Toward an immersion platform for the World Wide Web using autostereoscopic displays and tracking devices

Olivier Nocent, Sylvia Piotin, Aassif Benassarou, Maxime Jaisson, Laurent Lucas

Toward an immersion platform for the World Wide Web using autostereoscopic displays and tracking devices

Olivier Nocent, CReSTIC SIC
Sylvia Piotin, CReSTIC SIC
Aassif Benassarou, CReSTIC SIC
Maxime Jaisson, GRESPI
Laurent Lucas, CReSTIC SIC

CReSTIC EA3804 / GRESPI EA4301 - Université de Reims Champagne-Ardenne - France *

Figure 1: *Autostereoscopic technology allows 3D images popping out of the screen while natural interaction provides a seamless manipulation of 3D contents.*

Abstract

A few years ago, the introduction of the WebGL API has allowed displaying 3D content in web browsers very efficiently by using the power of 3D accelerators. Nowadays, 3D web applications, from games to medical imaging, tend to compete with their desktop counterparts. Among the main factors that can improve the feeling of presence in terms of impressiveness, immersion and natural interaction play a prominent role in enhancing the quality of the user experience. Both immersion and natural interaction rely on dedicated hardware like 3D displays and tracking devices. Unfortunately, browser makers do not supply JavaScript mechanisms for accessing hardware for security reasons. In this paper, we propose a plugin-free solution using the new features of HTML5 (WebGL and WebSockets) in order to handle autostereoscopic displays and widespread tracking devices like IR depth sensors for providing immersion and natural interaction within the web browser.

CR Categories: I.3.1 [Computer Graphics]: Hardware Architecture—Three-dimensional displays; I.3.2 [Graphics Systems]: Distributed/network graphics—; I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality; J.3 [Life and Medical Sciences]: Medical information systems—;

Keywords: Web graphics, 3D perception, autostereoscopic displays, natural interaction

*Centre de Recherche en Sciences et Technologies de l'Information et de la Communication, Rue des crayères BP1035, 51687 REIMS Cedex 2 (France). URL: http://crestic.univ-reims.fr, contact: olivier.nocent@univ-reims.fr

1 Introduction

A recurrent and key issue for 3D technologies resides in immersion. 3D web technologies try to reach the same goal in order to enhance the user experience. Interaction and depth perception are two factors that significantly improve the feeling of immersion. But these factors rely on dedicated hardware that can not be addressed through JavaScript for security reasons. In this paper, we present an original way to interact with hardware via a web browser, using web protocols by providing an easy-to-use immersion platform for the World Wide Web. This plugin-free solution leveraging the brand new features of HTML5 (WebGL, WebSockets) allows to handle autostereoscopic displays for immersion and different type of tracking devices for natural interaction (Figure 2). Because WebGL is a low-level API, we decided to develop the WebGLUT (WebGL Utility Toolkit) API on top of WebGL. WebGLUT enhances WebGL by providing extra features like linear algebra data structures (vectors, matrices, quaternions), triangular meshes, materials, multiview cameras to handle autosteroscopic displays, controllers to address different tracking devices, *etc*. WebGLUT was created at the same time (and with the same philosophy) as WebGLU [DeLillo 2010] and SpiderGL [Di Benedetto et al. 2010].

Our contribution is written as follows: Section 2 presents related work to 3D content on the web and on 3D displays. Section 3 describes the autostereoscopic display technology by providing equations and algorithms to generate multiple views. Section 4 is dedicated to our network-based tracking system. Finally, we mention very shortly in Section 5 a case study related to medical imaging using our immersion platform.

