A context-aware immersive interface for teleoperation of mobile robots

João Quintas
Instituto Pedro Nunes
Coimbra
jquintas@ipn.pt

Luís Almeida
Instituto Politécnico de Tomar
Tomar
laa@ipt.pt

Elísio Sousa Paulo Menezes
Universidade de Coimbra
Coimbra
pm@deec.uc.pt

Abstract

In this paper we present a context-aware immersive teleoperation interface to assist operators during navigation tasks. This new interface strategy aims to address the problems associated with mental overload, often experienced by operators of teleoperated devices. Our approach simplifies the high complexity of information displayed in control rooms. Our approach includes a context-based human-robot interaction framework that detects relevant information and automatically adapts the displayed interface in virtual windshield. Results showed that the proposed approach enhances user immersion while maximizes task performances and minimizes the operator physical and cognitive workload.

Keywords

Human-Robot Interaction, telepresence, Embodiment, Teleoperation, Context-Awareness

1 Introduction

A telepresence robot [Minsky 80] presents a solution for search and rescue, remote reconnaissance, space exploration or maintenance in contaminated areas. In these scenarios, operators often intervene in the robot control loop when the robot is deployed in remote and unstructured environments [Sheridan 93].

Figure 1. A typical ROV Control Room. Courtesy of Monterey Bay Aquarium Research Institute.

An example of a typical control room for Remote Operated Vehicles (ROV) is depicted in figure 1. In spite of operator's skills and expertise, human decisions in teleoperation rely on diverse remote information sources. While teleoperating, operators must be fully focused in their task, which requires processing all inputs and filtering relevant information in order to execute the appropriate action. This intense use of perceptual and cognitive skills may lead to mental and physical strain, which may cause catastrophic hazards. This fact was addressed in [Wickens 08], where the authors studied how humans capabilities vary while performing tasks that require processing information from multiple resources. The studies concluded that multiple sources of information contribute to a high mental workload, causing negative implications on task performance.

In order to reduce the difficulties and stress of teleoperation, several authors propose solutions that allow the user to have a better understanding of the remote environment without the need to keep a mental record of the same.

The sensation of being (inside) the robot improves operator's performance of driving it. Thus, by combining the concepts of telepresence and physical embodiment we are able to create tele-embodiment [Paulos 97]. As result, the operator feels the remote robot body as his own and he/she acts more naturally, minimizing the physical and cognitive workload.

In the other hand, studies showed that operators quite often are not sufficient aware of robot location and surroundings, resulting in most operator decisions are based on remote video information, which forces the operator to try understanding the remote environment through a "keyhole" [Woods 04].

In [García 15], a virtual cockpit was proposed for intervention underwater robots that simplifies the high complexity of information displayed through specifically designed Graphical User Interface (GUI).

To tackle this challenge, we propose an approach to create a context-aware immersive interface for teleoperation of mobile robots that extends typical teleoperation functionalities, allowing human operators to benefit from an improved user experience. Figure 2 illustrates the expected outcome for this approach, which aim to improve over typical control rooms as depicted in figure 1.

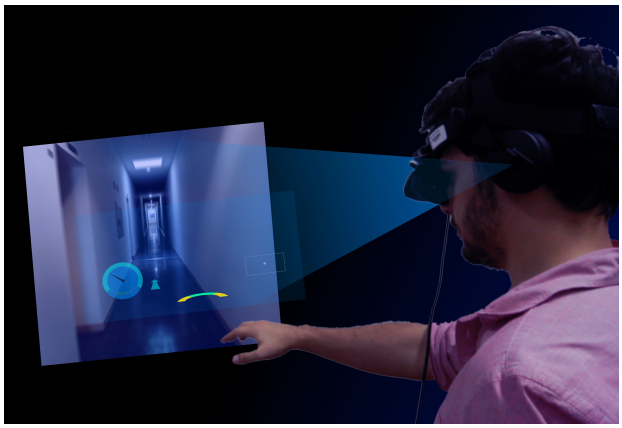


Figure 2. Context-aware immersive interface for teleoperation of mobile robots

Consider a teleoperation scenario where an operator is using our immersive interface. The operator is performing a navigation task, of a mobile robot, in remote environment with good weather conditions (e.g. partially sunny and low humidity). For this task, the operator could be more interested in paying attention in bearing and speed of the robot. The immersive interface would display a simplified control panel with widgets relevant only for that given context (i.e. navigating the robot with good weather). Suppose now, that during teleoperation, the weather changed and the robot's environment is rainy with wind gusts. Our context-aware immersive teleoperation interface will now adapt and display widgets related with wind direction and indication of applied torque in the wheels, as a muddy floor requires a more skillful driving to avoid getting stuck.

1.1 Context-awareness

Schmidt in [Schmidt 00] regarded situational context, such as location, surrounding environment or state of the device, as implicit input to the system. This extended the concept of context beyond the informational context into real world environments. The authors used this concept to define the term "implicit human-interaction" as *"... an action performed by the user that is not primarily aimed to interact with a computerised system but which such a system understands as input ..."*

Röning and Riekki in [Röning 01] proposed a context-aware mobile system, which included mobile personal robots. They proposed the "Genie of the Net" architecture as an ever expanding system providing helpful information and guidance when human capabilities are exceeded. Their proposed approach also aimed to be a technique to handle several individual robots so that they can co-operate with each other and human beings. Their initial application tests selected the approach of first building a teleoperated robot and then gradually shifting tasks from the human to the robot.

Celikkanat, et.al. in [Celikkanat 15] demonstrated on the iCub platform that using context resulted in an adaptive, online and robust approach for executing two important tasks: object recognition and planning.

In our approach we will define context as:

> *Context is the set of information that is relevant, affects or constrains how some action is taken, without being the center of interest of the action.*

1.2 Contributions and structure

Based on these principles, we present an approach that aims to improve the telepresence experience for the operator when remotely operating a mobile robots. The Augmented Reality based user interface (UI) proposed in [García 15] is now coupled with a context-aware module and will automatically adapt operator's UI to changing conditions that are relevant for the task being performed, resulting in a context-aware immersive interface. This auto-adaptation consists in providing cues to the operator that aim to simplify the teleoperation interface.

The paper proceeds as follows. Section 2 presents design aspects, including teleoperation mechanism and the role of context information in the interaction process of telepresence and teleoperation. Section 3 describes experimental and comparative results of different interface styles. As an application example, we address the scenario of an operator remotely controlling a robot while his context aware user interface adapts to help him during the navigation task, providing the necessary information for the given context while hiding the irrelevant one to avoid distracting or overloading the user. Section 4 summarize the conclusions.

