Johannes Luderschmidt
Business Information Service
Royal Holloway University of London

31. December 2007

Technological Foundations Of Second Life

1 Introduction

Recently Second Life (SL) has been intensively discussed. SL is a virtual world where (geographically dispersed) users usually represented by an avatar interact with other users, objects or agents by means of a client called viewer on their PC that communicates with a central server grid. SL seems to be the first commercial virtual world that is used by a big community. As of March 2007 according to the company behind SL Linden Labs 9.5m users have registered for SL. Averagely 50k of them are concurrently active in 8k to 10k regions. These 50k users and 10k regions have to be served by a server grid. However, there is a forecast of 16 million regions, 2bn users and a concurrency of 50m users for virtual worlds in the future. To serve that amount of users and data sophisticated server concepts have to be employed.

This report will try to present and critically analyse the current technological background behind SL. However, for this topic only little information is available in papers and books because the SL server software is still a proprietary system. Alternatively the information provided in this report is mainly based on information given by Linden Labs themselves in their Knowledge Base, their blogs and their Wikis. As the viewer software's source code is published under an Open Source licence plenty of documentation about it is available online. Despite efforts of Linden Labs to open their server protocols to the Open Source community as well documentation for the server software is currently hardly publicly available. Nevertheless, this report tries to interpolate the available online information to give an overview of the current state of the art of SL's technological foundations. This overview cannot be more than a snapshot of SL as it is developing quickly. In depth information is only presented if information was provided by LL. This report will present the most recent technological developments in SL even if they are still in a beta or Release Candidate phase at the time of this writing.

2 Architecture of Second Life

2.1 Second Life as a Virtual World

SL is a virtual world, which was publicly released to the Internet on June, 23rd 2003 (SL Forum, 2003, [23]). A user can log in to SL with a certain client application called viewer. This viewer is available for Microsoft Windows, Mac OS X and Linux operating systems and can be downloaded for free from the Internet. Before a user can log in to SL for the first time it must be registered for free on *secondlife.com*. In the virtual world of Second Life a user's avatar can own and use clothing, objects etc. called assets, which are available via an inventory in the viewer. SL has its own currency called Linden Dollar (abbr. L$) that can be used for payment within the boundaries of SL. The virtual word is subdivided into virtual land of virtual 256mx256m regions, which themselves can be partitioned by the owner into parcels. Each region is controlled on the server side by a so called simulator (see figure 1). Each viewer connects to Second Life via one simulator that is responsible for the region where a user wants to log in. By default it is the region where the user was logged in the last time. Every time a user changes between regions it will be passed over to the simulator that is responsible for that certain region. The viewer receives everything it needs to display from the simulator. A simulator is a basic server process on a so called Sim, which is a physical or virtual server with a URL like *sim1234.agni.lindenlab.com*. It is possible to run several simulator instances on one Sim. A grid according to Linden Labs is a collection of Sims. Linden Labs employs several grids for live use, internal and external testing (Second Life Wiki, 2007, [1]). For each type of grid another kind of viewer is necessary according to the features and software versions of the used simulator types. Behind the grid of simulators is a grid of data servers for instance the user server or the asset server. A simulator serves as a proxy to these databases. Thus every time a user retrieves an asset with its viewer the data has to be forwarded via the simulator to the viewer.

2.2 Simulator

A simulator...

- is responsible how a user's avatar interacts with other avatars and objects in a region.
- is responsible for saving land parcel states, object states and terrain height map states.

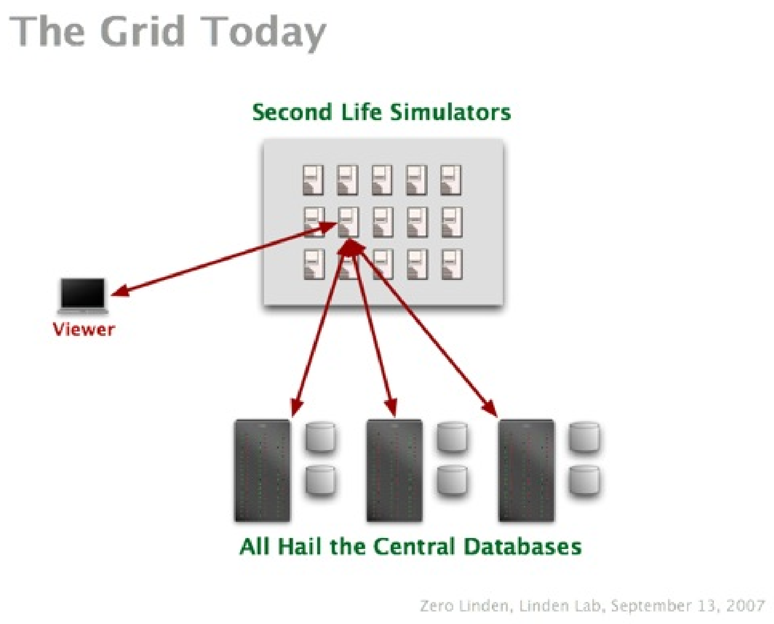
The Grid Today
Second Life Simulators
Viewer
All Hail the Central Databases
Zero Linden, Linden Lab, September 13, 2007

Figure 1: Second Life Application Server Structure. Source: [Figure1]

- carries out visibility calculations on land and objects and conveys the data to the viewer.
- executes physical calculations with the Havok library (see section 2.4).
- is responsible for chat and instant messages.
- runs a special instance of the Mono .NET platform (see section 3.1) to be able to run LSL scripts.

