DWeb3D: A toolkit for developing X3D applications in a simplified environment

First Author
Author 1
Author 1

Second Author
Author 2
Author 2

Third Author
Author 3
Author 3

ABSTRACT

The X3D standard is open, supported by an international consortium, mature and in constant development, but with a low adoption rate. In this work, X3D qualities and problems are discussed and correlated with other solutions. In this process it was detected some necessities in current applications and the complexity of X3D to deal with these issues. As an attempt to demonstrate that the complexity of X3D in some aspects may be reduced, the DWeb3D toolkit was built. DWeb3d is a toolkit to help the development of dynamic X3D applications, showing that it is possible to simplify the development process, possibly increasing the access to developers in this area. The toolkit provides tools to deal with publishing, synchronism, interactivity, multiple users management and disk persistence.

1. INTRODUCTION

One of the first solutions for 3D environments on the web was VRML [18]. The VRML was well received by the Web community, but its adoption was limited, and today much of what was created in VRML is disappearing or is outdated. In the early 2000's to replace the VRML and try to overcome its limitations, X3D was created [19, 6]. But even more than the VRML, the general community has not adopted X3D as the main 3D format. Several years after its creation, and even with all the qualities it has, its adoption is still very shy and there is no prospect of a short-term change to that reality. One goal of this research is to analyze the possible causes of this fact and propose methods that can change this.

Parallel to these technologies, there are attempts to provide 3D content using specific plug-ins. The most used currently are the Flash [1] and Shockwave 3D [7]. Flash has several limitations from its basic architecture designed to deal with animations and videos, not to 3D interactive content. Despite that, this tool has a much higher acceptance rate to display 3D on the Web than X3D.

According to Toni Parisi [14], commercial developers are expressing increasing interest in exploring real-time 3D in web applications, as a way to increase product value and to provide information in a significative manner. However, X3D and other 3D standards are mainly focused on 3D content transmission, and not on the application that involves them. Industry needs agile development environments for creating client-server applications for an agile and highly interactive web.

While X3D is an open standard, powerful, extensible and has a consortium, which maintains and supports it, it does not provide a very simple environment of quick development. This factor is probably a major cause of its low adoption rate. To analyze this phenomenon, it is possible to draw a parallel with Flash and analyze why it has greater acceptance. To that end, we may observe what Flash has that X3D does not:

1. A well-accepted Graphical Editing Interface [5] and a programming language known by many programmers.
2. A large base of programmers with knowledge of the language.

Once the programmers are who decide which technology will be used for their projects, it is important to offer resources and try to popularize the environment to programmers. To achieve these goals, it is suggested the development of a toolkit that simplifies the process of development and prototyping, thereby accelerating the acceptance of the technology.

In order to evaluate the usefulness of X3D it is important to delineate features that would be needed for applications in the real world. We can name a few of them with the reasons why they are useful and then explain how they can be achieved today with what X3D offers.

- Creating and manipulating a 3D scene [3, 2] – The objective is achieved through the X3D definition.
- Creating events that respond to user input [14, 4] – The objective is achieved through specific structures and the possibility of using ECMAScript and extensions of the standard.
- Loading and data persistence [10, 13] –This goal is achieved through complex routines in ECMAScript and web services that have a session control and state.
- Interaction with web GUI (Graphical User Interface) [14, 11] – Can be made through ECMAScript calls, which requires knowledge of the internal structure of X3D.

- Interaction between users accessing the same scene [3, 20] – It is possible to be achieved through complex solutions involving ECMAScript, code on the server and session control.
- Integration with other applications [8] – Can be done via webservices and ECMAScript.

It can be noted that X3D offers a good support for the first two itens, but needs complex mechanisms to support the others. The objective of this research is to develop tools for X3D and a demo application to show that the technology is flexible enough to support that features, required by the developers, thereby increasing the possibilities of adoption of the standard.

The paper is organized as follows. In Section 2 we present works related to X3D limitations, specially regarding the features mentioned above, namelly, collaboration, interaction with the Web GUI, integration with other applications, and persistence. In Section 3, the DWeb3D structure is presented, and in Section 4 it is shown how these features are treated in the proposed toolkit. Some examples are presented in Section 5, and conclusions in Section 6.

2. RELATED WORK

The subject of toolkits and creativity is widely exploited in [9] and is of fundamental importance for the development of this research. For a new area to develop it is necessary to give developers the ability to test their creativity through a cycle of prototype, testing, analysis, prototype, etc. This cycle provided in the methods of incremental development allows the developer to check the progress made in short periods of development. The discovery of ideas is a key aspect in the concept of focusing on creativity as it allows the developer to see if something that works has been created without the great burden for the project.

