8-2014

Overcoming Roadblocks in Introducing Virtual World Technology to High Schools

Casey Dylan Bailey
University of Arkansas, Fayetteville

Overcoming Roadblocks in Introducing Virtual World Technology to High Schools

Overcoming Roadblocks in Introducing Virtual World Technology to High Schools

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science

by

Casey Bailey
University of Arkansas
Bachelor of Science in Computer Science, 2010

August 2014
University of Arkansas

This thesis is approved for recommendation to the Graduate Council.

_____
Dr. Craig Thompson
Thesis Director

_____
Dr. Gordon Beavers
Committee Member

_____
Dr. Susan Gauch
Committee Member

ABSTRACT

The EAST (Environmental And Spatial Technology) Initiative is a non-profit educational organization that provides students in over two hundred schools in eight states with access to advanced computing technologies for the purpose of enabling students to develop technical skills early and to produce solutions to local community problems. Although many high-end technologies are available through EAST, they are desktop solutions that individual students use and there are none that enable students within a school or between schools to collaborate.

This thesis is a saga that documents the identification and removal of many roadblocks to introducing a 3D multi-user virtual simulation platform known as OpenSimulator into an EAST high school, Greenland High, located in Northwest Arkansas. The end result seemed compelling, simple and achievable -- with OpenSimulator, students from one or many EAST Labs would be able connect, chat, and work together within the same or nearby virtual areas to build models of (parts and aspects of) their communities. But getting to the point where students can begin to use this platform involved solving cost, safety, firewall, administrative, sustainability, and other puzzles. Most of this thesis is concerned with solving problems up to introducing OpenSimulator to Greenland -- more work is needed in understanding whether and how this kind of technology will benefit high school computing programs like EAST.

ACKNOWLEDGEMENTS

First and foremost, I thank my advisor, Dr. Craig Thompson, for never-ending support and patience. I would also like to thank my thesis committee members for their time spent reviewing and helping. Finally, I thank my family and friends for their constant love and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 Problem

The EAST Initiative (http://www.eastproject.org) is a non-profit organization seeking to increase high school students' experience and skill with modern, high-end technologies especially related to computing. These include commercial grade software packages spanning a variety of purposes: 3D animation with SoftImage XSI and Blender, drafting via AutoCAD and Microstation/J, web design and development, database administration, Global Positioning Systems, Geographic Information Systems, and Microsoft's Visual Studio programming suite [1]. Students are tasked with the goal of learning about and then using the available technology to implement projects which help their local community in some way.

One of the most interesting aspects of the initiative is that student work is self-selected – an EAST facilitator oversees the projects but does not teach and there is no set curriculum. Students are provided with the software, but the learning is self-driven. This results in students working together to experiment and develop skills with the provided technology rather than simply listening to an instructor lecture. The program has been successful since inception in 1996, both in its ability to encourage self-driven learning and in improving students' technical capacity by encouraging student-student cooperation.

One area that is lacking, however, is the ability for EAST students within a school or across the nation to collaborate with one another. This missing ingredient seems especially noticeably absent in an era when social networking tools like Facebook and 3D multiuser interactive gaming environments are commanding students' time and attention. EAST has shown that students teaching students is an effective method of learning, but thus far, only students within the same lab have the ability to teach one another, and then only because they are

physically near one another, not because their computing environment enables cooperative work. Given the 200 high schools participating, providing a method for schools to interact and work together seems likely to increase the technical skill development EAST enables.

1.2 Background

A long-term research project at the University of Arkansas Computer Science and Computer Engineering Department is Dr. Craig Thompson's *Everything is Alive* project, which focuses on pervasive computing and especially how to use technologies like agents (from Artificial Intelligence), radio frequency identification (RFID), and virtual worlds to build a smart world where every human and object can interact. In 2003-2004, the project focused on agent-based systems; in 2004-2006, the project focused on RFID middleware; and since then it has used 3D virtual worlds, especially Second Life, as a vehicle to demonstrate what a future world will be like when humans and objects can communicate and interact more as equals.

To make the project concrete, often we demonstrated our results using applications in the healthcare or supply chain domains as these domains seem to be early adopters of identity management and smart world technologies and might benefit from having virtual world simulations before general consumer applications (see Figure 1).

Figure 1: The EiA Project's "Hogspital" on University of Arkansas island in Second Life

The EiA project website is http://vw.ddns.uark.edu. The website provides a *Demos* tab listing several YouTube videos of project results and also a *Documents* tab that lists over thirty document project results.

1.3 Thesis Statement

This thesis documents a method for making virtual world technology available to high school students for the purpose of enabling widespread collaboration and transmission of computing skills within and between secondary schools with the intention to enhance students' ability to develop technical and collaborative skills.

1.3 Approach

The approach used in this thesis required three main steps:

- Construct a backbone grid of virtual world servers implemented on commercial commodity hardware. Configure and deploy the virtual world servers behind a router in a University research lab. Implement this solution ourselves, allowing us to maintain complete control over security, access, maintenance, and daily operations. This hardware-software platform is the foundation for both the OpenSimulator virtual world platform and associated project administrative tools.
- Deploy a modified version of the OpenSimulator server software on the backbone. This served to create a distributed virtual world building engine capable of scaling with the addition of hardware. Heavy modification and configuration were necessary to guarantee the high degree of security required when working with high school students. In addition, a widely-accessible yet centrally managed server

solution was low on the OpenSimulator development list, so much trial and error digging through code was a necessity.

- Develop and deploy a method of safely and efficiently allowing high school students access to the environment. Due to strict privacy standards and bureaucracy, this portion of the project showed itself to be at least as challenging as the technical portions. The Children's Online Privacy Protection Act of 1998 (http://www.coppa.org) sets strict guidelines regarding child privacy and security during online activities. It includes provisions setting standards for obtaining consent from a parent or guardian and keeping the students safe online. As a result, this project had to guarantee safety and security for the students involved.

1.4 Organization of this Thesis

Chapter 2 covers the project's background concepts and research. Chapter 3 discusses roadblocks encountered as well as the project's architectural evolution. Chapter 4 describes testing, results, and data analysis. Chapter 5 offers a final project summary as well as potential future work and impacts.

2. BACKGROUND

2.1 Key Concepts

As background for this thesis, this section describes 3D multi-user virtual worlds including the history of virtual worlds, commercial Second Life virtual world, its open source variant Open Simulator, and other virtual world commercial and research projects

2.1.1 Virtual World Development

A virtual world is a multi-player online 3D environment where users may build, script, and interact inside a simulated, dynamic world. The ancestry of today's virtual world include the Sensorama (see Figure 2), the first "virtual reality" machine, which was a mechanical device built as a prototype by Morton Heilig in 1962. The Sensorama worked by incorporating stereoscopic 3D images, physical body tilting, and both aroma and wind tunnels. The Sensorama had limited commercial success and was deployed in shopping malls and carnivals.

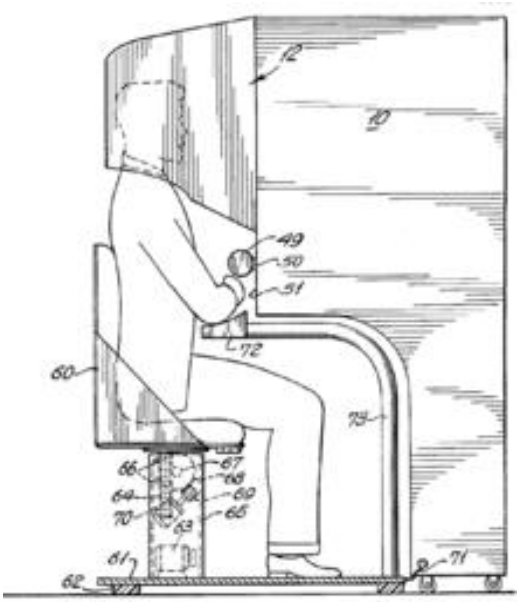

Figure 2: Heilig's Sensorama -- 1962

Before virtual worlds came graphics and games. In 1968, Ivan Sutherland (Turing Award Winner 1988) created Sketchpad (Fig 3), a first prototype of the sort of device seen in cheesy 1980s movies. Sketchpad enabled naive users to build 2D models of "objects" using a Graphical User Interface – the first ever. Although Sketchpad was no virtual world, it was one of the first computing applications to allow user-driven creation of "object" type content.

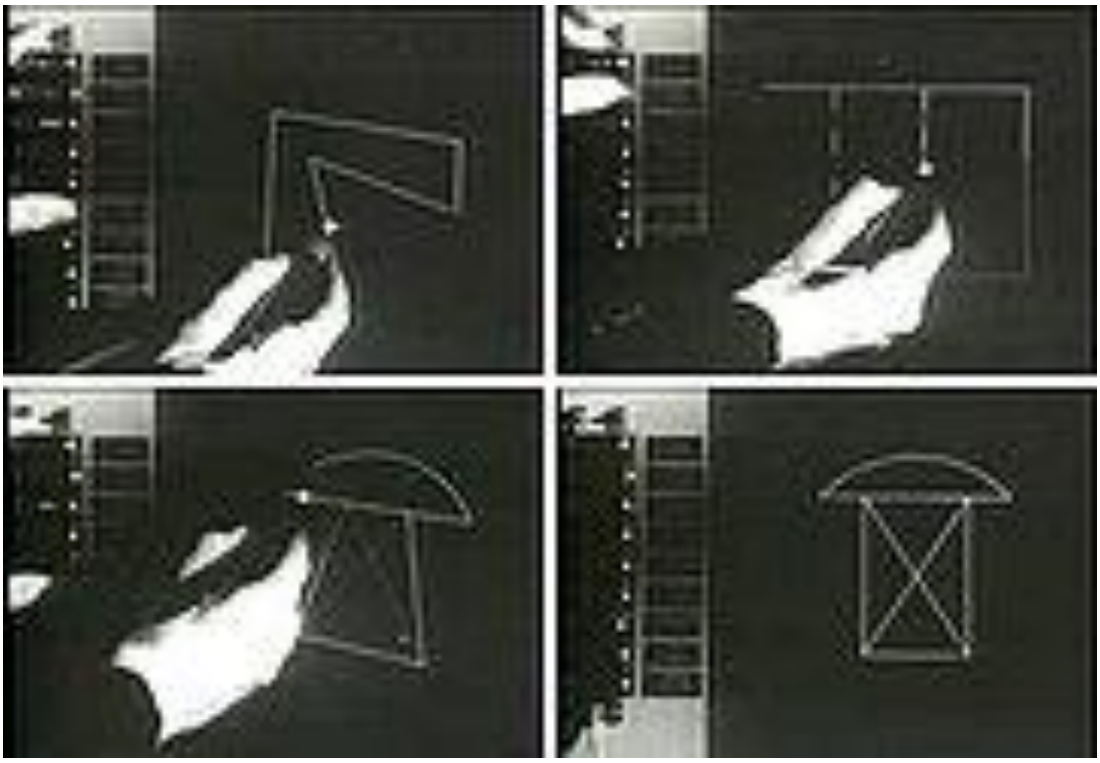

Figure 3: Sketchpad

Initial origins of multi-user online simulated worlds stemmed from Multi User Dungeons (MUDs). The first of these, MUD1, was released in 1978. MUDs allowed several people to participate in the same online world via text explanations of locations and events. The first of the 3D multi-user-across-network virtual worlds came in the form of *Maze War* [2]. It was written in the early 1970s to enable NASA Ames scientists to test a collection of machines with then-impressive 16 bits and vector graphics co-processors. Howard Palmer, creator of *Maze*

War, explained:

> "We had written a lot of useful software for the Imlac, but never any games, and we were feeling like it was time that we proved that we, too, could write fun code. As we were sitting around brainstorming one day, we came up with idea of a networked, multiplayer maze game. I suggested to Steve that it would be really cool if it could be 3D, but Steve didn't think that the Imlac had the necessary processing power. But then I pointed out that a maze with all 90 degree angles might enable a simpler 3D rendering than the general case. Steve got excited about this, and as he often did, came back the next day with an implementation of the graphics for the maze navigation. Within a short time, networking code was added to get the multiplayer version going (we had been experimenting with ad hoc LANS for the Imlacs as part of our more serious work)."

