Plug: Virtual Worlds for Millions of People

Shun-Yun Hu and Jehn-Ruey Jiang
Department of Computer Science and Information Engineering
National Central University, Taiwan, R.O.C.
{syhu, jrjiang}@csie.ncu.edu.tw

Abstract

*We propose the design of Plug, an application to find and keep contacts with friends within many inter-connected 3D virtual worlds. Users use an instant messenger (IM)-like interface to converse with friends and find new contacts, using a virtual representation of the user's self. It consists of three parts: a **plug** is an automatous agent / avatar situated at a user's computer that reflects and mimics its owner's behaviors and interests; a **plugspace** is a virtual environment that can be inter-connected, with scales ranging from a room to an entire virtual universe; and **plugtalk** is a set of packet formats and protocols that allow plugspaces to inter-connect, and individual plugs to navigate. Plug utilizes standards whenever possible, and is designed to be scalable, extensible, and customizable for various uses such as distance learning, virtual shopping, or online gaming. By combining the looks of 3D virtual worlds and the accessibility of IMs, we envision Plug as a step towards common virtual world experiences sharable by all Internet users.*

1. Introduction

In recent years, 3D multi-user virtual environments (VEs) [28] have seen tremendous growth in the form of massively multiplayer online games (MMOGs). MMOGs allow people to assume virtual character representations called *avatars* and participate in real-time, immersive interactions along with up to hundreds of thousands of other users. The most popular MMOG today, *World of Warcraft* [32], has over 10 million paid subscribers as of 2008, while the social virtual world *Second Life* also boosts over 10 million accounts at a growth rate of 120 additional servers per week [18]. These virtual worlds have not merely provided a new form of entertainment, but also a window on what is yet to come on the Internet. With current hardware trends, the growth and popularity of VEs likely will increase and diversify into other forms of applications such as learning, training, shopping, and socializing.

Arguably, the largest potential for online virtual worlds lies in the possibility to provide highly realistic, immersive interactions with other people in simulated environments across time and space. In the foreseeable future, graphics technologies may provide *photo-realistic* rendering for even average users; network bandwidth may increase to support content streaming for millions of easily browsable 3D sites; and imaging techniques may capture our facial expressions and body gestures to support *life-like, face-to-face* communications using our avatars.

However, presently VEs are not yet a popular form of Internet applications, and gamers only comprise a limited fraction of Internet users. This lack of adoptions may be contributed to three factors:

Incentive The most popular form of VE to date is MMOGs, and the majority of MMOGs today confines to the genre of role-playing games (RPGs), where players engage in quests to gain experience points, virtual money, and equipment upgrades. However, the RPG theme and genre may not appeal to all people, and the time required to master a game can deter causal Internet users.

Accessibility VEs today are still not as accessible as other Internet applications due to their hardware requirements, user interfaces, and time investments. As 3D content is often large in volume (e.g., hundreds to thousands of megabytes for a given MMOG), users often need to spend a long time to download and install the program, and patch for new updates. Compared with the experience of web browsing, accessing VEs today is still time-consuming and slow-paced.

Standard Two well-known elements to WWW's adoption and growth have been the HTML format and the HTTP protocol. HTML de-couples the efforts of content authors from software developers, and HTTP allows web browsers and web servers to be independently developed. The decoupling and interoperability of various efforts have contributed to scalable content and technology creations. However, while 3D content standards are emerging (e.g., Collada [7], VRML and its successor X3D [35]), no recognized protocols yet exist for VE applications.

Standards may not come easily given the diverse forms of VEs. On the other hand, incentive and accessibility are addressable with applications that are easy to install, quick to launch, and useful to specific needs. One important question related to incentive is the availability of *content*, as the scale of the virtual world and the amount of user interests are often determined by the amount of interesting content that exists. For MMOGs, a large amount of resources are often spent in content creation, where it is typically the most costly task during game production [1]. *User-generated content* hence is a viable alternative that could enable VEs to stay interesting without requiring expensive content creation pipelines (e.g., the creator of *SimCity* and *The Sims* Will Wright's most recent game *Spore* expects the users to create the majority of content [30]). The issue of accessibility is partially addressed as the average hardware improves. However, to deal with the typically large data volume, real-time *content streaming* is a more long-term solution, where users would only need a light installation, and download other relevant content as needed [12, 19]. User-generated content and content streaming thus have been the basis for a growing number of VEs to provide better incentives and accessibility (e.g., Second Life builds on user-generated content, and utilizes content streaming to deliver terabytes of a growing content base [31]. Other recent VEs such as IMVU, Metaplace, VastPark, and Google Lively[1] also adopt a similar approach). Despite these general trends, 3D content creation is still out of reach for ordinary users, and content streaming techniques are still in early stages where their scalability and efficiency are improvable.

