CEIG - Spanish Computer Graphics Conference (2016)
Alejandro García-Alonso and Belen Masia (Editors)

A Web 3D-Scene Editor of Virtual Reality Exhibitions and 360-degree videos

Vicente Martínez, Alba M. Ríos and F. Javier Melero

University of Granada, Spain

Abstract

This work consists in the development of a Web-based system which allows museum managers to generate their own virtual 3D exhibitions, using a two-dimensional graphical user interface. These virtual 3D exhibitions can be displayed interactively in the museum's website, or as a immersive experience in a virtual headset (e.g. Google CardBoard) via a mobile app. We use the SVG specification to edit the exhibition, handling 3D models of the rooms, sculptures and pictures. Furthermore, the user can add to the scene a list of cameras with their respective control points, and then generate several routes through the scene. The scene is rendered in the browser using standard technologies, such as WebGL (ThreeJS) and X3D (X3DOM), and the mobile app is generated via Unity3D . Also, the X3D and MPEG-4 compression standards allow us to transform the scene and its camera routes into 360° videos, where user can manipulate camera orientation while playing the track. Also, audio tracks can be added to each route and hence, inserted in the 360° video.

1. Introduction

During the last years, the virtual reality sector has had a huge upturn, especially in the entertainment industry. Although it is not its only scope since in such fields as medicine, scientific simulation or commercial sector it has become a crucial tool to work with. In this project, this technology is focused on the cultural sector, particularly in automating virtual exhibition generation for museums. Until now, museums have allowed from their websites to experience virtual tours, similar to Google Street View, in which the user could follow a fixed path through a room by clicking on the pictures. This project aims to give users greater scope for moving freely in rooms, buildings or any virtual scene and to interact with the exhibition objects more easily. It is also intended to offer the possibility of generating interactive routes through the use of 360-degree videos. This is cutting-edge technology and has been already adopted by tech giants such as Facebook or Youtube. By working towards a more immersive and realistic experience, we also advance work on virtual reality technology, a field that promises to be the next major computing and communication platform.

2. Creation of the exhibitions with SVG

The exhibitions' editor consists of a HTML5 canvas to which the users will place an exhibition room floor map and then will add sculptures, paintings and other objects and textual information. All those elements are added as nodes to a SVG DOM, and each element is modelled as a set of attributes such as identifier, coordinates and orientation. Using Javascript functions attached to the mouse behaviour, the 2D element attributes are modified so that all nodes are kept updated during execution time, and rendered coherently in the canvas.

The translations of the elements are carried out by the Drag and Drop event. As each available room has its own dimensions, as well as the canvas, we must scale properly the coordinates up so that the user moves in pixel coordinates but the elements are translated in real world coordinates. As each museum piece can have any arbitrary orientation, and the SVG nodes use a local coordinate axis when they perform the translation and rotation, it is necessary at first to align the room and artwork 2D coordinates, in order to prevent the wrong direction of the displacement. Users can also rotate elements in the exhibition, by selecting the element icon and performing a drag and drop operation over the rotation icon that is displayed. These actions modify in real-time the SVG rotation attribute. Also, when an object is selected, a delete icon is displayed nearby to remove it from the DOM if desired [Sch16].

3. Coordinates interpolation

We use a SVG canvas. The room floors images do not normally fit this SVG canvas. Therefore it is necessary to perform a visual scale not only of the room 2D image, but also of the objects icon. As the rooms can have any arbitrary shape and dimension, we request the user to mark upper-left and bottom-right points of the valid area of the room. Moreover, the user must also indicate the real dimensions of the room when including the map, icon and 3D shape on the rooms library.

DOI: 10.2312/ceig.20161308

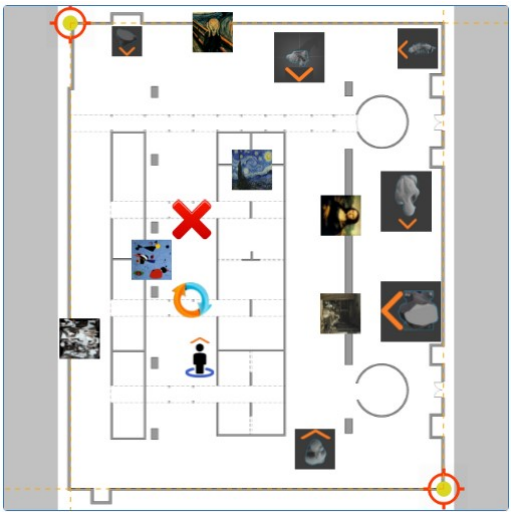

Figure 1: *The figure shows a museum room containing different sculptures and pictures. Over the selected picture appears the conceptual menu that allows the user to remove or rotate the object. The whole canvas is generated by using the SVG technology.*

4. Sculptures display

With the exhibitions' editor the user has the opportunity of visualizing the chosen sculpture by using ThreeJS, a WebGL based Javascript library. When an user selects an object from the library slider or from the SVG canvas, the object is rendered in a separate 3D viewer. Most of the sculptures 3D models have been obtained from web pages which allow their free download. These models can also be obtained through the 3D scanning of those sculptures. The sculptures and rooms must be uploaded to the system in the X3D and OBJ formats [Thr16].

