Bring Cultural Heritage 3D Content on the Web Using the X3DOM Framework

Anestis Koutsoudis (akoutsou@ceti.athena-innovation.gr)
Multimedia Department – ILSP/Athena Research and Innovation Centre

Abstract

The Web based visualisation of 3D models that are derived from our cultural thesaurus is a significant scheme for their dissemination in the modern world. Over the last fifteen years, a wide number of approaches to integrate 3D technologies in Web browsers have been developed. It is this evolution of real time 3D computer graphics technologies in combination with the currently available high bandwidth Internet connections and modern Web browsers that enable today users to explore online complex 3D scenes. This report is written within the framework of the CARARE ICT-PSP project and focuses on the development and use of the X3DOM framework. It gathers some of the core aspects of the framework's current version and provides a tutorial on how it can be used as a carrier for bringing 3D content of cultural heritage to Web's end-user. The tutorial is related with the development of a dynamic Website that integrates HTML, X3DOM and PHP technologies in order to deliver 3D content to the end-user without the need of installing any plug-ins to a Web-browser.

1. Introduction

It is a fact that the cultural heritage domain can intensely benefited by the advantages and the progress offered in content and knowledge sharing through a global medium such as the Web. The involvement of technology in the dissemination of cultural heritage enables a smooth transition from subjective promotion methodologies such as sketches, designs, drawings, paintings and text based descriptions to more objective such as photographs, video sequences and more recently 3D virtual reconstructions. It is safe to allege that a 3D model provides a better perception than a typical photograph or a video sequence. In fact 3D digital replicas are also able to provide solutions to divergent necessities such as digital preservation. Over the last years, numerous efforts regarding the implementation of Web-based 3D environments have already been reported. A number of surveys arguing the applicability and advantages of such 3D representations have also demonstrated that a photorealistic 3D reconstruction has a great impact on their visitors and encourage more people to visit the physical exhibitions [1][2].

In addition, nowadays 3D computer graphics are considered a commodity. We can find 3D graphics processing units embedded in low cost mobile phone devices and tablets. The high data bandwidth demand required by 3D graphics applications is actually available today by fast Internet access connections. Thus, the hardware backbone is already mature to provide a standardised technology of Web-based 3D content sharing and dissemination. Currently, research institutions are working towards the establishment of standards that will define 3D data visualisation, representation and interaction methods. Towards the same direction the European Union is funding research and development projects (e.g. Europeana [3], CARARE [4], 3D ICONS [5], 3D-COFORM [6], AIM@SHAPE [7], Sculpteur [8]) to take advantage of the 3D technologies and apply them in the cultural heritage domain. On a parallel development pathway, during the last decade several software digitisation applications appeared. They allow the low cost production of 3D replicas of real world movable and unmovable objects, their processing and their visualisation.

On the other hand, the development of Web-based software technologies and tools for 3D data sharing has begun in the early '90s but it is still a work-in-progress. As an initial effort,

one can consider the obsolete by now Virtual Reality Modelling Language (VRML) [9]. Bringing 3D content on the Web is a multidimensional procedure with numerous crucial aspects. Some of these aspects are related to the number of the available operating systems and Web browsers (user agents). It is important for these two aspects to be taken under serious consideration when developing a universal technology that will enable end-users with different Web browsers running on different operating systems to experience 3D content within a Website.

This report is focused on the use of the *X3DOM* Framework as *a current solution to deliver 3D content in a Website that is compatible with all*[1] *major Web browsers under different operating systems without installing any plug-ins*. The report is organised as follow: In Section 2, we describe in brief a number of related solutions that allow visualising and exploring 3D content in a Website. Section 3 contains a description of the *X3DOM* Framework by referencing the publications being made by the framework's authors. Then, we continue in Section 4 by providing a number of prototypes and case studies available on the Web that use the *X3DOM* technology. Finally, Section 5 provides a detailed tutorial of how a developer can use open source technologies to bring 3D content in a Website involving the *X3DOM* technology.

2. Related Work

The development of a standard that will natively provide real time 3D graphics through all Web browsers that also run on all the major operating systems is a difficult task to be achieved. In this Section, we briefly describe some of the related technologies available for bringing 3D content over the Web with the use of browser plug-ins.

The *Virtual Reality Modelling Language* (*VRML*) is a standard file format for representing 3D interactive vector graphics that been particularly designed with the World Wide Web in mind. Today, *X3D* file format [10] is considered the successor of *VRML*. Both *VRML* and *X3D* [10] have been accepted as international standards by the International Organization for Standardization (ISO). The first version of VRML was specified in November 1994 and it was first approved in 1997. This version was specified from, and very closely resembled, the API and file format of the Open Inventor software component, originally developed by SGI. The current and functionally complete version is *VRML97* (ISO/IEC 14772-1:1997). *VRML* has now been superseded by *X3D* (ISO/IEC 19775-1) [1]. Some of the available *VRML/X3D* plug-ins such as the *BS Contact* from Bitmanagement [11], provides custom extensions in order to allow the Web developer to access modern 3D graphics features (bump mapping, multi stage texturing, texture level of detail, normal mapping, etc), shading (*OpenGL* shader language *GLSL*), complex real time 3D shadows and lighting, scene post-processing and modern game like functionality. Furthermore, most of the available *VRML/X3D* plug-ins allow the transfer of compressed 3D models using the *GNU gzip* compression scheme [12].

Moreover, *O3D* is a Web oriented cross-platform JavaScript API which was written in *C* by Google. Google has recently start announced that *O3D* will be changing in order to run on top of the *WebGL* platform that is discussed further on. *O3D* was developed in order to provide an API for 3D games, 3D advertisements, model viewers, simulation, engineering and massive online virtual worlds

Another cross-platform technology is the *Unity 3D* [14]. It is a game engine (integrated authoring tool) for creating apart from 3D video games other interactive 3D content. The engine provides a Web browser plug-in able to run on Microsoft Internet Explorer, Mozilla

[1] Currently Microsoft's Internet Explorer (8,9,10pp) is the only major browser that requires the installation of a WebGL plug-in.

