Support for Mobile Augmented and Synthesized Worlds

December 17, 2007

Won J. Jeon and Roy H. Campbell
{wonjeon, rhc}@uiuc.edu
Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract

Virtual worlds provide 3D-immersive experiences to users and some of them already have already launched commercial service to users. As computing environment becomes more heterogeneous, more mobile users are anticipated to access the virtual world with their mobile devices. However, still there are challenges and problems to be addressed for mobile users. In this report, state-of-art virtual world platforms are presented and their key features are compared. We compare possible approaches to tackle these problems to support virtual worlds for mobile devices. Transcoding scheme at the proxy is presented and evaluated for a given computing and networking environment.

1. Introduction

A virtual world is a computer-simulated or synthesized environment where multiple users inhabit and interact to each other via avatars. The world mimics the real world via simulated real world physics and the persistence of the world comes from maintaining and updating the state of the world around the clock. Historically this concept is rooted from distributed interactive simulation (DIS) and massively multiplayer online role-playing game (MMORPG). DIS is used mainly by military organizations whereas MMORPG has gained huge popularity among general users or gamers.

The world is typically represented as 2D or 3D graphics to multiple users. Recent development of computer graphics technology enables real-time rendering of complex 3D scenes for immersive experience to users. MMORPGs have similar concepts but virtual worlds generally provide the platform for more diverse purposes including games, social networking, e-commerce, training, education, and collaboration.

As computing environment becomes more heterogeneous, more mobile users are anticipated to access the virtual world using their mobile devices via different types of wireless networks. However, current mobile systems do not fully match the system requirements needed for the virtual world applications that are currently designed for desktop computers, so providing immersive experiences to mobile users are challenging. Typically they do not have fast computing processors and graphics hardware to support real-time 3D rendering, enough memory space for run-time processing and caching for

information, or abundant network bandwidth to exchange the information to the world servers.

There would be different approaches to overcome such limitations due to resource scarcity of mobile devices and to provide the seamless virtual world experience to mobile devices no matter what kind of devices that users use. In section 2, we would like to compare key features and architectures of different virtual worlds that are currently available in open source community. In section 3, we discuss possible approaches for supporting virtual worlds for mobile devices. Among those approaches, transcoding of 3D to 2D to mobile devices for Second Life is implemented is evaluated in section 4. Finally, we conclude this report in section 5.

2. Virtual worlds

[Singhal99] and [Matijasevic02] summarized historical aspect of networked virtual environments and compared different systems developed in 80s and 90s in detail. SIMNET [Muller95], DIS (Distributed Interactive Simulation) [DIS1993], NPSNET (Naval Postgraduate School Networked Vehicle Simulator) [Macedonia1994], DIVE (Distributed Interactive Virtual Environment) [Hagsand96], CAVE [Cruz-Neira92] are good examples and mostly they were developed in research community and military organizations for collaboration and training.

Due to development of fast network and graphics hardware, recently developed virtual world systems enable the support for massive multi-users and rich multimedia contents including 2D/3D graphics as well as image, video, and audio. Some of them such as Second Life [SecondLife], There [There] have already launched their service to users and gradually gained commercial success, and the others such as OpenSim [OpenSim], Darkstar/Wonderland [Darkstar][Wonderland], Croquet [Croquet], OLIVE [OLIVE], Uni-verse [Uni-verse], MUPPETS [MUPPETS] sell or provide freely their solutions for the developers. Unlike the earlier virtual world systems and other 3D networked games, they target more general purposed architectures and try to support massive number of users.

2.1 Second Life / OpenSim

Overview

Second Life (SL) is the most popular virtual world service currently available. More than 11 million users are registered and more than 16 million financial transactions are made in a month [SLeconomics]. Currently, the client viewer code is open to public, whereas the server code is still in proprietary of Linden Lab. Recently there is an open source community to implement their own version of virtual server to operate with the official client viewer of Second Life.

Architecture

SL server consists of different server components including *Login server*, *Spaceserver*, *Dataserver*, and *Simulator*. The whole world is divided into grids, and a designated server manages the objects and interactions of them in a particular grid. Spaceserver handles the routing of messages based on grid locations. Dataserver passes queries to the database on behalf of the simulator. Simulator is the primary SL process. Each simulator simulates one 256x256 meter region. When an avatar moves, it is handed off from one simulator to the other via inter-process communication. It handles storing object state, land parcel state, and terrain height-map state. Also it performs visibility calculations on objects, land, and terrains to clients.

World representation

A 3D object is represented as a primitive such as box, cylinder, prism, sphere, torus, tube, ring, and sculpted, and multiple 3D objects can be grouped together in hierarchical manner. This representation is used for the entire lifetime of the object from creation from users, transmission between client and server, and representation in server's database. The texture information is mapped to the object in JPEG2000 format, which allows progressive transmission based on the level of details.

2.2 Project Wonderland & Darkstar

Overview

Project Wonderland is a client-server software platform developed by Sun Microsystems, for creating 3D virtual worlds, specifically for collaboration among users who are geographically distributed. It deploys Project Darkstar as a server-side platform that is originally designed for multi-player gaming technology. Project Darkstar now manages a collection of objects and provide APIs for update of the state of the objects in response to the events generated from multiple users. Project Wonderland & Darkstar are designed and developed in Java based on Java object-oriented components that are already developed by Sun Microsystems. For instance, 3D rendering on client is based on Java3D and Looking Glass (LG). LG is designed for embedding legacy 2D applications to 3D user interfaces. Audio communication among users is supported by jVoiceBridge framework [jVoiceBridge].

