12-2003

Virtual and Augmented Reality Simulation of Chattanooga Creek

Sirisha Vadlamudi
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Sirisha Vadlamudi entitled "Virtual and Augmented Reality Simulation of Chattanooga Creek." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Daniel B. Koch, Major Professor

We have read this thesis and recommend its acceptance:

Michael J. Roberts, Mostofa M. Howlader

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Sirisha Vadlamudi entitled "Virtual and Augmented Reality Simulation of Chattanooga Creek." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Daniel B. Koch

Major Professor

We have read this thesis and recommend its acceptance:

Michael J. Roberts

Mostofa M. Howlader

Accepted for the Council:

Anne Mayhew

Vice Provost

and

Dean of Graduate Studies

(Original signatures are on file with official student records.)

Virtual and Augmented Reality Simulation of Chattanooga Creek

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Sirisha Vadlamudi
December 2003

Abstract

Virtual reality involves reproducing all the details of a physical object or environment that has to be simulated in virtual space. Augmented reality is the mixing of computer-generated information with the user's view of the real world. Augmented reality is an alternative for virtual reality. This thesis presents the interactive virtual reality simulation and augmented reality simulation of Chattanooga Creek in Chattanooga, TN. The augmented reality part involves implementation of the basic concept of augmenting the virtual graphics to the real world.

Acknowledgements

I thank Dr. Daniel B. Koch for giving me the opportunity to work on this project and for his guidance and support throughout my graduate studies at the University of Tennessee, Knoxville. I also thank Drs. Michael J. Roberts and Mostofa Howlader for serving on my thesis committee.

I am grateful to Drs. Gary Sayler and John Sanseverino of the Center for Environmental Biotechnology for funding and supporting this project and to Drs. Larry McKay and Vijay Vulava from the Department of Earth and Planetary Sciences for providing the background information required for this project.

Many thanks to Marcus Dutton for helping me fix problems during programming with WorldUp while working on this project.

I specially thank my parents Rani and Prasad Vadlamudi who worked very hard for providing me with the best and my husband Vijay Vulava for helping me in every possible way during my graduate school, including formatting of this document in LaTeX2e.

Contents

List of Figures

Chapter 1

Introduction

The Tennessee Products site (TP site) in Chattanooga, TN is a former coke production plant. Areas surrounding the Chattanooga Creek and the TP site were contaminated with the wastes released from the coke plant. This thesis is concerned with developing virtual reality (VR) and augmented reality (AR) simulations for the Chattanooga Creek and the TP site as an aid to researchers investigating how best to clean up the site.

The United States Geological Survey (USGS) provides digital elevation model (DEM) data of various parts of the country. A DEM file consists of elevation data of a terrain at regularly spaced intervals with respect to a reference point. USGS also provides digital maps called digital raster graphics (DRG), which are high-resolution scanned images of paper maps. DEMs and DRGs for a particular area can be obtained by identifying the topographic quadrangle name or the southeast latitude and longitude corner coordinates. The quadrangle division is same for both the DEMs and DRGs.

The data format for transferring a DEM is the spatial data transfer standard (SDTS). SDTS format has to be decoded to the original DEM format using translators available for that purpose (SDTS2DEM converter). In order to visualize the elevation data in the form of a terrain one can use a software program that connects all the points in the DEM file using polygons. This shows the data in the form of a mesh. The DRG is then overlaid on

top of the mesh structure to create a textured three dimensional (3D) terrain model. The placing of a DRG onto a DEM is called texture mapping. Proper registration of the DEM and DRG is required to avoid any discrepancies in the final 3D model.

In a virtual environment (VE) all the objects are generated by the computer and are virtual. The VR simulation part of this thesis incorporates the DEM, which is also virtual and provides a graphical user interface (GUI) to interact with the VE. The GUI consists of various buttons and keyboard controls each providing a unique function.

The VE can be either immersive or non immersive. Immersive VR makes the user feel that he/she is actually located inside the VE. This can be achieved by using devices like a head-mounted display (HMD). The software tool used for developing the VR simulation is a product from Sense8 called WorldUp. Okino's NuGraf, a 3D model development tool, is used for texture mapping and for converting the files to different formats.

Development of a VR simulation for a terrain is limited by the size of the DEM and DRG. Expanding the simulation to contain multiple data sets in a single project may not be possible because the development environment, WorldUp in this case, cannot hold such huge amounts of data. Also, errors in the registration of the DEM and DRG distort the terrain model thereby decreasing the reality of the simulation.

Augmented reality (AR) does not face the problems mentioned above because it does not depend on the size or area of the terrain in this application and also eliminates the need to create 3D models for the data that can actually be seen. AR deals with augmenting the real world images with virtual images. Virtual objects are used only when additional objects are required in the original scene or where real world objects need to be replaced or deleted. AR can be realized using either a video or optical see-through HMD. A video HMD is used for the AR simulation in this thesis.

AR can also be interactive in nature like VR. Developing a real time AR simulation is complex since it involves the rendering of virtual objects in different views depending on the location and orientation of the person in the AR environment. The resolution of the

images seen depends on the HMD being used.

The following is an outline of the thesis. Chapter 2 introduces the concept, applications and the latest developments in the fields of virtual and augmented reality. Chapter 3 explains in detail the virtual reality part of the Chattanooga Creek simulation from building the 3D terrain model to developing the user interface. Chapter 4 describes the various issues involved in creating the augmented reality portion of the system. Chapter 5 presents a summary and conclusions of the work.

Chapter 2

Virtual and Augmented Reality

2.1 Introduction

The terms Virtual Reality (VR) and Virtual Environment (VE) refer to an artificial environment synthesized using a computer or an image generator (IG). VR is a developing technology and a considerable amount of work still has to be done before it can be used at full scale in applicable areas. Many commercial VR systems are available on the market today for use in various applications. Ivan Sutherland's work on head mounted display (HMD) technology is a major break-through in the development of VR. Though the idea of VR has existed for a long time, the major source of this technology may have come from the development of various simulators like flight simulators [1].

2.2 Virtual Reality

All the objects in a virtual reality simulation are computer generated. The major goal of VR is to generate the physical world with the highest degree of reality possible and to interact with that synthesized environment. The user interface is what makes the VR different from other graphics applications or animated movies. Immersion is another important factor in

VR but the type of application determines whether the simulation has to be immersive or not.

2.2.1 Types of VR

VR can be classified into two types depending on the degree of immersion the user experiences: immersive and non-immersive. In the case of immersive VR, the user becomes a part of the VE, being cut off from the real world. An HMD is required to view the simulation from this perspective. The images in the HMD are usually updated according to the orientation of the user's head with the assistance of a tracking device.

In the case of non-immersive VR, the user views and interacts with the VE externally from the real world. The display devices here can be either monitor-based or projector-based. Monitor-based systems are called *through-the-window* or *fish tank* VR [2, 3].

Virtual representations of the real world are not limited to visual applications. They can be aural, haptic or tactile. The simulation interface allows sending input control signals to the VE and the output can be seen or heard or felt by the user depending on the form of VR. The current form of tactile feedback provides only a simple touch stimulus but active research is going on in tactile VR technology for communicating the output directly to the brain [2].

Augmented reality (AR) is another technology extending VR. AR supplements the real world by adding virtual objects and does not try to recreate the physical environment. This will be discussed in Section 2.3 of this chapter and later in Chapter 4.

2.2.2 Components in Creating a VR Application

VR requires hardware and the software for its creation like any other computer-based application. The software needs to produce 3D virtual objects, to maintain a database for the virtual models, to convert the models to other formats, and to provide functions for visualizing and interacting with the simulation. Typical hardware includes image generators,

HMDs, input devices like a mouse or joystick, equipment for tactile or haptic feedback, a sound system for acoustics, graphics cards for stereoscopic vision, and a tracker to locate the user's head.

2.2.3 Applications of VR

VR is a developing technology and has applications in the fields of engineering, science, entertainment and so on. While using VR for a task improves the performance in some applications, this technology is the only way to do the job in a few other applications [4].

Some advantages in general of VR are: building virtual prototypes of a model is more effective in terms of time and cost compared to building physical mock-ups; VR offers more flexibility and efficiency in analyzing a design when the virtual prototypes are combined with the results of the simulation; the use of VR in training eliminates any risk of accidents in operation because the trainee is not physically involved in the given situation; VR also allows one to simulate conditions which cannot be created in the real world [5].

Flight simulation is one example where the above comments apply. Before 1980 rigid scale models were used to simulate the environment for new pilots. With rigid scale models specific objects have to be constructed for each airport instead of having one general model. Also, extreme weather conditions like fog could not be created. With the development of flight simulators and with the use of virtual objects produced by the IG in the simulator, separate models for each airport were no longer required. Also bad weather conditions could be simulated in the virtual environment.

Engineering

Designing an aircraft takes years of work. Installing the engine in the aircraft and removing it from the aircraft for maintenance is complex and time-consuming because of the huge size of the engine and the large number of components encountered in the process. Careful precautions have to be taken to see that the engine and the other parts do not collide with

each other and get damaged. In 1994 McDonnell Douglas used a VR system to explore various ways of installing and removing engines. By using VR for this application the virtual engine could be removed from the aircraft and the parts in it could be separated for repair. The virtual engine can then be reassembled and reinstalled in the virtual aircraft. Since none of this is happening in the real environment the process can be finished quickly and workers get insight into the steps involved in real engine installation [4].

Many architects use CAD systems to create designs for their projects. CAD tools aid the architect in designing the details of the building like plans, sections, elevations, etc. VR goes much further in the design process by providing an interface which lets the architect walk through the building. This is helpful to explore issues like spacing, placing supporting structures, and so on. Figure 2.1 shows the virtually reconstructed Dudley palace that existed in England in 1550s. Immersive VR lets the user walk inside the palace. The remains of the original palace can be seen in the bottom left corner of the picture [4].

Figure 2.1: Archaeological virtual reconstruction [6].

Science

Researchers in VR investigated the use of immersive VR systems in treating anxiety disorders and different phobias, which disrupt the everyday life of some people. Some persons have a very intense and irrational fear of certain objects or living things or other environments. For treatment, the stimulus causing the fear or anxiety is modeled virtually. The patient then gets into the VE by wearing an HMD. The virtual objects are modeled to be less realistic for the patient's first time exposure to VE. As the treatment goes on and the patient gets more exposed to the VE the level of realism will be increased gradually. This process continues till the patient will be able to cope with the real situation [4].

Interactive molecular modeling and molecular dynamics simulation can be used to dock a drug molecule to its molecular receptor. A researcher in the field of molecular biology can experiment with placing the drug molecule into the active site of the protein, while receiving real time feedback. Figure 2.2 shows the interactive virtual molecular model."Accurate and efficient methods of investigating the recognition and binding of drugs to their biomolecular targets will significantly enhance the drug discovery process" [7].

With the help of virtual visualization techniques, drug designers can interact with the molecular models both visually and aurally. The feedback from the simulation provides a more realistic atomic interaction between the molecules.

Training

Training provided through simulation proves to be more beneficial than other forms. The major advantage is that the person under training or the equipment used in it will not be physically involved in the environment thereby avoiding any risk of injuries or destruction of highly expensive machinery. These simulators are used for planes, air traffic management, submarines, nuclear power plants, fire fighting, etc.

Surgical residents get their experience in operations usually by assisting an experienced senior surgeon, reading the textbooks, and practicing on animals or physical models. But

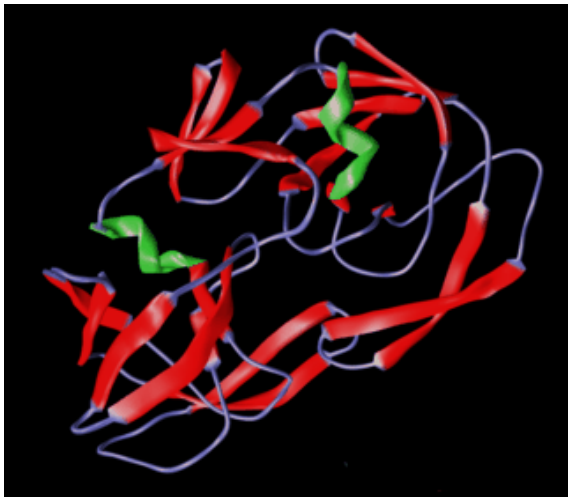


Figure 2.2: Interactive virtual molecular model [7].

assisting during an operation is different from performing the operation, which involves many minute and sensitive details. Also animal labs are expensive and animal anatomy is different from human anatomy [8, 9].

In laparascopic surgery the surgeon operates on the area of interest by looking at a video which is relayed by an endoscope. This procedure demands expertise from the surgeon. Using an immersive VR system the trainee can practice the procedure multiple times to gain hands-on experience. Figure 2.3 shows a trainee performing virtual laparascopic surgery. In the simulation virtual models of the human anatomy can be loaded and multiple views of the area of interest can be provided. Various kinds of disease can also be simulated using a VR system.

2.3 Augmented Reality

Augmented reality, also called mixed reality, is a composition of real objects and computer generated virtual objects. It is a developing area in the field of VR. AR does not try to reproduce the real environment unlike VR. Instead AR only replaces objects in the real

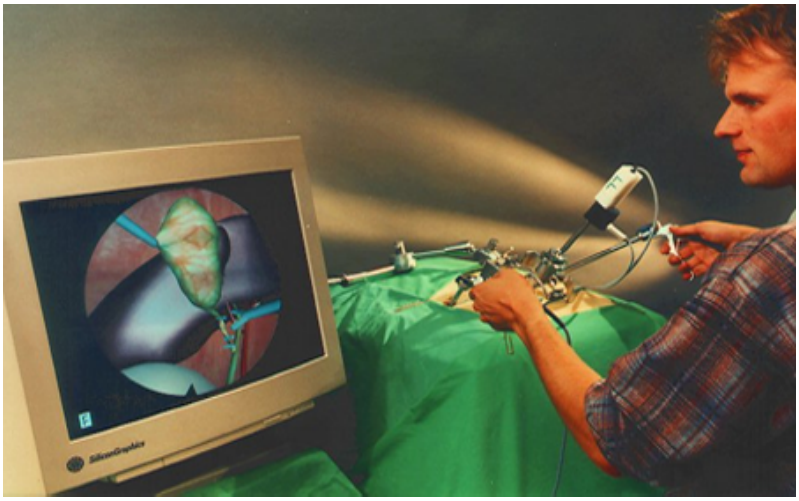


Figure 2.3: Virtual laparoscopic surgery training [8].

world with the virtual objects or adds new objects to the world. AR proves to be more useful when supplied with information from GPS and other Internet devices.

AR is relatively new compared to VR but much progress has been made in this field in the past few years. AR improves the visibility of important objects in the environment. For example, runway markers, which are not clearly visible, may cause runway accidents. So, they can be highlighted using AR. In Figure 2.4 text is annotated to the video of the real scene to help in navigation. AR also applies to aural, tactile and other forms [10, 11].

2.3.1 Creating AR

There are two ways to create an AR application using two types of HMDs: see-through and closed-view. Closed-view HMDs do not allow light from the real world into the user's eyes. The user can only see what is on the monitor of the HMD. See-through HMDs allow the user to partly see the real world and also the virtual objects generated by the IG. The two HMD types using this technique are optical see-through HMDs and video see-through HMDs.

Figure 2.4: Simple example of AR [10].

Optical See-through HMD

An Optical see-through HMD has an optical combiner in front of the user's eyes as shown in Figure 2.5. This combiner is like a half-silvered mirror, which is partially transmissive and partially reflective. This allows the user to see the real world combined with virtual objects. The percentage of real world light that has to be blended with the virtual images depends on the type of the AR application [12].

The tracker, which is attached to the HMD, tracks the location of the user and sends this information to the computer/IG. The computer then produces the virtual objects that are to be rendered at that particular location. The orientation of these virtual objects corresponds to the orientation of the user's head. These images are passed to the monitor of the HMD and then the combiner combines these virtual images with the input light coming from the real world.

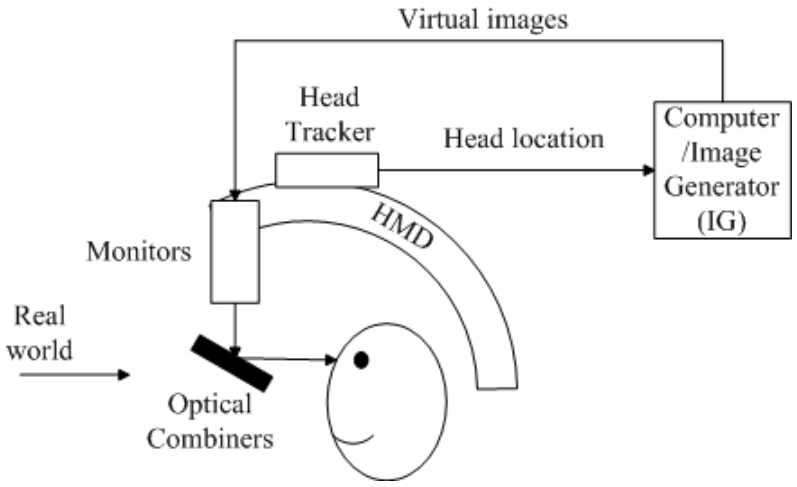


Figure 2.5: AR using an optical see-through HMD.

Video See-through HMD

Video see-through HMDs work by combining an optical see-through HMD with one or two video cameras. AR system with a video HMD is shown in Figure 2.6. Virtual objects are generated by the scene generator/computer according to the orientation and location coordinates of the user's head. These virtual images are combined with the video of the real world in a video compositor. The final output is seen on the monitor of the HMD [12].

Video composition is done using uses techniques like chroma keying or using depth information. Details of the video composition and registration are explained in Chapter 4.

Monitor-based Configuration

AR systems can also be developed using monitor-based configurations as shown in Figure 2.7. Video cameras can be attached to a moving object whose location can be tracked. The video is then combined with the virtual images generated from the computer by a combiner. The user can see the final AR output in a monitor in front of the user's eyes. In order to see the output in stereo the user should wear stereo glasses [12].

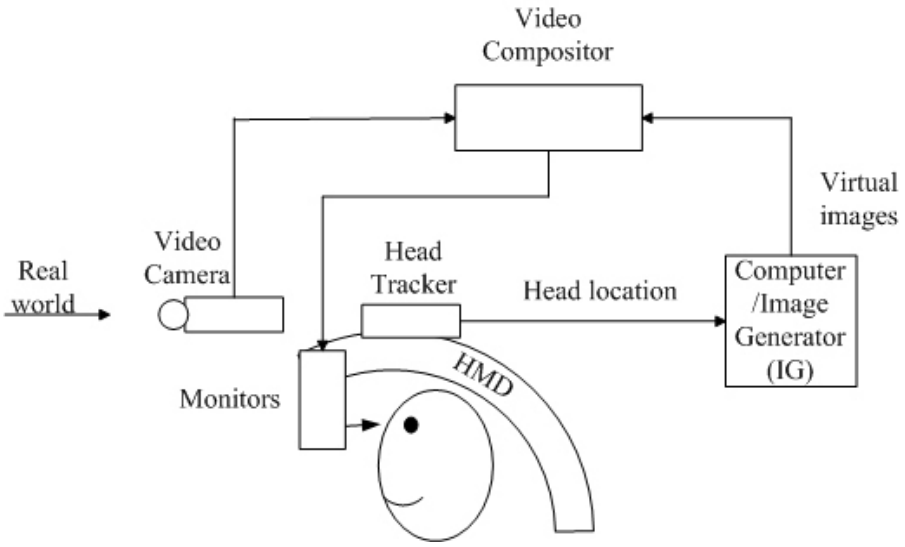


Figure 2.6: AR using the video see-through HMD.

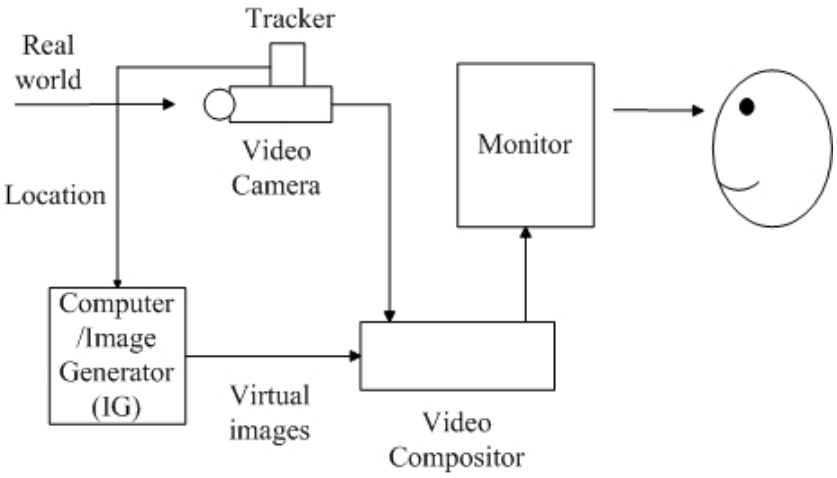


Figure 2.7: AR using the monitor-based configuration.

Comparison between Optical and Video See-through HMDs

The following are some comparisons between the optical and video see-through HMDs, though the type of application determines the HMD to be used [12, 13].

Cost: Optical HMDs (OHMD) are simpler and cheaper than video HMDs (VHMD) because the former do not need video cameras or the video compositor.

Resolution: OHMDs with a narrow field of vision (FoV) have better resolution. Between resolution and FoV one factor has to be traded off for the other depending on the application. In case of VHMDs the resolution of the images is limited to the resolution of the HMD.

Delay: The delay in OHMDs, which is in nanoseconds, is only due to the latency in rendering the virtual objects. Since VHMDs have to deal with two separate data streams for the real and virtual images the delay will be in milliseconds.

Eye offset: In case of VHMDs the cameras act as the eyes of the person using the device. So there will be an offset between what the person intends to see and what is seen.

With OHMDs the virtual objects are transparent and do not completely obscure the physical objects in the scene. Synchronization is a problem with OHMDs because the data stream from the real world cannot be delayed to correspond to the delay caused in rendering the virtual data.

