X3D and Java Fusion in a Medieval Fantasy Game

José Carlos Miranda[1,2], Nuno Martins[2]

[1] Faculdade de Engenharia da Universidade do Porto, Portugal
[2] ESTG, Instituto Politécnico da Guarda, Portugal
jcmira@ipg.pt, martins@sal.ipg.pt

Abstract. The presentation of 3D contents on the Internet is very interesting and presents many potentialities, especially if there is a great amount of interactivity within the applications. The X3D is a good format to use in the creation and visualisation of 3D contents, but the interactivity, which is possible to develop without using an external programming language, is limited. A prototype of medieval fantasy game has been developed, trying to demonstrate the integration capacities of the X3D format and the Java programming language. The integration of these two technologies was possible through the programming interface SAI (Scene Access Interface) that showed a good outcome, allowing a considerable increase of the power of interactivity necessary for this type of applications.

Keywords: Virtual Worlds, 3D modelling, X3D, Java, Xj3D, RPG game, Human-Animation.

1 Introduction

The 3D contents visualization on Internet has great potentialities in different areas like architecture, medicine, education, entertainment, amongst others. One of the most important standards that emerged in 1995 and that allowed the three-dimensional world visualization and interaction on the Internet was VRML – Virtual Reality Modeling Language [1], based on the Open Inventor technology of Silicon Graphics. The last version of this format is from 1997. Since then this language naturally evolved to another one, called X3D – eXtensible 3D. X3D is the new standard for 3D content distribution on Internet and its specification was approved by ISO in December 2004, and since then many revisions have been taken place [2].

It is important to refer to the great need for interactivity within this type of presentation. X3D is appropriate for the creation of 3D content visualization, but the interactivity that is possible to create without using another external programming language is limited. Due to the huge quantity and diversity of objects that exist within X3D, there are many potentialities to explore when using an external programming language. An adequate programming environment for this manipulation is Java [3], [4].

To assess the potentialities of X3D and the external programming language Java integration, an application has been developed for the specific area of entertainment.

A Role Playing Game (RPG) was developed once this application requires a big amount of interactivity between the player and the virtual world. The appropriate programming interface to establish the connection between these two technologies is called Scene Access Interface (SAI) which allows the exchange of events between the X3D scene and the Java application, producing much more powerful interactions.

To overcome the lack of information on this recent study-area, a Website has been built, where some examples of this integration can be found. This site [5] is recommended to users that already have some knowledge of X3D or VRML, since the main objective is to explain the integration between X3D and the Java language.

In section 2 some subjects related with the virtual world of the developed prototype are presented. The programming interface that allows the communication between the X3D and the Java language and the considered philosophy related to the game programming are described in section 3. Section 4 represents the results and section 5 includes the conclusions and a few orientative lines towards future work.

2 Virtual World Creation

In this section there are references to some theories about the creation of the virtual world. The first task to carry out is to put the plans on paper, in a textual format or by drawings. These drawings are going to be a kind of storyboard and the starting point for the creation of the virtual world.

2.1 Planning and Interaction Strategies

Some aspects that the creator should think of, are the tasks that the player will be able to carry out in that world. A virtual world where the player simply can walk will not be necessarily interesting. The basic idea underlying the creation of tasks that players will carry out, is that once they are finished the player will receive some kind of reward. This was taken into account in the developed game creating missions that the player should carry out and that lead to some rewards.

According to Roehl [6] it is also important that each scenario of the virtual world has different characters that the user interacts with, because the human being is social by nature. These new characters should have some kind of "artificial intelligence" creating a more realistic world. In the world built for the game there are some places that the user will be able to explore, as well as characters with which he can interact and communicate (Fig. 1).

