HTML5 3D Visualisations

Version of 20 May 2011

Abstract

This survey is meant to give the reader an in depth overview of currently used and actively developed 3D graphics libraries and plug-ins used for information visualization within the browser.
The focus of this survey is on WebGL, which is a graphics library for web browsers using JavaScript and OpenGL. WebGL is supported in HTML5 standard and can be used with the canvas element. Other technologies which enable 3D graphics in the browser are also examined and compared to HTML5 and WebGL.

Various example applications of the different technologies are shown throughout the survey. Some of them include complex 3D models that display the capabilities of the technologies while others show Information Visualisation (InfoVis) applications of 3D graphics in the web browser.

The survey will attempt to provide a conclusion on what technologies should be used by developers for what purposes in order to create 3D InfoVis applications, their advantages and disadvantages as well as the share of devices on the market that support them.

Contents

Chapter 1

Introduction

1.1 Introduction

This survey is meant to give the reader an in depth overview of currently used and actively developed 3D libraries and plug-ins used for information visualization within the browser.
The focus of this survey is on WebGL, which is a graphics library for web browsers using JavaScript and OpenGL. WebGL is supported in HTML5 standard and can be used with the canvas element. Other technologies which enable 3D graphics in the browser are also examined an compared to HTML5 and WebGL.

The underlying technique, which is actually needed to be able to make 3D content available in webbrowsers that support HTML5, is called Canvas. This new technology is part of the HTML5 standard, which slowly, but progressively is supported by more and more browsers.

There are quite a few libraries available that provide direct access to some underlying 3D application programming interfaces (APIs). In our survey we will start with a short description of the HTML5 Canvas element. Then we move on and have a detailed look on the WebGL API. In Addition to that we will examine VRML and X3D, which are standard formats used for describing 3D models and scenes, as well.

Some examples of applications using the technologies discussed in this survey will be included and briefly described.

Finally we will provide a conclusion which will consist of personal thoughts and experiences with the mentioned technologies. Advantages and disadvantages will be highlighted to provide developers with information on what technologies to use for what purposes.

Chapter 2

HTML 5

2.1 HTML 5

With the first introduction of HTML 5 in 2008 numerous additions were added to the HTML standard and a few deprecated tags have been removed. The most interesting novelty however, was the inclusion of the canvas element, not only with respect to the topic of this survey but in general.

2.1.1 HTML Canvas element

This canvas element provides programmers with the ability for dynamic, scriptable rendering of 2D shapes and bitmap images. Canvas is a procedural model which updates a bitmap and is not using a built-in scene graph.

Originally canvas was used inside the Mac OS X WebKit as a component, which improved applications like the Dashboard widgets. Later on it was adopted by a few companies, one of them was Opera, and standardized by the WHATWG (Web Hypertext Application Technology Working Group) for next generation web technologies [HTML5 2010].

The canvas element is basically a field within a browser with defined width and height, which can be used to ”draw” on. This field however can be accessed by JavaScript Code with a full set of drawing functions, allowing for dynamically generating images and graphics [HTML5 2010].

2.2 SVG

Another similar technique is called SVG (Scalable Vector Graphics). It was used for drawing shapes in a browser [SVG 2002]. This however was done by remembering each drawn shape as an object in a scene graph or Document Object Model (DOM), which is then rendered to a bitmap.

This however had a big disadvantage. For example if you were to draw a shape and just wanted to move it, the whole scene had to be rendered again, including all elements that are covered by that shape, as the system just had forgotten that it just had drawn that shape.

2.3 Compatibility

One of the biggest downsides of HTML5 is that only the newest versions of the most popular browsers support it. And even if a browser manufacturer claims that their browser is fully HTML5 compliant, it does not mean that the browser is able to process every official HTML5 command.

Hand in hand with the mentioned problem goes the WebGL compatibility. As it highly depends on the HTML5 Canvas element, WebGL needs a browser which at least rudimentarily implements the Canvas element.

2.4 WebGL

One of the most used and well developed libraries is called WebGL. WebGL stands for Web-based Graphics Library.

It represents a standard for programming in 3D with the Web-browser as platform, by extending the JavaScript programming language. The specs for the first final version of WebGL were published in March 2011 and were defined by the Khronos Group, which is in charge of the OpenGL and OpenCL standards as well.

Basically WebGL can be used as an interface between JavaScript and OpenGL ES 2.0. OpenGL ES stands for OpenGL for embedded systems and is a version of OpenGL designed for embedded devices. This allows native OpenGL code to be executed by directly accessing the hardware of graphics cards. Every current graphics card supports OpenGL, but an up-to-date driver might be required in order to display WebGL content.

Usually the operating system takes care that the correct OpenGL library is installed and used by WebGL. This direct access to the hardware enables hardware accelerated 3D graphics, which can give much better performance than software rendered graphics, which are computed on the CPU.

As previously mentioned, the whole rendering process is done by Canvas. So far nearly all browsers have adopted to the new HTML5 standard, which means that nearly all browsers, or at least their latest releases, can display 3D elements, programmed with WebGL.

As many people still have old versions of Internet Explorer or Firefox running it is very important that WebGL is supported by many companies so that this new technology can be pushed into the market, sooner or later.
Currently there are quite a few companies behind WebGL like Google, Nvidia, AMD and Mozilla, just to name the most important ones.

WebGL was first demonstrated by Vukicevic in 2006. In 2007 Mozilla and Opera implemented a similar system on their own. In 2009 Khronos started the WebGL Working Group together with Mozilla.

As we mentioned before, one of the biggest limitations to the usage of WebGL is the availability of an HTML5 browser. However there is another big problem which WebGL has to overcome before it can become more popular. As the API directly accesses the graphics card, there have to be operating drivers available which are compatible to WebGL.

2.5 WebGL API

Using the Canvas element, the only interface which is part of the Canvas specification is the 2D canvas rendering context (CanvasRenderingContext2D). WebGL provides another interface called WebGLRenderingContext, which actually represents the WebGL API.

2.5.1 WebGL Context

As 3D graphics can be used in many different ways, including information visualization purposes, the WebGL API has been built with the objective to provide flexible primitives that can be applied to any use case.

For special purposes specially tailored libraries can provide an API on top of WebGL, which can accelerate and simplify development. This however is not the main goal of WebGL. Due to the fact that WebGL inherits OpenGL ES 2.0, the whole WebGL library should be straightforward for developers who have already experience with OpenGL or OpenGL ES 2.0 development.

