PRT Simulation in an Immersive Virtual World

Cristina V. Lopes
Bren School of Information and Computer Sciences
University of California, Irvine
lopes@ics.uci.edu

Lorraine Kan
Bren School of Information and Computer Sciences
University of California, Irvine
lkan@uci.edu

Anton Popov
Bren School of Information and Computer Sciences
University of California, Irvine
apopov@uci.edu

Ricardo Morla
Faculdade de Engenharia
Universidade do Porto
Porto, Portugal
ricardo.morla@gmail.com

ABSTRACT

Immersive virtual world environments, such as Second Life^{TM} (SL), have the potential to dramatically improve the process of analyzing usability within technically correct system simulations, long before the system is built. We report our findings with the SL simulation of a Personal Rapid Transit (PRT) system. The SL model and simulation were done according to the original technical specifications. In interacting with this simulation, the system designers were able to identify several usability issues that would have gone unnoticed in a non-immersive simulation environment. Namely: (1) a problem with the design of the offramp to the station; (2) further requirements for the design of the top of the vehicles, so that the suspended track is out of direct sight of the people inside; (3) further safety requirements for dealing with unexpected obstacles along the path.

While all of these issues would have been identified upon deployment of the physical prototype, the contribution of our work is to show how usability issues like these can now be identified much earlier, using simulations in a virtual world.

Categories and Subject Descriptors

I.6.7 [**Simulation and Modeling**]: Simulation Support Systems; I.6.8 [**Simulation and Modeling**]: Types of Simulation—*Animation, Combined, Visual*

General Terms

Design, Human Factors

Keywords

Simulation, Virtual Worlds, Complex Engineering Systems

1. INTRODUCTION

Most engineering simulation tools focus on data, function, and operational performance as the main objects of simulation [6]. Usability by people is usually modeled as additional data following certain patterns of behavior, and then simulated using a variety of techniques. This practice is inherently limited, because real people tend to behave in unpredictable ways when using complex systems deployed in the real world.

Over the past decades, there has been considerable work in immersive virtual reality environments capable of supporting the virtual presence of people in those environments. The vast majority of this work has been in the military, and it is usually associated with sophisticated immersion techniques that may include special goggles, displays, and assorted instrumentation for people to wear. Recently, there has been a growing interest in virtual reality environments that can be interfaced through any ordinary computer. Examples include massive multi-player on-line games such as World of Warcraft [11], Club Penguin [2], and general purpose, user-programmable virtual world platforms such as Second Life [9] (SL) and Croquet [3]. This last group is particularly relevant for engineering systems. A visit to any part of SL reveals complex user-centered interactive virtual systems programmed by SL's users, suggesting the potential for combining technical simulation with immersive user experience at a level unseen before.

We describe a case study of using SL as the modeling and simulation environment of a real-world complex engineering system: a Personal Rapid Transit system (PRT) being developed by a company in Southern California. According to Wikipedia, PRT is "a public transportation concept that offers automated on demand non-stop transportation, on a network of specially built guideways." This concept accommodates a variety of hardware solutions that range from cars on rubber wheels guided by a supporting guideway, to cars suspended on guideways through magnetic levitation. The system we modeled uses suspension through magnetic levitation on guideways made of large numbers of special "bricks." Fig. 1 depicts concept images of this system. The first physical prototype being developed consists of a simple loop with one single station.