Later, more developed, virtual world prototypes continued to appear as the technology to support them rapidly developed. These included *WorldsAway* [3], CompuServe's *Dreamscape* [4], and *The Palace* [5], which later became an early exploration into natural language processing.

At this point, virtual world technology began spreading into more diverse areas than the video-game-based offerings which had come before. An early example was created in 1996 in Helsinki, Finland by a telephone company and was built from the ground up to be a virtual 3D version of the city [6]. Although the entire city was never finished, many subareas were completed in both modern and historical settings.

Another precursor to social aspects of virtual worlds was the development and widespread deployment of Bulletin Board Systems (BBSs), which enabled non-technical users to communicate instantly with each other around the world.

Modern 3D graphical Virtual Worlds we see today combine features of earlier BBSs and MUDs. Contemporary virtual worlds are a conglomeration of these early efforts at virtual reality and newer, computer-based gaming industry technology as well as social networking software. The difference is that rather than attempting to emulate reality to a perception-level of

degree, modern offerings scale back and attempt to engage users on an emotional level in an attempt to simulate the same immersion.

These modern simulations can be logically divided into two groups which sometimes overlap. The first group can be characterized as Massively Multiplayer Online Role-Playing Games (MMORPGs). These feature little to no real-life simulation, instead providing a virtual space where users can interactively play a game of some sort. Examples include Everquest and World of Warcraft. The second group is termed Massively Multiplayer Online Real-Life Games (MMORLG). These allow users to create and edit their virtual representation (avatar) at will. Rather than participating in a game or competition, these are used mostly as a social networking outlet. It is one of these MMORLGs which our project utilizes for the base framework.

2.1.2 Second Life

Second Life (SL), released in 2003 by Linden Labs Corp., has become popular with over 13 million players world-wide. It is a social networking tool that is an economic experiment as well in that it is the first world that allows users to own virtual land and build and trade interactive virtual goods within the virtual world. SL is divided into a large number of "regions," or subplots of land. Each region is 256 x 256 in-game meters. Regions can be stand-along islands (like *University of Arkansas* island) or they can be seamlessly connected to other regions. One or a small number of regions are hosted on one standalone server, which clients (avatars - see below) may visit. This architecture is similar to the architecture of the World Wide Web, which has servers at Sears, e-Bay and so on that many end-user clients can visit. But the virtual world architecture requires many additional services including services for avatars, inventories, messaging, and inter-grid communications.

Another difference is that SL is proprietary in that the SL developer Linden Labs physically houses and rents its grid of servers to region administrators, whereas in the World Wide Web, no one entity owns all servers. In SL, regions are entirely self-contained and administered by an "owner" who can grant permissions to others to own parcels (portions of a region) or build or script in designated areas. Any user can freely register and explore Second Life, but if a user wants to claim a parcel of property or an entire region, Linden Labs requires purchase. Thus, regions are "rented" from Linden Labs, giving the renter the ability to control aspects of his or her region. These controllable aspects include everything from allowing and disallowing visitors to granting various interaction permissions to one or more additional avatars. Thus, by careful use of permissions, owners can protect their region from malicious tampering while still allowing anonymous visitors to explore.

Users, also known in SL as "residents" according to Linden Labs, interact with one another and the world via virtual representations called "avatars." Residents can customize the appearance and clothing of their Avatar, creating a realistic world full of unique "people". Through her avatar, a user can take a variety of actions, for instance, walk, fly, interact with objects, speak, hear, teleport to another place, build, and script.

While Second Life itself is free for registration and basic interaction, players quickly find that many aspects within the virtual world is driven by a real-world economy. Users typically purchase in-world currency known as "Linden Dollars" via Linden Labs' website at an approximate rate of 250 LD per $1 US. Users then can charge one another in LD currency for services or items within the world. It is also possible to convert earnings back into real world currency via Linden Dollar exchanges. As of 2009, Second Life's total economy size was over

$550 million US Dollars and residents earned $55 million [7]. The SL economy has enabled a number of people to make a living entirely through Second Life related ventures.

Once inside SL, the simplest of actions is basic exploration. User-controlled avatars are able to travel from region to region instantaneously (via teleporting), downloading localized geographic content in real time as they move through the world. Each region contains an extensive collection of permissions and user rights which regulate users' ability to visit, interact with, and change that particular region's contents. As a result, as any given avatar moves from region to region, the collection of feasible actions changes. Some regions may block users from visiting entirely whereas others are wide open "sandbox" regions where all users are free to interact with full permissions.

Interactive actions available to avatars are another defining feature of SL. Whereas MMORPGs offer a static, unchanging world which users have no control over, SL allows users to build, script, and manage complex objects. This means SL's virtual landscape is in a constant state of flux as users create and destroy objects (which requires real-time updates to all avatars nearby who also see these changes as they are taking place).

Building primitive and complex objects in SL is accomplished via a proprietary, in-world modeling engine. This engine is rather simplistic compared to modern modeling packages such as SoftImage or Maya, but still allows a relatively high level of control. Rather than mesh-based modeling, Second Life's engine is based around "prims" or primitive objects. Complicated objects are then constructed by combining these prims in a very similar manner to building with Lego blocks. Prims can be "linked" together to construct more conceptually complicated objects, though Second Life has a variety of restrictions upon size and total number of linked prims. These restrictions seem draconian at first, but sharing dynamically shifting meshes

between 50 avatars requires a large amount of computing power. The nature of Linden Labs' business lends itself to a bigger and bigger audience, thus striving to keep each end-user's computer requirements low.

Once an object has been created, the user can then define specific behaviors for the object to follow. These behaviors are programmed using a proprietary scripting language called Linden Scripting Language, or LSL. LSL is syntactically similar to C/C++. LSL's programming model is a finite state machine that uses an event-driven architecture. An LSL script contains definitions of a variety of states. Within each state definition is a description of the script's necessary actions upon entering that state. Scripts can use the usual tools found in most programming languages, such as variables, functions, and operators, to express each state's behavior. The LSL script is capable of triggering state change events itself as well as receiving a number of other event triggers from the outside environment. Some of the most useful of these include messages from other scripts/agents, email, HTTP or XML-RPC messaging, and collisions. It is important to note that the concept of an LSL script is closely coupled with in-world objects. All scripts exist within and control only a specific object. Within that object, though, a script has nearly limitless power to modify object parameters (e.g., shape, size, texture, rotation) as well as object behaviors (e.g., movement, collision, message passing, chat).

Although LSL is adequate for scripting, it has several restrictions and shortcomings that become large obstacles to overcome in developing virtual world applications. First, there is no inherent method of data storage. While a script is capable of calculating values, there is no method within SL to store the results persistently. SL stores only the currently executing state of the object, all other data is lost when an object is removed from the world If this object is then "rezzed" (reinstantiated), it will resume from the state that it was previously operating in, only

without stored data. Objects are uniquely identified through Universally Unique Identifiers (UUIDs) which operate using a distributed hash method. UUIDs are used in a multitude of computing applications, from Unix file systems to encrypted partitions. A generated UUID can be formed from any system and used with relative assurance that the UUID will never be duplicated on any other system. While this is the method for managing Second Life's multitudes of objects, SL UUIDs are also not persistent across sessions. Every time an object is placed back into the world ("rezzed"), a new UUID is generated for it. This causes problems with tracking objects via a third party database, as the "same" object appears uniquely different every time an avatar picks it up and places it back down. Other restrictions include the lack of any array data types and a maximum of 64KB of memory available to any given script.

SL's LSL scripting language is also constructed with a variety of built-in, mandatory function delays. These delays are implemented with the intention of preventing resource-overload, but serve also as hard limits as to what is achievable through LSL alone. These delays impact a variety of highly important functions such as sending emails (20 second delay), HTTP communications (3 seconds), and moving objects (0.2 second delay). These delays cause many headaches for applications built on top of SL and will be discussed later.

One last limitation of SL is worth noting. Because it has an internal economy where objects can be sold to others, it is careful not to support arbitrary object copy operations. That is, export of objects out of SL or import into SL is difficult. This means that sharing objects across virtual world boundaries is problematic, a point we will describe in more detail in Chapter 3.

2.1.3 Open Simulator / Open Metaverse

Due in part to the above problems, the open source community interested in virtual worlds has begun establishing free, improved worlds rather than working within the Linden

Labs-imposed restrictions upon Second Life. The most important attempt to build an open source virtual world is OpenSimulator, a reverse engineering of the SL server platform which aims to provide a highly similar yet enhanced version of Linden Labs' offering. OpenSimulator succeeds in this well enough that users have the option of utilizing Linden Labs' Official Second Life Viewer as a client for OpenSimulator-based grids along with Second Life itself. Our project considered and adopted Open Simulator for reasons to be described in Chapter 3. .

2.2 Related Work

As described in Chapter 1, this thesis describes progress in removing roadblocks to make available a safe 3D virtual world to high school students. Others are pursuing a similar goal. Related work falls into two categories. Several industry leaders and educators are investigating the use of Second Life and OpenSimulator as ways of reducing cost, improving communication, establishing in-world commercial retail outlets, advertising, and holding virtual meetings and conferences to reduce travel. Other groups have decided to build their own improved virtual worlds in an attempt to circumvent problems inherent in the Second Life model.

2.2.1 Industry Ventures

Several commercial ventures have turned their attention to SL in the past few years to explore marketing possibilities. In the retail sector, Georgio Armani maintains a virtual store where avatars can make purchases. Diverse organizations such as Time-Warner and Adidas are jumping on board. [8]

Technology giants Microsoft [9], IBM [10], and Intel [11] use SL for research rather than immediate profit.

Microsoft owns and operates an island within Linden Labs' official grid. Recently, it began hosting all of its "Imagine Cup" Student Development Competition events virtually on the island [12]. This allows students from 24 countries to communicate and share their projects via real-time voice chat and video streaming as well as participate in Microsoft-sponsored events.

Intel appears to have a more diverse agenda of interests regarding Second Life. The main Intel island has its fair share of marketing in the form of a massive Core 2 Duo processor and free avatar accessories with corporate logos. However, Intel also maintains a Software College region with lessons on a variety of related topics [13] as well as a more outlandish partnership island with Orange County Choppers [14].

IBM is the major company with the largest virtual world presence via its Innovation Center. Its business ventures are mostly concentrated into a collection of regions known as the IBM Business Center. Visitors to the region are able to speak with an IBM representative present 24 hours a day and can explore virtual libraries and examples of Virtual World innovation [15]. The Business Center is also the location where IBM hosts speakers and conferences. IBM also utilizes this method for giving virtual tours of security-sensitive new technologies such as its Green Data Center Solutions [16] and IBM Healthcare Island.

On the academic side, IBM was the first to transport an entire company-internal conference into Second Life. The reported return on investment was around $320,000 with the total cost less than one-fifth the comparative cost for a real-world conference [17]. This initial event paved the way for IBM's Neil Katz to convince IBM to dedicate an entire team from the IBM Academy of Technology to the focus of virtual worlds. Later in 2008, the IBM Academy of Technology's President, Joanne Martin, was so impressed with results that she

rescheduled the Academy's Annual General Meeting to be held within Second Life as well. Martin went on to state:

> "Second Life provided an opportunity for us to have a positive social and technical exchange, addressing most of our collaboration objectives. And, we delivered the experience at about one fifth the cost and without a single case of jet lag."

Most recently, Katz and the IBM team collaborated with our University of Arkansas research team to host the X10 Workshop on Extensible Virtual Worlds in Second Life at IBM Business Center. The conference was well attended by a variety of academic, corporate, and open-source developers [18].

Second Life has been leveraged for many other educational uses already. Many universities maintain virtual land (islands) in Second Life for education, research, or other academic pursuits. Many schools, including the University of Arkansas, have explored or implemented programs holding classes or entire courses within Second Life. Others are building art museums, enacting plays, and hosting mock business competitions. Other research projects are exploring particular application domains such as healthcare, retail, disaster relief, special education, and library science.

