INTERACTIVE BRIEFING MAP ON RESPONSIVE WORKBENCH

by

Muhammed Kalkan

B.S., Computer Engineering, Istanbul Technical University, 2007

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2010

ACKNOWLEDGEMENTS

First of all I would like to thank my thesis supervisor Dr. Ali Vahit Şahiner with gratitude; for his support, significant comments and his guidance throughout this thesis study.

I would like to thank TÜBİTAK for time and material support required for this thesis study.

I am also thankful for Ali Tozan and Onur İnce for sharing their ideas and providing their support during my thesis study.

Finally, I would like to thank Semra Şenel and the rest of my family for their support and encouragement during and for my academic studies.

ABSTRACT

INTERACTIVE BRIEFING MAP ON RESPONSIVE WORKBENCH

This thesis presents a terrain visualization application developed on a responsive workbench system. The application enables stereo viewing of a terrain as if a physical mock up model of it is actually on a real table. Head tracking system allows the user to move around the terrain model.

The terrain visualization system developed has multi-texturing and level of detail capabilities. It also has tools that extracts culture feature data and displays it. Application also has a tool that performs visibility analysis on terrain. A remote controller is used to control system to give user more flexibility.

Effect of various stereo visualization parameters like eye separation is analyzed as part of evaluation in system usability.

ÖZET

ETKİLEŞİMLİ ÇALIŞMA TEZGAHI ÜZERİNDE BRİFİNG HARİTA UYGULAMASI

Bu tez etkileşimli çalışma tezgahı üzeride geliştirilmiş bir arazi görüntüleme uygulamasını ortaya koymaktadır. Uygulama arazinin fiziksel bir maketinin gerçek bir masa üzerinde bulunuyormuşçasına stereo olarak görüntülenmesini sağlamaktadır. Kafa takip sistemi kullanıcının arazi modeli etrafında hareket etmesine izin vermektedir.

Geliştirilen arazi görüntüleme sistemi çoklu kaplama ve detay seviyesi yeteneklerine sahiptir. Bu sistem aynı zamanda kültür özellik verisini çıkartan ve görüntüleyen gereçlere sahiptir. Uygulama aynı zamanda arazi üzerinde görülebilirlik analizi yürüten bir gerece de sahiptir. Sistemin kontrol edilmesi ve kullanıcıya daha fazla esneklik vermek için bir uzaktan kumanda kullanılmıştır.

Göz aralığı gibi çeşitli stereo görüntüleme parametreleri sistemin kullanılabilirlik değerlendirmesinin bir parçası olarak analiz edilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

RWB	Responsive Workbench
2D	Two Dimensions
3D	Three Dimensions
4D	Four Dimensions
5D	Five Dimensions
IB Map	Interactive Briefing Map
GIS	Geographical Information System
VR	Virtual Reality
AR	Augmented Reality
LOD	Level of Detail
DLOD	Discrete LOD
CLOD	Continuous LOD
LOS	Line of Sight
ROAM	Real-time Optimally Adapting Meshes
TUIO	Table-Top User Interfaces Objects
RWB	Responsive Workbench
TV	Terrain Visualization
VRML	Virtual Reality Modeling Language
DEM	Digital Elevation Model
DTED	Digital Terrain Elevation Data
USGS	United States Geological Survey
RAW	Raw Image Format
GCP	Ground Control Point
TIFF	Tagged Image File Format

GeoTIFF	Geographical TIFF
SHP	Shape File
DOF	Degrees of Freedom
VE	Virtual Environment
OS	Operating System
CABTT	Cached Aggregated Binary Triangle Trees
ROAM	Real-time Optimally Adapting Meshes
Wiimote	Nintendo Wii Remote Controller
API	Application Protocol Interface
OpenGL	Open Graphics Library
FOV	Field of View
HFOV	Horizontal Field of View
VFOV	Vertical Field of View

1. INTRODUCTION

Terrain is the term used to refer to the land surface of a geographical environment. An external terrain model is a uniform square grid of two attributes, the elevation and the color. It is visualized as a textured polygonal mesh. Visualization of terrain data as real – time display of multi resolution data significantly improves data access and quality of representation. Terrain visualization enables effective exploration of data. Without it, one can only carry out peripheral analysis.

In this work, a terrain visualization system called Interactive Briefing Map has been developed on a "Responsive Workbench" which is a head tracked stereo visualization environment. This approach enables a terrain to be visualized on a real table as if a physical mock up model of it is actually being viewed.

Interactive Briefing Map application is implemented using a software infrastructure which is developed as part of the thesis work. In order to improve real-time behavior, a level of detail algorithm is implemented. Interactive Briefing Map has a flexible architecture that allows visualization line of sight and terrain features other than geometry. System has 6 degrees of freedom input / output device to allow user interaction.

1.1. Outline

The organization of thesis is as follows: Chapter 2 gives information on terrain visualization, its areas of usage, terrain visualization basics, methods and data sets. Literature survey about responsive workbench systems is explained in chapter 3. Also stereoscopic visualization is deeply explained in this chapter. Chapter 4 explains visualization workshop responsive workbench system setup and its elements stationed in visualization laboratory. Chapter 5 provides the details of IB MAP application and the software organization. Chapter 6 gives the results and evaluations. Finally Chapter 7 presents conclusion and future work for the thesis.

2. TERRAIN VISUALIZATION

Digital terrain data has replaced old methods of cartography which used to employ handmade mockup or drawn maps. The rapid producing of digital terrain data using satellites offers significant cost reduction for cartographic operations and it provides a wide variety of data interactions and reducing the operation time [1].

Digital terrain data consists of height values for each predefined resolution. Data in the file is stored as uniform square grid of points. A data file captured with 3 meters resolution means, each height value in the file is sampled by 3 meters apart from each other on ground. Although there are many varieties of Digital Elevation Model (DEM) files, they all consist of data grid mentioned before. However each DEM file keeps different header data according to their type. Header data keeps information about file type; coordinate location information about stored data, datum information, coordinate system information and specific information like resolution and captivation date.

After the development of computer graphics, digital terrain data processing evolved to 3 Dimensions (3D) to obtain better perception and understanding of the terrain concept. Main purpose of the terrain visualization is to increase perception and understanding of the datasets.

There are many projects that specifically benefit terrain visualization. A battlefield project that uses terrain visualization is conducted to increase user's awareness of the terrain model. Banks *et al*. proposed a battlefield management system that uses terrain visualization. They suggest that system increases the ability to understand terrain and perception to help commander war time judgments according to terrain [2].

Moreover, works that requires good analyzing of terrain uses terrain visualization. For a highway construction process, terrain visualization can be used to increase performance and view the work before it is done [3]. Figure 2.1 shows highway planning by using terrain visualization.

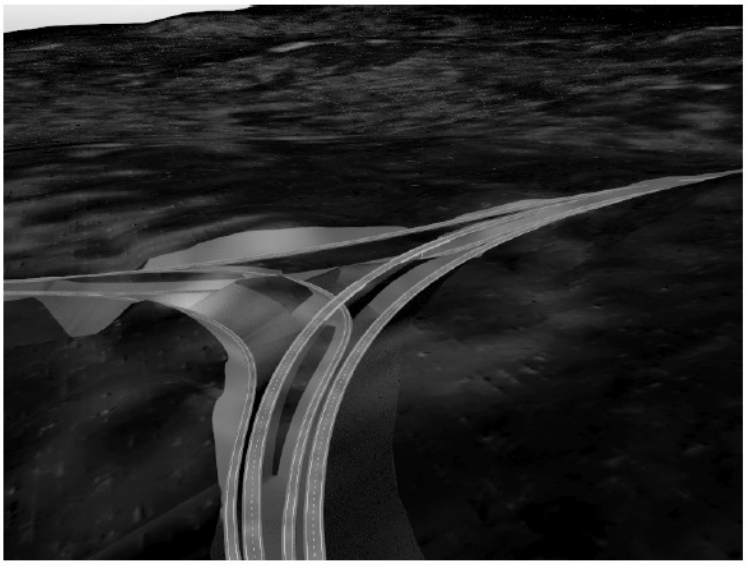


Figure 2.1. Terrain visualization in highway planning [3]

In the subject of terrain visualization, Marinos Kavouras suggests that visual realism can highly increase the phenomena modeled and increase awareness of flaws and inconsistencies [4]. In order to understand terrain dataset, visualizing plays an important role.

Moreover, Davis *et al*. suggests visual perception and real-time display of multi resolution data, results in interactive terrain visualization with significantly improved data access and quality of representation. They also claim that “Visualization is often the first step towards analysis and understanding. We must first "look at" the data before we can even know what to do next. Conversely, if we cannot even visualize the data, we cannot effectively explore and discover. We can only perform peripheral analysis; much of the data remains untouched [5].

Terrain visualization is used in science fields that needs environmental data or environmentally affected data visualizations like geography, city planning or social evaluations which is affected by environment. Training simulations, geographic research applications like population planning, evaluation of vegetation, soil, and road patterns mostly use terrain data [6][7]. Geographical Information Systems (GIS) applications also use terrain data and visualize it as 3D.

Terrain elevation data and terrain visualization is widely used for urban planning. Household analysis, city planning and urban affairs mostly use terrain visualization to increase utilization. Urban planning program by means of terrain visualization is demonstrated in Figure 2.2 [8].

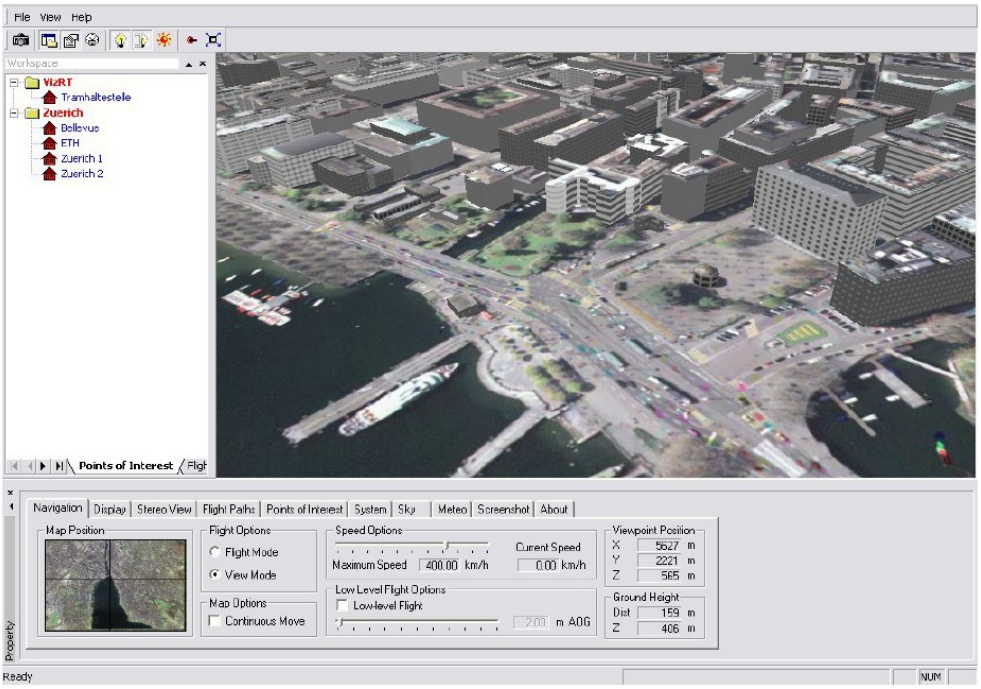


Figure 2.2. Terrain visualization in urban planning [8]

With all its benefits, terrain visualization is one of the most investigated subjects in computer graphics. Terrain visualization or geo-visualization may be described as the intersection of many different fields, including GIS, 3D scientific visualization, computer graphics, human-computer interaction and virtual reality [9].

There is also performance aspect of terrain visualization. In 3D graphics, drawing a terrain can be costly and reduces FPS value. In order to prevent low rendering performance LOD algorithms developed such as ROAM [10].

2.1. Terrain Visualization Applications

Terrain visualization has a wide variety of applications. The visualized terrain can depend on real data or an imaginary one. For the past few decades Terrain Visualization

(TV) is considered with Virtual Reality (VR). Moreover GIS applications' taxonomy gained a new type which is called Virtual Reality GIS. Especially with the development of Virtual Reality Modeling Language and GIS application projects' number increased dramatically [11].

There are many specific applications that uses terrain visualization. In a project called LIFE-SAVER, Nóbrega *et al*. used terrain visualization to simulate a flood disaster and evaluate its results. They pointed out that visually rich environment allows a fast perception of the emergency that is being simulated [12]. Figure 2.3 shows LIFE-SAVER disaster simulation system interface.

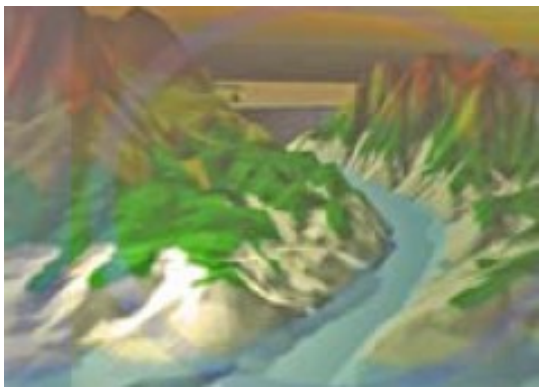


Figure 2.3. Life – Saver interface [12]

Also, Deussen *et al*. used terrain visualization to model plant ecosystems on computer environment. They used terrain data as well as plant ecosystem data to model the environment [13].

Terrain visualization is also used in space missions. Lewis Hitchner from NASA Ames Research Center suggested that they use terrain visualization to increase their exploration experience and guide them for incoming space missions [14].

We can classify terrain visualization usage areas in two groups as simulations and GIS applications. GIS applications focus on analysing of datasets. Simulations,on the other hand, focus on visual representation to increase realism effect. Next sub chapters explain general characteristics of these groups.

Simulation systems are test mechanisms in order to see an effect of a predefined incident without getting in danger or cost reducing. First examples were mechanical simulations. As the computer technology advanced, computer aided simulations took over mechanical ones. As of today, 3D graphical visions, sound effects, motion effects and even wind or scent effects are being delivered to user in modern simulators which are known as 4 Dimensions (4D) or 5 Dimensions (5D) simulators [15].

Most simulators need to run on real world thus needing a terrain environment to be able to show visual data. There are many application types of terrain visualization system in simulators. Space mission simulators are the one of the most important kinds of simulator systems since very high mission costs and dangers [16]. Also all of the vehicle simulators use terrain data with certain algorithms for simulating process. First of all size of the visualized terrain brings obstacles for the real time applications. Frame per second (FPS) rating in simulations and real time applications, is an important part for the system performance. User can give inputs and receive appropriate outputs at right time only if FPS in over an acceptable amount. Usually 30 FPS is considered acceptable for smooth visualization. Lower FPS rates decreases system stability and usability.

GIS applications are the most demanding area of science of TV. GIS applications mainly use datasets to create a virtual model of the targeted environment. As the satellite technology advances and users can reach out satellite data easily, GIS applications evolved to use satellite data. All data that is to be processed in a GIS application have to be in geo-coordinated format and using same coordinate system in order to achieve data consistency. GIS is an area that many science disciplines combined to achieve a goal. Computer graphics, remote sensing, image processing, photogrammetry, geology and database management systems are some of the areas that contribute to GIS. Modern GIS applications have three main components.

First of all, GIS systems need to use digital elevation models to be able to build 3D structure of the terrain. DEMs area generally produced from images taken above the targeted area either by plane or satellites itself. Images taken must be stereographic in order to calculate depth information and as a result build DEM. There are many formats to store DEM data. Digital Terrain Elevation Data (DTED), United States Geological Survey

DEM (USGS DEM), Digital Elevation Data (DED) and Raw (RAW) are some of the formats used to store DEM data.

Second component in GIS is cover imagery. Unlike stereoscopic images to build DEM data, cover imagery is taken to apply image onto the terrain to give area a more realistic look. These images must be processed with some specific applications before using in any GIS application. Images taken above the terrain have some adjustment problems because of curvature of earth. Images are in pure 2D environment but earth itself is not. Therefore some correction processes like orthorectification and ground control points to adjust image. These images have many formats. Most common are Tagged Image File Format (TIFF) and Georeferenced TIFF (GeoTIFF) imagery.

Third component is culture data in GIS applications. Culture data consists of many features. Vegetation, rivers, lakes and roads are some of the types of culture that can be stored. These features generally stored in 3 types such as point, line or area type. File formats for culture has a wide variety. Most common file type for culture information is Shape File (SHP) which is an industry standard.

Figure 2.4, shown below, represents a culture file layout.

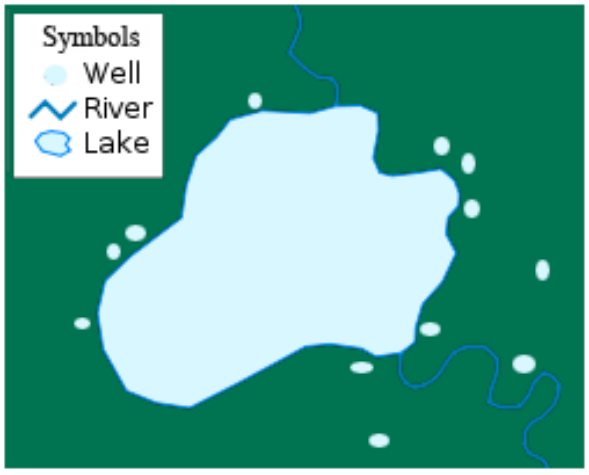


Figure 2.4. A culture file representation

2.2. Terrain Visualization Basics

Terrain visualization has a wide variety of applications as mentioned before. This rich application areas lead terrain visualization to get improvement in some aspects. For example planning a space flight to Mars can be simulated with a planet wide terrain simulation [17]. Planet wide terrain visualization requires a large amount of data to be shown and that needs some adjustments and algorithms to provide real time rendering. Real time rendering is a must for simulations. A good terrain representation must offer good performance and realism to achieve its purpose.

Terrain representation in graphics systems has two main properties. One of them is geometry of the terrain and other is picture data that is covered on terrain geometry. Since modeling terrains in high detail using polygons is very costly, systems designers use picture to detail virtual terrain.

2.2.1. Terrain Data Acquisition

Terrain data is simply divided into two categories. One of them is elevation data and another is image. Elevation data is derived from stereoscopic image pairs that can be either taken from a satellite or an aircraft. Moreover laser scanning, a.k.a Laser Imaging Detection and Ranging (LIDAR), can be used to capture elevation data. If we want to model undersea terrain then there are different techniques to get surface data under the sea. Bathymetry scanning and some sonar systems within ships can be used to extract surface elevation data. Elevation data can be stored as many file types like DTED, DED or DEM. Most of the elevation data includes geographical coordinates of the real world place of the data. Figure 2.5 shows a representation of the elevation extraction process.

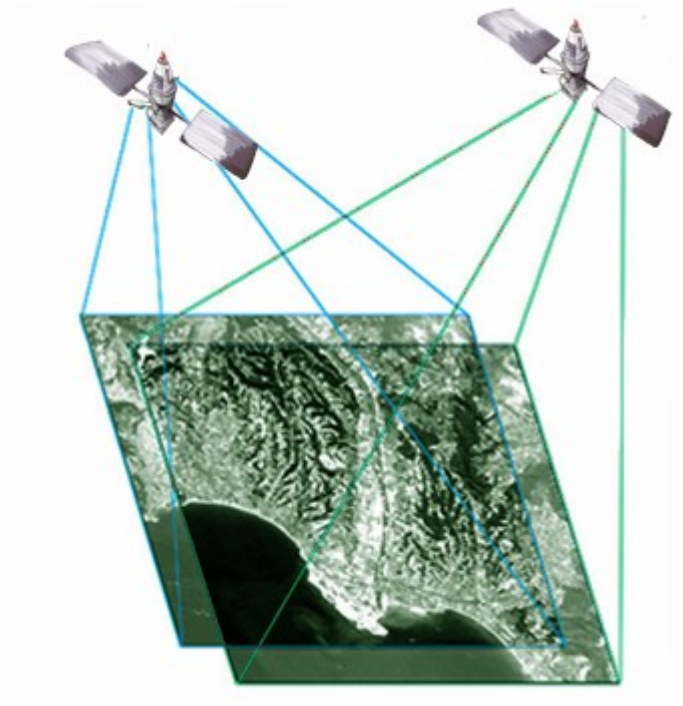


Figure 2.5. Elevation data extraction [18]

Also images can be taken from satellites to be used as terrain textures. In present day, there are commercially available with 0.60 m resolution image data. Resolution can be increased by mixing panchromatic images from satellites with color ones.

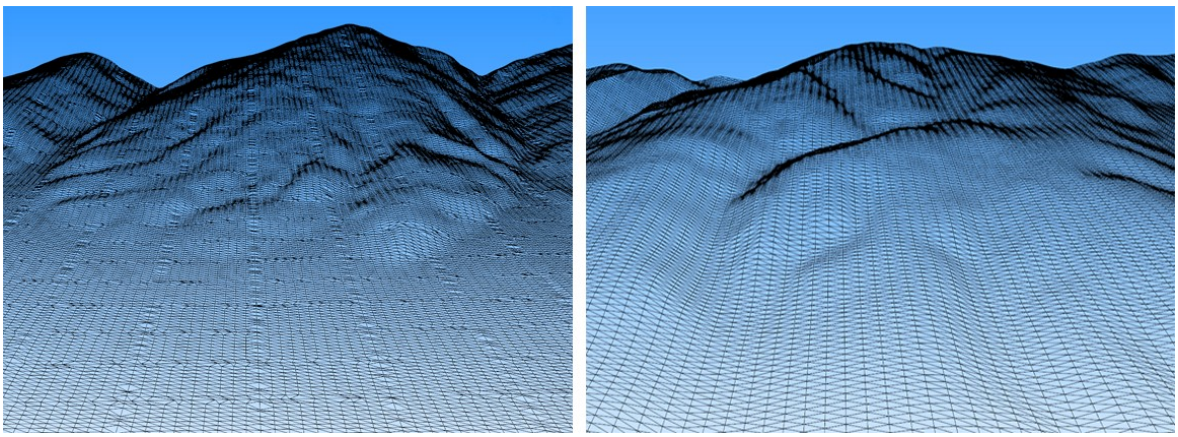
2.2.2. Terrain Representation

Representing a terrain in a computer simulated environment is a key point for terrain visualization. Visualizing terrain contains two main elements. One of them is representing geometry, and the other is detailing this geometry with textures to simulate real world.

Terrain geometry is the most important part of the terrain. In order to simulate a real world terrain in computer environment, first the terrain shape must be similar or identical to the original terrain respect to our use. Terrain geometry is built by using elevation data. Elevation data is a data file that includes height values by a certain resolution. They can be either captured by satellites or aerial photos by planes.

Terrain geometry is represented by polygons. Polygons are primitive component of the graphics system. Terrains are represented by triangles. These triangles can be placed irregularly that is sometimes called TIN, Triangle Irregular Network. This mesh type can represent original terrain more accurate since vertexes are placed according to real height. On the other hand to make drawing simple, regular triangle network can also be used. This however, can be a little bit different than the original data.

Figure 2.6 shows irregular mesh and regular mesh structures. Irregular mesh fits better to the terrain profile on the other hand regular mesh is easy for graphics hardware to draw.

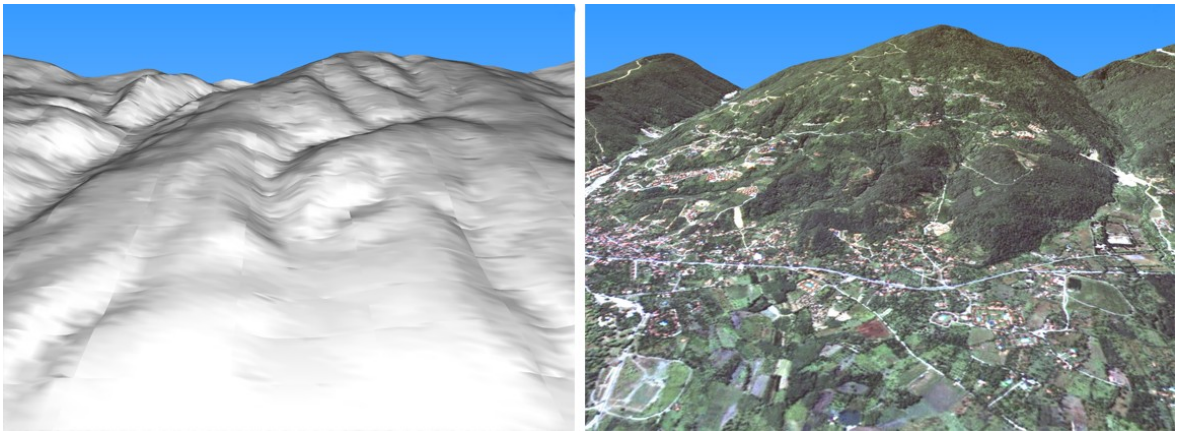


(a) (b)

Figure 2.6. Mesh representations (a) Irregular Mesh, (b) Regular Mesh

Terrain geometry representation with a detailed level is important for performance. And it depends heavily on some detail algorithms to be correct and fast.

After representing terrain geometry, we need to detail it and increase realism because the geometry itself cannot bring. Imagery used for a terrain can be either taken from satellites or aerial crafts (Figure 2.7). For large terrain visualizations with a big scale and detail, texture management and paging algorithms are represented.



(a) (b)

Figure 2.7. Terrain representations (a) Non-textured terrain, (b) textured terrain

2.2.3. Terrain Visualization Performance

The most important problem of terrain visualization is performance. Although computer graphics' rendering power capability has increased dramatically these days, still, rendering a planet wide terrain in real time will require algorithmic adjustments. There are several LOD algorithms developed in order to make large scale terrains visual in real time.

Generally, data used for terrain visualization can go up to 1 meter resolution. This means we need to draw approximately 2 million triangles to represent 1 km^2 terrain area. With this number of polygons to be able to render in real time, performance of the system must be enormous. LOD algorithms are developed to get around this problem.

2.2.4. Terrain Visualization Realism

Visualizing a terrain requires realism effect. There are several ways to add realism to the terrain scenery. One of the methods is texturing the terrain with appropriate images. But using low resolution images can decrease realism instead of increasing it and high resolution imagery can be huge in size for large scale terrain visualization. Detail texture maps are often implemented to increase realism. Another issue is to implement LOD without losing terrain geometry higher than an acceptable amount.

Moreover appropriate shading of the terrain not only increases 3D perception but also terrain realism. Shadowing and shading is a crucial material of 3D representation. Model artists always render their models with appropriate light and ambient to increase realism.

2.3. Performance Management in Terrain Visualization

Virtual Environment (VE) systems have two goals to achieve. First is to duplicate and emulate the real world and second is to build a virtual world similar to the real world [19]. In order to achieve these goals, we need to model VE realistic and support appropriate computer human interaction interfaces.

Visualizing terrain as an application has researches on many areas such as performance improvement, realism effect or input handling. In this thesis we used performance improvement algorithms.

Performance management for visualizing terrains is always a hot topic for computer graphics. Graphics hardware has limited drawing capabilities no matter how advanced it is. In real time, visualizing terrain with its full resolution is not possible for higher FPS values more than 23 FPS. 23 FPS is the value for that eye can perceive pictures as motion pictures. However 30 FPS is generally the minimum limit for VR applications. Even 30 FPS causes flickering and jittering in image stream. 60 FPS or higher values are suited well to see smooth stream of images [20].

2.3.1. Level of Detail in Geometry

Level of Detail (LOD) is the name of the common techniques for drawing objects less detailed in distance and more detailed that are closer to the observer. Objects which are drawn in computer graphics environment consist of polygons and thus by less detailed we mean reduced polygon count for geometry. Figure 2.8 shows LOD representation of a sphere to be rendered.

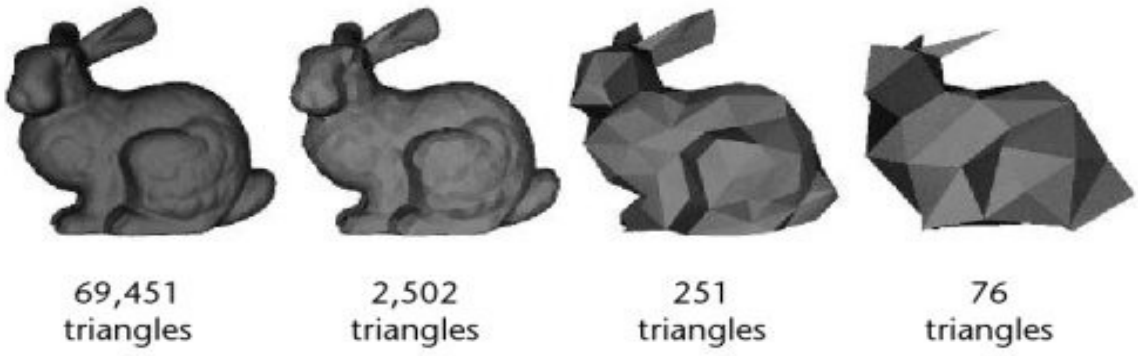


Figure 2.8. LOD representation of a rabbit model [21]

Algorithms for LOD vary according to object type and property to be rendered. There are two types of LOD algorithms:

- Continuous LOD (CLOD)
- Discrete LOD (DLOD)

In DLOD algorithms, objects get processed with and external algorithm to produce different LOD levels. These LOD levels are interchanged according to distance from the observer.

In CLOD algorithms, objects to be rendered got evaluated in rendering time and by some heuristic function, usually using distance from observer as parameter, LOD is calculated and object is rendered.

There are several LOD algorithms for TV. Arvaux *et al.* suggested parallel LOD management in order to achieve real time TV for large terrain datasets and big distributed simulations [22]. CABTT [23] is another proposed method to increase runtime rendering capability by dealing with clusters of geometry and storing on graphics card to improve performance of ROAM algorithm. ROAM algorithm is used for a long time and became a starting point for other LOD techniques in TV. Another algorithm for LOD calculation of TV is chunked LOD. Chunked LOD divides terrain into regular pieces and increases LOD according to observer [24]. Figure 2.9 shows representation of Chunked LOD.

