New Soccer Server Initiative

Artur Merke, Oliver Obst, Martin Riedmiller, ...

6 August 2002

1 Main Statement

The current soccer-server is a suited testbed for principle research on noisy, dynamic, multi-agent environments. However, with respect to the goal in 2050, it lacks of important features like 3D, correct physical modelling, flexiblity in sensor and actor modelling.

Therefore, we propose the following

- Freeze the current simulator. Keep it as a testbed for challenging multi-agent research. Idea: if we keep the soccer environment stable as it is, we can really measure scientific progress from year to year. Evaluation will become meaningful. This competition could become 'simulator classic' and can continue for say, the next 10 years. If the RoboCup organization or community decides not to support this any more, also an independent competition could be organized.
- Develop a new simulator from scratch (Features below)

2 Features of the New Simulator

Central Idea: Open-Source Development of a new simulator environment in the following basic steps:

- basic environment, based on correct physical modelling, 3D
- step-wise refinement of sensors and actors
- finally, simulation of humanoid robots

To be suited as a scientific testbed, the simulator should show the basic features:

- easy and reliable communication between clients and server (if noise in communication is desired as a feature of reality, it should be added explicitly)

- flexible time managemement: the simulator can both be accelerated (asynchronous mode) to allow learning and other cycle consuming scientific activities and slowed-down (to allow running on slow machines or via the internet)
- flexible modelling of physical objects and environments
- 3 D

3 Requirements and Directions

- 3D- Monitor
- Simulator Base-System
- Sample Clients
- physical modelling

3.1 Simulator Base System

The simulator base system contains the basic infrastructure for the core simulator of soccer server 3D. The key idea to gain flexibility with respect to the simulated robots, sensors, actuators and environment is to have a small number of extensible base entities by which the entire world model is made up. Base entities required are 3d physical objects, sensors, actuators and a representation of the terrain. Extensibility of soccer server 3D is achieved by using the concept of factories (from object-oriented programming) for each of these entity types and by deriving the needed entities from these entity base classes. A *class server* is used to manage these factory classes; an approach like this and some of the additional modules to gain flexibility below have been described in (1)[1].

3.1.1 Console Subsystem

The console subsystem is used for online configuration of soccer server 3D and for scripting facilities. The console subsystem is also responsible for printing and redirecting (error and warning) messages and uses a *variable subsystem* to store configuration variables.

[1] It's in German, so a summary of the concepts follows

3.1.2 Variable Subsystem

The variable subsystem stores the console variables and provides access to them via a variable server. Within this context, a variable is an object that stores type, value and other attributes of the variable. Console variables also allow referencing other (C++) variables so that they can be used to access variables of other modules of soccer server 3D. The console variable callback mechanism can execute methods within soccer server 3D whenever the value of the respective console variable changes.

3.1.3 Forwarder Subsystem

The forwarder subsystem is used to print and redirect messages intended to be read by humans.

3.1.4 Scene Graph (Tree)

Entities and their spatial relation are stored in a tree structure (scene graph), where each node contains an entity, the pose of the entity with respect to the entity of the parent node and possibly a number of links to child nodes. To represent for instance a simple robot on a soccer field we use a terrain, a cylinder-like physical object, a visual sensor and a kick device (actuator). In the scene graph, the root node contains the terrain entity and one link to a child node, the robot representation. This child node contains the cylinder-like physical object, the relative pose of the physical object wrt the terrain, and two links to child nodes representing the actuator and the sensor of the robot. Each of the child nodes stores one entity again plus the pose of the respective entity wrt the robot. If the physical representation of the robot moves on the terrain, the actuator and the sensor of the robot keep their relative pose without further changes within the scene graph.

3.1.5 World Objects

- entity
  A base class for all entities occurring in the simulated world. All entities have a name and can be stored in the scene graph.

- object3d
  A base class for representation of three dimensional objects in the world; derived from the entity base class. object3d and derived classes describe for instance (FIXME) the shape and the weight of 3d objects (but not the pose of the object in the world).

- sensor
  A base class for sensor devices, derived from the entity base class. A sensor device consists at least of a method that, given a certain pose in the world, creates some data simulating output from a sensor device. A sensor has no spatial extension or

other physical properties like weight by itself. To represent for instance a camera, a 3d physical object entity representing the camera and a sensor entity have to be used together.

- actuator
  A base class with a method creating a force that is applied to 3d physical objects for some time, derived from the entity base class. Actuators have a method for changing their current state, possibly for some time only. Like a sensor, an actuator alone does not have a spatial extension on its own.

- terrain
  A representation of the terrain (i.e. the soccer field), derived from the entity base class. A possible representation of the terrain is for instance a height map, a two dimensional array where each value stands for the height at a certain position.

References

[1] Jens Anhenn, Joschka Bödecker, Alexander Fuchs, Thilo Girmann, Frank Jacob, Marco Kögler, Martin Mack, Michael Nikelsky, Achim Rettinger, Christoph Ringelstein, Markus Rollmann, Timo Wallrath, and Christian Wohlgemuth. Gateway – Eine 3D Engine. Projektpraktikum, July 2002.