Figure 2: *HTML5 canvas where a 3D model is rendered using ThreeJS. In this case we are displaying the Venus de Milo sculpture.*

5. Detection of walls to insert paintings

A key point to properly place the paintings on the exhibition is to detect the walls and columns of the room. We perform an analysis of the corresponding X3D file of the room. We detect the floor by computing the normal values of each polygon, and selecting as wall or column those with a normal value near to horizontal. We also determine an horizontal band at each wall where the paintings must be hung. As the user moves the mouse in a 2D map of the room, we can determine by the picking position only the X,Z coordinates of the painting position, but the Y coordinate is located within that band at a visible height. The paintings are added to the system from the images which are used as textures for simple 3D models such as scaled hexahedrons.

6. Dynamic Generation of exhibitions with X3DOM

The developed software allows the users to visualize the exhibitions in two different ways: on the one hand, it is possible to visualize them by using an examine third-person behaviour thanks to which all the room can be seen; on the other hand, it is possible to visualize the exhibition in a first-person view by using the game mode of the X3DOM library.

In both cases, before running the X3DOM rendering it is necessary to create the X3D file that holds all the exhibition pieces in a standard manner. We analyse the SVG DOM, traversing all its layers. Each 2D figure is transposed into the X3D DOM with the appropriate translation and rotation attributes. In the concrete case of the sculptures, it will be necessary to obtain the coordinates attributes (position and orientation) and also the height and width of the pedestal. In order to properly place the paintings, it is necessary to search the closest point of the wall to the picking position that the user marked, and then place the painting at that height. At any time, the user can dynamically start a new viewport to check in real-time the current exhibition [W3C16].

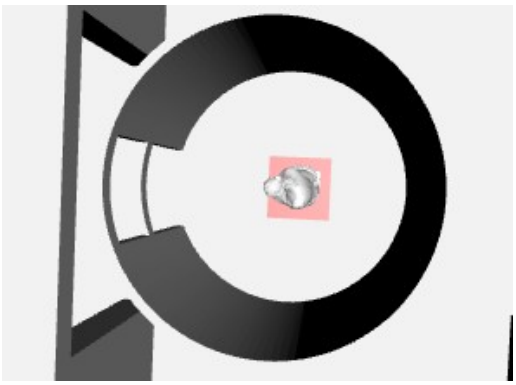

Figure 3: *The X3DOM display shows an overhead view of a museum room. A zoom allows us to identify a sculpture previously placed in the scene.*

7. Generating an X3D file for VR visualization from the web

Among the multiple functionalities which can be found within this web system, there is the possibility for a non-registered user to visu-

alize the expositions which have been created. To achieve this, the designer of the exposition can export it as a version that is adapted to goggles like those of Google CardBoard. The construction process is similar to dynamic generation, but apart from that it also requires a series of nodes and functions from X3DOM which allows for virtual reality visualization by using two cameras with a perspective which is similar to human eyes [W3C16].

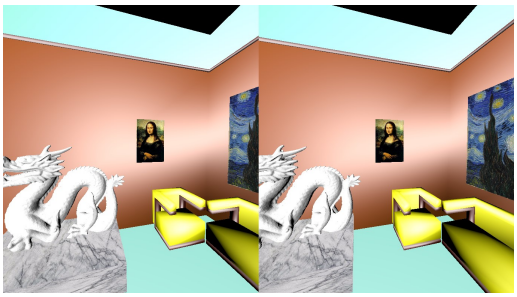

Figure 4: *Virtual reality camera view in X3DOM. We are displaying a dragon sculpture with two pictures.*

8. Virtual Reality Application

The mobile phone application has been generated by using the Unity 3D graphics engine and the Google VR SDK plug-in. The application imports the exhibitions which has been created from the web server and loads the 3D models during execution time. Heavy models are rendered using discrete multi-resolution techniques: each 3D model has three versions which will be loaded in the scene depending on the distance between the camera and the model. The movements around the scenario is performed in first person mode.