The full performance of a simulator is at 45 fps (Second Life Wiki, 2007, [2]).

2.3 Viewer

The viewer is responsible for client-side activities like downloading necessary data from the simulator and drawing everything that is visible to a user's avatar on screen (Second Life Knowledge Base, 2007, [3]). It takes care for positions of objects and handles velocity or other physical information and displays them with the according objects. The viewer does no collision detection as this is task is up to the simulator.

The Second Life viewer is released under an Open Source License and the source code can be downloaded from the Internet ([31]).

2.4 Physical Calculations

2.4.1 Functionality of Havok

Inside of Second Life a rigid body physics simulation is employed. All avatars and objects are dynamic, collidable and moving. For instance if a chair is created a user will be able to sit on it and if a sphere is tagged to be dynamic it will roll down hills (Secondlife.com, 2007, [8]).

The middleware physics engine that provides algorithms for rigid body dynamics and collision detection that are necessary for real world calculations is a commercial product called *Havok*. It is licensed by Linden Labs from the Irish company Havok ([11]). As of September 2007 according to the company Havok the current version of the Havok physics engine is 5 (Havok, 2007, [12]). However, Havok1 was the physics engine on which SL was built on and operated for several years. Havok 4 is the version that is in the final testing stages for use inside of Second Life.

Tasks for Havok are for instance to figure out what happens when...

- ... something or someone collides with something or someone else. Havok compares two objects' speeds and positions.
- ... something or someone is pushed. When two objects collide Havok calculates the forces on the colliding objects.
- ... friction or damping slows the motion of something or someone. That allows avatar movement like walking or flying. If for instance an avatar walks up a hill Havok will take care that the avatar is lifted vertically after each step and the walking speed is slowed down.
- ... something or someone is in motion and has momentum/energy. The energy can be calculated with mass * velocity. By using this calculation Havok can add up energies of colliding objects or avatars.

Source: (SL Wiki, 2007, [10])

2.4.2 Problems of Havok

Havok 1 sometimes had problems when two objects allocated the same space. They could end up interpenetrating themselves. As a result the simulator could go into a deep recursive loop to analyse the interpenetrating objects that consumes all CPU power and causes the simulator to wheeze. This incident is called *Deep Think* condition (Wikipedia, 2007,

[13]). Newer versions of Havok have an *overlap ejection* functionality, which allows to push apart two separate, interpenetrating objects.

2.5 Assets

An asset is a data resource like an image, sound, script, object, etc. Assets can be downloaded to the viewer or uploaded into a central asset store. Each asset has a unique UUID like. 'It's a 128-bit (16 byte) value which is generated in such a way as to make collisions very unlikely' (Second Life Wiki, 2007, [32]). This section will describe the asset types objects, textures and scripts. All assets are stored in a database. They are retrieved by the client application when a user makes use of them from the inventory.

2.5.1 Objects

An object in Second Life is 'a collection of one or more linked' primitives (Second Life Wiki, 2007, [28]). A primitive (prim) can have different forms and materials (Second Life Wiki, 2007, [30]). Textures can be mapped onto prims. Figure 2 shows different examples for prim forms and prim materials. Part 1 of the image shows prim forms like a cone, torus, sphere or a box that have the material wood. Part 2 shows a so called sculpted prim: It has the form that is described in a special sculpting texture that consists of 3D coordinates. This sculpting texture is mapped onto the prim. With the help of a sculpting texture a primitive can adopt quite complex forms. Part 3 shows different prim modifications of a cylinder such as cutting. Part 4 shows a cut cylinder with a texture mapped on it. Additionally prims can have attributes like colour, shininess etc. They also have a field 'physical'. Objects with 'physical' checked are dynamic. A dynamic sphere for instance will behave like a ball e.g. falling to the ground and rolling down a hill. Dynamic objects can consist of up to 31 linked prims (Second Life Wiki, 2007, [28]).

2.5.2 Textures

Images (Textures) in SL are encoded by means of the JPEG 2000 wavelet compression method. By conveying images as image codestreams according to "Appendix A of JPEG 2000 Part I Final Committee Draft Version 1.0 which are decoded in the client and sent to OpenGL as uncompressed 24 bit or 32 bit textures." (Second Life Wiki, 2007, [4]) By using that method preloaded parts of the image can be shown on objects.

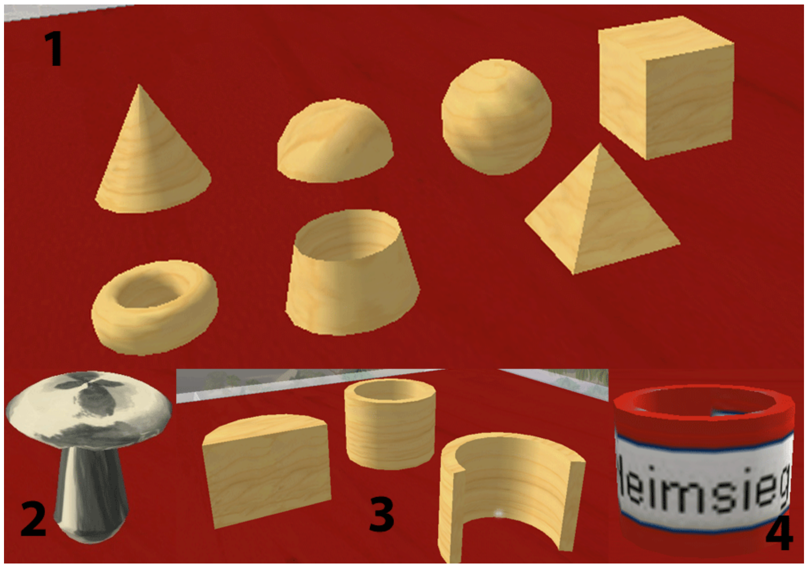

Figure 2: Primitives in Second Life.