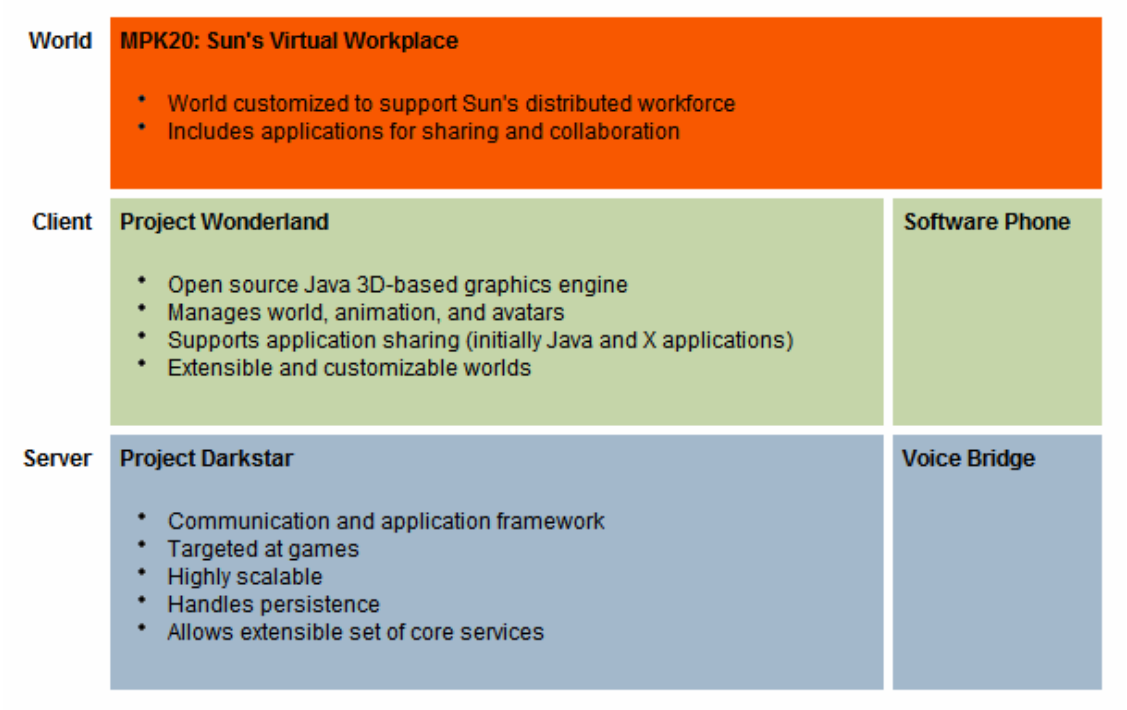

Figure 2-1 Wonderland/Darkstar summary [MPK20]

Figure 2-1 shows the relationship between Project Darkstar and Wonderland. Project MPK20 is an exemplary virtual world developed by Sun Microsystems, which is the mainly targeting for file sharing and business collaboration.

World representation

In Wonderland/Darkstar project, the world is defined by a group of *Cells* which are controlled by the server. The Cell is used to represent a managed object. As the avatar moves around the world, the server periodically informs the client which cells to load and which cells to unload. There are two types of cells in Wonderland - stationary and moveable. Stationary cells represent regions that do not move, while moveable cells represent graphic objects which do move, such as avatars. Each cell's position is defined by its origin and the physical extent of the cell is defined as its bounds. Each cell can have zero or more children and parents and they are grouped in a hierarchical structure.

The set of cells currently kept in client's memory is also managed by the server. This server-side cell is cached when the state of the cell changes – cell created, cell moved, cell deleted, and add parent to cell. The position of an avatar is used to determine the visibility of cells to a particular user. Visible cells are also cached on client especially for client boot-time.

Contents in a virtual world can be added in two ways – through static art files or programmatically using Java3D code. Currently Wonderland uses X3D data interchange format for loading content data into the world. Once imported, all data in Wonderland is

stored in the Java 3D SceneGraphIO binary format. Maya, Blender, Wings 3D, Sketchup, 3D Studio Max, Lightwave, Art of Illusion are the tools that can be used for importing contents to the world.

Client-server communication

The basic communication is based on client-server architecture. When a client connects to the server, a direct client-server communication link is established by Darkstar. The server also provides the publish/subscribe communication channels between clients and between a client and its server, which simplifies the programming tasks of client-server and P2P messaging for application developers.

As mentioned earlier, Darkstar is used for server-side solution for Sun's virtual world platform. As in Figure 2-2, Wonderland server-side code sits on top of Darkstar stack, meaning that actually Darkstar server can host multiple game servers other than Wonderland game server.

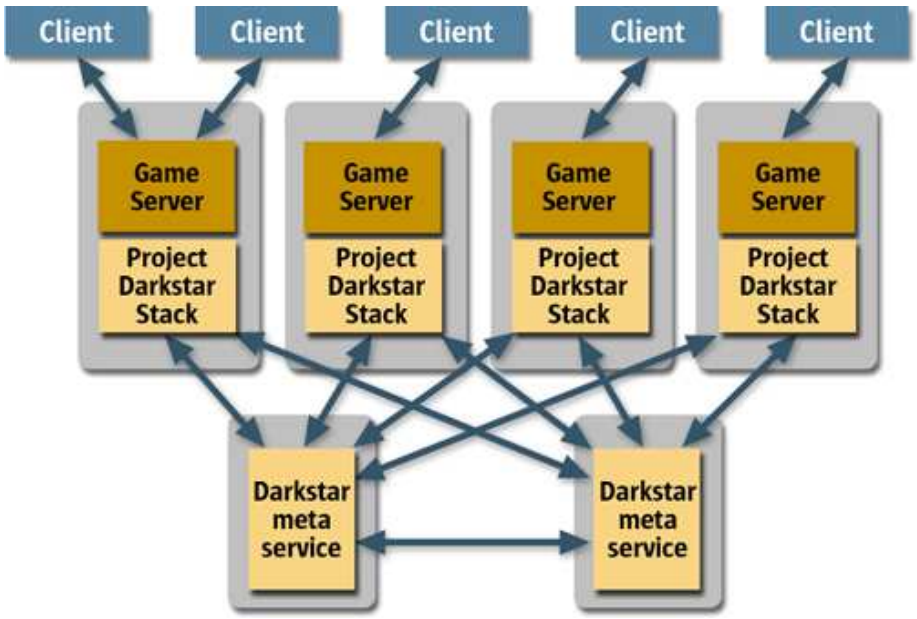

Figure 2-2 Architecture of Darkstar game server [Heiss07]

Project Darkstar and Wonderland provide a software platform for game developers and they are still on-going projects, so some of features are either not well defined or out of scope of these projects. For example, it is not clear to see how the collision among moveable cells is detected. Also, embedding user-defined scripts similarly to Second Life does not seem to be considered at this moment. Along with Project Darkstar, there are another open source projects called jMonkeyEngine [jMonkey] and ODENetworking [ODENetworking] for 3D gaming applications. jMonkeyEngine is another scene graph API and ODENetworking API provides server synchronization for gaming clients.