Fig. 1. Drawings of different scenes and some characteres of the game

Human beings like to discover and learn things. The user will feel a huge satisfaction upon discovering the "tricks" necessary to reach an objective like opening a door, finding several places that looked inaccessible, solving puzzles, especially if the user is rewarded after each discovery. A way to maintain a user "prisoner" to a virtual world is by rewarding him in different phases of the game, according to the task that he is doing. According to Shneiderman [7] a way to captivate the users of a virtual world is through the attribution of scores. In the implemented game, these scores were replaced by the characteristics of the character, whose value will increase during the game while the player is completing the missions.

Another characteristic of the real world that equally is important in a virtual world is the danger sensation. Many activities in the real world become more interesting due to the fact of the existence of danger, for example to ride in the russian mountains, doing bungee jumping, or climbing mountains. Fig. 2 shows some examples of weapons that had been imagined for the battles of the present game. In these battles there is the possibility that the player loses life points, or even dies. This sensation of danger is doubtlessly very important in a virtual world specially to turn it realistic and less monotonous, plus the fact that the human being likes to run risks and defeat adversaries.

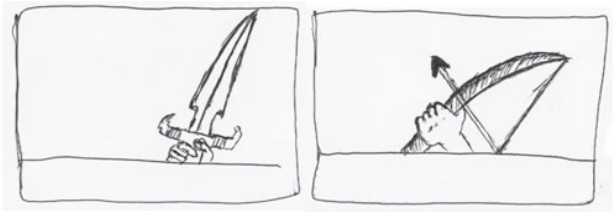

Fig. 2. Drawings of several weapons

2.2 Scenes and 3D Objects

After planning, the next task was the creation of the scenes and the 3D objects. A modelling tool should be used which controls the geometry at the vertices, edges and polygons levels (Fig. 3) not simply using the basic geometrical primitives: cube, cone, sphere, cylinder, etc.

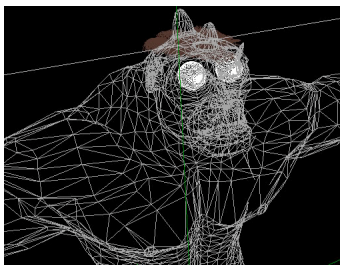

Fig. 3. Polygons of a X3D character

For the creation of the objects that compose the virtual world two modelling tools were used: the Cosmo Worlds and the Vizx3D.

The Cosmo Worlds can be defined as a world builder [8], having been used in the creation of some objects and respective scene positioning, because it includes precision tools that allow a huge amount of control. This is a VRML world builder software, that implies a subsequent conversion to the X3D format. On the other side, the Vizx3D [9] is a specific X3D world builder and, in our opinion, perhaps, the strongest software for the X3D model creation at the moment this project was developed, for it includes enough powerful modelling tools and it was used for the construction of more complex geometries. It was also within this software that the materials and the textures were attributed to the objects, as well as the creation of all animations. It was also in Vizx3D that the addition of environments and some simple interactivity were created.

Previously modelled objects in 3D Studio MAX were also used and later converted to the X3D format, using conversion tools, like the AccuTrans 3D [10].

In the creation of a virtual world for the internet, the optimization process of a 3D scene is very important, because it will allow for bigger loading speeds of the X3D files to the browser, as well as improve it's navigation performance by increasing the rendering speed of the 3D scene.

In the construction of the virtual world several optimization techniques were used, like:

1. Using the less ammount of polygons as possible, eliminating those that will always be invisible, despite of the point of view. According to Hartman [11] the planned worlds to be seen on the Internet should not have more than, approximately, 1000 visible triangles at every single moment.
2. Using materials and textures to create a higher complexity of surfaces effects, without using too many polygons. According to Ballreich [12], a texture should have the smallest possible size and should be squared.
3. Reusing textures and sounds whenever it is possible, with reduced sizes.
4. Limiting the use of lights to maintain the rendering speed at reasonable levels.
5. Structuring prudently the hierarchy of the objects that compose the scene, permitting the browsers to perform the visibility calculation in a more efficient way.
6. Applying characteristics of computer graphics that help to increase the efficiency, for example, different levels of detail (LOD), Billboards, Inlines and Collisions Detection.

These techniques are described, in detail, by Miranda [13].

