Development Architecture of a Humanoid Soccer Simulation Robot Team Description Proposal for Robocup2008 Apollo3D

Junqing Wang, Guoping Luo, Ke Ding, Zhiyong Zhang
Nanjing University of Posts and Telecommunications, China
wjqffff@hotmail.com

Abstract. This document describes the architecture of our humanoid soccer robot for RoboCup 2008 3D Soccer Simulation – Apollo3D. In order to make our research more efficient, we design and implement a solid system architecture, which is based on simspark. The server and the multiple clients are integrated and skills can be written in script.

Keywords: simspark ruby robocup

1 Introduction

The model of robots in RoboCup soccer simulation 3D turns to be humanoid and so many previous developing framework cannot be directly employed in such this league. In a traditional way, we program in C++, and have to edit the source code, compile it, and then test the binary. It is inefficient that clients have to connect to the server each time when we only want to write skills without playmode on, so we design a framework based on simspark [1] to avoid these problems.

2 Related Conceptions

Before introducing our development architecture, some conceptions should be reviewed, which are SimControlNode, TrainControl and Ruby Script.

2.1 SimControlNode

Simspark regards every components of the simulator as separated nodes, such as InputControl, RenderControl, NetClient, NetControl, AgentControl, MonitorControl, MonitorLogger[2].
When a client connects to simspark, it adds an agent in AgentControl, and AgentControl is responsible for the communications between the server and agents.

2.2 TrainControl

We design a TrainControl modular, and it omits the network of AgentControl. In other words, agents are embedded in simspark, without network connection.

2.3 Ruby Script

In simspark, library zeitgeist implements a mechanism[3] and works with classes objects, it provides a 'built-in' server: ScriptServer, which using Ruby script currently.

In general, we code in C++, and we have to do our work with the traditional method. However, compilation is time consuming, with the ScriptServer we can develop skills more efficiently with the new method:

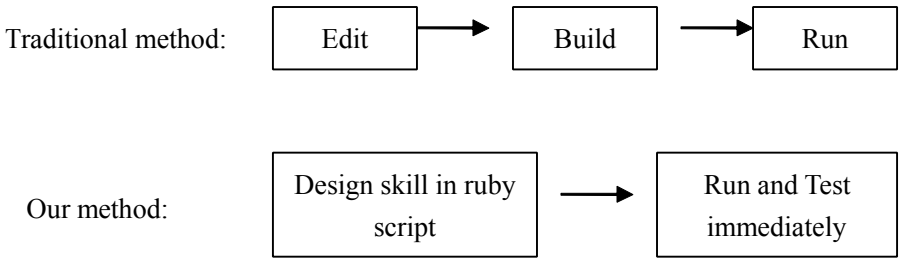

Fig. 1. Progress of developing skills with two methods

Admittedly, although programming in script can reduce the time of adjusting

parameters, it's not easy to debug in script. Nevertheless, we can still convert the ruby scripts into C++ at the last stage.

3 Apollo3D's Development Architecture

There are two modes that be switched in TrainControl:

1. Network Mode.
   An agent connects to rcssserver3d

   StartCycle:
   SenseAgent: network->GetMessage
   ActAgent: behavior->Think, network->SendMessage
   EndCycle:

   In this mode, only one simcontrolnode (TrainControl) is running.

2. Debug Mode.
   An agent connects to the internal server without network

   StartCycle: agent->RealizeActions
   SenseAgent:
   ActAgent: behavior->Think
   EndCycle: agent->QueryPerceptors

   In this mode, we can choose to render, input or logger to run, it's essentially the same as as simspark except that agents communicate without network.

We illustrate a graph of the debug mode as follows:

