2004-01-01

A Framework for Processing Viewer-based Directional Queries in a Simulated Mobile Environment

Keith Gardiner
Technological University Dublin

Recommended Citation

Gardiner, K.(2004). *A Framework for Processing Viewer-based Directional Queries in a Simulated Mobile Environment*. Masters dissertation. Technological University Dublin. doi:10.21427/D7NP68

A FRAMEWORK FOR PROCESSING VIEWER-BASED DIRECTIONAL QUERIES IN A SIMULATED MOBILE ENVIRONMENT

Submitted by

Keith Gardiner
B.Sc (Applied Computing)

In fulfilment of the requirement for a Masters Degree

To
Dublin Institute of Technology

Supervised by:

Dr. James Carswell & Dr. Ciaran Mac Donaill
Dublin Institute of Technology
Aungier St

May 2004

Declaration

I certify that this thesis which I now submit for examination for the award of Masters of Philosophy, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

This thesis was prepared according to the regulations for postgraduate studies by research of the Dublin Institute of Technology and has not been submitted in whole or in part for an award in any other Institute or University.

The Institute has permission to keep, to lend or to copy this thesis in whole or in part, on condition that any such use of the material of the thesis be duly acknowledged.

Keith Gardiner(DIT 2004)

Signature______________________ Date 2/2/05

Abstract

With the steady and fast advancements in the integration of geographic information systems and mobile location-based services, interest in exploiting this technology for Cultural Heritage (CH) data sharing has become apparent. In this area there has been an increasing need to integrate positional information with non-positional data and add a spatial dimension to the definition of a users "context".

In this research, the integration of Cultural Heritage information coupled with spatial technologies is investigated. The idea is based on a 3-tier architecture that positions the user at the client layer with a mobile device, a spatial query processor in the middle layer and a Cultural Heritage dataset at the database layer. This novel idea applied in this context is the basis for this study.

The research aims to develop a framework that enables the simulation of a mobile environment in which spatial applications can be implemented and evaluated without the current restrictions of mobile bandwidth in the context of availability and operational costs. The prototype system is demonstrated using a spatial application that processes user-based directional queries. The application gathers information about the position and orientation/direction of the user in a 3D environment. These parameters are evaluated along with the users profile and preferences and a user-based query is formulated. The query is processed using an Oracle spatial engine and the user-tailored results are delivered to the mobile device.

Finally, an experimental implementation shows how the query processor performs within a VRML model of Dublin linked to a spatially enabled CH dataset.

Acknowledgements

I would like to thank a number of people for their help and support during the course of my studies. I would like to thank my supervisors, Dr. James Carswell and Dr. Ciaran Mac Donaill, for their help and direction over the last couple of years. The project was developed by a team of researchers and I would like to thank Marco Neumann for his help during the development of the prototype system. His knowledge was very much appreciated. I would also like to thank all the people who triggered ideas over the course and all staff at the Digital Media Centre.

Table of Contents

Table of Figures

List of Tables

Chapter One

Introduction

1 Introduction

1.1 Introduction

The purpose of this research is:

- To investigate the area of spatial databases in the context of Cultural Heritage (CH) data-sharing applications and location-based services (LBS). The research focuses on how integrating Spatial Databases (SD) with Virtual Reality (VR) can be used as a medium to present Cultural Heritage artefacts to the user, depending on his/her location in a geo-spatial context. To achieve this goal, a comprehensive overview of the technologies that are needed to implement such a system has been carried out.

- To develop a system based on these findings that will enable users of the system, such as tourists, to navigate spatially within a VR environment on a standard PC from any location. In response to this navigation, the system will create a narrative based on a user's spatial context and deliver it by means of multimedia components to the user.

The main focus of the research is on the area of spatial databases, which is one of the core areas associated with location-based services. Initially a broad investigation is presented on the area of spatial databases, on which there has been substantial research carried out in the last two decades [Shekhar et al, 1999a]. Following this, the research highlights aspects of spatial querying before narrowing the focus to study "user-based queries in a spatial database environment".

1.2 Problem Statement

This research problem addresses the need to integrate spatial data with digitised cultural heritage information. The Cultural Heritages Interfaces (CHI) project examines how users interact within cultural spaces; how experts mediate information about those spaces; and it examines cultural heritage visitors' expectations and requirements. This information was used in the specification and design of a user-centred spatialised interface. A technology demonstrator of the navigation capabilities of such a spatial metaphor was developed, based on an existing repository of cultural heritage material drawn from the contents of a book called "Easter Rising 1916" [Collins and Kostick, 2000], presented through a database application and a web-based interface.

The goal of this research was to determine techniques that enable information related to cultural spaces to be delivered effectively via web-based interfaces. The results of the research should therefore provide an insight into the integration of existing technology as a delivery mechanism for cultural heritage information based on the user's context. A novel part of this research is the synergy that is created by the integration of a spatial database and VR, where every aspect of the system is driven by the data in the database.

1.3 Spatial Databases

Spatial data is a term used to describe data that pertain to the space occupied by objects in the real world. These are any data that are referenced by location - both absolute in terms of the earth's surface, and relative, in terms of distances and directions between objects. Spatial data are geometric, varied and consist of points, lines, polygons, surfaces, volumes and data of even higher dimensions [Guting, 1994]. They are usually found in conjunction with attribute or non-spatial data (e.g., name of a building, river).

Spatial data can be discrete or continuous. Discrete spatial data such as points in a multi-dimensional space can be modelled using traditional techniques from relational DBMSs. In particular coordinate values of the point can be treated as additional attributes in a tuple. On the other hand data such as lines and regions are continuous. Continuous data are data that span a region in space or time. In other words, the attribute value holds more than just one point or instance of time.

Spatial data presented special problems for computing in recent years. This is because spatial data are relatively more complex than traditional business data, and the traditional constructs are not adequate for handling it [Worboys, 1995], which makes it difficult to define and represent the data within a database. The more challenging aspect is querying of spatial data. For example in traditional Relational Database Management Systems (RDBMS), such as Business Information Systems, the queries are relatively simple. Queries like *"select the top 5 sales outlets"* are common in systems like these. These queries only require a B-Tree index lookup within the database, which has very little overhead because there are few calculations carried out to process them [Comer, 1979]. On the other hand spatial queries can be very complex. For example, queries like determining a polygon within which some location lies, and locating islands (internal polygons) within other polygons need to be performed.

Spatial data are significantly different from non-spatial data, but the differences are not always obvious and the structure of some intuitively obvious spatial operations can be surprisingly complex. It is the integration of these operations with VR that makes this research project innovative in terms of combining already available technology to create a novel platform that delivers worth well beyond what individual components acting separately can achieve.

1.4 Need for Research

In addition to addressing the core issue of user-centred interface design, the *CHI project* looks at areas of data and metadata schemas that are necessary for the creation of a 'semantic web,' without which an ambient intelligent environment cannot function.

Cultural heritage, defined both as the rich repository of content held by Ireland's leading memory organisations, as well as contemporary creative cultural activity, is now widely recognised both nationally and at a European level as a sector of strategic importance for digital media and informatics research and technology development. The preservation, re-use and access to intellectual capital which cultural heritage represents, is a cornerstone of future economic growth and development. In this context, the intersection between software, technology and cultural products has been a focus of significance for cultural heritage industries.

1.5 Significance of Study

The research offers a novel technological approach to accessing digital repositories of cultural heritage that integrates a toolset application with specific cultural content. It proposes a user-defined solution, based on research into spatial navigation in a cultural heritage context. The project offers an opportunity to develop an integral relationship between the use of virtual reality devices and the physical artefacts on which they are based by making the navigation of digital cultural heritage richer in content and more meaningful to different classes of user.

The long-term direction of R&D in the cultural heritage area is towards the realisation of the concept of a 'Cultural Portal'. A Cultural Portal harnesses the power of computers and the Internet to deliver very rich and valuable cultural knowledge and experiences. It combines two different concepts. One is the multimedia cultural database. This is a repository of cultural information that is a virtual gallery, museum, library and music centre that stores digital representations of objects, images, books, pieces of music, information texts and ephemera drawn from various different collections. Freed from the constraints of space and time the 'digital collection' can offer a degree of public access to a vast range of objects that would be impossible with any physical collection – and without any risk to fragile artefacts. Items and pieces of information from the 'digital collection' can be selected and presented in almost infinitely variable ways to support learning (at every level from elementary study to scholarly research), cultural tourism, and the cultural economy.

1.6 Research Objectives

The primary objective of this project is to research a methodology for the use of spatial navigation in hypermedia and demonstrate its applicability to the extraction and presentation of information from a cultural heritage database in ways appropriate to different classes of users. Using a database application and a management toolset, the project will develop a specification for the use of multimedia techniques and hypermedia rules to deliver relevant spatial local data to users through a Virtual Reality (VR) browser.

The project builds on ongoing research into virtual representations of cultural heritage and develops a new area of research into the use of spatial data to make the navigation experience more meaningful and rewarding. The project will contribute to research that aims to improve the functionality and usability of Information Communication Technologies (ICT) in cultural heritage sites and will enhance user-centred interactivity by making the location of the user a central element in the delivery of rich multimedia content.

The research has wide applications in the cultural, tourism and heritage industries. It is also relevant to emerging mobile computing and telecommunications devices and mobile applications capable of exploiting spatial navigation. The research will contribute to the formation of standards and protocols concerning spatialised hypermedia, such as XML and RDF data types that describe spatialised content and the mark-up of spatial data. The objectives of the research are:

- To research the spatio-temporal navigation of cultural heritage spaces.
- To examine database schemas and structures for the cataloguing and storage of cultural information.
- To identify possible interfaces, methods and schemas for mapping generalised nodal hypermedia to real and virtual space.
- To identify methods of navigation of a cultural database to users in a hypermedia environment.
- To implement a demonstrator of these navigational metaphors.
- To develop a software toolset for generating spatialised interfaces and linking them to an appropriate DBMS.
- To disseminate the demonstrator, findings and published papers to a wider research community.

1.7 Thesis Layout

This Thesis is divided into six chapters. The current chapter, Chapter 1 provides an introduction to the subject area of this research and provides justification for the topics investigated as well as giving a brief overview of the dissertation.

Chapter 2 is comprised of the literature review. In section 2.1, a general introduction to the main aspects of spatial databases is presented. This includes a brief description of the different components that are required in the architecture of a spatial database system. Section 2.2 gives an introduction to Desktop Virtual Environments (DVE) and Virtual Reality. In section 2.3, a detailed study of the different types of Database Management Systems (DBMS) that are currently available is given. The main designs and functions of the architectures are examined in detail, in particular the ability for each to store and process spatial data.

Section 2.4 focuses on spatial data models for spatial database systems. In particular the object-based model is discussed. The object-based model treats the world as a surface (embedding space) with discrete objects scattered around it. This section carries on to describe the specification for spatial data types defined by the OpenGIS (OGIS). These standards identify a set of spatial data types and the set of operations that can be applied to these data types.

In section 2.5, the Structured Query Language (SQL) is examined. The need for an extension to SQL is addressed. This is followed by a detailed explanation of the OGIS standard for extending SQL that enables it to accommodate spatial data types.

In order to query spatial data, there has to be present some kind of spatial access methods to index the spatial data. To ensure that the most efficient and cost effective spatial access methods are used in the system, a detailed insight into spatial indexing methods is needed. An overview of the literature on spatial index methods is presented in section 2.6.

In section 2.7, the area of query processing is covered; it begins with a general overview of querying standard data types. The discussion moves on to Spatial Query Processing and the different methods used to query data spatially. The concept of direction relations is discussed in section 2.8; these relations enable spatial database systems to process directional queries. Initially, the different types of direction are discussed. This is followed by a discussion of the two main types of directions namely, projection-based and cone-based directions. The most successful implementations of these directions are then explained.

Chapter 3 is an analysis of the literature review. In this section the technologies discussed in the literature review are analysed. This is followed by an explanation of the selected technologies for the *CHI project* and the synthesis of the conceptual design.

In Chapter 4, the design and development of the CHI framework is described. This includes a design of the *Initial Technology Demonstrator*. The description covers the three different aspects of the framework, namely, the client layer, the middle layer and the database layer all which are parts of the three-tier architecture used by the system. There is also the design and implementation detail of the initial database layer. This is followed by an expert internal evaluation of the system. The results of this evaluation lead into the design and implementation of the *Revised Technology Demonstrator*. This redesign incorporates a number of changes to the initial approach. This is followed by the details of the integration of all data into one central database. An extensive description of the database layer is described, as this is the main focus of the research. Following the description of the revised system set-up, an overview of the integration of the directional query methods into the *Revised Technology Demonstrator* is detailed. This includes the different techniques that are implemented to query the database using Direction and Line-of Sight algorithms.

To evaluate the *Revised Technology Demonstrator* implementation of the *CHI system*, Chapter 5 describes the details of an Internal Evaluation carried out by members of the research team. This involved performing an Internal Technical Evaluation and an Internal Usability Evaluation. This is followed by a description of an External Evaluation that involves performing an External Technical Evaluation and an External Usability Evaluation, which consists of peer reviews and a questionnaire. This includes the results and analysis of the overall findings.

Chapter 6 contains the final conclusions of the overall dissertation. This includes the overall finding of the project. Finally, recommendations for further research in the area are given.

Chapter Two

Review of Literature

2 Review of Literature

2.1 Introduction

This chapter describes the literature review and is divided into 8 sections. These sections range from the basics of database technology to the advanced aspects of query processing that can be achieved using the available underlying database framework that is an integral part of every type of Database Management System (DBMS) that complies with the Relational Database Model [Codd, 1970].

In the current section, a detailed layout of the literature review is provided, giving a brief description of the topics covered in this section.

In section 2.2, there is a definition of Virtual Reality (VR). This includes a description of Desktop Virtual Environments (DVE) as an alternative user interface and in particular 3D models. Following this, the current technologies used to build this kind of DVE are described.

In section 2.3, there is a comprehensive introduction to currently available Database Management Systems (DBMS). This section also gives an overview of DBMSs that are capable of storing and processing spatial data either by using a loosely coupled architecture or an integrated architecture.

In section 2.4 the techniques related to spatial data modelling of spatial database applications are discussed. These include the Field-Based and Object-Based-Models that are currently used. An introduction to Spatial Data Types (SDT) and the OpenGIS Specification for SDTs follows. Next, there is an analysis of embedding spaces and lastly in this section the 9-intersection matrix is discussed describing the operations that are used to describe how spatial objects interact topologically.