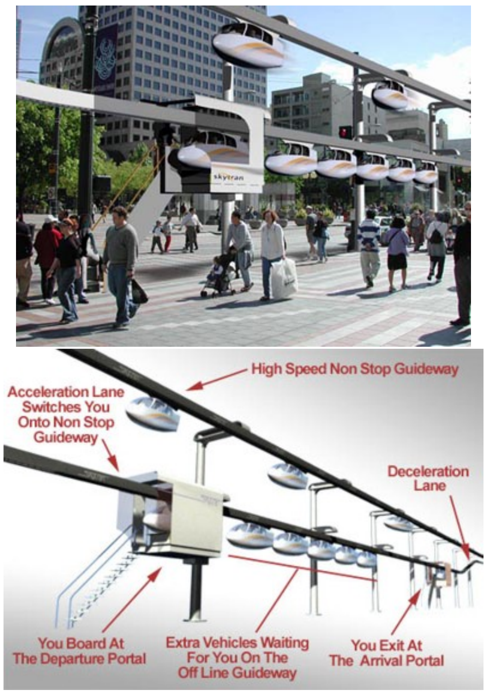
skytran
High Speed Non Stop Guideway
Acceleration Lane Switches You Onto Non Stop Guideway
Deceleration Lane
You Board At The Departure Portal
Extra Vehicles Waiting For You On The Off Line Guideway
You Exit At The Arrival Portal

Figure 1: Concept images of the PRT system of this study.

The purpose of this work was twofold. First, we wanted to find out if and to what extent, SL, the most popular general-purpose virtual world platform, can be used as a systems simulation tool by supporting technically accurate virtualizations of complex engineering systems; second, we wanted to find out if and to what extent those virtualizations are useful in the process of designing the systems. Our case study shows that (a) SL is capable of supporting technically accurate virtualizations of these systems, within certain limits; and (b) the virtualization of the system allowed the detection of several design issues, some of which were purely technical while others involved the system's usage and its context.

Second Life was chosen as the simulation platform because of its accessibility and the pre-existing virtual places created by other SL users. SL is free for all to use on desktop or laptop computers running Windows, Mac, and Linux. The simulation environment is accessed through the point of view of a customizable avatar. The avatar interacts with other avatars and manipulates nearby objects using keyboard and mouse inputs. SL's avatar based user-interface makes it easy for users to explore and increases the immersiveness of their experience. Additionally, SL already contains more than a million active members at the time of this publication. Other SL users have created many different virtual places, with some that mimic their real life counter parts. The pre-existing virtual locations will be excellent for deploying our simulation to create public awareness and to gain feedback about the system. Using SL does have one drawback: the physics engine in the system was designed to emulate only a limited amount of physical properties. However, our research is not focused towards the physics portion of the system so the limited physics engine proved to be adequate.

The remainder of the paper is organized as follows. Section 2 describes what the PRT system does, and how we implemented it in SL. Section 3 presents the findings of the study. Section 4 discusses related work, and Section 5 concludes the paper.

2. MODELING AND SIMULATION

The process used in this study was as follows. The PRT company's engineers provided us the specifications of their system, and we developed the SL model and simulation. Additionally, we had several meetings with them to clarify parts of their design.

The PRT system of the original specifications consists of a layered architecture that goes from the physical to the application layer (there are some similarities to the OSI model). Our simulation focused on the logic control software, i.e. the software that controls the vehicles' departure from the station onto the main driveway and that ensures that all vehicles remain in their assigned positions on the guideway; so, flow control problem reminiscent of the data link layer in the OSI model. We emulated the physical system to the point that we could capture all the meaningful input parameters for this control.

This section describes the major components of the model and simulation, starting with a brief introduction to SL programming which was used extensively.

2.1 Second Life Programming

SL is an end-user, fully programmable 3D virtual world with powerful features for real-world systems simulation such as a variety of sensors (touch, motion, and presence) and actuators (e.g. for generating light and sound, moving and spinning objects), as well as support for communications between simulated objects. Inworld programming of virtual objects is done using LSL, a C-like event- and state-based scripting language. Each object can hold a collection of concurrently-running LSL scripts that implement the object's behavior. LSL scripts can react to 37 kinds of events from the simulated environment and invoke the 300+ functions of the Second Life LL API. Table 1 shows a few examples of events and functions of SL.

SL applications are divided into server-side, where the LSL scripts run, and client-side, where the 3D graphics engine renders the 3D virtual world data sent by the SL servers. Given that the scripts execute on the server, all clients are able to observe the scripts in action – a key feature of SL that enables the simultaneous observation of the world's state by all avatars nearby.

