THE UNIVERSITY OF HULL

Immersive Visualization and Interaction of Multidimensional Archaeological Data

being a Thesis submitted for the Degree of Master of Science

in the University of Hull

by

Julien PANSIOT

September 2004

Abstract

The current standard method for gathering and representation of archaeological information consists of two-dimensional layer managers. This thesis presents an archaeological Geographical Information System (GIS) based on an immersive virtual environment. The aim is to allow the integration of many kinds of data, from the standard maps to more challenging numerical databases. Secondly, it should hasten the archaeological interpretations of the data by representing it within an environment that maximizes the usage of the human vision bandwidth and provides natural interaction.

Keywords: *Archaeology, Virtual Environment, Immersive.*

Acknowledgments

A one-year project involves many people from near and far, and this is quite a delicate task not to forget anybody. Hereafter are most of my acknowledgments, without any particular order.

I would like to thank all the testers that have performed the user evaluation of my application.

I would like to thank my chairperson Professor Roger Phillips, particularly for the agreement with the IIE which allowed me to pursue this master at the University of Hull.

I would like to thank Emma Smith-Howe, James Ward and Mike Bielby who were always here to help me with hardware and software issues.

I would like to thank Peter Halkon for the data sets and time spent describing important archaeological issues.

I would obviously like to thank my supervisors Warren Viant and Paul Chapman for the day to day support and profuse ideas.

I would like to thank Thibault and Bertrand for their ideas, comments, beta-testing and proof reading.

I really would like to thank my family and all my friends for their constant support from overseas and England.

Overall, I would like to acknowledge Stéphanie for her understanding and patience.

Contents

List of Figures

List of Tables

“Toute connaissance que n’a pas précédé une sensation m’est inutile.”

(All knowledge that is not preceded by sensation is useless to me.)

André Gide

Chapter 1

Introduction

1.1 Why virtual archaeology?

Archaeology is a science mainly based on the collection and the interpretation of information. The modern archaeologist can be overwhelmed with the very large quantity of data generated from current scanning and measurement technologies, as detailed for example in (Ashley López 2003). The vast quantity of data types, formats and representations compounds the problem.

The interpretation stage from which the archaeologist will draw his conclusions consists of gathering the whole information at once into a robust and consistent scenario. As "computer reconstruction allows the presentation of complex information in a visual way that enables it to be used to test and refine the image or model that has been created" (Forte & Guidazzoli 2004), a good visualization of the archaeological information may then improve the quality of the interpretation and as a matter of fact the pertinence of the conclusions.

Furthermore, "the very activity of excavation inevitably involves some degree of destruction of the information that is buried" (ibid). As the visualization relies on non-destructive methods, the virtual archaeologist may evaluate several hypothesis without altering the historical site, simply by simulation.

1.2 Aims of the project

The main aim of this project is to allow the representation of a large collection of archaeological data in a three-dimensional immersive virtual environment and to enable an archaeologist to easily interact with the information. This should hasten the conception of relations between finds, rather than using fastidious maps and pictures comparisons. For this purpose, three main objectives need to be reached.

First of all, the application must integrate as many archaeological data sets as possible. It should at least be able to use a three-dimensional terrain, some pictures and short texts, three-dimensional items and databases. This information must be easily represented, queried and displayed.

Secondly, the application should offer more than a simple three-dimensional rendering on a desktop screen. The visualization aims to immerse the user into a three-dimensional scene. It will rely on a large work wall and a motion capture system in order to reach such an immersion state.

Finally, as this kind of environment is not really common, developing some intuitive ways of navigation is required to provide the maximal efficiency to the archaeologists. This will include the most common interaction devices such as a mouse, a keyboard or a gamepad to the state of art motion capture systems. The navigation system will consist of a panel of metaphors based on these devices that allow the user to move, query information or select an object. As many options will be available, the ease of use, the consistency and the speed of the navigation will be decisive. At the same time, the application should still remain usable with a desktop PC.

Moreover, the modularity of the application is crucial. It must be quick and easy to include new pieces of information and new kinds of data even by a non-expert. Input devices could also be changed without reconfiguring the entire application, in order to extend the fields of utilization or to adapt it to the working environment.

1.3 Thesis structure

As this project involves many concepts from various fields of research, this thesis will present a solid background in order to introduce the context. This will cover the current archaeological methodology, the immersive virtual environments that may be applied to the science, two examples of current archaeological virtual environments, as well as the more technical concepts of Geographic Information Systems and of scenegraphs.

As the project strongly relies on the OpenSceneGraph toolkit, but no complete documentation for this library is available yet, its main concepts will be detailed in a chapter. Its architecture and functionalities but also secondary issues like the available add-ons will be listed.

The architecture of the system on which the application relies will be developed. This will cover the main input and output devices that were applied into the Hull Immersive Visualization Environment (referred to as HIVE). The work wall, the motion capture systems and the wireless gamepad will be introduced.

Logically following, the software architecture will then be detailed. This encompasses the global approach to integrated data in the environment, the modularity of the software design, the database design relying on the CIDOC model, and the database management using a Graphical User Interface (GUI).

The visualization tools that have been implemented for most of the information in the environment will then be detailed. Starting with the representation of the terrain, this chapter will then detail the enhancement of the crop marks representation, the integration of numerical databases, of short texts and picture using dynamic widgets and billboards. Finally, a summary of this case study will be addressed.

Associated to this visualization, the implemented navigation metaphors will then be described. First will be detailed the two main navigation metaphors (standard and 'map table' paradigm). Then the usage of a finger-bend tracked glove will be

demonstrated. Finally, some enhancing tools such as a status bar, a mini-map and a compass using the global density of find will be reviewed.

Then an evaluation of this system will be presented. The evaluation of the data representation consists of the English Heritage and Archaeological Data Service visits subjective feedbacks. The scientific user evaluation that has been carried out to validate the navigation metaphors and devices will then be described.

Finally, a conclusion about the state of this project will be given before detailing some issues for future work.

Chapter 2

Background

2.1 Introduction

The aim is to design an immersive virtual environment dedicated to multidimensional archaeological data. This issue involves many fields of study that will be described here. First of all, the current archaeological methodology, how data is collected and used, is described. This discussion highlights the valuable points and weaknesses within the current procedure. Given its crucial importance, the crop marks issue will be focussed on. Secondly, a container for our data and methods is required. The main aspects of a virtual environment for scientific purposes will be provided by an honest and large scaled vision of the current state of art. Then this will be refined in two steps: a study of the concept of Geographic Information System (abbr. GIS), which will be the basis of this application, and the specialization of such a GIS into an archaeological toolkit. The 'scenegraph' concept, which will be the back bone of our implementation, will be presented. And finally, an introduction to the CIDOC 'Conceptual Reference Model' that aims to store cultural information will be provided. All theses elements are demonstrated in Figure 2.1.

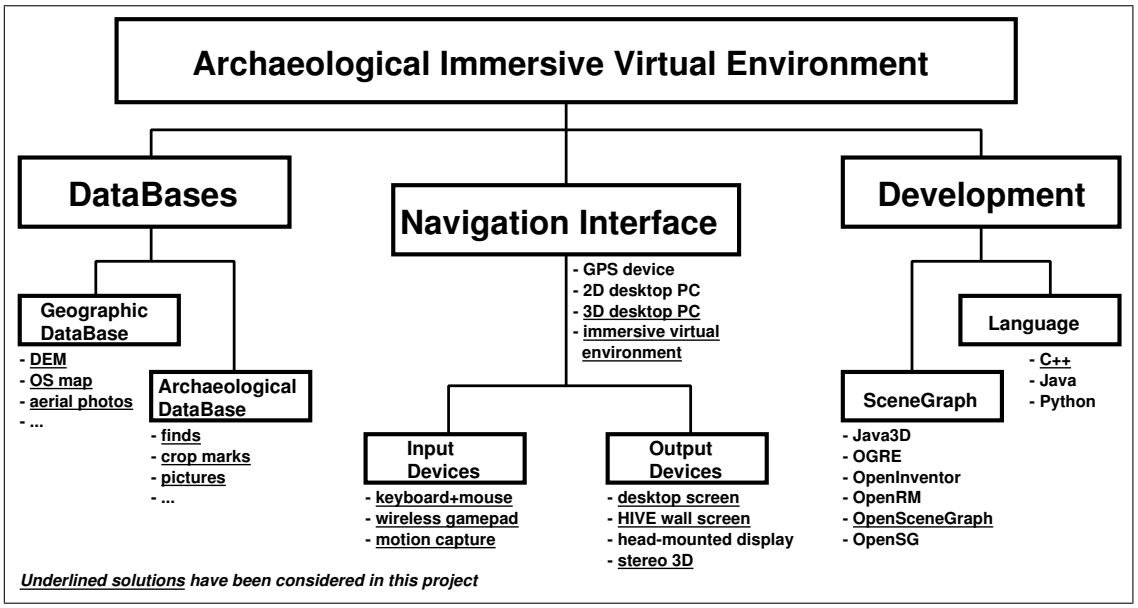


Figure 2.1: The main elements of an archaeological virtual environment

2.2 Current archaeological methodology

Peter Halkon (Lecturer in Archaeology - The University of Hull) has facilitated the research in introducing important archaeological concepts pertinent to this project.

2.2.1 Data collection and gathering

The archaeological data can be collected either on the historical fields or by exchange or purchase. In the former case the data consisting mainly of pictures, finds or maps of points, is raw and then needs to be treated (sorted, simplified, filtered, scanned, ...). In the latter case the data is already numerical or paper-printed and theoretically ready to use. At this stage, the amount of material may be quite large and consists of various kinds of elements. These can be numerical databases, photographs, maps, texts, etc.

A typical way to gather and visualise data is to use the Arcview software and its successors. The kernel of this software is basically a layer manager, allowing the loading and display of images. The source documents, generally maps, lie in

Ordnance Survey (OS), soil, finds, crop marks, or elevation models as well.

2.2.2 Usage and representation

Archaeologists try to draw some conclusions about the data they collected. For this purpose, they need to compare and link the whole information into a consistent model. Up to about twenty years ago, they had to use fastidious paper map comparisons. Nowadays, these tasks are hastened by the integration of the computer science in this process. The Arcview software is widely used for this purpose.

This piece of software consists of a two-dimensional map and a control panel as shown on Figure 2.2. The panel allows the user to switch on and off the layers and to change colours and symbols which are displayed on the map. The map can contain layers and also a Digital Elevation Model (DEM), that is simply represented in two dimensions by the way of grey scale (or any other colour-based scale). Two-dimensional areas can be represented at various scales and filled in with colours. Stand-alone objects are displayed as small symbols like crosses, squares or circles.

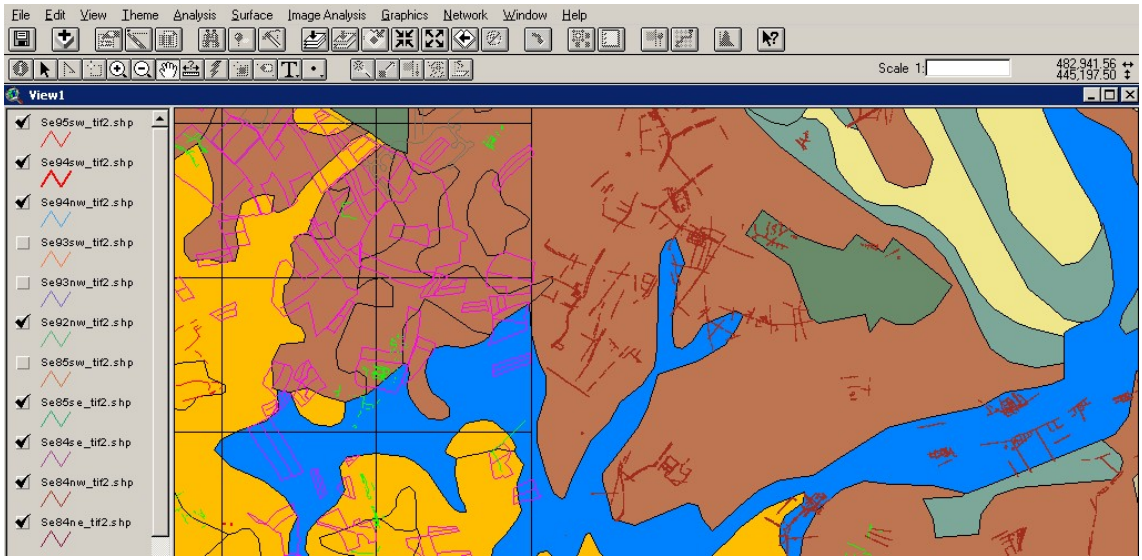


Figure 2.2: The Arcview software: soil map background and crop marks layer

The interaction with the map is achieved by the way of the keyboard and the

mouse, which enables the map to be grabbed and moved. The public display (for specialists or larger scale) of the data is mostly based on images extracted from this software associated with photographs and sketches. However, photographs and maps remain independent.

2.2.3 First limitations

Even though 3D models can be inserted into Arcview, this software is not designed to render the whole possibilities offered by a three-dimensional map. Lots of material cannot be integrated like stand-alone images (from finds for example), small animations, video or 3D objects (for example three-dimensional scans of tiles or building reconstructions).

It may be disappointing and quite frustrating for archaeologists to spend time and money in order to collect a large amount of data, but finally not be able to sort and represent it at a level similar or even better than its original quality. The main reasons for using this piece of software are the ease of use and the possibility to install it on almost any machine. It remains then quite important to take into consideration the simplicity of an archaeological application if it aims to be widely used.

2.3 Crop marks

2.3.1 What are crop marks?

Crop marks represent one of the most important aspect of archaeological research. These patterns on the ground "show only a two-dimensional view of buried features" (Stoertz 1997), left for example by ancient roads, walls, etc. They are visible only from their effects on the vegetation growth and are generally discovered using aerial photography (for example Figure 2.3).

Figure 2.3: An example of crop mark in an aerial picture (Photo: P. Halkon)

The interpretation of these single marks is simply impossible. Other knowledge and pieces of information about the studied place should been involved in this process. Some extrapolations are also necessary. However, a basic morphological classification can be performed.

2.3.2 Morphological classification

(Stoertz 1997) proposed such a classification. She pointed out four main classes.

The 'curvilinear' shapes enclose (totally or partially) surfaces which are rounded, whereas 'rectilinear' forms enclose (again, totally or partially) areas which are more rectangular. Both shapes can be either 'regular' (mainly circles and squares) or 'elongated' in the case of a breadth to length ratio of more than 1/1.25. Curvilinear shapes can also be classified as 'irregular', if no dominant form seems to rule. Most of the curvilinear shapes are related to barrows, "ritual enclosures and defended sites" (ibid). The rectilinear forms seem to be linked to barrows as well, but also settlements.

The 'linear' class includes all the elements that do not seem to be part of an enclosure. They can either be 'short' or 'long'. They are generally related to trackways,

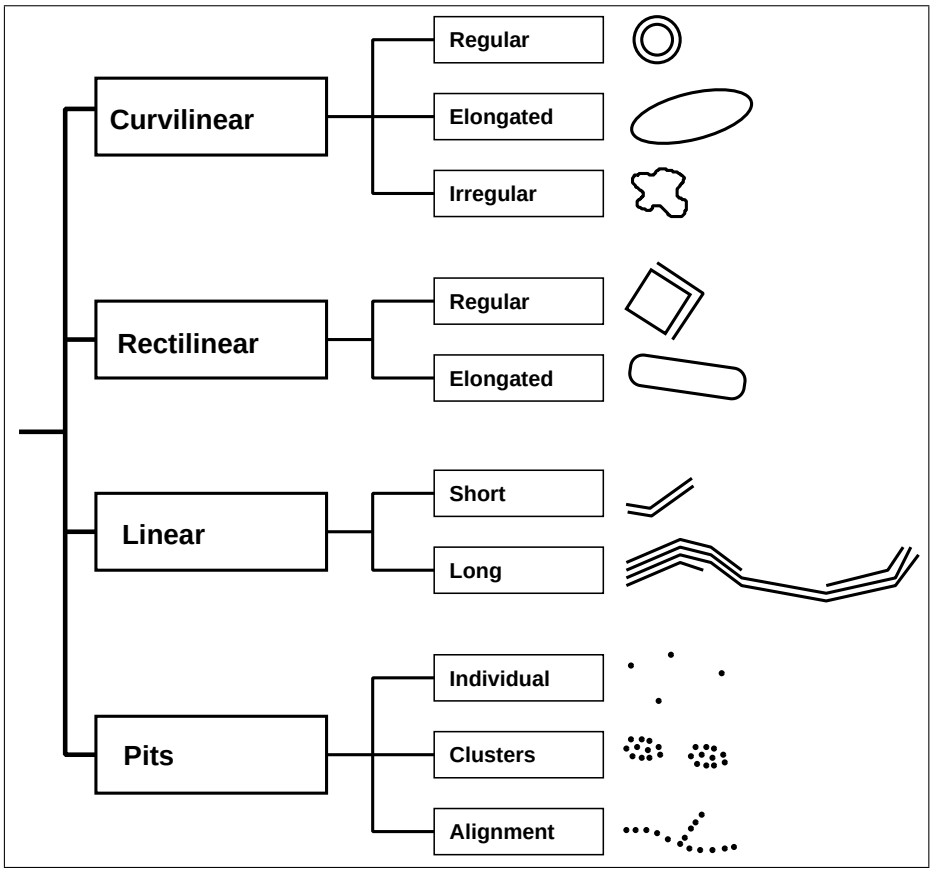


Figure 2.4: A summary of Stoertz's morphological classification of crop marks

droveways or boundaries. The 'pits' are very small crop marks similar to points. They can either be 'individual', in 'clusters' or in 'alignments'. These alignments are often considered as boundaries.

Overall, all theses forms can be 'single-ditched', 'double-ditched' or more generally 'multi-ditched', which means that the visible marks can be composed of one or several parallel lines. They are very often part of larger structures. The Figure 2.4 summarises the main aspects of this analysis.

2.3.3 Examples

Various examples of crop marks are presented within Figures 2.5, 2.6 and 2.7 in order to appreciate the different kinds that can be found. They are all proposed by

(Stoertz 1997).

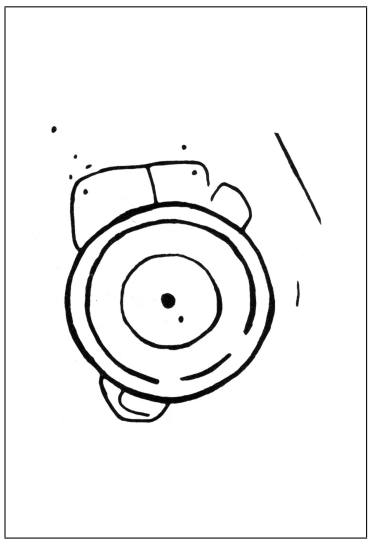


Figure 2.5: Crop marks: "regular curvilinear enclosure, 115m external diameter"

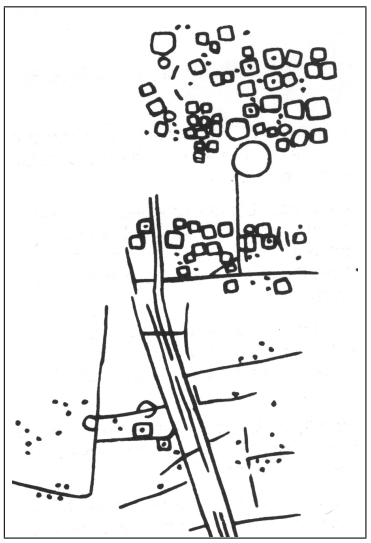


Figure 2.6: Crop marks: "square barrow cimetery" and "double-ditched trackway"

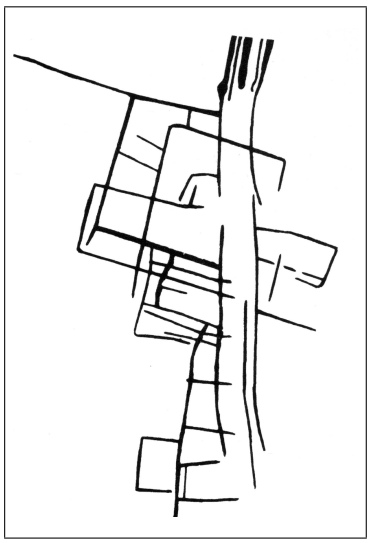


Figure 2.7: Crop marks: "Foxholes parish"

2.4 (Semi) immersive virtual environments for scientific objectives

2.4.1 General aims in science

Until recently, representing the information in two dimensions (on the paper or on a screen) has been more convenient, due to technical reasons. Considering the abilities of the human brain, trained to cope with a three-dimensional world and five senses, this is however a very poor metaphor. The aim of the virtual environment is to use a larger proportion of the human capacities by maximizing "the bandwidth to the brain by more fully engaging the human sensorium" (van Dam, Forsberg, Laidlaw, LaViola & Simpson 2002).

The great hopes carried by immersive virtual environments have spread their

applications to most of the scientific domains. In particular: anthropology, archaeology, astronomy, biology, chemistry, computer science, geology, applied mathematics, medicine, cognitive and linguistic sciences. It is really fundamental to observe that virtual environments are not generally designed for themselves, but to serve another field of research. This is an aid to the navigation in the available real or conceptual information. "The scientist visualizes the data to look for patterns, features, relationships, anomalies, and the like. Visualization should be thought of as task driven rather than data driven" (van Dam et al. 2002).

2.4.2 The immersion feeling

The capacities of a virtual environment can after be enhanced if no other signals are able to disturb the user. For example, when using a virtual environment on a simple personal computer, the field of view is largely occupied by the context like the desktop, the keyboard or some books which are not part of the virtual environment. Even though the user does not focus on these objects, they tend to break the illusion of the virtual world, which is reduced to the small window that is provided by the screen.

As a virtual environment aims to maximize the bandwidth of the human brain, this immersive effect must be taken into account. Reaching a state of immersion, which can be visual but also haptic or auditory requires a carefully designed coverage of the human field of view. A perfect virtual environment covers the whole human view angle.

The illusion can also be improved by adding to the virtual environment a more realistic perception of the third dimension. This can be achieved by the stereo 3D graphics, that simulates this third dimension by providing different images for each eyes, as discussed hereafter.

To complete this immersive feeling, the user should be able to interact with such

an environment via a device that does not seem to be intrusive or unusual in the considered virtual environment. Generally, the keyboard and the mouse break the feeling of immersion and in some cases are not usable at all (with head-mounted displays for example). A wireless gamepad is slightly better due to its usability even without looking at it and the more intuitive ways of navigation it can offer. The direct use of the human body by gesture recognition remains the most natural way of interaction.

2.4.3 Cognitive maps

While navigating through a virtual environment, the human brain naturally records some links, building a network of connections called a 'cognitive map'. The ease of use and the performance of the environment will depend on this cognitive map.

(Forte 2002) studied the example of a Roman building reconstruction. He found that this modeling was not accurate enough to provide a clear representation without distortions. Therefore, "lacking the appropriate navigational instruments, the cognitive maps (computational cartography) used for navigation and the appropriate hypermedial tools, it was almost impossible for a classical archaeologist to orientate himself in the three dimensional spaces." (Forte 2002) Moreover, this kind of virtual reconstruction can provide a partially wrong idea of the initial model and leads to rejection by the user.

The construction of a virtual environment must rely on very precise modeling. Designers should be very careful even about small details that may break the illusion and lead to "a complete cognitive de-territorialisation of the user, lacking orientation and without any informative increase, rather an informative loss" (ibid).

2.4.4 Display devices

Reaching a state of immersion for the user implies setting up the largest field of view possible. This is generally achieved with a head-mounted display or a large screen. In the case of the head-mounted display, the utilization of a tracking system, which updates the user virtual point of view according to the real one is critical to preserve the immersive sensation. In the case of a static large screen, several methods can improve such a feeling: the augmentation of the angle of view by choosing either a very large wall screen or a convex one (like the CAVE, which is a cubic display system presented within Figure 2.8) and the use of the stereo. A rear-projected screen is necessary to allow the user to walk as close as possible to the screen (and therefore increase the covered angle of view).

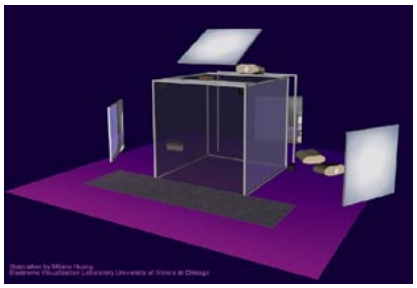


Figure 2.8: The CAVE cubic display system

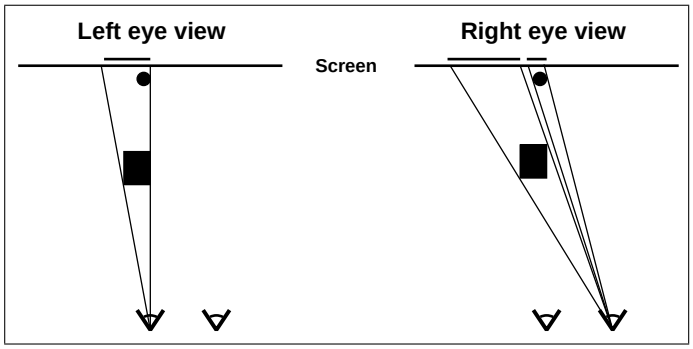


Figure 2.9: Stereo 3D: both eyes do not receive the same image of the scene

2.4.5 Stereo 3D

The stereo-3D method reproduces the natural way of the human vision: both eyes do not see exactly the same image as presented in Figure 2.9, and this allows the user to "see in three dimensions". There are three common ways of producing the stereo effects, which can be classified as active or passive.

One passive common method consists of using coloured glasses (also called

'anaglyphic'): red for the left eye and blue (or cyan) for the right one. The images for both eyes are displayed at the same time, but with a shift in the colour balance. The quality of such a display is generally not excellent as the colours are a little bit distorted[1]. Such glasses are really cheap and do not require any special display device.

A better quality of passive stereo can be achieved using cross-polarized glasses. This involves two projectors that produce polarized images (one vertical and one horizontal) and that should be perfectly aligned. The glasses will filter the images and let each eye see the image produced by the associated projector. There are no side effects between the views, but if the user tilts a bit his head, the polarized filters will shut off the image or even invert both left and right images. This method requires specific projectors.

The active stereo method consists of producing alternatively on the screen different images for each eyes. Associated shutter glasses shut off alternatively the left eye when displaying right image and the right eye when displaying the left one. The synchronization with the screen can be achieve either by a wire or using infra-red. The main weaknesses of this technology are the weight of the glasses (including batteries and infrared receptor) and the price.

There are two main ways of computing the eyes views (Bourke 2002): the classical but incorrect method, called 'toe-in', where both lines of sight converge in the middle of the screen (Figure 2.10), and the correct one called 'parallel axis asymmetric frustum perspective projection', often simply referred to as 'asymmetric frustum', in which lines of sight are parallel, as demonstrated within the Figure 2.11.

In any case, the point of view used to compute the difference between both views generally remains static. This leads to distortion effects when the user moves. The head tracked stereo aims to solve this problem.

