Survey of Popular Robotics Simulators, Frameworks, and Toolkits

Adam Harris
Department of Electrical and Computer Engineering and
Department of Software and Information Systems
University of North Carolina at Charlotte
Charlotte, NC

James M. Conrad
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
Charlotte, NC
jmconrad@uncc.edu

***Abstract*— Robotic simulators, frameworks, and related toolkits are very important in today's research community. The need to accurately simulate movements, algorithms, and interactions with the real world is quickly becoming a major research focus as humans and robots interact in more situations and differently than ever before. This is a survey of popular robotics simulators and some of the main frameworks and toolkits that are used to help bring robotics simulations to a one-to-one relationship with real-world interaction.**

I. INTRODUCTION

Robotics simulators have grown with the field of robotics in general. Since the beginning of the robotics revolution there has been a need to simulate the motions and reactions to stimuli of different machines. Simulations are conducted in many cases to test safety protocols, to determine calibration techniques, and to test new sensors or equipment among other things. Robotics simulators in the past were generally written specifically for a company's own line of robots or for a specific purpose. In recent years, however, the improvement of physics engines and video game engines as well as computer processing speed has helped spur a new breed of simulators that combine physics calculations and accurate graphical representations of a robot in a simulation. These simulators have the potential flexibility to simulate any type of robot. As the simulation algorithms and graphical capabilities the game engines become better and more efficient, simulations step out of the computer and become more real-world. Such engines can be applied to the creation of a simulator, and in the words of Craighead et al. "...it is no longer necessary to build a robotic simulator from the ground up." [1].

Robotics simulators are no longer in a class of their own. To ensure code portability from the simulator to real world robotic platforms (as is generally the ultimate goal) specific middleware is often run on the platforms. The performance, function and internals of this middleware must be taken into account when comparing the simulators. All of these factors (and many more) affect the fidelity simulation.

The ultimate goal of this survey is to compare some of the most popular simulators and toolkits currently available (both open source and commercial) to attempt to find one that can easily be used to simulate low-level, simple custom robotic hardware with both graphical and physical high fidelity. There is a lot of previous work in this field. This paper will add to the work of Craighead, Murphy, Burke, and Goldiez [1]. Out of the many simulators available, the particular robotics simulators described and compared in this survey are just a few that were chosen based on their wide use and specific feature sets. Each simulator being compared has one or more of the following qualities:

- Variety of hardware that can be simulated (both sensors and robots)
- Graphical simulation accuracy
- Physical simulation accuracy
- Cross-platform capabilities
- Openness of source code (for future development and addition of new or custom simulations by the user).

II. OPEN SOURCE SIMULATORS AND TOOLKITS

A. Player Project

Started in 1999 at the University of Southern California, the Player Project [2] is an open source (GPL License) three-component system involving a hardware network server (Player); a two-dimensional simulator of multiple robots, sensors or objects in a bit-mapped environment (Stage); and multi-robot simulator for simple three-dimensional outdoor environments (Gazebo). [3] Player is a Transmission Control Protocol (TCP) socket enabled middleware that is installed on the robotic platform. This middleware creates an abstraction layer on top of the hardware of the platform, allowing portability of code [4]. Being socketed allows the use of many programming languages [5]. While this may ease programming portability between platforms it adds several layers of complexity to any robot hardware design.

To support the socketed protocol, drivers and interfaces must be written to interact with each piece of hardware or algorithm. Each type of sensor has a specific protocol called an interface which defines how it must communicate to the driver [6]. A driver must be written for Player to be able to connect to the sensor using file abstraction methods similar to POSTIX systems. The robotic platform itself must be capable of running a small POSTIX operating system to support the hardware server application [3]. This is overkill for many introductory robotics projects, and its focus is more on higher level aspects of robotic control and users with a larger

978-1-61284-738-2/11/$26.00 ©2011 IEEE

budgets. The creators of Player admit that it is not fitting for all robot designs [5].

Player currently supports more than ten robotic platforms as well as 25 different hardware sensors. Custom drivers and interfaces can be developed for new sensors and hardware. The current array of platforms and sensor hardware available for Player robots can be seen on the Player project's supported hardware webpage [7].

Stage is a 2-dimensional robot simulator mainly designed for interior spaces. It can be used as a standalone application, a C++ library, or a plug-in for Player. The strength of Stage is that it focuses on being "efficient and configurable rather than highly accurate." [8]. Stage was designed for simulating large groups or swarms of robots. The graphical capability is also quite basic. Sensors in Stage communicate exactly the same as real hardware, allowing the exact same code to be used for simulation as the actual hardware [5]. This is no guarantee, however, that the simulations have high physical simulation fidelity [8].

Gazebo is a three-dimensional robotics simulator designed for smaller populations of robots (less than ten) and simulates with higher fidelity than Stage [9]. Gazebo was designed to model three-dimensional outdoor as well as indoor environments [5]. The use of plug-ins expands the capabilities of Gazebo to include abilities such as dynamic loading of custom models and the use of stereo camera sensors [10]. Gazebo uses the Open Dynamics Engine (ODE) which provides high fidelity physics simulation [11] It also has the ability to use the Bullet Physics engine [12].

