MASSEY UNIVERSITY

Geographic Data Visualization with Immersive Virtual Reality

Author
Yang JIANG

Supervisor
Dr Arno LEIST

A thesis submitted in fulfillment of the requirements
for the degree of Master of Information Sciences
in the

Software Engineering
159.888 Computer Science Professional Project

October 21, 2016

Abstract

Virtual reality technology has been proved to be beneficial to visualize geographic data. It can be broken down into the pseudo-3D display with 3D interaction and the immersible depth of vision of a true 3D experience. The former not only has been already studied for a long time in the domain of visualizing the earth and environmental sciences, but also got a successful result in both theory and practice area. On the other hand, the latter is yet to be completely revealed. In this paper, an immersive virtual reality based intuitive nature system that provides attractive and efficient methods for simultaneously visualizing geographic data from the different source is highlighted. It exposed by revealing the implementation of an application that includes both front (client) and back-end (web server) for geographic data visualization.

Keywords: Geographic Information, Visualization, Virtual Reality

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview and Objectives

The Geographic Information System (GIS) often refers to many different technologies, processes, and methods that designed to capture, store, manipulate, analyze, manage, and present spatial or geographical data [42]. A GIS combines a database management system and a graphic display system that tie to the process of spatial analysis [31]. Indeed, GIS has been widely used in the analysis of environmental data, but still significant problems have had exposed: first, GIS itself only handle 2D data; second, displays are limited to spatial views of the data; third, the capability of supporting user interaction with negligible data [32].

The Open Geospatial Consortium (OGC) is committed to making quality open standards for the global geospatial group. These standards were decided through a consensus based process and are freely available for anyone to sharing of the world's geospatial data. They have made contributions to many communities including government, commercial organizations, non-governmental organizations, academic and research organizations [29]. To use a markup language maintained by OGC for the creation of 3D geographic maps and associated spatial data allows scientists to publish the latest information in a single, simple data file format without technical assistance.

Since the technique progress in computers and the quick development in pipelined 3D graphics, GIS now can be used either with a workstation window based interface or with an immersive virtual reality environment [23]. In recent years, the immersive virtual reality not only frequent occurrences nearly in all sorts of media, but also it has explored a mess of related products. For example, immersive virtual reality headsets developed by manufacturers over the world, and the 3D camera that can capture a 360 degrees field of view. However, it is not mature enough to eliminate the equipment limitation and becomes a universal technology in daily life by comparison to the pseudo-3D virtual reality technology. For instance, when it comes to exploring, routing or getting to places, most people should just reach for Google Earth or Google Map.

There is a lack of research on visualizing geographic data in the immersive virtual reality environment. Therefore, we cannot yet say whether or not immersive virtual reality for geographic data visualization is better than other visualization and analysis approaches for certain data, if so, by how much; what basic considerations would be involved in immersive virtual reality

based geographic data visualization.

The purpose of this study are to explore how geographic data visualization with immersive virtual reality affect user interfaces and human-computer interactions; measure the ranges and capabilities of any necessary sensors; evaluate minimum equipment costs; takes advantage of the GIS and develop a virtual reality based geographic data visualization application including both front (client) and back-end (web server). In this thesis, a background of geographic data visualization is presented. Then, details of the related technology and implementation in respect of the application are described. Finally, there is a discussion and conclusion around the results and future research.

1.2 Background

There has been an increased interest in the exploration of Virtual Environments (VE) [19], sometimes called Virtual Reality (VR). Since beginning of 1990s when the development in the area of virtual reality became much more dynamic, and the term Virtual Reality itself became extremely popular, a wide range of applications were developed relatively fast, which offers significant benefits in many area, such as "architectural walkthrough", "scientific visualization", "modeling, designing and planning", "training and education", "telepresence and teleoperating", "cooperative working" and "entertainment" [25]. Among these applications, virtual reality technology has been proved it offers new and exciting opportunities for users to interact visually with and explore 3D geo-data [19].

In the past, GIS were mostly 2D, map-based systems, but the concept of taking advantage of GIS to visualize the earth and environmental sciences data has been already studied for a long time. That is called Virtual Globe (VG) technology, most of the virtual globe products use 3D representations of objects and display them onto a 2D monitor. This pseudo-3D nature of virtual globes allows users to interact in an environment that makes the data and information present easier to understand [34]. Then it has become a powerful tool for navigating geospatial data in 3D and contribute to all kind of communities across different usage till now.

Given the current rapid development of virtual GIS technology, they made a point of the motivation of virtual reality technology is that people always want more, they want able to step into the world and interact with it, instead of watching the 2D projection image on the monitor. VR provides an easy used, powerful, intuitive way of user interaction. The user can experience and manipulate the simulated 3D environment in the same way they act in the real world, without any preparation or understanding of the complicated user interface works. It soon became a perfect tool that is beneficial to architects, designers, physicists, chemists, doctors, surgeons etc. Without a doubt VR has a great potential to change our life, the expectation from this technology is much more than it can offer yet. [25] also discussed on an interesting

idea that they came up with: new invention brings fear, the more potential it has, the bigger the danger can be.

The success of virtual globes [34] is not only because the improvement of human understanding from its pseudo-3D representations of objects and spaces, but also the five features: transportability (digital data are easily transported), scalability, interactivity (users are in control of the experience), choice of topics (topics can be combined or presented individually or), currency (ability to adjust the data available to any given time period), and client-side [34]. Virtual globes can be beneficial to education ('For teaching spatial thinking, Virtual Globes offer tremendous opportunities, and it can be expected that they will greatly influence how a new generation will perceive space and geographical processes.' [28]), scientific collaboration research (such as the EarthSLOT [6]), and disaster response (VG is an invaluable tool in disaster response [4], [27]). Virtual globe technology has many exciting possibilities for environmental science. The easy-to-use, intuitive nature system, provide attractive and effective means and methods for simultaneously visualizing four-dimensional environmental data from different sources that driving a greater understanding and user experience of the Earth system [3].

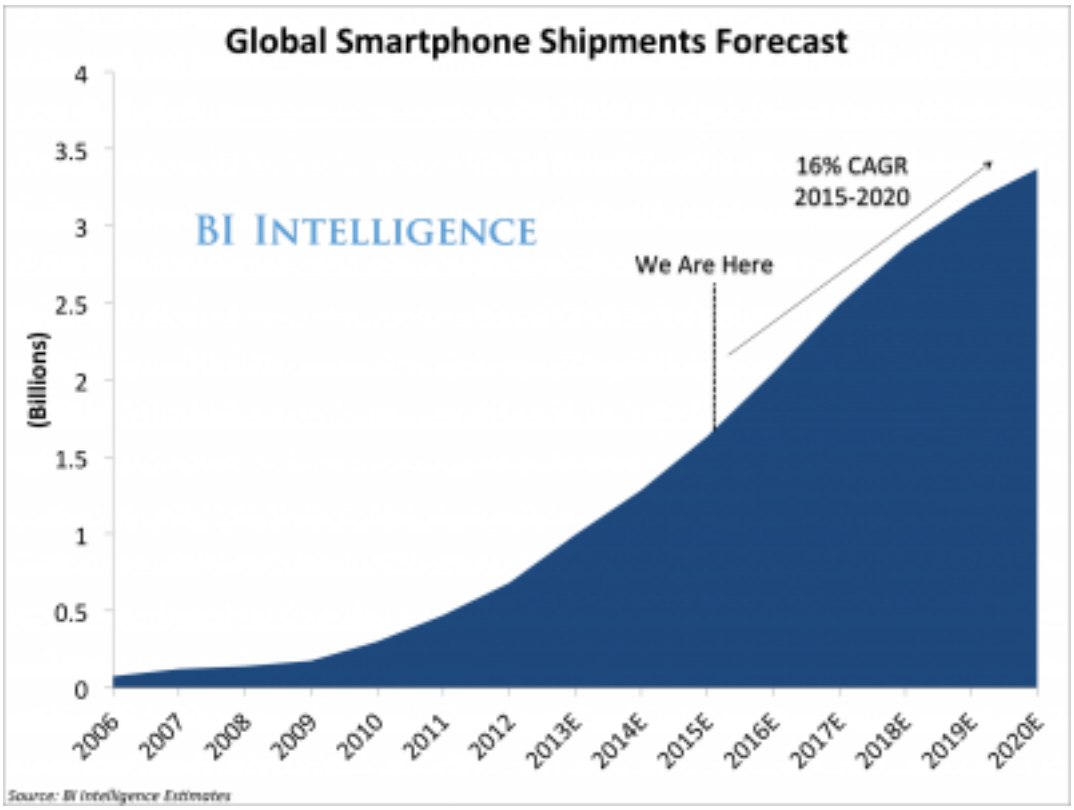
A markup language maintained by the Open Geospatial Consortium [29] plays an essential role in virtual reality implementation. By taking the use of a markup language, scientists are able to publish data in a single, simple data file format without technical assistance [3]. In spite of capabilities vary from products to products, but virtual globes always provide a support for a file format data exchange and the ability to simultaneously display multiple datasets. [3] points out Google Earth which has the largest community creates Keyhole Markup Language (KML) [14] files as its primary method for visualizing data (KML is an international standard maintained by the OGC); NASA World Wind [26] imports data from tile servers, OGC web services and a limited support for KML, it has more focus toward scientific users; ArcGIS Explorer [7] is a lightweight client to the ArcGIS Server, it can import data in a very wide range of GIS formats, including KML. Some of the virtual globes products are using Virtual Reality Modeling Language (VRML) [43] that is a language for describing 3D objects and interactive scenes on the World-Wide Web (WWW) [44], It has been superseded by X3D [45].

2 Technology

2.1 Android Phone

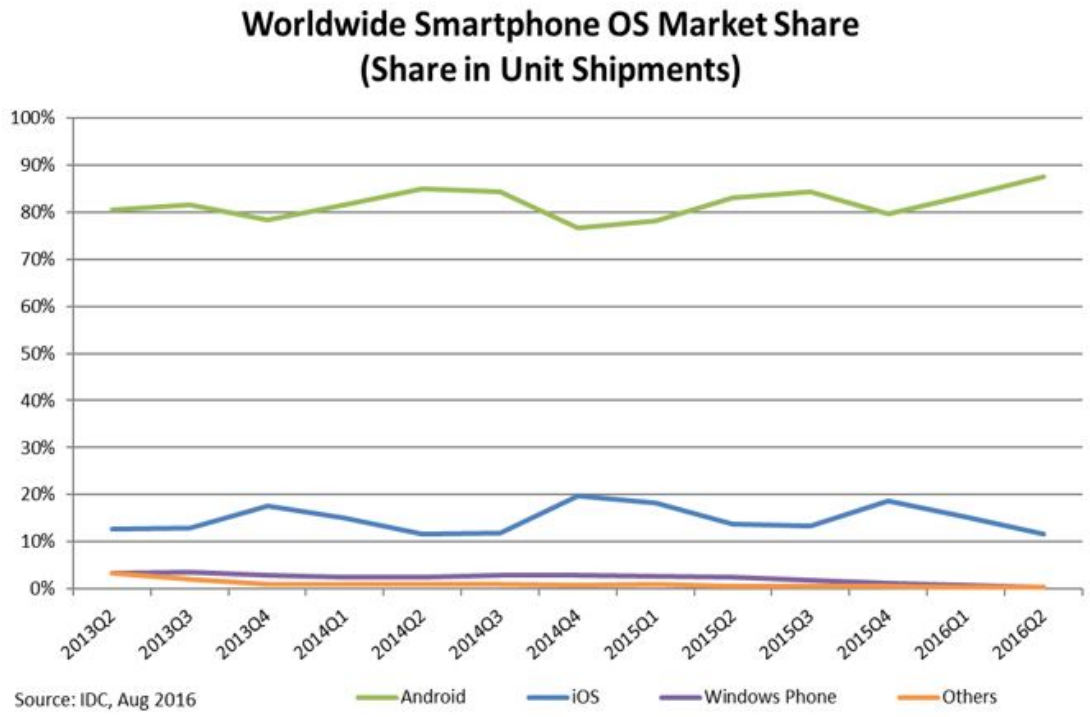
There are reasons we implemented this project on the Android device. We intend to build the virtual reality on a common and familiar device in the world. The smartphone fully deserves the title that not only it has an incredibly fast growth trend in the last decade and a good promising prospect, but also they have built-in sensors that measure motion, orientation and various environmental conditions. These sensors are capable of providing raw data with high precision and accuracy and are useful to monitor device movement - six degrees of freedom (DOF) - position coordinates (x, y and z offsets) and orientation (yaw, pitch and roll angles).