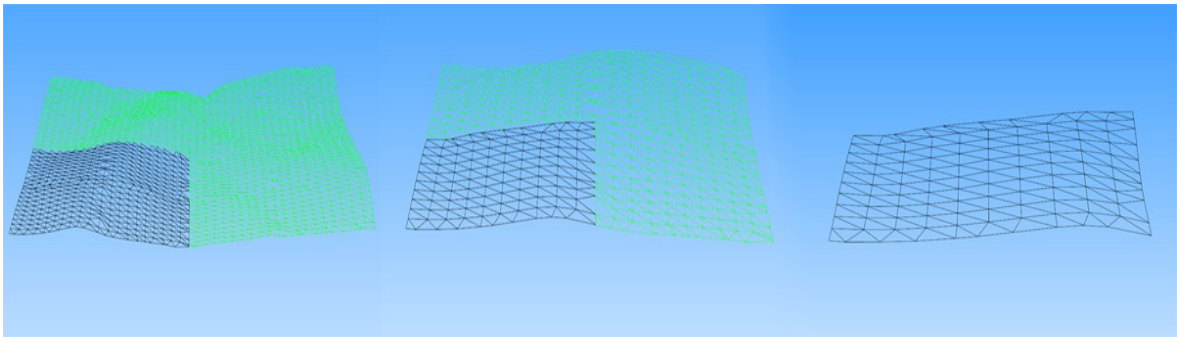


Figure 2.9. Representation of Chunked LOD

2.3.2. Level of Detail in Imagery

LOD is not only limited for object geometries. Images that are textured to geometries also can have LOD. LOD in images is called mipmapping. Mipmaps are optimized and pre-calculated smaller images according to original one. These images are stored along with the original image to be used in necessary conditions [25]. While observer is closer to the objects, image is rendered in full detail. However, when user get away from the object the renderer will switch to a suitable mipmap image. Figure 2.10 shows and example for a mipmap image.

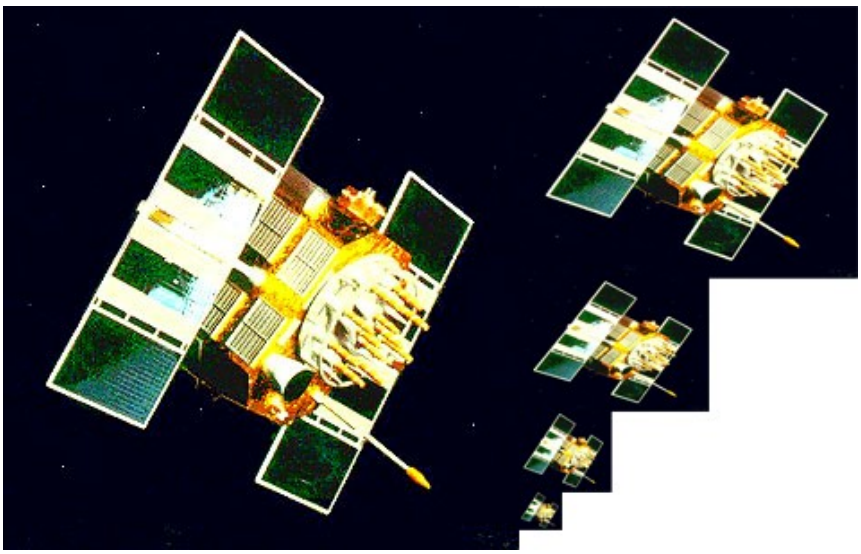


Figure 2.10. An example of a mipmap image

There are also some methods to switch or interpolate between mipmapped textures. Trilinear, bilinear or anisotropic texture filtering methods interpolate between mipmapped textures. Trilinear filtering woks as follows:

- Use the distance along the texture between the current pixel and the pixel to its right (or left, or above, or below) as the size of the pixel.
- Use the smallest (or biggest, or average) of the various sizes determined by using the above method.
- Determine the uv-values of the corners of the pixel, use those to calculate the area of the pixel, and figure out how many pixels of the exact same size would take up the whole texture.

2.3.3. View Frustum Culling

In order to avoid redundant polygon rendering that are not visible in the scene, it is required to intersect viewing frustum with the objects that are in the scene. This way, it is possible to rule out unseen polygons at the scene and reduce the workload of the graphics processor for drawing. In figure 2.11, red objects represent the polygons that are not inside the view frustum and therefore they are culled out for drawing. On the other hand, green objects intersect with the view frustum, thus they are drawn for viewing.

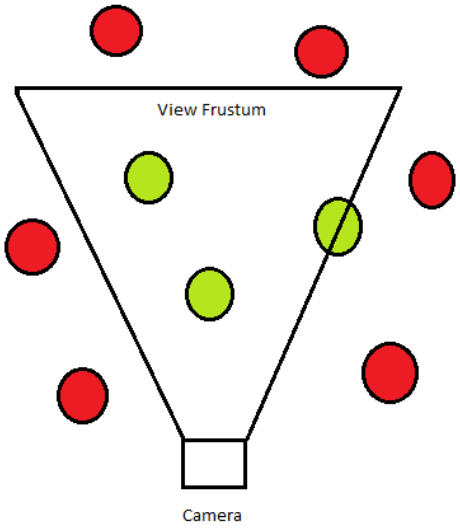


Figure 2.11. Frustum culling

2.4. Level of Detail in Terrain Visualization

In this chapter we will be explaining our implemented method for fast terrain visualization. As the terrain, grows in size, graphics hardware experiences difficulties to display objects in speed manner and computer may not handle inputs correctly. Sum of all, usability of application generally gets decreased. These difficulties are low FPS rates, lack of usability or even dead-locking computer. In our project, we have implemented a LOD algorithm to eliminate these problems and increase rendering performance.

LOD for terrains has some difficulties and easy parts apart from LOD methods for objects other than terrains. Easy parts are generally constrained geometry and more specialized, simple algorithms. On the other hand, models are continuous and very large to handle with LOD algorithm. Also we need to handle very far and close parts in the view frustum of the geometry at the same time.

Moreover, dealing with terrain LOD algorithm has some common problems like T-junctions or cracks in the terrain. Figure 2.12 shows crack and T-junction representation.

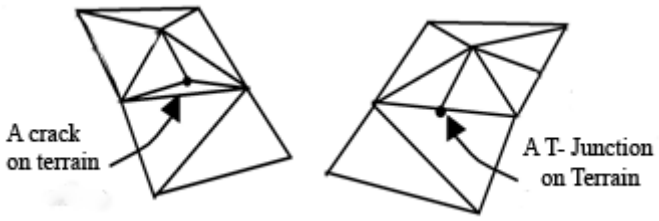


Figure 2.12. Cracks and T-junctions

With the ROAM algorithm's binary triangle tree approach, cracks and T-junctions are solved.

2.4.1. Roaming Terrain

ROAM is a LOD algorithm that is applicable to terrains, in order to have dynamic, view-dependent triangle meshes and texture maps at the required frame rate. Displaying large datasets like terrains contains several parts like disk paging, LOD for triangle

geometry, view frustum and triangle stripping. ROAM algorithm focuses on last three cases. Also terrain LOD algorithms must have a computable error metric. We will be investigating error metric for ROAM algorithm further. The most important property of ROAM is to be run-time calculated and view dependent LOD algorithm. Even today big budgeted simulation software, for example Presagis [26] products, uses pre-calculated LOD meshes and let scene graph application to deal with LOD level interchanges.

In a simple manner ROAM algorithm is dynamic mesh representation based on triangle binary trees. Binary tree definition will be explained in next subsection. ROAM algorithm contains a single preprocessing component and several run time components. The result of preprocessing component is nested, view independent error bounds, bottom-up, for a triangle binary tree. On the other hand, four components are performed for each frame in run-time:

- Recursive, incremental update to view-frustum culling
- Priority update for output triangles
- Triangulation update
- If needed updates for triangle strips affected by culling changes.

2.4.2. Mesh Representation

In this project, we use height map to reflect terrain elevation data. Height map we choose is the format that is cleansed from header information, called RAW. RAW files keep 8 bits of information for a cell. We than can scale height values that are between 0 – 255 according to our desire. Height maps can also be interpreted as image files. DTED and DED files can also be converted to RAW file format with appropriate programs. We have conducted our work on an imaginary terrain and real terrain information. Real world information is western side of Lake Sapanca.

ROAM uses triangle binary tree to represent mesh structure of the terrain. Triangle binary tree is a triangulation system that each edge is got split into two parts according to selected root node. Triangles defined in triplets as node names. When a triangle is split,

produced child triangles can continue to split recursively. Figure 2.13 shows a representation of a binary tree.

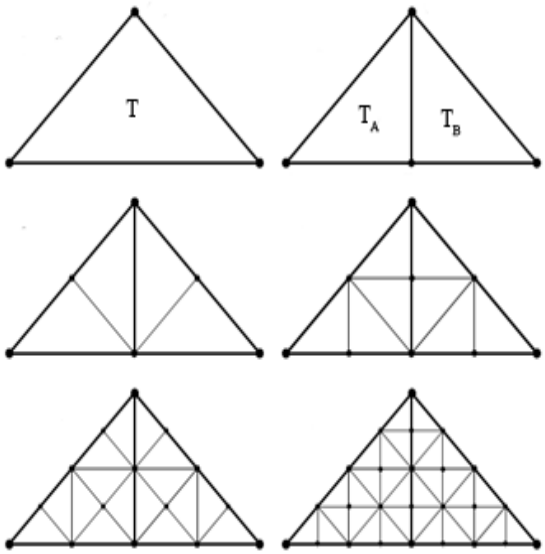


Figure 2.13. Triangle Binary Tree representation

As we can see in Figure 4.1 , triangle T is split from along its root node , called apex vertex, and resulting in 2 new child triangles, called T_A and T_B. By recursively splitting a triangle, triangle binary tree is formed. Complexity of triangle binary tree increases as the splitting increases thus next fine level for a triangle binary tree is the tree formed after one more recursive splitting.

Mesh in the world is formed by assigning world coordinates to each tree vertex. In order to obtain dynamic and continuous triangulations some triangles may overlap at a common edge or a vertex. This continuous structure is maintained by splitting or merging operations defined on triangle binary tree. Figure 2.14 shows split and merge operations on terrain mesh.

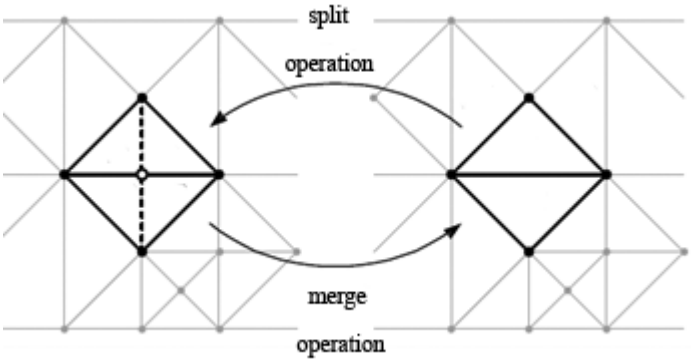


Figure 2.14. Split and merge operations on mesh

In our application we have implemented our triangle mesh structure as “PATCH”, which is stated in [27]. Figure 2.15 shows us a representation of base data structure of ROAM algorithm that we have used. While we are creating the mesh of the terrain, we add children to the triangle tree until we reach to the desired detail level. After completing constructing tree, we traverse it one more time in order to render it. For each frame, tree is reconstructed and allocations of vertices are done for every single one of them.

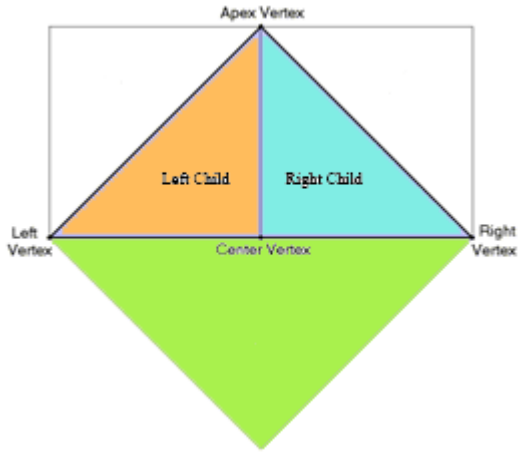


Figure 2.15. Triangle mesh data structure representation

In order to manage patches, we define one more class named “GEOMETRY”. This class keeps elevation data as well as visibility calculation parameters. Main importance of this class for ROAM algorithm is that we can chain more terrain elevation files and visualize larger terrains. We divide our terrain geometry into patches to increase management ability of terrain mesh. Each patch is actually made up of two discrete binary triangle trees fitted together into a square.

2.4.3. Error Management

In ROAM algorithm, in order to make a mesh approximation, we must define an error metric. This metric enables us to decide when a node should be split (to add detail), and how deeply to split it. Original ROAM algorithm uses an error metric called nested world-space bounds. It uses two priority queues, geometric screen distortions and line of site in order to handle merge or split operations. However the algorithm we implemented uses a simpler and a slightly faster method for error metric called variance.

Variance is the difference between height of the interpolated hypotenuse midpoint for a binary triangle node and the actual height field sample at that point. Calculation of this value is rather quick and only requires one memory reach. Formula is shown in Equation 2.1.

$$\text{triVariance} = \text{abs}(\text{centerZ} - ((\text{leftZ} + \text{rightZ}) / 2)) \quad (2.1)$$

However, error metric evaluated from a single root of the tree results in a high error rate. Thus we need to calculate deeper in the tree and average it up back to the root. In our project we predefined depth of this calculation for simplicity proposes. If we have a dynamic terrain structure, we need to calculate variance each frame. But since we assume our height field remains same as long as the duration of the application, we can build a variance tree along with binary triangle tree. Variance tree is a full-height binary tree written into a sequential array. Figure 2.16 shows a representation of a binary tree as an array.

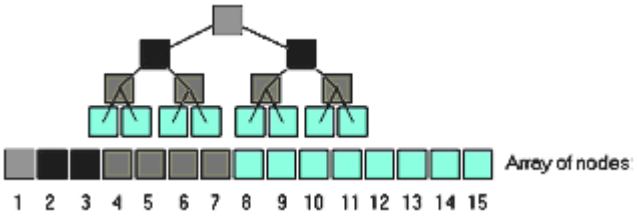


Figure 2.16. Binary tree as an array

After building our variance tree, we split up our binary triangle tree until we reach desired error metric. Each split reduces our error almost in half. We can split until running

out of variance steps or reaching to the resolution of the terrain height field. Figure 2.17 shows different variance effect on terrain.

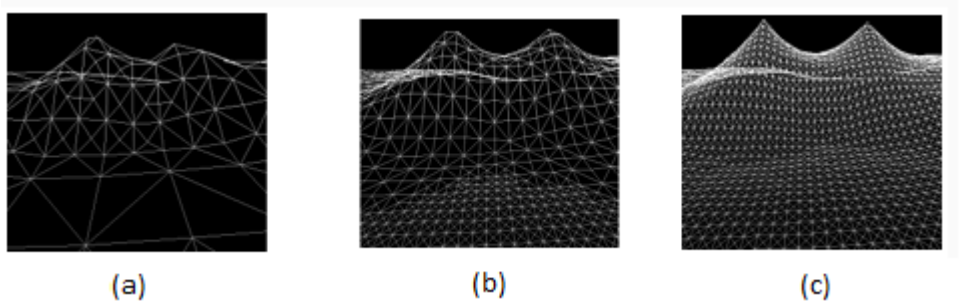


Figure 2.17. Variance (a) Low variance, (b) Medium variance, (c) High variance [27]

A common problem with terrain LOD is to prevent cracks on the terrain. Cracks occur in ROAM from splitting of the trees across patch boundaries. Figure 2.18 shows a sample crack on terrain.

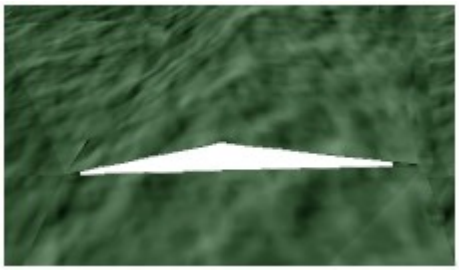


Figure 2.18. A crack on terrain

ROAM uses neighbor pointers and force split operation in order to prevent cracking on terrain. Force split operation is represented below in Figure 2.19.

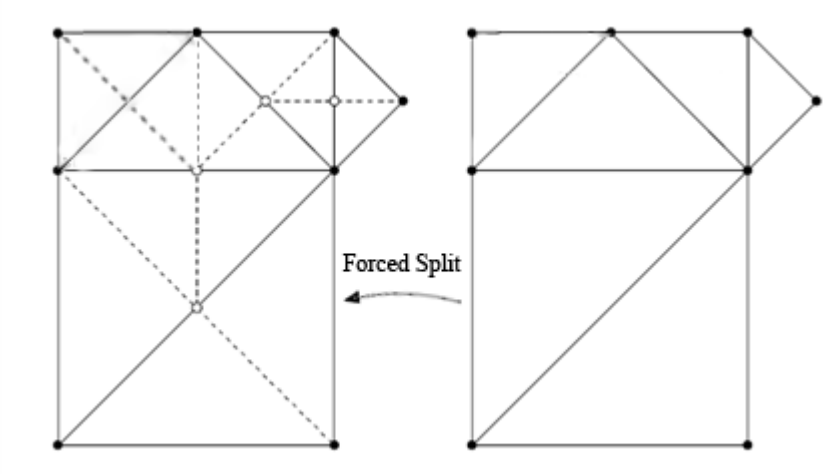


Figure 2.19. A force split operation on ROAM

3. RESPONSIVE WORKBENCH OVERVIEW

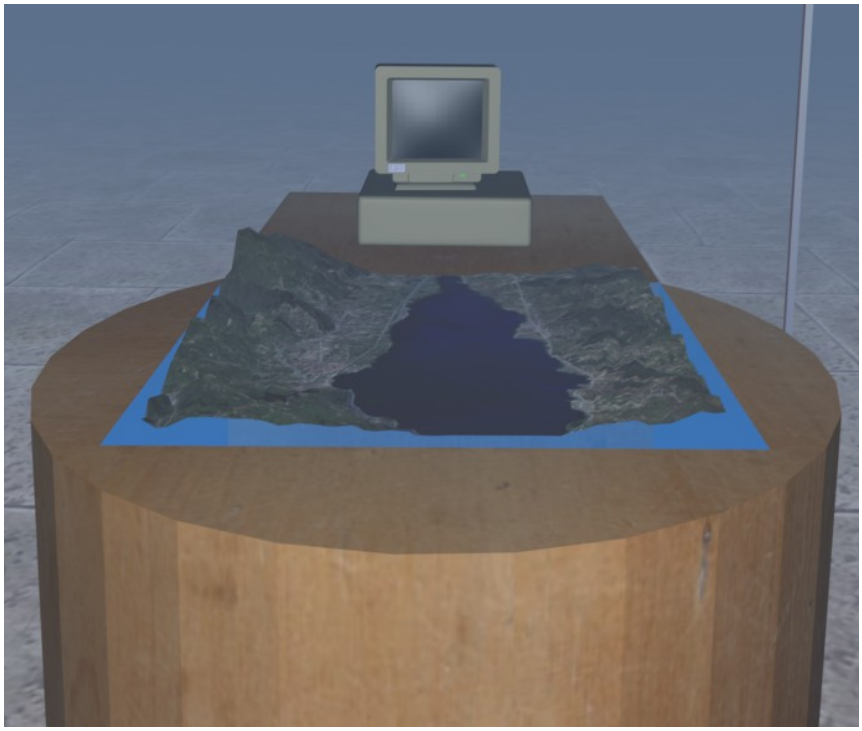


Figure 3.1. Responsive workbench system

Responsive workbench systems, also known as tabletop systems, are first proposed by Krueger *et al*. Responsive workbench systems contain a horizontal display for virtual object viewing. (see Figure 3.1). Virtual objects are projected onto this display as stereo image pairs. Image pairs are generated by a stereo capable computer and displayed by a stereo capable display. This setup of the workbench corresponds to a doctor's examination table or an engineer's design workbench. Responsive workbench systems can be integrated with various input and output capabilities such as motion, gesture and voice recognition systems as well as hand held control devices [28].

RWB systems change the classical idea of computer – human interaction. Today, there are companies that are developing table top systems to involve users in a different way of human-computer interaction like Sony, with atracTable [29] or Microsoft, with Surface [30].

Responsive workbenches' are used for multiple purposes. By using responsive workbench, an architect could inspect a virtual 3D model of a new building and its surrounding area before creating a physical model. A new car model could be displayed and annotated in a design studio before a 1-to-1 scale physical clay model is built.

Durbin *et al.* [31], used such a system for a battlefield simulation. In this work, it is aimed to increase commander situational awareness and fasten decision making on a battlefield map.

Piper *et al.* suggested a system for patient and doctor communication [32]. In this work, deaf patients and doctors are able to communicate with each other. Also it is possible that, a team of doctors could plan a surgery with 3D virtual representation of a patient's body.

Wesche *et al.* [33], produced a system to make user drawings easier and be aware of 3D object by using stereoscopic goggles.

Another solution of Wesche *et al.* is to use responsive workbench for investigating fluid dynamics [34]. System is used to evaluate huge datasets for an automobile manufacturer in an interactive way. Figure 3.2 shows system in use by Wesche *et al.*

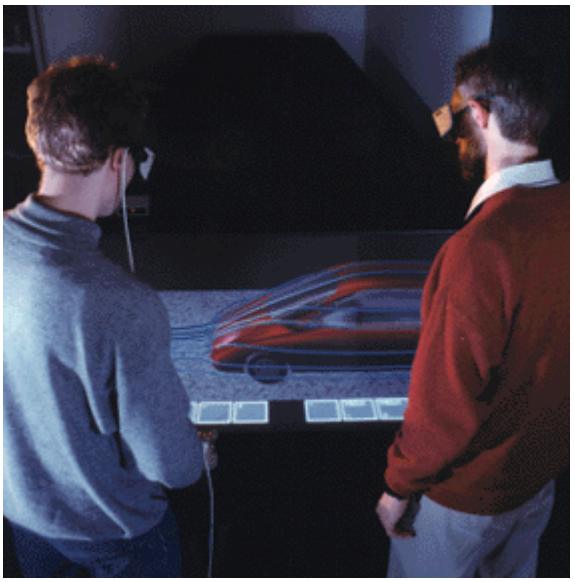


Figure 3.2. Responsive workbench in use for fluid dynamics [34]

The classification of tabletop systems proposed by Grossman *et al.* [35]. It is shown in Figure 3.3 shown.

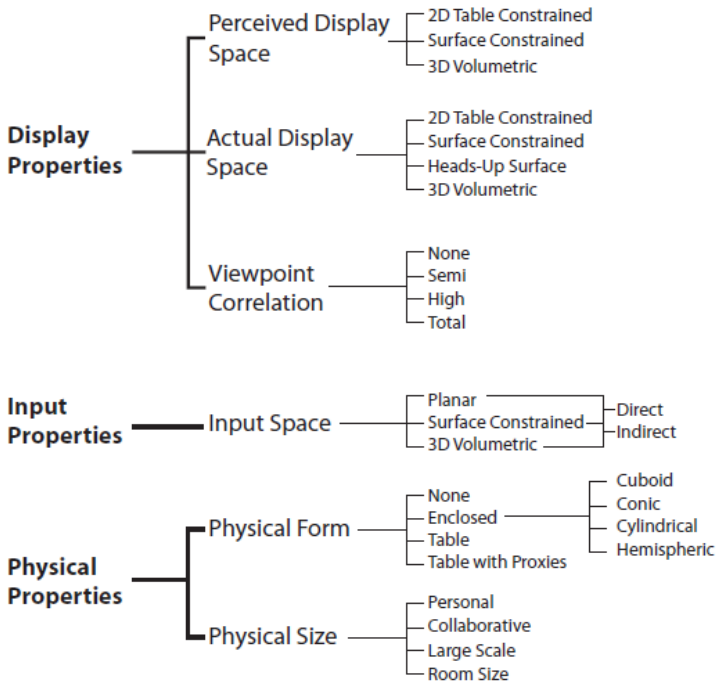


Figure 3.3. Taxonomy of Interactive 3D tabletops [35]

Display devices are classified in three different ways in table top display systems. First, perceived display space is important about classification table top display systems. The perceived display space is defined as the possible spatial locations for which displayed imagery can exist based on stereoscopic depth cues. Display space can either be constrained by nature of 2D table or 2D images that are shown on surface. Even if we show 3D objects, they are projected on 2D surface thus, display are will be constrained.

A variety of solutions proposed by researches for the display aspect. Wesche *et al*. used one vertical and one horizontal display. Durbin *et al*. used a single horizontal display. These two systems have different perceived display spaces. Durbin uses 2D table constrained type of workbench, on the other hand Wesche uses surface constrained workbench.

With respect to display space, Grossman *et al.* defines actual display space. The actual display space considers where the actual displayed imagery exists. In 2D surface constrained systems images appear on table itself. In surface constrained systems, perception is gained through physical objects. In heads – up systems, perception space is located up to table surface such as head mounted display systems. Finally, 3D volumetric displays have as 3D volumetric actual display space.

Final property of display systems of table top systems are viewpoint correlation. This group considers viewpoint ranges changing according to moving around the table physically.

Another important issue in workbench systems is how user interacts with the objects. For 2D display systems, touching the display surface to interact with the objects are used which is defined as Direct 2D interaction. But when using stereoscopic images, since objects are no longer constrained to the surface, touching is not an option. Direct 3D interaction takes place when user can hold and grab objects. When surface is big and user cannot reach all surfaces, indirect methods are used (Indirect 2D and Indirect 3D). Also for augmented reality systems, direct surface constrained input is defined. Painting a physical object with a virtual paint can be an example for this type. Here interaction is constrained with the surface of the object. On the contrary indirect surface constrained input type is defined if an intermediate input layer is used for interaction. Volumetric displays can be counted within this type.

Koutek *et al*. conducted a study to analyze the degree of realism while interacting with virtual environment in responsive workbench systems. In this study methods are represented for realistic behavior of virtual objects and a set of user input techniques. Koutek *et al*. introduced their method by using springs and physical principles [36].

Moreover, workbench proposed by Schmalstieg *et al.* is in the classification type for indirect surface constrained types. In this work, a small interaction board is used on workbench system and user interacts with responsive workbench [37].

Physical form of a system is important because form of the system defines input area and display surface. Size of the display system is important for it affects input space.

Paljic *et al*. conducted a research on distance of manipulation effect in a responsive workbench system. They concluded their work as close distances for manipulation is more effective for 20 cm distances than 40 – 55 cm distances. Large physical systems are hard to control according to this study [38].

There are personal sized, collaborative sized, large sized and room sized systems. Personal sized systems are for one person usage. Collaborative systems are for multiple user systems that all of the surface can be reachable for at least one user. Large systems are for multi user usage but some parts of the surface are not reachable for all users. Room sized type systems are large as a room and some parts of the system may even not be visible to all users.

3.1. Stereoscopic Visualization

Stereoscopic visualization helps to increase the illusion of reality in virtual systems. A quad buffered graphics card and shutter glasses are usually used for this purpose. This is achieved by providing a sequence of images for each eye. Figure 3.4 shows a rectangle block for left and right eye visions.

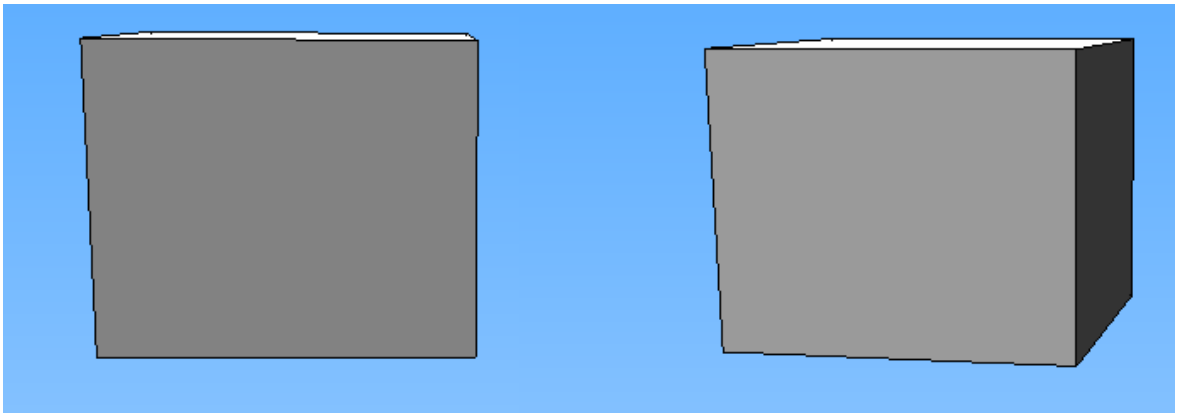


Figure 3.4. An example of angle difference between eye sights

There are number of stereoscopic image creation methods. Wiggle stereoscopy, anaglyph stereoscopy, polarized stereoscopy and shutter glasses stereoscopy are among these. Wiggle stereoscopy is achieved by alternating left and right images to both eyes. Polarized stereoscopy is achieved by superimposing both left and right images on the screen and filtering images for each eye by using polarized filters. Anaglyph stereoscopy is achieved by superimposing both left and right images in different colors and filtering images for each eye by colored glasses .Figure 3.5 shows an anaglyphic stereo image of a terrain.

Figure 3.5. An anaglyphic stereoscopic terrain view

A system using stereoscopic terrain viewing is suggested by Wartell *et al.* [39]. They suggested a head tracking display to manage camera and interaction with user.

3.1.1. Stereoscopy Geometry

In stereoscopic viewing, object that is viewed can be felt weather inside the screen or outside the screen. This effect depends on the image projection place. Therefore we must first understand the perspective projection in detail.

In perspective projection lines converge to a single point of projection and parallel lines in view direction merge. This enables to view far objects smaller and near objects bigger. View frustum has two planes to define a volume of a scene in Figure 3.6. Every object that falls into the volume is projected onto the screen via perspective projection and rendered afterwards.