Firefox Safari, Opera, Google Chrome and Camino. Unity is also able to export its projects as standalone applications for the mobile devices (e.g. IOS and Android). The non-Pro version of the game engine is provided for free and it involves an integrated development environment with hierarchical, visual editing and detailed property inspectors.

Another solution for bringing 3D content over the Web is the open source *OpenSpace3D* development platform [15]. It provides solutions for interactive real time 3D projects. The development team behind the platform continuously attempts to integrate the latest technologies in domains of virtual reality, speech recognition and real time 3D computer graphics. At the moment the software is provided as a plug-in for all major Web browsers under the Microsoft Windows operating system.

Furthermore, *TurnTool* is a similar platform for bringing 3D content in a website [16]. Turntool allows the conversion of 3D models/scenes from applications like 3D Studio Max, Autodesk Viz, ArchiCad and Cinema4D into a custom 3D file format viewable from their own Web browser plug-in.

3. The X3DOM Framework – Current Architecture and System Implementation

Although there is a number of solutions available to the developer to bring 3D content in a Website by using plug-ins (Section 2) there is currently one solution that attempts to offer 3D content natively, without the need of installing any plug-ins. The *X3DOM* (pronounced X-Freedom) framework is an experimental open-source framework and runtime developed by Fraunhofer IGD Visual Computing System Technologies [19] that provides support in the currently on-going discussion in the *Web3D* and *W3C* communities on how *HTML5* and declarative 3D content could be implemented. It actually attempts to fulfil the current *HTML5* specification for the declarative 3D content and allows the inclusion of *X3D* models-elements [10] to be a part of an *HTML5 DOM tree* [20]. The following figure (Figure 1) illustrates the relations between other associated standards such as the *SVG* 2D vector framework [21], the *HTML5* canvas element and *WebGL* (Web Graphics Library *Javascript* API) [23].

According to Behr et al. [18] *X3DOM* is a temporary solution for Web application developers until the *X3D* becomes a native (integral) part of *HTML5*. It is a framework that enables the integration of 3D content into Web pages without the need to forge new concepts as it relies on and utilises today's standards. It can be considered as a method to bring *X3D* (the open international standard for 3D on the Web [22]) to mass market and to promote 3D in a declarative manner for everyday use. One of *X3DOM's* main goals is to provide a *live-interactive X3D* scene in the *HTML DOM* that allows the developer to manipulate the scene's 3D content by adding, removing and altering *DOM* elements. It also supports other *HTML* events such as the on-click event [18].

Figure 1. Illustration of X3DOM as a part of the HTML5 specification and its relation to WebGL [1]

X3DOM can be viewed as a thin layer between *HTML* and *X3D* that enables a connection between the two standards. The authors of *X3DOM* state that this framework is their attempt to make declarative 3D a first class citizen (just like text, images, audio and video) of every Web browser. The framework proposes the integration of *X3D* content directly into the *HTML Document Object Model* (DOM)-tree. Additionally, the authors of *X3DOM* proposed using *X3D* since the *HTML5* specification already briefly references *X3D* for declarative 3D scenes [18]. *X3DOM* was accepted by the Web3D community [25] as one of the official solutions while the *X3D/HTML5* Working group [Consortium 2010] is focused on the integration of *X3D* into *HTML*. Their final goal is to make the authoring and use of declarative *XML-based X3D* scenes as natural and well-supported for *HTML5* authors as the current support of technologies such as *SVG* [5] and *MathML* [26].

Behr et al. [18] describe the system architecture being organised intro three building blocks. These are the *User Agent (UA)* which is the actual Web browser, the *X3D runtime* (*X3DR*) and the *connector (CON)*. The *UA* is responsible for holding the *DOM* tree. It integrates and composes the final rendering of the scene. It is equipped with a *URI*-resolver that is used to download images, video files, sounds and other types of data types that compose the scene's content. On the other hand, the *X3DR* block offers the services for building and updating the *X3D* scene and it handles the user's inputs that are related to point clicking and navigation in the 3D space. The *CON* block plays the role of the inner core of the systems architecture by connecting the *DOM*-tree hold by the *UA* with the *X3DR*. It distributes the relevant changes in both directions. Thus, any *DOM*-tree updates such as an addition, a removal or an attribute change in an object are forwarded to both directions. The *CON* block is also responsible for handling the media *up*- and *down*-streams.

The current implementation of the system provides a *JavaScript-layer* that automatically monitors any changes in the *DOM* and updates the *X3DR* and the rendering system in the background. This approach utilises the fact that modern *UAs* (Web browsers) parse and append tags from an *HTML* stream to the *DOM* even though they are not part of the current standard. According to the authors this intermediate framework supports a smooth transition from current temporary situation to a situation where all Web browsers will support *X3D* natively.

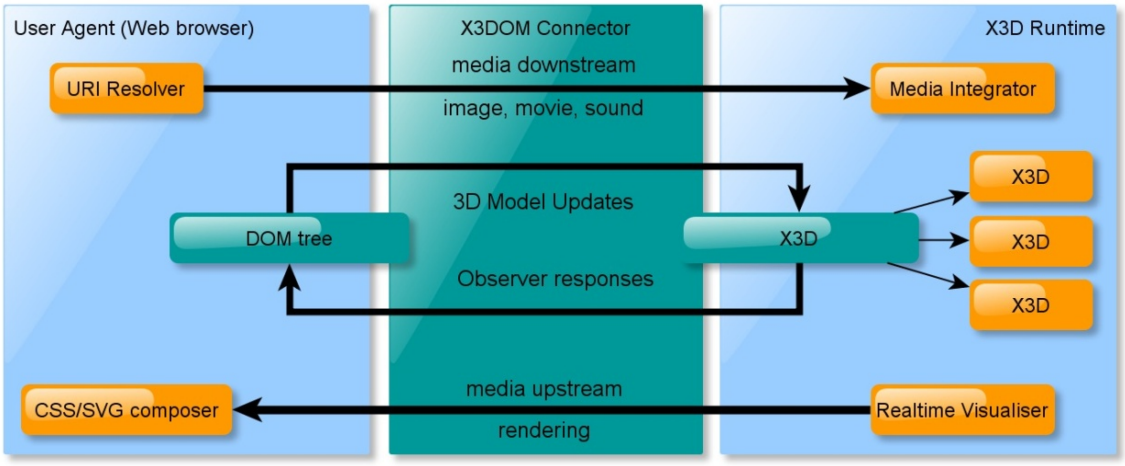

Figure 2. Proposed X3DOM system architecture based on three blocks [18][24]

