Simulation Framework in Second Life with Evaluation Functionality for Sensor-based Systems

Boris Brandherm[1], **Sebastian Ullrich**[1,2], **Helmut Prendinger**[1]
[1] National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, 101-8430, Tokyo, Japan
[2] Virtual Reality Group, RWTH Aachen University, 52074 Aachen, Germany
{boris, helmut}@nii.ac.jp, s.ullrich@ieee.org

ABSTRACT
This paper describes a simulation framework for sensor-based systems utilizing "Second Life", a popular virtual three-dimensional multi-user online world. With this platform, the components of a sensor-based system can be mapped to (or, represented by) virtual counterparts. The intuitive user interface of Second Life and its comprehensive visualization support evaluation tasks of ubiquitous computing applications. Developers can directly control and manipulate virtual counterparts of real devices. In this way, different settings of a sensor-based system can be tested. The main contribution of our work consists of a bi-directional interface between sensor-based systems and Second Life.

ACM Classification Keywords
I.6.7 Simulation and Modeling: Simulation Support Systems—*Environments*

General Terms
Design, Experimentation

Author Keywords
Simulation, testbed, sensor-based systems, evaluation, Second Life

INTRODUCTION
Ubiquitous computing environments and sensor-based systems are highly active fields of research. Many exciting new devices are being developed and the amount of powerful and versatile sensors is rapidly increasing through the advances in embedded systems and technological evolution.

The implementation, testing and evaluation of new ubiquitous systems in a real environment are laborious tasks. Significant time and effort has to be spent on designing and testing prototypes and simulators in order to avoid unforeseen problems, e. g. , regarding optimization, before the system is actually installed. While simulators for specific types of sensors are available, it is still difficult to evaluate a heterogeneous complex system. Hence the visualization of all the simulated sensor data is desirable and an intuitive interaction capability to change the parameters and spatial position of the devices would be helpful to optimize the system.

Because of these requirements, we propose a three-dimensional (3D) virtual environment for the simulation, testing and evaluation of sensor-based systems. Besides extensive research in virtual reality, nowadays even game engines and multi-user online worlds provide convincing 3D environments. So, instead of creating a new 3D-engine, we decided to use the 3D environment of Second Life [14]. In Second Life (SL), 3D content including buildings and props can be created easily, and anyone can interact with the environment in the form of an 'avatar' (a human-controlled graphical representation of the user).

The rest of the paper is structured as follows. We start with a brief review of related work. To motivate the merit of our simulation framework, we then report on the experience with an existing sensor-based system. After that, we explain two different approaches to simulation. First, we describe a rapid-prototyping approach for SL, and discuss its benefits and shortcomings. Second, we describe our own simulator framework [3, 16], and explain its functionality with respect to evaluation of sensor-based systems. We give an example where our system is used to evaluate the performance of an indoor-positioning system. The paper concludes with a discussion and summary.

RELATED WORK
Currently, the development, testing and evaluation of new systems is realized in different ways, ranging from real-world testing and evaluation [4] and miniature mock-ups for prototyping, to software-based simulators [1, 11]. Recent testbeds are MoteLab [17] for wireless sensors, eHomeSimulator [1] for smart environments, and ubiBuilding Simulator [11] for large scale context-aware systems. While these software-based testbeds are far more practical than physical models, all of them are limited to testing in two-dimensional space.

Ubireal [10] is a 3D simulator for smart environments. Yet, its focus lies on systematic testing to verify rules and user-specified programming between different smart devices and sensors. There is no support for interactive exploration and testing. Another simulator, called TATUS [12], is based on the Half-Life game engine. The system focuses on human interaction with ubiquitous computing environments rather than the setup of such environments.

While all these approaches demonstrate promising features, they either (1) do not work as testbeds for simulation, or (2) provide specialized (non-generic) solutions, or (3) lack 3D interaction with the simulated virtual environment.

Let us now take a closer look at solutions that feed real-world data from sensors into SL and/or SL data into the real world (e. g. to control a device). In [8] data from a specialized power-plug based sensor network are fed into the virtual world by means of a (latency restricted) SL script-based implementation of a XML-RPC protocol. The data is used for visualization but there is no support for interaction with it. [9] is a work where sensors embedded in commercial mobile phones are used to infer real-world activities (whether the user is idle, walking or running), that in turn are mapped to visual metaphors in the virtual environment. [5] reports about a real-world control panel that can both control objects in the virtual world of SL and in turn be controlled by them. Changes to the knobs or pushbuttons in the real world are translated to their virtual counterparts in SL, and pushing the virtual buttons controls the LEDs on the real world control panel.