Section 2.5 is dedicated to the Structured Query Language (SQL). This includes initially an introduction to SQL and thereafter an explanation of the OpenGIS extension to SQL that enables the DBMS to store and process spatial data. Each add-on mentioned in section 2.3 must adhere to these standards for interoperability purposes between the different implementations of the standards.

In section 2.6 the discussion moves on to the indexing structures that are used to index Spatial Data Types (SDT). This includes the two main types of spatial indexes namely Space Driven Structures that include Quad-Trees and Data Driven Structures that include R-Trees. The different implementations of each type of structure are described and the strengths and weaknesses associated with each are outlined.

Section 2.7 discusses the more focused area of Query Processing. The initial paragraphs look at relational query processing in general. Following that the focus gets narrower and the area of Spatial Query Processing is discussed. This section describes the different spatial operations and their uses in querying spatial data at a 2D level.

The final section of the literature review examines the specialised area of directional query processing in a spatial environment. The discussion is focused on how direction relations constitute an important class of user queries in a spatial database environment. The discussion then elaborates on this subject to concentrate on user-based queries that take into account the position of the user in a 3D environment. Finally, a discussion of the different frames of reference that can be taken is given.

2.2 Virtual Reality (VR)

In this section, the concept of Virtual Reality (VR) is discussed for use as a Desktop Virtual Environment (DVE) that is proposed as a user interface to simulate a real world environment. This includes a definition of VR and its background. Following this, a description of the Virtual Reality Modelling Language (VRML) is given.

The concept of VR is frequently associated with immersive virtual environments that require expensive hardware such as head-mounted displays and data gloves [Dalgarno and Scott, 1999]. However, these connections are becoming less visible with the emergence of standard desktop computer hardware that is capable of delivering the attributes of such immersive environments without specialised hardware on a desktop computer [Kelty et al, 1999].

It is because of the emergence of such VR technologies that require less and less customised hardware and software to run that makes the idea of the Desktop Virtual Environments (DVE) a possibility as alternative user interfaces. This type of interface has potential and can be applied in many application domains [Leach et al, 1997].

A common approach to developing a DVE is to use a 3D model that acts as the user interface [Lui et al, 1998]. It has been suggested by [Robertson et al, 1993] that these types of environments can be easier to use than their counterparts (immersive environments) because people are already familiar with the desktop computer.

Dalgarno asserts [Dalgarno, 2002] that a 3D environment has the following four principal characteristics:

- The environment is modelled using 3D vector geometry, meaning that objects are represented using x, y and z coordinates describing their shape and position in 3D space.

- The user's view of the environment is rendered dynamically according to their current position in 3D space, that is, the user has the ability to move freely through the environment and their view is updated as they move.

- At least some of the objects within the environment respond to user action, for example, a door might open when approached and information may be displayed when an object is selected with a mouse.

- Some environments include 3D audio, that is, audio that appears to be emitted from a source at a particular location within the environment. The volume of sound played from each speaker depends on the position and orientation of the user within the environment.

Due to the proliferation and development of the Internet, it is possible that these types of 3D environments can become distributed, and can be explored from desktop computers at many locations, making then easily accessible

There are a number of modelling techniques currently available that enable users to navigate 3D environments over the Internet, the most widely known are OpenGL and VRML. The following sections outline the main properties of each.

2.2.1 Open Graphics Language (OpenGL)

Open Graphics Language (OpenGL) is a graphics library standard used for the presentation of 2D and 3D visual data. It is currently the most widely adopted graphics standard and requires OpenGL API-compliant hardware to produce a consistent visual display. It is typically programmed in C, C++ or Java environments.

2.2.2 Virtual Reality Modelling Language (VRML)

The Virtual Reality Modelling Language (VRML) has significant potential as a platform for the development of virtual environments because of its separation of the navigation tools from the environment itself [Dalgarno, 2002]. The standardisation of the language enables virtual environments to be developed which are independent of navigation tools. Navigation tools become secondary in this approach and are developed as generic software tools, called VRML browsers that can be used to explore any VRML model. This makes VRML a very suitable technology for viewing 3D data over the web because the browsers can be downloaded automatically to display the VRML scene. The most common types of VRML browsers currently available are Cosmo Player [http://www.ca.com/cosmo], Cortona [http://www.parallelgraphics.com] and Blaxxuan [http://www.blaxxun.com].

2.2.3 Virtual Databases

To store the geometric description of a Virtual Environment (VE) a 3D database is required that describes it [Vince, 1995]. The data used to build virtual objects is specified by the designers of the virtual environment. The database is used to store and build the objects in the model as well as supplying information about the objects in the VE. A 3D interface used to display the data interfaces with the database in real-time to obtain the relevant data and information [Larijani, 1994]. In this way, all data relating to a particular 3D model can be stored in a database and rendered into a range of different formats. A Database Management System (DBMS) can be used to do this.

2.3 Database Management System (DBMS)

A database is a shared collection of logically related data (and a description of this data) that is stored in a structured repository [Connelly and Begg, 1998]. The database is stored within a computer environment where the data is persistent, which means that it survives unexpected hardware and software problems. Traditional applications of databases include management of stocks, travel reservations, and banking. Other non-standard applications have emerged in the last two decades, among them spatial databases, computer-aided design and manufacturing (CAD/CAM), software engineering and bio-informatics [Rigaux *et al,* 2002].

A Database Management System (DBMS) is a collection of software that manages the database structure and controls access to the storage, retrieval and manipulation of information through a Data Manipulation Language (DML) [Codd, 1970]. In general, a DBMS aids the process of:

- Defining a database – specifying the data types, structures and constraints to be taken into account.
- Constructing the database – storing the data itself into persistent storage.
- Manipulating the database – changing the database structure.
- Querying the database to retrieve specific data.
- Updating the database (changing the already existent values).

The structure of a database system is illustrated in Figure 2.0. It can be observed from the diagram that the only way to access the data is through the DBMS software, which includes initially the query processing software, which in turn has access to the data access software. This layer then has access to the physical data on disk. This is an example of how data independence is achieved at the physical level [Elmasri and Navathe, 1994].

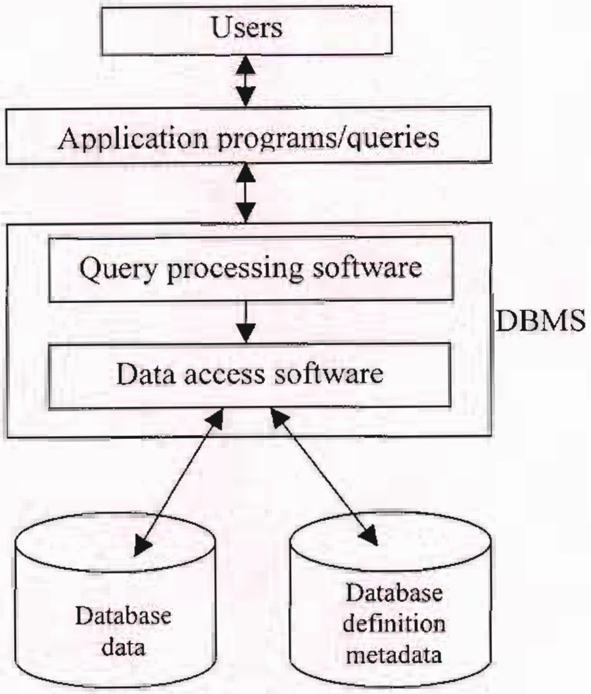

Users
Application programs/queries
Query processing software
Data access software
DBMS
Database data
Database definition metadata

Figure 2.0: A database system environment

This type of architecture is common to most database management systems [Rigaux et al, 2002] and in the next section there is an overview of the currently available types of DBMS and, where appropriate, their uses with regard to spatial data storage and manipulation are discussed.

2.3.1 Hierarchical Database Management System

One of the initial database architectures was the hierarchical database management system. The hierarchical data model is organised in a hierarchical manner, where the main relationships between nodes in the hierarchical structure of the database are one-to-many relationships. This means that each child node has only one parent and each parent can have many children. The architecture was capable of storing and manipulating data in this manner, but was not efficient when dealing with many-to-many relationships [Elmasri and Navathe, 1994, p.346].

This architecture offered centralised data control and other much-desired properties such as redundancy and concurrency control, but also had major shortcomings that caused great difficulty for users because of the limited and primitive commands used to navigate the hierarchical structure [Yu and Meng, 1994]. These problems prompted some research into interfacing the databases with relational database management, and using relational query languages, to query the structure [Chung and McCloskey, 1993]. Two of the major hierarchical database management systems were IBM’s IMS database system and System 2000.

The introduction of the relational data model was a major advance for database technology because, unlike other systems, it was easy to use and data could be retrieved using a declarative query language that instructed the system what to retrieve not how to retrieve it [Atzeni et al, 1999]. This technology paved the way for many future DBMSs.

2.3.2 Relational Database Management System (RDBMS)

E. F. Codd at IBM invented the Relational database Management System (RDBMS) in 1970 [Codd, 1970]. A RDBMS enables users to store related pieces of data in two-dimensional data-structures called tables (relations). This data may consist of many defined types, such as integers, floating-point numbers, character strings, and timestamps. Data inserted into the table can be categorized using a grid-like system of vertical columns, and horizontal rows (tuples). The relational model was built on a strong premise of conceptual simplicity, which arguably is both its most prominent strength and weakness

[Worsley and Drake, 2001]. Examples of RDBMSs are Oracle 9i, IBM Informix and Microsoft Access.

A Relational Database Management System (RDBMS) should satisfy Codd's 12 rules [Codd, 1985a; Codd, 1985b] but in practice there are few DBMSs that satisfy all these rules [Connolly and Begg, 1998]. In fact, some successful DBMS vendors consider their DBMSs to be relational even though they have implemented only a few relational features. Traditional DBMSs based on the network and hierarchical data models are two examples of two types of DBMS that claim to be in some way relational without implementing all the rules.

In the last two decades there has been an increasing need to store spatial data in an RDBMS. Spatial data is any data that is referenced by location. To handle this complex spatial data, a first approach was to use the Relational DBMS [Shekhar et al, 1999a]. While the Relational DBMS is good at handling topological relationships between data, an issue lies with its ability to handle the relationships between hierarchical sets of data that do not use indices. A topological relationship is a relationship that stays invariant under topological transformations such as translation, rotation and scaling [Papadias et al, 1995]. This meant that RDBMSs were incapable of handling spatial data because spatial data have a structure that does not naturally fit with tabular structures [Worboys, 1995, p.67]. For example, to represent a *polyline* in a relational database, would not be natural because *polyline* is not a built in data type. One solution to the problem is to create a set of overlapping linked tables each with a vertex coordinate to represent the *polyline* [Shekhar and Chawla, 2003, p.7]. Although this works it is not an efficient solution to the problem and the key point is that this type of mapping is not natural in a RDBMS environment.

2.3.3 Object-Oriented Database Management Systems (OODBMS)

Some commercial examples of OODBMS are O2 and Poet. Object-Oriented Database Management Systems (OODBMS) use a more advanced approach that extends the concepts of DBMS with characteristics of the object-oriented paradigm. This object-oriented approach evolved with the merger of database systems and object-oriented programming languages and is essentially an object-oriented programming language with persistence that offers transaction management, concurrency control and recovery mechanisms [Bancilhon and Buneman, 1990]. A persistent programming language gives the user the ability to preserve data across successive executions of a program that can exist beyond the lifetime of the program that created it [Connolly and Begg, 1998]. This object-oriented approach is better at processing topological relationships. It also performs very well in the area of hierarchical relationships between data, as it is based on an object-oriented model but its weakness is on its ability to deal with phenomena that are essentially continuous in space, for example, terrain data or population density distribution [Rigaux et al, 2002; Shekhar and Chawla, 2003]. Another constraint with object databases is that SQL is a query language that is tightly coupled with the Relational Database model. This means that is the most widely used query language and is integrated into many systems. SQL is a declarative language, meaning that it only specifies the desired result and not the algorithm to get the result; it's the database system that must determine the series of steps to obtain the required result [Shekhar and Chawla, 2003].

2.3.4 Geographic Information Systems (GIS)

Geographic Information Systems (GIS) are systems specifically designed to handle geo-spatial data, e.g. for the analysis and visualisation of geographic data. Geographic data is spatial data whose primary frame of reference is the earth's surface. GISs are designed to handle spatial data and provide a rich set of operations that can be applied to only a few objects and layers [Burrough and McDonnell, 1998]. For example, a county (Dublin) is an object and is part of the counties layer. Given the county boundary, a GIS can determine the neighbouring counties to Dublin in the layer.

These systems are primarily used for the analysis and visualisation of layer data based on the earth's surface. They are usually based on a loosely coupled architecture in which the geometric attribute data is handled by a different module than the non-spatial data. Due to the complexity of these systems expert knowledge is needed to formulate queries for these systems. A commercial example of a GIS is ArcInfo by ERSI [ESRI, 2003].

2.3.5 Spatial Database Management Systems (SDBMS)

A Spatial Database System offers Spatial Data Types (SDTs) in its data model and query language. It also supports spatial data types in its implementation, providing at least spatial indexing and efficient algorithms for spatial joins [Schnieder, 1997]. Spatial joins are queries that involve comparing spatial data from different tables that may be defined by different data types. For example, a query to determine how many Cultural Heritage (CH) points there are in Dublin city centre would require a spatial join, joining the CH table and the Dublin table. In this case the CH relation contains *Points* and the Dublin relation contains *Polygons*.

In contrast to a GIS, Spatial Database Management Systems (SDBMS) provide simple operations on a set of objects and layers [Shekhar and Chawla, 2003]. This means that set-based queries can be answered. Set-based queries use the set operations Union, Difference, Intersection and Cross-Product [Schneider, 1997] and as a relation (i.e. table) is fundamentally a set, these operations can be applied to relations providing the relations are union-compatible, i.e. if they have the same number of columns, share the same domain and their columns are in the same order.

SDBMS are designed to handle very large amounts of spatial data by using specialised indices and query processing techniques. They also inherit traditional DBMS functionality in providing database security and a concurrency control mechanism that allows multiple users to simultaneously access shared data, while preserving data consistency.

2.3.6 Distributed Database Management System (DDBMS)

Currently there is much research being carried out in the field of database systems [Kroenke, 2003]. With each type of DBMS, there is a need to have data from different locations combined together to provide a complete set. This may be for data interrelation purposes such as the interfacing of databases with heterogeneous data structures or for backup purposes, where the same homogeneous database may be at different locations in case of a database failure. In each case it is called a distributed database [Elmasri and Navathe, 1994, p.703].

A Distributed Database (DDB) is a logically interrelated collection of shared data, physically distributed across multiple servers connected by a network. A Distributed Database Management System (DDBMS) is the software system that facilitates the management of a DDB in such a way that the distribution aspects are transparent to the user.