After the user has optained a WebGLRenderingContext object for a given HTMLCanvasElement, the 3D drawing can start.
The WebGLRenderingContext object is used to manage all states of OpenGL and render to the drawing buffer. This drawing buffer has to be created at the same time of the context creation.

The API provides a getter method which always returns the given context of an HTML Canvas element called ”getContext()”. There is also a number of attributes which can, but don’t have to be submitted like alpha, depth, stencil, antialias etc. [Khronos 2009].

2.5.2 WebGL Drawing Buffer

Then there is the drawing buffer, where API calls are rendered into. This buffer should be defined at the same time as the context. The size of the drawing buffer is defined by the width and height attributes of the given HTML Canvas element.

So once the width or height attributes of the HTML Canvas element are changed, the drawing buffer will be resized to match the new attributes. There are two getter methods for the buffer which simply acquire the drawing buffer’s width (drawingBufferWidth) and height (drawingBufferHeight) [Khronos 2009].

2.5.3 WebGL Viewport

The Viewport is represented as a rectangular shape, which defines the where the results of the rendering in the drawing buffer should be placed. On creation time the viewport is set with origin at (0,0) and width and height are set equal to the HTML Canvas element width and height [Khronos 2009].

The state of the OpenGL viewport should not be directly affected by the implementation of the WebGL in response to resizing of the canvas element.

2.6 The Differences between WebGL and OpenGL ES

`https://www.khronos.org/registry/webgl/specs/1.0/#6`

Although WebGL is based on the OpenGL ES 2.0 API, there have been a few changes in order to achieve the highest possible degree of portability across browsers, operating systems and mobile devices.

Some of the differences are less impactful than others. In some cases the WebGL API received more functionality, such as in the case of Pixel Storage Parameters.

- There are three additional parameters for $pixelStorei$
- $boolUNPACK\_PREMULTIPLY\_ALPHA\_WEBGL$, multiplies the alpha channel of the source data into the color channels during calls to $texImage2D$ or $texSubImage2D$
- $ulongUNPACK\_COLORSPACE\_CONVERSION\_WEBGL$, applies conversions specific to browsers and file types

In other cases, WebGL simply doesn't support certain OpenGL ES features such as

- Fixed Point Support - no GL_FIXED data type
- Client Side Arrays
- Viewport Depth Range - no depth ranges with near plane ¿ far plane
- Blending With Constant Color - constant color and constant alpha can not be used as source and destination in blend functions
- Compressed Textures - no $CompressedTexImage2D$ and $CompressedTexSubImage2D$

Also there are some differences concerning the usage of buffers which is not at all surprising, given the nature of web-based applications. The Buffer Object Binding, Buffer Offset and Stride Requirements and Framebuffer specifics differ in various ways.

While these changes will sometimes make it harder to port existing OpenGL ES projects into WebGL, they are necessary. Developers which are already familiar with OpenGL or OpenGL ES will find working with WebGL fairly straightforward and will be able to use their experience regardless of some small differences in certain aspects of the API.

2.7 Learning WebGL

In addition to the very detailed specification of the WebGL API on `https://www.khronos.org/registry/webgl/specs/1.0/#6`, there are a variety of platforms where a newcomer can get to learn how to create WebGL applications such as:

- `http://learningwebgl.com/`
- `http://planet-webgl.org/`
- `http://www.peter-strohm.de/webgl/` (German)

The ”Learning WebGL” - Project is a blog by Giles Thomas, where he provides Lessons based on the NeHe OpenGL tutorials. Also on the blog there is The WebGL Cookbook - a wiki with the most basic samples and links to sites with further information as well as an FAQ section.

Mozilla also has a basic development intoduction at `https://developer.mozilla.org/en/WebGL`, providing simple tutorials and additional resources for people interested in WebGL.

As WebGL and HTML 5 in general become more known and more frequently used there will, without a doubt be a lot more useful resources to learn it. Additionally, as WebGL is based on OpenGL ES, tutorials and resources concerning OpenGL and OpenGL ES are definitely also a useful place to look for advice, knowledge and examples.

2.8 Demos

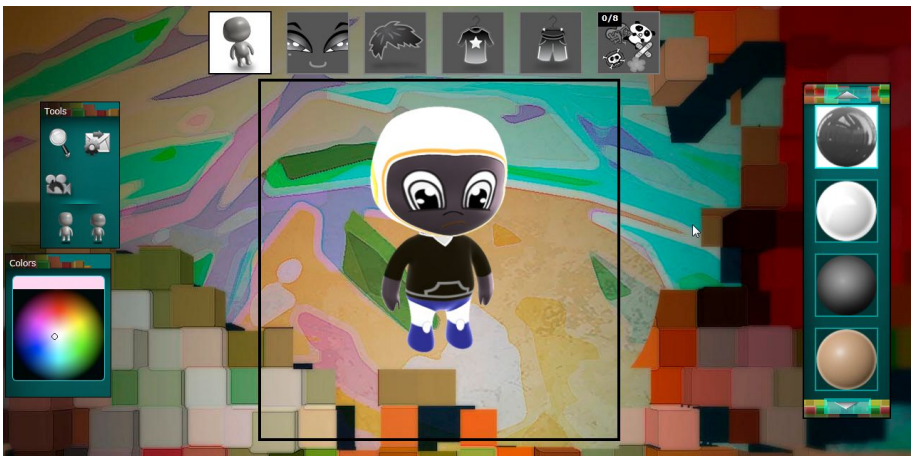
- GlobeTweeter (https://mozillademos.org/demos/globetweeter/demo.html)

The "GlobeTweeter" is a WebGL Demo-Project by Cédric Pinson and Guillaume Lecollinet. The Project uses GeoLocation, Twitter Web API and WebGL to render a 3D model of the earth where Twitter-activity across the world is shown in real-time. You can zoom and move the globe and it is an impressive example of how to visualize social network content with WebGL. GlobeTweeter is an open-source project which can be found on github: (https://github.com/cedricpinson/globetweeter/)

- Caves (http://webglsamples.googlecode.com/hg/caves/caves.html)