The user's interaction with the different objects of the exposition is performed by detecting when the user is looking at a concrete object. Then, if the user is close enough, the system displays the sculpture's or painting's information introduced by the manager of the exhibition through the website.

9. Route creation in 2D editor

Over every loaded scene in the editor users can activate the route generation mode, which allows to add and modify a camera path to the scene. In this mode two types of new objects are generated, both editable by the user, which are cameras and route points.

Each camera represents a full path and at the same time it represents the path's starting point. When a camera object is generated several route points can be attached to it. A route point matches up with a camera position at a specific time. This union can be visualized in the editor as a line linking the route point to the camera object, if it is the first point, or to the previous point (Figure 5).

Both the camera position and its points position can be modified in real time by selecting and dragging them with the cursor, as if they were any other scene object. If more precision is required, the position can be modified by introducing the desired coordinates in a form.

The user can just work with a route at a time as the form only shows information relative to the currently selected one. In addition, the far and near plane values can be also changed. In the same form the user must introduce the full route duration in seconds and automatically a specific time is assigned to each route point in it. As this time can be modified, it allows the user to indirectly control the camera movement speed between the points.

It is also possible to select and load an audio file from the user local machine into the web, which is later played with the 360-degree video. Once the file is loaded we display the audio wave and several buttons are also displayed to interact with the audio (play, pause, stop and remove). (Figure 6).

Information relative to every camera path is stored in a data base. An interpolation to transform the 2D editor coordinates to scene coordinates is applied after any position is stored. This way, te stored value is adapted to the real scene dimensions.

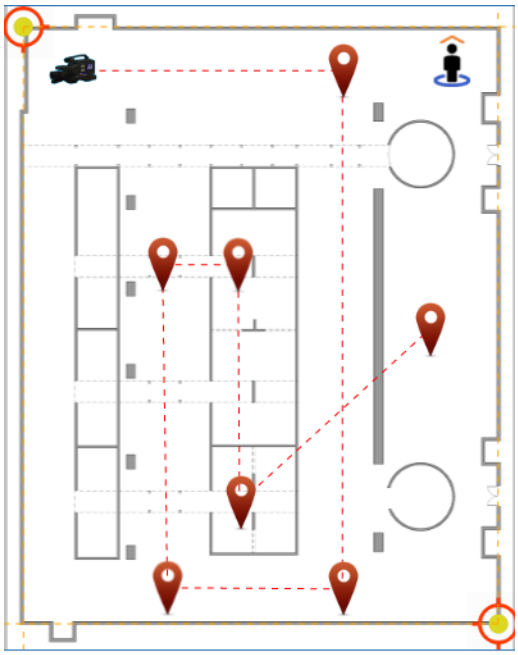

Figure 5: *Example of a camera path generated in the 2D editor over a loaded scene. The initial point is displayed as a camera and it is connected by a discontinuous line to the remaining route points.*

10. Route correspondence in X3D

In order to represent a route in a X3D file, every concept must be transferred into a valid X3D node or node attribute [W3C16].

The first approach consisted in using one viewpoint node for every control point in the path, which did not allow to modify the camera speed between points. In the end, the animation of a single viewpoint was chosen over this approach by means of transformation nodes. Camera position coordinates transformation is immediate, as the coordinate interpolation is done before storing them, as we previously stated.

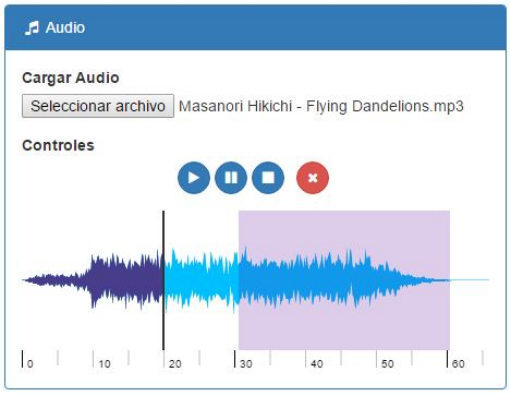

Figure 6: *Figure showing the result of loading an audio file into the web. Over it appears a coloured region that represents which part of the audio will be played during the 360-degree video.*

Table 1: *Correspondence between user-entered information and its X3D representation as a node or a node's attribute.*

Input data	X3D depiction	Name	Content
Route	Node	TimeSensor	-
Route duration	Attribute	cycleInterval	[0, inf]
Camera	Node	ViewPoint	-
Starting position	Attribute	position	[0, inf]
Near plane	Attribute	zNear	[0, inf]
Far plane	Attribute	zFar	[0, inf]
Route point	Node	Position-Interpolator	-
Point position	Attribute	keyValue	[x, y, z]
Point time	Attribute	key	[0, duration]
Audio	Node	AudioClip	-

The following table depicts the correspondence between the data base information and its representation in a X3D file (Table 1).