Some advantages of this type of architecture are the shareability of the data, and the availability and reliability of the system. This approach also improves the organisational structure and the overall performance [Connolly and Begg, 1998]. Some disadvantages to DDBMSs are the complexity of providing seamless distribution, the security of the systems and the lack of standards.

2.4 Spatial Data Models

In this section, the concept of data models is discussed. Initially, the definition of a model is given followed by a definition of a data model, and then the different types of spatial data models are explained.

A model is an artificial construction in which parts of one domain (source domain) are represented in another domain (target domain) [Worboys, 1995]. A structure-preserving function is used to transform from one domain to the other in a process called morphism. The constituents of the source domain may be entities, relationships, procedures or any other phenomena of interest. A data model is defined as an abstraction of real world data that incorporates only those properties of the data considered relevant to the application at hand. It uses logical concepts that may be easier for most users to understand and it supports data input, manipulation and result presentation [Shekhar *et al*, 1997]. A spatial data model of information is therefore a data model with spatial components.

Spatial information may be modelled using either field-based or object-based spatial data modelling techniques. These are methods of applying structures and patterns on spatial data for convenience of operations. First the field-based approach is explained followed by the object-based approach.[1]. In Figure 2.1 a map describing the distribution of area covered by different types of spatial phenomena in an urban area is shown. This map is used to describe the two models and their representation of space.

[1] As the object-based approach is the one that is used throughout the thesis it is explained in more detail than the field-based approach.

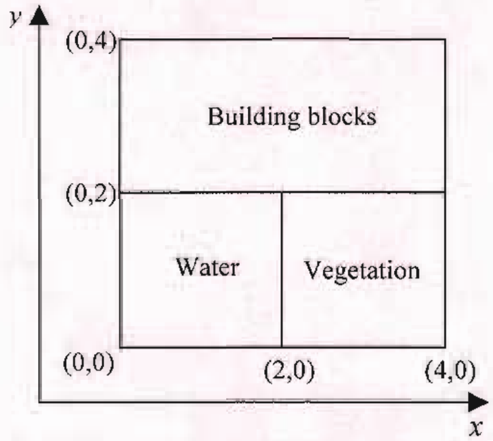


Figure 2.1: Typical distribution of blocks, water and vegetation

2.4.1 Field-Based Model

The field-based spatial data modelling approach treats the information space as a collection of fields where each field is a function from a spatial framework to a finite attribute domain. This model perceives the world as a continuous layer (surface) over which features (e.g., elevation) may vary. The field-based model uses layer algebra, which is a set of operations that can manipulate many layers to produce new layers [Delis *et al*, 1994]. In this approach each point on the layer has an associated value. An example of field-based data would be rainfall or vegetation distribution. This is illustrated by an area chart in Figure 2.1 and is represented by a field-based function in Figure 2.2b.

2.4.2 Object-Based Model

The object-based spatial data modelling approach treats the information space as being populated by discrete, identifiable and relevant entities, each of which is referenced to a given coordinate system and exist independent of their locations [Shekhar *et al*, 1997]. For example, a city can be characterized by the building blocks, water and vegetation that describe it. These elements are clearly recognisable as distinct objects where each object has a set of attributes that describe it (Figure 2.22a).

The difference between objects in the context of traditional database systems and objects in the object model is that the attributes in the object model fall into two categories, namely non-spatial and spatial. The non-spatial attribute, or alphanumeric attribute, is like the name of a building, park or river. The spatial attribute is the attribute of each object that interacts with the underlying coordinated (embedding) space. In the city example, the city boundary is the coordinated space; the polygon that represents the building block is the spatial attribute of the spatial object that interacts with that space and the object ID is the non-spatial attribute. The number of spatial attributes that can be associated is not just limited to one. A spatial object may have many spatial attributes associated with it, just as it is possible to have many non-spatial attributes. One example would be for generalisation or scaling purposes. In the city example, a map represented by a set of spatial objects could be viewed at different scales. At a small scale, a road would be represented as a line and at a large scale; the same road would be represented as a polygon. This spatial record would have two spatial attributes associated with it, one to represent the line and another to represent the polygon.

Area-ID	Coverage type	Boundary
A1	Block	[(0,2),(4,2),(4,4),(0,4)]
A2	Water	[(0,0),(2,0),(2,2),(0,2)]
A3	Vegetation	[(2,0),(4,0),(4,2),(2,2)]

(a)

$$f(x,y) = \begin{cases} \text{“Block”}\ 2 \leq x \leq 4;\ 2 < y \leq 4 \\ \text{“Water”}\ 0 \leq x \leq 2;\ 0 \leq y \leq 2 \\ \text{“Veg”}\ 2 < x \leq 4;\ 0 < y \leq 2 \end{cases}$$

(b)

Figure 2.2: (a) Object-based representation (b) Field-based representation

Figure 2.22(a) illustrates an object-based viewpoint of the map shown in Figure 2.1 as a collection of three objects. Each object has Area ID, Coverage type and Boundary attributes associated with it where its coordinates represent the boundary of the area covered, which is a polygon. Figure 2.22(b) illustrates a field-based viewpoint, where each point is mapped to the value of the relevant coverage type for each phenomenon.

2.4.3 Spatial Data Types (SDT)

The previous section introduced spatial data models. These models are used to extend database technology to meet the requirements of spatial applications. Spatial database systems represent spatial data as objects in space with attributes that define them. These systems have additional capabilities for representing, querying and manipulating spatial objects (Examples are shown in Figure 2.3). To achieve this, special data types called Spatial Data Types (SDT) are used. SDTs provide a basic abstraction for modelling geometric objects as well as their properties and operations. Consequently, there have been a number of proprietary implementations of SDTs and some formal definitions have been put forward [Schneider, 1997, pp.20-33]. The most widely accepted of these is the OGIS specification for spatial data types.

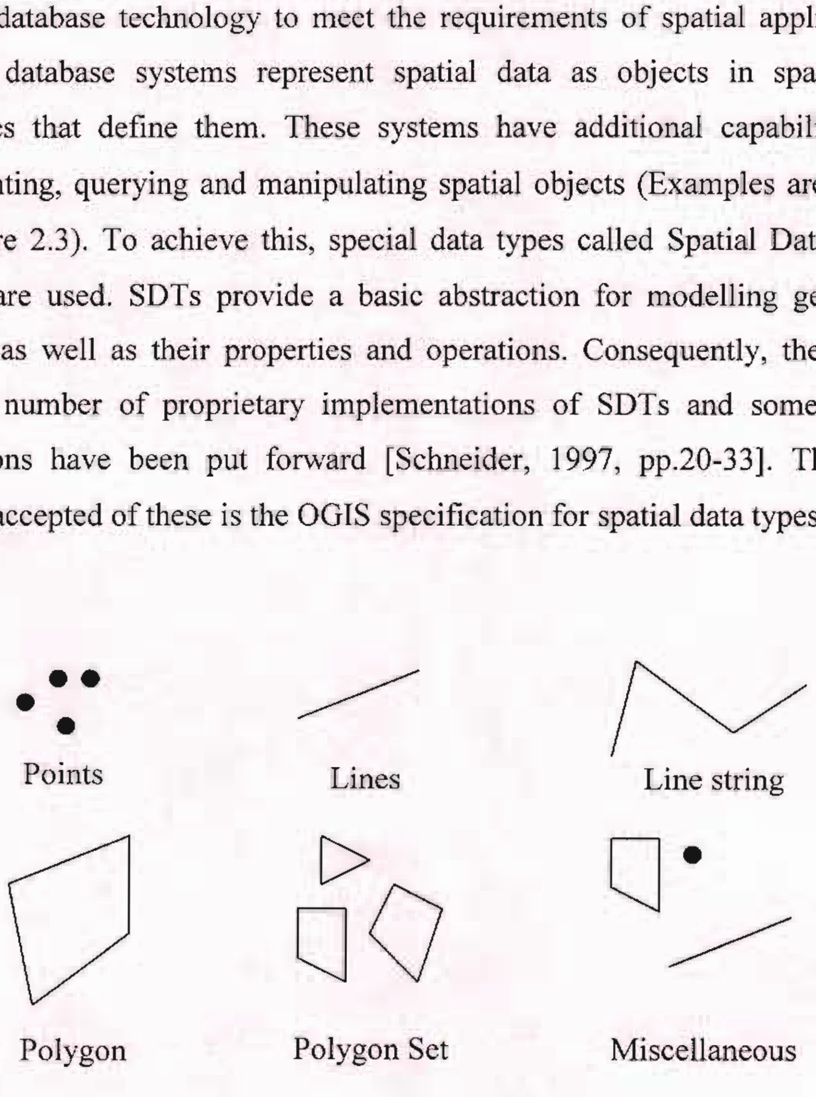


Figure 2.3: Examples of Spatial Objects

2.4.3.1 OGIS Specification for Spatial Data Types

There have been many proposals over the years for a standard set of spatial data types and operations. The OGIS standard [OGIS, 1999] describes the fundamental building blocks of two-dimensional geometries and their interrelationships. A brief description, using UML notation, of the Geometry class hierarchy is given in Figure 2.4

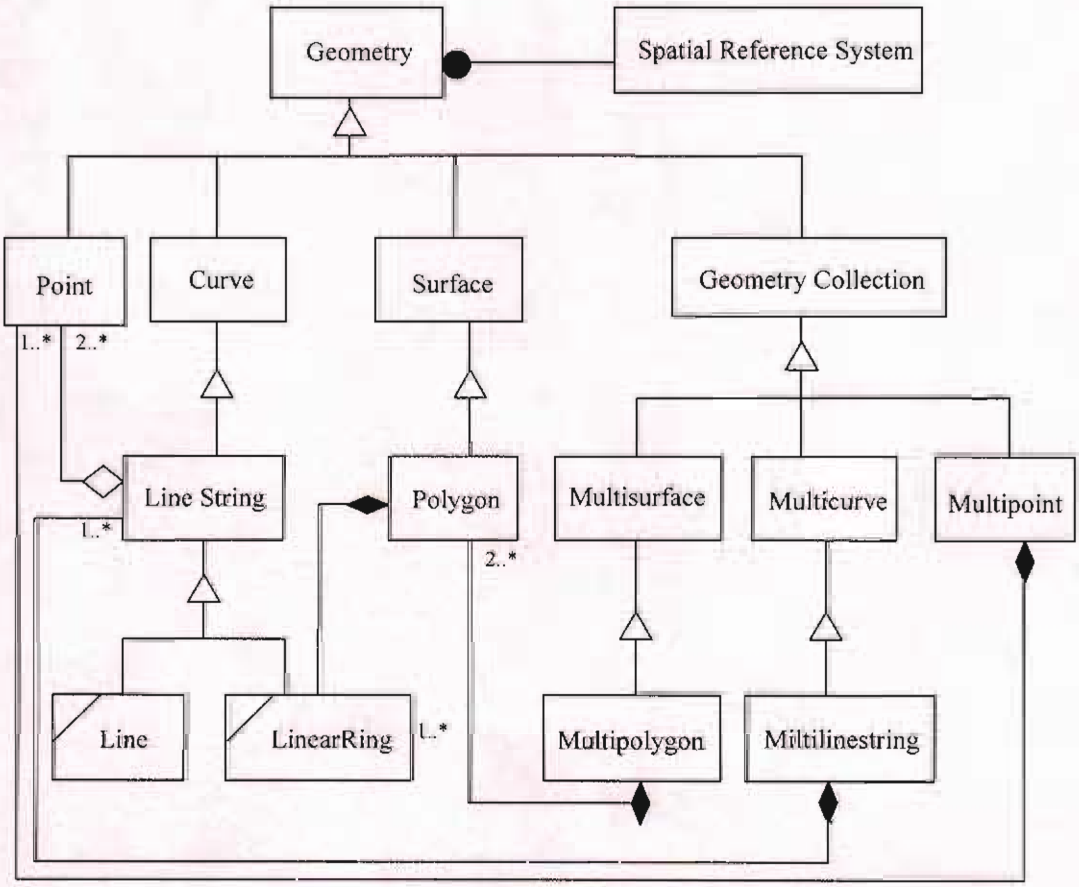


Figure 2.4: UML Representation of OGIS Specification for spatial geometry building Blocks [OGIS, 1999]

This specification describes a standard set of geometry types based on the **OpenGIS Geometry** Model, together with the functions on these types. The **base *Geometry* class** in the specification has four sub-classes. These are *Point*, *Curve*, *Surface* and *Geometry Collection*. Each of these geometric objects has associated with it a *Spatial Reference System* (SRS). This SRS describes the coordinate space in which the object is defined. *Point* describes the shape of a 0-d object such as a statue on a map. *Curve* describes a 1-d object such as a river or a road in a road network. A curve may be represented as a *Linestring*, which are two or more points connected with straight lines. *Surface* describes a 2-d object such as the outline of a building or park without any interior boundaries. A surface may also be represented by a *Polygon*. The fourth sub-class is *GeometryCollection* and is used to represent complex collections of geometries such as a collection of Cultural Heritage (CH) artefacts in a city. *GeometryCollection* consists of three-types; these are *Multipoint*, *Multicurve* and *Multisurface*. *Multipolygon* and *Multilinestring* are used to represent generalisations of *Multisurface* and *Multicurve* by modelling them using collections of *Polygon* and *LineString* objects [OGIS, 1999].

2.4.4 Operations on Spatial Objects

In order to display, process or analyse spatial information there needs to be some set of operations that can be applied to the spatial data sets that will identify the operands of the interaction and also the type of interaction that is occurring between the objects. Therefore, in [Egenhofer and Franzosa, 1991, pp.1-3] a formal and sound method to describe spatial relations was proposed.

In object-based modelling the types of relationships that may exist between objects are determined by the underlying embedding space [Worboys, 1995, pp.163-177; Shekhar and Chawla, 2003, p.27]. The operations that are allowed on these objects are defined for each particular embedding space. There are many types of embedding spaces in which these operations may be defined and this section outlines the principal ones.

2.4.4.1 Euclidean Space

The basis for modelling many spatial occurrences is an embedding in a coordinated space that enables measurements of distances and bearings between points [Worboys, 1995, p.98]. This coordinated model of space is called Euclidean space. In Euclidean space spatial properties are transformed into tuples of real numbers and all spatial relationships, including set, topological, metric, and directional (north/south), can be defined [Shekhar and Chawla, 2003, p.31].