2.3 Croquet

Overview

Croquet is an open source software development environment for collaborative application for multiple users on multiple operating systems and devices, developed by Croquet Consortium. It is derived from Squeak, which is an integrated software development environment for live software construction using Smalltalk object-oriented programming language. The Squeak provides a single development and run-time environment, so restarting the application and recreating the application state can be made dynamically.

TeaTime and *Island* are the basis for Croquet's replicated computation and synchronization. An Island is the unit of replication and maintains all the states on different machines connected by a network. The communication among objects is based on the synchronized message passing model. The Island's view time is defined by the order of messages in its internal queue.

Croquet inherently has a security mechanism for interaction among objects. The message passing is allowed only inside of the Island and also ensures the consistency between machines. Objects inside an Island maintain direct links to send messages to each other or themselves. They cannot send messages outside the Island, nor can object outside the Island send messages directly to the objects inside. A TFarRef is an object that exists outside of the Island, but can act as a proxy for an object that is actually inside the Island for exchanging messages.

Architecture

The Croquet is based on the Squeak software development platform. The Squeak is based on Smalltalk and it provides a single development and run-time environment, which means that changes in the world can be made dynamically, without needing to restart applications or recreating the application state. Croquet Harness manages the creation of Croquet islands located in different machines. Also it manages the user input and rendering output to the user's screen.

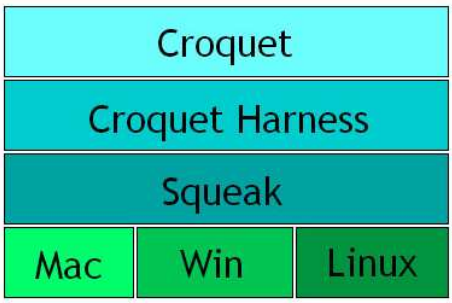

Figure 2-3 Croquet layered architecture [CroquetProgramming]

World representation

The Croquet rendering engine is built on OpenGL and programmers can always use OpenGL commands in programming his/her applications. When a new 3D object is created based on TObject class, the instance of the object is shared with other users. Currently Croquet has different 3D models to be imported from various graphics modeling tools such as 3DS Max, ALICE, Wavefront, VRML97, etc.

Importing 2D applications to 3D

Croquet also provides the mechanism to embed existing, non-collaborative, 2D applications to the 3D collaborative virtual world. The Croquet SDK includes a class called *TembeddedableApp* which can be used to define an application within the framework of Croquet. The *TembeddableApp* runs in *Squeak*, and Squeak provides many interfaces to the web, to the operating system, and so on.

Currently there are two types of implementation of sharing legacy applications in 3D virtual world in Croquet. One is called KRFB[1] shown in KAT demo. KRFB is a subclass of *TembeddedableApp* texture. The associated instance of morphing texture (*TmorphicWorld*) of a given *TembeddableApp* has a single client embedded in it, connected to a shared VNC (Virtual Networking Computing) session on some server. The other is called XRFB. The texture of XRFB can be given a URL to access. This is a similar way to how the media object is associated in Second Life.

2.4 OLIVE (On-Line Interactive Virtual Environment)

Overview

The OLIVE platform by Forttera Inc. is a suite of applications, tools and interfaces that enable non-programmers to create application-specific content and scenario, as well as enabling programmers to develop customized virtual world applications. The OLIVE core provides the foundation for a virtual world application and ensures the consisting environment across network and computing infrastructure. The extensible set of interfaces is defined as OLIVE APIs so that they can be used by third party content creators and plug-ins programmers.

Architecture

Similar to other virtual world platforms, OLIVE core has multiple server components such as Simulation Server, Application Server, Database Server, etc. They communicate to the other servers via Communication Server. Cluster Server can be deployed in enterprise-level OLIVE applications for the management of cluster configuration.

1 RFB stands for Remote Frame Buffer protocol which is used for Virtual Network Computing (VNC).

World Representation

Since OLIVE is not an open source software development platform for virtual worlds, the details of world representation and data model is not open to public. The world in OLIVE is represented based on OLIVE Application Object Model. It is a XML-based data binding describing object, their data, their interactions and relations to other parts of the system.

Similar to other virtual world platforms, OLIVE allows users to import various mesh-based 3D modeling formats from different graphics applications. It supports large area, multi-resolution terrain that can be developed using Autodesk 3ds Max for small areas and Terrex's Terra Vista for large areas. With a set of plug-ins, OLIVE allows to import photo-realistic human faces for avatars.

One of the advantages of OLIVE over the other virtual world platforms is that OLIVE supports an optional Distributed Interactive Simulation (DIS)/High Level Architecture (HLA) gateway to support integration with existing simulations and simulators. The gateway connects to an existing simulation network and establishes the necessary connections with OLIVE simulation servers, to proxy objects for the remote simulations within the OLIVE virtual world.

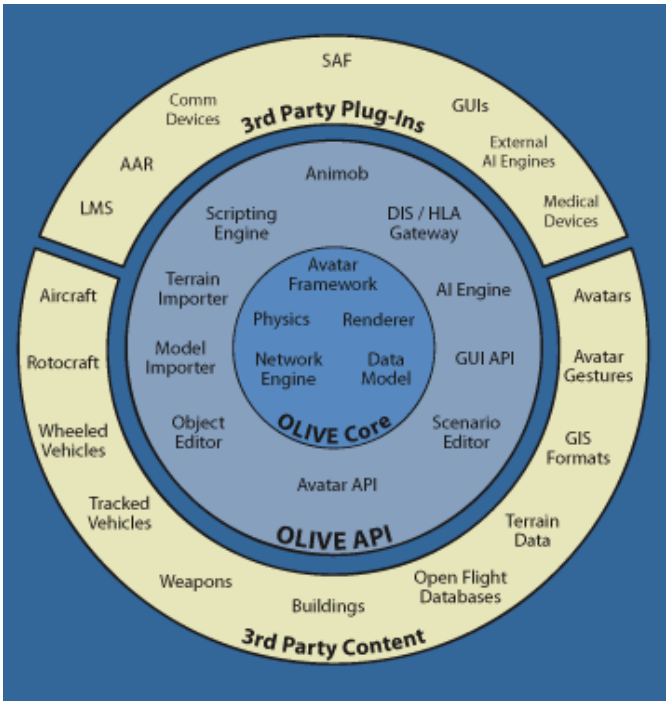

Figure 2-4 OLIVE architecture [OLIVEtech]

2.5 Uni-verse

Uni-verse is a European collaborative project for virtual worlds. It mainly focuses on the protocols for world modeling, 3D audio acoustics simulator, and 3D graphics engine.