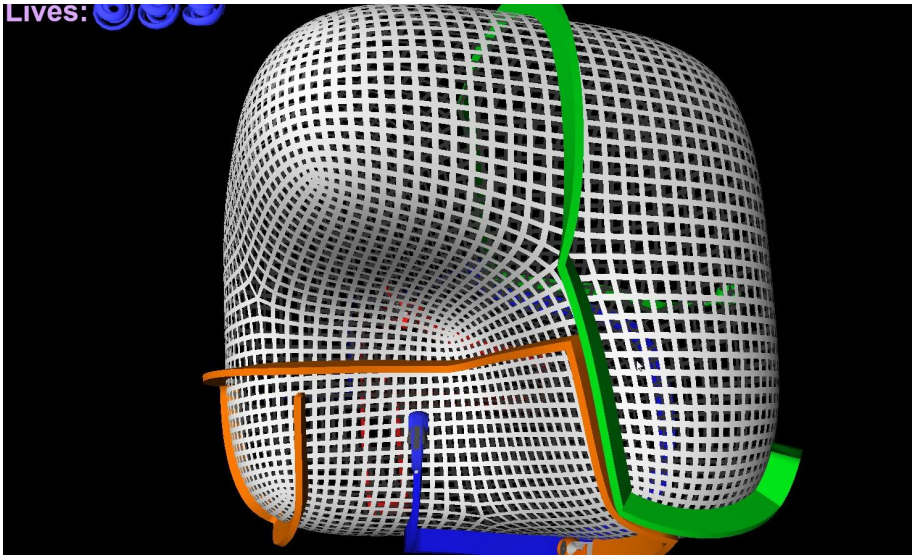
”Caves” is one of the sample applications from the ”webglsamples” Project at googlecode, where users can contribute WebGL applications under the New BSD License. The Application itself is basically a 3D game where the user can move around and carve caves into the ground. It has interactive controls and runs very smoothly.Caves is a nice example of how a pimped up MineCraft could look like with WebGL.

- Collectibles (http://webglsamples.googlecode.com/hg/collectibles/index.html)



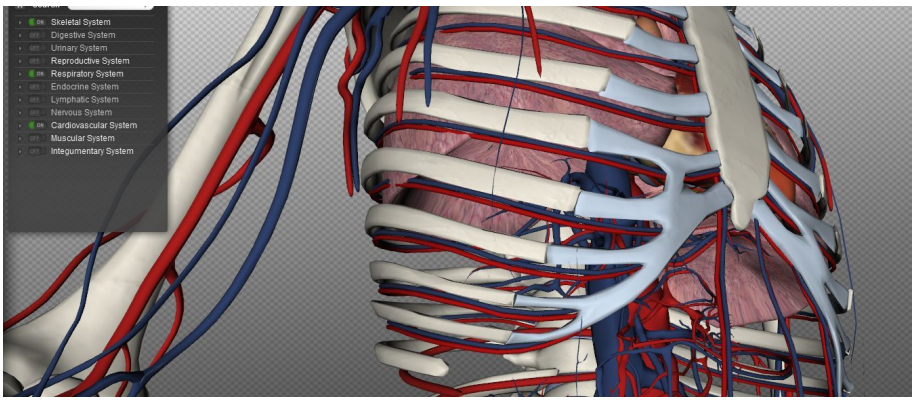
The Collectibles Painter is a WebGL application by Human Engines which allows the user to create a personalized toy collectible using paint, decals and materials. Users can share their favorite toys with their friends. This application even uses advanced graphic effects such as soft shadows and texture based projection. Applications like the Collectibles Painter may be used in the future in webshops to provide users with the ability to ”virtually” try on clothes before buying them.

- Cycleblob (http://cycleblob.com/)



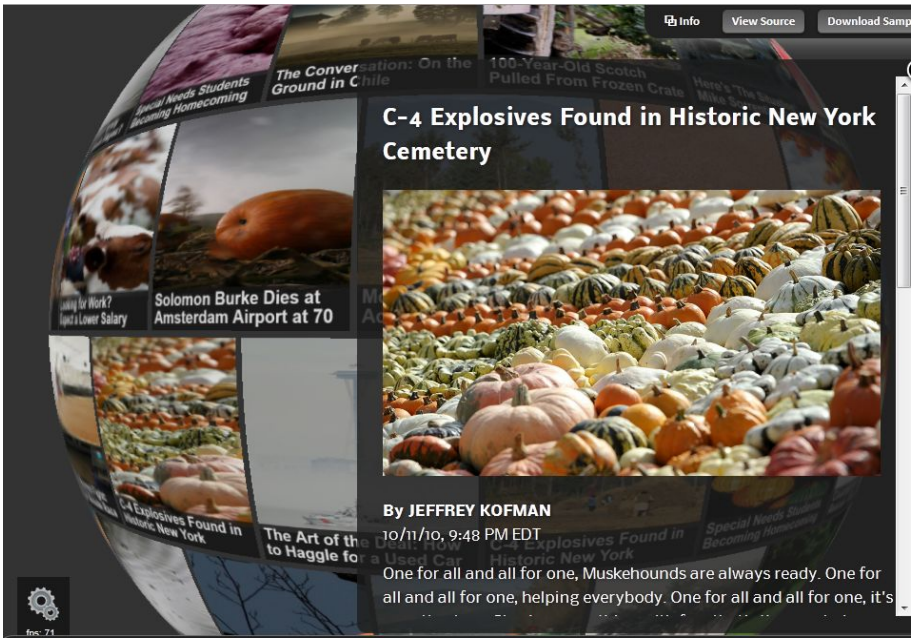
This is a 3D version of the Tron-Game ”Light Cycles”, where multiple players control a Light Cycle, trying not to crash into a wall made by themselves or another player. The application uses WebGL and is a lot of fun to play. This could be an outlook on how web-based arcade games could look in the future with WebGL.