2 Designing the context-aware immersive interface for teleoperation of mobile robots

In this section we will address some design aspects for our approach referring to teleoperation architectural details and necessary adaptations to achieve a context-aware immersive system.

2.1 Teleoperation architecture

Literature proposes teleoperation models with the human operator inside the control loop. Usually, the robot control commands are transmitted through a delayed transmission channel [Islam 14][Sheridan 93][Almeida 14] and, the action feedback is also affected by a transmission delay. The

model purpose is to integrate these delays and keep the robot controllable.

In our research, we explore the relationship between the human and the interface used to control the remote robot. We propose a simplified model composed by an outer teleoperation control loop that uses an inner perception control loop, see figure 3.

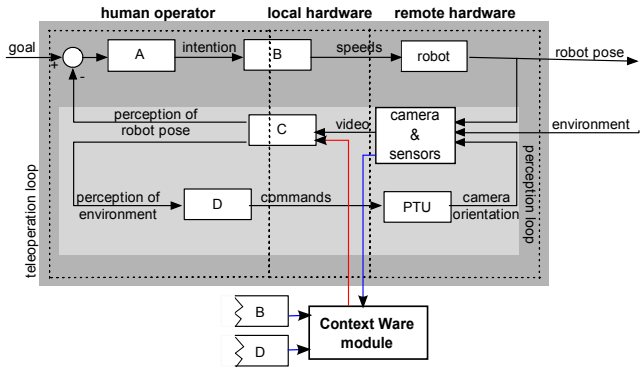


Figure 3. Teleoperation and perception as control loops

teleoperation loop – the robot teleoperation process can be modeled as a standard control loop. Basically, the human operator compares a given goal with the robot's position in the remote environment. The operator perceives the difference and develops an intention to compensate it, which is later translated into robot's commands by some interaction system. Figure 3 depicts this loop, where block A represents the perception of the error and the intention generation. This intention is converted into commands through block B, which models the human action into an interface that produces proper robot commands. This control loop will be closed only if the user can perceive the pose of the robot in the remote place.

Perception Loop – this research in teleoperation systems considers a camera point of view, as the operator being inside the robot. The camera's purpose is to enable user to perceive both the robot motion and the surrounding environment as being driving inside the mobile robot. This perception process can similarly be described as a control loop. In this case, the human operator controls the robot's camera orientation and utilizes the visual feedback to compensate the scanning process required for a task (ex: track objects, look around, inspect, or navigate). As in the control loop, the camera acquires images and sends them through a channel to the user. This visual information enables the operator to perceive the relative pose of the robot in the remote environment and, the environment itself. Block C represents this process.

The Context Aware module recognizes an activity based on robot's sensors information, operator's positional intention of the robot (block B) and operators visual point of view (block D) and, with it selects the useful information to be presented in the operator's windshield or UI (block C).

2.2 From teleoperation to remote embodied operation

Using the presented model lets map the different perception and control mechanisms into blocks A, B, C and D. We demonstrate how to evolve from a traditional teleoperation concept to new and more immersive approaches. In traditional teleoperation systems, block B represents the robot motion control using a joystick and, block D, represent the control of the pan-and-tilt camera unit using also a joystick. Block C provides the remote images to the user through standard screen, while block A, enables him to convert the positional perceived error into an intention to move the robot.

To create a more immersive interface we propose a viewpoint transfer. To solve the challenge of controlling the remote viewing camera, we suggest the use of a head mounted display (HMD) in which the operator can move his head and almost simultaneously control a pan-and-tilt unit (PTU) that supports the robot's camera. Block C provides the visual information that enables the user to perceive the difference between the visual goal, and the means to compensate. The human, through block D, acts into camera PTU to gather new point of views.

This type of camera control provides an egocentric view, as the camera movements are synchronous with the operator's head movements. It enables the user to have an egocentric perception of the remote environment just as if the human was at the robot position and orientation. The described process is a crucial step to give the user the sensation of being physically embodied in the remote robot, which means that "the user will see what the robot can see".

2.3 Creating a context-aware immersive interface for teleoperating mobile robots

In our approach we consider context recognition to be a periodic process that operates in the background of the system, while interacting with a user. This process is illustrated in figure 4.

It plays the role of detecting changes in the context and control the adaptation in the user interface during teleoperation.

To incorporate context-awareness into our architecture we designed a Context-based Human-Robot Interaction Framework (CB-HRI). Figure 5 illustrates the CB-HRI framework, which conceptually extends teleoperation architecture.

This framework acts as a middleware to integrate contextual information in the overall system and control the workflow related with human-robot interaction.

The main components of the framework are:

1. the *Message Bus* module that includes the interfaces with other components in the architecture.
2. the *Recognition algorithms* module includes the algorithms to match perceived information with contextual information.

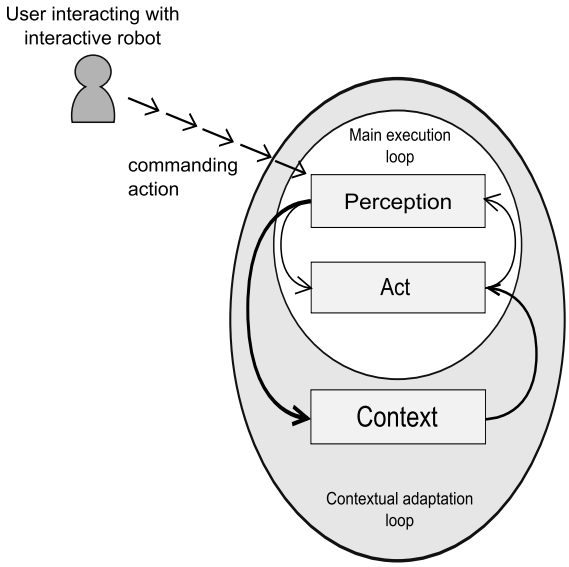


Figure 4. Context verification process

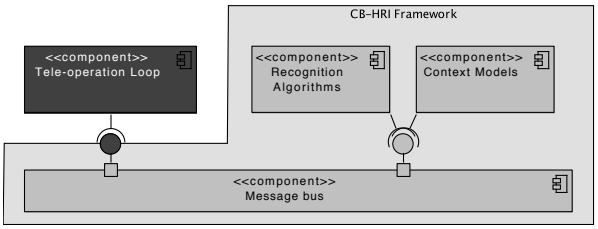


Figure 5. Context-based Human-Robot Interaction Framework extending existing architectures

3. the *Context models* is a repository with apriori context data models.