2.5.3 Scripts

Scripts are stored in two types in the asset system. As will be explained in section 3.1 the SL scripting language is LSL and scripts are being compiled to byte code and being run by a virtual machine. To enable a user to (re-)edit LSL code in Second Life the system stores as well the source code as the byte code as an asset in the asset system (Second Life Wiki, 2007, [5]). Scripts themselves are being edited directly in SL from the inventory by the user.

2.6 Networking

2.6.1 Login

A user can connect to Second Life by submitting its login data with its viewer. Alternatively a user can click on a so called *SL URL* in a browser, which will start the viewer and connect automatically with the stored login data. In a SL URL a location inside SL's virtual world can be encoded by means of SL coordinates.
E.g. clicking http://slurl.com/secondlife/Neu%20De%20Island/122/77/23/?title=Beach/ will connect the viewer to the simulator called "Neu De Island" and will put the user's avatar onto the coordinate 122/77/23.

In either of both possibilities (connecting manually with the viewer or by clicking a SL URL) the user's viewer will call a CGI script that runs behind the URL *login.agni.lindenlab.com*. This script verifies user name and password and decides to which simulator it connects the viewer. If a URL was given by clicking a SL URL then the script will try to connect to this simulator. Otherwise it will look up the simulator to which the user was logged in the last time. If the simulator verifies that a user is allowed to log in to it the CGI script will tell the simulator that this user's viewer is about to connect and sends the address of the simulator to the viewer (Second Life Wiki, 2007, [2]).

2.6.2 Circuits and Messages

In SL connections between viewers and simulators, simulators and simulators or simulators and utility servers (e.g. database servers) are called circuits, which are being provided as two-way UDP connections (Second Life Wiki, 2007, [6]).

Communication via circuits is established by the means of so called messages, which bear a strong similarity to UDP datagrams. Messages are being used to send serialised information about nearly any part of the system between hosts on the network. There are more than hundred different message types inside of SL that have all the same message format (Second Life Wiki, 2007, [7]). For further information about the format of messages and the messaging system please refer to [7].

2.6.3 Asset Transfer

Assets are requested by the viewer from an upstream provider. The appropriate upstream provider is the simulator to which a viewer is connected (Second Life Wiki, 2007, [5]).

3 Scripting

In Second Life independent, event-driven programs written in the programming language *Linden Scripting Language (LSL)* can be attached to every object in a region. For instance with an LSL script an object can become an independent agent like a fish searching for food, or an object that hands out notecards when touched. Programs can also be used to perform calculations, which are totally different from SL like SETI at home (Berkeley, 2007, [14]). As there are regions running more than 1000 scripts execution of scripts on simulators is a non-trivial problem. Section 3.1 will explain how this execution is accomplished by a simulator. Section 3.2 will explain different features of the scripting language LSL. Scripts are being developed inside of the viewer and are directly saved to the asset server (see section 2.5.3).

3.1 Microthreading with Mono

3.1.1 Microthreading

1000 scripts running in one region can be a big problem. If for instance 1000 scripts were run in separate system threads and each thread stack allocated only 32k of memory those scripts would lead a Linux operating system with the newest stack libraries to crawl (Official Linden Blog, 2007, [15]). Another question is how scripts can be migrated between two regions. If for instance a rocket flies from one region to the next the simulators that run these regions will have to migrate the script between those two regions.

SL solves this problem by employing microthreads based on .NET technology. Microthreading is injected into LSL script assemblies by means based on RAIL on top of .NET Reflection and .NET Reflection.Emit facilities. Therefore the injector looks for locations in the script assembly where the script code should yield and induces the type of stack at these locations. Now extra opcodes are inserted into the script assembly, which effect to copy the stack into a heap object and the script to yield. Afterwards another script instance can be restored from its heap object and being run. Employing this procedure many microthreads can be run within one system thread (Official Linden Blog, 2007, [15]).

This approach is efficient in terms of memory but expensive in terms of used CPU time to shift between microthreads and to insert opcode.

3.1.2 Mono

The foundation for Second Life's microthreading functionality is Mono. Mono is an Open Source software that implements the .NET platform developed by Microsoft. It includes a Common Language Interface (CLI) virtual machine. The CLI bears a Common Language Runtime (CLR), which can execute Common Intermedia Language (CIL) byte code programs. Therefore source code of programming language that should be used with a .NET platform needs to be compiled to CIL. The CLR compiles the CIL during runtime to machine code by means of a Just In Time (JIT) compiler.

Scripts written in LSL are being compiled to CIL byte code when the developer saves the code in Second Life. The asset server saves the source code as well as the byte code (see section 2.5.3). Before Mono has been introduced LSL code was executed by an LSL interpreter. According to Linden Labs LSL code that is being executed within Mono is 300-500 times faster than the former LSL interpreter. Another advantage of using .NET for script execution in SL is that every programming language to which a CIL compiler exists could be used as a programming language in SL in the future if the LSL API was transported to other languages. Additionally other APIs could be used within SL (Official Linden Blog, 2007, [15]).

3.2 LSL API

The LSL API offers language constructs for different purposes. One purpose is for instance the manipulation of size, position and rotation of a 3D object. Other purposes could be for instance interaction with avatars, chat communication within one region (see section 3.2.3) or communication with servers on the Internet (see section 3.2.4).