An interesting possibility lies in marrying the VE experience with the interface of *instant messengers* (IMs) [14], which currently form some of the largest online networks for Internet users (e.g., some IM networks today, such as AIM, MSN, and QQ, boost between 100 to 200 millions registered users, and up to 20 million peak concurrent users [14]). If VEs can be accessed in IM-style (i.e., easy to install, non-intrusive, and useful for quick interactions), they will likely become more useful and accessible.

2. Design of Plug

We propose to develop *Plug*, an IM-like, user-centered virtual world application, formed by two main components:

Personalized, autonomous avatars Plug is based on user-generated content to attract users. However, instead of 3D content, we observe that the most interesting and generative content is actually the people themselves. The behaviors and conversations inside virtual worlds are the strongest social glues that connect and retain people. However, people may not always be available online, or are always interested in being online. Yet, their personalities and behaviors often are still of interest to others. To maintain and keep interesting personalities and behaviors always online, we can encapsulate them in the forms of autonomous agents that follow certain behavior scripts to act similarly as, and in the interests of, the original users. Many of today's popular MMOGs are troubled by *bots* [6], which are autonomous agents that continuously act and mine the resources inside games. They are often banned by game companies due to their disruptions to gameplay balance and fairness (i.e., players would gain experience points or virtual money without actually playing). However, Plug takes the opposite approach and embraces the concept of bots, by requiring every user to start the virtual world experience by building a personalized self-acting avatar. The concept is similar to owning a virtual pet on the user's desktop [9], except that the pet is the user himself or herself, acting on behalf of the user, in manners similar to the owner. We call this representation a *plug* of the user. A key functionality for a *plug* is to navigate across different user-generated virtual worlds on its own, search and interact with other *plugs* of interests, and collect information on behalf of the user.

Inter-connected, streamable virtual worlds The universe in which the *plugs* live are a set of inter-connected virtual worlds called *plugspaces*. Each *plugspace* is basically a VE with its own authentication, physics, interactions, and script-based semantics. The *plugspace* server also acts as the final arbitrator and repository for all the *game states* in the virtual world. *Plugspaces* are scalable both in the number of worlds (via common communication protocols) and the number of participants within a world (via distributed state management [10]). To allow greater accessibility, each *plugspace* is stream-based, in the sense that all 3D content (e.g., geometric meshes, textures, animation, physics rules), and the interaction / behavior scripts are downloaded from the hosting site on demand. Users only need a single *universal client* program to access and navigate the various *plugspaces*. However, as each *plugspace* may have its own rules on physical behaviors and semantics (e.g., a space may have heavier or lighter gravity, support for item trading or competitions among users). As such, user clients will need to download and follow the script-based rules when they enter a particular *plugspace*. Similar to websites, *plugspaces* can be hosted by any individual or organizations, on any machines connected to the Internet. Often, a user starts the VE experience by creating a *plug* and a room-sized *plugspace*, then it may control the *plug* to enter and navigate *plugspaces* hosted by others. Different *plugspace* owners may choose to collectively build a larger space by following the same set of in-world semantic scripts, and linking individually created regions via see-through *portals* [29]. Depending on the *plugspace*'s chosen state management scheme, interactions among the *plugs* may or may not go through the *plugspace* server.

[1] http://mmve.wiki.sourceforge.net/Companies

2.1 Usage scenario

Figure 1. Avatar chatting in IMVU [15]

Plug aims to help *plug* owners to maintain and develop new social relations via natural and spontaneous interactions. This is achieved in two stages: first, a user's *plug* may autonomously wander and navigate various public or private *plugspaces* to find other interesting or like-minded *plugs*. This is done by first learning a few well-known public *plugspaces* or directories. The *plugs* could then navigate much like bots in MMOGs, and interact with other *plugs* according to their behavior scripts. If some interesting *plugs* are found, the interactions (e.g., a conversation) are recorded and brought back to the *plug* owners. Second, individual *plug* owners may follow up to learn more about, or initiate interactions with, the other *plug* owners, by controlling the *plugs* themselves (similar to avatar chatting in the 3D chatrooms of IMVU, see Fig. 1). Here the interactions change from agent-to-agent to human-to-human, where two or more people can converse or interact via their *plugs*, with voice or text, body languages or avatar gestures.