B. USARSim (Unified System for Automation and Robotics Simulation)

Originally developed in 2002 at Carnegie Mellon University, USARSim (Unified System for Automation and Robotics Simulation) [13] is a free simulator based on the cross platform Unreal Engine 2.0. It was handed over to the National Institute of Standards & technology (NIST) in 2005 and was released under the GPL license [14]. USARSim is actually a set of add-ons to the Unreal Engine, so users must own a copy of this software to be able to use the simulator. A license for the game engine usually costs around $40 US [15]. Physics are simulated using the Karma physics engine which is built into the Unreal engine [16]. This provides basic physics simulations [1]. One strength of using the Unreal engine is the built in networking capability. Because of this, robots can be controlled by any language supporting TCP sockets [17].

While USARSim is based on a cross platform engine, the user manual only fully explains how to install it on a Windows or Linux machine. A Mac OS installation procedure is not described. The installation requires Unreal Tournament 2004 (UT2004) as well as a patch. After this base is installed, USARSim components can be installed. On both Windows and Linux platforms, the installation is rather complicated and requires many files and directories to be moved or deleted by hand. The USARSim wiki has installation instructions [18]. Linux instructions were found on the USARSim forum at sourceforge.net [19]. Since it is an add-on to the Unreal tournament package, the overall size of the installation is rather large, especially on Windows machines.

USARSim comes with several detailed models of robots available for use in simulations [20], however it is possible to create custom robot components in external three-dimensional modeling software and specify physical attributes of the components once they are load into the simulator [21]. An incomplete tutorial on how to create and import a model from three-dimensional Studio Max is included in the source download. Once robots are created and loaded, they can be programmed using TCP sockets [22]. Several simulation environments are also available. Environments can be created or modified by using tools that are part of the Unreal Engine [21].

There have been a multitude of studies designing methods for validating the physics and sensor simulations of USARSim. Pepper et al. [23] identified methods that would help bring the physics simulations closer to real-world robotic platforms by creating multiple test environments in the simulator as well as in the lab and testing real robotic platforms against the simulations. The physics of the simulations were then modified and tested again and again until more accurate simulations resulted. Balaguer et al. built on the previous work of validating simulated components by testing virtual sensors against real-world sensors. A method for creating and testing a virtual GPS sensor that much more closely simulates a real GPS sensor was created [24]. Wireless inter-robot communication and vision systems have been designed and validated as well [20]. USARSim has even been validated to simulate aspects of other worlds. Birk et al. used USARSim with algorithms already shown to work in the real world as well as real-world data from Mars exploration missions to validate a robot simulation of another planet [25].

C. SARGE (Search and Rescue game Engine)

Figure 1. Screenshot of SARGE helocopter simulation (from the camera on the robot)

SARGE (Search and Rescue game Engine) [26] is a simulator designed to train law enforcement in using robotics in search and rescue operations [27]. It is released under the Apache License V2.0. A screen shot can be seen in Fig. 1. The developers of SARGE provided evidence that a valid robotics simulator could be written entirely in a game engine [1]. Unity was chosen as the game engine it was less buggy than the Unreal engine and because it proved a better option for physics simulations, PhysX. PhysX provides a higher level of fidelity in physics simulations [28]. SARGE currently only supports Windows and Mac platforms though it is still under active development. Currently, a webplayer

version of the simulator is available on the website http://www.sargegames.com.

It is possible for SARGE users to create their own robots and terrains with the use of external three-dimensional modeling software. Sensors are limited to LIDAR, three-dimensional camera, compass, GPS, odometer, inertial measuring unit (IMU), and standard camera [27], though only the GPS, LIDAR, compass and IMU are discussed in the user manual [29]. The GPS system requires an initial offset of the simulated terrain provided by Google Earth. The terrains themselves can be generated in the Unity development environment by manually placing three-dimensional models of building and other structures on images of real terrain from Google Earth [29]. Once a point in the virtual terrain is referenced to a GPS coordinate from Google Earth, the GPS sensor can be used [28]. This shows that while terrains and robots can be created in SARGE itself, external programs may be needed to set up a full simulation.

D. ROS (Robot Operating System)

ROS [30] is currently one of the most popular robotics toolkits systems. Only UNIX-based platforms are officially supported (including Mac OS X) [31] but the company Robotics Equipment Corporation has ported it to Windows [32]. ROS is fully open source and uses the BSD license [33]. This allows users to take part in the development of the system, which is why it has gained wide use. In its meteoric rise in popularity over the last three years it has added over 1643 packages and 52 code repositories since it was released [34].

One of the strengths of ROS is that it plays nicely with other robotics simulators and middleware. It has been successfully used with Player, YARP, Orcos, URBI, OpenRAVE, and IPC [35]. Another strength of ROS is that it can incorporate many commonly used libraries for specific tasks instead of having to have its own custom libraries [33]. For instance, the ability to easily incorporate OpenCV has helped make ROS a better option than some other tools. Many libraries from the Player project are also being used in certain aspects of ROS [4]. An additional example of ROS working well with other frameworks is the use of the Gazebo simulator.