2.4.4.2 Set-Oriented

Set-oriented embedding space is the simplest and most general of all embedding spaces [Shekhar and Chawla, 2003, p.27]. This is because sets are the simplest form for describing the interaction between objects. Sets are very abstract and provide very little in the way of constructs for modelling spatial properties and relationships but are fundamental in the modelling of spatial information systems [Worboys, 1995, p.104]. This will become apparent in the section on Topological space.

2.4.4.3 Topological

A topological property is one that is preserved by topological transformations of the space. To gain an understanding of what topological space is, imagine the Euclidean plane to be a sheet of rubber. On this sheet there are drawn two polygons that touch (meet) each other. If the sheet is then stretched or bent without cutting or folding it, then certain properties of the two polygons will be lost, while others will be invariant (or unchanged). For example, the area of the two polygons will change but the adjacency of the polygons does not change. This property of adjacency is a topological property whereas the area of the polygon is not as it has changed. The transformation caused by stretching the rubber is a called a topological transformation. Therefore, topology is the study of topological transformations and the properties that are left invariant by them [Worboys, 1995, p.111].

Topological relations can be described by the four intersections of the boundaries *b* and interiors *i* of two point sets of two spatial regions. With the help of an intersection matrix introduced by [Egenhofer and Franzosa, 1991, pp.8-14] it is possible to numerically evaluate clearly topological relationships (See Figure 2.5). This approach has sixteen possible intersections based upon the criteria of empty and non-empty intersections of boundaries and interiors. The same framework is also used for spatial regions, however not all of the sixteen relations between arbitrary point-sets exist between two spatial regions.

2.4.4.3.1 4-Intersection Matrix

It has been shown that spatial relationships can be derived automatically between pairs of simply connected polygons (regions), i.e. regions without holes, by determining their 4 intersection relations between their respective borders and interiors [Egenhofer and Franzosa, 1991, pp.10]. More specifically, a 2x2 matrix (Figure 2.5) is generated through the determination of whether the border *b* or interior *i* of region A intersects ($\bigcap$) either the border or interior of region B.

$$\begin{pmatrix} i(A)\bigcap i(B) & i(A)\bigcap b(B) \\ b(A)\bigcap i(B) & b(A)\bigcap b(B) \end{pmatrix}$$

Figure 2.5: 4-Intersection Matrix

The values for the entries in the 4-intersection matrix can be either empty (0) or non-empty (1) thereby giving 2^4=16 possible combinations of topological relations between 2 simply connected regions. Of these 16 possible combinations however, 8 are considered impossible to achieve. For example, the relation in Figure 2.6,

$$\begin{pmatrix} i(A) \cap i(B) & i(A) \neg\cap b(B) \\ b(A) \neg\cap i(B) & b(A) \neg\cap b(B) \end{pmatrix}$$

Figure 2.6: Inconsistent 4 Intersection Matrix

states that the boundary of region A doesn't intersect ($\neg\cap$) the boundary or interior of region B and the interior of region A doesn't intersect the boundary of region B. So far all is consistent, as this relationship expresses 2 regions that are disjoint (Figure 2.7).

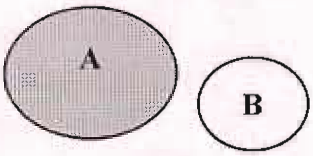


Figure 2.7: 2 Simply Connected Disjoint Regions

However, the final relation that, i(A) intersects ($\cap$) i(B) is not consistent given the 1st three relations. Therefore this 4-intersection matrix is impossible to achieve in practice. Considering that there are seven other inconsistent relations similar to the above example, one is left with only 8 possible relations.

2.4.4.3.2 9-Intersection Matrix

There are cases in which the 4-intersection method fails to distinguish between differing configurations of objects, differing at least from a human observer's perspective. For example, the cases illustrated in Figure 2.8 all have identical 4- intersection matrices [Egenhofer et al, 1993].

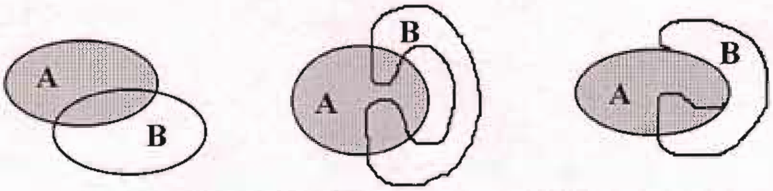


Figure 2.8: Topologically Distinct Configurations Having Same 4-Intersection Matrix

This problem led to the addition of a third term to the matrix, namely "exterior" (denoted by $^-$) intersection resulting in a 3x3 matrix of topological relations describing the 9-intersection method (see Figure 2.9) [Egenhofer and Herring, 1991].

$$\begin{pmatrix} i(A)\cap i(B) & i(A)\cap b(B) & i(A)\cap B^- \\ b(A)\cap i(B) & b(A)\cap b(B) & b(A)\cap B^- \\ A^-\cap i(B) & A^-\cap b(B) & A^-\cap B^- \end{pmatrix}$$

Figure 2.9 - 9 Intersection Matrix

In this matrix there are 2^9=512 possible relations amongst a single pair of objects. Allowing for the impossible cases, as in the 4 intersection method, one is left with 8 possible matrices describing area/area relations, 19 for line/area, 3 for point/area, 33 for line/line, 3 for point/line, and 2 for point/point.

Therefore, the spatial relationships that exist between objects include (see Figure 2.10):

- *Disjoint* – the boundaries and interiors of two geometries (objects) do not intersect.
- *Contains* – the interior and boundary of one object is completely contained in the interior of the other object.
- *Inside* – the opposite of Contains.

- *Touch* – the boundaries intersect but the interiors do not intersect.
- *Equal* – the two objects have the same boundary and interior.
- *Covers* – the interior of one object is completely contained in the interior of the other object and their boundaries intersect.
- *Covered by* – the opposite of Covers.
- *Overlap by intersect* – the boundaries and interiors of the two objects intersect.

In Figure 2.10, there is a graphical representation of the eight possible topological operations and directly below each one there is the corresponding matrix representation of the operations described with their equivalent set notation descriptions. In the diagram (A°) is the interior, (ƏA) is the boundary and (A^{-}) represents the exterior of object A.

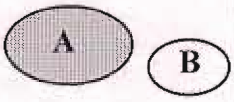
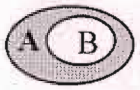
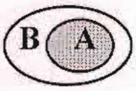
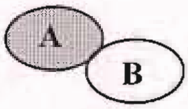
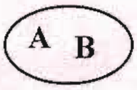
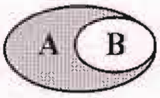
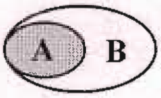
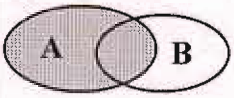
Disjoint	Contains	Inside	Meets
			
$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 0 & 0 & 1 \\ \partial A & 0 & 0 & 1 \\ A^- & 1 & 1 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 1 & 1 \\ \partial A & 0 & 0 & 1 \\ A^- & 0 & 0 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 0 & 0 \\ \partial A & 1 & 0 & 0 \\ A^- & 1 & 1 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 0 & 0 \\ \partial A & 0 & 1 & 0 \\ A^- & 0 & 0 & 1 \end{array}$
Equals	**Covers**	**Covered by**	**Overlap**
			
$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 0 & 0 & 1 \\ \partial A & 0 & 1 & 1 \\ A^- & 1 & 1 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 1 & 1 \\ \partial A & 0 & 1 & 1 \\ A^- & 0 & 0 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 0 & 0 \\ \partial A & 1 & 1 & 0 \\ A^- & 1 & 1 & 1 \end{array}$	$\begin{array}{cccc} & B^\circ & \partial B & B^- \\ A^\circ & 1 & 1 & 1 \\ \partial A & 1 & 1 & 1 \\ A^- & 1 & 1 & 1 \end{array}$

Figure 2.10 - Binary Topological Relations for Simply Connected Regions and Their Corresponding 9 Intersection Matrices [Egenhofer and Herring, 1991]

2.4.4.4 Metric Space

A Metric space is a model of space that has properties that include the concept of distance between objects in space. For a space S to be called a metric space, for any pair of points x and y there must be a distance d (or metric) from x to y that remains true for the following three properties:

1) $d(x, y) \geq 0$ and $d(x, x) = 0$

2) $d(x, y) = d(y, x)$

3) $d(x, y) \leq d(x, z) + d(z, y)$

The first condition requires that the distance between the points must be a positive number unless the points are the same in which case the distance is zero. The second condition ensures that the distance between two points is the same regardless of what way it is measured. The third condition stipulates that it must be at least as far to travel between the two points via a third point as it is to travel directly between them.

2.4.4.5 Directional Space

Directional relations refer to the order in which objects are identified in space. In a directional space cardinal directions such as above, north, southwest are used and depending on their definitions may yield topological information.

2.5 Spatial Query languages

In this section spatial query languages are discussed. Because SQL is the most common query language and can be extended to handle spatial data, it is examined in detail.

2.5.1 Structured Query Language (SQL)

The Structured Query Language (SQL) was originally developed in the late 1970's for IBMs relational DBMS System R and was formerly known as Sequel [Chamberlin and Boyce, 1976]. Since then, it has been widely adopted due to its standardisation by the ANSI (American National Standards Institute) and ISO (Organisation of International Standardisation).

It is a non-procedural or declarative programming language, which means that it is used to specify, *"what needs to be done"* and not *"how to do it"* and is based on a type of calculus called *relational calculus* [Rigaux et al, 2002, p.7]. Relational calculus is based on a type of formal logic called *predicate logic*, which has two variants; one is called *tuple relational calculus* and the other *domain relational calculus*. The *tuple relational calculus* has formed the basis for SQL [Worsley and Drake, 2001].

SQL provides users of relational database systems with the ability to define the structure or schema of a database (data definition) and to subsequently allow them to insert, update, delete and retrieve data from the database (data manipulation) [Chamberlin et al, 1976]. SQL is more than just a query language and is composed of the following two features:

1. Data Definition Language (DDL) – contains commands for the definition, manipulation and deletion of relational database schemas.

2. Data Manipulation Language (DML) – contains commands for the insertion, updating and deletion of tuples in tables. Data retrieval is by far the more complex aspect of SQL.

During the long standardisation lifecycle of the SQL standard there have been a succession of versions [Atzeni et al, 1999, p.86]. Initially, the first definition of the standard was called SQL1, which had all basic features for query formulation and support for schema definition and manipulation. An amendment to this specification produced a version called SQL89. The most notable addition to this was referential integrity. The next version of the specification was SQL2 or SQL92 and is to date the most widely used version and has a vast number of complex language features. A newer version called SQL3 or SQL99 exists, which is not yet widely adopted but contains advanced features such as triggers and new data type support [Eisenberg and Melton, 1999]. Oracle 9i is an example of a DBMS that conforms to the mandatory standards of SQL99 and is also a superset of these standards.

SQL provides only a basic set of standard data types in its specification. These data types are one-dimensional and contain only one value, e.g. number, character, String. These data types are insufficient for storing and manipulating complex spatial data. This is because SQL treats spatial data as integers and strings in relational form, which requires the user to know the implementation details of spatial data. A solution to this problem was not to create an entirely new query language but to extend an existing database query language with spatial concepts [Egenhofer, 1994].

2.5.2 Extending SQL for Spatial Data

Many enterprises today need the ability to store, manage and retrieve spatial data such as geometry, location and topology. There are two main ways of doing this. The first and more functional choice is to use a Geographic Information System (GIS). These systems are loosely coupled meaning that geometric attributes and descriptive attributes are handled by different parts of the system. For Example, *ArcInfo* has two modules for handling spatial data. The *Arc* module manages the geometric aspect of the system and the *Info* manages the descriptive part of the system. The system can also use any DBMS to replace the *Info* module. The other choice is to use an extension to the traditional DBMS to handle spatial data.

Many of the commercial DBMSs available today offer an extension to their systems for managing and querying spatial data [Guting, 1994, p.25; Shekhar et al, 1999a, p.1]. The aim of these extensions is to provide the user with a high-level query language capable of managing complex queries that mix descriptive and spatial conditions. Although this approach is effective, these systems usually fall short in functionality and complex query optimisation compared to fully-fledged GIS. Extended DBMS systems are more suited to dealing with 2D data applications [Rigaux *et al*, 2002].

The most common way to extend a DBMS to handle spatial data is to extend the Structured Query Language (SQL). This extension to SQL has to comply with one of two standards organisations. The International Organisation for Standardisation (ISO), which was responsible for the development of SQL and the Open GIS Consortium (OGC) [Melton and Eisenberg, 2001].

The main objective in developing a spatial SQL is to provide a high-level interface for incorporating spatial concepts into the host language. This is achieved (in the case of SQL) by incorporating Abstract Data Types (ADTs) into the host language [Shekhar and Chawla, 2003]. An ADT is a user-defined data type that has associated functions. ADTs are similar to objects in an object-orientated programming language because they are user defined and have associated functions or methods to operate on the object specific data. This is an optimal data structure for extending SQL because the user does not need to know the implementation details of the ADTs to use them. All that is required is that the user knows the function names and input parameter data types. To extend SQL with such data types it is required that standards are used when extending the data model. The most widely accepted standards are those specified by the OGC.

2.5.2.1 OGIS Standard for Extending SQL

The OGIS standard for extending SQL was formulated by the OGC. This specification describes a standard set of SQL Geometry Types based on the OpenGIS Geometry Model, together with the SQL functions on those types [OGIS, 1999]. The OGIS model can be incorporated into a number of programming languages including Java and SQL. This model is based on the geometry class hierarchy shown in Figure 2.4.

A base class GEOMETRY that has four main subclasses defines the geometry model. These are *Point, Curve, Surface* and *GeometryCollection* [OGIS, 1999]. Each of the subclasses has associated with them a set of operations that act on the instances of the classes. These operations fall into three categories:

1. *Basic Operations* – these operations apply to all classes
2. *Topological Operations* – these operations check for topological relationships between spatial objects, such as *touch* and *inside* (see section 2.4.4.3).
3. *Metric Operations* – these operations are used for spatial analysis, e.g. the distance between two objects (see section 2.4.4.4).

The OGIS focuses on select classes of operations for topological and metric spatial relations in the object model [Shekhar and Chawla, 2003]. Operations based on direction predicates are missing from the specification. Direction predicates are a central aspect of research into user-based egocentric queries and is considered more extensivly at the end of this chapter.

2.6 Spatial Indexing

This section covers the subject of spatial indexing which is an integral part of the directional query processor, the main subject of this thesis. Initially, the need for spatial indexes is explained and following that there is a comprehensive description of the main types of existing spatial indexes. This section covers the main types that are available in some commercial applications.

