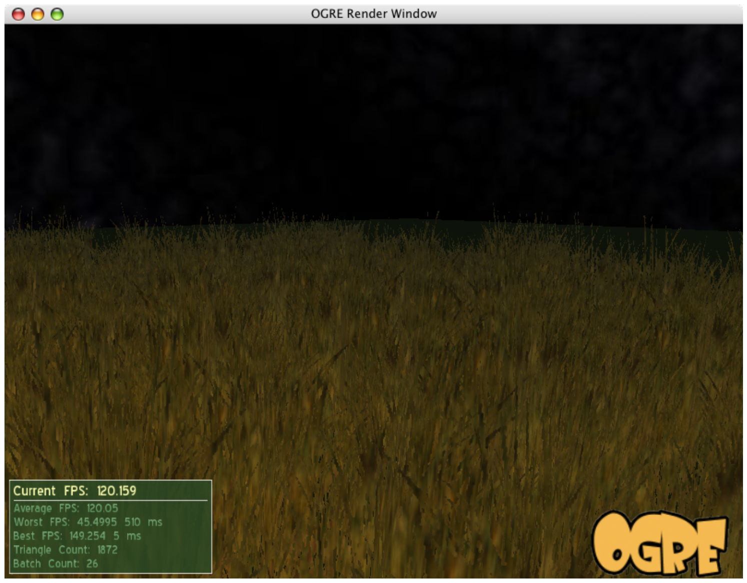
Goals of the single steps are described in the subsequent subsections, as indicated by the square brackets.

2.2 Simspark Mac OS X port

Our current development takes place on the Mac OS X platform. Because of this, a port to Mac OS X became necessary. The port serves also the purpose of making the Simspark simulator available to a wider community and to support an easy installation, as well as an improvement of the source code of Simspark itself.

2.3 Simple plugin to connect Simspark and Ogre3D

As a first step to develop a new monitor system, our development builds on the existing monitor plugins on the Simspark and rcssserver3D repositories. The new plugin will support the visualisation of scenes in the Simspark scene tree, but will contain no networking support, i.e. the simulation and the visualisation run on the same machine. The camera can be moved by pressing keys or by moving the view with the mouse. Features of Ogre3D include already some support for skyboxes, textures and different kinds of lights, so that the appearance of this simple visualisation will already be better than the current visualisation of Simspark.



```
Current FPS: 120.159
Average FPS: 120.05
Worst FPS: 45.4995 510 ms
Best FPS: 149.254 5 ms
Triangle Count: 1872
Batch Count: 26
```

Fig. 1. A plane with grass and skybox in Ogre3D (the grass is animated, i.e. moving). For the final version of our soccer visualisation, we promise to mow the lawn. Honestly.

2.4 Basic 2-D overlay with team name and score information

A simple HUD (Head-Up-Display) is responsible to display score and team information.

2.5 Autonomous camera-agents

The automatic camera is one of the important features for running the monitor autonomously without human interaction. Different cameras, following the ball on pre-defined tracks will provide a (hopefully) enjoyable presentation. The idea is to have the control realising the switch between single cameras as state machine, whereas the single cameras act autonomously according to own rules.

2.6 Tactical soccer information

A further step involves the creation of (soccer simulation league specific) tactical information, which might be useful for both debugging as well as presentation of soccer matches. Some of the ideas we will pursue are the representation of possible passes and distances in dead ball situations.

2.7 Generic graphics engine bridge

In order to follow the general philosophy of Simspark to keep things as generic as possible, a generic bridge in the *Kerosin* library of Simspark will allow for the connection of arbitrary graphics engines to the scene tree of Simspark. This would further support the development of simulation specific visualisations for (possibly different) simulations, as other developers might have other preferences for a graphics engine.

2.8 Quicktime movies

One of the problems of soccer simulation league matches (both 2-D and 3-D) has always been the presentation of matches in talks or on the web. While the 2-D matches can be presented as flash (swf) movies, Quicktime seems more suitable for 3-D presentation (flash/swf does not support 3-D graphics in a straightforward way).

2.9 Networking support

Transmitting a scene via network takes significant resources on both the network itself as well as processing time for serialising all data in a scene. To nevertheless make networked visualisations possible, three steps are necessary: (1) the monitor and the simulator have to work closer together, so that the simulator needs to transmit only data that is about to be presented on the screen, and can omit data that is not required; (2) the protocol between monitor and simulator has to be optimised so that the serialisation and de-serialisation takes only little processing time; (3) the networking itself has to be more efficient than just using TCP (which causes some overhead itself already). We intend to use the RakNet UDP library [4] for this. RakNet is a cross platform C++ network library that has been mostly used for games (but it is application independent and free for non-commercial applications).

3 Current state

The current state of our development is that the port of the core simulator to Mac OS X is finished (see also Fig. 2). We believe that, until RoboCup 2007, we will finish at least steps 1 - 5 of our visualisation, including the support to compile everything on both Mac OS X and Linux.

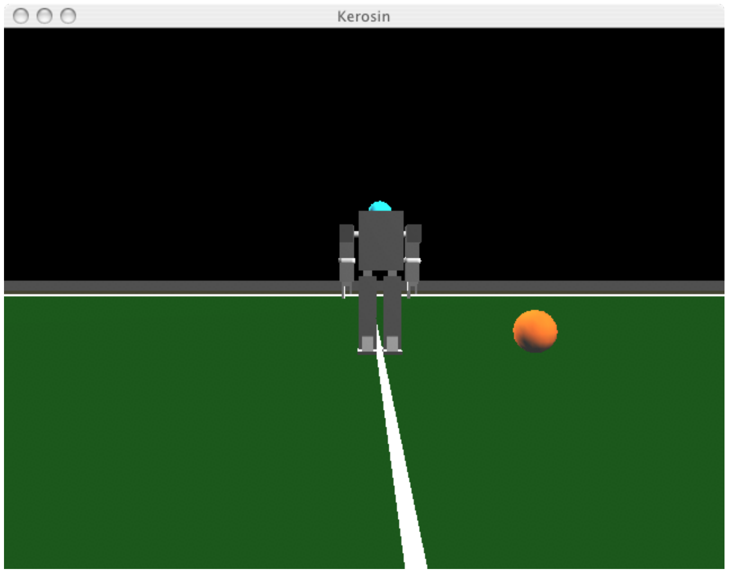


Fig. 2. A screenshot of the current version of the Mac OS X port.

References

1. Marco Kögler. Simulation and Visualization of Agents in 3D Environments. Diplomarbeit, AI Research Group, Universität Koblenz, February 2003.
2. Oliver Obst and Markus Rollmann. SPARK – A Generic Simulator for Physical Multiagent Simulations. *Computer Systems Science and Engineering*, 20(5):347–356, 2005.
3. Ogre3D Web page, March 2007. http://www.ogre3d.org/, last checked 10/03/07.
4. RakNet Web page, March 2007. http://www.rakkarsoft.com/, last checked 10/03/07.