Furthermore, the proposed system is able to instantiate and synchronise different backend 3D technologies. X3DOM follows the *fallback model* that is illustrated in Figure 3. Depending on the *X3DOM* profile and the current Web browser, the system automatically selects the appropriate backend rendering system. Each one of the supported backends provide

support for specific *X3D* profiles and also has specific features and performance criteria [18][24]. The system, as being illustrated in Figure 3, checks the requested 3D backend profile and tries to detect and initiate a match on the running computer system. According to the authors [24], the backends are organised in terms of performance and features. The lower the position in the fallback model the lower the quality of the 3D graphics. When the first node of the fallback model is executed, the framework checks if the Web browser supports natively X3D content. As currently there is no Web browser available that offers this option the fallback model continues with the rest of the possible backend 3D rendering options. The second node checks if an *SAI-plugin* is available to the Web browser. Such an example is the *Instant Reality X3D plugin* [28]. Then it continues with the *O3D-plugin* [23]. The current implementation supports, apart from others backends (Figure 3), the *WebGL API* [23]. This backend is supported by all major Web browsers such as Chrome, Firefox, Safari and Internet Explorer through the use of a plug-in.

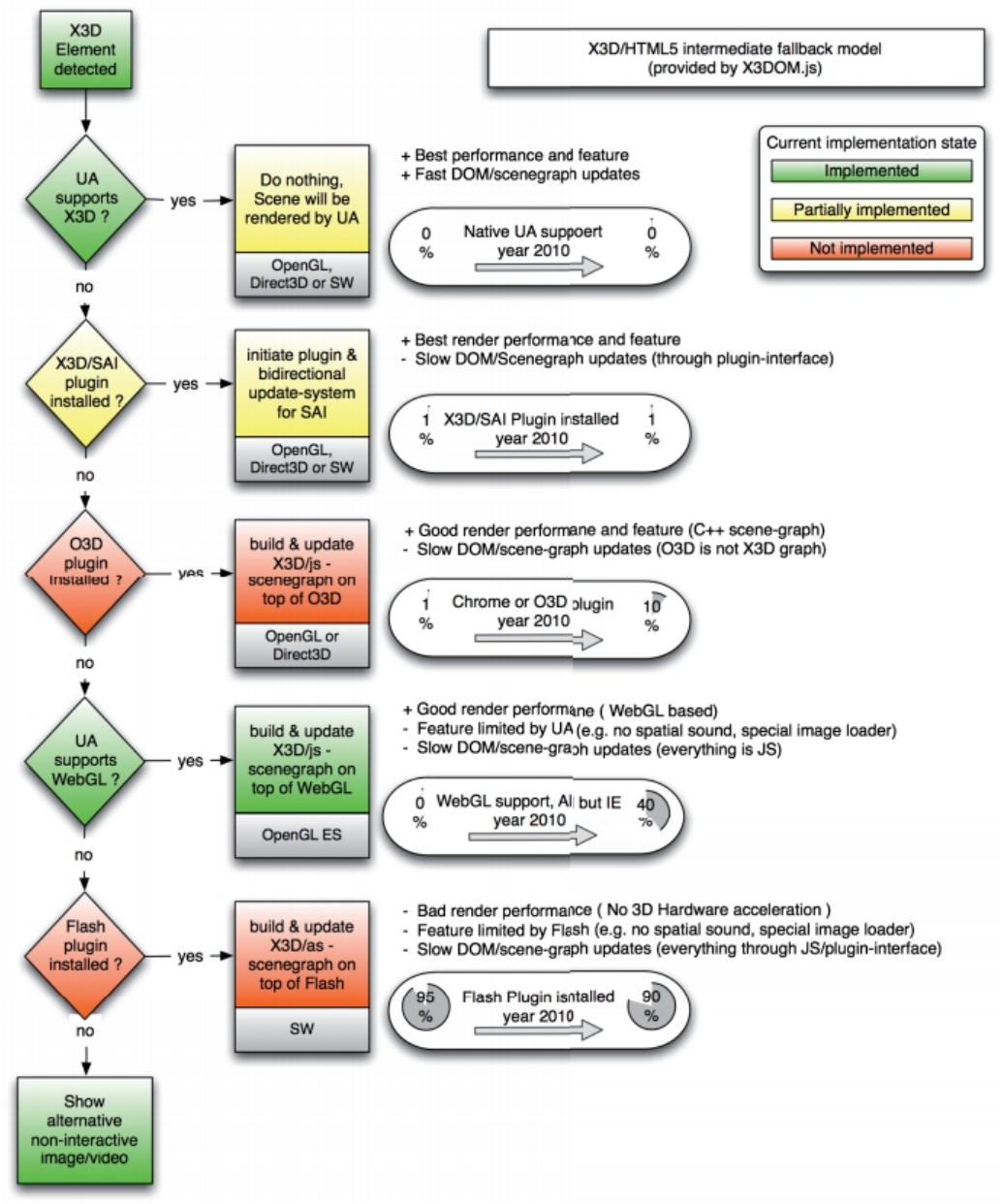

Figure 3. X3DOM fallback model as being illustrated by Behr et al. [18][24]*. An important feature of the X3DOM is the support of different 3D rendering backends.*

The *WebGL API* can be considered as a new technology that is embedded into JavaScript. *WebGL* is the only current backend that doesn't require from the end-user to install any additional plug-ins. It describes the 3D rendering context available through the HTML5 canvas element. This is achieved by defining a set of new *JavaScript* objects and methods [23]. *WebGL* is based on the *OpenGL ES 2.0* standard proposed by Munshi et al. [27][31]. It can be comprehended as an *OpenGL* dialect that was developed for embedded and portable devices (e.g. tablets, mobile phones, etc.) that currently are equipped with inferior in terms of performance graphics processing units (GPUs). Currently, *WebGL* is also implemented in *Mozilla Firefox* for both *Android* and *iOS* platforms. Thus, *X3DOM* can be seen as a powerful solution not only for desktop computer systems but also for mobile devices. In fact some of the major browser vendors such as *Apple (Safari)*, *Google (Chrome)*, *Mozilla (Firefox)*, and *Opera (Opera)* are actual members of the *WebGL* Working Group. *WebGL* is accompanied by the GLSL cross-platform shading programming language. The *Body Browser* made by Google is one of the first examples that exploit the *WebGL* technology to allow the user to interactively explore a detailed 3D model of the human body [32].