2.2 Modeling of Real-World Objects

One of the most important components of any PRT system is the guideway. The guideway of the particular system we studied is made of a sequence of "bricks" of about 50cm, each one comprising of magnetic levitation hardware, sensors, and communication capabilities. As stated before, the first prototype consists of a loop with an on/off-ramp to one single station; this guideway has been carefully designed by

Communications	
llSay	Broadcasts a message on a channel number; 20m range.
listen	Calls user-defined message handling code.
Sensing	
llVolumeDetect	Sets collision detection in an object's volume.
collision_start	Calls user-defined collision handling code.
Control	
llSetPos	Sets an object's position in the 3D space of the simulation region.
sensor	Calls user-defined handling code for in-range object events.
Object Creation	
llRezObject	Creates a new object on a given position in the 3D space of the simulation region.
on_rez	Calls user-defined constructor code of the created object.

Table 1: Example LSL functions and events.

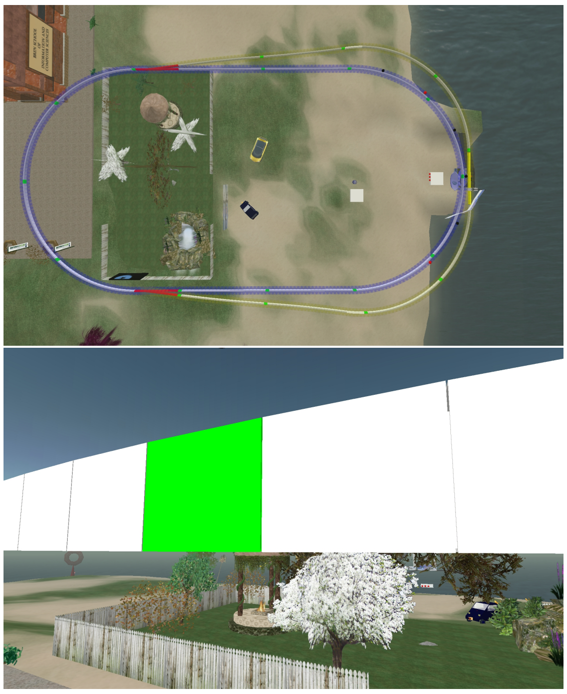

Figure 2: The guideway and its individual bricks.

the company's engineers to account for the exact position and rotation of each brick. Fig. 2 shows the overview of the SL model guideway and a detail highlighting the individual bricks. There are 312 bricks on the upper guideway, and 238 bricks on the lower one (the ramp/station).

The bricks, vehicle, and station are built and positioned programmatically by a creation object we developed. This allows for the PRT simulation to be deployed in different SL cities and immediately become a part of the environment for users to experience.

For the purposes of the logic control we were studying, we didn't need detailed physical modeling of the system. This was a good match with SL, because the current physics engine of SL is quite limited. Physical phenomena such as wind can be emulated by supplying additional input parameters to our simulation that affect the behavior of the vehicles; the controller simply needs to perform the appropriate procedures to deal with such vehicle errors.

From	**To**	**Message**
Brick	Station	*Vehicle detected at brick n*
Vehicle	Station	*Request to move*
Vehicle	Station	*Notify stop*
Station	Vehicle	*Accept/deny request to move*
Station	Vehicle	*When to start moving*

Table 2: Communications in the simulated PRT system.

2.3 Simulation

2.3.1 Sensing

Each of the bricks that compose the real-world system guideway contains sensors that capture the passing vehicle's presence and ID. As the top of the vehicle moves through the inside of a brick, the sensors are triggered and a detection notification is sent to the station. The station uses the information to determine if the vehicle is at the correct location and takes corrective action if necessary. The sensors in the real-world system can also detect how fast the vehicle is traveling based on the rate of change of positions.