2.2 Interoperability considerations

The most successful interoperable VE to-date is arguably the Distributed Interactive Simulation (DIS) protocol [22], which specifies a set of standard network packets such that any compliant simulators can join a simulation at any time, without *a priori* coordinations. On the other hand, API-level standards such as High Level Architecture (HLA) [17] has been less successful to offer interoperability among different implementations. This experience suggests that a common language at the network/protocol-level may be more essential to true interoperability[2]. However, we observe that a common set of network packets may not be enough for VEs, as the *interpretations* (i.e., processing) of the events and updates within the packets also need to be consistent to ensure consistent behaviors across hosts. In other words, VE nodes also need to agree at the semantic (i.e., game rules) level in order to be interoperable.

[2]http://www.web3d.org/x3d/workgroups/x3d-networking/

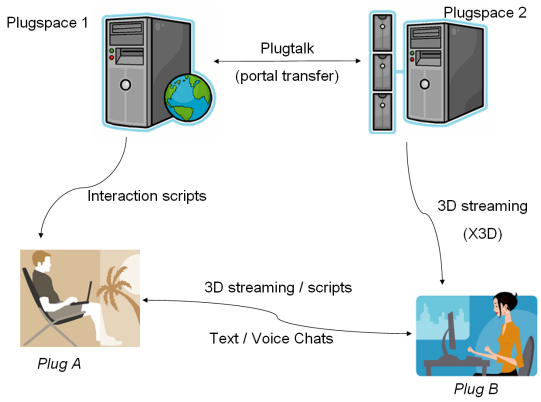

Figure 2. Data transfer in a Plug scenario

Plugspaces are intended to support interoperability with a common set of content formats, transmission protocols, and interaction scripts. As such, we try to utilize existing standards and protocols whenever possible. For example, the ISO standard X3D [35] is intended to be used as the main 3D content format. Although *Collada* [7] is another popular content format for games today, it lacks considerations for streaming and object interactions due to its design as an intermediate format in the content production pipeline [2] rather than as a deployment format. By using X3D for content representations, 3D avatars and objects can transit across *plugspaces* consistently.

For network communications, a set of protocols on top of TCP and UDP called *plugtalk* will be devised to allow different virtual worlds to negotiate the positions of portals and avatar transfer, so that users can walk from one *plugspace* to another. *Plugtalk* is mainly designed to aid the transitions between *plugspaces*, while leaving the internal semantics and state management to the *interaction scripts* of individual *plugspaces*. To allow flexible interpretations of VE schematics, while ensuring consistent interpretations for *plugspaces* that share the same semantics, the interaction scripts follow a layered design similar to the X3D standard, where various *profiles* [35] encapsulate and define different levels of semantic commonality. For example, at the most basic level, the data regarding avatars (e.g., appearance and animations) and physical properties (e.g., positions and movements) should be understood and consistently interpreted by all *plugspace* servers. Transitions of avatars between worlds should also be understood by all servers. Beyond this level (e.g., how many hit points will an attack takes), the semantics is left for the interaction scripts to decide, and consistent behaviors across VEs is achieved only if the *plugspace* owners agree to adopt a common set of scripts. See Fig. 2 for the communication flow in Plug.

3 Implementation Plans

Figure 3. realXtend test scene

Our goal for an initial Plug prototype is to bring a lightweight, easily installable VE client to users, where the virtual worlds can accommodate users scalably, and transit them smoothly. Our main approach is to convert an existing open source VE client to resemble IM in installation and usage, by transforming its content assets into X3D formats, and using peer-to-peer (P2P) networks to support VE's state management and content delivery [10, 12]. By adopting an existing VE client, we can focus our efforts on building and integrating relevant techniques that help current VEs to be more accessible and useful, instead of content generation. Adopting an existing client thus will provide us with real content to work with, while allowing us to focus on other aspects, such as improving incentives and accessibility for users. Likewise, we will also utilize and enhance existing open source libraries whenever possible. For example, we could use FLoD [12] for content streaming, and VAST [11] for distributive game state management.