- BioDigitalHuman (http://www.biodigitalhuman.com/)



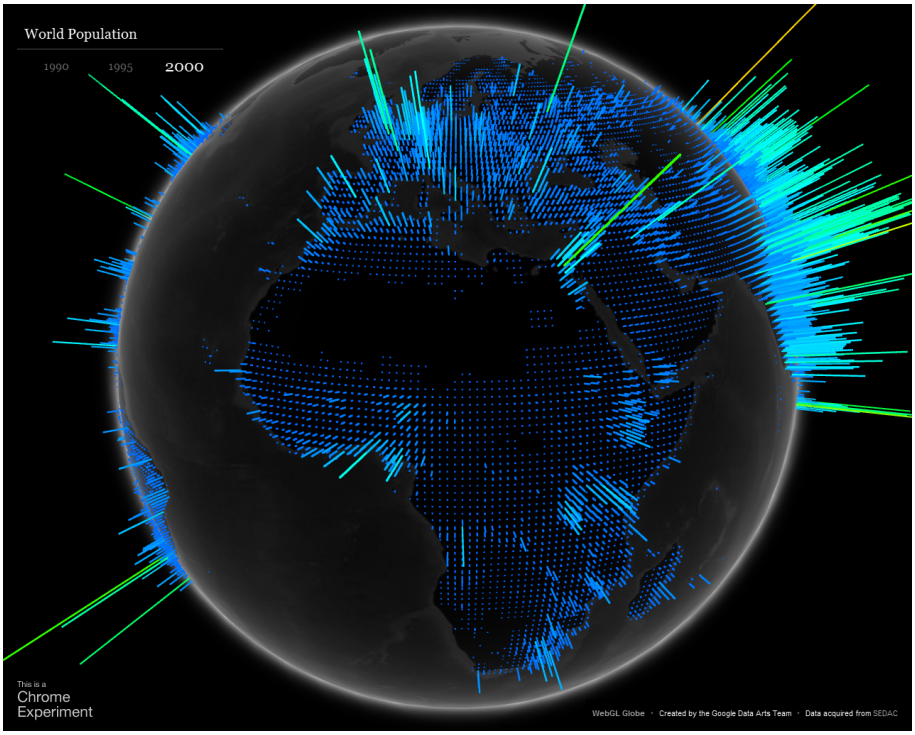
The BioDigitalHuman project is a very impressive application made by the BioDigital Team (http://www.BioDigital.com). It provides a detailed 3D model of the human body which the user can virtually explore. There are over 3000 specific 3D objects which can be hidden or displayed seperately. If the user double clicks on any body part, additional information on it is displayed. There is functionality implemented to search inside the model and to export images.

- WebGL Globe (by HTML5Studio) `(http://studio.html5rocks.com/#Globe)`



The WebGL Globe is an example application from html5rocks.com which shows an interesting approach on how to visualize news and pictures in general with technologies such as WebGL.

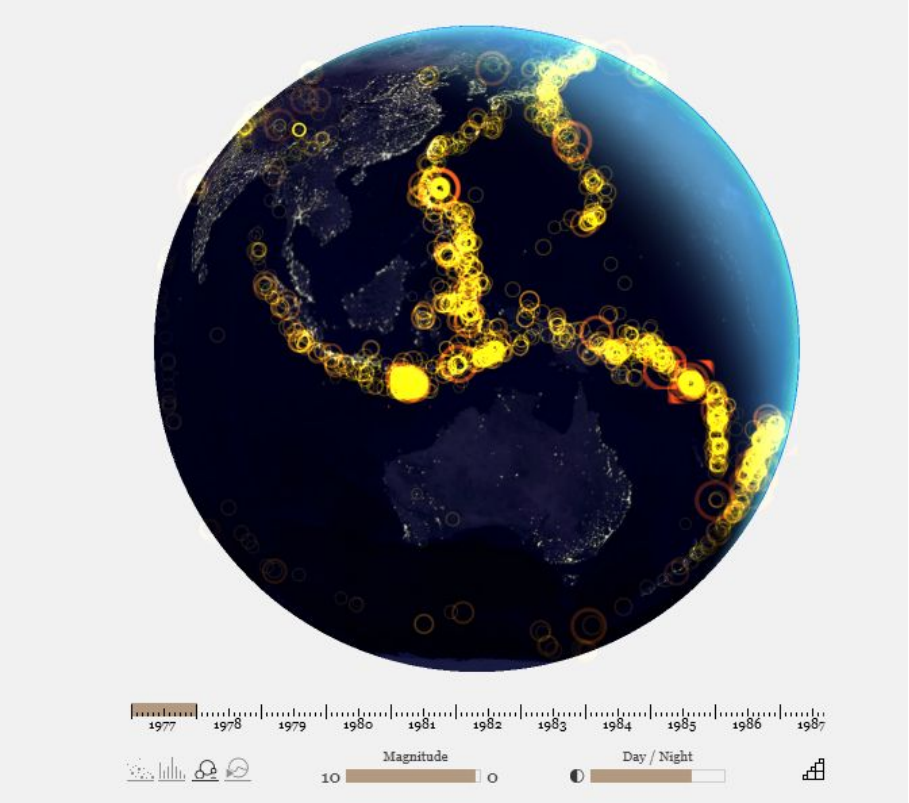
- WebGL Globe (by Google) `http://data-arts.appspot.com/globe`



Another example which is incidentally also called WebGL Globe comes from google. It is displayed on their Google Chrome Experiments gallery site (`www.chromeexperiments.com`). Google's WebGL Globe is an actual geographic visualisation. It visualises the world population by drawing boxes on the globe where the height of the box and its color correspond to the population in that city or area. The user can spin the globe, zoom in and out and also switch between different years.

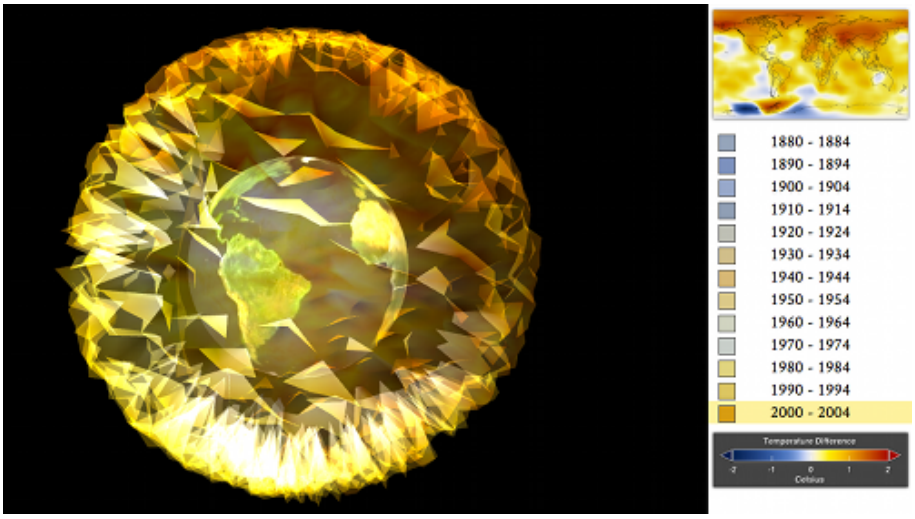
The implementation of the WebGL Globe includes various performance tweaks in order to make it run smoothly.
It uses the Three.js JavaScript 3D engine. A cube is generated for each data point, the bottom face is removed and cube is stretched in order to match the population data, which is present in the JSON format. The boxes' geometry is divided between states and rendered in a vertex shader program. Furthermore, two fragment shader programs are used: One simulates the earth's atmosphere and the the other is responsible for lighting. [GoogleChrome2011]