ROS is designed to be a partially real-time system. This is due to the fact that the robotics platforms it is designed to be used with like the PR2 will be in different situations involving more human computer interaction in real time than many current commercial research robotics platforms [4]. One of the main platforms used for the development of ROS is the PR2 robot from Willow Garage. The aim of using ROS's real-time framework with this robot is to help guide safe Human-Robot Interaction (HRI). Previous frameworks such as Player were rarely designed with this aspect in mind.

E. UberSim

UberSim [36] is an open source (under GPL license) simulator based on the ODE physics engine and uses OpenGL for screen graphics [37]. It was created in 2000 at Carnegie Mellon University specifically with a focus on small robots in a robot soccer simulation. The early focus of the simulator was the CMDragons RoboCup teams; however the ultimate goal was to develop a simulator for many types and sizes of robotics platforms [38]. Since 2007, it no longer seems to be under active development.

F. EyeSim

EyeSim [39] began as a two-dimensional simulator for the EyeBot robotics platform in 2002 [40]. The EyeBot platform uses RoBIOS (Robot BIOS) library of functions. These functions are simulated in the EyeSim simulator. Test environments could be created easily by loading text files with one of two formats, either Wall format or Maze format. Wall format simply uses four values to represent the starting and stopping point of a wall in X,Y coordinates (i.e. x1 y1 x2 y2). Maze format literally is a format in which a maze is literally drawn in a text file by using the pipe and underscore (i.e. | and _) as well as other characters. [40]

In 2002, the EyeSim simulator had graduated to a three-dimensional simulator that uses OpenGL for rendering and loads OpenInventor files for robot models. The GUI was written using FLTK [41] Test environments were still described by a set of two dimensional points as they have no width and have equal heights [42].

Simulating the Eyebot is the extent of this project. While different three-dimensional models of robots can be imported, and different drive-types (such as omni-directional wheels and Ackermann steering) can be selected, the controller will always be based on the EyeBot controller and use RoBIOS libraries [41]. This means simulated robots will always be coded in C code. The dynamics simulation is very simple and does not use a physics engine. Only basic rigid body calculations are used [42].

G. SubSim

SubSim [42] is a simulator for Autonomous Underwater Vehicles (AUVs) developed using the EyeBot controller. It was developed in 2004 for the University of Western Australia in Perth [42]. SubSim uses the Newton Dynamics physics engine as well as Physics Abstraction Layer (PAL) to calculate the physics of being underwater [1].

Models of different robotic vehicles are can be imported from Milkshape3D files [44]. Programming of the robot is done by using either C or C++ for lower-level programming, or a language plug-ins. Currently the only language plug-in is the EyeBot plug-in. More plug-ins are planned but have yet to materialize [44].

H. OpenRAVE (Open Robotics and Animation Virtual Environment)

OpenRAVE [45] is an open source (Lesser GPL) software architecture developed at Carnegie Mellon University [46]. It is mainly used for planning and simulations of grasping and grasper manipulations as well as humanoid robots. It is used to provide planning and simulation capabilities to other robotics frameworks such as Player and ROS [46]. Support for OpenRAVE was an early objective for the ROS team due to its planning capabilities and openness of code [47].

One advantage to using OpenRAVE is its plug-in system. Basically everything connects to openRAVE by plug-ins, whether it is a controller, a planner, external simulation engines and even actual robotic hardware. The plug-ins are loaded dynamically. Several scripting languages are supported such as Pythos and Matlab/Octave [48].

I. RT Middleware

RT Middleware [49] is set of a standards used to describe a robotics framework. The implementation of these standards is OpenRTM-aist, which is similar to ROS. This is released under the Eclipse Public License (EPL) [50]. Currently it is available for Linux and Windows machines and can be programmed using C++, Python and Java [49]. The first version of OpenRTM-aist (version 0.2) was released in 2005 and since then its popularity has grown. Version 1.0 of the framework was released in 2010.

OpenRTM-aist is popular in Japan, where a lot of research related to robotics takes place. While it does not provide a simulator of its own, work has been done to allow compatibility with the Player project [51].

J. MRPT (Mobile Robot Programming Toolkit)

The Mobile Robot Programming Toolkit (MRPT) [52,53] project is a set of cross platform C++ library and applications released under the GPL license. It is cross platform, but has only currently has only been tested on Windows and Linux.

MRPT is not a simulator or framework; rather it is a toolkit that provides libraries and applications that can allow multiple third-party libraries to work together. [MPRT Book] The main focus of MRPT is Simultaneous Localization and Mapping (SLAM), computer vision, and motion planning algorithms [54].

K. lpzrobots

lpzrobots [55] is a GPL licensed package of robotics simulation tools available for Linux and Mac OS. The main simulator of this project that corresponds with others in this survey is ode_robots which is a three-dimensional simulator that used the ODE and OSG (OpenScreenGraph) engines.

