A Universal Tool for Multirobot System Simulation*

Wojciech Turek[1], Robert Marcjan[1], and Krzysztof Cetnarowicz[1]

Institute of Computer Science, AGH University of Science and Technology
wojciech.turek@agh.edu.pl, marcjan@agh.edu.pl, cetnar@agh.edu.pl

Summary. The software tool presented in this paper is an advanced, distributed simulation environment for robot modelling, programming, and control program testing. It offers accurate, three-dimensional simulation of kinematics and dynamics of complex mechanical constructions, and ability of creating control programs in a convenient programming language. The article presents general architecture of the system, its abilities, and the process of distributed simulation, together with some results of experiments and examples of possible applications. The system was developed at the Institute of Computer Science, AGH University of Science and Technology and is used for research on multirobot systems.

1 Introduction

Research into robotics is one of the most interesting challenges of modern science and technology. This wide discipline interconnects number of different issues like material technology, analogue and digital electronics, wireless communication, control software creation, artificial intelligence and many others. Therefore design, building and programming of a new robot are very complex and sophisticated tasks, which usually take years to complete. In addition inevitability of number of prototypes and unforeseen problems during building process makes this kind of research extremely expensive. Some of these issues can be solved by use of simulation tools.

This article describes the RoBOSS Simulation System, which is an advanced software tool for robot modelling, simulation and control programs testing.

2 Applications of Simulation in Robotics

An advanced simulation software could significantly support both building and programming of robots. An accurate model of designed robot could be very

*This work was partially supported by the grant MEiN Nr 3 T11C 038 29

W. Turek et al.: *A Universal Tool for Multirobot System Simulation*, Studies in Computational Intelligence (SCI) **102**, 289–303 (2008)
www.springerlink.com
© Springer-Verlag Berlin Heidelberg 2008

useful in testing construction stability, estimating required actuators abilities and analysing ranges of robot's movement. This sort of test can quickly remove many mistakes from a design.

One of the most important advantages of virtual world is that it can exceed all restrictions of reality. Simulated model of a robot can be built of materials that are very light and resistant, its engines can be extremely powerful, it can also be equipped with fault-proof and perfectly accurate sensors.

Another very important application of robot simulator is control software development. First of all it is far safer to perform tests of control programs in virtual reality because even a small mistake can cause a very serious damages to the robot and its environment. Moreover, created simulation model can be easily simplified or modified to expose every interesting feature of the real robot. The robot can also be placed in any required environment in order to test its abilities.

What is more, many sophisticated control algorithms are very difficult to test on real robots. For instance, genetic algorithms or neuronal networks require long process of learning, that can be performed automatically by a simulation system. In addition time speed in a simulated world can be multiplied in order to reduce total learning time.

One of the most interesting issue concerning robots control algorithms is management of groups of cooperating mobile machines and multiagent systems development [19]. Due to large number of devices required, it is also one of the most expensive disciplines. Therefore usage of proper, high-performance simulator seems to be the only possible solution.

Described applications require the following features of an universal simulation system:

- three-dimensional virtual world
- ability of representing complex mechanical constructions, like mobile and stationary robots
- realistic and reliable simulation of rigid body kinematics and dynamics
- collision detection, friction and bounciness simulation
- support for modelling of typical robots subassemblies (actuators, sensors)
- reasonable units of required parameters and simulation results that allow simple usa of physical and mathematical equations
- convenient programming interface, various languages
- ability of simultaneous simulation of multiply robots
- adjustable accuracy of calculations and high performance

3 Available Tools

There are several software tools, that claim to be robot simulators, but very few of them meet any of the requirements listed above. Despite of careful

research, authors were unable of finding a simulator, that would fulfil all of those.

Commercial applications of assembly arms require CAD tools, that are used in assembly lines design process. One of the most advanced tools of this kind is "EASY-ROB" [3], which offers libraries of robots models manufactured by ABB, KUKA, Bosh and Staubli. There are a few simpler tools for simulation of serial-link manipulators, like a "Robotics Toolbox for MATLAB" developed by Peter Corke [4], or "Robotect" designed by Ophirtech [5]. These tools offer kinematics, dynamics and trajectory generation algorithms, but do not support advanced modelling capabilities needed for mobile robots simulation.

