JSpatial Explorer

Mihir S. Padaya, Pragati M. Patil, Sakshi S. Chavan, Saee K. Hirve

Department of Computer Engineering
Pune Vidyarthi Griha's College of Engineering &
S.S. Dhamankar Institute of Management

Project Guide: Prof. I. M. Shaikh

---***---

Abstract- The result of Web 2.0 sciences in the way that Asynchronous JavaScript and WebGL, netting-based requests accompanying rich elements of larger object are progressing to be to a greater extent like sane uses in facets, to a degree interactivity, functionality, and utility. This development creates it likely to create netting-located aids, providing maps for consumers to search and leaf through geographic facts. This belief is an judgment of use, usability and veracity for the four netting-located plan APIs: Google Maps, Microsoft Virtual Earth, Multimap and ViaMichelin After experiment and evaluation of the APIs, the judgment is that no one of bureaucracy maybe named as the winner. They all have benefits and disadvantages; dissimilarities in conditions of service, compatibility, utility, geocoding and growth support, and the choice of API is as a consequence reliant of the type of use. As a result of this, and the fact that the APIs are uniformly changeful in conditions of performance and coverage, it is main to found requests free of the outline service provider. This was favorably finished all the while the teacher's assistantship by creating a graph contemplation coating in-betwixt the requests and the maps, creating the likelihood to switch API, or outline internet access provider, outside changed the law.[1]

Keywords: *Quadtree, Mercator Projection, WebGL, topography, API, Geocoding, pipeline, texturing, Clipmapping, Tilebuffer.*

INTRODUCTION

Web applications are a very important platform because of their ability to reach a large number of people easily. The ongoing advent of HTML5 and WebGL opens a wide range of possibilities in all areas including Geographical information systems (GIS). Three-dimensional visualizations of the Earth are very modern-looking, popular, and allow us to display some attributes (elevation data, 3D buildings, etc.) that can't be shown with a classical top-down view. Many web pages need to present some data to the visitors on a map and sometimes even on a 3D model of the Earth. At the present time, the only way to achieve such 3D visualization is by using closed-source Google Earth API, which requires the installation of a browser extension. Many users are not entitled to perform this kind of operation in the operating system they're using and need to ask a system administrator to install it for them. The goal of the WebGL Earth project is to provide an alternative open-source solution implemented in JavaScript and readily available in all WebGL-enabled browser. Existing 3D imagination answers work only with sure browsers or accompanying additional gateway element installed. This disadvantage maybe overcome by assembling the imagination using WebGL and HTML5. Applying specific approach, 3D potential can now come into being straightforwardly in the browser outside some need for an supplementary plug-in or continuation. Another benefit of WebGL over added technologies is that it resorts to fitting drawings card thought for effecting and performing movements on 3D details and hence, it specifies fittings increased 3D functionality on computer network…Working with WebGL has a few restrictions, though. Major browsers like Google Chrome, Mozilla Firefox, Safari, and Opera all support WebGL, although some others, like Internet Explorer 11, only partially do so. WebGL is also a low-level API because it is made to communicate with graphics cards directly. Therefore, hardware with insufficient graphics card RAM may have serious performance problems. Additionally, using pre-defined WebGL libraries like Three.js, Scene.js, and GLGE is required to prevent complex low-level code.[2]

LITURATURE SURVEY

This chapter contains the existing and established theory and research in this report range. This will give a context for work which is to be done. This will explain the depth of the system. Review of literature gives a clearness and better understanding of the exploration/venture. A literature survey represents a study of previously existing material on the topic of the report. This literature survey will logically explain this system. [3]

The first context type that was made widely available was 2D context. It is designed as a state machine (similarly to OpenGL) and can be used for high-performance rendering of two-dimensional objects such as lines, rectangles, arcs, text or bitmap images. The specification also defines methods for controlling shadow rendering or creating gradient fills. The majority of modern web browsers already supports this context type and most implementations are already GPU-accelerated (on certain platforms). The 2D context can be (together with JavaScript) successfully used to create interactive web visualizations or even web games right in the browser itself. [4]

The second context type enables web developers to create high performance three-dimensional presentations, which was previously possible only by using browser plug-ins or extensions. The most significant disadvantage of this approach was that the user had to have some additional (often proprietary) software installed (such as Adobe Flash Player or Java Virtual Machine). WebGL context serves as a binding between high-level JavaScript and low-level GPU operations. WebGL (Web-based Graphics Language) is based on OpenGL ES 2.0 (Open Graphics Library for Embedded Systems) to be easily implementable on modern mobile platforms such as Android or iOS (iPhone OS).[5]