As mentioned in Chapter 1, our own Everything is Alive (EiA) project at University of Arkansas is exploring how to use 3D virtual worlds to model pervasive computing to model the real world. We have explored

- Protocols and standards for what makes an object a smart object
- How to reflectively import an object's API into a universal remote control to make it possible for humans to control any number of newly-encountered objects via one device.
- Workflow and logistics as well as inventory management systems

- Virtual World Search Engines – a much harder task when one searches a 3D world rather than an Internet full of text

2.2.2 Alternate Virtual Worlds In Development

Alternatives to Second Life and Open Simulator are beginning to appear as interest in virtual world technology grows. Some, such as Google's Lively, were simply explorations into the popularity of new offerings. Others, such as Open Metaverse, Unity3D, Open Wonderland, and Open Cobalt, are full-fledged commercial or academic endeavors with well-defined goals to rebuild virtual worlds from the ground up.

2.2.2.1 Google Lively

In 2008, Google introduced a beta version of Lively, a new virtual world integrated with the World Wide Web [19]. Lively used Adobe Flash and aimed to convey a new method of information access. The goal was to provide a method to embed Lively "rooms" into normal HTML web pages, allowing the webpage to continue to provide a traditional 2D content experience while users participated in a social networking type of communication in real time 3D rooms built around the page. Lively arose as part of Google's "20% Rule" that allowed developers one day per week to work on pet projects rather than job-related material. On debut, it featured the ability for up to 20 users to congregate in a single room, notably short of Second Life's maximum occupancy of 50 avatars per region. Even more restrictively, users were unable to generate their own content, though they could decorate virtual environments by hanging YouTube videos and Picasa pictures on walls. Lively was discontinued after six months, apparently due to low interest. Nonetheless, it was an interesting experiment in extending virtual

worlds to extend the web from its document centric paradigm to also accommodate a 3D space paradigm - a vision which many in the 3D virtual worlds community believe is coming.

2.2.2.2 Open Wonderland

Until recently, Open Wonderland was Sun Microsystems "Project Wonderland," a Java-based open source virtual world. Upon Oracle's acquisition of Sun, project development was terminated and the development team left Sun to continue development of Open Wonderland. Open Wonderland is best explained by its Executive Director, Nicole Yankelovich, who stated:

> "Open Wonderland is an open source toolkit for creating 3D virtual worlds. I emphasize toolkit because unlike Second Life or World of Warcraft, Wonderland is not a destination in and of itself. It's a set of tools that people can use to create virtual world destinations. The toolkit is quite rich, so out of the box, you can build some pretty nice worlds. And with a bit of software development effort, you can create highly customized, special-purpose virtual worlds.
>
> Think in terms of the World Wide Web. There's no single, giant document collection. On the web, millions of individual organizations own and operate highly specialized web sites both inside and outside their firewalls – travel sites, video sharing sites, endangered turtle tracking sites, charitable giving sites, financial planning sites, and so forth. To create these web sites, developers use tools like CSS, PHP, content management systems, Flash, blogging software, and many others which you can collectively think of as the web toolkit.
>
> The Wonderland toolkit is analogous to the web model. Individual organizations can use the Wonderland tools to create their own specialized virtual worlds" [20].

In light of this, Open Wonderland attempts to further Google's integrative model with a considerable twist. Rather than forbidding user-generated content, the entire toolkit is inherently extensible. The platform is entirely written in Java with the intent of creating a flexible environment developers could create not only objects for, but entire framework plugins. Open Wonderland also supports drag and drop functionality with many file types, from images to PDFs and PowerPoint slideshows. Another interesting import feature is the ability to create

animations within Alice, an introductory educational programming environment I also use to teach basic programming fundamentals. Finally, Open Wonderland exposes the ability to run arbitrary Java and Linux applications from within the virtual world. Thus far, applications have mostly targeted multi-user approaches such as shared whiteboards and sticky notes. Open Wonderland is still in rapid development, but already boasts a large list of education-related interests [21].

2.2.2.3 Others: Open Cobalt and Unity 3D

Several other virtual world platforms have become popular recently as well. Of these, Open Cobalt boasts the most promise as an education and research oriented application. Open Cobalt is being developed as open-source by Duke University and is written in Squeak, an offshoot of Smalltalk. Open Cobalt is notable for several reasons. It is fully open source and free to use, modify, and distribute. Open Cobalt runs on Unix and OSX, which is friendly to a variety of lab hardware, as we discovered in the Greenland EAST Lab, proves to be invaluable when dealing with such a variety of lab hardware. The platform was built around privacy, allowing more extensive control of regions than Second Life offers. It also grounded in de facto standards, supporting LDAP, Jabber, voice chat, collaborative document editing, in-world VNC, native importing of meshes/textures (both KMZ and OBJ) and media (both MPG and AVI), custom avatar animations using Collada and Ogre3D, and both building and scripting support. Interestingly, Open Cobalt's scripting support is for Smalltalk/Squeak, which shows Open Cobalt's first weakness, namely, users have to learn Smalltalk. Smalltalk was largely overshadowed by Java and is only beginning to regain popularity, so at this time it is not a likely language known by most developers like C and Java are.

Another emerging virtual world platform is Unity3D. Unity3D is an integrated development environment and API for creating interactive worlds of all sorts. While not offering an already-established virtual world specific platform as do the other technologies discussed, Unity3D has several unique advantages. Most importantly, Unity's level of technological development is unsurpassed, both in graphics and ease of use. While any academic research into using Unity3D as a basis for virtual world based education would first have to create the platform, the package makes this a manageable task. It is also natively deployable not only on Windows and OSX, but on a variety of handheld operating systems, iPhones, Nintendo Wiis, Google Android OS phones, Microsoft Xbox 360s, and Sony PlayStation 3s. The ability to deploy on mobile devices is particularly attractive. Similar to Open Cobalt, Unity supports mesh/texture importing as well as nearly all modern rendering techniques (including Parallax Mapping and Ambient Occlusion). In-game scripting is handled via Mono, so Unity offers the same scripting options as Open Simulator (JavaScript, C#, and Python). Unity3D has two major downsides: its implementation is proprietary and it is mainly used for video game development, so the bias is toward video games rather than virtual world development. A virtual world venture using Unity3D would be beginning basically from scratch. This bias also means Unity3D requires the use of high end computing hardware. The majority of students do not have access to such demanding computer equipment.

2.2.3 Virtual Worlds for Children

Due to the requirement to preserve children's privacy online and insure their safety, virtual worlds for sub-18 members have more restrictions than virtual worlds for adults. Many virtual worlds simply restrict anyone below 18 from registering, as Second Life does. However, a few offer places for children to congregate online. The largest of these is Linden Labs'

duplicate grid, dubbed the "Teen Grid". This grid exists entirely separately from the standard Second Life grid and caters only to children aged 13-17. Access is technically free, but each student must verify his or her identity via an SMS message or PayPal account. Adults may visit the island only after participating in a criminal background check. This maintains child safety and keeps the underage population partitioned off from the mostly uncensored adult grid. The teen grid is considerably smaller than the adult grid in both size and population, comprising only 93 mainland regions, 7 private regions, and 97 education/project based regions. Of the education oriented regions, many are like ours and inaccessible to most other residents. Several, however, are public projects, such as Global Kids island, UNICEF island, the Virtual Video Project, and Playing 4 Keeps. All of these projects are run by non-profit organizations with the intent of helping educate children.

3. ROADBLOCKS AND SOLUTIONS

As mentioned, our primary goal was to include high school students in our research that used virtual worlds to understand pervasive computing in the real world. This chapter documents a series of roadblocks we identified and overcame in realizing our objective.

Software engineering textbooks often give the impression that all requirements can be foreseen at the beginning of a project. But solving complex problems can involve partial solutions, roadblocks, and overcoming unforeseen problems, as described in this section.

3.1 Project Genesis

In Fall 2007, Dr. Craig Thompson and other faculty at the University of Arkansas wrote a proposal to Chancellor John White to fund an initial Second Life island. Linden Labs charges around $3,000 per year for one region -- we named ours *University of Arkansas* island.

In Spring 2008, Dr. Thompson incorporated Second Life as a pedagogical aid into his Artificial Intelligence course since it made it easier to teach several concepts including agents, path planning, and how to make objects more intelligent (e.g., chat bots, virtual world games) using a driving application of a healthcare facility. The choice of domain was, in part, driven by our associations with the university's RFID Research Center and also the then new Center for Innovation in Healthcare Logistics, which were both interested in modeling health care supply chains.

3.2 EAST Island on the Teen Grid

That summer, Dr. Thompson proposed to Arkansas Science and Technology Authority (ASTA) a project to involve EAST high school students in our growing virtual world research.

ASTA funded the project for around $29,000. Our interest in including high school students sprang partly from our hypothesis that they could learn how to build in SL (and we needed to flesh out our hospital with wheelchairs, beds, cabinets, equipment, medications, and patients) and perhaps could learn to script as well. If this experiment succeeded, we could perhaps fan out SL more directly into high schools as an outreach activity that helped us while giving the EAST students an enriched computational background.

Dr. Thompson's regular class of graduates and undergraduates met all summer to flesh out a virtual hospital (also known as a *hogspital*) on University of Arkansas island in the Second Life grid. As a fairly experienced Second Lifer, I acted as a de facto TA/RA for that class. In addition, as an EAST alumni in need of a Masters' topic, I became the link between the EAST involvement and the project at-large as well as the six EAST high school students who met together for the final six weeks of the summer for 20 hours a week on campus in the EAST Lab on the second floor of the JB Hunt Building.

While preparing for the involvement of the EAST students in our summer project, I discovered that high school students are not allowed on the main Second Life grid but are required to use a separate teen grid. This necessitated purchasing a teen grid island, which cost an additional $3,000.

I soon discovered that the price tag on our teen island did not guarantee much customer service. It took me approximately six weeks of interactions with Linden Labs customer support from payment date until we were able to access our teen region for the first time. Once the high school students arrived, it took around two of our six weeks to manually register them and lock them down to our island. We also had to designate adults to oversee their progress and this

required security background checks for myself and Adam Barnes, research staffer in the Center for Advanced Spatial Technology (CAST), who manages CAST's EAST Lab.

Once on the island, the EAST students rapidly picked up necessary skills and almost overnight turned into very productive developers. For the next four weeks, the biggest problem was formulating enough new ideas to keep them busy. This was much more time consuming than we imagined - it involved developing long lists of medical equipment, furnishings, and also a field trip to Washington Regional Hospital's Catheter Lab where we took pictures of catheter operation equipment and heard descriptions of associated procedures.

By the end of the six weeks, the small group of high school students had constructed a reasonably realistic second virtual hospital on the teen grid, complementing and in fact more extensive than the "hogspital" on *University of Arkansas* island. We captured their summer work in two YouTube videos available in the Demo tab of our project website:

- http://www.youtube.com/watch?v=M2TsVfatPZ8
- http://www.youtube.com/watch?v=9nqt6CENLFY

It was here that we encountered the next hurdle. As the teen grid and adult grids are separate, there was no simple way to transfer the students' work to our main project site. Upon consulting Linden Labs, we were left with no option except to wait for one of the students to turn 18 and thereby qualify to move to the adult grid. It took much of the next year to wait for a student to turn 18, to unprotect and transfer all teen hospital assets to the "mule" student, and then to follow Linden Labs' procedure for transferring an avatar from the teen grid to the adult grid (which strangely involved in part the student providing Linden Labs with a copy of his utility bill to prove he was now 18!).