There are many software tools designed to develop control programs for a particular model of robot. For instance, at least two simulators ("Khepera Simulator" [7], "Easybot" [8]) were implemented for the Khepera [6] robot, which is one of the most popular research mobile robot. Obviously, none of these programs can be called universal.

Another group of tools, that are designed for particular kinds of robots, are robot soccer simulators. The idea of robots playing soccer was introduced by professor Jong-Hwan Kim, who founded the Federation of Robot-soccer Association (FIRA [17]) on 1995, and became very popular over the last decade. The most advanced robot-soccer simulators are: MiS20 [9], UberSim [10], SimuroSot Simulator [17] and The RobotCup Soccer Simulator [12].

Very few programs offer functionalities of three-dimensional simulation of kinematics and dynamics for sophisticated mechanical constructions. The most advanced of those seems to be the "Webots 5" [15] simulator. It is a commercial tool based on a open-source dynamics library, supporting modelling of stationary, mobile and even flying robots. It simulates various sensors and effectors and other typical robot subassemblies. It is a suitable tool for robot design and simple control program testing, but it cannot be used for simulation of numerous groups of mobile robots due to performance issues.

Two other tools, that can simulate 3D kinematics and dynamics, are an open-source project Player/Stage/Gazebo [14] and Yobotics Simulation Construction Set [13]. The Yabotics is a commercial program which can simulate only one robot at a time. The Gazebo (3D version of the Player/Stage project) does not support robot modelling.

The lack of an open-source tool, that would be suitable for research into multirobot systems encouraged authors to develop a new, universal robot simulator.

4 RoBOSS Simulation System

RoBOSS is an universal, distributed robot simulation system, that supports modelling and testing of any mechanical structure consisting of rigid bodies. It offers accurate, three-dimensional simulation of kinematics and dynamics

of robots, and ability of creating control programs in convenient programming language. Distribution of necessary calculations guaranties high performance, that is needed during simulation of complex models.

4.1 A Subject of Simulation

A typical robot consist of the following kind of elements:

- rigid parts
- actuators, that connect parts
- sensors

Therefore a main subject of simulation for RoBOSS system is a mathematical model of a group of three-dimensional rigid objects.

Every moveable rigid objects is considered to be a part of a robot. A robot can consist of one or more connected parts, that can change their position and orientation in a virtual world according to rules defined by connections. Each connection can have a number of actuators attached that can force parts to move against each other. A robot can be equipped with a number of various sensors.

Part of a robot has two sets of features. Those which are constant during simulation process can be called part's properties:

- shape and size
- mass
- friction
- bounciness

A group of features which are variable during simulation are called part's state:

- shape and size
- rotation
- linear velocity
- angular velocity

Each actuator must define two parts that are connected. There can be various type of parts connections, differing in a number of blocked degrees of freedom.

Two parts that are not joined, have six degrees of freedom (movement and rotation around each of three axis). For example, a window attached to a wall has only one degree of freedom, which allows it to rotate around its hinges.

Every non-blocked degree of freedom can be equipped with a proper type of engine, that can force parts to move against each other. Engine can be described by maximum force and velocity that is possible to achieve.

Due to wide variety of sensors, that can be used by robots, it is very difficult to define an universal description format of a sensor. The only common feature

is a part that the sensor is connected to. Other properties depend on individual sensor features.

A simulated world is composed of a set of robots and a group of unmoveable rigid shapes, that create a static environment for robots. Environmental shapes are similar to robots parts, that do not have mass. Their state remains constant during simulation process.

A state of a simulated world is a set of robots states. A robots state is a set of its parts states. An initial configuration and one state of simulated world determine a comprehensive information about every simulated object.

