3D Web with Three.js

Caldarella Domenico, Di Carlo Michele, Napoletano Marco, Rosica Rossano

***Abstract*—The world of the Web has changed and consequently also the browsers, which are more and more similar to real operating systemsOne of the most interesting innovations was the introduction of WebGL. WebGL is a powerful and flexible graphics library derived from the world of OpenGL, which offers the ability to render 3D scenes within any modern browser. Given the above, the aim of this work is the implementation of a web site, specifically designed to display and navigate three-dimensional modelsIn particular, this technological innovation , has been applied to the Chan Chan archaeological site located in PeruParticular attention was given to website usability by defining a three-dimensional explorable environment, and as much as possible realistic in the web dimension. The goal has been achieved using the open source library Three.js. This library is implemented in JavaScript using WebGL, and provides provides an interface "simple and intuitive" and a high degree of abstraction to the developers. The end user will be able to virtually visit the archaeological site and individually explore the various bas-relief which are scattered among the building.**

I. Introduction

Just a few years ago, the visualization of 3D objects inside the most common web browser was bounded by the need of using external plugins. In fact in principle the 3d contents were exclusively used in specialized software designed to solve this specific task. In addition, the exploitation of 3D models was limited to the use of stand alone devices and not online. The use of external plugins introduced various limitations and problems that hindered developers and end users such as:

- mandatory third party software modules installation;
- proprietary implementations (with obvious difficulty of interfacing for developers);
- heterogeneous solutions [1].

The need for a standard increased the growing of flexible solutions to cope with both developers and users requests. The development of a standard API for 3D graphics on Web, provides the basis of the contextual development of three fundamental components: infrastructure, hardware and software. Previously, network infrastructure and hardware components were not yet adequate to support and develop online 3d graphics solutions. Nowadays, these limits are can be overcomes. The reasons are manifold: the introduction of HTML5, the birth of WebGL, the infrastructure development and the computer processing power increase are the most important. 3D models represent the best way to convey and manipulate a big volume of information about the representation of a real object: shapes, colors, textures, location, size and orientation. In addition a three-dimensional environment allows the user to change his standing point or move and rotate the object to improve his perception and understanding. This large amount of data makes the most of the 3d objects very heavy and this could badly influence their use, especially if the experience should be performed on line. One of the main issue when dealing with complex 3D models is the correct balance between quality and size, in order to of assure a good user experience without compromising the website usability. Archaeology is one of the fields that can benefit from the development of agile WEB 3D solution, since the dissemination and fruition of cultural goods is strictly related to the use of digitizing tools and smart visualization system. Through these tools it's possible to achieve the purpose of spreading complex information, by communicating the data collected in the field with through digital photogrammetry techniques to achieve a reality based result. In this paper we face with problems using as a case study the Huaca Esmeralda, an Archaeological building located in Peru. In particular, our tool will give the possibility for a virtual visit, from different points of view, of the entire archaeological complex. The web site will also give the possibility to have a close up visualization, with a high resolution, the various bas-reliefs scattered among the building.

II. Related Works

The potential of three-dimensional models representation applied to archaeology is growing its interest in research and application. One of the most viable alternatives used involves 3DHOP's software. 3DHOP (3D Heritage Online Presenter) is an open-source software package used for the creation of interactive Web presentations of high-resolution 3D models, oriented to the Cultural Heritage field. 3DHOP target audience goes from the museum curators with some IT experience to the experienced Web designers who want to embed 3D contents in their creations, from students in the CH field to small companies developing web applications for museum and CH institutions [2].

In the work of Capetown University, Zamani project, they capture the spatial domain of heritage, with a current focus on Africa, by accurately recording its physical and architectural structures, their dimensions and their positions. Sites are seen in the context of their physical environment, and wherever possible, the topography of landscapes surrounding the documented sites is mapped based on satellite images and aerial photography [3].

Barry Kemp (2011) presents a series of significant objects form the Egyptian site of Amarna. The digital objects are part of the Virtual Amarna Museum. A range of objects were involved - including stone stele, ceramics, pendants, moulds and selected architectural elements. It argues that the acquisition and creation of digital representations of heritage must be part of a comprehensive research infrastructure (a digital ecosystem) that focuses on all of the elements involved,

including recording methods and metadata, digital object discovery and access, citation of digital objects, analysis and study, digital object reuse and repurposing, and the critical role of a national/international digital archive [4].
The advantages of using this type of method are the fast startup time and the reduced network load. The model is immediately available for the user to browse, even though at a low resolution, and it is constantly improving its appearance as new data are progressively loaded. On the other hand, since refinement is driven by view-dependent criteria (observer position, orientation and distance from the 3D model), only the data really needed for the required navigation are transferred to the remote user. 3DHOP supports currently only the PLY format, but soon OBJ format will be supported. 3DHOP is still not a mature project compared to other solutions already implementedAlthough 3DHOP could have been a great alternative to address the work described in this paper, the instrument used has been Three.js, more mature than 3DHOP and considered one of the major successful alternatives. Three.js is very supported by community and this reason, for all developers, is a point in favor of this framework. It also supports multiple three-dimensional models format, like the *.obj*, used in this work.