Although these approaches demonstrate interesting results, (1) they are not generic, (2) they don't provide a direct bi-directional feedback loop (e. g. if we control a real device via the virtual counterpart from inside SL and the status of the real device changes this change is immediately fed into and represented in SL again and vice versa.) and (3) they don't take into account the context of the devices in the environment (e. g. the position and the orientation of a sensor can be crucial for the system behavior like a indoor positioning system).

Because of the lack of interactive and generic solutions, we have created a bi-directional simulation framework for SL [3]. This system has been extended from a scripted simulation within SL to a more flexible interface and will be described by an example application.

EXAMPLE APPLICATION

Positioning systems are often used in ubiquitous computing environments. As a simple motivating example for our simulation framework, we chose an existing indoor positioning system [2]. Sensor placement for such a system is a non-trivial task as it depends on several factors such as the infrastructure, the amount and type of available sensors, and interferences.

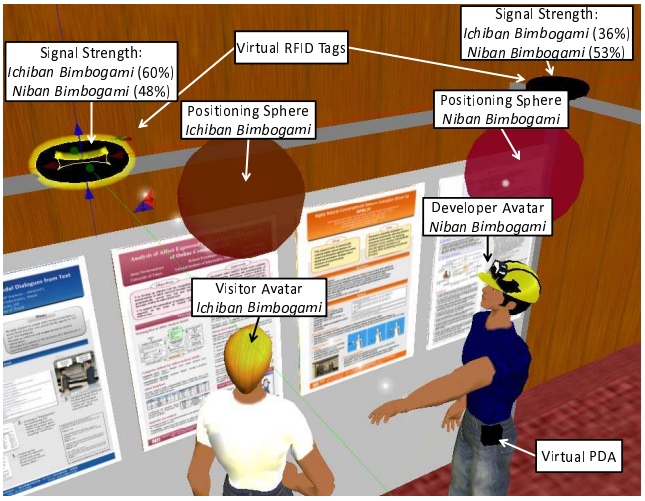

Figure 1. Example of the simulated positioning system in Second Life with a Visitor Avatar experiencing the system and a Developer Avatar who is interactively adjusting the properties of a virtual RFID tag.

The system described in [2] features different kinds of sensors and emitters: infrared beacons and radio-frequency identification (RFID) tags. The accuracy of the positioning system depends on the good placement of these sensors and emitters in the environment. The user carries a mobile device, which is equipped with the corresponding sensors. On this device the position of the user is computed based on the received sender data and afterwards displayed on a 2D map on the device.

When the system was installed in the environment, initial trials to improve accuracy by adding additional emitters failed. Furthermore, in boundary regions of the emitter field some unexplainable artifacts appeared in the position calculation. These open issues could be resolved easily in our SL based simulation system, as described in the following sections.

RAPID PROTOTYPING IN SECOND LIFE

Our first approach was to investigate the simulation capabilities of SL. Therefore we used the official API of SL that is called "Linden Scripting Language" (LSL). This programming language allows one to assign scripts to in-world objects. With over 300 library functions and different data and message types, scripts can control the behavior of virtual objects and communicate with other objects and avatars (users of SL). Limitations of LSL include time delays for movement of objects (0.2 sec) and memory constraints for scripts (16 KB). These constraints have a high impact on the achievable simulation accuracy, response times, and achievable simulation complexity within SL.

To simulate the previously discussed positioning system we created virtual objects in SL that represent RFID tags and can be positioned within the virtual environment interactively (see Fig. 1). Visitors who wish to be

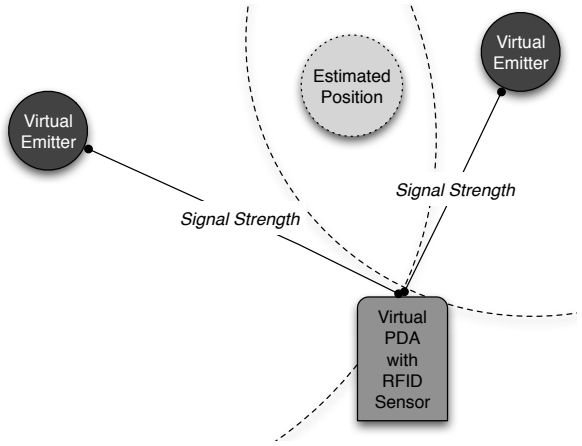

Figure 2. Conceptual overview of the example application: the virtual PDA reads the signal strengths of the virtual emitters and calculates an estimated position.