Described set of elements and their parameters, that define simulated world (figure 1), is probably insufficient in many cases, therefore it is very important to allow its further development. RoBOSS system stores simulation definition data using the XML language which is a very convenient solution for storing hierarchical data.

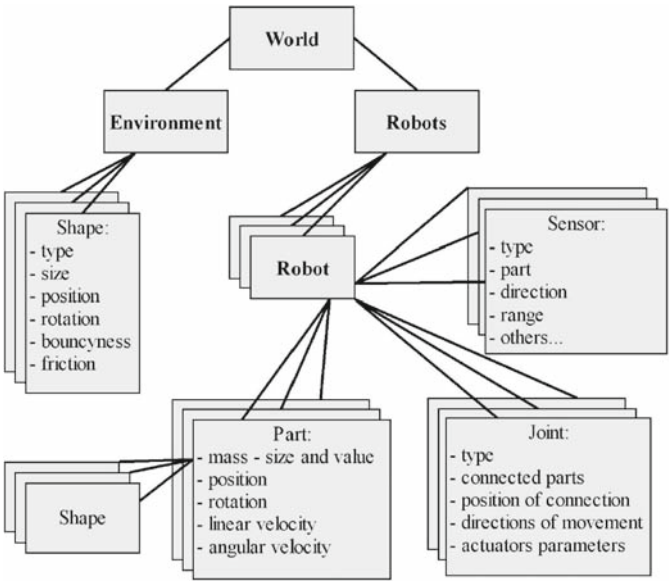

Fig. 1. Simulation subject definition structure

Whole simulated world can be defined by a single tree and stored in a single XML file. There are two basic groups of nodes: robots and environmental objects. Robots definition consists of three types of nodes representing Parts, Joints (connections and actuators) and Sensors. In order to reduce size of the file, every robot definition can be stored in a separated file. This allows user to include many copies of robot without repeating its definition.

Described file format can be easily developed by adding appropriate nodes and attributes to the tree.

4.2 System Architecture

The system is composed of several separated, cooperating programs that communicate using local area network. There are three types of programs: a Controller, Simulators and Clients. It is possible to run all these programs on one computer, however high performance can only be achieved by using a cluster of connected machines. The elements of the system and the connections between them are shown in figure 2.

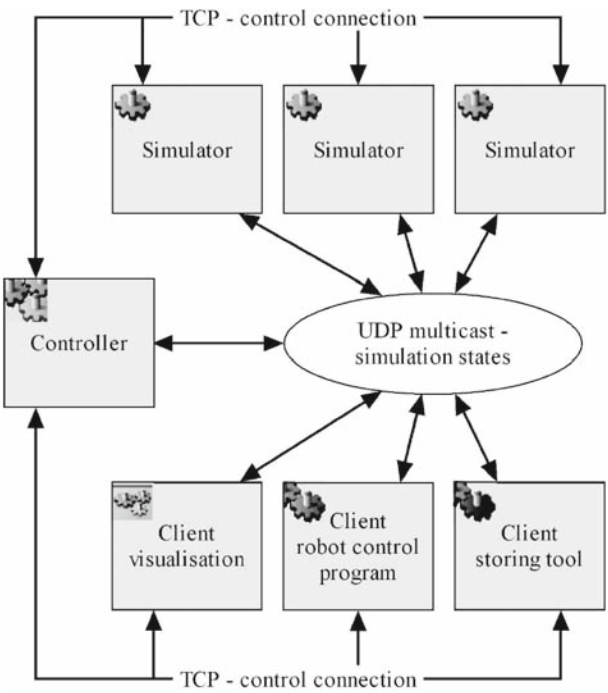

Fig. 2. Architecture of the RoBOSS system

The Controller is the most important element of the system. It is responsible for storing a simulation definition and state, supervising and synchronizing other programs and controlling simulation settings. It allows user to modify many simulation parameters, save or load simulation state or start additional Simulators on remote computers.

Main calculations of simulated robots behaviour are performed by Simulators. Number of Simulators connected to one Controller is not restricted.