2.3.2 Applications of AR

Engineering

Assembling wiring bundles, which are used in aircraft, is a tedious and complex process and is different for each individual aircraft. Every aircraft needs different bundles in large numbers and different wiring bundles have different configurations [3].

Traditionally, a 2D schematic of the bundle configuration is printed on mylar printouts and glued to a large 8ft. by 3ft. formboard. Connectors are then added at one end of the formboard before placing the wires on the board. Every wire has a serial number printed on it to recognize the slot to which it belongs. Next the assemblyperson creates the connections using the correct wires through proper routes. The pegs placed on the formboard will support the wires on the board.

The virtual systems research and technology group at Boeing Computer Services is trying to use AR in assembling these wiring bundles. A single blank board with an array of holes will be used for assembling the wires. The AR system attached to the person in Figure 2.8 shows the points where the pegs are to be placed on the board. Then the system shows the few connections that are to be made at any junction point, step by step.

The entire process then becomes simple because only a few connections are shown at a time leading to an increase in performance of the workers. Also, the storage space required for the formboards can be drastically reduced because a single board will serve for all the bundles.

Figure 2.9 shows the current state of a historical monument and the virtual reconstruction of the monument augmented to the real scene. Instead of recreating the entire surrounding as in Figure 2.1 this application builds only the missing structure and registers it exactly at the place of the original one. This augmentation can be visualized by wearing a HMD and carrying the AR system to the physical site. It should be noticed that the shadow of a real object is cast on the virtual structure making the scene more realistic. The tracking system for this outdoor AR is based on a GPS receiver [15].

Science

Paleontologists depend on excavated fossils for the study of dinosaurs . Fossils give general information on the animals like shape and size. The biology and locomotion of these animals can be studied by reconstructing the soft tissue, which forms the body of this creature.

Figure 2.8: Assembling wiring bundle using an AR system [14].

Figure 2.9: Augmentation of a virtual structure to the real world [15].

Scientists use mathematical models in order to study the growth of these organisms [16].

Paleontologists can use augmented paleontology (AP), the application of augmented reality to paleontology, to visualize the integration of the reconstructed soft tissue with the fossil structure. They can assess how the different organs in the skull share the limited space, how the muscle contraction of the jaws and bulging of the contracting muscles affects the conformation with the surrounding structures.

In Figure 2.10 the first part shows the skull of a dinosaur. The scanned skull geometry is then registered with the skull in the first part. This is shown in the second part of the picture. Then different muscle groups are augmented to the skull. Paranasal sinuses, eye rings and balls are then filled into the skull, and finally the skin is superimposed. The final form can be seen in the last part of the figure.

Advertising

In football matches shown on television a yellow first-down line can often be seen. This line is being inserted into the real broadcast with the aid of AR. In racecar competitions virtual text with the name of the driver, speed of the car, current place of the driver in the competition etc., can be seen annotated to the video on the television screen [13].

Virtual advertising allows virtual images to be augmented to a television broadcast. Delaying the video broadcast by a few video frames eliminates the registration problems

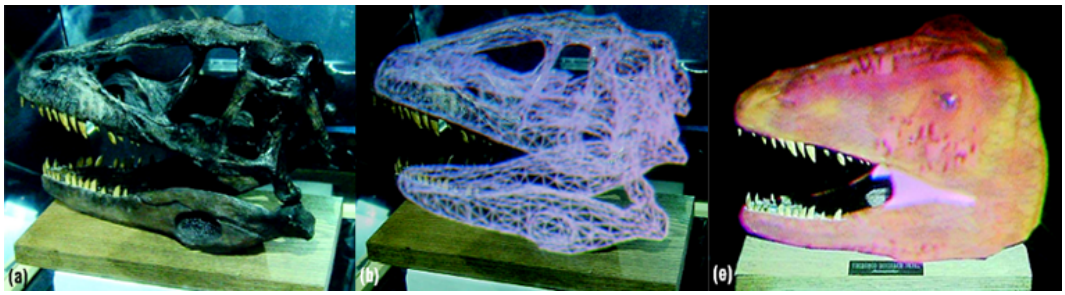


Figure 2.10: Augmented paleontology [16].

Figure 2.11: Virtual advertising [13].

caused by system latency. Video tracking is enough in this application to register the virtual image at the required position. The Pacific Bell advertisement shown in Figure 2.11 is a virtual image augmented to a live video broadcast [13].

2.4 Advantages and Disadvantages of VR and AR

Fewer objects need to be rendered in AR when compared to VR. This is because AR only supplements virtual objects in the real world and does not reproduce the existing environment. In the case of VR each and every pixel has to be rendered placing more demand on system resources.

Creating an AR application is more complex than creating a VR application. In VR all objects are computer generated and there is no problem of synchronization. Since AR deals with combining two sets of data streams there may be errors in the augmentation of the two sets. Tracking should be very accurate and fast in AR. Any disorientation due to improper augmentation can cause dizziness in the user using the system.

In AR the user has to be physically present at the site of the application to visualize

the augmentation, whereas this is not required in VR. The registration and tracking errors may be more serious in the case of a mobile AR system. The display devices and other related hardware of these technologies should be reliable, portable, robust, and cheap for safe regular use [12].

Chapter 3

Virtual Reality Simulation of Chattanooga Creek

3.1 Introduction

Virtual reality can be used for simulating any object and its behavior in the physical world. A VR project can be divided into two parts. The first task consists of creating the objects of interest and the second task deals with attaching behavior or action to those objects. Any 3D model development tool can be used to create the virtual objects. The 3D terrain model, which is the main virtual object in this project, can be built using the DEM and the corresponding DRG available from the USGS.

3.2 Types of Digital Elevation Models

A DEM represents the elevation data of a terrain at regularly spaced intervals with respect to a reference point. Elevation models are divided into three categories: large scale, intermediate scale and small scale. The USGS provides five primary types of elevation data: 7.5 minute DEM, 30 minute DEM, 7.5 minute Alaska DEM, 15 minute Alaska DEM and 1-degree DEM. Each type falls into one of the three categories mentioned above. These

products differ from each other in the reference coordinate system, coverage region and extent of coverage, units of elevation, spacing between the profiles, the manner of data acquisition and the data characteristics. The data set used for this project is a 7.5-minute DEM, which has 30x30 meter data spacing, with Universal Transverse Mercator (UTM) projection. DEMs are also classified into three levels of quality according to the data collection method and the availability of editing for that data set. They are called level 1, level 2 and level 3 DEMs [17]. Figure 3.1 shows the structure of a 7.5-minute DEM.

3.3 Format of the Digital Elevation Model

A DEM consists of three records A, B and C. Record A consists of the name of the DEM, its latitude and longitude, boundaries, scale information, minimum and maximum elevations, number of Type B records and projection parameters. "All type B records of the DEM files are made of data from one-dimensional bands called profiles. The number of complete profiles covering the DEM area is the same as the number of type B records in the DEM " [18]. "Type B records contain profiles of elevation data and the associated header information. Each profile has a type B record " [17]. "Type C records contain statistics on the accuracy of the data" [17]. A DRG is a scanned topographic map and a 7.5-minute map provides the same coverage as the 7.5 minute USGS DEM.

3.3.1 Record A from the 7.5-minute Chattanooga Quadrangle

```
CHATTANOOGA, TN-24000 LAT:: 35 0 0.0000 N LONG:: -85 15 0.0000
W SCALE:: 24000SDTS2DEM v.0.015

1 1 1 16 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 2 1 4 6.48294426D+005 3.874051802D+006 6.48068083D+005
3.887915336D+006 6.59459284D+005 3.888108436D+006 6.59702703D+005
3.874244581D+006 6.34D+002 2.144D+003 0.0D+000 03.00000e+001
3.00000e+0011.00000e+000 1 388
```

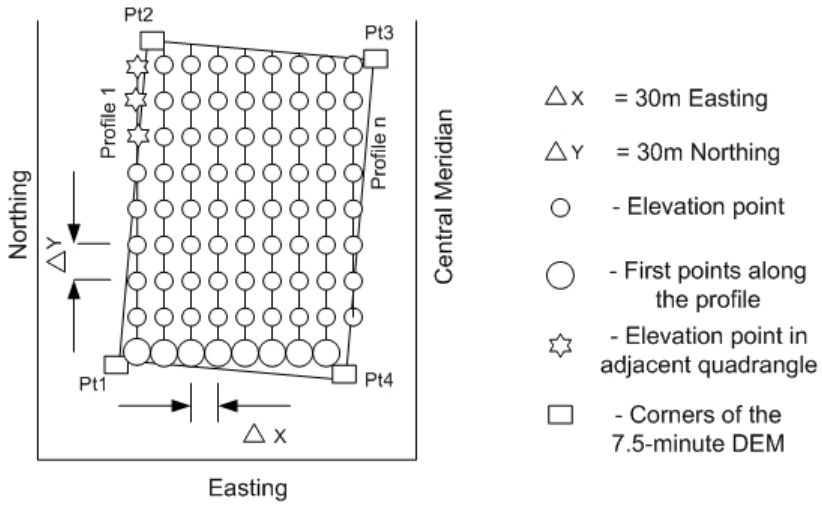


Figure 3.1: Structure of a 7.5-minute digital elevation model.

See Table 3.1 for a description of the Record A of the Chattanooga quadrangle.

3.3.2 First Profile in Record B from the 7.5-minute Chattanooga Quadrangle

```
1 1 45 1 6.4812D+005 3.88656D+006 0.0D+000 6.34D+002 1.507D+003
640 639 634 634 634 634 634 634 634 634 634 634 634 639 639 640
658 680 696 711 723 743 780 804 816 855 897 941 983 1030 1043
1047 1037 1050 1072 1109 1155 1210 1313 1346 1426 1478 1507 1505
1487
```

See Table 3.2 for a description of the first profile in Record B of the Chattanooga Quadrangle.

3.4 Errors in the DEM

DEM data contains three types of errors called blunders, systematic errors and random errors. Blunders are caused during the data collection process because of misreading contours, erroneous correlations or careless observations. These errors are easily recognizable

Table 3.1: Description of content of record A in the DEM file of Chattanooga Quadrangle [17].

Element#	Content	Description
1	`CHATTANOOGA`	Name of the quadrangle covering the area of Chattanooga, TN.
2	`1`	DEM level code; 1 implies this data set is in a standardized format.
3	`1`	Pattern code; 1 indicates a regular elevation pattern.
4	`1`	Planimetric reference system code; 1 indicates UTM coordinate system.
5	`16`	UTM zone of this data set.
6	`0.0` ⋯ `0.0`	Map projection parameters; All values are zero for the UTM coordinate system.
7	`2`	2 represents meters as the units of measure for planimetric coordinates.
8	`1`	1 represents feet as the units of measure for elevation coordinates.
9	`4`	There are 4 sides in the polygon, which defines the coverage of the DEM file.
10	`6.48...D+005` ⋯ `3.87...D+006`	This 4×2 array contains southwest, northwest, northeast and southeast quadrangle coordinates respectively.
11	`6.34D+002` ⋯ `2.144D+003`	Minimum and maximum elevation values in the DEM.
12	`0.0`	Counter clockwise angle from the axis of ground planimetric referenced to the axis of the DEM local reference system.
13	`0`	Indicates that this DEM does not have any records of accuracy, type C records.
14	`3.00000e+001` ⋯ `1.00000e+000`	DEM spatial resolution (x, y, z) set to these values in order.
15	`1 388`	Implies that there are a total of 388 profiles in this DEM in a one-dimensional array.

Table 3.2: Description of the content of first profile in the DEM file of Chattanooga Quadrangle [17].

Element#	Content	Description
1	`1 1`	Row and column identification number of the profile above.
2	`45 1`	Indicates that there are 45 rows and 1 column in this profile.
3	`6.4812D+005` `3.88656D+006`	Ground planimetric coordinates of the first elevation in the profile.
4	`0.0D+000`	Elevation of local datum for the profile.
5	`6.34D+002` `1.507D+003`	Minimum and maximum elevations in the profile.
6	The set of 45 elements from `640` to `1487`	Array of 45×1 elevations in this profile.

because they usually cross the maximum permitted error values and are removed prior to storing the data set. Systematic errors occur during the process of storing the data because of the procedures or systems used for storage and they show up in a pattern. Knowing the cause can help in reducing these errors. The third type of errors, random errors, are random in nature and are unpredictable. All these errors are minimized before they are supplied to the user but are not completely eliminated thereby causing bias and inaccuracy in the terrain [18].

3.5 Creating a Virtual Terrain Model with DEM in NuGraf

NuGraf is used to import the elevation data set and edit that data at either the block level or polygon level. Block- or instance-level editing allows operations on blocks of the data set. Any DEM imported into NuGraf will be divided into multiple blocks, the number of blocks depending on the size of each block. In applications where only a part of the entire 7.5-minute DEM is needed all the unwanted blocks in the data set can be deleted finally

leaving the area of interest. Once the required patch is obtained the polygon level mode can be used to delete the polygons in that block or to apply different textures to different polygons [19].

The quality of the DEM can be improved by changing the data-skip factor and its elevation can be scaled by changing the height-scaling factor. NuGraf builds the mesh structure from the DEM file by taking the data-skip factor into account. A value of 2 for that factor means that every second point in the DEM file is taken into consideration while creating the terrain. Giving a higher value for the factor reduces the quality by leaving empty spaces in the 3D model. It is useful when dealing with a large area and when resolution is not the first priority. Wireframe models of the Chattanooga DEM can be seen in Figures 3.2 and 3.3. The DRG in Figure 3.4 is applied as a texture to the DEM in Figure 3.3. Figures 3.5, 3.6 and 3.7 are the pictures of the 3D terrain after texture mapping.

3.6 Creating a Virtual Simulation Using the 3D Terrain

The 3D terrain model created in NuGraf can be converted into 3D Studio (3DS) or virtual reality modeling language (VRML) formats for use in WorldUp (WUP) to build the simulation. 3DS is more efficient than VRML in terms of memory.

WUP is the development tool used for creating the simulation around the 3D terrain. It provides primitive 3D objects for creating virtual models of real-world objects and a container for holding the objects and developing the simulation. The scripting language, called Basic Script, which comes with a library of built-in functions, is used to attach behavior to the objects in the environment. WUP comes with a set of properties for all of its objects and can be changed during the development phase or run time to update the objects' characteristics. WUP allows interacting with the simulation using input devices like a mouse and also supports the use of output devices like a head mounted display (HMD), for viewing the simulation in an immersive environment. Figure 3.8 shows the steps involved in creating the VR simulation of Chattanooga Creek.

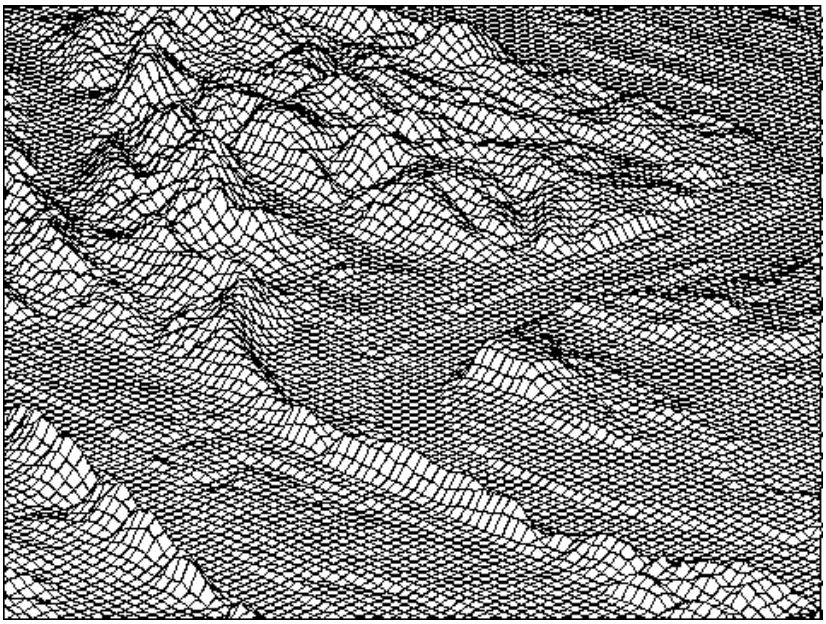


Figure 3.2: Wire frame model of part of 7.5-minute DEM of Chattanooga.

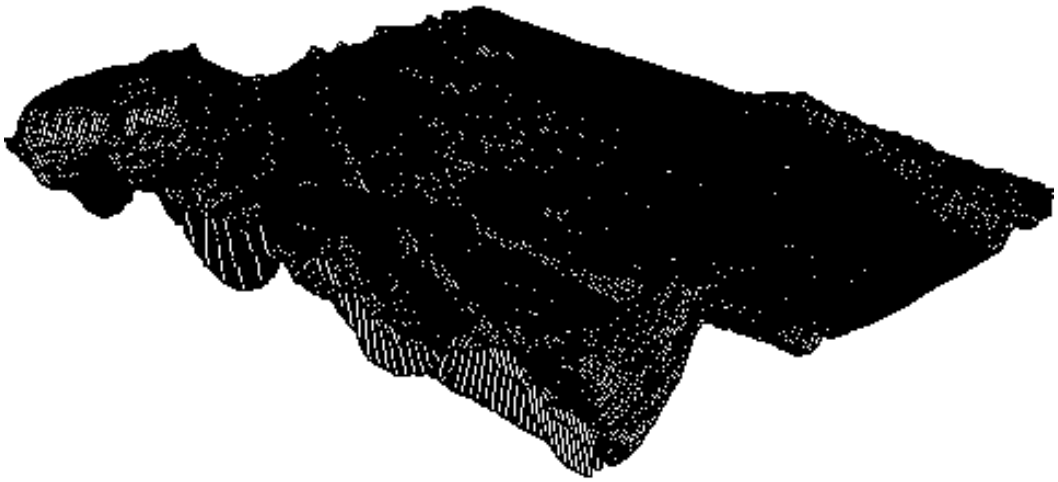


Figure 3.3: Wire frame model of 7.5-minute DEM of Chattanooga.

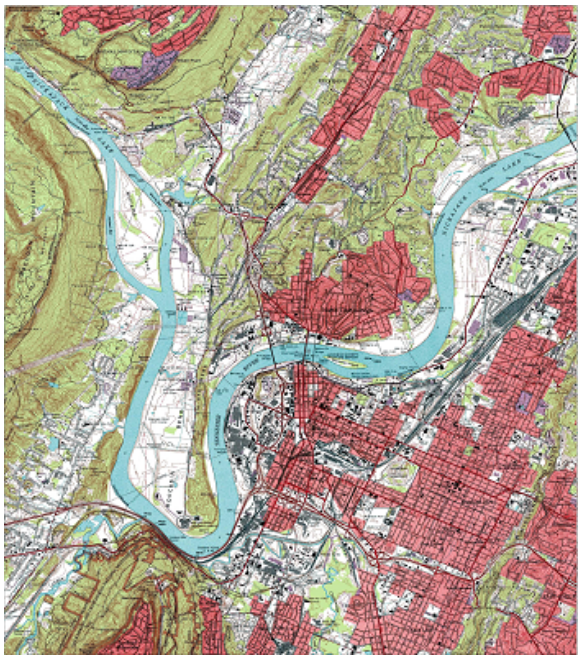


Figure 3.4: 7.5-minute DRG of Chattanooga.

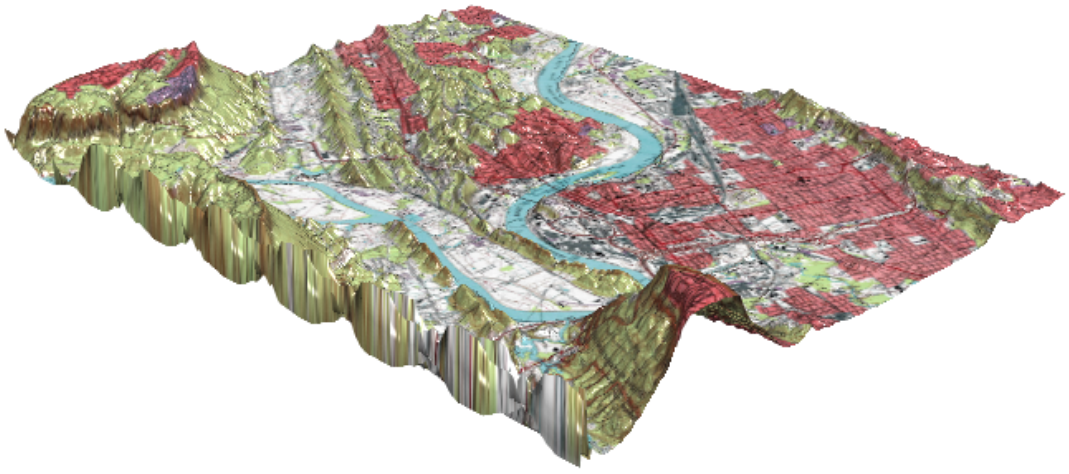


Figure 3.5: 7.5-minute DEM of Chattanooga after texture mapping.