There are some classic cases in which the presence of a structure for the abstraction of lower layers was a key factor for the success of a technology. The best-known example is the Web itself. The advent of HTML enabled rapid adoption of the environment as a repository of information, enough for developers to write their text in a structured and link them to others. Another familiar example is Flash, as its architecture allowed for the popularization of more dynamic pages, with interactive animations or movies.

To achieve these goals within a project with complex requirements there is a great need for code reuse. It is also necessary to eliminate unnecessary layers in low-level operations that do not affect the application. At this point a toolkit is extremely useful because it encapsulates several job applicants and allows the developer to focus on what is really wanted. The use of toolkits also promotes good programming practices, since they already carry a concept of decoupling, organization code and modeling.

This work presents the design and development of a toolkit for developing X3D applications covering the following topics:

- Collaboration between users;
- Interaction with the Web GUI;
- Integration with other applications;
- Data persistence.

In addition to implementing the above features, it is also purpose of the toolkit to provide the necessary abstractions for application developers using X3D to develop these applications faster and with lower learning curve.

2.1 Limitations of the X3D

Just as VRML, X3D describes an entire 3D scene, which in turn is interpreted by a browser itself or a plug-in for an existing web browser. Like VRML, X3D also proposes to make the control of the behavior of entities in the model within the file description of the scene. With this, it makes self-sufficient models. As in the new web applications, it is also expected that the environment of 3D applications offers collaboration between users, and this is not trivial to achieve in the model proposed by X3D. We can understand the world of X3D as several isolated 3D worlds, and they do not speak with each other. If you want to make a world talk to another or set one world where many users interact, it is necessary to make use of various complex solutions.

X3D also does not deal very well with too complex or too large scenes and this is due to its design. It must pass all information from the scene at once to carry it. This method makes it expensive for very large scenes, unviable for the web environment. Another limitation detected is little support for complex physical simulations. Only in 2006 X3D has started to provide a sub-set for physical simulations [17].

2.2 Collaboration

One of this project's goals is to enable the interaction of two or more people in a 3D scene [12]. This type of interaction is fairly common in multiplayer games because one player's action affects the online experience of others. Games and visualization applications created the concept of virtual world, i.e., a persistent and dynamic environment where several avatars coexist and interact, thereby altering the environment and the perception of reality. In late 2004 and early 2005 two products that have high expression of virtual environments were launched, World of Warcraft[1] and the Second Life[2]. World of Warcraft has achieved great success, with about seven million users. Second Life has a very different proposal, because it allows its users to freely create the contents of the virtual world. Its greatest merit is to offer the business world a virtual environment with great acceptance.

Even being a success and the current major reference in terms of virtual environments on the Web, Second Life does not have a good integration with web pages and has much of its infrastructure controlled by the company that created it, which limits the expandability and derivation of this product in new proposals. Even as a powerful product, it lacks the flexibility of an open standard, and the possibility of expansion that X3D has.

2.3 Interaction with the Web GUI

Among the desired characteristics for new web applications, interaction with other components of the GUI is listed. It is important because the technology of web pages is widely accepted and the basis of what the Internet is today.

The chat world was one of the first applications to spread the web using mostly the VRML, but there are some versions with X3D. This is a chat environment where you can visit the virtual world and

[1]http://www.worldofwarcraft.com
[2]http://www.secondlife.com

talk with others who are visiting the same world. Initial versions were little more than a page containing a 3D scene and a chat client located on the same page. There was no collaboration or interaction with the scene unless the navigation, there was an avatar or ways of seeing the other participants. We can cite the ABNet[3] as a client of this era that still exists today. This particular technology works via a Java client and ActiveX scripts that interact with the X3D/VRML browser and update the necessary messages. It still uses a separate dialog box to see the conversation, and interaction is limited to see the avatar of another guest and talk to him in the dialog box.

Although it was quite interesting for that time and several scenes have been created for this concept, this proposal includes several limitations such as inability to handle the scene and create greater interaction within it. But even with the limitations, this was one of the most adopted uses of VRML, because it provides a package that includes some of the features desired by developers and that were difficult to achieve with pure X3D/VRML. In addition, it has some level of interaction with the web pages, since it is inside the page, and the chat is a component of that page.

2.4 Integration with other applications

Although they are usually limited to their browser and the virtual 3D environment, X3D scenes may have to interact with data and events from other applications to be more interesting.

The Vivaty [4], developed by former MediaMachines, is a browser which main quality is the ability for users to use custom avatars. The product uses a concept called AJAX3D [14]. Using techniques of asynchronous calls via scripts it enables that the scene and web UI components interact. This product does much of its operations using the X3D standard, but not all, because you need a specific plug-in for the browser, and it uses a proprietary protocol to communicate with the server. Despite these limitations, it is an interesting product because it presents a new approach for X3D solutions, and can even integrate with popular community sites like FaceBook.

2.5 Persistent state

Another necessary feature is to allow the persistence of state between multiple instances of an X3D world. One example is the work presented in [3], which, among other points, explains that the persistence of the avatars position is important because it creates an immersive world where perception of the observer is placed inside the virtual representation.