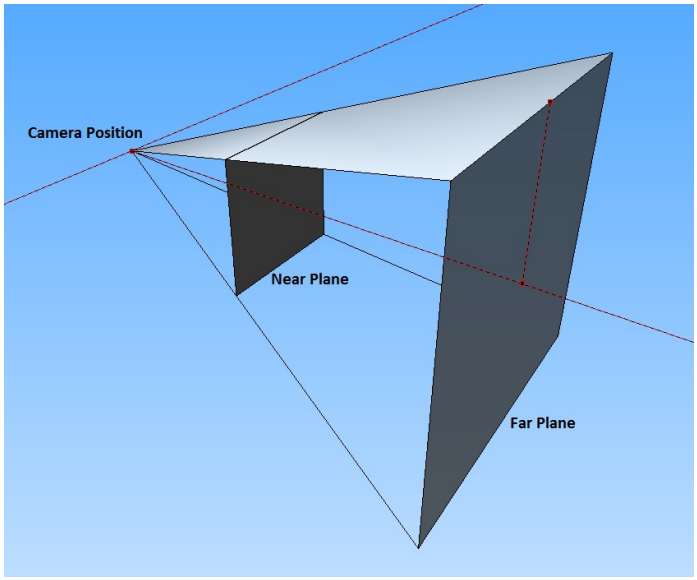


Figure 3.6. View frustum planes

In order to calculate projection result on 2D screen, 3D objects must be processed by a projection matrix. Assuming up direction length of the projection is as 1 for simplicity purposes. Figure 3.7 and Equation 3.1 shows camera distance relation with projection plane.

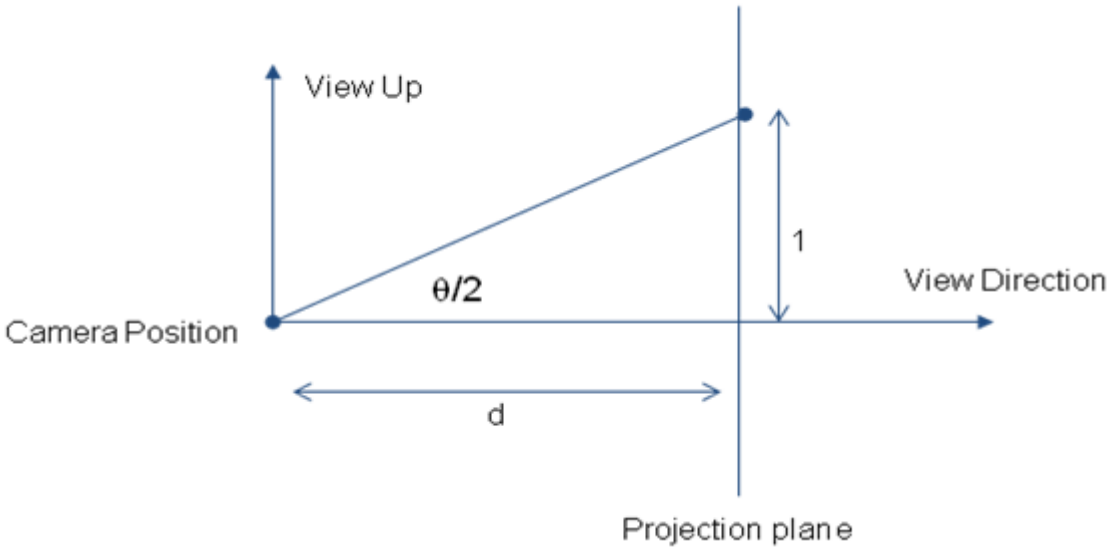


Figure 3.7. Camera distance to projection plane relation

$$d = \cot\left(\theta/2\right) \tag{3.1}$$

Similarity of triangles gives us Equation 3.2 in Figure 3.8.

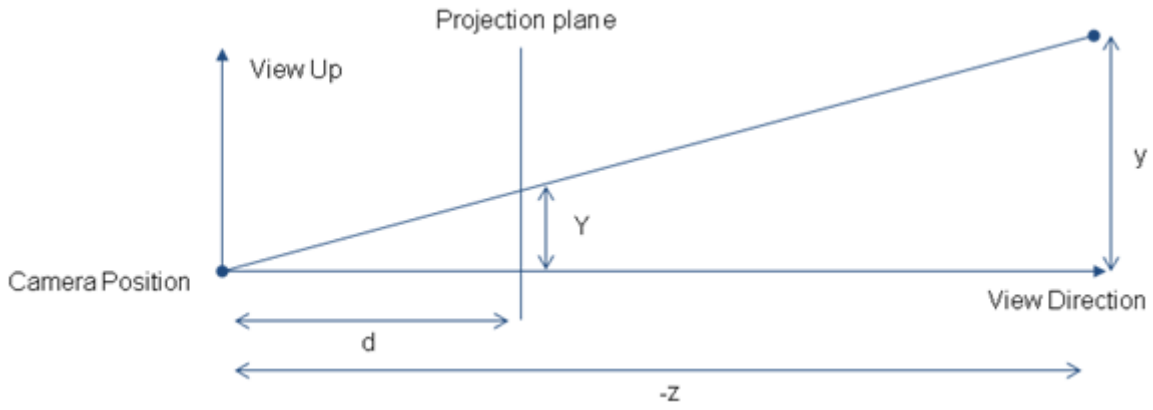


Figure 3.8. Similarity of triangles used in projection calculation

$$\frac{Y}{d} = \frac{y}{-z} \tag{3.2}$$

We also have to consider if our projection plane is square. If it is not, we must consider aspect ratio in calculations. Aspect ratio is proportion of screen width and height. Equation 3.3 shows aspect ratio formula.

$$a = \frac{w}{h} \tag{3.3}$$

After adjusting projection parameters according to aspect ratio in X and Y axis Equations 3.4 and 3.5 are obtained.

$$X = \frac{dx}{-az} \tag{3.4}$$

$$Y = \frac{dy}{-z} \tag{3.5}$$

We also need to adjust Z distance of projected item. We map near and far clipping planes defined in view frustum in 1 to -1 interval. Equation 3.6 which is derived from Figure 3.9 gives us Z mapping equation. In Equation 3.6, n variable represents near clipping plane and f variable represents far clipping plane distances.

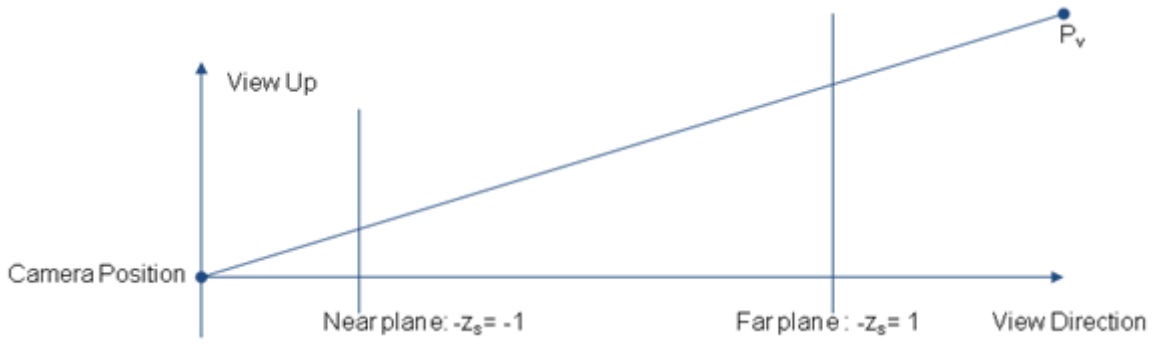


Figure 3.9. Z mapping for perspective projection

$$Z = \frac{z_v(n+f)+2nf}{n-f}\left(\frac{1}{-z_v}\right) \tag{3.6}$$

After the projection process, if the projection for the left eye is at the left and the projection for the right eye is at the right, then resulting image will create an illusion that the object is above the screen. This is called positive parallax. (see Figure 3.10).

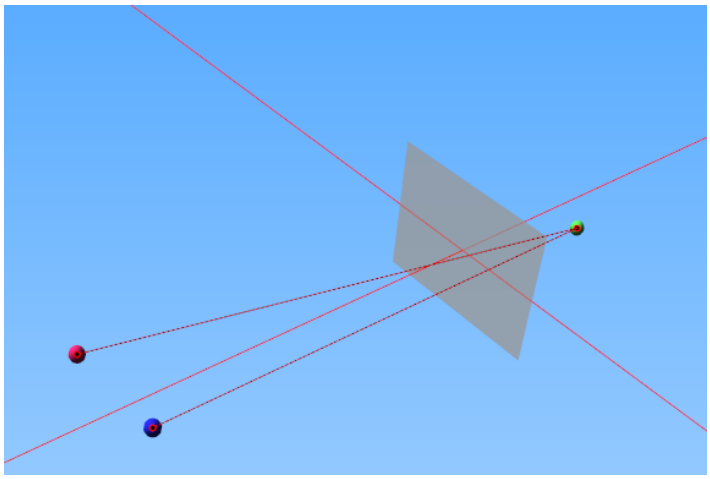


Figure 3.10. Positive parallax

There is an opposite situation of positive parallax which is called negative parallax. If the projection of the object for left eye is on the right and projection for right eye is on the left, then illusion is as if object is below the screen. (see Figure 3.11).

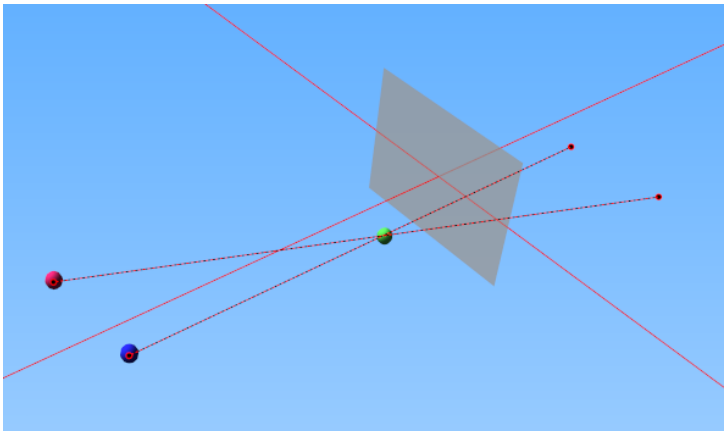


Figure 3.11. Negative parallax

Last situation is zero parallax. If projections for both objects intersect on screen, then image is felt on projection plane itself. (see Figure 3.12).

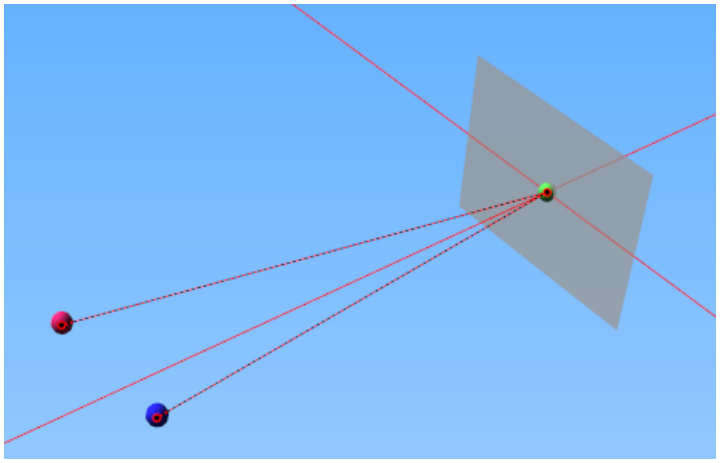


Figure 3.12. Zero parallax

3.1.2. View Frustum Management

In 3D graphics applications, to observe scene, we must define a camera Field of View (FOV). This FOV is in a shape of a pyramid which top is cut off with a plane. Camera FOV of the 3D scene is called view frustum. Objects that fall inside of the frustum is projected and displayed on screen. Figure 3.13 shows a drawing of view frustum.

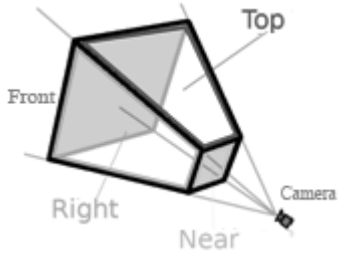
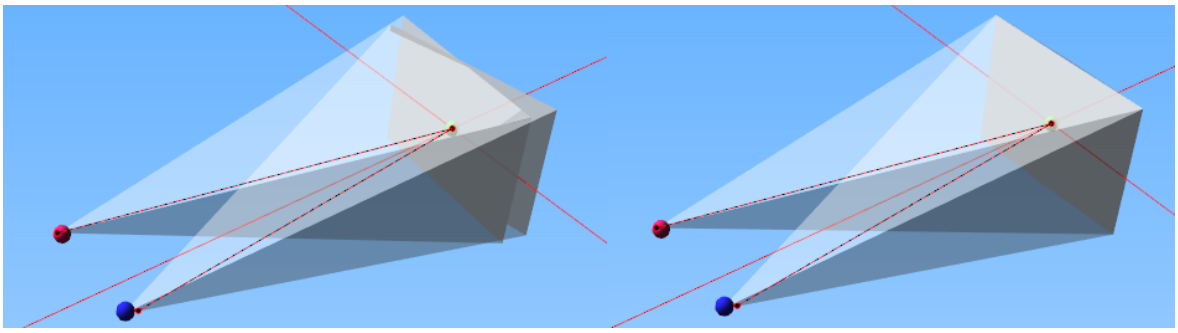


Figure 3.13. View frustum

Defining a regular view frustum can produce undesired results for stereoscopy generally as a discomfort at viewing the scene. In order to correct this, we must define a non-symmetric frustum pair for the projection plane which is called asymmetric frustum. Figure 3.14 shows wrong and right frustum definitions for stereo viewing.

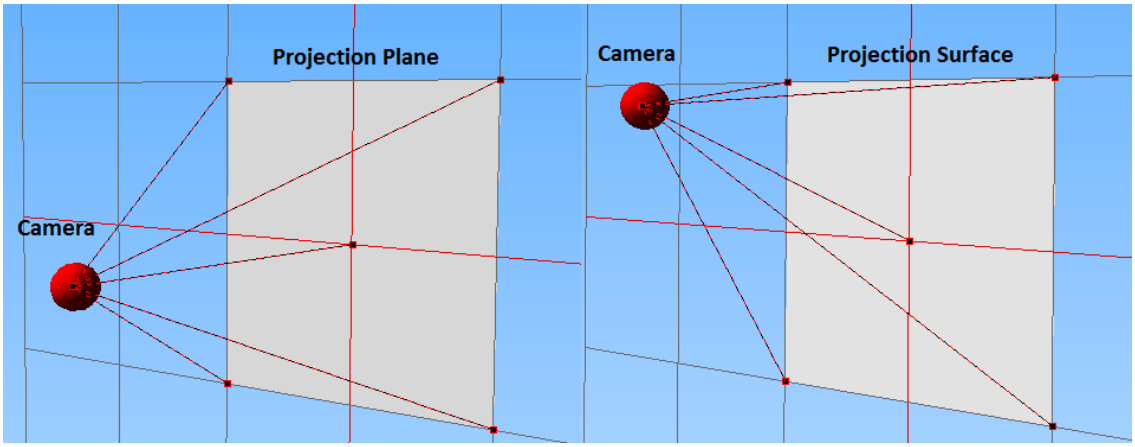


(a) (b)

Figure 3.14. Frustum pairs (a) Wrong frustum pair (b) Right frustum pair

3.1.3. On - Axis and Off – Axis Projections

There are two basic types of projection methods. One of the methods is on-axis projection. A frustum system is called on-axis if the axis of symmetry of all surfaces and the camera direction axis coincide. For display surfaces this is the case if the center of the camera frustum projects on the projection plane center. Vertical display devices are suitable for on-axis frustum viewing. Figure 3.15 shows on-axis and off-axis projections.



(a) On-axis Projection (b) Off-axis Projection

Figure 3.15. On-axis and Off-axis Projections

Off-axis projection is called systems where the center of projection projects onto a non-centered position on the display screen. This type of projection is suitable for horizontal display systems since user can not look directly above the screen. Figure 3.16 shows off-axis projection geometry.

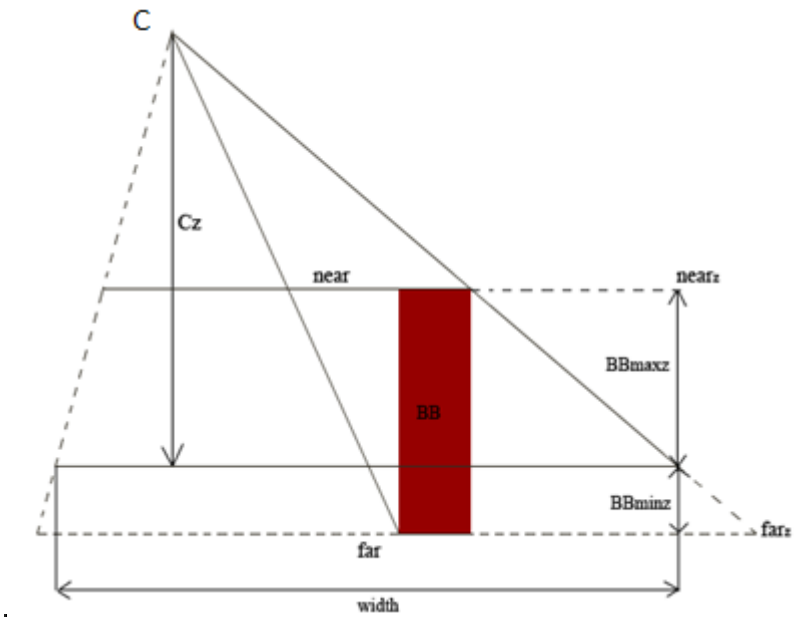


Figure 3.16. Off-axis projection geometry

In order to find the corresponding frustum for the off-axis projection, the scene's bounding box (BB) is required to computer the near and far clipping planes. Assuming C is

the camera position and the center of the projection, the near and far clipping planes are computed as follows:

$$nearz = Cz - BBmaxz - 1 \tag{3.7}$$

$$farz = Cz - BBminz - 1 \tag{3.8}$$

An offset value of 1 is required to prevent clipping of the front or backside faces which are aligned with the bounding box's top or bottom. For the on-axis situation, assuming camera position as C = [0, 0, z], near clipping plane can be determined as:

$$nearwidth2 = \frac{nearwidth}{2} = \frac{width}{2} ratio \tag{3.9}$$

$$nearheight2 = \frac{nearheight}{2} = \frac{height}{2} ratio \tag{3.10}$$

In Equation 3.9 and Equation 3.10, $width$ and $height$ are the dimensions of the projection plane, where the *ratio* is given by Equation 3.11.

$$ratio = \frac{nearz}{Cz} \tag{3.11}$$

Since our projection is off-axis, we need to compute a projectional shift before the parameters of the view frustum can be determined. Equation 3.12 and Equation 3.13 shows formulas for shift computation.

$$shiftx = Cx.ratio \tag{3.12}$$

$$shifty = Cy.ratio \tag{3.13}$$

After calculating these parameters, we can define our frustum by using OpenGL's glFrustum( ) method.

4. VISUALIZATION WORKSHOP RESPONSIVE WORKBENCH SYSTEM

Visualization workshop workbench is a horizontal display system with head tracking capability. Responsive workbench system displays virtual objects on a table top display. Display device is a stereo capable projector that projector images are transferred on table top screen via an auxiliary mirror. Images projected onto a translucent surface. Images are displayed at 120Hz and 1024 x 768 resolution. System uses a graphics computer in order to generate images to be displayed. Nvidia Quadro FX1400 graphics card is used in computer system and it generates stereoscopic image pairs. Since display device is stereo capable, system is used by wearing shutter glasses for true 3D viewing. Figure 4.1 shows RWB setup at the visualization workshop.

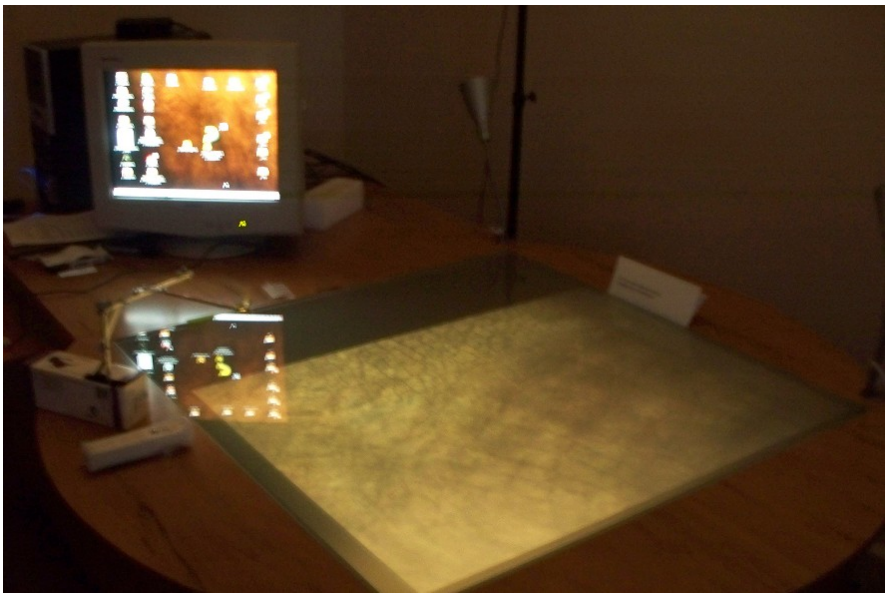


Figure 4.1. RWB in Visualization Workshop

Head tracking system tracks 3 LEDs mounted on the stereo goggles and calculates the relative position of the head. User can move around the table without an interrupted stereo vision. In order to track LEDs, a stereo camera captures images at 50 FPS is used.

System runs on Ubuntu Linux operating system, version 8.04. All of the system software is open sourced and free.

User can give input commands by using Nintendo Wii (wiimote) [40] controller device to interact with the table. Controller communicates with graphics pc via Bluetooth communication protocol.

4.1. RWB System Specifications

The system is composed of the following components:

- Display unit
- Stereo capable camera
- Graphics computer
- Shutter Glasses
- 6 DOF controller

Responsive workbench uses a stereo capable projector [41] to generate required images at 60 frames per second per eye. Projection is reflected via a mirror onto a matt horizontal surface inlay on the bench.

Crystal Eyes 3 [42] shutter glasses are a vital component for the stereovision. For each frame that is drawn for the each eye, glasses open one of the eye sights and close another. Since it is working synchronized with the graphics card, it makes sure that each frame that is produced for the particular eye is viewed by the targeted eye.

The graphics card used in system has double buffering (for each eye) ability that allows system to produce stereoscopic images. Some specifications of the graphics card can be found in [43].

The user can interact with the system via:

- Keyboard and Mouse

- Wiimote as a 6 DOF control device
- TUIO protocol as touch surface

4.1.1. Head Tracking System

Head tracking system is an important component of the RWB. System allows us to determine position and the orientation of the viewer's head. By knowing actual position of the user's head and viewing direction, we can manipulate camera in out virtual system and obtain a realistic observation of the VE running on RWB. Head tracking in our system also enables us to investigate stereoscopy distortions and effects for terrain rendering on RWB with respect to head movements. System is implemented and compiled as a library which is called by any application that uses head tracking property [44].

Head tracking system that is used on RWB has 3 main components. These components are:

- Stereo capable camera
- Head mounted LED system for head tracking
- Image processing software

First of all in order to track user's head, we need something to be tracked by stereo camera. Light dot in a dark canvas is a suitable tracking object. In order to do that, 3 LEDs fused via a mechanism for user's head. Figure 4.2 shows the LEDs to be tracked for head tracking information.

Figure 4.2. LEDs mounted on a frame attached to stereo goggles

Second component of the head tracking system is stereo capable camera. This camera gets image of the helmet by two slightly different angles. By calculating difference between two images, we can calculate distance of the tracked object thus we can find objects position in space that is relative to cameras position. Figure 4.3 shows the stereo camera that is being used by RWB head tracking system. A stereo camera by Point Grey Research (Bumblebee 2) is used for this purpose.

Figure 4.3. Stereo camera that is being used by RWB.

Last part of the head tracking system is calculations and setup for determining user's head position and view vector. Frame by frame, image processing based tracking is done by head tracking system. As we mentioned above, if we take 2 different images of a scene by slightly different two angles, we can find position of an observed point relative to the observing camera [45]. If we know relative position of cameras, then we can find observed point's distance relative to cameras. Figure 4.4 shows geometry setup for stereo camera used for head tracking system of RWB.

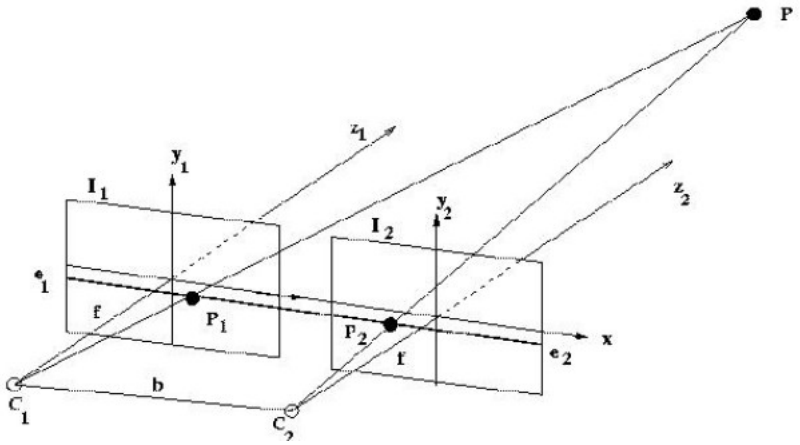


Figure 4.4. Geometry setup for head tracking [45]

3D coordinate reconstruction is done, after obtaining positions of the 3 LEDs coordinates from stereo camera. Initial calibration is required to reconstruct 3D coordinate data. User clicks LEDs positions on a window and initializes head tracking system. 3 LEDs on user's head forms a triangle. After obtaining 3D coordinates of each LED, head tracking system calculates user's head orientation from the triangle formed by LEDs.

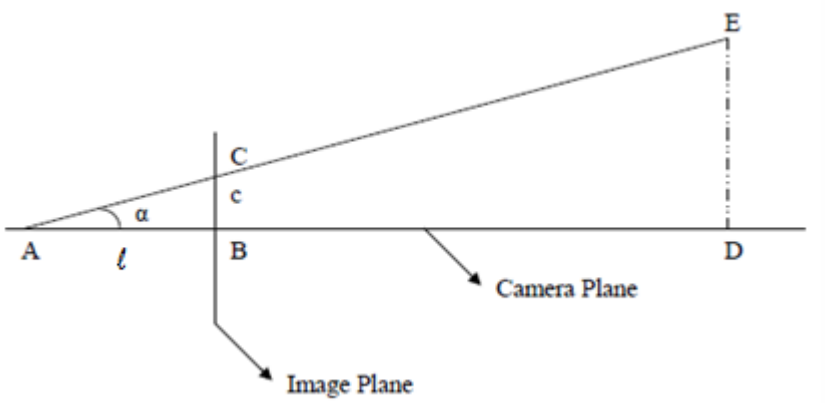


Figure 4.5. Side view of camera setup [45]

In Figure 4.5 side view of the camera system is represented. Here point A is the camera position and the E is the point that is being investigated for 3D reconstruction. The line segment is called c between points C and B; and the line segment is called ℓ between points A and B. Here we use some camera parameters like Horizontal Field of View (HFOV), Vertical Field of View (VFOV), image height and image width.

$$\tan\left(\frac{VFOV}{2}\right) = \frac{height/_2}{l} \tag{4.1}$$

$$\tan(\alpha) = \frac{c}{l} \tag{4.2}$$

Equations 4.1 and 4.2 are derived from Figure 4.5 applying tangent rule. When we divide Equation 4.1 and 4.2 to each other, we can eliminate length |AB|. Equation 4.3 is gained through this division.

$$\alpha = \arctan\left((2x\ \frac{c}{height})xtan(\frac{VFOV}{2})\right) \tag{4.3}$$

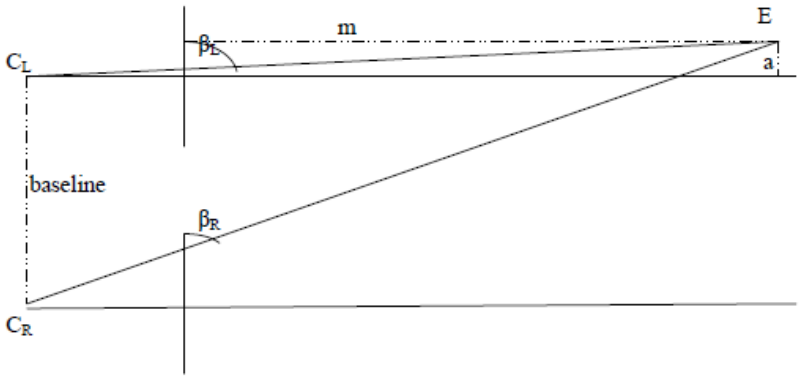


Figure 4.6. Top View of the camera setup [45]

Figure 4.6 shows the view of the camera setup on top. Baseline is the aperture between two cameras and m is the distance from point E to image plane. C_L is the left camera center and C_R is the right camera center respectively. Distance between point E and left camera direction is marked as a. d_L and d_R is the horizontal displacement of point E's projection onto image plane from the center of image plane.

$$\tan(\beta_L) = \frac{k}{d_L} = \frac{c}{d_L x \sin(\alpha)} \tag{4.4}$$

$$\tan(\beta_R) = \frac{k}{d_R} = \frac{c}{d_R x \sin(\alpha)} \tag{4.5}$$

$$\tan(\beta_L) = \frac{m}{a} \tag{4.6}$$

$$\tan(\beta_R) \approx \frac{m}{a+baseline} \tag{4.7}$$

If we divide Equation 4.6 and Equation 4.7 to each other we obtain Equation 4.8.

$$\frac{\tan(\beta_L)}{\tan(\beta_R)} = 1 + \frac{baseline}{a} \tag{4.8}$$

After reordering the terms we obtain Equation 4.9 and Equation 4.10 for variables a and m.

$$a = \frac{baseline}{\frac{\tan(\beta_L)}{\tan(\beta_R)} - 1} \tag{4.9}$$

$$m = a\ x\ \tan(\beta_L) \tag{4.10}$$

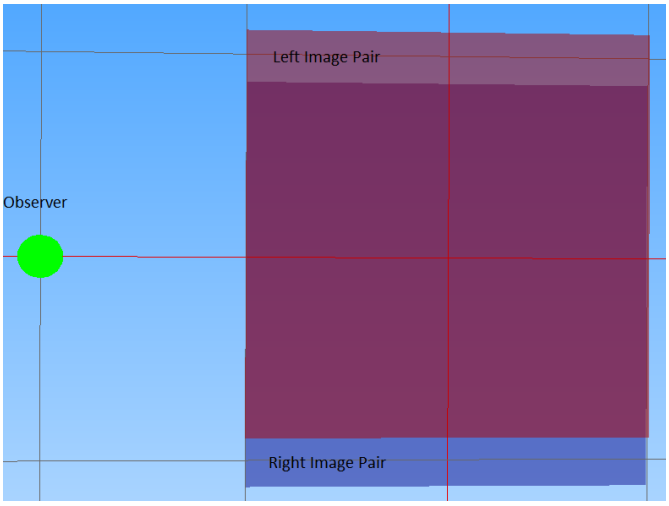
Finally we obtain displacement for point E in following equations.

$$m_X = a + \frac{baseline}{2} \tag{4.11}$$

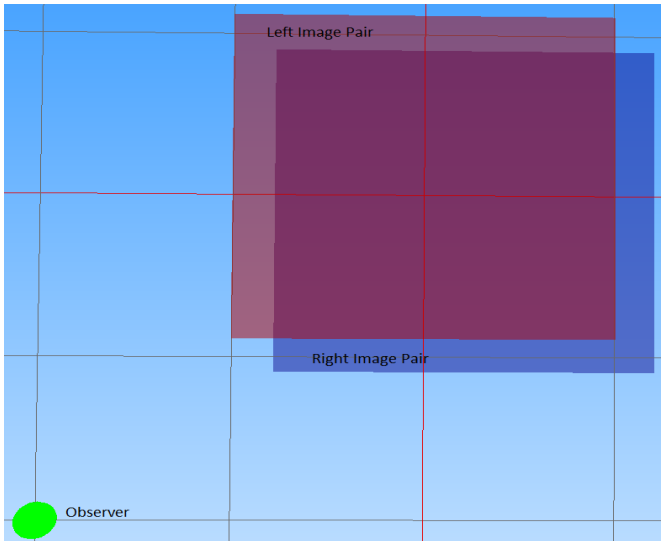
$$m_Y = m\ x\ \sin(\alpha) \tag{4.12}$$

$$m_Z = m\ x\ \cos(\alpha) \tag{4.13}$$

Stereo viewing in workbench is only possible from predefined positions if a head tracking system does not exist. In order to enable stereo viewing anywhere around workbench image pairs for left and right eye must be adjusted relative to user position. (see Figure 4.7).