In our approach we propose contextual information as an integration mechanism between a variety of available algorithms and other resources that are known to perform well under a certain conditions.

Therefore, the CB-HRI framework must be integrated with the components dealing with perception, reasoning, data storage and actuation.

3 Implementation and Results

In our experiment the objective was to navigate as quickly as possible, without colliding against walls or obstacles, as illustrated in figure 6. The operator could make the robot move forward, move backward, turn 360º on itself, and control the robot's camera point of view. The robot on-board sensors could provide the following information: movement speed, movement direction, 360º proximity information, camera's pose and battery levels.

Choosing only the most relevant information at any given time, provides uncluttered field of view and decreases the user's mental workload. Furthermore, for the information that is always present, this can be slightly transparent as not to block the user's view. The graphical elements representing the information should not be too big and placed near the user's view centre, as not to strain the user's eye and focus.

Figure 6. Navigation task, comparison of different teleoperation interaction styles designed to enhance embodiments sensations

3.1 Modeling and recognizing Contexts

In our approach we represent Context as a vector, where each element is a numerical representation of a feature (i.e. context features).

In order to select coherent context features we take as baseline the same measured information in previous works, where we explored immersive teleoperation interaction styles and their application in a virtual cockpit in a navigation task. Thus, we enumerate the context features used as:

- Task (e.g. reserved for future use)
- User proficiency (e.g. 0 = beginner, 1 = amateur, 2 = professional)
- Distance to obstacle (e.g. 0 = close, 1 = near, 2 = far)
- Safe speed limit (e.g. 0 = slow, 1 = fast)
- Bearing to obstacle (e.g. 0 = no adjustment, 1 = adjust right, 2 = adjust left)

In this set of features we can neglect Task, as this information will be irrelevant because we are only performing navigation.

We create a set of rules based on the previous features to define Context classes, as follow:

- Context0: Navigating in open space
- Context1: Narrow space / Close proximity to obstacle

- Context2: Too much speed to avoid obstacle without colliding
- Context3: Wrong bearing to overcome the obstacle

The result of the classification is then used by the user interface, which loads the appropriate widgets that provide relevant information to the user while navigating in a specific environment. Specifically, taking into account our experiment, we have the following adaptations:

- No widgets are displayed in Context0;
- Display the speedometer widget in Context1;
- Display the bearing indicator widget in Context2.

3.2 Implementing the immersive interface

To implement the graphical interface, a combination of OpenGL and OpenCV was used. Setting the video stream as background with OpenCV, the 2D elements are created with OpenGL and then placed on top of the stream. The robot movement speed was designed according to modern speedometers to offer a degree of familiarity to the user, since it is one of the most common and intuitive ways to display an object's speed. As such, a speedometer background containing a gauge was designed and a needle (with a transparent background) was created and placed on top of the background, with the lower end of the needle aligned with its center. By applying rotation to the needle, we can make it move and indicate the robot speed. The robot's direction is simply an arrow indicating forwards or backwards. It is designed to look like it's pointing along the Z-axis (depth) for better intuitiveness (see figure 7). Proximity information is represented as a circle with as many sections as there are sensors. Each section changes color depending on the distance from the robot to the nearest object in the corresponding direction. The circle is designed to look like it's aligned with the Z-axis to facilitate the user's perception of which sensors are displaying information. To create the sections, various circles with single sections are stacked upon each other and controlled independently. The camera pose is represented by a circle inside a square. Pan is represented through the X-axis and tilt is represented through the Y-axis. The square represents the minimum and maximum pan and tilt limits. The battery level is represented by a numerical percentage inside a drawing of a common battery.

Other widgets, like battery level will be displayed if context awareness module considers that is important for the task.

3.3 Validating user interaction

To validate our approach we compared experimentally four interaction styles, which included from traditional joystick approaches to more innovative based on deictic gestures and natural body postures. We carried out a quantitative and subjective task performance analysis involving 13 participants. All the participants had to teleoperate a mobile robot and navigate through a predefined obstacle course.

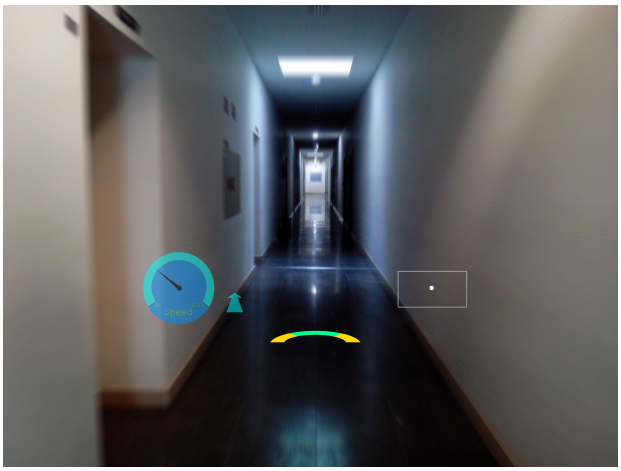


Figure 7. Operator's immersive windshield with smart widgets: semi transparent speedometer, robot direction motion arrow, proximity sensor and camera pose.

The experiment goals were to maximize the task performance and minimize the operator's physical and cognitive workload. We induced in the operator the sensation of being at the remote environment; to generate the remote physical embodiment feeling, the approach consisted in letting user perceive the robot's structure as his/her own body.

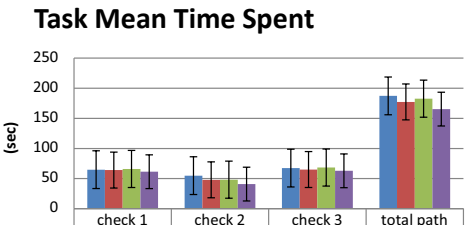
To evolve from teleoperation to embodied operation we explored 3 approaches:

1) view transfer using an HMD (i.e. with an egocentric controlled view, the user will see what robot can see);
2) pointing gestures to control the robot (i.e. user sees him as being the robot, or inside of it, and his pointing gestures are used to control his own motions);
3) body posture to control the robot.

To understand the influence of the four different interaction styles on the teleoperation of a mobile robot; and to assess how natural can a user interact and perceive the remote robot structure as his own body, a quantitative and subjective task performance analysis were carry out (13 participants in driving tasks)(figure 6).