The persistence in database is also important for allowing multiple servers have access to a common environment in a simple way, and a more efficient storage of the virtual environment. But we could not find any work proposing this specifically for X3D. Thus the implementation of DWeb3D was made through the use of generic techniques of object-oriented database as described in [10]. With these techniques you can store the entire universe of application objects and restore them when needed.

3. DWEB3D

This research proposes to develop a toolkit that reduces the learning curve and time required for the development of X3D applications with the features mentioned above (collaboration, interaction with the Web GUI, integration with other applications, and persistence), while promoting the rational reuse of code. Following this line of reasoning, we have developed a toolkit to assist the development of collaborative applications using X3D and an application that makes use of this toolkit as a proof of concept. A toolkit is defined as a set of development libraries that deal with various challenges encountered in developing an application with certain characteristics [9].

For the development of the toolkit C# in .NET platform was used. For the case study, a rendering plug-in for Unity3D[5] was developed. The Unity3D is a graphics engine that aims to be modularized and extensible, supporting a wide range of formats for import. The choice was made because the Unity3D has a scripting language based on C#, it has a plug-in for use on the Web, and it enables a richer interface than using an existing X3D browser, since with the X3D browser, the interface would have to be done in ECMAScript and would be more complex.

3.1 DWeb3D Structure

The toolkit is divided into modules organized by function. Each module represents a dynamic library. This enables a more rational use, since if not all the functions are needed, it is not necessary to add all the dynamic libraries to the application (Figure 1).

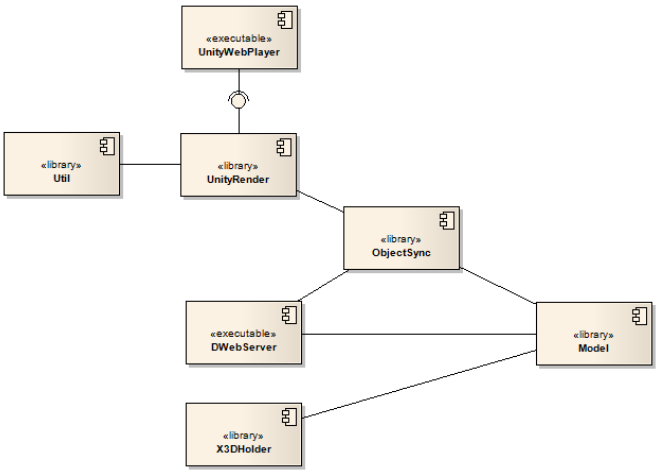

Figure 1: Components of DWeb3D.

3.2 Model

This constructor is the base of the toolkit's operation. It contains the graph and all classes that represent the structure of X3D. Also here are the core classes for reading and writing X3D files, such as X3DRepresentation and X3DHolder. X3DRepresentation is the main class for manipulation of the graph. It is responsible for reading X3D files and writing the graph in XML format. X3DHolder is the class responsible for the use of the structure in an asp.net environment.

3.3 ObjectSync

This is the package responsible for synchronization of multiple objects across the network graph. To use ObjectSync it is only necessary that the target object be of type ObjectSync or inherit it and it be recorded in the sync manager. The interface defines which properties are synchronized and allows the synchronization shooting by a single function call. The importance of this library is that it

[3]http://kimballsoftware.com/abnet/
[4]http://www.vivaty.com
[5]http://unity3d.com/

handles the entire network layer and makes it very simple to make synchronization.

3.4 DWebServer

Library responsible for creating a server. It only receives the synchronization and forwards it to other clients. The main feature is that it makes use of classes contained in ObjectSync that in turn has capabilities to forward synchronizations. It keeps a copy of the graph itself. Remember that synchronization is always made from client to server.

3.5 Unity Render

Unit Redender is a plug-in which lets you use the X3D graph within the Unity3D environment. It creates Unity objects through representations of the X3D and keeps them synchronized with the objects of the graph.

3.6 Util

This package contains code for internal use of the toolkit, such as conversions and other code reused within the toolkit.

3.7 Unity WebPlayer

Executable scripts developed for DWeb3D, to serve as a demonstration of the interaction of the plug-in with the Unity game engine and of how to put the plug-in to work. The WebPlayer will be used as a substitute for a browser.

4. IMPLEMENTATION OF THE GOALS IN DWEB3D

In order to illustrate the suitability of the toolkit to the proposed objectives, it is presented how it can solve existing problems and how its structure was designed for these cases.

4.1 Collaboration

To meet the goal of facilitating collaboration, some factors are important:

- Facilitate the creation of a server.
- Enable the synchronization of the positions of objects between the views of several clients.
- Enable the creation of new objects via the client interface.