3 X3D and Java Fusion

While X3D is an adequate language for the creation of three-dimensional objects, the Java language can be used to add complex behaviours to those objects. The integration of these two technologies is possible through SAI - Scene Access Interface [14]. This programming interface has some functions that permit the manipulation of the X3D scene through the utilization of the Java programming language.

When this work was developed, the only browser that allowed the viewing of X3D scenes manipulated through Java is Xj3D [15], a stand-alone browser entirely developed by the Web3D Consortium and that has been used as the only tests platform that integrates X3D with Java [16], [17].

In the following section the integration of X3D with Java will be explained, using the SAI programming interface.

3.1 SAI – Scene Access Interface

SAI is the programming interface (API) used to establish the communication between the X3D and Java. Through this connection it was possible to exchange events between the X3D nodes and the Java application.

The graphic interface components of the developed game in this project are the X3D browser and the Java application. While the browser allows the user to navigate in a three-dimensional environment, the 2D interface created in Java offers some tools that enable the interaction and manipulation of the objects of that 3D environment. In this way, pressing a button will send an event for the X3D scene, changing the characteristics of an object.

SAI allows two kinds of access to the 3D scene, one named by internal access and another one by external access. Both cases can be used together, using the advantages of each one of these approaches.

In the internal access, there is a script node inside the X3D file that will make the connection with the Java class. So, it is the X3D scene that, after being loaded in the X3D browser, will call the Java class where the code exists that will perform the intended alterations in the 3D scene (Fig. 4).

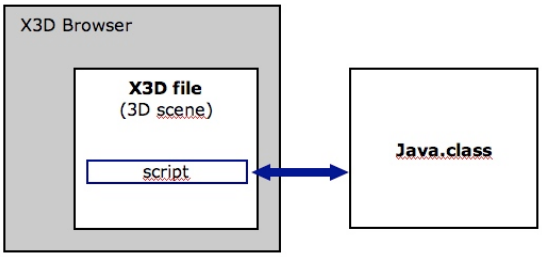

Fig. 4. Internal access

Within the external access, the Java class will create a 3D window and, after that, will load the X3D scene to the previously created 3D window (Fig. 5). In this type of access it is the Java application that will load the X3D file. This becomes very useful when an application with two windows is needed, one composed by the 3D scene and another one by the controlling interface, like the developed prototype in this work.

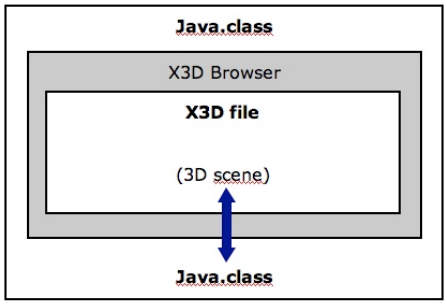

Fig. 5. External access.

In the following section the considered philosophy related to the game programming and some used methodologies will be explained.

3.2 Game Programming

Whenever the interaction and manipulation of objects on the X3D scene through the Java programming language are necessary, it will be necessary to access to those objects properties. After this access to those properties it is possible to proceed to its manipulation. To better clarify this integration process, one case of the implemented game – the battles of the player with the existing monsters – will be explained.

To simulate the monster's field of view, a proximity sensor was placed on the area where the monster will be found (Fig. 6).

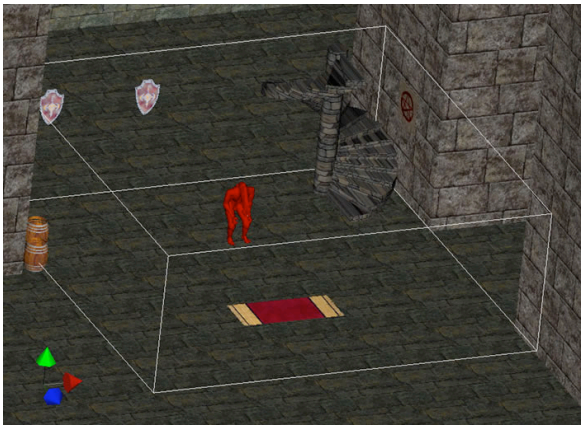

Fig. 6. Proximity sensor in the area where is found the monster

Whenever the player enters the proximity sensor area, an event is triggered. The Java application is automatically notified of that event occurrence, setting the necessary actions, which in that case, would be the beginning of the monster attack.