We implemented the equivalent sensing mechanism in SL by equipping the individual bricks with a sensor for presence and identification, and notifying the station of the detected vehicle. The sensor is implemented using the SL function `llSensorRepeat` on each brick. `llSensorRepeat` performs a scan within a specified sensor range every half a second within 0.3 meters of the brick's center. When `llSensorRepeat` detects an object passing through the brick, the event `sensor` is triggered. the vehicle's unique identifier is then retrieved using the SL function `llDetectedKey`.

2.3.2 Communications

Table 2 summarizes the messages that are sent between the several components of the system.

In the real-world system, the communications medium is still an open design decision. In SL, we used an emulation of wireless communications using SL's channels: communication between the system's different objects is passed as plain text through selected channels using the SL function `llShout`. Each object listens to a specific set of channel(s) using `llListen` and parses the text message to determine the next action. The communication in SL is sent immediately regardless of the distance of the recipient object, but, just as in the real physical medium, there is the possibility of congestion.

2.3.3 Flow Control

The system requires the vehicles to keep a specific amount of distance between each other. To enforce the distance, the station pre-determines the amount of slots available for a given guideway. When the vehicle requests to move, the station assigns an empty slot and notifies the vehicle when it is cleared to go. The vehicle then accelerates and merges with the main guideway at the correct time in order to be at the assigned slot. The station then marks the slot as occupied until the vehicle returns to the station and sends a notification that it has stopped. If a vehicle requests to move and the station has already assigned all available slots, the vehicle will not be cleared to move and is forced to wait for a free slot.

3. FINDINGS: THE IMMERSION FACTOR

Design mistakes are a normal and unavoidable part of any design process. The purpose of engineering simulations, in general, is to identify design mistakes and other issues of concern, long before the systems are built. The more problems and insights are identified during simulation, the more successful that simulation is considered to be. Conversely, if no problems or insights are identified during simulation that is usually an indicator that the simulation itself is inappropriate, since it is highly unlikely that the design is flawless.

As stated before, the goal of this study was to investigate the appropriateness of SL as simulation tool for complex engineering systems through this particular case study. There were two parts to this goal: (1) feasibility of a technically accurate virtualization of the system; and (2) study of the kinds of design issues that would be identified, if any, with such a simulation.

With respect to the first part, and as described here, we were able to successfully produce a fairly accurate virtualization of the PRT system. At several times during this study we discussed the need for more detailed physical modeling. Since the focus was on the logic control, we were able to make progress without having to model the physical phenomena more accurately. This was accomplished by emulating the physical phenomena as additional input parameters to the scripts in the object models. For example, the effect of wind on the vehicles is emulated by an additional random error parameter on the movement of the vehicle.

Our conclusion is that, as of now, SL is inappropriate for detailed physical modeling. If that is the focus of a simulation, then another tool should be used.[1] But for simulations pertaining to logic control problems in physical systems, SL proved to be an excellent fit. Its virtualization capabilities, along with the programming API, make a powerful combination for general-purpose physical systems simulation.

With respect to the second part of our goal (i.e. detection and nature of design issues), our findings exceeded our expectations. Not only the simulation detected routine technical mistakes, but it also enabled the detection of additional mistakes of a very different nature: usability. Table 3 lists the main design issues that were identified with this simulation.

SL has two properties that are rarely seen in engineering simulation tools: (1) the virtual presence of people through "avatars;" and (2) the loose control of the environment where the virtual model is deployed – that is, the site has public access, and other people have been doing virtual constructions in it. We call these two properties the "immersion factor." Fig. 3 illustrates this immersion.

[1]We note, however, that the SL architecture is not tied to any particular physics engine. Documents describing the future of SL mention the possibility of varying the specific physics engine for different simulators of the SL network. Therefore, it is predicted that detailed physical modeling will be supported in the future.

	Design Issue
1	*Guideway calculation errors.*
2	*Guideway stop section not big enough.*
3	*View of upper guideway at the station too uncomfortable.*
4	*Direct sight of the guideway on moving vehicle might cause panic or epileptic attacks.*
5	*Unexpected Obstacles.*

Table 3: Design issues identified with this simulation.