One approach to storing spatial data is to store the data explicitly as a set of extra records in a relational database. This is not sufficient, as storing spatial data in this format does not exploit the spatial properties of the data by the query processor. For example, this approach does not allow for spatial queries that involve the space occupied by the actual data to be carried out. Spatial data needs to be queried based on the spatial properties of the data, without each relationship being stored explicitly in the database. Storing the relationship information would result in excessively large datasets. This identifies the need to store spatial data implicitly in the database so that a wide range of spatial queries can be executed [Samet, 1995b].

Storing spatial data implicitly in a database requires extending the database with spatial data-types (See section 2.5.2). Extending a DBMS by adding new data types to manage spatial data also means that there is the need for new index structures to index these data types. These index structures are called spatial indexes. Spatial indexes are hierarchical data structures that sort data in relation to the space that the data occupies and are based on the approximation of the spatial objects by using their Minimum Bounding Rectangles (MBR) or the recursive decomposition of the underlying space. An MBR is the smallest aligned rectangle enclosing an object.

In regular Database Management Systems (DBMS), the tuples of a relation are sorted in some manner, usually by indexing the primary key field. In the case of spatial data the data is sorted based on the spatial key of the tuples, in relation to the occupied space. This is called Spatial Indexing and just as the primary key is used to index the relational data in the table, a spatial index is used to index the spatial component. There are two main approaches used to represent spatial data adequately. These are object mapping and object bounding.

Object Mapping

One solution is to approximate the spatial data by reducing the dimensionality of the data. This is achieved by mapping the data into points using some mapping technique. The problem with this approach is that it is only possible to query spatial data that has been mapped. For example, if a line and a point are mapped to a lower dimension into points, the query "find the point where the point intersects the line" is valid because they are both mapped, but for a query that involves a point or object that is not mapped, the query can only be achieved in the source domain (Initial coordinated space). This is because the relationship in the source domain would not necessarily be mapped to the target domain [Samet, 1995b].

Object Bounding

An alternative approach to this problem is to use data structures that decompose the source domain space into regions based on the area that is occupied by the spatial objects. This method is also used to decompose target domain space, but the solution lies in the decomposition of the source domain where proximity is preserved. There are two main types of structures when referring to spatial decomposition [Rigaux *et al*, 2003]. These are Space Driven Structures (Quadtree) and Data Driven (R-tree) Structures. A basic example of each is shown in Figure 2.11. A detailed description of both types follows.

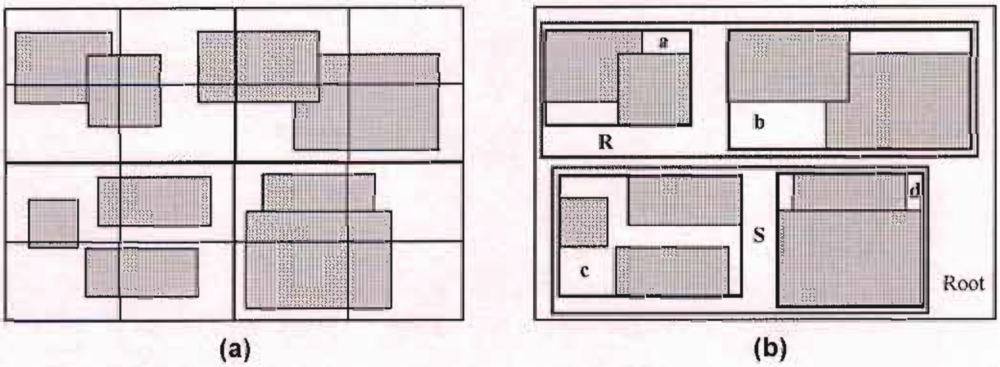


Figure 2.11: (a) Quadtree Cell Decomposition (b) R-tree MBR Approximation

2.6.1 Space Driven Structures

These structures are based on the principal of recursively partitioning the underlying 2D embedding space into cells regardless of the distribution of objects in the plane. The spatial objects are subsequently associated with and identified by the cells that contain them. In this section we will look at one type of Spatial Access Method (SAM) called Quadtrees. These methods are covered in particular because of the fact that they enable the spatial extension to the

DBMS by using their B-tree index, to index multi-dimensional data. A B-tree is an indexing structure used in standard DBMSs to index records in a database table (Relation) based on a unique numerical key.

2.6.1.1 Quadtree

The quadtree data structure is based on recursive decomposition of space into quadrants. It is used to index 2D data because of its simplicity. The space is recursively decomposed into quadrants, labelled NW, NE, SW, and SE. Each non-leaf node has four children and each leaf node has the MBR of the object associated with it [Finkel and Bentley, 1974]. This tree approximates spatial objects by using the cells of the geometry's quadtree decomposition.

Quadtrees use a space-filling curve called a z-ordering curve to order the cells of the 2D grid. This order labels the cells of the grid and provides uniformity to the cells and in some cases preserves the proximity of the objects represented by the grid. Z-ordering is explained in detail later in this section.

There is a large number of variations of quadtrees designed for different applications. There are two main types [Samet and Webber, 1985], (i) point quadtrees are used to represent points and (ii) region quadtrees are mainly used to represent regions. The first discussion in this section covers point quadtrees, the second covers region quadtrees. The third section covers linear quadtrees, which are types of region quadtree that use the Minimum Bounding Rectangle (MBR) of a spatial object to approximate the geometry.

When using this method, the MBR of an object is stored in as many nodes as it intersects. In the case of the quadtree a node is a page and has a capacity of four. When inserting a node into a quadtree, an MBR for an object is inserted into the quadrant or node that it overlaps. The tree is checked; if the node is full the space is decomposed into four segments and the new object is inserted; if the node is not full the object's MBR is inserted into the node. The nodes that are overlapping are reorganised to reflect the current changes. This method is illustrated in Figure 2.14.

2.6.1.2 Point Quadtree

A point quadtree is an unbalanced search tree and each node in the tree has exactly four children. Each non-leaf node contains a point in two-dimensional space and all leaf nodes are empty. The entire search space is divided up from the root into four quadrants, NE, NW, SW, and SE. Each non-leaf node in the tree divides the subspace into four quadrants. Each one has the following structure (X, Y, P1, P2, P3, P4), where X, Y is the point associated with the node and P1→P4 represent pointers to the four children. This is illustrated in Figure 2.12

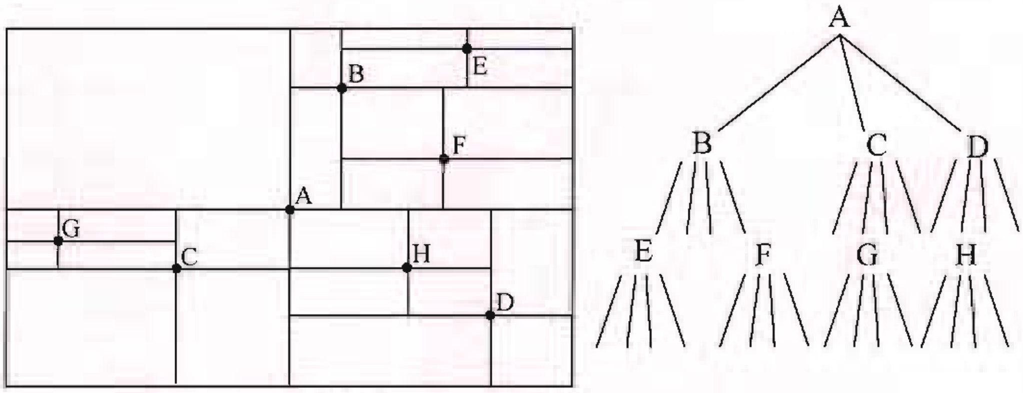


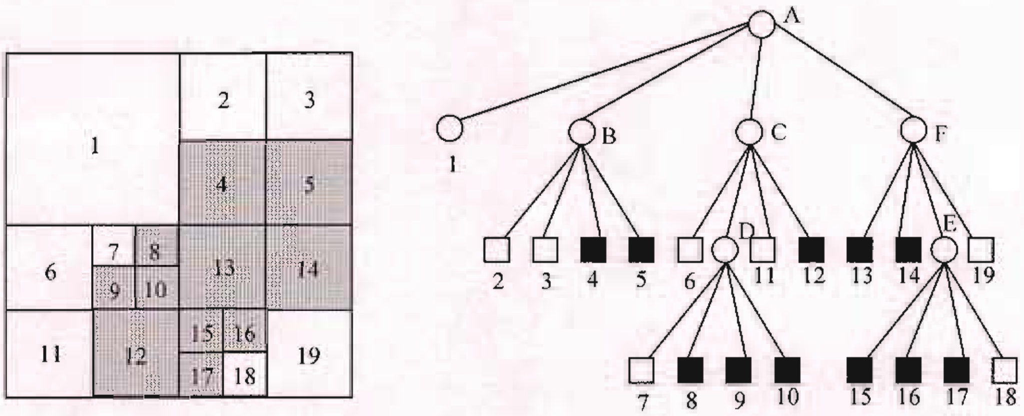
Figure 2.12: (a) Point Quadtree Distribution (b) Point Quadtree

One application of point quadtrees is in indexing composite keys [Finkel and Bentley, 1974, p.1]. For example, if a relation R has a composite key with attributes A1 and A2 then it is possible to build an index on A1 using the x-axis and A2 using the y-axis that speed up the search based on the attribute values. This requires that an additional pointer to the location of the corresponding tuple be added to each non-leaf node [Finkel and Bentley, 1974].

2.6.1.3 Region Quadtree

The region quadtree is a tree that requires that the maximal blocks are disjoint, have standard sizes and are located in standard locations [Samet, 1995b]. It is based on repeated decomposition of an image (that represents the search space) into quadrants. If the quadrants in the region are not composed entirely of 1's or 0's, the quadrant is subdivided until the region consists of blocks made of 1's and 0's entirely.

Region quadtrees are often used to represent black and white images on a two-dimensional space and the space that is used in a region quadtree is bounded and of limited area [Samet and Webber, 1985].



(a)

(b)

Figure 2.13: (a) Region Quadtree cells (b) corresponding quadtree [Samet, 1995]

The grey area in Figure 2.13(a) is an example of a region that is to be represented by a quadtree. Initially the root space A is decomposed into four quadrants. This process continues in the children nodes until each leaf node either contains all black or all white quadrants. The non-leaf nodes are said to be grey. The quadtree structure associated with this decomposition is illustrated in Figure 2.13(b).

2.6.1.4 Linear Quadtree

The linear quadtree is a slight variation of the regular quadtree. In the regular quadtree each node of the tree has four pointers to its child nodes. In the linear quadtree approach the non-empty quadtree blocks are represented by a list of quaternary quadtree codes [Gargantini, 1982]. Quadtree codes describe a quadrant using labels. These labels are then indexed using a B-tree index in which each block contains the ID of the spatial object that it belongs to [Aboulnaga and Aref, 2001].

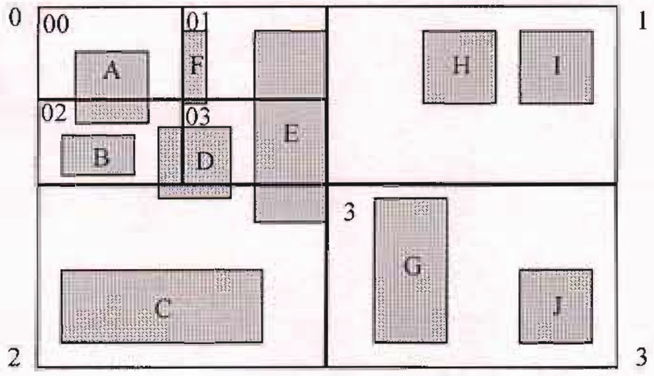


Figure 2.14: Linear Quadtree spatial decomposition

After each node at the leaf level of the tree has been assigned a leaf label *l* and stored in a page *p*, the pairs are indexed using a B+-tree index. A B+-tree index is a tree structure that allows associative access based on a *key* without placing constraints on the physical location of the tuples in a relation [Atzeni et al, 1999]. This provides a natural packing of the quadtree leaves into a B+-tree that is dynamic, thus allowing the insertion and deletion of nodes without losing the persistence of the tree [Abel and Smith, 1983].

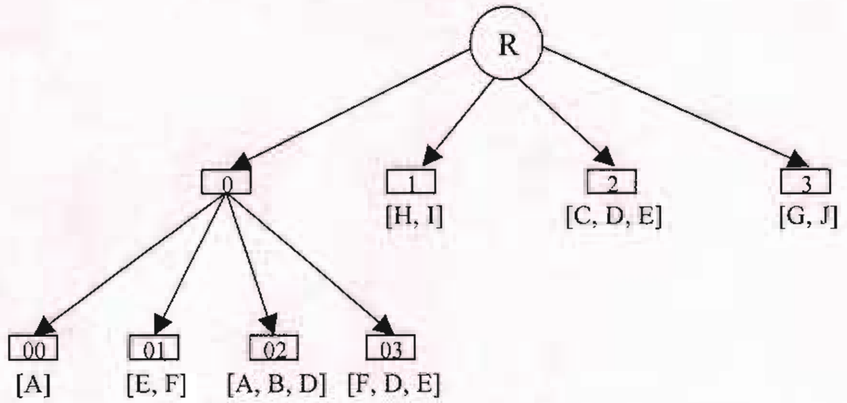


Figure 2.15: Linear Quadtree using a labelling technique

The method used to encode quadtree blocks as integers is called the Morton order [Morton, 1966]. A Morton block is an integer that represents a quadtree block. This labelling of the quadtree cells is also known as z-orderering [Orenstein, 1986].

2.6.1.5 Z-Ordering Tree

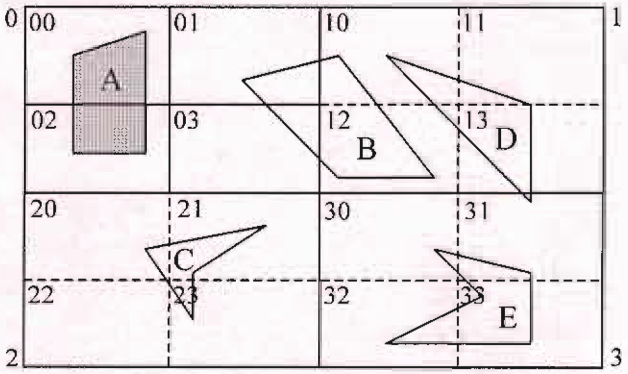
The z-ordering tree is a tree that uses a quadtree decomposition of the space and uses a z-order curve to label the space. The leaf labels are then indexed by inserting them into a B+-tree. In contrast to previous structures this data structure does not use an object's MBR as its approximation but instead, the actual geometry values are mapped to the cells.