FIGURE 2.1: Global Smartphone Shipments Forecase [5]



According to data from the International Data Corporation (IDC), Android dominated the smartphone market with a share of 87.6% in the worldwide.

FIGURE 2.2: Smartphone OS Market Share [20]



Moreover, the perfect part is the Google VR SDK [13] for Android supports and the affordable Cardboard product [12] designed for different kind of mobile devices.

2.2 OpenGL

Android includes support for high-performance 2D and 3D graphics with the Open Graphics Library, specifically, the OpenGL ES API [15]. OpenGL ES is a flavor of the OpenGL specification intended for embedded devices. Android supports several versions of the OpenGL ES API:

TABLE 2.1: OpenGL ES API specification supported by Android

OpenGL ES Version	**Android Version**
OpenGL ES 1.0	Android 1.0 and higher
OpenGL ES 1.1	Android 1.0 and higher
OpenGL ES 2.0	Android 2.2 (API level 8) and higher
OpenGL ES 3.0	Android 4.3 (API level 18) and higher
OpenGL ES 3.1	Android 5.0 (API level 21) and higher

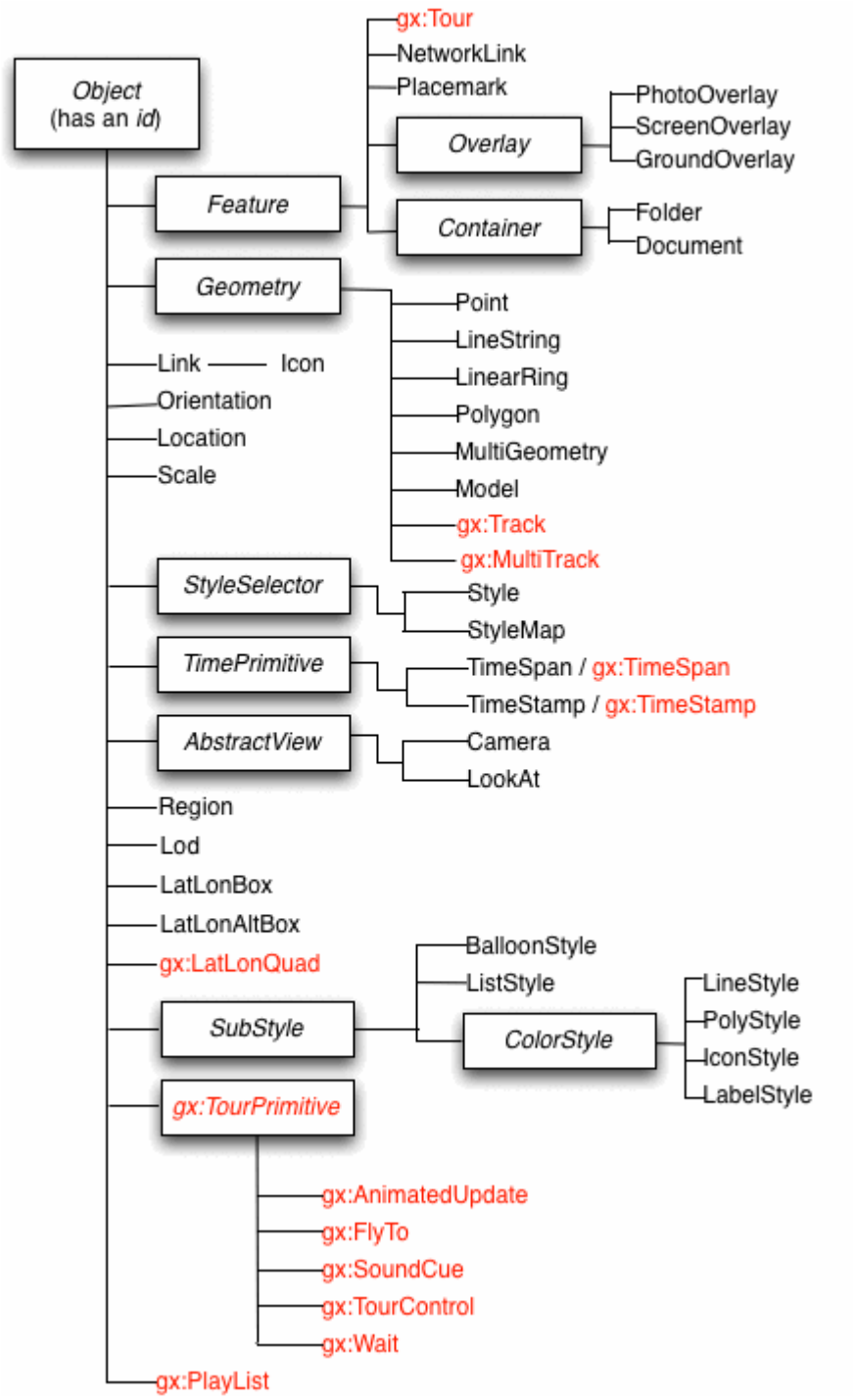
2.3 Keyhole Markup Language

We were looking for a simple markup language that we can publish and consume data in interoperable formats without the need for technical assistance. In the 1.2, we introduced KML (it can be combined with other supporting files such as imagery in a zip archive, producing a KMZ file), which is the markup language we are using in the project. It is not only because it can simply satisfy our application purpose, more importantly, it is supported by many virtual globes and other GIS systems and is therefore already becoming a de facto standard [3]. Moreover, the annotations of KML features are not designed as machine-readable XML, but a human readable plain text or simple HTML. Moreover, real-time data are important in the environmental sciences. The Networklink facility in KML allows all or part of the dataset to be automatically refreshed by the URL, to ensure the user always sees the latest information.

From an environmental science point of view, KML is a somewhat limited language. It can only describe simple geometric shapes on the globe (points, lines, and polygons) and is not extensible. It is, in many respects, analogous to Geography Markup Language (GML) 3.0+ is much more sophisticated and allows the rich description of geospatial features such as weather fronts and radiosonde profiles. For the above reasons, KML is currently not suitable as a fully-featured, general-purpose environmental data exchange format. Nonetheless, it earns the acceptance from an increasing number of scientists. It is important to be aware of that virtual geographic data visualization (and KML) do not attempt to replace more sophisticated systems.

Figure 2.3 shows the KML schema. From the point of view of usability, KML spans a gap between very simple (e.g. GeoRSS) and more complex (GML) formats, that makes it easy for non-technical scientists to share and visualize simple geospatial information which can then be manipulated in other applications if required.

FIGURE 2.3: KML schema [14]



2.4 Network

The key strengths of virtual reality applications are not only easy-to-use, and intuitive nature, but also the ability to incorporate new data very easily. Therefore, real-time data are very

important in the environmental sciences [3]. To do that, a web server is needed. In this project, we implemented a RESTful web server to support communication with the client, along with a file server to synchronize data. In the client side.

Go (often referred to as golang [17]) is an open source programming language, and it is compiled, concurrent, garbage-collected, statically typed language developed at Google in late 2007. It was conceived as an answer to some of the problems we were seeing developing software infrastructure [9]. Also, it growing fast that each month the contributors outside Google is already more than contributors inside the Go team.

We are using Go to build the server, it is well suited for developing RESTful API's. The net/http standard library provides key methods for interacting via the HTTP protocol. On the other hand, since our client is Android phone, we introduced Volley for transmitting network data (Volley is an open sourced HTTP library that makes networking for Android apps easier and most importantly, faster [18]), and jsoup (Java HTML Parser [22]) for analyzing HTML format response.

3 Implementation

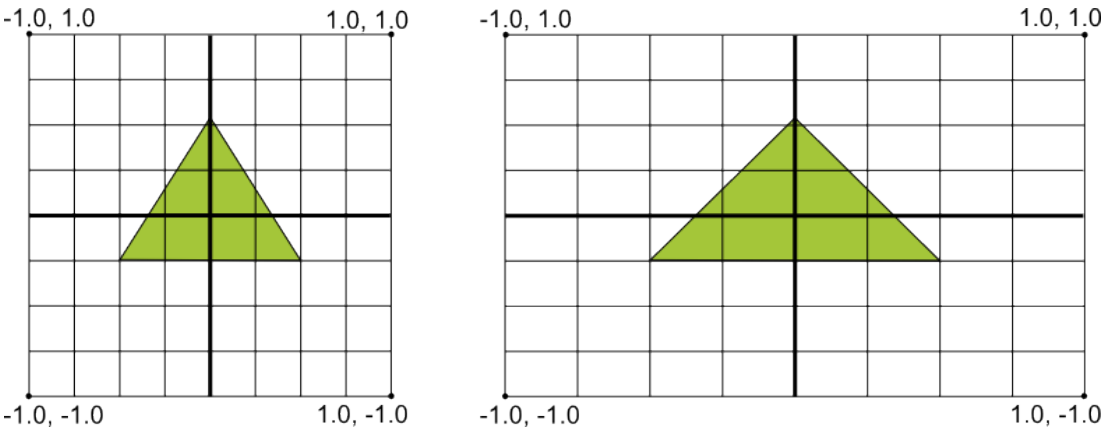
3.1 Google VR SDK

The Google VR SDK repository is free and accessible from *https://github.com/googlevr/gvr-android-sdk*, where we can get the necessary libraries and examples. The SDK libraries are in the libraries directory of the repository as *.aar* files [10]. This project has two dependencies on *base* and *common* modules.

3.2 OpenGL ES

OpenGL assumes a square, uniform coordinate system and, by default, happily draws those coordinates onto your typically non-square screen as if it is perfectly square. But the problem is that screens can vary in size and shape:

FIGURE 3.1: Default OpenGL coordinate system (left) mapped to a typical Android device screen (right) [15]



The illustration above shows the uniform coordinate system assumed for an OpenGL frame on the left, and how these coordinates actually map to a typical device screen in landscape orientation on the right. To solve this problem, you can apply OpenGL projection modes and camera views to transform coordinates so your graphic objects have the correct proportions on any display.

In order to apply projection and camera views, you create a projection matrix and a camera view matrix and apply them to the OpenGL rendering pipeline. The projection matrix recalculates the coordinates of your graphics so that they map correctly to Android device screens. The camera view matrix creates a transformation that renders objects from a specific eye position.