[1]Moreover, a 'phantom' of the representation of one side (left or right) can also appear slightly on the other side.

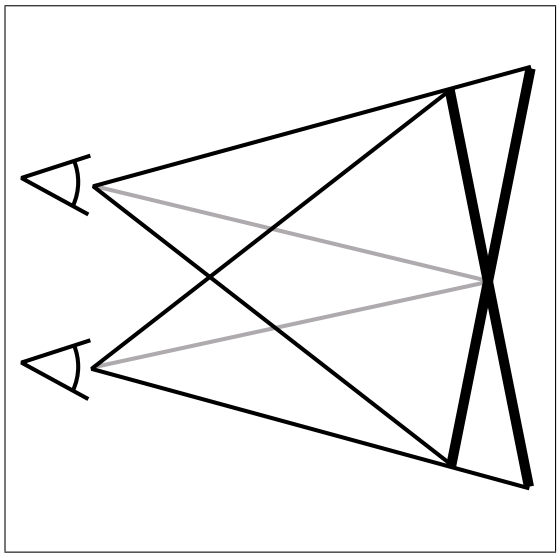


Figure 2.10: Stereo 3D: 'toe-in' (rotation-based method)

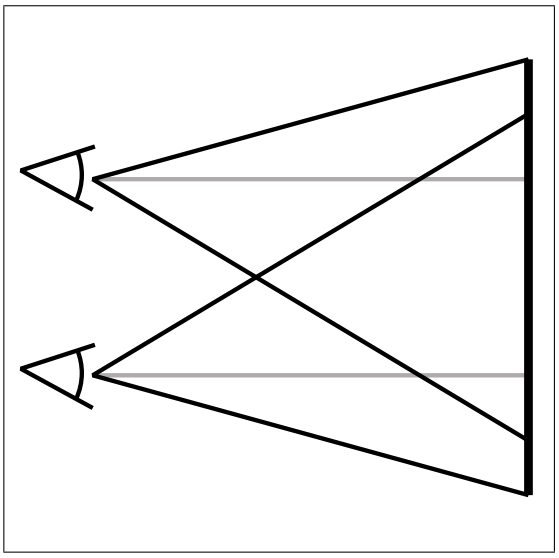


Figure 2.11: Stereo 3D: asymmetric frustum (translation-based method)

2.4.6 Head tracked stereo

The head tracked stereo (also referred to as Head-Coupled Stereoscopic (HCS)) (Arthur, Booth & Ware 1993) updates the virtual user point of view in order to display a coherent information. For example, the left face of an object will be displayed if the user seems to look at it from the left side. Such a technique allows the user to rotate around an object to watch (and even 'touch') all its faces. For this reason, this paradigm has been called 'fish-tank virtual reality'. It really increases the realism. However, as in the physical world the screen remains flat, big objects like landscapes can easily cross the border of the screen and partially disappear, which breaks the immersive feeling.

2.4.7 Tracking systems

Three motion capture (tracking) technologies are widely used: electro-mechanical, magnetic and optical. Many other ones are in development or used in very specific domains, for example those using acoustic or gravitational methods.

The electro-mechanical systems use mechanical links between the electronic sensors and the human body. These links can be as various as wires, telescopic sticks, articulated arms, optical fiber, etc, bonded on the hand or any other part of the user's body. The field of action of such systems is obviously reduced, and most of them are limited to a specific area of the user like his hand. They are generally very accurate and reliable, as there are almost no degrading factors.

The magnetic systems are based on a static magnetic field emitter and on a set of sensors (electro-magnets) bonded on the body of the character to be tracked. The sensors, wire-linked to a controller, can evaluate the intensity and direction of the main magnetic field and then compute their relative position and orientation to the emitter. This kind of system is quite sensitive to metallic objects and other magnetic fields that may degrade their accuracy[2]. They are also very sensitive to other developers that may use the same frequency in a near room.

The optical systems are generally based on a set of cameras acquiring two-dimensional images of the scene containing the character to be tracked (as any traditional camera could do). Knowing the exact position and orientation of the cameras, a processing applied on these images can extract the three-dimensional position and orientation of the character. Very often, special markers are bonded on the part of body to be tracked, in order to hasten the recognition. These markers are most of the time small white balls that can be lighted with either visible or infra-red light. The main advantage of this method is to free the user of cables. The workspace can be extended to any size simply by adding some more cameras.

The Table 2.1 sums up the main points of this comparison. Note that the accuracy and workspace are representative of the current average tracking systems, they do not aim to provide exact values for any devices.

[2]Most of the current systems use a pulsed magnetic field in order to reduce dramatically this kind of error.

System	Accuracy	Workspace	Degrading factors	Constraints
Electro-mechanical	0.1mm	30cm – 2m	-	wires, sticks
Magnetic	1mm	sphere, r=3m	large metallic pieces, magnetic fields (screens)	wires
Optical	1mm	room	intense lights, white items	bonded balls

Table 2.1: Comparison of the main motion capture systems

2.5 Geographic Information System

2.5.1 What is a GIS?

A Geographic Information System (referred to as GIS) is a complete concept or toolkit that should encompass at least a database and an interface. The database contains the geographic information. The way of storing this information will strongly influence the capacities of the interface. For example if the database only contains some standard 1:25000 scale Ordnance Survey (abbr. OS) maps in a raster (grid of pixels) format, the representation of the terrain in a three-dimensional way will probably be quite difficult, because it would rely on the recognition of the contour lines on the map. This is not an easy task, which can even be impossible to perform in many cases. The functionalities of the interface can then be various, depending on the database content. Providing information to trucks on motorways is totally different than to an alpinist during a hill challenge or to an archaeologist in his office.

The information provided by the database generally contains maps like the OS ones, that can be raster or vectorial if a better scalability and flexibility are required. Other standard large scaled two-dimensional data are aerial and satellite photographs, but also soil maps. The third dimension is generally provided separately by the way of a Digital Elevation Model (referred to as DEM). This is basically

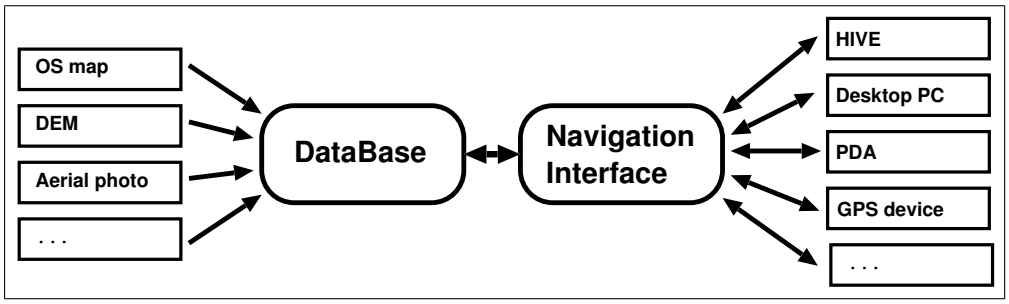


Figure 2.12: Concept of the GIS

a map of elevations. Beside these common pieces of information and depending on the purpose of the considered GIS, various kinds of data can be integrated, for example soil, vegetation or radioactivity maps, but also price of terrain, underground pipes, etc.

The interface can be developed as a simple two-dimensional layer manager for a personal computer as previously discussed with Arcview, for a personal digital assistant (abbr. PDA) or even a small Global Positioning System (abbr. GPS) device. It can also be represented within a full three-dimensional aspect and even integrated in an immersive virtual environment.

2.5.2 GIS-based virtual environment

As previously stated, an immersive virtual environment provides the best interface for the interaction with many kinds of bi and three-dimensional data. GIS are naturally part of these due to their close relationship with a real world that is three-dimensional. However, the efficiency of the integration of a GIS in a virtual environment depends on the quality of three main features pointed out by (Feliz 1999):

- "Realistic representation of the three-dimensional nature of real geographic areas."
- "Freedom of movement for the user within the selected terrain."

- "Access to standard GIS capabilities such as query, selection, spatial analysis, etc."

Building a fairly realistic representation of the geographic area is not a significant problem if a fair source of data containing accurate three-dimensional map and high-fidelity aerial photographs are available. However, many researchers like (Cavens, Lange & Schmid 2003) also pointed out the importance of the vegetation, which complicates importantly the representation. The freedom of movement is also easy to implement, even though an excess of freedom could generate some disorienting effects. This last point is still a challenge. Porting some two-dimensional ways of interaction to a three-dimensional immersive environment generates many difficulties. Actually, a totally novel way of query and selection is generally created from scratch, by the way of gesture recognition for example.

2.6 Existing archaeological virtual environments

2.6.1 Photo-realistic vs non photo-realistic

Several aims can motivate the development of an archaeological virtual environment. The main goals are scientific reconstruction, public display and educational tools. Archaeological environments designed for historians and archaeologists have followed for many years the "as realistic as possible" paradigm, which means that the navigation in the environment should look like a video. This kind of method is widely implemented for public displays or educational purposes where a complete "beautiful" representation is required. The content of such a display depends obviously on the pre-interpretation of the designers of the environment, and then may be only by hypothesis. However, a new paradigm based on a "believable" environment that does not require a photographic accuracy (Roussou & Drettakis 2003) may be the new guide line for scientific virtual environments. This concept prefers

using techniques that render a view similar to human perception rather than a flat photographic one, which means that some parts would not be represented as a photograph, but with metaphors like sketches. A more efficient evaluation of the hypothesis should be carried out, without being influenced by the realism of the scene.

Moreover, in the case of a scientific environment, the highest degree of freedom and interaction is required in order to reach the best evaluation system. The archaeologist may be able to change some hypothesis on the studied location, like the shape of some buildings or the position of a footpath for example. This is generally impossible in a realistic environment, due to the lack of pertinent information.

2.6.2 Examples of current archaeological virtual environments

Archaeological virtual environments can be classified in three main categories: the local video-realistic reconstruction, generally designed for public demonstrations, the local believable environment, for scientific purposes applied to a particular location, and the GIS-based archaeological environment, which can be more or less realistic. The first class, which is not designed for scientific purposes, will not be discussed.

(Roussou & Drettakis 2003) provided an excellent example of local believable environment applied to the "Hellenistic Asklepieion of Messene in Greece". Nowadays, only the basements of the temple remain on the physical site. As several plaster models have been designed, the aim of this project is to evaluate the authenticity of such reconstructions. The interface proposes many tools like a switch between a couple of models or semi-transparent reconstructions. A 'paper-grain' and 'pen-and-ink' metaphors have been implemented (see Figure 2.13) in order to represent the three-dimensional scene as paper-designed and then remove partially the undesirable effects of realism that could influence scientific conclusions.

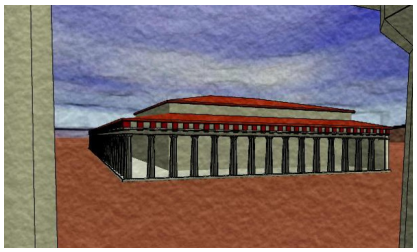


Figure 2.13: Concepts of paper-grain and pen-and-ink (published in VAST 2003)

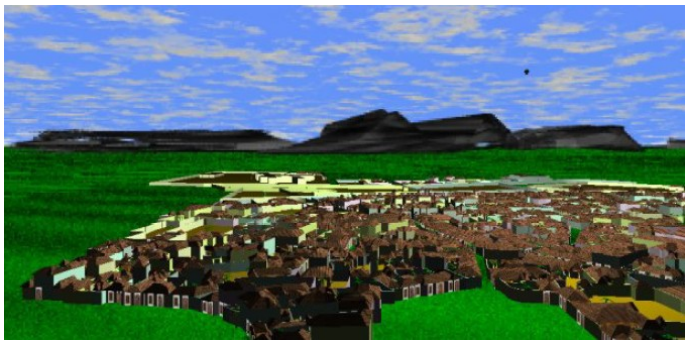


Figure 2.14: The Walled City Nicosia reconstruction (published in VAST 2003)

GIS-based archaeological environments appeared only a couple of years ago. According to (Feliz 1999): "to my knowledge, there are no projects trying to link GIS and VR in archaeological research". Although the situation has obviously evolved a bit since, archaeological GIS-based virtual environments remain not widely spread yet and a lot of research and discoveries are expected. (Dikaiakou, Efthymiou & Chrysanthou 2003) proposed a reconstruction of the Walled City of Nicosia. The main interesting point of this project is the automatic generation of the buildings. The good accuracy of the GIS allowed recognition of ancient roads and buildings. Linking this classification algorithm with a database of building elements like walls, windows, etc, they dynamically reconstructed the ancient buildings and then the entire environment in a very realistic way as shown on Figure 2.14.

2.7 SceneGraphs

2.7.1 What is a scenegraph?

The OpenGL library has been created to accelerate the rendering of three-dimensional graphics. OpenGL is polygon-oriented, which means that all the objects populating an OpenGL-like environment, however complex they can be, are based on polygons.

The art of programming with OpenGL consists of correctly sorting these polygons, in order to render back the expected result, but also of gathering these polygons into lists that allow a faster rendering. While the application size and the amount of data represented increase, the complexity of the development and the maintenance of such applications become problematic. The developers naturally need more high-level interfaces that could gather properly a great number of polygons. Such a group can then be called 'object' and placed into an environment called a 'scene'. As a result, the 'SceneGraph' was developed.

Using full capacities of a high-level object-oriented language such as C++, Java or Python, a scenegraph naturally encapsulates the properties of the 'objects' into classes. Each class contains the displayable information, but also a behaviour. All these objects are positioned into the scene and can then be linked together by the way of a tree, or more generally by a graph. From now, the name 'node' will then be preferred to 'object'. A node can have very various kinds of behaviour. It can be a simple displayable three-dimensional object, but also a container for multiple nodes, a position matrix that allows to rotate or translate another node upon it, a switch between child nodes, etc.

Figures 2.15 and 2.16 illustrate how a scene composed of several objects can be easily modeled by a simple graph. The drawable objects representing the Oxygen atom and the Hydrogen one are created only once. These are grouped into molecules H_2 and H_2O. Multiple instances of such molecules can be positioned in the scene by the way of the 'Position/Orientation' nodes.

Another important issue implemented by most of the scenegraphs is the camera metaphor. A camera is the encapsulation of all the information about the point and the angle of view of a virtual user in the scene, but it also has eye separation, the projection distance, etc.

Around the kernel previously described, scenegraphs also generally provide high-

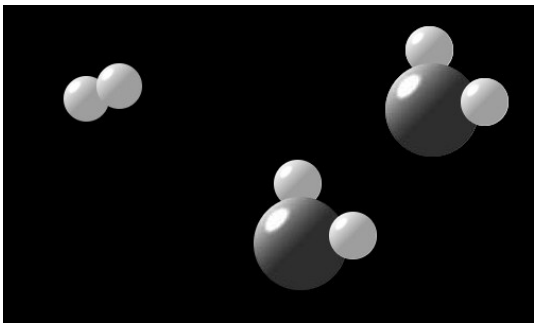


Figure 2.15: The representation of 3 molecules

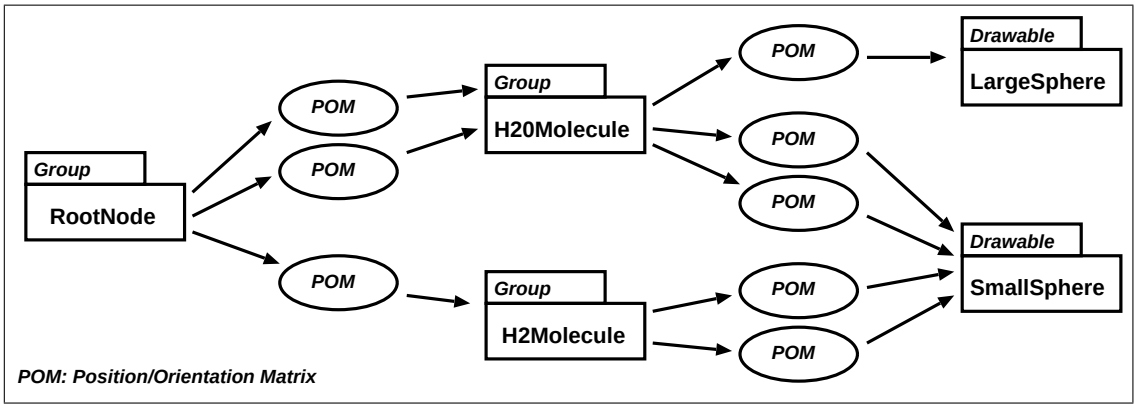


Figure 2.16: The graph associated to the molecules representation

level functionalities. These consist of convenient three-dimensional model and image loaders, multi-threading, multi-pipe rendering support, easy access to stereo modes and nodes manipulators or culling. The culling is a powerful filtering method applied on the scene in order to speed up the rendering. This consists of removing from the display list the nodes that are not in the field of view, using low-cost methods. It can dramatically improve the frame rate while visualising large scenes.

2.7.2 Presentation of the main scenegraphs

The main available scenegraphs according to the criteria required to develop an immersive archaeological virtual environment are presented hereafter. A good scenegraph must be cross-platform (which means 'open source' most of the time), efficient

because the data collection is to reach a large volume, flexible (particularly due to the dual-projector architecture detailed later), be able to load an important number of file formats and provides enough development support (documentation and community).

Java3D

Java3D is the Java-based portable scenegraph, natively object-oriented and platform independent. One of the main feature of this API is the ease of exportation through the Internet via the Java applets embedded in HTML pages. The developers would like to position this API at the highest programming level without loosing any performance due to the increasing distance to the hardware layer. As any Java application (run on a virtual machine) is slower that a C++ equivalent binary, it is hard to achieve such performance. The application considered in this project does not aim to be exported trough the internet, but may require complex multi-pipe rendering, therefore this API was not selected.

NVIDIA SG

At the end of the project, NVIDIA announced that they were about to release a new Scene Graph Software Development Kit called NVSG SDK (NVIDIA 2004). No version was available yet, but it was supposed to be based on C++ and OpenGL, to support cluster and multi-thread and to be extensible. It will be quite cross-platform: MS Windows and Linux may be supported.

OGRE - Object-oriented Graphics Rendering Engine

OGRE is a very young (launched in 2002) open source project that, at the opposite of all the other scenegraphs presented here, is not specifically designed for OpenGL and can use the Direct3D implementation (DirectX) as well. This scenegraph was

not created primarily for game development, but this utilization represents a large majority. The coverage of supported file formats is sufficient. However interesting its potential is, it remains a bit too immature for scientific visualization.

OpenInventor

Open Inventor is a powerful open source and cross-platform, object-oriented scene-graph initially developed by Silicon Graphics. Based on C++, it is built on top of OpenGL, which provides the most powerful graphic rendering. This is probably the most widely used scenegraph at the moment. It supports multi-pipe and is multi-threadable, which allows the exploitation of the full capacities of multi-processor architectures. Various high-level features like collision detection between objects in the scene or between the camera and the scene, CAVE integration, but also an image render via the internet are available. A very large panel of 3D model formats is supported as well as 2D images. A binding for Java is also available.

OpenRM Scene Graph

OpenRM is a multi-pipe scene graph based on a commercial technology. It really misses some plug-ins to read various kind of 2D and 3D files as common as 'VRML', 'obj' or 'bmp'. It is not implemented on Mac OSX. However, an interesting point is the publication of a programming guide, available for purchase.

OpenSceneGraph

OpenSceneGraph is a powerful and flexible object-oriented scenegraph relatively close to OpenInventor. This is an open source cross-platform scenegraph whose design is lead by two main developers. However young it is, it already supports many functionalities thanks to a very active community around it: for example multi-threading, multi-pipe rendering, some methods dedicated to object picking

or collision detection. It also supports one of the largest number of supported file formats. A binding for Python is available for prototyping purposes. As it was finally chosen for this project, its features will be detailed later.

OpenSG

OpenSG is a cross-platform scenegraph initially designed to become the successor of Open Inventor and Open Performer in 1999. Its capacities are unfortunately still at a quite low level, even though it supports multi-thread, hidden faces culling, etc. However, some interesting features are already available like improvement of performance by gathering and stripping automatically the polygons in the scene.

SGL

SGL is a cross-platform open source scenegraph developed by a small team composed of nearly ten programmers. This is a fair, balanced library. Its syntax is very close to OpenSceneGraph. It provides some geometry routines like Bezier surfaces and supports a fair number of file formats. The documentation mainly consists of Doxygen-based HTML pages.

2.7.3 Choice of the scenegraph

Considering the key features compared in Table 2.2, there are only two scenegraphs worth considering further: OpenInventor and OpenSceneGraph. All the others miss an important issue such as cross-platform, supported file formats number or efficiency. OpenInventor provides more documentation and is older (and then more stable) than OpenSceneGraph. However, trusting the active OpenSceneGraph community and considering that flexibility is the keystone of such an undefined environment like this project, OpenSceneGraph was finally selected.

Scenegraph	Core language	Cross-platform	Supported files	Efficiency	Flexibility & extensibility	Documentation
Java3D	Java	+++	+	-	++	+++
NVIDIA SG	C++	+	?	?+++	?++	?
OGRE	C++	-	+	+	-	+
OpenInventor	C++	++	+++	+++	++	+++
OpenRM SG	C++	-	- -	++	?	++
OpenSceneGraph	C++	++	+++	+++	+++	-
OpenSG	C++	+	++	+	?	+
SGL	C++	++	++	++	++	+

Table 2.2: Summary of the main scenegraphs capacities

2.7.4 VR Juggler

Although VR Juggler is not a scenegraph and has not been used in this project, it should be shortly introduced. VR juggler is an open source, operating system independent toolkit that provides easy access to most of the virtual reality systems. It allows to develop easily a portable multi-scalable application. This takes into account the size of the display (desktop screen to work wall), its shape (flat, parabolic, cubic like CAVE, etc) and even the tracking systems than may be used. Simple XML configuration files allow to change the configuration, even during runtime.

2.8 The CIDOC 'Conceptual Reference Model'

The CIDOC (International Committee for Documentation of the International Council of Museums) has worked for about ten years on a 'Conceptual Reference Model' (CIDOC 2004) referred to as CRM. This emergent model provides an accessible and flexible storage of cultural heritage documentation. It defines explicit and implicit relationships and concepts. By providing such a framework, it aims to integrate all

sources of documentation from museums to libraries. This model is sufficiently open to match any requirements: it can define the structure of a museum database as well as a chess game or a car description.

Technically speaking, the CRM proposes hierarchical entities and properties (relations) between these entities. Both can be declared by a description (string) or a unique identifier ('En' for the entities, 'Pn' for the properties). For example, the concept of **'digging'**, instance of the entity `(E7: Activity)` can be linked to **'the archaeologist'** `(E39: Actor)` via the property `(P14: carried out by)`. The following sentence as been stored: '**Digging is carried out by the archaeologist**'.

2.9 Summary

Most of the elements constituting an archaeological virtual environment have been reviewed.

The current archaeological information software allows nowadays to gather and interact with two-dimensional layers and pay little attention to the third dimension. For example the crop marks, these effects of buried material on the vegetation growth, are considered as a simple layer, despite their crucial meaning in archaeology. The digital elevation models are represented by grey scale, which gives almost no feeling of their three-dimensional nature.

An immersive environment surrounds the user and therefore provides him the feeling that he is part of this virtual 'world'. The power of such an environment for scientific visualization, as demonstrated by many authors, has the potential to significantly improve some archaeological tasks and methods. Examples of current archaeological virtual environments have been demonstrated.

As the back bone of such an environment would naturally be a Geographic Information System, the author presented some aspects of the GIS. This consists mainly of a database and an interface. It seems valuable to use a scenegraph to

implement such a GIS, as the development is rapid and the performance improved compared to a standard OpenGL application.

Given its flexibility, the CIDOC 'Conceptual Reference Model' is to be the glue between all the information contained in an archaeological virtual environment. This should also open the environment to more data sets.

Chapter 3

OpenSceneGraph

3.1 Introduction

OpenSceneGraph (referred to as OSG) is a powerful cross-platform graphics toolkit on top of OpenGL, designated for three-dimensional rendering. It runs on MS Windows, OSX, GNU/Linux, IRIX, Sun Solaris and FreeBSD operating systems. As its name indicates it, OSG is an open source project lead by Robert Osfield and Don Burns and supported by a small number of developers. It can easily be qualified as young, as it was started in 1998 and still remains under development, in 2004. For this reason, the OpenSceneGraph source code is updated daily and frequent CVS (Concurrent Versions System) check-ins are required to use the full power of this growing library. A dedicated active mailing list allows the developer to track and solve very quickly appearing bugs, but also to submit ideas of improvements and of course to seek for help. This last issue is very useful since there is no specific documentation to OpenSceneGraph. As a matter of fact, the only way of learning OSG is to browse the provided panel of examples and the source code. Therefore it is appropriate to provide a detailed overview within this thesis.

3.2 Architecture

3.2.1 A scenegraph

The OpenSceneGraph library is based on a robust scenegraph, a concept described earlier. The classes that are defined in the core of OSG can be separated into three groups: the low-level objects, the states and the graph itself. The low-level objects consist of vectors, arrays, clean pointers, matrices, etc. The states are generally some simple encapsulations of the OpenGL functionalities such as fog, drawing attributes, textures or drawable objects. Finally, the graph of the scene can be designed by the classes inherited from the 'Node' class. Theses classes re-encapsulate a set of states than finally create an entity, like a drawable object, a positioning node or a grouping one.

The implementation of the hierarchical structure of this graph is very robust, allowing the programmer to manipulate the objects, without thinking too much about memory management. For example the destructors of all objects of a sub-graph are automatically applied if the considered sub-graph is not linked anymore to the root node of the main scene graph.

3.2.2 An open project

The well-designed hierarchy of OpenSceneGraph provides useful flexibility. Therefore, there is an intense activity around OpenSceneGraph and many developers had the clever idea to submit not only some improvement proposals, but also some complete pieces of source code[1] includable in OpenSceneGraph. This is for example the case of some specific picking techniques which were really useful for this project and have been integrating into OSG a couple of days after the help request by the author, but also a wide range of file readers as shown in the next section. The port

[1] An independent piece of code on top of a scenegraph is generally referred to as 'nodekit', merging the node of the graph with the typical toolkit.

of OSG to the Python prototyping language is another excellent demonstration of the OpenSceneGraph flexibility.

3.3 Main plug-ins and add-ons

3.3.1 Producer and Viewer

The Producer toolkit is a high-level nodekit designed to manage the interaction between the user and the scene. Its 'Viewer' class automatically configures a set of cameras which will be the 'windows' to the scene. This consists of the lens of the cameras, the stereo mode or the clipping planes for example.

The Producer also proposes a very powerful navigation manager. Three navigation methods based on keyboard and mouse are available. First an object manipulator, which basically simulates that the mouse is grabbing the scene as an object in order to move it. Secondly, a driving simulator that follows the terrain surface and uses the mouse as a steering wheel. Finally, a simple flight simulator based on the classical is a steering gaze metaphor.

Like some other scenegraphs, this viewer uses three main stages to render a graph:

- **'sync'** waits for the other tasks (update/frame) to complete.
- **'update'** traverses the graph and update the nodes to consider the manipulators, the animations, etc (simulation process).
- **'frame'** processes the culling (remove invisible nodes) and renders all the cameras.