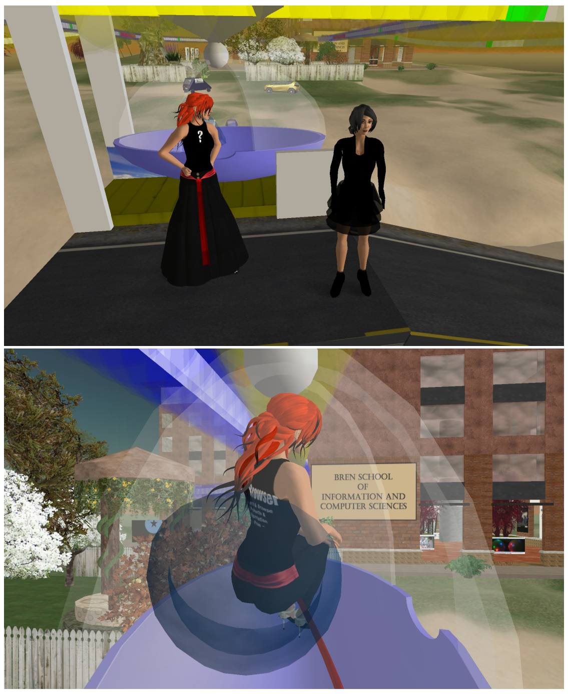

Figure 3: Top: avatars at the station. Bottom: an avatar riding one of the cars.

Looking at Table 3, we notice that issues 1 and 2 would have also been detected with a non-immersive simulation environment. For 1: the initial guideway specification used the wrong radius for the arcs' curvatures, which caused the bricks to not line up correctly. For 2: initially, the stop section at the station was only 4 bricks long, which wasn't even enough for one single car. These are routine mistakes in any physical modeling process, and they are usually detected fairly early, either by unfolding the calculation on a drawing, or by modeling the artifact in a modeling tool. In our case, they were immediately detected as soon as we started laying the bricks in space. In turn, the engineers promptly redid the calculations and generated the correct specifications.

Issues 3, 4, and 5, however, have a different nature. They have to do with usability of the system by people, and the

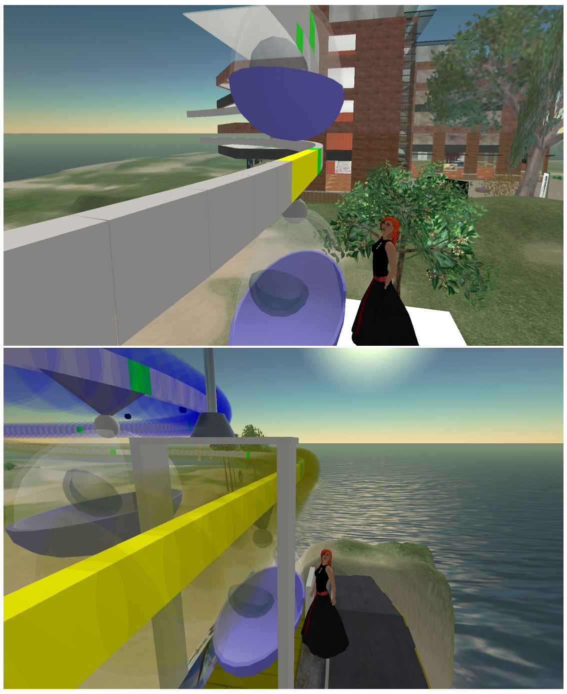
Figure 4: Design problem in a prior version of the guideway. Top image: the original station design. Bottom image: the redesigned station.

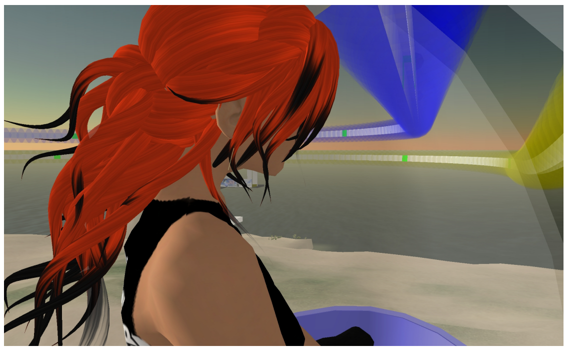
Figure 5: View forward, from inside the car.