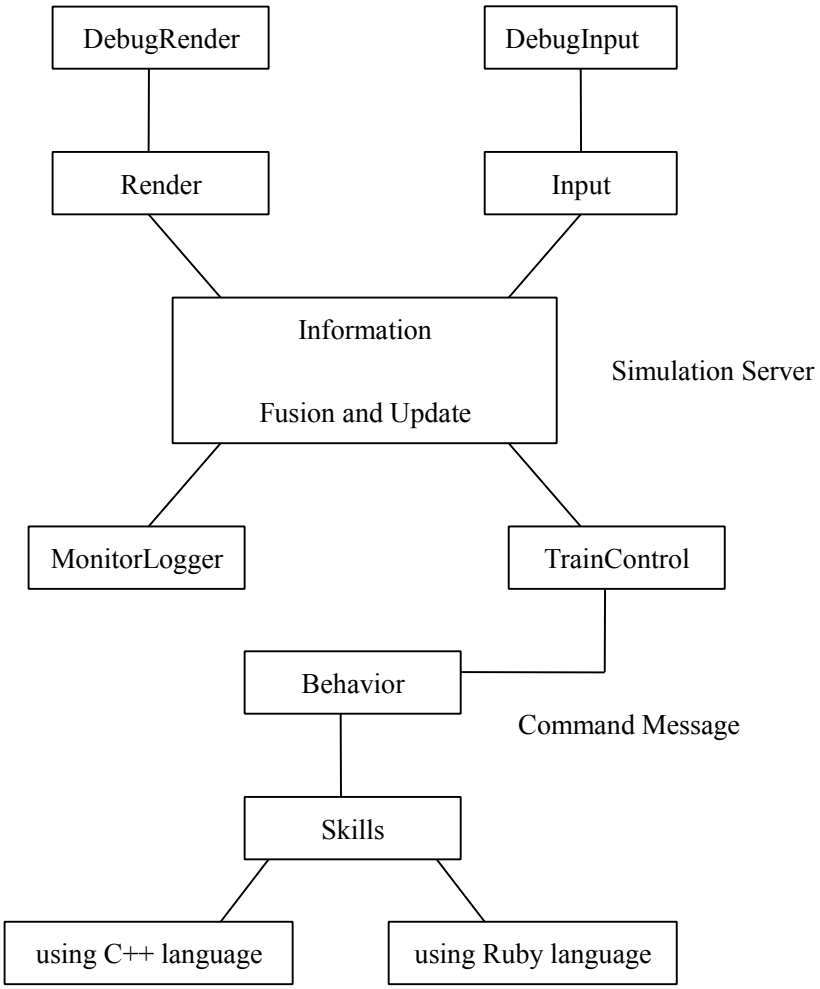

Fig. 2. Architecture Graph in Debug Mode

In the loop of Simulation Server, simcontrolnodes are specific for different purposes, we get two import aspects used:

(1) DebugRender, DebugInput: we can draw our debug information in the screen directly when the simulation is running, and with the usage of debuginput,

reset the position of objects and create a scenario that can be easily achieved at any simulation time.

(2) TrainControl: as described above, it's responsible for communicating with agents' behaviors.

Additionally in the debug mode, we can do some records of dynamic information, for example, the prediction of ball's movement has been successfully achieved using this method.

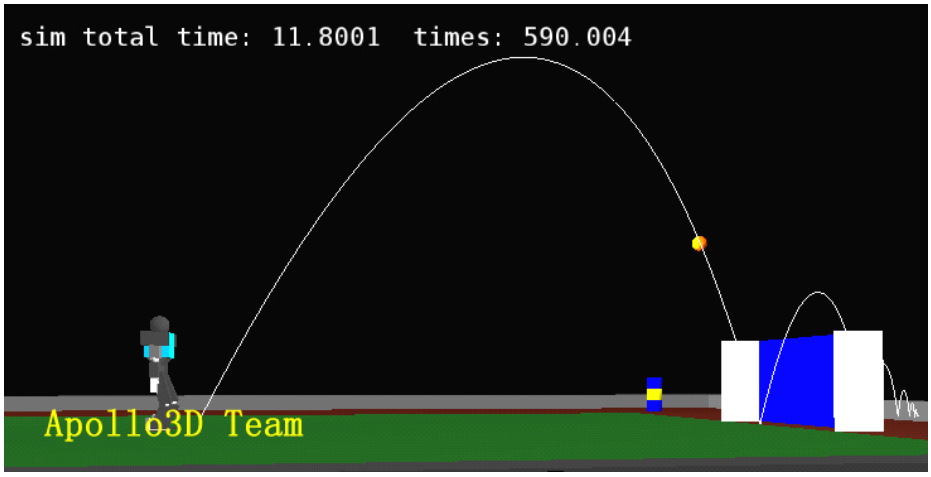

Fig. 3. Prediction of ball's Movement

4 Conclusion

In this paper we present an architecture for developing a humanoid soccer player agent, we'll continue to work out some advanced features, such as setting scenes for creating agents' skills and for cooperation among agents.

References

1. Markus Rollmann, Spark-a generic simulator[D]. Diploma thesis, Koblenz-Landau University, 2004.
2. RoboCup3d server develop group. RoboCup Soccer Server 3D Manual[Z], 2007
3. RoboCup3d server develop group. RoboCup Soccer Server 3D Howtos[Z], 2007