The Producer toolkit also implements multi-threading and many functions such as picking (function that computes a ray intersection algorithm and extracts the

name of the object represented at a given pixel), switch between point, wire and surface representation, texture map toggling, frame rate or CPU usage display.

3.3.2 Supported file formats

This non-comprehensive (OSG grows everyday) list of supported file formats provides an idea of the huge ability of OpenSceneGraph to integrate almost any kind of file.

2D images

- **bmp** - *MS Windows bitmap*
- **gif** - *CompuServe*
- **jpeg** - *Joint Photographic Experts Group*
- **tiff** - *(partially supported)*
- **pic** - *Macintosh Quickdraw/Pict*
- **png** - *Portable Network Graphics*
- **rgb** - *Silicon Graphics image*
- **tga** - *Truevision Targa*
- **x** - *Stardent AVS X*

3D models

- **3dc, 3ds** - *3D Studio Max*
- **acd** - *AC3D modeler*
- **dds** - *DirectDrawSurface*
- **dw** - *Design Workshop*
- **dx** - *IBM Data Explorer*
- **geo** - *Carbon Graphics*
- **iv** - *Open Inventor (close to VRML)*
- **lwo, lws** - *Light Wave*
- **md2** - *Quake 2 Player Model*
- **obj** - *Alias Maya Wavefront*
- **osg** - *OSG Native Format*
- **pnm** - *supports PPM (Portable Pixmap), PGM (Portable GreyMap), PBM (Portable Bitmap)*
- **txp** - *Terrex Terrain Page*
- **wrl** - *VRML (Virtual Reality Modeling Language)*

Miscellaneous

- **flt** - *OpenFlight*
- **tgz** - *Tar+Gnu-zipped file*
- **zip** - *Zipped file*

Noticeable not supported format

- **psd** - *Adobe Photoshop*

3.4 OpenSceneGraph-based projects

Many high level projects rely on OpenSceneGraph for their three-dimensional rendering. These are as various as games, simulators or scientific applications. 'Pirates of the XXI century' (Diosoft 2004) is a beautiful commercial first-person action game which takes in tropical climate, illustrated in Figure 3.1. (The Combat Simulator Project 2004) is an open source project, whose flight simulator is being developed using OSG. 'BARC' is a realistic virtual environment that simulates sailing races, but can also follow the boats in real time by the way of a broadcast of the embedded GPS coordinates, shown in Figure 3.2. The 'NASA Blue Marble' open source project aims to represent the whole NASA satellite photographs using a sphere as a metaphor of the Planet Earth as represented on Figure 3.3.

Figure 3.1: OSG example: Diosoft's game 'Pirates of the XXI century'

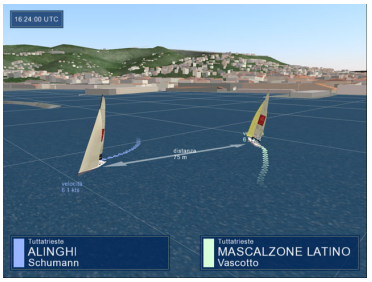


Figure 3.2: OSG example: BARC sailing races

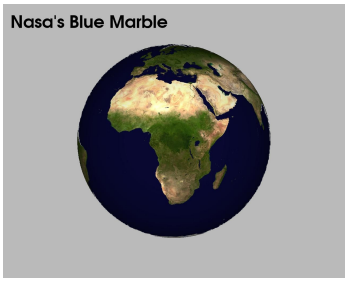


Figure 3.3: OSG example: NASA Blue Marble

3.5 Summary

The directions of research within this project changed frequently, necessitating the robustness and flexibility offered by OSG. As the aim is to represent the maximal amount of data in the same context, the wide scale of supported file formats will obviously be very useful to improve the application without any additional effort. More generally, using OSG should reduce significantly the technical aspects of a 3D-rendering context to focus on the interactions between the scene and the user.

Chapter 4

System Architecture

4.1 Overview

Two variations in architectures were used during this project: a development platform on a simple PC and the immersive one based on the HIVE (Hull Immersive Visualization Environment). Both were running Microsoft Windows XP and supported the following input devices: a wireless gamepad, a finger-bend tracked glove and a motion capture system (also called 'tracking system') for the head-coupled stereo as shown within the Figure 4.1. Two main differences between these platforms are the size of the screen and the tracking system, but the conceptual architecture is the same. Each phase of development was followed by validity and tuning tests in the HIVE.

4.2 Development platform

4.2.1 Operating system and compiler

OpenSceneGraph could have been used with Python, a prototyping language that allows methods to be tested quickly. However, C++ is the native language of Open-

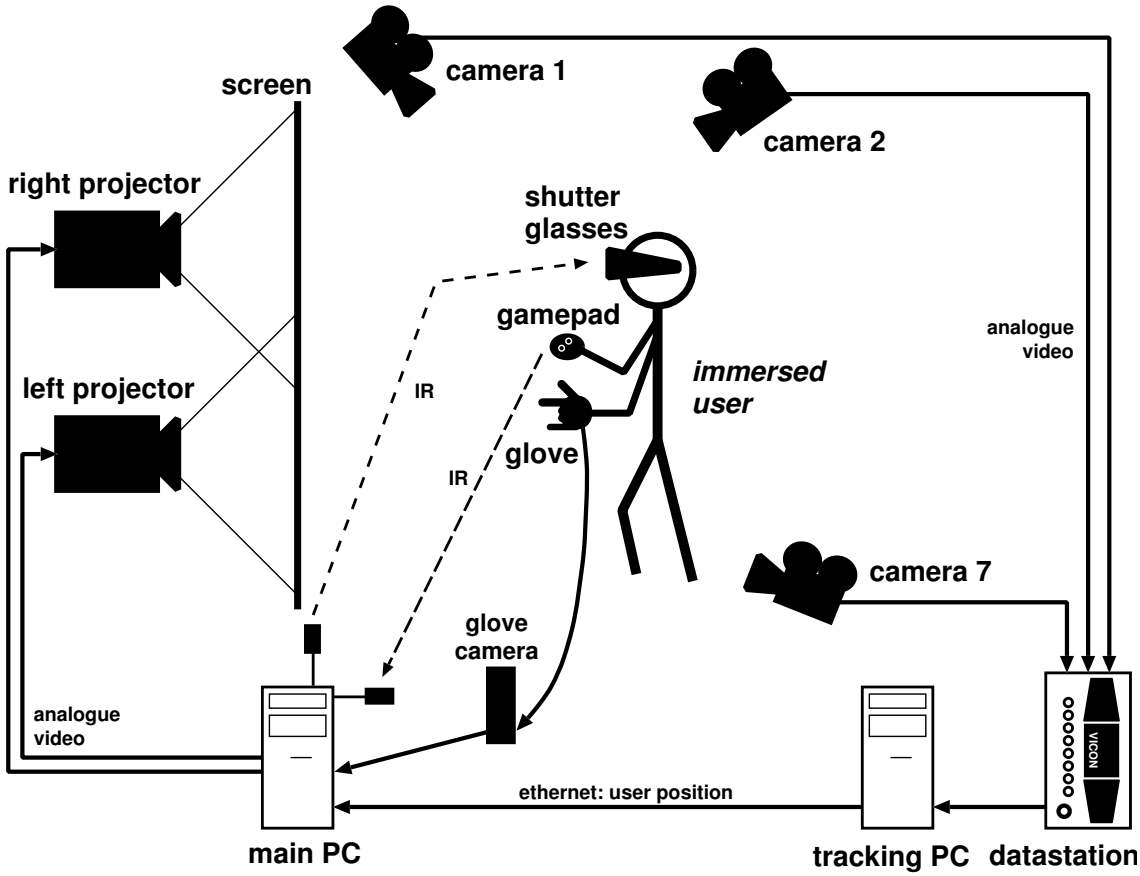


Figure 4.1: Overview of the architecture of the HIVE

SceneGraph and the Python binding[1] is not always up-to-date. Moreover, Python is slightly slower than C++, and a second implementation in C++ of the critical methods would have been necessary. Therefore, C++ has been retained.

Microsoft Windows XP and the C++ compiler of Microsoft Visual Studio .NET running on a desktop PC have been used as the development framework[2]. The final application also runs on a Microsoft Windows system, displayed on a rear-projected wall screen. However, the application was developed in such a way that it could also be compiled and run on GNU/Linux operating system[3]. This should increase

[1] A Python binding is a wrapping piece of code that allows to use the C/C++ functions/classes inside a Python source code.

[2] Except the Qt-based interface which has been developed with (Mandrake Linux 10.0 / gcc 3.3.2) and ported to MS Windows afterward.

[3] All the libraries involved were cross-platform. However, no global tests have been performed on GNU/Linux.

the reusability and the ease of exporting the application onto other visualization facilities.

4.2.2 Tracking system for test purposes

In order to test the motion capture in a simple environment based on a desktop PC, a light and easy to install magnetic tracking system was used. A Polhemus Fastrak (Polhemus 2004*b*) with a single sensor has been connected to the PC. The Figure 4.2 shows this device.

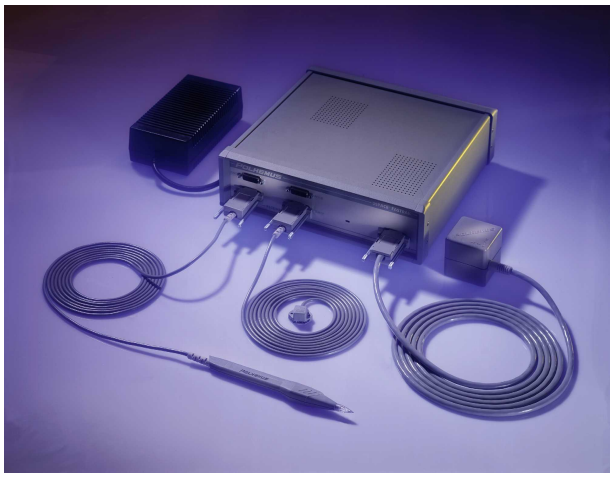


Figure 4.2: Polhemus Fastrak (magnetic motion capture system)

This system is able to capture a six-degrees-of-freedom (position and orientation) value with an excellent accuracy in a radius of 75cm (0.7 millimetres RMS for position / 0.15 degrees RMS for orientation) and a correct one within 3 metres. It works quite fast as well, with an update rate of 120 Hertz and a latency of 4 milliseconds. However, it remains sensitive to stray magnetic fields. Cathode-ray screens are particularly prone to throw the sensors into panic, and even the metallic structures of the tables and the PC are disturbing as well. The worst side effect experienced remains the field emitted by other similar tracking devices.

Up to four sensors can be connected to the same emitter[4], but only a single one was used for this test purpose. It was bonded on the user's head in order to acquire the point of view for the tracked stereo. The system was accessed via the serial port using the toolkit of J. Ward.

4.3 The HIVE (Hull Immersive Visualization Environment)

4.3.1 Wall screen

The display system of the HIVE Auditorium is based on a wall screen (also referred to as 'work wall'), whose size is about 6m long by 2.5m high. In order to cover the entire human viewing angle of 60 degrees, the user has then to stand up at less than d metres from the screen where $d = \frac{\frac{2.5}{2}}{tan(\frac{60}{2})} = 2.2$. In order to allow the users to walk very close to the screen, this screen is rear-projected. This means that the projectors are located behind the screen and this system eradicates totally the classical user shadows problem as can be seen in Figures 4.3 and 4.4. As the ratio $\frac{covered\_surface}{user\_distance}$ is quite large, a very good resolution is required to avoid an obvious rastering of the image. As a matter of fact, two projectors have been installed to cover the whole wall.

This leads to a minor problem of smooth interface between the coverage of each projector. An overlap of 25 percent has been set up. This means that the centre of the screen is covered at the same time by the right quarter of the left projector and the left quarter of the right one (see Figure 4.4). A dedicated hardware (blender) applies a linear decrease on the intensity of the video signal in the middle part. The screen display is then smoothly blended and its intensity remains constant along the projectors overlap.

[4]The update rate is then divided by the number of sensors.

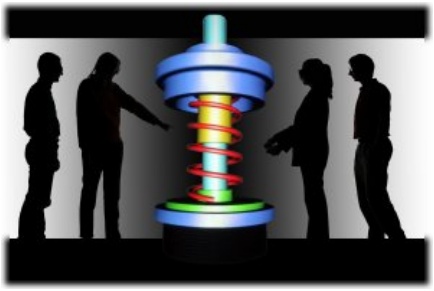


Figure 4.3: The HIVE in action

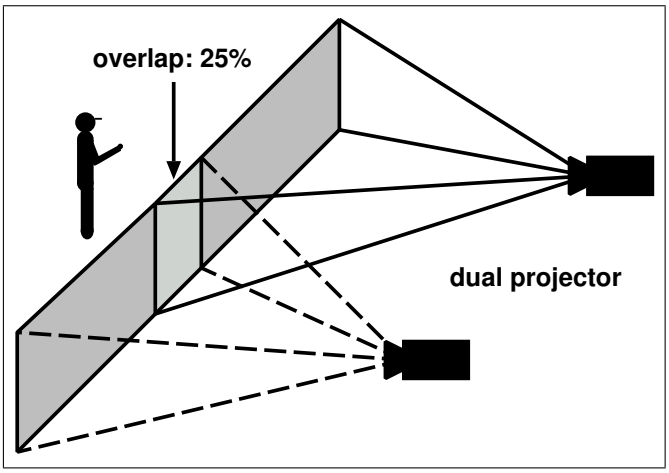


Figure 4.4: The dual rear-projected wall screen of the HIVE

4.3.2 Tracking system

The impressive size of the work wall of the HIVE allows the user to move in a large volume in front of the screen. Actually, this is the volume of a complete room: around $6m \times 3m \times 2.5m = 45m^3$. The magnetic tracking system previously described, with its wire attaching the user to the ground and a maximal work volume of around $\frac{4}{3}\pi 1^3 \simeq 4.2m^3$ is not convenient in such an environment. The HIVE relies then on an optical tracking system.

This Vicon motion capture system (Vicon 2004*a*) is composed of seven[5] infrared cameras dispatched all around the work volume and a DataStation that computes the position of the tracked items. The 'Vicon Mcam2' camera (Vicon 2004*b*) presented within Figure 4.5 also embeds a set of infrared diodes. The infrared signal is reflected by the markers (small white balls) to be tracked and captured by the camera. This camera can acquire up to 1000 frames per seconds with a resolution of 1280×1024 pixels (total 1.3 mega-pixels). Up to 24 cameras can be connected to the 'Vicon 8i DataStation' illustrated in Figure 4.6. This station calculates the position (in

[5]The volume in front of the screen of the HIVE is covered by seven cameras. However, more cameras are available in the room.

each camera frame) of the white markers and sends these positions to a standard PC. This computer runs the Vicon iQ software (screen capture in Figure 4.7) that computes the position and the orientation of predefined items (for example shutter glasses, gloves or magic wand). The positions are then served via the ethernet to any computer that requires them. J. Ward's Tracklib library was used to perform the connection.

Figure 4.5: Vicon Mcam2

Figure 4.6: Vicon i8 DataStation

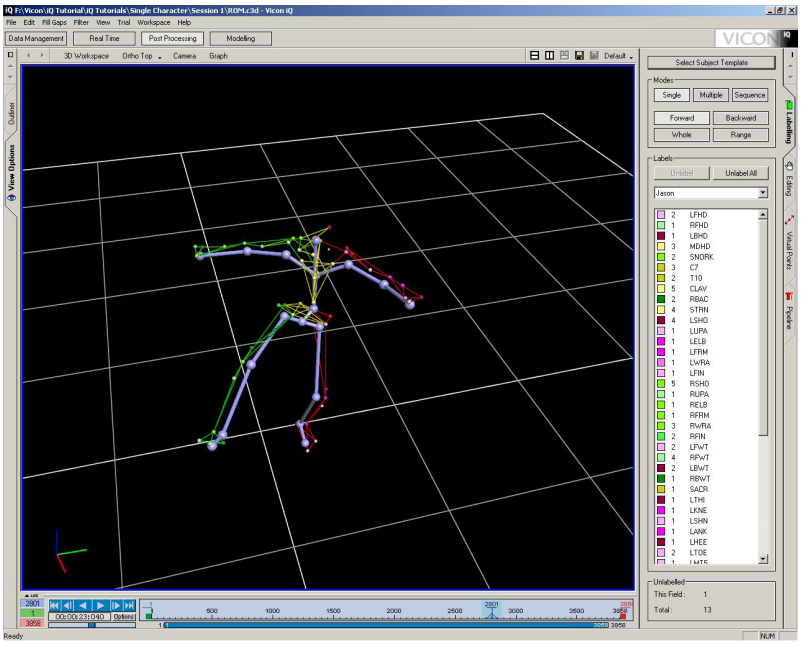


Figure 4.7: Vicon iQ software

The average precision was around one millimetre, which is sufficient for the needs of the application. The coverage provided by seven cameras is excellent and avoids most of the time the occlusion of the markers by the user's body.

4.3.3 Graphical processor

The classical method for using such a display system is to associate one computer per projector. This leads obviously to some difficulties in terms of management of the machines and of clean display. The best solution consisted of launching the same application on both of the computers and synchronizing them using the ethernet. A

first approach lead to consider a master and a slave. The master got all the input parameters from the user and simply sent some UDP[6] packets to invite the slave to change its state. These packets could convey information like the camera position, an object selection, as demonstrated within Figure 4.8. However, as UDP does not ensure the transmission of the packets and as this transfer can be relatively long, the synchronization was not accurate enough. In the best case, a small delay could be observed between both sides of the screen, but in the worst, some objects of the scene could be displayed on one side only.

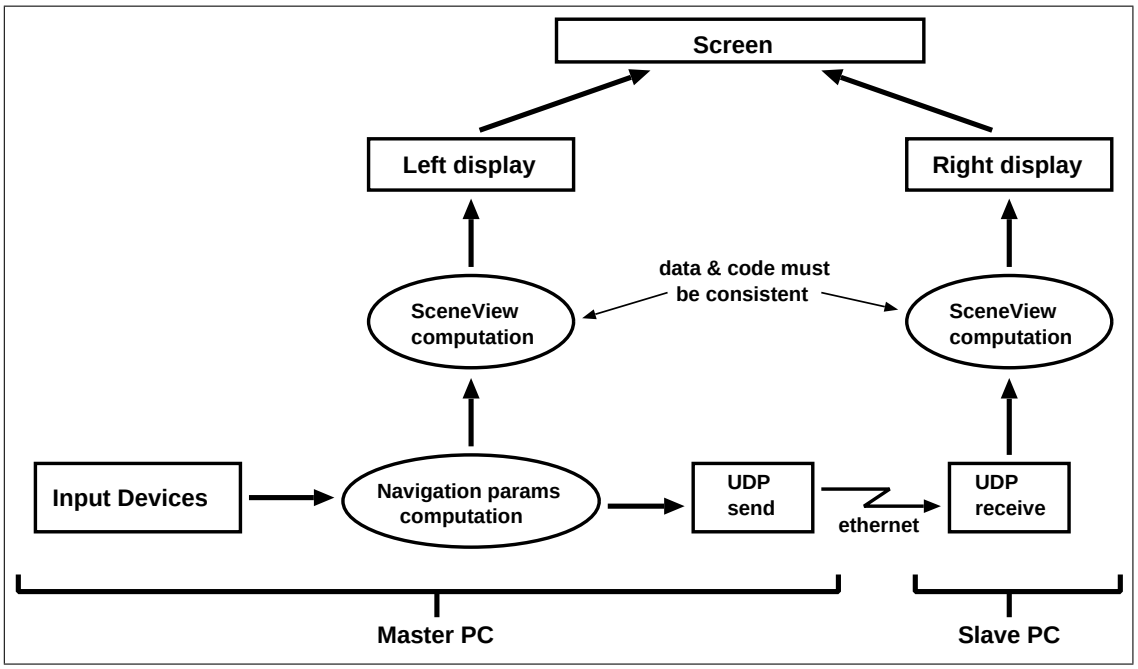


Figure 4.8: Dual computer scene rendering (finally not selected)

For this reason, and in order to improve the graphic performance, a new machine including a dual output video card has been installed. The wall screen is then considered as a single screen. At the end of the project, the data set available and loadable into the application was very large (around 200MB are available). The visualization of all the information at the same time is only possible on very powerful station including a high end graphic card. 128MB of video memory seems to be a

[6]UDP: User Datagram Protocol.

minimum to load correctly two layers of high quality textures. An Intel Pentium 4 Extreme clocked at 3.2Ghz was used, with 2GB of memory and a NVIDIA Quadro FX3000 as video card. This card embeds 256MB of memory and displays a total of 2560x1024 pixels, which represents 2240x1024 pixels after overlap blending.

4.4 Input devices

4.4.1 Wireless gamepad

A single wireless gamepad has been included in the interface: a Logitech WingMan Cordless, consisting of two mini-joysticks, a slider, one 'hat' (eight binary cardinal directions on a flat circle) and eleven simple buttons as partially shown within Figure 4.9. The navigation with a gamepad seems to be more intuitive than the mouse version and also provides the necessary movement freedom.

As OpenSceneGraph does not support gamepads for the moment, the Simple DirectMedia Layer (SDL) library was used to interface such hardware. SDL is an open source largely cross-platform (at least twenty supported platforms) and cross-language (more than fifteen languages bindings) multimedia library designed to interface easily audio, keyboard, mouse, joystick, 2D and 3D hardware. The utilization of this library is consistent with a cross-platform application.

4.4.2 Finger-bend tracked glove

A 'P5 glove' device, produced by Essential Reality (Essential Reality 2004) has been integrated into the application. This includes an optical three-dimensional tracking system (based on a static camera and eight infrared LEDs bonded on the body of the glove) and bend sensors on each finger. A noticeable weakness is that the glove is linked to the camera by a wire (length: 180cm). Both glove and

Figure 4.9: Logitech WingMan Cordless

Figure 4.10: P5 finger-bend tracked glove

camera are illustrated within the Figure 4.10[7]. The camera is USB-connected to the computer. It is important to note that this low-cost glove is mostly designed for video game addicts, and therefore its accuracy does not reflect the state of art in tracking systems.

In the framework of this project, the finger sensors were particularly useful. They simply measure the bend in the fingers independently, therefore it is impossible to retrieve the angles between the fingers. It provides an accuracy of 0.5 degree for a refresh rate of 60Hz. This is sufficient to recognize simple movements or static postures of the hand, but not for complete gesture recognition.

Although less innovative, the tracking system was used as well. The working space is a small one-metre-radius half-sphere. The refresh rate of the 6 degrees of freedom (position and orientation) is 45Hz, which is good enough for simple motion capture. The accuracy of the tracking system at one metre from the camera is 12mm

[7]Note that the scale of the glove and the camera is not the same. The camera is actually around 30cm high.

for the position and 3 degrees for the orientation (around 10 to 20 times worse than the Fastrak and the Vicon systems).

Two libraries could have been used to access the glove: the official SDK (Software Development Kit) library provided by the manufacturer for both MS Windows and GNU/Linux and an open source library. The main weakness of the SDK is that the origin of the coordinates is not static, which involves many inconvenient calibrations. The open source library that also supports OSX was also tested, but the source code is really messy and the values of the positions have not a fair accuracy at all. Finally, the SDK was used.

Note that, as described later, the tracking system of the glove was eventually replaced by the Vicon.

4.5 Summary

The large work wall of the HIVE is an appropriate support for immersive virtual environment: its surface easily covers the human field of view and a pair of rear-projectors ensures the ability for the user to walk as close as required to the screen to reach an immersive state. The motion capture (tracking) system involved in the head-tracked stereo mode is dedicated to a Vicon set of seven infrared cameras.

The computation of the three-dimensional rendering provided by a powerful PC should ensure an excellent frame rate, even in stereo mode. In order to maintain a correct immersion feeling and freedom of movements, a wireless gamepad and a finger-bend glove interfaces are also integrated.

Chapter 5

Software Architecture

5.1 Introduction

The software architecture will be detailed following a top down approach. Firstly, a brief presentation of the conceptual steps required to include a piece of archeological information in the application. Secondly, the software design: the libraries used, the way to ensure modularity. This will be concluded by a UML graph of the main groups of classes. Then the database design using the CIDOC Conceptual Reference Model will be described. Finally, a short description of the Graphical User Interface (GUI) for system configuration will be provided.

5.1.1 Disclaimer

All source codes of this prototype described in this chapter and the two following ones have been written by the author with the exception of the bitmap input/output class, developed by W. Viant, the gamepad access class, which is based upon a code from J. Ward and the ASCII DEM loader which is based upon a W. Viant's code.

The OpenSceneGraph[1], SDL, libxml2[2], P5 SDK and Tracklib libraries were also used in this project.

5.1.2 The operation chain

The Figure 5.1 demonstrates the global way to visualise the raw data by the application. First of all, the data is filtered in order to remove useless components, to ensure its consistency and to store the information in a convenient format. This can be for example the translation from the 3DS file format to the OSG one or the crop of a huge satellite image according to a grid. The result is an orthogonal, well-formatted database. The modeling aims to interpret this set of data in a human-understandable way. For example, some tables of numerical values have been modeled by a spatial field of sticks. Finally, the data applied to the models is presented to the final user.

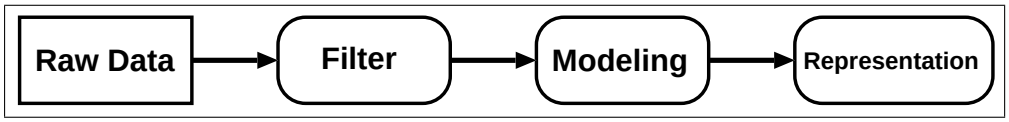


Figure 5.1: Filter-Modeling-Representation approach

5.2 Software design

5.2.1 Libraries dependencies

The Figure 5.2 represents the libraries that were relied on. OpenSceneGraph is compulsory, as many classes of this project inherit from OSG. The other libraries are optional, depending on the functionalities available in the application.

[1]The 'Producer' and its dependencies ('OpenThreads') have been used as well. The 'nodekit' OSGterrain has not been used.

[2]The utilization of libxml also involves 'iconv' and 'zlib'.

