Comparative Evaluation of Free/Open Source Software to Develop Virtual Reality Systems

Eduardo Islas Pérez[1], Ivonne Ávila Gutierrez[1], Ilse Leal Aulenbacher[1] and Benjamin Zayas Pérez[1]

[1] Instituto de Investigaciones Eléctricas, Av. Reforma 113 Col Reforma, Cuernavaca, Morelos, México, 62490
{eislas, ivon_ag, ileal, zayas}@iie.org.mx

Abstract. In this paper we describe an evaluation methodology for virtual reality (VR) free/open source software tools. A Multi Criteria Decision Making (MCDM) methodology based on a criteria set and weight assignment was applied. The analysis and evaluation aimed to help decision makers to select the most appropriate software tools to develop VR applications. The selected tools were used to develop a virtual reality system to teach concepts related to generation, transmission and distribution of electricity from a power plant to consumption centres.

1 Introduction

The success of a virtual reality project or system relies on many factors. One of the most important aspects is good planning, in which software resources must be considered and managed. An adequate selection of VR software tools may determine the success or failure of a project.

In order to develop a VR system with a high degree of interaction, immersion and realism, we need different types of free/open source and commercial software tools. Therefore, software analysis and evaluation must be carried out to identify the most appropriate tools that can be integrated to support the development process. The results of these activities should reflect in the selection of software with the right features according to the system requirements. These results can improve the planning and developing of a project.

The system described in this paper explores the use of VR as a training tool to help learners (utility workers and students) to become familiar with the equipment and facilities of a power system: from electricity generation in a power plant, to the distribution lines for domestic supply, going through the transmission towers and the substations. This application provides the users with different levels of immersive, interactive and three-dimensional experience allowing them to explore the power system freely in order to know the elements and equipment involved with power generation, transmission and distribution.

Received 18/09/07
Accepted 19/10/07
Final version 24/10/07

2 Free/Open Source Software

Besides the obvious low cost of Free/Open Source Software (FOSS), there are many other reasons why public/private organizations are adopting this kind of technology [1]. The most important are: security, reliability/stability, open standards / vendor independence, reduced reliance on imports, developing local software and capacity.

- **Security**. Development method, program architecture and target market can greatly affect the security of a system and consequently make it easier or more difficult to violate. There are some examples where FOSS systems are superior to proprietary systems [2].

  Three reasons are often cited for FOSS's better security record:

  - **Availability of source code**: Availability has made it easier for developers and users to discover and fix vulnerabilities as soon as they are found.
  - **Security focus, instead of user-friendliness**: It is more focused on robustness and functionality, rather than ease of use.
  - **Roots**: These systems are mostly based on the multi-user, network-ready Unix model. Because of this, they come with a strong security and permission structure.

- **Reliability/Stability**. FOSS is well known for their stability and reliability. For example, Vaughan and Steven conducted a reliability test between Red Hat Linux, Caldera Systems OpenLinux and Microsoft's Windows NT Server 4.0. The result was that NT crashed once every six weeks but none of the FOSS systems crashed at all during a period of 10 months [3].

  In other example Prof. Miller from Wisconsin University has been measuring reliability by feeding programs random characters and determining which ones resisted crashing and freeze-ups (Fuzz testing). This approach is unlikely to find subtle failures, the study found that their approach still manages to find many errors in production software and is a useful tool for finding software flaws. What is more, this approach is extremely fair and can be broadly applied to any program, making it possible to compare different programs fairly [4].

- **Open standards and vendor independence**. Open standards give users flexibility and the freedom to change between different software packages, platforms and vendors. Proprietary, secret standards lock users into using software only from one vendor and leave them at the mercy of the vendor at a later stage, when all their data is in the vendor's proprietary format and the costs of converting them to an open standard is prohibitively high.

- **Reduced reliance on imports**. A major incentive for developing countries to adopt FOSS systems is the enormous cost of proprietary software licenses. Because virtually all proprietary software in developing countries is imported, their purchase consumes precious hard currency and foreign reserves. These reserves could be better spent on other development goals in developing countries.

