MWF2014: Museums and the Web Florence 2014

February 18-21, 2014 | Florence, Italy

An Advanced Solution For Publishing 3D Content On The Web

Paper

Marco Potenziani, Italy , Massimiliano Corsini, Italy, Marco Callieri, Italy, Marco Di Benedetto, Italy, Federico Ponchio, Italy, Matteo Dellepiane, Italy, Roberto Scopigno, Italy

Keywords: 3D Graphics, Cultural Heritage, Online Virtual Museums, Online 3D Content, WebGL

1. INTRODUCTION

The deployment of 3D content on the Web has gained momentum in many contexts. The Cultural Heritage (CH) field is a good example, since 3D data need to be published on the Web for many different purposes (presenting museum collections, displaying virtual reconstructions of ancient sites, showing connections between artefacts of interest, etc.); moreover, 3D data are often paired by other multimedia information. The current approach for developing 3D graphics applications on the Web is to employ WebGL, a graphics JavaScript API that is embedded in modern Web browsers. WebGL was derived from the standard OpenGL graphics library and all the major Web browsers have recently adopted it. People who access a Web site based on WebGL do not need to install any additional applications to use the interactive 3D content of the webpage. The main problem related with the use of WebGL is the baggage of technical skills the developers should master, such as Web design, JavaScript programming and knowledge of Computer Graphics (CG), in order to implement a CG Web application.

3D Heritage Online Presenter (3DHOP) is an advance technological solution that allows people with few or no programming skills to be able to publish 3D content on the Web in several forms. This technology consists of a set of components and templates for online visualization that have in common the characteristic to be easily modifiable with a minimal effort to obtain the desired final effect.

3DHOP has been developed to meet the needs of the specific user group that works in the context of CH, so it is specifically designed to be able to handle very detailed digital representations of real artifacts and at the same time to be able to present these 3D models with an intuitive and user-friendly interface. However, even if it has been already used in different Virtual Museum projects, this tool can be useful in many other fields where high-fidelity popular presentations of virtual models is required (for example in scientific visualizations in the medical field).

The most interesting peculiarity of 3DHOP, i.e. the ability to work with gigantic 3D meshes (tens of million triangles) in an environment rich with performance bottlenecks like the Web, is made possible by using the JavaScript implementation of the Nexus [6] technology, a view-dependent multi-resolution scheme.

Ease of use is achieved thanks to the use of a declarative-style scene setup, composed of basic blocks for building the interactive visualization, each one configurable, but with sensible, ready-to-use defaults. Moreover, the modular structure of the 3DHOP makes it accessible at different levels, so as to be used effectively by users with heterogeneous backgrounds in software development.

3DHOP is based on Web technology, like the WebGL component of HTML5, and SpiderGL [8], a JavaScript support library oriented to CG programming. Thanks to this, 3DHOP works without the need for plugins on most modern browsers (Chrome, Firefox, Internet Explorer and Opera) on all platforms.

3DHOP has been recently released as open source (GPL license), and it is already available to be used (http://vcg.isti.cnr.it/3dhop). Although 3DHOP is at its first release, it is already a mature tool due to the fact that it has been employed in several CH projects (Figure 1). So, we hope that it will become a solid tool in the CH community, reducing the learning curve and speeding up the implementation of Virtual Museums and online interactive presentations.

— Figure 1: an example of 3DHOP real usage (presenting a collection of 3D objects for the “Virtual Museum Of Sculpture And Architecture” in Pietrasanta, Italy).

2. RELATED WORK

As three-dimensional content became a consolidated type of multimedia, its visualization in the context of webpages became an issue, since 3D models were not considered to be a “native” type of data. Hence, their visualization and use was devoted to embedded software components, such as Java applets or ActiveX controls [1]. This led to a lack of standardization, and to a quite limited use of this type of content for many years in the past. The efforts linked to Virtual Reality Modelling Language (VRML) [18] action, started in 1995, and of the more recent X3D [10] (2007), were a first step to finding at least a common format for data. However, 3D scene visualization was still delegated to external software components.