L. SimRobot

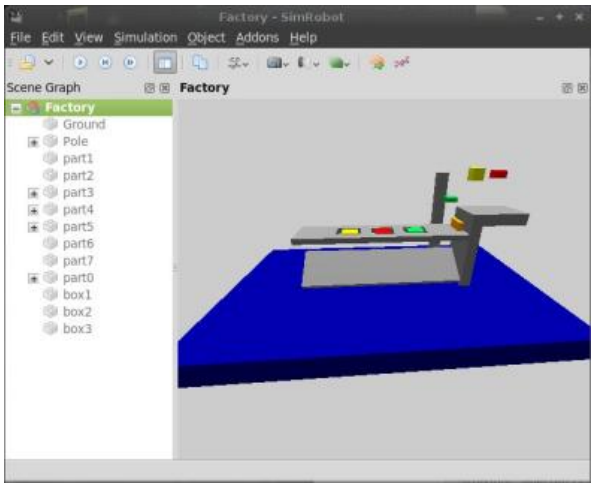

Figure 2. Screenshot of SimRobot.

SimRobot [56] (shown in Fig. 2) is a free, open source completely cross-platform robotics simulator started in 1994. It uses the ODE for physics simulations and OpenGL for graphics [57]. It is mainly used for RoboCup simulations, but it is not limited to this purpose.

A simple and intuitive drag and drop interface allows custom items to be added to scenes. Custom robots can be created and added as well [58].

Unlike many of the other robotics simulators, SimRobot is not designed around client/server interaction. This allows simulations to be paused or stepped through which is a great help to debugging simulations [58].

SimRobot does not simulate specific sensors as many of the other simulators do; rather it only provides generic sensors that users can customize. These include a camera, distance senor (not specific on a type), a "bumper" for simulating a touch sensor, and "actuator state" which returns angles of joints and velocities of motors [58].

Laue and R¨fer admit that there is a "reality gap" in which simulations differ from real-world situations [57]. They note that code developed in the simulator may not translate to real robots due to distortion and noise in the real-world sensors. They also note, however, that code that works in the real world may completely fail when entered into the simulation because it may rely on that distortion and noise. This was specifically noted with the camera sensor and they suggested several methods to compensate for this difference. [57]

M. Moby

Moby [59] is an open source (GPL 2.0 license) rigid body simulation library written in C++. It supports Linux and Mac OS X only. There is little documentation for this simulation library.

III. COMMERCIAL ROBOTICS SIMULATORS

There are many commercial robotics simulators available. Many of them are designed for industrial robotics or a manufacturer's own robotic platforms. The commercial simulators described and compared in this paper will be focused on research and education.

As with any commercial application, one downfall of all of these applications is that they are not open source. Commercial programs that do not release source code can tie the hands of the researcher, forcing them in some cases to choose the less than optimal answer to various research questions. When problems occur with proprietary software, there is no way for the researcher to fix it. This problem alone was actually the impetus for the Player Project [4].

A. Microsoft Robotics Developer Studio (MRDS)

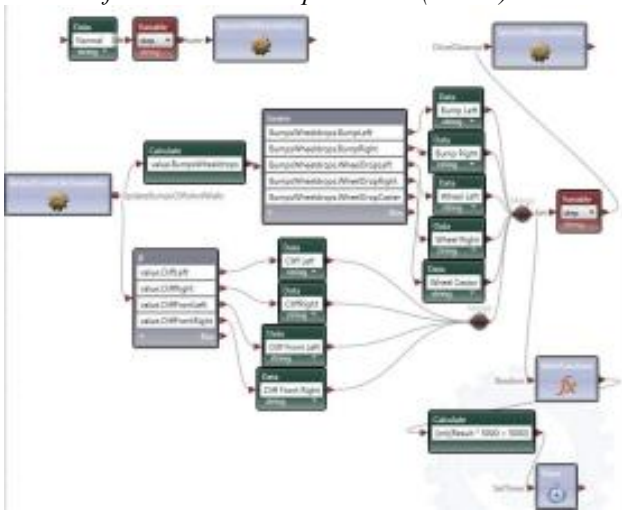

Figure 3. Screenshot of a Microsoft Robotics Developer Studio's Visual Programming Language (VPN) program

Microsoft Robotics Developer Studio (MRDS) [60] uses Phys X physics engine which is one of the highest fidelity physics engines available [1]. A screenshot can be seen in Fig. 3. MRDS robots can be programmed in .NET languages as well as others. The majority of tutorials available online mention the use of C# as well as a Visual Programming Language (VPL) Microsoft developed. Programs written in VPL can be converted into C# [61]. The graphics are high fidelity. There is a good variety of robotics platforms as well as sensors to choose from.

Being a Microsoft product, it is certainly not cross platform. Only Windows XP, Windows Vista, and Windows 7 are supported. MRDS can, however, be used to program robotic platforms which may run other operating systems by the use of serial or wireless communication (Bluetooth, WiFi, or RF Modem) with the robot [62].