TABLE 3.1: OpenGL compute

What	**How**	**Where**
Model Matrix	translationMatrix * scaleMatrix * rotationMatrix * matrix(1)	CPU
Camera Matrix	Matrix.setLookAtM(positionV, lookAtV, upV)	CPU
View Matrix	eye.getEyeView() * cameraM	CPU
Perspective Matrix	eye.getPerspective(zNear, zFar)	CPU
Projection Matrix	perspectiveM * viewM * modelM	GPU
Vertex'	projectionM * vertex	GPU

3.3 Server

As mentioned in 2.4, Go is well suited and super easy for developing the network. A simple localhost file server on port 8080 to serve a directory on disk (/tmp) under an alternate URL path (/files/), use StripPrefix to modify the request URL's path before the FileServer sees it.

```
http.Handle("/files/", http.StripPrefix("/files", http.FileServer(http.Dir("./tmp"))))
http.ListenAndServe(":8080"), nil)
```

For RESTful APIs, we introduce a free framework Go-Json-Rest [21], it is a thin layer designed by KISS principle (Keep it simple, stupid) and on top of native net/http package that helps building RESTful JSON APIs easily.

Note that, a file server is satisfied all need from the client at this moment. Although the RESTful is setup, but there is no RESTful APIs is actually in use yet.

3.3.1 Assets

Following is the folder structure served by file server:

TABLE 3.2: Assets structure

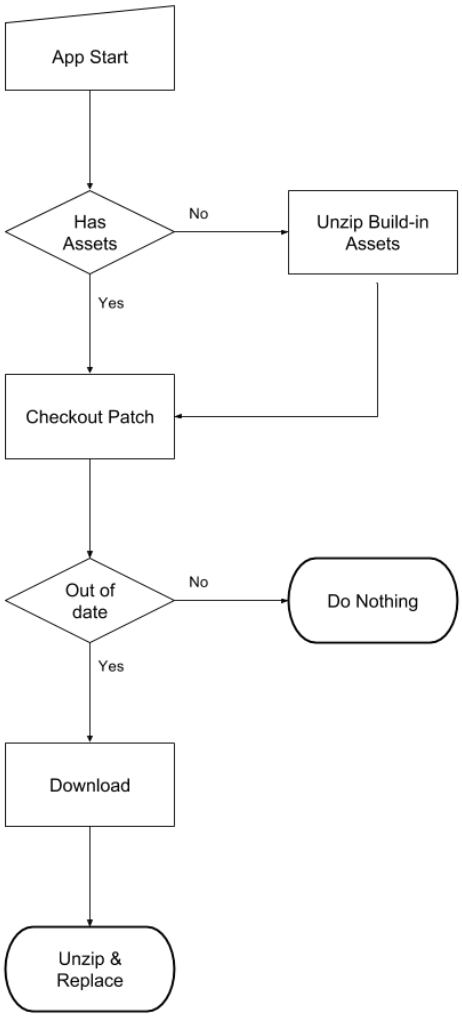
Path	**Usage**
\assets	Root
\assets\static.zip	Patch (see 3.3.2) compressed file
\assets\static\kml	KML (see 3.4.1) storage
\assets\static\layer	KML storage (see 3.4)
\assets\static\model	OBJ model (see 3.6.3) storage
\assets\static\resource	Image storage

3.3.2 Patch

Patch check is happening every time when the app starts. First of all, client checks out the patch file (*assets**static.zip*) from the file server, comparing the*lastModifiedTime* with local patch file, and only continue to download if local patch out of date. Once the patch file is downloaded, replace any existing files.

Note that a built-in default patch is included in the apk (Android app binary) in a case of client disconnect from the internet for the first time launch time that no available data should be avoided.

FIGURE 3.2: Patch check



3.4 Scene

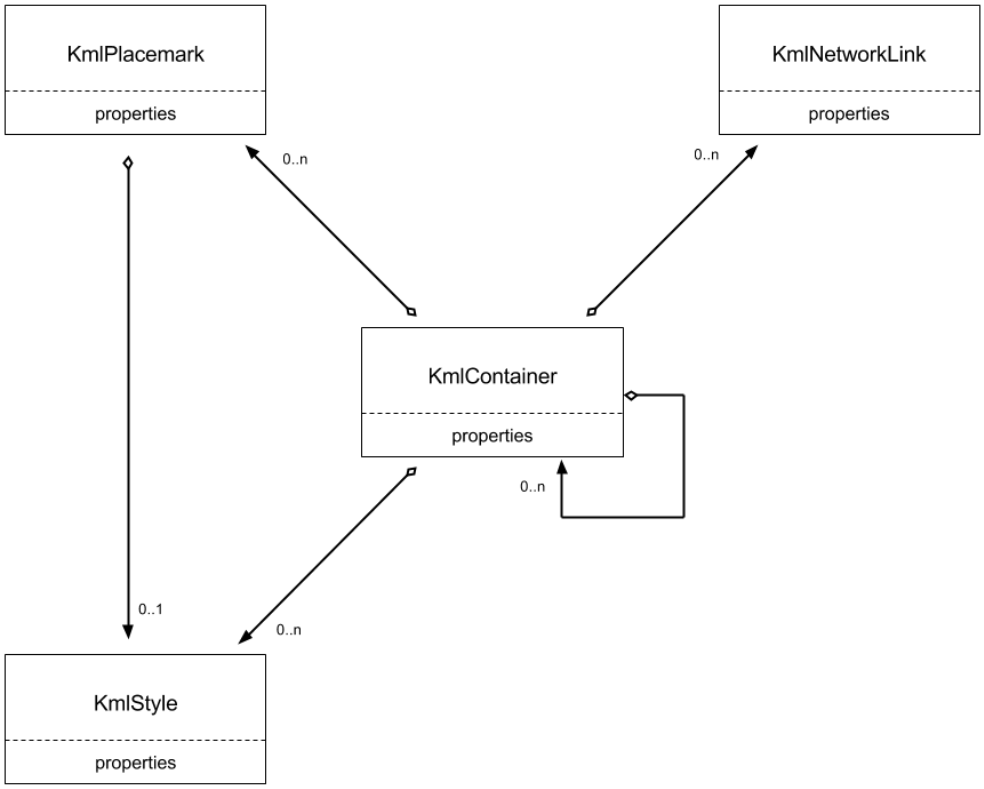
A layer list shows available KML files from *\assets\static\layer*, scene will be created according to which KML is selected. The KMLs in *\assets\static\layer* is literally same as KMLs in *\assets\static\kml*, only different is that user can only able to see laye that achieving the idea of KML categorization by KML NetworkLink feature (see 3.4.1). The NetworkLink feature allows a KML file (*\assets\static\layer*) includes one or more KMLs (*\assets\static\kml*).

3.4.1 Keyhole Markup Language

In this project, we only take use of few feature of KML 2.3: Container, Style, Placemark, and NetworkLink. The KML parser we are using is not coded from scratch, but is based upon the open-source library `android-maps-utils` [11] (NetworkLink is one of the unsupported

features in the library). Main modifications are getting rid of `GoogleMap` dependency, and extending `NetworkLink` feature support in accordance with the current design pattern.

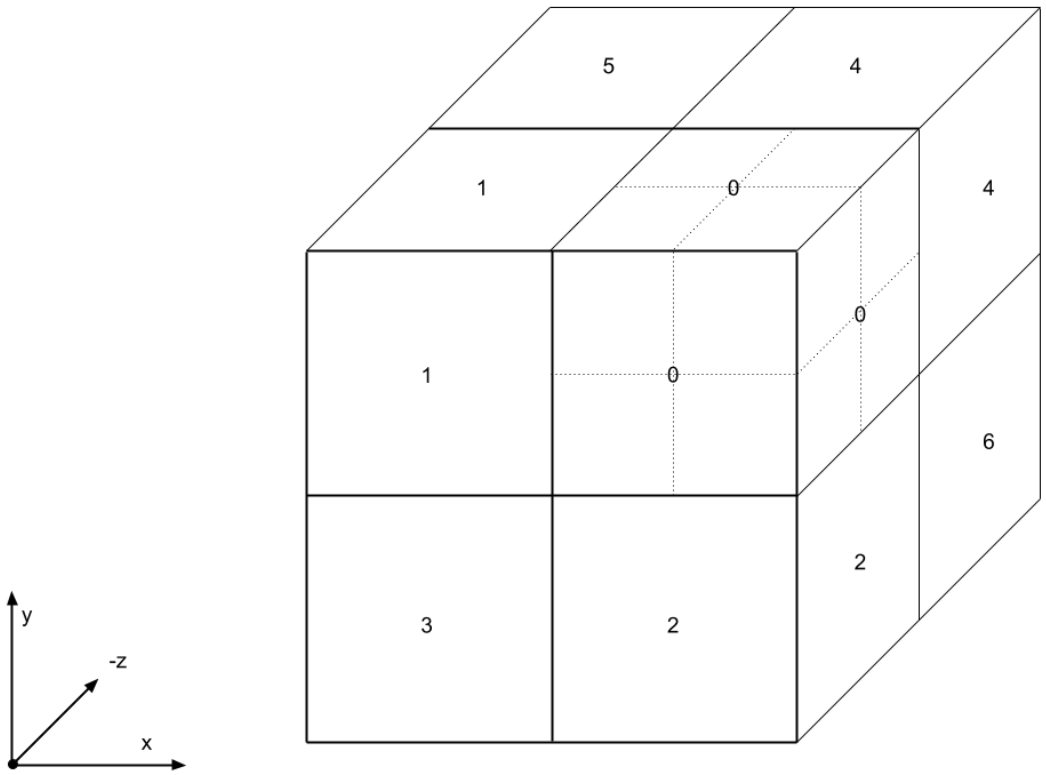
FIGURE 3.3: kML parser simple



3.4.2 Octree

To reduce the ray-object intersection tests, space partitioning is needed. The main requirement is not to use a spatial data structure for a ray intersect with irregular geometry, but to determine what objects are in the same cell to avoid doing an n^2 check on all objects. In this case, it is spherical placemark. Therefore, we do not need overlapping volumes, and contained objects do not need to be cut on volume boundaries. It is actually 3D space partitioning process with a predefined restricted maximum number of objects in the same cell. A simple axis-aligned Octree is fully satisfied in here 3.4.

FIGURE 3.4: Octree split



For each box splitting process, we also generate eight indexes to indicate the relative position inside the box. These indexes are important for the next partition, where we might need to relink contained objects to the new corresponding box. To insert an object into the box only if the existed contained number of objects is less than the predefined constant value, otherwise splitting the box then relink existed object and insert the new object again.