(a)



(b)

Figure 4.7. Stereo pair adjustment (a) Observer at west (b) Observer at southwest

When we run workbench without head tracking, we can only observe stereoscopy effect from some specific predefined positions. This is because stereo image pairs must be adjusted according to user position to view stereoscopy effect independent of user position. Distortions in this mode are different than the distortions in head tracked mode.

4.1.2. Off Axis Projection on VW Workbench

VW workbench supports stereoscopic visualization. The display infrastructure of the workbench is programmed using C++ and OpenGL. OpenGL draws every frame two times for each eye respectively. OpenGL launches buffers of the graphics card and renders by using left and right buffers to draw frames for each eye. Figure 4.8 below, shows code sample to obtain stereoscopy in OpenGL environment.

```
Input:None
Select left buffer of graphics card.
Clear buffer.
Draw scene from left eye point.
Select right buffer of graphics card.
Clear buffer.
Draw scene from right eye point.
Swap buffers.
Output:  Stereoscopic 3D scene
```

Figure 4.8. Stereoscopy pseudo code in OpenGL

The GLUT library is used for platform independent input / output and window management. Off axis projection is required for horizontal displays. If symmetric frustum is used for horizontal displays, the scene looks sheared. However, user can only see a correct view by looking directly downwards from above the horizontal display. Off axis projection is a special asymmetric frustum that is consistent with observer position. In this method, a perspective transformation is applied and looking angle is adjusted relative to observer position. Figure 4.9 below, shows code sample for adjusting off axis frustum pairs.

```
Input:Camera position C and frustum values
Define left, top, bottom, near and far frustum variables.
Frustum left = C.x - resolution width / 2 - C distance x.
Frustum right = C.x + resolution width / 2 - C distance x.
Frustum bottom = C.y - resolution height / 2 - C distance y.
Frustum top = C.y + resolution height / 2 - C distance y.
Define frustum with calculated values.
Place camera to position.
Begin drawing.
Output:  Off axis projected scene
```

Figure 4.9. Off axis frustum pseudo code

4.1.3. 6-DOF Control Device

Wiimote is a game controller that is produced by Nintendo Company for their gaming console Nintendo Wii. Controller has accelerometer and camera to receive motion & position changes. It also has buttons, vibration property and sound warning. By using accelerometer, controller receives orientation changes like pitch, roll and yaw. Device has camera at the front side to track infrared LEDs. Tracking infrared LEDs makes it possible to calculate position changes of controller.

Figure 4.10 shows controller in detail.

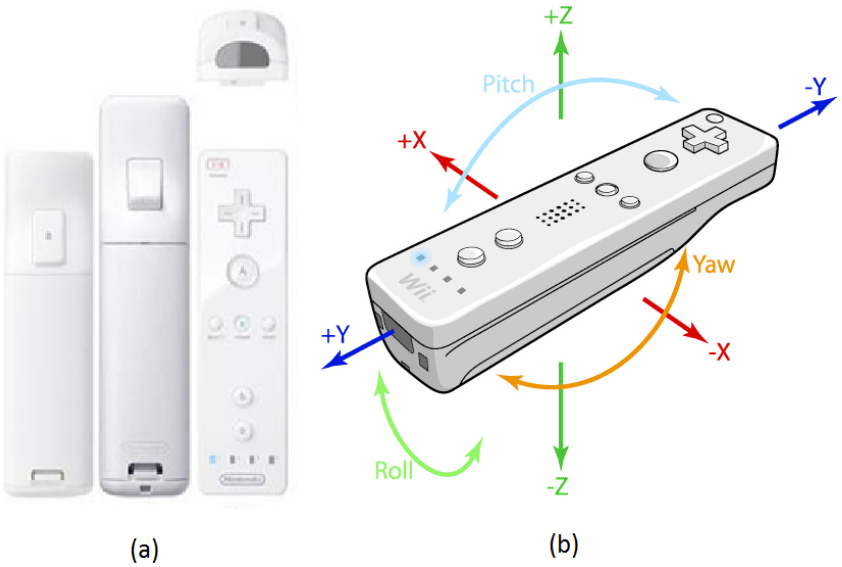


Figure 4.10. Wii Remote Controller.
(a) Wiimote view top and down (b) Wiimote motion axes

In our work, we used Nintendo Wii Remote as a controller device. Wii supports IR point tracking, bluetooth and integrated accelerometer 6- DOF motions. This control device is often preferred for academic and research purposes because of its high capabilities and open source API.

Schou *et al.* [46] used wii controller for a VR application. They build a virtual theatre and controlled by using remote device. Similarly wii is used for low cost head tracking and as a control device for a VR application and evaluated accuracy of control [47].

4.1.4. TUIO Protocol

Table top systems usually interact with user by touching. Touch systems works with two parts. First part is the touch surfaces that is capable of sensing touch gestures and deliver it to software. Second part is the software that gets input and handles in appropriate way. We have used a library that provides API for multi-touch surfaces and especially table top systems named TUIO [48].

TUIO is a library that aims to build a standard for table-top systems. It has both client and server sides to implement API. Server side gets input from device and delivers it

to the client application. Library has many implementations and open source. Figure 4.11 shows structure of a TUIO application.

Figure 4.11. TUIO protocol scheme

TUIO is a protocol that aims to standardize communication between touch surfaces and software that interprets data. In our theses we implemented TUIO protocol in order to use table top system display as a multi-touch surface. Basically, we use a wiimote, and one third party application to achieve our goal. In our thesis, TUIO works as described in figure below.

Figure 4.12 shows TUIO and wiimote usage as a touch surface.

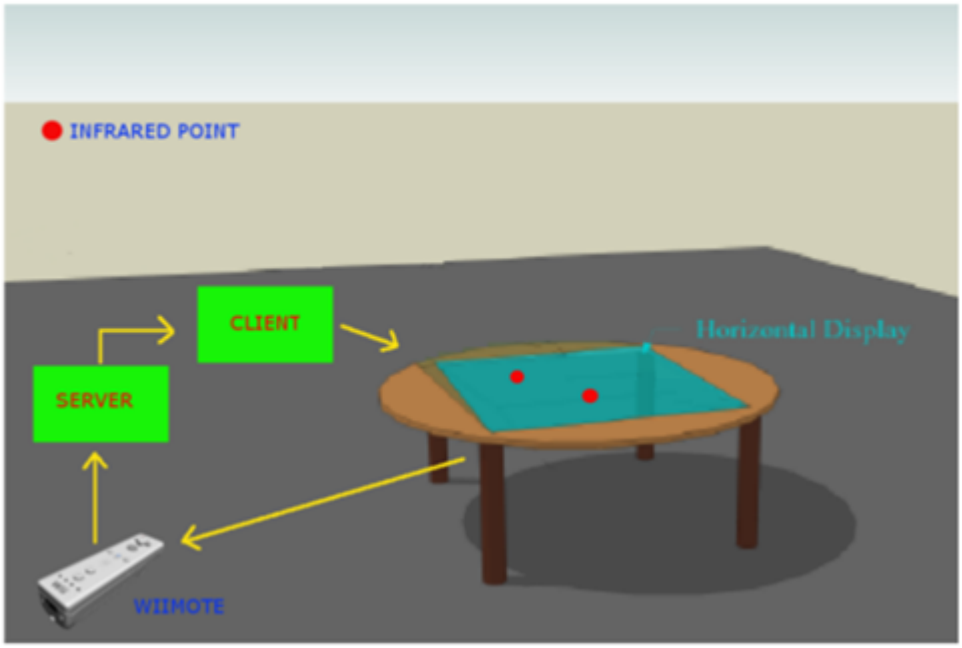


Figure 4.12. RWB and TUIO multi-touch setup

In our thesis, we implemented TUIO client application to software framework. As we mentioned before wiimote can track infrared points. Either wiimote or tracked infrared points can be mobile. In this input setup, we mobilize our infrared points and treat them as

touch surface pointers. In order to do that we use an infrared LED pen. Figure 4.13 shows infrared pen we use. In our setup, there is a third party application that is connected with wiimote which has TUIO server implemented. Server application sends tracked infrared cursors by wiimote to the client application. We receive cursor positions and interpret them as needed.

Figure 4.13. Infrared Pen

In order to receive infrared points as cursors we need to convert them to screen coordinates. In our approach, wiimote application makes interpretation and sends suitable cursor positions with our application. When we first initialize system, we calibrate setup and tell wiimote program our touch screen corners. Then program maps real coordinates to a rectangle coordinate system which has corners defined as (X, Y) pair. Corners for this rectangle are (1, 1), (-1, 1), (1,-1), (-1,-1). Figure 4.14 shows calibration result of tracking surface.

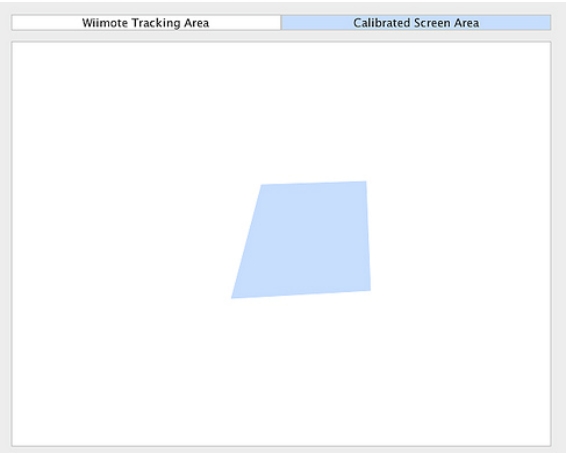


Figure 4.14. Calibrated tracking surface.

Our TUIO method has three benefits for the system. Above those three benefits, with this method, we obtain a touch surface with a very low cost. Advantages are:

- First we can use wiimote alternatively for user input. Not only as a hand held device but also a camera for tracking cursors.
- Second we can use our application as a touch controlled not only on glass display surface but also other surfaces that we project our image onto. By using that we can use a regular table as a touch surface and use our application.
- Third benefit is since we implemented TUIO in our framework; we can use any other cursor tracking methods other than wiimote and immediately use it with our system.

4.2. Software Structure

Software we developed, which is called Interactive Briefing Map (IB MAP), is not only a standalone application, but also a framework that is available for development other applications upon. In this sub chapter, we will explain basic software structure of IB MAP framework and RWB setup for this unique framework.

4.2.1. Framework Structure

IB MAP framework has a modular software structure. A singleton class called Mediator is used to build all main components of the application. Mediator is used to control system component managers. There are five managers:

- Terrain Manager
- Camera Manager
- Culture Manager
- Wiimote manager
- TUIO manager

Developers can add additional managers in order to develop new applications or make enhancements to the IB MAP application. These managers handle input / output streams or organize / process data.

In order to add different capabilities to the IB MAP software or to entirely change it, it is very is to do after completing a few steps. A developer must take these steps:

1- Create a class and define an instance inside the manager class Mediator.
2- Initialize newly defined class in Mediator's constructor.
3- If further initialization depends on another object, initialize in main function before draw loop.
4- In draw callback function, call new objects draw function
5- If input handling is required, GLUT input callback functions can be modified as wished
6- Wiimote controls can be used by modifying and calling newly created object's methods
7- Camera manager class and methods can be modified in order to change camera behavior.
8- Head tracking is bound to camera model and can be detached and used as wished. Head tracking is working independent from object or class that is being drawn.
9- It is important to remember that wiimote input handling is running in a different thread.

Figure 4.15 shows software organization from object orientation aspect.

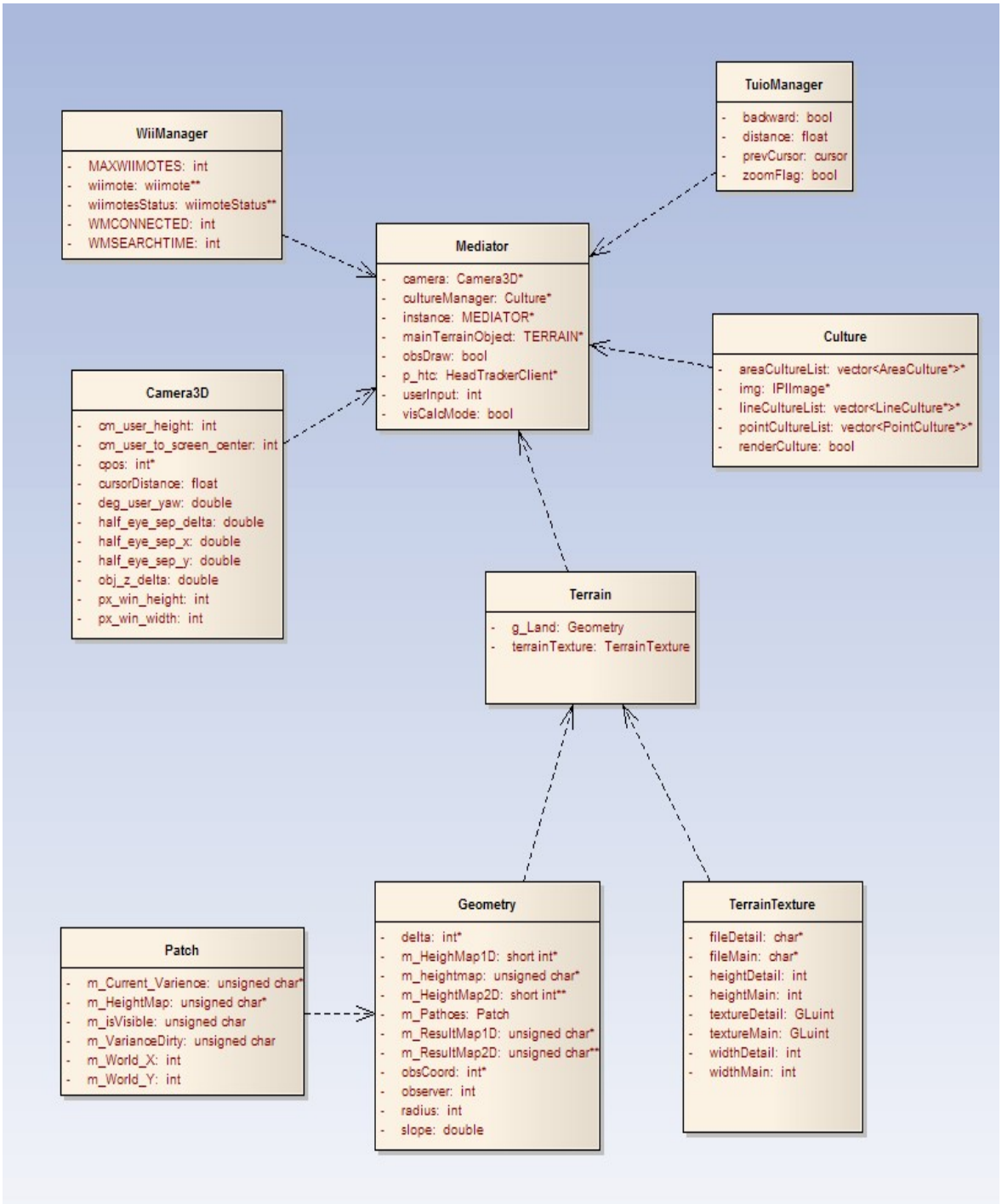


Figure 4.15. Software organization of IB MAP

The IB MAP framework is implemented as a multithreaded application. When the framework starts, it initializes a mediator singleton and initializes managers that are registered to mediator class. After initializing managers, framework branches into two threads. For this purpose we have used POSIX Thread library, also known as pthread, which is originally a Linux library.

POSIX library enables to create, destroy and manage threads. Threads rather than processes are used because threads are easier for OS to handle. When we compare process handling workload between thread handling, it can be observed that threads are easier to manage. Figure 4.16 shows workload comparison between threads and processes for Unix OS [49]. Table shows timing results of thread and process creating routines. 50.000 processes / threads created and evaluated with time utility without optimization flags. Values are in seconds, hence lower is better.

Platform	fork()			pthread_create()		
	real	user	sys	real	user	sys
AMD 2.3 GHz Opteron (16cpus/node)	12.5	1.0	12.5	1.2	0.2	1.3
AMD 2.4 GHz Opteron (8cpus/node)	17.6	2.2	15.7	1.4	0.3	1.3
IBM 4.0 GHz POWER6 (8cpus/node)	9.5	0.6	8.8	1.6	0.1	0.4
IBM 1.9 GHz POWER5 p5-575 (8cpus/node)	64.2	30.7	27.6	1.7	0.6	1.1
IBM 1.5 GHz POWER4 (8cpus/node)	104.5	48.6	47.2	2.1	1.0	1.5
INTEL 2.4 GHz Xeon (2 cpus/node)	54.9	1.5	20.8	1.6	0.7	0.9
INTEL 1.4 GHz Itanium2 (4 cpus/node)	54.5	1.1	22.2	2.0	1.2	0.6

Figure 4.16. Comparison of thread and process workload on Unix OS [49]

After program is branched into two threads, one thread deals with OpenGL drawing routines that are used with GLUT library. Not only drawing routines are processed but also callback functions that are registered to GLUT start to run within this thread.

The second thread handles input / output routines that are registered to the thread routine. Wiimote, head tracking and TUIO can be registered to this thread using the developer structure. The developer can also register other application specific routines to this thread. (see Figure 4.17).

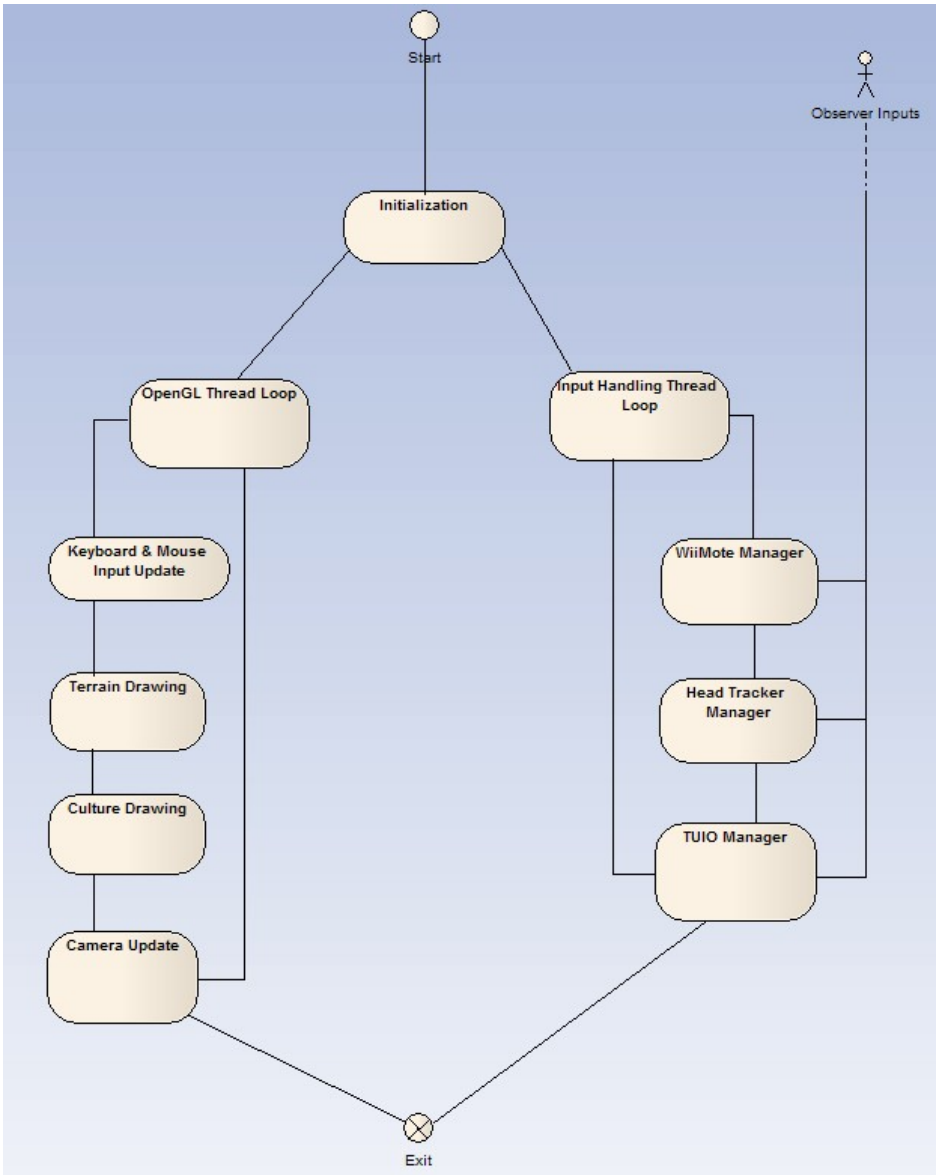


Figure 4.17. IB MAP Thread Scheme

4.2.2. RWB and IB MAP Setup

RWB is a different platform than a personal computer for viewing purposes. Main idea is to model 3D environment according to user position and work with real world dimensions as much as possible. For example, camera in OpenGL environment must model the user's head in real world. Head tracking system must calculate distances and angles correctly according to the user position, RWB position, both RWB and user dimensions and relative positions according to each other. Stereo viewing in RWB makes us to consider some more calculations and adjustments. Zero parallax placements must

coincide with the surface of the RWB. This allows us to view items either inside the RWB or outside the RWB. Wrong displacement can cause undesired viewing results or disappearing of the stereo effect. Figure 4.18 shows RWB setup.

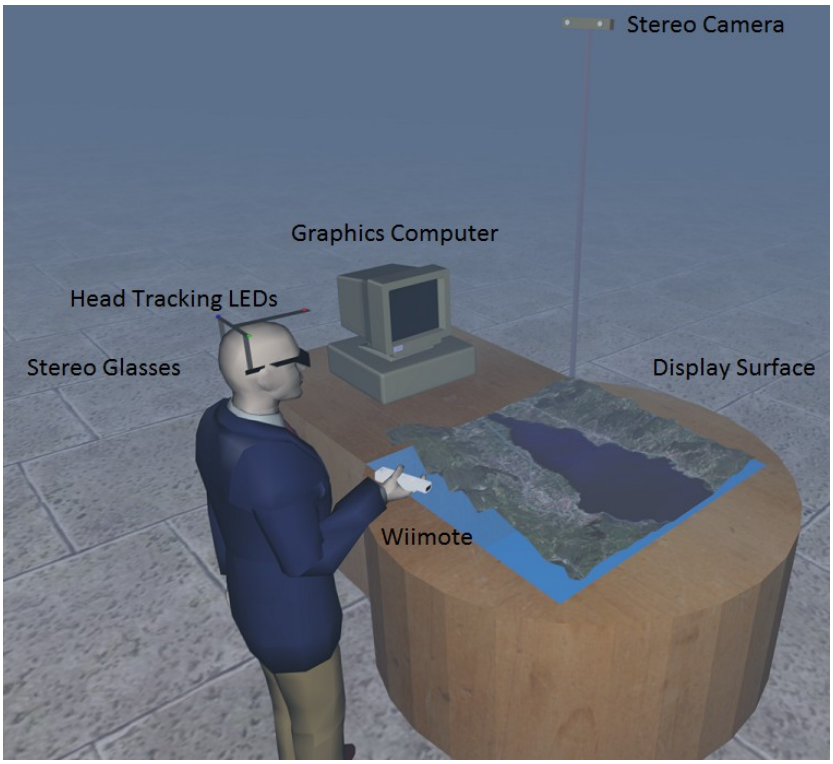


Figure 4.18. RWB Setup

One of the most important properties of IB MAP is to be able to zoom in and out of the terrain. In RWB system, however, it is not same as moving camera closer to the terrain or moving out of terrain. In a VR system that consists of a head-tracked display or a stereoscopic display zooming by translation works poorly [50]. In order to apply an appropriate zooming method, we apply scaling to the object that is being rendered. Also we are translating terrain in order to roam on terrain freely. We cannot move camera because users head position represents camera in our system. It causes inconsistencies between head tracking and camera position values.

4.3. IB MAP Data Preparation and Calibration

In order to run IB MAP application first, we must provide data needed and calibrate for correct viewing. Application won't work smoothly if we do not apply required calibration or data.

IB MAP software needs to be calibrated according to the data we want to view. Application takes 2 main files to work on. Image file and elevation file. Elevation file must be in RAW format because IB MAP has parser for RAW format. Other elevation formats, like DTED, DED or DEM, can be converted easily with 3rd party programs to RAW file format. For example Global Mapper is a handy GIS tool to convert GIS data to other file formats like image files. Afterwards Adobe Photoshop can take image data and convert it to RAW format. On the other hand, image file format can be several of common types. JPG, BMP, TGA, DDS or HDR can be used as image file for IB MAP.

There are a few calibrations that need to be done before we can use our data. We must define our image data and elevation data to be the power of two as a square. We can define our elevation data as 1024 pixels, 2048 pixels or so on. If our data is not power of two by default, we must scale it and recalculate pixel to pixel measure in order to calculate lengths correctly on IB MAP. We are doing this to simplify LOD calculations. After we define elevation size we must define necessary LOD patch size for each side of the square. Figure 4.19 below shows slicing for LOD.

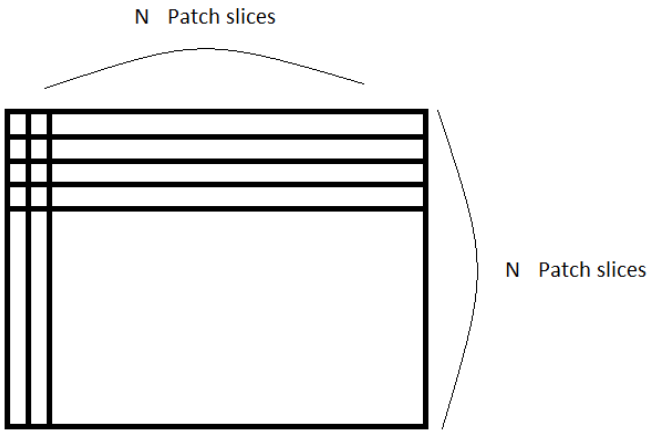


Figure 4.19. Slice representation of height map

Patch slice number is important for LOD performance of application. If we give slice number so small, we will end up with too many patches and calculation will take so much time thus decreasing performance. On the other hand, if we define slice number per side too small, we won't have necessary patch side to calculate visibility correctly. Since visibility is calculated for per patch center, some patches will be culled out even if they fill some of the viewing frustum. Also we can set height scale multiplier if we want to increase height of the terrain.

After we prepare our data to be viewed, we can change IB MAP options to view our data sets correctly. IB MAP application reads some initial values from an INI file. This file can be changed in order to maintain options we needed to set up. Description for INI file is given below in table.