2.6.1.5.1 Z-Ordering Space Filling Curve

This curve orders the grid by using the repetition of the order NW→NE→SW→SE. This ordering continues for all decompositions of the space. The cells also use a labelling technique that identifies the depth of the quadrant within the tree. For example, 0,1,2,3 represents a space divided into four. A further decomposition of the first quadrant caused by overflow yields quadrants 00, 01, 02, 03 and so on. This is shown in Figure 2.16.

The organisation of quadtree leaves into a B+-tree index is known as a z-ordering tree. This method involves the decomposition of each object's geometry using quadtree decomposition. The leaves of the quadtree are then indexed to approximate the geometries.

It is the order in which the nodes of the tree are labelled that is the focus of this section. The construction algorithm works as follows. Given a geometry g and a quadrant q, a check is carried out to determine if g overlaps all leaves of the quadrant routed at q. If so, then q is part of the z-order list. If not, the quadrant is divided into the number of pieces that g overlaps (≤ 4). This process re-occurs until the maximum depth of the tree is reached.



A=(00, 02)

B=(01, 10, 03, 12)

C=(2)

D=(1, 31)

E=(3)

Figure 2.16: Objects with z-ordering decomposition

Figure 2.16 illustrates the z-ordering decomposition of a set of objects A, B, C, D, E. If the object overlaps all the sub-quadrants in a quadrant, the association with the object is made using the quadrants z value. For example, the value 2 represents the object C. If the object does not cover all four sub-quadrants the object is represented by the individual sub-quadrants. For example, 00 and 02 represents object A.

2.6.2 Data Driven Structures

This approach uses Minimum Bounding Rectangles (MBR) to approximate spatial data where the MBR is the smallest aligned rectangle enclosing an object. The MBRs are grouped hierarchically and subsequently indexed with a B-Tree index [Comer, 1979]. This method partitions the objects not the embedding space, as is the case with Space-Driven Structures. The R-tree is the original and most common of this type of structure [Aboulnaga and Aref, 2001, p.2]. The following is a description of the R-tree and some of its variations, these include the Packed R-tree, R+-tree and the R*-tree.

2.6.2.1 R-trees

The R-tree is a natural extension to the B-tree index [Bayer and McCreight, 1972] that is widely integrated into relational database systems and is used to index multi-dimensional data. It is based on MBR approximations represented on a height-balanced tree [Guttman, 1984]. R-trees are characterised by the following:

- Each leaf node contains between m and M index records where $m \leq M$
- Each record in a leaf node is represented by (B, ID) where B is the smallest MBR that contains the spatial object in the record.
- Every leaf node has between m and M children unless it's a leaf node.
- Each record in a non-leaf node is represented by (B, C) where B is the MBR of a rectangle that spatially contains the MBRs of its child nodes rectangles and C is a child pointer.
- The root node has at least two children.
- All leaves are at the same level on the tree.

- The MBRs of the spatial objects are parallel to the global coordinate space.

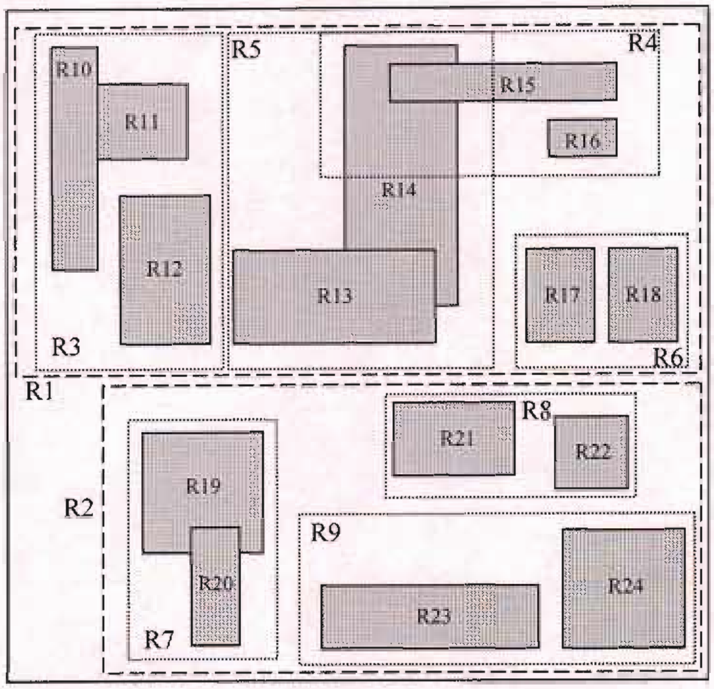


Figure 2.17: Spatial Objects MBRs

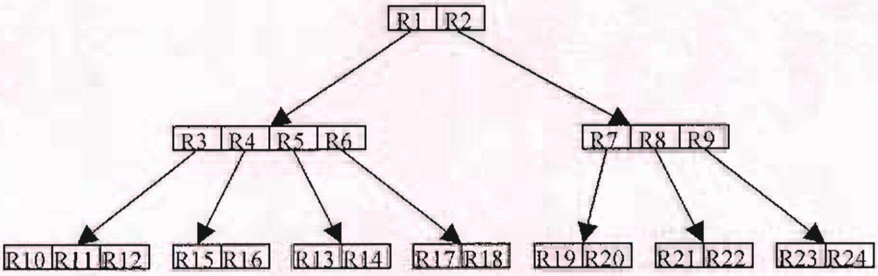


Figure 2.18: R-tree Structure for MBR of Spatial Objects

The queries in an R-tree are processed in a top-down manner as illustrated in Figure 2.18 and the performance of the search algorithm depends on two important factors namely, *Coverage* and *Overlapping* [Samet, 1995b].

Coverage

Coverage refers to the overall amount of space occupied by the R-tree's MBRs. The more empty space occupied by the tree is an indirect measure of the performance of the tree. This coverage is given by the total area of the non-leaf nodes just below the root. This is illustrated in Figure 2.17 where R1 and R2 represent the total coverage.

Overlapping

Overlapping is a major problem with the R-tree spatial index [Kriegel, 1993]. It refers to the spatial objects that are represented by more than one MBR. This is caused by the MBRs of different spatial objects overlapping and the total overlapping at any level is the total area of space covered by more than one MBR. This overlapping may cause the search algorithm to visit more than one node in the tree when searching for a spatial object. It is worth mentioning here that each node represents a page in memory, so the greater the overlapping the more pages in memory are required to be accessed. Therefore, it is essential that Coverage and Overlapping be kept to a minimum to achieve optimal search performance.

There has been some work done in that area of overlap reduction. This has resulted in the development of variations of the R-tree such as the Packed R-tree [Roussopoulos and Leifker, 1985], R+-tree [Stonebraker and Rowe, 1986] and R*tree [Beckmann *et al*, 1990].

2.6.2.1.1 Packed R-tree

This tree works on the premise that the geometry of the spatial objects is known before tree construction. Knowing this, the database uses a compaction technique that packs the index at the initial construction phase thus reducing the coverage and overlapping to a minimum [Roussopoulos and Leifker, 1985].

The compaction algorithm accepts a set of spatial objects as an input and produces an optimal packed R-tree. However further insertions and deletions are handled in the same way used for the regular R-tree. This means that if an insert operation is carried out, it is the nature of the R-tree insert algorithm to insert the data object into the node that requires the least enlargement. Since the tree is so tightly packed by the packing algorithm it is inevitable that splitting will occur. This causes node splits to propagate upwards possibly increasing coverage and overlapping. A solution to this problem was proposed and it involves invoking the compaction algorithm after an insert or delete operation on the tree [Kamel and Faloutsos, 1993].

2.6.2.1.2 R+-tree

This tree works by splitting rectangles of objects that overlap in the non-leaf nodes. Therefore, the object may be represented twice in the leaf nodes. [Stonebraker and Rowe, 1986]. The tree shown in Figure 2.19 is as a direct result of the overlapping between R4 and R5, in Figure 2.17, being eliminated to produce two non-overlapping nodes, each containing a reference to R15. This action eliminates the overlapping between the nodes but in some cases may cause two nodes of the tree to be visited in a search, for example if the query involved finding R15.

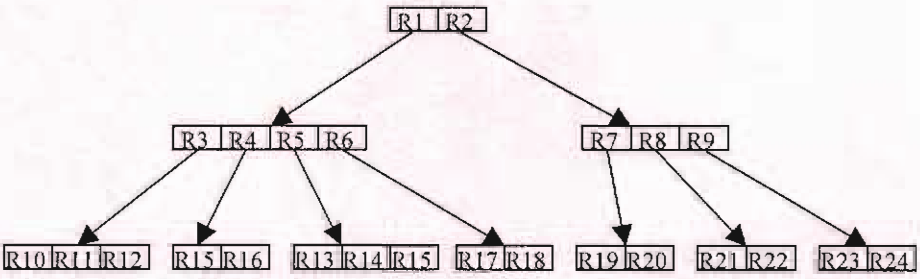


Figure 2.19: The R+-tree Structure

The R+-tree index is more efficient because it provides continuous efficiency, compared to the Packed R-tree that only provides this at the beginning. It is also possible to pack the R+-tree with a packing algorithm to ensure that the tree is arranged efficiently initially. The R+-tree eliminates overlapping by partitioning the MBRs and storing object parts in different nodes of the tree. This gives rise to a requirement for very complex insertion and deletion algorithms to ensure that each node on the tree is at least half full [Sellis et al, 1987].

2.6.2.1.3 R*-tree

The R*-tree is a major improvement of the original R-tree because it aims to improve three problems associated with the R-tree, as follows:

1. Node overlapping (Overlapping between directory rectangles), reducing the number of possible search paths.
2. Area covered by a non-leaf node (Area covered by a directory rectangle and not by the leaf enclosing rectangle).
3. The margin size of a nodes directory rectangle

The first aspect of the R*-tree is how it improves the split algorithm. For example the R-tree structure requires that if a node on the tree becomes overloaded (i.e. > M), a new page has to be allocated. In this context, M is the maximum number of entries that will fit on one node and m is the minimum. It does this by initialising two groups with the two objects that are furthest from each other, then it assigns to each node the remainder of objects in turn.

The example in Figure 2.20 illustrates the advantages of this method. Given M=4 and m=2. The rectangles furthest away from each other are chosen (1 and 2) as the basis for two groups. The fact that object 1 is larger than object 2 results in the next iterations being in favour of group 1. After three rectangles are placed in group 1 the remainder are put into group 2, regardless of their position. This is seen to be a non-optimal split as it clearly causes unnecessary overlapping.

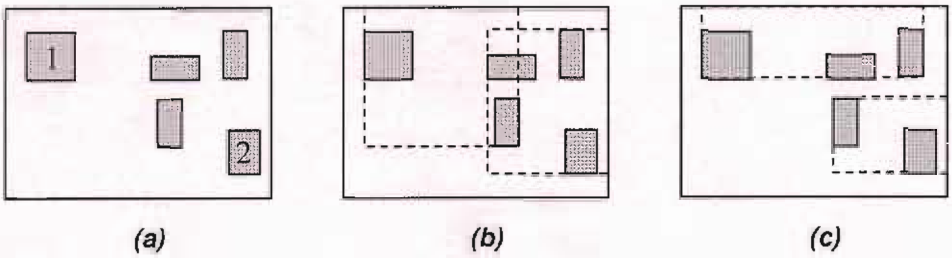


Figure 2.20: (A) R-tree Overflow (B) R-tree Solution(C) R-tree Solution*

In the case of the R*-tree the splitting algorithm assumes that the split has to be along one axis (x or y) and then it explores all the possible combinations each side of the axis. If the optimal axis is chosen, this yields from minimal to no overlapping (this is achieved by sorting the upper and lower rectangle values and subsequently calculating the minimal sum of perimeters). Given the most optimal axis selection, the distribution with minimal overlapping is chosen. If they are the same, the one with the minimum area is chosen [Beckmann et al, 1990].

The R-tree index is very sensitive to the order in which the data are inserted into the tree. It is possible that a tree will behave differently with the same data inserted in a different order. Beckmann [Beckmann *et al*, 1990] describes a technique that involves deleting and reinserting nodes. If a node is about to overflow, the reinsertion algorithm sorts the entries in decreasing order based on the distance between the centroid of the rectangle and the directory rectangle and reinserts the first p entries. In some cases the entries are reinserted back into the same nodes, causing splitting to occur anyway.

In general, the R*-tree seems to be the most efficient of the other R-tree variants. The experiments outlined in [Beckmann *et al*, 1990] also indicated that the R-tree using the linear splitting algorithm was less efficient than the quadratic splitting algorithm.

It is important to mention here that the Spatial Access Methods presented here are only used to speed up the filter (first) step of a two-step query process. The refinement (second) step is also a crucial factor in the overall speed of the query as a whole. This is the step where the exact object is checked against the query predicate and is covered in section 2.7.2.1.2.

2.7 Query Processing

In the field of query processing, there are a number of aspects to be considered. Initially there is the concept of a view. A view can be described as a window into a database, which is usually controlled by assigning users into groups with different levels of control. A view provides the users of a particular domain with a personalised view of the data contained in the database that they are authorised to access. This is called a local conceptual schema as opposed to the global conceptual schema, which is the data model for the whole database [Connolly and Begg, 1998]. If the user wants to access the database directly they have to use a query language such as SQL (detailed in section 2.5.1).

This section gives an introduction to general query processing and then a detailed overview of spatial query processing techniques is presented.

2.7.1 Relational Query Processing

When a user requires data from a relational database system, a query language is used. SQL is the most commonly used query language in database systems today and will be used throughout this section (See section 2.5.1). Relational query processing is defined by a set of operators that are integrated into SQL and is used to formulate queries to a DBMS, where the main aim is to reduce the execution time of the query, which is the sum of all individual operations that make up the query [Selinger et al, 1979]. When querying a relational database, the data are usually stored and sorted in one-dimensional arrays making it easy and fast to query the data. A detailed overview of relational query processing is given in [Connolly and Begg, 1998]

2.7.2 Spatial Query Processing

In contrast to relational query processing, spatial query processing is much more complex [Kriegel et al, 1993?]. First of all, spatial databases deal with large amounts of very complex data that has to be indexed using spatial indexes (covered in the previous section 2.6) and cannot be indexed using regular array indexes. Secondly, there is no fixed set of operators for spatial query processing. Each spatial database system either has its own set of available operators or complies in part with some standards such as the OGIS Simple Features Specification for SQL [OGIS, 1999]. Further to this, the algorithms that can perform such operations as spatial selection and spatial joins need to be seamlessly integrated with the spatial database system. These operators are called spatial operators.