The Controller is dynamically distributing computations between available Simulators, so that each of them is equally loaded.

Any other program that is participating in the simulation is considered to be a Client. The most important type of a Client is robot control program, but there can be others, like visualisation or storing tool.

The simulation system is designed to work under the MS Windows operating system. Its components are written in C++ and C#, and reqire .NET Framework 1.1 or newer. The 3D visualisation uses Direct3D technology, it requires DirectX 9.0 libraries.

Communication between programs is based on two network protocols:

- Connection-oriented control protocols
- Connectionless simulation protocol

The first is used for initializing new programs and synchronizing the simulation process. The second is based on multicast UDP protocol and is used to publish a simulation state changes. This solution allows Clients to receive complete and most recent information without additional communication.

4.3 Clients

Programs, that cooperate with the simulation system can be written in any programming language — network interface allows them to communicate with the controller and the simulators. The easiest way of writing a client is to use a communication library, that implements both necessary protocols. The communication library provides high level application programming interface, and can be used by programs written in C++, Managed C++, Visual Basic and C#.

Each client, has access to complete information about the simulation state, therefore tools like visualisation do not require any special protocol or treatment. Clients can alter robots state and behaviour by using a set of orders, which are divided into four levels:

- direct modification of robots position and orientation
- direct modification of robots linear or angular velocity
- adding forces or torques to robots parts
- setting orders for robots actuators

The communication library is also responsible for calculating values returned by sensors. Provided that it has all necessary information, it can successfully perform the task, relieving the Simulators and the network. There are four types of sensors supported:

- laser rangefinder
- odometer
- GPS
- gyrocompass

Implementation of another types of sensors is in progress.

4.4 Simulation Process

The simulation of rigid body system is a continuous process. Each Simulator periodically calculates and publishes state changes of assigned robots that occurred during given, very short time step. Calculation of state changes consists of following steps:

- collision detection, creation of temporary connections between colliding objects
- solving a problem of moving every rigid body

Collision detection algorithm is based on hierarchical detection of bounding objects intersections. Two types of bounding objects, shown in figure 3, are used:

- sphere, that surrounds maximum possible size of robot, regardless of its current state,
- axis aligned box, that is calculated as necessary for a robot and each of its parts.

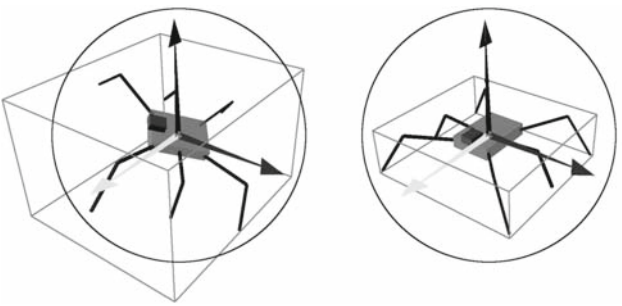

Fig. 3. Axis aligned bounding box and bounding sphere

A radius of the bounding sphere is calculated once, at the beginning of the simulation. It requires analysing of the robot architecture - maximum range of movement of its parts must be estimated. If there is a part in a robot definition, that is not joined to any other part, the radius of the bounding sphere is infinite.

Usage of bounding objects can significantly reduce number of necessary collision tests. Only one comparison is required to determine, if two bounding spheres collide, six comparisons are needed for bounding boxes test. Usually very few pairs of parts remain after this procedure. Shapes associated with those are tested against each other. If a collision is detected, a temporary connection between colliding parts is created, and appropriate reaction forces are added.

Each connection between robots parts (temporary or permanent) is a restriction in a number of degrees of freedom. Every part must be moved according to its velocity and attached forces, without violating any of these restriction, which is a complex computational problem. Main calculations are based on Open Dynamics Engine [16], an open-source library for simulating rigid body dynamics, written by Russell Smith. The library is usually used by games developers, but due to its high accuracy, performance and open-source license it can be successfully used in robot simulation.

All modified states of robots are published using the simulation protocol. Afterwords all orders received from the Controller and Clients are carried out, and another simulation cycle is being started.