Results demonstrated that visual feedback through an HMD improved significantly users task performance (figure 8). The introduction of natural deictic gestures based robot control presented some gain in task performance when compared with joystick. Body intention-based robot control was the operator's choice in all subjective questionnaires, and was confirmed by time performance measures in path driving. As conclusions of the introduced gesture, postures and view control mechanisms improves the physical embodiment sensation. Sensation of controlling the robot from inside reduces mental workload of the opera-

tor. There is a positive effect on user satisfaction and task performance.



	check 1	check 2	check 3	total path
joystick + monitor	65	55	67	187
joystick + HMD	64	48	65	177
deictic gesture + HMD	66	48	68	183
body intention + HMD	61	41	63	165

Figure 8. Navigation task performance time comparison while using different teleoperation interaction styles

4 Conclusion and future work

The paper addressed the challenges that enhances telepresence and teleoperation. We considered the importance of integrating contextual information in the interaction process with the objective to improve user experience while performing a teleoperated task.

To understand the influence of 4 different interaction styles on the teleoperation of a mobile robot; and to assess how natural can a user interact and perceive the remote robot structure as his own body, a quantitative and subjective task performance analysis were carry out demonstrating that visual feedback through an HMD improved significantly users task performance.

Moreover, we addressed the navigation task as an example task where the integration of context information can improve usability of the system by providing pro-active cues that help the operator to perform the task. An rational management of the information and a egocentric point of view maximizes task performances and minimizes the operator physical and cognitive workload.

References

[Almeida 14] L. Almeida, B. Patrao, P. Menezes, and J. Dias. Be the robot: Human embodiment in tele-operation driving tasks. In RO-MAN, 2012 IEEE, Aug 2014.

[Celikkanat 15] H. Celikkanat, G. Orhan, N. Pugeault, F. Guerin, E. Sahin, and S. Kalkan. Learning Context on a Humanoid Robot using Incremental Latent Dirichlet Allocation. IEEE Transactions on Autonomous Mental Development, 2015.

[García 15] Juan Carlos García, Bruno Patrão, Javier Perez, Joaão Seabra, Paulo Menezes, Jorge Dias, and Pedro J. Sanz. Towards an immersive and natural gesture controlled interface for intervention underwater robots. In Oceans'15 MTS/IEEE Genova, Genova (Italy), 05/2015 2015.

[Islam 14] Shafiqul Islam, XiaopingP. Liu, AbdulmotalebEl Saddik, Lakmal Seneviratne, and Jorge Dias. Control schemes for passive teleoperation systems over wide area communication networks with time varying delay. International Journal of Automation and Computing, 11(1):100–108, 2014.

[Minsky 80] Marvin Minsky. Telepresence. Omni, pages 45–51, 1980.

[Paulos 97] Eric Paulos and John Canny. Ubiquitous tele-embodiment: Applications and implications. International Journal of Human-Computer Studies, 46:861–877, 1997.

[Röning 01] Juha Röning and Jukka Riekki. Context-Aware Mobile Robots: Part of Smart Environment. In HRI-workshop at KTH, 2001.

[Schmidt 00] Albrecht Schmidt. Implicit human computer interaction through context. Personal Technologies, 4(2-3):191–199, June 2000.

[Sheridan 93] T.B. Sheridan. Space teleoperation through time delay: review and prognosis. Robotics and Automation, IEEE Transactions on, 9(5):592–606, Oct 1993.

[Wickens 08] Christopher D. Wickens. Multiple Resources and Mental Workload. Human Factors: The Journal of the Human Factors and Ergonomics Society, 50(3):449–455, June 2008.

[Woods 04] D.D. Woods, J. Tittle, M. Feil, and A. Roesler. Envisioning human-robot coordination in future operations. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 34(2):210–218, May 2004.

3D and the Web
The last twenty years and the future

Vitor Cardoso
Universidade Aberta (UAb) / CIAC
Lisbon
Vitor.Cardoso@uab.pt

Abstract
A critical view on two decades of 3D graphics "standards" for the Web. Starting with the first W3C standard (VRML) and its evolution (X3D), the reasons for decay and why according to several evidences WebGL, by the Khronos group, seems to be the unavoidable road ahead. Among them, a new phase of "WebGL era" is coming, with full grown 3D applications being WebGL ready.

Keywords
Web3D, Virtual Reality, Virtual Worlds, VRML, X3D, WebGL.

1. INTRODUCTION

In October 2014, HTML's 5.0 final version arrived. It includes in the norm significant definitions for several media (audio, video, 2D graphics) but, against what was expected by some developers. 3D is absent. At least apparently because, on one hand, VRML/X3D[1] - the "official" norm - was not yet revoked although never widely adopted and still relaying on third party plugins. A X3D Working Group inside W3C and the Web3D consortium have been working for years with "the purpose of fully integrating X3D with HTML" (Web3D Consortium, 2011) but had no success; on the other hand another Web Graphics norm, WebGL, proposed by the Khronos group (well known by its open standard OpenGL) is strongly emerging and already runs plugin less over the major web browsers.

Summarizing, this article addresses the following points:

- VRML/X3D: the first 3D graphics standard for the Web
- VRML/X3D technology: what went wrong?
- WebGL, the road ahead for 3D on the Web?
- Tools for designers with some programming skills
- 3D Graphical editors and Web deployers

2. VRML/X3D: THE FIRST 3D GRAPHICS STANDARD FOR THE WEB

Before any attempt to foresee the future of Web 3D graphics one needs to briefly rationalize and review what has been learned from the history of 3D graphics standards on the Web.

In the first years of the Web era (started in 1991) and the excitement over a multimedia global experience, 3D come naturally as the next level. Proposed in 1994 (Pesce, Kennard, & Parisi, 1994) and endorsed shortly after (Bell, Parisi, & Pesce, 1994) by vrml.org (now web3d.org), an organism under w3c consortium, VRML/X3D technology has been around for two decades as "the" intended 3D graphics standard for the Web. Yet, despite this "official" status, it was never implemented natively on any major web browser. Until today, users must consider installing a plugin in order to run VRML/X3D content inside a browser.