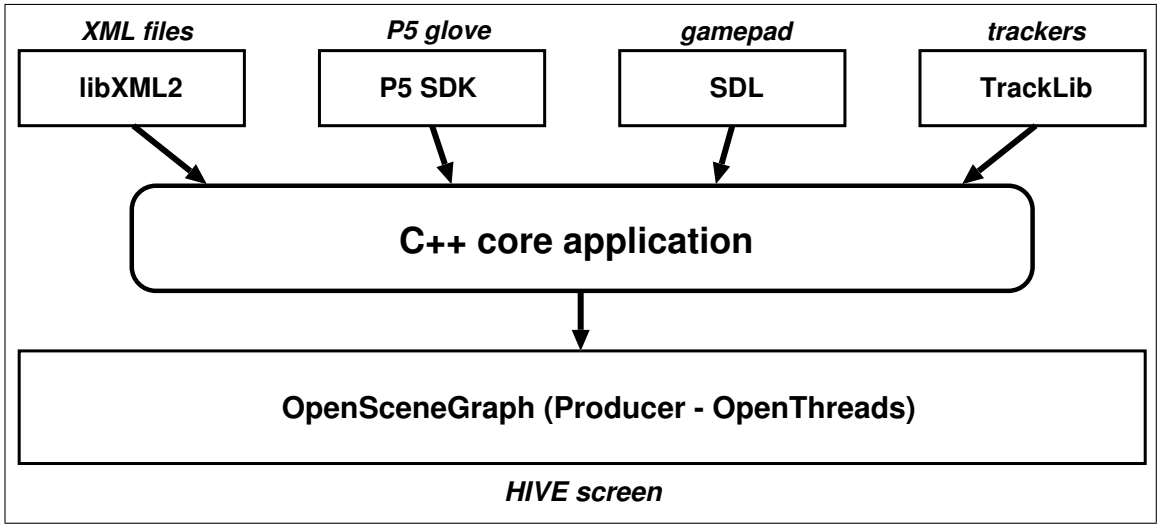


Figure 5.2: Libraries dependencies

5.2.2 Generalities

The application has been designed to be as modular as possible, therefore it is totally object-oriented. Each part of the virtual environment is an independent class or a set of classes. This allows easy addition or removal of the parts of the implementation (except a couple of classes that are central). This was useful, and even necessary during the implementation phase, as it was very often difficult to predict the precise direction of the research. Adding the support for a new type of information in the application involves typically three types of classes:

- **A file loader** converts the information as expected by the application. It may include a simple parser.
- **A graphic part** defines the representation and sometimes a simple behaviour. Most of these classes directly inherit from the OpenSceneGraph classes.
- **A linker** describes the complex behaviour of the object in the application by linking it to all the other components.

5.2.3 Linking the elements

The dispatcher

All these sets of classes are initialized by the 'main' function at the starting up of the application, and unified by a central 'dispatcher' class, that contains the (dynamic) links between the classes. It also routes the signals arriving from the keyboard, the mouse, the gamepad manager and the finger-bend tracked glove via the 'key dispatcher' to the corresponding classes. Therefore, if a component has not been added to the environment or has not been registered by the dispatcher, this will not affect the other classes. They will continue to inform the dispatcher of their 'events' which will simply ignore them if necessary.

The key dispatcher

(Faisstnauer, Schmalstieg & Szalavári 1997) presented a method for dynamically allocating input devices like keyboard, mouse, trackball, joystick or tracked glove that could have been useful. The configuration of a new device only consists of setting up the parameters of a new device like the number of degrees of freedom (DOF) or the types of button. The final user can transparently plug in and out these devices. However, this method leads easily to non-intuitive systems due to some weak automated associations between incoming events and output tasks.

The 'KeyDispatcher' class allows the change, without compilation, of the associations between the keys pressed (from keyboard, mouse, wireless gamepad or finger-bend glove) and the tasks. A simple text file composed of links is loaded at startup (`keylinks.ini`). The following small example shows how it is easy to connect the End key of the keyboard to the task which removes the widget selection, the PageUp key of the keyboard and the X button of the gamepad to expand the billboard set, the R1 button of the gamepad to pick a widget, or even the right thumb up (using the finger-bend glove) to raise up the sea level.

keyboard.end	widget.unselect
keyboard.pageup	billboardset.expand
gamepad.x	billboardset.expand
gamepad.r1	widget.pick
righthand.thumb	sea.upper

Several keys can be linked to the same task, which launchs a task using either the keyboard, the gamepad or the finger-bend glove. Several tasks can also be connected to the same key, therefore several actions can be performed while pressing only one key.

5.2.4 Main groups of classes

The Figure 5.3 represents the main groups of classes. Note that there are more than 70 classes in the core project: not all the classes are represented. Moreover, some groups of classes have been represented as a single class and some links have been omitted. With the exception of 'P5SDK' and 'SDLlib' which are two libraries and 'Keyboard' and 'Mouse' that are only simple symbolic names, all the classes here have been developed by the author, and based upon OpenSceneGraph. Some simple examples of functions calls chains are demonstrated within Figure 5.4.

Moreover, some pre-processing is completed off-line and therefore not linked directly to the virtual environment. They simply produce or modify files that will be used by the main application. This is the case of the crop marks classifier, the density map generator and the Graphic User Interface (GUI) for the configuration files.

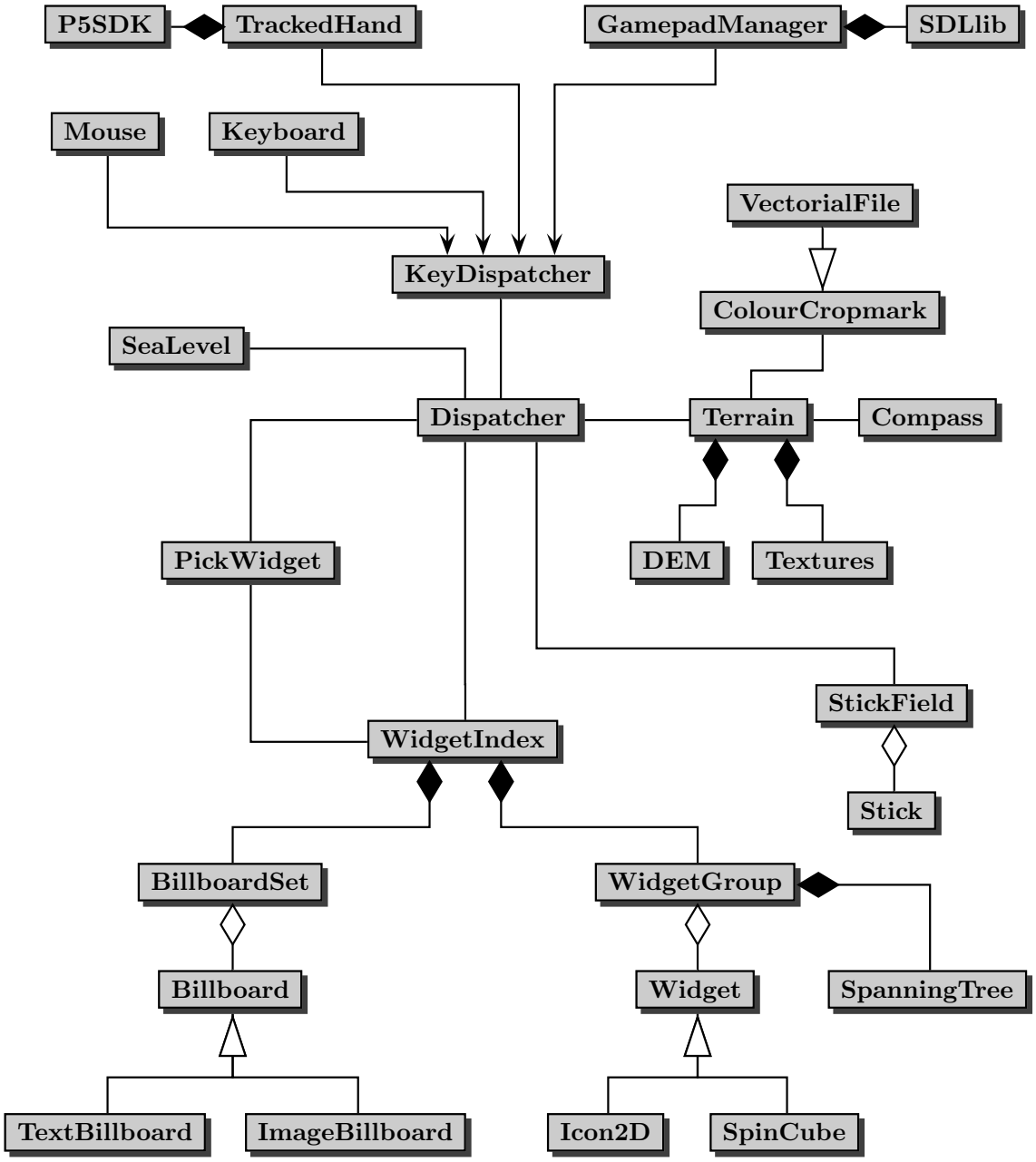


Figure 5.3: Main classes of the implementation

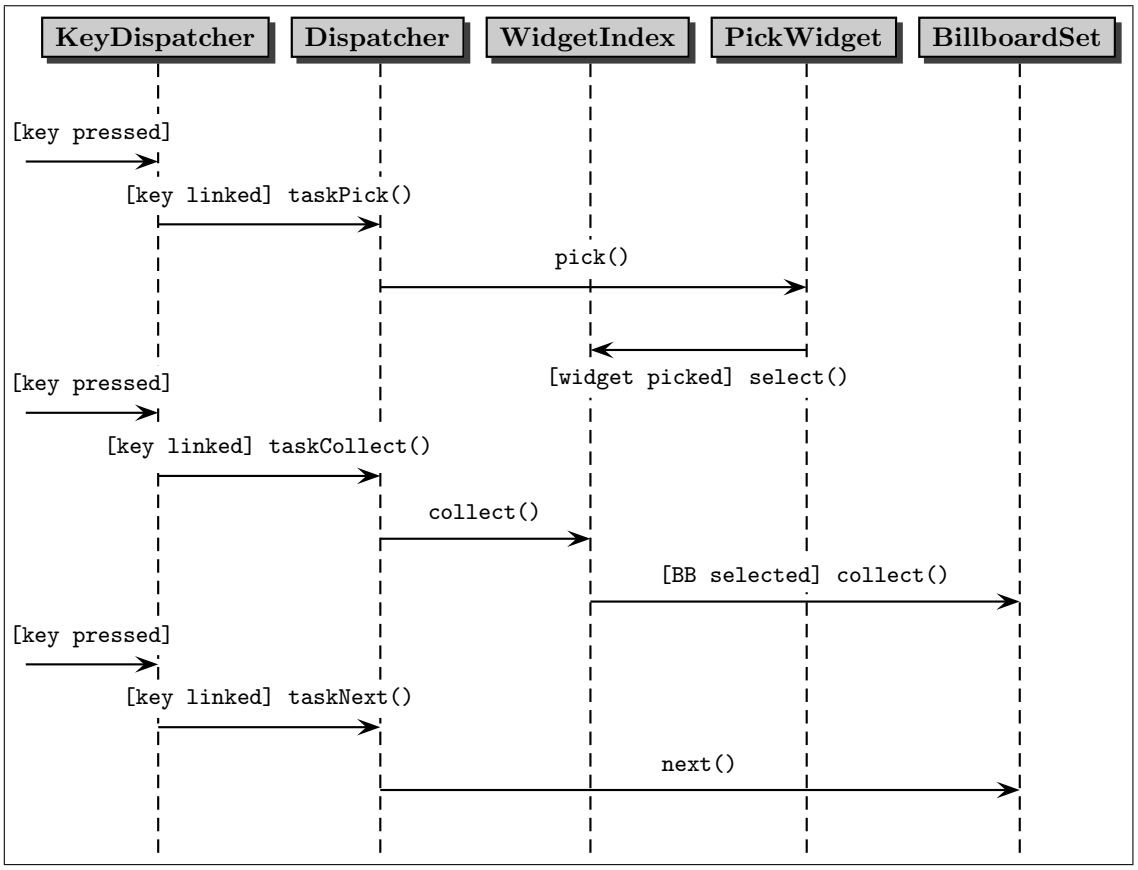


Figure 5.4: Sequences of functions calls

5.2.5 Extensibility

Configuration flexibility

As some parts of this project were experimental and as the application was supposed to run on two different platforms (PC and HIVE), a set of files allows the developer and the user to tune the configuration of the virtual environment. These files allow changes to many parameters without recompiling. This includes for example the sensitivity of the gamepad, the tracking system, the key-task links previously described or the size of the screen. Most of these files are described in Appendix B.

Dependency to OpenSceneGraph

It is noticeable that the source code strongly relies on the OpenSceneGraph (referred to as OSG) structure. Most of the basic objects like vectors or matrices are stored using the OSG classes. Almost the whole classes involved in the project inherit (directly or indirectly) from the OSG classes. As OpenSceneGraph grows quickly, this strong dependency often generates deprecated functions calls, or useless pieces of code. For example, the new module 'osgTerrain', not used in this project, could load and represent the Digital Elevation Model (DEM).

5.3 Database design using the CIDOC 'Conceptual Reference Model'

5.3.1 Aim

The aim is to design a dictionary of archaeological objects stored using the widely used CIDOC 'Conceptual Reference Model', in order to be able to integrate finds records from a museum. The required information consists of a name, a location in the scene (on the terrain), a type of widget used (the concept of widget will be detailed later – it consists basically of a representation and a behaviour) and a file which is the container of the archaeological information.

The Extensible Markup Language (W3C 2004), referred to as XML, was naturally chosen to implement the 'Conceptual Reference Model', as this is the most common file format for this purpose.

5.3.2 Modeling of the information

Considering the attributes required to store an archaeological object, the more appropriate base class of the CRM was the 'Physical Object'. It was then decided to

use the following CRM entities and properties to store the information, presented within Figure 5.5. They match the requirements of the widget index class. It is important to note that there is no unique way to formalize the data set into the CRM. Many different combinations of entities and properties have been considered and could have been used as well. The author finally decided to choose the simplest one.

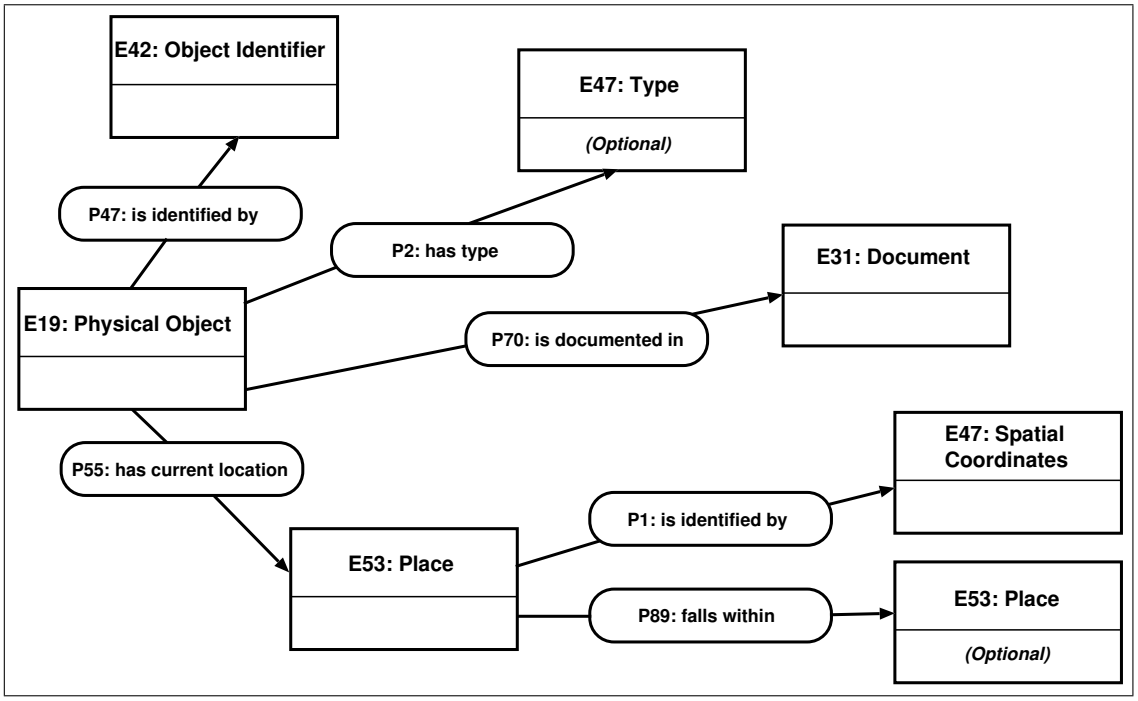


Figure 5.5: CIDOC entities and properties usage

An interpreter of the CRM (version 4.0, formerly 3.4.10) has then been developed, based upon libxml2 (libxml 2004). The utilization of this library is consistent with a cross-platform application. It can load XML CRM files, on the proviso that for each object is specified a reference document and a place (location) in GPS-like coordinates.

The XML CRM file generated was considerably larger that the simple dictionary used at the beginning of this project. Note that the place was described twice: using the exact GPS coordinates; and less precisely ('Hayton Region') by the way

of the 'falls within' property. Note also that the same property description (is identified by) can belong to more than one unique identifier, depending on the domain of application (in the entities space). The 'Type' and the second 'Place' entities are optional. Any other entity that would be added to this model is simply ignored.

```
<CRM_Entity>Aerial Picture of Hayton
    <in_class>E19 Physical_Object</in_class>
    <P47F_is_identified_by>AerialPicture
        <in_class>E42 Object_Identifier</in_class>
    </P47F_is_identified_by>
    <P55F_has_current_location>5100 22600
        <in_class>E53 Place</in_class>
        <P1F_is_identified_by>5100 22600
            <in_class>E47 Spatial_Coordinates</in_class>
        </P1F_is_identified_by>
        <P89F_falls_within>Hayton Region
            <in_class>E53 Place</in_class>
        </P89F_falls_within>
    </P55F_has_current_location>
    <P2F_has_type>cube
        <in_class>E55 Type</in_class>
    </P2F_has_type>
    <P70F_is_documented_in>/Hayton/items/Hayton011.jpg
        <in_class>E31 Document</in_class>
    </P70F_is_documented_in>
</CRM_Entity>
```

The author is still not sure that the syntax is perfectly correct: several examples of files or parsers found on the CIDOC-CRM web site were inconsistent. For example, another syntax used in some examples looks like:

```
<is_identified_by>AerialPicture
     <in_class>E42: Object Identifier</in_class>
</is_identified_by>
```

The numerical references of the properties do not appear, the names of the entities can contain space characters and the entity references are followed by a

colon. These are obviously simple syntax differences that do not affect the global XML structure, but they can be problematic with some parsers.

5.3.3 Results

The XML files have been validated by a tool that converts the XML 'Conceptual Reference Model' files into RDF format by H.G. Humet. This small program, based on the formal CIDOC-CRM specifications (`CIDOC_v3.4.dtd`), can also act as a code validator[3]. It was assumed that this validator was more reliable than the contradictory examples found on the web site.

The XML-CRM file is about four to five times larger than the traditional finds dictionary. Moreover, it is not easily readable and understandable. The quickest way to add an object remains to copy, paste and modify another object in the collection. However, if a collection of items has already been created and stored using the CRM, it will be ready to use with minor or even without any changes. It remains difficult to evaluate the quality of the implementation since no appropriate external CRM file was available.

5.4 Graphical user interface for configuration files

In order to easily manipulate the numerous configuration files involved in this project, a Graphical User Interface (GUI) was developed.

5.4.1 Aims and position in the project

Configuration files for the virtual environment were numerous and are quick to set up and modify for a trained user. However, these operations may be quite difficult

[3]Although this program was built two years ago, a slight mistake that prevents the parsing of large files was detected and corrected by the author of this research. This leads one more time to wonder about the consistency of the XML implementations of the CIDOC-CRM.

for non-expert users such as archaeologists. That is why a Graphical User Interface, referred to as GUI, has been developed. It allows the user to tune the virtual environment configuration within a friendly and standard WIMP (Windows Icon Menu Pointer) environment. The Qt (Trolltech 2004*b*) or WxWidgets (formerly WxWindows) toolkits could have been chosen for this purpose, as both of them are powerful, easy to use and cross-platform. Qt has been retained simply due to the author knowledge of this toolkit.

Is is quite important to note that this configuration should not be processed during the virtual environment run time, but before. The Figure 5.6 demonstrates how this configuration interface takes place in the concept of GIS previously described within Figure 2.12.

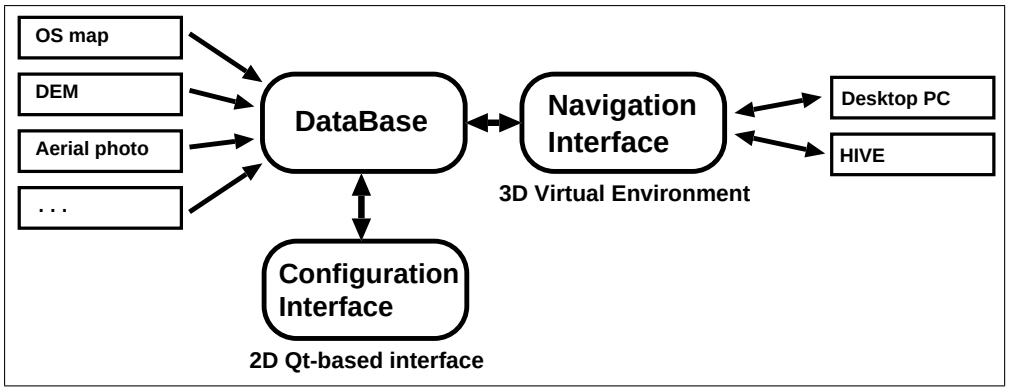


Figure 5.6: GIS configuration interface

5.4.2 Description

The main root file (generally '`main.ini`') is detailed in a large main window that presents all parameters as shown within Figure 5.7[4]. This allows the user to open, modify and save any main configuration file. Some secondary files like the archaeological item collection and the DEM files are editable from this interface in dedicated

[4]Screen captures of Figures 5.7, 5.8 and 5.9 have been generated with GNU/Linux + KDE3.2 which style is similar to MS Windows XP. MS Windows 2000 version and below may be less beautiful.

children windows.

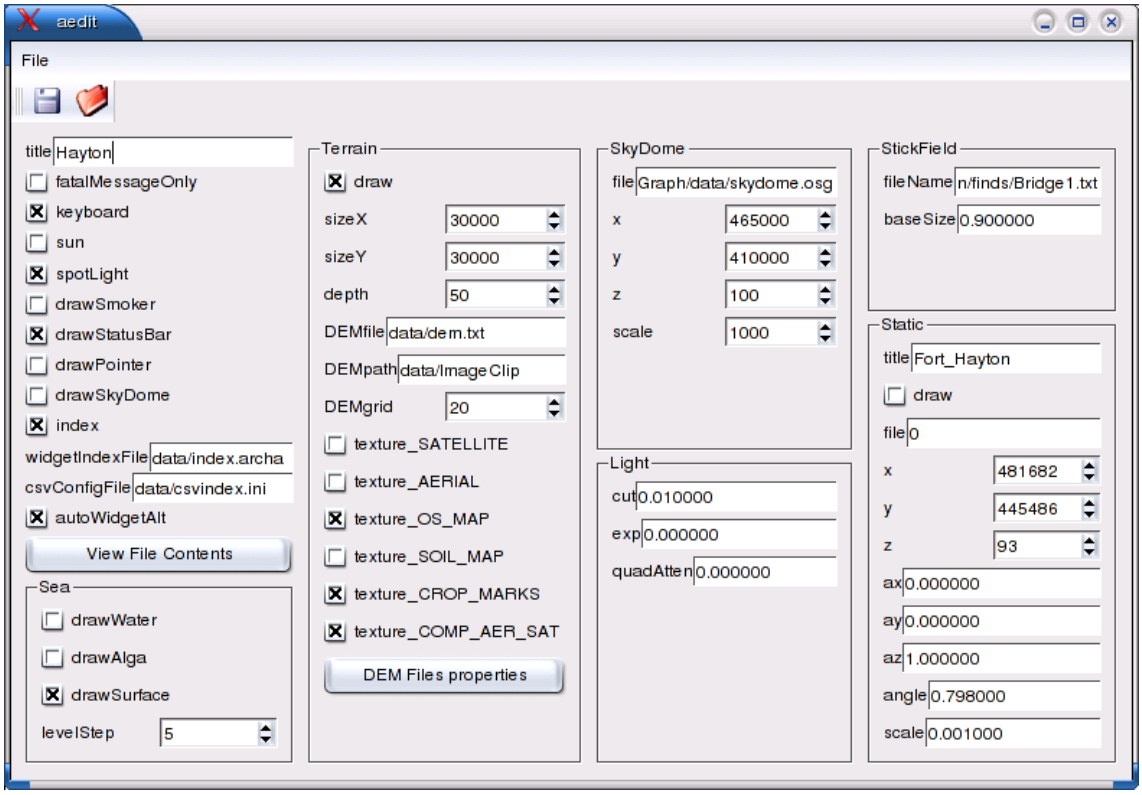


Figure 5.7: GUI: main configuration window

Figure 5.8 presents the list that allows the user to manage the item collection files (typically '`region.archa`'). The user can easily modify, remove or add new items. This last function can be operated via a drag and drop. For example, the archaeologist can browse his picture folders with the MS Windows file explorer, select the relevant ones with the mouse and drop them directly into the interface. The only manual operation is the simple specification of the location of these pictures in the three-dimensional virtual environment.

The DEM file and texture path for the terrain are visualised without available edition. Figure 5.9 illustrates the tool that shows the coverage of each kind of texture in the specified path for the current DEM file.

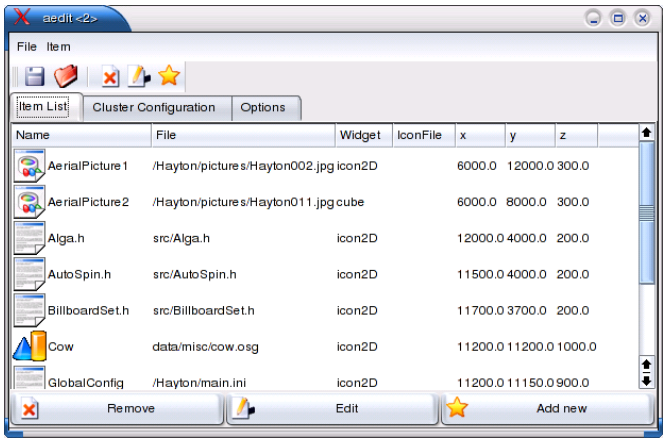


Figure 5.8: GUI: archaeological items window

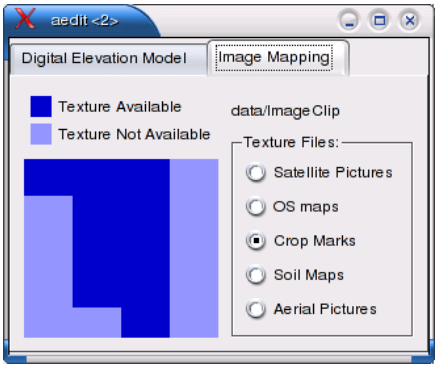


Figure 5.9: GUI: texture availability (coverage) window

Chapter 6

Visualization

6.1 Introduction

This chapter describes most of the visualization tools that have been developed. Note that many of these tools encompass also a navigation component. Therefore some of them will be detailed in the navigation chapter.

First described is the terrain representation, as this is the basis of the scene. Following this, the enhanced crop marks and their classification is described. An important part of this chapter is devoted to the dynamic widgets and their clustering. Then detail is given of the representation and manipulation of short texts and images using a rotational billboard set. Finally the stick fields, aimed at representing numerical databases is detailed, before concluding with a summary of the data integration in the environment.