– **Developing local software capacity**. It has been noted that there is a positive correlation between the growth of a FOSS developer base and the innovative capacities (software) of an economy. There are three reasons for this:
  - **Low barriers to entry:** FOSS, which encourages free modification and redistribution, is easy to obtain, use and learn from.
  - **FOSS as an excellent training system**: The open and collaborative nature of FOSS allows a student to examine and experiment with software concepts at virtually no direct cost to society. Likewise, a student can tap into the global collaborative FOSS development network that includes massive archives of technical information and interactive discussion tools.
  - **FOSS as a source of standards**: FOSS often becomes a de facto standard by virtue of its dominance in a particular sector of an industry. By being involved in setting the standards in a particular FOSS application, a region can ensure that the standard produced takes into account regional needs and cultural considerations.

FOSS has significant market share in many markets, is often the most reliable software, and in many cases has the best performance. FOSS scales, both in problem size and project size and often it has far better security, perhaps due to the possibility of worldwide review. Total cost of ownership for FOSS is often far less than proprietary software, especially as the number of platforms increases. These statements are not merely opinions; these effects can be shown quantitatively, using a wide variety of measures. This does not even consider other issues that are hard to measure, such as freedom from control by a single source, freedom from licensing management (with its accompanying risk of audit and litigation) [5].

In the case of our application, we need to accomplish some of these features. For instance, security is needed for systems that run over the Internet or a public network; in fact, we aim to develop this kind of applications in the near future. In the reliability/stability context, not only the mentioned aspects are relevant but also the stability of developers or vendors. For example, the well-known company Sense 8 is no longer available in the VR field since a couple of years ago, leaving some customers without any kind of support for their VR systems or projects.

3 VR Software Description

We have identified four types of software tools commonly used to develop VR applications, which are: toolkits and graphic environments for programming and developing VR applications, tools for 3D modeling, tools for developing mathematical models and tools for 3D visualization. Figure 1 depicts this type of software tools. The analysis and evaluation described in this paper are mainly based on this classification.

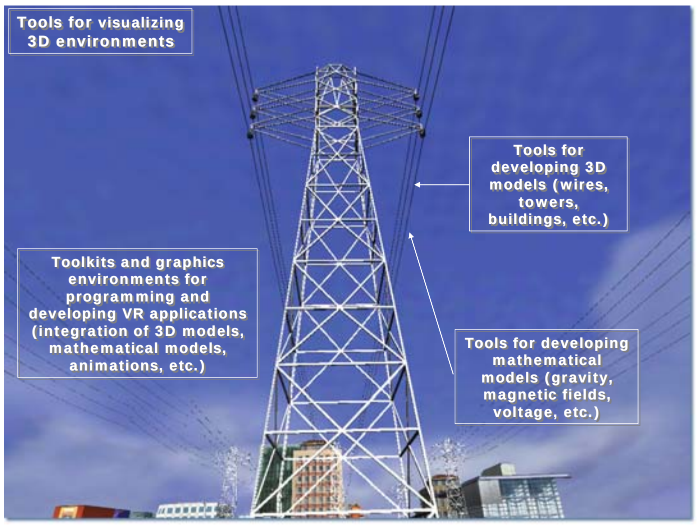
Tools for visualizing 3D environments
Tools for developing 3D models (wires, towers, buildings, etc.)
Toolkits and graphics environments for programming and developing VR applications (integration of 3D models, mathematical models, animations, etc.)
Tools for developing mathematical models (gravity, magnetic fields, voltage, etc.)

