Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

AN INTRODUCTION TO OPENSIMULATOR AND VIRTUAL ENVIRONMENT AGENT-BASED M&S APPLICATIONS

Paul A. Fishwick

Computer & Information Sci. and Eng. Dept.
Bldg. CSE, Room 301
Gainesville, FL 32611, USA

ABSTRACT

An "agent" in a computer simulation is an object with a dynamic model driving its actions. There are different classifications for agents, for example: autonomous, intelligent, and software. A cell within a cellular automaton might be considered an agent with the complete environment being a multi-agent system. An object containing an artificial intelligence could also be considered an agent. Our purpose is to introduce the "personal" aspect of agents through first-person perspective—by becoming one of the agents in the simulation. When a level of presence on the part of the human's relationship to the agent is incorporated in this fashion, we must incorporate methods found typically within multi-user virtual environments. This tutorial is centered on one particular open-source, multi-user, virtual environment system called OpenSimulator (or OpenSim). We introduce OpenSim to allow the reader an opportunity for understanding how this software is used within the context of agent-based computer simulations.

1 INTRODUCTION

We should begin by establishing conditions and requirements for agent-based approaches. While a network of manufacturing services tied together might ostensibly be considered a network of agents, this type of application is not generally considered to be an agent-based simulation, and so we should endeavor to first seek out common qualities of agents. Historically, there are several qualities attributed to agent. Macal and North (2005) present a two-part tutorial that identifies agent qualities, a subset of which are included here:

- Identity: a discrete individual with a set of characteristics
- Interaction: the agent is situated in an environment where there is interaction with other agents
- Goal-directed: the agent seeks an objective
- Autonomy: the agent is autonomous

When considered at a glance, these qualities are those of living organisms. Based on this observation, the following might be considered valid agents: virus, bacterium, cow, or a human. However, to be flexible in interpretation, and to allow for maximal use of the agent concept, we can extend the definition of an agent to incorporate any adaptive object that has organic, autonomous qualities. In this extended use of "agent", a software module may be considered to be an agent since it may be adaptive and highly reconfigurable and mobile. For example, an internet worm or virus is an agent according to this definition.

The agent concept is indeed a powerful one and relevant to many domains from modeling communities within an ecological niche to modeling the interaction between software components. A panel was held recently considering different ways in which agents can be utilized within defense and civilian simulation use-cases (Oren et al. 2000). There is a compendium of knowledge about the state of the art from modeling humans from individuals to societies (Zacharias et al. 2008) or ecosystems (Fishwick et al. 1998). The concept of *scale* is a key concern: how to model agents at different levels of abstraction and aggregation while mainlining semantic interoperability among levels? Different scales allow us to study the system while scaling along the *axis of agent population*: from one to many.

However, with the increasing interest in agents and populations, there is a missing component: the scale associated with human reference. It may be assumed when simulating agents that the person who is running the simulation maintains a relative distance from the model and its behaviors. For example, if one executes a cellular automaton, one sees cell colonies growing, dying, and transforming—creating an array of patterns and sometimes, self-organization. The relative position of the human running this automaton is similar to that of a scientist running an experiment perhaps through a microscope or in situ by studying the agents "from the outside." It is as if we can maintain that we understand how populations undergo their dynamics without becoming a member of that population. Is this possible? To a large extent, this is true—we routinely do science in this fashion but primarily because we are unable to see the environment through the eyes of the agent. Even though we may gather useful information about population behavior in this manner, we may be collectively missing the opportunity to act-out roles for the agents to better understand how an agent functions within a specific context. One way of summarizing the issue of "human becoming the agent" is the observation that agents have bodies and the point of the body is that one employ its affordances through role-playing.

2 SCALE OF HUMAN REFERENCE

If an agent can have a body then we introduce another dimension to agent-based modeling and simulation where we have to consider a new scale: the relative position and orientation of the body to the environment. In physical reality, this process is enhanced through special instruments such as the microscope and telescope: to get closer to the colony of bacteria, one must gaze through the microscope and see them up-close. However, it is not possible to become a bacterium. For this more refined scale of human reference to the environment, we need to leverage newer technologies. The field of virtual environments (Sherman and Craig 2003) suggest how we may achieve new positions and orientations of the human with respect to either physically perceived, or actual, phenomena. In the simplest case, desktop virtual environments create the illusion of being in another space through no other means than the use of a display screen used in conjunction with the common input devices of the keyboard and a mouse.