6.2 Terrain and GIS

As stated in the background chapter, the GIS (Geographic Information System) are particularly useful to cope with large amounts of geographic information. “GIS and related technology help greatly in the management and analysis of large volumes

of geographically referenced data, allowing for better understanding of terrestrial processes" (Fabrizio 2001). Therefore it was logically chosen to base the project on such a GIS. The terrain representation and the navigation metaphors are then to stand for the back bone of the application.

The implementation of the GIS representation is based on a Digital Elevation Model (DEM). A DEM is a simple grid of elevations (altitudes). The terrain, represented as a three-dimensional surface, can be draped with various kinds of pictures such as satellite or aerial photographs, Ordnance Survey (OS) maps, crop marks or soil maps, as illustrated within Figure 6.1.

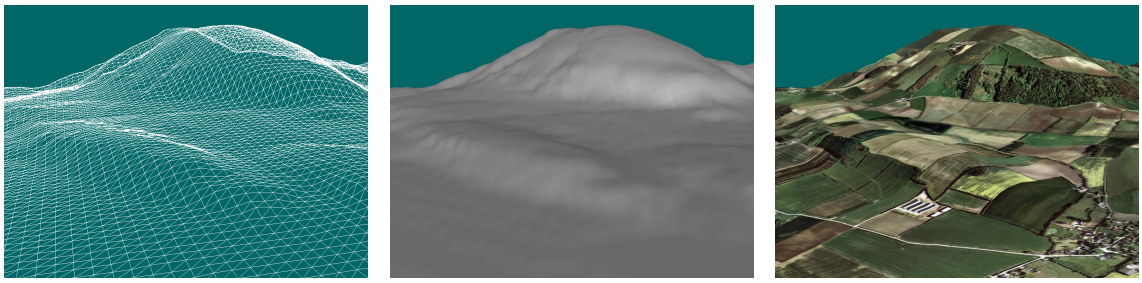


Figure 6.1: Simple DEM, 3D surface and texture mapping

Although it could be possible to load as many textures as available[1], it was decided to limit this number to two only: it was noticed than blending more textures leads to confusion. Moreover, a texture can be defined as predominant on others. This is useful to choice automatically the higher quality texture available. However, it was possible to change the maps during the execution of the application. When two textures are applied at the same time, they are blended in order for both to be visualised. For example the roads of an OS map can be superimposed upon an aerial picture as shown in Figure 6.2.

Moreover, a vertical exaggeration has been used. This means that the vertical scale of the terrain is multiplied by a factor $f_{vertical}$ compared to the horizontal one in order to enhance the visualization of the terrain relief. A typical factor

[1]This number is only limited by the video card.

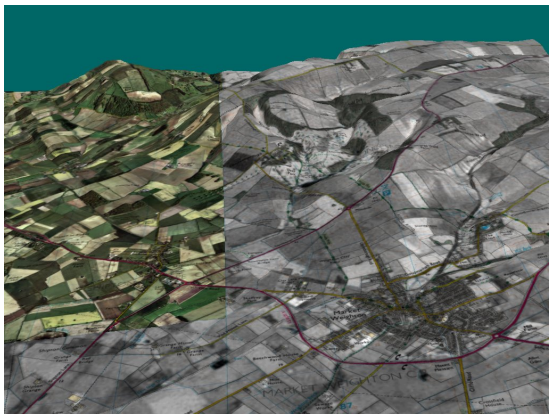


Figure 6.2: Blend of OS map and aerial/satellite photographs

Figure 6.3: OS map and variable sea level (dramatically raised up here)

($f_{vertical} = 5$) used for both cardboard relief maps and virtual environments has been chosen (Jingtong, Donghua, Young, Berry, Keeler, Willan & Sechrist 1996).

A variable sea level masking the underwater terrain has also been integrated as shown in Figure 6.3. This enables the simulation of variation of sea level through time. However, the one-metre accuracy of the current DEM is not really enough to provide a fine result.

6.3 Enhanced crop marks

6.3.1 Raising to the third dimension

The crop marks can easily be integrated in the application as simple images draped on the terrain. However, given the crucial meaning of these marks, an enhancement of this representation has been developed. This consists of raising the crop marks to the third dimension. For this purpose, a vectorial definition of the crop marks was required. A basic interpreter for text CGM format has been developed for this.

The user should be careful with this functionality. Even though the crop marks are represented as a kind of small wall, many of them are actually due to ditches.

Overall, the height or depth of a current crop mark is not representative at all of its ancient dimension. That can mislead the user, so the height of this enhanced crop marks should not be set too large (no more that one metre). According to some archaeologists used to deal with simple pictures, a learning stage will be necessary to take advantage of this metaphor. Figures 6.4 and 6.5 demonstrate the enhancement of the crop marks.

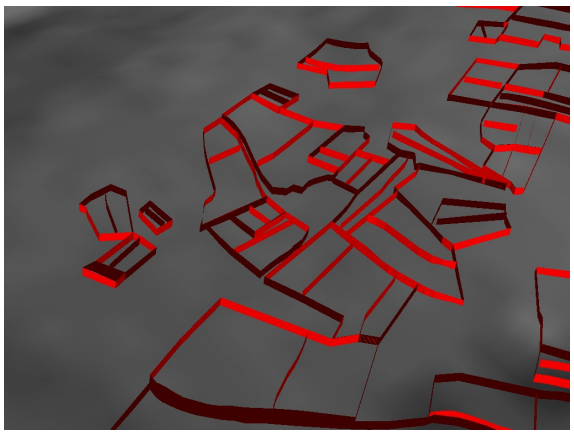


Figure 6.4: Enhanced crop marks (interpreted by P. Halkon)

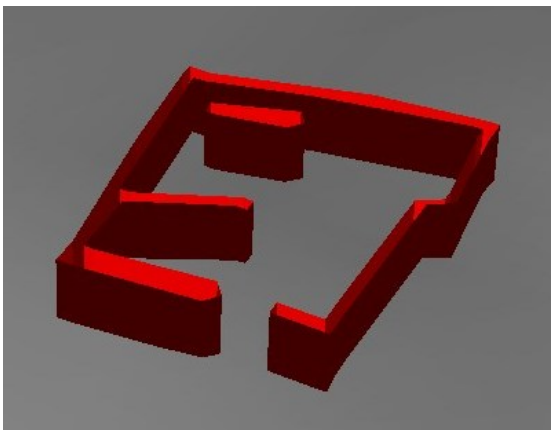


Figure 6.5: Enhanced crop marks (original by C. Stoertz)

6.3.2 Automated classification

An automated morphological classification of the crop marks according to (Stoertz 1997) has been developed. It aims to identify the main characteristics of the shape in a raw file of crop marks. Two steps are necessary to reach this objective: first the crop marks need to be clustered into small entities and secondly, each of these entities have to be classified. As these processes are time consuming[2], they are performed off-line (and only once), as shown within Figure 6.6.

[2]On an Intel Pentium IIII 2.8GHz - 512MB RAM, the execution of the clustering algorithm on an area of about $9km^2$ requires up to 2.5 minutes.

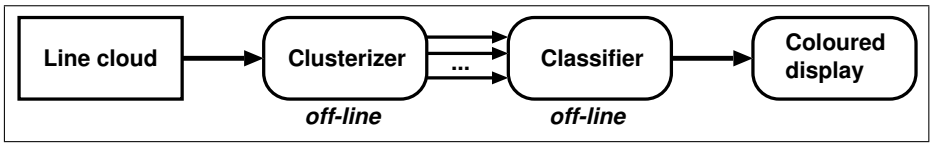


Figure 6.6: Clustering and classification of the crop marks

Clusterizer

A simple clusterizer has been developed. It consists of gathering all the segments that intersect together into an entity. Given the amount of information available, this method is really CPU-consuming. It fails to gather crop marks entities that can be composed of two separated parts. This is not important in the case of a double-ditched crop mark: both parts will be similar and then fairly classified. But in the typical case of a house 'cropped' by its gates, the different parts can be classified in various ways, or in a way that is not the global one (for example, two half-circles could be independently classified as 'edgy', whereas the global entity is circular).

Classifier

Typical classifiers based on neural network could have been used for this pattern recognition. As the features of the crop marks are clearly described, it was not necessary to rely on such heavy algorithms. Therefore, simple mathematic methods were used.

The classifier is based on Oriented Bounding Boxes (OBB). An OBB is a rectangle that contains the object to study, which aims to match as close as possible this item, as shown within Figure 6.7. The computation of such an OBB is based on the covariance matrix of the cloud of points as detailed in (Lahanas, Kemmerer, Milickovic, Karouzakis, Baltas & Zamboglou 2000): given a cloud of N points $\{p_i\}$, its centre can be computed: $c = \frac{1}{N}\sum_{i=1}^{N} p_i$. The covariance matrix C of this cloud is then given by:

$$C = \frac{1}{N} \Sigma_{i=1}^{N} (p_i - c)(p_i - c)^T$$

Computing the eigenvectors of this matrix provides an optimal base for this cloud of points. If the multiplicity of the eigenvalue is two, only a single eigenvector can be computed. This means that the OBB is flat. The orientation of the eigenvector base is the orientation of the optimal OBB.

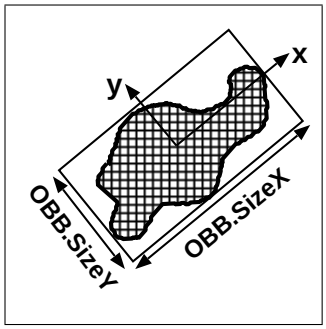


Figure 6.7: An oriented bounding box (OBB)

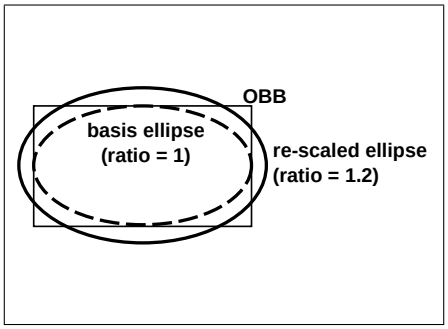


Figure 6.8: Ellipse used for the rounded test

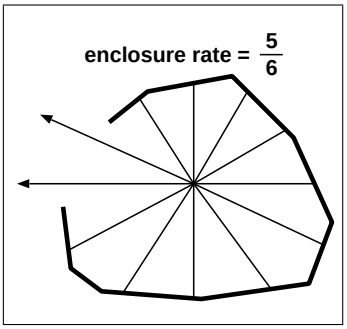


Figure 6.9: Rays emitted for the enclosure test

Hereafter, $OBB.SizeX$ and $OBB.SizeY$ represent the sizes of the Oriented Bounding Box along the x and y axis respectively. $OBB.MinSize$ and $OBB.MaxSize$ are the minimum and the maximum of $\{OBB.SizeX, OBB.SizeY\}$.

Using this OBB, the ratio between $OBB.MaxSize$ and $OBB.MinSize$ can be computed easily in order to differentiate the regular objects from the elongated ones. An object is considered as elongated if this ratio is greater than 1.25 (Stoertz 1997):

$$\frac{OBB.MaxSize}{OBB.MinSize} > 1.25$$

In order to evaluate if a crop mark is rounded or not, an ellipse attached to the OBB is used. The radiuses of this ellipse are the width and the height of the bounding box, multiplied by a ratio greater than one as demonstrated in Figure 6.8. A point $p(x, y)$ (in the OBB coordinate system) is included in the ellipse if:

$$\left(\frac{x}{OBB.SizeX/2}\right)^2 + \left(\frac{y}{OBB.SizeY/2}\right)^2 \leq ratio^2 \mid ratio \geq 1$$

Computing the rate of crop marks included in this ellipse aims to distinguish between curvilinear objects and rectilinear ones.

A crop mark archaeologically 'closed' is not necessarily mathematically closed. For example, a gate in a house opens the associated crop mark, which remains however considered as an enclosure. The method used to detect an enclosure consists of sending rays from the center and of computing the intersections with the crop mark, as shown within Figure 6.9. The rate of rays that do not collide is the opening rate.

The whole process is detailed in Figure 6.10. Note that this algorithm contains some shortcuts to hasten the process which is a bit long. For example, if the ratio $\frac{OBB.MaxSize}{OBB.MinSize}$ is greater than a certain threshold, there is no need to apply the enclosure tests: it is obvious that this shape will be linear.

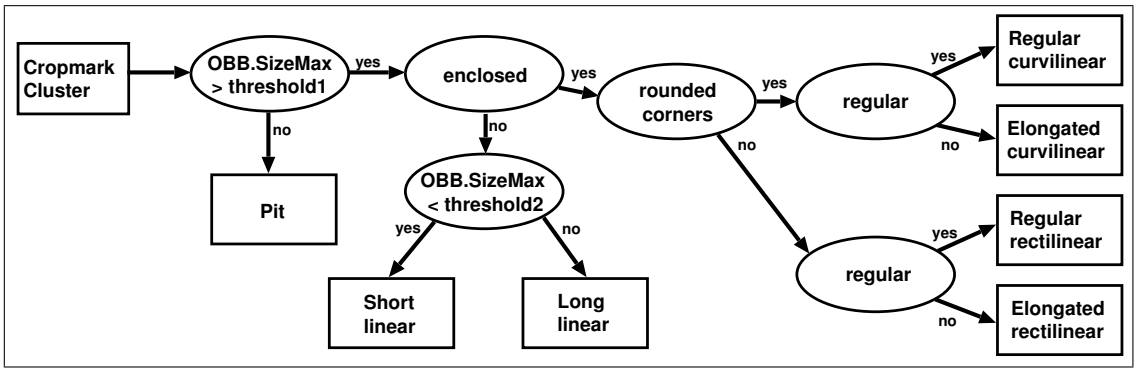


Figure 6.10: Classification process

As one can notice, the classification process does not distinguish between individual, clustered and aligned pits. There is no test to conclude that a shape is 'irregular curvilinear' neither. This class of crop marks will then fall into the 'regular curvilinear' and 'elongated curvilinear' ones.

Results

Several factors affected the recognition of the shapes; the most important one is the precision of the file. The vectors have an accuracy of only one metre. Considering small shapes of this size, this is clearly very difficult to distinguish between a square and a circle. Moreover, the large majority of the pits (point-shaped crop marks) are not represented with such a precision. Another important issue is the evaluation of all the thresholds that have been used to distinguish between classes (ellipse size, enclosure rate, etc). (Stoertz 1997) only provides the rounded ratio, all the others are approximative.

However, after some tuning of the parameters, the simple crop marks were fairly well classified and represented using various colours in the main application, as shown in Figure 6.11. It is quite difficult to provide some quantitative results, because even for the human eye it is often not obvious to classify the crop marks.

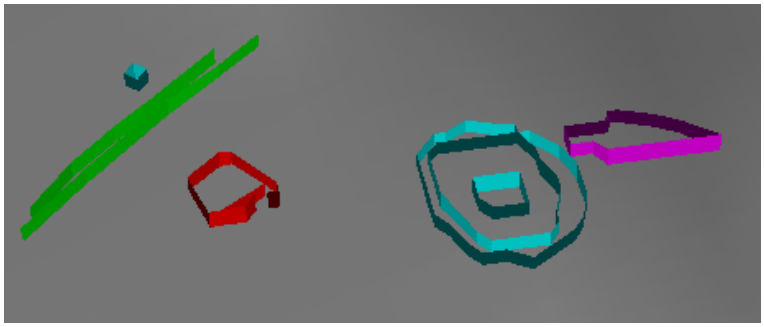


Figure 6.11: Classification result

6.4 Dynamic widgets

6.4.1 Concept

"We define a widget as an encapsulation of geometry and behaviour used to control or display information about application objects." (Conner, Snibbe, Herndon, Rob-

bins, Zeleznik & van Dam 1992). For example, the traditional widgets in a graphical two-dimensional interface are Windows, Icons, Menus and Pointing, referred to as WIMP (Beaudouin-Lafon 2000).

Widgets can have three aims in this virtual environment: represent a (numerical) value, symbolize an action or gather some children widgets. They have to be designed very carefully in order to associate an intuitive representation to their meaning (Bertin 1967).

6.4.2 Genesis

Dynamic widgets representing the content of a file can be created automatically simply by referencing this file and its location in the scene in an index. An example of such an index file is provided here:

```
### Picture of Hayton landscape
Object LandscapePicture
    title    Landscape_Picture_of_Hayton
    widget   cube
    x        6000.0
    y        2000.0
    file     /Hayton/pictures/Hayton002.jpg
EndObject
```

Note that the elevation is not specified, as a result the widget will automatically be positioned on the surface of the terrain. The '`file`' will be the content (static part) of the widget. This is the minimal necessary information required to create a widget. Optionally, the type of representation (size, shape, texture) and the action when clicked or passed over can also be specified.

The widgets can obviously also be stored using the CIDOC Conceptual Reference Model previously described.

6.4.3 Representation

Representing all the images, texts and three-dimensional objects directly on the terrain leads to an overload of information. Therefore, the user is easily lost. Pickable widgets avoid such an overload by representing a symbol instead of full information. More details can be queried by selecting the widget with the mouse pointer or the gamepad.

Two symbols for the pickable widgets representation have been implemented: a two-dimensional icon that always faces the user (for the common finds) and a spinning cube (for more important finds and widget clusters). Both can be textured by any image. There is no automated process to distinguish between a valuable find and a average one, and therefore the user is responsible for choosing between icon and cube. In a standardization purpose, the widget is draped by default with the KDE 3.2 standard MIME-type icon (Crystal series). As they are at the same time homogeneous and quite intuitive, archaeologists using the application should not be lost with too many new symbols.

Three widgets are demonstrated in Figure 6.12. In the foreground, two icons wrapping a text (a configuration file and a C++ source), an icon delivering a three-dimensional Roman helmet and a spinning cube containing a poppable picture. In the background, a large spinning cube clustering children widgets (method detailed in the next section). It is important to notice that the style of the widgets (edgy, highly coloured) strongly contrasts with the terrain. This should help to notice the finds.

When these widgets are selected (by mouse clicking or gamepad picking), the contained information pops up. In the case of text (ASCII) or images files, a billboard is used (see next section). In the case of a three-dimensional object, the item is popped up and spins automatically.

The names of the widgets are automatically displayed on the screen when the user

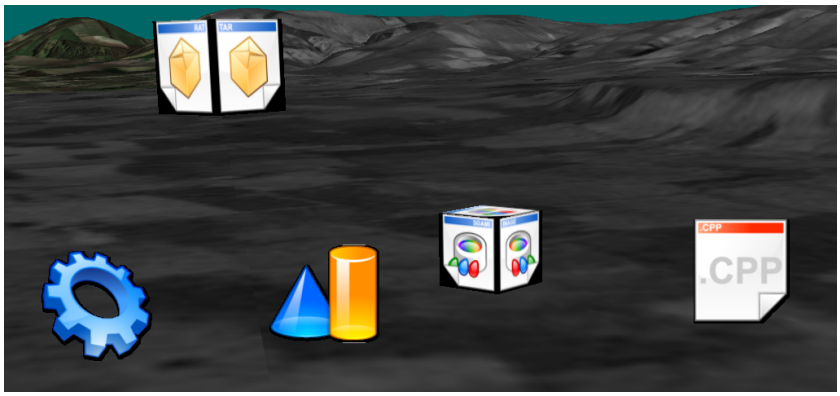


Figure 6.12: Widgets examples

moves the mouse pointer upon the widget representation (this feature is generally called 'hint' in many widgets managers terminology).

6.4.4 Widget clustering

The clustering of the widgets allows the application to display only one widget instead of a group of children widgets. Generally, the cluster widget is displayed when the user navigates far away from the group. The detailed display of every widget is reserved for closer points of view. The transition between these two representations happens automatically while the user moves.

The clusters can be automatically generated. The user has only to provide the maximal radius of a cluster. A basic spanning tree algorithm has been implemented for this purpose. "This hierarchical method starts by considering each component of the population to be a cluster. Next, the two clusters with the minimum distance between them are fused to form a single cluster. This process is repeated until all components are grouped into the final required number of clusters" (Center for the New Engineer 1995). The ending condition of this algorithm is the number of the clusters. However, the size of the cluster widget provides a clearer indication as the number of clusters is abstracted data. The size was then preferred for the implementation.

An example of a user clustering configuration is provided hereafter [3]. '`minDistance`' is the size of the clusters for the ending condition.

```
Object ClusterConfig
    clustering      auto
    minDistance     17.0
    viewDistance    500.0
EndObject
```

The optional '`viewDistance`' field allows to specify the limit distance between the representation of the cluster widget and those of each of its children.

However, this solution has some weaknesses, in particular in the case of diffuse objects, as shown within Figure 6.13. In the third set of points, the human eye cannot distinguish clearly between the groups of clusters. These points should easily be classified into 4, 5 or 6 groups.

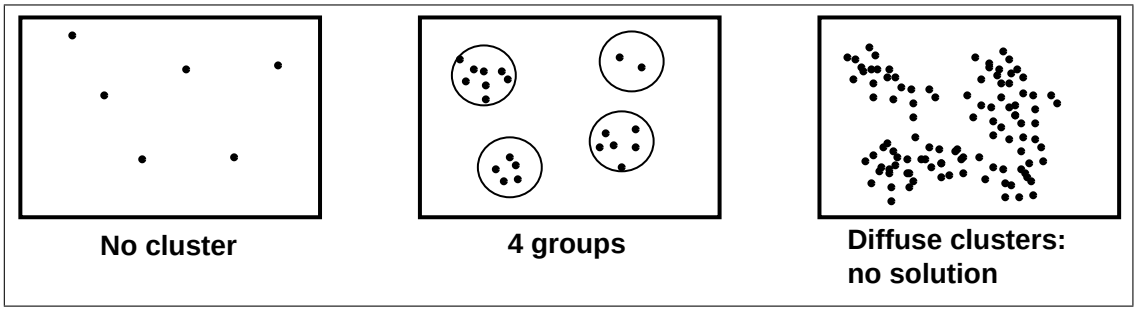


Figure 6.13: Diffuse clustering problem

6.5 Billboards

In order to represent and manipulate two-dimensional information such as short texts and images, the billboard was developed.

[3] A slight technicality appears while using the CIDOC Conceptual Reference Model: it is impossible to store the parameters of the clustering algorithm in the model. A separated and specific file must then be used.

6.5.1 Concept

A billboard is a two-dimensional container encapsulating text or images. It can be attached to another object, collected and manipulated. Even though the billboards are flat, it is important to understand that they do not consist of a simple projection on the screen: they are 'real' objects, part of the three-dimensional environment. In this project, the billboards have been designed to face the user at any time. This allows the user to move freely inside the environment without taking into account the orientation of this class of objects: they are always 'ready-to-be-seen'.

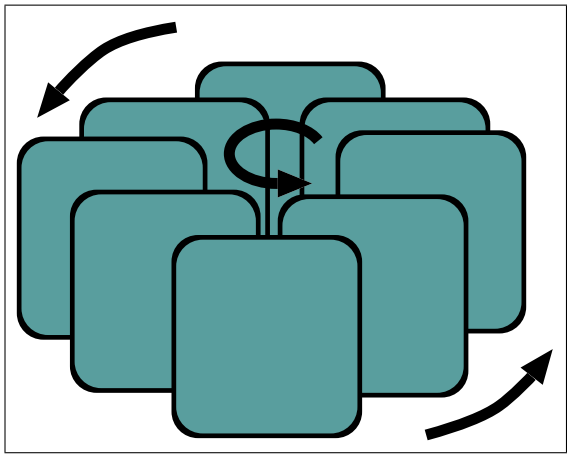


Figure 6.14: BillboardSet – principle

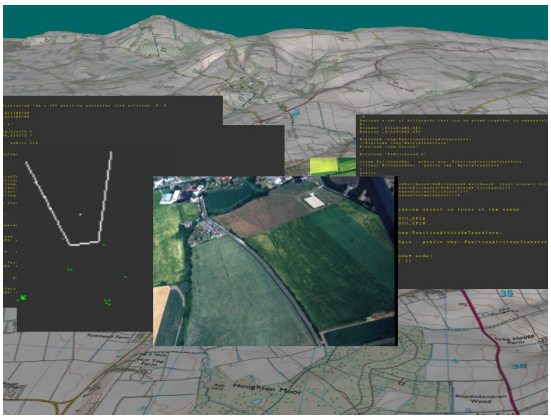


Figure 6.15: BillboardSet – application

6.5.2 Gathering and manipulation

The billboard has been considered as the standard way of representation and interaction with an image or a text file in the interface. When a widget representing one of these objects is selected, the associated billboard pops up smoothly, and remains attached to the widget. Then the user may decide to collect it into his billboard set.

A billboard set is a tool allowing the user to collect and gather numerous billboards. A rotational set has been implemented. This can be considered as the first level of a cone tree (Cockburn & McKenzie 2000). All the billboards face the user,

grouped in a circle whose axis can be changed. The smooth rotation of the billboards leads to a three-dimensional slide show. Other operations like billboard removal are also available. Figures 6.14 and 6.15 demonstrate respectively the principle and the application of a rotational billboard set.

6.6 Static widgets

The static widgets cannot be selected or clicked on. Two types of static widgets have been implemented.

6.6.1 Three-dimensional model

The first one, very basic, simply displays a three-dimensional model in the scene. The model itself can be very complex, like the representation of the Roman Fort of Hayton modeled by an archaeologist-designer. The user must simply specify the main file containing the model, its location and orientation and a scale ratio if necessary.

6.6.2 Stick field

The second one, slightly more complex, is based on sticks "planted" vertically on the terrain. They are useful to represent multi-dimensional data, like databases or arrays of numerical values. Such a database can contain for example the density and the number of roman pottery, iron age glass and medieval tiles finds per metre-square. An real example of the raw data is given within the Table 6.1.

A value (density, number, etc) can be represented by planting a stick at the studied site. The visible area of the stick has been chosen as proportional to the represented value. All dimensions (and not only the height) of the stick have been scaled according to represented value. Therefore the user can interpret the three-

cell	xcoord	ycoord	Roman pottery	Greyware	Gritted types	Dales ware	Huntcliff
A1	482244	445496	19	12	5	0	5
B1	482251	445503	7	2	2	1	1
C1	482257	445511	25	21	4	0	4
D1	482264	445518	16	8	1	1	0

Table 6.1: Database example

dimensional data visualization from all viewpoints: from a bird's eye point of view as well as a human walking on the ground. Considering a large area, a field of sticks is then built, as represented in Figure 6.16.

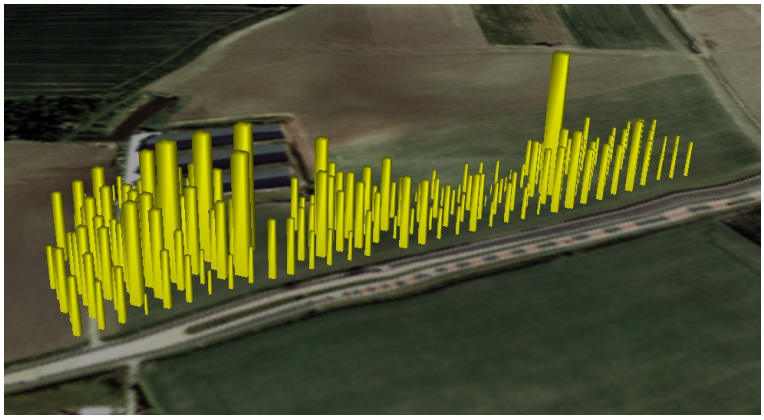


Figure 6.16: Stick field

In the case of multi-dimensional values, there are several values to represent at the same site. One can then choose between two options: visualize only one value at the same time and letting the user swap between values or stacking the sticks representing different values by using different colours. Note that the latter metaphor is not appropriate for more than four values as the blend of colors becomes confusing.