World representation

Different from Second Life, the 3D geographical data is based on subdivided polygons. All data in Uni-verse is organized in hierarchical nodes using OpenSG [OpenSG]. Uni-verse server maintains the original representation of 3D data objects, whereas the client can render the part of the objects based on different subdivision algorithms. Currently, Catmull Clark subdivision surface algorithm is implemented and integrated in the rendering engine.

2.6 Other applications using 3D interface

In order for the virtual worlds to provide a certain level of immersive experience to users, underlying system support from hardware and software on clients is necessary. Especially high-speed network and fast computing processor are the most crucial hardware components in order for the client to keep up the persistence of virtual worlds in real-time and render it to its local display.

There are different types of applications using 3D graphical user interface for their purposes and they can be divided into the following categories.

MMORPG (Massive Multi-player Online Role-Playing Game)

Most MMORPGs are deployed using client-server architecture. Each user controls its character and interacts with others. Events are generated from actions from clients and interactions among the clients, and propagated in real-time to the server as well as the other clients that share the same world. Since the delayed delivery of such events to clients and slow feedback like the delayed rendering to its display are frustrating to gamers, high-speed network with lower network delay and fast CPU and graphics engine for faster 3D rendering at the client are usually required.

Scientific visualization

3D rendering based on complex physical objects such as molecules in microscopic scale requires huge amount of computation for visualization, so typically geometric computation and rendering is done at the server side upon client's request and only the rendered image is transmitted to the client. Dynamic interactions such as change in viewpoint and viewing distance from clients are not provided or provided in a given scenario. The server maintains the "pipelines" of consecutive 3D-rendered images for simulating the change of viewpoint or distance from the information of the client, and streams them to the client.

GIS applications

Traditional GIS applications now have client-server architecture in wide area networks, so they also have schemes for 3D representations for transmissions over the network. Recently some of GIS applications such as Google Earth allow users to embed 3D objects to their tiled map. The tiled map is organized as texture pipelines in different levels of details and coordinates of geographical locations. When a user interacts with the map by moving the interesting point or focusing the particular location, the server asynchronously streams the textures of the particular map. The details of how Google Earth work is described in [GoogleHow].

Embedding 2D legacy applications to 3D environments

There have been efforts to reuse 2D legacy applications to 3D computing environments without any change in application code. Typically the visual representation from the application is captured by the underlying display server and translates it appropriately for 3D visual environments.

Sun Microsystems' Looking Glass (LG3D) is an open source project to provide richer desktop experience to users through 3D windowing and visualization capabilities. It is based on the existing Java technology including Java 3D API and provides Java APIs for developing LookingGlass-aware 3D applications to developers, while C++ bindings will be also provided in the future for native applications.

LG3D also supports existing 2D applications in a 3D space. Figure 2-5 shows the high level diagram of LG3D. For the integration with legacy X applications, LG3D captures the client's visual representation and sends it to the LG3D Display Server. The rendering of the 3D space is managed by the *DisplayServer* and integrated into the X.org's X server. For Looking Glass-aware applications, Java APIs to LG3D library is provided and the library communicates the Display Server for 3D visuals and events.

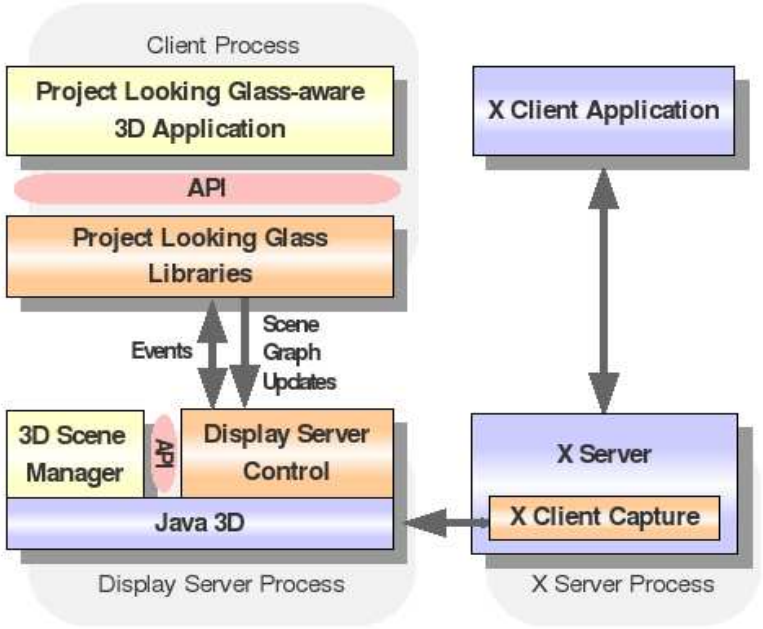

Figure 2-5 LG3D architecture [LookingGlass]

The Task Gallery is a research prototype of 3D user interface that expands the desktop into an entire office, which allows users to move quickly and easily from one task to another with a simple mouse and keyboard command.

Since the system does not assume any change, rewriting, or recompilation of existing applications, they implement their redirection mechanism for both output and input. Output redirection renders the screen output of the application to off-screen bitmaps and provides them as a texture in the 3D environment. Input redirection captures the mouse and keyboard events received by the application, and the mouse coordinate is translated from 3D to 2D. The underlying components of the window manager and graphical subsystem in the driver were modified to support legacy applications without any recompilation. Figure 2-6 shows the redirection of output and input in Task Gallery.