Movements of both legs and arms of the monster, as well as the movements of rotation and translation in the direction of the player make the produced animation.

To control the monster translation movements towards the player, the basic principle of animation was used. A position interpolator and a clock were applied. The position interpolator has uniquely two keyFrames: an inicial keyFrame, made by the initial coordinates of the monster and the final keyFrame, that will always be updated with the actual coordinates of the player while he is moving (Fig. 7).

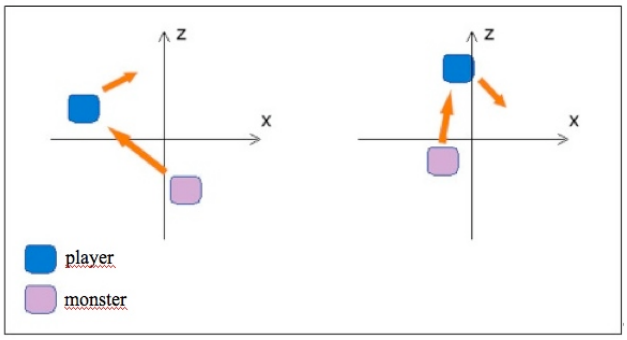

Fig. 7. Monster translation movements

To control the rotation movements of the monster in order to be face to face with the player all the time, the basic principles of analytic geometry were used. The schematic representation is showed in Fig. 8.

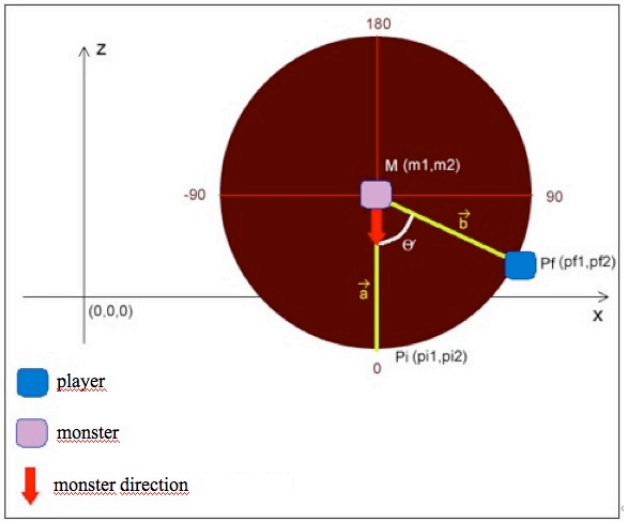

Fig. 8. Schematic representation of monster rotation movements

Initially, the monster is turned into a specific direction, represented by vector $\vec{a}$. The moment the player enters the field of view of the monster is represented by

position Pf. The monster is in position M and will have to have a rotation θ to face the player. It will be necessary to calculate the angle between vectors $\vec{a}$ and $\vec{b}$ in order to know the rotation angle of the monster. The presented formulae in equation 1 was used to determine the inner product of vectors $\vec{a}$ and $\vec{b}$.

$$\theta = \arccos \frac{\vec{a} \times \vec{b}}{\left\|\vec{a}\right\| \times \left\|\vec{b}\right\|} \quad (1)$$

$\left\|\vec{a}\right\|$ is the norm of vector $\vec{a}$, $\left\|\vec{b}\right\|$ is the norm of vector $\vec{b}$ and θ is the angle between the two vectors.

The calculation of the angle θ will have to be continuously performed and applied to the coordinate system of the monster, ensuring that the monster will always be facing the player.