Fig. 1. Types of software tools involved in a VR system development

- **Toolkits and graphic environments for programming and developing VR applications.** Although there are many kinds of toolkits to develop three-dimensional environments or virtual worlds, we have only considered two types of tools:
  - **Graphic environments to develop VR applications**. These tools provide a graphical environment to develop applications. Basic nodes or primitives are used to build more complex virtual worlds. In addition to the graphical interface, this type of tools also offer some built-in basic animations and object behaviors, which make the development of applications easier than with development toolkits. In particular, a good background in programming is not needed to develop applications. This is one of the biggest advantages of these environments. However, the limited functionality they provide precludes the development of complex applications.
  - **Toolkits for programming VR applications.** A toolkit is an extensible library of object-oriented functions designed specifically for developing VR applications. Toolkits are in a middle stage between low-level graphic language, such as Open GL [6] and graphic environments such as Open Inventor [7]. These tools afford functionality through a rich set of function libraries such as: connectivity with input/output hardware, behaviors, animations, lighting techniques, etc. Complementary functions can also be programmed by developers using high-level programming languages such as C++ or Java.
- **Tools for 3D modeling**. Three-dimensional models can be created and edited with this type of tool. The usage of a diversity of techniques, objects, scenes and environments can be replicated. Some of these tools allow developing simple object animations, object behaviours and special effects through scripting. These tools are important in the sense that they create the visual part of a VR application.

- **Tools for developing mathematical models**. With this kind of tool, object behaviors can be modeled in a virtual environment. Simulations are based on mathematical models of behaviors such as gravity, inertia, weight, acceleration, etc. Object behaviour makes the representation of physical settings more realistic.
- **Tools for 3D visualization**. These types of tools are used for model visualization, interacting with virtual objects in a scene and exploring a virtual environment. These tools are divided into two categories: general-purpose or developed for a specific application. Usually, toolkits or other virtual reality software offer viewers for their particular applications. However, those general-purpose viewers are limited in functionality. Viewers with extended functionality can be built with the tools aforementioned in this section. These viewers can be distributed to final users without buying additional development licenses for toolkits and graphic environments.

4 VR Software Evaluation

The MCDM methodology used in this project is based on some concepts from the methodology described in [8] which was applied in the evaluation of VR hardware and software tools [9] and appraisal of Learning Management Systems (LMSs) [10]. It is worth pointing out that at some degree it is a general-purpose methodology, in the sense that depending on the kind of items to be evaluated, a set of matching criteria (or parameters) must be defined. We applied this methodology to evaluate different VR software tools. However, due to lack of space, only the evaluation details of toolkits and graphic environments for developing VR applications are presented in this paper. Results for other software tools are only shown.

a) **Identification and selection of evaluation parameters**. The list of parameters considered in the toolkits evaluation and graphic environments is shown in Table 1:

Table 1. Evaluation parameters for toolkits and graphic environments

	Parameter		Parameter
1.	*Drivers to ease hardware integration*	2.	Use of communication networks
3.	Classes or functions library	4.	Multiplatform and portability
5.	Import and export 3D models and scenes	6.	Import and export animations
7.	Geometries library	8.	Optimization
9.	Audio	10.	Realism level
11.	Animation	12.	Rendering and visualization
13.	Use of databases	14.	Open source and versions
15.	Availability of demos	16.	Management aspects
17.	Company Profiles		

b) **Value assignment for each parameter**. Table 2 shows in detail the features that are taken into consideration when grading and scaling the *Drivers to ease hardware integration* parameter. Details on the other parameters have been intentionally omitted due the lack of space.

Table 2. Grading for drivers to ease hardware integration

Features
Drivers to ease the use of:
1. Video equipment (*HMDs*, *eyeglasses*, *CAVEs*, etc.)
2. Audio equipment (*headphones*, *speakers*).
3. Haptic devices (*gloves*, *cybergrasp*, etc.)
4. Equipment for movement (input devices)(*mice*, *josticks*, etc.)
5. Positional gadgets (*trackers*, *nestbirds*, etc.)

c) **Weight assignment for all of the parameters**. Weights assigned to each parameter were: 1, 1.5 and 2 where 1 means an optional parameter, 1.5 a parameter to improve immersion, interaction or development and 2 means a very important parameter.

d) **Identification and selection of tools**. Because of the great number of software tools available nowadays and based on the most important attributes, we made a pre-selection of software tools. The final list of the evaluated software tools is shown in Table 3.