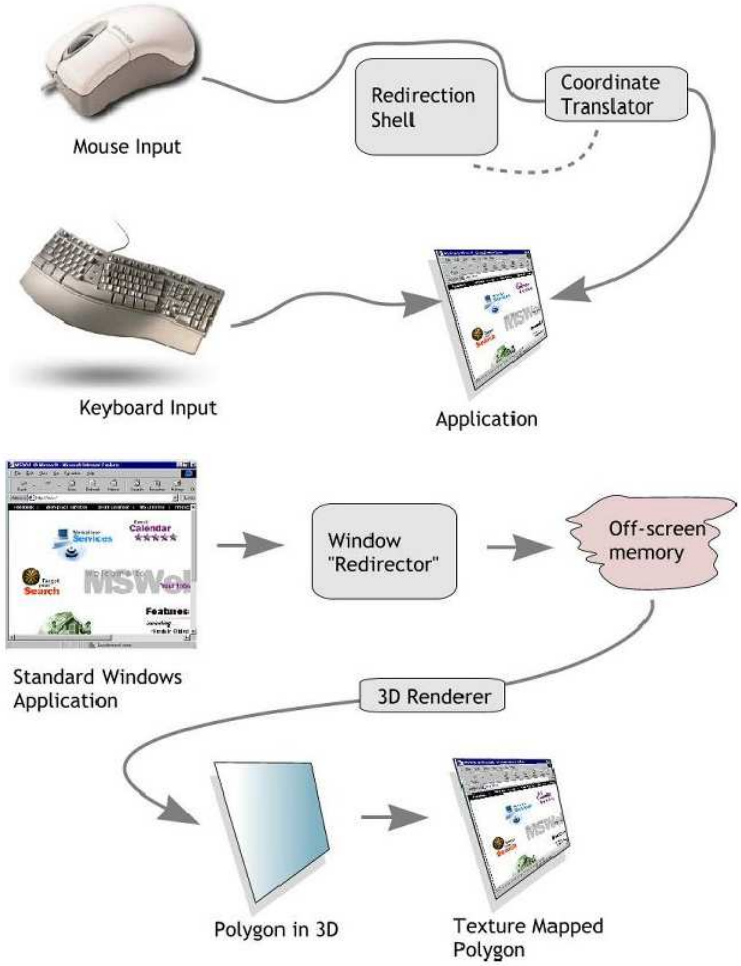

Figure 2-6 Redirections of output and input in Task Gallery [TaskGallery]

3. Virtual Worlds Support for mobile devices

We address the challenges and problem to be addressed to support virtual world application for mobile devices. Different approaches based on the resource availability on the client devices are presented.

3.1 Challenges of 3D applications on mobile devices

Currently popular 3D applications such Second Life, Google Earth, and Microsoft Virtual Earth have client-server architecture. Usually clustered-servers store the information of geographical/terrain and 3D objects on their storage space and send them to clients upon their request. Rendering of the objects and viewpoint change is performed on client side in order to offer instant visual feedback to users. Therefore, these applications require a very high-end client desktop computer with high-speed network, fast computing processor, abundant memory and storage, and 3D graphics accelerators. However, current

specifications of mobile devices including smart phones and Internet tablets do not fully match the requirements for such applications. The characteristics of computing resources of mobile devices and the effects due to short of system requirements are explained as follows.

Low network bandwidth

Whereas rich client desktops have usually wired network connection such as DSL, cable network, (Gigabit-) Ethernet, with the bandwidth of ranging from 256kbps to 100Mbps, mobile devices based on cellular and broadband networks have limited network bandwidth. For example, the current data service based on CDMA and EV-DO networks only supports up to 144kbps and 2.4Mbps, respectively, which is practically slower than 802.11b/g wireless networks.

Slow computing processor and lack of graphics hardware

Typically the 3D applications mentioned above require at least 800 MHz or faster computer processor and recommends 1.5GHz or faster processors. In addition, they require a 3D hardware accelerator on video/graphics card for real-time rendering at the client. However, most embedded processors based on the ARM-based core on smart phones are running at 500 MHz or slower, and most of them either do not have hardware 3D accelerator or software components on them do not fully support for the 3D rendering.

Low memory and storage capacity

3D objects are usually represented in meshes or primitives with texture information embedded. They are transmitted from server to client and the 3D clients re-construct 3D objects and calculate the appropriate visible portion of the objects based on client's viewpoint and camera location. These reconstruction and rendering tasks require a good amount of run-time memory on clients, so typically 3D applications requires more than a couple of hundred Mbytes. Moreover, in order to reduce the latency to retrieve 3D objects from server, clients usually pre-fetch and cache them in their secondary storage space. However, mobile devices typically have limited primary and secondary storage space in solid-state memory for run-time execution and object caching.

Limited and small display

Typically smart phones and Internet tablets have smaller 2D displays with the size of WVGA (800x480) or QVGA (320x240) and with lower color depth than desktop computers. Therefore, even they receive relatively complex 3D objects from the server, they do not have an ability to precisely display 3D objects in detail in such small displays from user's perspective, which make hard to create immersive experiences to users.

3.2 Possible approaches

Depending on the resource availability on the client device, there are a couple of possible approaches to enable 3D experience to users[2].

3D rendering on the client

This is the case where the client device has similar level of resource availability as the desktop that can run the off-the-shelf 3D applications such as Second Life or Google Earth. Typically UMPCs (Ultra Mobile PC) meet those system requirements for the applications.

Filtering at the proxy

If the client does not have full capability of rendering 3D objects on its display, the intermediate node, proxy can unburden the load of 3D rendering by filtering out some of unnecessary 3D objects to a particular user. The determination of which objects should be kept or filtered out could be based on different parameters, such as viewing distance, viewing angle, display size, etc. The hierarchical representation of scene description and the levels of details on 3D objects are captured and modified by the proxy to filter out unnecessary information.

In Second Life, each sub-object in a particular object has a weak notion of hierarchical relationship, which means that the server does not have a full hierarchical description of the object. Instead, it only maintains the parent-child relationship for each sub-object. Similarly, when the server sends a sub-object in an object, it only sends the parent-child relationship of the particular sub-object to the object. One disappointing fact of this representation is that there is no relation between the hierarchical relationship of objects and visual representation such as level of details, so it is not quote obvious for the proxy to determine which object is more important than the other in visual context.

3D to 2D transcoding at the proxy

In this case, the proxy renders the 3D image on behalf of the client and sends the 2D images to the client. This scheme is suitable for the client that does not have 3D graphics rendering capability or low rendering capability. The image can be compressed in order to reduce the amount of data transmitted over the network.

4. Experiments

Among the possible approaches mentioned in the previous section, 3D to 2D transcoding scheme was implemented and tested with proxy and client. The proxy captures bitmap

[2] Different approaches could be mixed together to enhance the overall usability of 3D applications. For example, the client viewer could have 3D rendering for avatars only, whereas the 3D objects on the background can be transcoded to 2D image textures.

images from OpenGL rendering buffer when it renders to its display, compresses them with JPEG library, and transmits them to the client over TCP/IP. While receiving frames from the proxy, the client decompresses and displays them periodically in every 200 milliseconds[3]. At the same time, when the user moves his/her avatar by pressing the navigation keys on the client device, the key presses are transmitted back to the proxy, stored to the event queues at the proxy, and eventually transmitted to the server.