4.5 Distribution of Computations

Fast simulation of a numerous group of complex robots exceeds abilities of a single computer. To overcome this problem RoBOSS simulator can perform parallel computations using a cluster of computers connected with a local area network.

Used simulation algorithm requires simultaneous calculations of all temporarily or permanently connected parts. It is safe to assume, that all parts of a single robot are always colliding (they are connected directly or indirectly), therefore a single robot is considered to be an unbreakable unit of computations. The Controller is dynamically distributing simulated robots among Simulators using an algorithm that has two aims:

- balancing of Simulators load
- reduction of duplicated computations

Duplication of computations occurs when colliding robots are assigned to different Simulators. This situation forces both Simulators to calculate state changes of both robots. The algorithm of robots distribution is based on analysis of their position and velocity and prediction of possible collisions. The Controller dynamically divides robots into potentially colliding groups (called "islands"), estimates a computational cost of each island and distributes these island among available Simulators. Performance of computers in the cluster is also taken under consideration by measuring time of computations and confronting it with the estimated cost.

This approach gives the best results for groups of robots that are distributed in a large environment and do not create numerous islands. Maximum number of Simulators that can be efficiently used is restricted by the number of islands - when all robots are colliding (directly or indirectly) the are all assigned to one Simulator. This is the most pessimistic situation, which should not occur, provided that robots are designed to avoid collisions. Even if the pessimistic situation takes place, total performance of the cluster is not worse than the performance of a single computer. In an optimistic situation,

time needed by a cluster of n computers for a single simulation step equals $c + \frac{t_1}{n}$, where c is a constant time needed for collision detection and state publication, and t_1 is a time needed by a single Simulator to calculate a state changes in a single simulation step.

4.6 Performance Tests

Number of tests have been carried out in order to verify efficiency of described method of computations distribution. A robot used was a simple model of a car built of four wheels and a body. A controlling algorithm was based on a range-finder sensors that provided information about the closest objects. It allowed the robot to manoeuvre between static obstacles without any collisions.

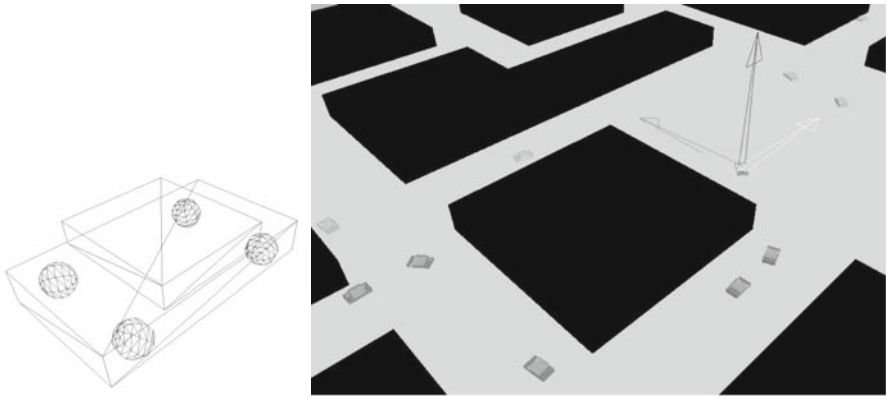

Fig. 4. Model of a car and its environment

A group of fifty robots was situated in a model of a small city, shown in figure 4. Robots were driving around, causing a random number of collisions. The simulation system was running on a cluster of fourteen computers, twelve of them were used for calculations, one for visualization and one for Controller program. The number of islands in the simulated model was very close to the number of robots and much bigger than the number of simulators, therefore each simulator was approximately equally loaded. During the experiment one simulator was detached from the system every 20 seconds.

Table 1. Efficiency of calculations distribution.