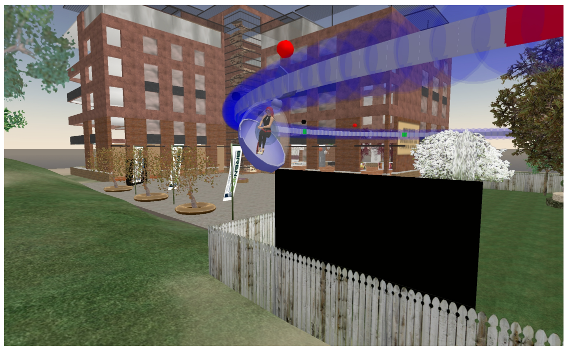
Figure 6: Obstacle along the path.

context in which the system is deployed. They were detected because of the immersion factor. The remainder of this section focuses on these.

Upper guideway at the station. Fig. 4 illustrates the situation. In the initial design, the two guideways aligned vertically according to the concept images shown in Fig. 1. As such, the vehicles moving at high speed in the upper guideway would pass on top of the platform where people stand. This looked nice in the concept image. However, when we placed the avatar at the station and looked up through the "mouse view" (a view of the world through the avatar's eyes), the view of the vehicles on top made this design feel uncomfortable, if not unsafe. Once this was detected, the engineers promptly redid the calculations of the guideway, and generated the specifications leading to the new guideway seen on the bottom of Fig. 4. In this second design, the lower guideway shifted horizontally outwards from the upper one. As such, the moving vehicles on the upper guideway are at a confortable angle and distance from the people at the station.

Direct sight of the guideway on moving vehicle may cause panic or epileptic attacks. Fig. 5 attempts at illustrating the situation, although it can only be fully perceived while riding on the moving vehicle. The problem here is that seeing the guideway on top as the vehicle moves can be highly unsettling, especially for people who are prone to panic or epileptic attacks. The system engineers detected this problem the first time they interacted with the simulation using the "mouse look" view, i.e. the view of the world through the avatar's eyes. As a result, they issued further requirements for the vehicles so that the view of the top is obstructed as much as possible.

Unexpected Obstacles. Fig. 6 illustrates this situation. This problem is predictable even without any simulation. Any system that involves autonomous vehicles must account for unexpected obstacles, and the engineers were aware of that. However, the simulation made it an urgent concern. In this case, the interference of the obstacle along the path happened in a natural manner: we wanted to deploy the system near a pre-existing building, to make it more realistic. Along with the building, there were several other objects already there, including a garden with trees and a movie screen – in Fig. 6, the vehicle is hitting this movie screen. All that construction had been done before, and was unrelated to the PRT simulation. Real-world deployments must take into account the existing environment, and avoid existing obstacles, or take them down, so this particular situation would have been solved in real life by taking down the movie screen and some trees. However, unless the guideway is relentlessly monitored, it will be impossible to control obstacles along the path *after* the guideway is deployed: vandals may place them on purpose; nearby trees may grow onto the path; etc. For the engineers, it became clear that if their PRT technology is to be accepted, they need to address collision avoidance right from the start.

Further Observation on Deployment. The ability to generate the track anywhere in SL provides the system engineers with the opportunity to see and minimize the visual impact of the track in different environments such as cities, parks, and natural habitats. The large collection of environments built by SL users, contribute to the immersion factor and provide a unique way to test and minimize the visual footprint of the track. Other, non-immersive simulation tools do not have a readily available collection of environments which render the visual-impact test impossible. Furthermore, the simulation exists and runs on Second Life even if we, the creators, are not there, allowing anyone to experience it at any time.

Public Access. The virtual PRT system we developed is publicly accessible in `secondlife://TechCoast`. Anyone with an SL account can visit and interact with the system.