B. *Marilou*

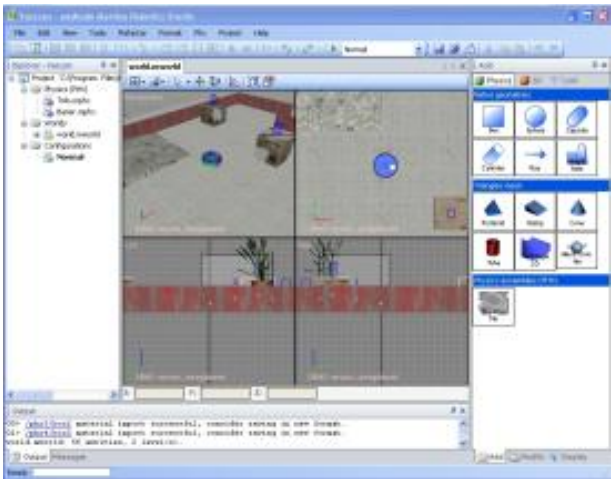

Figure 4. Screenshot of Marilou vacuum simulation

Marilou by anyKode [63] (shown in Fig. 4) is a full robotics simulation suite. It includes a built in modeler program so users can build their own robots using basic shapes. The modeler has an intuitive CAD-like interface. The physics engine simulates rigid bodies, joints, and terrains. It includes several types of available geometries [64]. Sensors used on robots are customizable, allowing for specific aspects of a particular physical sensor to be modeled and simulated. Devices can be modified using a simple wizard interface.

Robots can be programmed in many languages from Windows and Linux machines, but the editor and simulator are Windows only. Marilou offers programming wizards that help set up projects settings and source code for based on which language and compiler is selected by the user [65].

Marilou is not open source or free. While there is a free home version, it is meant for hobbyists with no intention of commercialization. The results and other associated information are not compatible with the professional or educational versions. Prices for these versions range from $360 to $2,663 [66].

C. *Webots*

The Cyberbotics simulator Webots [67] (shown in Fig. 5) is a true multiplatform three-dimensional robotics simulator that is one of the most developed of all the simulators

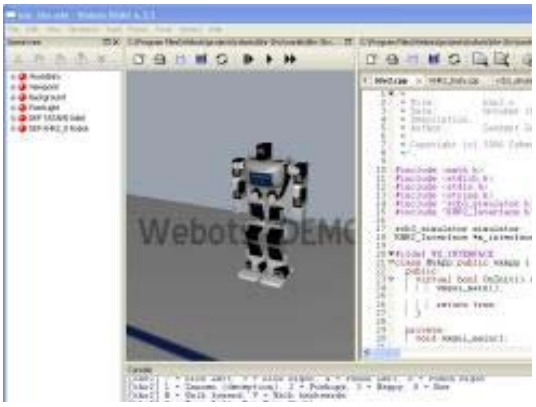

Figure 5. Screenshot of Webots humanoid demonstration

surveyed [68]. Webots was originally developed as an open source project called Khepera Simulator as it initially only simulated the Khepera robot platform. The name of the project changed to Webots in 1998 [69]. Its capabilities have since expanded to include more than 15 different robotics platforms [70].

Webots uses the Open Dynamics Engine (ODE) physics engine and, contrary to the criticisms of Zaratti, Fratarcangeli, and Iocchi[22], Webots has realistic rendering of both robots and environments. It also allows multiple robots to run at once. Webots can execute controls written in C/C++, Java, URBI, Python, ROS, and MATLAB languages [70]. This simulator also allows the creation of custom robotics platforms; allowing the user to completely design a new vehicle, choose sensors, place sensors where they wish, and simulate code on the vehicle.

Webots has a demonstration example showing many of the different robotics systems it can simulate, including an amphibious multi-jointed robot, the Mars Sojourner rover, robotic soccer teams, humanoids, multiple robotic arms on an assembly line, a robotic blimp, and several others. The physics and graphics are very impressive and the software is easy to use.

Webots has a free demonstration version available (with the ability to save world files crippled) for all platforms, and even has a free 30 day trial of the PRO version. The price for a full version ranges from $320 to $4312 [71].

D. *robotSim Pro / robotBuilder*

robotBuilder [72] is a software package from Cogmation Robotics that allows users to configure robots models and sensors. A screenshot can be seen in Fig. 6. Users import the models of the robots, import and position available sensors onto the robots, and link these sensors to the robot's controller. Users can create and build new robot models piece by piece or robotBuilder can import robot models created in other three-dimensional CAD programs such as the free version of Google Sketchup. The process involves exporting the Sketchup file as a Collada or 3DS file, then importing this into robotBuilder [73].

robotSim Pro is an advanced three-dimensional robotics simulator that uses a physics engine to simulate forces and collisions [74]. Since this software is commercial and closed source, the actual physics engine used could not be determined. RobotSim allows multiple robots to simulate at