- NinePointFive (http://www.ninepointfive.org/)



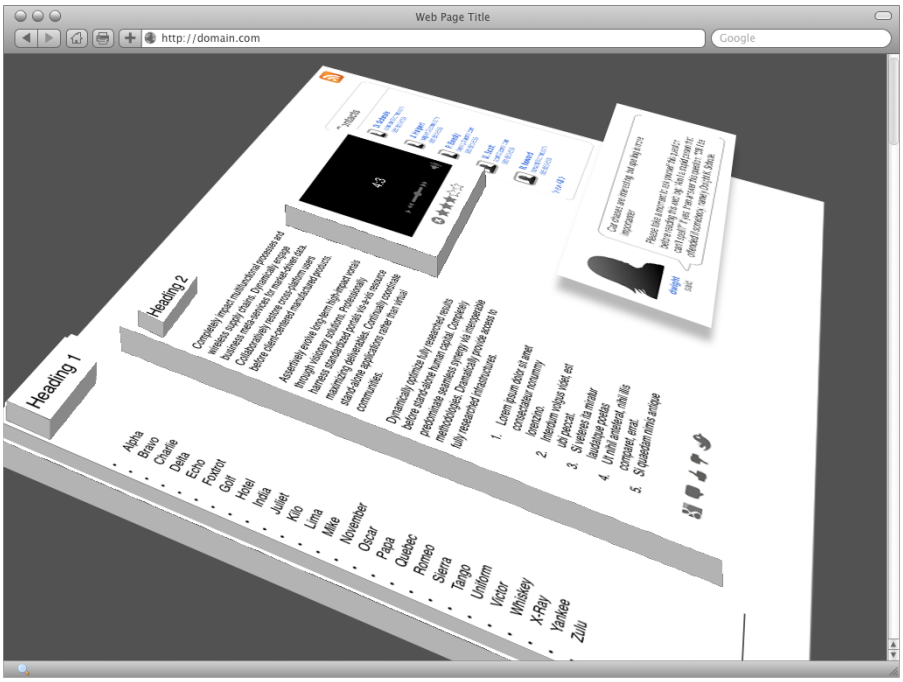
The NinePointFive Project is an impressive WebGL-based visualization of earthquakes around the world for the past 30 years. The user can switch between different views, select certain earthquakes, scroll through time and even filter by the magnitude of the quakes. A very nice example for a data visualization with a time axis.

- PhiloGL (http://senchalabs.github.com/philogl/)



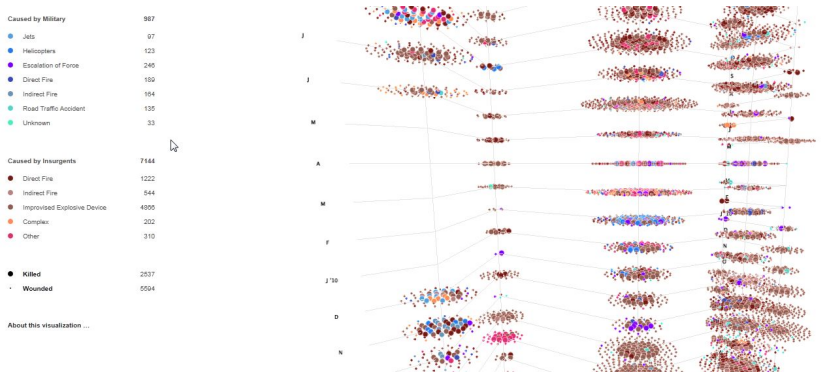
PhiloGL is a WebGL Framework from the people behind the InfoVis toolkit. It provides a wide functionality for Data Visualization, Creative Coding and Game Development. The provided example picture shows a visualization of Temperature Anomalies from 1880 to 2010. It looks extremely nice and is very fluently navigatable.

- Tilt (https://wiki.mozilla.org/Tilt_Project_Page)



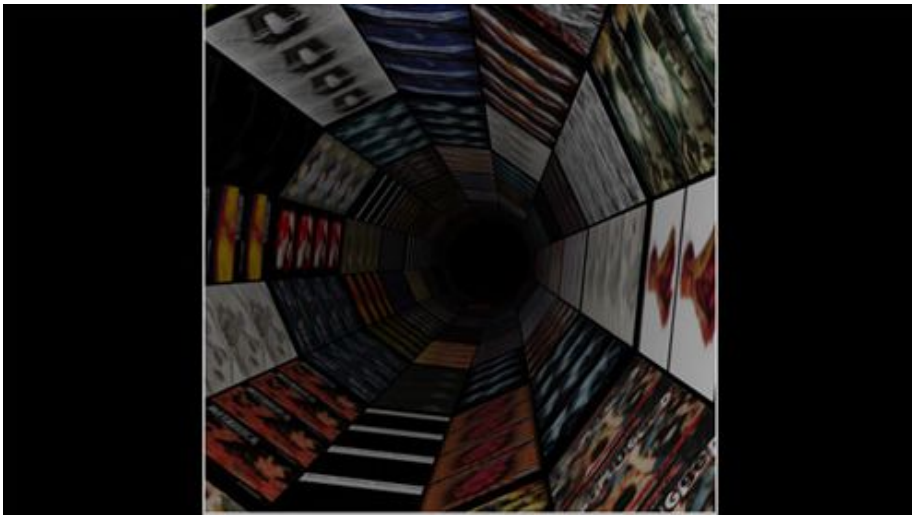
The Tilt Project is a WebGL-based visualization tool for websites. It is basically a webpage-inspector, so it represents the DOM of the tested site, where the BODY is the baselayer and every other layer is added 3-Dimensionally on top of this layer. This lets the user see relationships and parents of elements in a very clear way. The user can zoom and move around the model and thus explore the DOM to its full extent.