positioned by the system in SL have their avatar take a copy of a PDA object. The virtual PDA embeds a LSL script that is the core of the simulated positioning system and performs the calculation of the position. The PDA communicates with the virtual emitters and takes the signal strength to estimate the position, just as the real system (see Fig. 2). In the real world application, the result is shown on a 2D map on the PDA. In SL the calculated position is visualized in three dimensions, as a floating sphere in the virtual environment (see Fig. 1). If a user logs in to SL as a "visitor avatar", he or she can experience and interactively test and evaluate the sensor-based system. As a "developer avatar" the user can additionally adjust the virtual sensors and devices in the virtual environment.

The artifacts in the example application could be experienced in the simulation in SL. In both cases the signal range of the RFID tags has been too high compared to the distance of each RFID tag to another. This explanation was found after interactive testing (repositioning the sensors and adjusting their sender range). Consequently, the best results were achieved with just a small overlapping of the RFID signals.

Benefits of this approach include the ability of rapid prototyping of coarse simulations with reduced complexity and no need for additional software or servers: the scripts in SL reside within the objects. Shortcomings of this approach are the aforementioned limitations of the scripting language. Additionally, it can be very tedious or even impossible to map the functionality of an existing system into LSL. A better solution is to provide an interface to reuse existing systems, this approach is described in the following section.

SIMULATION FRAMEWORK WITH TWIN-WORLD MEDIATOR

Our architecture (Fig. 3) consists of three components: (1) the Twin-World Mediator, (2) a sensor-based system, and (3) the SL 3D environment. The main task of the Simulation Framework is to provide an interface to existing systems, sensors and simulators and to mediate the exchange of data with SL for visualization and interaction.

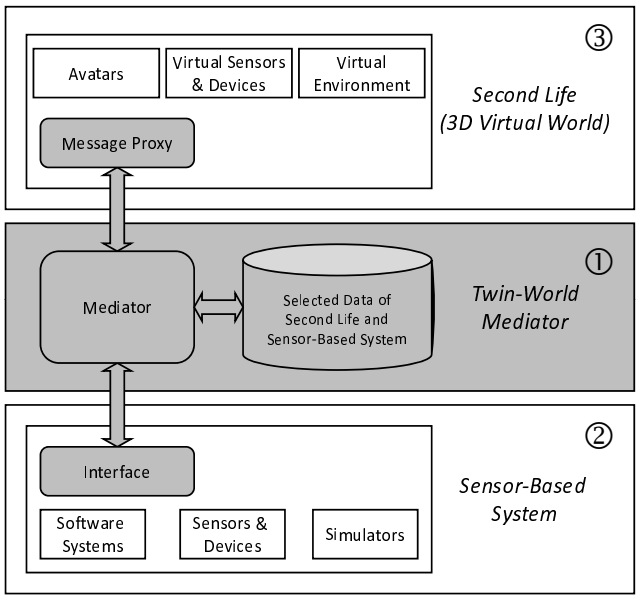

Figure 3. Architecture that embeds existing sensor-based systems (2) with Second Life (3) through the Twin-World Mediator (1).

In the following we first describe the architecture in more detail. To illustrate the usage of the system, we show how we embedded our example application.

Components of the Architecture

The *Twin-World Mediator* (1) consists of the Mediator, the Message Proxy, the Interface and a database. The key component of the simulation framework is the Mediator which handles the data exchange between Message Proxy, Interface and the database of the simulation framework. It ensures the data exchange of the components of the sensor-based system with their virtual counterparts in SL.

A *sensor-based system*(2) typically consists of software systems and sensors & devices. Simulators are used for unavailable sensors and devices. The components of the sensor-based system register as listeners through the interface for the desired data.