III. Materials and methods

A. Basic libraries

The work has been realized through the use of some basic libraries for the 3d Web rendering.

1) WebGl: Khronos describes WebGL thus:
WebGL is a cross-platform, royalty-free web standard for a low- level3D graphics API based on OpenGL ES2.0, exposed through the HTML5 Canvas element as Document Object Model interfaces. OpenGL ES ("Embedded Systems") 2.0 is an adaptation of the standard OpenGL API, designed specifically for devices with more limited computing power, such as mobile phones or tablets. WebGL is designed to use, and be used in conjunction with standard web technology; thus, while the 3D component of a web page is drawn with the WebGL API via Javascript, the page itself is built with standard HTML [5].

2) Three.js: The Three.js library provides a easy-to-use JavaScript API based on the features of WebGL and a large number of features and APIs that can be used to create 3D scenes directly in the browser. It provides a high abstraction level hiding to the developer the execution of the WebGL calls and the ways in which the information is sent to the graphics card. To be able to display anything with Three.js, we need three things: A scene, a camera, and a renderer so we can render the scene with the camera. The scene variable is a container that is used to store and keep track of all the objects that we want to render. The renderer is responsible for calculating what the scene will look like in the browser based on the camera angle. There are different renderers type and the one of the best performance, if the browser supports WebGL, is WebGLRenderer otherwise Three js offers CanvasRenderer. The camera determines what we'll see in the output so it projects from 3d world onto the 2d surface of computer monitor. There are a few different cameras in Three.js and the most used is PerspectiveCamera, figure 1 [6].

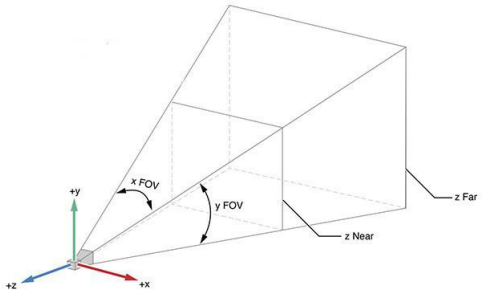

Fig. 1. Perspective camera description

3) Tween.js: Tween.js is a open-source JavaScript library that we can use to define the transition of a property between two values. All the intermediate points between the start and end value are calculated. This process is called tweening. Tweening is an abreviation of inbetweening, The tween will simply generate the needed values needed inbetween two limits. This is done with a function. The most obvious one is to go straight from source to target, this is called the linear function. There are several standard function available to animate objects [7].

B. 3D models

3d modelling is the process of developing a mathematical representation of any three-dimensional surface of an object (either inanimate or living) via specialized software in 3D computer graphic. The product is called a 3D model. It can be displayed as a two-dimensional image through a process called 3D rendering or used in a computer simulation. 3D objects used in this work were provided by the Department of Information Engineering. Each of them is featured by two separate files, respectively in *.obj* format (object) and *.mtl* (material file library). An object file (obj) is a format developed by Wavefront Techonologies used to define the geometry and other graphics properties about a general objects. By this format all the information for the definition of lines, polygons, curves and freeform surfaces can be listed. Lines and polygons are described by their vertex, curves and surfaces are defined by special check points and by other parameters that depend on the curve type. Often a *.obj* file is used to graphics objects interchange between different visualisation platforms. Material library files contain one or more material definitions, each of which includes the color, texture, and reflection map of individual materials. The material is like the skin of the object, which defines what the outside of a geometry looks like. The obj file must call the *mtl* file referenced to itself using the keyword "mtllib"Inside a *mtl* files are defined all the materials for the reference scene. In this work some 3D objects are been pre-processed by MeshLab, an open source software to edit and to process three-dimensional objects. In this circumstance two specific operations to 3D objects have been applied;

the first was about rotation around axes to display objects correctly, the second was to shift 3D objects in the origin of the axes.
To apply textures to the bas-reliefs scattered among the building and to export the model in *obj* format (the initial 3D model was in *3dm* format, not compatible with Three.js) was used the software Rhynoceros. The first thing can be achieved with these steps:

1) open the model;
2) drag the corresponding texture on the part of the model that represent the bas-relief.

To successfully export the file, so as to make it compatible with MeshLab and Three.js, we must do the following steps:

1) in the Rhynoceros command line write "_SaveAs" and press Enter;
2) write the name of the model and select the *obj* format, select also "Save textures" and click on "Options";
3) select "Export material definitions", unselect "Wrap long lines" and press "Ok" to save the file.