The official version of Second Life client viewer by Linden Lab was used for implementing the proxy with Microsoft Visual Studio 2005. The client was implemented in C and GTK+ graphic library using Maemo SDK [Maemo] and GCC tool-chains for ARM processors. The network and display module on the client were implemented as separate threads. The JPEG codec (jpeglib version 6b [JPEG]) developed by Independent JPEG Group was integrated with the proxy and client code.

4.1 Setup

Table 4-1 shows the details of hardware and software configurations of both proxy and client.

	Proxy	Client
Manufacturer	Dell	Nokia
Model	Inspiron 8600	N800
Operating system	Windows XP Service Pack 2	Internet Tablet OS 2007
CPU	Intel Pentium M 1.7GHz	TI 1710 OMAP 250MHz
Memory	1GB	128MB
Graphics	NVIDIA GEForce FX Go5200	Imagination Technologies PowerVR MBX graphics coprocessor
Display	-	800x480 (16-bit color depth)
Network	802.11b/g	802.3u Fast Ethernet

Table 4-1 Hardware and software configurations

4.2 Scenario

In order to measure the performance of proxy and client, a user navigated in a particular region (“Hanja” and its neighbor regions in the experiment) using four direction keys on the client device for about two to three minutes. Figure 4-1 shows the typical view of the region that the avatar navigated. This region has relatively delicate modeling in architectures and grounds, so it is quite popular to users, as seen in the screenshot.

[3] The period of display was determined by the measurement of the network bandwidth from bulk data transfer. The client was connected to the security-enabled wireless network (dcs-wpa). For the given network configuration, the maximum network throughput that the client could achieve from bulk data transfer was about 1.2Mbps.

Figure 4-1 Screenshot of "Hanja" region in Second Life

With this scenario, two different image sizes, 800x480 (WVGA) and 320x240 (QVGA), and two different JPEG compression qualities (25 and 50) were used for the measurement. The quality of JPEG compression is the number between 1 and 100. If it is closer to 1, the best compression ratio is achieved, whereas if it is closer to 100, the best picture quality is achieved[4].

4.3 Results

Table 4-2 shows the summary of the experimental results in four different system configurations. When each image frame was sent from the proxy to the client, the frame size and the duration of the transfer were recorded on the client and the proxy. The frame rates were calculated by dividing the sum of size of frames by the number of frames transmitted from the proxy to the client during the entire experiment. The compression ratio is the size ratio of the compressed frame to the original RGB raw image with the same display size. During the experiment, the network bandwidth between server and proxy was about 200-300 kbps, depending on the complexity of the scenes.

[4] Even though there is a difference in terms of frame size with different JPEG compression quality, 25 and 50, the visual difference was hardly noticeable on such a small display of N800. Both qualities look quite acceptable visually for gaming or virtual world applications.

	Frame Rate (FPS)		Frame Size (Bytes)	Compression Ratio
	At Client	At Proxy		
320x240 (25)	4.48	4.54	3891.2	0.017
320x240 (50)	5.77	5.70	6171.1	0.027
800x480 (25)	3.16	3.07	22093.2	0.019
800x480 (50)	2.96	2.98	26780.9	0.023

Table 4-2 Summary of experimental results

Figure 4-2 shows how the frame size changes over time for different system configurations. Note that four different experiments were performed separately, so each of them has different starting time of the experiment. Generally, the frame size looks proportional to the display size and compression quality.

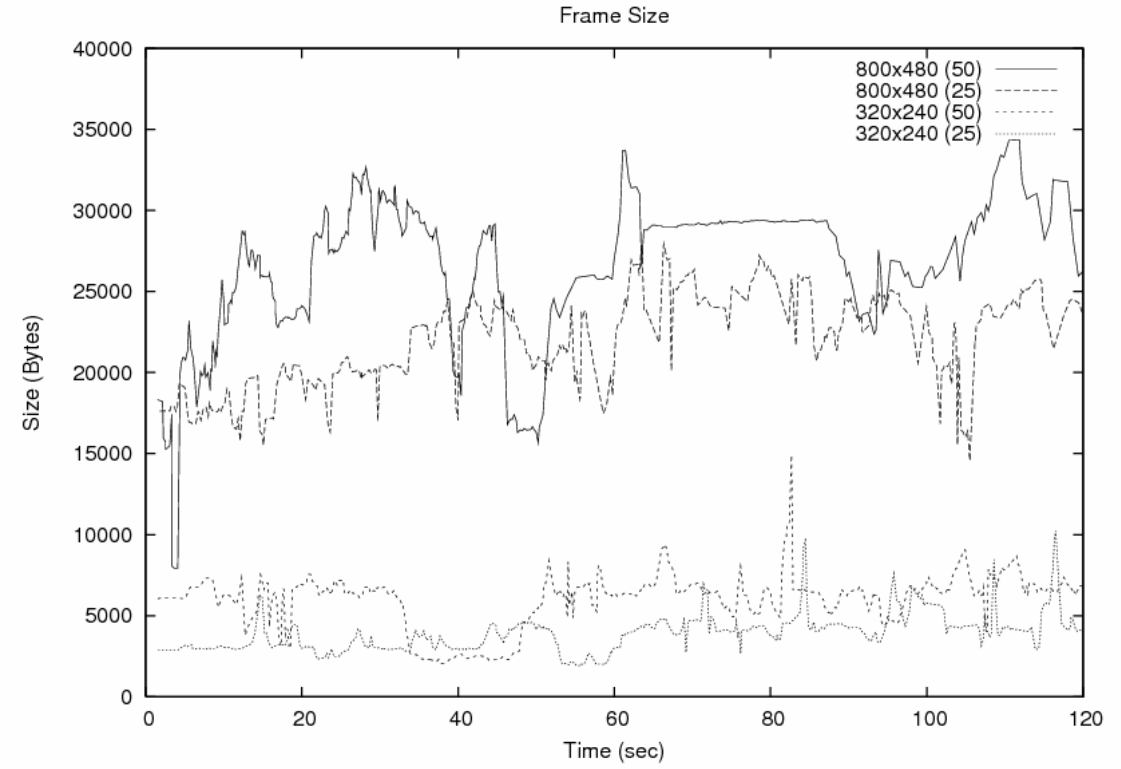

Figure 4-2 Frame size changes over time

Figure 4-3 and 4-4 show the instant frame rates measured at the client and proxy. The instant frame rate was calculated by dividing the size of the transmitted frame by the time interval between two consecutive frames.