The Message Proxy connects to *Second Life*(3) and gathers data about the components in SL and sends them to the Mediator. We are especially interested in the state of the avatars, the virtual sensors & devices, and the virtual environment because they represent the virtual counterparts of the sensor-based system which is to be tested based on their data. Obstacles in the virtual environment such as walls and other objects can influence the simulation.

```
float refresh_rate = 20; // 20 times per second
rotation old_rotation;
rotation new_rotation;
vector old_position;
vector new_position;

default
{
    state_entry()
    {
        // Activate the timer listener every
        // 'refresh_rate' times per second
        llSetTimerEvent(1 / refresh_rate);
    }

    timer()
    {
        new_rotation = llGetRot();
        new_position = llGetPos();

        // Rotation changed?
        if (new_rotation != old_rotation)
        {
            // Position changed?
            if (new_position != old_position)
            {
                llSay(0, "<Rotation " + (string)new_rotation + "> " +
                         "<Position " + (string)new_position +">");
                old_rotation = new_rotation;
                old_position = new_position;
            }
            else
            {
                llSay(0, "<Rotation " + (string)new_rotation + ">");
                old_rotation = new_rotation;
            }
        }
        else
        {
            // Position changed?
            if (new_position != old_position)
            {
                llSay(0, "<Position " + (string)new_position + ">");
                old_position = new_position;
            }
        }
    }
}
```

Figure 4. Example of an update script for virtual objects, which informs about changes of position and/or rotation and is implemented in the Linden Scripting Language.

System Setup

In the following we explain how to integrate an existing system to our architecture. As an initial step the simulation framework has to be prepared for the specific simulation task. The developer has to register the existing sensor-based system with its hard- and software, in order to inform the simulation framework which data (from SL or other components) are required for the simulation task. This is done by registration as listener through the interface for the desired data.

In our example the sensor-based system consists of the PDA positioning software, and a simulator for the behavior of the RFID tags. The required data from SL comprises the coordinates and the orientation of the emitters, sensors, and avatars.

The Twin-World Mediator configures the message proxy according to the registered listeners. Then, the message proxy connects to SL and continuously listens (in SL) for the requested data and communicates it to the database of the Twin-World Mediator. Some objects are static and their data will be gathered only in the initial step (e. g. parts of the virtual environment like walls), whereas other objects are potentially moving; so they have to report about their changes (e. g., avatars, virtual sensors and devices). In those objects we have embedded specific LSL scripts to send the updates to the message proxy (see below).

Update Scripts for Virtual Objects

The example LSL script (shown in Fig. 4) continuously sends updates about position changes and/or rotation with a predefined refresh rate (here 20 times per second). During each refresh cycle first the rotation and the position of the object are determined by the functions llGetRot and llGetPos. The new values are then compared with the previous values in order to determine whether the rotation and/or the position of the object has changed. Only the changes are transmitted (function llSay) to keep the traffic low. It depends on the object which refresh_rate is necessary and which object's changes (e. g. rotation, position, size or color) has to be transmitted. For example, for a RFID tag only the position is important and a refresh rate of 4 times per second is sufficient. The scripts are rather easy to understand and therefore quite easy to adapt for their specific task.

One of the most important differences to previous and related work is that the message proxy uses communication methods of both the scripting language LSL, and the libsecondlife API [7]. (Libsecondlife is an unofficial API that interfaces SL as a client and enables access to data of the virtual environment.) Thus the performance can be improved, technical limitations like the time limitations of the XML-RPC method of the LSL are circumvented, and SL can be interfaced more effectively.

Simulation Update

After the system has been initialized, the Twin-World Mediator synchronizes the update loops of the sensor-based system and of SL to allow for development and testing with 3D interaction and visualization. As said before, the static data is collected only once (to save bandwidth) and sent to the database of the Twin-World Mediator.

In the second step the message proxy continuously collects (in an infinite loop) all the subscribed dynamic data and sends it to the mediator. The Twin-World Mediator notifies data changes to the listeners of the components of the sensor-based system. Then the components process this data and send the results back to the Twin-World Mediator. Data which is meant to be visualized in SL is sent via the message proxy to SL.

In the case of our example (the indoor positioning system), the coordinates of the virtual RFID tags in SL are sent to the RFID-simulator. The coordinates and orientations of avatars are also sent to the simulator. The simulator computes the results and sends them back to the Twin-World Mediator, which in turn delivers the data to the positioning system. In our case the PDA positioning system has been slightly modified in order