4. Showcases, Examples and Prototypes based on the X3DOM Framework

A number of publicly available Websites have been created that exploit the *X3DOM* framework in order to bring 3D content experience to their visitors. Some of them present content derived from the cultural heritage domain and this is the main reason why they are mentioned in the report.

The EU founded project 3D-COFORM utilises the binary compression method available in the *X3DOM* to visualise low polygon models captured with shape-from-silhouette technique as well as large 3D datasets (up to 4 million polygons) in real time. The models have been digitised and provided by the Victoria & Albert museum in London (Figure 4) [33].

The Athena Research Centre through its Multimedia Department has implemented a showcase where a number of cultural heritage related objects are presented as a 3D gallery. The user is able to select one of the 3D models through a 2D user interface and explore an artefact in 3D using the navigation controls offered by the *X3DOM* framework (Figure 5). The showcase is taking advantage of the inline element available in the *X3DOM* framework in order to dynamically create a *X3D* scene and stream it to the Web browser. This showcase is an implementation of the tutorial included in this report (Section 5).

The Digital Irish Light Experience is a project that produced a thematic Website of lighthouses in Ireland and exploits again the *X3DOM* framework (Figure 6). Furthermore, Lornet-Design has implemented a thematic website about the city of Dijon located in Eastern France. The visitor can perform a virtual visit to a low complexity 3D model of a part of the city. A number of predefined viewpoints are also available to the virtual visitor to be transferred directly to specific points of interest within the 3D model.

Michaelis et al. presented a conceptual approach of how we can develop web apps that provide real-time 3D support, behave like native apps and run platform independently on smartphones, tablets and desktop computers. Their concept is completely based on standard web technologies like *HTML5*, *CSS3*, *DOM* scripting, and *Ajax* [35]. Pimsuwan et al. [36] created a VR book store using *X3DOM*. To accomplish that, they have used technologies such as the *X3DOM*, JavaScript embed in a *HTML* page, serversided scripts and PHP. Prieto et al. [37] focused on the visualization of 3D city models in mobile devices natively through

Web browser using *X3DOM*, driven by the use of *CityGML*[2] as the reference data model. Hering et al. have presented a campus information system based on WebGL [38].