The most straightforward way of envisioning the scale of human reference at least for human vision is to imagine a virtual camera that can be easily moved. At a distance, and with the correct field of view and other camera attributes, we can see with this camera a cluster of agents acting over a space. Without visual perspective, this view would appear much like the 2D cellular automata simulations or the simulation of a commander exercising planning options on a battlefield—moving this battalion in one direction and that company in another. As this virtual camera gradually moves toward an agent, we realize that the agent has a corporeal existence and it ceases to exist as solely a dot on a map. In the case of a human agent, we begin to notice the person's physical attributes such as their height, gender, ethnicity, and clothing. Moving the camera just behind a person, but over the shoulder as not to obscure our view, we create a more personal view. The most direct view is when the camera is positioned in the head reference frame as to simulate what the agent sees.

This relationship between being far away from the agents vs. becoming them can also be viewed in terms of the distance part of the human reference scale. Table 1 illustrates two axes: Topic to be considered, and Proximity of the user to the environment. The human interface in a case where the human is distant from a target involves little "sense of being there" (i.e., presence). The utility for "being there" is that one gains a body-centered understanding of phenomena, often more useful for learning and training rather than for pattern analysis.

Table 1: Different approaches for contextualizing agents within a simulation.

Topic/Proximity	*Distant*	*Near*
Human Interface	Little Presence	Significant Presence
Utility	Pattern	Experience
Cognition	Reasoning	Doing

3 INTRODUCTION TO OPENSIM

3.1 Basic Overview

The development of OpenSim is situated historically within the development and deployment of the Second Life platform (Weber et al. 2007). Second Life, which is owned and distributed by Linden Labs, Inc. (Linden Labs 2009), is composed of two key components: a *server* that hosts a virtual world composed of a massive rectangular grid of regions, and a *client* program that runs on the user's computer and that communicates with the server. The client and server have different functions:

- Client: a software program that has a window in which the user navigates a perceived three-dimensional space and performs other key functions such as search, map view, inventory management, chat communications, user interface settings, and client administration.
- Server: software that communicates with the client by accepting requests and producing responses. The server has one or more databases that store all user assets and inventory items.

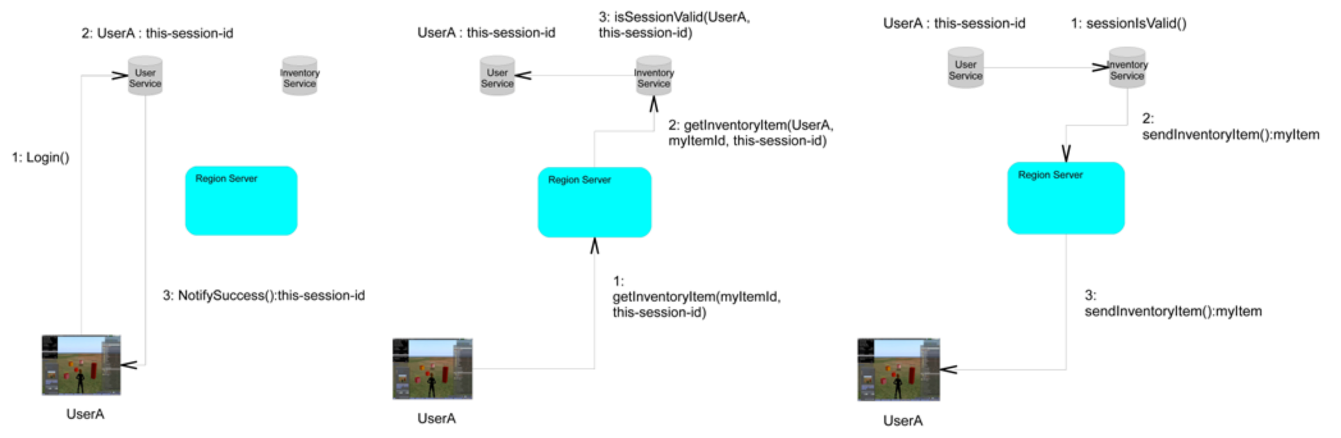
In January 2007, Linden Labs “open sourced” the client software, thus making the internals of the code available to a broad array of users and developers. One small set of developers beginning during that time frame produced an alternate “back end” server that serviced the client. This was made possible since the client’s functioning was made evident as part of the open source disclosure. This alternative server resulted in a project called OpenSimulator (2009), abbreviated OpenSim.