11. 360-degree video generation

By the emergence of the MPEG-4 standard [PE02] [MD04], there appeared the possibility of introducing 3D scenes in a video file. Taking as the starting point an existing X3D file which contains the scene and over which a camera path has been added, we aimed to generate a mp4 video file. This video duration has to match the full route duration specified in the X3D file and during the reproduction the camera looking point has to be adaptable. In this way, a user reproducing the video could dynamically interact with it by rotating the camera [NWSW05].

In order to do so, we tried to use the open source multimedia framework called GPAC [LFCM16], more specifically the MP4Box packager which can be used to codify a X3D file in a MP4 video file. However, after several tries no successful results were obtained: the resulting videos could not be played by any video player. Some of the players used were VLC, BSPlayer, MPV or GNOME Videos. After further research we came to the conclusion that 360 videos cannot be played by a regular player, instead they can just be played as Web content or a specific piece of software has to be developed.

Because of the unfortunate results obtained, we decided to develop our own Web player to simulate the expected behaviour. The X3D scene is included as XML into the HTML code of the proyect site and it is loaded when the user access the player. This player allows the user to play, stop and interact with the scene as if it was a video.

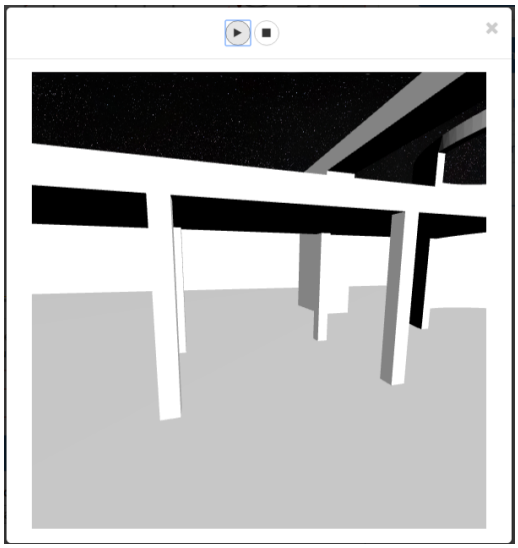

Figure 7: *Image of the route player developed as a solution for the problem with MP4 videos.*

12. Conclusions

In conclusion, as it was seen previously in the paper, this project presents a series of functionalities which allow for the creation of virtual expositions for their use with virtual reality. It is a system whose objective is to introduce museums into the virtual reality market, by giving them the opportunity to create their own expositions and also a new way to publicize them. It may also be seen as a tool which facilitates the educational labour in educational entities by allowing them to organize their own virtual tours for students. By paying special attention to the future, this software should be improved by adding new functionalities which allow for the exportation of scenarios which have been developed in other systems, as well as the improvement of the interaction between the user and the 3D environment by adding some other interaction options such as obtaining information about the sculptures and paintings, or the user's movement control through the virtual reality goggles of the web system. Also, further research can be done on the development of a software to play 360 videos or the Web solution applied can be improved.

13. Acknowledgments

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness and the EU (via ERDF funds) through the research project TIN2013-47276-C6-3R.

References

[LFCM16] LE FEUVRE J., CONCOLATO C., MOISSINAC J.: Gpac project web site, May 2016. URL: http://gpac.sourceforge.net/. 4

[MD04] MOERITZ S., DIEPOLD K.: *Understanding MPEG-4: Technology and Business Insights*. Focal Press, 2004. 4

[NWSW05] NEVE W. D., WOLF K. D., SCHRIJVER D. D., WALLE R. V. D.: Using mpeg-4 scene description for offering customizable and interactive multimedia presentations. In *Proceedings of Workshop on Image Analysis for Multimedia Interactive Services* (April 2005), WIAMIS 2005. 4

[PE02] PEREIRA F., EBRAHIMI T.: *The MPEG-4 Book*. Prentice Hall, 2002. 4

[Sch16] SCHEPERS D.: svg-whiz, May 2016. URL: http://svg-whiz.com/. 1

[Thr16] THREEJS: Threejs documentation, May 2016. URL: http://threejs.org/. 2

[W3C16] W3C: X3dom documentation, May 2016. URL: http://doc.x3dom.org/. 2, 3