A Web-Based 3D Mapping Application using WebGL allowing Interaction with Images, Point Clouds and Models

Alexandre Devaux
Université Paris Est, IGN, Laboratoire MATIS
alexandre.devaux@ign.fr

Mathieu Brédif
Université Paris Est, IGN, Laboratoire MATIS
mathieu.bredif@ign.fr

Nicolas Paparoditis
Université Paris Est, IGN, Laboratoire MATIS
nicolas.paparoditis@ign.fr

ABSTRACT
Nowadays we see a lot of Rich Internet Applications (RIA), but real interactive 3D applications have just started years ago, and it is only in 2011 that a GPU library was integrated directly in the major internet browsers. This opened a new way of creating 3D applications on the web and a new public, as much more users can access them without complicated plugin installations and this is just what 3D web mapping was waiting for.

We present our street view application allowing to move through cities from a pedestrian point of view and interact with it in order to create and update precise urban maps enabling accessibility diagnostics, services and applications around mobility. We care about precision in the data acquisition (images and laser) in order to be able to make accurate 3D measurements for professional usages. The user can show large laser clouds in his web-browser, access information on any points, draw 3D bounding boxes and export selections to databases. He can also make measurements from images, integrate collada models, display OpenData layers, etc. Finally we use projective texture mapping in real time to texture any mesh with our images.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous;
H.5.2 [Information Interfaces and Presentation]: User Interfaces

General Terms
Algorithms, Design

Keywords
WebGL, Html5, Shader, City Modeling, Street view, Panoramic, Mapping

1. INTRODUCTION

Serving street-level imagery on a large scale is a challenge. Interact with it and manipulate real 3D data is even more challenging. [1] presented some great work on next generation map making based on multi source acquisition but this data is not available on the web. In order to offer to web users large textured city models, millions of laser points with an interactive interface on it and other visualization services we had to work on 3D rendering in a web context. As lots of multimedia web-based applications needed GPU acceleration since a long time, developers had to create sophisticated library to make 3D rendering possible using only the CPU (Papervision, Away3D, Sandy, etc). In 2009, Google launched O3D: an open source JavaScript API (plugin) for creating interactive 3D graphics applications that run in a web browser. A year and a half later WebGL was functional and O3D was integrated in it.

The ability to put hardware-accelerated 3D content in the browser provides a mean for the creation of new web based applications that were previously the exclusive domain of the desktop environment. The new standard HTML5 allows WebGL to use its canvas element so it leverages the power of OpenGL to present accelerated 3D graphics on a web page.[3] have shown that 3D in the browser using WebGL makes possible new approaches and strategies allowing motion capture studios to communicate with each other. [2] worked on both performance and scalability of the volume rendering by WebGL ray-casting in two different but challenging application domains: medical imaging and radar meteorology.

The WebGL specification was released as version 1.0 on March 3, 2011. It is an extension to the JavaScript programming language to allow the generation of interactive 3D graphics within any compatible web browser. Today in 2012 it is supported by all browsers apart Internet Explorer which still have 42% of the browser market. This plays in favour of Flash which is available in all browsers since you install the version 11 for GPU-rendering. But we believe that WebGL is going to be the standard for 3D visualization on the web, at least for scientific/mapping applications. However both flash3D and WebGL rely on OpenGL ES which makes them close and we can imagine to make a Flash version of our application for users with incompatible browser or card. Finally, WebGL is more oriented to mobile devices than Flash, some mobile phones can already render complex 3D models using integrated GPU on WebGL website.

This paper will demonstrate our experience on creating a GPU accelerating geo-viewer for the web (Figure 1). We will first introduce the data used, then the architecture of
2. DATA MANIPULATED

All the data (images, lidar, positioning information) is collected using a sophisticated vehicle we created for the acquisition of panoramic images and laser data which are precisely georeferenced (sub-metric absolute localization). Figure 2 shows the image acquisition. The data is then slightly post-processed and organized on servers using file systems and PostGIS databases. We want to stay as close as possible to the original data to shorten post-processing, minimize server space usage (no duplication) and to work on real native values.