4. RELATED WORK

As mentioned before, most engineering modeling and simulation tools focus on data, functionality, and operational performance. For example, modeling tools such as LabView [6] and Matlab [7] provide numerous toolbox libraries and textual and graphical interfaces for composing system models that provide complex data processing functionality. Applications of these tools include sound and image processing, communication systems modeling, financial analysis, and control system design. General purpose simulation tools such as Simulink [10] and C++SIM [4] also focus on supporting the performance assessment of e.g. algorithm and models, providing basic simulation functionality such as random number generators, queuing algorithms, and event generation and management. Similarly, domain-specific simulators provide constructs that make it easier to build systems in those domains and to assess their performance. For example, the ns network simulator [8], *en route* to its third version includes simulation objects for most of the OSI stack e.g. at the physical layer (e.g. simulation of two way ray propagation in wireless networks) and at the MAC, IP, and TCP layers, as well as logging throughout the different layers and simulation objects. The same also happens in the domain of transportation systems, where traffic simulators such as [19] can be used to determine the performance (traffic congestion) of a city or freeway grid, incorporating simulation details such as multi-lane streets with lane changing, lane-to-lane connections, and different vehicle types. Research based on these simulation tools is often based on batches of numerous simulation runs with different input parameters, mostly random, the output of which is typically processed to provide statistically meaningful results. While immersive virtual world environments such as SL provide the programming building blocks for developing technically correct simulations, they are not designed for batch runs or to run in faster than real time. In fact, in SL it is typically a human-controlled avatar that triggers the response of the simulated system, which does not lend it self to the same number of simulation runs per time unit that a simulation system without a human in the loop can perform.

Architecture-wise, the SL virtual world is accessible online, much like web-based simulations [16]. Moreover, SL is designed to support multiple users connected simultaneously to a grid of simulation servers. The servers communicate with each other through UDP connections and the client-server protocol is open source. This is reminiscent of a number of simulation and emulation architectures [14, 15, 20] although its general purpose nature does not lie on the different types of simulators that can be included in the simulation architecture, as is the case with [14, 15, 20]. Rather, SL's general purpose nature lies in the variety of interactive 3D objects that can be simulated – much like what happens in [21] for grid and P2P applications.

To some extent the ability to identify usability issues on SL derives from its 3D support. SL has embedded 3D object creation tools and can import 3D objects as sculpted textures from other 3D tools such as Blender [1] and SketchUp [5]. Although SketchUp can export 3D objects into Google Earth (which is, to some extent, a virtual environment), neither SketchUp nor Blender objects can be programmed and given a simulated behavior – unlike SL objects. A variety of 3D virtual worlds have been developed and used so far but fall short of having the potential for improving the usability analysis of technically correct systems. Namely, they do not have one or more of the following properties: fully online (e.g. UbiWise [12], a ubiquitous computing-focused 3D simulation environment); they are not general purpose nor end-user programmable (e.g. WoW [11]); or they have not reached the same growth and critical mass as SL (e.g. Croquet [3]) that supports deploying a simulation prototype with a large number of people that can individually assess the usability of that simulation prototype.

There are a number of research efforts into the issue of assessing the usability of a prototype without fully developing it. The "Wizard of Oz" approach is often explored [18] in which an application whose functionality has not been fully developed is tried out and evaluated with users. The application functionality is typically provided by a researcher that is hidden from the user or by model-based simulations [13]. This has a number of disadvantages, e.g. incorrect model simulation and technical issues hiding the researcher/wizard. With SL simulation prototypes, a technically correct simulation of a somewhat complex system (e.g. the PRT system that we developed) can be quickly developed and the usability of that system explored without need for a Wizard of Oz. Providing technically correct functionality is not the only issue when evaluating usability with virtual environments; the interface with the user plays an important role and is often used as an argument against using virtual environments as tools for helping assessing usability. However, recent research [17] shows that even crude foam board-based user interfaces can prove useful when exploring how prototypes can be used – in this case a smart home prototype was explored. This points to the direction in which 3D virtual reality simulations can be used to explore the usability of real world prototypes such as the PRT simulation presented in this paper.