To create a basic server, the necessary code is simply:

```
public void StartDefaultServer()
{
   // Create the server on port 1717
   SceneServer s = new SceneServer (1717);
   // Setting and creating the basic scene
   s.SceneGraph X3DRepresentation = new ();
   s.SceneGraph.CreateBasicScene ();
   // Setting this to be the default server
   s.IsDefault = true;
   // Adding the server to the scene manager
   DefaultServer = s;
   SceneServers.Add (s);
}
```

It may be noted that there are four operations in the code above. Creation of the object, definition of the default scene (an empty scene), definition that it is the default server (we may have several on the same machine) and registration of the server in the scene manager.

The various low-level functions like create socket, process threads, and create loops of reading required for a server are embedded in DWeb3D and therefore the developer doesn't need to deal with them. Thus the size and complexity of the code produced is greatly reduced.

To deal with the synchronization ObjetSync module was created. To use it, it is simply necessary that objects to be synchronized implement a specific interface. This interface lets you define which fields we want to be synchronized on the object and thus can promote the timing by a simple function call as shown in Figure 2.

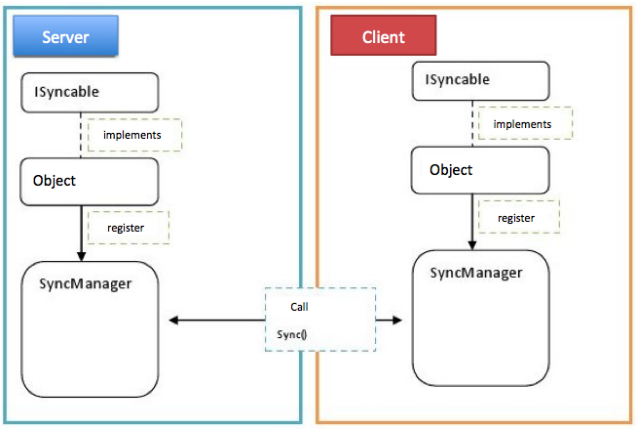

Figure 2: Exemplifying the synchronization process

A small snippet of code showing this can be seen below:

```
// Creating a transformable object
var t = new Transform ();
t.LoadFromReader (XmlReader);
// Setting the object as a ISyncableISyncable
tmp = t;
// Defining a valid NetId
tmp.NetId = SyncManager.Instance.NextNetID;
// Registering the object manager
SyncManager.Instance.Register (ref tmp);
```

With this code you get the object registration. In case of objects from the graph that step is not necessary, since they are already registered in their creation. In this case, just the following line is needed:

```
SyncManager.Instance.Syncronize()
```

4.2 Persistence and load

Persistence is also done through the mechanisms that allow collaboration, but with a new engine, X3DHolder. As it is responsible for the exposure of the object in a web environment, it records a cookie that identifies who the user is. It also reads the user's cookie if it exists, and by the UserManager class, detects who the user is on the server and allows it to recover the characteristics of his/her last visit.

4.3 Interaction with other applications

In order to interact with other applications, an X3D application needs a mechanism that exposes the events within the scene to other applications. Two ways are identified:

1 - To program using ECMAScript calls to external mechanisms such as webservices, which are programmed with the desired logic.

2 - To program an X3D viewer that allows the addition of scripts to events of the graph (a node access, modification of some property, etc.) and triggers.

For DWeb3D it was decided to follow the second option and for this UnityRender was developed. This module converts the X3D graph into a Unity 3D graph, so that you can publish a web viewer. This viewer can have more plug-ins in the form of scripts in C# that may come from the graph. Figure 3 illustrates its operation.

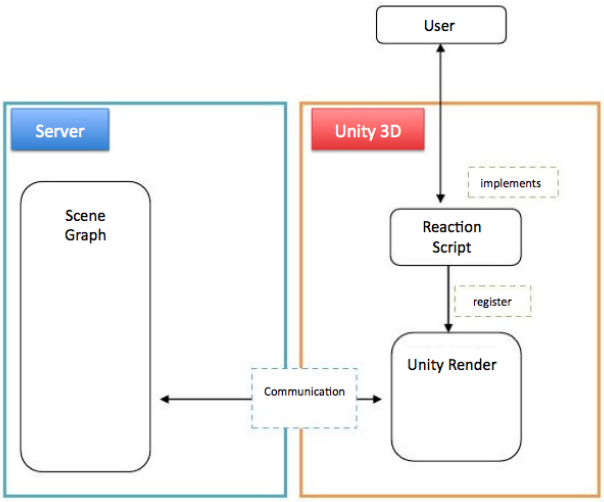

Figure 3: Scheme illustrating the operation of the renderer.

With this plug-in the interaction is achieved, because the scripts can change the graph, which in turn synchronizes with objetSync and changes the graph on the server and this in turn may have programmed code to interact with other applications. Figure 4 illustrates the proposed timing scheme.