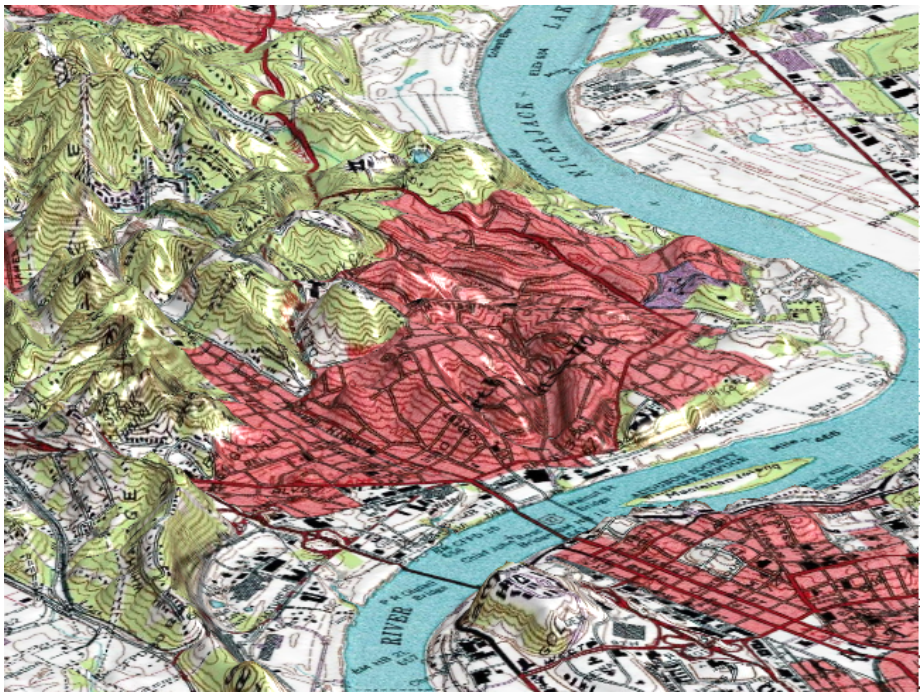


Figure 3.6: Part of 7.5-minute DEM of Chattanooga after texture mapping.

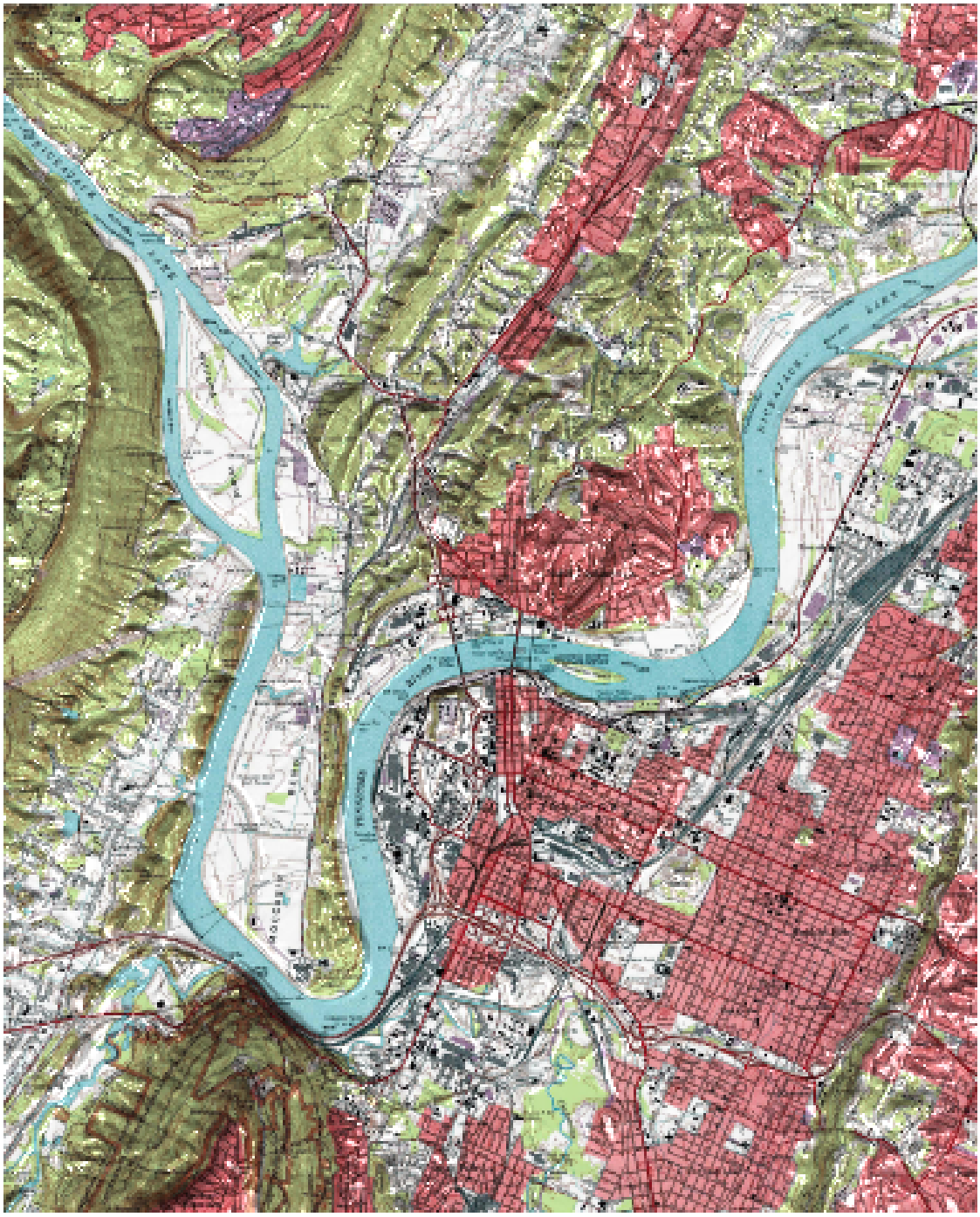


Figure 3.7: Virtual terrain of Chattanooga, TN.

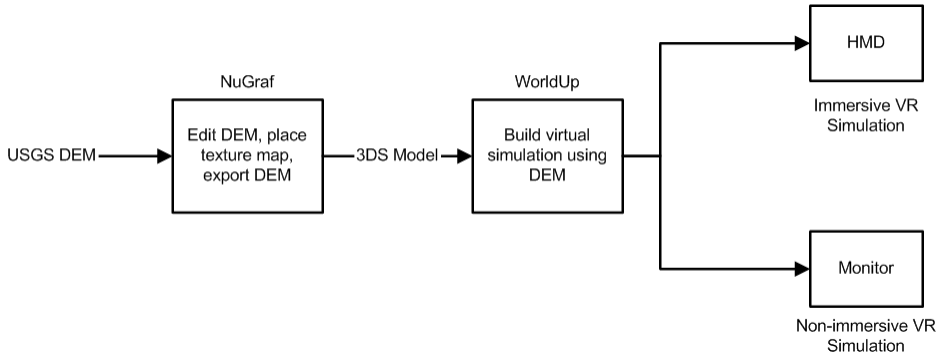


Figure 3.8: Block diagram of virtual simulation of Chattanooga Creek.

3D models that cannot be created in WUP using the primitive objects can be imported from other sources if the file formats are compatible. WUP supports various formats of 3D geometries including 3DS and VRML. The 3D terrain of Chattanooga Creek and the coke plant, which cannot be created in WUP, are imported as a 3DS file from NuGraf.

The purpose of the virtual reality simulation of Chattanooga Creek and the coke plant is to present visually the data that scientists gather and work with and to provide an interface to interact with the virtual environment. The interface must allow the user to perform functions in the virtual environment, which can be done when located physically in the field. A geologist working on this project collects samples of soil from the bores drilled at various points along the creek and the plant area. The data obtained from the samples is entered into files and analyzed for further information. This process is the basis for developing the Chattanooga Creek simulation.

Various functions provided in this simulation appear in the form of button or keyboard controls in the program.

3.7 Functions Provided in the Simulation

Refer to Table 3.3 for the buttons used in the VR simulation.

3.7.1 Button Controls

Select Map

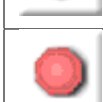
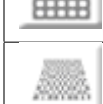
The virtual environment for this simulation consists of the Chattanooga Creek terrain that has been created in NuGraf using the DEM file. The creek and the coke plant areas are in two separate DEM files. Therefore three terrains have been created: one for the creek area, one for the coke plant area and the last one for the joint model of the first two terrains. The three sets can be viewed in the window one at a time by using this button. When pressed this function switches to the next terrain in the cycle and disables the other two thereby keeping only one set enabled at a time.

Import

Sampling locations are represented by cylindrical wells. The locations of the wells are kept track of in UTM coordinates. Adding new sampling points means adding new wells. Wells can be added onto the terrain accordingly using the *import* button. The same function can be used to place historical buildings that were previously located on the coke plant site but were destroyed with time. Hitting the *import* button opens a dialog box, which shows the various types of objects that can be added to the scene. The new object can either be a well, a sensor or a building. The next step opens a new dialog box and asks for the information of the new object like its label, its location in UTM coordinates, and its dimensions. These values are stored as the object's properties. The importing process is finished after clicking the add button, which adds the type of object selected and assigns the value of the label as the object's name.

The import function gets the UTM coordinates entered and checks if that point lies on

Table 3.3: Buttons used in the VR simulation.

Button	Name	Description
	Exit	Exits current simulation.
	Save/Open	Saves/opens current/new simulation.
	Help	Opens help file.
	Decrease Light	Decreases intensity of light in simulation.
	Increase Light	Increases intensity of light in simulation.
	Change Viewpoint	Toggles between multiple viewpoints.
	Drag Mode	Toggles between drag and navigate mode.
	Navigate Mode	Toggles between drag and navigate mode.
	Run Simulation	Toggles between run and stop simulation.
	Stop Simulation	Toggles between run and stop simulation.
	Import Object	Imports new object to the scene.
	Toggle Grid	Toggles coordinate grid.
	Toggle Wire Frame	Shows the objects in wire frame mode.
	Select Map	Toggles between multiple maps.
	Spacer	Dummy button used as a separator.

the terrain. If the point is outside the terrain it gives an error message. The UTM values of the corners of the terrain are stored in a text file and are read from that file during the simulation. If the entered values are valid then the function converts the UTM values to the local terrain translation values. The new object is then constructed using the built-in WorldUp function and added to the environment at the computed terrain translation.

UTM coordinates give the precise location of a point on a large-scale map and are also easy to read. To measure the coordinates of a point on a map, which shows the UTM grid ticks, measure the horizontal distance to the left from the point to its nearest vertical grid line. Identify the vertical grid line's coordinate. Append the distance measured to the coordinate, which gives the point's easting coordinate. Similarly, to measure the northing coordinate of the point, identify the nearest horizontal grid line below the point and measure the distance between the point and the grid line. Append this distance to the line's coordinate, which gives the northing component [20]. In Figure 3.9 the easting coordinate of the point is *652700*. This is obtained by appending the horizontal distance of the point from the grid line, 700 meters, to the grid tick 652. Similarly the northing component of the point is *3874750*.

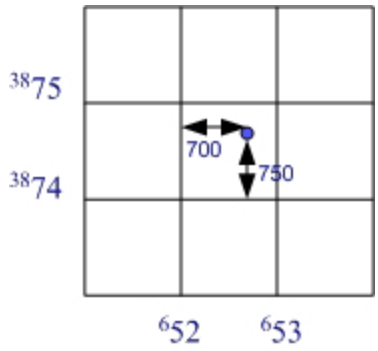


Figure 3.9: Computing UTM coordinates of a point on the grid.

Run/Stop Simulation

This button, when hit, shows the contaminants present at all the sampling points. They are represented using spheres of different colors and sizes depending on the type and amount of the pollutant. This function shows all the wells or sensors on the terrain in the current view. It then reads the contaminant information at each well or sensor from a text file and creates the virtual contaminants beneath the corresponding wells and sensors.

Drag/Navigate

Provision to browse through the virtual environment is essential in order to get a closer view of the terrain and of objects present on it. This operation is provided by a button, which toggles between navigate mode and drag mode, to either go through the scene or to drag the objects in the scene. A series of steps take place when the button is hit in the *drag* mode. The cursor is first changed to an arrow icon to indicate that the mode has changed to *navigate* state from the *drag* state. Next the motion link of the currently-enabled view is obtained and, depending on the angle of the icon, the viewpoint direction is changed when the mouse is moved. Pressing the left mouse button while in navigate mode zooms in or out of the environment depending on the cursor shape and pressing the right mouse button in this mode helps the user to pan above, below, left or right or to any other angle according the cursor direction.

Similarly when the button is hit in the *navigate* mode, the state is changed to *drag* mode. The motion links of all the viewpoints are then disabled and the cursor is changed to a *hand* icon. The selected object becomes active and a bounding box appears around it. The target of the motion link is now changed to the selected object and its position and orientation values are updated according to the movement of the mouse. All this happens so fast that the object can be handled in the virtual world in the same manner as it can be handled in the real world.

A motion link is the link between the source and the target, i.e., the mouse and the view-

point in this example. The position and orientation information of the source is conveyed to the target and the target's position is changed accordingly.

Change Viewpoint

This button cycles between six different views namely perspective, top, front, back, right and left. This helps the user to look at the same terrain in various views because a single view is not sufficient to see the details of the vast terrain clearly. The name and view of the current terrain in the scene is displayed on the top of the screen.

Toggle Wire Frame

This button toggles between showing the terrain in wire frame format and the full textured format. It uses the built-in WorldUp property to change back and forth between these formats.

Increase Light

This button increases the intensity of light used in the scene.

Decrease Light

This button decreases the intensity of light used in the scene.

Help

This button opens a dialog box containing other buttons, which provide links to the project related web sites. It also provides a link to the documentation file of the Chattanooga Creek.

Grid

The grid button gets the currently enabled terrain in the scene, its dimensions, and position and orientation values. It then creates a 3D grid, using 3D lines, below the surface of the

terrain.

Save/Open

This button is used to save or open a simulation.

Exit

The simulation can be exited by using this button.

3.7.2 Keyboard Controls

The following are the keyboard controls provided in the simulation:

L Displays a list of all the objects on the terrain, currently present in the scene, in a dialog box. The focus of the view is changed to the selected object after hitting the *Go To* button. This feature is useful since wells, sensors or any other objects on the terrain are relatively small compared to the terrain and can be missed by the user.

V The terrain may go out of the view sometimes after browsing through a scene. This key helps to get back to the default view. It gets the default view information from a text file.

I Selecting an object in the scene and hitting this key displays the name of the selected object on the top right portion of the window and opens the corresponding Excel file, if the object is a well. The Excel files are opened in a Explorer window and the file can be edited and the changes can be saved.

D Wells or any other objects can be deleted when they are not required in the simulation using this key.

C If multiple objects are needed in the simulation, a single object of that type can be imported and copied using this control for the required number of times.

R Objects like buildings on the terrain, whose orientation matters, can be rotated using this key. This function uses a set of orientation commands provided by WorldUp.

H This key toggles the keyboard control menu. It enables and disables the text display of the commands alternately when the button is pressed.

When an object is selected and dragged using the drag button, the object's current position, in UTM coordinates, is displayed on the top of the screen. This is for the user to know where an object is being dragged. The mouse coordinates over the terrain are converted into the corresponding UTM values for this purpose.

3.8 Errors and Problems in Building the 3D Terrain

3.8.1 Creating a Terrain Using an Entire DEM

A 3D terrain has to be created from the DEM file. The larger the area of terrain is in the application, the larger the memory occupied and the lesser the flexibility is to work with it. The DRG, which sits on top of the DEM, also increases in size with the DEM. A DEM has inherent errors that are mentioned in the previous pages. These errors introduce distortion and empty spaces in the terrain. Though the errors cannot be eliminated, the quality of the terrain can be improved by decreasing the data skip factor in NuGraf. But decreasing the data skip factor drastically increases the memory. So, terrain resolution has to be traded off for less memory. A DRG, which is a digital map, comes in a Tiff file and also occupies a huge space. It therefore has to be converted to a JPEG file since JPEG is the best format in terms of resolution and size when compared to others due to its compression techniques. The final 3D terrain with both the DEM and the DRG is very big except when divided into multiple parts.

3.8.2 Creating a Terrain Using the Center Portion of the DEM

If a terrain has to be created using a whole DEM, it is easy to texture map the DRG onto the DEM. Aligning the DEM and the DRG vertically at one corner aligns the two data sets at every other point. If a part of the DEM, which is not along the edges has to be used for creating the terrain there are two ways to do that. One way is to texture-map the entire DEM with the DRG and then cut the required area. The second way is to cut the DEM chunk that is necessary and then get an exact portion of the DRG that corresponds to the DEM. This is a difficult step because there are no markings on the DEM to find out which point of the DEM corresponds to which point of the DRG. If the exact part of the DRG is not obtained, there will be displacement of DRG on the DEM leading to a weird terrain. For example a river may be misplaced on the elevation of a hill and looks like water is flowing uphill.

3.8.3 Creating a Terrain by Combining Two DEMs

Chattanooga Creek is the area of interest in this project. But part of the creek lies in the Chattanooga quadrangle and the other part lies in the Fort Oglethorpe quadrangle. The DRGs are also present on two separate maps. There are two choices here to create the terrain as mentioned in the paragraph above. One is to map the DEMs with the DRGs, join the two terrains and then cut the portion needed. But this increases the size of the final terrain because the entire texture file is attached to the DEM though only a portion of the DEM is used. The second choice is to cut the areas in the two DEMs, cut the corresponding DRGs for the required areas, map the two DEMs with their DRGs and finally join the terrains. This method is very economical in terms of memory but causes large displacement errors in the 3D terrain.

3.9 Results of the VR Simulation

Figures 3.10 to 3.14 show the results of the virtual reality simulation of Chattanooga Creek. The values shown in Figure 3.10 at the top right corner of the window are the UTM coordinates of the selected well. The selected well can be seen in the white bounding box. In Figure 3.12 the distance between the horizontal and vertical rows of the grid can be changed by the user. Figure 3.13 shows the wells and and contaminants beneath them. The dialog box lists all the objects in the scene and zooms the view to the selected object after hitting the *OK* button.

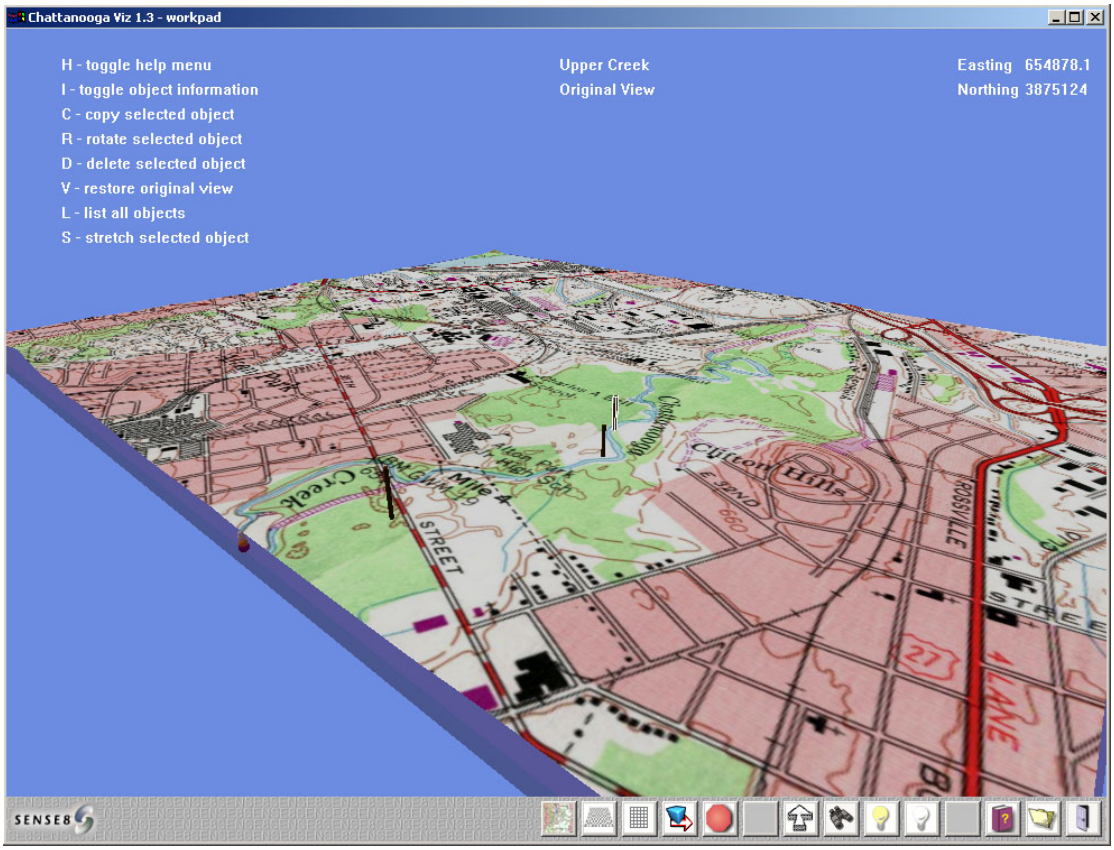


Figure 3.10: Chattanooga Creek virtual reality simulation showing the upper part of the creek.

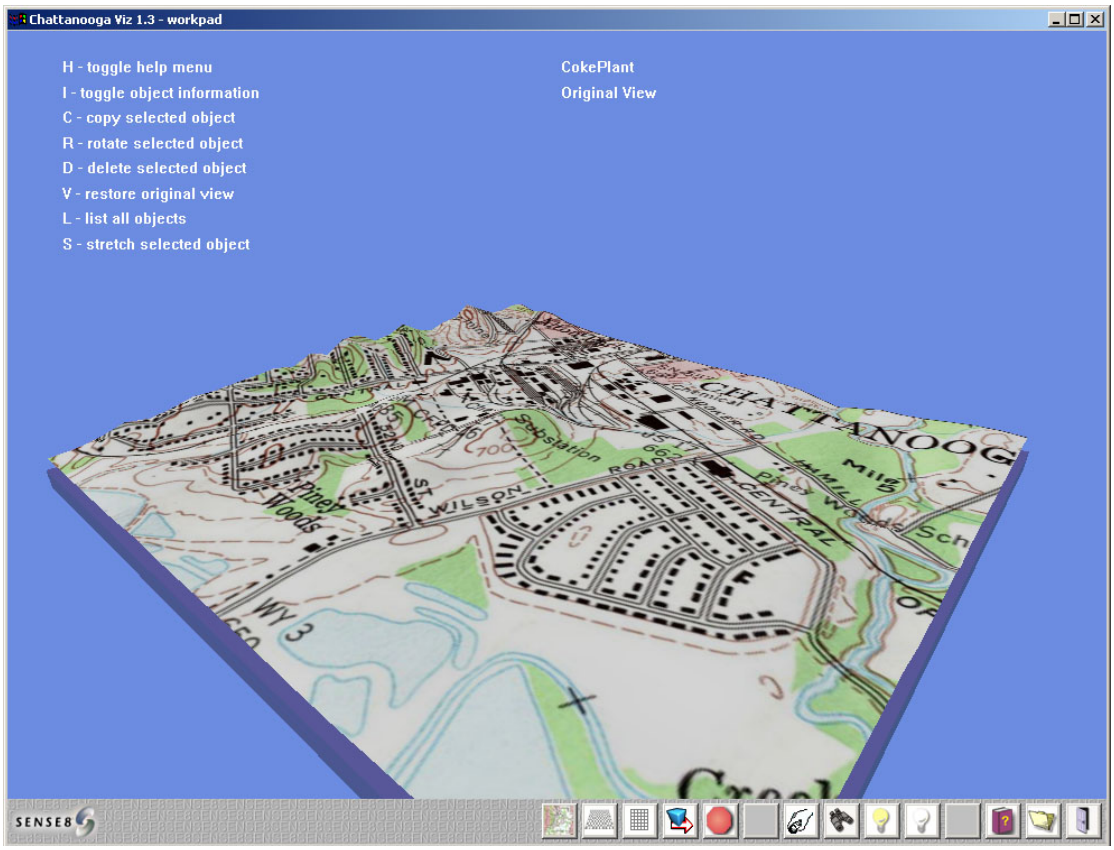


Figure 3.11: Chattanooga Creek virtual reality simulation showing the lower part of the creek and the coke plant.