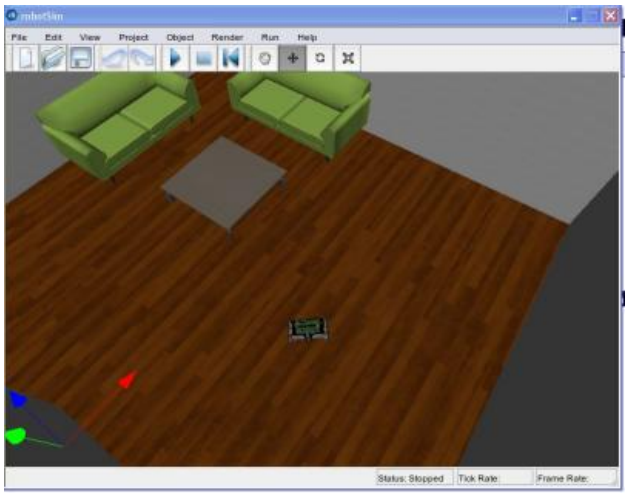

Figure 6. Screenshot of robotSIM National Instruments simulation

one time. Of all of the simulators in this survey, robotSim has some of the most realistic graphics. The physics of all objects within a simulation environment can be modified to make them simulate more realistically [74]. Test environments can be easily created in robotSim by simply choosing objects to be placed in the simulation world, and manipulating their positions with the computer mouse. Robot models created in the robotBuilder program can be loaded into the test environments. Robots can be controlled by one of three methods; the Cogmation C++ API, LabVIEW, or any socketed programming language [72].

robotSim is available for $499 or as a bundle with robotBuilder for $750. Cogmation offers a 90-day free trial as well as a discounted Academic License.

IV. Conclusion

While this is certainly not an exhaustive list of robotics simulators, this is a simple comparison of several of the leading simulator packages available today. Most of the simulators in this survey are designed for specific robotics platforms and sensors which are quite expensive and not very useful for simpler, cheaper systems. The costs and complexities of these systems often prevent them from being an option for projects with smaller budgets. The code developed in many of these simulators requires expensive hardware when porting to real robotics systems. The middleware that is required to run on actual hardware is often too taxing for smaller, cheaper systems. There simply isn't a very good three-dimensional robotics simulator for custom robotic systems designed on a tight budget. Many times a user only needs to simulate simple sensor interactions, such as simple analog sensors, with high fidelity. In these cases, there is no need for such processor intensive, high abstraction simulators.

References

[1] J. Craighead, R. Murphy, J. Burke, and B. Goldiez. "A Survey of Commercial & Open Source Unmanned Vehicle Simulators." Proceedings 2007 IEEE International Conference on Robotics and Automation, Apr. 2007: 852-857.

[2] Player Project, unpublished. http://playerstage.sourceforge.net (accessed Nov 2010)

[3] B.P. Gerkey, R.T. Vaughan, K. Stoy, A. Howard, G.S. Sukhatme, and M.J. Mataric. "Most valuable player: a robot device server for distributed control." Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium 2001: 1226-1231.

[4] Interviewing Brian Gerkey at Willow Garage, Aug. 28, 2008. unpublished. http://getrobo.typepad.com/getrobo/2008/08/interviewing-br.html (accessed Nov. 2010)

[5] Vaughan, Richard T, and Andrew Howard. "On device abstractions for portable , reusable robot code." Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003), October 2003: 2421-2427.

[6] "The Player Robot Device Interface" unpublished. http://playerstage.sourceforge.net/doc/Player-cvs/player/index.html (accessed Nov. 2010)

[7] Player Manual: Supported Devices unpublished. http://playerstage.sourceforge.net/doc/Player-cvs/player/supported_hardware.html (accessed Nov. 2010)

[8] B.P. Gerkey, R. T. Vaughan, and A. Howard. "The Player / Stage Project : Tools for Multi-Robot and Distributed Sensor Systems." Advanced Robotics, Icar 2003: 317-323.

[9] Player Project Basic FAQ, unpublished. http://playerstage.sourceforge.net/wiki/Basic_FAQ (accessed Nov. 2010)

[10] Gazebo Main Page unpublished. http://playerstage.sourceforge.net/index.php?src=gazebo (accessed Nov. 2010)

[11] J. Craighead, R. Gutierrez, J. Burke, and R. Murphy. "Validating the Search and Rescue Game Environment as a robot simulator by performing a simulated anomaly detection task." 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Sept. 2008: 2289-2295.

[12] M. E. Dolha, "3D Simulation in ROS Outline (slides)." Simulation (2010) unpublished. http://www.ros.org/wiki/Events/CoTeSys-ROS-School?action=AttachFile&do=get&target=3d_sim.pdf (accessed Dec. 2010)

[13] USARSim Main Page unpublished. http://usarsim.sourceforge.net/wiki/index.php/Main_Page (accessed Nov. 2010)

[14] G. Roberts, S. Balakirsky, and S. Foufou. "3D Reconstruction of rough terrain for USARSim using a height-map method." Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems - PerMIS '08, 2008: 259.