At the end of these steps we have the *obj* and the associated *mtl* file fully compatible with Three.js.

C. Three.js usage

The project's implementation is inspired by Three.js online examples. Below some library's key elements used are introduced.

1) Control: Control is the element that allows users to interact with the objects on the scene by input devices such as keyboard and mouse. Three.js has a number of camera controls that you can use to control the camera throughout a scene. OrbitControls and TrackBallControls are the most used. The first simulates a satellite in orbit around a specific scene and allows to move around with the mouse and keyboard. The second allow to use the mouse (or the trackball) to easily move, pan, and zoom around the scene. The mainly difference between both is that OrbitControls enforces the camera up direction, and TrackballControls allows the camera to rotate upside-down. The control used in this work is the OrbitControl. It was necessary to edit a library section to optimize the the bas-reliefs navigability. In particular has been added a function (setLimite(min, max)) that prevents the camera, controlled by the user, to left the the scene's portion in which objects are visible.

2) Raycaster: In Three.js, it is quite common to detect when a user clicks an object in 3D space. The raycaster element allows to find objects which intersect the outgoing ray that leaves from the camera and through mouse cursor's direction. In the figure below, the raycaster would build a two elements vector sorted by distance from the origin of the ray. In this work, this approach has been used for the bas-reliefs clicks management on the structure. One transparent clickable panels has been put on top each bas-relief and it has been associated them by an identifier. The panels hover makes them highlighted(we use a THREE. Raycaster object, returned from the projector. pickingRay function, to send out a ray into the world from the position we've clicked on, on the screen). By intersectObjects function the raycaster builds an array containing all the meshes intersected by the ray. Although the ray intersects more than one object (for example: a clickable panel ahead the structure) only the click on the closer object is managed. In case a panel is selected, the corresponding 3D bas-relief model is loaded.

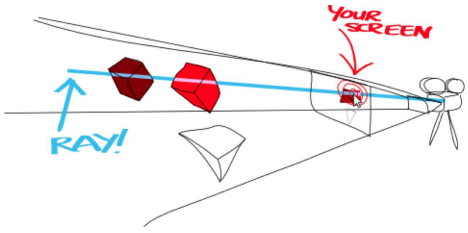

Fig. 2. Raycaster description

3) Light and realism: There are several lights available in the Three.js library that have specific behavior and usage. In this work have been used two types of light: DirectionalLight and HemisphereLight. A DirectionalLight source can be seen as a very far away light. All the light rays that it sends out are parallel to each other. In this case it was used to simulate the sun light. The sun is so far away that the light rays we receive on Earth are almost parallel to each other. HemisphereLight is a special light and can be used to create a more natural-looking outdoor lighting by simulating a reflective surface and a faintly emanating sky. In fact, outdoors, not all the light comes directly from above; much is diffused by the atmosphere, reflected by the ground, and reflected by other objects. The combination of these two types of lights makes the scene more realistic. The user has still the possibility to manipulate the light through keyboard hotkeys. A further contribution to realism it's given by the use of fog and shadows. The fog has the goal of making the horizon blurred and therefore a more natural-looking.

D. Animation

The animations, created by help of Tween js have been applied during the objects and camera transitions to make them more pleasant. For the swipe animation between structure and bas-relief, the structure will come out from the scene moving to the right compared to the camera view. Its comeback takes place in a specular manner compared to its release. The animation is also applied when the bas-reliefs are browsed. The switch between objects and the camera prospective change follows the Exponential. InOut trend type and the Sinusoidal. InOut trend type as shown in figure 3. Obviously it must be specified a starting point(coinciding with the current position of the model) and a destination point.

IV. Results and Discussion

The first outcome after Huaca Arco Iris work is summarised in Figure 4. The screen layout has a main area which included the Huaca Arco Iris structure with their bas-reliefs in the real position. The visitor is placed in a first person perspective view

Fig. 3. On the left the Sinusoidal. InOut trend, on the right the Exponential. InOut trend

here. Clicking, holding and moving the mouse will change the direction of view around the model. The entire element can be zoomed-in or out,within limits, using the scroll wheel on the mouse . There are two buttons on the scene: camera and full screen. Pressing the camera button, the structure will start to auto rotate until the user pushes a second time on the same icon or pushes on a bas-relief. The second button allows to change in full screen mode the user view. Pressing the letter “h” or “d” will toggle the hemisphere light or the directional light. Pressing the letter “l” or “d” will change lighting or the camera view around the model.