Table 3. Identification and selection of software tools

Software Tools	**Company**	**Price (USD)**
MetaVR [11]	MetaVR, Inc.	10,500.00
IRRLicht [12]	IRRLicht	FOSS
Cult3D [13]	Cycore	7,700.00
Torque [14]	GarageGames, Inc.	395.00
Open Inventor [7]	Mercury Inc.	5,000.00
Horizon Scene Graph [15]	DigiUtopikA Lda.	Not available on line
OpenGL Performer [16]	Silicon Graphics	Not available on line
Panda 3D [17]	Disney and Carnegie Mellon University	FOSS
Java 3D [18]	Sun Developer Network	FOSS
OpenSceneGraph [19]	*OpenSceneGraph*	*FOSS*
X3D [20]	Web 3D Consortium	FOSS
VR Juggler [21]	Iowa State University's Virtual Reality Center	FOSS

e) **Analysis and evaluation of each tool**. Table 4 shows the results for the OpenSceneGraph evaluation using an MCDM method. This method is the additive value function and non-hierarchical weight assessment which is briefly described below [10].

$$MAX\ V(x_j) = \sum_{i=1}^{n} w_i v_i (x_{ij}) \tag{1}$$

where:

x_{ij} = The value of criterion i for alternative j

$v_i(x_{ij}) =$	A single criterion value function that converts the criterion into a measure of value or worth. These are often scaled from 0 to 1, with more being better. In this method these values were not scaled
$w_i =$	Weight for criterion *i*, representing its relative importance.
$n =$	Number of criterions

Table 4. Evaluation results of OpenSceneGraph

Parameter	x_{ij}	$v_i(x_{ij})$	w_i	$w_i v_i(x_{ij})$
Drivers to ease hardware integration	*1*	*1*	*2.0*	*2.0*
Use of communication networks	2	2	1.5	3.0
Classes or functions library	4	4	2.0	8.0
Multiplatform and portability	2.5	2.5	2.0	5.0
Import and export 3D models and scenes	3	3	2.0	6.0
Import and export animations	3	3	2.0	6.0
Geometries library	2	2	1.0	2.0
Optimization	5	5	2.0	10.0
Audio	1	1	1.5	1.5
Realism level	3	3	2.0	6.0
Animation	3	3	2.0	6.0
Renderization and visualization	4	4	2.0	8.0
Use of databases	2	2	1.5	3.0
Open Source and versions	4	4	1.0	4.0
Demos availability	3	3	1.0	3.0
Management aspects	3	3	2.0	6.0
Company profiles	1	1	1.5	1.5
		$\sum_{i=1}^{n} w_i v_i(x_{ij})$		**81.0**

f) **Obtaining a graph to compare tools**. The results obtained for this group of tools are shown in Figure 2. The results show the best commercial and FOSS toolkits and graphic environments for programming and developing VR applications. Figure 3, Figure 4 and Figure 5 show the evaluation for the other types of software considered in this assessment, which were obtained by applying this methodology.

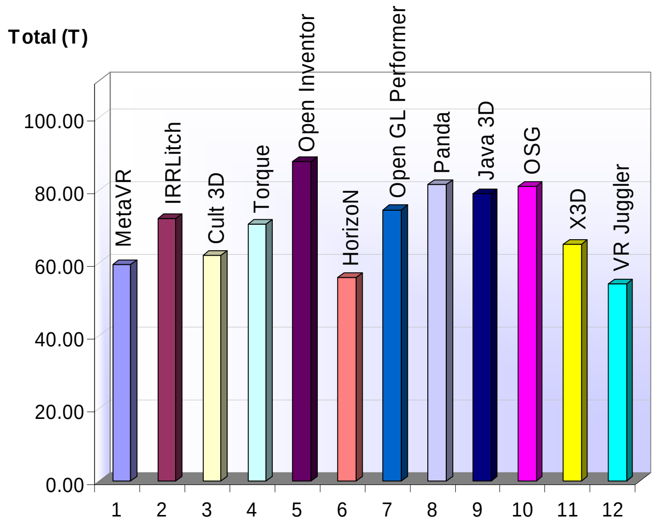

Fig. 2. Toolkits and graphic environments for programming and developing VR applications