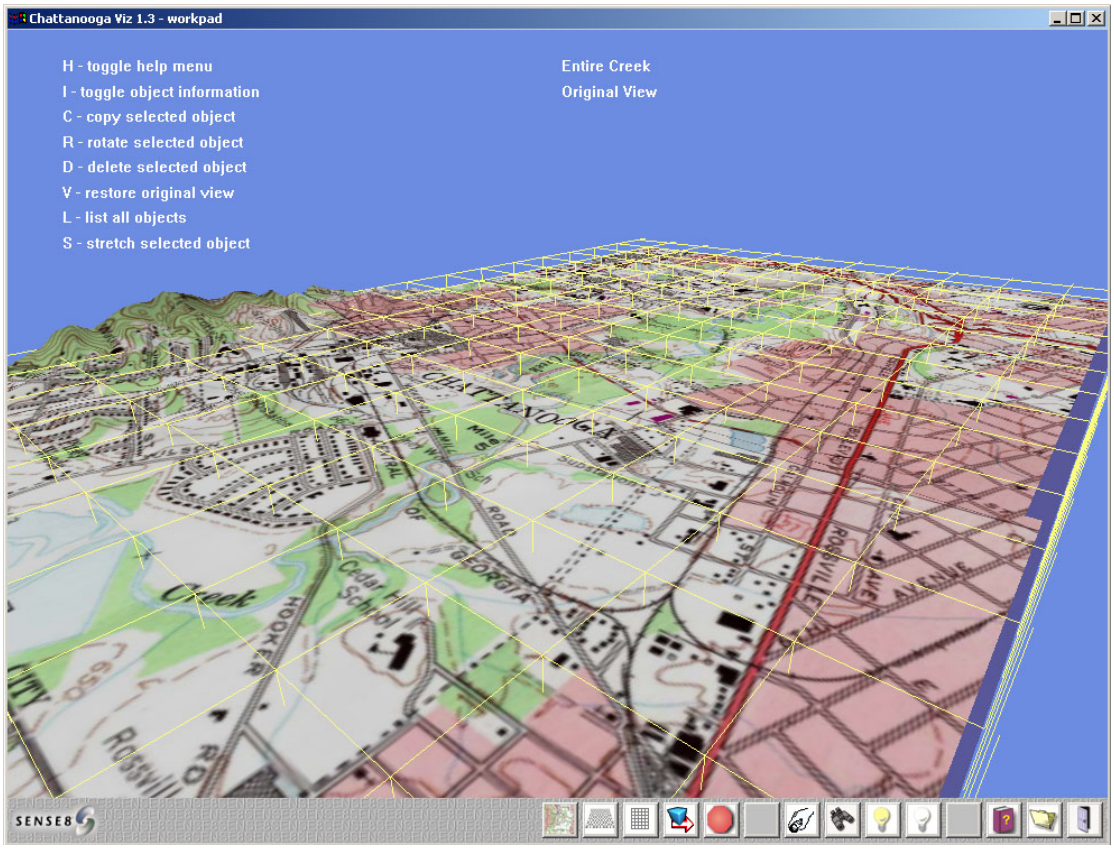


Figure 3.12: Chattanooga Creek virtual reality simulation showing the grid above the surface of the terrain.

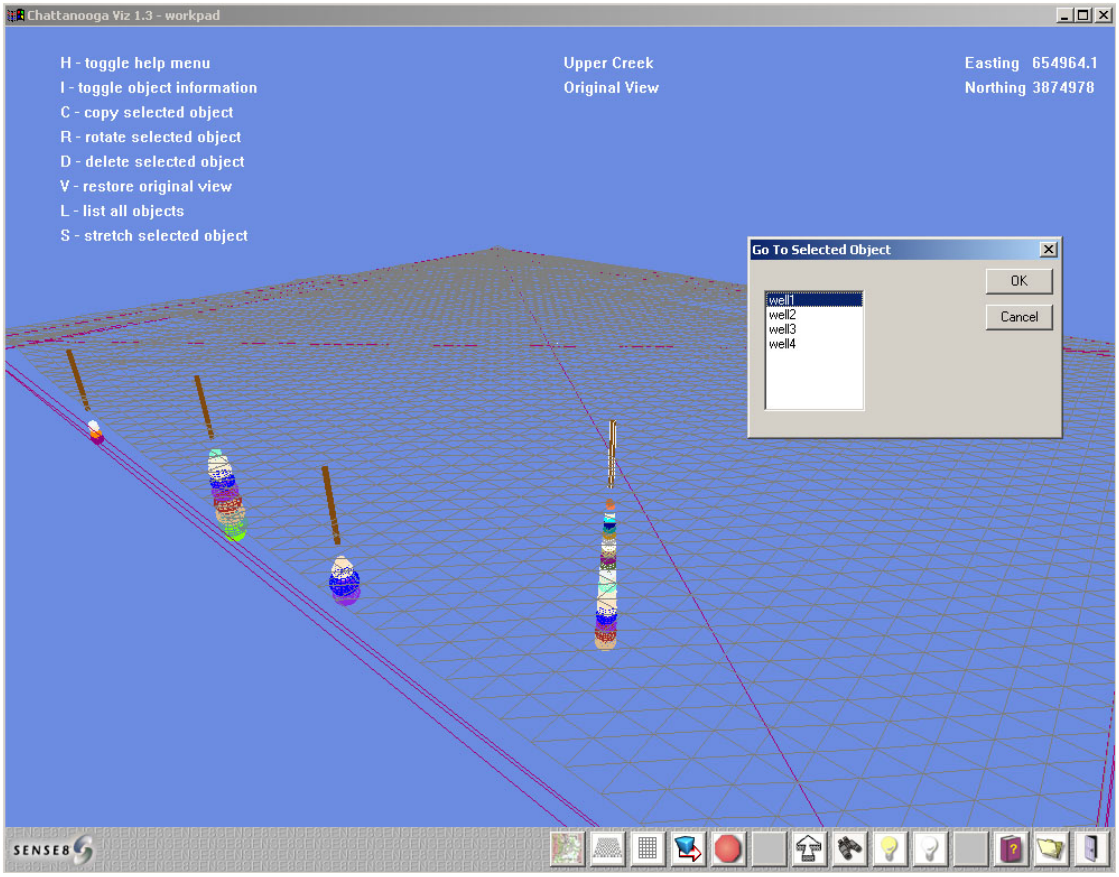


Figure 3.13: Chattanooga Creek virtual reality simulation showing the terrain in a wire frame mode.

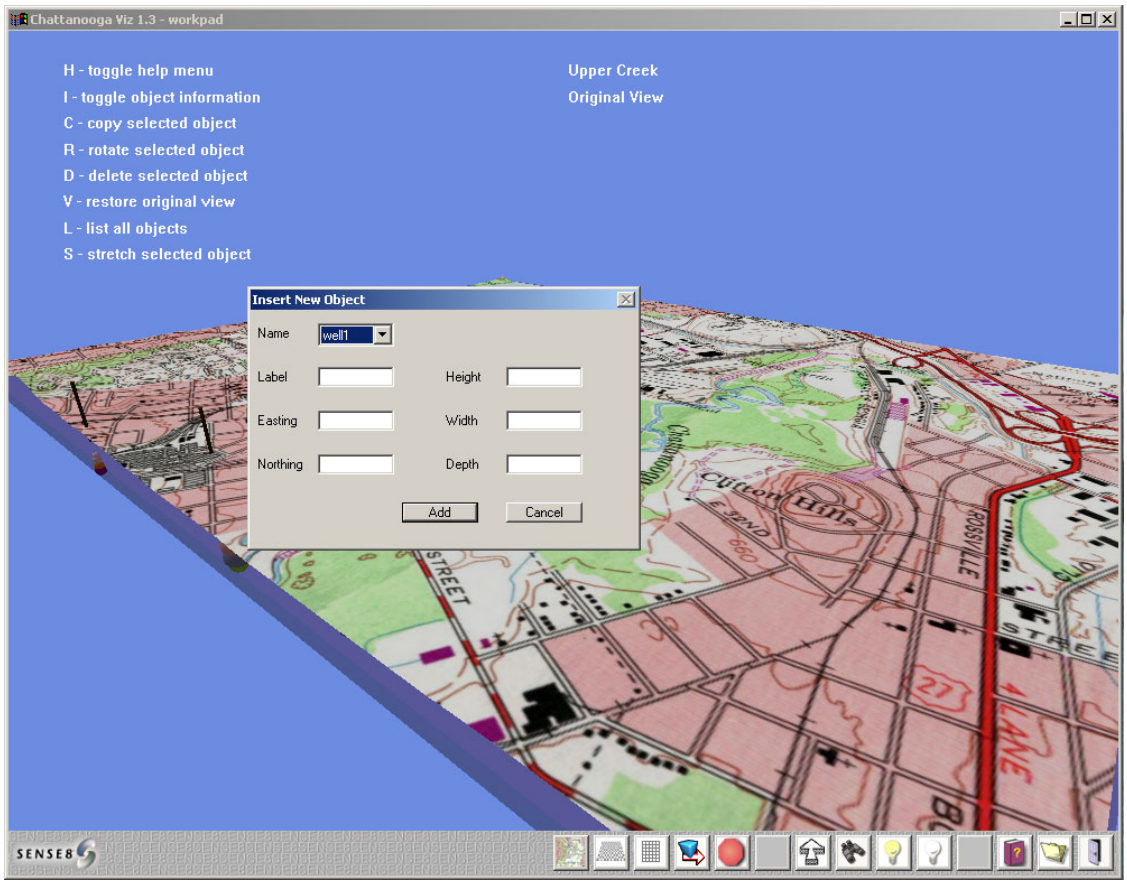


Figure 3.14: Chattanooga Creek virtual reality simulation showing the dialog box to import new objects into the scene.

Chapter 4

Augmented Reality Simulation of Chattanooga Creek

4.1 Introduction

Augmented reality can be broadly divided into two categories: context-dependent AR and context-free AR. Context-dependent AR requires the location coordinates of the person using the AR system. Such systems are complex to implement and usually require sensors. An augmented reality system that needs to augment the name and the associated information of a street or a building on a digital map in the display unit, while the user is navigating, is an example of context-dependent system. A context-free system is independent of the position of the user and does not need sensors. The application that needs to show the instructions on the display unit for assembling machinery is an example of context-free AR [21].

Depending on the demand for accuracy of the position information in an application, augmented reality can also be divided into two groups: finely grained AR and coarsely grained AR. Finely grained AR requires pinpoint accuracy of the position of the object under interest and is mostly needed in surgical applications. Augmentation of the name

and other details on a digital map for help in navigation does not require accurate reading and is called coarsely grained AR. The representation of AR can be determined depending on the accuracy requirements. The virtual image to be augmented can either be a simple line of a text to aid in an assembly process or a complex textured object.

The process of creating an augmented reality simulation changes with the type of the HMD used. The series of steps involved in developing a mobile AR simulation using a video HMD (VHMD) are: tracking the location of the user, synthesizing the virtual objects according to the user's position and combining the two channels of information, real and the virtual, with exact registration.

4.2 Tracking

Tracking involves identifying the location of the person or object using the AR system. The type of tracking differs for indoor and outdoor applications. In the case of indoor AR the physical environment is confined to a limited area and the objects located in that area can be moved to a new position when needed. Also, there are no restrictions on the size, weight, and power consumption of the equipment used since the system will not be a mobile unit. But outdoor applications mean large, unpredictable areas where the environment cannot be modified. In addition, there may be atmospheric disturbances like wind, lightning, etc., which distort the output signal from the tracker device. Factors such as the size of the tracking equipment, weight, and power consumption of the entire system should be taken into account for outdoor applications. The speed at which the mobile AR unit is used plays a major role in the accurate registration of the real and the virtual images. Accurate tracking is different from accurate registration though error in tracking causes error in registration.

Tracking technologies that are used in AR applications can be divided into three classes: active, passive and inertial. Active tracking systems contain signal emitters and sensors that are placed in prepared and calibrated environments. These systems may transmit optical,

magnetic, radio or acoustic signals to the receiver. The short signal sensing range and the interference caused by man-made and natural sources limit the active systems. Passive systems are those, which use markers, fiducials or natural features of the environment as trackers. The signals emitted by these systems occur naturally. Vision systems and compasses that sense intentionally placed or natural markers and earth's field respectively are other examples of passive systems. These systems are limited because of the signal degradation caused due to factors like poor lighting, closeness to the buildings and metallic structures etc. Inertial systems track the position of the user by sensing the person's linear and angular motion. "Inertial systems require double integration of the linear acceleration for computing the position and single integration of the rotation rate for orientation [22]." Each system has its own advantages and limitations and a combination of two or more of the above technologies can be used for accurate results. Tracking systems that are a combination of active, passive or inertial systems are called hybrid systems. Artificial markers, fiducials, etc., are suitable for indoor tracking applications, whereas a GPS unit with a compass is suitable for unprepared and unpredictable outdoor tracking. Normal GPS units are accurate within 30 m whereas differential GPS units are accurate within 3 m. Natural features of the environment can also be used for outdoor tracking in prepared and calibrated environments [22].

4.3 Synthesizing the Virtual Objects

A dynamic renderer should be used to produce the virtual objects that are to be overlayed on the real world scene. Synthesizing the virtual objects involves building the polygon structure for the object and setting the associated properties like texture, location, etc. The position of the virtual objects within the scene of the monitor is computed from the transforms of the graphics and world coordinate systems. Both the real and virtual worlds should be seen from the same frame of reference for correct registration of the virtual images with the real world.

Frame rate, update rate and system lag are a few factors that affect the display in augmented reality applications. The number of times an image is presented to the user per second or number of times per second that a screen is refreshed is called frame rate. The rate at which new images are rendered on the screen is called update rate. These factors depend on the complexity of the virtual objects, software used for creating the virtual objects and also on the underlying hardware such as the image generator and the HMD [2].

4.4 Video Keying

Video keying is a process that combines two channels of images to give a single output signal. In AR it is used to mix the computer-generated graphics with the real world imagery. From the two signals sent to the keyer one acts as a background layer and the other as a foreground layer. Keying can be two types: composite or component. Composite keying takes into account the color, luminance, and synchronization information of the two input channels to output one signal. Component keying combines luminance and synchronization information into a single signal and the chroma information is sent separately. Keying is also divided into two other types, chroma keying and luminance keying, depending on whether color or luminance is used as a keying factor [2].

Video keying is not necessary when an optical see-through HMD (OHMD) is used in the AR system. With OHMD the real world is directly seen through the semi-transparent combiner and only the synthesized virtual object stream is sent to the HMD's monitor. If a VHMD is used for an augmented reality application, a video compositor is required to combine the two sets of information. One is the video signal from the head mounted camera and the other is the computer-generated graphics signal. A VHMD may have one or two cameras. For a monoscopic view, a single camera VHMD is used and for a stereoscopic view a VHMD with the two cameras is used. The calibration parameters of the cameras can be adjusted for obtaining proper parallax.

In chroma keying a foreground key color has to be specified. The background image

replaces the foreground image at places where the key color is present. If red is the key color, the red colored parts in the foreground image become transparent and the background image at that part becomes visible. "A chroma keyer has the ability to replace the pixels in the background image when a particular color is used in the foreground image [2]." From the video and the graphics signals, one can be used as a foreground and the other as a background depending on which signal occupies the major part in the final output.

Luminance keying is similar to chroma keying. It can be either hard or soft keying. A desired threshold value of the foreground luminance (FL) has to be specified in this process. The background luminance (BL) replaces the foreground luminance at places where FL is below the threshold level. This is called hard keying. In soft keying two levels of luminance have to be specified: I_L and I_H. The background replaces the foreground if FL is less than I_L and if FL is greater than I_H , the foreground remains in the output image. For values of FL between I_L and I_H the output image luminance contains a combination of the foreground and background luminance as shown in Figure 4.1 [2].

4.5 Image Registration

The challenging part in creating an AR application is the proper registration of the two sets of information. Finding the location of the synthesized objects within the monitor screen

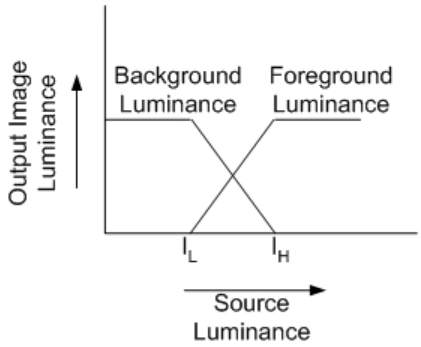


Figure 4.1: Soft luminance keying.

in relation to the real objects is a complex issue. Though the problem of registration exists in AR systems regardless of the type of HMD involved, using a VHMD can reduce the registration errors. This is because of the availability of the video signal for applying image processing or computer vision techniques that help in registration. The depth information of the objects in the real world is necessary for calibrating the real and virtual environments and several frames of reference must be considered for acquiring accurate depth of the objects in the scene. Also, different frames of reference must be taken into account while calibrating the two environments for exact registration of the virtual and real objects.

Video-based systems that have feedback on how closely the virtual and the real images are registered are called closed-loop systems. The image processing techniques are applied to the video signal depending on the received feedback. Systems without such feedback are called open-loop systems [2].

4.5.1 Graphics and World Coordinate Systems

Consistent registration of the real world with the virtual world is necessary for creating a high-fidelity augmented reality application. The relationship between the coordinate systems: object-to-world, O, world-to-camera, C, and camera-to-image plane, P, should be determined before registering the synthetic objects with the real objects. The position and orientation of a virtual object with respect to the world coordinate system is defined by the object-to-world coordinate system as shown in Figure 4.2. The location and orientation of the video camera that is used for viewing the real world is obtained from the world-to-camera transform. The projection parameters required by the camera to create a two-dimensional image of the three-dimensional real world scene are specified by the camera-to-image plane transforms [1].

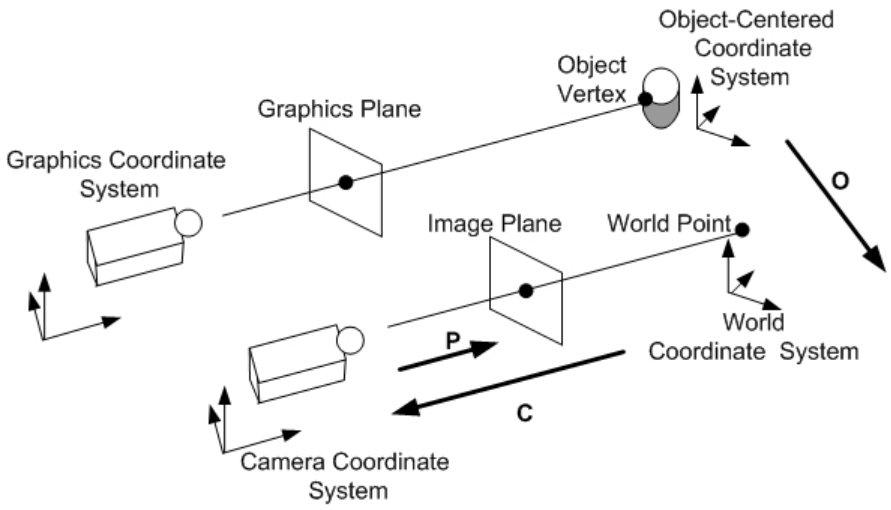


Figure 4.2: Graphics and the world coordinate systems.

4.5.2 Errors in Registration

To create and register virtual images consistently with the user's current view of the world and to place the virtual objects in the real world requires the geometric relationships between the virtual and the physical objects. Registering the virtual images with the real images when there is no motion involved in the user wearing the tracking system is static registration. If the two sets of images have to be registered while the user is in motion, it is called dynamic registration. The errors that occur in the process of registering the real and the virtual images are classified into two types: static and dynamic [2].

Static errors are the errors that cause distortion in the registration of the virtual images even without any movement in the user's head. These errors are caused because of the distortion and mechanical misalignments in the HMD optics, errors in the tracking system, difference in the tracker to eye position and orientation, improper viewing parameters like field-of-view, etc [1]. Static errors cause the virtual objects to be misplaced from the intended location. "The user perceives the static errors as differences in the placement of appearance of the virtual objects when viewed from different viewpoints [2]."

Dynamic errors are the errors that cause distortion in the registration of the virtual images with the real scene when the user is in motion. These errors are caused when the computer-generated graphics stream is not in synchronization with the real world scene or the video signal. Computational, sensing or control latencies resulting in system time lags are the main factors responsible for dynamic errors. The end-to-end system latency is the time difference between the instant at which the user changes his position and the instant at which the virtual objects are rendered on the scene. This is a dynamic error because it has no effect on the simulation until the user moves his head. The virtual objects are rendered on the monitor at improper times because of dynamic errors [1].