The advent of WebGL standard [17], promoted by the Khronos Group [15], brought to a remarkable change. WebGL, which is a mapping of OpenGL|ES 2.0 specifications [16] in JavaScript, allows Web browsers to access graphics hardware directly. WebGL has been the starting point for a number of actions for implementing advanced 3D Graphics on the Web: a very interesting and updated version is provided by the survey from Evans [11], which presents and compares a number of technological solutions.

From a general point of view, the proposed solutions can be divided in two groups: the first one extended the effort of X3D by structuring the description of three dimensional content in a declarative fashion [13], essentially based on the concept of a scenegraph. Two interesting examples of declarative programming solutions are X3DOM [2, 3] and XML3D [19]. In contrast to the declarative approach, the imperative approach considers computation as “a series of statements that change a program state”. A number of high-level libraries have been developed to help non-expert users using WebGL. Most of them are based on the use of JavaScript as a basic language. They range from scene-graph-based interfaces, such as Scene.js [14] and GLGE [4], to more programmer-friendly paradigms, such as SpiderGL [8] and WebGLU [7]. The most successful of these libraries is Three.js [9], which has been used in several small and medium projects. The comparison between declarative and imperative approach is not trivial, since none of them is able to perform better in all the possible cases of application. Evans [11] points out that declarative approaches have had a major impact in the research community, while imperative approaches have been mainly used in the programming community.

From a more general point of view, the system presented in this paper deals also with the issue of integrating 3D models with other types of data, such as text or images. This has been recently taken into account by works [12, 5]

which explored the possibility of integrating text and 3D models on the Web. The Smithsonian's X3D explorer (http://3d.si.edu) is an alternative example where 3D models are associated with additional content, but the structure and flexibility of the proposed system are not known.

Other examples of tools nowadays used widely for displaying interactive 3D content on the Web are represented by Unity 3D and Sketchfab. Unity 3D (http://unity3d.com) is a full-fledged gaming engine, extremely powerful and complete, providing advanced rendering, sound, physics and a lot of pre-defined components and helpers. By using Unity, it is possible to build interactive visualizations at the same level of graphics and interaction complexity of a modern videogame. It is characterized by a rapid development time when creating a simple visualization, but the difficulty of use increases quickly with the more complex interaction features. Moreover, Unity is not well suited to manage high-res geometry (except for terrains) and its streaming capabilities are a paid feature, requiring server side computation. Finally, the interconnection of 3D visualization with the webpage is possible but it is one of the more complex features to set up. Sketchfab (https://sketchfab.com) is indeed a popular solution for fast online deployment of 3D models. It is extremely simple to use and that offers data storage. On the downside, it has strong limits on geometrical complexity, making it difficult or impossible to upload models coming from 3D scanning at full resolution. Moreover, the interaction with the 3D models is only partially configurable, making it in some cases difficult to navigate the model's geometry.

3. DESIGN

The rationale underlying 3DHOP was the lack of tools to manage interactive visualization on the Web of high-resolution 3D geometries in a simple way. While, as said in the previous section, there are various products (both free and commercial) that can re-create complex virtual scenes made of simple 3D entities, they are not suitable for the management of complex 3D geometry, and are mainly aimed at skilled CG developers.

By contrast, 3DHOP can be used by anyone with basic Web design skills, without the need for specific CG expertise. One of the main goals of this project has always been to create a tool that is easy to use and easy to learn, and so each design chosen has been taken in this direction. In the next subsections we will provide an overview of 3DHOP's main features, starting from this fundamental milestone.

3.1 Declarative scene setup

3DHOP is thought to bridge the gap between the worlds of declarative and imperative programming, by providing a tool that combined the ease of use of the declarative style with the potential provided by low-level programming.