[15] B. Balaguer, S. Balakirsky, S. Carpin, M. Lewis, and C. Scrapper, "USARSim: a validated simulator for research in robotics and automation," in IROS'08.Workshop on robot simulators: available software, scientific applications and future trends, Nice, France, 2008.

[16] S. Okamoto, K. Kurose, S. Saga, K. Ohno, and S. Tadokoro, "Validation of simulated robots with realistically modeled dimensions and mass in usarsim," in Safety, Security and Rescue Robotics, 2008. SSRR 2008. IEEE International Workshop on, Oct. 2008, pp. 77–82.

[17] S. Carpin,M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. "USARSim: a robot simulator for research and education." Proceedings 2007 IEEE International Conference on Robotics and Automation (Apr. 2007): 1400-1405.

[18] USARSim Installation on Windows, unpublished. http://usarsim.sourceforge.net/wiki/index.php/4._Installation (accessed Dec. 2010

[19] USARSim Installation on Linux, unpublished. http://sourceforge.net/projects/usarsim/forums/forum/486389/topic/3873619 (accessed Dec. 2010)

[20] S. Carpin, T. Stoyanov, Y. Nevatia, M. Lewis, J. Wang. "Quantitative assessments of USARSim accuracy," in Proceedings of PerMIS 2006, 2006.

[21] S. Carpin, J. Wang, M. Lewis, A. Birk, A. Jacoff. "High fidelity tools for rescue robotics: results and perspectives", in Robocup 2005: Robot Soccer World Cup IX, Springer, LNAI Vol. 4020, Springer, 2006, pp. 301-311.

[22] M. Zaratti, M. Fratarcangeli, L. Iocchi. "A 3D simulator for multiple legged robots based on USARSim", in Robocup 2006: Robot Soccer World Cup X, Springer LNCS