2.7.2.1 Spatial Operators

When designing a spatial database, one of the main problems to overcome is how to connect the operations of a spatial algebra to the facilities of a DBMS query language that also includes predicates to express spatial relationships such as spatial selection and join operations [Guting, 1994]. In this section, the fundamental operations needed to manipulate sets of spatial objects are discussed. These spatial operations can be classified into four groups [Gaede and Gunther, 1998].

2.7.2.1.1 Update Operations

Update operations in a database are standard operations that are available in all relational database systems. These include operations such as *create*, *modify* and *delete*.

2.7.2.1.2 Selection Operations

Spatial selection operations are among the most important type of operations within the set of spatial queries and operations. This is because not only do they represent a query class but also form the basis for other important spatial operations such as nearest neighbour and spatial join [Kriegel *et al*, 1993]. There are two main types of spatial selection:

- *Point Query* – Given a query point P and set of objects O, the point query returns all objects O that contain P.
- *Window Query* – Given a rectangle R and a set of objects O, the window query returns all objects O that interact with the window R.

To process a spatial selection query, a two-step process is required. The initial step is called the primary filter, which uses a spatial index to reduce the candidate set in a spatial query. Spatial indexes (covered in section 2.6) are used to organise pages of secondary memory containing sets of entries with an MBR, an object ID and a reference to the exact geometry of the spatial object. This is because it is not efficient to use the exact geometries of the spatial objects due to the computational cost and therefore, approximations of the spatial objects are used by spatial indexes to reduce the cost. A spatial index attempts to store objects that are close together in a data space on the same page in memory. This reduces the number of page accesses to secondary memory. This approximation technique is the first step of the two-step process and is called the primary filter and is shown in Figure 2.21.

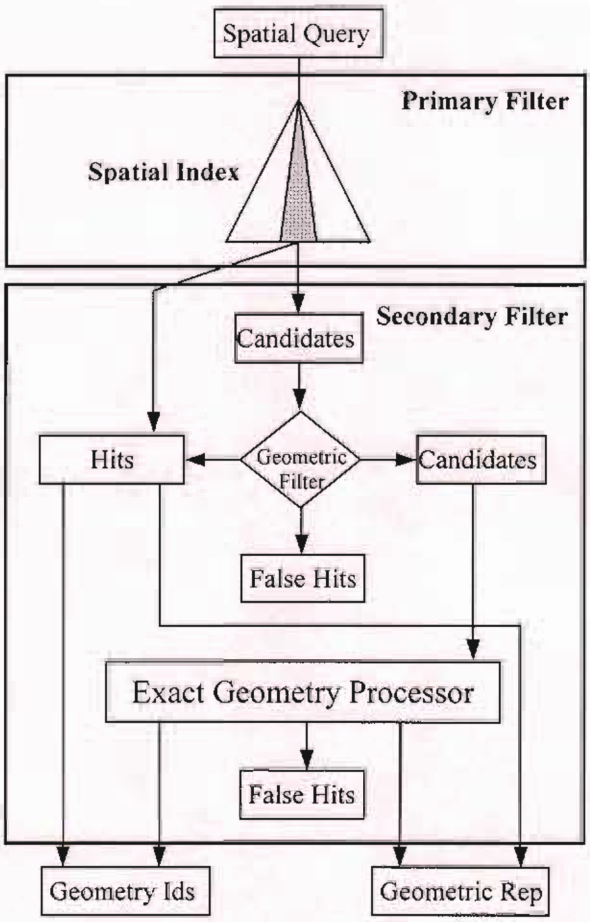


Figure 2.21: Spatial Query Processor

Primary Filter

The primary filter is the first filter in the spatial selection process. If the page area satisfies the query condition, which means that all approximations in the page are totally contained in the query window, these are classed as *hits*. These geometries are definitely part of the result and are loaded into main memory. The remainder of the intersecting object approximations are then sent on to the secondary filter.

Secondary Filter

In the secondary filter, the pages that do not fulfil the query exactly but have entries that may have, need to be checked by an additional geometric filter. This filter applies a further approximation test using the geometric key. The result of this filter yields three classes of outcomes. These are *hits* (satisfying the query), *false hits* (not satisfying the query) and *candidates* (possibly satisfying the query). At this point the geometry IDs and geometric properties of the current hits are recorded.

The next step in the secondary filter is the exact geometry processing. Candidate *hits* that were not detected in the primary filter or the initial part of the secondary filter need to be processed using the exact geometries of the spatial objects. This part of the filter is expensive with regard to I/O cost. This step involves loading the exact geometry of the remaining candidates into main memory from secondary storage and applying complex computational geometry algorithms to identify the candidates that are *hits* [Brinkhoff et al, 1993b]. The complex algorithms referred to are well known in the field of computational geometry, but are beyond the scope of this thesis.

This two-step approach to spatial querying dramatically reduces the cost of the queries because the primary filter approximation technique greatly reduces the number of geometries that have to be tested by the exact geometry processor. The intermediate geometry processor also reduces the candidate set thus providing a minimal number of candidates for the expensive exact geometry processor.

The next section that is examined is spatial joins. The spatial join is an operation that combines two sets of spatial objects to create a new set.

2.7.2.1.3 Spatial Joins

A spatial join is similar to an ordinary join operator in a relational database except that it is more complex because the columns of the relations that are being joined are no longer one-dimensional joins but are joins between spatial data types that may be of two or more dimensions. It is one of the most important operations and is the result of a join between two tables using a spatial predicate. An example of a spatial predicate is *"overlaps"* (the spatial predicates resulting from Egenhofer's 9-intersection matrix are covered in section 2.4.4) [Egenhofer, 1991]. The spatial join is part of the filter (first) step in the query process and the computational geometry algorithms used in conjunction with them are beyond the scope of this thesis.

In spatial database management systems there are two main types of queries. These are identified by the time it takes to run a query based on object accesses. A single-scan query needs only to access an object at most once to satisfy the query [Brinkhoff *et al*, 1993a]. Examples of single scan join queries are *point queries* and *window queries*. An example of a point query is *"find the building that cultural heritage data point A lies inside"* and an example of a window query is *"find all cultural heritage data points that lie within a 2m square window at location X*". The other type of query is called a multiple-scan query. In this type of query, objects have to be accessed more than once to execute the query. The spatial join is a multiple-scan query that uses the filter and refinement approach and is one of the most important queries along with the map overlay operation [Orenstein, 1986].

The spatial join operation is used to combine spatial objects of different sets according to the spatial properties of the spatial attribute associated with the relations. One of the main problems with the spatial join operation is that its counterpart in a conventional relational database system, the join operation, is not efficient when applied to spatial columns unless they are modified [Brinkhoff *et al*, 1994].

When performing a spatial join, the defined procedure is to take the two relations to be joined and compare them. After the comparison of the relations, a selection is made to retrieve the results of the comparison operation. For example, consider the spatial relations *"Building Blocks"* and *"Cultural Heritage (CH) Data Points"* where the spatial attributes represent the borders of each. The query *"Find all CH data points that are inside buildings"* is an example of a spatial-join query. This is seen to super-linear (O(N^2)) in the number of objects because it involves computing the cross product before doing the selection. An example of the cross-product operation of relations R and S (RXS) is shown below:

R	R.Name	R.Geom

S	S.Name	S.Geom

R X S	R.Name	R.Geom	S.Name	S.Geom

There have been many attempts to replace the need to carry out the cross-product operation to reduce the cost of the join operation. In this section, some of the algorithms that are used to process spatial joins are examined.

Nested Loop

This algorithm uses the nested loop to perform the search for pairs of tuples. Given two relations L and M, the algorithm tests each tuple in L with each tuple in M. For each element in L, scan the entire relation M for a match. This method is a brute force approach and yields very high I/O costs. Since most spatial relations are large, this approach is deemed to be unacceptable as an efficient way of processing spatial joins [Brinkhoff et al, 1993a]. An algorithm that tests for overlap between L and M is as follows:

```
forall tuple l ∈ L
        forall tuple m ∈ M
                if overlap(L.geom., M.Geom)
                then add < l,m > to result
```

One solution that can speed up the processing of this operation is to use a spatial index; if a spatial index is set up on the inner loop, a candidate set is produced that is a subset of the inner relation thus reducing the need to scan the entire inner relation completely each time. Instead the algorithm only compares the outer relation with a candidate set that the index provides. This approach is called the *Scan and Index Strategy* (SAI) [Papadopoulos et al, 1999].

Tree Matching

Tree matching is an approach that requires that both the relations associated with the join have R-tree (section 2.6.2.1) indexes available. This algorithm works on the fact that the rectangles of the R-tree in the non-leaf nodes contain all rectangles of their children nodes. So, if the MBRs of the non-leaf nodes in one relation do not intersect the non-leaf nodes in the other relation in any way then it is not possible that they intersect in the children nodes. However, if the directory rectangles do intersect then it is possible that they may intersect further down the tree at some stage [Brinkhoff *et al*, 1993a].

Partition-Based Spatial Merge

This implementation of a spatial join introduces a new algorithm for performing a spatial join operation. In addition to the filter step of the spatial join it also includes the refinement step. It partitions the inputs of the join into manageable partitions that fit into memory. A computational geometry plane sweep algorithm is then used to join objects from the two inputs [Patel and DeWitt, 1996]. The specifics of this algorithm are complex and are beyond the scope of this thesis as the focus is on user-based queries that can be processed using spatial joins.

This concludes the section on spatial joins. The next section focuses on spatial aggregates.

2.7.2.1.4 Spatial Aggregates

Spatial aggregates commonly refer to the nearest-neighbour query or one of its variants. An example of a nearest neighbour query would be *"Find the closest Cultural Heritage artefact to me"*. These types of spatial queries usually involve more than one spatial operation to obtain a result. For example, in the query above, the first pass calculates the distances of all objects within the given range and the second pass runs a range query to determine the closest one. A range query describes a region in space and then requests all points in the region [Roussopoulos et al, 1995].

2.8 Directional Query Processing

In this section, directional query processing is discussed. Direction-based spatial relationships are needed in many domains and are frequently used as selection conditions in spatial queries. This section gives a description of the most recent attempts by the research community to develop an effective method of processing directional queries.

2.8.1 Direction Relations

Direction relations deal with order in space, for example *North, East, South, West*, *above* and *below*, etc [Papadias et al., 1994]. The approach of using direction to query spatial data is the focus of substantial research efforts within the spatial database community [Goyal and Egenhofer, 2001; Liu et al, 2003; Liu et al, 2000; Papadias et al, 1994; Pfoser et al, 2000; Shekhar et al, 1999a; Shekhar et al, 1999b; Theodoridis et al, 1996]. Direction relations therefore represent an important class of user queries in spatial databases and their applications to geographic information systems but as of yet are not available in most spatial database systems.

The main reason for this is because of a lack of definitions and standards for direction relations [Theodordis et al., 1996]. When describing direction relations, there is a common vocabulary used but the definitions are vague. For example, Co. Donegal is *North* of Co. Dublin, but also it is *North-West.* It is these types of ambiguous descriptions of direction relations that provoke the need for research to be carried out in this area. There is a great need for a set of standard definitions for direction relations that can be totally interoperable between applications.

There has been little work carried out in this area with regard to the formalisation of direction relations in a two-dimensional space [Shekhar and Chawla, 2003, p.67]. In contrast to topological relations where there is a widely accepted set of relations [Egenhofer and Herring, 1990], there are no such definitions for direction relations.

To understand direction, a reference frame must be established. The next section describes the three main frames of reference and following this, the two main direction models, projection-based and cone-based, are outlined.

2.8.1.1 Absolute Direction

Absolute direction is direction that is defined as the relationship between objects based on their locations in an embedding space [Shekhar and Liu, 1998]. Absolute direction uses an *extrinsic* frame of reference where the reference frame is established independently of the orientation of the objects or the observers. Examples of absolute directions are: *north*, *south*, *east* and *west*.

2.8.1.2 Object-Based Direction

Object-based direction is the direction of a target object with respect to the orientation of a reference object. In such a case, the reference object has direction and the target object may or may not [Liu, 2000]. Object-based direction uses an *intrinsic* frame of reference where the reference frame is in respect to the orientation of a reference object. Examples of object-based direction are *front, back, left* or *right.*

2.8.1.3 Viewer-Based Direction

Viewer-based direction refers to the direction relation, which is measured from the viewer's perspective [Liu, 2000]. Viewer-based direction uses a *deictic* frame of reference where the reference frame is relative to each individual looking at the scene. Examples of viewer-based directions are: *'in front of me'*, *'behind me'*, *'to the left of me'* and *'to the right of me'*. And for another person what is *'in front of me'* might be *"to the left of him'*.

2.8.2 Projection Based Model

The projection-based approach to defining direction relations assumes an absolute frame of reference and a set of orthocanonic x and y-axes (i.e. parallel to axes). This model defines direction relations using projection lines that are perpendicular to the coordinate axis [Theodoridis et al., 1996].

2.8.2.1 Point Direction Relations

In [Papadias et al., 1994], a method of defining direction relations using the relationships between points is described. The definitions are represented by universally and existentially quantified formulas, which are subsequently used to compare point coordinates [Theodoridis et al., 1996, p.2].

Using this approach, direction relations between objects are defined using first order predicate logic to compare point coordinates. In the following example, p_i is a point of object A (primary object) and q_i is a point of object B (reference object) also X and Y are functions that return the x and y coordinate of a point. In the example in figure 2.22a, the definition of a low-resolution direction relation called *strong_north* is shown. This is called low resolution because its acceptance area is large. In figure 2.22b a higher resolution of the *strong_north* is shown called *strong_bounded_north*. This is high resolution because the acceptance area is small.

In the *strong_north* example the definition states that for all points p_i in A and all points q_i in B every Y value in p_i must be greater than the Y values in q_i. This is illustrated with the following definition and by the diagram in figure 2.22a.

$$\forall p_i \, \forall q_i \, Y(p_i) > Y(q_i)$$

In this example, the horizontal axis is the projection line and is perpendicular to the Y-axis. The resolution of this direction relation is low because, as may be seen, there is only a loose definition of requirements, in this case the Y values for primary object A must all be greater than the highest Y values in the reference object B.