- Webgl Visualization of Afghan Casualties (http://plumegraph.org/)



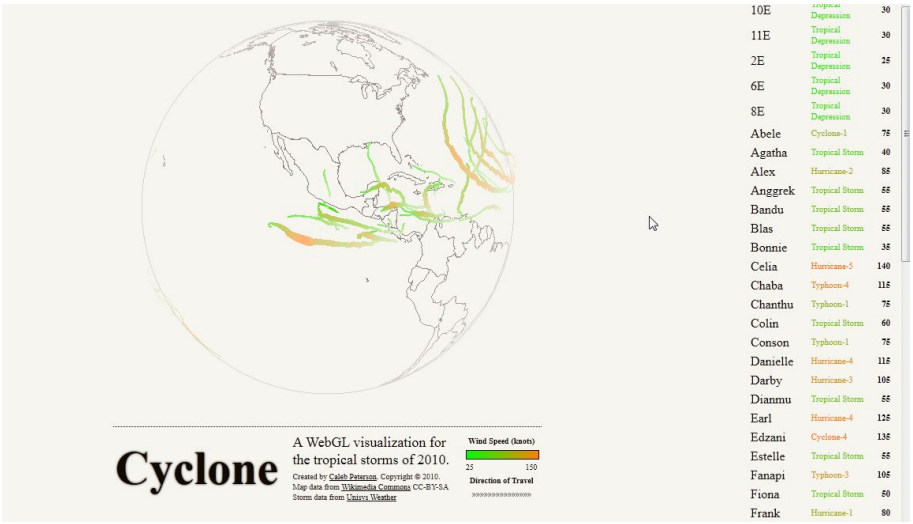
This is another great example for data visualization with WebGL. It is a Project by George Michael Brower and John Bohannon visualizing a data set which shows the Afghan casualties during the war in Afghanistan. The user can filter the data, move around and zoom through the data in a very smooth and intuitive way.

- Last.fm Data Visualization (http://www.rozengain.com/blog/2010/07/23/visualizing-lastfm-data-with-webgl-glge-jquery/)



This is a WebGL based visualization of Last.fm albums, using the GLGE library for WebGL and jquery for the data. With the help of the GLGE library, a visual tunnel is created, through which the user can fly and navigate through his albums by looking at their respective covers. It works smoothly, and looks great, but lacks actual usability.

- Cyclone (http://www.cubicle6.com/2011/01/31/cyclone-a-webgl-visualization/)



Cyclone is a data visualization project by Caleb Peterson using WebGL. It is a 3D view of the Earth and of the tropical storms in the year 2010. The colored lines on the globe show the path of the storm and their coloring indicates their respective speed at each position. It is easily filterable by each storm that occured in 2010. A simple, yet effective way of representing a complex data set in WebGL.

2.9 More demos

Other great resources for demos are:
http://www.chromeexperiments.com/webgl
http://code.google.com/p/webglsamples/

Chapter 3

3D Standards for use on the web

3.1 VRML

VRML (which is often pronounced ”vermal”) stands for ”Virtual Reality Modeling Language”. It is a file format used for the representation of 3D vector graphics. VRML was designed to be a readable file format for creating 3D graphics to be displayed in web browsers.

VRML is an ISO/IEC standard (ISO/IEC 14772-1:1997), but it has been superceded by the X3D standard (see 3.1). The VRML file format can still be read by all X3D applications, so it can still be used. The first draft of the VRML specification was published in 1994, while the latest version of the standard, VRML97, dates to 1997.
VRML files are called ”worlds” and have the extension .wrl. These files are human readable text files, which are commonly compressed using gzip. [WEB3D97]

VRML itself is used create and define 3D graphics. In order to display these graphics, a separate VRML viewer application or browser plugin is required. Many different viewers and plugins are available. One popular viewer, which is an open source project is FreeWRL (`http://freewrl.sourceforge.net/`). A list of other viewers can be found on the the web3d consortium's site: `http://www.web3d.org/x3d/vrml/tools/viewers_and_browsers/`.

3.2 X3D

X3D stands for Extensible 3D and is the successor to VRML. Much like VRML, it is a standard format to describe 3D models which can then be viewed in a web browser. Its intended areas of application include engineering and scientific visualization, multimedia presentations, entertainment, education, web pages with 3D content and virtual worlds.
The file format is still human readable, but is based on XML in contrast to VRMLs text file format. The file extension is .x3d. It is also the new ISO/IEC standard (ISO/IEC 19775/19776/19777).

X3D is intended to become a universal interchange format 3D graphics models and multimedia. It has some improvements over X3D and adds new features, such as animations, programmable shaders, reflections or advanced lighting techniques. [WEB3D08]

Like VRML, X3D is only used for the description of 3D scenes and models and still needs a viewer application or a browser plugin in order to be rendered. A list of available viewers and plugins can be found on the the web3d consortium's site: `http://www.web3d.org/x3d/content/examples/X3dResources.html`.

Chapter 4

Alternative technologies

4.1 XML3D

4.1.1 Overview

The XML3D project was invented by the Saarland university and is based on various technologies, all of them part of W3C standards and recommendations. XML3D extends the frameworks of WebKit and Mozilla, which means the XML3D scenes can be displayed straight in the website. Due to its implementation as extension of the browser frameworks, XML3D has the ability to use 3D-specific CSS properties.

The principle of XML3D is to reuse the existing web technologies to leverage the existing work and upcoming improvements [XML3D 2010].

4.1.2 Design

A XML3D scene starts with the <xml3d>tag. This tag defines the area on the website, where the scene is displayed. It is part of the standard HTML hierarchy, thus CSS styles like z-positioning and background color, can be applied. By default an XML3D scene has a transparent background.

The next tag should be the <defs>tag. The content of this tag is not displayed in the scene directly, it contains the definition of resources used and instantiated later through references. In general, these resources may include geometry data, transformations and all kinds of shaders. The <defs>tag is similar to SVG (see **??**).

The geometry resources are referenced in the <mesh>tag using the src attribute. In order to render the scene efficiently the <mesh>tag uses, like the OpenGL the VertexArray, an array to serialize the data mapped to the rendering engines. The elements of these arrays contain several data types, such as different float types and integer values.

There are no complex primitives like for example cylinders, implemented in the XML3D framework, which results in a lightweight standard. Through grouping, using the <group>tag, and the CSS shader extension it is possible to apply material properties and transformations. The <group>element can also include elements like <light>and <view>for light sources and view points.