Whenever a new script is created in SL following default script is presented:

```
default
{
    state_entry()
    {
        llSay(0, "Hello, Avatar!");
    }

    touch_start(integer total_number)
    {
        llSay(0, "Touched.");
    }
}
```

Listing 1: LSL default script.

As LSL is state driven each script can enter different, predefined states. In this example only default is implemented. default describes the default state. state_entry() and touch_start() are event listeners. If for instance this script is put in an object everytime the script is saved state_entry() will be called because the script was started and the API function llSay() will echo out "Hello, Avatar!" on the chat console. On the other hand everytime an avatar touches the object will say "Touched." on the chat console.

3.2.1 Basics

LSL uses typed variables like float g_target_distance. LSL offers the simple data types integer, float, string, vector, rotation, key and list. rotation consists of four vectors that define a rotation. key is a type that references the UUID of an asset. list is an array like data type that offers additional manipulation possibilities (for details see [16]). However, there is no dedicated array data type.

LSL uses arithmetic operators for assignment (=), hexadecimal entries (e.g. 0xff), boolean operators (e.g. <, >, !), binary arithmetic operators (e.g. +, -, *, /, «, ») and bitwise operators (&, |, ~).

LSL offers language constructs for flow control like for, do−while, if −else, jump, return, while and state. state can be used to switch to another state. E.g.

```
state SpinState;
```

will switch to the state SpinState.

Also LSL offers the possibility to define functions e.g. the signature setMatch(string homeTextur) would define a function with the name setMatch with one argument called homeTextur of the type string.

3.2.2 States and Events

LSL employs an implicit state machine with at least one state (the default state). When a script is reset or first started, it will enter the default state. States contain event handlers, which are triggered by the LSL virtual machine (see listing 1).

The most important event handler is state_entry(). It is called whenever a state is entered e.g. by calling state SpinState; the state SpinState will be entered and the event state_entry will be triggered.

Multiple events exist for different purposes. For instance there are certain event listeners of an object that can be used ...

- ... as a sensor to trigger actions if avatars pass by within a certain distance around the object.
- ... to receive and react to emails.
- ... to receive and process data from the Internet.
- ... to react if being touched by avatars.
- ... to trigger events after a timer had run out.
- ... to perform actions after being pulled from a users inventory in world.
- ... to perform actions while moving.

- ... to react after money was paid to the object.
- ... to listen to ongoing communication on an island (e.g. to listen to the chat engine). For instance this can be used to create intelligent agents that communicate with avatars.
- ...

Please refer to [18] for more information on events.

3.2.3 In-World Communication

Inside of a region there are different means of communicating with each other. A common method is to use the chat engine. A user just enters some text in the text field of its viewer and presses enter. This text now appears in the viewers of all avatars within a certain surrounding. Additionally instant messages can be sent from one avatar to another without being visible to other users. SL employs a channel concept for communication. Communication can take place on up to 2,147,483,649 channels. Channel 0 is the public channel that is used by default from the chat engine.

Communication tasks can be carried out as well by objects with an attached LSL script. The LSL API offers several functions for this purpose:

- llSay outputs some text in the viewers of avatars within chat distance.
- llShout outputs some text in the viewers of all avatars in a region.
- llWhisper outputs some text in the viewers of all avatars within whisper distance.
- llEmail sends an email.
- llInstantMessage sends an instant message to a certain avatar.
- llListen listens to chat on a certain channel and can react on certain keywords.

Listing 2 shows a simple voting script that employs events and functions from the LSL communication API. An object in which this script is placed will start to listen to avatars that touched them at least once. The listening is triggered via llListen (0, "", toucherKey, ""). That causes the script to listen to avatars that touched it at least once on the public channel 0. Everytime the avatar with the UUID toucherKey submits a message the script figures out if it starts with "/vote". If so it will extract the text after "/vote".

```
default{
        touch_start(integer total_number){
                key toucherKey = llDetectedKey(0);
                llListen(0, "", toucherKey, "");
        }

        // user submits something like "/vote Bubo Lubitsch" to vote ...
        listen(integer channel, string name, key id, string message){
                string command = "/vote";
                integer startStringLength =  llStringLength(command);

                if(llGetSubString(message,0,startStringLength)==command){
                        string votedFor = llGetSubString(message,
                                llStringLength("/vote")+1,
                                llStringLength(message));

                        //treat extracted name...
                }
        }
}
```

Listing 2: Simple vote script in LSL.

3.2.4 Internet Communication

LSL scripts in Second Life can communicate with the Internet by the means of XML-RPC or HTTP. However, XML-RPC does communicate over HTTP as well. The approach to communicate with the Internet via LSL scripts is provided in a typical LSL manner. At first a request is sent with a request function of the LSL API or an XML-RPC event is triggered by a function call. Afterwards specific event handlers will (send requests and) receive the responses.