3. ARCHITECTURE

A very special effort has been made to efficiently transfer data from the servers to the client. As we work with large panoramic images (21Mpixels) and billions of 3D points we had to find a way to stream images and lidar data fastly enough to keep an immersive dynamic sensation. Figure 3 shows the global architecture.

3.1 Images

We use IIPImage fastCGI on the server allowing dynamic multi-resolution tile streaming from JPEG2000 images (we use Kakadu library for decompressing). It allows reducing to a single image source file per camera photography on the server and works in a pyramidal way. The client can ask a specific part of the original image at a specific resolution with a specific compression. When a client navigate through the city, at every new position he loads the ten camera pictures in order to make a panoramic view. The loading starts with very low resolution highly compressed images hence taking a short time, then new tiles with better quality appears automatically.

3.2 Laser clouds

It is possible to display the laser cloud in the viewer superposed to the image or alone. While activating only the laser cloud you can evolve fluidly in the city through 3D points which are streamed dynamically around your position. For that purpose we created a file system on the server to organize the binary lidar data. We divided thousands of square meters of laser data into blocks of one meter square and used a geo hierarchy on the web server, so the user actually just loads the laser data around its position quite fastly because everything is already preprocessed. We are currently working on another architecture subdividing the global laser cloud into façade packets which is more efficient when measuring functions are enabled. It is also possible to request a specific temporal period of acquisition. The idea behind the temporal representation of lidar data is that we can ask for 3D points that were acquired at the same time as the panoramic picture. In some streets the vehicle passed several times so the physical correlation between the panoramic picture and the lidar data can be low if we had a lot of moving objects or if we have a derivation from the geo-referencing system.

Each laser point can be picked with a mouse click in order to have access to its property (3D absolute position, GPS time, intensity) thus allowing to make absolute and relative measurements as we have created an interface to draw lines.
between points showing distance, polygon showing surface and even 3D volumes. All this is possible with or without showing the laser clouds. Within the panoramic imagery alone every click in the image is then projected on the laser cloud to get direct absolute 3D position.

3.3 3D points extraction

The application interface is simple and effective for visualization, creation and edition of polyline based features. The user can easily create 3D bounding box around an object using just 3 clicks he can shape a box around the object he wants to extract (Figure 4 shows a user selecting a box around a car). Then the application automatically colors and addresses a specific class to the 3D points inside the box. It can recognize the object using gabarit features and the user can adjust the class of the object or add specific information. The application gives also the absolute position of the gravity center of the object, the length, width, height, volume, etc. Then the user can drag the object where he wants and decide to save it to the object extraction database. Useful to create urban maps, this can also be useful for scientists allowing them to create ground truth database of specific class of object.

3.4 Projective texturing on urban generated mesh

Working in WebGL allows us to use shaders. They are very useful to calculate rendering effects on graphics hardware with a high degree of flexibility. We created a projective shader so we can project any image to any mesh 3D (city models for us). The process is the following: We have a map of building print (2D). We create a mesh of the building with the a-priori that the facades are perpendicular to the ground. We use the laser cloud to obtain the height of buildings or we give a specific height that is supposed to be the maximum height of buildings we can meet and create a mesh. It is also possible to load more precise meshes if they are available. Then with multiple shaders we texture the mesh using photography taken from different points of view which in our case allows us to fill holes because our acquisition geometry didn’t capture a full 360° sphere. The navigation can now be fluid between the different points of view as we now evolve in a real textured 3D model (Fig 5).

4. DEMONSTRATION

The setup for the demonstration is as follow: First we will show how a non-professional public can use it. Second, we will present the possibilities of precise measurement, integration of future public infrastructures for professional users.

5. FUTURE WORKS

For more than a decade, rich 3D content on the web has only been available via external, often proprietary, browser plugins. However, a new standard has emerged to change this: WebGL. Within that decade, local bandwidth and server capacity have exploded allowing new possibilities for the internet.

We have shown here how powerful was already WebGL with a case-study, a Street View navigation web-application and how well suited WebGL is for modeling/mapping applications. We will now work on optimizing data integration and accessibility to allow an even more collaborative application. With the other popular GPU technology (Flash 11) we will see a lot of powerful 3D applications on the web in the next years and we are pretty excited about contributing to sculpting the web.

6. REFERENCES