4.6 Chattanooga Creek AR Simulation

For the Chattanooga Creek AR simulation, a GPS unit is used as the tracker and WorldUp (WUP) is used as the rendering software. The GPS receiver obtains the location of the user and sends this information to WUP. WUP then displays the position coordinates, in text, on the virtual reality simulation window of Chattanooga Creek. Also, the view of the VR simulation is changed according to the orientation of the GPS unit. An opaque HMD is used as the display unit. The user cannot see the real world through the glasses of this HMD. The input to this device is the VR simulation of Chattanooga Creek augmented with the text of the user's location. This system does not use a video camera and so there is no video signal of the real world. This is a slight deviation from a regular AR application. There is no problem of mismatch in this application while registering the virtual objects with the real world. Figure 4.3 shows the AR system implemented for this thesis.

4.6.1 Transmitting Data from GPS to WUP

The GPS unit, eTrex Vista, from Garmin, is used in the Chattanooga Creek project for tracking the user location. The output from this unit can be in proprietary Garmin format or NMEA, etc. NMEA is a standard, provided by National Marine Electronic Association,

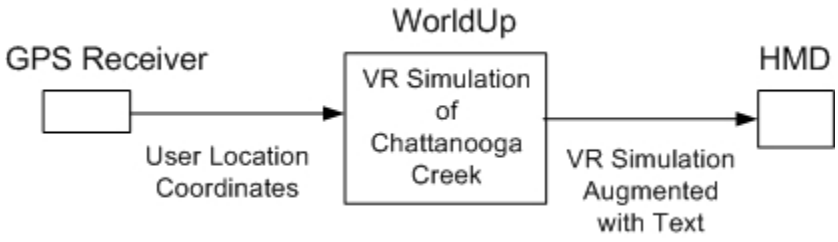


Figure 4.3: Chattanooga Creek AR system.

which defines the interface for communications between marine instrumentation. It consists of ASCII text with parameters like the location, orientation, elevation, speed, local time and so on [23].

The Hyper Terminal program that comes with the windows operating system allows the location coordinates to be transmitted from the GPS to the personal computer (PC). The terminal port settings should be changed to 4800-baud rate, 8-bits, no parity and no handshake according to the specifications provided in the GPS manual. The eTrex Vista connected to the PC with a serial cable should show strings of data, which are updated once every few seconds as the user moves, in the terminal window. This can be seen in Figure 4.4. The orientation coordinates of the user are then taken from the strings in Hyper Terminal and sent to WUP for the rendering of virtual objects at the computed position. The stream of data from the GPS unit can be captured to a text file, in real time, from the Hyper Terminal and the text file can be read by WUP. Or WUP can be set to read the incoming data directly from the serial port.

NMEA Sentences

NMEA information is sent from the *talker*, the GPS, to the *listener*, the PC. The information is transmitted in sentences called *NMEA sentences*. The maximum length of characters in a given sentence is 80. The type of sentences from the talker unit varies with the type of the equipment used. Any talker device outputs a set of standard sentences along with a

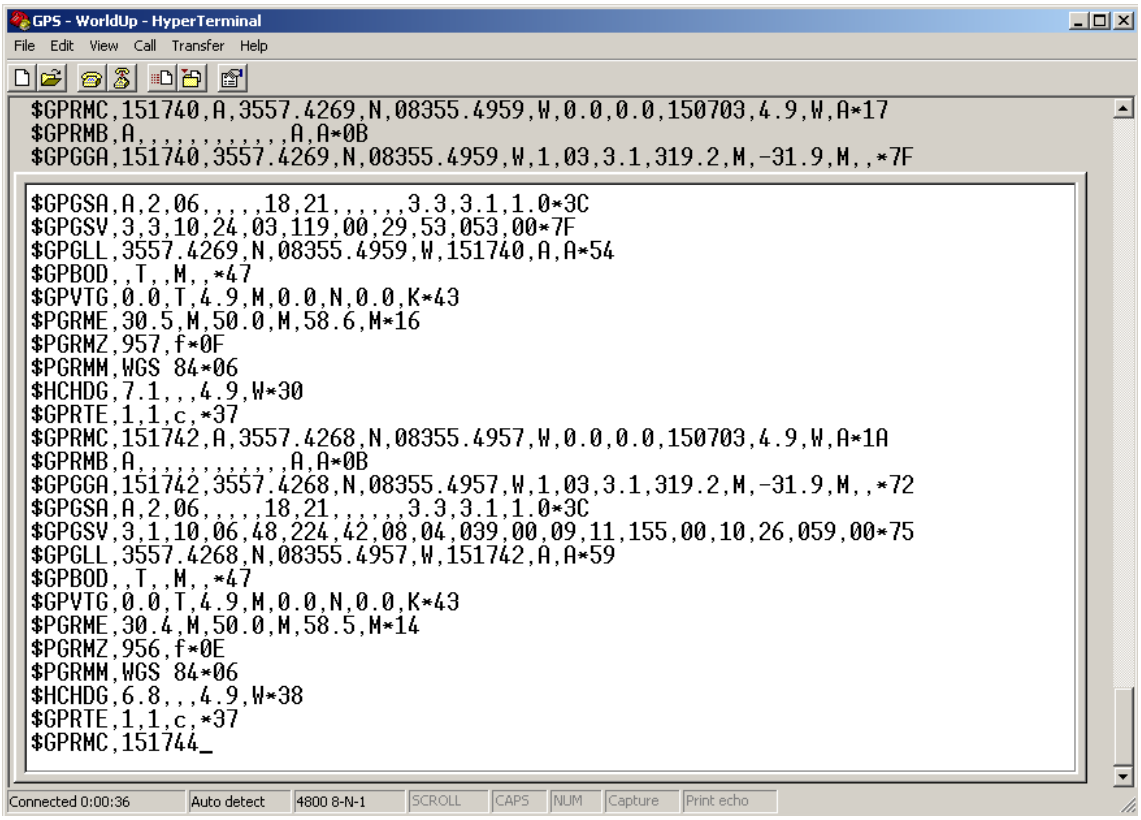


Figure 4.4: Output sentences from eTrex Vista.

set of proprietary sentences. Every sentence starts with a *$*, a two letter *talker ID*, and a three letter *sentence ID*. The information that each sentence provides follows the talker and the sentence IDs. When a data is not available the fields are shown as null but are still separated by comma delimiters [23].

The Garmin eTrex Vista used sends out the following set of sentences to the listener: GPGLL, GPGGA, GPBOD, GPVTG, GPRMB, GPRMC, GPRTE, GPGSA, GPGSV, PGRME, PGRMM, PGRMZ, HCHDG. Figure 4.4 shows the above sentences in the Window's Hyper Terminal. The talker ID *GP* means the GPS receiver, *HC* means the magnetic compass, and *PG* means the proprietary Garmin.

Format of HCHDG Sentence

The following is the structure of the HCHDG sentence, which gives out the orientation information of the user. This can be seen in Figure 4.5 [24].

Field Number:

1. Magnetic Sensor heading in degrees
2. Magnetic Deviation, degrees
3. Magnetic Deviation direction, E = East, W = West
4. Magnetic Variation degrees
5. Magnetic Variation direction, E = East, W = West
6. Checksum

The magnetic sensor heading given out by the HCHDG sentence is read by WUP in order to change the viewpoint of the user in the virtual reality simulation. The location coordinates displayed on the simulation window are read from the GPGLL sentence. The result of the AR simulation that augments the location coordinates of the user on the display window can be seen in Figure 4.6.

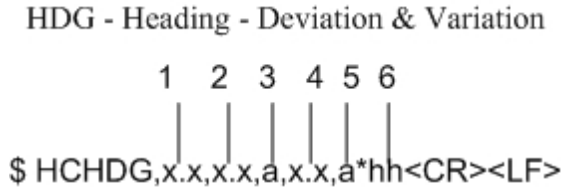


Figure 4.5: Format of HCHDG sentence.

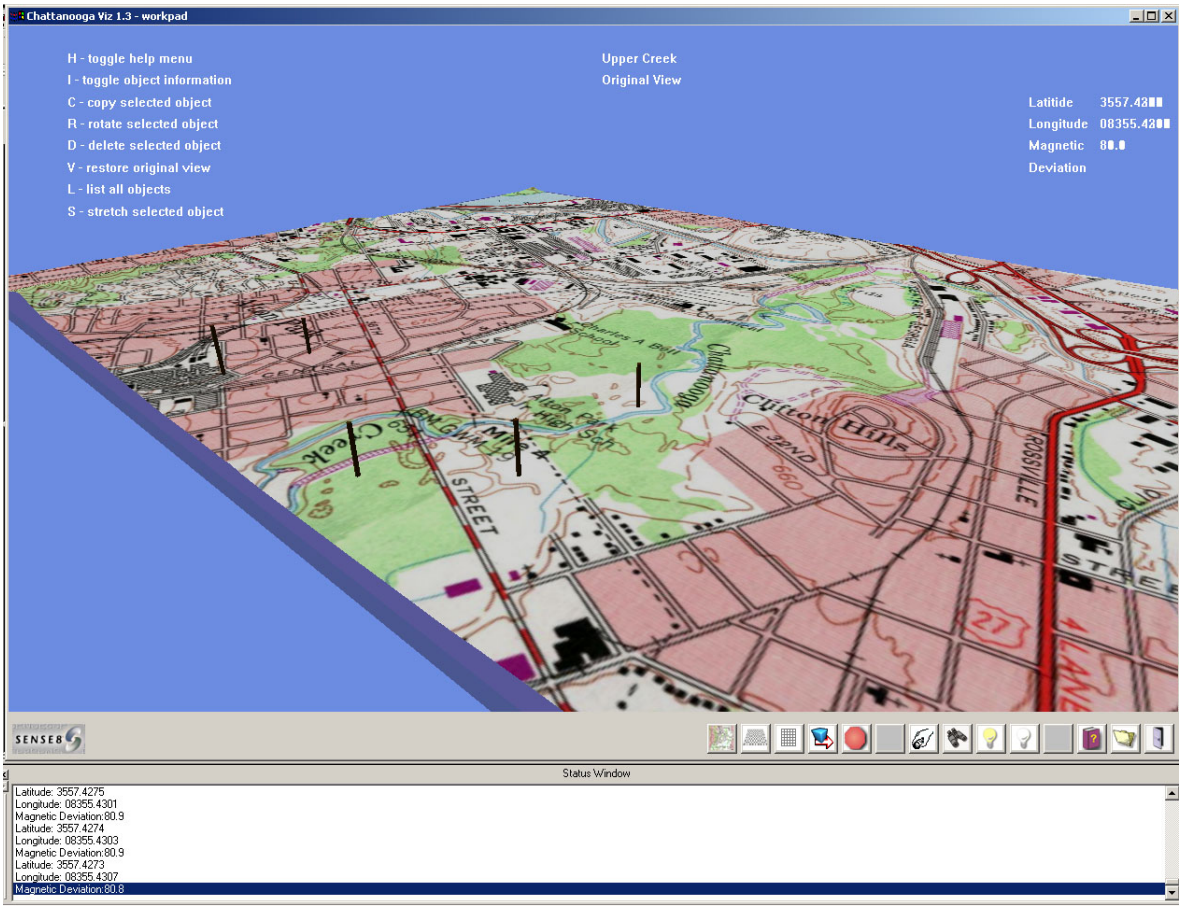


Figure 4.6: Augmentation of the latitude and longitude on the display window.

Chapter 5

Summary and Conclusions

5.1 Contribution

The work done for the Chattanooga Creek project in this thesis is in two parts. One is the virtual reality (VR) simulation and the other is the augmented reality simulation. The VR part includes a user interface with buttons and keyboard controls for the end user to interact with the simulation. It is developed from the Checkpoint Viz software base created at the Applied Visualization Center (AVC) at the University of Tennessee. The AR application is a basic simulation demonstrating augmented reality. It does not provide an interface for the user unlike the VR simulation. It augments the text of the position coordinates on the screen of the VR simulation and changes the viewpoint according to the orientation of the GPS receiver.

5.2 Future Work

The VR simulation can be extended to import new DEMs in addition to importing objects like wells, sensors and buildings. This makes the simulation more useful because it is no longer tied to a specific elevation model. The next step can be to show the contaminant distribution in the soil in the form of contours. This requires the mathematical models of

the flow of the contaminants.

The AR simulation can be developed into a full-scale mobile application. A person can walk through the affected area of the creek, wearing the AR system consisting of an optical see-through HMD, GPS receiver, a laptop and a mouse. The GPS unit identifies the location of the user and sends the position coordinates to the computer. The rendering program, WUP, generates the virtual stream of contaminants and sends it to the combiner in the HMD in front of the user's eyes. This enables the person to view the affected creek area directly along with the virtual contaminants from the computer.

5.3 Conclusions

VR applications place a large demand on the resources of a computer. This is because the real world has to be reproduced with utmost fidelity for these applications. Augmented reality eliminates the need to reproduce every detail of the real world. But it is more complex for the user to implement in terms of dynamic synthesis of objects and registration of the virtual and the real images. Most of the AR systems developed up to now serve indoor applications. Much work has to be done on tracking, image registration, and the display units before these systems can be used outdoors. Applying AR for the contaminant visualization of a terrain is a better choice than using VR if the tracking and image registration are accurate.

Bibliography

Bibliography

[1] J. R. Vallino, *Interactive augmented reality.* PhD thesis, University of Rochester, Rochester, NY, 1998.

[2] W. Barfield and T. A. Furness III, *Virtual environments and advanced interface design.* Oxford Univ. Press, 1995.

[3] W. R. Sherman and A. B. Craig, *Understanding virtual reality interface, application, and design.* Morgan Kauffman Publishers, 2003.

[4] J. Vince, *Virtual Reality Systems.* Addison-Wesley Publishing Company, 1995.

[5] J. A. Adam, "Virtual reality is for real," *IEEE Spectrum*, pp. 22–29, 1993.

[6] C. Johnson, "Introduction to JTA studio, `http://www.imint.freeserve.co.uk/dudcas.htm`."

[7] C. Cruz-Neira and B. Kohlmeyer, "Virtual reality applications center, `http://www.vrac.iastate.edu/research_archive/visualization/molecule/index.php`."

[8] C. Kuhn, "Karlsruhe Endoscopic Surgery Trainer, `http://iregt1.iai.fzk.de/TRAINER/mic_trainer1.html`," 1997.

[9] M. Downes, M. C. Cavusoglu, W. Gantert, L. W. Way, and F. Tendick, "Virtual environments for training critical skills in laparascopic surgery," *Medicine Meets Virtual Reality*, vol. VI, pp. 316–322, 1998.

[10] HowStuffWorks.com, "How augmented reality will work, http://www.howstuffworks.com/augmented-reality.htm."

[11] Popular Science, "Augmented reality, http://www.popsci.com/popsci/computers/article/."

[12] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperatots and virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.

[13] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, pp. 34–47, 2001.

[14] TriSen, "Augmented Reality Company, http://www.trisen.com/applications/wiring.html."

[15] V. Vlahakis, J. Karigiannis, and N. Ioannidis, "Cultivate interactive issue 9: Augmented reality touring of archaeological sites with ARCHEOGUIDE system, http://www.cultivate-int.org/issue9/archeoguide/," 2003.

[16] O. Bimber, S. M. Gatesy, R. R. Lawrence M. Witmer, and L. M. Encarnacao, "Merging fossil specimens with computer-generated information," *IEEE Computer*, vol. 35, no. 9, pp. 25–30, 2002.

[17] U. S. Geological Survey, "Digital elevation models, http://edc.usgs.gov/glis/hyper/guide/usgs_dem."

[18] U. S. Geological Survey, "Digital elevation model standards, http://rockyweb.cr.usgs.gov/nmpstds/demstds.html."

[19] *NuGraf Rendering System User Manual*.

[20] N. G. Terry, Jr., "How to read the Universal Transverse Mercator (UTM) grid, http://www.nps.gov/prwi/readutm.htm."

[21] D. J. Haniff, C. Baber, and W. H. Edmondson, "Categorizing augmented reality systems, http://www.bham.ac.uk/ManMechEng/IEG/ar_cat.pdf."

[22] S. You, U. Neumann, and R. Azuma, "Hybrid inertial and vision tracking for augmented reality registration," in *Proceedings of IEEE VR'99*, (Houston, TX), pp. 260–267, IEEE, 1999.

[23] P. Bennett, "The NMEA FAQ,version 6.4, http://vancouver-webpages.com/peter/nmeafaq.txt," 2003.

[24] K. Vogel and W. Piechulla, "Understanding NMEA 0183, http://pcptpp030.psychologie.uni-regensburg.de/trafficResearch/NMEA0183/," 2002.

Appendix

Appendix

The scripting files in WorldUp are saved with the extension of .ebs. The following pages provide the .ebs files involved in this project.