In listing 3 the procedure for sending XML-RPC requests as well as HTTP requests in LSL is illustrated. Whenever this script is started an XML-RPC event is triggered via llOpenRemoteDataChannel() in line 4, which causes the script to trigger the event remote_data in line 11. The type of the event triggered by this function is REMOTE_DATA_ CHANNEL (line 16), which causes the code in line 17 of the listing to send an XML-RPC request (over HTTP) via llHTTPRequest to url. A reply from url will automatically have the type REMOTE_DATA_REQUEST. In this implementation the received data would be echoed out to chat in line 22. In the case of XML-RPC in SL all requests and responses are handled by the CGI script http://xmlrpc.secondlife.com/cgi-bin/xmlrpc.cgi. Each object may only send one request at a time that is queued on xmlrpc.secondlife.com. Any additional request that is sent from this object before the preceding request was treated by xmlrpc.secondlife.com will overwrite the previous one. Additionally a response coming from an external server will be delayed for three seconds. As the performance of xmlrpc.secondlife.com has been steadily degrading XML-RPC requests can take up to 60 seconds before they are completed (LSL Wiki, 2007, [19]).

```

default{
        state_entry() {
                llOpenRemoteDataChannel();
        }
        touch_start(integer foo){
                llHTTPRequest(URL, [] ,"");
        }

        //xml rpc event handler
        remote_data(integer type,
                key channel, key message_id,
                string sender, integer ival, string sval){

                //sends request after calling llOpenRemoteDataChannel()
                if(type == REMOTE_DATA_CHANNEL){
                        llHTTPRequest(url,[HTTP_METHOD,"GET"],"");
                }

                //handles incoming answers
                if(type == REMOTE_DATA_REQUEST){
                        llSay(0,sval);
                }
        }

        //http response handler
        http_response(key request_id, integer status,
                list metadata, string body){
                llSay(0,body);
        }
}
```

Listing 3: Internet communication with LSL.

Requests must always be directed from Second Life to the Internet. It is not possible to send a request from the Internet to a facultative object in Second Life. Because of this resources coming from the Internet that should be used in Second Life need to be pulled by the objects themselves from SL. In listing 3 the responses are only echoed to chat. Usually they would be proceeded by LSL code. As LSL does not offer an API for XML processing the treatment of string data must be implemented manually. Also there is an additional 2048 byte limit for the payload of an XML-RPC request forwarded by xml-rpc.secondlife.com to the according script (Second Life Wiki, 2007, [33]). If for instance information encoded in an RSS feed should be forwarded to Second Life a sophisticated server side preprocessing of the XML information will be necessary. In this case all tags should be removed and replaced by stop codes that can be recognised by the processing LSL script to save bandwidth. E.g. <key>name</key><value>Johannes</value> could be encoded to :−:name%johannes:−:. If the data that should be conveyed is still bigger than 2048 bytes it needs to be subdivided in more XML-RPC requests. That means for every XML schema a new server side and LSL side implementation is obligatory.

4 Second Life search engine

In a virtual world consisting of more than 9.5m users and more than 8k regions sophisticated search possibilities are crucial. Especially for Second Life where virtual business with virtual goods like clothing takes place it is important that a user can find those goods. In the past there was no simple way to find a green pair of shoes in SL. At the time of this writing Second Life recently changed their search system. Now the new search tool in Second Life ranks search results according to relevance and not alphabetically or according to the fact how much traffic they generate. However, objects can only be found if they were correctly tagged in their properties (Technology Review, 2007, [21]). The new search system is provided by off the shelve Google search appliance products that were integrated in the SL server system. These appliances perform indexing and produce search results. Assets that can be found include:

- Objects (if search checkbox is checked. If an object is tagged "for sale" it is per default searchable.)
- Land Parcels (if fee of 30L$ is paid by parcel owner)

- Groups
- Avatar profiles (can be switched off for own avatar but is switched on per default)
- Events

The order of search matches is provided by the Google search algorithm, which means that they are ordered by relevance e.g. how close search terms are together (Official Linden Blog, 2007, [20]). Objects on the asset servers that are searchable are being rendered/updated to static HTML pages on world.secondlife.com twice daily (3 AM and 3 PM Pacific time). However, it takes one to three days for those updates before they are reflected in the search results (Second Life Knowledge Base, 2007, [22]). The static HTML pages are public and could be found as well on web search machines like Google. But currently public search machines are not explicitly asked to crawl them.

5 Critical Analysis

Second Life is the first virtual world with a really huge community of around 9.5m users in March 2007. The steadily growing amount of users and especially concurrently active users demands sophisticated server concepts to allow parallel activity in a region. In section 5.1 the server concepts of Second Life will be analysed with regard to scalabilty and performance. Section 5.2 dissects the opportunities for users to enhance Second Life with information retrieved from the Internet or imported contents.

5.1 Grid

5.1.1 Problems

Second Life employs a one simulator per region approach. Additionally the server performs the main part of the physical calculations (see section 2.4). Because of these two facts the amount of avatars in one region is restricted. According to the Second Life Knowledge base the amount of avatars in one region without causing a serious server lag is about 40 ([24]). 40 people in an area of 65k m2 is a fairly small amount of people especially for big events. On the other hand there will be lots of areas that are hardly used at all at the same time during others are crowded. It seems that the current Second Life Grid solves scalability issues fairly unsatisfying.

Another problem is the solution of maintenance tasks for the grid. If the simulator software changes because of security or feature releases the simulator must be halted, the new software installed and the simulator booted. As Second Life is used around the world there is no time slot in which the system is not or only scarcely used. Additionally for some simulator releases the download of a new viewer is obligatory. The size of the viewer software is around 40 MB. According to the experience of the writer of this text the download of a new viewer is necessary around every four weeks.

5.1.2 Linden Labs Future Plans

A problem of the Second Life community is the binding of regions to the proprietary software on the Second Life grid. Some users and/or companies wish to maintain their own server(s) with their own land on it. This would demand heavy changes from the current concept to enable seamless collaboration of Linden Labs own simulators with foreign ones.