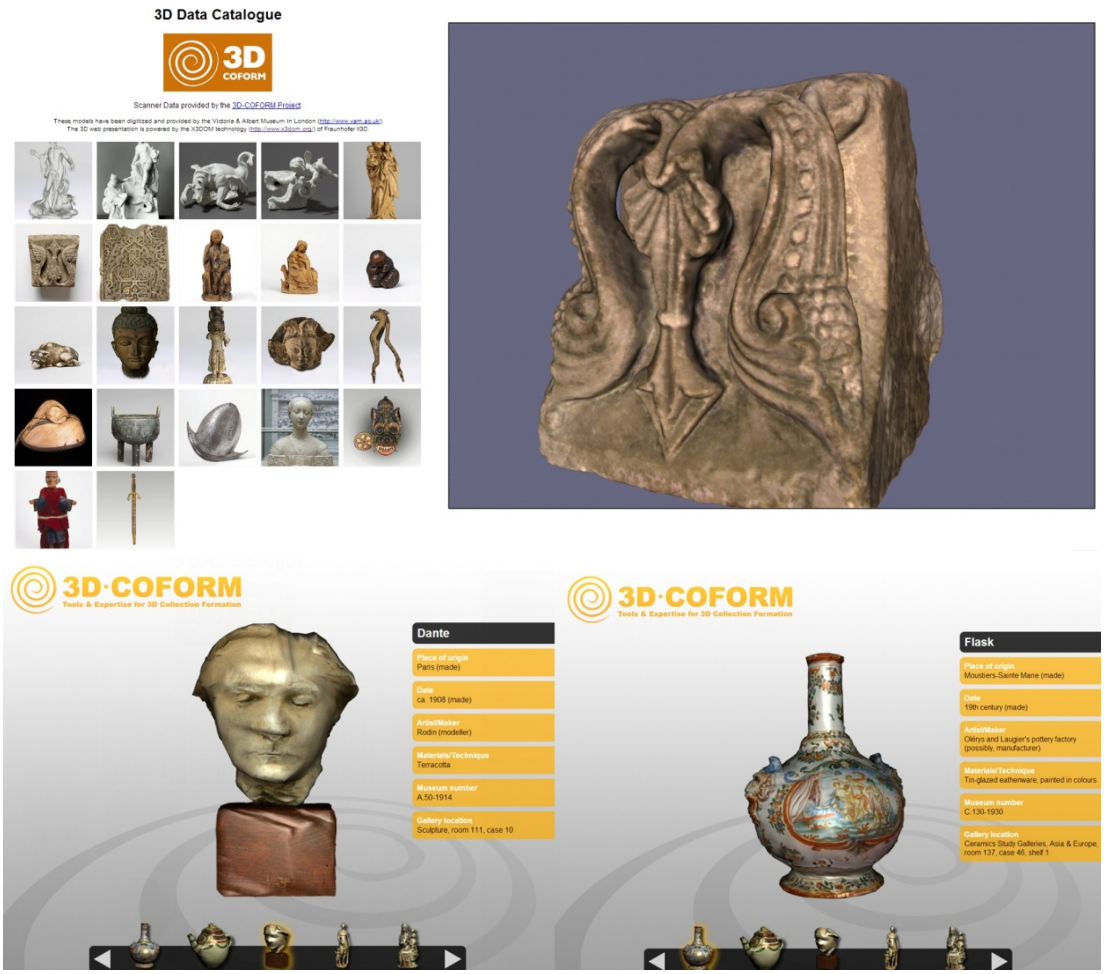

Figure 4. 3D Data Catalogue example based on the X3DOM – 3D COFORM project

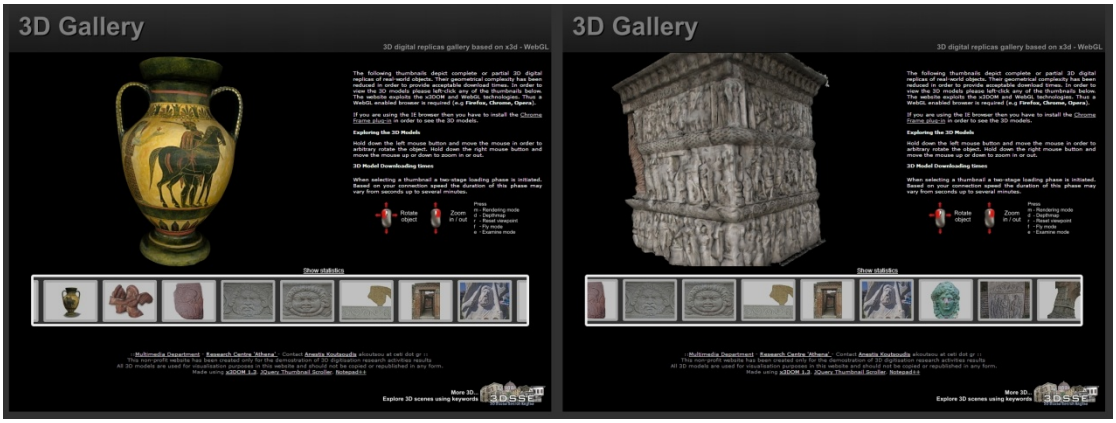

Figure 5. Complete or Partial 3D Digital Replicas of Real World Objects – Multimedia Department - Athena R.C.

[2] CityGML allows representing and storing information of 3D city models in a single data model and facilitates the use and the interoperability of the same by different agents [18].

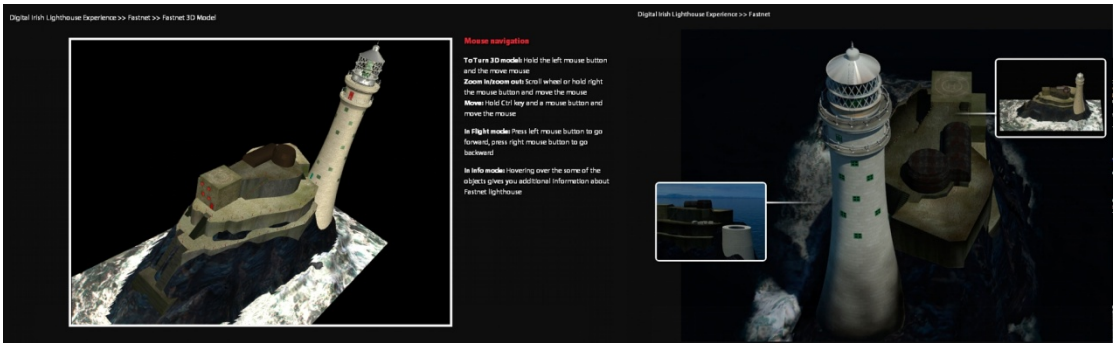

Figure 6. Irish Lighthouse 3D Model visualised using X3DOM

Figure 7. 3D Virtual Tour in Dijon, France

5. Using the X3DOM Framework – A tutorial

This tutorial describes in detail an integration of common Web technologies such as *HTML* and *PHP* with *X3DOM* in order to demonstrate a simple and efficient pipeline for embedding 3D content in an *HTML* Website. The tutorial has been built on source code examples found in the official Website of *X3DOM's* documentation [45]. The proposed system has as prerequisites some basic Web development knowledge and the installation of a Web server (e.g. Apache *HTTP* Web server [39]) along with a *PHP* hypertext preprocessor [40]. An open source 3D model processing software (Meshlab [34]) is also required in order to perform the transformation of the 3D content in the X3D file format that is required by the proposed pipeline.

The following figure (Figure 8) illustrates the components of the described system. The system is composed by two parts, the *online* and the *offline part*.

The *online* part requires as mentioned before the installation and setup of an *HTTP Webserver* and a *PHP hypertext preprocessor*. The installation procedures of these components are out of the scope of this tutorial and thus the reader will need to refer to their own installation guidelines [39][40].

On the other hand, the *offline* part is related with the preparation (transformation) of the 3D content in the required 3D file format (*X3D*). This is also considered as the initial step of the *Offline* part. In order to achieve this, a mesh processing tool like *Meshlab* can be used.