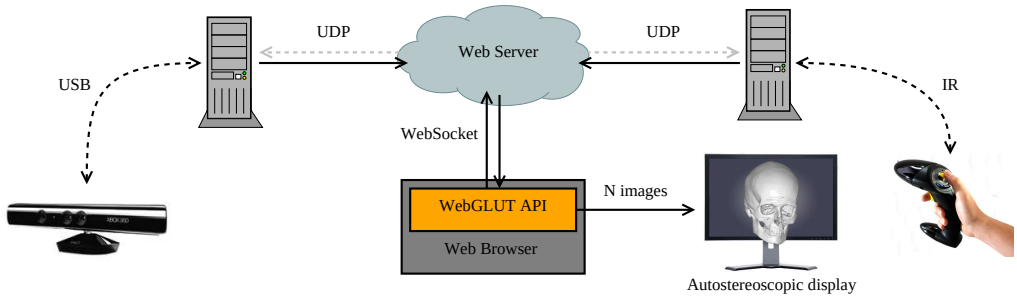

Figure 2: *Global structure of our immersion platform for the World Wide Web*

2 Related work

Web browsers have acquired over the past years the ability to efficiently incorporate and deliver different kinds of media. 3D content can be considered as the next evolution to these additions although requirements of 3D graphics in terms of computational power and unifying standard are more restrictive than still images or videos. Several technologies have been developed to achieve this integration. The Virtual Reality Markup Language (VRML) [Raggett 1994] replaced afterward by X3D [Brutzman and Daly 2007] was proposed as a text-based format for specifying 3D scenes in terms of geometry and material properties and for the definition of basic user interaction. Even if the format itself is a standard, the rendering within the web browser usually relies on proprietary plugins. But the promising X3DOM [Behr et al. 2009] initiative aims to include X3D elements as part of the HTML5 DOM tree. More recently, the WebGL [Khronos Group 2009] API was introduced to provide imperative programming mechanisms to display 3D contents in a more flexible way. As its name suggests, WebGL is the JavaScript analogous to the OpenGL|ES 2.0 API for C/C++. It provides capabilities for displaying 3D content within a web browser which was previously the exclusive domain of desktop environments. Leung and Salga [Leung and Salga 2010] emphasize the fact that WebGL gives the chance to not just replicate desktop 3D contents and applications, but rather to exploit other web features to develop richer content and applications. In this context, web browsers could become the default visualization interface [Mouton et al. 2011].

Even if real-time 3D rendering has become a common feature in many applications, the resulting images are still bidimensional. Nowadays, this limitation can be partly overcome by the use of 3D displays that significantly improve depth perception and the ability to estimate distances between objects. Therefore, the content creation process needs to be reconsidered For computer generated imagery, the rendering system can seamlessly render one or more related views depending on the application [Abildgaard et al. 2010; Benassarou et al. 2011]. 3D contents are obviously clearer and more usable than 2D images because they doe not involve any inference step. Finally, 3D contents can also address new emerging devices like 3D smartphones [Harrold and Woodgate 2007] and mobile 3DTV, offering new viable platforms for developing 3D web applications.

3 Autostereoscopic technology

The term *stereoscopy* denotes techniques where a separate view is presented to the right and left eye, these separate views inducing a better depth perception. Different solutions exist for the production of these images as well as for their restitution. For image restitution with non time-based techniques, one can use anaglyph and colored filters, polarizing sheets with polarized glasses or autostereoscopic displays [Halle 1997]. The technology of autostereoscopic displays presents the great advantage to allow multiscopic rendering without the use of glasses. Therefore, the spectator can benefit from a stereoscopic rendering more naturally, and this is especially true for 3D applications in multimedia.

3.1 Multiple view computation

The geometry of a single camera is usually defined by its position, orientation and viewing frustum as shown in Figure 3. But in stereo rendering environments, we need two virtual cameras, one for each left/right eye. And for multiview autostereoscopic displays [Prévoteau et al. 2010], we need up to N virtual cameras. Each virtual camera has a given offset position and its own off-axis asymmetric sheared viewing frustum, the view direction remaining unchanged. The near/far planes are preserved, and a focus plane has to be manually defined where the viewing zones converge. The choice of the focus distance will determine if the objects appear either behind or in front of the screen, providing or not a pop-out effect. Given a perspective projection matrix $\mathbf{P}$, the following calculations allow to identify six parameters l, r, b, t, n, f as defined in the OpenGL `glFrustum` command: l and r are the left and right coordinates of the vertical clipping planes, b and t are the bottom and top coordinates of the horizontal clipping planes, n and f are the distances to the near and far depth clipping planes. First, the distances n and f (refer to Figure 3 for the geometric signification of these terms) are given by Equation 1.

$$n = \frac{1-k}{2k}\mathbf{P}_{34} \qquad f = nk \quad \text{where} \quad k = \frac{\mathbf{P}_{33}-1}{\mathbf{P}_{33}+1} \tag{1}$$

In order to compute l, r, b and t, we need to define the half-width w_F (respectively w_n) of the image at the focus distance F (respectively at the near distance n) from the horizontal field of view α:

$$w_F = \tan(\alpha/2)F \qquad w_n = \tan(\alpha/2)n \tag{2}$$

where $\tan(\alpha/2) = \mathbf{P}_{11}^{-1}$ according to the definition of the projection matrix $\mathbf{P}$.