According to Christian Scholz, one of the members of the Second Life Architecture Working Group ([35]), Linden Labs want to enable collaboration of simulators by providing open communication protocols for the communication between the necessary parts of the server system. However, they are not planning to publish their simulator source code under an Open Source licence because of proprietary parts like the Havok physics library in it. Nonetheless, alternative implementations of simulators could be realised like OpenSim (Open SL, 2007, [25]). The current server architecture illustrated in figure 1 is considered to be changed to the architecture illustrated in figure 3. Blue areas are Agent Stores. Agent Stores offer services like storing avatar and inventory data, login of avatars or retrieval of user data. The green areas are Region Domains, which are going to provide similar services like simulators today. Avatars can change between regions. These types of Domains and Stores offer well defined interfaces to their services. Basically a Store or a Domain can consist of any amount of computers and databases. A user logs in to a Store and can enter a Region with its viewer. Second Life will still have their own server system and user data indicated by *Second Life Agents* and *Second Life Mainland*. Additionally Linden Labs will offer central utilities like the currency system, a centralized search or a map of locations (Christian Scholz, 2007, [34]).

5.1.3 Improvement Notes

Linden Labs themselves show that they are planning to open up their system to other implementations of the server system. However, the problem of a better scalability of one or multiple simulators remains. If one simulator is idle and another suffers from lags the server system should be able to schedule simulator tasks in a way that idle simulators

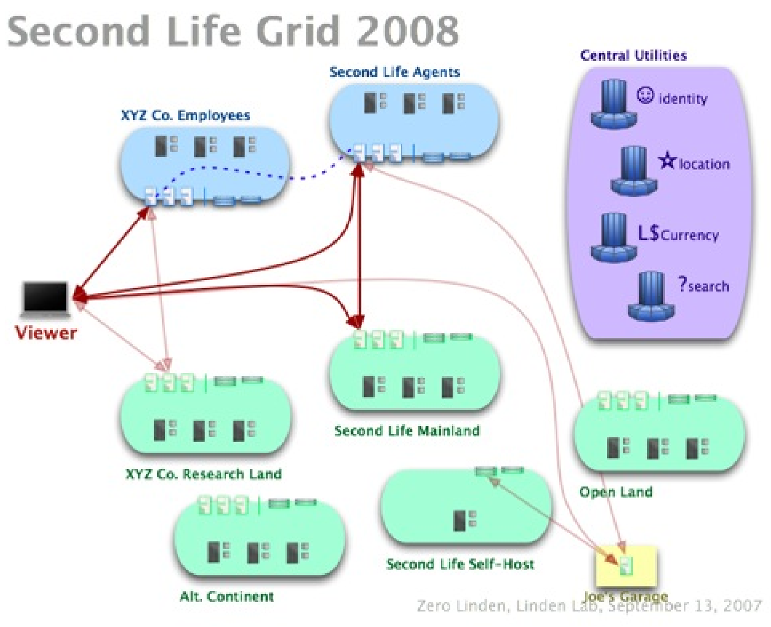

Figure 3: Future Second Life Server Structure. Source: [Figure2]

can perform tasks of the lagging simulator. E.g. if there are 100,000 users connected and 10,000 simulators available each simulator should be responsible for 10 users. But the accomplishment of physical calculations like collision detection demands multiple context information that seems to be not so easily parallelizable on different simulators/computers. Another approach could be to leave expensive calculations to the viewer. In terms of collision detection that seems to be an impossible task because of network latency. But maybe there are other calculations that can be carried out by the viewers instead of the servers. Maybe the Mono surrounding for scripting can be migrated and carried out by the viewers. Even with the planned new architecture user amounts within one region should remain fairly restricted. But maybe with a grid system and better algorithms in Region Domains the amount of parallel active users can be increased a lot. On the other hand the server system could stop expensive calculations when the amount of users exceeds a certain amount. E.g. collision detection may be not be equally important as fluid displaying of graphics and communication.

Another point are maintenance tasks for servers. It should not be obliging to shut down a simulator for an update that is necessary quite regularly. Instead it should be possible to start a new and an old instance of the simulator and to switch dynamically from the old to the new instance. Sophisticated protocols for the migration of current data between the old and the new instance would be compulsory.

5.2 Objects And Extensibility

5.2.1 Problems

In has been explained in section 2.5.1 that objects in Second Life consist of so called primitives that can be linked with a maximum count of 31 primitives per dynamic object. A problem of this approach is that existing 3D models of other file formats e.g. the obj file format ([36]) cannot be imported into Second Life. As today's engineering processes of e.g. buildings and cars employ CAD proceedings that yield perfect 3D models it is a pity that those models cannot be used inside of SL. If they should be used in SL they will have to be completely rebuild inside of SL. Additionally the amount of primitives inside a region is restricted meaning that designers of 3D objects in SL always have to bear in mind not to use too many primitives. Under these circumstances it is a fairly complex task to build complex models in Second Life. Also, not everybody may build objects in a region. The owner may decide who is allowed to build objects or to put objects into a region. As avatars can exploit permissions to use harmful or ugly objects most owners tend to restrict permissions. The result is watch-only regions.