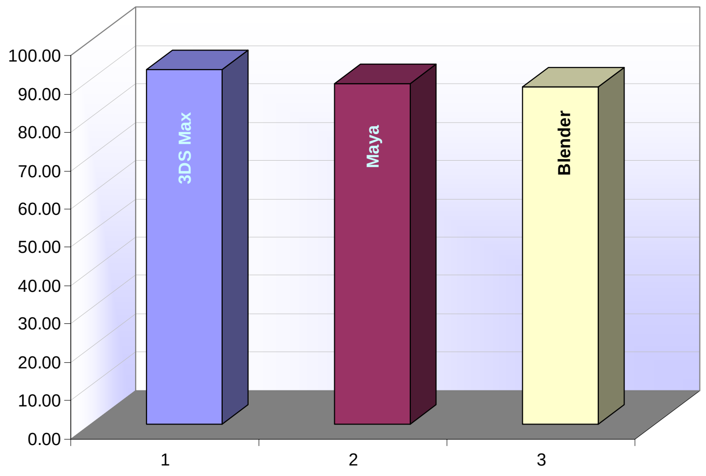

Fig. 3. Tools to develop 3D models (3DS Max, Maya, Blender)

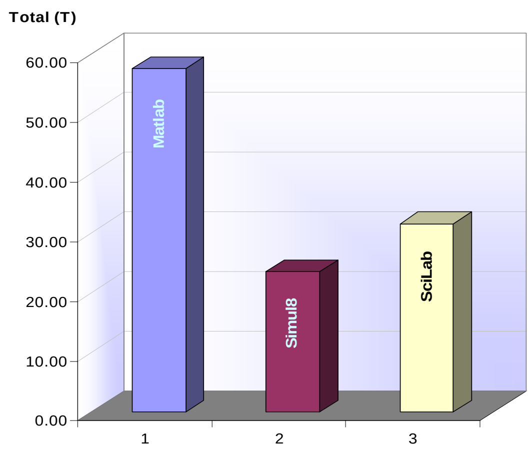

Fig. 4. Tools to develop mathematical models (Matlab, Simulink, SciLab)

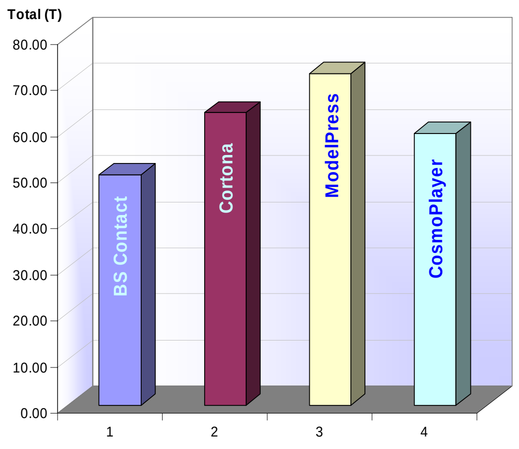

Fig. 5. Tools to visualize 3D environments (BS contact, Cortona, ModelPress, CosmoPlayer)

g) **Documentation of results and conclusions**. According to the results analysis and evaluation, the best FOSS tools to develop VR applications can be identified. These results can help make decisions about the best configuration to build a platform considering the software tools evaluated in previous stages.

Based on the evaluation results, the best combination of FOSS to develop an interactive virtual environment is shown in Figure 6.