Integer indexes of box is defined by three boolean value that indicates three axis-relative value:

Any position P in the box with known center$O$:

$$dx = P_x - O_x$$
$$dy = P_y - O_y$$
$$dz = P_z - O_z$$

TABLE 3.3: Octree octant

Index	Octant	Geometric Meaning
0x00000000	T, T, T	dx > 0, dy > 0, dz > 0
0x00000001	F, T, T	dx < 0, dy > 0, dz > 0
0x00000010	T, F, T	dx > 0, dy < 0, dz > 0
0x00000011	F, F, T	dx < 0, dy < 0, dz > 0
0x00000100	T, T, F	dx > 0, dy > 0, dz < 0
0x00000101	F, T, F	dx < 0, dy > 0, dz < 0
0x00000110	T, F, F	dx > 0, dy < 0, dz > 0
0x00000111	F, F, F	dx < 0, dy < 0, dz < 0

Octant solution:

```
octant[] = (index & 1, index & 2, index & 4)
```

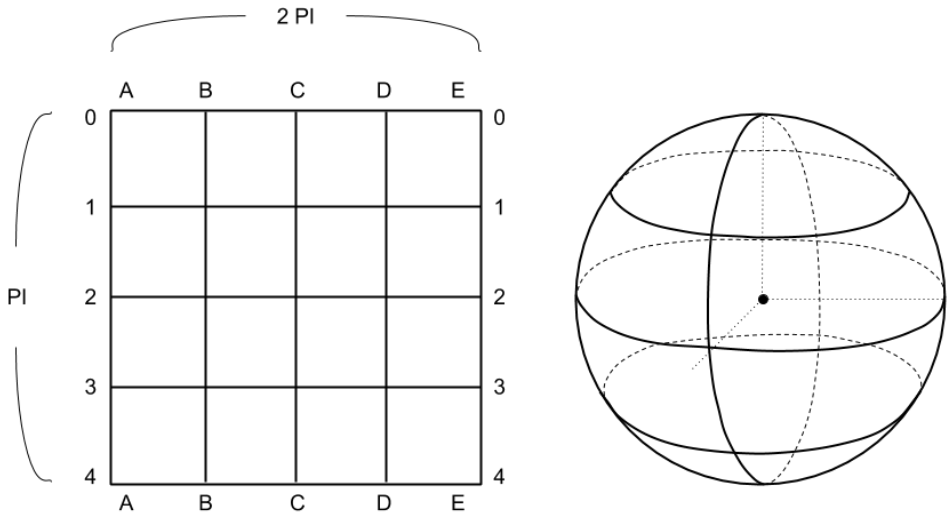
Index solution:

```
For Each octionn[i]:
                    index |= (1 < < i)
```

3.5 Earth

The Earth is created as a UV Sphere, which somewhat likes latitude and longitude lines of the earth, uses rings and segments. Near the poles (both on the Z-axis with the default orientation) the vertical segments converge on the poles. UV spheres are best used in situations where you require a very smooth, symmetrical surface.

FIGURE 3.5: UV sphere mapping

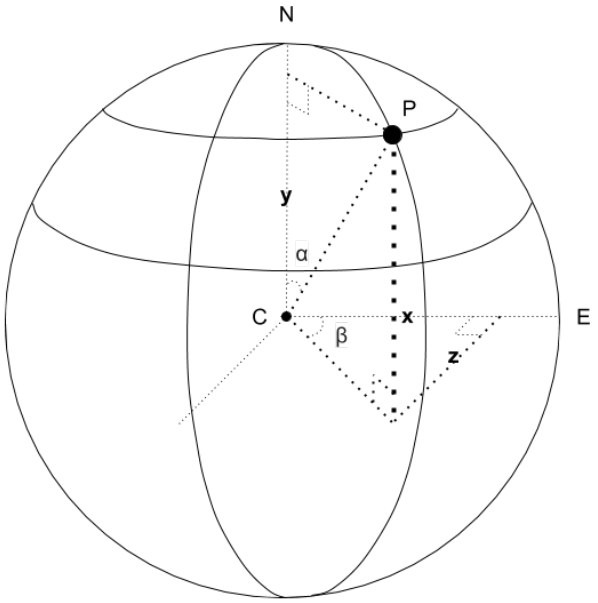


As we can see the mapping from 3.5. Vertex $A0,\ A1,\ A2,\ A3,\ A4$ and $E0,\ E1,\ E2,\ E3,\ E4$ are duplicated, and $A0,\ B0,\ C0,\ D0,\ E0$ converge together as well as $A4,\ B4,\ C4,\ D4,\ E4$. So we can simply define it as a UV sphere has 5 rings and 4 segments. Also be noticed that each ring spans $2\,\pi$ radians, but each segment spans π radians in the sphere mapping.

The total vertex number is:

$$Vertices = Rings \times Segments \tag{3.1}$$

FIGURE 3.6: UV sphere vertex



For each vertex P on sphere from ring r and segment s, we have:

$$
\begin{aligned}
v &= r \times \frac{1}{rings - 1} \\
u &= s \times \frac{1}{segments - 1} \\
\measuredangle\alpha &= v \times \pi \\
\measuredangle\beta &= u \times 2\,\pi
\end{aligned}
$$

$\therefore$ P (x, y, z)

$$
\begin{aligned}
x &= (\sin(\alpha) \times radius) \times \cos(\beta) \\
y &= \cos(\alpha) \times radius \\
z &= (\sin(\alpha) \times radius) \times \sin(\beta)
\end{aligned}
$$

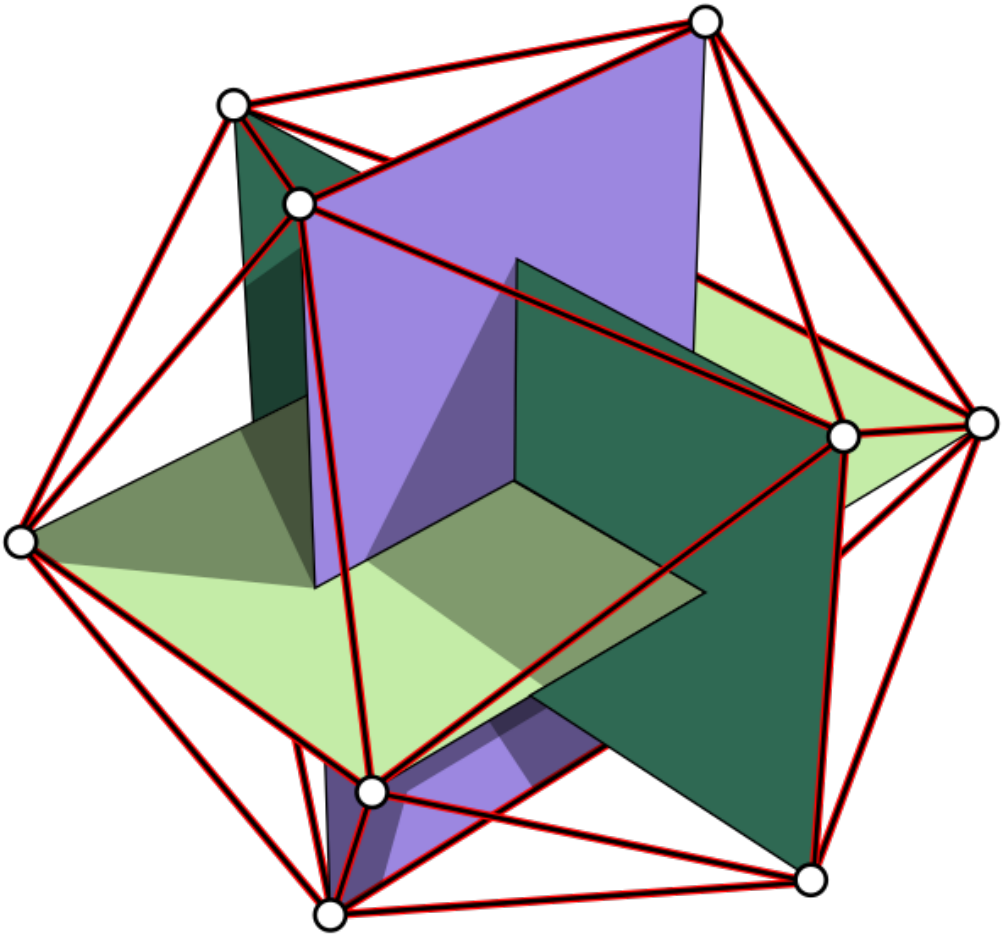
& 2D Texture (x, y) mapping for vertex P is:

$$
\begin{aligned}
x &= u \\
y &= v
\end{aligned}
$$

3.6 Placemarker

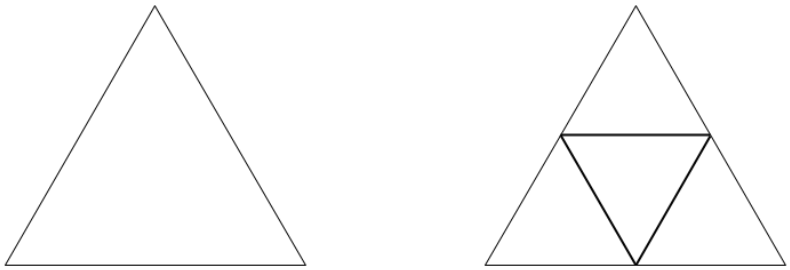
Generation of vertices for placemarker is a recursion process of subdividing icosphere. Figure 3.7 shows that the initial vertices of an icosahedron are the corners of three orthogonal rectangles.

FIGURE 3.7: Icosahedron rectangles [38]



Rounding icosphere by subdividing a face to an arbitrary level of resolution. One face can be subdivided into four by connecting each edge's midpoint.

FIGURE 3.8: Icosphere subdivide



Then, push edge's midpoints to the surface of the sphere.

FIGURE 3.9: Icosphere refinement

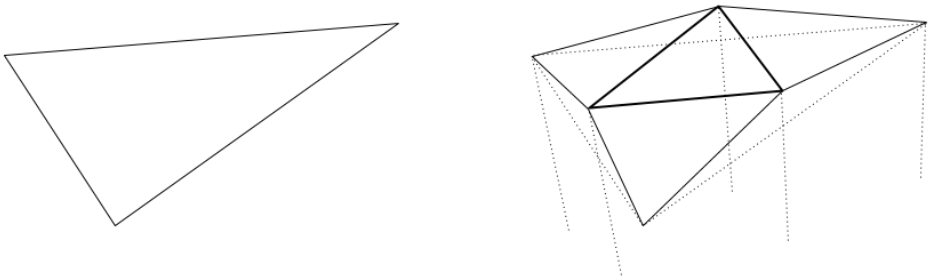


TABLE 3.4: Rounding Icosphere

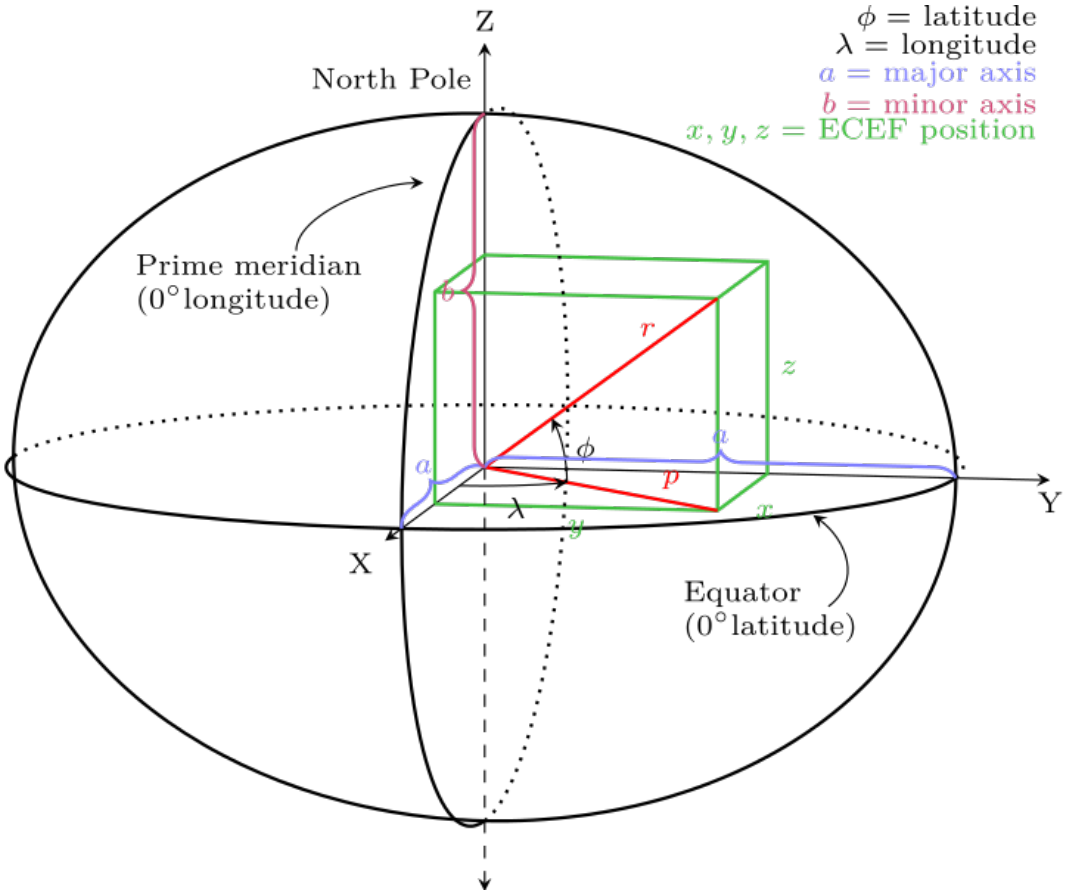
Recursion Level	Vertex Count	Face Count	Edge Count
0	12	20	30
1	42	80	120
2	162	320	480
3	642	1280	1920

3.6.1 Geographic Coordinate System

A geographic coordinate system is a coordinate system that enables every location on the Earth to be specified by a set of numbers or letters, or symbols [41]. A common geodetic-mapping coordinates are latitude, longitude, and altitude (LLA), which also is the raw location data read from KML.