3.2 Operation and Architecture

OpenSim has three major operational instances—defined as “modes”: Standalone, Grid, and Hypergrid. OpenSim code is written in C# which is supported, more generally, as Mono, an open source .NET platform (Mono 2009). These modes are described below:

- *Standalone Mode*: The client interacts with one server and the client user authenticates on that server before being placed in-world. The server contains all basic services wrapped into one executable image which invokes dynamic linked libraries as necessary. These services are described separately in the next mode.
- *Grid Mode*: This mode includes specific services abbreviated as UGAIM. Each letter in this phrase stands for a separate service: User, Grid, Asset, Inventory, and Messaging. The User service manages authentication and knows about users who log into the grid. The Grid service knows the general layout of the rectangular grid including the endpoint internet addresses for each region. A grid is composed of regions in much the same way that a floor contains a grid of tiles. The entire grid comprises the “world”. The Asset service manages all assets in regions. Examples of assets include basic geometry, texture maps, audio files, and terrain geometry. The Inventory server manages which assets are tied to a particular agent, so that when a user authenticates as an agent, that agent will “own” a set of inventory items. The Messaging server handles text chat messaging.
- *Hypergrid mode*: A hypergrid is a loosely connected set of simulators without a global grid manager. In many ways, it resembles hypermedia on the internet. The equivalent of the web hyperlink for OpenSim is the teleport from one region to another. Hypergrid concepts are more complex than on the web since teleporting requires significantly more exchange of messages and updating of region information.

Figures 1(a) through 1(c) provide schematic diagrams showing the process of a client (i.e., User A) logging into an OpenSim server to obtain an inventory item via the region server. That inventory item might be any sort of asset from a texture to a linkset of prims (i.e., atomic shapes and their attributes).


2: UserA : this-session-id
User Service
Inventory Service
1: Login()
Region Server
3: NotifySuccess():this-session-id
UserA
UserA : this-session-id
3: isSessionValid(UserA, this-session-id)
User Service
Inventory Service
2: getInventoryItem(UserA, myItemId, this-session-id)
Region Server
1: getInventoryItem(myItemId, this-session-id)
UserA
UserA : this-session-id
1: sessionIsValid()
User Service
Inventory Service
2: sendInventoryItem():myItem
Region Server
3: sendInventoryItem():myItem
UserA

(a) Login sequence (b) Get Inventory Item (from client) (c) Send Inventory Item (to client)

Figure 1: Sequence of operations using the protocol for logging into OpenSim and retrieving an inventory item. Image Copyright © Justin Clark-Casey. Used with permission.

We should relate the general architecture of OpenSim to computer and console games. All OpenSim modes can support "multi-player" scenarios where multiple users can simultaneously be present within a single virtual space; however, one unusual aspect of both OpenSim and the client is the ability to create all objects *in situ*, without having to perform the more routine import/export process required of other design software. The concept of inventory fits nicely within the concept of massively multiplayer online role playing games (MMORPG) since players in that class of game manage an inventory where they can get and put objects that they find when exploring the game space.

3.3 Building OpenSim

The following instructions are for obtaining and building OpenSim. The server can be built on different platforms such as Windows, Mac, and Unix. The procedure for Windows is described below:

(a) Obtain Visual C# Express, which is a free download from the Microsoft web site. Mono can also be used for Windows, but is necessary for the Mac and Unix installations.
(b) Obtain the OpenSim distribution from the instructions at: http://opensimulator.org/wiki/Download.
(c) Follow the build instructions at: http://opensimulator.org/wiki/Build_Instructions.
(d) The default installation runs a server on address 127.0.0.1 (localhost). To execute the server so that other clients can access the server, it is necessary to use the external IP address for the server machine.