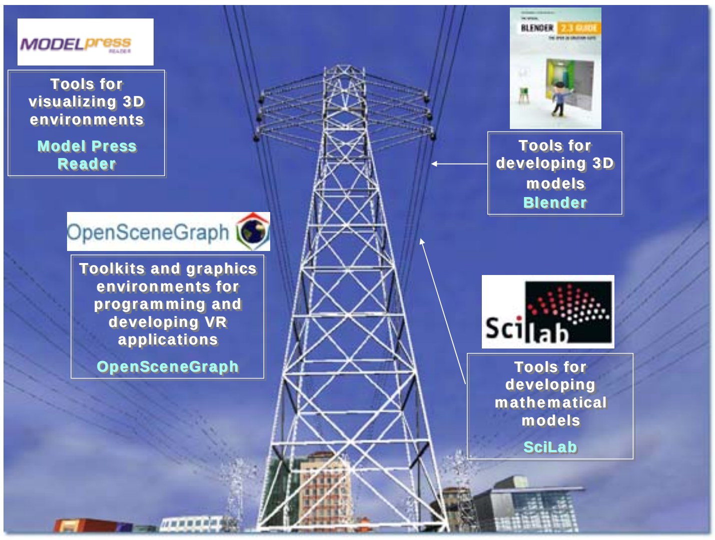
MODELpress
Tools for visualizing 3D environments
Model Press Reader
BLENDER 2.3 GUIDE
Tools for developing 3D models
Blender
OpenSceneGraph
Toolkits and graphics environments for programming and developing VR applications
OpenSceneGraph
Scilab
Tools for developing mathematical models
SciLab

Fig. 6. FOSS tools recommended for developing an interactive virtual environment

5. Development of a VR Application using FOSS

With the results we were able to know the best possible combination of software tools to develop a virtual power system tutorial. In this section we provide some snapshots to illustrate the way Blender was used to create 3D models and OpenSceneGraph to program and develop the VR application [22].

5.1 Tool for 3D Modeling

We propose the use of Blender (v 2.44) to develop the 3D models for the tutorial development. This FOSS tool runs on several platforms (Windows, MacOS, Linux, FreeBSD, Irix and Solaris). Furthermore, it has very important animation features such as: physics features and particles functions [23].

5.2 Tools for programming and developing VR applications

The use of OpenSceneGraph (v 1.9.8) is justified because it was the best VR FOSS tool obtained from the evaluation. One of its most important features is that many of its libraries were developed in C++ .Net. Additionally, it includes optimization methods such as: culling, different levels of detail, etc. Finally it presents many features for rendering and visualization [19].

5.3 Developing a Virtual Power System Tutorial using FOSS

The virtual power system tutorial allows a better learning experience than just reading or viewing photographs, because users can acquire knowledge using some of the activities proposed by Moshel and Hughes to improve users learning: constructivist, constructionist and situated [24]. Constructivist learning involves the exploration of prebuilt worlds and discovery, which is obtained with the exploration of the virtual world. Furthermore the users learn by means of situated learning because the students can have interaction with the virtual world using most of their senses to explore a power system in an immersive environment that gives him/her the sense of actually being there. This kind of VR applications with additional features can be further applied to personnel training in power plants, equipment maintenance, etc. without the risk of accidents or equipment damage.

In the developed tutorial the user/student can interact with all the objects situated in the virtual power system. The following figures show some power system tutorial screenshots [22]. For example, figure 7 shows an exterior view of a power plant; this particular application is focused on a fossil power plant where electricity is generated using petroleum. Other kind of power plants could be modeled and integrated into the system to learn about them. For example, the user can learn about differences between geothermal, hydropower and nuclear power plants.

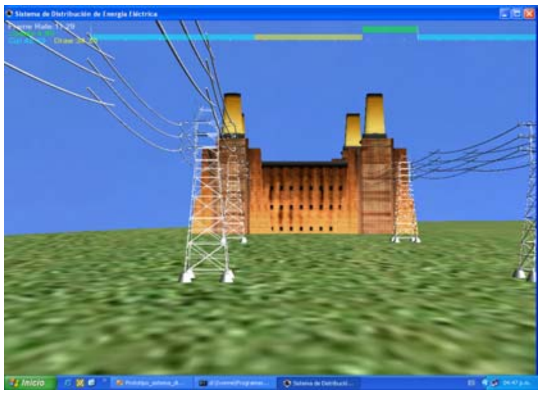

Fig. 7. A screenshot of the exterior view of a fossil power plant

In figure 8 a view of the transmission lines and towers is depicted, where the user can learn about different voltage levels for transmission (400, 230 and 161 KVs) and how the lines arrive at a power substation in order to reduce the level of voltage (34.5, 23, 13.8, 6.6, 4.16 y 2.4 kVs) for the distribution system.