Other models have been implemented, tested but judged not efficient enough and abandoned. For example, another method using sticks consisted on planting several stick fields at the same time, with a little shift and using different colours. But from

some points of view, it was impossible to see all the colours at once, especially if one stick field is globally smaller than another. In another method, the values were represented using a colour scale map. In the case of multidimensional databases, several maps have been superimposed, using half-transparency. But most of the information was not visible, and it interferes with the texture of the terrain.

6.7 Case study: Hayton region (the Foulness Valley)

Thanks to archaeologists for the raw data, a virtual environment around the region of Hayton ('The Foulness Valley', also called 'The First Iron Masters Valley'), East Yorkshire, UK, was built (Halkon & Millett 1999). The accuracy of the Digital Elevation Model (DEM) was 1 metre. The total size of the terrain was 20km by 24km, with an elevation range of [0..163m]. It consists of a plain, some hills and a small part of the Humber River estuary (which can be considered as the sea).

The whole model could be draped with low resolution monochromatic satellite pictures and colour Ordnance Survey (OS) maps. Half of the terrain could be covered by red crop marks provided by archaeologists and about a quarter by high quality colour aerial photographs. Some crop marks were also available in a vectorial format and could be enhanced. Also added into the environment were two different databases of finds with their properties, various stand-alone pictures or shorts texts and a reconstruction of the Hayton Roman Fort in three dimensions designed by an archaeologist (M. Faulkner) and partially represented in the Figure 6.17. The integration of this Fort remained incomplete because of some exporting issues from 3ds Max: some textures could not be extracted properly (they appear white on the figure).

The location around the Fort was really rich in information and includes the

Figure 6.17: Hayton Roman Fort partial integration

visualization of the red crop marks fitted to real crop marks within the aerial photographs. Moreover, these crop marks are aligned with the Hayton Fort three-dimensional reconstruction.

6.8 Supported data summary

The data file types presented in the Table 6.2 are currently supported, with their integration modalities and conditions of use. The coverage is quite large, encompassing almost all the kinds of information currently generated and used by archaeologists. A noticeable exception is the integration of video.

The format of the databases and tables remains the main pending issue. Almost each file that was provided was unique. Moreover some pieces of tuning for their integration had to be processed manually. An effort in the normalization of this kind of data to improve the ease and efficiency of their integration has been done by using the CIDOC 'Conceptual Reference Model', but it does not support all the item types at the moment.

Type	File format	Integration	Conditions
DEM	ASCII	terrain model	
Image	bmp	terrain texture	grid of squares
Vectorial Image	cgm	enhanced crop marks	
CIDOC CRM	XML	link the content (files, notes)	E19 'Physical Object' entity, spatial coordinates
Text	ASCII	icon/cube + billboard	should not be more than 20 lines
Image	jpeg, bmp, png, ...	icon/cube + billboard	
3D model	3ds, VRML, ...	icon/cube + auto-spin	
3D model	3ds, VRML, ...	static in the scene	
Arrays of finds	CSV*	fields of sticks (eventually multicolour)	use template
Database of finds	CSV*	fields of clustered icons	use template

* *Comma Separated Values.* (Can be exported from Arcview, Microsoft Excel, ...)

Table 6.2: Supported file formats

Chapter 7

Navigation

7.1 Introduction

Two global metaphors for the navigation in the virtual environment have been developed. The first one is a traditional virtual environment. The user can navigate using both keyboard and mouse but also a wireless gamepad, optionally with the finger-bend tracked glove. This first mode supports the stereo 3D, but this is only an option that can be unactivated. It can run on almost any desktop PC and on the HIVE wall screen as well. The second mode is called 'map table'[1].

This chapter first describes these two metaphors. Secondly, the usage of the P5 finger-bend tracked glove is presented. Then some low-level navigation tools like the status bar and the mini-map are demonstrated. Finally, some more high-level tools such as the density map and the density compass are detailed before concluding with a summary of the navigation devices and modules integration.

[1] 'Map table' refers to the table used in the ships to read and manipulate the maps.

7.2 Standard navigation

The standard OSG navigation metaphors (object manipulator, drive and flight simulator) are used as the traditional method of interaction with the scene by the way of the mouse and the keyboard. However, this method of navigation is not really intuitive, especially for beginners. Even used by trained people, these metaphors generate jerky movements.

The wireless gamepad (Logitech WingMan Cordless) has been integrated in the interface in order to provide a more intuitive and smooth method of navigation. This has the advantage of letting the user move freely into the room instead of remaining stuck on the desktop. The 'flight mode' (steering gaze) has been implemented: one of the mini-joystick stands for the direction of the gaze (two axis) and the other for its power (one axis only). The main functions of the interface like picking objects, displaying, moving or masking the rotational billboard set, are accessed directly from the gamepad buttons. The slider of the gamepad has been assigned to a sensitivity value, that allows the user to choose at any time the velocity and precision of the movement commands of the gamepad. This is quite useful (and even necessary) in this kind of environment: the user may want to move either with a velocity of 1km/s while scanning quickly the landscape but 1m/s inside the Hayton Roman Fort reconstruction.

While using the stereo 3D with the standard mode, the terrain always appears behind the screen to avoid any side effects if the representation crosses the border of the screen, as shown within the Figure 7.1.

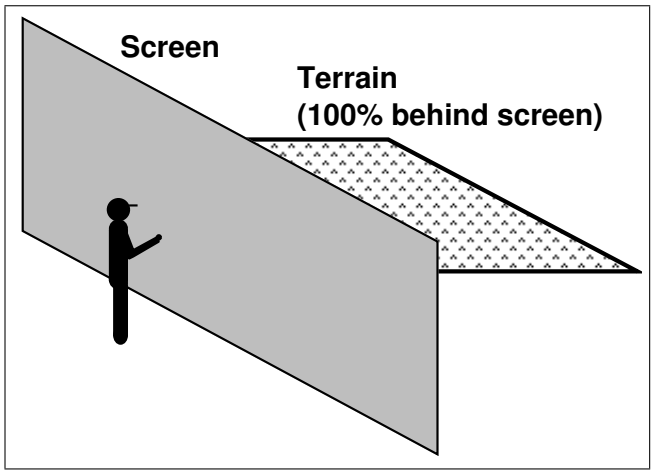


Figure 7.1: Standard navigation: the screen is a window

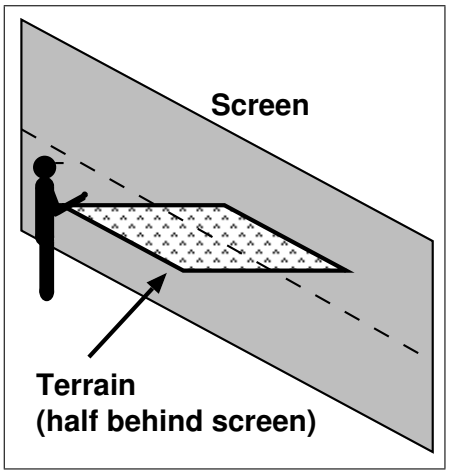


Figure 7.2: Map table paradigm: the terrain seems to be horizontal

7.3 Immersive navigation

7.3.1 The map table paradigm

The 'map table' display method aims to represent the terrain as if it was a map put on a table, as demonstrated within the Figure 7.2. This map should appear half behind and half in front of the screen. The user must be able to move around the table and to feel that he could touch the map. This map must then seem to remain stationary in the user's point of view.

As described in the background chapter (p. 27), the application must then apply changes to the image that is displayed according to the user's point of view, in order to let him believe that the map remains at the same place.

7.3.2 Head tracking system

Before proceeding further, it is important to note that trying to simulate something physical by using a projection on flat screen is not obvious, as the human perception considers many cues to determine if something is real or not. As a matter

of fact, coupling physical and virtual environments involves several modules, each one enhancing a particular aspect of the perception of the virtual environment. The number and the quality of this set of modules affect the global efficiency of the environment. There is no way to classify virtual environments into two simple classes 'immersive' and 'non-immersive', a virtual environment is always more or less immersive.

Global movement

The global movement consists of moving the virtual camera in the virtual environment as if the user were in this environment. However, it is not as simple as that, because the user also moves in the real world, relatively to the screen. For example, if the user walks backward from the screen, then simply moving the camera backward in the virtual environment is not valid. Because this will reduce the apparent size of the environment on the screen, but at the same time the user is already moving backward, and therefore there is no need to reduce the apparent size of the environment, this will be done by the user physical distance to the screen.

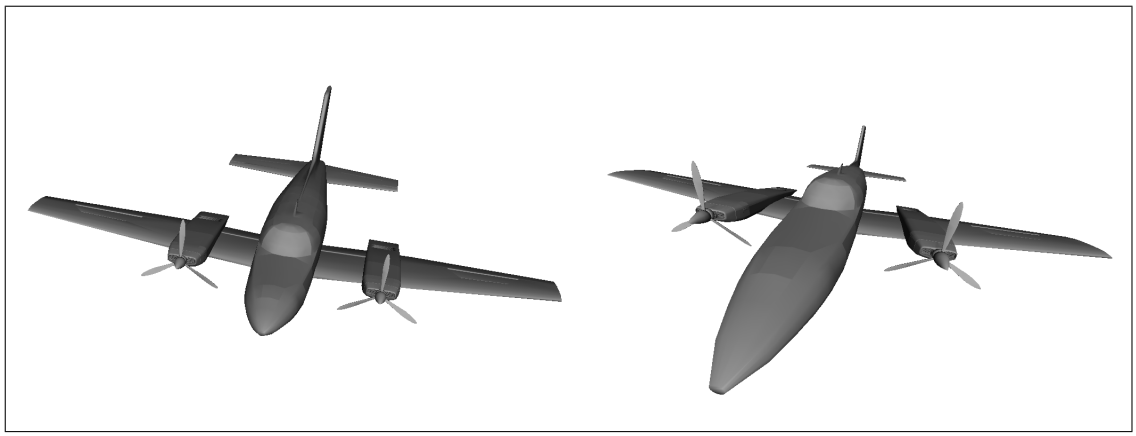


Figure 7.3: An airplane viewed from far (narrow angle) and near (wide angle)

However, looking at an object from near or far is not equivalent, even though it should have the same size on the screen. The difference is due to the proportion

of visible angle covered by the object, as demonstrated within the Figure 7.3. The wide angles create distortions, with the nearest parts of the object appearing as over-dimensioned.

The modeling of the visible angle in OpenGL (and wrapped in OpenSceneGraph) is the camera frustum, demonstrated within the Figure 7.4. When the user moves backward, the method consists of reducing the frustum opening to give the impression of a far object, and to move the camera backward to keep the apparent size.

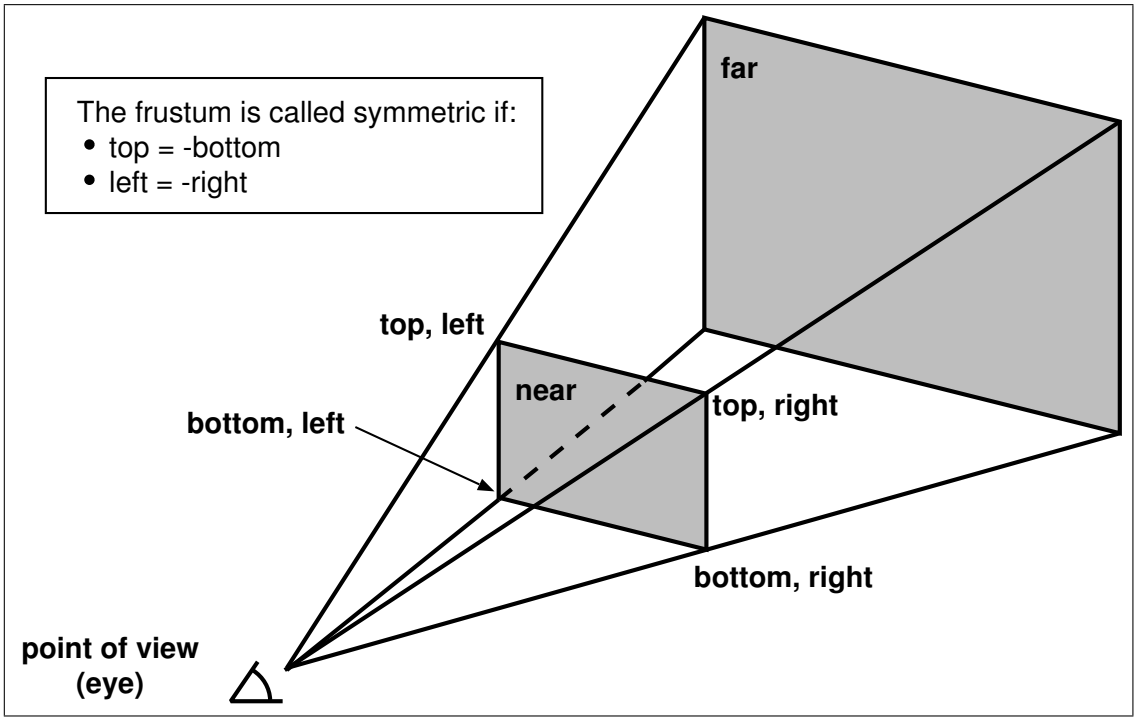


Figure 7.4: The concept of frustum view

The rotation of the camera around the environment can follow the user. But the angle to the screen implies an asymmetric frustum to compensate that the user does not face it directly. This method is exactly the same as that employed by Hans Holbein in his painting 'The Ambassadors' (Holbein 1533), where a skull appears only if the visitor stands beside the picture, as illustrated within Figure 7.5. The skull is on the bottom of the picture.

Figure 7.5: 'The Ambassadors' (Hans Holbein)

Orientation of the eyes

The OpenSceneGraph native stereo support takes into account the eye separation, the screen distance and the fusion point. It assumes then that the user watches at the screen keeping his head horizontal. Therefore the points of view for each eye are slightly shifted on the left and on the right. Within the framework of this project, the user stands up and is easily tempted to tilt his head. For example, if the user turns his head with an angle of 90 degrees, the left eye will be strictly above the right one; the points of view should be shifted vertically and not horizontally. A new callback for the OpenSceneGraph 'SceneView' class has been developed in this project to take into account the angle of the head around the line of sight (also called 'roll axis') during the calculation of the projection and view matrices of the stereo mode. The other angles are not considered, as it was assumed that the user always faces the screen.

7.3.3 Navigation using hand motion capture

The navigation using the finger-bend tracked glove involves two main ways of interaction: by using the fingers bend and the hand tracking system itself.

Finger bend glove

Using the P5 glove previously described, around fifteen postures were defined and they can be used like any key of the keyboard or any button of the gamepad: they can be linked to the virtual environment tasks using the linker file. It is not fair to speak about 'gesture recognition', as only static states or discrete changes between these states have been considered. The postures that have been considered are represented within the Figure 7.6.

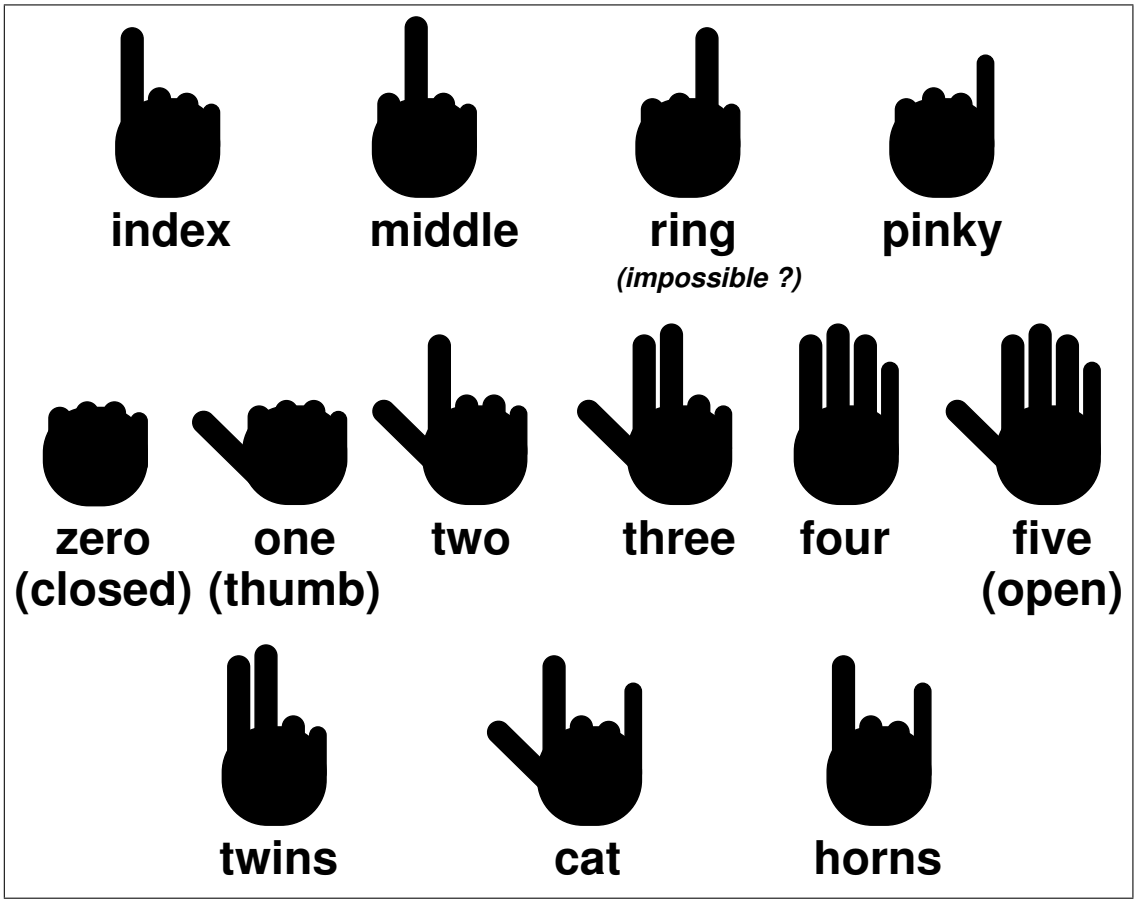


Figure 7.6: Main recognized hand postures

An important limitation of this glove is that it must be calibrated before use, and this calibration strongly depends on the morphology of the user's hand. Frequent changes of users are therefore not really convenient.

Moreover, linking too many of these postures to different tasks can create some troubles: the hand can take a stray recognized posture while moving from one to another. For example, it is unfortunately very easy to do the 'five' (open) posture while moving from 'four' to 'one' (thumb up).

The glove is comfortable to wear, even though the movements are a bit limited by the bend sensors. It remains difficult to completely bend the fingers because the sensors physically prevent this action. However, it is possible to completely close the hand: if all the fingers are bent at the same time, the body of the glove follows the fingers. This leads to the conclusion that if the sensors of each finger are independent, the movement of the fingers are linked, which limits the number of physical postures[2].

Hand tracking

The hand tracking was used mainly via the 'grab-and-move' metaphor, which consists of grabbing the terrain by closing the fist and then moving the hand horizontally, the terrain following it. When relaxing the hand, the terrain remains stationary. This metaphor is really intuitive and is commonly used by two-dimensional windows managers, for example to copy-paste by moving an icon with the mouse. Following a similar model, a zoom control was developed, using a 'grab-and-zoom' metaphor on the vertical axis.

Another simple but powerful and intuitive metaphor called 'show-to-pick' has been developed. It consists of picking with a virtual index. A small yellow stick representing the hand moves into the scene, reproducing the movements of the user's

[2]For example, the postures 'cat' or 'horns' are really difficult to produce with a P5 glove.

glove. When the user bends all fingers apart from the index, the object under the pointer is then selected (picked up).

The only weakness of this navigation system is the one-centimetre accuracy of the P5 glove. There is nothing apparent while moving at a fair speed, but the noise is disturbing when moving very slowly to reach a precise target. Moreover, this accuracy is even worst while standing far from the base, as it is the case while standing up in the HIVE with the base put on the floor. This problem has been partially fixed while porting the glove to the HIVE. The accurate Vicon tracking system has replaced the gloves' cameras (three markers have been bonded on the glove for this purpose). The work volume is also considerably larger, but the wire from the camera to the glove was still rather inconvenient.

Moreover some side effects on the stereo glasses were noticed: the image flickered. This effect is due to the infrared rays emitted by the glove LEDs that confuse the glasses infrared synchronization. These LEDs have been masked and the effect was reduced, but not eradicated.

Conclusion

An implementation using the finger-bend tracked glove has been successfully integrated in the application. The use of such a glove seems likely to be a really intuitive and powerful interaction device, in particular for the common navigation tasks like moving and picking objects. The low accuracy of the P5 glove has been fixed using the HIVE. However, a wireless glove should really be integrated to free the user of any constraints.

7.4 Status bar

The main navigation information is available via a status bar in the lower part of the screen. A high-contrast LCD (Liquid Crystal Display) representation has been

developed in order to display efficiently the numbers. The spatio-temporal references as the three GPS coordinates and the time spent in the environment are embedded in this bar. Simulation parameters like the number of objects currently collected in the billboard set and the applied sea level are also displayed. Moreover, a time-limited message can be sent and displayed by any metaphor within the scene. This is used for example to provide the name of a pickable widget while the mouse cursor is passing over it, or the label of a part of database represented by a stickfield.

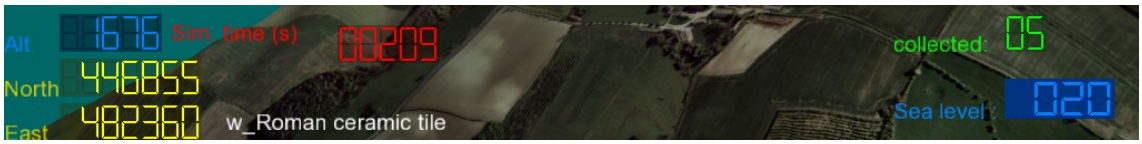


Figure 7.7: Status bar

As adding a background to this interface was too intrusive even with a semi-transparent colour, it was decided to let the counters float. However, as this status bar is still a simple two-dimensional projection, it breaks the effects of the stereo and should be turned off in this case. It was not used with the map table mode.

7.5 Mini map

As the terrain may be quite large, a map summary has been developed. "The "mini-map" is a simple graphic tool to provide a user orientation between the media and the virtual environment." (Shilling, Morgan, Mosbruger, Beilstein & Orichel 2003). The author chose to represent it as a two-dimensional terrain from bird's eye. All widgets are also represented as simple multicolour[3] points. Another important feature is the representation of the current displayed zone of the terrain by a bounding polygon. This mini map should hasten the research of objects in the scene and avoid the user getting lost. This tool is currently considered as a billboard and then inherits from

[3]The colour depends on the type of widget.

all their properties of collection and representation. This map is demonstrated in the previous chapter: Figure 6.15, page 84 in the billboard set, on the left.

Due to the projection onto a flat representation, this map is not perfectly accurate, especially with regard to horizontal points of view. However, it remains precise enough to reach its aim of basic localization. An improvement would be a better interaction with this map. This could consist of a 'select-and-go' tool that automatically moves the user to a selected position.

7.6 Navigation using the density of finds

7.6.1 Density map

In order to improve the perception of the finds from a significant distance, widget gathering and crop marks enhancement have been developed. However, this does not fit well with very large distances. Moreover, the usage of various improvement metaphors could be confusing. For this reason, a global density map has been developed. It unifies all the information about the finds (crop marks, stand alone finds) into a single global texture map that can be applied to the terrain.

For this purpose, a dilatation function was applied on these finds. It consists basically of expanding each point to a surface. Four functions describing the value of the density according to the distance to the find have been tested: linear, inverse, square inverse and gaussian, as shown within the table 7.1[4]. Both inverse and square inverse functions did not provide interesting results (due to their strong variations).

These dilatation functions are then applied to all the points representing a find: the crop mark textures, the isolated finds, etc. As the isolated finds consist of only one point whereas the crop marks are large sets of pixels, a hand-made balance

[4]Although the dilatation function is defined for a positive distance, the interval of the function plot has been considered as symmetrical around zero as it provides a better overview of the function behaviour.

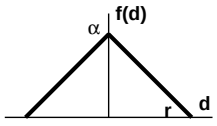
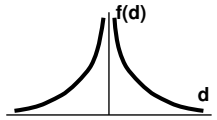
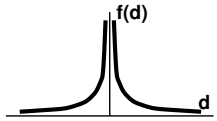
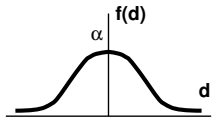
	Parameters	Interval	Function(d)	Plot
Linear	α, r	$d \in [0, r]$	$\alpha(1 - \frac{d}{r})$	
Inverse	r	$d \in \Re_*^+$	$\frac{r}{d}$	
Square inverse	r	$d \in \Re_*^+$	$\frac{r^2}{d^2}$	
Gaussian	α, r	$d \in \Re^+$	$\alpha e^{-(\frac{d}{r})^2}$	

Table 7.1: Dilatation functions

between the various types of finds was required (isolated points where considered as more intense than crop marks). The dilatation result is a valued map to be displayed in the main application. Several palettes of colours have been tested to represent this density:

- **Grayscale** is the simplest way to represent a range of value, but interferes in the lighting effects and the relief representation.
- **Red to blue** is the classical cold/hot metaphor, but not really impressive in this context.
- **Black/red/yellow** is probably the most impressive colour scale for two-dimensional representations. However, in a three-dimensional environment the black colour dissolves the relief perception.
- **Blue/red/yellow** is a compromise between the two previous ones. It allows at the same time an efficient representation and a fair render of the third dimension. It has been retained for this project and demonstrated within

Figure 7.8.

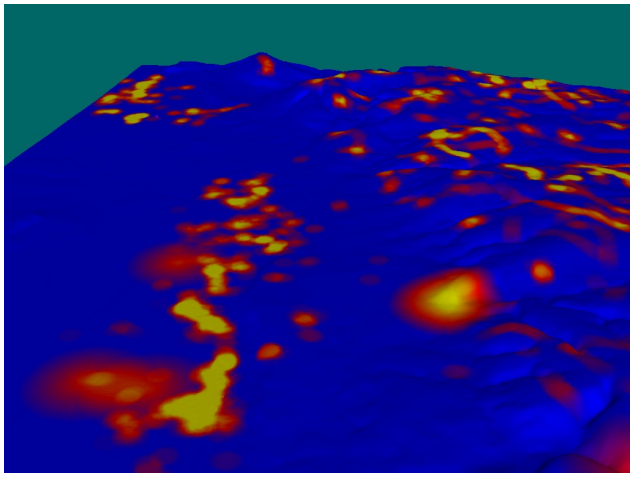


Figure 7.8: Density map

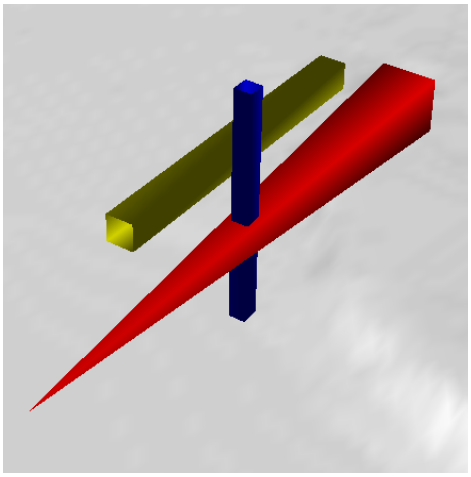


Figure 7.9: Navigation compass

7.6.2 Density compass

Visualising the density map provides the user a global view of the scene. However, in the case of important zoom factors (navigation close to the terrain), the contrast of colours is not really obvious. For this reason, a compass that indicates the nearest maximum of density has been developed.