Although we eventually made the transfer, it was clear that this method was in no way sustainable. Thus, at the end of summer, we had learned two very important things: the EAST integration was a great success in that high school students do have the ability to contribute to a 3D virtual world research project. But also that Linden Labs' offering was unsuitable as a platform because of their policies involving teens and transfer of assets across grid boundaries. We also learned a few other important lessons:

- six students had filled their island to capacity (around 15,000 primitive objects) in one month of half time work. So for EAST to provide a 3D virtual world to high schools, it would need at least one region and perhaps more per high school.
- the cost per region ($3,000 per year) for many regions is prohibitive and unsustainable for most or all public high schools

Concurrently with the above, during downtimes with the EAST students, I spec-ed, ordered, and provisioned a project information server to manage our project's growing knowledgebase and website-related tasks. The server acted as host for my newly created project website (http://vw.ddns.uark.edu) as well as offering Subversion repositories, development/deployment automation, wiki integration, backup, bug tracking, MYSQL database and related administration tools.

Most of these are typical project necessities, but the Wiki and registration API have particular significance. As much of our workforce consisted of students doing class projects, our developer pool suffered a massive turnover rate (with students coming and going every semester). At first, this was ok, but as the project grew, we were spending more and more time getting new developers up to speed. At the same time, leaving developers often took much of their new knowledge with them, leaving us mostly clueless as to how portions of progress were

actually accomplished. In addition, we could not track which avatar corresponded to which student. Finally, we began receiving requests from across campus to use our island and needed a quick way to grant permissions for building and scripting. To solve the problem of documenting student work, I configured a basic Wiki and we enforced a documentation-goes-on-Wiki rule. The Wiki quickly grew and expanded, later becoming a project of its own by another student.

Regarding the registration API, because avatars are mostly anonymous, we had no way to track individual developer actions or contributions. After talking with Linden Labs and acquiring special permissions, I was able to obtain a barebones API specification for interfacing with the Second Life Avatar Registration system. I learned Ruby and then created a web-facing proprietary registration system (See Appendix). By routing all incoming project developers through the new API rather than Linden Labs' version, we were able to compile a database table relating student ID to avatar ID as well as keep track of a variety of developer-specific information. The process is transparent to new developers and offers the same feature set as the official registration but with the addition that it is now very easy for us to grant a new student rights to build and script on our island and to keep track of what their avatar has developed.

Another related hurdle was that Second Life contains very detailed access control so that every object and every script it by default owned by its creator and the creator can provide specific rights to others. When one avatar creates an object on the island and does not remove permissions, only an administrator can remove it and no one else can further modify it. Our few administrators had to become human garbage collectors but worse, sometimes very useful objects had to be removed because we no longer owned permissions to move, operate, or improve them. Recognizing this problem early on, working with a few other project members, I

developed a procedure for preserving our assets. The procedure involved an Project Inventory Avatar as the keeper of past student projects. Before semester's end, a student would remove their permissions (which we manually double checked) and then check their project contributions into the Project Inventory Avatar.

The new and improved Second Life-based project backbone helped automate our larger virtual world project, but by the end of summer, there was no question that an alternative framework would have to be found in order to continue working with the EAST program and with high school students. The Linden Labs Teen Grid simply lacked key specifics we required. Inability to share and export work, high cost of island rental, low maximum object and processing limits, and languid customer service all contributed to the final decision to migrate to an open source platform.

3.3 Migration to OpenSimulator

I began exploring other avenues and found that at the time (Fall 2008), there were only a few options. Of them, only one offered a real framework rather than pre-alpha code chunks. The open source OpenSimulator (which aims to mirror Second Life) was considered an "alpha" release, but appeared to offer a mostly-stable platform. A significant benefit to our project was the ability to deploy a "grid" of OpenSimulator regions on a local area network, allowing free creation of regions limited only by hardware. Thus, for every region hosted under our local network, we saved $3,000 US by not having to purchase from Linden Labs. It is also possible to remotely attach a region to a separate grid, making it possible to establish a widespread academic virtual network which educational entities can connect to and disconnect from at any time. In addition, any number of users may register avatars and participate in the virtual network regardless of whether they attach a server of their own. After a quick trial run,

we decided to go ahead and select that platform for an experiment of trying to provide a virtual world platform for high school students.

The first step was purchasing a small collection of machines to serve as our backbone. In total, our "grid" ended up being three low end commodity PCs loaded with RAM located behind a gigabit router. Thus began a long period of establishing the OpenSimulator framework on the grid. OpenSimulator is written entirely in C#, yet theoretically supported Unix deployment via Mono, an open source .NET compiler. All of our existing project structure was Unix based, so we decided to deploy OpenSimulator in the same environment.

Early testing went well - we deployed the alpha release of OpenSimulator and did some scalability testing. We found we could load around four regions per server and up to 60,000 prims per region. We also knew we had control of import and export with this platform making it much easier to preserve work from project contributors.

But we rapidly ran into our next roadblock -- platform stability: OpenSimulator, like many other open source projects, suffered from rapid undocumented changes and woefully out of date documentation. This meant the platform could become unstable if we continuously updated to keep pace with OpenSimulator development. In addition, a distributed grid framework such as the one we were attempting was experimental only, leading to further complications. Only one other distributed grid based on OpenSimulator existed at the time, OSGrid. OSGrid was closely tied with the OpenSimulator project and was supported by several developers. Our startup project, on the other hand, was not, and thus a constant stream of errors had to be debugged manually.

Instead, we chose to freeze the version of OpenSimulator and stabilize that platform. I fixed the last of the compilation/configuration problems a couple months later and managed to

successfully deploy the first iteration of what came to be our final architecture design. The architecture consisted of a number of modular process daemons which interacted across the local network to create a single overall OpenSimulator Grid. In total, there were five independent daemons: User, Grid, Asset, Inventory, and Messaging. We could also create a hardware-limited number of independent Region servers which served each in-world region. The first deployment featured four Region servers on each of two $500 commodity servers, giving us a total of 800% more virtual land than what we currently pay Linden Labs $3,000 per year to rent for our main *University of Arkansas* island.

The daemons are mostly standard. The User daemon handles all avatar registration, authentication, and log on/log off procedures. It is the gateway through which clients interact with the world. The Asset server manages all information relating to in-world assets, which include such things as clothing, textures, and streaming media objects. It interfaces directly with MySQL through a server plugin configured to operate with our previously-established project framework databases. The inventory server acts much the same, only handling avatar inventories rather than world objects. It also interfaces directly to the MySQL backbone. The Messaging server not only controls all in-world chat but offers the ability to connect via the IRC chat protocol. Finally, the Grid daemon acts as mother hen, overseeing the interactions of the other servers and maintaining consistent world state. It has no direct database or user interaction, but required the most configuration.

The successful deployment was great news, but bugs due to the immaturity of the code soon surfaced. In our more comprehensive testing, around 40% of LSL functionality had bugs or was not yet implemented. Worse, we began running into a crippling yet inconsistently reproducible bug. The entirety of the MySQL database would unpredictably wipe itself out,

resulting in a loss of all in-world work. Increasing the frequency of database backup served as a temporary band aid solution as I attempted to track the source of the problem. Eventually, I located an obscure pair of bugs in the MySQL adapter/region handling code which worked together to allow some regions' database connections to time out. The time out was flagged as a critical crash, driving the MySQL adapter to clear the database in an attempt to recover. The problems were easily fixed once found and I submitted a patch to OpenSimulator's codebase (See Appendix B).

Having established a stable deployment, our attention turned to recruiting a high school EAST lab to participate.[1]

3.4 Deploying OpenSimulator Virtual World in a High School

As Dr. Thompson and I considered how to deploy OpenSimulator in high schools, we discussed two approaches. We could either deploy the system internally to a high school which meant that it would be administered locally -- but this seemed a non-starter since there were generally no local administrators competent to take on this task and it would be difficult to share our work with the high schools without firewall access. The second alternative was to host OpenSimulator on University of Arkansas campus so that the region servers were administered by our project and then to open required ports so students could remotely access our virtual world servers. We chose this latter approach, knowing we could at least temporarily afford to

[1] Around this time (Spring 2009), our EiA project became closely involved with the IBM Academy of Technology and together we formed plans to host our own virtual worlds conference in-world -- the X10 Workshop on Extensible Virtual Worlds. To support this conference, I deployed a conference-specific website (http://vw.ddns.uark.edu/X10) to manage information, registration, and paper submissions.

administer the machines and hoping to transfer this responsibility to EAST's central offices eventually.

During Fall 2009, we approached the EAST program at Fayetteville High School as a possible venue to try our experiment of deploying virtual world technology in a high school. We encountered our first social roadblock when their computing administrators refused to open firewalls. This decision was made for several apparent reasons: it would involve taking a risk with a new technology; it would involve reconfiguring firewalls required some learning; it would involve an unknown stress on the network if many students accessed a virtual world remotely; and finally it might involve student safety. We learned that, by default, many sites on the web (and in some schools the entire web) are blocked to insure students cannot reach sites like YouTube, Facebook, World of Warcraft, and many other popular sites.

Due to national child privacy protection laws, namely the Children's Online Privacy Protection Act (COPPA) [22], it was necessary to convince school administrators that their students' safety would be protected. This required that only students from a given school could inhabit a given region in the virtual world - since visiting avatars from another school or adults (including university students) could not mix with a given schools students (similar to insuring student safety on a high school campus by not allowing uninvited guests on campus).

The inherent authentication and administration functionality we have built into our OpenSimulator guaranteed this from our end, as only manually approved students were allowed access. However, due to administrative roadblocks, opening the necessary ports turned out to be too big a hurdle for Fayetteville High at that time. We ceased pursing this avenue at that time because we found another school to work with.

During Spring 2009, we talked to John Diesel, the EAST Facilitator at Greenland High School (just south of Fayetteville) and he was extremely receptive to the idea of Greenland being the vanguard school for trying the virtual world experiment providing I would manage the deployment.

The OpenSimulator deployment requires several special ports available for platform communication. Our local project was correctly configured, but it was necessary to establish the same openings at Greenland High School's network. Our first step was to task the Greenland facilitator with convincing the school's systems administrator that the project was both safe and legitimate, which I successfully accomplished.

However, once I was in communication with the administrator, I hit another roadblock. While perhaps not an issue for larger, better funded schools, Greenland's budget does not support an experienced systems administration team. As such, I found that the systems administrator did not know how to manually configure the school's various firewalls and web-filtering software. After network testing and research, I managed to locate the necessary information for opening port communications and preventing web filtering and communicated that to the administrator, who made the changes. This process occurred over many iterations and a few months.

At this point, the lines of communication were open and all that was left was to deploy the project on Greenland's EAST computers. In Fall 2009, Dr. Thompson and I visited Greenland to install Second Life clients on their EAST Lab computers and to talk to them about virtual worlds. Again owing to budget issues, it turned out that Greenland's EAST Lab contains several older generation machines and only a few graphics capable machines (they are currently upgrading to all modern hardware). Second Life is not a demanding application client-side, but

it requires graphics cards that can handle a fully rendered 3D world. I found that several of the Greenland EAST lab's machines were incapable of the performance required to participate in the project.

3.5 Mission Accomplished

After deploying the client-side software on the few capable machines at the Greenland East Lab, first contact was made with the grid server! Our initial mission was successful.

3.6 Final Architecture

The final architecture is less of a planned design than a solution which evolved over the course of the project. The current iteration layout is shown below in Figure .

The largest portion of the architecture is the server grid located in the Everything is Alive Lab on the fourth floor of JB Hunt on University of Arkansas campus. The grid consists of a variety of servers, each with specific functionalities, which communicates through a single access point router. This allows the ability to connect to the Internet without sacrificing security. The Internet connection ability allows both clients and remote region servers to connect and interact with the grid in predetermined ways.