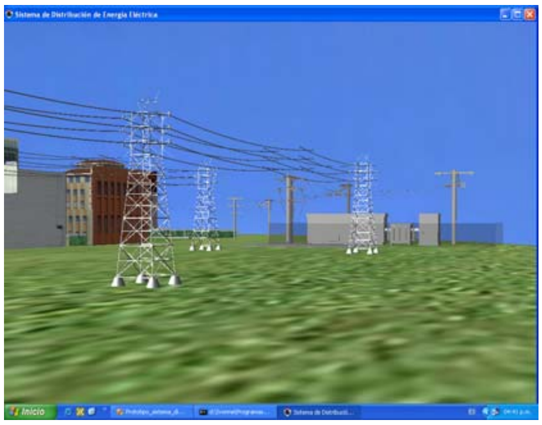

Fig. 8. Transmission towers and a power substation screenshot

Figure 9 shows a narrow view of the electrical substation. In the tutorial, the user can navigate and interact with the substation equipment to learn how voltage is reduced for its use in the distribution system and gain knowledge about different components of a substation: switchgears, power transformers, surge protection, controls, metering, etc.

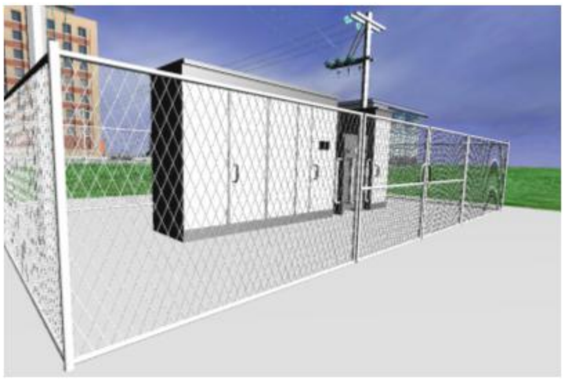

Fig. 9. A narrow view of the electrical substation

Figure 10 shows a screenshot of a city. In this part, the user can learn about the distribution of electricity at low voltages (440, 220 and 127v) and how electricity is supplied to final consumption centers such as: buildings, streets, houses, factories, etc.

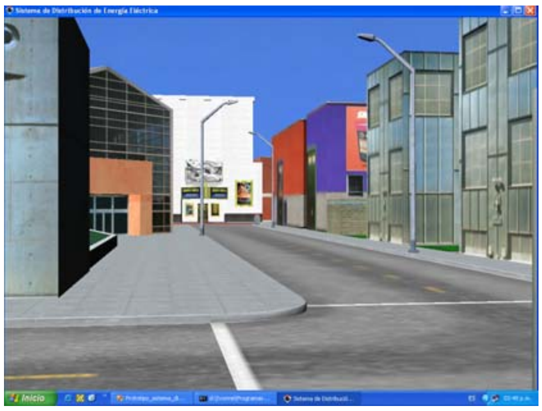

Fig. 10. A screenshot of the use of electricity in consumption centers

6 Conclusions and Future Work

The methodology suggested in this paper helped to evaluate objectively, the characteristics and functionality of commercial and open source software tools. A Multi Criteria Decision Making methodology based on a criteria set and weight assignment was useful to facilitate the selection of VR software tools. The Virtual Reality Group reduced time and effort in the development process of virtual reality systems. In so doing, we have obtained further information to improve and refine the methodology.

Future work includes, review of the evaluation methodology according to the fast changes in technology and keep track of software updates in order to obtain a reliable and updated evaluation. New criteria should be introduced to take into account factors derived from our implementation experience. For instance, geometric format compatibility, reliable documentation, functionality, and development support.

Additionally, in the near future real physical behavior will be added to the objects for simulation. We will develop mathematical models for behaviors using SciLab which is the top rated FOSS tool according to the evaluation. After that, we will add those mathematical models (for example gravity, inertia, weight, magnetism, etc.) to the power plant tutorial in order to have a more realistic environment.