More detailed instructions for building OpenSim can be found in the above web pages. Practically speaking, OpenSim runs well on most computers, including laptops, and performance issues arise mainly when there are too many objects stored as assets or when there are too many agents (i.e., clients) attempting to access the server. At the time of this writing, most regions can comfortably support 20 avatars. Differences in computer architecture and machine capabilities most certainly have an effect of performance. For example, more memory can provide significant performance improvement with more avatars and objects possible in-world.

3.4 Using OpenSim

To use OpenSim, one must choose a client and then choose which server to access. The default client is the open source one provided by Linden Labs, Inc. An example of this interface is shown in Figure 2. In this figure, one can see the main operating area (i.e., the place where navigation occurs) as well as a sample prim in the foreground, texture-mapped with the default wood image, two open sub-windows showing an overhead map and text communications window, and the avatar. There are three key terms used for people who interact with this type of environment: (1) *user*: the human who is the user of the client software, (2) *agent*: the user's identity in-world, and (3) *avatar*: the user's visual representation in-world. One user may have several agents, and each agent may have several different visual identities or appearances.

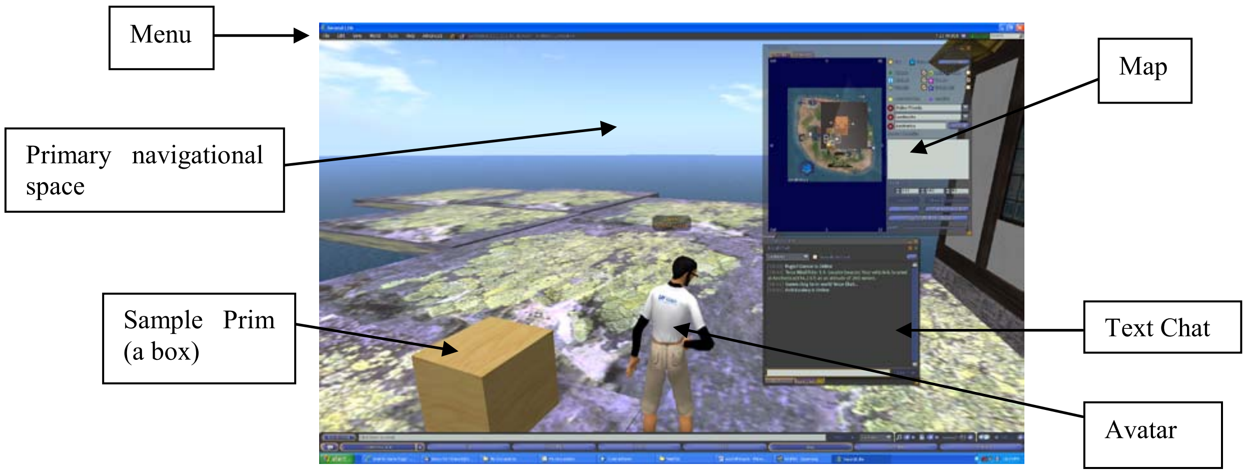
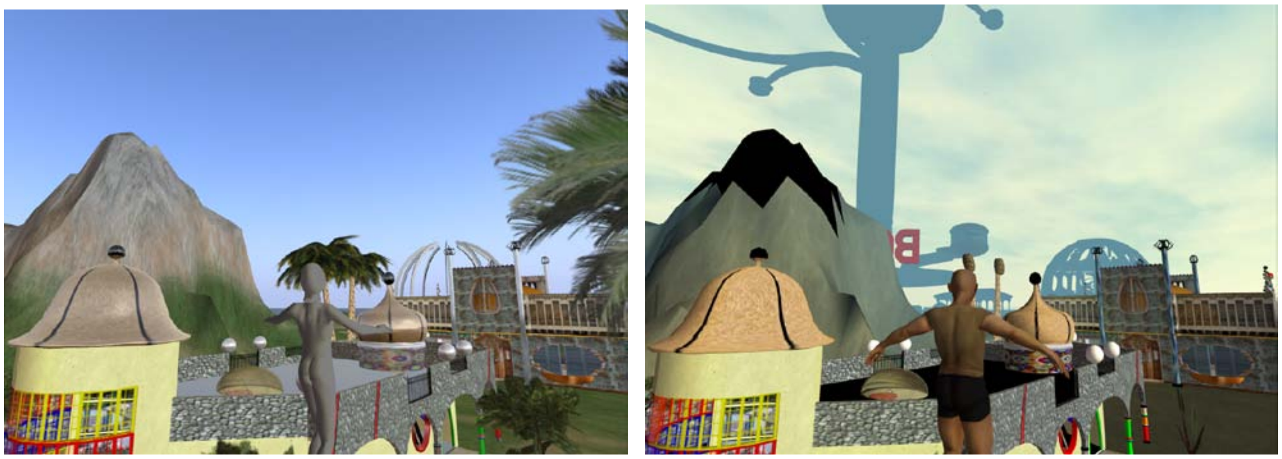