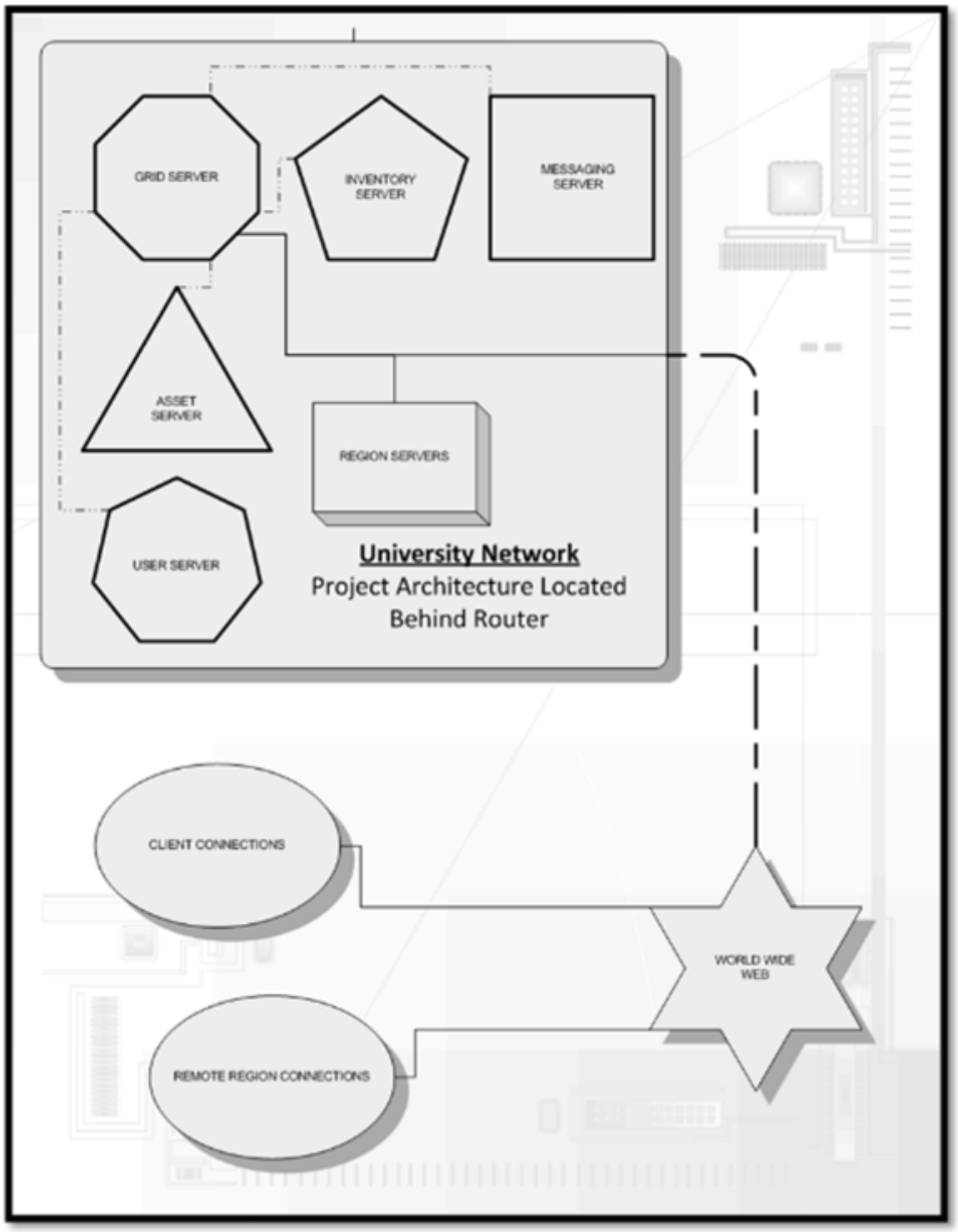

Figure 4: High Level Architecture of OpenSimulator as Deployed

4. TEST METHODOLOGY, RESULTS, AND ANALYSIS

4.1 Methodology

The methodology for administering the EiA virtual worlds project and deploying it to high schools was iterative and incremental and followed the natural progression of the project itself. As the project evolved over time, we were able to identify new requirements, devise new solutions and test them as they were added. Doing this sort of testing at the end never would make sense as we could not foresee the problems until we came to them.

During the academic year Fall 2009-Spring 2010, I taught the CSCE Department's introductory course for non-majors CSCE 1013: Explorations In Computing. Dr. Thompson assigned a senior Capstone student Kevin Calderon to work week-to-week with the Greenland EAST students to build and script a smart school using the OpenSimulator platform.

By utilizing the bug tracker as well as email interactions, I was able to quickly adapt and solve all problems the EAST students encountered. Aside from a few problems, the virtual world platform has been very stable. One problem surfaced when the students, in the first few days of experimenting, tried to terraform their region with gigantic mountains that surfaced an internal bug in OpenSimulator. A similar OpenSimulator-related bug surfaced when students flew object-laden aircraft into the ground at high speed. A humorous early configuration problem surfaced when the Greenland students asked us why all the avatars they created only appeared as women. Once the system stabilized, only intermittent support was needed during the year. A few times when the server went down due to power failures in the JB Hunt building, Greenland students asked us to reboot. One sign of success was when the Greenland students ran past the prim limit and requested additional regions.

4.2 Results

Our current implementation contains a full Greenland High School integration. We now have four regions dedicated to Greenland High students. The below table (Fig. 5) tracks platform user accounts as well as total numbers of scripts and objects created since initial deployment.

Total Registered Users	36
Persistent World Objects	1647
Inventory Items	1104
Assets (textures, media, etc.)	2541
LSL Scripts	829
Total Prims	28,773 (17.47 prims/object average)

Fig. 5: Compiled Statistics

4.3 Analysis

Since the first establishment of the client-server functionality, Greenland students have rapidly become engrossed in the project. We recently quadrupled the region allocation to support continued work and the students show no signs of slowing down. As the table above shows, a relatively small number of total students are participating, yet both object and script counts are high. This suggests full integration with the over 200 total EAST labs across the nation could lead to an astounding amount of learning and work being produced.

5. CONCLUSIONS

5.1 Summary

Over the last two years, our project at the University of Arkansas, which started as a simple exploration into using virtual worlds to model RFID and more generally as a platform to simulate pervasive computing environments in the real world, has grown to an integrated collection of research projects that is beginning to receive national attention. This thesis reports on one of our thrusts – to develop a high functioning 3D collaborative platform capable of supporting widespread high school participation. Based on initial deployment during a summer course at University of Arkansas and over a school year at Greenland High School, we have demonstrated that today's high school student developers can become productive virtual world developers. We have also established a solid foundation for continuing research into virtual worlds, an exciting new area of computer science.

5.2 Potential Impact

In the small, our project represents an outreach effort that may be able to attract high school students to computing programs including at the University of Arkansas Computer Science and Computer Engineering Department. More broadly, our project explores a path for attracting more students into science, technology, engineering, and mathematics (the so-called STEM) disciplines. It enables these students to collaborate with other students in their own schools but more widely to engage in research on smart world technologies and may enable cooperative work with other students in other EAST schools or even anywhere in the world. Virtual world projects facilitate and encourage cross-disciplinary collaborations. Finally, our

work exposes students to a likely future direction for the World Wide Web, toward a 3D Web, and familiarizes them with technologies to build next generation smart world applications.

5.3 Lessons Learned and Future Work

Our original aim was to include high school students in our research that involves using virtual worlds to explore a future smart world where computing is ubiquitously present and every real world object is a networked object. We overcame some initial hurdles during the summer of 2008 when six high school students joined our project for six weeks during the summer. Their contribution to our project was so impressive that we aimed further to involve them during the school year while they participated remotely from their EAST Labs.

This thesis documents an extended exercise in identifying and solving unexpected hurdles. While it provides the basic functional framework for a collaborative EAST grid, more work is needed in several areas. Our initial goal was to eliminate all barriers against involving collaborative groups in our project as possible. As specified previously, technical solutions were found to most of these hurdles. However, a few of the roadblocks remain unsolved or solved ineffectively and will be discussed further.

In order to add a diverse array of collaborative groups, we must ensure a robust collection of security and permissions options. The ability to partition student avatars to only their own regions, for example, is analogous to one school disallowing entry of students from other schools. We have not tackled this problem in general yet though we can currently give each school a separate set of regions and we believe, given time and more work, we could permit the ability to dynamically collaborate in a controlled manner in a Second Life-like architecture.

As discussed above, endpoint school systems administrators are not uniformly capable of changing firewalls or administrating a system like OpenSim. Thus, we have chosen to

administer OpenSim servers from campus and this requires some of our research team to be available for system maintenance. What happens, then, when core project developers graduate? Although the research is blossoming, spending developer resources on maintenance of older projects rather than the creation of new ones is suboptimal. Having the EAST Initiative (central offices in Little Rock) add virtual worlds to their recommended list of software and having vanguard schools adopt this technology and highlight it in the annual EAST Conference would go a long way in legitimizing virtual worlds within EAST, reducing any concern that this technology is just a distraction and a cover story for introducing games to high schools. (We have not heard this complaint but can imagine it as a possible concern.)

To avoid the long term administrative burden to our little research project of supporting virtual worlds for a growing number of EAST schools, our initial plan was to establish a strong beachhead and transition responsibility to the EAST Initiative. Alternative solutions for this could include requesting funding from NSF or other sources based on the interesting results achieved or transplanting the project to future virtual worlds platforms with better central administrative support. But some significant problems remain before we believe this can succeed. First, OpenSimulator needs to become more stable (or be replaced by another platform that wins community approval). Second, standards for preserving and sharing assets are needed so that work done in the past can easily be upgraded to be useful in the future. Third, we need more experience with working through the administrative and firewall issues. While we succeeded with one school, spending weeks to months with each school would be cost prohibitive. Still, it seems likely that we could find other early adopter schools that are interested in this technology.

EAST students have shown themselves to be rapid learners and can be highly productive with the right level of guidance. When told to "do something productive" and given a high level goal like "Build a smart school," however, we have so far found that there was a lack of ability to successfully design useful additions to the project. At present, it is left to us to provide specific guidance, which is an infeasible proposition when dealing with more than a few schools and was even difficult when we had just six dedicated students. Without guidance, students thus far are prone to random acts of creativity resulting in guns that shoot cats or a rubber duck dispenser rather than subprojects that are integrated into a full demonstration of an idea, but as project scale expands, we are stretched to our limits to provide the necessary guidance. Creative freedom and a non-slave driving approach is certainly necessary for keeping students motivated and interested, yet to achieve our goals, project involvement must be maintained. Explorations into how best to prepare student leaders to take initiative in their research are yet to be done, but it is clear that an open-ended approach must be favored over lists of very specific projects.

On the hardware side, we have already seen rapid demand for increasing resources from the solitary high school currently involved. Our small three-machine grid is capable of supporting our local projects at Greenland High School, but the time is rapidly approaching when upgrades will become necessary. Fortunately, we assembled the entire framework with this scalability in mind, so architectural changes will be minimal aside from acquiring new hardware.

Finally, Second Life and OpenSimulator by association are showing their age. New virtual world technologies are springing up. Some of these have interesting advantages over Second Life and OpenSimulator, though none beyond Open Simulator have both a large installed base with easy model sharing worldwide. Still, that open source system is a moving target.

Continued research into virtual world platforms will help us transition our project if and when the time is right. One next step we are considering is a Reference Model document that compares the various virtual worlds along a collection of architectural and maturity dimensions.

5.4 News Update

In recent weeks (after the design, development, deployment, testing, and documentation of our Open Simulation-based solution and this thesis), it has come to our attention that a Harvard project, The *Immersive Education Initiative* [23], has prepared and recently fielded a system similar (and in several small ways more extensive) than our own. Their system is similar to ours in that it also depends on OpenSimulator, has been stabilized, and includes similar administrative and security improvements. In addition, they have made further changes, one being that they also include in their release an augmentation to OpenSimulator called RealXtend (a third party improvement which would be a possible straightforward addition to our project) which adds mesh graphical data types in addition to Second Life compatible prims (which are efficient but not industry standard). The Harvard system is more widely fielded than our system and is available to the public, not just to EAST high schools. Still, at this time, it is unknown how widely it has been tested in secondary schools. In a way, this Harvard effort may be a blessing in disguise since we can contribute to their effort and it may give us a way to involve Greenland and other EAST high schools in virtual worlds without our project having to carry the heavy burden of deployment and support.