2.1 Enthusiasm in the 1990s

VRML looked, back then, cool and unique for the Web. It supported 3D geometry, animation, and scripting (Web3D Consortium, 2015). From the beginning and especially on the second half of the nineties VRML/X3D gathered considerable attention and enthusiasm among artists, engineers and educators. Since it was doing well and achieving an interesting (moderate) widespread use over the Web, the enthusiasm spread, engaging some reference public and private organizations, from NASA to SUN Corporation. Even today search engines still show a significant evidence of the quantity and quality of VRML work done back then. Beside its own potential and "official" status as a web technology, some key important factors have contributed in our view to this early enthusiasm and use of VRML:

- **General Interest/curiosity for 3D**

In the nineties the curiosity and general interest about 3D

[1] We will use the term VRML/X3D to refer the continuity of this technology with two norm versions that coexist, even today.

was growing and the possibility of associating it to the, also new, Web was appealing.

- **Browser Wars and VRML endorsement**

During the first “browser war" days, in mid-nineties, Netscape and Microsoft had their focus on every relevant novelty to promote their browser and, in no time, Netscape Navigator and Internet explorer had, each one, their free VRML plugin to offer. This was in the public eyes an obvious technology endorsement and clear incentive for users and content developers to install VRML capability on their browsers and use/create 3D for the Web.

- **Multiuser worlds, avatars & 3D Communities**

The third key element was surely the quite immediate availability of 3D multiuser solutions. Innovative 3D multiuser (VRML) worlds with avatars, most using technology from "Black Sun" (latter reborn as Blaxxun), formed at Sun Technologies (Dammer, 1997). Black Sun's worlds open in 1997, Cybertown in 1997 (Poster, 2014) and “Le Deuxième Monde” from 1998 are among the most significant and iconic early projects done in VRML. They gathered on the nineties a significant amount of world users around this technology.

It’s important to highlight that in addition to 3D graphics, provided by VRML, Blaxxun and others, brought 3D multiuser technology that constituted a solid base for virtual communities on the Web (Blaxxun Interactive, 1998). The solution was quite *easy* to implement, working nicely and well integrated in the Web ecosystem. This looked back then incredibly innovative and appealing, gathering an enthusiastic community which remained the hallmark of VRML and X3D evolution throughout many development cycles (Butzman & Daly, 2007)

3. VRML/X3D: WHAT WENT WRONG?

In this point we rationalize about how VRML/X3D technology changed from a phase of initial enthusiasm to the present low pace.

In 2001 Web3D.org proposed X3D, a new 3D web graphics norm, meant to solidify the VRML path and ideally be included in Web browsers core, avoiding plugin need, but this never happened. Unfortunately in the early 2000s Netscape had already lost the browsers war and the winner, Microsoft, backed the posture of browser innovation, dropped the support for his own VRML plugin and removed it from their site! Moreover, including VRML in Internet Explorer core was also out of the question.

From then on, there were no more "official" VRML/X3D plugins for specific browsers and users had to take the risk of installing a third party plugin. The opportunity was lost and, despite being an official "3D standard for the Web", since then the interest for VRML/X3D technology dropped significantly.

Cybertown, a vibrant and innovative free community until 2001 was sold to IVN in 2002 and started to charge users a fee for membership. That led to a massive abandon. At first users and creators looked for other free VRML/Blaxxun related communities and multiuser servers (even Blaxxun had its own free server). ABNet/Babel X3D[2] was one of them.

In early 2002, Blaxxun went out of business and the support for its free VRML/X3D plugin (Blaxxun contact) was at risk. Bitmanagement took the plugin development, renamed it as bs contact on a new version and ... started to charge for it! The users of their “unrestricted” demo had to cope with an annoying floating logo over the 3D scenes and worlds. Bitmanagement did a fine technical work evolving bs contact to the most recent 3D graphics norms and enhancements, but that ugly logo was probably the last drop that disgusted users and creators.

Many moved away from VRML/X3D to other emergent technology communities, including Second Life[3] that was opening its doors in 2002.

From 2002 on, 3D over the web took a low pace but did not die. The "need” was there but now, instead of one, several 3D technologies along with VRML/X3D concurred to fulfill the demand. Among them Unity3D, with its 3D Web plugin and especially Flash.

> “2001 saw Adobe’s notable rise to web 3D power with version 8.5 of their Director software. Featuring Shockwave 3D technology, Adobe Director allowed creatives to produce hardware-accelerated 3D graphics with scripted interaction using the Lingo language. Full 3D browser based games could be created, such as Xform Games’ GoKartGo!Turbo!” (Helix Design Studio, 2013)

For the past decade Flash has been regarded as the de facto standard for deploying rich graphics (including 3D) and multimedia on the web. Unfortunately Adobe professional tools remained out of reach from common 3D non-profit or educational creators since they are expensive and the company never had a policy of free tools (as others like Autodesk do, for example).

3.1 The swan song of VRML/X3D

From the VRML/X3D side, among other interesting projects. Vivaty, a 3D virtual worlds community, deserves special mention. One of the VRML founders, Tony Parisi, took the lead of a brilliant team, including Keith Victor (creator of VRML/X3D editors, Spazz3D/VizX3D/Vivaty Editor) and Rick Kimball (creator of ABNet multiuser server) and founded Vivaty in 2007 (Parisi, 2010). In there we could see a new level of professional quality graphics and innovative social interaction rarely seen before in VRML/X3D. It showed how VRML/X3D was a fantastic Web 3D technology up level with others much more recent. Vivaty closed in 2010 and was, in several ways, the VRML/X3D well deserved swan song.

[2] http://www.odisseia.univ-ab.pt/abnet2

[3] Second Life, a non-web 3D technology, uses a specific client program as is out of scope in this paper.

4. WEBGL, THE ROAD AHEAD FOR 3D ON THE WEB?

Now we rationalize about WebGL and the available evidence pointing it as the road ahead.

4.1 What is WebGL?

In mid 2000s the non-profit Khronos group deploys OpenGL ES, a 3D rendering API for mobile and "embedded systems" (ES), based on the desktop long-established 3D rendering standard OpenGL but optimized for mobile/handheld devices. As an industry standard and royalty free, OpenGL ES became universally adopted on small computing devices, most notably phones and tablets to deliver a hardware-accelerated 3D experience.

In early 2009, the non-profit technology consortium Khronos Group started the WebGL Working Group. WebGL is a Web version of OpenGL ES 2.0. The designers felt that, by basing the API on OpenGL ES's small footprint, it would be more achievable to deliver a consistent, cross-platform, cross-browser 3D API for the web (Parisi, 2014).