In fact, if the use of the declarative approach in the development of a 3D Web

application mimics the way which the rest of the webpage is composed and managed (3D entities are declared and controlled as to be part of the DOM structure like, for example, an HTML DIV), conversely, imperative language for 3D applications works in a way that is more similar to the implementation of a stand-alone visualization software. Usually, this last type of language is close to the low-level programming that allows use of the capabilities of the graphics card. Of course, things are never black and white, and a lot of effort has been spent on both sides to reduce the respective advantages/disadvantages.

3DHOP is a mixed solution: in effect the core of the rendering is based on an imperative language (SpiderGL) and exploits low-level programming for efficiency, but the creation of the scene is similar to a declarative style, enabling a quick and intuitive (yet, highly customizable) deployment. The scene setup in fact is subdivided into a few logic blocks (*meshes, model instances, trackballs, space*), that are nothing but text fields, where the matching between the keywords and the values chosen by the user takes place.

3.2 Exhaustive default

Another essential design feature of 3DHOP is the ability to provide, by default, sensible values for each setting's parameters. If the user, starting from the empty example, only specifies the name of the 3D model and does not specify/configure anything, the components act to produce a result that make sense in the majority of the cases. The basic view, the scene composition and the interaction settings will be configured automatically. Each component is configurable by the user, but it is never mandatory to modify/update each parameter; the developer may just change a single parameter, and rely on defaults for the rest.

This smart feature obviously aims to simplify the life of the developer providing ready-to-use visualization components and a fast start-up for new users. In this way, the tool is much more accessible, and can be easily learned gradually using the examples provided.

3.3 Access levels and modularity

3DHOP was born as a tool capable of helping users without specific CG skills to create simple but performant 3D interactive presentations up to actual Virtual Museums. However, although the initial target user was conceived as a person with a non-specialized background, with just a little knowledge of programming, its particular structure allows a wide target user range.

3DHOP technology is in fact distributed on different kinds of files, each one customizable depending on the user skills. It is possible to distinguish three separate levels of usability for the viewer, and so three different groups of files upon which to operate.

The first one, dedicated to the naïve user (one with just some Web

programming experience), allows the entire 3D scene to be set by just modifying the HTML file containing the declarative scene descriptor, and if desire a few other parameters in the CSS file.

The second one, conceived for the medium skilled user (a Web designer for instance), allows editing of all the JavaScript and CSS files within the viewer initialization parameters, so a more complete control of all the final look of the Web application.

Finally, we have the group of people capable of working at the library level; this requires solid CG knowledge and WebGL expertise. People in this group can customize basically everything, for example, design a new trackball to account for specific navigation/interaction needs.

3.4 Online and offline

While the 3DHOP tool has been developed for the use online, it is fully possible to use it for the creation of multimedia kiosks and interactive displays running on local machines inside a museum or an exposition. Given its minimal interface, compatible with touchscreens and the ability to work without a dedicated server, 3DHOP is a good candidate to create kiosk-based interactive multimedia presentation.

4. ADVANCED FEATURES

As said, 3DHOP was designed in order to be used with ease. However, this does not prevent this tool being used in a more advanced mode. To this aim a series of exposed functions and event handlers were added, usable by a developer to make the 3DHOP visualization interact with the rest of the webpage logic. The next subsections will provide an overview of 3DHOP's advanced designed features.

4.1 Interactions

The user's interaction with the virtual scene is one of the most important components in a 3D viewer. 3DHOP uses the object-in-hand metaphor: the camera is fixed and the object is manipulated by the user in front of it, generally using a trackball.