Next it becomes necessary to control when the player is near the monster, the moment when the monster will start his attack, causing damage. To control this situation, a proximity sensor is placed near the body of the monster. This proximity sensor wraps the monster and is placed inside the group that is part of the animation of that monster, so that when the monster is moving, he transports the proximity sensor with himself (Fig. 9).

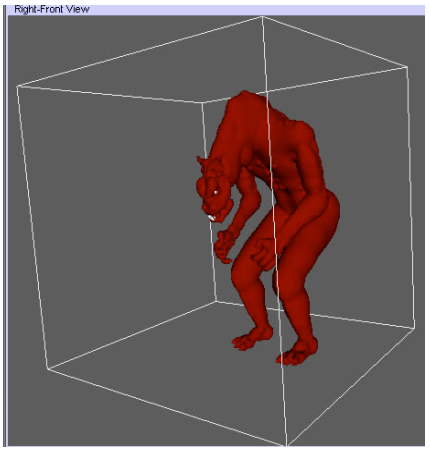

Fig. 9. Proximity sensor of the monster

From the moment that the player enters the defined region by this second proximity sensor, another event will be triggered indicating the program that the attacks can begin. This way, the attack animations of the both monster and the player will start. The damage caused by these attacks is controlled by specific formulae that are based in the physical characteristics of the player, the characteristics of the equipped weapon and the monster characteristics.

A battle finishes when the points of life of the player or of the monster get to zero, causing dead. This is the way through which the battles and both the monster -and player damage are controlled.

After examining this example it is possible to better understand the philosophy underlying this game programming. The created world is made up by a large number of sensors, for example, proximity sensors, touch sensors, visibility sensors, and others. This way, each sensor will notify the player's interactions to the Java application that will trigger an action in the 3D world specific object. Several programming code examples of this game can be found in the Website [5] developed in the field of this project.

4. Results

To assess the potencial of X3D technology and the Java programming language integration, a medieval fantasy RPG prototype has been developed. The graphic interface is composed by two components: the X3D browser and the Java application (Fig. 10). While the browser allows the user to navigate in a three-dimensional environment, the interface created by Java offers several tools that enable the interaction and manipulation of the specific 3D environment objects.

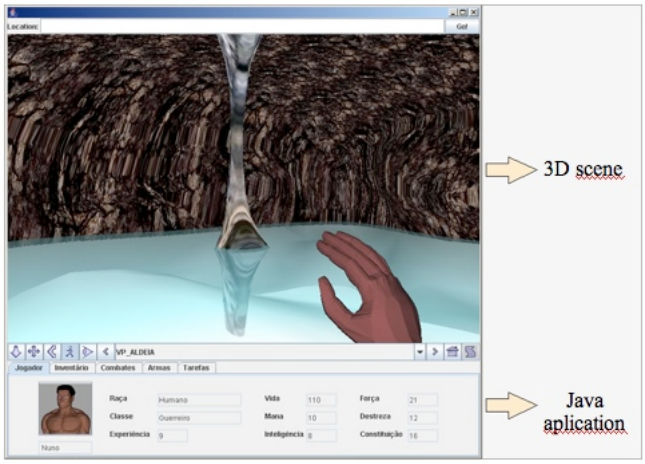

Fig. 10. Graphic interface with two components: 3D scene and Java application

In the beginning of the game the user must select and configure the characteristics of his character, like the race, colour, strength, dexterity, intelligence, etc. In this initial phase there are many user interaction possibilities within the three-dimensional scene. Through the Java developed interface it was possible not only to manipulate some characteristics of the character, but also to add new elements in the 3D scene. After this, the player will be able to explore a virtual world in a medieval fantasy environment. The created world is made of different environments, such as mountains, caverns, villages, castles and forests. The player should carry out some

tasks and explore the world the way that he wants. There is not a linear way to play the game, and the player has a big range of freedom in its exploitation. He will be able to decide with which character he speaks, which monsters attack, when he should flee, etc.