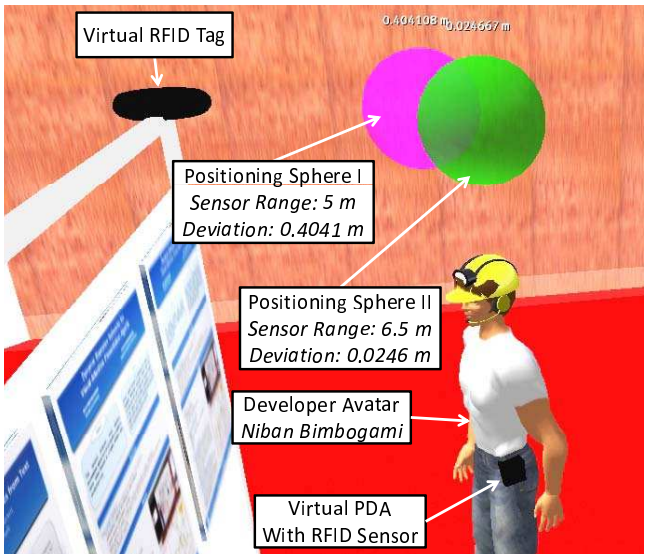

Figure 5. Basic evaluation of an indoor positioning system by comparing different sensor ranges and measuring the accuracy/deviation (i) sensor range 5 meters and (ii) sensor range 6.5 meters.

to be able (i) to receive the simulated sensor data of the virtual RFID tags, and (ii) to provide the estimated user position for the Twin-World Mediator.

The positioning system sends its results (the estimated position of the avatar) to the Twin-World Mediator. The message proxy takes this data and visualizes it in SL. We use a sphere to indicate the estimated position of the avatar in SL. In this way, the developer can inspect the results of the positioning system in SL (see Fig. 1).

In the following section we show how this can be extended for the evaluation of a sensor-based system.

EXAMPLE EVALUATION OF SENSOR-BASED SYSTEM

[4] reports about the evaluation of the accuracy of a installed tag-based location system (Ubisense [15]) and the influence of the human body on it based on the fact that many tag-based systems use communication frequencies that cannot pass easily through the human body. To measure the accuracy of the system, they use a combination of definitions suggested by [6].

As a simple example how our approach could be used for the evaluation of a sensor-based system we compare the accuracy with different system settings. To measure the accuracy of the estimated position the deviation is computed by calculating the distance between the real position of the avatar and the estimated position. The measured deviation of the estimated position is displayed on the top of the positioning sphere (see Fig. 5). Obviously, the evaluation of a sensor-based system with our approach heavily depends on the quality of the used simulators. If the simulator takes into account obstacles for the sensors (like walls or human bodies) then our approach would facilitate the evaluation of a sensor-based system and it even would enable to compare two different sensor-based systems with the same data under different conditions.

In Fig. 5, the aforementioned indoor positioning system is tested with two different sensor ranges of the RFID sensor. RFID tags are placed on a uniform grid with 6 meter spacing. The Positioning Sphere I indicates the estimated position for a RFID sensor with a sensor range of 5 meters and shows a deviation of 0.4041 meters. Positioning Sphere II, which indicates the estimated position for a RFID sensor with a sensor range of 6.5 meters, shows a better accuracy (deviation 0.0246 meters) than Positioning Sphere I. Thus the issues that have been encountered with the real system (as mentioned before in Example Application), could be evaluated in a virtual setup and lead to the conclusion that the best results are achieved with a small overlapping of the RFID signals. With the current RFID tag placement in the environment Positioning Sphere II matches this condition better than Positioning Sphere I.

So far, we compared the system behavior of the same indoor positioning system with different sensor ranges and placements of the RFID tags in the environment. But obviously, modified versions of the current underlying algorithm or completely distinct algorithms could be compared with each other (using their best sensor range and RFID tag placement in the environment). For the evaluation of systems other metrics than the deviation could be desirable. The values of these parameters can be fed into the system via the Twin-World Mediator and displayed on the top of the positioning sphere as well (by a short LSL script). In addition to text-display, color-coding and resizing can also be realized with LSL.

DISCUSSION AND CONCLUSIONS

The paper proposes a novel simulation framework based on the 3D virtual environment of SL, which can be used as an evaluation testbed for sensor-based systems. A core feature of our approach is the bi-directional interaction with our Twin-World Mediator. Events from the real world are reflected to the virtual world and vice versa.

With the example of a positioning system we have illustrated how our simulation framework can be used and how the virtual environment can be utilized for evaluation and optimization purposes.