Number of simulators	1	2	3	4	5	6	7	8	9	10	11	12
Efficiency of distribution	1	0.95	0.94	0.94	0.73	0.83	0.82	0.67	0.77	0.73	0.71	0.65

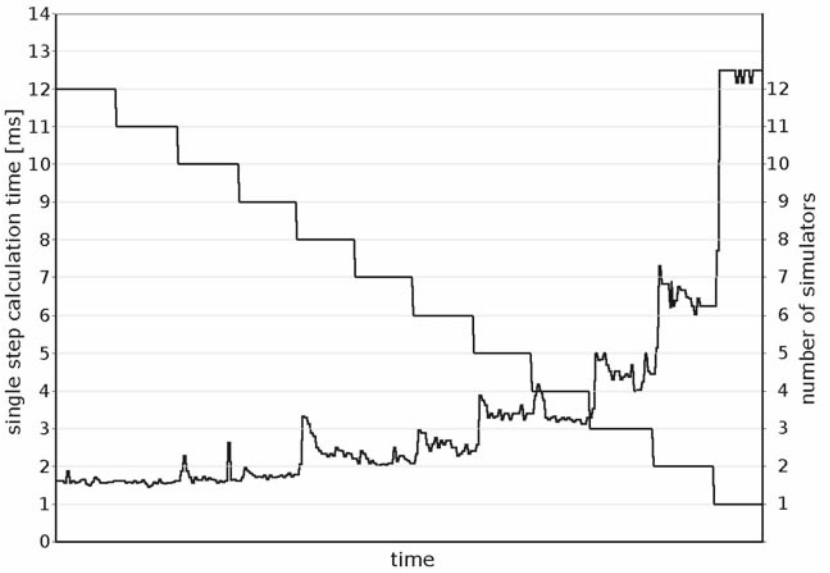

Fig. 5. Relation between number of simulators and the single step calculation time.

The diagram presented in figure 5 shows the relation between the number of simulators connected to the Controller and the time required to calculate a single step of simulation. The peaks of step calculation time, that occur after a simulator is disconnected, show that the time needed by the system to recalculate distribution parameters is very short.

Efficiency of calculations distribution is quite satisfactory - use of a cluster of computers can significantly increase general performance of the simulation system.

5 Examples of Applications

RoBOSS simulation system is capable of simulating behaviour of almost every mechanical construction. Following examples show applications of the system in design and testing of some typical types of robots.

5.1 Industrial Arms

Industrial arms are most common commercial application of robots [2] [18]. They are used in most modern assembly lines, due to reliability, precision, speed and repeatability. These are also the most important parameters that describe abilities of an arm.

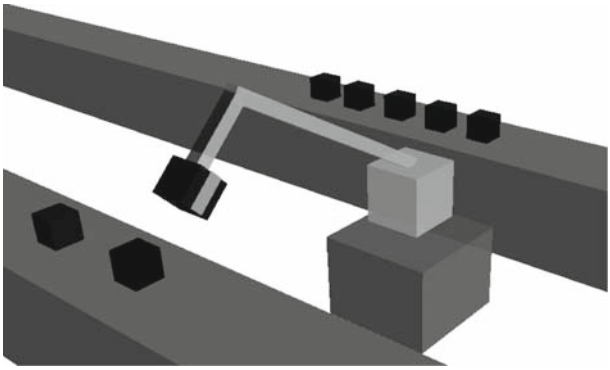

Fig. 6. Model of four-degrees-of-freedom industrial arm

RoBOSS simulator is a suitable tool for design and programming of complex industrial arms. It can be used to estimate necessary number of degrees of freedom, required sizes of parts and powers of actuators. Moreover it allows user to design whole assembly line, predict required space and avoid collisions between arms.

Figure 6 shows a four-degrees-of-freedom arm equipped with a simple gripper. Its task is to move boxes from one conveyor belt to another. The diagram presents changes in position of a gripper controlled by a simple, sensor-based algorithm.