WebGL is implemented as low-level API JavaScript. It uses the HTML5 canvas element and is accessed using DOM (Document Object Model) interfaces. Automatic memory management is provided as part of the JavaScript language. As such, WebGL runs directly in browsers (desktops or mobiles) without the need for a specific plugin to harness the full power of the computer's 3D rendering hardware. It is today supported by all the major browsers (IE, Firefox, Chrome and Safari) on desktops and mobile platforms..

4.2 Why WebGL? Some relevant aspects

- Technically sound and proved standard. WebGL is based on long experienced, widely adopted, open and free standards and is already supported by all major browsers (IE, Firefox, Chrome and Safari) on desktops and mobile platforms. It is the long waited 3D on the web without plugins!
- It's light and fast. Based on standards with a small footprint, it is more capable to deliver a consistent, cross-platform, cross-browser 3D API for the web and capable to deliver a hardware-accelerated 3D experience using the device GPU directly. We should say here that we were amazed by the incredible speed a demo scene ran on a three year old galaxy note II mobile phone.
- Perfectly integrated in HTML 5 canvas. WebGL wires the GPU to the browser with a JavaScript-based OpenGL ES API, thanks to the HTML5 canvas tag. This means WebGL content is a DOM element (this was never true with VRML, because it operated as a plug-in) and can be manipulated with the same procedural or formatting techniques as any other element. It's finally at reach a seamless 2D/3D web content integration.

4.3 WebGL risks, performance and compatibility

Comparing to VRML/X3D, WebGL as a lower level language is much harder to cope with and direct content creation is for serious 3D graphics programmers.

"WebGL is simply a programming API for JavaScript built on top of OpenGL which is a graphics abstraction layer. Since WebGL makes direct JavaScript calls to OpenGL, content creation is for serious 3D graphics programmers who know how to deal with a 4×4 transformation matrix and who can speak the GLSL shader language fluently. Exposing OpenGL as JavaScript is nice, but we do not expect web-page authors to become graphics programmers …
X3D describes scene graphs and 3D content declaratively. This means that authors define what geometry and interaction belongs in a model, rather than programming the low-level details for how polygons get built and drawn. Authors can write XML descriptions for their content in a manner similar to (X)HTML.

Therefore X3D authoring is much more like Web development. Content creators can also easily export VRML or X3D models from their favorite authoring tool, from whatever format, and publish them using the Web." (Anita Havele, 2012)

There are some concerns with WebGL security risks coming with direct access to the GPU. Due to these risks, initially Microsoft and Apple refused to support WebGL but that changed in time partially because the browsers war is back again and no one wants to be left behind and also because the upcoming WebGL versions addressed the security concerns. According to WebGL Security white paper, by the Khronos Group, WebGL conforms to all the security principles of the web platform and was designed with security in mind from day one.

WebGL performance is in general not as good as native execution - it is limited by the dynamic nature of JavaScript. Even so, performance has increased over time and the current browser implementations do a great job of optimizing it. In situations mostly GPU-bound, we can now expect WebGL to perform very similar to native code.

Universal WebGL availability and compatibility is growing but, in the meantime, here and there problems may occur. We have experienced problems with Android running in virtual machines, which was to be expected, but problems may also occur, as we also have experienced, on systems with older graphics cards (GPU) and/or outdated drivers. In some cases certain features or all of WebGL isn't available. The Khronos WebGL wiki has a list of supported configurations.

4.4 How are the alternatives to WebGL going?

Comparing to VRML/X3D, WebGL advanced clearly in browser support and it's popular "plugin less 3D graphics" feature is highly appealing. Google Trends shows clearly that in search statistics WebGL term has become much more popular than VRML and X3D immediately after version 1.0 launch in 2011 (Figure 1).

On the other side comparing to Flash it did not. Flash has been an incredible platform, done a lot for the web as an interactivity and entertainment platform. The reasons to change from Flash to WebGL in the foreseen future are more "political" than technical.

The exact timing is not entirely known right now but there

is a declared intention (started by Apple in 2010) to end, sooner or later, the support for plugins in major browsers, especially plugins that run proprietary compiled code. It affects Java, Silverlight, *bs contact* (VRML/X3D), Unity 3D Web plugin, Flash and many others. This is one of the reasons why Unity dropped flash to switch all development efforts in the creation of a WebGL deployer.

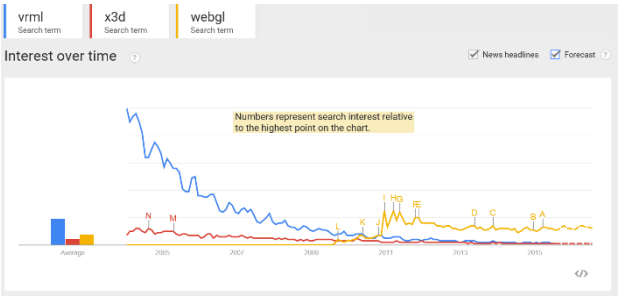


Figure 1 - Comparing VRML, X3D and WebGL Search terms. Source: Google Trends, 2005-2015

Facing an uncertain future, Flash's popularity has been falling in search term statistics (Figure 2).

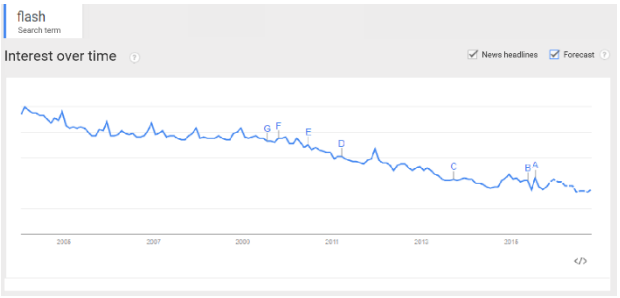


Figure 2 - Flash in Search terms. Source: Google Trends, 2005-2015

4.5 WebGL is already supported on more devices than Flash

"The times they are a-changin'". The declared intentions to end the support for third party plugins in major browsers and the mobile market growth are changing the landscape dramatically and Flash is no longer the largest platform on the planet for interactive browser graphics. Flash is not supported on mobile browsers out of the box; it primarily reaches Mac and PC desktop browsers. On the contrary, WebGL reaches all platforms, desktop and mobile, via the browsers but can also be packaged into native apps and run independent of a browser on both desktop and mobile. It is now supported on more devices than Flash (Krüger, 2014).