The viewing frustum shift for the camera j for $j \in \{1, \ldots, N\}$ in the near plane, denoted as s_n^j is given by Equation 3 where d is the interocular distance and w_s the physical screen width.

$$s_n^j = \frac{dw_n}{w_s}\left[(j-1) - \frac{N-1}{2}\right] \tag{3}$$

Finally,

$$l = -w_n + s_n^j \qquad r = w_n + s_n^j \qquad t = \rho w_n \qquad b = -t \tag{4}$$

where ρ is the viewport aspect ratio. The camera position offset along the horizontal axis is given by s_F^j, the viewing frustum shift

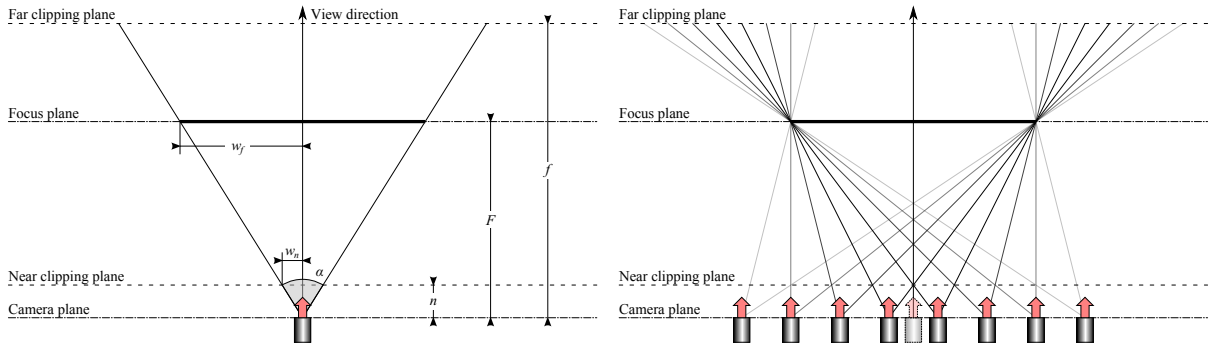

Figure 3: *Geometry of a single camera (left) and multiple axis-aligned cameras (right).*

in the focus plane.

$$s_F^j = \frac{dw_F}{w_s}\left[(j-1) - \frac{N-1}{2}\right] \tag{5}$$

3.2 Autostereoscopic image rendering

Thanks to the computations presented in the previous section we are able to produce N separate perspective views from N different virtual cameras. The use of these images depends on the chosen stereoscopic technology. One of the simplest cases relies on quad-buffering, where the images are rendered into left and right buffers independently, the stereo images being then swapped in sync with shutter glasses. Other techniques, which are not time-based, need the different images to be combined in one single image.

Let $\mathbf{I}^j$ for $j \in \{1, \ldots, N\}$ be the image generated by the virtual camera j, the color components $\mathbf{I}_c^{final}(x, y)$ for $c \in \{R, G, B\}$ of the pixel (x, y) in the final image are given by:

$$\mathbf{I}_c^{final}(x, y) = \mathbf{I}_c^{\mathcal{M}(x,y,c)}(x, y) \quad c \in \{R, G, B\} \tag{6}$$

where $\mathcal{M}$ is a mask function and R, G, B stand for red, green and blue channels. As Lenticular sheet displays consist of long cylindrical lenses that focus on the underlying image plane so that each vertical pixel stripe corresponds to a given viewing zone, the function $\mathcal{M}$ does not depend on the color component c and is simply given by Equation 7.

$$\mathbf{I}_c^{final}(x, y) = \mathbf{I}_c^{x \bmod N}(x, y) \tag{7}$$

It is worth noticing that Equation 7 clearly shows that the horizontal resolution of the restituted image is reduced by a factor $1/N$ compared to the native resolution m of the display. This resolution loss is one of the main drawbacks of autostereoscopic displays, even if it is only limited to $3m/N$ while using wavelength-selective filters because each pixel's RGB components correspond to three different view zones [Hübner et al. 2006]. Technically speaking, the WebGL implementation of the autostereoscopic image rendering is a two-step rendering process using deferred shading techniques.