It is difficult, if not impossible, to create a single all-purpose trackball, able to cope with the specific geometric characteristics of every possible object. For this reason some different basic trackballs have been created, allowing the user to pick the more appropriate one. In the current moment, distribution includes three different trackballs (full-sphere, turn-table and pan-tilt), but more trackballs will be added in the future. The turn-table is the default trackball. With this trackball is possible to reach almost all view positions around an object in a simple way, maintaining its verticality (it provides rotation around the vertical axis and tilting around the horizontal axis). The pan-tilt, on the other hand, is tailored to be used to present bas-reliefs or objects whose detail is

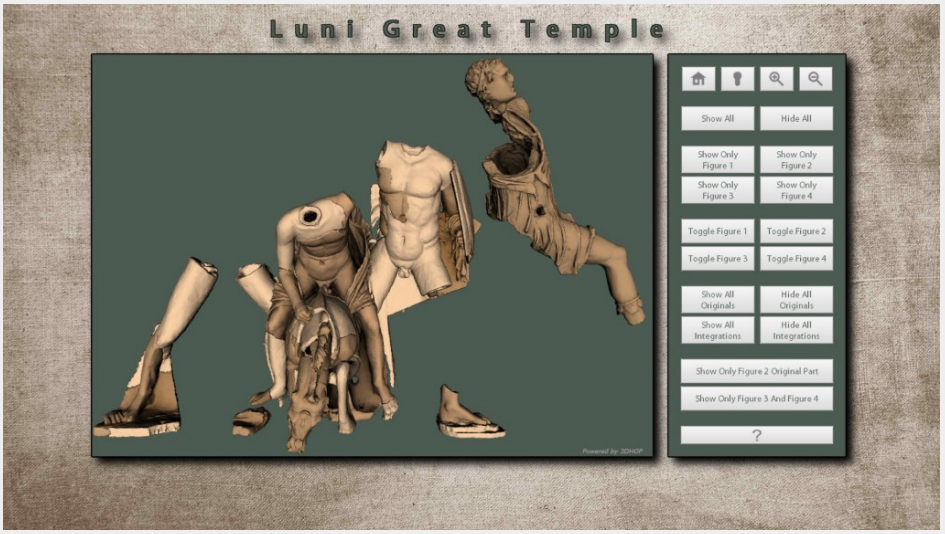
mostly located on a single plane. The full-sphere is the trackball providing the more free interaction, enabling the user to rotate the object around all axis at the same time.

These trackballs, distributed is a separate file to make so easier to use them as codebase, are configurable with limits on their axis and able to give feedback on their current position. An exposed JavaScript function ("getTrackballPosition") returns a structure containing the current state of the trackball, while using another JavaScript function ("setTrackballPosition") can be used to instantly move the trackball to a specific position by feeding it a new state description. Additionally, it is possible to make the trackballs animate to a certain position: instead of instantly changing its state, a smooth animation is played from the current position to the specified one.

4.2 Visibility

Most presentation tools implement the control of the visibility of the different models. Model instances in 3DHOP can be configured in order to be visible or invisible at startup (visible is the default), and their visibility status can be changed at runtime using specific JavaScript functions exposed by the tool.

An interesting solution is the tag-based selection of groups: in order to make it possible for the user to turn groups of objects visible and invisible, the visibility function does not work on a single instance, but on all instances that have a specific tag. Model instances have a tags property, which is basically a series of strings. Using this simple mechanism, it is possible to address single entities as well as groups (Figure 2). 3DHOP exposes a function to set visibility ("setInstanceVisibility") and another one to toggle the visibility ("toggleInstanceVisibility") of a set of instances. Finally, a tag constant value ("HOP ALL") is used to select all the instances.



— Figure 2: showing different switchable modules of a composite 3D model (the different color of the statues in figure represent two different visibility groups, the final user can choose which one to view).

4.3 Picking

The last (but not least) advanced feature described here is related to the possibility to pick the geometries in the virtual scene. This feature is often connected to something happening in the 3D visualization or elsewhere in the webpage. Depending on the visualization scheme, it may be interesting to be able to detect a click on a hotspot, but also to detect a click on a model instance.

3DHOP does support both of these two levels of interaction through two exposed handle functions. The first of these (hooked to “onPickedInstance”) is invoked every time a model instance has been clicked, and returns the name of the picked instance. The second one instead (hooked to “onPickedSpot”) is invoked every time a spot is clicked, again returning its name. The developer may decide to register one or both of these functions, and to assign a different semantics to each one, depending on its need.