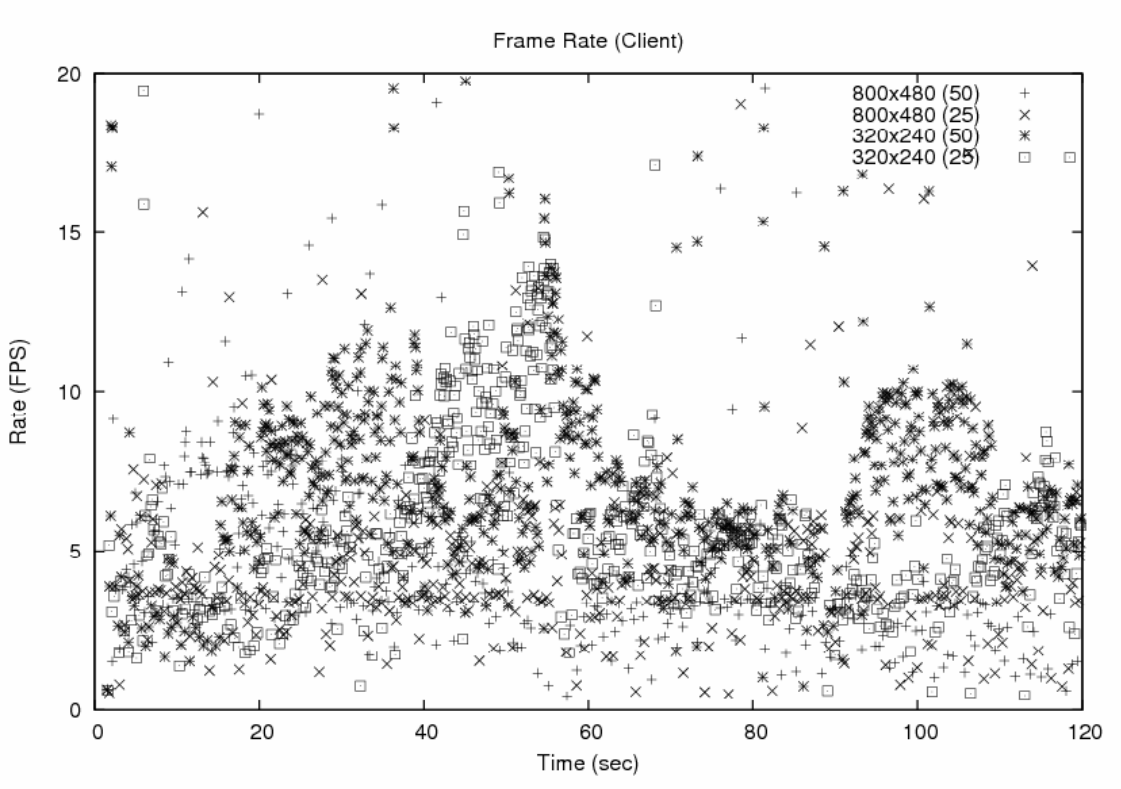

Figure 4-3 Instant frame rates at client over time

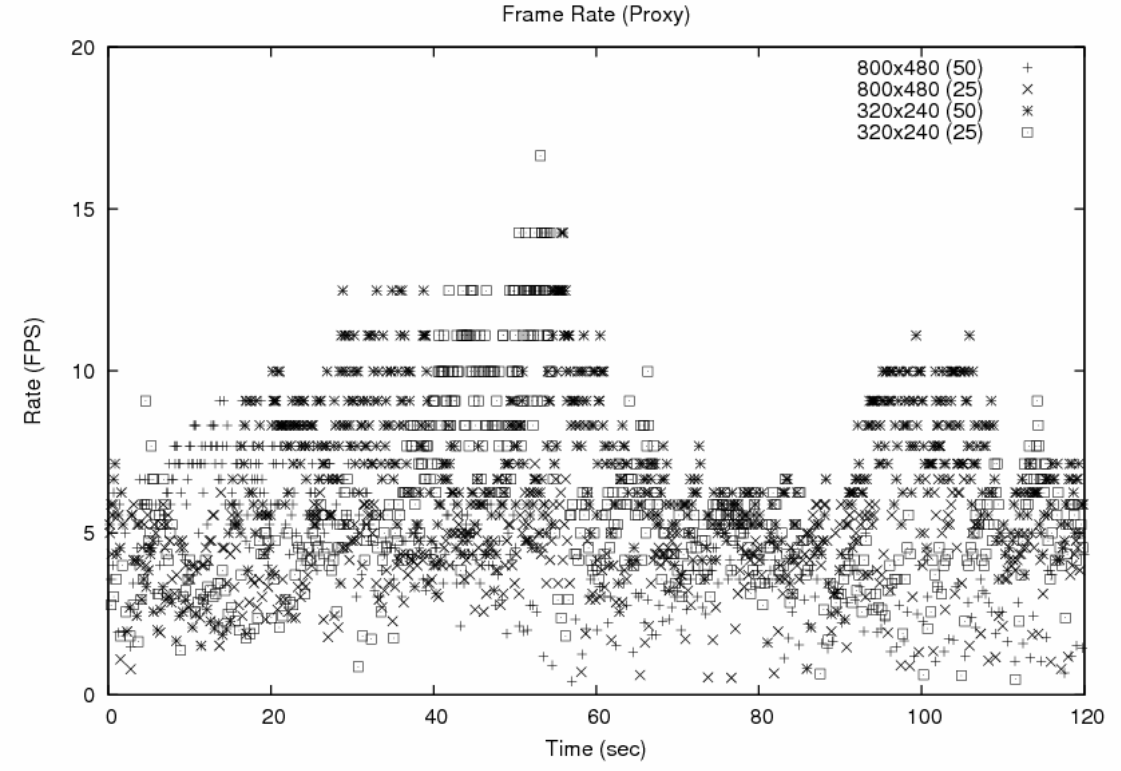

Figure 4-4 Instant frame rates at proxy over time

4.4 Discussions

From Table 4-1, the frame rates measured on the client is quite similar to that measured on the proxy, which means that the maximum frame rate that the client can achieve is determined by how fast the proxy can render, compress, and send frame data to the client. It took more time for the proxy to handle larger image size (800x480) than the smaller size (320x240). We confirmed that the maximum throughput for the given network configuration was about 1.4Mbps in application layer, but we could get about 636kbps with the large display size (800x480) and high compression rate (50), which means that the network bandwidth between proxy and client is not fully utilized. Therefore, the bottleneck to achieve higher frame rate on the client seems to be the 3D rendering capability on the proxy. Different JPEG quality parameters for a given image size does not seem to directly affect the frame rate that the client can achieve.