We introduce ECEF ("earth-centered, earth-fixed") coordinate system for converting LLA coordinates to position coordinates. According to, the z-axis is pointing towards the north but it does not coincide exactly with the instantaneous earth rotational axis. The x-axis intersects the sphere of the earth at 0 latitude and 0 longitude [40].

FIGURE 3.10: Earth-centered, earth-fixed [40]



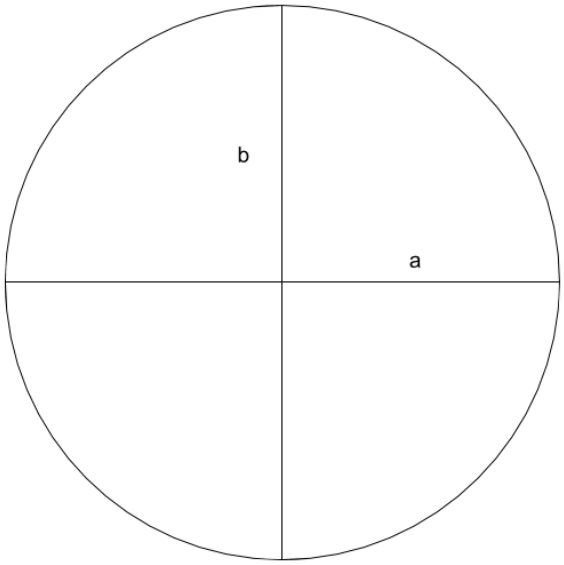
The ECEF coordinates are expressed in a reference system that is related to mapping representations. Because the earth has a complex shape, a simple, yet accurate, method to approximate the earth's shape is required. The use of a reference ellipsoid allows for the conversion between ECEF and LLA [35].

A reference ellipsoid can be described by a series of parameters that define its shape and which include a semi-major axis (a), a semi-minor axis (b), its first eccentricity (e_1) and its second eccentricity (e_2) as shown in Table 3.5.

TABLE 3.5: WGS 84 parameters

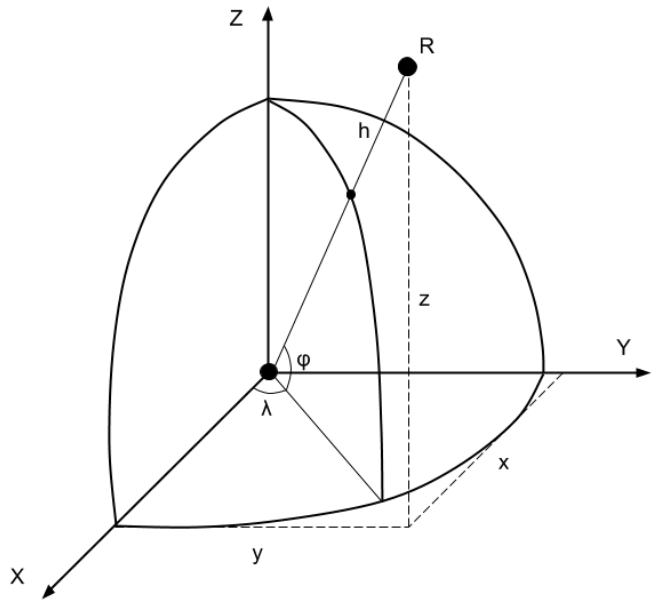
Parameter	**Notation**	**Value**
Reciprocal of flattening	$1/f$	298.257 223 563
Semi-major axis	a	6 378 137 m
Semi-minor axis	b	$a\,(1-f)$
First eccentricity squared	e_1^2	$1 - b^2/a^2 = 2\,f - f^2$
Second eccentricity squared	e_2^2	$a^2/b^2 - 1 = f\,(2-f)/(1-f)^2$

FIGURE 3.11: Ellipsoid parameters



The conversion from LLA to ECEF is shown below.

FIGURE 3.12: LLA to ECEF



$$x = (N + h)\ \cos(\varphi)\ \cos(\lambda)$$
$$y = (N + h)\ \cos(\varphi)\ \sin(\lambda)$$
$$z = (\frac{b^2}{a^2}\, N + h)\ \sin(\varphi)$$

Where

$$\varphi = \text{latitude}$$
$$\lambda = \text{longitude}$$
$$h = \text{height above ellipsoid (meters)}$$
$$N = \text{Radius of Curvature (meters), defined as:}$$
$$= \frac{a}{\sqrt{1 - e^2\ \sin(\varphi)^2}}$$

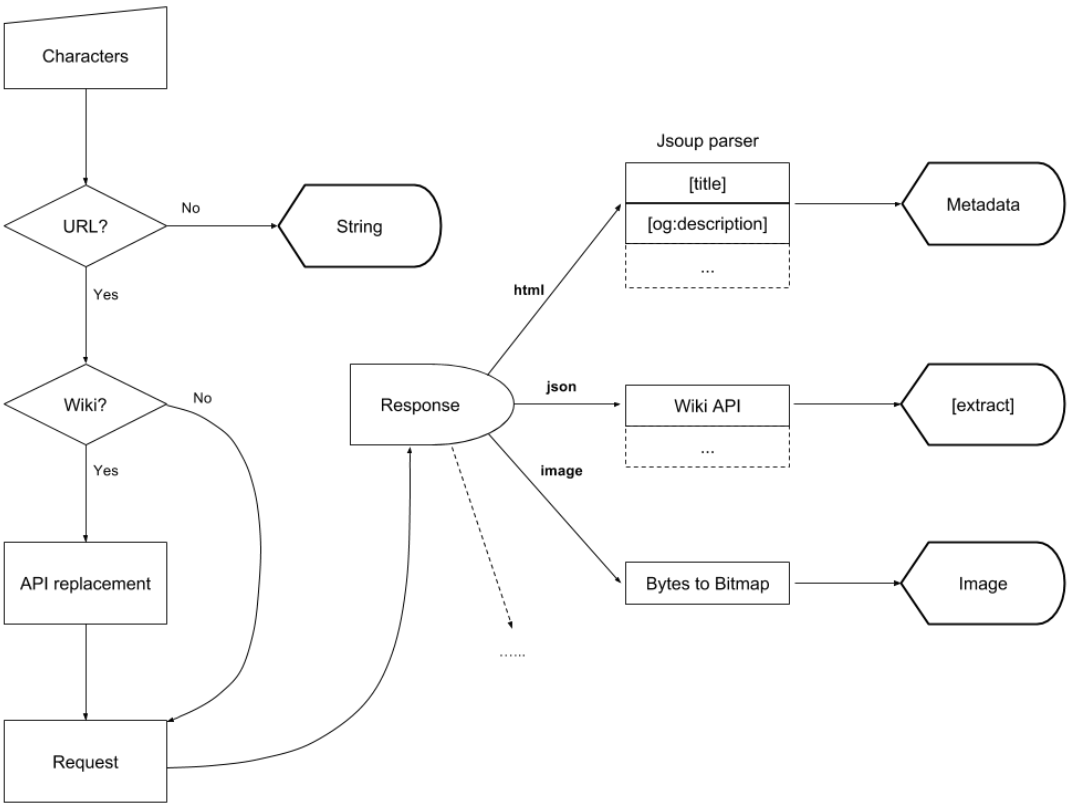
At last, for this project usage, where high accuracy is not required, a equals to b. And also the ECEF coordinate system is y-east, z-north (up), and x points to 0 latitude and 0 longitude, but for project specific, we still need to convert ECEF to x-east, y-north (up), and x points to 0 latitude and 180 longitude.

3.6.2 Description

Description of placemarker requires an appropriate analysis for display. The raw data of description is a set of characters that could be a normal text, an image URL, a URL returns different type of content, or maybe just some meaningless characters.

Although the implementation of analysis in this project did not cover every situation, but it is flexible and extendable for more functionality.

FIGURE 3.13: Description analysis



In order to get an extracted content from a Wikipedia page, we can transform the URL to a Wiki-API based open-search URL [39], which will return a json format raw data that we can easily get what we need from different json tags.

Replace `.wikipedia.org/wiki/`

To `.wikipedia.org/w/api.php?`$APIs$

Where $APIs$ is:

```
format=json
          &action=query
          &redirects=1
          &prop=extracts
          &exintro=
          &explaintext=
          &indexpageids=
          &titles=
```

For $html$ parser, we introduced jsoup (it is a Java library for working with real-world HTML [22]), to get the basic information we need, such as $title$, and some other metadata. In this project, I am also using $og : description$ (one of the open graph meta tags [30]) from the HTML source if it exist.

3.6.3 OBJ Model

A simple and common OBJ format model can be loaded as an extra model for the placemarker. OBJ model can be generated by Blender [2]. A simple OBJ parser is created only support v (vertex indices), vn (vertex normals), fv (face vertex), fvn (face vertex normals), and MTL syntax is ignored [33].

3.7 Information Display

A textfield is a rectangle vertex based renderable component to display text on a flat plane. Since it is a GL scene, the actual text will be drawn as a texture. By a constant width and native `android.text.StaticLayout` support, the height of the texture can be calculated.

A menu contains multi-textfield can be seen as an empty textfield based which texture is fill-full a pure background color, and several textfields are laid out on the top of it with a certain vertical dimension.

A head rotation matrix (quaternion matrix [37]) is required for locating object in front of camera [24].

3.8 Camera Movement

In general, there are two sensors can be useful to manage camera movement: ACCELEROMETER (API level 3), LINEAR_ACCELERATION (API level 9) and STEP_DETECTOR (API level 19).

LINEAR_ACCELERATION is same as ACCELERATION which measures the acceleration force in meter per second repeatedly, except linear acceleration sensor is a synthetic sensor with gravity filtered out.

$$
\begin{aligned}
&LinearAcceleration = AccelerometerData - Gravity \\
&v = \int a\,dt \\
&x = \int v\,dt
\end{aligned}
$$

First of all, we take the accelerometer data and remove gravity that is called gravity compensation, whatever is left is linear movement. Then we have to integrate it one to get velocity, integrated again to get the position, which is called double integral. Now if the first integral creates drift, double integrals are really nasty that they create horrible drift. Because of this noise, using acceleration data it isn't so accurate, it is really hard to do any kind of linear movement [8].