In order to be more flexible, instead of just a single point, a spot may have an arbitrary shape and geometry. This is obtained by associating a 3D model to the spot, similarly to the way a 3D model is specified when declaring an instance. Spots may be made active or inactive using a tag-based mechanism similar to the one used in the visibility control, making it possible to define “hotspot groups” which can be independently activated/deactivated (Figure 3).

— Figure 3: introducing a 3D model with interactive related hotspots (in blue are visible the activated clickable hotspots).

5. IMPLEMENTATION

3DHOP has been created using standard Web technologies (like JavaScript and HTML5), without relying on external components; it can run on all the most widely used browsers (Chrome, Firefox, IE and Opera) and all the principal OS (Win, Mac-OS and Linux) without the need for plugins. It uses general Web

programming mechanisms, and the various components of the viewer (the 3D meshes, the trackballs, the scene) are treated like HTML entities in the same way as the rest of the structural elements of the webpage, and are declared and customized using simple parameters.

3DHOP is based on WebGL (as said the standard API for the CG on the Web developed as the JavaScript equivalent to OpenGL|ES 2.0), and on two main libraries: SpiderGL and Nexus.

SpiderGL (http://spidergl.org) is a JavaScript utility support library for WebGL, which provides typical structures and algorithms for real-time rendering of 3D graphics Web applications, without requiring compliance with some specific paradigm (i.e. scene graph) nor preventing low-level access to the underlying WebGL graphics layer. More specifically SpiderGL includes mathematical routines and algorithms, linear algebra functions, geometric structures, mesh importers, user-interface event handlers and much more…

Nexus (http://vcg.isti.cnr.it/nexus) is the technology that supports the streaming of high-resolution 3D meshes over the HTTP protocol, enabling the exploration of very large models (millions of triangles) even on commodity computers with standard internet connections. This JavaScript library implements a multiresolution scheme able to make the rendering and the data transfer efficient even on through the Web. This multiresolution scheme splits the geometry of the 3D model into smaller chunks. For each chunk, multiple levels of detail are available. Transmission is on demand, requiring only the loading and rendering of the portions of the model strictly needed for the generation of the current view. 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.

The Nexus package also provides the tools to convert a single-resolution 3D model into the multi-resolution representation supported. This is a one-time operation done in preprocessing. 3DHOP supports single-resolution models too (currently only in PLY format, but soon OBJ format will be supported).

Finally, we underline that a third library (jQuery) is occasionally used to manage the interaction between HTML webpage elements and 3DHOP.

6. FUTURE DEVELOPMENTS AND CONCLUSIONS

The first version of 3DHOP was released just few months ago, so it is an already usable tool. However, the project is still a work in progress, and its

development is currently focused in two main directions.

The first one concerns the evolution of the present component for the interactive visualization of high-resolution 3D models. Many new features can be added to this powerful viewer, some of which are already in testing phase. The ongoing works regards the enrichment of the user-interaction experience (adding new trackballs, touch gesture supports, re-design of the hot-spots and model selection, etc.) and the improvement of the rendering stage (providing different configurable shaders and support for textured models).

The second direction concerns the evolution of 3DHOP towards the management of other kinds of 3D data, in particular, the integration of the existing toolbox with new visualization components for the handling of different 3D media, such as, for example large 3D terrains or re-lightable images.

In conclusion, with 3DHOP we hope to offer a novel and versatile solution for publishing 3D content on the Web, facilitating deployment to non-experts in 3D programming and making it possible for users to access high-resolution 3D resources. This new technology, particularly suitable for the development of online Virtual Museums, should encourage the global dissemination and deployment of high-quality 3D cultural heritage content via the Internet. However, due to space limitations this paper introduces only a glimpse of the full potential and of all the features of 3DHOP. For more exhaustive information about this versatile tool we recommend visiting the official website (http://vcg.isti.cnr.it/3dhop), where the download package, a series of tutorials and some running examples can also be found.