Start.ebs

```
Public oldPickedObject As Geometry
Public showHelp As Boolean
Public viewptNum As Integer
Public firstCycle As Boolean
Public collisionDetection As Boolean
Public showTime As Boolean
Public state As String
Public fixedMouse As Boolean
Public pathState As String
Public newButtonclicked as Boolean
Public wireframe as Boolean
Public showsgrid as Boolean
Public windowTitle as String
Public showcont as Boolean

Declare Function verifyEntry(entry As String) As Boolean

Sub Main()

showsgrid = False
newButtonclicked = True
showcont = FALSE

Dim win As AppWindow
Set win = GetAppWindow( "Window-1" )
win.NavBarOptions NAVBARSHOW+NAVBUTTONSHIDE+NAVBARMENUHIDE
win.height = 445
win.width = 545
win.leftedge = 12
win.topedge = 12
windowTitle = "Chattanooga Viz 1.3 - " & win.UpFile
win.title = windowTitle
WinMaximize windowTitle
```

```
SetCursor "GRAB"
showHelp = TRUE
firstCycle = TRUE
fixedMouse = FALSE
wireframe = FALSE
viewptNum = 1
Dim viewPt
Set viewPt = GetViewpoint("Viewpoint-1")
Set win.viewpoint = viewPt
collisionDetection = FALSE
showTime = FALSE
xt# = TimeValue(Time$())
message xt
xh# = Hour(xt#)
message xh
xm# = Minute(xt#)
message xm
xs# = Second(xt#)
message xs
newSeed# = xh*xm*xs
'msgbox "The new seed for the RN generator is " & newSeed
Randomize(newSeed)

' Define initial state
state = "viewing"
pathState = "go"

' Setup window and viewport
Dim port as Viewport
Set port = GetViewport( "Window-1-Viewport" )
port.viewportheight = win.height   ' matches up viewport dimensions
port.viewportwidth = win.width     ' with window dimensions

' Disable motionlinks upon startup except 1st one
Dim link as MotionLink
Set link = GetMotionLink( "MotionLink-1" )
link.enabled = TRUE
Set link = GetMotionLink( "MotionLink-2" )
link.enabled = FALSE
Set link = GetMotionLink( "MotionLink-3" )
link.enabled = FALSE
Set link = GetMotionLink( "MotionLink-4" )
link.enabled = FALSE
Set link = GetMotionLink( "MotionLink-5" )
link.enabled = FALSE
Set link = GetMotionLink( "MotionLink-6" )
link.enabled = FALSE

' Setup user buttons and bitmaps
Dim action As Script
```

```
Set action = GetScript( "exitdoorScript" )
win.AddUserButton action, "exitdoor", "Exit"
win.SetUserButtonBitMap "exitdoor", ".\images\door.bmp"

Set action = GetScript( "saveitScript" )
win.AddUserButton action, "saveit", "Save/Open"
win.SetUserButtonBitMap "saveit", ".\images\save.bmp"

Set action = GetScript( "helpmeScript" )
win.AddUserButton action, "help", "Help"
win.SetUserButtonBitMap "help", ".\images\helpme.bmp"

Set action = GetScript( "" )
win.AddUserButton action, "dummy2", ""
win.SetUserButtonBitMap "dummy2", ".\images\null.bmp"

Set action = GetScript( "lightdnScript" )
win.AddUserButton action, "lightdown", "Decrease Light"
win.SetUserButtonBitMap "lightdown", ".\images\lightdn.bmp"

Set action = GetScript( "lightupScript" )
win.AddUserButton action, "lightup", "Increase Light"
win.SetUserButtonBitMap "lightup", ".\images\lightup.bmp"

Set action = GetScript( "chgviewScript" )
win.AddUserButton action, "chgview", "Change Viewpoint"
win.SetUserButtonBitMap "chgview", ".\images\viewpt.bmp"

Set action = GetScript( "chgmodeScript" )
win.AddUserButton action, "ChangeMode", _
"Drag/Navigate"
win.SetUserButtonBitMap "ChangeMode", ".\images\view.bmp"
 'caches 2nd image
win.SetUserButtonBitMap "ChangeMode", ".\images\drag.bmp"

Set action = GetScript( "" )
win.AddUserButton action, "dummy1", ""
win.SetUserButtonBitMap "dummy1", ".\images\null.bmp"

Set action = GetScript( "runstopScript" )
win.AddUserButton action, "contaminant", "Run/Stop Simulation"
win.SetUserButtonBitMap "contaminant", ".\images\golght.bmp"
win.SetUserButtonBitMap "contaminant", ".\images\stoplght.bmp"

Set action = GetScript( "importScript" )
win.AddUserButton action, "import", "Import Object"
win.SetUserButtonBitMap "import", ".\images\import.bmp"

Set action = GetScript( "toglegrdScript" )
win.AddUserButton action, "togglesg", "Toggle Grid"
```

```
win.SetUserButtonBitMap "togglesg", ".\images\coordinatesgrid.bmp"

Set action = GetScript( "toglemapScript" )
win.AddUserButton action, "showlatitude", "Toggle Wireframe"
win.SetUserButtonBitMap "showlatitude", ".\images\grid.bmp"

Set action = GetScript( "selmapScript" )
win.AddUserButton action, "showgrid", "Select Map"
win.SetUserButtonBitMap "showgrid", ".\images\map.bmp"

' Initialize oldPickedObject to avoid errors
Dim iter as Iterator
Set oldPickedObject = GetFirstGeometry(iter)
While oldPickedObject is not nothing
Message oldPickedObject.name
oldPickedObject.BoundingBox = FALSE
Set OldPickedObject = GetNextGeometry(iter)
Wend
Set oldPickedObject = GetFirstGeometry()  'initializes variables

End Sub

' This function verifies the textbox entries of dialog
' boxes to detect illegal characters
Public Function verifyEntry(entry As String) As Boolean

length% = Len(entry)

If (length=0) Then
verifyEntry = FALSE
Exit Function
End If

'Separate string into 1-character elements of an array
Dim test$(length)
num%=1
while num <= length
test(num) = Right$(Left$(entry,num),1)
num = num + 1
wend

'Check each element for illegal characters
num=1
While num <= length
Select Case test(num)
Case ","
verified = FALSE
Case "."
verified = FALSE
Case "/"
```

```
verified = FALSE
Case ";"
verified = FALSE
Case "'"
verified = FALSE
Case "["
verified = FALSE
Case "]"
verified = FALSE
Case "\"
verified = FALSE
Case "`"
verified = FALSE
Case "!"
verified = FALSE
Case "@"
verified = FALSE
Case "#"
verified = FALSE
Case "$"
verified = FALSE
Case "%"
verified = FALSE
Case "^"
verified = FALSE
Case "&"
verified = FALSE
Case "*"
verified = FALSE
Case "("
verified = FALSE
Case ")"
verified = FALSE
Case "="
verified = FALSE
Case "<"
verified = FALSE
Case ">"
verified = FALSE
Case "?"
verified = FALSE
Case ":"
verified = FALSE
Case ""
verified = FALSE
Case "␣"
verified = FALSE
Case "{"
verified = FALSE
Case "}"
```

```
verified = FALSE
Case "|"
verified = FALSE
Case "~"
verified = FALSE
Case Else
verified = TRUE
End Select

If verified = FALSE Then
message test(num)
verifyEntry = FALSE
Exit Function
Else
verifyEntry = TRUE
End If
num = num + 1
Wend

End Function

Saveit.ebs

Declare Function verifyEntry(entry As String) As Boolean
Public windowTitle as String

Sub Main()

Dim theFile$

'Find existing simulation files (.up) in current directory:
Dim simFiles$()
Dim numExistingSims as Integer
FileList simFiles$,"*.*up"
numExistingSims = UBound(simFiles)
Dim existingSims$(numExistingSims)
count%=0

'Remove .up extension for nice viewing in dialog box
While count <= numExistingSims
message simFiles(count)
length% = Len(simFiles(count))
existingSims(count) = Left(simFiles(count),length-3)
message existingSims(count)
count = count + 1
Wend

'Setup 3 dialog boxes

Begin Dialog UserDialog ,,146,77,"Save or Restore Scene"
```

```
PushButton 20,8,32,14,"Save",.SaveButton
PushButton 20,32,32,14,"Open",.OpenButton
CancelButton 16,56,40,14,.CancelButton
Text 56,12,72,8,"the current scene",.Text1
Text 56,36,92,8,"a previously saved scene ",.Text2                    290
End Dialog

Begin Dialog SaveDialog ,,117,134,"Save the Current Scene"
PushButton 8,116,44,14,"Save",.SaveButton
CancelButton 68,116,40,14                                             295
Text 8,8,104,8,"Enter a new name for this scene:",.Text1
ListBox 32,56,52,48,ExistingSims$,.ListBox1
Text 32,44,68,8,"Existing Scenes:",.Text2
TextBox 28,20,56,12,.Filename
End Dialog                                                            300

Begin Dialog RestoreDialog ,,159,97,"Restore a Saved Scene"
PushButton 108,24,40,14,"Open",.OpenButton
CancelButton 108,48,40,14
ListBox 16,24,68,64,ExistingSims$,.ListofFiles                        305
Text 20,4,60,16,"Select a previously saved scene:",.Text1
End Dialog

'Open main dialog box and process user actions
Dim result As Integer                                                 310
Dim SaveRestore as UserDialog
Dim SaveFileDlg as SaveDialog
Dim RestoreFileDlg as RestoreDialog
iterate% = 1
Dim dispError as Boolean                                              315

r = Dialog(SaveRestore)
WinActivate windowTitle

SaveSim:                                                              320
If r = 1 Then              'user wants to save simulation
s% = Dialog(SaveFileDlg)
WinActivate windowTitle

If s = 1 Then                                                         325
theFile$ = SaveFileDlg.Filename
Dim success As Boolean
success = verifyEntry(theFile)
If Not(success) Then
MsgBox "Please enter a scene name with 10 or less                     330
        alphanumeric characters",16,"Invalid Scene Name"
GoTo SaveSim
End If
theFile = theFile + ".up"
If Not theFile$ = ".up" Then                                          335
```

```
If FileExists(thefile) Then
Beep
If theFile$ = "generic.up" Then
temp% = MsgBox ("ERROR: Cannot overwrite original file."
        + Chr(13) + Chr(10) + "Please choose another filename.", _
16,"Error Saving File")
WinActivate windowTitle
Exit Sub
End If
t% = MsgBox ("Filename already exists. Overwrite?", 52,"Overwrite?")
WinActivate windowTitle
If t = 6 Then
Result = SimulationSave(theFile)
End If
Else
result = SimulationSave(theFile)
End If
Else
MsgBox "Error: No filename entered"
WinActivate windowTitle
End If
End If
ElseIf r = 2 Then
'user wants to load a saved simulation
s% = Dialog(RestoreFileDlg)
WinActivate windowTitle

If s = 1 Then
thefile$ = simfiles(RestoreFileDlg.ListofFiles)
if Not thefile$ = "" Then
LoadWorld thefile, TRUE
End if
End If
End If

End Sub

Helpme.ebs

Public windowTitle as string
Sub Main()

Begin Dialog AboutHelpDialog ,,215,252,"For Information on.."
PushButton 80,230,44,14,"Close",.Close
PictureButton 8,16,52,28,".\images\ceb_logo.bmp",0,.CEBPicture
Text 76,20,132,28,"Click the logo to browse the CEB website at
         The University of Tennessee.",.Text2
PictureButton 8,60,52,28,".\images\hydro_logo.bmp",0,.HYDROPicture
Text 76,64,132,28,"Click the logo to browse the Hydrogeology website
         at The University of Tennessee.",.Text3
```

```
PictureButton 8,104,52,28,".\images\avclogo.bmp",0,.AVLPicture
Text 76,108,132,28,"Click␣the␣logo␣to␣browse␣the␣AVC␣website␣at
␣␣␣␣␣␣␣␣The␣University␣of␣Tennessee.",.Text4
PictureButton 8,148,52,28,".\images\readme.bmp",0,.HelpPicture
Text 76,152,136,24,"Click␣the␣Book␣icon␣for␣instructions␣and␣tips␣on
␣␣␣␣␣␣␣␣how␣to␣use␣this␣software.",.Text5
PictureButton 8,192,52,28,".\images\manual.bmp",0,.ManualPicture
Text 76,194,128,20,"Click␣the␣Adobe␣Acrobat␣logo␣to␣view␣a␣description
␣␣␣␣␣␣␣␣about␣the␣project.",.Text1
End Dialog

Dim userDialog As AboutHelpDialog
result% = Dialog (userDialog,2,0)

Dim id As variant

Select Case result
Case 1
WinActivate windowTitle
Exit Sub
Case 2
id = Shell("explorer␣http://www.ceb.utk.edu/",ebNormalFocus)
Case 3
id = Shell("explorer␣http://web.utk.edu/~hydro/",ebNormalFocus)
Case 4
id = Shell("explorer␣http://viz.utk.edu",ebNormalFocus)
Case 5
id = Shell("notepad␣.\readme.txt",ebNormalFocus)
Case 6
id = Shell("explorer␣.\report.pdf",ebNormalFocus)
End Select

End Sub

Root.ebs

Public pickedObject As Geometry
Public oldPickedObject As Geometry
Public pickedPos As Vect3d
Public mouseDown As Boolean
Public collisionDetection as Boolean
Public state As String
Public fixedMouse As Boolean
Public newpos as vect3d
Public finalutm as vect3d
Public showUTM as Boolean
Public newLatitude as string, newLongitude as string
Public newDeviation as string
Public newLatDir as string, newLongDir as string
Public showGPS as Boolean
```

```
Public i as integer, j as integer

Sub Task(obj As Root)

Dim dem as Geometry                                                    440
Dim dem_dims as vect3d, dem_origin as vect3d, dem_trans as vect3d
Dim a1 As Geometry
Dim a2 as geometry
Dim floorgrid2 as geometry
                                                                       445
Set a1 = GetGeometry("chattcreek")
Set a2 = GetGeometry("focreek")
Set floorGrid2 = GetGeometry("wholecreek")

if a1.enabled = TRUE then                                              450
set dem = a1
else
if a2.enabled = TRUE then
set dem = a2
else                                                                   455
if floorGrid2.enabled = TRUE then
set dem = floorGrid2
end if
end if
end if                                                                 460

dem.getDimensions dem_dims
dem.getOrigin dem_origin
dem.getTranslation dem_trans
                                                                       465
If Not(fixedMouse) Then
spf = FrameDuration
If spf > 0.001 Then
'frameduration is valid value
Dim mickey As Mouse                                                    470
Set mickey =GetFirstMouse
mickey.Sensitivity = (203*spf)+28
fixedMouse = TRUE
End If
End If                                                                 475

If state <> "dragging" Then Exit Sub

Dim deviceCtr As Vect3d
Dim offset As Vect3d                                                   480

'Initialize mouse
Dim m As Mouse
Set m = GetFirstMouse
Dim screenPt As Vect2d                                                 485
```

```
m.GetPosition screenPt
Dim win As AppWindow
set win = GetAppWindow("Window-1")

'Display bounding box when device is selected                    490
If m.MiscData = 9 Then
Set oldPickedObject = pickedObject

If oldPickedObject Is Not NOTHING Then
oldPickedObject.BoundingBox = FALSE                             495
End If

Set pickedObject = PickGeometry(screenPt, pickedPos)
Set devObject = CastToDevice(pickedObject)
                                                                500
If devObject Is NOTHING Then
Set pickedObject = NOTHING
Exit Sub
Else
pickedObject.BoundingBox = TRUE                                 505
pickedObject.GetTranslation deviceCtr
Vect3dSubtract deviceCtr, pickedPos, offset
Vect3dPrint offset
win.SetOffset offset
                                                                510
End If
End If

'Toggle mouseDown and remove bounding box
If m.MiscData And Not(mouseDown) then                           515
mouseDown = TRUE
Else
If mouseDown Then
mouseDown = FALSE
Set pickededObject = NOTHING                                    520
End If
End If

'Changes object translation by finding mouse coordinates
Dim lastPt As Vect2d                                            525
win.Getlastpt lastPt

If Not(screenPt.x=lastPt.x And screenPt.y=lastPt.y) Then
win.Setlastpt screenPt
                                                                530
If m.MiscData = LeftHeld And Not(pickedObject Is NOTHING) Then

'Get the current orientation of viewpoint (normal of proj plane)
Dim viewPt As Viewpoint
Set viewPt = win.viewpoint                                      535
```

```
Dim normal As Orientation
viewPt.GetOrientation normal
'Dim newPos As Vect3d
Dim clickOffset As Vect3d
Dim dist as Integer                                                    540
PickPlane screenpt, pickedPos, normal, newPos, dist

'Set the new translation (must first get the current
'location to keep the same global orientation)
Dim origTrans As Vect3d                                                545
pickedObject.GetTranslation origTrans
'newpos.y = origTrans.y       'Get/Save original y translation
win.GetOffset clickOffset
newpos.x = newpos.x + clickOffset.x
newpos.y = newpos.y + clickOffset.y                                    550
newpos.z = newpos.z + clickOffset.z
pickedObject.SetTranslation newpos

'Move Contaminants Along With The Well
Dim pickedwell as Device                                               555
Dim pickedcont() as sphere
Dim contaminants() as string
Dim noofcontaminants as integer
noofcontaminants = 0
set pickedwell = getDevice(pickedObject.Name)                          560

Dim firstTrans as vect3d, nextTrans as vect3D
Dim dimens as vect3D
pickedWell.getDimensions dimens
firstTrans.x = newpos.x                                                565
firstTrans.y = newpos.y
firstTrans.z = newpos.z

If pickedwell.Manufacturer = "Wells" Then
Open ".\scripts\wells.txt" For Input As #13                            570
While not EoF(13)
line Input #13, firstLine$
firstWord$ = Word$(firstLine$,1)
cmpConts$ = pickedWell.Name & "contaminants"
                                                                       575
if firstWord$ = cmpConts$ then
noofcontaminants = wordCount(firstLine$)
Redim contaminants(noofcontaminants-1)
for i = 2 to noofcontaminants
contaminants(i-2) = word$(firstLine$,i)                                580
next i

end if
Wend
Close                                                                  585
```

```
firstTrans.y = firstTrans.y + (dimens.y/2) + 50
If noofcontaminants <> 0 then
Redim pickedCont(noofcontaminants-1)
for i = 0 to noofcontaminants-2
pickedSphere$ = contaminants(i) & "_" & pickedWell.Name & "_cont"
Set pickedCont(i) = getSphere(pickedSphere$)
next i

pickedCont(0).setTranslation firstTrans
for i = 1 to noofcontaminants-2
pickedcont(i-1).getTranslation firstTrans
pickedCont(i).getTranslation nextTrans
nextTrans.x = firstTrans.x
nextTrans.y = firstTrans.y + pickedCont(i).initialradius
nextTrans.z = firstTrans.z
pickedCont(i).setTranslation nextTrans
next i
end if
end if

'End of Move contaminants with the well

'Display UTM coordinates
'Get UTM coordinates for the bottom left corner of the DEM
Dim origutm as vect3d
'Get WorldUp coordinates for the bottom left corner of the DEM
Dim origpos as vect3d
origpos.x = dem_origin.x - (dem_dims.x)/2
origpos.y = 100
origpos.z = dem_origin.z - (dem_dims.z)/2

'Dim finalutm as vect3d
finalutm.x = abs(newpos.x - origpos.x) + origutm.x
finalutm.y = newpos.y
finalutm.z = abs(newpos.z - origpos.z) + origutm.z
showUTM = TRUE
'End of Display UTM Coordinates

'Check for collisions
If collisionDetection Then
Dim collideObj As Device
Set collideObj = CastToDevice(pickedObject)
If collideObj Is Not NOTHING Then
Dim lastPos As Vect3d
collideObj.getlastposition lastPos
If collideObj.intersectsuniverse() Then
collideObj.settranslation lastPos
Else
collideObj.gettranslation lastPos
```

```
collideObj.setlastposition lastPos
End If
End If
End If
else
showUTM = FALSE
End If
End If
End Sub

Chmode.ebs

Public state As String
Public viewPtNum As Integer

Sub Main()

Dim pos As Iterator
Dim iter As Iterator
Dim link As MotionLink
Dim win As Window
count% = 0

Set win = GetWindow( "Window-1" )

If state = "dragging" Then
win.SetUserButtonBitmap "ChangeMode", ".\images\drag.bmp"
state = "viewing"
SetCursor "FORWARD"
'enable correct motionlink
Select Case viewPtNum
Case 1
Set link = GetMotionLink("MotionLink-1")
Case 2
Set link = GetMotionLink("MotionLink-2")
Case 3
Set link = GetMotionLink("MotionLink-3")
Case 4
Set link = GetMotionLink("MotionLink-4")
Case 5
Set link = GetMotionLink("MotionLink-5")
Case 0
Set link = GetMotionLink("MotionLink-6")
End Select
link.enabled = TRUE
Else
win.SetUserButtonBitmap "ChangeMode", ".\images\view.bmp"
state = "dragging"
SetCursor "GRAB"
Set link = GetFirstMotionLink(iter)
```

```
While link Is Not NOTHING
link.enabled = FALSE
Set link = GetNextMotionLink(iter)
Wend
End If

End Sub

Chgview.ebs

Public viewPtNum as Integer
Public state as String
Public firstCycle as Boolean
Public startCount as Boolean
Public tempCount as Integer

Sub Main()

Dim win as Window
Set win = GetWindow( "Window-1" )
Dim port as Viewport
Set port = GetViewport( "Window-1-Viewport" )
port.viewportheight = win.height
'matches up viewport dimensions
port.viewportwidth = win.width
'with window dimensions

Dim viewpt as Viewpoint
Set viewpt = win.viewpoint
Dim link as MotionLink
Dim iter as Iterator

'disable all motionlinks
Set link = GetFirstMotionLink(iter)
While link Is Not Nothing
link.enabled = FALSE
Set link = GetNextMotionLink(iter)
Wend

'get the motionlink attached to the next viewpoint
Dim mlink as MotionLink
Dim pos as Iterator
count%=0

Set mlink = GetFirstMotionLink(pos)
While count < viewPtNum
Set mlink = GetNextMotionLink(pos)
count = count + 1
Wend
```

```
'DETERMINE IF USER HAS CHANGED THE CURRENT VIEWPOINT (bug fix)
If startCount Or firstCycle Then
Dim cdir as Vect3d
viewpt.GetDirection cdir
Dim cpos as Vect3d                                                    740
viewpt.GetPosition cpos
Dim corient as Orientation
viewpt.GetOrientation corient
Dim dirSame as Boolean
Dim posSame as Boolean                                                745
Dim oriSame as Boolean

Dim viewptname as String
Dim dir as Vect3d
Dim orie as Orientation                                               750
Dim posit as Vect3d
Dim foundIt as Boolean
foundIt = FALSE

Open ".\scripts\viewpts.txt" for Input as #1                          755
While Not foundIt
Input #1,viewPtName,dir.x,dir.y,dir.z,orie.x,
        orie.y,orie.z,orie.w,posit.x,posit.y,posit.z
If viewPtName = viewPt.name Then
foundIt = TRUE                                                        760
dirSame = Vect3dEqual(dir,cdir)
posSame = Vect3dEqual(posit,cpos)
oriSame = OrientEqual(orie,corient)
End If
Wend                                                                  765
Close #1

If firstCycle And Not Startcount and
Not (dirSame And posSame And oriSame) Then
startCount = TRUE                                                     770
tempCount = 0
End If

If startCount Then
If tempCount < 5 Then                                                 775
tempCount = tempCount + 1
Else
tempCount = 0
firstCycle = FALSE
startCount = FALSE                                                    780
End If
End If
End If

'determine the next viewpoint                                         785
```

```
Select Case viewPtNum
Case 0
Set viewPt = GetViewpoint("Viewpoint-1")
viewPtNum = 1
Case 1
Set viewPt = GetViewpoint("TopViewpoint")
viewPtNum = 2
Case 2
Set viewPt = GetViewpoint("FrontViewpoint")
viewPtNum = 3
Case 3
Set viewPt = GetViewpoint("RightViewpoint")
viewPtNum = 4
Case 4
Set viewPt = GetViewpoint("BackViewpoint")
viewPtNum = 5
Case 5
Set viewPt = GetViewpoint("LeftViewpoint")
viewPtNum = 0
End Select

Set win.viewpoint = viewPt

If state="viewing" or state="flowviz" Then mlink.enabled = TRUE

If firstCycle Then SendKeys "v" 'fixes bug on first cycle

End Sub
```