Due to the hierarchical structure of the XML3D framework, the mesh element has to be instantiated using a <use>tag.

XML3D uses CSS 3D transitions, specified by the W3C [W3C 2009a]. These transitions can be applied to the <group>tag and for this reason also for all of its children.

There are two different ways of defining shaders to change the appearance of the surface. The shader properties can either be individually defined using the <shader>tag or they can be defined using a set of default shader models with CSS properties [XML3D 2010].

A simple way to animate a scene is to use the standardized W3C CSS animations [W3C 2009c]. This feature is a good example for the principle of the XML3D project to reuse as many web technologies as possible.

Due to the XML3D data structure, it is fully accessible in the Document Object Model (DOM) and the 3D data can be accessed in a standardized way[XML3D]. The event handeling in XML3D is provided through the DOM Events specification[W3C 2009b]. However, the DOM Events specification does not supply the 3D scene with enough possibilities to trigger events. For this reason, there were some additional 3D-specific events implemented for XML3D, for example picking and proximity detection.

4.1.3 Implementation

XML3D is implemented in two different ways. For WebGL and native build-in for Firefox and WebKit. The WebGL implementation is not able to provide all CSS features. The XML3D scene (for each ¡xml3d¿ tag) is shown in a HTML canvas element [**?**].

The second is build in to the Firefox and WebKit frameworks. This implementation uses RTSG [Rubinstein et al. 2009], a scene graph management library in an extended version to store data in an efficient way. The RTSG structure holds most of the data and is accessed by a DOM API.

In order to prevent problems with different renderers (and therefore different shading languages) in different browsers, XML3D is based on an independent multi-language shading description called AnySL [AnySL2010]. AnySL supports many different shading languages and provides a very high performance for each language.

4.1.4 Limitations

There are some XML3D limtations which will be, referring to the authors of ”XML3D - Interactive 3D Graphics for the Web”, solved in future development [XML3D 2010].

Future implementations of web browsers have to deal with the upcoming flood of data generated by XML3D through an embedded DOM element. Current DOM parsers integrated in web browsers are not optimized for large amounts of data.

All vertex data in DOM are stored as full text representations, which leads to a computational overhead concerning the serialization and de-serialization when synchronizing with the RTSG structure [XML3D 2010].

There are also some general limitations to WebGL implementations using the OpenGL ES 2.0 specification. The array index values are limited to short and unsigned byte types, which means 65536 is the maximum number of vertices for an array. Complex 3D models can easily exceed this vertex count.
As an alternative glDrawArrays can be used, which does not have this limitation but does not support shared vertices which leads to an overhead of copying vertex data [XML3D 2010].

4.2 Flash 3D

4.2.1 Flash

Adobe Flash is a proprietary integrated development environment which is used to create interactive multimedia content, such as animations, video, and interactive content which can be included in web pages.
Flash is commonly used to create animations, advertising and games in web pages.

In order to display Flash content, the user needs a browser and the Adobe Flash Player plugin. The Flash Player plugin is available free of charge for common web browsers and for some mobile phone platforms, such as Android and iOS. [Flash FAQ]

4.2.2 Native 3D support

The main focus of Flash is on 2D graphics. Before Flash Version 10 display objects have two properties, x and y, for positioning them on a 2D plane. Starting with Flash Player 10 an additional z property was introduced for every object, which indicates its position along the z-axis.

Support for 3D effects was also introduced with Flash 10, but only in a limited fashion. Display objects are still inherently flat. The 3D features enable the placement, movement, rotation, and other transformations of these planar objects in all three dimensions.

4.2.3 3D Engines for Flash

A variety of 3D engines developed by third parties exists for Flash. This section show a few examples.

Papervision3D

Papervision3D is an open source 3D engine for the flash platform. It is still in a public beta version (as of May 2011). There are two versions of the engine: The first one builds on Adobe Actionscript 2 and requires Flash Player 8 or higher, the other one builds on Actionscript 3 and requires at least Flash 9. The use of the latter is recommended by the developers unless compatibility with Flash Player 8 is desired.
Papervision3D supports polygon-based 3D models and is capable of various shading models. While there are various tutorials to Papervision the documentation is not complete yet. Users must therefore check out the source code from the project repository and and look at its inline documentation in order to use all features.[Papervision3D 2010]

Sandy 3D Engine

The Sandy 3D Engine is another open source 3D API for Flash. It uses Adobe Actionscript 2 and 3. It is actively maintained and has an extensive online documentation.
Features of the Sandy 3D Engine include:

- Lighting models and shading effects such as Phong, Gouraud or toon shading
- Transparency
- View frustum culling and clipping for performance improvements
- Materials to change objects' appearances (including video textures)
- Compatible with Adobe Flash Player versions 7 to 10

- Parsing of various 3D model formats
- Many 3D primitives for easy model creation

[Sandy 3D 2011]

4.2.4 Limitations

One drawback of Flash is that it needs the Flash Player installed on the user's device in order to work. While Adobe Flash Player 10.0 has a spread of 93.74% [Statowl 2011], it is not pre-installed on many operating systems, which is a barrier especially to inexperienced computer users.
Also, for creating advanced 3D graphics additional plugins need to be used in addition to the proprietary Flash IDE, which means a lot of effort to developers with no previous experience.

4.3 JavaFX

JavaFX is a software technology which is used for creating rich internet applications. It is part of the Java specification and enables building applications for desktop operating systems, web browsers and mobile phones. Further support for TV set-top boxes, gaming consoles, Blu-ray players and other platforms is planned.
While the technology itself is proprietary, both the IDE and the runtime environment are available for free.

To build JavaFX apps developers use a statically typed, declarative language called JavaFX Script; Java code can be integrated into JavaFX programs. JavaFX is compiled to Java bytecode, so JavaFX applications run on any desktop and browser that runs the Java Runtime Environment (JRE) and on top of mobile phones running Java ME.

JavaFX needs the Java Runtime Environment in order to be displayed in the browser. However, many platforms don't use the official Java Runtime Environment (by default), and while some mobile phone developers have integrated JavaFX into their Runtime Environments, it is currently not supported on Android and iOS devices. [JavaFX partners]

4.3.1 3D Grahpics in JavaFX

Support for 3D graphics was introduced in version 1.3 of JavaFX, which was released on April 22, 2010. Although many features are available through the already existing Java 3D API, their uses in JavaFX are not well documented or widely used yet.