Another issue is restricted extensibility caused by constrained communication possibilities of SL with the Internet. If for instance a developer wants to build a wallpaper that changes its texture (image) every 2 minutes then the displaying object must contain all displayed textures. There is no simple way to dynamically download images from the Internet and use them as dynamic textures in SL (Matt Biddulph, 2006, [26]). The simple reason is that Linden Labs gets 10L$ per upload of a texture. Those imported images can be observed more easily and gives Linden Labs a legal security (Dynamic textures could be easily exploited to display porn (Second Life Wiki, 2007, [27]) or spam). Another example for an extensibility difficulty is to display dynamic text information from the Internet. The whole manufacturing chain suffers from severe processing lacks. At first there is no simple possibility to display text in Second Life. A script can echo out text to the chat system. But for instance large billboards that can display text are scarce goods inside of SL. If they can display text the size of them is about 10x40 characters based on 10x40 small character textures that are set dynamically according to the contents. Additionally LSL makes it difficult to import information from the Internet because it is missing the support of XML libraries or web services directly. If a web service should be used inside of SL it is fairly difficult (see section 3.2.4).

All of these reasons lead to the impression that the current Second Life is analogue to Web 1.0. A region owner provides information like sophisticated 3D models (e.g. the

Cologne Dome [37]) that cannot be changed or commented by an average visitor. But visitors tend only to come back to a region if something is changing inside of the region or social activities take place. Thus a region owner has to spend lots of time to entertain visitors. If there was an easy way for an owner to add dynamic content to its region or easier means for user generated content it would be easier for region owners to provide interesting offers to visitors and particularly to keep visitors coming back.

5.2.2 Improvement Notes

The writer would recommendate ...

1. ...to support web services e.g. RSS feeds with easy to use LSL XML or web service APIs.
2. ...to provide billboards or OpenGL text views for dynamic text contents.
3. ...to provide an easy import mechanism for 3D file formats e.g. the obj file format and that those objects can be dynamically downloaded and used from the Internet. This would support object-focused interaction of enginering tasks. For instance engineering teams with dispersed members could meet and discuss about 3D models of their CAD projects in Second Life (see e.g. Hindmarsh et al [29]). Additionally lots of existing, sophisticated 3D models could be reused in SL.
4. ... to provide means for dynamic textures from the Internet.

If Linden Labs are afraid of legal problems with dynamic contents then the future plans to open up the server platform (see section 5.1.2) should take the legal burden from Linden Labs and put it in the responsibility of server operators. However, dynamic and user generated content is a must for Internet applications nowadays.

6 Summary and Conclusion

6.1 Summary

Second Life brought a successful virtual world to the Internet. Technically it founds on a client server model in which a Second Life viewer communicates with a server called simulator. One simulator serves one region. Avatars of users whose viewers connect with the same simulator are in the same region and can by means of the simulator interact with each other by chat or physical actions. A simulator acts as a proxy for viewers to data sources like assets or login data. Assets are data like textures, LSL scripts or landmarks that are stored in databases. Viewers communicate with simulators by the means of circuits over UDP. Simulators offer physical interaction with the help of a physical library called Havok. It also runs a Mono .NET platform implementation to execute scripts written in the scripting language LSL. Scripts are run as microthreads inside of Mono (see section 3.1). LSL itself is a state-driven scripting language that offers among others an API for 3D manipulation, In-world and Internet communication functionalities. Second Life recently employed a new searching machine concept in which assets like objects, avatar profiles, events etc. can be found by the help of an In-world search that is based on Google appliances. Each asset that is searchable is rendered into a static HTML page that is indexed by the Google appliances. In section 5.1 a critical analysis was presented that illustrated scalability problems of the current Second Life Grid implementation, presented new server concepts of Linden Labs and finally gave some improvement ideas. Section 5.2 explicated problems in Second Life like displaying of dynamic elements like text and images downloaded from the Internet and the lacking ability to use 3D file formats in Second Life. Finally some improvement notes for better extensibility are given.

6.2 Conclusion

Second Life has been groundbreaking in terms of a first impression and implementation of a potential 3D Internet. But SL cannot be called the 3D Internet as it is not free like the *real* Internet. It still is proprietary software that is owned by only one company. Nobody can tell today how Second Life will develop over the next few years. But Second Life gives an idea how a 3D Internet can be realised and yields experiences what has to be kept in mind regarding technical foundations. It seems that regions in Second Life were not designed to work with a large amount of users what can be infered from the fact that for simulators there is a max number of 40 users for a good performance of a region. Legal problems seem to hinder the integration of dynamic contents from the web.

With the success of Second Life now comes the pressure to think about a more scalable grid for Second Life. Future implementations of virtual worlds should consider the successes and the mistakes that Linden Labs made when they planned Second Life.

References

[1] Second Life Wiki, 2007, *Glossary - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Glossary> [Accessed 12. December, 2007]

[2] Second Life Wiki, 2007, *Server architecture - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Server_architecture> [Accessed 12. December, 2007]

[3] Second Life KB, 2007, *About the Heterogenous Grid*, Available from: <https://support.secondlife.com/ics/support/KBAnswer.asp?questionID=4560> [Accessed 12. December, 2007]

[4] Second Life Wiki, 2007, *Image System - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Image_System> [Accessed 12. December, 2007]

[5] Second Life Wiki, 2007, *Asset System - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Asset_System> [Accessed 12. December, 2007]

[6] Second Life Wiki, 2007, *Circuits - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Circuit> [Accessed 12. December, 2007]

[7] Second Life Wiki, 2007, *Message - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Message> [Accessed 12. December, 2007]

[8] Secondlife.com, 2007, *The Technology behind the Second Life Platform*, Available from: <http://s3.amazonaws.com/download.grid.secondlife.com/Fact_Sheet_Technology.pdf> [Accessed 12. December, 2007]

[9] Second Life Wiki, 2007, *Havok 4 Beta Home - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Havok_4_Beta_Home> [Accessed 12. December, 2007]