Table 4.1. IB MAP INI file description

##Offaxis	true
##Stereo	true
##PixelMeasure	1.0
##Viewtype	RWB
##GridColor	-1 -1 -1
##Lighting	false
##RWBScaleValue	0.1
##PCScaleValue	0.3
##HeightMapPath	./Media/Height1024.raw
##HeightMapSize	1024
##OrthoImagePath	./Media/texture1024.jpg
##OrthoImageSize	1024
##DetailMapPath	./Media/detail.jpg

Description file can manage several parameters about the software. These parameters are:

- Off axis : Determines if application will run with offaxis frustums or parallel axis frustums. True is for offaxis enable mode and false is for parallel axis enabled mode.
- Stereo: Determines if application will run in stereoscopic mode or in monoscopic mode. True is for stereo and false is for mono modes.
- Pixel Measure: Determines the length that each pixel represents. Used to measure lengths in line culture tool

- View type: Determines if the application camera is adjusted for normal PC view or RWB view. Can take RWB or PC parameters.
- Grid Color: Determines the color of grid on screen. Values show red, green and blue values respectively. -1 value for each color means grid is same color as terrain
- Lighting: Determines if lights in the application will be enabled. True is enabling false is disabling parameters.
- RWBScaleValue: Determines the scale value of the viewed object for RWB viewing.
- PCScaleValue: Determines the scale value of the viewed object for PC viewing.
- HeightMapPath: Determines the input path of the given height map. Must be relative to the application root.
- HeightMapSize: Determines the size of the given height map. Can be powers of 2.
- OrthoImagePath : Determines the input path of the given image. Must be relative to the application root.
- OrthoImageSize: Determines the size of the given image.
- DetailMapPath: Determines the input path of the given detail image. Must be relative to the application root.

After we execute application, we need to press wiimote's 1 & 2 buttons simultaneously in order to open wiimote and connect it with the application.

After connecting wiimote, we can change camera position by mouse. But arranging invalid looking position different than where we stand will distort stereoscopy effect. In order to prevent it we may use predefined 5 different angles for predefined user positions. However system will not be updated in camera position manner, so we have to perceive stereoscopy from a single spot. This can be used to test either stereoscopy is working or not. Figure 4.20 below shows predefined positions. Each time we press keyboard '1' key, starting from position 1 we travel all 5 looking positions and return to first looking position.

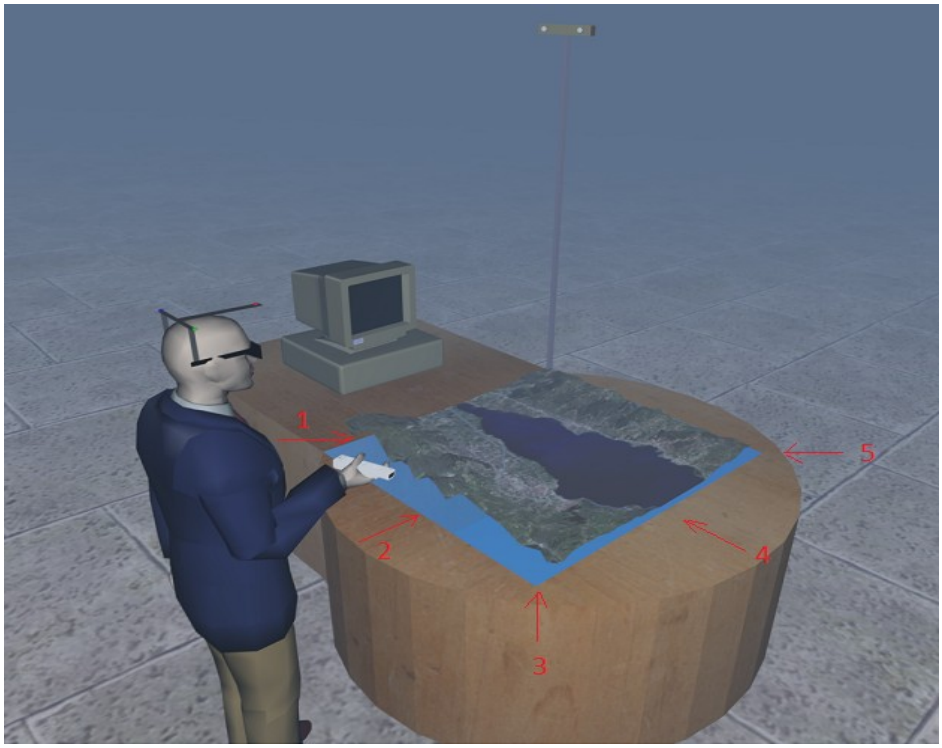


Figure 4.20. Predefined look positions

If we want to enable head tracking in system, we press key ‘p’ to connect head tracking. Then we calibrate head position by clicking led images that appear afterwards. We can than move around the table and preview the terrain.

If we want to enable TUIO port listening for cursor data, we just press button ‘o’ to activate TUIO port 3333 connection and update camera position from the data that comes from TUIO server.

We can also extract or view culture features. We can work with 3 types of culture feature. In order to extract point feature, press button ‘l’. A new feature visualization window appears after pressing ‘l’ button. We can click places on window to place point features and after we are done, simply click middle mouse button to observe features in 3D window. If we want to extract line feature, we press on button ‘k’. A new window will appear for culture extraction. By clicking the control points that our line feature will pass, we create the line feature. After completing extraction, simply click on middle mouse button to view feature on 3D window. Length calculation results will appear on console window. If we want to display area features, we just press button ‘j’ to open area feature

visualization window. As we did in line feature we click on image and when we are done, this time we click right mouse button to enclose the field and then click middle mouse button to display feature on 3D window.

We can change stereoscopy parameters on runtime or translate viewing object. Some buttons on keyboard helps us to manipulate RWB. These buttons are:

- Buttons ‘z’ and ‘x’ will increase and decrease eye separation of stereo mode respectively.
- Buttons ‘c’ and ‘v’ will increase and decrease z height of object respectively.
- Buttons ‘q’ and ‘e’ will shift camera forward and backward respectively.
- Buttons ‘d’ and ‘a’ will shift camera left and right respectively.
- Buttons ‘w’ and ‘s’ will zoom in and out respectively.

We can also navigate terrain with wiimote using arrow keys. If we want to zoom in or out of the terrain, we can press buttons on wiimote that are ‘+’ or ‘-’. We can return to original position by pressing ‘home’ button. (see Figure 4.21).

Figure 4.21. Wiimote buttons

When using RWB, we can use analysis tool to test terrain data for terrestrial element placement. After connecting wiimote, when we press button ‘A’ we can go into analysis mode. By help of arrow buttons we place element and pressing button ‘B’ (button on the back) we analyze terrain. After pressing button ‘A’ again we get out of analysis mode and

we can even clear analysis effect on terrain by pressing button ‘B’ again. Buttons ‘1’ and ‘2’ decreases and increases radius of the element respectively. Buttons ‘+’ and ‘-’ increases and decreases element height in analysis mode respectively.

5. APPLICATION: INTERACTIVE BRIEFING MAP

Interactive Briefing Map (IB MAP) is a workbench application that allows 3D terrain visualization. Terrain on workbench appears as a physical mock up model. Stereo illusion of system increases perception of the terrain data. User can move around the model and view it independent from position. We can control program from mouse, keyboard, Wiimote and head tracker. Program not only allows us to view terrain as we like, but also it has ability to place terrestrial objects on terrain in order to calculate and view LOS coverage area on terrain in real time such as base stations, sharp shooters or radars.

Moreover IB MAP is not only a standalone application, but also is a framework for the RWB that can organize control of several inputs with a drawing capable interface. Any developer can change method implementations and/or create their classes/objects to reuse application as they like.

Application is originally developed for RWB that is stationed at Boğaziçi University. Since RWB is a stereoscopic device, IB MAP also designed to work with stereoscopic or monoscopic viewing modes. Stereoscopic mode works with shutter glasses. Program is explained in more detail in the following sections.

5.1. User Interface of the IB MAP

IB MAP application has a simple interface that shows us the terrain as a whole. Main program screen is an OpenGL window which also greets the user. By default we display a textured terrain. (see Figure 5.1). Terrain can be displayed either in side view or top view. Side view is more usable with vertical monitors. On the other hand, top view is very well adjusted for RWB.

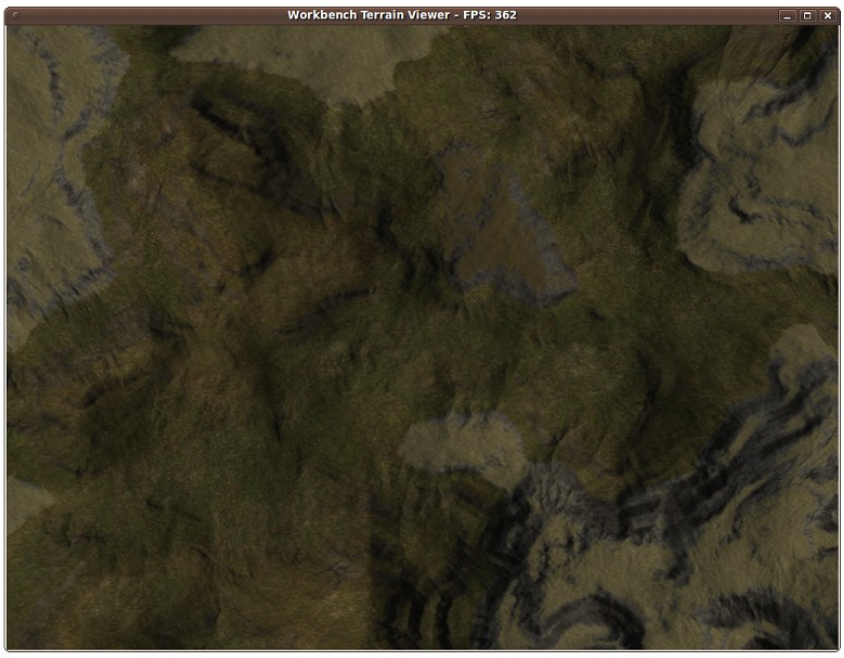


Figure 5.1. Initial start of IB MAP application

Figure 5.2 shows IB MAP viewing states according to different screen setups.

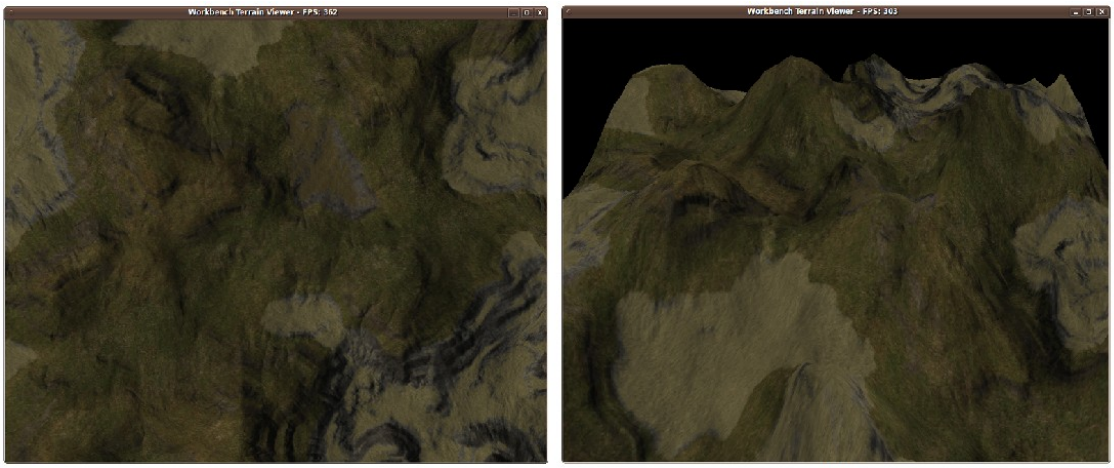
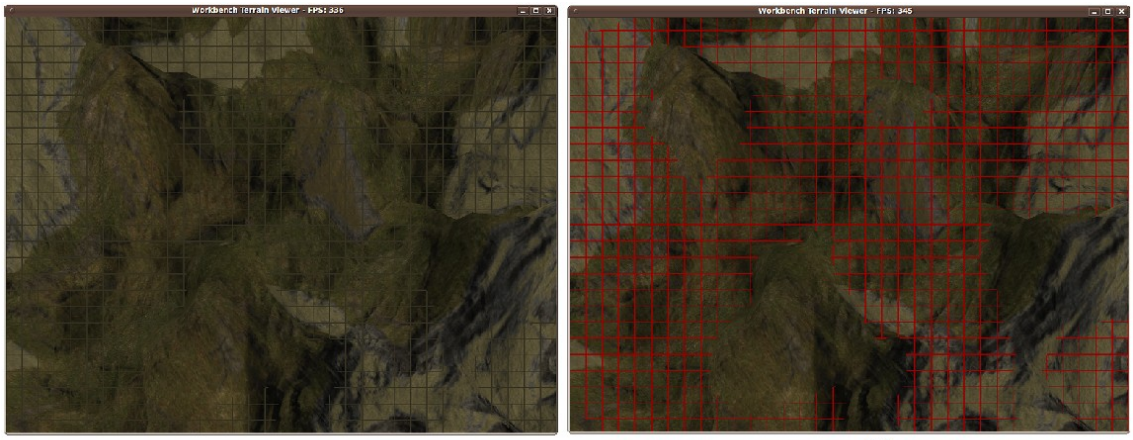


Figure 5.2. IB MAP views (a) Top view of IB MAP, (b) Side view of IB MAP

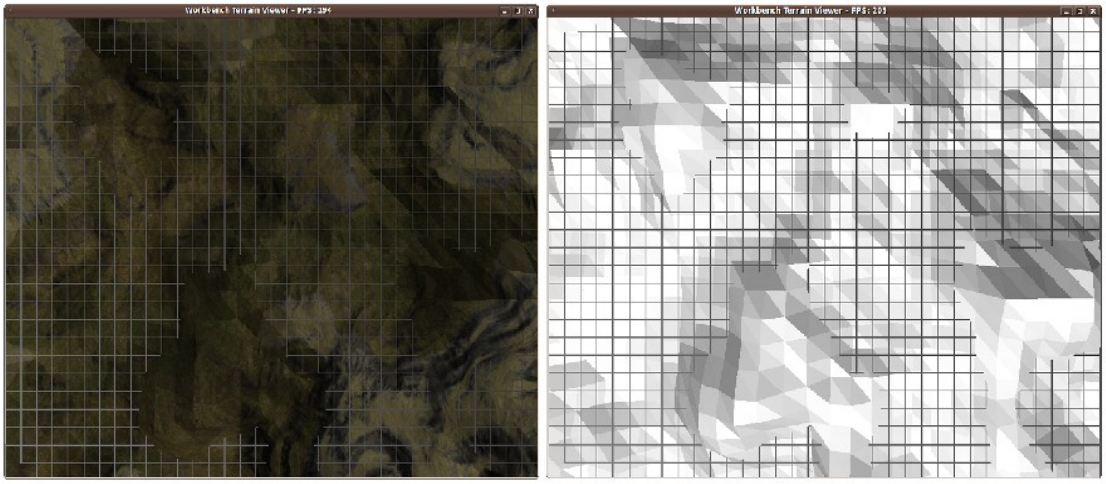
When we display terrain in top view mode, we can display a grid to help us measure distances on terrain relatively. Also we can change color of grid whatever color user feels comfortable with. (see Figure 5.3).



(a) (b)

Figure 5.3. Grid in IB MAP (a) A grid in grey, (b) A grid in red

IB MAP also uses OpenGL lights. User can toggle on and off lights that is activated within the scene. Figure 5.4 shows when lights enabled within the scene. On the contrary,



(a) (b)

Figure 5.4. Lights activated in IB MAP (a) texturing enabled (b) texturing disabled

Figure 5.5 shows lights disabled within the scene. There are textured and no textured views for a better comparison.

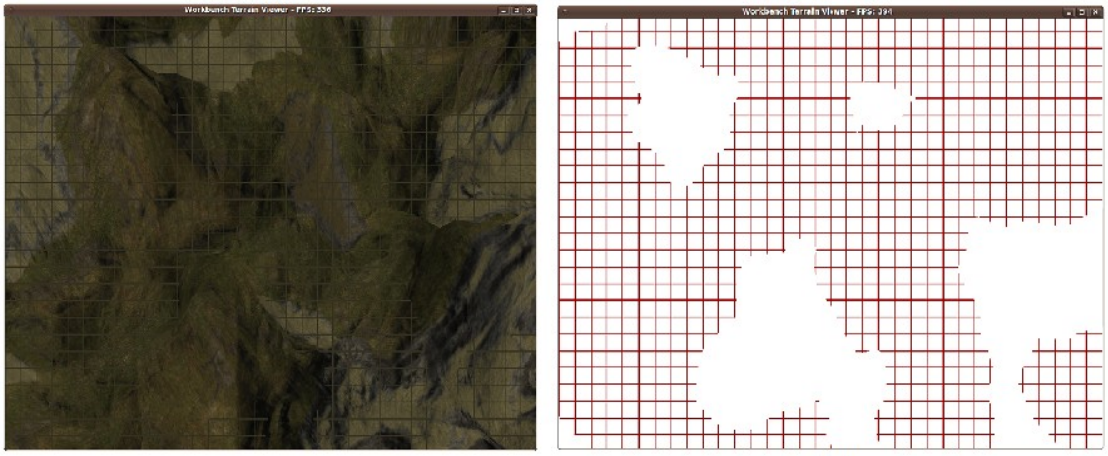


Figure 5.5. Lights deactivated in IB MAP, (a) texturing enabled (b) texturing disabled

In order to add lighting effect on terrain, first we need to calculate terrain polygon normals since OpenGL uses Groud Shading which requires polygon normals at each polygon vertex. (see Figure 5.6). We calculate each terrain polygon normal right before polygon is constructed. That is because, in IB MAP system we use dynamic continuous LOD algorithm and scene is reconstructed at each frame displayed. Since our terrain consists of triangles, we use corners of triangle to calculate each normal per vertex for a triangle right before it is drawn. When we have 3 points in space we can calculate 2 distinct lines or vectors. Cross product of these vectors gives us another vector that is perpendicular to other 2 vectors. When we unitize result vector, we find vertex normals for vertexes of our processed triangle polygon. This calculation allows us to implement flat shading on terrain.

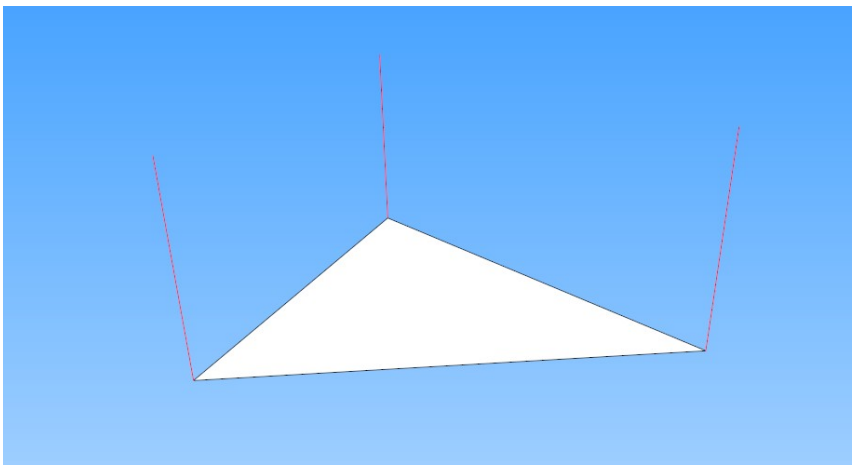
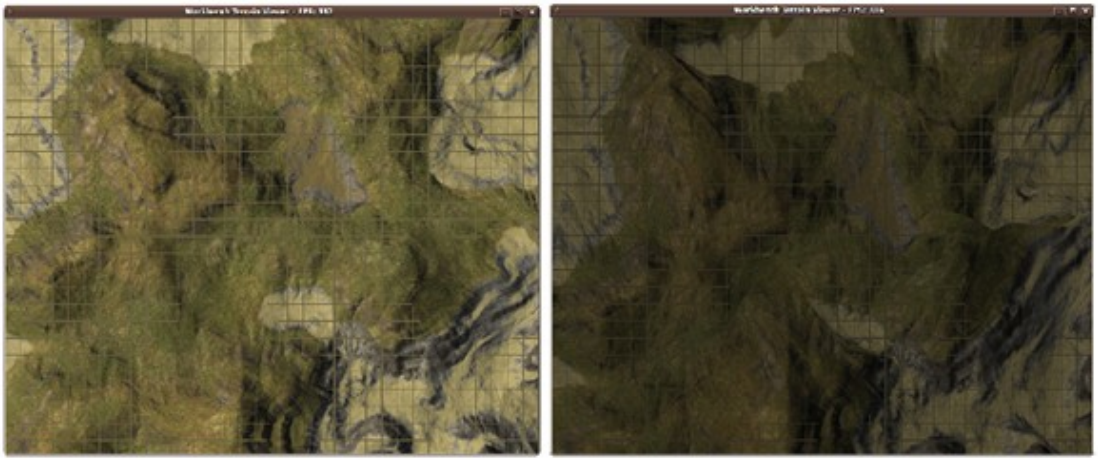


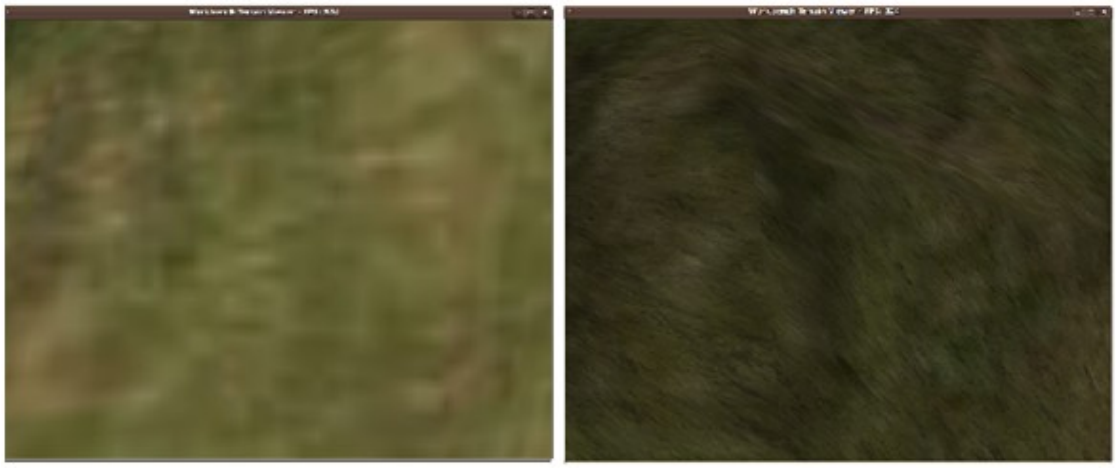
Figure 5.6. Normals of a triangle polygon

IB MAP employs, multi-texturing to increase realism. Multi-texturing is using more than one texture on the same polygon. By this method, we add more detail to the terrain without increasing polygon numbers. OpenGL supports more than 8 levels for multi-texturing. IB MAP includes 2 texturing levels for more detail effect.

Figure 5.7 illustrates the effect of multi-texturing. In Figure 5.7 (a) terrain is shown without multi-texturing. It is obvious that not using multi-texturing creates a bright scene. Figure 5.7 (b) shows multi-textured terrain. Contrary to the Figure 5.7 (a), it is darker. Multi-texturing is not very effective for distant viewing of the terrain. But for close up viewing, it increases level of detail. Figure 5.7 (c), and (d) shows terrain with multi-texturing and without multi-texturing closer to terrain ground.



(a) Terrain from sky without multi-texturing (b) Terrain from sky with multi-texturing



(c) Zoomed in without multi-texturing (d) Zoomed in with multi-texturing

Figure 5.7. Multi-texturing effect on terrain.

IBMAP uses mipmapping for texture level of detail. For this purpose, we use an open source texture library for OpenGL named SOIL. OpenGL supports mipmaps and graphics card preferences let us choose filtering level according to our choice of performance.

In modern GIS systems and applications standard data sets elevation data. A major component called culture features in addition to texture images and culture features represent a large spectrum of information in GIS systems [51].

Culture features are generally referred as the vector data. Vector data is used to represent almost anything on earth surface including vegetation, rivers, roads, structures and so on. Most common method to generate culture data is to extract it from satellite imagery. Alternative is to work on the terrain and manually collect GPS coordinates of items that are to be tagged as culture features

In IB MAP, we have implemented a vector extraction tool. This tool allows us to create culture features in vector data primitive format. These primitives are points, lines and area types.

Figure 5.8 shows the interface window for Feature visualization tool.

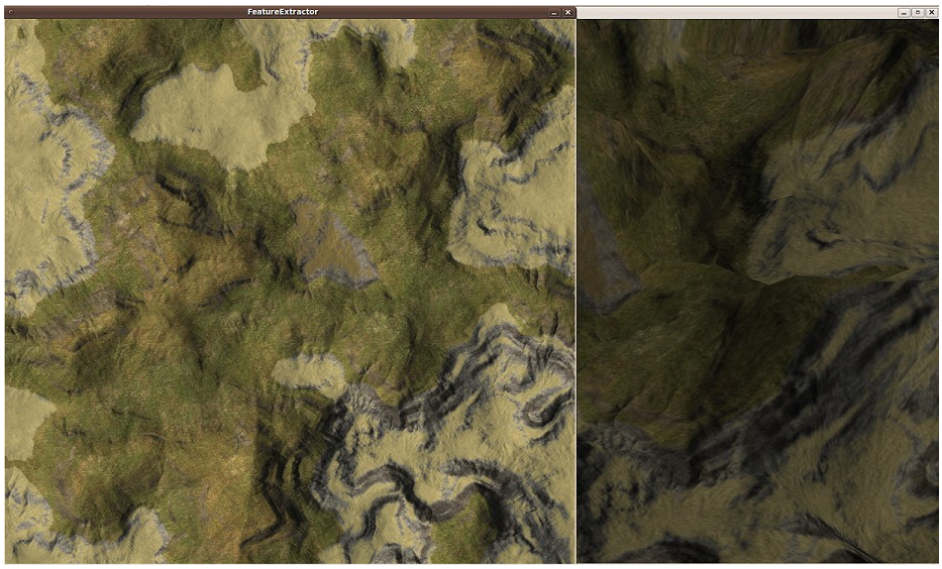


Figure 5.8. Feature Extractor tool interface in foreground. IB MAP interface is shown at background

5.2. Culture Data Creation

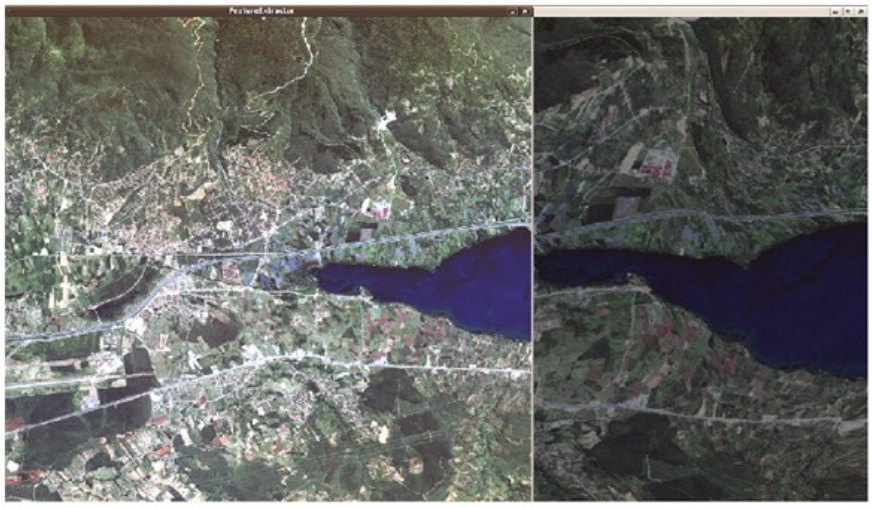
Features over a terrain are very important components of the GIS systems. These datasets have many area of usage. VE modelers that needs to model real world needs these data types to correctly place models over terrain or scientists need these data to evaluate vegetation layer corruption or flood simulation analysis if a dam breaks through. Map building, city planning, construction engineering and many working areas need these data sets in order to work with actual data.

In our thesis, we implemented a feature visualization tool as well. This tool helps us to create our own vector data in RWB coordinate system. We can create culture datasets, display them and save or load them. Culture data can store terrain features in order to display or work with them. In industry standard culture files, like SHP or DFD, culture components are stored within 3 main kinds of primitives. These primitives are:

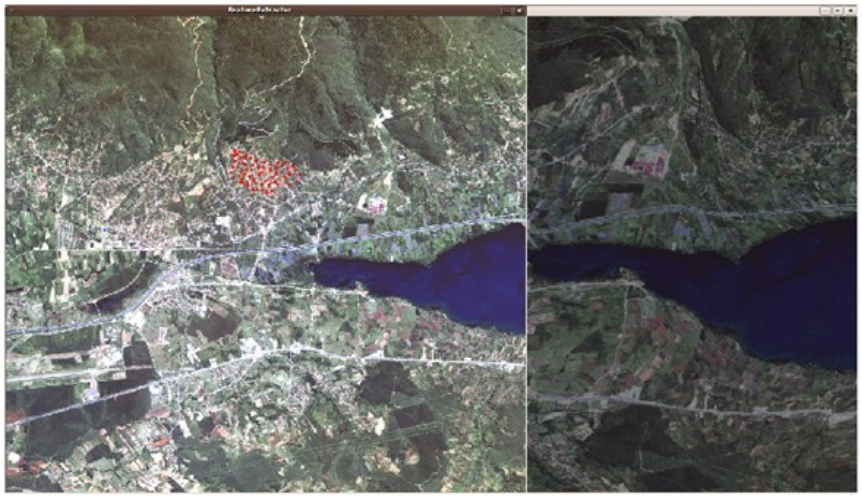
- Point Type Culture
- Line Type Culture
- Areal Type Culture

Point type culture primitive holds singularly definable objects in real world. For example buildings in a city or trees of a forest can be defined singularly. These types are generally used in city planning, road planning and detailed vegetation analysis. In our tool, we open point culture extraction window if we want to create point type of culture over our terrain. When we click on the terrain image, there appears a 2 pixel sized red dot on the map. If we see a red dot on the map, then it means we successfully created a point type culture primitive. There is not a limit of point type culture count in IB MAP software; however system memory limit may cause inconsistencies if too many points entered for systems memory limit.

Figure 5.9 shows point culture entry process steps.