Fig. 4. The structure when the browser is opened. The darker areas are the bas−reliefs visible from the current point of view

Moving the mouse over any bas-reliefs, the user can note the entire bas-relief area marked by a color that stands out from everything else and a small circle hint that indicates the bas-relief name ordered by number and position (internal or external the structure’s wall). This is shown in Figure 5.

Fig. 5. The mouse is above the bas−relief number 6. Its area is highlighted and a tip is shown

Clicking on any bas-relief will cause the visitor to jump in another area which is useful to display in the middle of the scene the bas-relief selected. Clicking, holding and moving the mouse will change the model view direction and by mouse wheel the visitor can zoom-in or out to the finest details. A back button at the top left of the screen teleports the visitor to the first main area. Holding the right mouse button and moving it on the screen the user can shift the work along the four cardinal points. It’s is also possible using the arrows keys. in In the bottom left corner there is a structure’s mini-map that highlights the bas-relief selected by user. In addition The mini-map allowing the user to orientate himself in the structure. This is shown in Figure 6.

Fig. 6. The bas−relief number 6 has been selected. The slider is semi transparent until the user moving the mouse on it. The mini map indicates that you are observing the work number 6

A semi transparent sidebar is placed at the bottom of the screen. It’s became opaque and totally visible when user moves the mouse over it. it provides thumbnails about all bas-reliefs and it is focused on the current bas-relief selected. All thumbnails areclickable and the visitor can jump to any bas-relief by clicking on the desired bas-relief on the sidebar. This is shown in Figure 7 and Figure 8.

Fig. 7. The sidebar is active and the current bas−relief is highlighted

Fig. 8. The user has push on thumbnail corresponding internal bas−relief number 7. The layout switch to it

A. 3D Meshes Management

In this work were not been applied editing or simplification operations on 3d models but were been used the formats provided by the University. The main check done was to verify that each *mtl* file, linked to 3d object, specified the correct texture to apply. The aim to give to users the opportunity to navigate the structure and observe all the bas-reliefs in all details involves the use of 3d mesh that can be very heavy. This could be a problem, especially if they should be loaded and displayed on the Web browser. This problem would be very clear if all the work was available on a remote Web site

and the users could upload time to time a big amount of data. In this work, the issue is more transparent because the Web server is local and the 3D objects are loaded much faster.

B. Usability

Regarding the usability, particular attention was given to simplicity of use and to ensure that the user can reach the desired result in an accurate and complete way. With just a few intuitive steps it's possible to access inside all visitable areas of the site. Each bas-relief is highlighted on the structure so the attention can be focused on itself. All shown buttons are labeled on mouse hover with an explanatory string. A further aid to usability is given by the fact that with simple keyboard operations it is possible to change the camera position and to choose the type of light that lights up the scene. The bas-reliefs rotation angles were restricted to ensure a proper frontal visualization. Likewise zoom, horizontal and vertical displacement are bound to not give to the user the feeling that the object can leave the scene. All the scene objects are suspended from the ground for two reasons, the first is to make the user aware of their interactivity and the other is to put more emphasis by releasing them from the rest of the scene. During the switch between structure and bas-relief, the camera position is saved in such a way that it can be restored at the end of the bas-reliefs tour.

V. Conclusion

Currently the user is able to navigate the entire archaeological structure, in many directions and from different perspectives. Furthermore, the user has the chance to interact with it in order to view the various artistic compositions shown, allowing a view of all the appreciable details in the three-dimensional model. The approach used forces the user to interact almost with only the three-dimensional models(for example the bas-reliefs are accessible by clicking directly on certain points of the structure without the need of a secondary menu for their selection). One of the most significant problem is that the site is actually accessible and browsable only after the entire three-dimensional model loading is complete and that, as the current network infrastructure state, could lead the user to a long waiting time. In addition this new browsing concept in an object interactive web-site may not be easily accepted by the average of the users becaause they probably feel more comfortable with a standard web-site. The project is still a work-in-progress, and its development is currently focused in a general direction. New features could enhance the user-interaction experience and the improvement of the rendering stage, by a deeper Three.js use, might lead to a better web site usability. Future developments could lead to the use of much smaller 3D models (trying to maintain unaltered their quality) and the enhancement of the loading techniques used could reduce the user's waiting time.

References

[1] WebGL e 3D su Web, Marco Potenziani, http://vcg.isti.cnr.it
[2] 3DHOP, http://3dhop.net/
[3] Zamani Project, http://zamaniproject.org/index.php/project.html
[4] Amarna Project, http://archaeologydataservice.ac.uk/archives/view/amarna_leap_2011/overview.cfm
[5] WebGL specification; 2011, https://www.khronos.org/registry/webgl/specs/1.0
[6] Jos Dirksen, Learning Three.js: The Javascript 3D Library for WebGL
[7] Tween.js description, http://learningthreejs.com/blog/2011/08/17/tween-js-for-smooth-animation/