Pass #1 (multiple images rendering) renders N low-resolution images by shifting the viewport along the y-axis. The N images are vertically stored in a single texture via a *FrameBuffer Object* (FBO).

Pass #2 (image post-processing) renders a window-aligned quad. Within the fragment shader, each color component of the output fragment is computed according to Equation 7 where the N images are read from an input 2D texture.

These two rendering passes are encapsulated in the method `shoot()` of the `MultiViewCamera` object.

4 Tracking system

Another characteristic of our immersion platform for the World Wide Web resides in the use of tracking devices in order to interact with a 3D scene in a straightforward way. Our aim is to address a large range of tracking devices like mouses, 3D mouses, flysticks or even more recent devices like IR depth sensors (Microsoft® Kinect, ASUS® Xtion Pro). In the same fashion we handle autostereoscopic displays, we propose a plugin-free solution to interact with *ad-hoc* tracking devices within a web browser by using HTML5 WebSockets [W3C® 2012]. The proprietary ART® DTrack tracking system is an optical tracking system which delivers visual information to a PC in order to be processed. The resulting information, 3D position and orientation of the flystick is then broadcast on every chosen IP address, using the UDP protocol. A PHP server-side script, which can be seen as a *WebSocket Server*, is running on the web server. The WebSocket server is waiting for UDP datagrams, containing locations and orientations, from the DTrack system. At receipt, the data is parsed and sent via WebSocket to the client using the JSON format. Thanks to this networked architecture, we are able to stream JSON encoded data coming from the tracking system to the web browser. The location and the orientation of the flystick are then used to control the WebGLUT virtual camera.

Using the same approach, we have also imagined a more affordable solution to interact with a 3D scene in an even more straightforward way. Indeed, we use recent IR depth sensors like Microsoft® Kinect and ASUS® Xtion Pro to perform *Natural Interaction*. Just like the ART® DTrack system, our C++ program acts as a UDP server and, at the same time, collects information about the location and the pose of a user facing an IR depth sensor. Thanks to the OpenNI framework [OpenNI™ 2010], we are able to track the user's body and detect characteristic gestures. This information, streamed over the network using our hardware/software architecture can be used to interact with a 3D scene: move the virtual camera, trigger events, *etc.* These aspects are exposed within the WebGLUT API through the concept of `Controller`. A `Controller` can be attached to any type of `Camera`. This controller is responsible for updating the properties of the camera (position, orientation, field of view, *etc.*) depending on its state change. At the time of writing, we manage three types of controllers: `MouseController`, `DTrackController` and `KinectController`.

5 Case study: ModJaw®

The ModJaw® (Modeling the Human Jaw) project has been developed by our research team and Maxime Jaisson who is preparing a PhD thesis in odontology related to the impact of *Information Technology* on dentistry practice and teaching. ModJaw® [Jaisson and Nocent 2010] aims to provide 3D interactive educational materials for dentistry tutors for teaching mandibular kinematics. There exist several similarities with a former project called MAJA (Modeling and Animation of JAw movements) [Reiberg 1997]. One main difference between ModJaw® and MAJA consists in the data nature. Indeed, we use real-world data obtained by motion capture and CT scanners. In this way, students are able to study a wide variety of mandible motions according to specific diseases or malformations. Among the future works, Jörg Reiber wanted to add an *internet connection* to MAJA. In this way, ModJaw® can be seen as an enhanced up-to-date version of MAJA relying on real-world data sets and cutting edge web technologies exposed by the brand new features of HTML5 (WebGL, WebSockets, *etc.*). The choice of web technologies to develop this project was mainly dictated by the following constraints:

Easy-to-use: users just have to open a web browser to access to the software and its user-friendly graphical interface. As it is fully web-based, ModJaw® also incorporates online documentation related to anatomy, mandibular kinematics, *etc*.

Easy-to-deploy: the framework does not require any install, it can be used from all the computers within the faculty or even from your home computer if you use a HTML5 compatible web browser. Since the software is hosted on a single web server, it is really easy to upgrade it.