REFERENCES

[1] ActiveX Controls, Microsoft. http://msdn.microsoft.com/it-it/en-es/library/aa751968

[2] Behr, J. (2009) and P. Eschler, Y. Jung, M. Zollner. “X3dom: a dom-based html5/x3d integration model”. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D 09. New York, NY, USA: ACM, 127–135.

[3] Behr, J. (2012) and Y. Jung, T. Franke, T. Sturm. “Using images and explicit binary container for efficient and incremental delivery of declarative 3d scenes on the Web”. In *Proceedings of the 17th International Conference on 3D Web Technology*, Web3D 12. New York, NY, USA: ACM, 17–25.

[4] Brunt, P. (2010). “GLGE: WebGL for the lazy”. http://www.glge.org

[5] Callieri, M. (2013) and C. Leoni, M. Dellepiane, R. Scopigno. “Artworks narrating a story: a modular framework for the integrated presentation of three-dimensional and textual contents”. In *18th International Conference on 3D Web*

Technology, Web3D 13.

[6] Cignoni P. (2005) and F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio, R. Scopigno. “Batched multi triangulation”. In *Proceedings IEEE Visualization.* Minneapolis, MI, USA: IEEE Computer Society Press, 207–214.

[7] DeLillo B. (2009). “WebGLU: A utility library for working with WebGL”. http://sourceforge.net/projects/webglu

[8] Di Benedetto M. (2010) and F. Ponchio, F. Ganovelli, R. Scopigno. “Spidergl: a javascript 3d graphics library for next-generation www”. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D 10. New York, NY, USA: ACM, 165–174.

[9] Dirksen, J. (2013). In J. Dirksen (Ed.) “*Learning Three.js: The JavaScript 3D Library for WebGL”*. Packt Publishing.

[10] Don Brutzmann, L.D. (2007). “*X3D: Extensible 3D Graphics for Web Authors”*. Morgan Kaufmann.

[11] Evans, A. (2014) and M. Romeo, A. Bahrehmand, J. Agenjo, J. Blat. “3d graphics on the Web: a survey”. *Computers & Graphics*, 41(0):43 – 61.

[12] Jankowski, J. (2012) and S. Decker. “A dual-mode user interface for accessing 3d content on the world wide Web”. In *Proceedings of the 21st international conference on World Wide Web*, WWW 12. New York, NY, USA: ACM, 1047–1056.

[13] Jankowski, J. (2013) and S. Ressler, K. Sons, Y. Jung, J. Behr, P. Slusallek. “Declarative integration of interactive 3d graphics into the world-wide-web: Principles, current approaches, and research agenda”. In *Proceedings of the 18th International Conference on 3D Web Technology*, Web3D 13. New York, NY, USA: ACM, 39–45.

[14] Kay, L. (2009). “SceneJS”. http://scenejs.org

[15] Khronos Group. (2009) “Khronos: Open Standards for Media Authoring and Acceleration”.

[16] Khronos Group. (2009) “OpenGL ES – The Standard for Embedded Accelerated 3D Graphics”.

[17] Khronos Group. (2009) “WebGL – OpenGL ES 2.0 for the Web”.

[18] Raggett, D. (1995). “Extending WWW to support platform independent virtual reality”. *Technical Report.*

[19] Sons, K. (2010) and F. Klein, D. Rubinstein, S. Byelozyorov, P. Slusallek. “Xml3d: Interactive 3d graphics for the Web”. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D 10. New York, NY, USA: ACM, 175–184.

Cite as:
M. Potenziani, M. Corsini, M. Callieri, M. Di Benedetto, F. Ponchio, M. Dellepiane and R. Scopigno, An Advanced Solution For Publishing 3D Content On The Web. In *Museums and the Web 2013*, N. Proctor & R. Cherry (eds). Silver Spring, MD: Museums and the Web. Published May 30, 2014. Consulted September 12, 2014 .
http://mwf2014.museumsandtheweb.com/paper/an-advanced-solution-for-publishing-3d-contents-on-the-web/