BIBLIOGRAPHY

[1] EAST Initiative. (2009) EAST Initiative Introduction. [Online]. http://www.eastproject.org

[2] Howard Palmer, "DigiBarn," 2004. [Online]. http://www.digibarn.com/collections/games/xerox-maze-war/index.html#origins

[3] Bruce Damer, "WorldsAway," 1997. [Online]. http://www.digitalspace.com/avatars/book/fullbook/chwa/chwa1.htm

[4] Business Wire, "DreamScape Avatar Communities by Fujitsu Expand Onto The Internet; Fujitsu Launches Web Services," (1997, October) [Online] http://www.allbusiness.com/technology/software-services-applications-internet/7054929-1.html

[5] Jim Bumgardner, "Psychology of Cyberspace - On Being A God In The Palace," 2001. [Online]. http://www-usr.rider.edu/~suler/psycyber/jbum.html

[6] Elisa Group, "Virtual Helsinki," (1996) Helsinki, Finland. [Online] http://www.virtualhelsinki.net/english/index.htm

[7] Website: Second Life, Linden Labs, December, 2009. [Online]. http://secondlife.com/whatis/economy_stats.php

[8] Business Week. (2006, November) BW Magazine - Second Life Lessons. [Online]. http://www.businessweek.com/magazine/content/06_48/b4011413.htm

[9] Grant Robertson. "Microsoft Builds Second Life Game To Promote Visual Studio," (2007, May) Download Squad. [Online]. http://www.downloadsquad.com/2007/05/16/microsoft-builds-second-life-game-to-promote-visual-studio/

[10] Shruti Gandhi. "IBM Dives Into Second Life," (2009, August) IBM Corporate. [Online]. http://www.ibm.com/developerworks/opensource/library/os-social-secondlife/?ca=drs-

[11] Intel Corp. "Second Life: Intel Engagement," (2010, January) Intel Corporation. [Online]. http://www.intel.com/general/second-life.htm?iid=personal+gaming_second-life

[12] Microsoft Corp. (2009) Microsoft EMEA Press Centre. [Online]. http://www.microsoft.com/emea/presscentre/SecondLife/default.mspx

[13] Intel Corp. (2010) Intel Ignite. [Online]. http://softwarecommunity.intel.com/articles/eng/1283.htm?iid=secondlife+body_isn

[14] Intel Corp. “Intel Ignites Orange County Choppers Island In Second Life,” (2010) Intel Software College. [Online]. http://download.intel.com/pressroom/kits/embedded/pdfs/OCC_2L_MediaAlert.pdf

[15] IBM Corp. (2010) IBM Business Center. [Online]. http://www.ibm.com/3dworlds/businesscenter/us/en/

[16] IBM Corp. (2010) Addressing Soaring IT Energy Costs. [Online]. https://event.on24.com/eventRegistration/EventLobbyServlet?target=registration.jsp&eventid=113685&sessionid=1&key=D404FB0B032A78672EDCFA3A7DFF3A54&partnerref=web&sourcepage=register

[17] Linden Labs. (2009, February) Second Life Work - Case Studies: IBM. [Online]. http://work.secondlife.com/en-US/successstories/case/ibm/

[18] Casey Bailey. (2010) X10 Workshop on Extensible Virtual Worlds, University of Arkansas Virtual Worlds. [Online]. http://vw.ddns.uark.edu/X10

[19] Brad Stone. “Google Introduces a Cartoonlike Method for Talking in Chat Rooms,” (2008, July) New York Times Online. [Online]. http://www.nytimes.com/2008/07/09/technology/09google.html?_r=1&th&emc=th&oref=slogin

[20] Open Source Community. (2010) Open Wonderland. [Online]. http://openwonderland.org/

[21] Nicole Yankelovich. (2010, March) Open Wonderland. [Online]. http://mfeldstein.com/open-wonderland/

[22] United States Federal Government. (1998) COPPA - Child Online Protection and Privacy Act. [Online]. http://www.coppa.org

[23] Harvard Immersive Education Initiative. (2010, April) Immersive Education. [Online]. http://immersiveeducation.org

APPENDIX A

This Registration API developed for this thesis creates a method for users to register for both Linden Labs' Second Life and OpenSimulator. Originally, Linden Labs' registration forced the use of Ruby as a language, thus I created an initial script *register.rb*. PHP support was added later, along with an upgraded security/authentication specification. Since a rewrite was necessary anyway, I migrated the API to PHP, which is the foundation of the rest of our web server. The API itself (llsd.php) handles data transmission and authentication with Linden Labs server daemons. This API relies on data supplied by the registration script *register.php*. This script is fed data from a web-facing HTML form *regapi.html*. When the project migrated to OpenSimulator, we ceased to need a full API and instead were able to register users by directly adding their data to the grid database, so I was able to take the previous registration code and adapt it to SQL rather than XML transmission.

LLSD.PHP

```
<?php
// VERSION 1.0
// Read the incoming LLSD post data, and decode it into native PHP objects

if (version_compare(PHP_VERSION,'5','>='))
 require_once('domconvert.php');

function llsd_parse_body()
{
        $doc = file_get_contents("php://input");
        return llsd_decode($doc);
}


// Do a simple get of an URL, return the LLSD doc as native PHP objects
function llsd_get($url)
{
        $ch = curl_init($url);
        curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
   curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
```

```
	$doc = curl_exec($ch);

	$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
	curl_close($ch);

	if ($code >= 400)
	{
		return NULL;
	}

	return llsd_decode($doc);
}


// Post a native PHP object as an LLSD document, return the resulting LLSD doc as native PHP
objects,
function llsd_post($url, $node)
{
	$str = llsd_encode($node);
   return llsd_post_string($url, $str);
}


function llsd_put($url, $node)
{
	$str = llsd_encode($node);
	return llsd_put_string($url, $str);
}


function llsd_delete($url)
{
	$ch = curl_init($url);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
	curl_setopt($ch, CURLOPT_POST, TRUE);
	curl_setopt($ch, CURLOPT_FAILONERROR, 1);
	curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
	curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "DELETE");
	curl_setopt($ch, CURLOPT_HTTPHEADER, Array("Content-Type: application/xml"));
	curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
	$doc = curl_exec($ch);
	$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
	curl_close($ch);
	return llsd_decode($doc);
}
```

```
function llsd_post_string($url, $str)
{
	$ch = curl_init($url);
   curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, FALSE);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
	curl_setopt($ch, CURLOPT_POST, TRUE);
	curl_setopt($ch, CURLOPT_FAILONERROR, 1);
	curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
	curl_setopt($ch, CURLOPT_HTTPHEADER, Array("Content-Type: application/xml"));
	curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
	$doc = curl_exec($ch);
	$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
	curl_close($ch);

	return llsd_decode($doc);
}

function llsd_put_string($url, $str)
{
	$tmp = tmpfile();

	// FIXME: Drops the string into a temporary file and uses CURL
	// to PUT it, blech.
	fwrite($tmp, $str);
	fseek($tmp, 0);
	$ch = curl_init($url);
	curl_setopt($ch, CURLOPT_RETURNTRANSFER, TRUE);
	curl_setopt($ch, CURLOPT_PUT, TRUE);
	curl_setopt($ch, CURLOPT_INFILE, $tmp);
	curl_setopt($ch, CURLOPT_INFILESIZE, strlen($str));
	curl_setopt($ch, CURLOPT_FAILONERROR, 1);
	curl_setopt($ch, CURLOPT_FOLLOWLOCATION, 1);
	curl_setopt($ch, CURLOPT_HTTPHEADER, Array("Content-Type:
application/xml","Transfer-Encoding: chunked"));
	#curl_setopt($ch, CURLOPT_POSTFIELDS, $str);
	$doc = curl_exec($ch);
	$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
	curl_close($ch);
	fclose($tmp);
	return llsd_decode($doc);
}

// Encode a native PHP object into an LLSD representation
function llsd_encode(&$node)
{
```

```
	$doc = domxml_new_doc("1.0");
	$llsd_element = $doc->create_element("llsd");
	$doc->append_child($llsd_element);
	$llsd_element->append_child(llsd_encode_node($doc, $node));
	return $doc->dump_mem();
}


// Takes in an llsd document, returns a native PHP object
function llsd_decode($str)
{
	// error_log($str);
	// Generate the DOM tree from the text document
	$error = array();
	if (!$dom = domxml_open_mem($str, DOMXML_LOAD_PARSING, $error))
	{
		// Probably should generate something with errors, but just bail for now

		return NULL;
	}

	$dom_root = $dom->document_element();
	if (!$dom_root)
	{
		return NULL;
	}

   $child = $dom_root;
	// Iterate through all of the children looking for
	// the LLSD node.
	// $child = $dom_root->first_child();
	while ($child)
	{

		if (!($child->is_blank_node()))
		{
			switch ($child->node_type())
			{

			case XML_TEXT_NODE:
				// Skip text not enclosed in the LLSD tag
				break;
			case XML_ELEMENT_NODE:
				// Now fill in cur_key or cur_value depending on the node
				if ('llsd' == $child->node_name())
				{
```

```
                                        // We've found the root node of the LLSD tree, now have at
it.
                                        return llsd_parse_root($child);
                                }
                        default:
                                break;
                        }
                }
                $child = $child->next_sibling();
        }

        // No valid LLSD node, return an empty object.
        return array();
}


//
// Implementation details below here, you shouldn't need to use any of this functionality
//


// Generates the DOM tree for a particular branch
function llsd_encode_node(&$doc, &$node)
{
        if (is_array($node))
        {
                // Figure out if it's a map or array,
                // Assume anything with sequential integer keys starting at 0 is an array
                $keys = array_keys($node);
                $is_array = true;
                $cur_key = 0;
                foreach ($keys as $key)
                {
                        if (is_int($key))
                        {
                                if (!$cur_key == $key)
                                {
                                        $is_array = false;
                                        break;
                                }
                                $cur_key++;
                        }
                        else
                        {
                                $is_array = false;
                                break;
```

```
                        }
                }
                if ($is_array)
                {
                        return llsd_encode_array($doc, $node);
                }
                else
                {
                        return llsd_encode_map($doc, $node);
                }
        }
        else if (is_int($node))
        {
                return llsd_encode_integer($doc, $node);
        }
        else if (is_float($node))
        {
                return llsd_encode_real($doc, $node);
        }
        else
        {
                // Default to string for everything else
                return llsd_encode_string($doc, $node);
        }
}

function llsd_encode_array(&$doc, &$node)
{
        $map_element = $doc->create_element("array");

        $count = count($node);
        for ($i = 0; $i < $count; $i++)
        {
                $value_element = llsd_encode_node($doc, $node[$i]);
                $map_element->append_child($value_element);
        }
        return $map_element;
}

function llsd_encode_map(&$doc, &$node)
{
        $map_element = $doc->create_element("map");

        foreach ($node as $key => $value)
        {
                $key_element = $doc->create_element("key");
```

```
                $key_text = $doc->create_text_node(utf8_encode($key));
                $key_element->append_child($key_text);

                $value_element = llsd_encode_node($doc, $value);

                $map_element->append_child($key_element);
                $map_element->append_child($value_element);
        }
        return $map_element;
}

function llsd_encode_integer(&$doc, &$node)
{
        $element = $doc->create_element("integer");
        $text = $doc->create_text_node(utf8_encode($node));
        $element->append_child($text);
        return $element;
}

function llsd_encode_real(&$doc, &$node)
{
        $element = $doc->create_element("real");
        $text = $doc->create_text_node(utf8_encode($node));
        $element->append_child($text);
        return $element;
}

function llsd_encode_string(&$doc, &$node)
{
        $element = $doc->create_element("string");
        $text = $doc->create_text_node(utf8_encode($node));
        $element->append_child($text);
        return $element;
}

function llsd_parse_root($node)
{
        // Root can have only one "value".  Skip all text fields.
        // FIXME: We ignore "extra" values in the root node if there are more than one.
        // should we be more forceful and error out here?

        // Iterate through all of the children looking for
        // an xml element node
        $child = $node->first_child();
        while ($child)
        {
```

```
		if (!($child->is_blank_node()))
		{
			switch ($child->node_type())
			{
			case XML_TEXT_NODE:
				// Skip text
				// FIXME: Should only skip whitespace, should error out on non-
whitespace?
				break;
			case XML_ELEMENT_NODE:
				// Now fill in cur_key or cur_value depending on the node
				return llsd_parse_value($child);
				break;
			default:
				break;
			}
		}
		$child = $child->next_sibling();
	}

	// No LLSD node found, return an empty array
	return array();
}

function llsd_parse_value($node)
{
	$cur_value = "";

	switch ($node->node_type())
	{
	case XML_ELEMENT_NODE:
		switch ($node->node_name())
		{
		case 'map':
			$cur_value = llsd_parse_map_contents($node);
			break;
		case 'array':
			$cur_value = llsd_parse_array_contents($node);
			break;
		case 'binary':
			// FIXME: Implement binary handler (pull out encoding, decode base64
binary?
			$cur_value = NULL;
		default:
			// Everything else is handled via the generic handler, which will default to
			// treating it as a string
```

```
                        // $cur_value = llsd_parse_contents($node->node_name);
                        $cur_value = llsd_parse_contents($node, $node->node_name());
                        break;
                }
                break;
        case XML_TEXT_NODE:
                // Skip this, it's not a "value".  Should never happen, the caller should notice this.
                break;
        }
        return $cur_value;
}

//
// Parse the contents of an LLSD map from the DOM branch
//
function llsd_parse_map_contents($branch)
{
        $object = array();
        $objptr = &$object;


        $cur_key = '';
        $cur_value = '';
        // Iterate through all of the children.
        // The children need to alternate between keys and values
        $child = $branch->first_child();
        while ($child)
        {
                if (!($child->is_blank_node()))
                {
                        switch ($child->node_type())
                        {
                        case XML_TEXT_NODE:
                                // FIXME: Should verify that this is whitespace?
                                break;
                        case XML_ELEMENT_NODE:
                                // Now fill in cur_key or cur_value depending on the node
                                if ('key' == $child->node_name())
                                {
                                        $cur_key = llsd_parse_contents($child, 'string');
                                }
                                else // Switch based on different LLSD types
                                {
                                        $cur_value = llsd_parse_value($child);

                                        // We've got a key/value pair, add it to the map.
```

```
                                        $object[$cur_key] = $cur_value;
                                        $cur_value = '';
                                }
                                break;
                        }
                }
                $child = $child->next_sibling();
        }
        return $object;
}

//
// Parse an LLSD array from a DOM branch
//
function llsd_parse_array_contents($branch)
{
        $object = array();
        $objptr = &$object;

        $cur_key = '';
        $cur_value = '';
        // Iterate through all of the children.
        // The children need to alternate between keys and values
        $child = $branch->first_child();
        while ($child)
        {
                if (!($child->is_blank_node()))
                {
                        switch ($child->node_type())
                        {
                        case XML_TEXT_NODE:
                                // FIXME: Should verify that this is whitespace?
                                break;
                        case XML_ELEMENT_NODE:
                                // We've got a key/value pair, add it to the map.
                                $object[] = llsd_parse_value($child);
                                $cur_value = '';
                                break;
                        }
                }
                $child = $child->next_sibling();
        }
        return $object;
}

// function llsd_parse_contents($branch, $type)
```

```
function llsd_parse_contents($branch, $type)
{
	$child = $branch->first_child();
	while ($child)
	{
		if (!($child->is_blank_node()))
		{
			switch ($child->node_type())
			{
			case XML_TEXT_NODE:
				switch ($type)
				{
				case 'integer':
					return (int)$child->get_content();
				case 'real':
					return (float)$child->get_content();
				case 'bool':
					if ("true" == $child->get_content())
					{
						return true;
					}
					else
					{
						return false;
					}
				default:
					// Treat everything else as a string
					return $child->get_content();
				}
			}
		}
		$child = $child->next_sibling();
	}
	return false;
}
```