6 Conclusion

In this paper, we have presented an original solution for providing immersion and natural interaction within a web browser. The main benefits of our contribution rely on the seamless interaction between a web browser and autostereoscopic displays and tracking devices through new HTML5 features like WebGL and WebSockets. This plugin-free framework allows to enhance the user experience by leveraging dedicated hardware via JavaScript. Thanks to its network-based approach, this framework can easily be extended to handle other devices in the same fashion. For instance, we began to explore the possibility to manage haptic devices. As our tracking system is completely plugin-free and fully network-based, it could be seamlessly integrated in web-based collaborative environments allowing users to remotely interact with shared 3D contents displayed in a web browser.

Acknowledgements

The authors would like to thank Romain Guillemot, research engineer at CReSTIC SIC, for his expertise in autostereoscopic displays and his precious help for porting the source code of multiview cameras from OpenGL to WebGL.

References

ABILDGAARD, A., WITWIT, A., KARLSEN, J., JACOBSEN, E., TENNE, B., RINGSTAD, G., AND DUE-TNNESSEN, P. 2010. An autostereoscopic 3D display can improve visualization of 3D models from intracranial MR angiography. *International Journal of Computer Assisted Radiology and Surgery 5*, 549–554.

BEHR, J., ESCHLER, P., JUNG, Y., AND ZÖLLNER, M. 2009. X3DOM: a DOM-based HTML5/X3D integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '09, 127–135.

BENASSAROU, A., VALETTE, G., DEBONS, D., REMION, Y., AND LUCAS, L. 2011. Autostereoscopic visualization of 3D time-varying complex objects in volumetric image sequences. In *Three-Dimensional and Multidimensional Microscopy: Image Acquisition and Processing XVIII*, SPIE, vol. 7904.

BRUTZMAN, D., AND DALY, L. 2007. *X3D: Extensible 3D Graphics for Web Authors*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

DELILLO, B. P. 2010. WebGLU development library for WebGL. In *SIGGRAPH Posters*, 4503–4503.

DI BENEDETTO, M., PONCHIO, F., GANOVELLI, F., AND SCOPIGNO, R. 2010. SpiderGL: a javascript 3D graphics library for next-generation WWW. In *Proceedings of the 15th International Conference on Web 3D Technology*, ACM, New York, NY, USA, Web3D '10, 165–174.

HALLE, M. 1997. Autostereoscopic displays and computer graphics. *SIGGRAPH Comput. Graph. 31*, 2, 58–62.

HARROLD, J., AND WOODGATE, G. J. 2007. Autostereoscopic display technology for mobile 3DTV applications. SPIE, vol. 6490.

HÜBNER, T., ZHANG, Y., AND PAJAROLA, R. 2006. Multi-view point splatting. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, ACM, New York, NY, USA, 285–294.

JAISSON, M., AND NOCENT, O., 2010. ModJaw®: a kind of magic. `http://www.modjaw.com`.

KHRONOS GROUP, 2009. WebGL - OpenGL ES 2.0 for the Web. `http://www.khronos.org/webgl/`.

LEUNG, C., AND SALGA, A. 2010. Enabling WebGL. In *Proceedings of the 19th international conference on World wide web*, ACM, New York, NY, USA, WWW '10, 1369–1370.

MOUTON, C., SONS, K., AND GRIMSTEAD, I. 2011. Collaborative visualization: current systems and future trends. In *Proceedings of the 16th International Conference on 3D Web Technology*, ACM, New York, NY, USA, Web3D '11, 101–110.

OPENNI™, 2010. OpenNI framework. `http://www.openni.org/`.

PRÉVOTEAU, J., CHALENÇON-PIOTIN, S., DEBONS, D., LUCAS, L., AND REMION, Y. 2010. Multi-view shooting geometry for multiscopic rendering with controlled distortion. *International Journal of Digital Multimedia Broadcasting (IJDMB), special issue Advances in 3DTV: Theory and Practice 2010* (Mar.), 1–11.

RAGGETT, D. 1994. Extending WWW to support platform independent virtual reality. In *Proceedings of the INET/JENCS*, 242/1–242/3.

REIBERG, J., 1997. MAJA: Modeling and Animating the Human Jaw. `http://www.reiberg.net/project/maja/overview.html`.

W3C®, 2012. The WebSocket API. `http://dev.w3.org/html5/websockets/`.