Showtext.ebs

```
Public showSpecs As Boolean
Public showHelp as Boolean
Public collisionDetection As Boolean
Public showTime As Boolean
Public pickedObject As Geometry
Public frameTime As Single
Public simTime As Single
Public state As String
Public a1 As Geometry
Public a2 as geometry
Public floorgrid2 as geometry
Public finalutm as vect3d
Public showUTM as Boolean
Public newLatitude as string, newLongitude as string
Public newDeviation as string
Public showGPS as Boolean

Sub Task(win As Window)
```

```
win.SetColor 1,1,1

Dim viewpt as Viewpoint
set viewpt = win.Viewpoint
Dim viewptname as String                                                    840
viewptname = viewpt.name

Dim devObject As Device
Set devObject = CastToDevice(pickedObject)
                                                                            845
If showSpecs Then
If devObject Is Not NOTHING Then
win.DrawText .80,.95, devObject.name
End If
End If                                                                      850

If showHelp Then
win.DrawText .05,.95,"H - toggle help menu"
win.DrawText .05,.92,"I - toggle object information"
win.DrawText .05,.89,"C - copy selected object"                             855
win.DrawText .05,.86,"R - rotate selected object"
win.DrawText .05,.83,"D - delete selected object"
win.DrawText .05,.80,"V - restore original view"
win.DrawText .05,.77,"L - list all objects "
win.DrawText .05,.74,"S - stretch selected object "                         860

'Display the current view name on the window
If viewptname = "Viewpoint-1" Then
win.DrawText .5,0.92,"Original View"
else                                                                        865
if viewptname = "TopViewpoint" Then
win.DrawText .5,.92,"Top View"
else
if viewptname = "BackViewpoint" Then
win.DrawText .5,.92,"North View"                                            870
else
if viewptname = "FrontViewpoint" Then
win.DrawText .5,.92,"South View"
else
if viewptname = "RightViewpoint" Then                                       875
win.DrawText .5,.92,"East View"
else
if viewptname = "LeftViewpoint" Then
win.DrawText .5,.92,"West View"
End If                                                                      880
End If
End If
End If
End If
End If                                                                      885
```

```
'Display the DEM name on the window
If a1.enabled = TRUE Then
win.DrawText .5,0.95,"Upper Creek"
else
If a2.enabled = TRUE Then
win.DrawText .5,0.95,"CokePlant"
else
If floorgrid2.enabled = TRUE Then
win.DrawText .5,0.95,"Entire Creek"
End If
End If
End If
End If
If showUTM = TRUE Then
win.DrawText .86,.95,"Easting"
win.DrawText .92,.95,finalutm.x
win.DrawText .86,.92,"Northing"
win.DrawText .92,.92,finalutm.z
End If

Open ".\scripts\gps5.txt" for Input as #20
While Not EoF(20)
line Input #20, newline$
newword$ = word$(newline$,1)
If newword$ = "GPGLL" then
newLatitude = word$(newline$,2) & "." & word$(newline$,3)
newLatDir = word$(newline$,4)
newLongitude = word$(newline$,5) & "." & word$(newline$,6)
newLongDir = word$(newline$,7)
win.DrawText .86,.89, "Latitide"
win.DrawText .92,.89,newLatitude
win.DrawText .86,.86,"Longitude"
win.DrawText .92,.86,newLongitude
else
if newword$ = "HCHDG" then
newDeviation = word$(newline$,2) & "." & word$(newline$,3)
win.DrawText .86,.83,"Magnetic"
win.Drawtext .86,.80,"Deviation"
win.DrawText .92,.83,newdeviation
else
if newword$ = "    " then
Message" No data available from the GPS"
end if
end if
end if

Wend
Close #20
```

```
simTime = simTime + frameTime
If showTime Then
min% = simTime/60
If min*60 > simTime Then min = min - 1
hr% = simTime/3600
If hr*3600 > simTime Then hr = hr - 1
sec% = simTime - (3600*hr) - (60*min)
If sec < 0 Then sec = sec + 60
If min < 10 Then
If sec < 10 Then
digClock$ = hr & ":0" & min & ":0" & sec
Else
digClock$ = hr & ":0" & min & ":" & sec
End If
Else
If sec < 10 Then
digClock$ = hr & ":" & min & ":0" & sec
Else
digClock$ = hr & ":" & min & ":" & sec
End If
End If
win.DrawText 0.85,.1,digClock
End If


End Sub

Viewcurs.ebs

Declare Sub SetCursorBasedOnCode (code as Integer)
Public state as String

Sub Task(obj as Root)
Dim code as Integer
code = 0

If state<>"viewing" AND state<>"flowviz" Then Exit Sub

Dim mousePos as Vect2d
Dim win as Window
Set win = GetWindow("Window-1")
win.GetMousePosition mousePos

Dim windowWidth as Integer, windowHeight as Integer
windowWidth = win.ClientWidth
windowHeight = win.ClientHeight
If windowWidth = 0 or windowHeight = 0 Then Exit Sub

' Convert to percentage of screen
mousePos.X = mousePos.X / windowWidth
```

```
mousePos.Y = mousePos.Y / windowHeight

If mousePos.X > 0 And mousePos.X < 1
        And mousePos.Y > 0 And mousePos.Y < 1 Then
code = 0                                                    990
If mousePos.Y < .49 Then
code = 1
ElseIf mousePos.Y > .51 Then
code = 2
End If                                                      995

If mousePos.X < .35 Then
code = code + 3
ElseIf mousePos.X > .65 Then
code = code + 6                                             1000
End If
Else
code = 9
End If
                                                            1005
SetCursorBasedOnCode code

End Sub

                                                            1010
Sub SetCursorBasedOnCode( code as Integer )
Select Case code
Case 0
SetCursor "FORWARD"
Case 1                                                      1015
SetCursor "FORWARD"
Case 2
SetCursor "BACKWARD"
Case 3
SetCursor "TURN_LEFT"                                       1020
Case 4
SetCursor "FORWARD_LEFT"
Case 5
SetCursor "BACK_LEFT"
Case 6                                                      1025
SetCursor "TURN_RIGHT"
Case 7
SetCursor "FORWARD_RIGHT"
Case 8
SetCursor "BACK_RIGHT"                                      1030
Case 9
SetCursor "ARROW"
End Select
End Sub
                                                            1035
```

Keyboard.ebs

```
Public showSpecs as Boolean
Public showHelp as Boolean
Public viewPtNum as Integer
Public pickedObject as geometry
Public collisionDetection as Boolean
Public state as String
Public windowTitle as String
Public a1 As Geometry
Public a2 as geometry
Public floorgrid2 as geometry
Public newname as string
Public objgroup as Node

Public cont_trans as vect3D
Public dimens As vect3d

Declare Sub CopyObject (oldObject as Geometry)
Declare Function FixHeight(newDevice as Device) as Boolean
Declare Function FindNewName
        (origObject as String) as String
Declare Function ReadSpecs
        (newName as string, origObject as String) As Boolean

Public colors() as string
Public pollutants() as string

Begin Dialog RotateDialog ,,126,80,"Rotate Object"
OptionGroup .Direction
OptionButton 24,24,56,8,"Clockwise",.CW
OptionButton 24,36,76,8,"Counter-Clockwise",.CCW
TextBox 84,4,36,12,.RotationAngle
Text 12,8,68,8,"Enter Rotation Angle:",.Text1
PushButton 12,56,44,14,"Rotate",.Rotate
PushButton 68,56,44,14,"Cancel",.Cancel
End Dialog

Sub Task(root as root)
Dim id as Variant
k$ = UCase$(GetKey()) 'converts entry to upper case
If k = "" Then Exit Sub
If k = "," Then k = "<"
If k = "." Then k = ">"
counti% = 0
Dim theLight as Light
Dim noofcolors as integer
noofcolors = 0
Open ".\scripts\colors.txt" For Input As #12
while Not EoF(12)
```

```
line Input #12, color$
noofcolors = noofcolors + 1
wend
close
Redim colors(noofcolors)
Redim Pollutants(noofcolors)
Open ".\scripts\colors.txt" For Input As #13
while Not EoF(13)
for i = 0 to (noofcolors-1)
line Input #13, color$
colors(i) = Word$(color$,3)
pollutants(i) = Word$(color$,2)
next i
wend
Close

Dim objcts as list
Dim windoh as Window
Set windoh = GetWindow("Window-1")
If a1.enabled = TRUE Then
set objgroup = getNode("chatt_objects")
else
If a2.enabled = TRUE Then
set objgroup = getNode("fo_objects")
else
If Floorgrid2.enabled = TRUE Then
set objgroup = getNode("whole_objects")
End If
End If
End If

Select Case k
Case "H"
If showHelp Then
showHelp = FALSE
ElseIf showHelp = FALSE Then
showHelp = TRUE
End If

Case "I"
filename$ = pickedObject.name + ".xls"
If showSpecs = TRUE Then
showSpecs = FALSE
else
If showSpecs = FALSE Then
id = shell("Explorer" &"␣" &
        ".\data\"&filename , ebNormalFocus)
showspecs = TRUE
end if
end if
```

```
Case "D"
Dim pickDev as Device
Set pickDev = CastToDevice(pickedObject)
Dim pickedCont() as Sphere
Dim contaminants() as string
Dim noofcontaminants as integer
noofcontaminants = 0

If pickDev Is Not Nothing Then
z%=MsgBox("Do you really want to delete the" &
        " " & pickdev.name & "?", 52, "Are you sure?")
WinActivate windowTitle
If z = 6 Then
DeleteObject PickedObject
Set PickedObject = Nothing
If pickdev.Manufacturer = "Wells" Then
Open ".\scripts\wells.txt" For Input As #13
While not EoF(13)
line Input #13, firstLine$
firstWord$ = Word$(firstLine$,1)
cmpConts$ = pickDev.Name & "contaminants"

if firstWord$ = cmpConts$ then
noofcontaminants = wordCount(firstLine$)
Redim contaminants(noofcontaminants-1)
for i = 2 to noofcontaminants
contaminants(i-2) = word$(firstLine$,i)
next i

end if
Wend
Close
If noofcontaminants <> 0 then
Redim pickedCont(noofcontaminants-1)
for i = 0 to noofcontaminants-2
pickedSphere$ = contaminants(i) &
        "_" & pickDev.Name & "_cont"
Set pickedCont(i) = getSphere(pickedSphere$)
DeleteObject pickedcont(i)
Set pickedcont(i) = Nothing
next i
else
Message "KeyBoard Script:No Contaminants to Delete"
End If
End if

else
Message "You selected not to delete the object"
End If
```

```
End If

Case "C"
Dim pickedDev as Device
Set pickedDev = CastToDevice(pickedObject)
If pickedDev Is Not Nothing Then
z%=MsgBox("Do you really want to copy" & "␣the
␣␣␣␣␣␣␣␣" & pickeddev.name & "?", 36, "Are you sure?")
If z = 7 Then
'MsgBox "Operation Canceled"
ElseIf z = 6 Then
CopyObject pickedObject
End If
WinActivate windowTitle
End If
Case "R"
If pickedObject Is Not Nothing Then
Dim ori as Orientation
Dim roty as Single
Dim addori as Orientation
Dim newori as Orientation
Dim userdialog as RotateDialog
userDialog.rotationangle = "90"
action% = Dialog (userdialog,1,0)
WinActivate windowTitle
roty = Csng(userdialog.rotationangle)
If (userdialog.direction=1) Then roty=roty*-1
If action = 1 Then
PickedObject.getrotation ori
OrientSet addori,0,roty,0
OrientAdd ori,addori,newori
PickedObject.setRotation newori
End If
End If
Case "S"
Dim pickedObj as Device
Set pickedObj = CastToDevice(pickedObject)
If pickedObj is Not Nothing Then
Begin Dialog UserDialog4 ,,125,65,
        "Stretch Selected Object"
OKButton 80,12.5,35,12
CancelButton 80,32,35,12
Text 6,10,20,10,"X:"
TextBox 26,10,25,11,.TextBox1,0
Text 6,25,20,10,"Y:"
TextBox 26,25,25,11,.TextBox2,0
Text 6,40,20,10,"Z:"
TextBox 26,40,25,11,.TextBox3,0
End Dialog
Dim strchObj As UserDialog4
```

```
Dim strchParams as Vect3D
res% = Dialog(strchObj,1,0)
If res = -1 Then
strchParams.x = strchObj.Textbox1
strchParams.y = strchObj.Textbox2                                    1240
strchParams.z = strchObj.Textbox3
pickedObj.setScale strchParams

Else
If res = 0 Then                                                      1245
exit sub
End If
End If
End If
Case "V"                                                             1250
Dim win as Window
Dim viewPt as Viewpoint
Set win = GetWindow("Window-1")
Set viewPt = win.Viewpoint
Dim viewPtName as String                                             1255
Dim dir as Vect3d
Dim orie as Orientation
Dim posit as Vect3d
Dim foundIt as Boolean
foundIt = FALSE                                                      1260

Open ".\scripts\viewpts.txt" for Input as #1
While Not foundIt
Input #1,viewPtName,dir.x,dir.y,dir.z,
        orie.x,orie.y,orie.z,orie.w,posit.x,posit.y,posit.z          1265
Message viewPt.name
Message viewPtName
If viewptname = viewpt.name Then
foundIt = TRUE
viewPt.name = viewPtName                                             1270
viewPt.direction = dir
viewPt.Setorientation orie
viewPt.position = posit
End If
Wend                                                                 1275
Close #1
Case "Z"
Dim wind as Window
Set wind = GetWindow("Window-1")
wind.ZoomAll                                                         1280

Case "P"
state = "dragging"
Dim windo as Window
Set windo = GetWindow("Window-1")                                    1285
```

```
windo.SetUserButtonBitmap "ChangeMode", ".\images\view.bmp"
SetCursor "GRAB"

'disable motionlinks
Dim link As MotionLink                                          1290
Dim iter As Iterator
Set link = GetFirstMotionLink(iter)
While link Is Not NOTHING
link.enabled = FALSE
Set link = GetNextMotionLink(iter)                              1295
Wend

'turn off all bounding boxes
Dim tempdev as Device
Dim iter2 as Iterator                                           1300
Set tempDev = GetFirstDevice(iter2)
While tempDev Is Not NOTHING
tempDev.boundingbox = FALSE
Set tempDev = GetNextDevice(iter2)
Wend                                                            1305

Case "F"
Dim spf As Single
Dim fps As Single
Dim intNum As Integer                                           1310
Dim decNum As Integer
spf = FrameDuration
fps = 1/FrameDuration
intNum = Cint(fps)
decNum = Cint(10*(fps - intNum))                                1315
If decNum < 0 Then
intNum = intNum - 1
decNum = decNum + 10
End If
MsgBox str$(intNum) + "." + str$(decNum) +                      1320
        "␣␣␣frames/sec", 0, "FPS␣Info"
WinActivate windowTitle

Case "L"
                                                                1325
' Get all the objects on the dem into the list
Set objcts = objgroup.Children

'delete the contaminant objects from that list
Dim delgroup as Vbase                                           1330
set delgroup = objcts.GetFirstObject()
objcts.removeFromList delgroup

'Count the number of remaining objects and pass them
'to an array to write them to a file                            1335
```

```
objctnum = objcts.Count
dim objarr(objctnum) as device
set objarr(0) = CastToDevice(objcts.GetFirstObject())
if objctnum>1 then
for i = 1 to objctnum-1                                          1340
set objarr(i) = CastToDevice(objcts.GetnextObject())
next i
end if
Dim objarrname(objctnum) as String
if objctnum = 0 then                                             1345
message"no objects!"
else
objarrname(0) = objarr(0).name
if objctnum>1 then
for i = 1 to objctnum-1                                          1350
objarrname(i) = objarr(i).name
next i
end if
end if
Begin Dialog LBTemplate                                          1355
        500,100,184,96,"Go To Selected Object"
ListBox 8,16,60,72,objarrname,.selobj
OKButton 140,4,40,14
CancelButton 140,24,40,14
End Dialog                                                       1360
Dim selectedObject as LBTemplate
actionLB = Dialog(selectedObject,1,0)
Dim ZoomedNode as Device
Dim selectedNode as string
selectedNode = objarrname(selectedObject.selobj)                 1365
set ZoomedNode = getDevice(selectedNode$)

winActivate windowTitle
If actionLB = -1 Then
windoh.ZoomToNode ZoomedNode                                     1370
Else
If actionLB = 0 Then
exit sub
End If
End If                                                           1375

End Select

End Sub
                                                                 1380
Sub CopyObject( oldObject as Geometry )
Dim root as Root
Set root = GetRoot("Root-1")
Dim newDevice as New Device
Dim newName as String                                            1385
```

```
'x% = Len(oldObject.name)   'find length of name

Begin Dialog UserDialog3 ,,
        105,55,"Copy Selected Object"
OKButton 6,35,35,12                                          1390
CancelButton 50,35,35,12
Text 6,5,60,10,"Label the object"
TextBox 6,15,45,10,.TextBox1,0
End Dialog
                                                             1395
Dim entername As UserDialog3
result% = Dialog (entername,-1,0)

If result = 0 then
'WinActivate windowTitle                                      1400
Exit Sub
End If

If result = -1 Then
newName = entername.textbox1                                 1405
'create new object
Dim pickeddevice as device
Set pickeddevice = CastToDevice(oldObject)
newDevice.filename = pickedDevice.filename
origObject$ = pickedDevice.Devicename                        1410
Message "Old Object name :" & oldObject.name

Dim deviceScale as Vect3D
pickedObject.getScale deviceScale
                                                             1415
success = newdevice.Construct(newName)
If success Then
objgroup.AddChild newDevice
newDevice.setScale deviceScale
message "Just created " & newName                            1420
Else
MsgBox "Error creating " & newName
End If

'newDevice.optimized = TRUE      'optimize it!               1425
successful = ReadSpecs(newName,origObject)
If Not successful Then MsgBox "Error reading specs"
End If
End Sub
                                                             1430
Exitdoor.ebs

Public usedDoor As Boolean
Public windowTitle as String
                                                             1435
```

```
Sub Main()

ans% = MsgBox
        ("Would you like to save the current scene?",35,"Save Scene?")
                                                                    1440
If ans = 2 Then
'Cancel was pressed
WinActivate windowTitle
Exit Sub
ElseIf ans = 7 Then                                                 1445
'No was pressed
usedDoor = TRUE 'prevents exitsim from being called
AppClose windowTitle
Exit Sub

                                                                    1450
ElseIf ans = 6 Then
'Yes was pressed
usedDoor = TRUE
'prevents exitsim from being called

                                                                    1455
'Find existing simulation files (.up) in current directory:
Dim simFiles$()
Dim numExistingSims as Integer
FileList simFiles$,"*.*up"
numExistingSims = UBound(simFiles)                                  1460
Dim existingSims$(numExistingSims)
count%=0

'Remove .up extension for nice viewing in dialog box
While count <= numExistingSims                                      1465
message simFiles(count)
length% = Len(simFiles(count))
existingSims(count) = Left(simFiles(count),length-3)
message existingSims(count)
count = count + 1                                                   1470
Wend

'Setup dialog box

Begin Dialog SaveDialog ,,117,134,"Save the Current Scene"          1475
PushButton 8,116,44,14,"Save",.SaveButton
CancelButton 68,116,40,14
Text 8,8,104,8,"Enter a new name for this scene:",.Text1
ListBox 32,56,52,48,ExistingSims$,.ListBox1
Text 32,44,68,8,"Existing Scenes:",.Text2                           1480
TextBox 28,20,56,12,.Filename
End Dialog

'Open main dialog box and process user actions
Dim result As Integer                                               1485
```

```
Dim SaveFileDlg as SaveDialog
iterate% = 1
Dim dispError as Boolean

WinActivate windowTitle                                                    1490

SaveSim:
s% = Dialog(SaveFileDlg)
WinActivate windowTitle
                                                                           1495
If s = 0 Then
WinActivate windowTitle
Exit Sub
ElseIf s = 1 Then
theFile$ = SaveFileDlg.Filename                                            1500
Dim success As Boolean
success = verifyEntry(theFile)
If Not(success) Then
MsgBox "Please enter a scene name with 10 or
        less alphanumeric characters",16,"Invalid Scene Name"              1505
GoTo SaveSim
End If
theFile = theFile + ".up"
If Not theFile$ = ".up" Then
If FileExists(thefile) Then                                                1510
Beep
If theFile = "generic.up" Or theFile = "tys.up" Then
temp% = MsgBox ("ERROR: Cannot overwrite original file."
        + Chr(13) + Chr(10) + "Please choose another filename.", _
16,"Error Saving File")                                                    1515
WinActivate windowTitle
Exit Sub
End If
t% = MsgBox
        ("Filename already exists. Overwrite?", 52,"Overwrite?")           1520
WinActivate windowTitle
If t = 6 Then
Result = SimulationSave(theFile)
End If
Else                                                                       1525
result = SimulationSave(theFile)
End If
Else
MsgBox "Error: No filename entered"
WinActivate windowTitle                                                    1530
End If
End If
AppClose windowTitle
End If
                                                                           1535
```

```
End Sub
```

Lightup.ebs

```
Sub Main
Dim theLight As Light
Set theLight = GetLight("Light-1")
If theLight.intensity < 1.0 Then
theLight.intensity = theLight.intensity + 0.05
End If
Set theLight = GetLight("Light-2")
If theLight.intensity < 1.0 Then
theLight.intensity = theLight.intensity + 0.05
End If
End Sub
```

Lightdn.ebs

```
Sub Main
Dim theLight As Light
Set theLight = GetLight("Light-1")
If theLight.intensity > 0 Then
theLight.intensity = theLight.intensity - 0.05
End If
Set theLight = GetLight("Light-2")
If theLight.intensity > 0 Then
theLight.intensity = theLight.intensity - 0.05
End If
End Sub
```