Field of 'force'

As a traditional magnetic compass would behave, this device reacts to a field of force. The density of finds is considered as a potential, and the force then derives from this potential. The standard model of the gravitational and electrostatic fields was followed:

$$\mathbf{f}(\vec{\mathbf{x}}) = C \cdot \Sigma_{\vec{\mathbf{v}} \in \xi - \{\vec{\mathbf{x}}\}} \frac{\rho(\vec{\mathbf{v}})}{\|\vec{\mathbf{v}} - \vec{\mathbf{x}}\|^3} \cdot (\vec{\mathbf{v}} - \vec{\mathbf{x}})$$

where:
- $\mathbf{f}(\vec{\mathbf{x}})$ is the 'density' force at the point $\vec{\mathbf{x}}$
- $\vec{\mathbf{v}}$ is a point of the considered space ξ
- $\rho(\vec{\mathbf{v}})$ is the density potential at the point $\vec{\mathbf{v}}$
- C is a constant (optional)

The force is steered in the direction of the emitter point, and its intensity is proportional to the square inverse[5] of the distance to this point. This strong decrease with the distance is prone to a substantial scalability: the main field lines associated to the main maxima of density are detected when navigating with a low zooming rate, but this main tendency does not affect the minor maxima visualised while zooming in.

It is obvious that the computation of such a field is CPU-consuming, as all the points of the space must be integrated (summed) for each point of computation. Due to this significant computation time[6], the process happens off-line. A separated piece of software must be run independently to generate the density map texture and the field of forces.

Representation

In map table mode, a representation of a traditional compass has been developed. It consists of a static vertical blue axis, a red horizontal arrow that shows the direction of the field at the current point and a yellow stick remaining parallel to the main arrow, which variable size depends on the intensity of the field. These elements are presented within Figure 7.9.

The compass stands over the terrain, remaining centered in the visible part of the terrain. Some effects that happen with traditional magnetic compass were observed.

[5]The cube power in the sum comes to compensate the length of the steering vector.

[6]More than 5 minutes on a Pentium IIII 2.8GHz to generate 3 points per kilometre on a terrain of 24x20km.

For example in the zones containing a lot of finds, this compass throws into a panic as a magnetic one when approaching magnets.

7.7 Summary of the main differences between navigation modes

Regarding the differences between the application in standard mode and in map table mode, it appears that many devices and modules are specific. The integration of all the components in both standard mode and map table are given within the tables 7.2 and 7.3 for the interaction devices and for the navigation modules, respectively.

	Standard navigation	**Immersive map table**
Keyboard	Access way to most of the functions	*Tests and debugging tasks only*
Mouse	Can be a primary navigation device (movement and picking only), but the keyboard is still required for advanced tasks	*Not used*
Gamepad	Independent interaction device	Main interaction device
Head tracker	*Not used*	Updates the view according to the user's point of view
Hand tracker	*Not used*	Grab-and-move and show-to-pick metaphors
Bend-finger glove	*Not used*	Many postures used as any button

Table 7.2: Interaction devices integration

	Standard navigation	**Immersive map table**
Compass	*Not used*	Static on the middle of the terrain
Dynamic widgets	Pickable with mouse or gamepad pointer (computes the projection with the scene to choose the object)	Pickable with the compass axis: the user must move the terrain to put the widget at its center
Billboard set	Bonded on user's view	Static on the middle of the terrain
Mini map	The projection of the corners of the screen onto the terrain representation provides the edges of the polygonal displayed surface	*Not used*
Status bar	Bonded at the bottom of the screen (optional)	*Not used*

Table 7.3: Navigation modules integration

Chapter 8

Evaluation

8.1 Evaluation methods

The evaluation of the application by the author cannot be considered as objective. Therefore external comments were collected from archaeologists from the Department of History at the University of Hull and during a departmental poster presentation.

Representatives from 'English Heritage' and from 'Archaeology Data Service', two major organizations of the cultural heritage provided strong feedback about the information visualization during their visits.

At this final state of the project, the author has considered a scientific survey to evaluate the navigation metaphors and devices. The main results of this user evaluation will be described and interpreted.

8.2 English Heritage feedback

8.2.1 Visits

English Heritage acts as the English Government's adviser on the historic environment. It is funded by both the Government and the revenue earned from their properties but also sponsored by many Ministerial Departments (English Heritage 2004). Their mission is "to make sure that the historic environment of England is properly maintained and cared for. By employing some of the country's very best architects, archaeologists and historians, [they] aim to help people understand and appreciate why the historic buildings and landscapes around them matter." (ibid).

An early version of this project has been demonstrated on the 6x2.5m wall of the HIVE to three representatives of English Heritage: Pete Horne (Head of Air Photography Unit), Paul Bryan (Metric Survey Team Leader & Head of the Photogrammetric Unit) and Keith Miller (Inspector of Ancient Monuments) on the 4th March 2004 and a second time to Ian Panter (Regional Science Adviser for English Heritage) on the 19th March.

It is important to note that the project did not support head-tracking system at the time of the visit. First of all the main concepts were introduced, via a demonstration of the project applied to the 'Foulness Valley' large scaled landscape previously described. Focused then shifted to Hayton, where information gathering functionality was demonstrated. This included for example the possibility to swap between textures to demonstrate how the crop marks matched the aerial pictures, and how the reconstruction of the fort fits these crop marks. Finally, one of the representative navigated alone (but assisted) in the virtual environment for about ten minutes, using the wireless gamepad.

8.2.2 General comments

During and after the demonstration, the English Heritage representatives provided comments on general issues. The most impressive aspect was the immersive simultaneous visualization of a large amount of data. This was mainly due to two factors: the management of a multi-scaled environment and the visualization of abstract data. For example, the possibility to zoom quickly from the bird's eyes view to some detailed information was really appreciated. This changes a lot from Arcview, which takes a long time to perform such an operation. Numerical information like the databases and tables provided and rendered as stick fields were also well received, even though it was a simple metaphor.

More generally, favour was given to the functions that provided a more intuitive access to the information. Tables of numbers are not intuitive for the human brain, whereas coloured sticks are much more representative. Moving in a semi-realistic landscape is easier than a flat stack of layers as well.

8.2.3 Demonstrator weaknesses

Due to initialisation errors, the stereo mode was not correctly configured for the wall. Therefore, it was quite painful for the eyes to watch some of the details like the modeling of the Hayton Fort. While looking at close (and small) objects, the separation between the left and the right eye representations was really too large and the eyes could not easily refocus. Reducing artificially the human eye separation distance to 1cm provided a more affordable modeling, but the stereo effects on the whole landscape were lost as demonstrated within Figure 8.1. This issue was fixed later.

Independently to the virtual environment, some two-dimensional maps with small symbols like circles, squares or triangles produced by some archaeologists were not considered as comprehensible enough for the general public. This has been taken

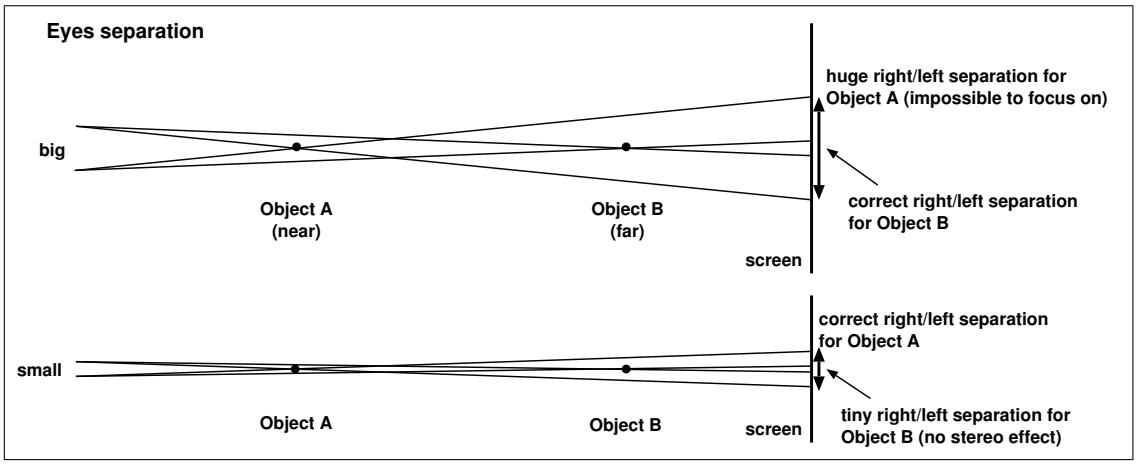


Figure 8.1: Stereo 3D: eyes separation effect (objects are represented as points)

into consideration during the design of more recent parts of the project, therefore more comprehensible symbols have been preferred. The three-dimensional sticks for example are only a translation of that kind of small symbols in the superior dimension. The massive use of icons and billboards attached to each object (in the limit of the available computational power) may improve the average readability.

8.3 Graduate research conference (poster presentation)

The main features of the environment were demonstrated through a poster presentation. This presentation did not involve the environment itself, hence it provided comments around more theoretical aspects. Most of the feedback that has been collected can be summed up in two main points.

The main weakness was the lack of scalability of the hardware support of such a virtual environment. At this state of the project, it can only be run on the HIVE (unique and not movable) or, with some limitations on a simple PC. In particular, it was asked if a support for a Personal Digital Assistant (abbr. PDA) or for augmented reality devices were available. It was argued that the case of the PDA is not really

relevant, as it cannot produce any immersive feeling. Augmented reality could have been considered, but this involves many other considerations (open air motion capture, camera registration (Doganis 2001), real and virtual image superimposition, etc) that were not in the scope of this project.

The main advantage was the ability to gather many various kind of information into a single environment. In particular the cohabitation of three-dimensional and two-dimensional information was appreciated as well as the modeling of the databases by the fields of sticks.

8.4 Archaeological Data Service feedback

The 'Archaeology Data Service', part of the 'Arts and Humanities Data Service' (AHDS) is a British national service that supports access to digital archaeological information. As this is exactly the purpose of this project, the visit of Dr William Kilbride (Assistant Director of AHDS-Archaeology) on 27th July 2004 was really valuable.

The state of the software application at the time of this visit was more advanced than during the English Heritage visit. The stereo mode was well configured, using an average eyes fusion distance[1], and solutions to the focussing problem.

Two positive aspects were pointed out. The first point was about the stick field automatically generated from a database. More than the three-dimensional visualization of the data, the automated process that leads to such a representation seems to be interesting. The second point was the usage of the CIDOC-CRM (Conceptual Reference Model). Apparently, the CRM is talked about a lot without really being used. This leads to think that the integration of the CRM in this project is really a matter of novelty.

[1]The eyes fusion distance is the distance from the user to the point where the lines of sight for both eyes intersect.

8.5 Scientific survey

8.5.1 Objectives

A scientific survey has been carried out. It aimed to evaluate the navigation (movements and object selection) in the immersive environment, by experienced and novice users. Non-experienced users are crucial in this experiment, as it aims to serve archaeologists which may not be familiar with virtual environments. The survey considered both wireless gamepad and tracked glove.

8.5.2 Method

A simplified mode has been developed to navigate within the immersive environment. The environment was run within the HIVE, in map table mode with the stereo glasses. The P5 glove was tracked using the Vicon system.

To avoid any side effect, the navigation was very limited: the user can only move and select widgets (referred to as 'targets' in this section). He had to follow a path containing fifteen targets and select all of them in a pre-defined order. A three-dimensional medieval soldier positioned on top of the target shows the current objective to the user. Moreover, the compass previously described has been designed to behave not according to the density of finds, but to show the direction of the current target to reach.

The target selection using the glove (show-to-pick) did not use the small stick representing the virtual index previously described. It used the pointer of the compass, whilst navigating with the gamepad. This metaphor is less powerful than the original, but avoided confusion whilst changing from the gamepad to the glove.

While navigating in the environment, the actions of the user have been recorded into a file. This includes the position of the user in the virtual environment all along the survey and the attempts of target selection (successfully achieved or not).

After a short explanation, the user had to navigate once with the gamepad and once again using the tracked glove. Once the navigation task completed, the user was asked to fill in a short questionary, in order to collect informal information about the general feeling. This form is available in Appendix C, page 139.

It is important to note that considering the time available, no user-specific calibration of the glove was performed before the test.

8.5.3 Subjective comments

Many comments have been given by the user during their navigation and on the feedback questionaries.

The most common criticism was about the picking system, with both glove and gamepad, that does not show clearly if the user is close enough to collect the target. Most of the testers would have liked to show a change of color of the compass or the target. Sound could have been used as well. The only people that were not disturbed by that were the video game addicts that apparently are used to evaluate quickly such implicit parameters.

Generally people preferred the gamepad to the glove as it is a more common device, and therefore easier to use. However, for small movements, the glove was sometimes preferred as it appears more precise. While crossing the whole scene, the user had to grab many times the terrain, which was a bit painful for the arm. It was proposed to introduce a scalable grab-and-move metaphor: closing the full hand to grab would move quickly, whereas closing part of it would move more slowly.

Considering the targets picking, the testers could be classified into two classes: those who liked the glove and those preferring the gamepad. In the former where found people with relatively small hands that fit the glove better, and for whom it was easier to do the picking posture. In the latter are the gamers and people with larger hands, less adapted to the glove.

Surprisingly, the disturbing effect of the glove wire was never noticed by the testers.

Around half of the testers answered that they were more confident with the navigation tools at the end of the experiment than at the beginning.

8.5.4 Formal results

Seven people participated to this survey.

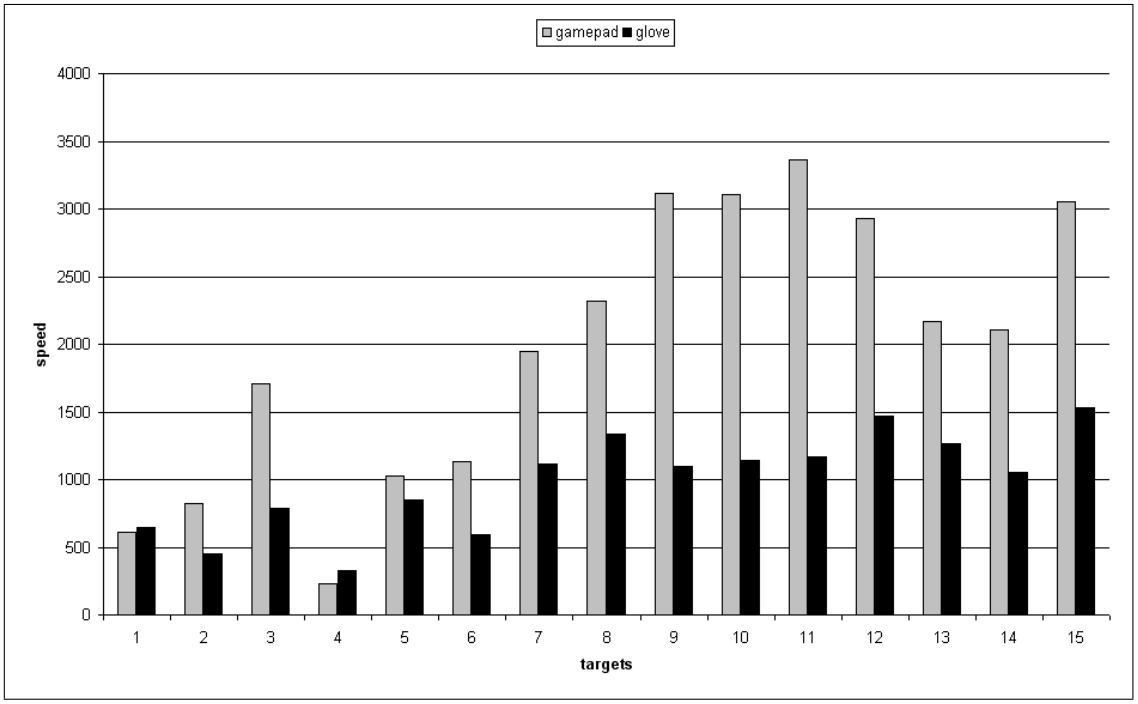


Figure 8.2: Speed (m/s) against target number

The Figure 8.2 provides an overview of the experiment: it represents target after target the average speed of the users with the glove and the gamepad. Note the trend of increase: the learning effect is obvious.

The main results are given within Tables 8.1 and 8.2 for the gamepad and the glove respectively.

Considering the average time needed to reach a target, an mean of 6.5s was recorded with the gamepad against 11.4s using the glove. However, the standard

	Min	Max	Mean	Deviation
Total Time (s)	39.4	178.5	97.1	43.3
Mean Target Time (s)	2.6	11.9	6.5	2.9
Moving Time (s)	26.7	83.9	50.4	19.5
Static Time (s)	11.3	94.6	46.6	26.2
Distance (m)	226759	329127	270124	37953
Total Speed (m/s)	1796	7637	3664	1751
Moving Speed (m/s)	3411	11275	6321	2280
Mistake Number	1	8	3.4	1.8

Table 8.1: Gamepad test bed main results

	Minimum	Maximum	Mean	Deviation
Total Time (s)	93.5	212.9	171.6	38.0
Mean Target Time (s)	6.2	14.2	11.4	2.5
Moving Time (s)	46.7	132.5	93.3	20.6
Static Time (s)	46.9	118.9	78.3	20.3
Distance (m)	234759	337105	292259	33353
Total Speed (m/s)	1139	3293	1854	512
Moving Speed (m/s)	2058	6600	3484	1010
Mistake Number	118	411	249	89

Table 8.2: Glove test bed main results

deviation is only 2.5s for the glove, against 2.9s using the gamepad (although its value is half of the glove's one). This is due to the fact that the gamepad was already well known by a couple of testers, whereas the glove is a novelty for everybody. This leads to balance a bit the difference of results between glove and gamepad. Another test carried out after a learning stage would be useful to conclude firmly about this issue.

It is also noticeable that the distance of the trajectory is higher with the glove than with the gamepad.

The number of mistakes when attempting to select an object with the gamepad is very low: the mean is 3.4, which equates to 16%. It is impossible to compare this figure with the glove's (249). The glove uses a continuous picking method (tries to

pick all the time as far as the index is pointing), whereas the gamepad's picking is discrete (picks once when pressing the button).

The average speed to reach a target against the distance to cover is given within the Figure 8.3. It shows clearly that for the large distances, the gamepad is faster, whereas for the short ones the glove is slightly better.

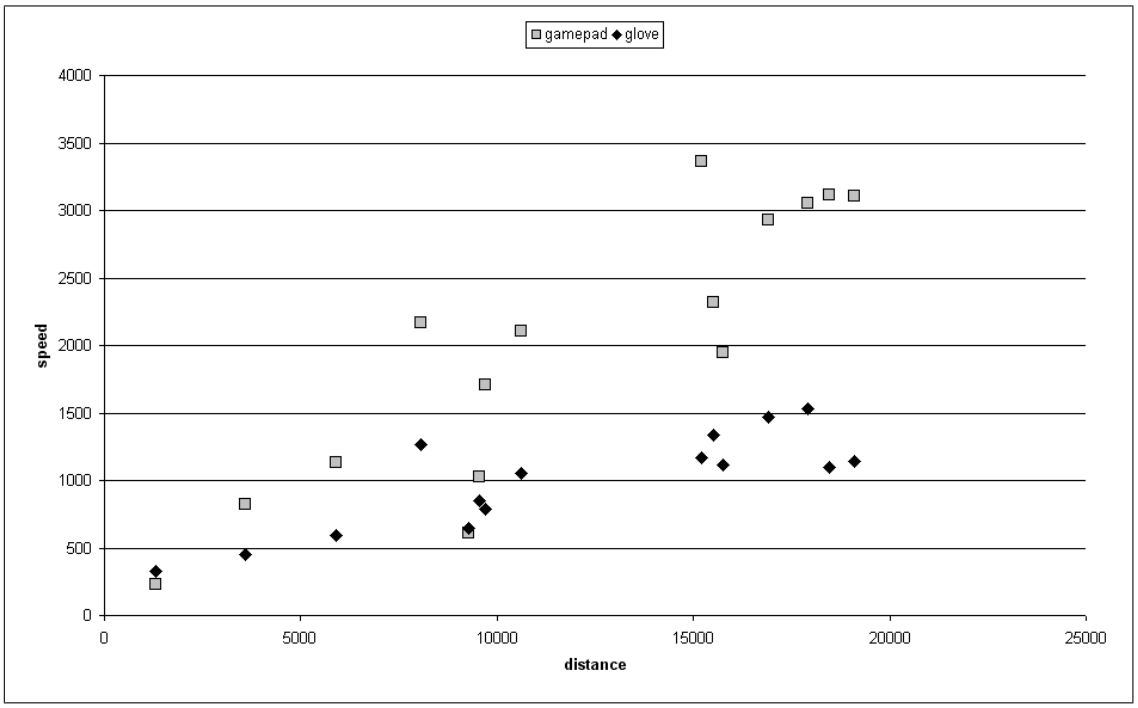


Figure 8.3: Speed (m/s) against distance (m) to cover

This chart also demonstrates how the average speeds with the glove are grouped, whereas those using the gamepad are more dispersed.

8.5.5 Survey conclusion

It is quite easy to start with a gamepad and it is already well known by some users. The gamepad seems to be adapted to the gross movements, whereas the glove is preferred for precise selection. A learning stage should improve the usage of both tools, but particularly of the glove.

The intensity bar that shows the distance from an object to the user is not

adapted to this kind of experiment. The targets that can be picked should be distinguished using an animation, a color code or by playing a sound.

Another test after a learning stage should be performed to eradicate the effects of pre-knowledge.

8.6 Summary

The dynamic representation of large amount of data seems to be a key feature of an efficient archaeological virtual environment. The integration of various sources of data in the same environment, visualised at the same time and understandable without any kind of help must be the goal.

The static elements like the terrain representation which appear quite realistic should remain but considered more as a basis for the project than as an end itself. This can lead to a virtual environment that would be at the same time realistic and symbolic, referred to as 'believable' (Roussou & Drettakis 2003).

The test that has been carried out, in order to evaluate the navigation metaphors, demonstrated that the glove could be useful for precise movements, whereas the gamepad is more adapted to gross movements. The compass metaphor could also be improved.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

An immersive virtual environment has been successfully built. The objectives of the project are now discussed against what has been achieved.

Most of the pieces of information that are typically available to the archaeologist have been integrated. This includes the classical data contained in a GIS toolkit (Geographic Information System) such as Arcview: aerial pictures, OS maps, crop marks, soil maps and DEM (Digital Elevation Model). However, these kinds of information have been strongly enhanced to use the full power of a virtual environment. The DEM has been naturally represented in a three-dimensional way. All the textures that can be draped on it are easily blended and swapped or switched on and off. The crop marks have been automatically classified and coloured using the Stoertz's method and enhanced to the third dimension as well for a better location by the user. According to most of the testers of the application, the visualization of the terrain on the wall screen, even without the head-coupled mode switched on, really provides a good perception of the landscape and thus should hasten their archaeological research.

Gathering stand-alone pieces of information like short texts, single pictures or

three-dimensional objects is the other main strength of the application. The representation of diverse kinds of data in one single scalable environment is also a novelty that helps the archaeologists, as traditional photo albums do not offer a positioning of the finds on the terrain. The collection of the information in a rotating billboard set provides a useful and intuitive way of dealing with large amount of pictures and texts at the same time. The challenging representation of numerical databases of finds in a fancy and useful way (the fields of sticks) leads to great hopes about the gathering possibilities of any kind of data set that would be available. Including an implementation of the emergent standard for cultural data storage and exchange, the CIDOC 'Conceptual Reference Model', reinforces such hopes. Help to navigation offered by the density map and compass unifies strongly the information available.

The experimentation of novel ways of navigation in the environment, particularly using the finger-bend tracked glove was very productive. Even though the interaction metaphors are not perfectly tuned at the moment, the use of a higher quality glove should provide a better interaction, be less intrusive and be more intuitive. The successful implementation of the 'grab-and-move' metaphor already demonstrates the potential of such a navigation device.

Although only in a pilot stage, the immersive representation using the HIVE wall screen and the head-coupled stereo already proves how valuable the immersion feeling could be, using the maximum of brain bandwidth available. The attention of the archaeologist in the virtual environment is maximized, as there is nothing else other than the screen in their field of view. Moreover, it is more intuitive to walk around the terrain to look at its opposite face than to click and move the mouse.

A scientific survey involving computer scientists and non-computer scientists has been carried out, in order to evaluate the navigation metaphors only. It demonstrated that the glove could be useful for precise movements, whereas the gamepad is more adapted to gross movements. The compass seems to be a fair metaphor,

even though it could be improved.

To conclude, the majority of the project's objectives have been reached. A virtual environment that encompasses any kind of data set available to the archaeologist, incorporating multidimensional ones, and that represents them within an immersive environment, using intuitive navigation metaphors such as a tracked glove has been successfully built and runs. It as been informally validated by archaeologists and representatives from major national organizations such as English Heritage and Archaeological Data Service.

9.2 Future work

Some aspects of the project are about to be completed. Other ones are simple ideas provided to the potential successor(s) of the author.

The tuning of the immersive visualization and the tracked glove should be completed soon. This should lead to a firm conclusion about the usage of a fully immersive virtual environment, including visualization and navigation.

The GUI-based database manager currently deals with ready-to-use information. A tool that facilitates the importation of data (crops the textures, verifies file formats, etc) would be useful for archaeologist to integrate new data.

The mini-map is currently only a display device. It could be improved by adding some more interactive tools, such as a 'select-and-go' metaphor.

Performance enhancement can be expected by using the module 'OSGterrain' of the most recent version of OpenSceneGraph (0.9.7) (Osfield & Burns 2004). It can load a Digital Elevation Models (DEM) and map it with a texture. Moreover, it also generates dynamically several levels of detail and split the terrain database into pieces (data pager) which should improve dramatically the rendering speed. Improved portability could be achieved with the integration of VR Juggler instead of the current specific tracking system implementation.

The integration of some more high-level and multimedia pieces of information such as HTML or video streams should improve slightly the gathering power of the application. Video streams are currently supported by OSG for GNU/Linux and OSX, but not for MS Windows.

References and Bibliography

References

Arthur, K. W., Booth, K. S. & Ware, C. (1993), 'Evaluating 3d task performance for fish tank virtual worlds', *Information Systems* **11**(3), 239–265.
`citeseer.ist.psu.edu/arthur93evaluating.html`

Ashley López, M. (2003), 'Managing the real with the virtual: A role for digital media recording in archaeological fieldwork', *Proc. VAST 2003: The 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage* pp. 29–35.