REGISTER.PHP

```
<?php
// VERSION 1.0
// Read the incoming LLSD post data, and decode it into native PHP objects
if (version_compare(PHP_VERSION,'5','>='))
 require_once('domconvert.php');

require_once('llsd.php');
```

```
// FILL THESE IN WITH YOUR OWN CAPABILITY URLS
define('URI_CREATE_USER', 'https://cap.secondlife.com/cap/0/a8564453-97b7-4078-993d-
718fbd56929b');
define('URI_GET_LAST_NAMES', 'https://cap.secondlife.com/cap/0/5a5ecced-ec12-428b-
9887-dcf8387e885a');
define('URI_CHECK_NAME', 'https://cap.secondlife.com/cap/0/68006f23-ea34-42a9-a614-
2dde08923c89');

if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
   if (is_name_available($_POST['username'], $_POST['last_name_id']))
   {
      $user = array
      (
         'username'     => $_POST['username'],
         'last_name_id' => (int)$_POST['last_name_id'],
         'email'        => $_POST['email'],
         'password'     => $_POST['password'],
         'dob'          => $_POST['dob_year'].'-'.$_POST['dob_month'].'-'.$_POST['dob_day']
      );
      $result = llsd_post(URI_CREATE_USER, $user);
      if($result['agent_id'])
      {
        echo "<h3>Successful!</h3> <p> Agent ID is: ";
         echo $result['agent_id'];


         $host = "localhost";
                     $user = "casey";
                     $pass = "test";
                     $db = "casey";
                     $last_nameID = (int)$_POST['last_name_id'];

                     $pFirst = $_POST['personalfirst'];
                     $pLast = $_POST['personallast'];
                     $email = $_POST['email'];
                     $aFirst = $_POST['username'];

                     $last_names = llsd_get(URI_GET_LAST_NAMES);
                     $aLast = $last_names[ "$last_nameID" ];

                     $connection = mysql_connect($host, $user, $pass) or die ("Unable to
connect!");

                     mysql_select_db($db);
```

```
                $query = "INSERT INTO avatarDB (pFirst, pLast, email, aFirst, aLast)
VALUES ('$pFirst','$pLast','$email','$aFirst', '$aLast')";
                mysql_query($query) or die ('Error, insert query failed');

                mysql_close($connection);
                   echo "<br>Successfully added entry. <br> First = " . $aFirst . " <br>Last =".
$aLast. "<br>";

                   echo "<br><br>You can download the client from <a
href=\"http://download-secondlife-com.s3.amazonaws.com/Second_Life_1-21-6-
99587_Setup.exe\">Here</a>.";

                   echo "<br>You can teleport directly to our island by clicking here:  <a
href=\"http://slurl.com/secondlife/University%20of%20Arkansas/71/81/152/\">Class
Platform</a>.<br>";

      }
     else if ($result[0] == 95)
      {
         print "Failed: Email already in use!\n";
      }
      else
      {
         print "Failed!  Error code: " . $result[0] . "\n";
      }

   }
   else
   {
      echo 'SL name not available.';
   }

}
if (!($_POST["testval"]))
{
?>
<html>
<body>
<h1 align="center">Create Second Life Account - Adult Grid</h1>

<form action="./index.php?page=register" method="post">

<table border="0" cellpadding="3" cellspacing="0">
<tr>
  <td><h2 align="center">Personal Information</h2></td>
</tr>
```

```
<tr>
  <td>First Name:</td>
  <td><input type="text" name="personalfirst" size="25" maxlength="31" value="" /></td>
</tr>
<tr>
  <td>Last Name:</td>
  <td><input type="text" name="personallast" size="25" maxlength="31" value="" /></td>
</tr>

<tr>
  <td><h2 align="center">Avatar Information</h2></td>
</tr>
<tr>
  <td>First name:</td>
  <td><input type="text" name="username" size="25" maxlength="31" value="" /></td>
</tr>
<tr>
  <td>Last name:</td>
  <td>
  <select name="last_name_id">
  <?php
  $last_names = llsd_get(URI_GET_LAST_NAMES);
  foreach ($last_names as $last_name_id => $name)
  {
     print '<option value="'.$last_name_id.'">'.$name.'</option>';
  }
  ?>
  </select>
  </td>
</tr>
<tr>
    <td>Password:</td>
    <td><input type="password" name="password" size="20" value="" /></td>
</tr>
<tr>
    <td>Email:</td>
    <td><input type="text" name="email" size="35" value="" /></td>
</tr>
<tr>
    <td>Date of birth:</td>
    <td>
    <select name="dob_day">
    <?php
    $days = get_days();
    foreach ($days as $key => $value) { print '<option value="'.$key.'"
'.$selected.'>'.$value.'</option>'; }
```

```
   ?>
   </select>

   <select name="dob_month">
   <?php
   $months = get_months();
   foreach ($months as $key => $value) { print '<option value="'.$key.'"
'.$selected.'>'.$value.'</option>'; }
   ?>
   </select>

   <select name="dob_year">
   <?php
   $years = get_years();
   foreach ($years as $key => $value) { print '<option value="'.$key.'"
'.$selected.'>'.$value.'</option>'; }
   ?>
   </select>
   </td>
</tr>
<tr>
   <td><input type="hidden" value="1" name="testval" id="testval" /></td>
   <td><input type="submit" value="Create SL Account" /></td>
</table>
</form>
Please note registering certifies you agree to the <a
href="http://secondlife.com/corporate/tos.php">Second Life TOS</a> and the <a
href="http://uits.uark.edu/policies/code.htm">UARK TOS</a> .<br>
</head>
</html>
<?php
}
function get_months()
{
   $months = array();
   for ($i = 1; $i <= 12; $i++)
   {
      $key = date('n', mktime(0, 0, 0, $i, 1, 2000));
      $value = date('M.', mktime(0, 0, 0, $i, 1, 2000));
      $months[sprintf("%02d", $key)] = $value;
   }
   return $months;
}

function get_years()
{
```

```
    $today = getdate();
    $max_year = $today['year'] - 90;
    $min_year = $today['year'] - 18;

    $years = array();
    for ($i = $min_year; $i >= $max_year; $i--)
    {
        $years[$i] = $i;
    }
    return $years;
}

function get_days()
{
    $days = array();
    for ($i = 1; $i <= 31; $i++)
    {
        $days[sprintf("%02d", $i)] = sprintf("%02d", $i);
    }
    return $days;
}

function is_name_available($username, $last_name_id)
{
    $params = array('username' => $username, 'last_name_id' => (int)$last_name_id);

    if (llsd_post(URI_CHECK_NAME, $params) == 'true')
    {

        return true;
    }

    return false;
}
?>
```