We have surveyed and found two suitable open source large-scale VEs that may be targets for conversion: *WorldForge* [34] and *realXtend* [26] (Fig. 3). WorldForge is a VE project started in 1998 and has subsequently developed a number of related libraries as a game-building platform; realXtend is a modified Second Life client that uses the open source 3D engine OGRE [23] to support mesh-based models (which is more flexible than the primitive, or *prim-based* modeling used in Second Life). It is also compatible with both the official Second Life server and the open source Second Life server *Open Simulator* [24]. As WorldForge is still of prototype quality without the full functionalities of large-scale VEs, our current choice will be to use realXtend as the basis to prototype Plug, while leveraging existing Second Life functions. To make a virtual world IM-like, it is important to reduce its memory footprint so that only a minimal amount of resource is needed when not in use. As an example, the physical memory footprint of WorldForge is 200MB, and realXtend is between 80MB to 100MB, but both the 3D chatroom IMVU and the open source IM *Pidgin* [25] use only about 10MB of physical RAM. Our first task will thus be to study the code structure of realXtend and remove non-essential functionalities. We will also make the client program minimized when not in use, so that rendering is done only when active. As Second Life (and thus realXtend) already supports *3D streaming* [12] (i.e., 3D content is initially located at server-side), our second task will be to utilize P2P content streaming to improve streaming efficiency and scalability.

3.1 Avatar system

Automatous avatars require models, textures, animations, and behavior scripts to appear and function properly. We will base the 3D representations of the avatars in the X3D format, as it may be easier to transport avatars across different *plugspaces*. To provide automatous agent behaviors, initially we will allow users to build avatar profiles based on a list of keywords regarding personal interests. Avatars will wander randomly between the *plugspaces* of friends and public areas, then interchange and compare with other *plugs* for similar interests based on keywords. If keywords are matched, then a remote *plug* may be considered as a potential friend reportable to the *plug* owner.

Regarding the search for like-minded users, we observe that each of us often has questions that are best answered by some knowledgeable persons. As everyone is knowledgeable in certain areas, if knowledgeable people can be located online to answer specific questions, timely help can be provided, while new social interactions may also be stimulated. We thus would also provide a mechanism for users to search for and initiate conversations with appropriate users given any question [13].

3.2 State management

Successful IM networks can easily reach into the range of millions of concurrent users. It is thus important that the Plug architecture can accommodate various user sizes scalably. Recently peer-to-peer virtual environment research has provided solutions ranging from overlay management [11] (i.e., the discovery of neighbors within a visible area without the help of servers), to state management [3, 10] (i.e., the division of game object states onto different client nodes, with considerations for consistency, load balancing, and fault tolerance), to client-assisted services (e.g., voice chatting [16] or content streaming [12]). We will adopt Voronoi State Management (VSM) [10] as the main object state management system, by distributing the existing client-server event-processing to selected super-peers.

3.3 Content streaming

As realXtend is based on the official Second Life client, content streaming is already supported. However, there are still two main tasks for us:

1) **Distributed streaming:** Second Life and realXtend currently use the client-server model for content delivery, which makes content streaming fairly slow under the limited amount of server bandwidth. A scene typically takes over 30 seconds to one minute to download adequate details for interactions. We seek to improve the situation by utilizing idling client upload bandwidth, so that concurrent downloads from other clients may speed up the time to obtain a scene. We will achieve this by using and improving the FLoD library [12], which supports content discovery and peer selection for distributed, P2P-based 3D streaming.

2) **Format conversion:** The two main considerations for the content data itself are whether the content is *progressive* (so that full download is not necessary before navigation) and *standardized* (so that the same client may access different hosted worlds, and different client implementations can likewise access the same world). VE content mainly includes triangular meshes, 2D bitmaps (textures and light maps), animation data, sound effects, and music. As standardized progressive formats already exist for texture image and music / sound (e.g., textures may be stored as JPEG or PNG files, music as MP3 or OGG files), the main format issue thus lies in mesh models and animations. It will be our task to investigate whether the current formats used by realXtend (i.e., the *.mesh* format in OGRE) can support progressive transmission and be interchanged with X3D, so that interoperability across implementations is possible.