Beaudouin-Lafon, M. (2000), Instrumental Interaction: An Interaction Model for Designing Post-WIMP User Interfaces, University of Aarhus. [Accessed 19 March 2004].
`http://www.daimi.au.dk/CPnets/CPN2000/download/chi2000.pdf`

Bertin, J. (1967), Sémiologie graphique, Paris.

Bourke, P. (2002), '3D Stereo Rendering Using OpenGL (and GLUT)'.
`http://astronomy.swin.edu.au/~pbourke/opengl/stereogl`

Cavens, D., Lange, E. & Schmid, W. (2003), 'Virtual Alpine Landscapes and Autonomous Agents', p. 5. [Accessed 22 March 2004].
`http://www.sim.inf.ethz.ch/projects/alpsim/dessau03.pdf`

Center for the New Engineer (1995), Minimal Spanning Tree Clustering Algorithm. [Accessed 19 March 2004].
`http://cne.gmu.edu/modules/dau/stat/clustgalgs/clust4_bdy.html`

CIDOC (2004), Definition of the CIDOC object-oriented Conceptual Reference Model (version 4.0). [Accessed 06 July 2004].
`http://cidoc.ics.forth.gr`

Cockburn, A. & McKenzie, B. (2000), 'An evaluation of cone trees'.
`citeseer.ist.psu.edu/cockburn00evaluation.html`

Conner, D. B., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C. & van Dam, A. (1992), Three-dimensional widgets, *in* 'Proceedings of the 1992 Symposium on Interactive 3D Graphics, Special Issue of Computer Graphics, Vol. 26', pp. 183–188.
`citeseer.ist.psu.edu/conner92threedimensional.html`

Dikaiakou, M., Efthymiou, A. & Chrysanthou, Y. (2003), 'Modelling the Walled City of Nicosia', *Proc. VAST 2003: The 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage* pp. 57–66.

Doganis, F. (2001), Augmented reality in archaeology: Registration issues, Master's thesis, The University of Hull.

Essential Reality (2004). [Accessed 06 August 2004].
`http://www.essentialreality.com`

Fabrizio, T. C. (2001), 'Interacting with gis: From paper cartography to virtual environments'.
`citeseer.ist.psu.edu/502290.html`

Faisstnauer, C., Schmalstieg, D. & Szalavári, Z. (1997), 'Device-independent navigation and interaction in virtual environments', *submitted to VRAIS '98* .
`citeseer.ist.psu.edu/307288.html`

Feliz, D. A. (1999), 'ARCHAVE: A virtual reality interface for archaeological 3D GIS'. [Accessed 19 March 2004].
`http://www.cs.brown.edu/research/graphics/research/sciviz/archaeology/archave/ARCHAVE_proposal.pdf`

Forte, M. (2002), About *virtual archaeology*: disorders, cognitive interactions and virtuality, Institute for Technologies Applied to Cultural Heritage, p. 1. [Accessed 19 March 2004].
`http://www.mmi.unimaas.nl/eculturenet/publicPDF/VRarcheo.pdf`

Forte, M. & Guidazzoli, A. (2004), About Virtual Archaeology. [Accessed 19 March 2004].
`http://mosaic.infobyte.it/project/towardsa.html`

Halkon, P. & Millett, M. (1999), *Rural Settlement and Industry: Studies in the Iron Age and Roman Archaeology of Lowland East Yorkshire. Yorkshire Archaeological Report*, Vol. 4, Yorkshire Archaeological Society.

Holbein, H. (1533), 'The Ambassadors or 'Jean de Dinteville and Georges de Selve''. [London National Gallery – NG1314].

Jingtong, J., Donghua, W., Young, P. M., Berry, R. D., Keeler, R. H., Willan, J. M. & Sechrist, D. R. (1996), Creating Relief Maps Of China. [Accessed 28 April 2004].
`http://gis.esri.com/library/userconf/proc96/TO350/PAP349/P349.HTM`

Lahanas, M., Kemmerer, T., Milickovic, N., Karouzakis, K., Baltas, D. & Zamboglou, N. (2000), 'Optimized bounding boxes for three-dimensional treatment planning in brachytherapy', *Medical Physics* **27**(10), 2333–2342.

NVIDIA (2004), 'New NVIDIA Scene Graph Technology Streamlines and Accelerates 3D Content Development'. [Accessed 11 August 2004].
`http://www.nvidia.com/object/IO_14925.html`

OpenSceneGraph (2004). [Accessed 10 August 2004].
`http://www.openscenegraph.org`

Osfield, R. (2003). [Accessed 09 March 2004].
`http://osgcvs.no-ip.com/osgarchiver/archives/August2003/0261.html`

Osfield, R. & Burns, D. (2004), 'Openscenegraph 0.9.7 release'. [Accessed 10 August 2004].
`http://openscenegraph.sourceforge.net/news/index.html#OpenSceneGraph-0.9.7`

Roussou, M. & Drettakis, G. (2003), 'Photorealism and Non-Photorealism in Virtual Heritage Representation', *Proc. VAST 2003: The 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage* pp. 47–56.

Shilling, R., Morgan, D., Mosbruger, M., Beilstein, D. & Orichel, T. (2003), Enhancing Information Fusion Using Spatial Auditory Displays and Videogame Interfaces, Rensselaer Polytechnic Institut. [Accessed 12 March 2004].
`http://www.movesinstitute.org/Publications/ShillingONRWorkshop.pdf`

Stoertz, C. (1997), *Ancient Landscapes of the Yorkshire Wolds – Aerial photographic transcription and analysis.*

van Dam, A., Forsberg, S. A., Laidlaw, H. D., LaViola, J. J. & Simpson, M. R. (2002), Immersive VR for Scientific Visualization: A Progress Report, pp. 26–27.

Vicon (2004*a*). [Accessed 06 May 2004].
`http://www.vicon.com`

Vicon (2004*b*), 'Vicon Mcam2 technical sheet'. [Accessed 21 September 2004].
`http://www.neat.co.kr/hardware/vicon/reference/VMS054A_MCam2_TechSheet.pdf`

General bibliography

3ds Max (2004). [Accessed 04 May 2004].
`http://www4.discreet.com/3dsmax`

Archaeology Data Service (2004). [Accessed 27 July 2004].
`http://ads.ahds.ac.uk`

CAVE system (2004). [Accessed 05 March 2004].
`http://www.evl.uic.edu/pape/CAVE/oldCAVE/CAVE.html`

CIDOC (The International Committee for Documentation of the International Council of Museums) (2004). [Accessed 06 July 2004].
`http://www.cidoc.icom.org`

CVS (2004). [Accessed 05 March 2004].
`http://www.cvshome.org`

Diosoft (2004). [Accessed 25 March 2004].
`http://www.diosoft.com`

DirectX (for Direct3D) (2004). [Accessed 23 March 2004].
`http://www.microsoft.com/windows/directx/default.aspx`

Doxygen (2004). [Accessed 18 May 2004].
`http://www.doxygen.org`

English Heritage (2004). [Accessed 05 March 2004].
`http://www.english-heritage.org.uk`

ESRI (2004), Arcview. [Accessed 19 March 2004].
`http://www.esri.com/software/arcview`

Everaldo (2004), Crystal svg icons. [Accessed 20 September 2004].
`http://www.everaldo.com/crystal.html`

GCC (GNU Compiler Collection) (2004). [Accessed 23 April 2004].
`http://gcc.gnu.org`

HIVE (Hull Immersive Visualization Environment) (2004). [Accessed 05 March 2004].
`http://www.hive.hull.ac.uk`

Java3D (2004). [Accessed 05 March 2004].
`http://java.sun.com/products/java-media/3D`

KDE (2004). [Accessed 05 March 2004].
`http://www.kde.org`

libxml (2004), 'The XML C parser and toolkit of Gnome'. [Accessed 06 July 2004].
`http://www.xmlsoft.org`

Logitech (2004). [Accessed 23 March 2004].
`http://www.logitech.com`

Mandrake Linux (2004). [Accessed 23 April 2004].
`http://www.mandrakesoft.com`

Microsoft Office (2004). [Accessed 05 March 2004].
`http://office.microsoft.com/home/default.aspx`

Microsoft Visual C++ (2004). [Accessed 05 March 2004].
`http://msdn.microsoft.com/visualc`

OGRE - Object-oriented Graphics Rendering Engine (2004). [Accessed 23 March 2004].
`http://www.ogre3d.org`

Open Inventor (2004). [Accessed 05 March 2004].
`http://oss.sgi.com/projects/inventor`

Open Performer (2004). [Accessed 05 March 2004].
`http://oss.sgi.com/projects/performer ; http://www.tgs.com`

OpenGL (2004). [Accessed 23 March 2004].
`http://www.opengl.org`

OpenRM Scene Graph (2004). [Accessed 05 March 2004].
`http://www.openrm.org`

OpenSG (2004). [Accessed 05 March 2004].
`http://www.opensg.org`

Ordnance Survey (2004). [Accessed 05 March 2004].
`http://www.ordnancesurvey.co.uk`

PNG (2004). [Accessed 05 March 2004].
`http://www.libpng.org`

Polhemus (2004*a*). [Accessed 07 May 2004].
`http://www.polhemus.com`

Polhemus (2004*b*), Fastrak. [Accessed 07 May 2004].
`http://www.polhemus.com/FastTrak/fastrak.pdf`

Python (2004). [Accessed 23 March 2004].
`http://www.python.org`

Silicon Graphics (2004). [Accessed 05 March 2004].
`http://www.sgi.com`

Simple DirectMedia Layer (SDL) (2004). [Accessed 31 March 2004].
`http://www.libsdl.org`

The Combat Simulator Project (2004). [Accessed 25 March 2004].
`http://csp.sourceforge.net`

Trolltech (2004*a*). [Accessed 25 March 2004].
`http://www.trolltech.com`

Trolltech (2004*b*), Qt Presentation. [Accessed 23 April 2004].
`http://www.trolltech.com/products/qt/qtpresentation.pdf`

VR Juggler (2004). [Accessed 20 August 2004].
`http://www.vrjuggler.org`

W3C (2004), 'Extensible Markup Language (XML)'. [Accessed 09 July 2004].
`http://www.w3.org/XML`

WxWidgets (formerly WxWindows) (2004). [Accessed 23 April 2004].
`http://www.wxwindows.org`

Appendix A

Glossary

Billboard *In our case:* a two-dimensional container encapsulating text or images.
(p 83)

Billboard set *In our case:* a rotative three-dimensional slide-show that contains the billboards collected by the user during his navigation.
(p 84)

CIDOC International Committee for Documentation of the International Council of Museums
(p 39)

CIDOC-CRM 'Conceptual Reference Model' of the CIDOC. This emergent model provides an accessible and flexible storage of cultural heritage documentation and should make the join between all the sources of documentation, from museums to libraries.
(pp 39, 65)

Contour line Virtual line on a geographic map including all the points having the same elevation.

Crop marks Remains of the past under the ground that are visible only from their effects on vegetation growth.
(pp 19, 74)

DEM Digital Elevation Model: a map of high of a piece of terrain.
(p 73)

Finger-bend glove A motion tracking device that captures the bending of the fingers.
(pp 55, 96)

Frustum A truncated cone or pyramid. In a visualization context, this refers to the volume represented on the screen (the field of view).
(p 94)

Fusion distance The eyes fusion distance is the distance from the user to the point where the lines of sight for both eyes intersect.

GIS Geographic Information System: a concept or toolkit that encompasses at least a geographic database and an interface for this database.
(pp 29, 73)

GUI Graphic User Interface.
(p 68)

HIVE Hull Immersive Visualization Environment, which is mainly based on a rear-projected wall screen (6m wide by 2,5m high). A pair of video-projector ensures the coverage of the whole surface. Seven infrared cameras are dedicated to the tracking system.
(pp 49, 51)

Navigation *In our case:* the actions of the user in the virtual environment: movement, item selection and information query
(pp 30, 91)

OSG OpenSceneGraph, a powerful open source, cross-platform scenegraph toolkit on top of OpenGL.
(p 42)

OS (map) Ordnance Survey: Britain's national mapping agency.

Parallax (Binocular parallax) The apparent difference in position of an object as seen separately by both eyes.

Scenegraph A scenegraph is an (object-oriented) data structure or toolkit that allows to manage efficiently a set of three-dimensional objects stored in a tree (a graph).
(p 33)

SDK Software Development Kit: a collection of development tools.

Stereo 3D This method reproduces the natural way of the human vision by producing a different image for each eye and then provide a feeling of third dimension.
(p 25)

Tracking Also referred to as **motion capture**, this encompasses all methods that use the movements of the user.
(pp 27, 50, 52, 92)

Wall screen Also referred to as **work wall**, a screen that covers an entire wall in order to immerse the user.
(p 51)

Widget "An encapsulation of geometry and behavior used to control or display information about application objects." (Conner et al. 1992)
(p 79)

Appendix B

Configuration Files

Hereafter are detailed most of the configuration files entries. For each entry is describe the range of values and its meaning. In most of the configuration files the command '`Object name`' begins the object 'name' and the command '`EndObject`' closes it. Note that the attributes of such objects are written here '`name.attribute`', which is **not** valid in a file.

B.1 Main configuration files

The following configuration files must be stored in the '`./config/`'.

B.1.1 gamepad.ini

The configuration of the gamepad.

`cut_value`	**0..1**	The minimal value of the analogic axis that is taken into account.
`rotate_scale`	**R+**	The scale applied to the rotation.
`translate_scale`	**R+**	The scale applied to the translation.
`zoom_scale`	**R+**	The scale applied to the scale.
`translate_slider_scale`	**R+**	The scale applied to the slider.
`inverseXYaxis`	**bool**	Inverse the X and Y axis.
`directionX`	**+/- 1**	Direction of the X axis.
`directionY`	**+/- 1**	Direction of the Y axis.

B.1.2 headtracked.ini

Parameters for the head tracking system.

`tracker.type`	**fastrak, vicon**	Tracking system to be used.
`tracker.port`	**auto, net:harrier**	Port of the tracking system.
`modelA`	**0..2**	Rotation model.
`modelB`	**0..2**	Translation model.
`modelC`	**bool**	*Experimental – put 0.*
`variableFrustum`	**bool**	X-Y variable frustum.
`nearFrustum`	**bool**	Z variable frustum.
`xclip`	**R+***	X-size of the displayed piece of terrain.
`yclip`	**R+***	Y-size of the displayed piece of terrain.
`eyeSeparation`	**R+***	User eye separation.
`screen.width`	**R+***	Width of the screen.
`screen.height`	**R+***	Height of the screen.
`screen.distance`	**R+***	Average distance from the user to the screen.
`RoomMatrix.originX`	**float**	X-origin of the tracker.
`RoomMatrix.originY`	**float**	Y-origin of the tracker.
`RoomMatrix.originZ`	**float**	Z-origin of the tracker.
`RoomMatrix.r00 .. r22`	**float**	Tracker matrix.
`GlassesMatrix`	**...**	Idem, for the shutter glasses.

B.1.3 fingers.ini

The configuration of the fingers: the P5 glove.

`useP5tracker`	**bool**	Use the p5 tracker instead of Fastrak/Vicon.
`realToVirtualScale`	**>1.0**	The scale factor between the reality and the virtual environment.

`zoomFactor`	**R+**	The factor to zoom using the glove.
`threshold1`	**0..32**	Straight threshold.
`threshold1`	**16..48**	Middle bend threshold.
`threshold1`	**32..63**	Totally bend threshold.
`RoomMatrix`	**...**	See 'headtracked.ini'

B.1.4 keylinks.ini

The links between the keys of the keyboard, gamepad and finger-bend glove and the tasks (actions) in the application. The content of this file has been previously detailed on page 61. Hereafter are provided the lists of most of the keys and the tasks available.

Keys

`keyboard.`*x*	`keyboard.f5`	`gamepad.r1`	`righthand.open`
`keyboard.insert`	`keyboard.f6`	`gamepad.r2`	`righthand.closed`
`keyboard.delete`	`keyboard.smaller`	`gamepad.hatN`	`righthand.ok`
`keyboard.home`	`keyboard.greater`	`gamepad.hatS`	`righthand.point`
`keyboard.end`	`keyboard.f7`	`gamepad.hatW`	`righthand.twins`
`keyboard.pageup`	`keyboard.f8`	`gamepad.hatE`	`righthand.cat`
`keyboard.pagedown`	`keyboard.f9`	`gamepad.a`	`righthand.horns`
`keyboard.up`	`keyboard.f10`	`gamepad.b`	`righthand.0`
`keyboard.down`	`keyboard.f11`	`gamepad.c`	`righthand.1`
`keyboard.left`	`keyboard.f12`	`gamepad.x`	`righthand.2`
`keyboard.right`	`keyboard.plus`	`gamepad.y`	`righthand.3`
`keyboard.f1`	`keyboard.minus`	`gamepad.z`	`righthand.4`
`keyboard.f2`	`gamepad.slide`	`righthand.thumb`	`righthand.5`
`keyboard.f3`	`gamepad.l1`	`righthand.index`	
`keyboard.f4`	`gamepad.l2`	`righthand.ring`	

Note that the posture contained in each of these sets are equivalent:
(`righthand.thumb, righthand.ok, righthand.1`),
(`righthand.index, righthand.point`),
(`righthand.open, righthand.5`) and
(`righthand.closed, righthand.0`).

Tasks

none	widget.pick3d	static.toogle
billboardset.collect	widget.unselect	stickfield.rotatedirect
billboardset.delete	textures.load0	stickfield.rotateindirect
billboardset.expand	textures.load1	spot.smaller
billboardset.reduce	textures.load2	spot.greater
billboardset.farer	textures.load3	sea.lower
billboardset.closer	textures.load4	sea.upper
billboardset.previous	textures.load5	video.record
billboardset.next	textures.toogle0	video.stop
widget.pick	textures.toogle1	

B.1.5 survey.ini

The file defining the survey mode properties.

inputFile	**string**	The name of the file containing the path (see below).
loopMode	**bool**	Restart the path after the last target.
recordTrajectory	**bool**	Record the complete trajectory (all frames).
Pointer.file	**string**	The file name of the 3D object that will be used for the target pointer.
Pointer.scale	**R+***	The scale applied to this object.
Pointer.offset	**R**	The offset of elevation for the object.
Cube.icon	**string**	The name of the file containing the icon for the targets (cubes).
Cube.size	**R+***	The size of the cubes.

B.1.6 survey-path.txt

This file simply contains the positions of the path to be followed within the survey mode (couple of floats).

B.2 Data configuration files

These files are generally stored in the root of the data repository.

B.2.1 main.ini

The main file, given as parameter to the application.

survey	**bool**	Survey mode

surveyFile	**string**	The configuration file for the survey mode
title	**string**	The title of the environment
fatalMessageOnly	**bool**	Display only fatal messages
keyboard	**bool**	Use the keyboard
sun	**bool**	Ambient light
spotLight	**bool**	Spot light
drawStatusBar	**bool**	Display the status bar
drawPointer	**bool**	Display the gamepad pointer
drawSkyDome	**bool**	Display the sky dome (deprecated)
index	**bool**	Use a widget index file
widgetConfigFile	**string**	The widget configuration file
widgetIndexFile	**string**	The widget index file (CIDOC-CRM or archa)
csvConfigFile	**string**	The configuration file for the CSV data sets
autoWidgetAlt	**bool**	Automatically add the widgets on the surface of the terrain
compass	**bool**	Draw the compass
Sea.drawWater	**bool**	Draw the water fog (under-water)
Sea.drawSurface	**bool**	Draw the water surface
Sea.levelStep	**R+**	The water level movement step size
Terrain.draw	**bool**	Draw the terrain
Terrain.shape	**DEM, flat, cosine, grade**	The shape of the terrain
Terrain.sizeX	**R+**	E-W Size of the terrain
Terrain.sizeY	**R+**	N-S Size of the terrain
Terrain.depth	**R**	Depth of the terrain (not valid for DEM)
Terrain.DEMfile	**string**	The DEM file
Terrain.DEMpath	**string**	The textures path
Terrain.DEMgrid	**R+**	Grid size for the DEM
Terrain.texture_DENSITY_MAP	**bool**	Draped with the density map

`Terrain.texture_SATELLITE`	**bool**	Draped with the satellite pictures
`Terrain.texture_AERIAL`	**bool**	Draped with the aerial picture
`Terrain.texture_OS_MAP`	**bool**	Draped with the OS maps
`Terrain.texture_SOIL_MAP`	**bool**	Draped with the soils maps
`Terrain.texture_CROP_MARKS`	**bool**	Draped with the crop marks
`Terrain.texture_COMP_AER_SAT`	**bool**	Draped with a composite aerial-satellite
`Terrain.densityField`	**string**	The density map file
`SkyDome.file`	**string**	The sky dome file (deprecated)
`SkyDome.x`	**R**	
`SkyDome.y`	**R**	
`SkyDome.z`	**R**	
`SkyDome.scale`	**R**	
`Light.cut`	**R+**	Angle of the spot light
`Light.exp`	**R+**	Exponential attenuation of the light
`Light.quadAtten`	**R+**	Quadratique attenuation of the light
`StickField.multicolor`	**bool**	Multicolour stick mode
`StickField.fileName`	**string**	The data set file
`V.baseSize`	**R+**	Base size of the sticks
`Static.title`	**string**	The title of the object
`Static.draw`	**bool**	Draw a static object
`Static.file`	**string**	The file containing the object
`Static.x`	**R+**	E-W position of the object
`Static.y`	**R+**	N-S position of the object
`Static.z`	**R+**	Elevation of the object
`Static.ax`	**R**	X-axis of the object
`Static.ay`	**R**	Y-axis of the object
`Static.az`	**R**	Z-axis of the object
`Static.angle`	**R**	Angle of the object
`Static.scale`	**R+**	Scale of the object

B.2.2 widgetConfig.ini

The configuration file for the behaviour of the widgets.

`offsetX`	**R+**	The west offset of the widgets.
`offsetY`	**R+**	The north offset of the widgets.
`iconSize`	**R+**	The standard icon size.
`spinCubeSize`	**R+**	The standard spinning cube size.
`clusterSizeFactor`	**R+**	The ratio between the size of a standard cube and a cluster cube.
`scale`	**R+**	A position scale (not used).
`ClusterConfig.clustering`	**auto, user, off**	define the mode of clustering.
`ClusterConfig.minDistance`	**R+**	the minimum distance required between clusters.
`ClusterConfig.viewDistance`	**R+**	the distance of swap between the detailed and cluster view.
`ClusterConfig.iconFile`	**string**	The cluster icon file.

B.2.3 csvindex.ini

The configuration file for the CSV (Comma Separated Value) data sets.

`siteFile`	**string**	The site file.
`pinsFile`	**string**	The pins file.
`positionX`	**int**	Column of the X position in the file

The parameters `offsetX`, `offsetY`, `iconSize`, `spinCubeSize`, `clusterSizeFactor`, `ClusterConfig.clustering`, `ClusterConfig.minDistance` and `ClusterConfig.viewDistance` are the same than those from `widgetConfig.ini` below.

B.3 Additional files

B.3.1 Crop marks classifier: config.ini

The configuration file for the crop marks classifier.

`pitMaxSize`	**R+**	Maximal size of a 'pit'
`enclosureRate`	**R+**	The enclosure rate that distinguish between an open and a closed crop mark.
`linearShortLong`	**R+**	The maximal size of a 'short linear' crop mark.
`linearRatio`	**R+**	The width/height ratio to consider automatically as 'linear'.
`ellipticalSize`	> 1	The size of the ellipse for the rounded test.
`ellipticalRate`	< 1	The rate of points in the ellipse to consider rounded.
`regularRatio`	**R+**	The regular/elongated ratio.

B.3.2 Density map generator: config.ini

The configuration file for the density map generator.

`DEMfileName`	**string**	The DEM file.
`texturePath`	**string**	The texture path.
`widgetIndexFile`	**string**	The widget index file.
`fieldConstant`	**int**	Deprecated.
`DEMgrid`	**R+**	Grid size for the DEM
`pixel2km`	**int**	The number of pixels by kilometre.
`fieldStep`	**int**	The integration step for the field of force.
`CropMarksFilter.type`	**linear, inverse, square_inverse, gauss**	The type of filter (dilatation).
`CropMarksFilter.size`	**int**	The size of the crop mark filter.
`CropMarksFilter.radius`	**R+**	The radius of the crop mark filter.

`CropMarksFilter.alpha`	**R+**	The intensity of the crop mark filter.
`CropMarksFilter.palette`	**standard, blue_red, black_red_yellow, blue_red_yellow**	The colour palette.
`ItemFilter.size`	**int**	The size of the single widget filter.
`ItemFilter.radius`	**R+**	The radius of the single widget filter.
`ItemFilter.alpha`	**R+**	The intensity of the single widget filter.
`ItemFilter.factor`	**R+**	A constant factor of the single widget filter (deprecated).

Appendix C

Survey Feedback Form

Immersive experience: feedback form

Would please take 30 seconds to fill in this form ? It should help me to classify the numerical results acquired during the experiment. Thank you very much !

1. Your background

Are you a computer scientist ? Yes ☐ No ☐

	never	once a month	once a week	every day
Do you use a computer ?	☐	☐	☐	☐
Do you play video games ?	☐	☐	☐	☐
Do you use such environments ?	☐	☐	☐	☐
Do you use GIS software ?	☐	☐	☐	☐

2. General feeling

	inadequate	fair	good	excellent
Ease of eyes' focus	☐	☐	☐	☐
Stability of the map	☐	☐	☐	☐
Compass system	☐	☐	☐	☐

3. Movements

How were your movements ...

	difficult	fair	easy	very easy
... while beginning with the gamepad ?	☐	☐	☐	☐
... at the end with the gamepad ?	☐	☐	☐	☐
... while beginning with the glove ?	☐	☐	☐	☐
... at the end with the glove ?	☐	☐	☐	☐

4. Picking targets

How did you feel about the target picking ? *(you may select several answers)*

gamepad	glove	
☐	☐	No relevant problems.
☐	☐	It was quite confusing at the beginning.
☐	☐	It costed me time and patience.
☐	☐	I needed to try a couple of times to reach the target.
☐	☐	I had to try too many times before reaching the target.
☐	☐	It got on my nerves !

5. Free comments

If you have any other comments...

Appendix D

Trademarks

The following trademarks used in this thesis remain the property of their owner.

3ds Max	Java3D	OpenGL
Adobe Photoshop	KDE	Open Inventor
Arcview	Logitech WingMan Cordless	Open Performer
Compuserve	Macintosh	Ordnance Survey
Diosoft	Mandrake	OSX
Direct3D	Microsoft Access	Polhemus Fastrak
DirectX	Microsoft Excel	Silicon Graphics
Essential Reality	Microsoft Internet Explorer	Sun Solaris
IBM	Microsoft Visual Studio .NET	Trolltech Qt
Intel Pentium 4 Extreme	Microsoft Windows	Vicon Motion Systems Ltd
IRIX	NVIDIA Quadro FX3000	
Java		