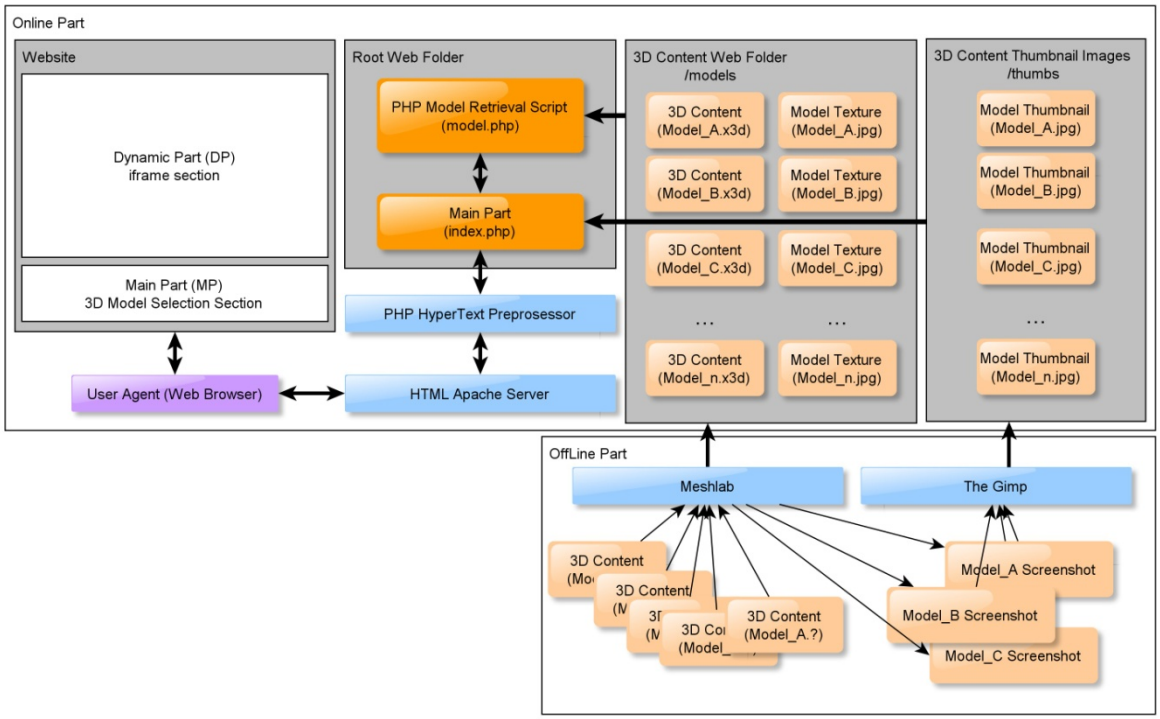

Figure 8. Visualisation of the online and offline parts of the proposed example

5.1 OffLine Part Implementation

Given a number of 3D models in the *OBJ* 3D file format (e.g. `Model_A.obj, Model_B.obj, Model_n.obj`) followed by texture map images (e.g. `Model_A.jpg, Model_B.jpg, Model_n.jpg`) Meshlab can be used to convert them into the *X3D* file format. As a result a number of new files is produced (e.g. `Model_A.x3d, Model_B.x3d, Model_n.x3d`) followed again by their texture map images (e.g. `Model_A.jpg, Model_B.jpg, Model_n.jpg`). These X3D files should be stored in a folder[3] accessible by the Webserver. This folder is the *3D Content Web Folder* shown in Figure 8. Additionally, Meshlab can be used for the production of screenshots of each 3D model. These can be used under an image processing tool (e.g. The Gimp [34]) in order to produce a set of thumbnail images. These should also be stored again in another Webserver accessible folder[4] and will be used within the Website as hyperlinks to the 3D models.

Once the 3D content is converted in the *X3D* format, a UTF-8 text file editor can be used to explore the content of these files. In this tutorial, we consider the case where a single polygonal mesh is described within the *X3D* file. The following table (Table 1) presents an example of an *X3D* file format.

The *X3D* source code found in Table 1 points to an external *JPEG* image file (`Model_A.jpg`) which is the texture map image file. Additionally, the source code includes the traditional data structures one can find in almost any 3D triangular mesh file format. There are elements to declare the vertices coordinates (Element `Coordinate point`), their organisation into triads (Element `IndexedFaceSet coordIndex`), the normal vector of each triangle (Element `Normal vector`) and *UV* texture map coordinates (Element `TextureCoordinate`).

[3] Within all the source code snippets found in this tutorial, the name `models` will be used as the folder's name.
[4] Within all the source code snippets found in this tutorial, the name `thumbs` will be used as the folder's name.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.1//EN" "http://www.web3d.org/specifications/x3d-3.1.dtd">
<X3D   profile="Immersive"   version="3.1"   xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-
3.1.xsd" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
 <head>
  <meta content="1.x3d" name="title"/>
  <meta content="Generated from Meshlab X3D Exported" name="description"/>
  <meta content="" name="created"/>
  <meta content="Meshlab X3D Exported, http://meshlab.sourceforge.net" name="generator"/>
 </head>
 <Scene>
 <Transform DEF="mesh_0_0_5_part_5" translation="0.0 0.0 0.0" scale="2.4 2.4 2.4"  >
  <Shape>
   <Appearance>
    <ImageTexture url="Model_A.jpg"/>
   </Appearance>
   <IndexedFaceSet creaseAngle='4' coordIndex="0 1 2 -1 3 2 1 -1 2 4 0 -1 1 5 3 -1 6 1 ...">
    <Coordinate point="0.194701 -1.09601 0.224906 0.107794 -1.10002 0.188525 0.221676 ...">
    <Normal vector="0.00025962 -0.992483 0.122384 -0.00217215 -0.99286 0.119269 0.000175785...">
    <TextureCoordinate point="0.479 0.7378 0.4904 0.762 0.4494 0.7593 0.4608 0.7835 0.4585...">
   </IndexedFaceSet>
  </Shape>
  </Transform>
 </Scene>
</X3D>
```

Table 1. Example of X3D content

5.2 Online Part Implementation

Furthermore, on the *online* part of the system, the proposed Website composed by a *Main Part (MP)* and a *Dynamic Part (DP)*. Both parts are implemented as PHP source code files as they contain script sessions (Shown in Figure 8 as `index.php` and `model.php` under the Root Web folder).

The MP will be used by the end-user to select the 3D model he or she would like to explore. This will be achieved by clicking on a set of typical *HTML* hyperlink elements. On the other hand, the *DP* part will be responsible for fetching the actual X3D content. Every time the user selects one of the available, in the *MP*, hyperlinks, an *X3D* model will be retrieved and will be presented within an `iframe` element that will be declared in *MP* (Figure8). The content of the `iframe` is the *Dynamic Part* (DP) of the Website. More specifically, *MP* (index.php) can define at some point an `iframe` element. An example of such declaration is given below:

```
<iframe     name="3d"     src="null.html"     frameborder="no"     width="100%"
height="100%" scrolling="no"></iframe>
```

Table 2. Declaring the iframe where the 3D model will be presented

The `iframe` is named *"3d"* and this name will be used by the *X3D model retrieval PHP script* (named `model.php` in this example) as the `target` frame in which its results will be presented. Furthermore, the `iframe` can point to a *null HTML* file that will initially fill the `iframe` area with some other content (e.g. Descriptive text, a logo, etc).

Apart from the `iframe` declaration, *MP* also involves a number of hyperlinks to the 3D models. These hyperlinks can be assigned to the thumbnail images that were previously created and may point to the *X3D model retrieval PHP script* (`model.php`) which will accessed along with a *URL-based* parameter that will contain the filename of a 3D model. The result of this hyperlink will be directed within the *`iframe`* named '3d' (*`target="3d"`*) that has been previously defined in *MP* (Table 3).