Furthermore, the architecture is flexible and *extensible* and thus ensures that new sensor types, such as temperature sensors, accelerometers, or light sensors, can be included. Spatial characteristics of devices can be modeled and visualized to easily identify problems and interferences, e. g. when walls or other objects in the virtual environment influence the characteristics of the devices. Sophisticated simulators can also be adapted and connected to the system. Metrics for evaluation

purposes can be updated via the Twin-World Mediator and visualized in SL.

The *3D interaction* capability of SL combined with the embedded simulators offers many advantages and opportunities. Virtual sensors and devices can be moved intuitively by 'direct' (avatar-mediated) manipulation (Fig. 1) and their parameters can also be changed easily by editing the object properties through the user interface of SL. Most importantly, these changes can also be fed back in real-time via the Twin-World Mediator and affect the connected system.

In our future work, we plan to implement a user-friendly interface and toolbox for developers of ubiquitous computing systems. To reach a broader audience and ensure higher flexibility, the Twin-World Mediator will be adapted to the emerging and open-source virtual worlds system OpenSimulator [13]. Furthermore, we intend to use the simulation framework for running systematic experiments of sensor-based systems. Specifically, computer-controlled agents, i.e. SL "bots", will populate the environment, and the behavior of the sensor-based system will be evaluated in the multi-agent setting.

REFERENCES

1. I. Armac and D. Retkowitz. Simulation of smart environments. In *IEEE International Conference on Pervasive Services (ICPS'07)*, pages 257–266. IEEE Computer Society Press, 2007.
2. B. Brandherm and T. Schwartz. Geo referenced dynamic Bayesian networks for user positioning on mobile systems. In *Proceedings 1st International Workshop on Location- and Context-Awareness (LoCA'05)*, pages 223–234. Springer LNCS 3479, 2005.
3. B. Brandherm, S. Ullrich, and H. Prendinger. Simulation of sensor-based tracking in Second Life. In *Proceedings of 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 1689–1690. ACM Press, 2008.
4. L. Coyle, J. Ye, E. Loureiro, S. Knox, S. Dobson, and P. Nixon. A proposed approach to evaluate the accuracy of tag-based location systems. In *USE 07: Workshop on Ubiquitous Systems Evaluation. UbiComp Workshop Proceedings*, pages 292–296, 2007.
5. A. Fundinger. Real Life Control Panel for Second Life. http://channel3b.wordpress.com/2007/01/24/real-life-control-panel-for-second-life/, January 2007.
6. J. Hightower and G. Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, 2001.
7. libsecondlife. `http://www.libsecondlife.org`.
8. J. Lifton, M. Feldmeier, Y. Ono, C. Lewis, and J. A. Paradiso. A Platform for Ubiquitous Sensor Deployment in Occupational and Domestic Environments. In *Proceedings of the 6th International Symposium on Information Processing in Sensor Networks (IPSN)*, pages 119–127, 2007.
9. M. Musolesi, E. Miluzzo, N. D. Lane, S. B. Eisenman, T. Choudhury, and A. T. Campbell. The second life of a sensor: Integrating real-world experience in virtual worlds using mobile phones. In *Proceedings of 5th Workshop on Embedded Networked Sensors (HotEmNets 2008)*. ACM, 2008.
10. H. Nishikawa, S. Yamamoto, M. Tamai, K. Nishigaki, T. Kitani, N. Shibata, K. Yasumoto, and M. Ito. UbiREAL: Realistic smartspace simulator for systematic testing. In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp2006)*, Springer LNCS 4206, pages 459–476, 2006.
11. Y. Oh, A. Schmidt, and W. Woo. Designing, developing, and evaluating context-aware systems. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 1158–1163. IEEE Computer Society, 2007.
12. E. O'Neill, M. Klepal, D. Lewis, T. O'Donnell, D. O'Sullivan, and D. Pesch. A testbed for evaluating human interaction with ubiquitous computing environments. In *Proceedings of the 1st International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, pages 60–69. IEEE Computer Society, 2005.
13. OpenSimulator. `http://opensimulator.org`.
14. Second Life. `http://secondlife.com`.
15. P. Steggles and S. Gschwind. Ubisense-a smart space platform. Technical report, Ubisense, May 2005.
16. S. Ullrich, B. Brandherm, and H. Prendinger. Simulation Framework with Testbed for Sensor-based Systems in Second Life. In *Proceedings of 10th International Conference on Ubiquitous Computing (UbiComp 2008) Demo Session*. ACM, 2008.
17. G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: a wireless sensor network testbed. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 68–73. IEEE Press, 2005.