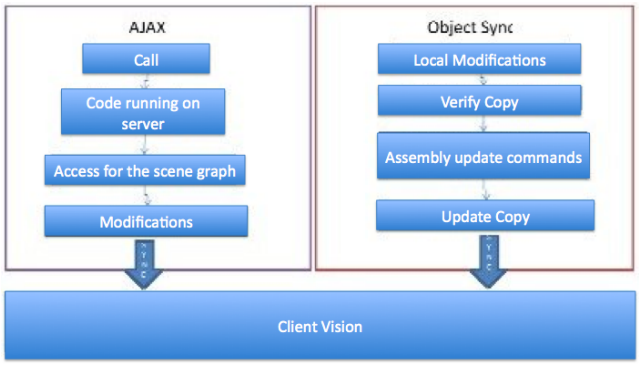

Figure 4: Timing scheme.

4.4 Interaction with the Web GUI

Although it is also a form of interaction with other applications, the web GUI has a special role because it is the environment where the X3D scene stays.

Currently one of the fastest growing fields in the web are the RIAs (Rich Internet Application) [15], which are applications that enable a richer interface with the user. Evolution of browsers and personal computers has allowed many of the applications originally made to run on desktop migrate to the web. The applications most commonly reach this goal through application of Ajax techniques (Asynchronous JavaScript And XML) [16]. Thus, DWeb3D will make use of Ajax techniques to achieve rich interaction with the 3D application.

Ajax methodology is the use of technologies such as JavaScript and XML, provided by browsers to render pages more interactive with the user, making use of asynchronous requests for information. Ajax is an initiative to build more dynamic and creative web applications. Ajax is not a single technology, there are several known technologies working together, each doing its part by offering new features. Ajax incorporates in its model:

- Standards-based presentation using XHTML and CSS.
- Dynamic display and interaction using the DOM.
- Data exchange and manipulation using XML and XSLT.
- Asynchronous data retrieval using XMLHttpRequest.
- JavaScript binding everything together.

Ajax3D comes to apply the same idea of the traditional Ajax for X3D objects. Since X3D can receive scripts in ECMAScript and exposes its features to the browser, it is possible through ECMAScript and asynchronous calls to interact with X3D. With this idea Ajax3D project was developed, providing a small library with basic functions for this interaction.

The operation of the interaction Ajax / 3D scene can be seen in Figure 5. The crucial point is that the server is always updated, thereby ensuring not only the interaction with this user, but also the change of the graph, which in turn enables the change of scene. The role of Ajax in DWeb3D is to allow web GUI components to communicate seamlessly with the scene.

5. CASE STUDIES

As a way to organize the tests and demonstrations small test projects were created, each one focused on a specific feature. These features were designed to validate the possible uses of DWeb3D, demonstrating that the complexity of the development can be hidden in the code of the toolkit.

5.1 Scene Graph

The main functions available in the X3D Model package, which encapsulates the graph and the structure of X3D are:

- Load an X3D file.
- Exposure of the structure of the scene.

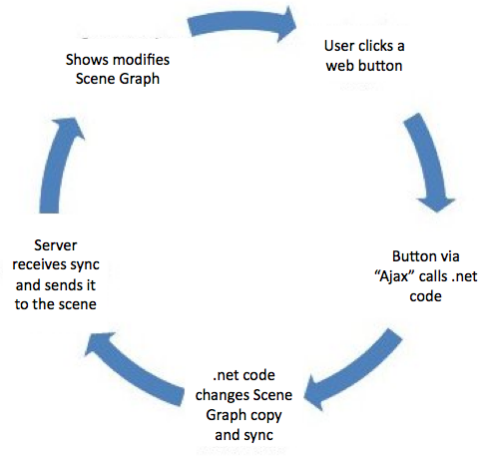

Figure 5: Cycle interaction Ajax / 3D scene.

- Rendering a changed structure again to a file.

The load is obtained through a recursive function that reads each item of the X3D file and transforms it into an equivalent node, respecting the hierarchy. To demonstrate the operation of the model load we can observe the following piece of code:

```
public static void TestRead()
{
   // Create object to hold classes
   X3DRepresentation xrep X3DRepresentation = new ();
   // Load the file
   xrep.LoadFromFile ("../Samples/BeckyRoad.x3d ");
   // Rendering to the screen
   Console.WriteLine(xrep.RendertoX3D());
}
```

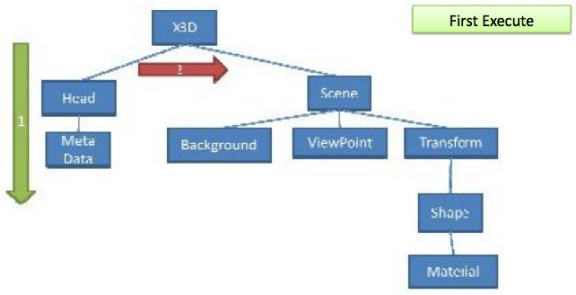

Figure 6: Rendering order of the graph.