To play sounds in a X3D scene it was necessary the installation of OpenAL [18], a library used in applications and games that require audio 3D.

In Fig. 11 several images of the prototype developed in this project are presented.

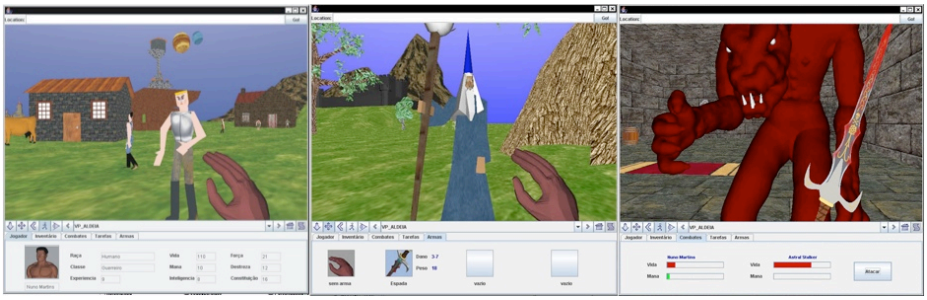

Fig. 11. Images of the developed game

A Website where several programming code examples of the X3D and Java integration can be found, has also been developed (Fig. 12).

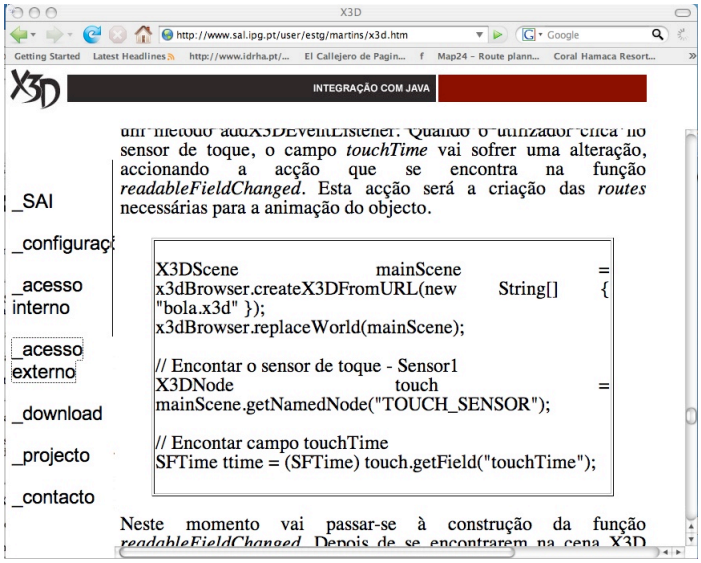

Fig. 12. Web tutorial with programming code examples.

5. Conclusion

In this work a decision was taken to use open source 3D technology – X3D. The choice of the X3D instead of the VRML seem logic, considering that X3D came after VRML, containing, amongst other advantages, a much stronger scene codification. The objects of the 3D scene are, actually, in XML code, instead of text, as it used to happen with VRML.

The first objective of this project was not the creation of a game, but the creation of an application that would demonstrate the capacities and potencialities of the conection between the X3D world and a Java application. The implementation of the game demonstrates many characteristics of this integration. The capacity of using an external programming language to add specific behaviours to 3D scene objects raised the power of interactivity turning the X3D much more powerfull.

Although the prototype developed was directed to the specific area of entertainment, the integration of these two technologies can be used in different fields of application like architecture, medicine, teaching, publicity, and others.

The used browser – Xj3D – works properly with the fusion of the X3D with Java and supports most of the characteristics of X3D. But, it still represents some instability due to the fact of still beeing in the phase of development. It is expected that a new version of this browser will be soon released, with many of its problems solved.

It is still important to mention the need of extra time for the correct configuration of the computer, in order to allow the comunication between X3D and Java.

The main dificulty in the development of this work was the lack of information about this theme. The existing information is not very clear, specially when it concerns the integration of X3D and Java. In order to overcome this lack of information in this recent area and, in some way, contribute to the development and dissemination of this new field, a website with several practical examples has been created. This website is online and can be consulted on the oficial site of Xj3D, in the tutorials section [19].