REGISTER.RB

```
require 'llsd'

  class Array

    def pick_random
     self[rand(self.length)]
    end
```

```
  end

get_capabilities_url = "https://cap.secondlife.com/get_reg_capabilities"

first = "Dillan"
last = "Potez"


if ARGV.length == 5
   reg_pw = ARGV[0]
   first_name = ARGV[1]
   password = ARGV[2]
   email = ARGV[3]
   dob = ARGV[4]
else
   puts "Please pass arg values: API password, first name, new password"
end


#GET CAPABILITIES
puts "===Getting capabilities===\n"

post_body = "first_name=#{first}&last_name=#{last}&password=#{reg_pw}"
response_xml = LLSD.post_urlencoded_data get_capabilities_url, post_body

capability_urls = LLSD.parse(response_xml)
capability_urls.each { |k,v| puts "#{k} => #{v}" } #Print capabilities

#### ERROR CODES
puts "===Get Error Codes==="

if capability_urls['get_error_codes']
   response_xml = LLSD.http_raw capability_urls['get_error_codes']

   # puts response_xml  ## Write out xml response

   error_codes = LLSD.parse( response_xml )

   error_codes.each {|error| puts "#{error[0]} => #{error[1]}"} #Print error codes
else
   puts "Get error codes capability not available to user. Exiting\n"
   exit
end


#### LAST NAMES
```

```
puts "\n\n===Get Last Names===\n"

if capability_urls['get_last_names']
   response_xml = LLSD.http_raw capability_urls['get_last_names']

   #puts response_xml  ## Write xml response

   last_names = LLSD.parse(response_xml)
   last_names.each {|k,v| puts "#{k} => #{v}"}

   ### CHECK NEW REGISTRATION NAME ###
   puts "\n\n=== Check Name ===\n"
   if capability_urls['check_name']
      valid_surname_id = last_names.keys.pick_random
      params_hash = {"username" => first_name, "last_name_id" => valid_surname_id}

      xml_to_post = LLSD.to_xml(params_hash)

      response_xml = LLSD.post_xml(capability_urls['check_name'], xml_to_post)

      puts "Posted xml:"
      puts xml_to_post

      puts "xml response: "
      puts response_xml

      is_name_available = LLSD.parse(response_xml)

      puts "Result (Is name available?): #{is_name_available}"


      ## CREATE USER
      puts "\n\n===CREATE USER===\n"
      if is_name_available and capability_urls['create_user']:
         params_hash["email"] = email
         params_hash["password"] = password
         params_hash["dob"] = dob

         xml_to_post = LLSD.to_xml(params_hash)

         response_xml = LLSD.post_xml(capability_urls['create_user'], xml_to_post)
         puts "XML SENT"
         puts xml_to_post


         puts "XML RESPONSE:"
```

```
        puts response_xml

        result_hash = LLSD.parse(response_xml)
        puts "DEBUG"
        puts result_hash
        puts "DEBUG"

        puts "New Agent ID: #{result_hash['agent_id']}"
        puts "Name: #{first_name} #{last_names[valid_surname_id]}"
      else
        puts "Create_user capability not granted"
      end
   else
      puts "Check_name capability not granted"
   end
else
   puts "Get Last Names capability not granted"
end
```

REGISTER.HTML

```
<html>
<body>
<h1 align="center">Create Second Life Account - Adult Grid</h1>
<form action="./index.php?page=register" method="post">
<table border="0" cellpadding="3" cellspacing="0">
<tr>
  <td><h2 align="center">Personal Information</h2></td>
</tr>
<tr>
  <td>First Name:</td>
  <td><input type="text" name="personalfirst" size="25" maxlength="31" value="" /></td>
</tr>
<tr>
  <td>Last Name:</td>
  <td><input type="text" name="personallast" size="25" maxlength="31" value="" /></td>
</tr>
<tr>
  <td><h2 align="center">Avatar Information</h2></td>
</tr>
<tr>  <td>First name:</td>
```

```
  <td><input type="text" name="username" size="25" maxlength="31" value="" /></td>
</tr>
<tr>
  <td>Last name:</td>
  <td>
  <select name="last_name_id">
  <br />
  </select>
  </td>
</tr>
<tr>
   <td>Password:</td>
   <td><input type="password" name="password" size="20" value="" /></td>
</tr>
<tr>
   <td>Email:</td>
   <td><input type="text" name="email" size="35" value="" /></td>

</tr>
<tr>
   <td>Date of birth:</td>
   <td>
   <select name="dob_day">
   <option value="01" >01</option><option value="02" >02</option><option value="03"
>03</option><option value="04" >04</option><option value="05" >05</option><option
value="06" >06</option><option value="07" >07</option><option value="08"
>08</option><option value="09" >09</option><option value="10" >10</option><option
value="11" >11</option><option value="12" >12</option><option value="13"
>13</option><option value="14" >14</option><option value="15" >15</option><option
value="16" >16</option><option value="17" >17</option><option value="18"
>18</option><option value="19" >19</option><option value="20" >20</option><option
value="21" >21</option><option value="22" >22</option><option value="23"
>23</option><option value="24" >24</option><option value="25" >25</option><option
value="26" >26</option><option value="27" >27</option><option value="28"
>28</option><option value="29" >29</option><option value="30" >30</option><option
value="31" >31</option>    </select>
   <select name="dob_month">
   <option value="01" >Jan.</option><option value="02" >Feb.</option><option value="03"
>Mar.</option><option value="04" >Apr.</option><option value="05" >May.</option><option
value="06" >Jun.</option><option value="07" >Jul.</option><option value="08"
```

```
>Aug.</option><option value="09" >Sep.</option><option value="10" >Oct.</option><option
value="11" >Nov.</option><option value="12" >Dec.</option>    </select>
   <select name="dob_year">
   <option value="1992" >1992</option><option value="1991" >1991</option><option
value="1990" >1990</option><option value="1989" >1989</option><option value="1988"
>1988</option><option value="1987" >1987</option><option value="1986"
>1986</option><option value="1985" >1985</option><option value="1984"
>1984</option><option value="1983" >1983</option><option value="1982"
>1982</option><option value="1981" >1981</option><option value="1980"
>1980</option><option value="1979" >1979</option><option value="1978"
>1978</option><option value="1977" >1977</option><option value="1976"
>1976</option><option value="1975" >1975</option><option value="1974"
>1974</option><option value="1973" >1973</option><option value="1972"
>1972</option><option value="1971" >1971</option><option value="1970"
>1970</option><option value="1969" >1969</option><option value="1968"
>1968</option><option value="1967" >1967</option><option value="1966"
>1966</option><option value="1965" >1965</option><option value="1964"
>1964</option><option value="1963" >1963</option><option value="1962"
>1962</option><option value="1961" >1961</option><option value="1960"
>1960</option><option value="1959" >1959</option><option value="1958"
>1958</option><option value="1957" >1957</option><option value="1956"
>1956</option><option value="1955" >1955</option><option value="1954"
>1954</option><option value="1953" >1953</option><option value="1952"
>1952</option><option value="1951" >1951</option><option value="1950"
>1950</option><option value="1949" >1949</option><option value="1948"
>1948</option><option value="1947" >1947</option><option value="1946"
>1946</option><option value="1945" >1945</option><option value="1944"
>1944</option><option value="1943" >1943</option><option value="1942"
>1942</option><option value="1941" >1941</option><option value="1940"
>1940</option><option value="1939" >1939</option><option value="1938"
>1938</option><option value="1937" >1937</option><option value="1936"
>1936</option><option value="1935" >1935</option><option value="1934"
>1934</option><option value="1933" >1933</option><option value="1932"
>1932</option><option value="1931" >1931</option><option value="1930"
>1930</option><option value="1929" >1929</option><option value="1928"
>1928</option><option value="1927" >1927</option><option value="1926"
>1926</option><option value="1925" >1925</option><option value="1924"
>1924</option><option value="1923" >1923</option><option value="1922"
>1922</option><option value="1921" >1921</option><option value="1920" >1920</option>
</select>
```

```
   </td>
</tr>
<tr>
   <td><input type="hidden" value="1" name="testval" id="testval" /></td>
   <td><input type="submit" value="Create SL Account" /></td>
</table>
</form>
Please note registering certifies you agree to the <a
href="http://secondlife.com/corporate/tos.php">Second Life TOS</a> and the <a
href="http://uits.uark.edu/policies/code.htm">UARK TOS</a> .<br>
</head>
</html>
```

APPENDIX B

This patch was submitted and accepted by OpenSimulator's development team. It introduces a fuzzy leeway period into the database connection's timeout period. Prior to this addition, database operations which took longer than the hard-coded "wait" period would frequently cause database corruption as the grid force-dropped the database connection while data was being transmitted.

CODE

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.IO;
using System.Reflection;
using log4net;
using MySql.Data.MySqlClient;
using OpenMetaverse;
using OpenSim.Framework;
namespace OpenSim.Data.MySQL
{
   /// <summary>
   /// A MySQL Database manager
   /// </summary>
   public class MySQLManager
   {
      private static readonly ILog m_log =
LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

      /// <summary>
      /// The database connection object
      /// </summary>
      private MySqlConnection dbcon;
      /// <summary>
      /// Connection string for ADO.net
      /// </summary>
      private string connectionString;
```

```
        private const string m_waitTimeoutSelect = "select @@wait_timeout";
        /// <summary>
        /// Wait timeout for our connection in ticks.
        /// </summary>
        private long m_waitTimeout;
        /// <summary>
        /// Make our storage of the timeout this amount smaller than it actually is, to give us a
margin on long
        /// running database operations.  Adding this prevents database corruption when operations
overflow the wait Timeout
        /// </summary>
        private long m_waitTimeoutLeeway = 60 * TimeSpan.TicksPerSecond;
        /// <summary>
        /// Holds the last tick time that the connection was used.
        /// </summary>
        private long m_lastConnectionUse;
        /// <summary>
        /// Initialises and creates a new MySQL connection and maintains it.
        /// </summary>
        /// <param name="hostname">The MySQL server being connected to</param>
        /// <param name="database">The name of the MySQL database being used</param>
        /// <param name="username">The username logging into the database</param>
        /// <param name="password">The password for the user logging in</param>
        /// <param name="cpooling">Whether to use connection pooling or not, can be one of the
following: 'yes', 'true', 'no' or 'false', if unsure use 'false'.</param>
        /// <param name="port">The MySQL server port</param>
        public MySQLManager(string hostname, string database, string username, string password,
string cpooling,
                            string port)
        {
            string s = "Server=" + hostname + ";Port=" + port + ";Database=" + database + ";User
ID=" +
                username + ";Password=" + password + ";Pooling=" + cpooling + ";";
            Initialise(s);
        }
        /// <summary>
        /// Initialises and creates a new MySQL connection and maintains it.
        /// </summary>
        /// <param name="connect">connectionString</param>
        public MySQLManager(String connect)
```

```
        {
            Initialise(connect);
        }

        /// <summary>
        /// Initialises and creates a new MySQL connection and maintains it.
        /// </summary>
        /// <param name="connect">connectionString</param>
        public void Initialise(String connect)
        {
            try
            {
                connectionString = connect;
                dbcon = new MySqlConnection(connectionString);

                try
                {
                    dbcon.Open();
                }
                catch(Exception e)
                {
                    throw new Exception("Connection error while using connection string
["+connectionString+"]", e);
                }
                m_log.Info("[MYSQL]: Connection established");
                GetWaitTimeout();
            }
            catch (Exception e)
            {
                throw new Exception("Error initialising MySql Database: " + e.ToString());
            }
        }

        /// <summary>
        /// Get the wait_timeout value for our connection
        /// </summary>
        protected void GetWaitTimeout()
        {
            MySqlCommand cmd = new MySqlCommand(m_waitTimeoutSelect, dbcon);
```

```
        using (MySqlDataReader dbReader =
cmd.ExecuteReader(CommandBehavior.SingleRow))
        {
            if (dbReader.Read())
            {
                m_waitTimeout
                    = Convert.ToInt32(dbReader["@@wait_timeout"]) * TimeSpan.TicksPerSecond
+ m_waitTimeoutLeeway;
            }
            dbReader.Close();
            cmd.Dispose();
        }
        m_lastConnectionUse = DateTime.Now.Ticks;
        m_log.DebugFormat(
            "[REGION DB]: Connection wait timeout {0} seconds", m_waitTimeout /
TimeSpan.TicksPerSecond);
    }
    /// <summary>
    /// Should be called before any db operation.  This checks to see if the connection has not
timed out
    /// </summary>
    public void CheckConnection()
    {
        //m_log.Debug("[REGION DB]: Checking connection");
        long timeNow = DateTime.Now.Ticks;
        // Check to make sure our connection is still open and NOT TIMED OUT, lest database
corruption occur
        if (timeNow - m_lastConnectionUse > m_waitTimeout || dbcon.State !=
ConnectionState.Open)
        {
            m_log.DebugFormat("[REGION DB]: Database connection has gone away -
reconnecting");
            Reconnect();
        }
        // Strictly, we should set this after the actual db operation.  But it's more convenient to set
here rather
        // than require the code to call another method - the timeout leeway should be large
enough to cover the
        // inaccuracy.  This caused problems before the leeway was added.
        m_lastConnectionUse = timeNow;      }
```