Figure 2: The interface window for the Linden Labs, Inc. open source client software.

The Linden Labs client viewer is not the only way to connect to OpenSim servers. For comparative rendering purposes, Figure 3 shows two main client windows, minus the auxiliary menu items and sub-windows. On the left in Figure 3(a), we have the Linden Lab client as shown in Figure 2, and on the right in Figure 3(b), the RealXtend (2009) client, both showing different renderings of the same scene taken from the same camera position. RealXtend uses a different underlying approach to rendering the scene. This alternative engine creates the possibilities of shadows, more flexible lighting conditions, and level of detail (LOD) effects.



(a) Linden Lab client (b) RealXtend client

Figure 3: Two clients access the same server region and using the same camera angle, prior to the snapshot.

4 THE UTILITY OF OPENSIM

From the standpoint of the simulation researcher, practitioner, or individual interested in simulations, a key question that arises when considering the panoply of Agent-based M&S options is:

What type of agent-based simulation should I use?

Even though this tutorial has focused on one specific type of simulation, this question is posed by most of us when considering agent-based approaches. The following three questions need to be asked to be able to answer this general question: (1) What is the goal of the simulation?, (2) What will be simulated?, and (3) Who will use the simulation?. Sometimes, we may forget to ask these questions but it is critical that we ask them when simulating agents. Let's take each question in turn. For the first question, goals may be different. One goal may be "To do prediction" or "To study patterns". These goals taken together suggest that the user may be a scientist. An engineer might answer differently with the same requirement to predict, but then for to validate against physical phenomena for the goal rather than pattern analysis and free-form exploration. The second question, while seemingly obvious, is equally as subtle since we must decide at what level of abstraction or aggregation we should be employing our simulation. Do agents represent individuals or groups, or even large segments of a population? The second question also encourages us to re-ask the question "What (or who) is the agent in this simulation?" The third question is just as critical—who is going to use this simulation, how many users will use it, and what sort of representations do they prefer? It may be sufficient to state "the users are simulation experts", and one must assess whether this is the correct goal for a project or whether there are other users who have not been considered—such as high level decision makers, simulation non-experts, or indeed the general population.

The software that we overviewed, OpenSim, can be used by many different groups and for different purposes. The software encourages collaboration, a sense of presence, and a closer-to-real-life world representation. Unless, the stated goal of a project is pure science and the target user population are simulation experts, systems such as OpenSim and its associated human interface clients can play a key role within the simulation community. At the University of Florida, we have a research project (Fishwick et al. 2008) where OpenSim is being tested as a way to train the warfighter for cross-cultural awareness. For training, teaching, and communication purposes, OpenSim tends to perform well.

5 SUMMARY

The intent of this presentation has been a brief introduction to OpenSim, which is a set of services used for virtual world construction. We discussed the concept of agent-based simulation and mentioned that there are different scales when considering how a simulation researcher or practitioner may use each type of agent. If the goal of the agent-based software is to support a scientist who is trying to understand emergent behaviors of agents then the reader is advised to use software that emphasizes this type of analysis. Conversely, if the goal is to support a person who needs full-body access to an environment, or perhaps someone is a non-simulation specialist who may feel more comfortable with scenes that look-like familiar physical objects, then OpenSim has signficant potential for these audiences.