```
<?php
         $a = 'ModelA.x3d';
          $b = 'ModelA_Thumbnail';
          echo '<a href="model.php?fname='.$a.'" target="3d">
                    <img src="thumbs/'.$b.'.jpg" />
                </a>';
?>
```

Table 3. Creating links for the X3D models

Furthermore, the content of the `model.php` script can be as follow:

```
<html>

  <head>
    <meta http-equiv='Content-Type' content='text/html;charset=utf-8'></meta>
          <meta http-equiv="X-UA-Compatible" content="chrome=1" />
          <link rel='stylesheet' type='text/css' href='http://www.x3dom.org/x3dom/release/x3dom.css'></link>
          <script type='text/javascript' src='http://www.x3dom.org/download/1.3/x3dom.js'></script>
  </head>

<body    onLoad="init()"    bgcolor="#000000"    text="#000000"    leftmargin="0"    topmargin="0"    marginwidth="0"
marginheight="0">

 <div  style="background-color:#000000;" align="right">

  <x3d  id='the_element' showStat='false' showLog='false' x='0px' y='0px' width='680px' height='433px'>
    <scene>
      <Background skyColor='0 0 0'></Background>
      <inline url='<?php echo 'models/'.$_GET["fname"].'.x3d';?>'></inline>
     <scene>
  </x3d>

 </div>

</body>

</html>
```

Table 4. A PHP script that uses the X3DOM framework to visualise and X3D model

A link to the *Javascript X3DOM* framework is included in the `HEAD` section of the PHP script file. The URL included points to the current 1.3 version.

```
 <script type='text/javascript' src='http://www.x3dom.org/download/1.3/x3dom.js'></script>
```

Furthermore, an `X3D scene` is defined within the limits of a `DIV` element along with some additional properties such the width and height of the window where the 3D model will be displayed. These properties can be set to the same dimensions as the `iframe` in *MP* in order for the 3D content to cover the whole `iframe` area.

```
<x3d  id='the_element' showStat='false' showLog='false' x='0px' y='0px' width='680px' height='433px'>
```

A background colour is also defined in order for the `iframe` background to match with the rest of the Website. The *X3DOM* background's opacity can also be defined [36]. Then, the X3D scene in actually enriched with one of the 3D models stored in the /`models` folder. This is achieved by using the `inline` element.

```
<inline url='<?php echo 'models/'.$_GET["fname"].'.x3d';?>'></inline>
```

Once the above script is executed, the *X3DOM Javascript* is initiated and it will then be responsible for parsing the *X3D* file and visualise it.

The current version of *X3DOM* offers a number of options to explore the 3D scene. More specifically, by holding down the left mouse bottom and by moving the mouse the object is rotated towards the direction defined by the mouse movement. Zooming in and out is enabled by holding down the right mouse button and moving the mouse. By pressing the key *'M'* a number of rendering methods are available to the user (The current version shifts between coloured point cloud and 3D mesh rendering). The key *'R'* resets the viewpoint of the 3D scene, while keys *'F'*, *'E'*, *'W'* enable the fly, examine and walk mode respectively. The key *'D'* enables a rendering window where the model is visualised using a normalised colour encoded depthmap method that is used for detecting the point on the model which the user is point to.

Furthermore, a number of properties about the current 3D scene (e.g. *Frames-per-second, Number of Triangles, Number of Vertices*, etc.) are available to the end user by including the following *JavaScript* in the `model.php` script (Table 5).

```
<script>

        var $element;
        var debug = false;
        var pick_mode_info;
        var nav_mode_info;
        var ab_info;

        function init() {
                $element = document.getElementById('the_element');
                updateAbInfo('Viewpoint');
                updateNavInfo();
        }

        function updateNavInfo() {
                nav_mode_info = document.getElementById('nav_mode_info');
                nav_mode_info.innerHTML = $element.runtime.navigationType();
        }

        function updateAbInfo(typeName) {
                var bindable = $element.runtime.getActiveBindable(typeName);
                ab_info = document.getElementById('ab_info');
                ab_info.innerHTML = bindable.tagName+"/ " + bindable.getAttribute('description');
        }

        function toggleStats(link) {
                stats = $element.runtime.statistics();
                if (stats) {
                        $element.runtime.statistics(false);
                        link.innerHTML = '<font color=#ffffff>Show statistics</font>';
                } else {
                        $element.runtime.statistics(true);
                        link.innerHTML = '<font color=#ffffff>Hide statistics</font>';
                }
        }

</script>
```

Table 5. JavaScript to display additional properties about the content of the 3D scene [44]

In order to enable or disable the visualisation of these properties within the *X3DOM* window the following *HTML* code snippet can be used together:

```
<a href="#" onClick="toggleStats(this);"><font color=#ffffff>Show statistics</font></a>
```

Concluding, a working implementation of this example can also be found at http://www.ceti.gr/~akoutsou/x3d/demo. For further information about the X3DOM Framework, its use and additional showcases can be found at the official X3DOM Website resources [45]-[47].

6. Conclusions

An important factor in Web content authoring is the interoperability of the content among all the different platforms. The example presented in this report verifies that *X3DOM* can actually achieve this by exploiting technologies such as *JavaScript* and *DOM-tree* infrastructure found in modern Web browsers. The authors of *X3DOM* Framework mention that the future work will be mainly influences by the discussions and results of the *W3C* incubator group "Declerative 3D". Additionally, they also mention that they would like to explore in how far new *HTML5* elements such as the "*device"* tag can be utilised towards interactive Augmented Reality applications [18][24].

References - Bibliography

[1] N. Di Blas, E. Gobbo, P. Paolini, *3D Worlds and Cultural Heritage: Realism vs. Virtual Presence*, International Conference on Museums and the Web 2005, Vancouver, British Columbia, Canada, 2005.
[2] W. A. Thomas, S. Carey, *Actual/Virtual Visits: What are the Links?*, International Conference on Museums and the Web 2005, Vancouver, British Columbia, Canada, 2005.
[3] Europeana, http://www.europeana.eu/portal (last access 22-11-2012).
[4] The Carare Project, http://www.carare.eu (last access 22-11-2012).
[5] 3D-Icons Project, http://3dicons-project.eu (last access 22-11-2012).
[6] 3D-Coform, http://3dcoform.eu/index.php (last access 22-11-2012).
[7] AIM@Shape, http://www.aimatshape.net (last access 22-11-2012).
[8] Sculpteur Project, http://www.sculpteurweb.org (last access 22-11-2012).
[9] VRML97, http://www.web3d.org/x3d/vrml (last access 22-11-2012).
[10] X3D, A royalty-free open standard file format, http://www.web3d.org/x3d (last access 22-11-2012).
[11] BS Contact – Bitmanagement, http://www.bitmanagement.com
[12] The Gzip GNU compression utility, http://www.gzip.org (last access 22-11-2012).
[13] O3D, http://code.google.com/p/o3d (last access 22-11-2012).
[14] Unity3D, http://unity3d.com (last access 22-11-2012).
[15] OpenSpace3D, http://www.openspace3d.com (last access 22-11-2012).
[16] TurnTool, http://www.turntool.com (last access 22-11-2012).
[17] Khronos Group - Media Authoring and Acceleration, Open Standards for Media Authoring and Acceleration, http://www.khronos.org
[18] J. Behr, Y. Jung, J. Keil, T. Drevensek, M. Zoellner, P. Eschler, D. Fellner, *A Scalable Architecture for the HTML5/X3D Integration Model X3DOM*, in the proceedings of the Web3D 2010 conference, Los Angeles, California, July 24-25, 2010.
[19] Fraunhofer IGD – Visual Computing System Technologies, http://igd.fraunhofer.de/Institut/Abteilungen/Visual-Computing-System-Technologies
[20] *X3DOM Instant 3D the HTML way*, Fraunhofer IGD, http://www.X3DOM.org (last access 21-11-2012).
[21] Scalable Vector Graphics, W3C, http://www.w3.org/Graphics/SVG, (last access 21-11-2012).
[22] L. Daly, D. Drutzman, *X3D: Extensible 3D Graphics Standard*, IEEE Signal Processing Magazine, Vol. 24 (6), 2007, pp. 130-135.

[23]WebGL – OpenGL ES 2.0 for the Web (Web Graphics Library Javascript API), http://www.khronos.org/webgl,(last access 21-11-2012).

[24]J. Behr, Y. Jung, T. Drevensek, A. Aderhold, Dynamic and Interactive Aspects of X3DOM, in the proceedings of I3D conference, Paris, France, 20-22 June, 2011, pp. 81-88.

[25]Web3D consortium, Open Standards for Real-Time 3D Communication, http://www.web3d.org/realtime-3d,(last access 21-11-2012).

[26]MathML, W3C, http://www.w3.org/Math, (last access 21-11-2012).

[27]A. Munshi, D. Ginsburg, D. Shreiner, *OpenGL ES 2.0 Programming Guide*, Addison-Wesley, Boston, 2009.

[28]Instant Reality Framework, http://www.instantreality.org/downloads, (last access 21-11-2012).

[29] Google O3D plugin, http://code.google.com/p/o3d, (last access 21-11-2012).

[30] A.M. Manferdini, F. Remondino, *A Review of Reality-Based 3D Model Generation, Segmentation and Web-Based Visualisation Methods*, International Journal of Heritage in the Digital Era, Vol. 1 (1), 2012, pp. 103-123.

[31] WebGL 1.0 Specifications, https://www.khronos.org/registry/webgl/specs/1.0, (last access 21-11-2012).

[32] Zygoty Human Body, http://www.zygotebody.com

[33] 3D Data Catalogue – 3D COFORM Project, http://www.3dcoform.eu/X3DOMCatalogue, (last access 21-11-2012).

[34] X3DOM Demo – Athena RC, http://www.ipet.gr/~akoutsou/x3d/demo, (last access 21-11-2012).

[35] N. Michaelis, Y. Jung, J. Behr, *Virtual Heritage to Go*, in the proceedings of Web3D conference, Los Angeles, CA, 4-5 August, 2012, pp. 113-116.

[36] H. Pimsuwan, S. Phosaard, P. Rattanawicha, W. Chantatub, *X3DOM Virtual Reality Book Store,* in the proceedings of Web3D conference, Los Angeles, CA, 4-5 August, 2012, pp. 183.

[37] I. Prieto, J. L. Izkara, *Visualization of 3D City Models on Mobile Devices*, in the proceedings of Web3D conference, Los Angeles, CA, 4-5 August, 2012, pp. 101-104.

[38]N. Hering, M. Runz, L. Sarnecki, L .Priese, 3DCIS: A Real-time Browser-rendered 3D Campus Information System Based on WebGL, in proceedings Of the 2011 International Conference On Modeling, Simulation & Visualization Methods, pp.10-15.

[39] Apache Web Server, http://httpd.apache.org (last access 24-11-2012).

[40] PHP HyperText Preprocessor, http://www.php.net (last access 24-11-2012).

[41] Meshlab, http://meshlab.sourceforge.net (last access 24-11-2012).

[42] THE Gimp, http://www.gimp.org (last access 24-11-2012).

[43] X3DOM Background Opacity Demo, http://x3dom.org/x3dom/example/x3dom_backgroundOpacity.xhtml

[44] X3DOM runtime API example, http://x3dom.org/x3dom/example/x3dom_runtime.html.

[45] X3DOM's Documention, http://x3dom.org/docs/dev (last access 24-11-2012).

[46] X3DOM's Showcases, http://www.x3dom.org/?page_id=2429 (last access 24-11-2012).

[47]X3DOM's Source Code Examples, http://www.x3dom.org/?page_id=5.