3.4 Interaction scripts

Commercial large-scale VEs often utilize *scripts* to dictate behaviors for non-player characters (NPCs) and game objects. For example, Second Life uses *Linden Script Language* (LSL) [20] as the internal script for all in-world object behaviors. Likewise, *Lua* [21] has been used by many MMOGs. We will simply adopt and continue the usage for LSL as it is currently supported within realXtend, but will also investigate the suitability for other script languages (e.g., object behaviors currently supported within X3D) if other behaviors require them.

3.5 Networking protocols

Plugtalk is the main communication protocol for Plug, whose main function is to allow user avatars be identified by different *plugspaces* and facilitate avatar transitions between them. *Plugtalk* will be a network-level, instead of API-level protocol, in order to ensure its interoperability and future extensibility. Common data for all avatars, such as the representation for the 3D mesh model, texture, and animation, as well as the position, scale, and orientation of the avatar, will need to be commonly communicated. If two worlds are connected via *portals* and desire a seamless transition among them, they could indicate in their communications that a common set of scripts are used by both *plugspaces*. This way, the two *plugspace* servers could exchange state updates directly as if a continuous world is being hosted.

4 Related work

The most successful large-scale VEs to-date are MMOGs (e.g., World of Warcraft and Second Life). However, most of these games require tedious downloads that make them inconvenient to access. Guild Wars [33] utilizes background download so that only a 100 kb loader is required at first. The execution of the loader then initiates the download of the first 100 MB of data to start up the initial game. Afterwards, continuous download occur in the background for content in nearby regions. In this way, download time may become unnoticeable once the initial game starts. However, Guild Wars is still an RPG that may not appeal to the wider audience, and the 100 MB download still does not compare with the almost instant access of the web.

Recent research on peer-to-peer virtual environments (P2P-VEs) attempt to support scalable and affordable VEs by distributing the bandwidth and processing loads traditionally assigned to centralized servers to all the participating client machines. Overlay designs such as Solipsis [8] and VON [11], state management such as Colyseus [3] and VSM [10], and systems such as HyperVerse [4] are some examples. Additional work based on these overlay also result in P2P-based 3D streaming design such as FLoD [12] and LOD-DT [5], or consistency framework such as Peers@Play [27]. We will utilize existing P2P-VE research effort such as VON and FLoD to distribute game states and support P2P-based 3D streaming in Plug.

5 Discussions

The designs of Plug address the following key issues for VE adoptions:

Incentive Plug is designed as a non-intrusive, simple application that sits on the user's desktop like an IM. The main function of Plug is to provide users a way to customize and train their avatars to search for other interesting people, and interact with them in a virtual world setting. These short socialization-oriented interactions are potentially more appealing and easier to use than current RPG-based gaming scenarios, by providing useful real-world functions.

Accessibility Making the user experience IM-like addresses the accessibility issue by allowing users to quickly enter a virtual world and start interactions any time, without having to wait for tedious downloads. This is enabled via distributed, concurrent streaming of the 3D content on P2P networks, which helps to speed up content streaming and enhance accessibility.

Standard Although standard is a difficult issue to address, by making the basic infrastructure highly scalable yet affordable to adopt, it will help standards to materialize once a large enough userbase is formed. We intend to do so by adopting peer-to-peer networks for all the underlying content delivery and state management mechanisms.

6 Conclusion

A successful implementation of Plug will allow any casual Internet users to quickly initiate conversations or interactions with other users within a virtual world setting. Internet users thus may interact with others in a new way, and enjoy the benefits of current virtual world technologies. With Plug, we would like to see virtual worlds be as easy to host as websites, and as accessible to use as web browsers or instant messengers. Techniques developed with Plug, namely, P2P-based 3D content streaming, state management, as well as the networking protocol for avatar transitions, will be useful as the basis for other types of VE applications. In time, Plug may serve as the first step towards a massive 3D Web based on virtual worlds, usable by millions of people.

Acknowledgments

We would like to thank the following individuals for invaluable discussions: Ye-Zen Chang Chen and Chen-Yu Yeh (Yakko) for the conception of *Plug*; Shao-Chen Chang, Jyun-Jie Huang, Davild Liu, Guan-Yu Huang, and Chien-Hao Chien for usability ideas; Jon Watte and Don Brutzman on interoperability issues.