It is still unclear whether having a high level of presence, independent of whether a training task is being performed, is useful for enhanced cognitive skills such as short and long term memory. We are in the midst of a year-long experimental design to evaluate the role of presence and interaction for measuring memory, empathy, and motivation.

Some have anecdotally coined OpenSim "the Apache of virtual worlds" using the analogy of the Apache tools being the predominant open source vehicle for implementing web services. It is too early to determine whether this anecdote will hold true; however, the community is fairly active and has many core developers. The current dearth of available clients for OpenSim services will hopefully be rectified over the next year with some emerging projects centerend on different approaches to rendering.

ACKNOWLEDGMENTS

The author would like to acknowledge Christa Lopes of University of California at Irvine and Justin Clark-Casey for online discussions regarding OpenSim service protocols, and for Justin Clark-Casey for permission to use the images in Figure 1. The author also acknowledges useful technical discussions on the two OpenSim mailing lists (opensim-user and opensim-dev), as well as technical discussions with his University of Florida students on OpenSim, particularly Hyungwook Park, Zachary Ezzell, and Ryan Tanay.

REFERENCES

Fishwick, P. A., J. G. Sanderson, and W. F. Wolff. 1998. Multimodeling basis for across-trophic-level ecosystem modeling: the Florida everglades example. 15 (2): 76-89. *Transactions of the Society for Computer Simulation International*.

Fishwick, P. A., J. Henderson, E. Fresh. F. Futterknecht, and B. D. Hamilton. Simulating culture: an experiment using a multi-user virtual environment. 2008. In *Proceedings of the 2008 Winter Simulation Conference*, ed. S. J. Mason, R. R. Hill, L. Monch, O. Rose, T. Jefferson, and J. W. Fowler, 786-794. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Linden Labs, Inc. 2009. `<http://www.lindenlab.com>`

Macal, C. M. and M. J. North. 2005. Tutorial on agent-based modeling and simulation. In *Proceedings of the 2005 Winter Simulation Conference*, ed. M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, 2-15. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Macal, C. M. and M. J. North. 2006. Tutorial on agent-based modeling and simulation part 2: how to model with agents. In *Proceedings of the 2006 Winter Simulation Conference*, ed. L F. Perrone, F. P. Wieland, J. Liu, B. Gl. Lawson, D M. Nicol and R. M. Fujimoto, 73-83. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

Mono Project. 2009. `<http://www.mono-project.com>`

OpenSimulator. 2009. `<http://www.opensimulator.org>`

Oren, T. I., S K. Numrich, A. M. Uhrmacher, L. F. Wilson, and E. Gelenbe. 2000. Agent-directed simulation: challenges to meet defense and civilian requirements. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 1757-1762. Piscataway, New Jersey: Institute of Electrical and Electronic Engineers, Inc.

RealXtend. 2009. Open source platform for interconnected virtual worlds. `<http://www.realxtend.org>`

Sherman, W. R., and A. B. Craig. *Understanding virtual reality: interface, application, and design*. 2003. Morgan Kaufmann.

Weber, A., Rufer-Bach, K., and R. Platel. 2007. *Creating your world: the official guide to advanced content creation for second life.* Sybex Publishing.

Zacharias, G. L., J. MacMillan, and S. B. Van Hemel. 2008. *Behavioral modeling and simulation: from individuals to societies*. National Research Council. Washington, DC.

AUTHOR BIOGRAPHY

PAUL A. FISHWICK is Professor of Computer and Information Science and Engineering at the University of Florida. Fishwick's research interests are in simulation modeling methodology, aesthetic computing, and the use of virtual world technology for modeling and simulation. He is a Fellow of the Society of Modeling and Simulation International, and recently edited the CRC Handbook on Dynamic System Modeling (2007). He served as General Chair of the 2000 Winter Simulation Conference in Orlando, Florida. His email address for these proceedings is <fishwick@cise.ufl.edu>.