On the other hand, use step counter from STEP_DETECTOR, and pedometer algorithm for pedestrian navigation, that in fact works very well for this project.

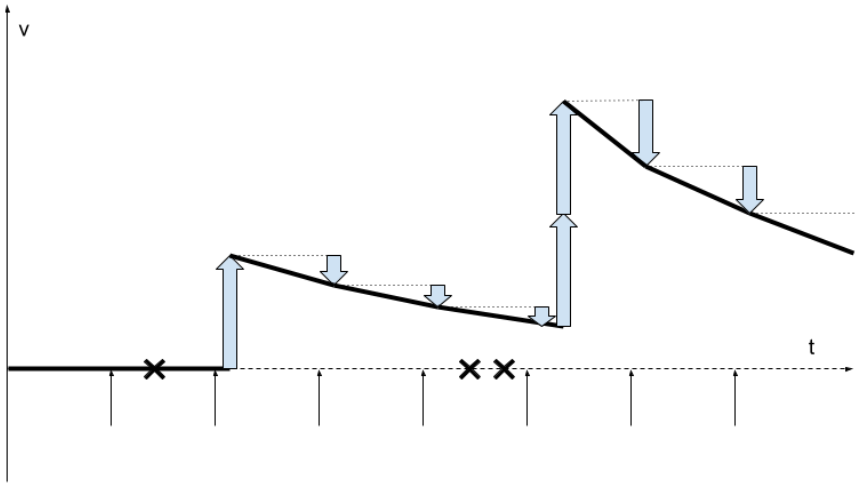
$$
\begin{aligned}
p_1 &= p_0 + v_0 \times dt \\
v_1 &= v_0 + a \times dt
\end{aligned}
$$

The accuracy of this depends on how precision we can get for changing velocity. Considering that velocity is made of 3-axis directions, the current heading direction is required for a correct velocity calculation. Since the frame life cycle is implemented based on [13], which provide the heading direction in each frame callback. So I collect everything I need from the last frame to new frame and update both velocity and position for each new frame.

First of all, damping is required. I reduce velocity by a percentage. It is simply for avoiding that camera taking too long to stop. Damping by percentage can stable and stop the camera in a certain of time that won't be affected by the current camera speed.

Secondly, a constant value in head forwarding direction is been used as a pulse for each step. Because a step is happening instantaneously which implies $a\,dt$ made by each step is actually can be replaced by a constant value.

FIGURE 3.14: Camera movement



For each new frame:

$$\begin{aligned}
\overrightarrow{V_0} &= \overrightarrow{V_0} \cdot Damping \\
\overrightarrow{P_1} &= \overrightarrow{P_0} + \overrightarrow{V_0} \cdot dt \\
\overrightarrow{V_1} &= \overrightarrow{V_0} + \overrightarrow{Forwarding} \cdot Pulse \cdot Steps \\
Damping &\in [0,\ 1] \\
Pulse &\in [0,\ \infty)
\end{aligned}$$

3.9 Ray Intersection

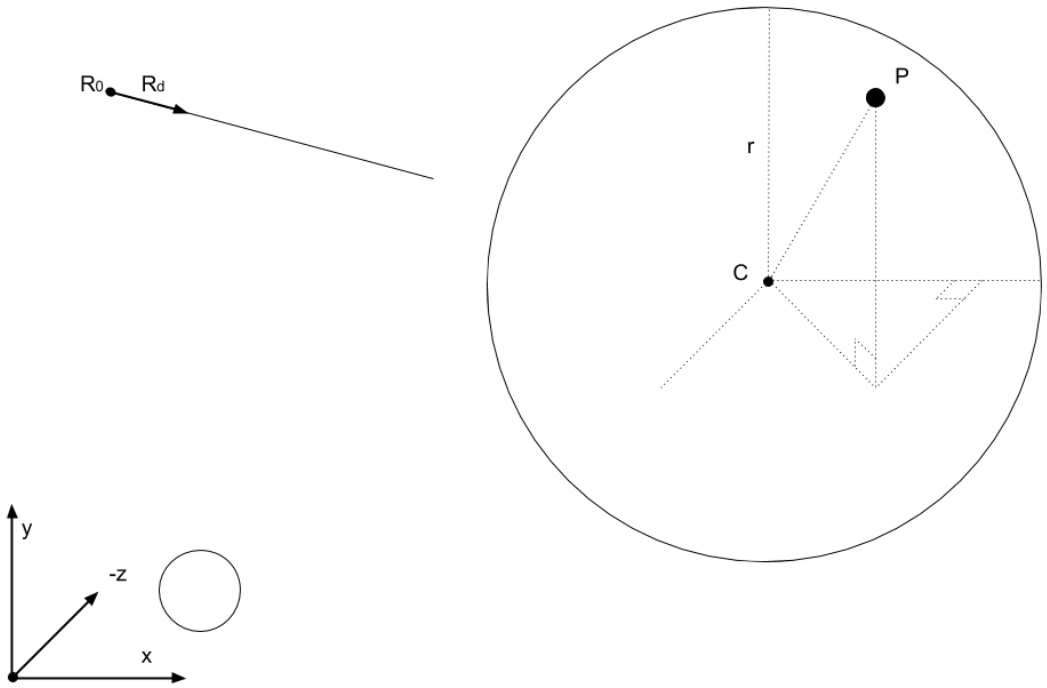
Detect collisions between ray and models are the key to allowing user selecting objects in the VR would, which is one of the important experience for user interaction.

A ray can be describe in a equation with known ray start position $\overrightarrow{R_0}$ and ray direction $\overrightarrow{R_d}$.

$$\overrightarrow{R(t)} = \overrightarrow{R_0} + \overrightarrow{R_d} \cdot t \tag{3.2}$$

3.9.1 Ray-Sphere

FIGURE 3.15: Ray-Sphere intersection



A point P on the surface of sphere should match the equation:

$$(x_p - x_c)^2 + (y_p - y_c)^2 + (z_p - z_c)^2 = r^2 \tag{3.3}$$

If the ray intersects with the sphere at any position P must match the equation 3.2 and 3.3. Therefore the solution of t in the cointegrate equation implies whether or not the ray will intersect with the sphere:

$$\begin{aligned}
&(x_{R_0} + x_{R_d} \cdot t - x_c)^2 + (y_{R_0} + y_{R_d} \cdot t - y_c)^2 + (z_{R_0} + z_{R_d} \cdot t - z_c)^2 = r^2 \\
&\qquad\qquad \vdots \\
&x_{R_d}^2\, t^2 + (2\, x_{R_d}\, (x_{R_0} - x_c))\, t + (x_{R_0}^2 - 2\, x_{R_0}\, x_c + x_c^2) \\
&+ y_{R_d}^2\, t^2 + (2\, y_{R_d}\, (y_{R_0} - y_c))\, t + (y_{R_0}^2 - 2\, y_{R_0}\, y_c + y_c^2) \\
&+ z_{R_d}^2\, t^2 + (2\, z_{R_d}\, (z_{R_0} - z_c))\, t + (z_{R_0}^2 - 2\, z_{R_0}\, z_c + z_c^2) = r^2
\end{aligned}$$

It can be seen as a quadratic formula:

$$a\,t^2 + b\,t + c = 0 \tag{3.4}$$

At this point, we are able to solved the t:

$$t = \begin{cases} \dfrac{-b \pm \sqrt{b^2 - 4\,a\,c}}{2\,a} & \text{if } \; b^2 - 4\,a\,c > 0 \\ \dfrac{-b}{2\,a} & \text{if } \; b^2 - 4\,a\,c = 0 \\ \varnothing & \text{if } \; b^2 - 4\,a\,c < 0 \end{cases}$$

Then, I take a further step to get rid of formula complexity.

$\because$ Equation 3.3, 3.4

$$\begin{aligned} a &= x_{R_d}^2 + y_{R_d}^2 + z_{R_d}^2 \\ b &= 2\,(x_{R_d}\,(x_{R_0} - x_c) + y_{R_d}\,(y_{R_0} - y_c) + z_{R_d}\,(z_{R_0} - z_c)) \\ c &= (x_{R_0} - x_c)^2 + (y_{R_0} - y_c)^2 + (z_{R_0} - z_c)^2 - r^2 \end{aligned}$$

&

$$\begin{aligned} |\overrightarrow{R_d}| &= \sqrt{x_{R_d}^2 + y_{R_d}^2 + z_{R_d}^2} = 1 \\ \overrightarrow{V_{c\_R_0}} &= \overrightarrow{R_0} - \overrightarrow{C} = \overrightarrow{(x_{R_0} - x_c,\;\; y_{R_0} - y_c,\;\; z_{R_0} - z_c)} \end{aligned}$$

$\therefore$

$$\begin{aligned} a &= 1 \\ b &= 2 \cdot \overrightarrow{R_d} \cdot \overrightarrow{V_{c\_R_0}} \\ c &= \overrightarrow{V_{c\_R_0}} \cdot \overrightarrow{V_{c\_R_0}} \cdot r^2 \end{aligned}$$

$\because$ The formula for t can also be optimized

$$\begin{aligned} &\frac{-b \pm \sqrt{b^2 - 4\,a\,c}}{2\,a} = -\alpha \pm \sqrt{\beta} \\ &\alpha = \frac{1}{2}\,b \\ &\beta = \alpha^2 - c \end{aligned}$$

$\therefore$ The final solution for t

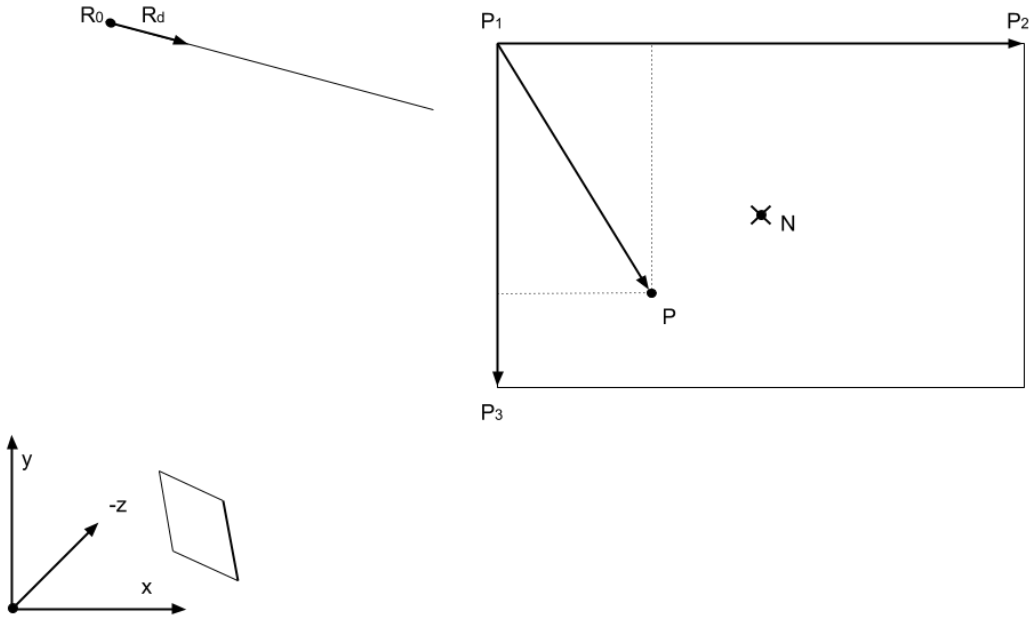
$$t = \begin{cases} -\alpha \pm \sqrt{\beta} & \text{if } \; \beta > 0 \\ -\alpha & \text{if } \; \beta = 0 \\ \varnothing & \text{if } \; \beta < 0 \end{cases}$$