(a) Initial entry window



(b) Entry window after point culture entrance

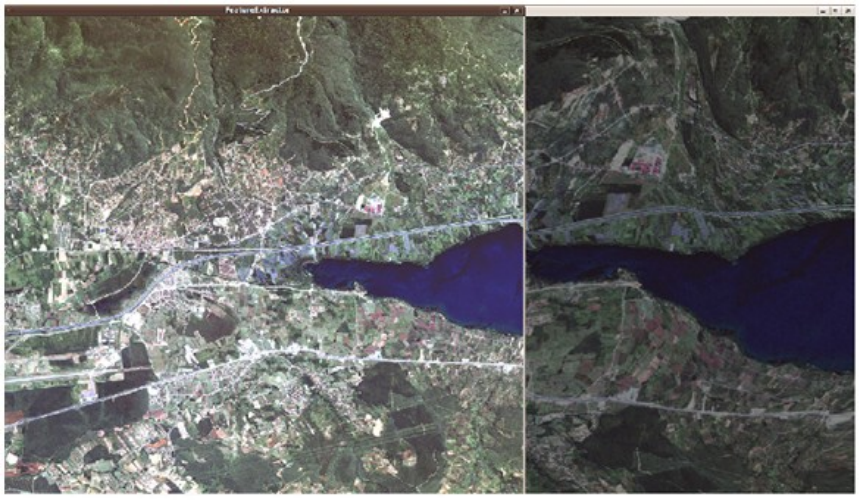
(c) Point culture rendering on 3D after entry process is done

Figure 5.9. Point culture entry process.

Line type culture, on the other hand, keeps information about linear culture features. Rivers, borders, roads and anything that can described as linear shape is included in this primitive culture type.

In our tool, we open line culture data creation window if we want to create line type of culture over our terrain. When we click on the terrain image, there appears a 2 pixel sized red dot on the map. After clicking more dots to create a line, a line shape occurs on 2D map of terrain. If we see a red line on the map, then it means we successfully created a linear type culture primitive. There is not a limit of linear type culture count in IB MAP software; however system memory limit may cause inconsistencies if too many lines entered for systems memory limit.

Figure 5.10 shows line culture entry process steps.



(a) Initial entry window

(b) Entry window after line culture entrance

(c) Line culture rendering on 3D after entry process is done

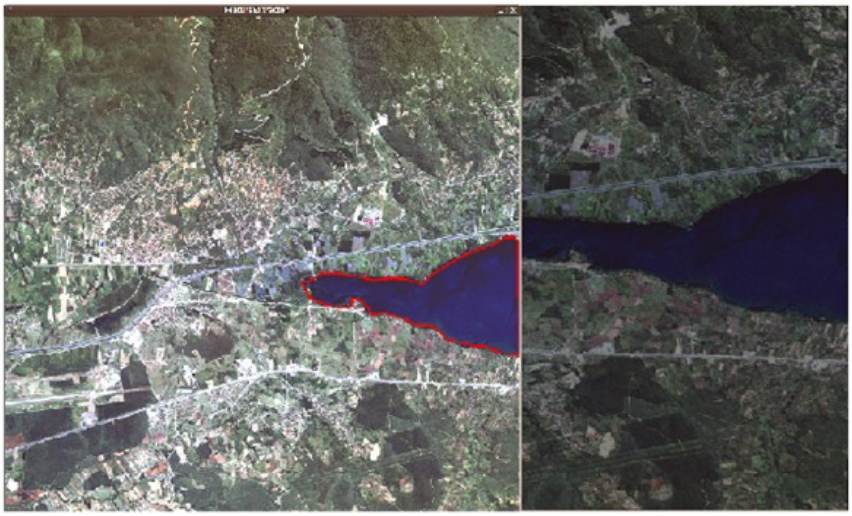
Figure 5.10. Line culture entry process

Area culture is the last kind of our defined vector data primitive. This type keeps information about areal definable objects such as lakes, fields, military zones or forests.

In our tool, we open area culture extraction window if we want to create area type of culture over our terrain. When we click on the terrain image, there appears a 2 pixel sized red dot on the map. After clicking more dots to create an area shape, a line shape occurs on 2D map of terrain. After completing our area lines, we right click to enclose the area we defined by lines. If we see a red enclosed area shape on the map, then it means we successfully created an area type culture primitive. There is not a limit of area type culture count in IB MAP software; however system memory limit may cause inconsistencies if too many areas entered for systems memory limit.

Figure 5.11 shows areal culture entry process steps.

(a) Initial entry window



(b) Entry window after area culture entrance

(c) Area culture rendering on 3D after entry process is done

Figure 5.11. Area culture entry process

We can use these features to examine terrain in more detail. We can enhance perception of certain features by creating vector data of these features. Either for militaristic purposes or research purposes, these datasets makes easier the comprehension of the terrain features.

In line extraction tool, we calculate total length of the line feature for eye's bird length and real length on terrain. Line feature visualization tool can also be used to measure distance between two points for eye's bird length and real length on terrain.

While we implement culture extraction tool, we considered if user enters less points according to terrain elevation data resolution. We interpolated user points on terrain to prevent culture polygons to interfere with the terrain itself. (see Figure 5.12).

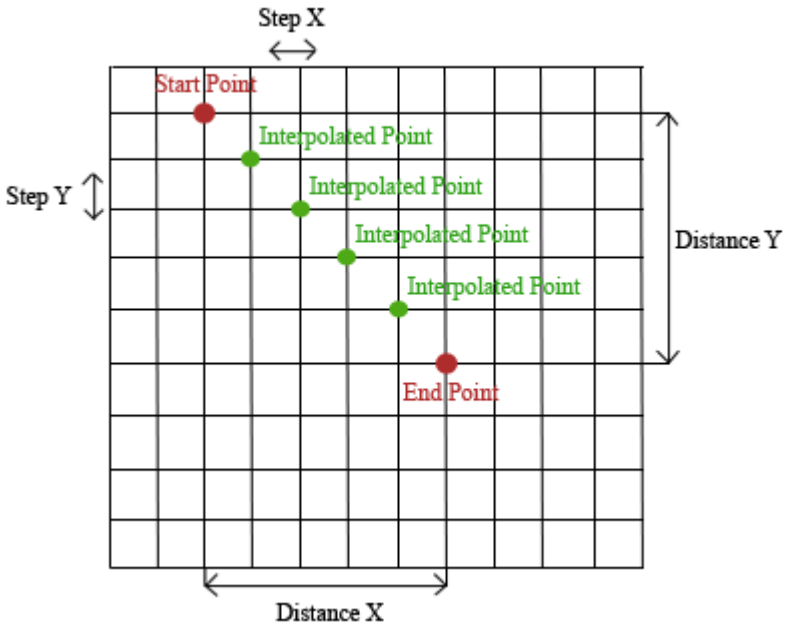


Figure 5.12. Interpolation using terrain data

Interpolation is done by using digital terrain data loaded in application. After user enters points on terrain via line feature or area feature interface, user points are interpolated using an algorithm. Main component of the algorithm is the method to interpolate between given two points. Figure 5.13 shows pseudo code for interpolation algorithm.

```
Input:Two points to interpolate between {P = (P1, P2)}
Define X and Y coordinate difference distance between points.
Define P[N] point list to be returned as a result array.
Set interpolation start point.
if(absolute X difference > absolute Y difference)
{
    Find interpolation step amount according to X difference.
    Find interpolation loop boundaries according to X difference.
}
else
{
    Find interpolation step amount according to Y difference.
    Find interpolation loop boundaries according to Y difference.
}
while i < loop boundaries
{
    Increase interpolation start point by step amount
    Control values and find a new interpolation point
    Push new calculated interpolation point to P[N]
}
Output:  Interpolated point array list of P[N]
```

Figure 5.13. Interpolation pseudo code

Figure 5.14 shows results with interpolation and without interpolation.

(a) Line culture evaluation without interpolation

(b) Line culture calculation with interpolation

Figure 5.14. Interpolation effect over culture calculation in IB MAP

5.3. Line of Sight Evaluation

LOS evaluation is an important part of the GIS applications. It simply calculates if two points on terrain are directly visible to each other. Line of sight evaluation is among most common queries in GIS systems like Global Mapper [52] or ArcGIS [53]. View shed calculation and direct distance measuring are usually the techniques that are employed. Ben-Moshe *et al.* proposed an algorithm to find the visible region from a given point on a terrain [54]. In [55] Franklin *et al.* uses an algorithm that considers observer and target height as additional parameters.

In our thesis, we have used XDraw [56] algorithm. The algorithm approximates LOS of a given point according to given radius. (see Figure 5.15).

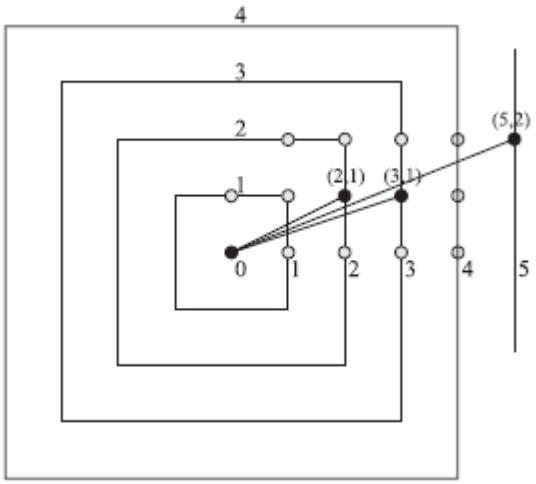


Figure 5.15. XDraw Algorithm Calculation steps [56]

For each point Xdraw algorithm calculates:

a) Whether the point is visible or not
b) If it is hidden, the elevation of a LOS from the observer passing over it.

XDraw algorithm calculates LOS values as levels. First, algorithms computes closest ring to the given observer point. In Figure 5.15, XDraw algorithm is shown in ring calculation manner. Point (2, 1) is calculated as visible from point (0, 0). Also point (3, 1) is evaluated as seen. After algorithm calculates inner rings, it considers previously calculated LOS values to evaluate outer ring points.

Properties of Xdraw algorithm is stated as:

- Algorithm calculates LOS from a given observer to every other point the approximately in the database.
- Xdraw allows both the observer and the target to be a specified elevation above ground level.
- Algorithm calculates the height that the LOS passes over the point if that point is hidden.

Figure 5.16 shows pseudo code for XDraw algorithm.

```
Input:observer coordinates {O = (O.x, O.y)}
Define bounds for calculation with given radius from observer.
Define resultMap 2D array for storing visible points.
Mark observer position as seen on resultMap.
Calculate observer altitude including observer height.
if(Line of Sight can not be calculated)
    return.
while ip < perimeter
{
    Get a start point considering bounds.
    if(Observer is at edge)
    {
        Continue with the next item.
    }
    Start with the most changing value of X or Y coordinates.
    while items are in targeted bound
    {
        Process points in the defined area.
        if(Point is visible)
        {
            Set resultMap as marked.
        }
    }
}
Output:  Line of sight 2D array
```

Figure 5.16. XDraw algorithm pseudo code

In our thesis, we applied a line of sight calculation algorithm. But we simply don't measure if two points can see each other or not. We implemented a different algorithm, called XDraw. Our tool places an observer on terrain, and we calculate which terrain parts this observer can directly view. After we calculate terrain parts, we draw it marked as green on terrain structure. This method has 3 inputs in order to make necessary calculations. These arguments are:

- Observer height
- Observer radius
- Target height

Observer height is the height that user is above the terrain. For example a guarding tower may be 10 meters tall; on the other hand a sharp shooter can be at 0.20 cm above the ground.

Observer radius is the area range that observer can handle. Again, defense artillery batteries may launch over 5 km length but other weapons may have limited range.

Target height is the targets distance above the terrain. If we are looking for a standing person on the terrain, our observer will be looking for a person for a 1.75m tall approximately. If observer is looking for a sharp shooter, this time observer will be looking for a person not more than 0.5m above the ground level.

When we supply these parameters to IB MAP program, we will see a rendered terrain with the visible parts painted as green with red point observer identification. We can then evaluate if the position of the observer is right for us and meets our expectations such as covering enemy's passage direction.

Figure 5.17 shows result of a LOS evaluation for an observer with a given radius of 300m, observer height of 2m and target height of 1m

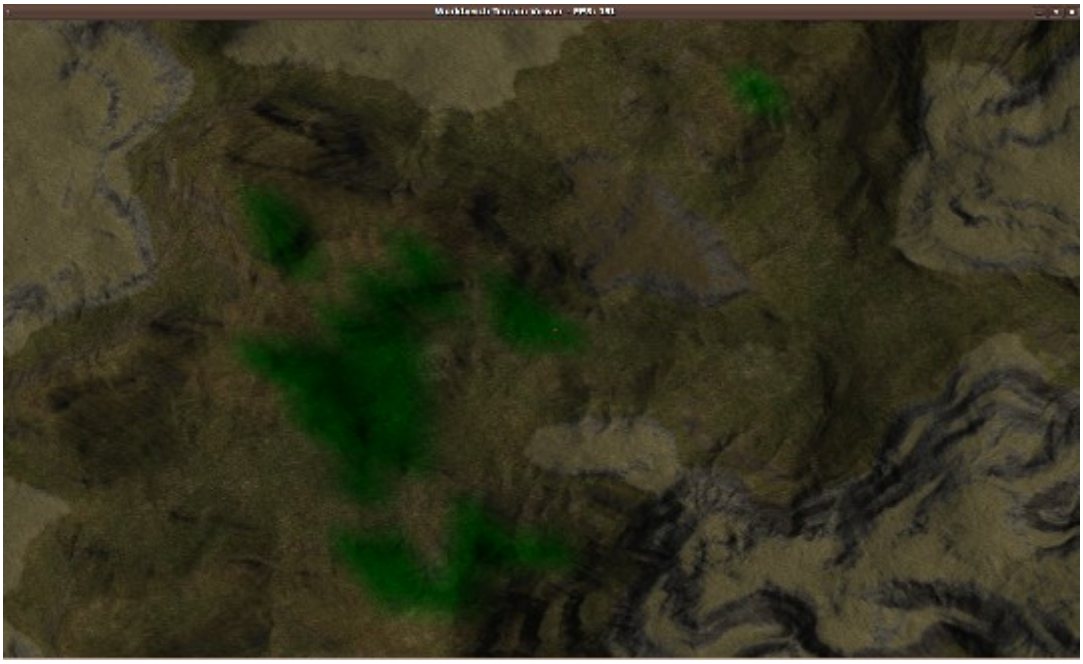


Figure 5.17. LOS evaluation for Observer height = 2m, Observer radius 300m, Target Height = 1m

When we move our camera to the observer's position and look around, we see that algorithm has calculated LOS evaluation correctly and everywhere we can see within our range is painted with green color. Figure 5.18 shows observer view for different viewing angles.

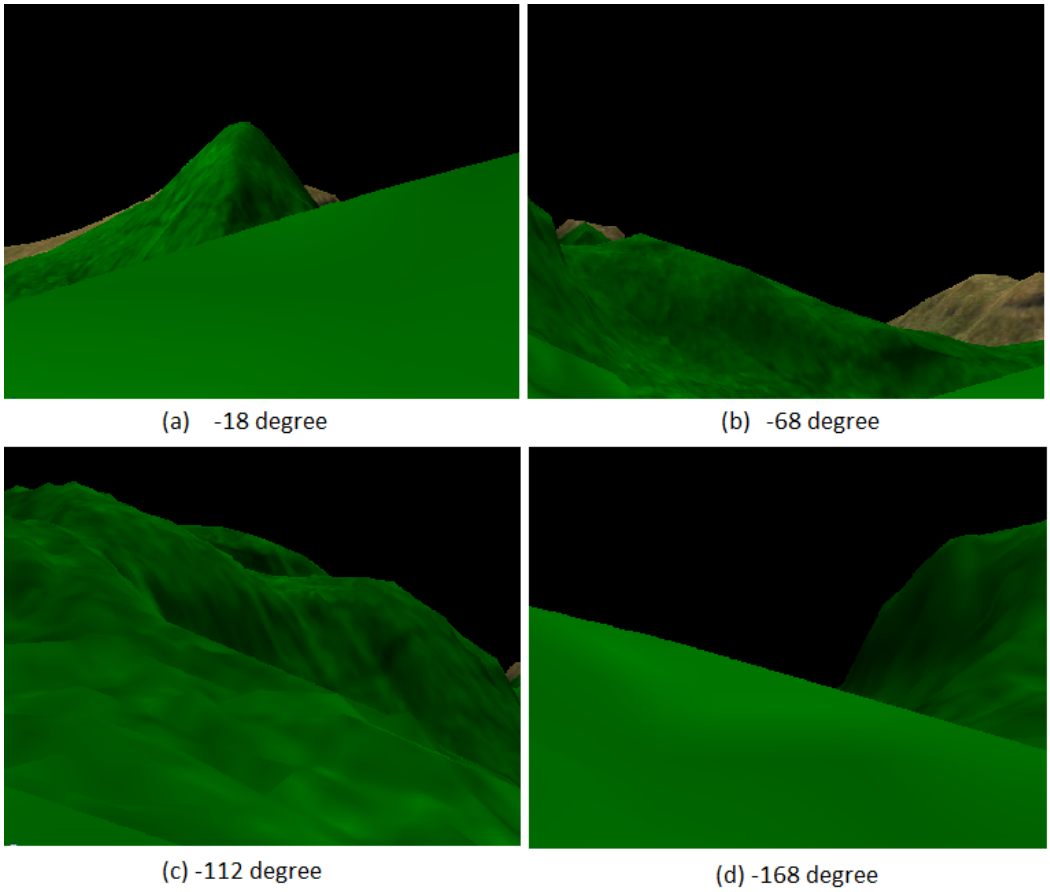
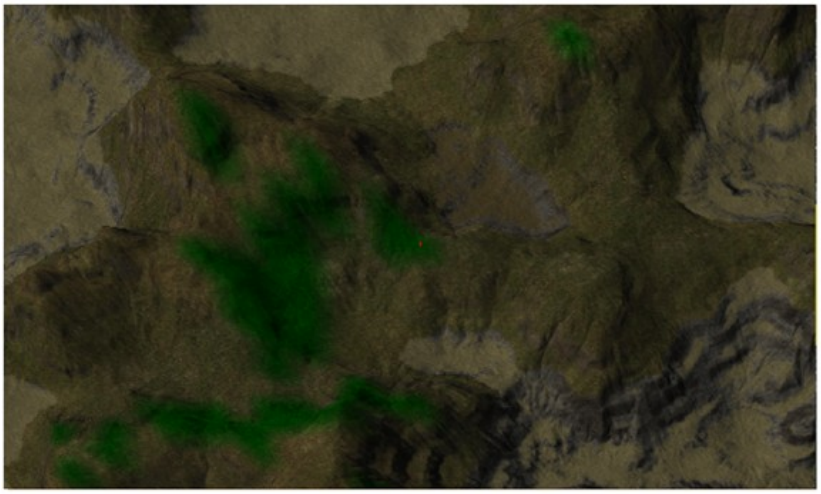


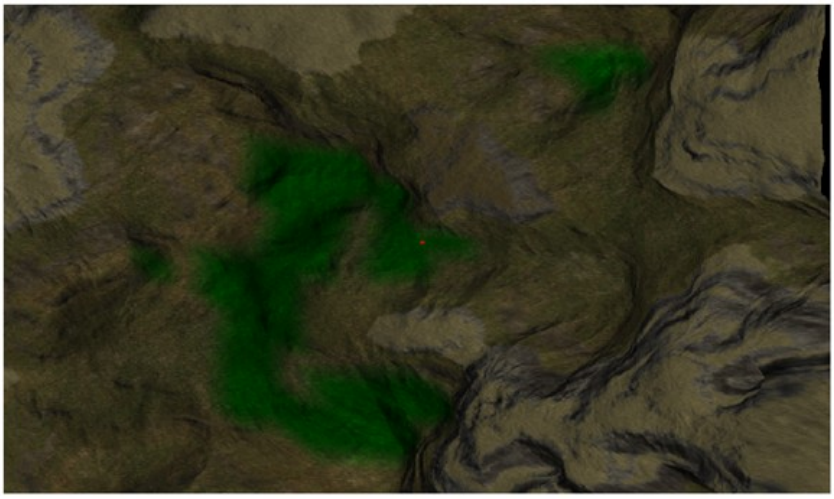
Figure 5.18. Observer position view. (a) Heading is -18 degrees, (b) heading is -68 degrees, (c) heading is -112 degrees and (d) heading is -168 degrees

Figure 5.19 shows result of changing radius, target height and observer height to the visibility calculations according to control group that is shown is Figure 5.17.

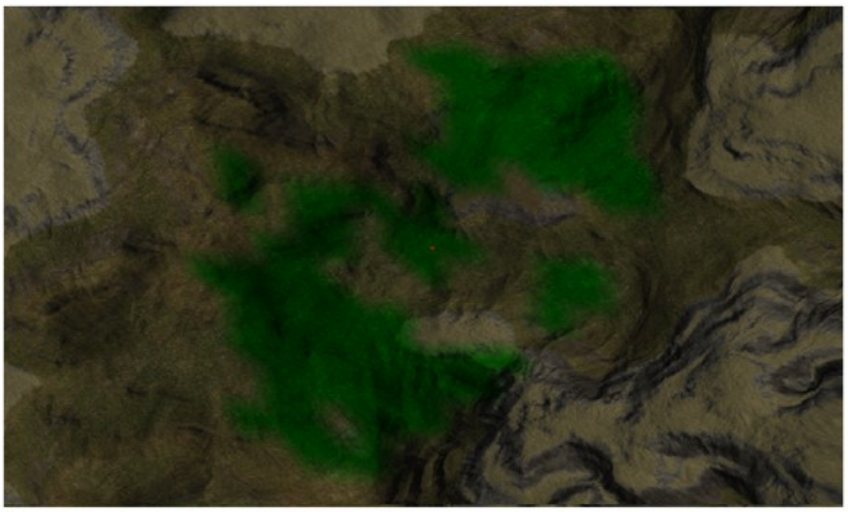
Figure 5.19 (a) is calculated when observer radius is 600m, observer height is 2m and target height is 1m. Figure 5.19 (b) is calculated when observer radius is 300m, observer height is 2m and target height is 10m. Figure 5.19 (c) is calculated when observer radius is 300m, observer height is 10m and target height is 1m.



(a) observer radius is 600m, observer height is 2m and target height is 1m



(b) observer radius is 300m, observer height is 2m and target height is 10m



(c) observer radius is 300m, observer height is 10m and target height is 1m

Figure 5.19. Different input values' effect for algorithm

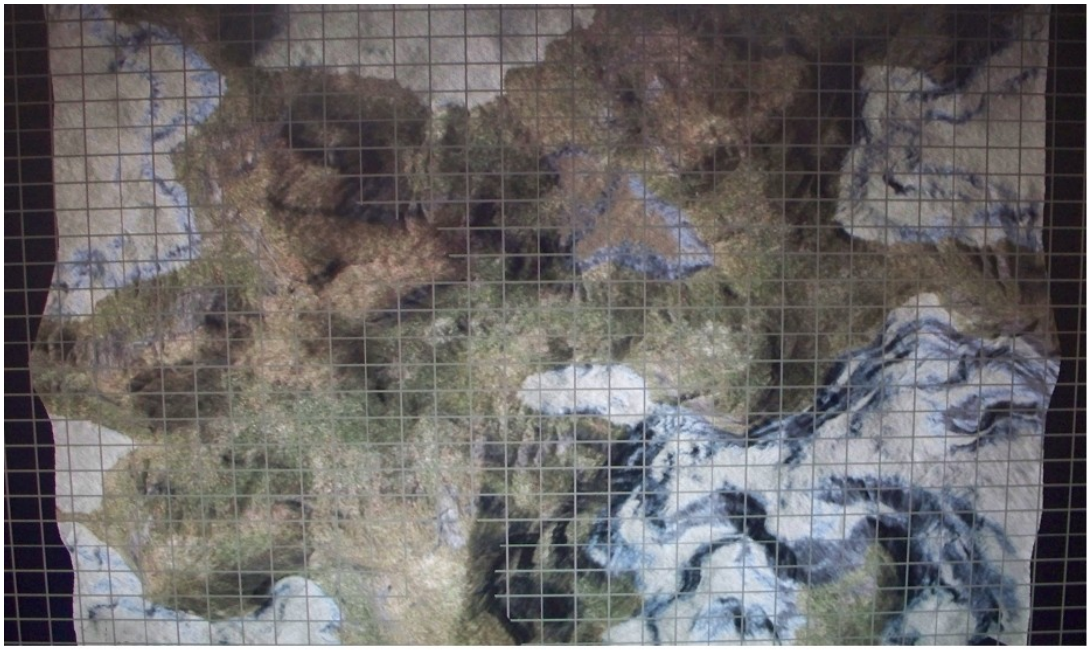
6. EVALUATION

Our framework is developed to run on RWB in stereoscopic mode. Our main concern is to evaluate terrain viewing on RWB in stereoscopic mode. We have evaluated the effect of eye separation parameter used in stereo viewing, static stereo distortion due to head movement without head tracking and dynamic stereo distortion without head tracking. Also viewing the terrain on RWB and evaluating its positive properties compared to classical personal computer usage. Other than that, we have developed a GIS like program with different functionalities to be used on RWB and review its functionalities.

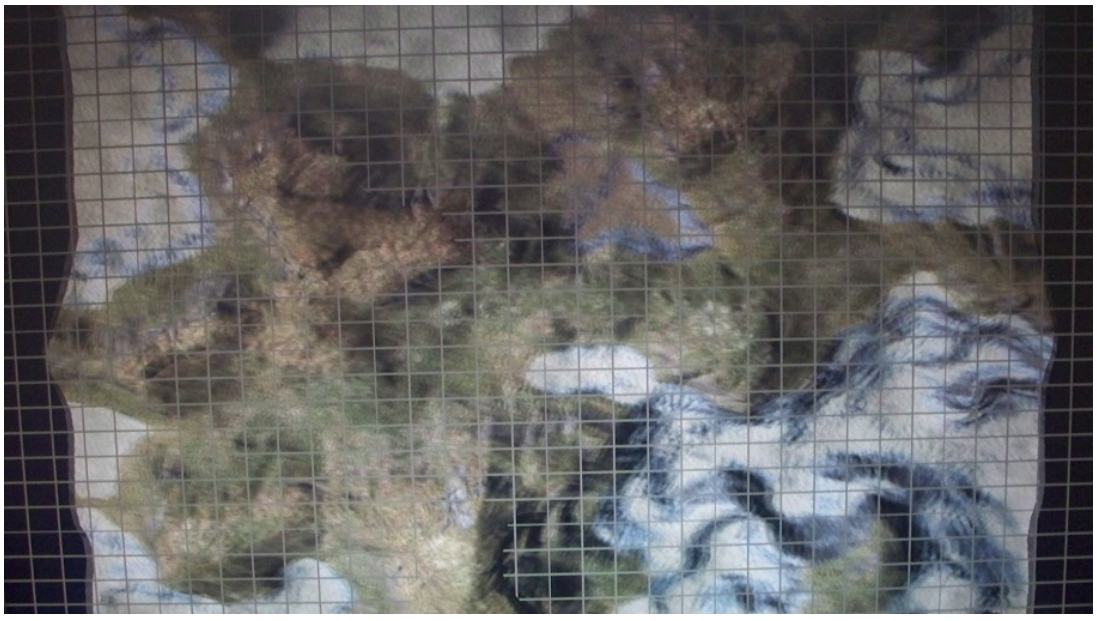
6.1. Eye Separation Parameter Effect over Stereoscopy

In our thesis, we used eye separation parameter to build a stereoscopic camera in OpenGL. In order to model a stereoscopic camera, we need eye separation parameter. This parameter is simply the distance between observer's eyes. However this effect is very important about fusing images and distortion models.

There are various eye separation distances that users can be comfortable with. This is about fusion tolerance and perception of the user. However best stereoscopic depth perception is experienced when eye separation value is near 5 cm. Figure 6.1, Figure 6.2 and Figure 6.3 show various eye separation values effect on IB MAP stereoscopy.

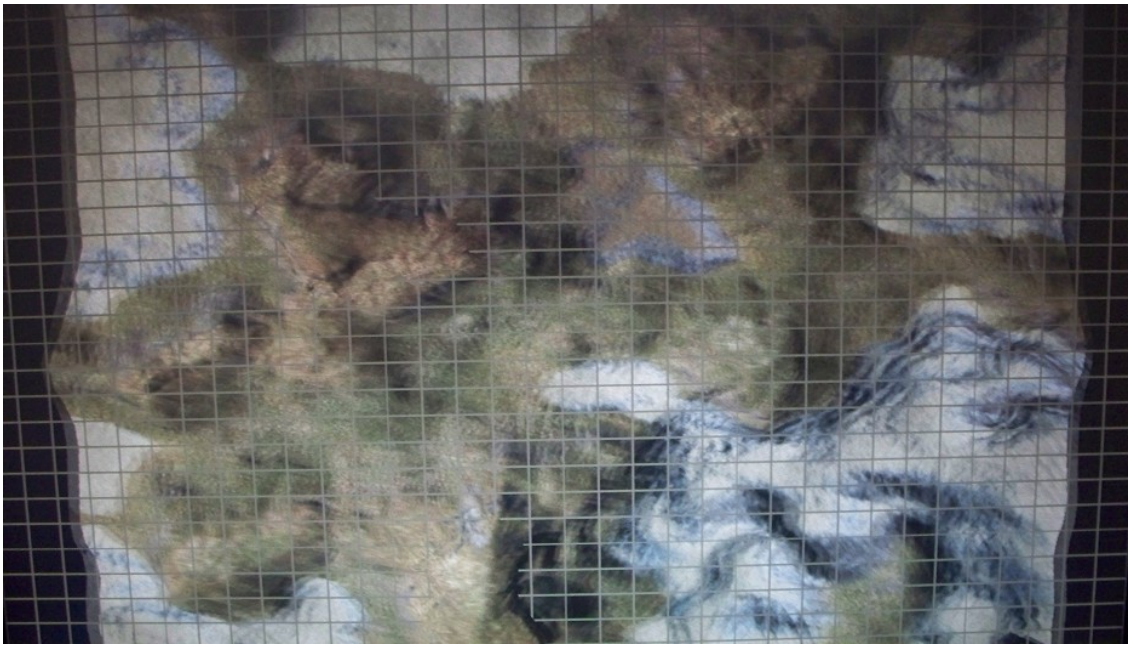


(a) Eye separation is 0 cm

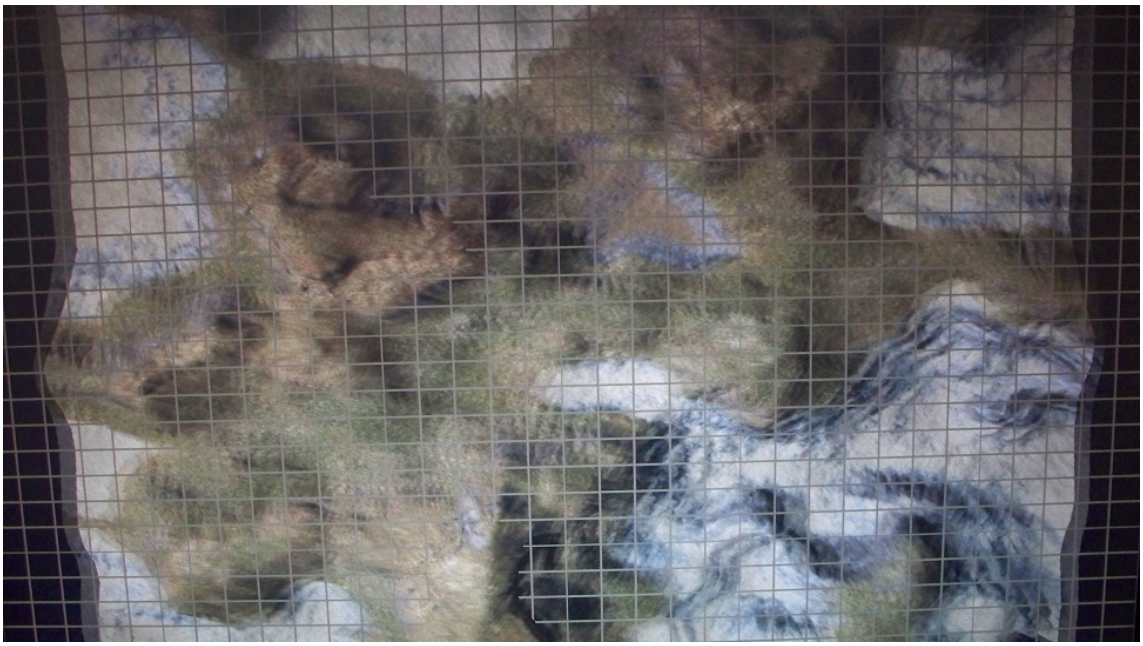


(b) Eye separation is 3 cm

Figure 6.1. Eye separation effect for values 0 cm and 3 cm.