Cgrid.ebs

```
Public a1 As Geometry
Public a2 as geometry
Public floorgrid2 as geometry
Public b1 as geometry
Public b2 as geometry
Public showsgrid as Boolean

sub task(W as window)

Set a1 = GetGeometry("chattcreek")
Set a2 = GetGeometry("focreek")
Set floorGrid2 = GetGeometry("wholecreek")

Dim basetrans as vect3d, basedimens as vect3d
Dim baseorient as Orientation
Dim startpt as vect3d, endpt as vect3d
Dim horst as vect3d, horend as vect3d
```

```
Dim startpty as vect3d, endpty as vect3d
Dim griddepth as integer
Dim startptz as vect3d, endptz as vect3d
Dim gridwidth as integer
Dim verst as vect3d, verend as vect3d
Dim gridlength as integer
Dim baseobj as geometry
Dim zst as vect3d, zend as vect3d

if Not(showsgrid) then
Exit sub
else
W.set3Dcolor 1,1,0.5
w.set3DlineWidth 1.45
if a1.enabled = True then
set baseobj = a1
else if a2.enabled = True then
set baseobj = a2
else if floorGrid2.enabled = True then
set baseobj = floorGrid2
end if
end if
end if

baseobj.getTranslation basetrans
baseobj.getDimensions basedimens
baseobj.getRotation baseorient

startpt.x = basetrans.x - (basedimens.x/2)
startpt.y = basetrans.y + 0.5
'to be entered by the user
startpt.z = basetrans.z - (basedimens.z/2)

endpt.x = basetrans.x + (basedimens.x/2)
endpt.y = startpt.y
endpt.z = startpt.z

'generate horizontal lines for the
'grid taking the width from the user
horst = startpt
horend = endpt

'generate vertical lines for the grid
startpty = startpt
'to be given by the user
depth = 1000
endpty.x = startpty.x
endpty.y = startpty.y + depth
endpty.z = startpty.z
```

```
'generate lines along z axis for the grid
startptz = startpt
endptz.x = startptz.x
endptz.y = startptz.y
endptz.z = startpt.z+basedimens.z

' to be given by the user
deltawidth$ = 300
deltadepth$ = 300

gridwidth = basedimens.z/deltawidth
griddepth = depth/deltadepth
'generate multiple layers of horizontal lines
for i = 1 to griddepth
horst = startpt
horend = endpt
for j = 1 to gridwidth+1
W.Draw3Dline horst, horend
horst.z = horst.z + deltawidth
horend.z = horst.z
next j
startpt.y = startpt.y + deltadepth
endpt.y = startpt.y
next i
verst = startpty
verend = endpty
' to be given by the user
deltalength$ = 300
gridlength = basedimens.x/deltalength
'generate multiple layers of vertical lines
for i = 1 to gridwidth
verst = startpty
verend = endpty
for j = 1 to gridlength+1
W.Draw3Dline verst, verend
verst.x = verst.x + deltalength
verend.x = verst.x
next j
startpty.z = startpty.z + deltawidth
endpty.z = startpty.z
next i
zst = startptz
zend = endptz
'generate multiple layers of lines along z-axis
for i = 1 to gridlength
zst = startptz
zend = endptz
for j = 1 to griddepth+1
W.Draw3Dline zst, zend
zst.y = zst.y + deltadepth
```

```
zend.y = zst.y
next j
startptz.x = startptz.x + deltalength
endptz.x = startptz.x
next i
end if
end sub

Runstop.ebs

Public showcont as Boolean

Sub Main()

Dim group1 as Node,group2 as Node, group3 as Node
Set group1 = getNode("chatt_objects")
Set group2 = getNode("fo_objects")
Set group3 = getNode("whole_objects")

Dim group11 as Node, group21 as Node, group31 as Node
Set group11 = getNode("chatt_contaminants")
Set group21 = getNode("fo_contaminants")
Set group31 = getNode("whole_contaminants")

Dim win as window
Set win = GetWindow( "Window-1" )

if group1.enabled = True then
if showcont = True then
set group11.enabled = TRUE
showcont = False
win.SetUserButtonBitMap "contaminant", ".\images\stoplght.bmp"
else
set group11.enabled = FALSE
showcont = True
win.SetUserButtonBitMap "contaminant", ".\images\golght.bmp"
end if
else
if group2.enabled = True then
if showcont = True then
set group21.enabled = TRUE
showcont = False
win.SetUserButtonBitMap "contaminant", ".\images\stoplght.bmp"
else
set group21.enabled = FALSE
showcont = True
win.SetUserButtonBitMap "contaminant", ".\images\golght.bmp"
end if
else
if group3.enabled = True then
```

```
if showcont = True then
set group31.enabled = TRUE
showcont = False
win.SetUserButtonBitMap "contaminant", ".\images\stoplght.bmp"
else
set group31.enabled = FALSE
showcont = True
win.SetUserButtonBitMap "contaminant", ".\images\golght.bmp"
end if
end if
end if
end if
end sub

Import.ebs

Public pickedObject As Geometry
Public oldPickedObject As Geometry
Public state as String
Public newname as String
Public windowTitle as String
Public dem as imported
Public dem_dims as vect3d, dem_origin as vect3d, dem_trans as vect3d

Public cont_trans as vect3d
Public dimens As vect3d

Declare Function readSpecs
        (newName As String, origObject As String) As Boolean
Declare Function findDeviceName
        (origObject As String) As String
Declare Function fixHeight (newDevice As Device) As Boolean
Declare Function findNewName (origObject As String) As String
Declare Function verifyEntry(entry As String) As Boolean

Public colors() as string
Public pollutants() as string

Sub Main()

Dim root As Node
Dim givenname as string
Dim root1 as Node

Dim dem1 as imported, dem2 as imported, dem3 as imported
set dem1 = getImported("chattcreek")
set dem2 = getImported("focreek")
set dem3 = getImported("wholecreek")

if dem1.enabled = TRUE then
```

```
set root = GetNode( "chatt_objects")
set root1 = GetNode( "chatt_contaminants")
else
if dem2.enabled = TRUE then
set root = GetNode( "fo_objects" )
set root1 = GetNode( "fo_contaminants")
else
if dem3.enabled = TRUE then
set root = GetNode( "whole_objects")
set root1 = GetNode( "whole_contaminants" )
end if
end if
end if

Dim noofcolors as integer
noofcolors = 0
Open ".\scripts\colors.txt" For Input As #12
while Not EoF(12)
line Input #12, color$
noofcolors = noofcolors + 1
wend
Close
Redim colors(noofcolors)
Redim Pollutants(noofcolors)
Open ".\scripts\colors.txt" For Input As #12
while Not EoF(12)
for i = 0 to (noofcolors-1)
line Input #12, color$
colors(i) = Word$(color$,3)
pollutants(i) = Word$(color$,2)
next i
wend
Close

Dim manufacturers(3) As String

manufacturers$(0) = "DEMs"
manufacturers$(1) = "Buildings"
manufacturers$(2) = "Wells"
manufacturers$(3) = "Sensors"

Begin Dialog MainDialog ,,175,100,"Insert␣New␣Object",,,1
PushButton 116,28,52,14,"Show␣Objects",.ShowDevices
PushButton 124,72,40,14,"Cancel",.Cancel
DropListBox 4,28,100,120,manufacturers$,.manufacturer
Text 8,12,72,8,"Select␣an␣object:",.Text2
End Dialog

Dim userDialog As MainDialog
DisplayMainDialog:
```

```
action% = Dialog (userDialog,1,0)
If action = 2 Then   'if user cancels
WinActivate windowTitle
Exit Sub
End If                                                          1840

manuf$ = manufacturers(userDialog.manufacturer)

'Change to directory with correct devices
Select Case userDialog.manufacturer                             1845

Case 0
manFolder$ = "DEMs"
Case 1
manFolder$ = "Buildings"                                        1850
Case 2
manFolder$ = "Wells"
Case 3
manFolder$ = "Sensors"
End Select                                                      1855

currentDir$ = CurDir$    'gets current directory
message "initial directory: " + currentDir
deviceDir$ = currentDir + "\models\" + manFolder
message "new directory: " + deviceDir                           1860
ChDir (deviceDir$)

Dim modelName() As String                    'device name

                                                                1865
If userDialog.manufacturer <= 3 Then
'Create string of 3DS files in device directory


Dim deviceNames$()                                              1870
FileList deviceNames$,"*.3ds"

If (ArrayDims(deviceNames$)<>0) Then
numDev% = 0
countDev% = UBound(deviceNames) + 1                             1875
ChDir (currentDir$)
message "final directory: " + currentDir
message "there are " & countDev & " possible objects"
Dim deviceList$(countDev-1)      'filename without .3ds
ReDim modelName$(countDev-1)     'vendor name                   1880
numDev = 0
While numDev < countDev
deviceList(numDev) = FileParse$(deviceNames(numDev),4)
modelName(numDev) = findDeviceName
        (FileParse$(deviceNames(numDev),4))                     1885
```

```
numDev = numDev + 1
Wend
Else
MsgBox "Error: No devices exist for
          " & manuf,16,"No devices available"
ChDir (currentDir$)
Goto DisplayMainDialog
End If
End If

'Display dialog box with requested devices
manImage$ = ".\images\" + manFolder + ".bmp"


Begin Dialog DeviceDialog ,,230,130,"Insert New Object"
Text 4,8,64,8,"Name",.Text1
DropListBox 40,7,45,68,modelName$,.thedevice

Text 4,32,85,10,"Label"
TextBox 40,31,45,11,.TextBox1,0

Text 4,56,85,10,"Easting "
TextBox 40,55,45,11,.TextBox2,0

Text 4,80,85,10,"Northing "
TextBox 40,79,45,11,.TextBox3,0

Text 116,32,85,10,"Height"
TextBox 152,31,45,11,.TextBox4,0

Text 116,56,85,10,"Width "
TextBox 152,55,45,11,.TextBox5,0

Text 116,80,85,10,"Depth "
TextBox 152,79,45,11,.TextBox6,0

PushButton 90,105,45,11,"Add",.AddButton
PushButton 152,105,45,11,"Cancel",.Cancel

End Dialog

Dim devDialog as DeviceDialog
action = Dialog (devDialog,1,0)
WinActivate windowTitle
If action = 2 Then Exit Sub
Dim origObject As String

enteredName$ = devDialog.TextBox1
        'used for findnewname() for the well
```

```
'Determine possible object selected
'& get the next available name
theObject = deviceList(devDialog.thedevice)
theFilename = theObject + ".3ds"
message "the filename is " & theFilename                    1940
origObject=theObject


'UTM Coordinates
'Get the easting and northing co-ordinates                  1945
Easting$ = devDialog.TextBox2
Northing$ = devDialog.TextBox3

strchx$= devDialog.TextBox5
strchy$= devDialog.TextBox4                                 1950
strchz$ = devDialog.TextBox6

'Get the currently enabled DEM
'and set it's co-ordinates to d_x, d_z
'these d_x and d_z are used for verification                1955
'Easting
d_x1$ = left(dem_eastborder$,1)
d_x2$ = right(left(dem_eastborder$,2),1)
d_x3$ = right(left(dem_eastborder$,3),1)
d2_x4$ = right(left(dem_eastborder2$,3),1)                  1960

'Northing
d_z1$ = left(dem_northborder$,1)
d_z2$ = right(left(dem_northborder$,2),1)
d_z3$ = right(left(dem_northborder$,3),1)                   1965
d_z4$ = right(left(dem_northborder$,4),1)
d2_z5$ = right(left(dem_northborder2$,4),1)

'verify the entered easting and northing components
Dim proceed_east as Boolean                                 1970
a1 = left(Easting,1)
a2 = right(left(Easting,2),1)
a3 = right(left(Easting,3),1)

b1 = left(Northing,1)                                       1975
b2 = right(left(Northing,2),1)
b3 = right(left(Northing,3),1)
b4 = right(left(Northing,4),1)


                                                            1980
'Easting
if a1 = d_x1 then
if a2 = d_x2 then
if (a3 >= d_x3) and (a3 <= d2_x4$) then
proceed_east = TRUE                                         1985
```

```
else
proceed_east = FALSE
MsgBox "Check the easting coordinate"
Exit  Sub
end if
else
proceed_east = FALSE
MsgBox "Check the easting coordinate"
Exit Sub
end if
else
proceed_east = FALSE
MsgBox "Check the easting coordinate"
Exit Sub
end if

Dim proceed as Boolean
'Northing
if proceed_east = TRUE then
if b1 = d_z1 then
if b2 = d_z2 then
if b3 = d_z3 then
if (b4 >= d_z4) and (b4 <= d2_z5) then
proceed = TRUE
else
proceed = FALSE
MsgBox "Check the Northing coordinate"
Exit  Sub
end if
else
proceed = FALSE
MsgBox "Check the Northing coordinate"
Exit Sub
end if
else
proceed = FALSE
MsgBox "Check the Northing coordinate"
Exit Sub
end if
else
proceed = FALSE
MsgBox "Check the Northing coordinate"
Exit Sub
end if
end if


'Compute the translation from the Easting
'and Northing coordinates above
Dim dem_orient as orientation
```

```
dem.getDimensions dem_dims
dem.getOrigin dem_origin
dem.getTranslation dem_trans
dem.getRotation dem_orient

Dim newDevice as New Device
Dim newDevice_trans as vect3d
Dim newDevice_strch as vect3d

newDevice_strch.x = strchx
newDevice_strch.y = strchy
newDevice_strch.z = strchz

'Dim dimens As vect3d

east_distance$ = abs(easting - dem_eastborder)
north_distance$ = abs(northing - dem_northborder)
newDevice_trans.x = (dem_origin.x)-(dem_dims.x/2)
        +east_distance$
newDevice_trans.y = (dem_trans.y)-50
newDevice_trans.z = (dem_origin.z)-(dem_dims.z/2)
        +north_distance$
Message"Dem X-coordinate:"& newDevice_trans.x
Message"Dem Y-coordinate:"& newDevice_trans.z

'End of UTM Coordinates

'Add Contaminants
Dim newName as String
newName = enteredname     'findNewName(origObject)

If userDialog.manufacturer = 2 then
addcont$ = TRUE
else
addcont$ = FALSE
End If

Dim modelContaminant as Sphere
Dim cont() as Sphere
Dim noofContaminants as Integer
Dim contaminants() as String
Dim concentrations() as String, contColor() as String
noofcontaminants = 0
set modelContaminant = getSphere("Sphere-1")

If addcont$ = TRUE then
Open ".\scripts\wells.txt" For Input As #13
While not EoF(13)
line Input #13, firstLine$
firstWord$ = Word$(firstLine$,1)
```

```
cmpConts$ = newname & "contaminants"
cmpConcs$ = newName & "concentrations"
if firstWord$ = cmpConts$ then
noofcontaminants = wordCount(firstLine$)
Redim contaminants(noofcontaminants-1)
for i = 2 to noofcontaminants
contaminants(i-2) = word$(firstLine$,i)
next i
else
if firstWord$ = cmpConcs$ then
noofcontaminants = wordcount(firstLine$)
Redim concentrations(noofcontaminants-1)
for i = 2 to noofcontaminants
concentrations(i-2) = word$(firstLine$,i)
next i
end if
end if
Wend
Close

if noofcontaminants <> 0 then
Message "NoOfContaminants: " & noofcontaminants
Redim contColor(noofcontaminants-1)
Redim cont(noofcontaminants-1)
for i = 0 to noofcontaminants-2
for j = 0 to 129
'Message "Pollutants(j) : " & j & ":" & pollutants(j)
cmpStr$ = StrComp(contaminants(i),pollutants(j),1)
if cmpStr$ = 0 then
contColor(i) = colors(j)
end if
next j
next i
end if
else
Message "Added object is not a well."

End If


Dim firstTrans as vect3D, nextTrans as vect3D

'End Of Add Contaminants

'Create new object
newDevice.fileName = theFilename

success = newDevice.Construct(newName)
If success Then
root.addchild newDevice
```

```
newDevice.GetDimensions dimens

'Change the device dimensions to
'those specified in the dialog box
newdevice.setRotation dem_orient                                    2140
newdevice.setTranslation newdevice_trans
newdevice.setScale newdevice_strch
set newDevice.latitude = easting
set newdevice.Longitude =  northing
message "just␣created␣" & newname                                   2145

firstTrans.x = newDevice_trans.x
firstTrans.y = newDevice_trans.y
firstTrans.z = newDevice_trans.z
                                                                    2150
'Set Contaminant Properties
firstTrans.y = firstTrans.y +
        ((newdevice_strch.y)*(dimens.y/2)) + 50
If addcont = TRUE Then
if noofcontaminants <> 0  then                                      2155
for i = 0 to noofcontaminants-2
set cont(i) = DuplicateSphere(modelContaminant)
if cont(i) is not Nothing then
root1.addChild cont(i)
set cont(i).name = contaminants(i) & "_" & newname & "_cont"        2160
set cont(i).initialradius = concentrations(i)
end if
next i
cont(0).Material = contColor(0)
cont(0).setTranslation firstTrans                                   2165
for i = 1 to noofcontaminants-2
cont(i-1).getTranslation firstTrans
cont(i).getTranslation nextTrans
nextTrans.x = firstTrans.x
nextTrans.y = firstTrans.y + cont(i).initialradius                  2170
nextTrans.z = firstTrans.z
cont(i).setTranslation nextTrans
cont(i).Material = contColor(i)
next i
end if                                                              2175
else
Message "Contaminant␣construction␣not␣successful"
addedcont = FALSE
End If
'End of Set Contaminant Properties                                  2180

'adjusts object's translation
successful = fixHeight(newDevice)
If successful Then message
"fixed␣height!" successful = readSpecs(newName, origObject)         2185
```

```
If not successful Then MsgBox
"Error reading specs from devices.txt"
'Make bounding box & change to drag mode
Set oldPickedObject = pickedObject                                    2190
If oldPickedObject Is Not NOTHING Then
oldPickedObject.BoundingBox = FALSE
End If

Set pickedObject = GetGeometry(newName)                               2195
pickedObject.boundingbox = TRUE
Message"Picked Object name:" & pickedObject.name
state = "dragging"
SetCursor "GRAB"
'disable motionlinks                                                  2200
Dim iter As Iterator
Dim link as MotionLink
Set link = GetFirstMotionLink(iter)
While link Is Not NOTHING
link.enabled = FALSE                                                  2205
Set link = GetNextMotionLink(iter)
Wend
Dim win as Window
Set win = GetWindow("Window-1")
win.SetUserButtonBitmap "ChangeMode", ".\images\view.bmp"             2210
Else
errorString$ = "Error Creating " & newname
MsgBox errorString, 48, "Import Failed"
End If
                                                                      2215
End Sub

'___
' Function Definitions
'___                                                                  2220
Public Function readSpecs(newName As String,
        origObject As String) As Boolean

Dim obj As Device
Dim objName As String                                                 2225
Dim mfdName As String
Dim devName As String
Dim processTime As Single
Dim offset as Single
Dim foundIt As Boolean                                                2230

Dim commentLines() As String
lineNumber% = 0
numComments% = 0
                                                                      2235
```

```
Open ".\scripts\devices.txt" For Input as #1
foundIt = FALSE

While Not EOF(1)
lineNumber = lineNumber + 1                                    2240
Line Input #1, lineInput$
message lineInput
firstChar$ = Left$(lineInput,1)
commentChar$ = "#"
If StrComp(firstChar,commentChar) = 0 Then                     2245
ReDim Preserve commentLines(numComments)
commentLines(numComments) = lineNumber
numComments = numComments + 1
message "comment line!!!!"
End If                                                         2250
Wend
Close #1

Dim iter As Integer
                                                               2255
Open ".\scripts\devices.txt" For Input as #1
lineNumber = 1
While (Not foundIt) And (objName <> "END")
iter = 0
'message "line number: " & lineNumber                          2260
While iter < numComments
commentLineNumber = commentLines(iter)
'message "comment line number: " & commentLineNumber
If lineNumber = commentLineNumber Then
Line Input #1, trashComments$                                  2265
'message "trashed line: " & trashComments
lineNumber = lineNumber + 1
End If
iter = iter + 1
Wend                                                           2270
Input #1,objName,mfdName,devName',processTime,offset
lineNumber = lineNumber + 1
Message "origobject is " + origObject
Message "objname is " + objName
strCompare% = StrComp(objName,origObject,1)                    2275
If strCompare = 0 Then
Set obj = GetDevice(newname)
obj.manufacturer = mfdName
obj.devicename = devName
Message "Name: " + objName                                     2280
Message "Manufacturer: " & mfdName
Message "Device Name: " & devName
foundit = TRUE
Else
foundIt = FALSE                                                2285
```

```
End If
Wend

If foundIt Then readSpecs = TRUE
If Not foundIt Then readSpecs = FALSE                                    2290

Close #1

End Function

                                                                         2295
Public Function findDeviceName
        (origObject As String) As String

Dim obj As Device
Dim objName As String                                                    2300
Dim mfdName As String
Dim devName As String
Dim processTime As Single
Dim offset as Single
Dim deviceName As String                                                 2305
Dim foundIt As Boolean
foundIt = FALSE

Dim commentLines() As String
lineNumber% = 0                                                          2310
numComments% = 0
Open ".\scripts\devices.txt" For Input as #1

While Not EOF(1)
lineNumber = lineNumber + 1                                              2315
Line Input #1, lineInput$
message lineInput
firstChar$ = Left$(lineInput,1)
commentChar$ = "#"
If StrComp(firstChar,commentChar) = 0 Then                               2320
ReDim Preserve commentLines(numComments)
commentLines(numComments) = lineNumber
numComments = numComments + 1
message "comment line!!!!"
End If                                                                   2325
Wend
Close #1

Dim iter As Integer
                                                                         2330
Open ".\scripts\devices.txt" For Input as #1
lineNumber = 1
While (Not foundIt) And (objName <> "END")
iter = 0
message "line number: " & lineNumber                                     2335
```

```
While iter < numComments
commentLineNumber = commentLines(iter)
message "comment line number: " & commentLineNumber
If lineNumber = commentLineNumber Then
Line Input #1, trashComments$                                          2340
message "trashed line: " & trashComments
lineNumber = lineNumber + 1
End If
iter = iter + 1
Wend                                                                   2345
Input #1,objName,mfdName,devName',processTime,offset
lineNumber = lineNumber + 1
Message "origObject is " + origObject
Message "objName is " + objName
strCompare% = StrComp(objName,origObject,1)                            2350
If strCompare = 0 Then
deviceName = devName
foundIt = TRUE
Else
foundIt = FALSE                                                        2355
End If
Wend

If foundIt Then findDeviceName = deviceName
If Not foundIt Then findDeviceName = origObject                        2360

Close #1

End Function
                                                                       2365


Public Function fixHeight (newDevice As Device) As Boolean
FixHeight = TRUE
End Function                                                           2370

Toglemap.ebs

Public wireframe as Boolean
                                                                       2375
Sub Main

Dim env as Universe
set env = getUniverse("The Universe")
                                                                       2380
if wireframe = False then
env.Wireframe = True
wireframe = True
else if wireframe = True then
env.Wireframe = False                                                  2385
```

```
env.Textured = False
env.Textured = True
wireframe = False
end if
end if
End Sub

Toglegrd.ebs

Public showsgrid as Boolean

sub Main()

if showsgrid = True then
showsgrid = False
else
showsgrid = True
end if
end sub

Selmap.ebs

Public a1 As Geometry
Public a2 as geometry
Public floorgrid2 as geometry
Public b1 as geometry
Public b2 as geometry
Public showcont as Boolean

Sub Main()

Set a1 = GetGeometry("chattcreek")
Set a2 = GetGeometry("focreek")
Set floorGrid2 = GetGeometry("wholecreek")

Dim group1 as Node,group2 as Node, group3 as Node
Set group1 = getNode("chatt_objects")
Set group2 = getNode("fo_objects")
Set group3 = getNode("whole_objects")

if a1.enabled = TRUE then
Set a1.enabled = False
Set a2.enabled = True
Set floorGrid2.enabled = False

Set group1.enabled = False
Set group2.enabled = True
Set group3.enabled = False
else
if a2.enabled = TRUE then
```

```
Set a1.enabled = False
Set a2.enabled = False
Set floorGrid2.enabled = True

Set group1.enabled = False                                          2440
Set group2.enabled = False
Set group3.enabled = True
else
if floorGrid2.enabled = TRUE then
Set a1.enabled = True                                               2445
Set a2.enabled = False
Set floorGrid2.enabled = False

Set group1.enabled = True
Set group2.enabled = False                                          2450
Set group3.enabled = False
end if
end if
end if
                                                                    2455
End Sub
```

Vita

Sirisha Vadlamudi

Sirisha Vadlamudi was born in Nagarjuna Sagar, AP, India. She attended the Velagapudi Ramakrishna Siddhartha Engineering College from 1995 to 1999, and graduated with a Bachelor of Engineering degree in Electrical and Electronics Engineering in 1999. She came to the University of Tennessee in the Fall of 2001 for graduate studies in Electrical Engineering.