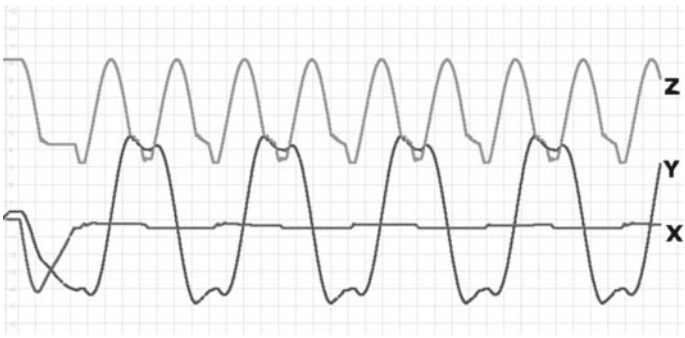

Fig. 7. Tests of accuracy and repeatability

The diagram in figure 7 was created by RoBOSS Analyser program. It can be used to estimate both precision and repeatability of the arm. After a few experiments with control program it was possible to determine all properties and abilities of the robot.

5.2 FIRA MiroSot Robots

Another example of RoBOSS system application is mobile robots modelling. Devices, that were used during experiments are FIRA MiroSot soccer-playing, differential drive robots [1] [17]. These semi-autonomous vehicles are equipped with two engines, a battery, a programmable microcontroller, and a radio receiver. They are only 7.5 centimetres long, and their maximum speed can reach 5 m/s.

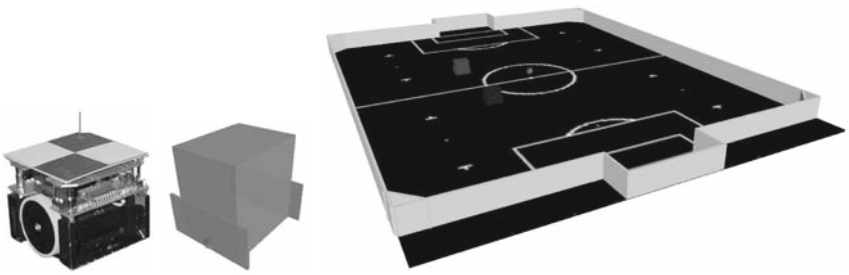

Fig. 8. FIRA MiroSot robot [17], its model, and a simulated pitch

Two values had to be calibrated in order to achieve realistic behaviour of simulated robots:

- power of robots engines
- friction between wheels and a pitch

A number of tests have been carried out and the results show that the difference between simulation and real robots behaviour are similar to those between two real robots. Final version of the model is very useful for control programs development - several groups can work independently because constant access to hardware is no required. Moreover, there is no risk of damaging the robots by using an untested program.

5.3 Spider Robot

One of the most interesting types of mobile robots are legged machines. This type of propulsion is very complicated in design and assembly, but also very flexible - legged creatures can reach places that are inaccessible for wheeled machines. Research into this matter are very popular, but also very expensive. The simulator can be very useful during the design process and may help to avoid some costly mistakes.

The following example presents a model of a Spider robot - six-leg walking machine. The aim of this model is to estimate required powers of actuators and

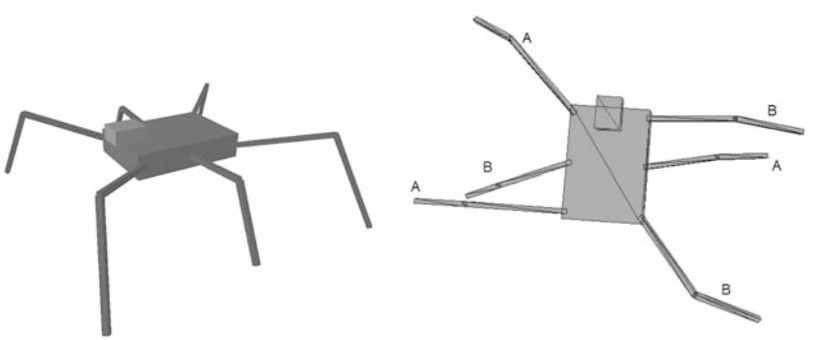

Fig. 9. Spider robot

create a control program, that will allow the spider to move fast and stable. Defined model is 40cm long, each of its legs has three degrees of freedom.