References

[1] Multiplayer and network programming forum faq. http://www.gamedev.net/community/forums/showfaq.asp?forum_id=15, 2008.

[2] R. Arnaud and T. Parisi. Developing web applications with collada and x3d. http://www.khronos.org/collada/presentations/, 2007.

[3] A. Bharambe, J. Pang, and S. Seshan. Colyseus: A distributed architecture for multiplayer games. In *Proc. NSDI*, 2006.

[4] J. Botev et al. The hyperverse - concepts for a federated and torrent-based "3d web". In *Proc. MMVE*, 2008.

[5] R. Cavagna, C. Bouville, and J. Royan. P2p network for very large virtual environment. In *Proc. VRST*, 2006.

[6] K.-T. Chen, J.-W. Jiang, P. Huang, H.-H. Chu, C.-L. Lei, and W.-C. Chen. Identifying mmorpg bots: A traffic analysis approach. In *Proc. of ACM SIGCHI ACE 06*, Jun 2006.

[7] Collada. The khronos group. http://www.collada.org/, 2008.

[8] D. Frey et al. Solipsis: A decentralized architecture for virtual environments. In *Proc. MMVE*, 2008.

[9] GoPets. http://www.gopetslive.com/, 2008.

[10] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang. Voronoi state management for peer-to-peer massively multiplayer online games. In *Proc. NIME*, 2008.

[11] S.-Y. Hu, J.-F. Chen, and T.-H. Chen. Von: A scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006.

[12] S.-Y. Hu et al. Flod: A framework for peer-to-peer 3d streaming. In *Proc. INFOCOM*, 2008.

[13] J.-J. Huang, S.-C. Chang, and S.-Y. Hu. Searching for answers via social networks. In *Proc. CCNC*, 2008.

[14] IM. http://en.wikipedia.org/wiki/Instant_messaging, 2008.

[15] IMVU. http://www.imvu.com, 2008.

[16] J.-R. Jiang and H.-S. Chen. Peer-to-Peer AOI Voice Chatting for Massively Multiplayer Online Games. In *Proc. P2P-NVE*, 2007.

[17] F. Kuhl, R. Weatherly, and J. Dahmann. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall, 1999.

[18] I. Lamont. Second life's population problems. http://www.computerworld.com/blogs/node/5122, March 2007.

[19] F. W. B. Li, R. W. H. Lau, and D. Kilis. Gameod: an internet based game-on-demand framework. In *Proc. ACM VRST*, 2004.

[20] LSL. Linden script language. http://wiki.secondlife.com/wiki/LSL_Portal, 2008.

[21] Lua. http://www.lua.org/uses.html, 2008.

[22] D. C. Miller and J. A. Thorpe. Simnet: The advent of simulator networking. *Proc. IEEE*, 83:1114–1123, 1995.

[23] OGRE. http://www.ogre3d.org/, 2008.

[24] OpenSimulator. http://opensimulator.org/, 2008.

[25] Pidgin. http://www.pidgin.im, 2008.

[26] realXtend. http://www.realxtend.org/, 2008.

[27] G. Schiele et al. Consistency management for peer-to-peer-based massively multiuser virtual environments. In *Proc. MMVE*, 2008.

[28] S. Singhal and M. Zyda. *Networked Virtual Environments: Design and Implementation*. ACM Press, 1999.

[29] D. A. Smith, A. Kay, A. Raab, and D. P. Reed. Croquet - a collaboration system architecture. In *Proc. C5*, 2003.

[30] D. Terdiman. Wright hopes to spore another hit. http://www.wired.com/gaming/hardware/news/2005/05/67581, 2005.

[31] M. Wagner. Inside second life's data centers. http://www.informationweek.com/news/showArticle.jhtml?articleID=197800179, 2007.

[32] Wikipedia. http://en.wikipedia.org/wiki/World_of_Warcraft, 2008.

[33] Wikipedia. Guildwars content delivery architecture. http://en.wikipedia.org/wiki/Guild_Wars, 2008.

[34] WorldForge. http://www.worldforge.org/, 2008.

[35] X3D. Web3d consortium. http://www.web3d.org/about/overview/, 2008.