[23] C. Pepper, S. Balakirsky, and C. Scrapper. "Robot simulation physics validation." Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems - PerMIS '07 (2007): 97-104.
[24] B. Balaguer and S. Carpin, "Where Am I? A Simulated GPS Sensor for Outdoor Robotic Applications," in Simulation, Modeling, and Programming for Autonomous Robots, vol. 5325, Springer Berlin Heidelberg, 2008, pp. 222-233.
[25] A. Birk, J. Poppinga, T. Stoyanov, and Y. Nevatia "Planetary exploration in USARsim: a case study including real world data from Mars" In L. Iocchi, H. Matsubara, A. Weitzenfeld, & C. Zhou (Eds.), Lecture notes in artificial intelligence (LNAI). RoboCup 2008: Robot WorldCup XII. Berlin: Springer.
[26] SARGE Games, unpublished. http://www.sargegames.com/page1/page1.html (accessed Nov. 2010)
[27] J. Craighead, J. Burke, R. Murphy. "Using the Unity Game Engine to Develop SARGE: A Case Study". Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008). Sept. 2008
[28] J. Craighead, R. Gutierrez, J. Burke, R. Murphy. "Validating The Search and Rescue Game Environment As A Robot Simulator By Performing A Simulated Anomaly Detection Task". Proceedings of the 2008 International Conference on Intelligent Robots and Systems (IROS 2008). Sept. 2008.
[29] User Guide for SARGE 0.1.7, Apr. 24, 2008 unpublished. Downloaded with Source Code for SARGE from http://sourceforge.net/projects/sarge/ (accessed Nov. 2010)
[30] ROS Documentation, unpublished. http://www.ros.org/wiki/ (acessed Nov. 2010)
[31] "ROS/Introduction", unpublished. http://www.ros.org/wiki/ROS/Introduction (accessed Nov. 2010)
[32] Robotics Equipment Corporation ROS for Windows, unpublished. http://www.servicerobotics.eu/index.php?id=37 (accessed Dec. 2010)
[33] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng (2009). ROS: an open-source Robot Operating System. In Open-source software workshop of the international conference on robotics and automation (ICRA).
[34] "Happy 3rd Anniversary, ROS!", unpublished. http://www.willowgarage.com/blog/2010/11/08/happy-3rd-anniversary-ros (accessed Nov. 2010)
[35] ROS FAQ, unpublished. http://www.ros.org/wiki/FAQ (accessed Dec. 2010)
[36] "Carnegie Mellon UberSim Project" http://www.cs.cmu.edu/~robosoccer/ubersim/ (acessed Dec. 2010)
[37] "CORAL Downloads" http://www.cs.cmu.edu/~coral/download/ (accessed Dec. 2010)
[38] B. Browning and E. Tryzelaar, "Ubersim: A multi-robot simulator for robot soccer" In Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems.
[39] "EyeSim - EyeBot Simulator" http://robotics.ee.uwa.edu.au/eyebot/doc/sim/sim.html (accessed Nov. 2010)
[40] T. Bräunl. "The EyeSim Mobile Robot Simulator The EyeSim Mobile Robot Simulator." Report University of Auckland, Mar. (2000).
[41] A. Koestler, and T. Bräunl. "Mobile Robot Simulation with Realistic Error Models (eyesim)." Autonomous Robots 51, no. 6 (2004): 46-51.
[42] A. Waggershauser, "Simulation of small autonomous mobile robots." PhD Thesis, Kaiserslautern University of Technology Dec. (2002)
[43] SubSim Website, unpublished. http://robotics.ee.uwa.edu.au/auv/subsim.html (accessed Dec. 2010)
[44] T. Bielohlawek "SubSim - An Autonomous Underwater Vehicle Simulation System", PhD Thesis, Kaiserslautern University of Technology, Apr. (2006)
[45] OpenRAVE Main Page, unpublished. http://openrave.programmingvision.com/index.php/Main_Page (accessed Dec. 2010)
[46] R. Diankov and J. Kuffner "Openrave: A planning architecture for autonomous robotics" Tech. Rep. CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon University (2008)
[47] "OpenRAVE and ROS" http://www.willowgarage.com/blog/2009/01/21/openrave-and-ros (accessed Dec. 2010)
[48] "Introduction to OpenRAVE…" http://openrave.programmingvision.com/index.php/Main_Page#Introduction_to_OpenRAVE_-_the_Open_Robotics_Automation_Virtual_Environment (accessed Dec. 2010)
[49] OpenRTM-aist Official Website http://www.openrtm.org/ (accessed Dec. 2010)
[50] "RT-Middleware : OpenRTM-aist version 1.0 has been released", unpublished. http://www.aist.go.jp/aist_e/latest_research/2010/20100210/20100210.html, (accessed Dec. 2010)
[51] G. Roberts, S. Balakirsky, and S. Foufou. "3D Reconstruction of rough terrain for USARSim using a height-map method." Proceedings of the 8th Workshop on Performance Metrics for Intelligent Systems - PerMIS '08 (2008): 259.
[52] "The Mobile Robot Programming Toolkit", unpublished. http://www.mrpt.org/ (accessed Dec. 2010)
[53] J. L. Claraco. Development of Scientific Applications with the Mobile Robot Programming Toolkit. The MRPT reference book. Machine Perception and Intelligent Robotics Laboratory, University of M´laga, M´laga, Spain, October, 2010.
[54] "About MRPT", unpublished. http://www.mrpt.org/About (accessed Dec. 2010)
[55] "Research Network for Self-Organization of Robot Behavior", unpublished. http://robot.informatik.uni-leipzig.de/software/ (accessed Nov. 2010)
[56] "SimRobot - Robotics Simulator", unpublished. http://www.informatik.uni-bremen.de/simrobot/index_e.htm (accessed Nov. 2010)
[57] T. Laue and T. Rofer Simrobot - development and applications. In Proceedings of the International Conference on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR 2008), Venice, Italy, 2008.
[58] T. Laue, Tim, K. Spiess, and T. Rofer. "SimRobot – A General Physical Robot Simulator and its Application in RoboCup." In: Proc. of RoboCup Symposium. Universität Bremen (2005)
[59] Moby Project Main Page, unpublished. http://physsim.sourceforge.net/index.html (accessed Nov. 2010)
[60] "Microsoft Robotics", unpublished. http://msdn.microsoft.com/robotics (accessed Nov. 2010)
[61] J. Jackson Microsoft robotics studio: A technical introduction. Robotics and Automation Magazine 14(4) (2007) 82–87
[62] Overview of Microsoft Robotics, unpublished. http://msdn.microsoft.com/en-us/library/bb483024.aspx (accessed Dec. 2010)
[63] "anyKode Marilou - Modeling and simulation environment for Robotics", unpublished. http://www.anykode.com/index.php (accessed Nov. 2010)
[64] Marilou Physics, unpublished. http://www.anykode.com/marilouphysics.php (accessed Nov. 2010)
[65] Marilou Programming Wizard, unpublished.s http://www.anykode.com/marilouwizard.php (accessed Nov. 2010)
[66] Marilou Licensing Information, unpublished. http://www.anykode.com/licensemodel.php (accessed Nov. 2010)
[67] "Cyberbotics Webots", unpublished. http://www.cyberbotics.com/products/webots/ (accessed Nov. 2010)
[68] M. Olivier "Webots TM : Professional Mobile Robot Simulation." Int. Journal of Advanced Robotic Systems (2004): 39-42.
[69] "Khepera Simulator Homepage", unpublished. http://diwww.epfl.ch/lami/team/michel/khep-sim/ (accessed Nov. 2010)
[70] Webots User Guide, Published Dec. 2010, unpublished. www.cyberbotics.com/cyberbotics/guide.pdf (accessed Dec. 2010)
[71] Webots Order Page, unpublished. http://www.cyberbotics.com/order/ (accessed Nov. 2010)
[72] "Cogmation Robotics – roboSim Pro", unpublished. http://www.cogmation.com/robot_builder.html (accessed Dec. 2010)
[73] "Create model with Google Sketchup", unpublished. Forum post http://cogmation.com/forum/viewtopic.php?f=9&t=6 (accessed Nov. 2010)
[74] roboSim User Guide, unpublished. http://cogmation.com/pdf/robotsim_doc.pdf (accessed Dec. 2010)