With such a wide support for WebGL, the demand for content is growing rapidly and the software industry is already committing. Tool producers are working on WebGL exporters or already have the solutions to offer. Unity 3D stopped Flash export development (the feature was removed in recent versions) and is now committed to WebGL. The recent version 5 outdoes the former Web plugin by a WebGL export format. Even Adobe itself has adapted its professional tools to produce content in HTML5 and WebGL alongside with Flash. Adobes position is incredibly pragmatic and, obviously, endorses WebGL as the road ahead for 3D graphics on the Web.

In the remaining parts of the article we comment about some fundamental tools for programmers and designers to create WebGL content for the ongoing virtual Web.

5. TOOLS FOR DESIGNERS WITH SOME PROGRAMMING SKILLS

Firstly we comment on tools intended for programmers and designers with some programming skills: the JavaScript middleware libraries for WebGL. Intended for creators with programming skills, they are fundamental to convert and create WebGL content.

WebGL is a low level programming language; this means its code appears very technical and, without a layer of helpful abstraction to assist in reading and learning the syntax, it may frustrate designers with low programming skills who want to produce creative, 3D interactive scenes for webpages. Thankfully a good number of developers have already produced tools and JavaScript libraries to help increase the accessibility of WebGL (a popular one is Three.js). These JavaScript "wrappers" provide an alternative set of commands for creating objects in 3D space. For creators with low programming skills, but that don't fear coding, these libraries simplify the development of WebGL applications and the conversion from other 3D formats to WebGL.

5.1 X3DOM

Web3D Consortium's member, Fraunhofer, using WebGL has developed a JavaScript based interface for X3D intended as a useful framework for WebGL development and transition from X3D. It runs in any HTML 5 browser and supports native X3D within an HTML page. Former VRML/X3D creators will find X3DOM interesting since it converts VRML/X3D objects and scenes to WebGL using a X3D interpreter written in JavaScript. It works well for static scenes and simple animations but, to our knowledge, does not implement the full X3D spec; that would be a major undertaking.

This has been a fine development as the projects available in the site (x3dom.org) demonstrate. Although not exclusively X3DOM is of particular interest to VRML/X3D developers.

5.2 Other JavaScript middleware libraries for WebGL

- Three.js (http://threejs.org), is the most popular and has become a reference platform for WebGL development.
  A quick browse of the demos page (http://threejs.org/examples) shows how powerful and practical this library is. Three.js can convert to WebGL from many popular formats including VRML.
- A list of other JavaScript libraries for WebGL is available at: http://www.webgl-game-engines.com/

6. 3D GRAPHICAL EDITORS AND WEB DEPLOYERS

Finally we comment about useful 3D graphical editors to convert and create WebGL content.

6.1 3D editors and WebGL converters

Although JavaScript middleware libraries may be fundamental at some point in the building process, common graphic creators might have a hard time dealing with them. Nothing like using a good graphical 3D editor, indeed. Fortunately a significant number of new editors, tools and plugins (for mainstream 3D editors) appear daily. Look for news and info about them in places such as http://learningwebgl.com/blog/.

One of the most interesting new editors is *Coppercube* (http://www.ambiera.com/coppercube/), a commercial product. It appeared a couple of years ago and was the first WebGL ready 3D editor. It uses CopperLicht, an open source WebGL library developed by the same owner. Coppercube is not (yet) as full featured as mainstream 3D editors like Blender, but is clean, easy to use and has been until now one of the few tools around that made WebGL content development easy, practical and exciting, as its demos and user forums show.

Unfortunately, the absence of a free editor version for nonprofit users and education somehow restrained a potentially larger dissemination that this fine product has merited. Now that the big named editors are committing to be WebGL ready, Coppercube faces a harder fight.

Bitmanagement´s BS Content Studio, a commercial product, is another interesting authoring tool that, in the steps of the company's long experience on VRML/X3D, now also exports to WebGL among other format platforms. Unfortunately, also here, the absence of free versions for nonprofit users and education has restrained Bitmanagement fine products from a potentially larger dissemination.

6.2 Free mainstream 3D editors to deploy WebGL

As we see it today, by the products already in place, the ones coming soon and the intentions declared, Unity 3D seems to be best positioned to build and deploy high quality dynamic Web 3D content, whether it's games or interactive virtual scenes and environments, for multiple purposes. Since Unity is not a 3D modeler, designers will need to choose a modeler. Blender and Sketchup are two among several possible options.

6.2.1 3D modellers

Blender is free and one of the best tools for 3D. Does not have (yet) its own native WebGL exporter, out of the box, but we can install the excellent Blend4Web exporter plugin and enjoy its promising WebGL conversions.

Blender is a complete tool and can also create games (includes its own game engine) and interactive virtual scenes. With Blend4Web plugin scenes with limited interactivity can be deployed to the web but not yet games (not yet at least) nor highly interactive virtual scenes. In a WebGL scenario Blender is much more interesting when used with Unity 3D. In this *duo virtuoso* Blender is the modeler for objects and scenes that Unity (which reads blender files directly) uses as assets to create the action of the game/virtual environment and deploy to end user platforms. Unity manual (Unity Technologies, 2015) has useful info on how to integrate both seamlessly.

One should have in mind that Blender is a serious choice, surely a good one in the long run, but is not necessarily the easiest choice for beginners.

Sketchup instead is a powerful 3D editor, highly intuitive and, from our experience, beginners find it much easier to work with than Blender. Sketchup free version is still a great 3D creation tool and exports to *collada* (.dae), a universal format that Unity3D can import quite well. It has good documentation, instructional videos, a huge collection of free 3D objects and plugins. Adding to this, Sketchup has a legion of followers and tons of free tutorials and resources on the Web.
There are several ways/services to display a Sketchup model in WebGL; the easiest is to upload the model to 3D Warehouse (https://3dwarehouse.sketchup.com) which will render exactly this kind of stream automatically and without cost. Once the model has been processed, can be embedded in any web page.

6.2.2 Scene assembler and deployer

The best placed for the moment is probably Unity3D. The reasons why in our vision is a fundamental tool to build and deploy Web 3D content are:

- It is one of the best tools available to deploy games and dynamic virtual environments on several platforms: desktop, mobile and Web. Unity has committed itself to WebGL, starting on version 5 released in early March 2015. Apart from that it deploys content to more than 20 platforms including the former Unity Web plugin, now overtaken by WebGL.
- Until 2014, Unity 3D had a hardly interesting free version since it did not include some essential features of 3D (like dynamic shadows and advanced water, among others) and that disappointed users. That has changed completely and now Unity3D Editor has the same features on free and paid versions. Only some very "high" advanced features, related to special services and game optimization, more directed to professional developers, are reserved for paying customers. In our view, the free version is totally adequate for non-profit users and education.
- In addition, Unity renewed the interface editor (GUI) starting from version 4.6. The 3D editor is now more intuitive and easier to learn.
- Multiuser worlds have for long been created with Unity using its network capabilities, but it was not an easy task so Unity team has committed itself (in the official blog) to deploy with Unity 5.1, recently released, tools and features that simplify the creation of multiuser 3D environments and games, an area much appreciated by former VRML/X3D creators and from other 3D communities.