(a) Eye separation is 5 cm



(b) Eye separation is 7 cm

Figure 6.2. Eye separation effect for values 5 cm and 7 cm.

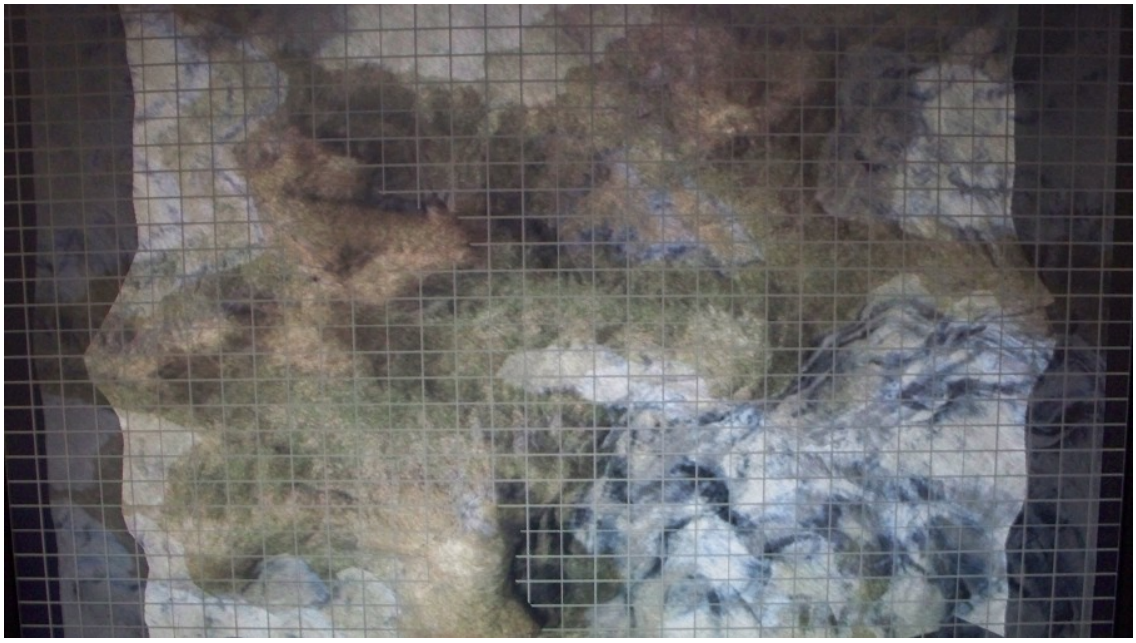


Figure 6.3. Eye separation parameter effect for value 30 cm.

When we changed eye separation value, fusing effect of the stereoscopy changed significantly for the RWB. Greater eye separation values than 0 cm increases depth perception of stereoscopic effect as expected. Fusing images is not a problem for the eye separation interval between 5 cm and 0 cm for terrain object translation value 0. If we increase eye separation, increasing parameter value causes fuse errors and diplopia. Diplopia is double vision error caused by relative eye misplacement.

Also, we translated terrain object on a direction vertical to RWB surface plane. Translating object and changing its place according to RWB surface, distorted the realism effect. In order to maintain stereoscopy fusing effect we adjusted eye separation value.

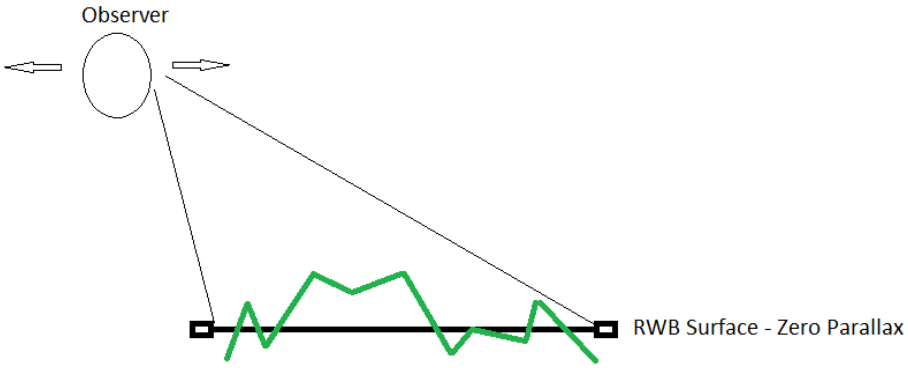
High eye separation values show that fusing is not possible higher than that value. Low separation values show that no fusible stereoscopy effect is possible below that value.

6.2. Distortion on Stereoscopic RWB

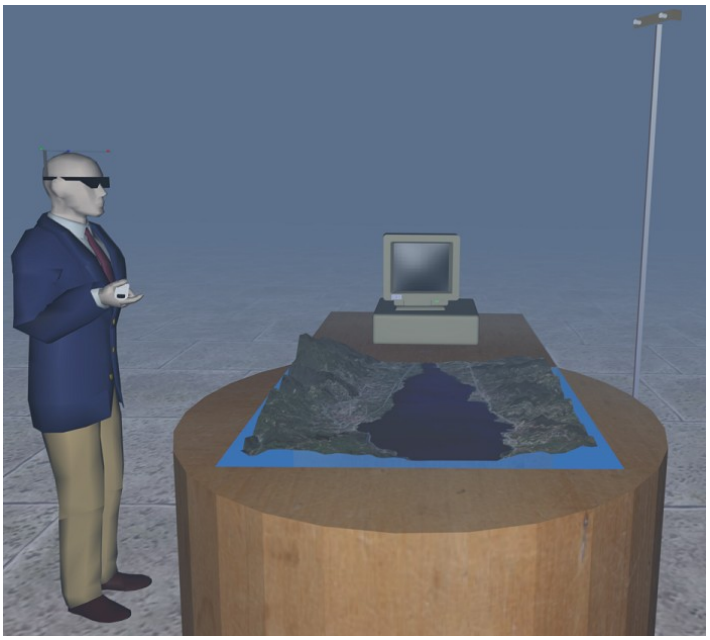
We test our application for distortion issue. In RWB, we tested terrain viewing for non head tracked system. We observed distortions of shearing effect on viewed terrain.

Also we tested terrain viewing with head tracking property. We also observed some distortion on viewing the terrain.

Viewing terrain in IB MAP has some stereoscopic distortion issues. Ideally when we look at RWB, we should be able to see a solid 3D shape that has no distortions. Changing position or moving our head should not be a problem for viewing terrain correctly. Figure 6.4 and Figure 6.5 show head movements relative to RWB.

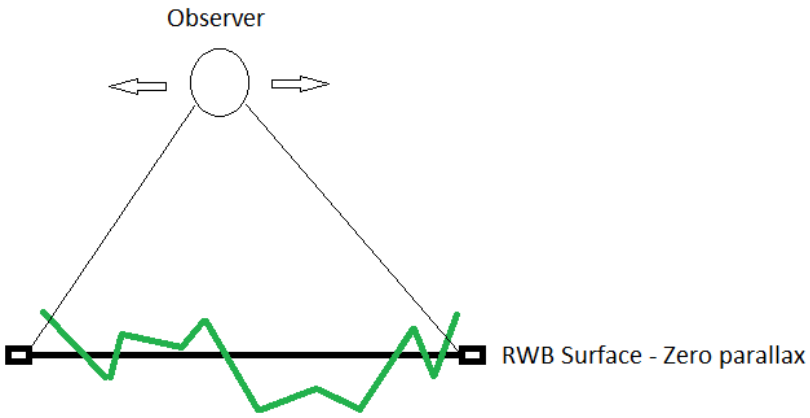


(a) Moving observer's head back and forth

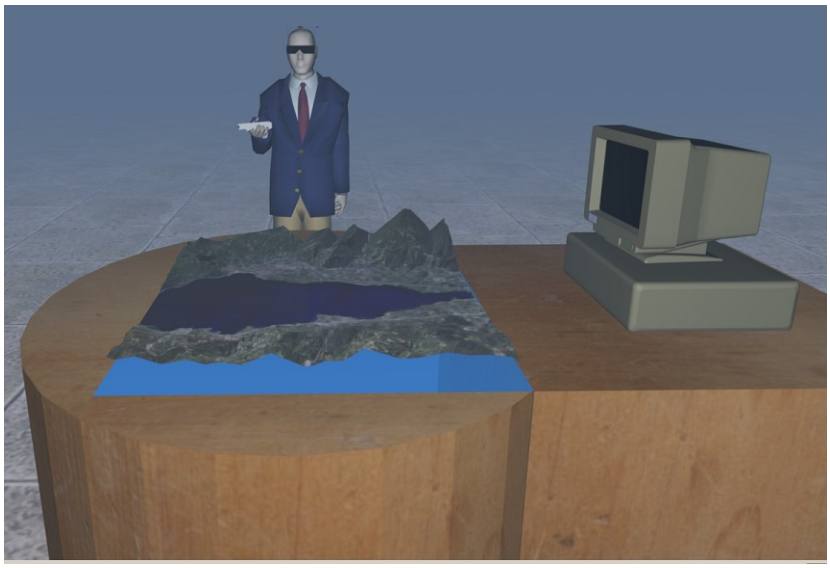


(b) Observer moving back and forth

Figure 6.4. Back and forth head movement in responsive workbench.



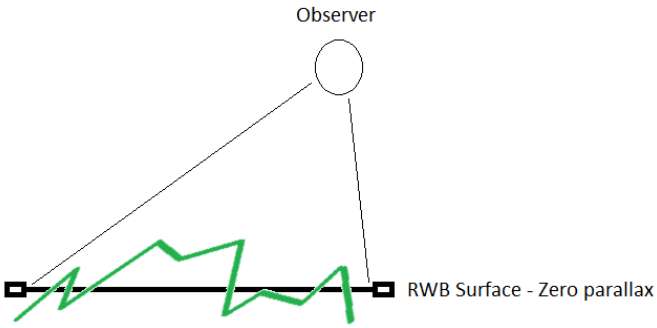
(a) Moving observer's head right and left



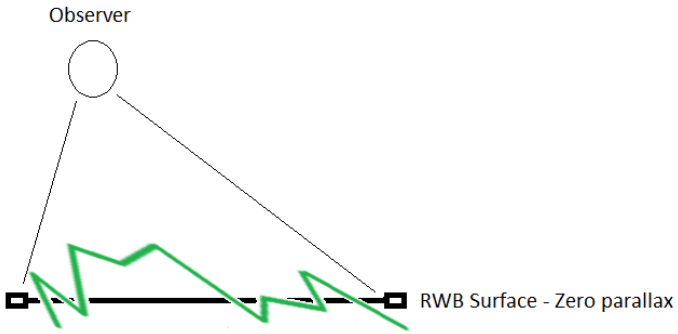
(b) Observer moving left and right

Figure 6.5. Left and right head movement in responsive workbench.

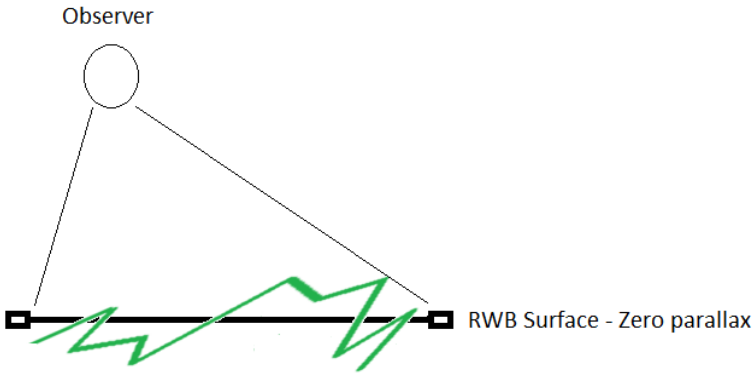
Moving observer's head from initial position ideally must not distort the image perceived. But when we test without head tracking system, we observed distortions on image. These distortions are shearing of the image we experience on the viewing surface. When observer moves towards to the left of him, image also shears to the left side. When observer moves towards to the right of him, image also shears to the right side. These distortions are also counted for forward and backwards moving. When observer moves forward, images shears forwards. And when observer moves backwards, image also shears backwards. Figure 6.6 shows the distortions we experienced without head tracking is on.



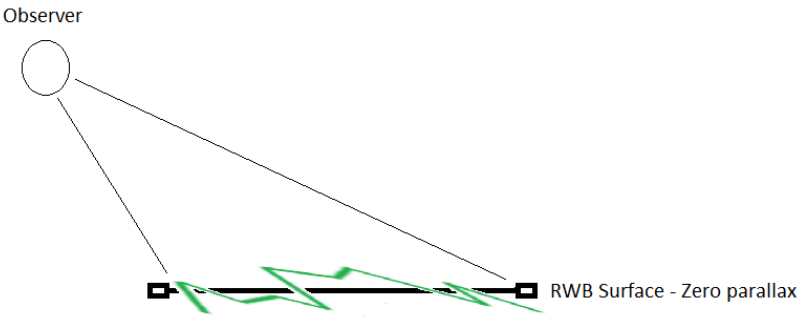
(a) Distortion when user moves to the left



(b) Distortion when user moves to the right



(c) Distortion when user moves forward



(d) Distortion when user moves backwards

Figure 6.6. Distortion on RWB relative to observer movement without head tracking

When viewing terrain, we realized that top of the mountainous areas are distorting more compared to skirts. Thus we come to the conclusion that distortion amount is not similar for all terrain parts. While experiencing distortions, we translated terrain completely above the zero parallax and completely below zero parallax. We saw that object distortion increases when it gets further away from zero parallax (see Figure 6.7).

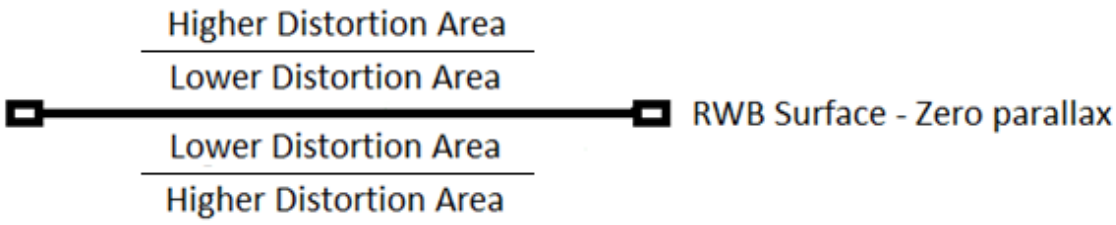
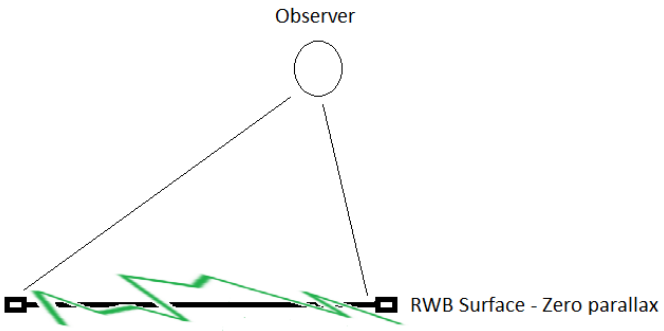
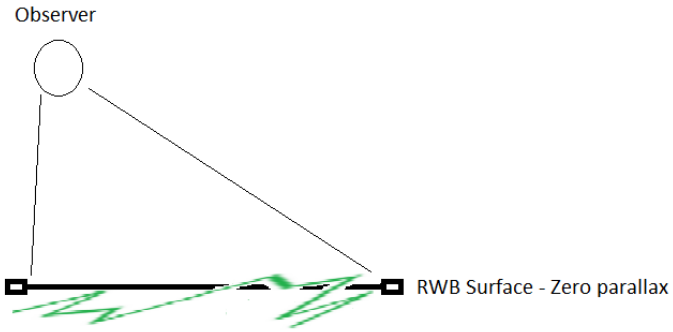


Figure 6.7. Distortion areas

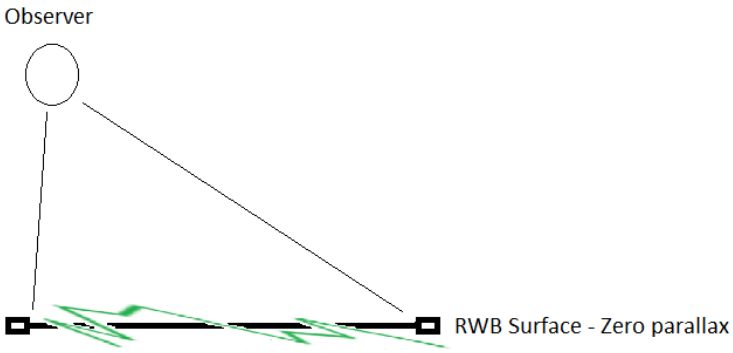
We also tested system while head tracking is on. After activating head tracking system, we applied same movements for the observer and observed distortions. We observed different distortions this time. On contrary to non-head tracked distortions, we observed shear effect opposite direction of movement this time. However distortion amount is not as much as non-head tracked shearing effect (see Figure 6.8).



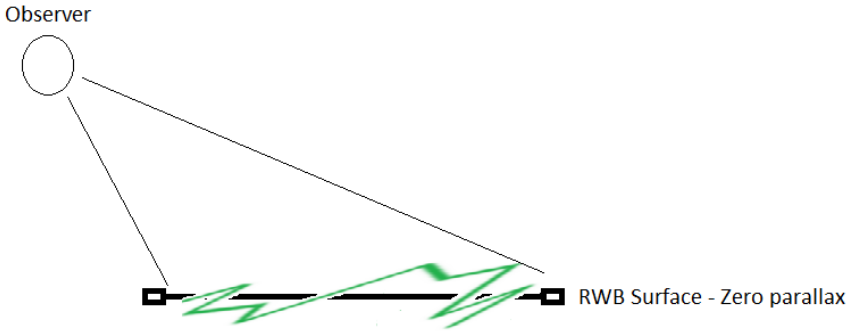
(a) Distortion when user moves left



(b) Distortion when user moves right



(c) Distortion when user moves forward



(d) Distortion when user moves backwards

Figure 6.8. Distortion on RWB relative to observer movement with head tracking

We also changed disparity value and tested distortion relative to different disparity values. Distortion without head tracking is on, shows a decrement when we lower disparity value. Moreover, distortion without head tracking increases when we increase disparity value as well.

6.3. RWB and PC Comparison

In our thesis, we wanted to evaluate the utilization of the RWB and a regular vertical non stereoscopic computer. RWB used wiimote and horizontal top table display system. On the other hand vertical display used classical keyboard and mouse input methods. Our experiences of using the vertical display and RWB showed that RWB is more effective for user to understand terrain data. We concluded that terrain viewing is suitable for horizontal display systems in stereo mode rather than classical vertical computer display.

6.4. Experimentation

In our thesis study we have conducted some experimentation about stereoscopy on RWB. Mainly we have focused on eye separation parameter effect over stereoscopy on RWB. In these experiments, our test subjects were computer engineer students. Totally there are 17 subjects including B.S, M.Sc and Ph. D students. Experimentations are made in 4 parts. First part is about stereo fusing and distortion change relative to eye separation. Second part is adjusting eye separation value with respect to viewed object place relative to zero parallax. Third part is distortion change with respect to viewed object place relative to zero parallax. Finally, determining which system that users would choose between different terrain viewing setups is the experimentation. Experimentation questionnaire is in Appendix A.

6.4.1. Distortion and Stereo Fusing

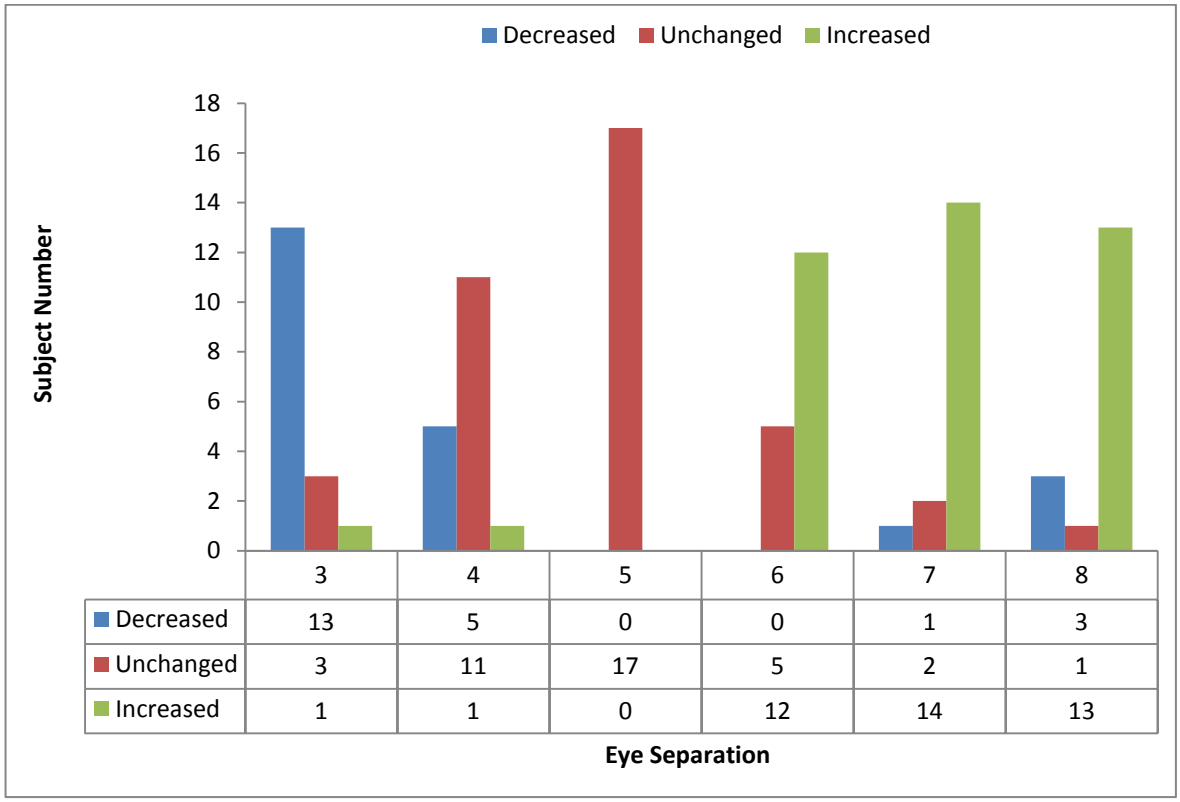
First question of the experimentation is to find out eye separation effect on distortion and stereo fusing. Stereo fusing is the ability to merge to separate images in brain or in different terms, feeling the stereoscopy effect. It is also considered as increasing in 3D perception. We asked subjects to answer stereo fusing effect is increased if they can merge

image pairs with respect to control group. If user can merge both values of eye separation, we asked them to answer as increased according to 3D perception increment. We also asked if user can not merge image pairs with respect to control group. If user can not merge both values of eye separation, we asked them to answer as decreased according to 3D perception decrement. If user decides perception and fusing is same, then they just answered as unchanged. Control group we used here is, 5 cm eye separation value. Subjects considered other experimentation eye separation values compared to 5 cm eye separation control value.

Distortion is also experimented with respect to eye separation. Distortion is the amount of image shearing (bending) when user moves his/her head forward / backward or left / right. Amount of distortion is determined by shearing amount according to same head movement.

Figure 6.9 shows distortion and stereo fusing results with respect to eye separation change. Value 5 is the control value and rated by all users as default unchanged.

According to results we obtained, most of the test subjects rated stereo fusing as increased when we increase eye separation value starting from 5cm. However after 7 cm we observe that stereo fusing is decreased. From those results we can say that eye separation value is changing optimally between 5 cm and 7 cm. When eye separation increases further than 7 cm, fusing starts to decrease. Also we decreased eye separation value starting from 5 cm to 3 cm. Test subjects yield to rate stereo fusing started to decrease below 4 cm. However from the results, we can obtain that there is not much of a noticeable difference between 5cm and 4 cm eye separation values.



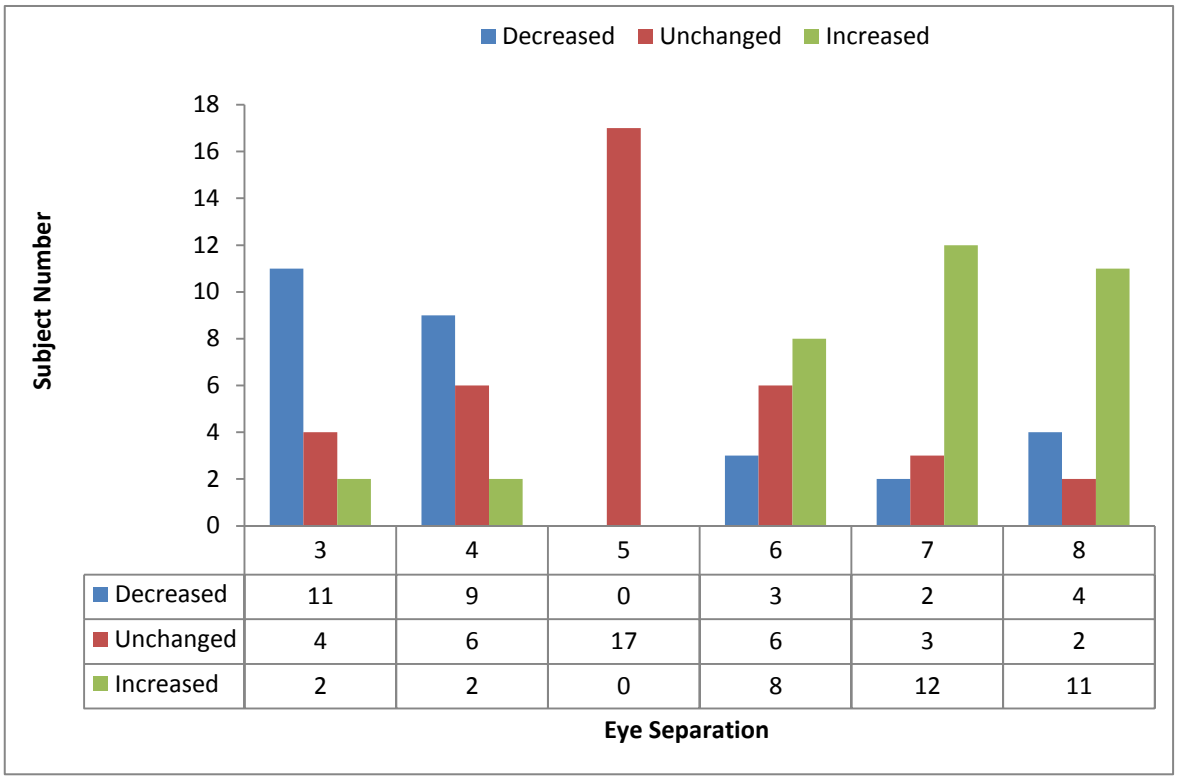
	3	4	5	6	7	8
Decreased	13	5	0	0	1	3
Unchanged	3	11	17	5	2	1
Increased	1	1	0	12	14	13

Figure 6.9. Stereo fusing with respect to eye separation

Figure 6.10 shows experiment results about distortion with respect to eye separation. Value 5 is the control value and rated by all users as default unchanged.

We also asked test subjects to rate distortion with respect to eye separation value change. Distortion amount is determined by shearing amount of the viewed object relative to same amount of head movement. Most of the test subjects rated distortion as decreased while eye separation value is decreased. On the other hand, increasing eye separation value resulted in increasing distortion, while eye separation value varies between 5 cm and 7 cm.

However while eye separation value is beyond 7 cm, subject numbers that decided distortion as increased is lower than the eye separation value is smaller than 8 cm. We can conclude that distortion is not recognizable while fusing starts to decrease.