[10] Second Life Wiki, 2007, *What's Changed With Havok4 - Second Life Wiki*, Available from: <https://wiki.secondlife.com/wiki/What%27s_Changed_With_Havok4> [Accessed 12. December, 2007]

[11] Havok, 2007, *Havok - Home*, Available from: <http://www.havok.com> [Accessed 20. December, 2007]

[12] Havok, 2007, *Havok - Havok 5 Launches Integrated Character & Physics Solution*, Available from: <http://http://www.havok.com/content/view/538/53/> [Accessed 20. December, 2007]

[13] Wikipedia, 2007, *Second Life - Wikipedia, the free encyclopedia*, Available from: <http://en.wikipedia.org/wiki/Second_life> [Accessed 20. December, 2007]

[14] Berkeley, 2007, *SETI@home*, Available from: <http://setiathome.berkeley.edu/> [Accessed 20. December, 2007]

[15] Official Linden Blog, 2007, *Microthreading Mono - Official Linden Blog*, Available from: <http://blog.secondlife.com/2006/05/05/microthreading-mono/> [Accessed 12. December, 2007]

[16] Second Life Wiki, 2007, *Category:LSL List - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/List> [Accessed 21. December, 2007]

[17] Second Life Wiki, 2007, *State - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/State> [Accessed 27. December, 2007]

[18] Second Life Wiki, 2007, *Category:LSL Events - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Category:LSL_Events> [Accessed 27. December, 2007]

[19] LSL Wiki, 2007, *LSL Wiki : XMLRPC*, Available from: <http://www.lslwiki.net/lslwiki/wakka.php?wakka=xmlrpc> [Accessed 28. December, 2007]

[20] Second Life Wiki, 2007, *New Search Currently Under Development - Official Linden Blog*, Available from: <http://blog.secondlife.com/2007/10/19/new-search-currently-under-development/> [Accessed 12. December, 2007]

[21] Technology Review, 2007, *Better Search in Virtual Worlds*, Available from: <http://www.technologyreview.com/Infotech/19664/> [Accessed 12. December, 2007]

[22] Second Life KB, 2007, *New Search FAQ*, Available from: <https://support.secondlife.com/ics/support/KBAnswer.asp?questionID=4722> [Accessed 12. December, 2007]

[23] Second Life Forum, 2003, *Second Life 1.0 Available today!*, Available from: <http://forums.secondlife.com/showthread.php?t=3297> [Accessed 29. December, 2007]

[24] Second Life KB, 2007, *How many avatars can I have on my region at once without causing serious problems with lag?*, Available from: <https://support.secondlife.com/ics/support/KBAnswer.asp?questionID=4426> [Accessed 12. December, 2007]

[25] Open SL, 2007, *Main Page - OpenSim*, Available from: <http://opensimulator.org/wiki/Main_Page> [Accessed 29. December, 2007]

[26] Matt Biddulph, 2006, *Alas, Second Life! Web 2.0 in a virtual world*, Available from: <http://www.hackdiary.com/archives/000085.html> [Accessed 30. December, 2007]

[27] Second Life Wiki, 2007, *Web Textures - Second Life Wiki*, Available from: <https://wiki.secondlife.com/wiki/Web_Textures> [Accessed 12. December, 2007]

[28] Second Life Wiki, 2007, *Object - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Object> [Accessed 12. December, 2007]

[29] Hindmarsh, J., Fraser, M., Heath, C., Benford, S., Greenhalgh, C. 2000. *Object-Focused Interaction in Collaborative Virtual Environment*, In ACM Transactions on Computer-Human-Interaction, Vol. 7, No. 4, December 2000, Pages 477-509

[30] Second Life Wiki, 2007, *Primitive - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Primitive> [Accessed 12. December, 2007]

[31] Second Life Wiki, 2007, *Source downloads - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Source_downloads> [Accessed 30. December, 2007]

[32] Second Life Wiki, 2007, *UUID - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/UUID> [Accessed 31. December, 2007]

[33] Second Life Wiki, 2007, *User:Zero Linden/Office Hours/Discussion - Second Life Wiki*, Available from: <https://wiki.secondlife.com/wiki/User:Zero_Linden/Office_Hours/Discussion> [Accessed 31. December, 2007]

[34] Christian Scholz, 2007, *Linden Lab diskutiert die neue Second Life Architektur - mr-topf.de*, Available from: <http://mrtopf.de/blog/secondlife/linden-lab-diskutiert-die-neue-second-life-architektur/> [Accessed 31. December, 2007]

[35] Second Life Wiki, 2007, *Architecture Working Group - Second Life Wiki*, Available from: <http://wiki.secondlife.com/wiki/Architecture_Working_Group> [Accessed 31. December, 2007]

[36] Florida State University, 2007, *Object Files (.obj)*, Available from: <http://www.csit.fsu.edu/~burkardt/txt/obj_format.txt> [Accessed 31. December, 2007]

[37] Grid Grind, 2007, *Historical Cologne Cathedral Goes Virtual*, Available from: <http://www.gridgrind.com/?p=177> [Accessed 31. December, 2007]

Figures:

[Figure1] Copied from: <http://wiki.secondlife.com/w/images/b/be/SLGArchWG1-01-Grid_Now.jpg> [Accessed 12. December, 2007]

[Figure2] Copied from: <http://wiki.secondlife.com/w/images/b/be/SLGArchWG1-24-SL_Grid_2008.jpg> [Accessed 12. December, 2007]