It may be noted that the model encapsulates the whole process of loading and rendering necessary to handle a X3D file in .NET code. In the end we have a hierarchical graph with all nodes of the scene. The rendering is done in turn by an in depth recursive function as illustrated in Figure 6.

Figure 7 illustrates the result of the rendering done by the model. It serves as a way to show the correct loading and rendering.

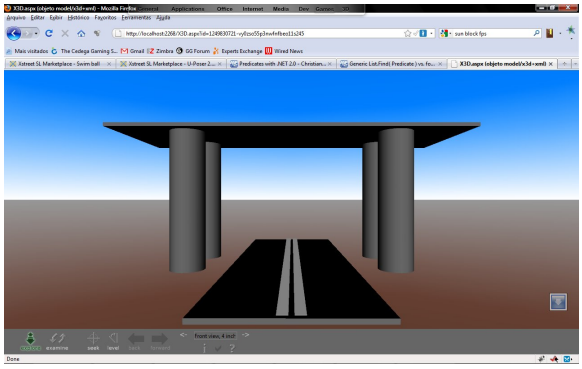

Figure 7: Rendering on Flux 3D model output.

5.2 Collaboration

To demonstrate how to code a simple application that can be collaborative, a demonstration is illustrated in Figure 8.

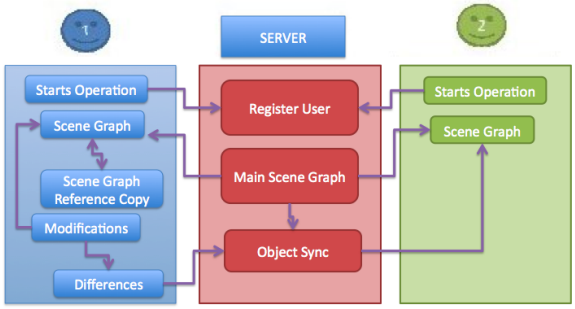

Figure 8: Two users viewing a scene.

The first task to be done is to define a server. It should be able to accept multiple users and sync messages from them. For this a simple scene was set that will be loaded on the server and two clients. Then a client will modify an object and synchronize and the second client is able to see the changes.

The code for the server is the same example in Section 4.1. For clients the code is described below:

```
public void StartClient()
{
   // Create the client on port 1717
   Client c = new Client (1717, "127.0.0.1" ');
   // Setting and creating the basic scene
   c.SceneGraph X3DRepresentation = new ();
   c.SceneGraph.CreateBasicScene ();
   // Registering the client
   ClientManager.Add (c);
}
```

For the client the main difference is that it needs to know which is the server. Once connected, client and server are transparent to the application. Thus, the user simply makes the necessary changes and synchronizes. It is important to note that the synchronization starts from the client.

Finally, an object was created out of the graph so that it could simulate an operation of conversation between two users. The following code shows how this works.

```
var chat = new Chat ();
// Defining the object as a ISyncable
// (It has to implement the interface)
ISyncable tmp = chat;
// Defining a valid netID
tmp.NetId = SyncManager.Instance.NextNetID;
// Registering the object manager
SyncManager.Instance.Register (ref tmp);
// Sending a message
chat.SendMessage ("Hello");
```

In the code above, function SendMessage sets a variable and then synchronizes it. The server replicates this message, which can be read by other clients. A similar code generated without DWeb3D is much more extensive and difficult to understand.

5.3 Persistence and load

To obtain the persistence it is just necessary to use the X3DHolder as a way to expose the code for the viewer. It will be responsible for recording and retrieval of the cookie. You can expose the X3DHolder through the following code.

```
<%@ Register NameSpace="DWeb3D.X3DHolder"
        TagPrefix="X3DH" Assembly="X3DHolder"  %>
<X3DH:X3DHolder ID="XHolder" runat="server">
        </X3DH:X3DHolder>
```

In this aspect the demo application illustrates how to use the second level of persistence, which is the persistence to disk. For this, the server received a Save () method as shown in the code below:

```
public void StartDefaultServer()
{
   // Create the server on port 1717
   SceneServer s = new SceneServer (1717);
   // Defining and creating the basic scene
   s.SceneGraph X3DRepresentation = new ();
   s.SceneGraph.CreateBasicScene ();
   // Saving to disk
   s.Save ();
 }
```

The code above creates a server, sets a scene and calls the default method to save it. This method will verify the existence of a standard database on disk, if it does not exist, it will be created. After that db4o will call the library to be responsible for dealing with the complexities of storing objects to disk. The toolkit makes db4o startup, verifying the existence and basic settings of the library, as well as finding the correct node in the graph, so that the storage can be successfully done.