5. CONCLUSIONS

We have presented a case study of using the virtual world Second Life as simulation tool for a complex real-world engineering system. SL supports 3D virtualization of objects and people, along with a general-purpose programming API that allows the association of behavior with the virtual objects. While we found SL to have several limitations in terms of physical modeling, we found it to be very effective for simulating logic control of physical objects. Our SL simulation of the real-world PRT system allowed the system engineers to detect several design mistakes and concerns, including some of usability nature – a kind of design concern that is rarely detected with engineering simulation tools.

Modeling tools such as Google's SketchUp provide solid support for detailed 3D modeling; advanced physics engines being developed by Intel and others will enable accurate physics modeling; and finally user-programmable virtual worlds such as SL allow the deployment of interactive 3D constructions with sophisticated sensing, communicating, and control behavior, in relatively uncontrolled social environments. The convergence of these three technologies may revolutionize the process of design in engineering.

6. ACKNOWLEDGMENTS

Dr. Lopes is partially supported by the US National Science Foundation grant No. CCF-0347902. Dr. Morla was supported by the Portuguese Fundação para a Ciência e Tecnologia. The authors would like to thank Unimodal Inc., Irvine, CA for sharing their confidential design documents, and for their generous gift supporting our research on simulation in virtual worlds.

7. REFERENCES

[1] Blender web site. http://www.blender.org/.

[2] Club Penguin Community Site. http://www.clubpenguin.com/.

[3] Croquet Consortium. http://www.opencroquet.org/.

[4] C++SIM web site. http://cxxsim.ncl.ac.uk/.

[5] Google SketchUp. http://sketchup.google.com/.

[6] LabView web site. http://www.ni.com/labview/.

[7] Matlab web site. http://www.mathworks.com/products/matlab/.

[8] ns-3 Network Simulator web site. http://www.nsnam.org/.

[9] Second Life Community Site. http://www.secondlife.com/.

[10] Simulink web site. http://www.mathworks.com/products/simulink/.

[11] World of Warcraft Community Site. http://www.worldofwarcraft.com/.

[12] J. J. Barton and V. Vijayaraghavan. UBIWISE, A ubiquitous wireless infrastructure simulation environment. HPLabs Technical Report HPL-2002-303, 2002.

[13] R. Chatley, J. Kramer, J. Magee, and S. Uchitel. Model-based simulation of web applications for usability assessment. In *ICSE Workshop on SE-HCI*, 2003.

[14] J. S. Dahmann, R. M. Fujimoto, and R. M. Weatherly. The department of defense high level architecture. In *29th Winter Simulation Conference*, 1997.

[15] O. Dalle. The osa project: an example of component based software engineering techniques applied to simulation. In *Summer Computer Simulation Conference (SCSC'07)*, 2007.

[16] A. D'Ambrogio and D. Gianni. Using corba to enhance hla interoperability in distributed and web-based simulation. In *19th International Symposium on Computer and Information Sciences (ISCIS'04)*, 2004.

[17] S. Davidoff, M. K. Lee, A. K. Dey, and J. Zimmerman. Rapidly exploring application design through speed dating. In *Conference on Ubiquitous Computing (Ubicomp'07)*, 2007.

[18] S. Dow, B. MacIntyre, J. Lee, C. Oezbek, J. D. Bolter, and M. Gandy. Wizard of oz support throughout an iterative design process. *IEEE Pervasive Computing*, 2005.

[19] D. Krajzewicz, G. Hertkorn, P. Wagner, and C. Rossel. An example of microscopic car models validation using the open source traffic simulation sumo. In *14th European Simulation Symposium*, 2002.

[20] R. Morla and N. Davies. Evaluating a location-based application: A hybrid test and simulation environment. *IEEE Pervasive Computing*, 2004.

[21] M. Quinson. Gras: A research & development framework for grid and p2p infrastructures. In *18th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2006)*, 2006.