5.1 Future Work

The development of a game is not an easy task once it requires a teamwork action, each element with a specific function. The future work will be directed to the creation of new game-areas and the implementation of new missions and objectives for the player to reach. Adding inteligence to the characters, making a more realistic world, corresponds to another future aim.

During this work it was impossible to use SAI in on-line application, due to the fact that the Xj3D is a stand-alone browser. Thus another objective will be to make the created game available on the Internet.

Acknowledgments. Special thanks goes to Alan Hudson, Xj3D project manager and code programmer, for his unconditional support and valuable comments.

References

1. The Virtual Reality Modeling Language, Dezembro 2007 http://www.web3d.org/x3d/specifications/vrml/VRML1.0/index.html
2. X3D and Related Specifications, Dezembro 2007 http://www.web3d.org/x3d/specifications/
3. Harney, J., Blais, C., Hudson, A., Brutzman, D.: Visualizing Information Using SVG and X3D: X3D Graphics, Java and the Semantic Web. In: Geroimenko, V., Chen, C. (eds.) 2005. LNCS, vol.5555, pp. 10-10. Springer, Heidelberg (2005)
4. Rudolph, M., Zhang, Y. J.: A universal java interface to native 3D rendering platforms using multiple scenegraph representations. 7th International Conference on Computer-Aided Industrial Design & Conceptual Design. pp. 413-417. IEEE Xplore (2006).
5. X3D – Integração com Java, Dezembro 2007 http://www.sal.ipg.pt/user/estg/martins/x3d.htm
6. Roehl, B.: Playing God – Creating Virtual Worlds with Rend386.Wait Group Press, EUA (1994)
7. Shneiderman, B, Plaisant, C..: Designing the user interface – Strategies for Effective Human-Computer Interaction. Addison Wesley, EUA (1998)
8. Cosmo Worlds 2.0 gives 3-D Web authoring a boost, Janeiro 2008 http://www.infoworld.com/cgi-bin/displayArchive.pl?/98/20/i12-20.81.htm
9. Media Machines - Developer Resources and Forums, Janeiro 2008 http://www.mediamachines.com/developer.php
10. AccuTrans 3D by MicroMouse Productions, Janeiro 2008 http://www.micromouse.ca/
11. Hartman, J., Vogt, W.: Cosmo Worlds 2.0 User's Guide. Silicon Graphics, Inc, EUA (1998)
12. Ballreich, C.: Late Night VRML 2.0 with Java. Texture Map Creation. Ziff-Davis Press, Macmillan Computer Publishing (1997).
13. Miranda, J.C., Sousa, A.A.: Urbanismo e Ambientes Virtuais – Divulgação e Discussão na Comunidade. Virtual - Revista Electrónica de Visualização, Sistemas Interactivos e Reconhecimento de Padrões - ISSN: 0873-1837. (2000)
14. Extensible 3D (X3D) - Part 2: Scene Access Interface (SAI) ISO/IEC 19775-2:2004, Janeiro 2008 http://www.web3d.org/x3d/specifications/ISO-IEC-19775-X3DAbstractSpecification/
15. Xj3D - Java based X3D Toolkit and X3D Browser, Janeiro 2008 http://www.xj3d.org/
16. Hudson, A.D., Couch, J., Matsuba, S.N.: The Xj3D browser: community-based 3D software development. ACM SIGGRAPH 2002 conference abstracts and applications, pp. 327-327. ACM Press, Texas (2002)
17. 3dtest - 3D Temps reel et interactive, Janeiro 2008 http://www.3d-test.com/interviews/xj3d_1.htm
18. openAL - Cross-Platform 3D Audio, Janeiro 2008 http://www.openal.org/
19. Xj3D Tutorials - Portuguese SAI Tutorial, Janeiro 2008 http://www.xj3d.org/tutorials/index.html