References

1. Wikibooks. FOSS A General Introduction/Why FOSS?, http://en.wikibooks.org/wiki/FOSS_A_General_Introduction/Why_FOSS%3F#_ref-19, last modified 13 May 2007.
2. Pescatore, J., Commentary: Another worm, more patches, CNet News.com; available from http://news.com.com/2009-1001-273288.html?legacy=cnet&tag=nbs ; 20 September 2001.

3. Vaughan-Nichols, S. J., Can You Trust This Penguin?, ZDNet SmartPartner. http://web.archive.org/web/20010606035231/http://www.zdnet.com/sp/stories/issue/0,4537,2387282,00.html ; 1 November, 1999.
4. Miller B., Fuzz Testing of Application Reliability, http://pages.cs.wisc.edu/~bart/fuzz/fuzz.html, Last modified: Jun 2 2006.
5. Wheeler D., Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers, http://www.dwheeler.com/oss_fs_why.html, Last modified: April 16, 2007.
6. Open GL. The Industry's Foundation for High Performance Graphics, http://www.opengl.org/, visited on Jun 2007.
7. Open Inventor from Mercury, Overview, Open Inventor main features, 2002, http://www.tgs.com/pro_div/oiv_overview.htm, visited on Jun 2007.
8. Pérez M., Zabre E. and Islas E., Prospectiva y ruta tecnológica para el uso de la tecnología de realidad virtual en los procesos de la CFE, Instituto de Investigaciones Eléctricas, Cuernavaca México, Technical Report IIE/GSI/022/2003, 2003.
9. Islas E., Zabre E. and Pérez M., Evaluación de herramientas de hardware y software para el desarrollo de aplicaciones de realidad virtual; Boletín IIE, vol. 28, pp. 61-67, Apr-Jun 2004.
10. Islas E., Pérez M., Rodriguez G., Paredes I., Ávila I. and Mendoza M., E-learning Tools Evaluation and Roadmap Development for an Electrical Utility, Journal of Theoretical and Applied Electronic Commerce Research, Vol 2, pp. 63-75, Apr 2007.
11. Virtual Reality Scene Generator (VRSG) With Tracker Support, MetaVR, http://www.metavr.com/products/vrsg/immersim.html, visited on Jun 2007.
12. Irrlicht, Features, http://irrlicht.sourceforge.net/features.html, visited on Jun 2007.
13. Cult3D, Welcome, http://www.cult3d.com/, visited on June 2007.
14. Torque, Torque Game Engine SDK, http://www.garagegames.com/products/1, visited on June 2007.
15. DigiUtopikA, HorizoN Scene Graph, Portugal, http://www.utopika.net/horizon/EN/horizonsg.html, visited on Jun 2007.
16. OpenGL Performer, Overview, http://www.sgi.com/products/software/performer/overview.html, visited on Jun 2006
17. Panda3D, About Panda3D, What is Panda3D, http://www.panda3d.org/what.php, visited on Jun 2007.
18. Sun Developer Network, Java 3D API, http://java.sun.com/products/java-media/3D/, visited on Jun 2007.
19. OpenSceneGraph, Homepage, http://www.openscenegraph.com/index.php, visited on Jun 2007.
20. Web 3D Consortium, X3D Developers, http://www.web3d.org/x3d/, visited on Jun 2007.
21. VR Juggler, Juggler Suite of Tools Project Overview, Iowa State University, http://developer.vrjuggler.org/, visited on Jun 2007.
22. Avila I., Desarrollo de un prototipo de realidad virtual como tecnología de apoyo para capacitación de personal en CFE, Tesis de Licenciatura, Agosto 2005
23. Blender, Features and Gallery, http://www.blender.org/features-gallery/, visited on Jun 2007.
24. Moshel M and Hughes C., Virtual Environments as a Tool for Academic Learning, in K. Stanney (Ed.), The Handbook of Virtual Environments Technology, Erlbaum, Mahwah, NJ, 2002.