	3	4	5	6	7	8
Decreased	11	9	0	3	2	4
Unchanged	4	6	17	6	3	2
Increased	2	2	0	8	12	11

Figure 6.10. Distortion with respect to eye separation

6.4.2. Eye Separation Value Estimation

In this experiment, we aim to evaluate optimum eye separation value for a varying height parameter. Here, height parameter is the distance of object being drawn to the projection surface. We changed terrain distance from projection surface and asked test subjects about a suitable eye separation value which they can best perceive terrain in stereo mode.

Figure 6.11 shows estimated eye separation value which respect to varying terrain height.

From the experimentation result graph (Figure 6.11), we can see that for terrain Z values near 3, subjects prefer smaller eye separation values. When terrain Z value goes to 9 or -9, eye separation preference of test subjects are increased to 10 cm border. When we average results, we can see that eye separation choice is smaller when terrain z value is

near 3. Also when terrain z value gets away from value 3, eye separation choice increases (see Figure 6.12).

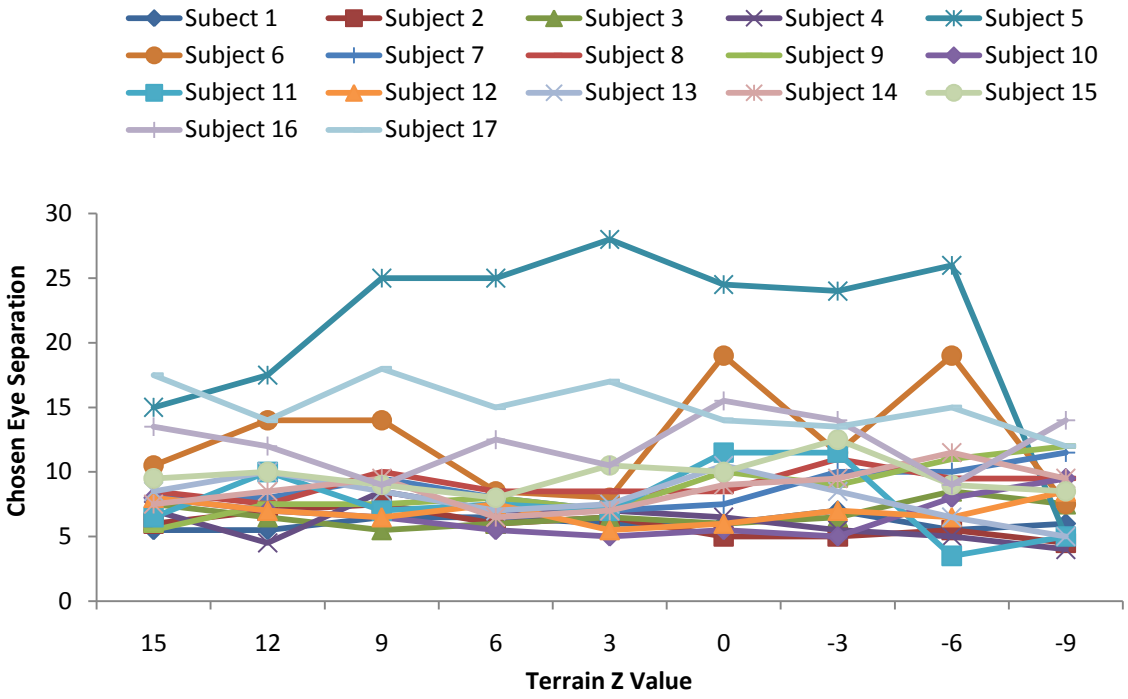


Figure 6.11. Eye separation estimation with respect to Z value

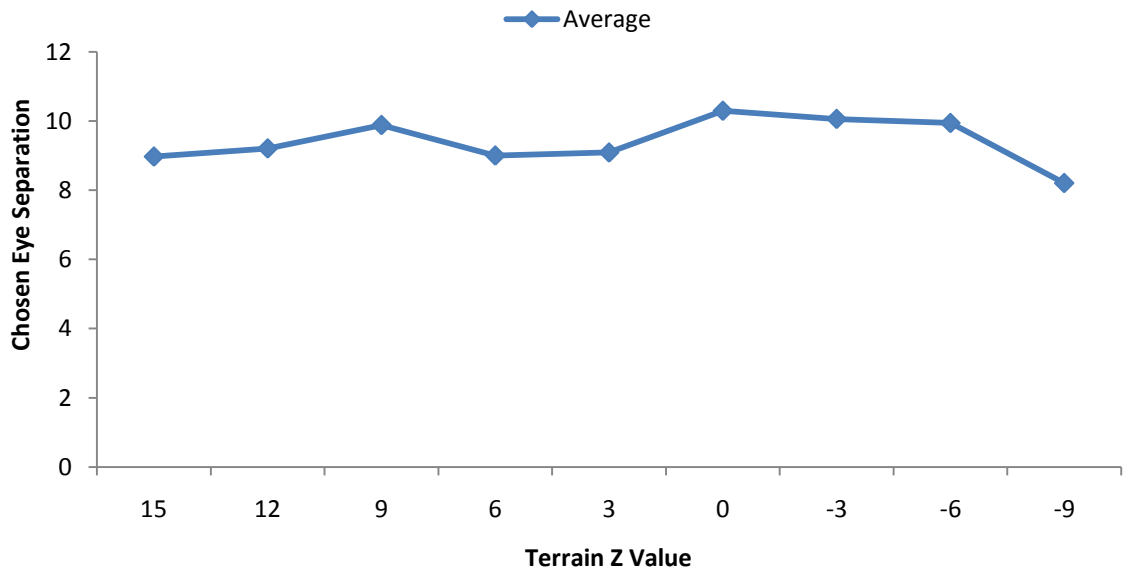


Figure 6.12. Eye separation value estimation average

Moreover, after we eliminate 3 extreme test results (Subject 5, Subject 6 and Subject 17), we can see clearly choice of eye separation value with respect to terrain Z value (6.13). Also we reevaluate average graph and see test subjects' choices more reliably (6.14).

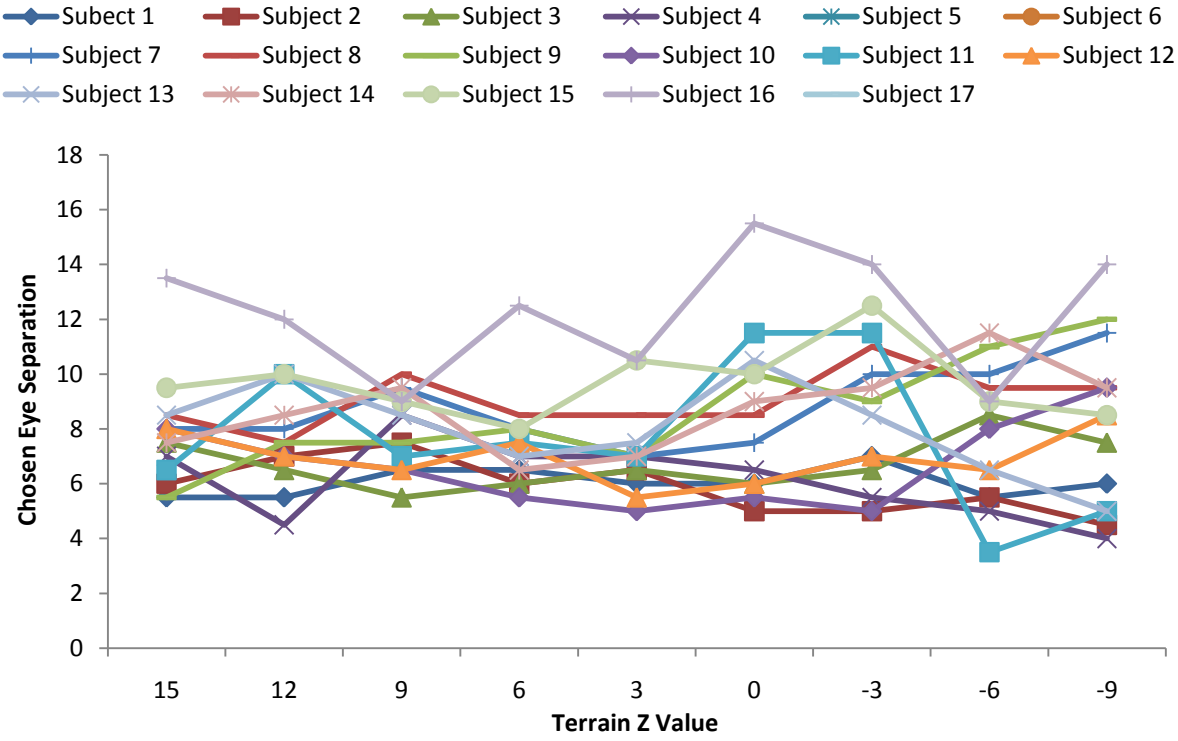


Figure 6.13. Extreme results eliminated eye separation estimation chart

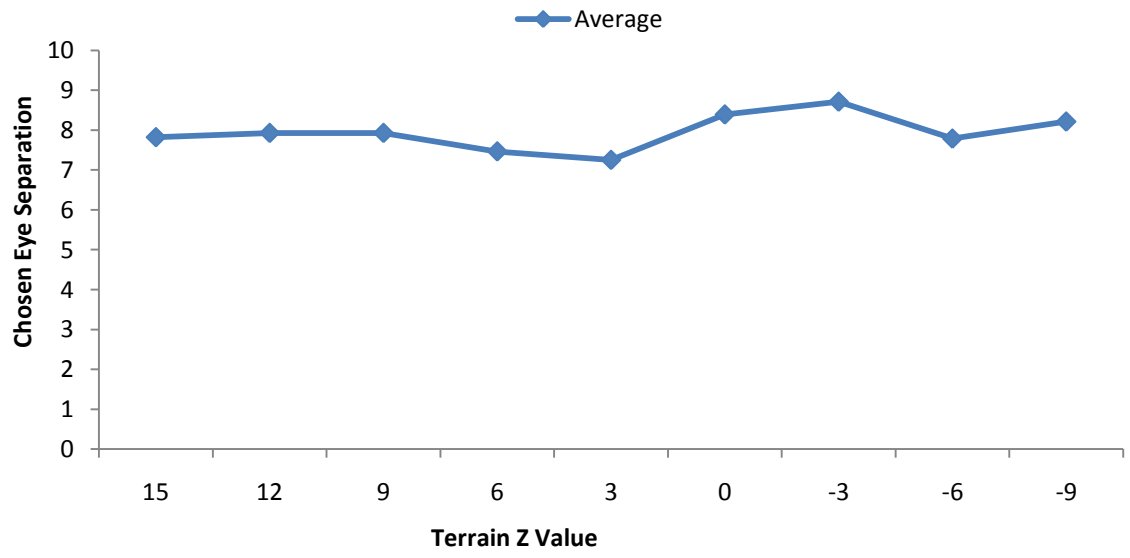
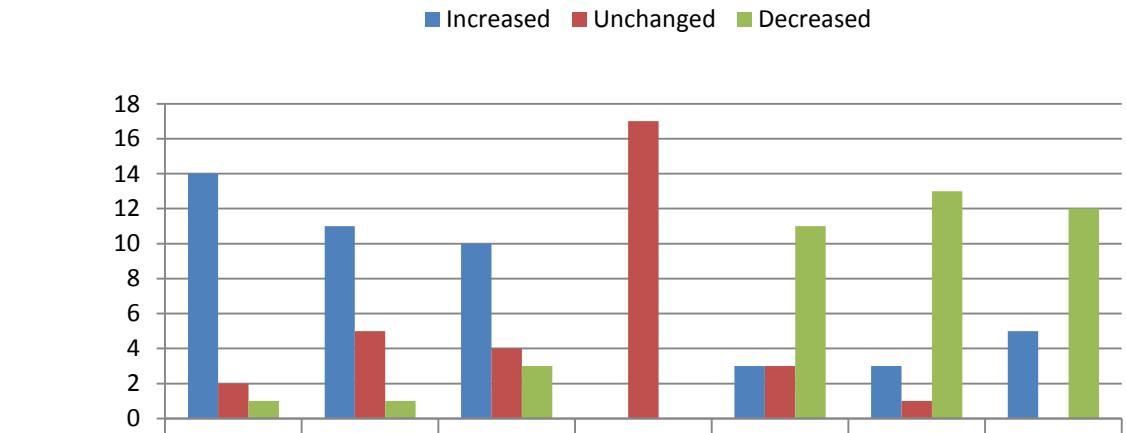


Figure 6.14. Extreme results eliminated eye separation estimation average chart

6.4.3. Distortion Estimation

In this experiment, we have asked test subjects to evaluate distortion amount with respect to terrain z value. Test subjects decided whether distortion increased or decreased. Control group is the terrain z value of 0 and distortion at this parameter. Figure 6.15 shows answers of test subjects, with respect to varying terrain z value.



	9	6	3	0	-3	-6	-9
Increased	14	11	10	0	3	3	5
Unchanged	2	5	4	17	3	1	0
Decreased	1	1	3	0	11	13	12

Figure 6.15. Distortion with respect to terrain Z value

As we can see from graph in Figure 6.15, distortion is increased when terrain gets closer to projection surface. On contrary, distortion is decreased when terrain gets further from projection surface. However this distortion value is observed for top sides of the viewed object. It is also asked to users, objects further parts from screen distorted more when z value is decreased. From these results, we can conclude that distortion is increased when object's parts gets further from zero parallax.

6.4.4. Users' Choice

In this experiment, for the last question we asked users about what kind of a system they would prefer for a terrain viewing. Figure 6.16 shows answers of the users.

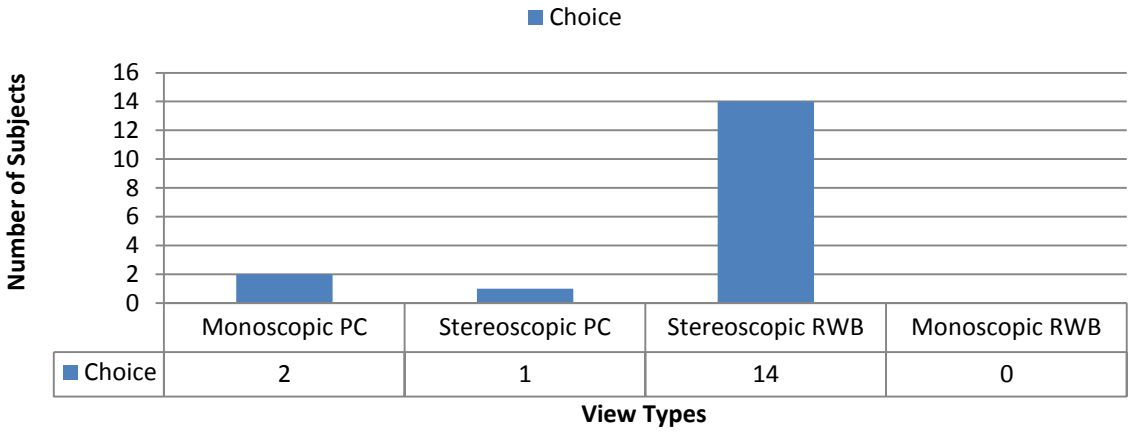


Figure 6.16. User choice for terrain viewing system

Most of the users choose RWB system for a better perception of terrain viewing. Some users pointed out that stereoscopic viewing is causing headache and therefore they choose monoscopic computer viewing. One user chooses stereoscopic pc viewing for better resolution and classical usage. None of the subjects choose monoscopic RWB viewing.

We also made test subjects to wear head tracking device and made them try system with head tracking. Afterwards we asked explicitly to the users if they prefer static or head tracked system usage. All of the users that choose RWB stereoscopic viewing also choose head tracked system for a better perception.

7. CONCLUSIONS

We have successfully developed a real-time terrain visualization application on the responsive workbench which is a head tracked stereo visualization system. Responsive workbench has a horizontal displaying area placed on a bench. The user wears stereo goggles and interacts with the system using a Nintendo Wii remote.

We believe this system is the first terrain visualization system that allows the terrain to be visualized on a real table as if a physical mock up model of the terrain is actually being viewed. We also believe that this approach has certain advantages for certain type of terrain visualization applications where fly through is not necessary in applications like urban planning, plant population monitoring, highway design. The use of a mock up model built to scale is found to be useful in planning applications. User can walk around the model that sits on the table and interacts with it.

We have evaluated the potential stereo distortion problems on system. We have also evaluated the effect of various stereo visualization parameters like eye separation and model height from display surface.

Interactive Briefing Map application is implemented using a software infrastructure which is developed as part of the thesis work. In order to improve real – time behavior, a level of detail algorithm is implemented. Interactive Briefing Map has a flexible architecture that allows visualization line of sight and terrain features other than geometry. System has 6 degrees of freedom device to allow user interaction

Short term future work includes accuracy improvement for head tracking system and better vision quality on RWB. Also correcting distortions for stereoscopic horizontal viewing is an important future work for system's utilization. Level of detail algorithm will be improved to allow system to handle visualization of large terrains. Also terrain editing will be implemented to add more capability to system.

APPENDIX A: INTERACTIVE BRIEFING MAP EXPERIMENT

1 – For each Eye Separation value answer the questions below. Rank as you perceive the terrain. Use evaluation ranking values to fill in the blanks. Evaluation subjects explained below.

Stereo Fusing is the ability to merge to seperate images in brain or in different terms, feeling the stereoscopy effect.
Distortion is the amount of image shearing (bending) when user moves his/her head forward / backward or left / right

Evaluation Ranking
(Least) 1 – 2 – 3 (Most)

TIP: Use keyboard button Z to increase eye separation
Use keyboard button X to decrease eye separation

Table A.1. Stereo fusing table

• While Eye Separation value equals to 3	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .
• While Eye Separation value equals to 4	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .
• While Eye Separation value equals to 5	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .
• While Eye Separation value equals to 6	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .
• While Eye Separation value equals to 7	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .
• While Eye Separation value equals to 8	
Evaluation Type	Value
Stereo Fusing	. . . . .
Distortion Amount	. . . . .

2– For each Z value of the terrain, arrange Eye Separation to obtain best depth perception and answer the questions below. Fill in the blanks with the Eye Separation values that you best perceived the stereoscopy effect with respect to Z value.

TIP: Use keyboard button Z to increase eye separation
Use keyboard button X to decrease eye separation
Use keyboard button C to increase Z value
Use keyboard button V to decrease Z value

Table A.2. Z values table

Z Values	Eye Separation Value
While Z value equals to 15	. . . . .
While Z value equals to 12	. . . . .
While Z value equals to 9	. . . . .
While Z value equals to 6	. . . . .
While Z value equals to 3	. . . . .
While Z value equals to 0	. . . . .
While Z value equals to -3	. . . . .
While Z value equals to -6	. . . . .
While Z value equals to -9	. . . . .

3- For each Z value answer the questions below. Rank as you perceive the terrain. Use evaluation ranking values to fill in the blanks. Evaluation subjects explained below.

Distortion is the amount of image shearing (bending) when user moves his/her head forward / backward or left / right

Evaluation Ranking
(Least) 1 – 2 – 3 (Most)

TIP: Use keyboard button C to increase Z value
Use keyboard button V to decrease Z value

Table A.3. Distortion table I

• While Z value equals to 9	
Evaluation Type	Value
Distortion	. . . . .
• While Z value equals to 6	
Evaluation Type	Value
Distortion	. . . . .
• While Z value equals to 3	
Evaluation Type	Value
Distortion	. . . . .

Table A.4. Distortion table II

• While Z value equals to 0	
Evaluation Type	Value
Distortion	. . . . .
• While Z value equals to -3	
Evaluation Type	Value
Distortion	. . . . .
• While Z value equals to -6	
Evaluation Type	Value
Distortion	. . . . .
• While Z value equals to -9	
Evaluation Type	Value
Distortion	. . . . .

4- Compare PC perception of terrain viewing and RWB perception of terrain viewing. Select only one choice and tick it. Make selection after trying all types. Glasses must be wear on while trying stereoscopic modes.

TIP: Execute MonoPC to experience monoscopic PC view in monitor.
Execute StereoPC to experience stereoscopic PC view in monitor.
Execute MonoRWB to experince monoscopic RWB view in RWB.
Execute StereoRWB to experince stereoscopic RWB view in RWB.

Table A.5. Users' choice

Evaluation Type	Value
Monoscopic PC	. . . . .
Stereoscopic PC	. . . . .
Monoscopic RWB	. . . . .
Stereoscopic RWB	. . . . .

REFERENCES

1. Doyle, F. J., “Digital terrain models - An overview”, *Photogrammetric Engineering and Remote Sensing*, Vol. 44, pp. 1481-1485, 1978.

2. Banks, R. and C. D. Wickens, “Commanders’ Display of Terrain Information: Manipulations of Display Dimensionality and Frame of Reference to Support Battlefield Visualization”, Aviation Research Laboratory, Institute of Aviation, University Illinois.

3. Hernandez, L., "Application Of Digital 3D Models On Urban Planning And Highway Design", *Third Conference on Urban Transport and the Environment*, Computational Mechanics Publications.

4. Kavouras, M., "Human-Computer Interaction Considerations In Terrain Modeling And Visualization", *Cognitive Aspects of Human-Computer Interaction for Geographic Information Systems*, pp. 213-219, 1995.

5. Davis, D., T. Y. Jiang and W. Ribarsky, "Intent, Perception, and Out-of-Core Visualization Applied to Terrain", *Report GIT-GVU-98- 12*, *IEEE Visualization*, pp. 455-458, 1998.

6. Koller, D., P. Lindstroma, W. Ribarskyb and L. Hodges, "VIRTUAL GIS: A Real-Time 3D Geographic Information System", *IEEE Visualization Proceedings of the 6th conference on Visualization*, p. 95, 1995.

7. He, Y., "Real-Time Visualization of Dynamic Terrain for Ground Vehicle Simulation", Ph.D Thesis, University of Iowa, 2000.

8. Beck, M.," Real Time visualization of Big 3D City Models, International Archives of the Photogrammetry", *Remote Sensing and Spatial Information Sciences*, Vol. XXXIV-5/W10, 2003.

9. Faeth, A., M. Oren and C. Harding, "Combining 3-D Geovisualization With Force Feedback Driven User Interaction", *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, 2008.

10. Duchaineau, M., M. Wolinsky, D. Sigeti and M. Millery, "ROAMing Terrain: Real-time Optimally Adapting Meshes", *Proceedings of IEEE Visualization Proceedings of the 8th conference on Visualization,* pp. 81–88, 1997.

11. Mordechay, E. H., "Virtual Reality and GIS: applications, trends and directions", pp. 47-57, 1998.

12. Nobrega, R., A. Sabino and A. Rodrigues, "Flood Emergency Interaction and Visualization System", *Visual Information Systems, Web-Based Visual Information Search and Management*, Lecture Notes in Computer Science, pp. 68- 79, 2008.

13. Deussen, O., P. Hanrahan and B. Lintermann, "Realistic Modeling And Rendering Of Plant Ecosystems", *Proceedings Of The 25th Annual Conference On Computer Graphics And Interactive Techniques*, pp. 275 - 286, 1998.

14. Hitchner, L. E., "Virtual Planetary Exploration: AVery Large Virtual Environment", *ACM SIGGRAPH Tutorial on Implementing Immersive Virtual Environments*, pp. 6.1 - 6.15, 1992

15. Kuka Entertainment. http:// www. kuka-entertainment.com, 11 2009.

16. Pomerantz, M., I. A. Jain and S. Myint, "Dspace: Real-Time 3D Visualization System for Spacecraft Dynamics Simulation", *Third IEEE International Conference on Space Mission Challenges for Information Technology*, pp. 237-245, 2009.

17. Olanda, R., M. Pérez and P. Morillo, "Entertainment Virtual Reality System for Simulation of Spaceflights Over the Surface of the Planet Mars", *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 123 - 132, 2006.

18. Centre National D'études Spatiales, Comitee on Earth Observation Satellites. http://ceos.cnes.fr:8100/cdrom-00/ceos1/satellit/spotsys/images/stereosp.jpg, 03 2010.

19. Zhao, Q., "A Survey On Virtual Reality", *Science in China Series F: Information Sciences*, Vol. 52, Number 3 , pp. 348-400, March, 2009.

20. Frame Rate Test Videos. http://spng.se/frame-rate-test/, 02 2010.

21. Polygon Reducer. http://polygon-reducer.pc-guru.cz/blog/polygon-reducer-lod-1.jpg, 04 2010.

22. Arvaux, S., J. Legau, S. Limet, E. Melin and S. Robert, "Parallel LOD for Static and Dynamic Generic Geo-Referenced Data", *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pp. 301-302, October 27-29, 2008.

23. Levenberg, J., "Fast View-Dependent Level-of-Detail Rendering Using Cached Geometry", *IEEE Visualization*, pp. 259-265, Boston, MA, USA, 2002.

24. Ulrich, T., "Chunked LOD: Rendering massive terrains using chunked level of detail control", http://www.vterrain.org/LOD/Papers/index.html, 2002.

25. Williams, L., "Pyramidal Parametrics". *Proceedings of Computer Graphics SIGGRAPH'83*, Vol. 17, pp. 1–11, July 1983.

26. PRESAGIS Modeling and Simulation. http://www.presagis.com/, 02 2010.

27. Bryan, T., "Real-Time Dynamic Level of Detail Terrain Rendering with ROAM", http://www.gamasutra.com/view/feature/3188/realtime_dynamic_level_of_detail_.php, 2000.

28. Krüger, W., A. Bohn, B. Fröhlich, H. Schüth, W. Strauss and G. Wesche, "The Responsive Workbench", *IEEE Computer Graphics and Applications*. Vol.14, pp. 12-15, 1994.

29. Atracsys. http://www.atracsys.com/_products/atracTable.php, 05 2010.

30. Microsoft Surface. http://www.microsoft.com/surface/en/us/default.aspx, 05 2010.

31. Durbin, J., J. Edward Swan and Brad Colbert, "Battlefield Visualization on the Responsive Workbench", *Proceedings of the conference on Visualization '98*, pp. 463 - 466, Research Triangle Park, North Carolina, USA, 1998.

32. Piper, A.M., and J.D. Hollan, "Supporting Medical Conversations between Deaf and Hearing Individuals with Tabletop Displays", *ACM Conference on Computer Supported Cooperative Work*, pp. 147 - 156 , 2008

33. Wesche, G. and Hans-Peter Seidel, "FreeDrawer. A Free-Form Sketching System on the Responsive Workbench", *Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 167 - 174, 2001.

34. Wesche, G., J. Wind and M. Göbel, "The Responsive Workbench for Visualization of Fluid Dynamics", *ERCIM News No.31*, 1997

35. Grossman, T. and D. Wigdor, "Going Deeper: a Taxonomy of 3D on the Tabletop", *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer System*, pp. 137–144, 2007.

36. Koutek, M. and F. H. Post, "Dynamics in Interaction on the Responsive Workbench", *Proceedings of Eurographics Virtual Environments* , pp. 43-54 ,2000.

37. Schmalsteig, D., L. M. Encamcao and Z. Szalavari, "Using Transparent Props For Interaction With The Virtual Table", *ACM Symposium on Interactive 3D Graphics*, pp. 147-153, 1999.

38. Paljic, A., S. Coquillart, J.M. Burkhardt and P. Richard, "A Study of Distance of Manipulation on the Responsive Workbench", *Proceedings Immersive Projection Technology Workshop*, Orlando, Florida, 2002.

39. Wartell, Z., W. Ribarsky and L. Hodges, "Third-Person Navigation of Whole-Planet Terrain in a Head-tracked Stereoscopic Environment", *IEEE Virtual Reality Conference*, page 141, 1999.

40. Nintendo Controllers, http://www.nintendo.com/wii/what/controllers, 05 2010.

41. DepthQ. http://www.depthq.com/, 05 2010.

42. Crystal Eyes 3. http://www.reald.com/Content/Crystal-Eyes-3.aspx, 05 2010.

43. Nvidia Quadro FX 1400 .http://h18000.www1.hp.com/products/quickspecs/ 12093_na/ 12093_na.html, 05 2010.

44. Ugur, B., A. V. Şahiner and I. B. Fidaner, " Off-Axis Stereo Projection and Head Tracking for a Horizontal Display", *ACE'09*, October 29–31, Athens, Greece, 2009.

45. Çizmeci, M. R., "3D Head Tracking And Vrml Editor For The Responsive Workbench System", B.Sc Thesis, Boğaziçi University, June 2009.

46. Schou, T. and H. Gardner, "A Wii Remote, a Game Engine, Five Sensor Bars and a Virtual Reality Theatre", In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, pp. 231-234, Adelaide, Australia, 2007.

47. Chow, Y., "Low-Cost Multiple Degrees-of-Freedom Optical Tracking for 3D Interaction in Head-Mounted Display Virtual Reality", *International Journal of Recent Trends in Engineering*, Issue. 1, Vol. 1, May 2009.

48. Kaltenbrunner, M., T. Bovermann, R. Bencina and E. Costanz, "TUIO: A Protocol for Table-Top Tangible User Interfaces", *Proceedings of the The 6th International Workshop on Gesture in Human-Computer Interaction and Simulation*, 2005.

49. POSIX Thread Programming. https://computing.llnl.gov/tutorials/pthreads/#Thread, 09 2009.

50. Wartell, Z. J., "Stereoscopic Head-Tracked Displays: Analysis And Development Of Display Algorithms", Ph.D Thesis, Georgia Institute of Technology, August 2001.

51. Kersting, O. and J. Döllner, "Interactive 3D Visualization of Vector Data in GIS", *Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pp. 107 - 112, 2002.

52. GIS Mapping Software, http://www.globalmapper.com/, 04 2010.

53. ArcGIS: A Complete Integrated System, http://www.esri.com/software/arcgis/index.html, 04 2010.

54. Ben-Moshe, B. and P. Carmi, M. Katz, "Approximating the Visible Region of a Point on a Terrain", *GeoInformatica,* Vol. 12, No. 1, pp. 21-36, March, 2008.

55. Franklin, R., C. Ray and S. Mehta, "Geometric Algorithms for Siting of Air Defense Missile Batteries", 1994.

56. Kaucic, B. and B. Zalik, "Comparison of Viewshed Algorithms on Regular Spaced Points", *Proceedings of the 18th spring conference on Computer graphics*, pp. 177 - 183, Budmerice, Slovakia, 2002