And the collision position for each t is:

$$\overrightarrow{P} = \overrightarrow{R_0} + \overrightarrow{R_d} \cdot t$$

3.9.2 Ray-Plane

FIGURE 3.16: Ray-Plane intersection



If a point *P* on the plane and also belongs to the ray, we have quadric equation:

$$\begin{aligned} &(\overrightarrow{P} - \overrightarrow{P_1}) \cdot \overrightarrow{N} = 0 \\ &\overrightarrow{P} = \overrightarrow{R_0} + \overrightarrow{R_d} \cdot t \end{aligned} \tag{3.5}$$

Solution for the *t* is:

$$t = \begin{cases} \dfrac{-\overrightarrow{N} \cdot (\overrightarrow{R_0} - \overrightarrow{P_1})}{\overrightarrow{N} \cdot \overrightarrow{R_d}} & \text{if } \overrightarrow{N} \cdot \overrightarrow{R_d} \nsim 0 \\ \varnothing & \text{if } \overrightarrow{N} \cdot \overrightarrow{R_d} \sim 0 \end{cases}$$

At last, we have to verify if the collision is inside of the quadrangle by putting *t* back to 3.5, [36] the *t* is valid only if:

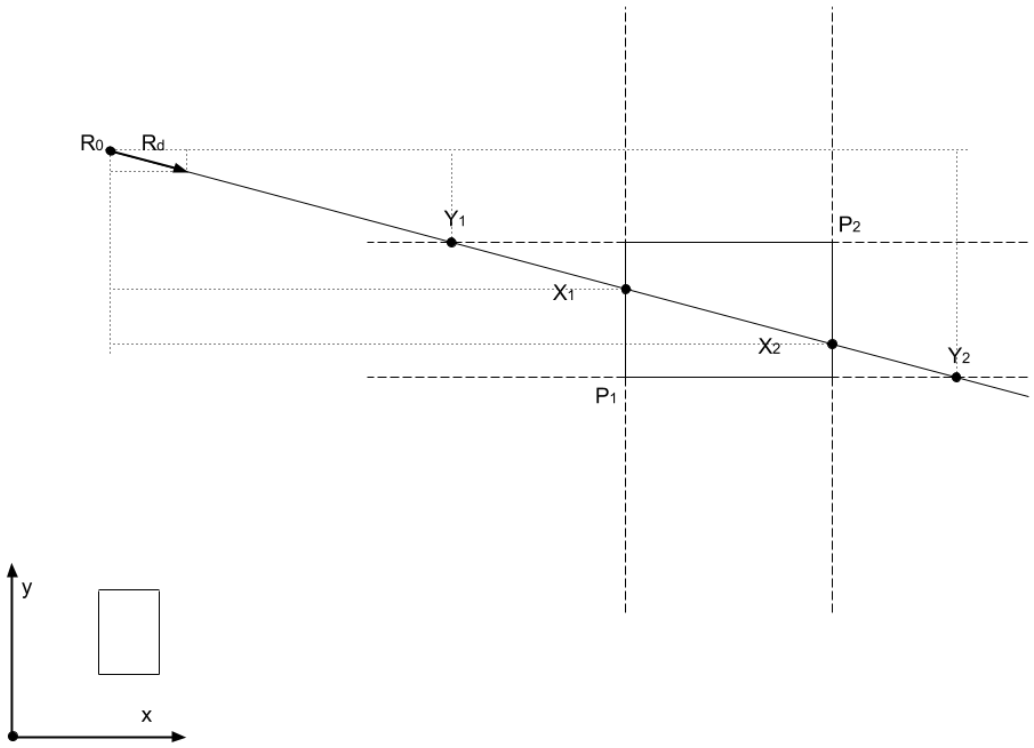
$$\mu = \sqrt{(\overrightarrow{P} - \overrightarrow{P_1}) \cdot (\overrightarrow{P_2} - \overrightarrow{P_1})} \in [0,\ |\overrightarrow{P_2} - \overrightarrow{P_1}|]$$
$$\nu = \sqrt{(\overrightarrow{P} - \overrightarrow{P_1}) \cdot (\overrightarrow{P_3} - \overrightarrow{P_1})} \in [0,\ |\overrightarrow{P_3} - \overrightarrow{P_1}|]$$

3.9.3 Ray-Box

There is an octree implementation 3.4.2 in the VR 3D world that separates the 3D world to invisible 3D boxes that each box contains a certain number of other models. It is to avoid unnecessary ray-object collision detection. In this section, I am going to first explain Ray-Box-2D collision detection [1], then derive out Ray-Box-3D intersection.

Ray-Box 2D

FIGURE 3.17: Ray-Box 2D intersection



$\because$ Known $R_0,\ R_d,\ P_1,\ P_2$

$$X_1 = \begin{cases} x_{P_1} - x_{R_0} & \text{if } x_{R_d} > 0 \\ x_{P_2} - x_{R_0} & \text{if } x_{R_d} < 0 \end{cases} \qquad X_2 = \begin{cases} x_{P_2} - x_{R_0} & \text{if } x_{R_d} > 0 \\ x_{P_1} - x_{R_0} & \text{if } x_{R_d} < 0 \end{cases}$$

$$t_{X_1} = \frac{X_1}{x_{R_d}}$$
$$t_{X_2} = \frac{X_2}{x_{R_d}}$$

$$Y_1 = \begin{cases} y_{P_1} - y_{R_0} & \text{if } y_{R_d} > 0 \\ y_{P_2} - y_{R_0} & \text{if } y_{R_d} < 0 \end{cases}$$

$$Y_2 = \begin{cases} y_{P_2} - y_{R_0} & \text{if } y_{R_d} > 0 \\ y_{P_1} - y_{R_0} & \text{if } y_{R_d} < 0 \end{cases}$$

$$t_{Y_1} = \frac{Y_1}{y_{R_d}}$$
$$t_{Y_2} = \frac{Y_2}{y_{R_d}}$$

& When collision happens, we have formula:

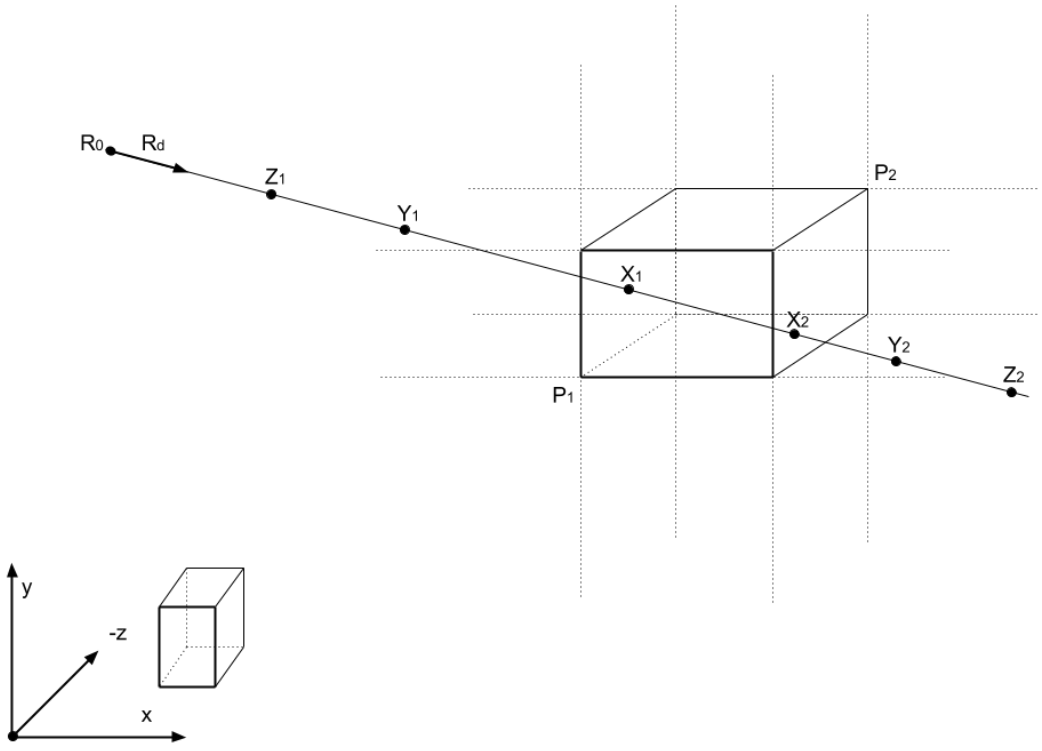
$$t_{X_1} < t_{X_2}$$
$$t_{Y_1} < t_{Y_2}$$

∴ Which is

$$max(t_{X_1},\ t_{Y_1}) < min(t_{X_2},\ t_{Y_2}) \tag{3.6}$$

Ray-Box 3D

FIGURE 3.18: Ray-Box 3D intersection



$\because$ Known $R_0,\ R_d,\ P_1,\ P_2$

$$X_1 = \begin{cases} x_{P_1} - x_{R_0} & \text{if } x_{R_d} > 0 \\ x_{P_2} - x_{R_0} & \text{if } x_{R_d} < 0 \end{cases} \qquad Y_1 = \begin{cases} y_{P_1} - y_{R_0} & \text{if } y_{R_d} > 0 \\ y_{P_2} - y_{R_0} & \text{if } y_{R_d} < 0 \end{cases}$$

$$X_2 = \begin{cases} x_{P_2} - x_{R_0} & \text{if } x_{R_d} > 0 \\ x_{P_1} - x_{R_0} & \text{if } x_{R_d} < 0 \end{cases} \qquad Y_2 = \begin{cases} y_{P_2} - y_{R_0} & \text{if } y_{R_d} > 0 \\ y_{P_1} - y_{R_0} & \text{if } y_{R_d} < 0 \end{cases}$$

$$t_{X_1} = \frac{X_1}{x_{R_d}} \qquad t_{Y_1} = \frac{Y_1}{y_{R_d}}$$
$$t_{X_2} = \frac{X_2}{x_{R_d}} \qquad t_{Y_2} = \frac{Y_2}{y_{R_d}}$$

$$Z_1 = \begin{cases} z_{P_1} - z_{R_0} & \text{if } z_{R_d} > 0 \\ z_{P_2} - z_{R_0} & \text{if } z_{R_d} < 0 \end{cases}$$

$$Z_2 = \begin{cases} z_{P_2} - z_{R_0} & \text{if } z_{R_d} > 0 \\ z_{P_1} - z_{R_0} & \text{if } z_{R_d} < 0 \end{cases}$$

$$t_{Z_1} = \frac{Z_1}{z_{R_d}}$$
$$t_{Z_2} = \frac{Z_2}{z_{R_d}}$$

& When collision happens, we have formula:

$$\begin{cases} t_{X_1} < t_{X_2} \\ t_{Y_1} < t_{Y_2} \\ t_{Z_1} < t_{Z_2} \end{cases}$$

$\therefore$ Which is

$$max(t_{X_1},\ t_{Y_1},\ t_{Z_1}) < min(t_{X_2},\ t_{Y_2},\ t_{Z_2}) \tag{3.7}$$

4 Discussion

Our approach to explore geographic data visualization with immersive virtual reality is to actually develop one. We take use of Google Cardboard for turning an Android phone into an immersive virtual reality device. A virtual reality-specific application is developed to import KML format geographic data source and display them in the application. The user is able to make a six degrees of freedom (DOF) - position coordinates (x, y and z offsets) and orientation (yaw, pitch and roll angles) - movement. User is able to do simple interactions: selecting a placemark; viewing the information of the placemark on a popup message board; displaying a customized 3D model (eg: OBJ model) of the placemark; or any further information from a URL (which could be a image or a piece of summarized information extracted from Wikipedia or any HTML text).