4.4 JavaScript 3D

4.4.1 Overview

JavaScript is an interpreted programming language with object-oriented capabilities. It is most commonly used in web browsers, where it is run by the client computer rather than the web server. The core JavaScript language and its built-in datatypes are the subject of international standards, and compatibility across implementations is very good. [JavaScript 2006]

JavaScript 3D libraries are libraries written in JavaScript, which are used to create and interact with 3D scenes in web browsers.

Canvas 3D JavaScript Library

The Canvas 3D JavaScript Library (short C3DL) is such a JavaScript 3D library, The library is built on top of canvas 3D, which means it manipulates and it can be manipulated by other parts of the web page. C3DL tries to hide the complexity of OpenGL programming from web developers [C3DL 2008].

The vertices, textures and all other model data are stored in Collada files [Collada]. The following steps are necessary to create a 3D scene with C3DL [C3DL 2008]:

- Decide on the context
- Create a scene object
- Associate the scene with the id of the canvas the scene will be displayed in
- Add a camera
- Add objects and lights

- Start the scene

The implementation of C3DL follows closely the OpenGL ES 1.1 specification. As canvas 3D tries to make use of the video card whenever possible, C3DL has good performance. [C3DL 2008].

Chapter 5

conclusion

Some of the examples in this survey clearly show that WebGL is well suited for displaying 3D InfoVis applications in browsers. Also, WebGL should be wel suited for displaying high-dimensional data, because 3D not only provides another dimension, but also a whole lot of navigation-methods that are hard to implement and use in a 2D visualization. We also found out that there are, at the moment, some major drawbacks to WebGL, like the browser support. WebGL is only supported by the most modern browsers like FF4 and Google Chrome and it needs a fairly powerful graphics adapter with installed opengl drivers to work properly.

Another problem, for casual programmers or beginners, will be the steep learning courve of WebGL or 3D programming in general as it is way more taxing to generate working 3D code than simply putting together some jQuery snippets. But as it seems, there are certain frameworks such as PhiloGL in the development stage, which will allow people to use a highlevel toolbox to create their 3D visualizations. So far there are only few examples of actual InfoVis applications using WebGL. Most examples rather showcase advanced graphical features than present information in a well-readable fashion. It should, however, only be a matter of time until good InfoVis libraries for WebGL become available.

We believe that, with the increase of browser support and especially the major support by Google and Mozilla, WebGL, same as HTML 5, will be established as a new web standard for 3D visualization.

Bibliography

[XML3D 2010] XML3D - Interactive 3D Graphics for the Web, Kristian Sons, Felix Klein, Dmitri, Rubinstein, DFKI Saarbrüuecken, Saarland University, DFKI Saarbruecken , Saarland University, Sergiy Byelozyorov Saarland University, Philipp Slusallek DFKI Saarbruecken Saarland University, 2010

[AnySL2010] PROGRAMMING AND COMPUTER GRAPHICS GROUPS OF SAARLAND UNIVERSITY, 2010. AnySL., http://www.prog.unisaarland.de/projects/anysl/.

[Rubinstein et al. 2009] RUBINSTEIN, D., GEORGIEV, I., SCHUG, B., AND SLUSALLEK, P. 2009. RTSG: Ray Tracing for X3D via a Flexible, Rendering Framework. In Proceedings of the 14th International Conference on Web3D Technology 2009, (Web3D Symposium '09), ACM, New York, NY, USA, 43-50.

[XML3D 2010] XML3D - Interactive 3D Graphics for the Web, Kristian Sons, Felix Klein, Dmitri, Rubinstein, DFKI Saarbruecken, Saarland University, DFKI Saarbruecken , Saarland University, Sergiy Byelozyorov Saarland University, Philipp Slusallek DFKI Saarbruecken Saarland University, 2010

[XML3D] http://www.xml3d.org/

[W3C 2009a] W3C, 2009. Cascading Style Sheets. http://www.w3.org/Style/CSS/

[W3C 2009b] W3C, 2009. DOM, http://www.w3.org/DOM/

[W3C 2009c] W3C, 2009. CSS Animations Module Level 3. http://www.w3.org/TR/css3-animations/

[Flash FAQ] http://www.adobe.com/products/flash/faq.html

[Flash 3D] Understanding the 3D features of Flash Player and the AIR runtime. `http://help.adobe.com/en_US/as3/dev/WS5467498E-BCF8-454f-8607-A51AD392CC07zephyr_serranozephyr.html`

[JavaFX features] JavaFX for Designers and Developers http://www.javafx.com/about/at-a-glance.jsp

[JavaFX partners] `http://javafx.com/partners/details/device_manufacturers.jsp`

[JavaScript 2006] JavaScript: the definitive guide, David Flanagan, O'Reilly Media, Inc., 2006

[Collada] Collada.org, Collada wiki `http://www.collada.org/mediawiki/index.php/Main_Page`

[C3DL 2008] Canvas 3D JS Library, Catherine Leung, Andor Salga, Andrew Smith, Proceeding Future Play '08 Proceedings of the 2008 Conference on Future Play: Research, Play, Share, 2008

[GoogleChrome2011] Visualizing geographic data with the WebGL Globe Doug Fritz `http://googlecode.blogspot.com/2011/05/visualizing-geographic-data-with-webgl.html`

[WEB3D97] VRML97 and Related Specifications The web3d consortium `http://www.web3d.org/x3d/specifications/vrml/`

[WEB3D08] X3D and Related Specifications The web3d consortium `http://www.web3d.org/x3d/specifications/`

[HTML5 2010] Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development, Peter Lubbers,Brian Albers,Frank Salim,Ric Smith, Apress, 2010

[SVG 2002] Scalable vector graphics (SVG): the world wide web consortium's recommendation for high quality web graphics, Dean Jackson, SIGGRAPH '02: SIGGRAPH 2002 conference abstracts and applications, ACM 2002

[WebGL 2010] Enabling WebGL, Catherine Leung, Andor Salga, WWW '10: Proceedings of the 19th international conference on World wide web, ACM 2010

[Khronos 2009] KHRONOS, 2009. WebGL. http://www.khronos.org/webgl/.

[Sandy 3D 2011] Sandy 3D Engine, 2011. `http://www.flashsandy.org/about`

[Papervision3D 2010] Papervision3D, 2010. `http://code.google.com/p/papervision3d/wiki/Getting_Started_FAQ`

[Statowl 2011] Statowl, 2011. `http://statowl.com/flash.php`