6.3 A new phase of WebGL era with full grown 3D applications being WebGL ready

We do not want to promote a specific product, but would not be fair to omit that Unity's commitment to WebGL is

not to be seen as one more in a long list. Unity 3D is the first "big named" 3D/game editor to have a full WebGL deployer (not a limited exporter for simple stuff). As a reference tool for games and interactive virtual scenes/worlds creation with almost everything we need, including avatars, special effects, and so on, Unity's WebGL deployer has to be incredibly advanced, a first of a kind, in order to generate correctly all the interaction and special effects demanded by commercial games.

This defines a new era for WebGL where we change from experiments with limited tools (even Coppercube is an ongoing experiment, some relevant features are not entirely implemented or still in development), limited exporters and "specific" tools to full grown 3D applications that offer in WebGL all we had before for other technologies/3D formats. An article about the first commercially available Unity WebGL game (Schwartz & Nyman, 2014) has important information and is a vivid confirmation of what we said about Unity´s WebGL deployer relevance.

We have commented here on a selection of tools, most free, to deploy 3D on the Web via WebGL. There are, of course, other alternatives and choices, free or paid, to deploy WebGL content.

7. CONCLUSIONS

VRML appeared in the mid-nineties and was rapidly endorsed by main browser producers through their “official” free plugins. This, joined to a relatively accessible language syntax and the novelty effect of a cool and unique 3D standard for the Web, led to a considerable early enthusiasm that has grown rapidly into powerful multiuser technology developments and the appearance of multiuser worlds and communities of considerable dynamism and influence in the nineties.

The 2000s did not bring the expected boom. Despite the norm upgrade to X3D in 2001, VRML/X3D was never included in major bowsers native code. On the contrary, the major browser producer Microsoft, that had recently gained the browser war with Internet Explorer, ended its endorsed free VRML plugin and thus removed millions of users from this 3D technology confining it to a very small market niche. Shortly after, major players like Blaxxun and Cybertown went out of business. Some of their heirs, like Bitmanagement, kept evolving the technology and tried to survive the difficult years but some disregard for nonprofit users, marketing policy and prices, kept moving common users and creators away to other web 3D technologies (among them Flash and Unity3D) and to non-Web as well (like Second Life and Open Sim).

In early 2009, Khronos consortium started the WebGL Working Group and Version 1.0 of WebGL specification was released in March 2011. Developed upon OpenGL ES 2.0 free standard WebGL benefitted immediately from a wide acceptance since it was supported natively by major browsers such as Firefox and Chrome on desktop platforms. The successive announcements of Flash’s inglorious deprecation in major platforms also helped WebGL that widespread rapidly to all major browsers and platforms. The initial lack of good content creating tools for non-programmers has changed over time and now major 3D content creators, like Unity3D, are WebGL ready or aiming to.

Today everything apparently points to WebGL as the major 3D graphics technology running on Web browsers from now on. However, WebGL itself is not a language created for Web designers and direct content creation with the language is reserved for advanced 3D graphics programmers. Middleware JavaScript libraries and mainstream 3D graphical editors reduce this problem significantly but do not eliminate it. At some point, in an exploding 3D on the Web scenario, a Web designer may need to tweak the code directly but WebGL was not designed with that user in mind as VRML/X3D was. Some progress in this regard would be highly welcome

8. REFERENCES

Anita Havele. (2012, July 6). *Research 2.0 - Thought Leader Interview* . Retrieved from web3d.org: http://www.web3d.org/wiki/index.php/Interview_with_Research_2.0

Bell, G., Parisi, A., & Pesce, M. (1994). The VRML 1.0 Specification. *Proceedings of the Second International Conference on the World Wide Web, Chicago, October 1994.* Chicago.

Blaxxun Interactive. (1998). *blaxxun Community Platform - Server SDK*. Retrieved from Blaxxun.com: http://www.mission-base.com/uni-weimar/peter/mututorial/api/doc/apextern.html

Butzman, D., & Daly, L. (2007). *X3D: Extensible 3D Graphics for Web Authors.* Morgan Kaufman.

Helix Design Studio. (2013, January 21). *From VRML to WebGL, a look at 3D on the Internet.* Retrieved from blog.helixwebsites.co.uk: http://blog.helixwebsites.co.uk/2013/01/from-vrml-to-webgl-a-look-at-3d-on-the-internet/

Krüger, M. (2014). *Flash Is Dead ... Long Live WebGL.* Retrieved from wired.com: http://www.wired.com/2014/05/flash-dead-long-live-webgl/

Parisi, T. (2010). *Real-time rants, reflections and ruminations on building a 3D Web*. Retrieved from the immersive Web(log): http://flux.typepad.com/

Parisi, T. (2014). *Programming 3D Applications with HTML5 and WebGL.* O'Reilly.

Pesce, M., Kennard, P., & Parisi, A. (1994). Cyberspace. *Proceedings of the First International Conference on the World Wide Web, May, 1994.* Geneva.

Schwartz, A., & Nyman, R. (2014, October 14). *Unity games in WebGL: Owlchemy Labs’ conversion of Aaaaa! to asm.js.* Retrieved from hacks.mozilla.org:

https://hacks.mozilla.org/2014/10/unity-games-in-webgl-owlchemy-labs-conversion-of-aaaaa-to-asm-js/

Unity Technologies. (2015). *Importing Objects From Blender*. Retrieved from Unity Manual: http://docs.unity3d.com/Manual/HOWTO-ImportObjectBlender.html

Web3D Consortium. (2011, February 17). *Goals: X3D and HTML5*. Retrieved from Public X3D Wiki: http://www.web3d.org/wiki/index.php/X3D_and_HTML5

Web3D Consortium. (2015, July). *X3D & VRML, The Most Widely Used 3D Formats*. Retrieved from http://www.web3d.org: http://www.web3d.org/x3d-vrml-most-widely-used-3d-formats