Without the use of the toolkit, it would be necessary to deal with the initialization of db4o and the verification of the existence of a database, besides having to search the correct node in the graph to save it successfully.

5.4 Interaction with other applications

An example of how we can interact with other application is the server, because through him the X3D scene is interacting with the .NET code on a server that itself is a different application. Yet another example was developed in the form of Unity3D Render. This is a code that can be called in the graphics engine and so the graph is synchronized there. Thus we can add specific behaviors through scripts of the Unity3D nodes.

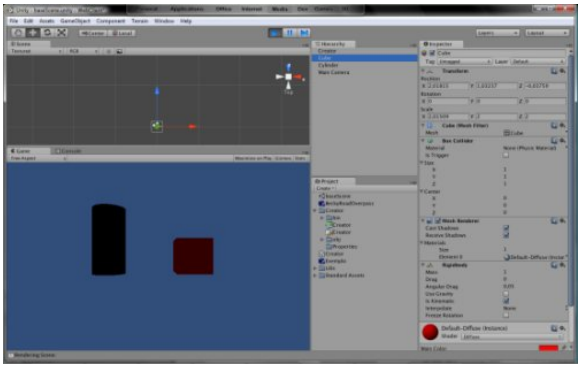

Figure 9: Unity after 3D rendering.

To demonstrate the operation of the plug-in, a simple scene with only one cylinder, a cube, a viewport and some transformations will be used. The process of rendering in Unity3D is presented Figure 9.

In Figure 9 it can be noted that the nodes we created are selectable and they have the characteristics of the nodes of the X3D scene. The process was a conversion of the X3D model graph to the graph of Unity. Another interesting feature of the graphics engine is that it allows scripts to attach scripts to nodes. Using this technique you can add scripts that respond to specific events as a click of the node for example.

To get this same result without the use of DWeb3D it would be necessary to write a conversion method from X3D to the the graph of Unity. This method would have to contain: an XML reader, an interpreter, a function to transform this structure into something that the program could understand, and a function to transform this understandable structure into the structure of Unity.

The DWeb3D approached the problem of interaction with other applications focusing on integration with Unity3D. For a possible interaction with other graphics engine, the DWeb3D implements a scheme for rendering plug-ins. This means, simply rewriting the rendering code, you can enjoy all the other mechanisms of the toolkit like graph handling, among others.

5.5 Interaction with the Web GUI

The web GUI is another application. In this case we are dealing with a more specific case of integration. However, as the web GUI is the environment that hosts the 3D scene, the integration is even more important. Their integration can be made via Ajax calls in common .NET code running on the server. This code will be on the same environment as the X3D server, and through SceneServers manager may have access to the graph server and change it thus affecting the scene. To illustrate this possibility, the following code is presented.

```
using System;
using DWeb3D.Model;
namespace TestWebApp
{
  public partial class _Default : System.Web.UI.Page
  {
    // Create the storage object
    private X3DRepresentation x3dm;
    void Page_Load(object sender, EventArgs e)
    {
```

```
        x3dm = new X3DRepresentation();
        x3dm.LoadFromFile("..\\Samples\\Example.x3d");
        XHolder.Representation = x3dm;
      }
      // Code called by the button to raise the box
      void BtRaiseBox_Click(object s, EventArgs e)
      {
        ((Box)XHolder.Representation.Scene.Nodes[1]).Size
                      += new Vector3D(2, 2, 2);
      }
    }
}
```

In the above example, when the user clicks the raise box button the code go to the storage place and adds 2 to all values. To obtain the same results without the DWeb3D, the simplest way would be to develop an ECMAScript code that via the browser API could access the node and make the change.

6. CONCLUSIONS

This research proposes a toolkit to speed up the development of X3D applications. We also addressed some weaknesses detected in X3D, especially in direct comparison with other 3D technologies on the web. The issues addressed were: collaboration, persistence, interaction with other applications and interaction with the web GUI.

Regarding collaboration, we developed a client/server mechanism to enable message exchange and scene synchronization. Regarding persistence, we created a server control to manage cookies and users, allowing the persistence of camera positions between user accesses to the file. Still related to persistence, we developed a mechanism to save the scene in disk. For interacting with other applications, the messagem exchange mechanism is used, in a way that it is possible to trigger programmed behaviours from scene events. Finally, for the interaction with the Web GUI, Ajax methods were used to access the scene graph.

The development of the toolkit was motivated by the idea that by simplifying the lifecycle of application development for developers, we can accelerate the acceptance of technology. Application tests have shown us, even empirically, that this toolkit can facilitate the process of developing applications that have the requirement of the features proposed.

We saw examples in this paper showing how the use of DWeb3D reduces the amount of code needed to obtain 3D applications. This code reduction not only facilitates the process of initial development, but also brings a number of advantages for those who need to develop these codes. By having a lower development cost, the applications' risk decreases. Another positive aspect is the possibility of freeing developers to focus on the goals of their applications. This occurs because if the developers do not need to deal with the basic layers, which are laborious and complex, they can focus on desired features in its software.