During the movement, the spider stands on three of his legs. When legs marked as A are raised and moved forward, legs B are held low and moved backward. After reaching desired position, functions of legs are switched - spider stands on legs A and moves legs B forward. This simple method required a number of tests to reach an optimal coordination between legs, that would result in a stable movement of the spider. The final effect is quite impressive - the robot is able to move with maximum speed of $2\frac{m}{s}$, when required angular velocity of his legs is smaller than $\frac{0.8\pi}{s}$.

Unfortunately, due to high powers of effectors used in this model, it is impossible to build a real robot - sources of energy and engines would exceed the mass and the size assumed.

6 Conclusions and Further Work

Described simulation system can be a very useful tool in robot design and prototyping. It boost a process of new robot creation and significantly reduces required resources. The RoBOSS system simulates 3D kinematics and dynamics of complex, rigid-body based constructions, supports many types of sensors and actuators and provides a convenient programming interface.

Unlike the other advanced 3D robot simulators, RoBOSS can offer high performance due to distribution of computations and adjustable accuracy of computations. Therefore it can be used in research into complex multirobot systems.

The system is being continuously improved. A camera sensor, simulated radio communication and a new, fully 3D visualization will be added soon. Support tools for storing and analysis of performed simulations will be developed. Moreover new types of real robots controllers will be implemented in

the communication library to allow transparent switching between simulation and hardware.

References

1. Kim JH, Vadakkepat P (2000) Multi-agent systems: A survey from the robot soccer perspective. Journal of Intelligent Automation and Soft Computing, vol. 6, r 1
2. Arkin R (1998) Behavior-Based Robotics, MIT Press, ISBN 0-262-01165-4
3. Anton S (2006) EASY-ROB simulator documentation, http://www.easy-rob.de/
4. Corke P (1996) A Robotics Toolbox for MATLAB. IEEE Robotics and Automation Magazine, vol. 3, nr 1, pp.24–32
5. Ophirtech (2005) EASY-ROB simulator documentation, http://www.ophirtech.com/
6. K-Team Corporation (2006) Khepera II robot specification, http://k-team.com/robots/khepera/
7. Michel O (2005) Khepera Simulator documentation, http://diwww.epfl.ch/lami/team/michel/khep-sim/
8. Iwe H (2005) Easybot simulator documentation, http://easybot.htw-dresden.de/
9. Fikkert W, Dollen H (2005) MiS20 simulator documentation, http://hmi.ewi.utwente.nl/MiS20/
10. Weloso M, Browning B, Go J (2005) UberSim simulator documentation, http://www.cs.cmu.edu/~robosoccer/ubersim/
11. Iwe H (2005) Easybot simulator documentation, http://easybot.htw-dresden.de/
12. The Robocup Soccer Simulator Maintenance Group (2005) RobotCup Soccer Simulator documentation, http://sserver.sourceforge.net/
13. Pratt J, Krupp B, Paluska D, Morse C (2005) Yabotics simulator documentation, http://yobotics.com/simulation/simulation.htm
14. Gerkey B, Vaughan R, Howard A (2003) The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems. Proceedings of the International Conference on Advanced Robotics (ICAR 2003) Coimbra, Portugal, June 30 – July 3, pp. 317–323
15. Michel O (2005) Webots simulator documentation, http://cyberboticspc1.epfl.ch/cdrom/common/doc/webots/reference/reference.pdf
16. Smith R (2005) Open Dynamics Engine, http://ode.org/ , http://ode.org/ode-latest-userguide.html
17. Federation of International Robot-soccer Association (2005) Specification of soccer-playing mobile robots, http://www.fira.net/
18. Murphy R (2000) An Introduction to AI Robotics, MIT Press, ISBN 0-262-13383-0
19. Browning B, Bruce J, Bowling M, Veloso M (2005) STP — Skills, tactics and plays for multi-robot control in adversarial environments. IEEE Journal of Control and Systems Engineering, nr I1, pp. 33–52