By comparison to virtual globes, geographic data visualization with immersive virtual reality device allows to do similar things, but it has more easy used intuitive interface. Firstly, they can share with geographic data that created by a universal markup language (eg: KML), which means almost every data based feature in the existing virtual globe can also migration to the immersive virtual reality application. Secondly, they are both able to have a remote server database that provides synchronous data, such as a server processes the requests and delivers the result in a standard web format back to the client.

There are five human senses provide the information and passed to our brain for capturing our attention: sight (70%), hearing (20%), smell (5%), touch (4%), and taste (1%) [25]. The immersive virtual reality has certainly improved the feedback of sight sense, and also by given the existing Spatial Audio technology (such as [16]), it is able to use a spatial audio as a simultaneous response from the user for "fooling" the hearing sense.

Sensor fusion creates a huge drift during the nasty double integration process, we alternatively using the Step sensor (pedometer) as the pedestrian navigation (it is not the most logical way). It allows move forward in the current heading direction. Nonetheless, it doing very well for navigating through all scene that satisfies our application purpose.

The performance of this immersive virtual reality application is good (55 - 60 FPS) when the there is less than 250 placemarks exist in the scene. Although, we have an Octree based object intersection algorithm avoid most of the invalid recursive detection, and optimized matrix reconstitution, but there is a performance limitation of actual OpenGL ES native call in the Android SDK. To solve this issues require further investigation for reducing the times of OpenGL

ES render call for each frame.

Also, there is a limitation of gesture recognition and perception technology that suppress the development of immersive virtual reality technology.

Due to the time, and geographic data resource limitation, this project is simply developed with some unfinished features, which are very important as a geographic visualization tool, but it not particularly critical for an exploring purpose.

A key requirement for environmental scientists is to be able to visualize four-dimensional data (i.e. time-dependent three-dimensional data). Indeed, we are able to visualize the environment data from the data file which created from the different period of time. We also can do a fake real-time data visualization by a certain frequently refreshing rate on both client and server, but there is a limitation on both client (performance) and server (data creation), none of them make any sense. However, an implementation of dynamic graphic animation would be excellent for improving user understanding of any environmental data visualization. That is to say, an animation transform from one piece of time-dependent data visualization to another.

One of the main features of geographic data visualization is the Level Of Detail (LOD) rendering based on distance from the viewpoint. Textures of the virtual reality environment should be separately prepared, and attached as the circumstances may require. They are updatable and detailed on different levels. It can also provide a solution for visualizing a large amount of overlapping data.

Most of the geographic data markup language (eg: KML) supports multiple layers in a single file so that we need a layer switch, and more geometric shapes supporting, such as lines and polygons. Under the LOD implementation, we are able to see the architectural structure or plan if we are close enough.

There is always room for improvement, especially in the immersive virtual reality visualization, when something related to human intuitive nature system, because the feeling will be always not real enough compare to the real world interaction.

5 Conclusion

The immersive virtual reality provides a highly integrated easy-to-use, intuitive real-time 3D GIS for geographic data visualization. Due to the limitation of human-machine interaction, the VR is not yet able to do everything that the pseudo-3D virtual globes can do, but it has the potential to do more than people expected when there is a revolution for gesture recognition and perception.

There are at least three reasons indicate that Android Phones are extremely suitable to use as an immersive virtual reality device. First, always about the money, 15$ Google Cardboard kit turns Android or iOS smartphone to immersive virtual reality device; second, the existing VR specific open source SDK provided by Google, includes necessary graphic, and spatial audio development; third, Android includes support for high-performance graphics with OpenGL.

The most logical way to calculate the movement in the immersive virtual reality environment is to use Gyroscope to measures angular velocity relative to the body, or in other words, to get the device orientation. Then, using Accelerometer to inject the correction term that keeps the orientation correct with respect to gravity, and a correction due to the magnetic north from Compasses is also required. However, it is really hard to get an accurate position out of them due to a horrible drift comes from the nasty double integration process. Alternatively, a pedestrian navigation was implemented for this project is based on the Step sensor (pedometer) with a certain algorithm to calculate the velocity was turn out working very well. The limitation of this approach is can only move forward in the current heading direction.

KML is not only a human-readable markup language can and very suit for visualizing geographic data, but also it has very well compatibility with current major virtual globes, such as the well-known Google Earth. Moreover it also powerful enough to describe a small subregion, such as a building hierarchical plan. On the other hand, immersive virtual reality also can be used as a tool that able to visualizing different sort of data, or natural system by integrating another or new spatial markup language.

A Source

Related source repository:

https://github.com/jiangyang5157/virtual-reality

https://github.com/jiangyang5157/tookit

https://github.com/jiangyang5157/vr-server

https://github.com/jiangyang5157/massey-master-thesis-2016

Bibliography

[1] T. Barnes. (2011). Fast, branchless ray/bounding box intersections, [Online]. Available: https://tavianator.com/fast-branchless-raybounding-box-intersections.

[2] Blender. (2016). Open source 3d creation, [Online]. Available: https://www.blender.org/.

[3] J. D. Blower, A. Gemmell, K. Haines, P. Kirsch, N. Cunningham, A. Fleming, and R. Lowry, "Sharing and visualizing environmental data using virtual globes", 2007.

[4] D. Butler, "Virtual globes: The web-wide world", *Nature*, vol. 439, no. 7078, pp. 776–778, 2006.

[5] T. Danova. (2015). The global smartphone market report, [Online]. Available: http://www.businessinsider.com.au/global-smartphone-market-forecast-vendor-platform-growth-2015-6?r=US&IR=T.

[6] Earthslot. (2016). Earth maps & world geography, [Online]. Available: http://www.earthslot.org/.

[7] Esri. (2016). Explorer for arcgis, [Online]. Available: http://www.esri.com/software/arcgis/explorer.

[8] Google. (2010). Sensor fusion on android devices: A revolution in motion processing, [Online]. Available: https://www.youtube.com/watch?v=C7JQ7Rpwn2k&feature=youtu.be&t=23m21s.

[9] ——, (2012). Go at google: Language design in the service of software engineering, [Online]. Available: https://talks.golang.org/2012/splash.article.

[10] ——, (2016). Aar format, [Online]. Available: http://tools.android.com/tech-docs/new-build-system/aar-format.

[11] ——, (2016). Android maps utils, [Online]. Available: https://github.com/googlemaps/android-maps-utils/tree/master/library/src/com/google/maps/android/kml.

[12] ——, (2016). Google cardboard, [Online]. Available: https://vr.google.com/cardboard/.

[13] ——, (2016). Google vr sdk for android, [Online]. Available: https://developers.google.com/vr/android/.

[14] ——, (2016). Keyhole markup language, [Online]. Available: https://developers.google.com/kml/.

[15] ——, (2016). Opengl es, [Online]. Available: https://developer.android.com/guide/topics/graphics/opengl.html.

[16] ——, (2016). Spatial audio, [Online]. Available: https://developers.google.com/vr/concepts/spatial-audio.

[17] ——, (2016). The go programming language, [Online]. Available: https://golang.org/.

[18] ——, (2016). Transmitting network data using volley, [Online]. Available: https://developer.android.com/training/volley/index.html.

[19] B. Huang and H. Lin, "A java/cgi approach to developing a geographic virtual reality toolkit on the internet", *Computers & Geosciences*, vol. 28, no. 1, pp. 13–19, 2002.

[20] IDC. (2016). Smartphone os market share, [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[21] A. Imbert. (2016). Go json rest, [Online]. Available: https://github.com/ant0ine/go-json-rest.

[22] Jsoup. (2016). Jsoup: Java html parser, [Online]. Available: https://jsoup.org/.

[23] D. Koller, P. Lindstrom, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner, "Virtual gis: A real-time 3d geographic information system", in *Proceedings of the 6th conference on Visualization'95*, IEEE Computer Society, 1995, p. 94.

[24] Mathworks. (2016). Quaternion rotation, [Online]. Available: http://au.mathworks.com/help/aeroblks/quaternionrotation.html.

[25] T. Mazuryk and M. Gervautz, "Virtual reality-history, applications, technology and future", 1996.

[26] NASA. (2016). Nasa world wind, [Online]. Available: https://worldwind.arc.nasa.gov/.

[27] I. Nourbakhsh, R. Sargent, A. Wright, K. Cramer, B. McClendon, and M. Jones, "Mapping disaster zones", *Nature*, vol. 439, no. 7078, pp. 787–788, 2006.

[28] A. Nuernberger. (2006). Virtual globes in the classroom, [Online]. Available: http://www.csiss.org/SPACE/resources/virtual-globes.php#GISdata.

[29] OGC. (2016). Open geospatial consortium, [Online]. Available: http://www.opengeospatial.org/.

[30] OGP. (2014). The open graph protocol, [Online]. Available: http://ogp.me/.

[31] T. M. Rhyne, "Going virtual with geographic information and scientific visualization", *Computers & Geosciences*, vol. 23, no. 4, pp. 489–491, 1997.

[32] T. M. Rhyne, W. Ivey, L. Knapp, P. Kochevar, and T. Mace, "Visualization and geographic information system integration: What are the needs and the requirements, if any?", in *Visualization, 1994., Visualization'94, Proceedings., IEEE Conference on*, IEEE, 1994, pp. 400–403.

[33] H. W. Shen. (2011). Guidance to write a parser for .obj and mtl file, [Online]. Available: http://web.cse.ohio-state.edu/~hwshen/581/Site/Lab3_files/Labhelp_Obj_parser.htm.

[34] B. T. Tuttle, S. Anderson, and R. Huff, "Virtual globes: An overview of their history, uses, and future challenges", *Geography Compass*, vol. 2, no. 5, pp. 1478–1505, 2008.

[35] Ublox. (1999). Datum transformations of gps positions, [Online]. Available: http://www.nalresearch.com/files/Standard%20Modems/A3LA-XG/A3LA-XG%

20SW%20Version%201.0.0/GPS%20Technical%20Documents/GPS.G1-X-00006%20(Datum%20Transformations).pdf.

[36] User3146587. (2014). 3d ray-quad intersection, [Online]. Available: http://stackoverflow.com/questions/21114796/3d-ray-quad-intersection-test-in-java.

[37] J. V. Verth. (2013). Understanding quaternions, [Online]. Available: http://www.essentialmath.com/GDC2013/GDC13_quaternions_final.pdf.

[38] Wikipedia. (2006). Icosahedron golden rectangles, [Online]. Available: https://commons.wikimedia.org/wiki/File:Icosahedron-golden-rectangles.svg.

[39] ——, (2016). Api, [Online]. Available: https://www.mediawiki.org/wiki/API:Opensearch.

[40] ——, (2016). Ecef, [Online]. Available: https://en.wikipedia.org/wiki/ECEF.

[41] ——, (2016). Geographic coordinate system, [Online]. Available: https://en.wikipedia.org/wiki/Geographic_coordinate_system.

[42] ——, (2016). Geographic information system, [Online]. Available: https://en.wikipedia.org/wiki/Geographic_information_system.

[43] ——, (2016). Vrml, [Online]. Available: https://en.wikipedia.org/wiki/VRML.

[44] ——, (2016). World wide web, [Online]. Available: https://en.wikipedia.org/wiki/World_Wide_Web.

[45] ——, (2016). X3d, [Online]. Available: https://en.wikipedia.org/wiki/X3D.