The DWeb3D is itself a useful tool to facilitate the development of complex X3D applications and fits the principles cited above, where a toolkit promotes the use of technology by simplifying development.

7. REFERENCES

[1] J. Allaire. Macromedia Flash MX next-generation rich client. Whitepaper, Adobe, 2002. www.adobe.com/devnet/flash/whitepapers/richclient.pdf.

[2] J. Behr, P. Dähne, and M. Roth. Utilizing X3D for immersive environments. In *Web3D '04: Proceedings of the ninth international conference on 3D Web technology*, pages 71–78, New York, NY, USA, 2004. ACM.

[3] C. Bouras, C. Tegos, V. Triglianos, and T. Tsiatsos. X3D Multi-user Virtual Environment Platform for Collaborative Spatial Design. In *ICDCSW '07: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops*, page 40, Washington, DC, USA, 2007. IEEE Computer Society.

[4] R. Dachselt and E. Rukzio. Behavior3D: an XML-based framework for 3D graphics behavior. In *Web3D '03: Proceedings of the eighth international conference on 3D Web technology*, pages 101–ff, New York, NY, USA, 2003. ACM.

[5] J. Dehaan. *Flash MX 2004 - guia autorizado Macromedia.* Elsevier, 2004.

[6] Y. Doi and K. Kagawa. An X3D generator plug-in for Eclipse in a Web-based Educational System for Programming. In *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, pages 2523–2528, Chesapeake, VA: AACE, 2006. EDMEDIA.

[7] R. J. dos Santos, A. L. Battaiola, and R. P. Dubiela. Aspectos Fundamentais da Criação de jogos em Shockwave 3D. WJogos / SBGames 2004, Simpósio Brasileiro de Jogos de Computador e Entretenimento Digital, 2004.

[8] M. E. Frincu and D. Petcu. Remote Control for Graphic Applications. In *Symbolic and Numeric Algorithms for Scientific Computing, International Symposium on*, pages 304–309, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[9] S. Greenberg. Toolkits and interface creativity. *Multimedia Tools Appl.*, 32(2):139–159, 2007.

[10] R. Grehan. Complex Object Structures, Persistence, and db4o. Whitepaper, DBO, 2005. http://www.odbms.org/download/006.01/Grehan/Complex/Object/ Structures/May/2005.pdf.

[11] J. Huang and B. Cheng. Interactive Visualization for 3D Pipelines Using Ajax3D. In *Networking and Digital Society, International Conference on*, pages 21–24, Los Alamitos, CA, USA, 2009. IEEE Computer Society.

[12] S. Jourdain, J. Forest, C. Mouton, B. Nouailhas, G. Moniot, F. Kolb, S. Chabridon, M. Simatic, Z. Abid, and L. Mallet. ShareX3D, a scientific collaborative 3D viewer over HTTP. In *Web3D '08: Proceedings of the 13th international symposium on 3D web technology*, pages 35–41, New York, NY, USA, 2008. ACM.

[13] A. Lugmayr and S. Kalli. *Using Metadata-based SVG and X3D Graphics in Interactive TV* . Springer London, 2005.

[14] T. Parisi. Ajax3D: The Open Platform for Rich 3D Web Applications. Whitepaper, Media Machines, Inc, 2006.

[15] Y. S. Park, J. H. Lee, H. R. Choi, H. S. Kim, J. U. Jung, and J. Y. Park. Development of an RIA-based user interface for promotion of effectiveness in marine transportation. In *ACS'08: Proceedings of the 8th conference on Applied Computer Science*, pages 366–372, Stevens Point, Wisconsin, USA, 2008. World Scientific and Engineering Academy and Society (WSEAS).

[16] R. Riordan. *Head First Ajax*. O'Reilly Media, 2008.

[17] R. Turkowski. Web3D Consortium X3D Revision to add Physics, Particle Systems, UI Enhancements, Realistic Motion. Whitepaper, Media Machines, Inc. Disponível em:

http://www.web3d.org/images/uploads/pdfs/Web3D_Consortium-X3D_Revision_1.pdf, 2006.

[18] Web3D Consortium. Virtual Reality Modeling Language. Available at: http://www.web3d.org/x3d/specifications/vrml/ISO-IEC-14772-VRML97/, 1997.

[19] Web3D Consortium. Extensible 3D (X3D). Available at: http://www.web3d.org/x3d/specifications/ISO-IEC-19775-1.2-X3D-AbstractSpecification/, 2008.

[20] J. C. Weber and T. Parisi. An Open Protocol for Wide-area Multi-user X3D. *Proceedings of the twelfth international conference on 3D Web technology*, pages 133–136, 2007.