Also, when the proxy handles the larger frame size (800x480), it captures the frames in less frequency, so from the user's perspective, the rendering of the frames looks a little jumpy on the client display. The rendering of the smaller frame (320x240) on the other hand was quite smooth, quite comparable to that on the desktop client viewer. In interaction point of view, the user could get less than 1 second of the visual feedback for most of time, but there were some variances. The time of the visual feedback to the user is the time between when he/she presses the navigation keys on the client device and the time that he/she notices the movement of his/her avatar on the display.

5. Conclusions

Virtual worlds provide 3D-immersive experiences to users and some of them already have already launched commercial service to users. As computing environment becomes more heterogeneous, more mobile users are anticipated to access the virtual world with their mobile devices. However, still there are a couple of challenges and problems to be addressed for mobile users. In this report, state-of-art virtual world platforms are presented and their key features are compared. We compare possible approaches to tackle these problems to support virtual worlds for mobile devices. Transcoding scheme at the proxy is presented for the given computing and networking environments.

We achieved 4.48-5.77 and 2.96-3.16 frames per second with the Second Life viewer at the client, for 320x240 and 800x480 image sizes, respectively. The visual feedback from the user interaction with the server is less than one second in most of time. We identified the system bottleneck at the rendering proxy, and underutilization of the network bandwidth between proxy and client.

6. References

[Croquet] Croquet Consortium, http://www.opencroquet.org/index.php/Main_Page

[CroquetProgramming] Programming Croquet, Croquet Consortium, http://www.opencroquet.org/index.php/Programming_Croquet

[Cruz-Neira92] C. Cruz-Neira, D. J. Sandin, T. A. DeFanti, R. V. Kenyon, and J. C. Hart, "The CAVE: Audio visual experience automatic virtual environment," Communications of ACM, vol. 35, no. 6, pp. 65-72, June 1992

[Darkstar] Project Darkstar Community, http://www.projectdarkstar.com/

[DIS1993] IEEE standard for information technology - protocols for distributed simulation applications: Entity information and interaction. IEEE Standard 1278-1993, New York: IEEE Computer Society, 1993

[GoogleHow] How Google Earch [Really] Works, http://www.realityprime.com/articles/how-google-earth-really-works

[Hagsand96] O. Hagsand, "Interactive multiuser VEs in the DIVE system," IEEE Multimedia, vol. 3, no. 1, pp. 30-39, 1996

[Heiss07] Janice J. Heiss, Project Darkstar's New World of Online Games: A Conversation with Jeff Kesselman, October 2007, http://java.sun.com/developer/technicalArticles/Interviews/kesselman_qa.html

[jMonkey] jMonkeyEngine, http://www.jmonkeyengine.com/

[JPEG] Independent JPEG Group, http://www.ijg.org/

[Jung05] G. S. Jung, S. I. Kwon, M. W. Park, K. H. Won, S. K. Jung, "A New Dynamic Client-Server Scene Graph for MobileNPRQuake on Heterogeneous Frameworks," International Conference on Computer Animation and Social Agent, October 2005

[jVoiceBridge] jvoiceBridge, https://jvoicebridge.dev.java.net/

[LookingGlass] Hideya Kawahara, Paul Byrne, Deron Johnson, and Krishna Gadpalli, "Project Looking Glass: A Comprehensive Overview of the Technology," technical report, Sun Microsystems, Inc., March 2006

[Macedonia1994] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz, "NPSNET: A network software architecture for large-scale virtual environments," Presence: Teleoperators and Virtual Environments, vol. 3, no. 4, pp. 265-287, 1994

[Maemo] Open source development for Internet Tablets, http://maemo.org/

[Matijasevic02] M. Matijasevic, D. Gracanin, K. P. Valavanis, and I. Lovrek, "A Framework for Multiuser Distributed Virtual Environments," IEEE Transactions on

[Metaverse] Metaverse Roadmap: Pathways to the 3D web, http://metaverseroadmap.org/

[Miller95] D. C. Miller and J. A. Thorpe, "SIMNET: The advent of simulator networking," in Proceedings of IEEE, vol. 83, pp. 1114-1123, August 1995

[MPK20] MPK20: Sun's Virtual Workspace, http://research.sun.com/projects/mc/mpk20.html

[Multiverse] Multiverse, http://www.multiverse.net/

[MUPPETS] The MUPPETS Project, http://muppets.rit.edu/muppetsweb/about/index.php

[ODENetworking] ODENetworking, https://odenetworking.dev.java.net/

[OLIVE] On-Line Interactive Virtual Environment, http://www.forterrainc.com/products.php

[OLIVEtech] Forterra Systems, On-Line Interactive Virtual Environment (OLIVE), A Multi-user, Virtual World Software Platform, December 2006

[OpenSG] Open SG scene graph, http://www.opensg.org

[OpenSim] OpenSim, http://opensimulator.org/wiki/Main_Page

[Renaud] C. Renaud, Virtual Environments and their impact on the Network, CISCO, September 2007, http://xianrenaud.typepad.com/weblog/files/TechTalk2.pdf

[SecondLife] Second Life: Your World. Your Imagination, http://secondlife.com/

[Singhal99] Sandeep Singhal and Michael Zyda, Networked Virtual Environments – Design and Implementation, ACM Press, 1999.

[SLeconomics] Second Life, Economic Statistics, http://secondlife.com/whatis/economy_stats.php

[TaskGallery] The TaskGallery, Microsoft, http://research.microsoft.com/adapt/TaskGallery

[There] There - the online virtual world that is your everyday hangout, http://www.there.com

[Uni-verse] Uni-verse Home, http://www.uni-verse.org/

[VirtualGL] Virtual GL, http://www.virtualgl.org/

[Wonderland] Project Wonderland, https://lg3d-wonderland.dev.java.net/