There are many ways of projecting surface of the Earth to a plane. One of the most popular is the Mercator projection, which is defined by the following equations, where λ and φ are longitude and latitude and [x, y] are coordinates of the projected point on the map. The principle of this projection is also often described as a balloon (representing the Earth) placed inside a cylinder. The goal of WebGL Earth is to display a virtual globe and to be able to zoom down to street-level. We need to be able to display maps with up to 23 zoom levels — 223 × 223 tiles at the most detailed zoom level. With tiles being 256 px×256 px and having a 32-bit depth, we would have to be able to store more than 4 exabytes1 of data directly in video memory. That is impossible on today's mainstream hardware, which usually has 512 megabytes or 1 gigabyte of video memory. Therefore, we had to implement some form of tile management system, that would take care of dynamic tile streaming, caching, and real-time lookup in a GLSL shader program during rendering. [6]

Virtual Textures (as described by Mittring [MG08]) is a method of large texture management that, as its name indicates, adopts the fundamental ideas of memory virtualization from computer operating systems. In implementations of virtual texturing, the whole texture is divided into pages (we can use tiles directly as pages, because 256 × 256 px division provides enough granularity). There is a large buffer in video memory which serves as a page buffer and where the needed tiles are placed. The application also has to maintain a page table (or lookup table) that will provide information (in constant time) whether the page with given coordinates is available in the buffer. Virtual texturing implementations also usually employ the so-called feedback buffer into which the scene is being rendered with a special shader program applied. The feedback buffer is then transferred to RAM and analyzed. It contains information about what tiles should optimally be in the buffer at the given moment. [7]

This method was first introduced by C. Tanner [TMJ98] and is based on findings from mipmapping.

According to L. Williams' definition of a mip map [Wil83], a mip map is a pyramid-shaped collection of images with progressively lower resolutions until they are 1 1 pixels. The mipmap pyramid can be created "on-the-fly" by modern graphics cards as they buffer texture data. When rendering a geometry with mipmapped texture, the relevant texels (texture elements) from the pyramid's levels are selected based on a number of different considerations. Camera distance, angle from the surface, and texture filtering are the three factors that matter most. This improves performance and simultaneously addresses aliasing problems by reducing the amount of texels that must be read from the visual memory. As we've previously mentioned, virtual texturing's lack of scalability is its fundamental drawback. We tried a reverse strategy rather than using a huge lookup database with a lot of empty space. We added metadata that describes each buffer's "slot's" content (in context of this method called the TileBuffer. [8]

One of the most crucial services offered by geographical information systems and web-based map APIs is geocoding. This method can be described as the act of determining points, or geographic coordinates, on Earth's surface using alphanumeric addressing information. Parsing, matching, and locating are the three stages of the geocoding process. The address data given to the geocoder is examined and converted into an organized, standardized database tuple during the parsing step, which is then used in the matching phase. It is crucial to deal with diverse methods to write an address because this is the geocoding face where the human input is read. Common errors, various separators, and fictitious place names must all be addressed. [9]

AIM & OBJECTIVES

MOTIVATION

Existing browsers or additional browser plug-ins are required for certain existing 3D visualization technologies to function. By putting the visualization together using WebGL and HTML5, this restriction can be bypassed. By using such a method, 3D capabilities can now be realized without the use of an additional plug-in or extension, straight in the browser. Another advantage of WebGL over competing technologies is that it uses the graphics card memory of the hardware to show and process 3D content, offering hardware-accelerated 3D functionality on the web. Working with WebGL has a few restrictions, though. Major browsers like Google Chrome, Mozilla Firefox, Safari, and Opera all support WebGL, although some others, like Internet Explorer 11, only partially do so. WebGL is also a low-level API because it is made to communicate with graphics cards directly.

SYSTEM ARCHITECTURE

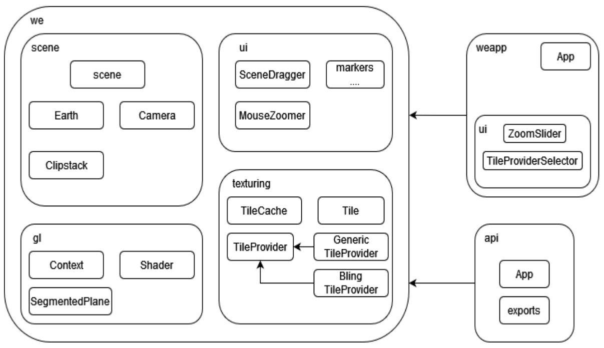

Fig -1: System Architecture Diagram

In fig-1, the project is divided into three main parts. The namespace we (WebGL Map) serves as a codebase and provides all of the core functionality for the other two namespaces called weapp and api. As their names suggest, weapp represents the final application (and additional UI (User Interface) controls) running on http://localhost:8000/, while api contains an application and exports for the WebGL Map API. If a programmer wants to build his own, complex application, the codebase in the we namespace should be used together with the Closure Compiler. However, for simple applications that use only the most common operations the API, should provide enough functionality, and can be easily used together with a wide variety of JavaScript libraries and compilers. This approach is similar to Google Maps model, where the core code base is shared between the Google Maps application itself (running at http://maps.google.com/) and the public JavaScript API.

FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS

Functional requirements: may involve calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements; these are captured in use cases.

Nonfunctional Requirements: (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

Functional requirements

- Multiple API
- Use of Quadtree
- Search Bar and Current Location

Design Constraints:

1. Database
2. Operating System
3. Web-Based Non-functional Requirements

Security:

1. User Identification
2. Login ID
3. Modification

Performance Requirement:

1. Response Time
2. Capacity
3. User Interface
4. Maintainability
5. Availability

APPLICATION:

- Dynamic web and Mobile Apps
- Satellite Imagery
- Data Visualization
- Game Development

SYSTEM REQUIREMENTS

Software Used:

- Python 3.9.0 or above, NodeJS and WebGL.

Hardware Used:

- I3 processor or above
- 150 GB Hard Disk or above
- 4 GB RAM or above

ALGORITHM

Step 1: Start

Step 2: Run app.js NodeJS file from Cmd.

Step 3: Browse: http://localhost:8000 in any browser.

Step 4: System Use the Algorithm developed for showing WebGL map.

Step 5: The map is shown on the browser, user can freely interact with map.

Step 6: User can check and change the API of map according to their usage.

Step 7: 3D WebGL map is showed on the screen

Step 8: Stop. [10]

CONCLUSION

Rich 3D material on the web has only been accessible for over a decade through third-party, frequently exclusive browser plugins. However, WebGL has arisen as a new standard to alter this. The produced graphics-based applications that support both 2D and 3D types of applications primarily adhere to the fundamentals of the WebGL and WebGL pipelines. The client user data conditions are verified by the HTML, CSS, and JavaScript codes. Consequently, it works with all sorts of web browsers. Local bandwidth and server capacity have multiplied in that time, opening up new opportunities for the internet. We have demonstrated how effective WebGL is for modelling and mapping applications using a case study, a Street View navigation web application, and other examples. In order to make the application even more collaborative, we will now focus on improving data accessibility and integration. The produced graphics-based applications that support both 2D and 3D types of applications primarily adhere to the fundamentals of the WebGL and WebGL pipelines. In order to make the application even more collaborative, we will now focus on improving data accessibility and integration. With the other widely used GPU technology, there will be a ton of potent 3D apps on the web in the upcoming years, and we are very happy to help shape the web. [11]

REFERENCES

[1] G. Qiu and J. Chen, "*Web-based 3D map visualization using WebGL*," 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018.

[2] Y. Yikang, L. Xinxing and L. Lei, "*A WebGL-Based Method for Visualization of Space Environment*," 2012 Fifth International Symposium on Computational Intelligence and Design, 2012.

[3] R. Miao, J. Song and Y. Zhu, "*3D geographic scenes visualization based on WebGL*," 2017 6th International Conference on Agro-Geoinformatics, 2017.

[4] Chaturvedi, Kanishk. (2014). *Web based 3D analysis and visualization using HTML5 and WebGL*.

[5] J. Jo and I. Jang, "*A cross-browser, web-based geospatial open platform using HTML5 and WebGL*," 2017 International Conference on Information and Communication Technology Convergence (ICTC), 2017.

[6] Sloup, Petr. "*WebGL Earth*." (2012).

[7] F Sun Z Zhang and D Liao "*A lightweight and cross-platform Web3D system for casting process based on virtual reality technology using WebGL*" International Journal of Advanced Manufacturing Technology vol. 80 no. 5-8 pp. 801-816 2015
.

[8] H. Kim, S. Nam, J. Park and D. Ko, "*Direct canvas: Optimized WebGL rendering model*," 2018 IEEE International Conference on Consumer Electronics (ICCE), 2018.

[9] Näslund, Magnus. (2008). *Web-based mapping : An evaluation of four JavaScript APIs.*

[10] Y. Chen, W. Shuai and X. Chen, "*A probabilistic, variable-resolution and effective quadtree representation for mapping of large environments*," 2015 International Conference on Advanced Robotics (ICAR), 2015.