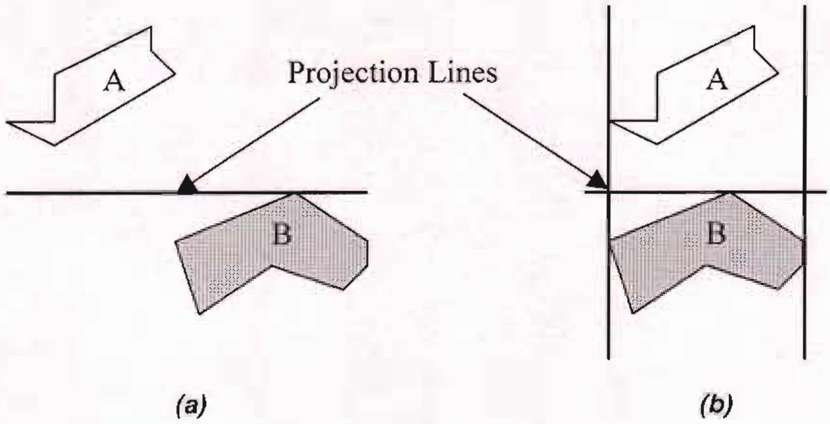


Figure 2.22: (a) strong_north (b) strong_bounded_north direction relations

Figure 2.22b illustrates the *stong_bounded_north* direction relation. In this case, all points of object A must be in the region bounded by the horizontal line that passes from B's north most point and by the vertical line that also bounds B.

$$\begin{aligned}&\forall p_i \forall q_i \;\; Y(p_i) > Y(q_i) \wedge \\ &\forall p_i \exists q_i \;\; (Y(p_i) > Y(q_i) \wedge X(p_i) > X(q_i)) \wedge \\ &\forall p_i \exists q_i \;\; (Y(p_i) > Y(q_i) \wedge X(p_i) < X(q_i))\end{aligned}$$

This first order predicate logic notation illustrates how universally and existentially quantified identifiers are used to compare the point coordinates of the relations. This approach can be used to represent all other directional relations in that they can refer to several levels of resolution and it can be extended to correspond to certain application needs which are readily defined by comparing point coordinates [Theodoridis et al., 1996].

When integrating direction relations into a DBMS using these methods traditional two-dimensional approximation techniques are used. MBRs are used to approximate the spatial objects because they are efficient and only need two points to represent them. The spatial access methods used to represent the approximations can utilize traditional one-dimensional access methods or specialised indexes for spatial data. In the next section there is an example of each type.

The first approach uses a B+-tree index [Comer, 1979] to retrieve direction relations [Theodoridis et al., 1996]. To represent the MBR described in Figure 2.22b four B+-tree indexes are used, one for each ordinate of the two coordinates that approximate the object. The second approach to representing direction relations in a DBMS is to use an R-tree data structure [Guttman, 1984]. (R-trees are covered in section 2.6.2.1). Using this method involves approximating the objects with MBRs that are the leaf nodes of an R-tree index [Theodoridis *et al.*, 1998].

2.8.2.2 Direction Relation Matrix

More recent work into determining the direction between spatial objects by Goyal uses the direction relation matrix [Goyal and Egenhofer, 1997]. In this approach an extended MBR is placed around object A and the percentage of the area of object B that falls within the various regions is recorded in the direction relation matrix dir(A,B). This is illustrated in Figure 2.23.

$$dir(A,B) = \begin{pmatrix} 0 & .04 & .46 \\ 0 & 0 & .5 \\ 0 & 0 & 0 \end{pmatrix}$$

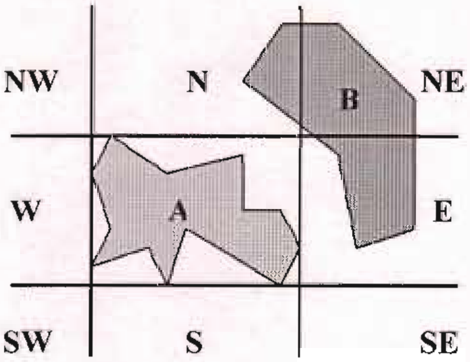


Figure 2.23: Direction Relation Matrix dir(A,B)

By building direction relation matrices for each pair of features, the configuration of the input query scene can be compared to configurations of the same objects in the database. When the difference between the query scene direction matrix and the database scene direction matrix is minimum, the best match has been found. However, there are cases where this approach will not distinguish between two different configurations of objects. For example in Figure 2.24, where both objects lie within the MBR, both configurations have identical direction relation matrices.

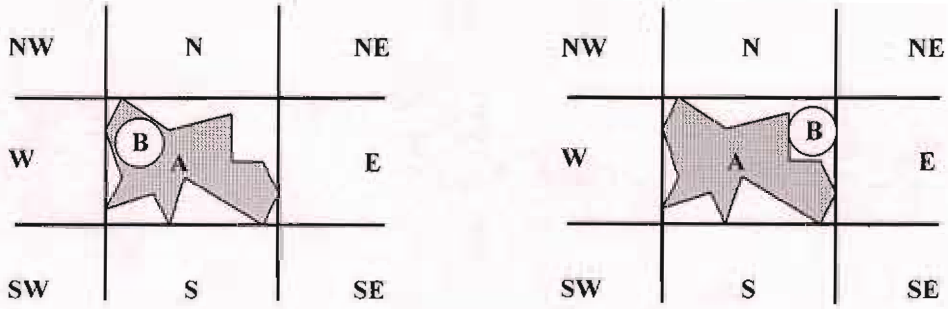


Figure 2.24: Identical Direction Relations dir(A,B)

However, if topology and direction is taken into account, the above two configurations can be distinguished. But, even topology and direction cannot distinguish between all combinations of object configurations. For example, the configurations in Figure 2.25 prove to be identical configurations based on their topology and direction alone.

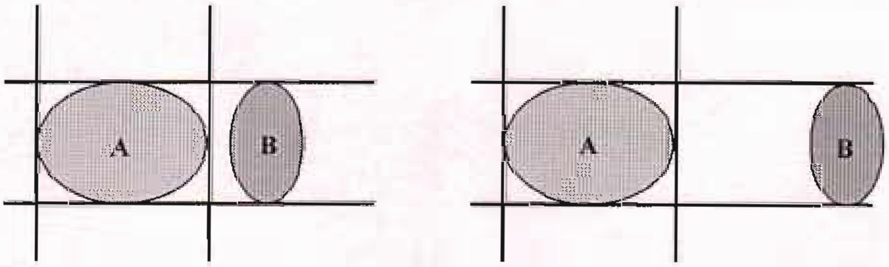


Figure 2.25: Identical Topology/Direction Relations

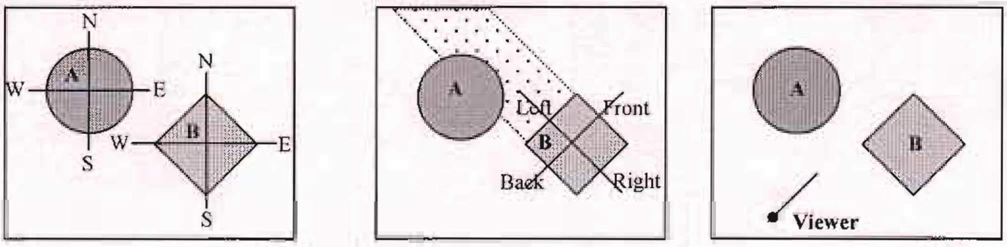
It is clear from the illustration that distance is also required in order to distinguish properly between a pair of spatial objects.

2.8.2.3 Direction as a Spatial Object

In contrast to all the previously discussed attempts to model direction as a relational predicate between spatial objects, the approach in [Shekhar and Liu, 1998] is different. This basic approach models direction as a spatial object that is represented by a unit vector. Modelling the vector as a direction object has the advantage that each object has its own attributes and operations, giving it a rich set of vector predicates and operators on direction that makes direction reasoning easier [Shekhar et al, 1999].

Using this approach direction can be described using the absolute, object and viewer-based frames of reference. When defining absolute direction, each cardinal direction (e.g. *North, South, East* and *West*) represented by a single unit vector. This is shown in Figure 2.26a where A is *West* of B and B is *East* of A.

When defining object-based direction, each object's direction in a scene is described relative to a reference object, which is oriented. This is shown in Figure 2.26b where object A is to the left of the reference object B. An example query using object-based direction is *"List all CH data points to the left of B"*. This means that every CH object contained in the region left of B (represented by the dotted lines) is a candidate. Usually, it would be the MBR of this area that would be used to query the database and then a secondary filter applied to find the exact candidates, but in [Liu et al, 2003] an Open Shape-based Strategy (OSS) is introduced that uses the direction region as the query window thus eliminating the need for a secondary filter. This new approach requires the definition of a new data type to represent an open shape but significantly outperforms the regular Range Query Strategy (RQS).



(a) *(b)* *(c)*

Figure 2.26: (a) Absolute, (b) object-based and (c) viewer-based direction

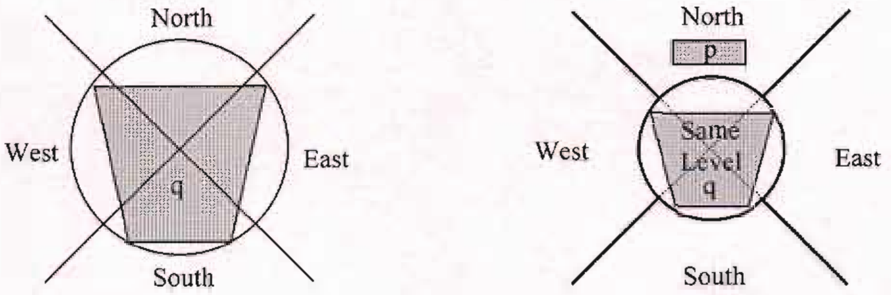
In the viewer-based example in Figure 2.26c, the direction of object A is determined relative to B from the viewer's perspective, which is similar to the object-based approach except that the viewer is able to change direction (unlike static objects) by using translate and rotate operations. A vector pointing in the direction the user is facing represents the user's direction.

Viewer-based directions are the easiest to understand and use when navigating, as the instructions are relative to the user, compared to absolute directions, which have little meaning if north is unknown. Viewer-based directions are used a great deal in navigation systems. For example, if using a car navigation system, the instruction *"turn left"* is much better than *"turn east"* and the query *"List all CH data that are left of the GPO from my viewpoint"* is better than *"List all CH data east of the GPO"*.

This concludes the section on projection-based direction relations. The following section describes an alternative method of defining direction relations, called cone-based direction relations.

2.8.3 Cone-Based Model

Cone-based directions are based on the division of an object's direction domain into *Cones* of equal size, each of which identifies a direction with reference to the object. *Cones* are the triangles that are created when a circle is divided up into four or more parts by straight lines that run through the centre of the circle at 45 degrees and extending out past the extents of the object. Subsequently, each cone represents a direction (*North, South, East, West*). This is illustrated in Figure 2.26a.



(a) (b)

Figure 2.26 (a) Cone-based Directions (b) Using MBCs

2.8.3.1 MBC Approximation Based Direction Relations

Up to now each method of processing topological and direction relationships has used MBR approximation techniques to index spatial objects. The most common and efficient index structure used to index rectangles is the R-tree or some variation of it [Brinkhoff et al., 1993a]. However when it comes to indexing circles, a spatial access method called the *Sphere-tree* [Oosterom and Claassen, 1990] is used.

The approach taken in [Safar and Shahabi, 1999] uses a slightly different method to index spatial objects. Instead of using an MBR to approximate the spatial objects a Minimum Bounding Circle (MBC) is used. As circles are insensitive to orientation, using MBCs to approximate these objects is an advantage as thier MBC's are unique and invariant when it comes to translation and rotation.

It is this similarity between cone directions and MBC approximations that makes it suitable for processing direction relations. The direction relations are defined using the MBC of a *reference* object q as shown in Figure 2.26b. The MBC of q is divided into five parts *North, South, East, West* and *Same Level* (Figure 2.26b). If the primary object p interacts with any of the partitions, it is said to be in the direction the partition represents, therefore object p is *North* in relation to object q. In addition to these single cardinal directions, if the primary object spans over more than one direction partition the object is given both cardinal directions. For example, if the primary object spans over the *West* and *North* partitions the direction relation would be defined as *Northwest* of the reference object.

2.8.4 2D+ String

This approach is based on the knowledge that spatial reasoning requires very large geometric computation power. This computation is doubled when reasoning direction relations as both toplogical and directional relations are calculated separately. The 2D+ string approach tries to avoid this repetition and improve the processing time by introducing a new metadata representation scheme [Bowon and Kyhyun, 1999]. It works by creating a two-dimensional string that contains direction and topology information between spatial objects. The 2D+ string is a combination of two one-dimensional strings, *u* and *v*. The u string represents the directional relationships for the x-axis and topological relationships for objects that have their MBRs overlapping and the *v* string describes directional relationships for objects on the y-axis. It is based on the 2D string proposed by [Chang, 1989] but is constructed from the MBR of the object as opposed to a fixed grid and the time needed to generate the string is shorter. The main advantage of this method is that the directional and topological relationships (query results) are not handled separately but together. This method assumes that the objects are vector line strings and that the objects are two-dimensional.

Inference rules are then used to reason direction and topological relationships. Analysing the metadata strings and applying the information to the inference rules provides the information needed to process these queries. This method reduces the amount of storage and processing time needed for the analysis.

2.9 Summary

In this section, the main methods of processing directional queries in a spatial database system have been outlined and the theoretical and experimental work of a number of researchers critically presented. This concludes the literature review undertaken for this research. In the next chapter, a discussion on the findings of the literature review is presented and the chosen technologies for the project are identified.

Chapter Three

Analysis and Synthesis of Literature Review

3 Analysis and Synthesis of Literature Review

3.1 Analysis

A discussion of the observations and findings that surfaced in the literature review is presented in this Chapter in order to provide justification for the topics covered by the research. Initially, an analysis of the literature review is given followed by the synthesis of a conceptual design for the proposed system.

The main objective of the project was to create a web-based application that is capable of delivering content, based on the users context, where context includes location. In principle, the *CHI system* is a framework that enables the user of the system to navigate spatially in a 3D environment while querying a CH dataset based on a contextual query where the primary query parameter is location. The results of such a query are subsequently exhibited to the user by publishing the relevant information in a separate pane of the system that will simulate mobile devices. It was decided that Cultural Heritage (CH) information would be used as the content and a particular event "The 1916 Rising" was chosen.

As an exemplar of what could be done, it is clear that to deliver this content to the user, based on location would require the user to have a mobile device capable of determining its location using a Global Positioning System (GPS) and the bandwidth to deliver multimedia content efficiently using technology such as General Packet Radio Service (GPRS). Currently, the area of mobile telecommunications is rapidly evolving and some devices are capable of providing such advanced capabilities [Schiller, 2000]. Although the promise of Location-Based Services (LBS) based on these technologies has been publicised, the LBS sector is still relatively young, and the cost high [O2, 2004]. Considering this, a decision was made to simulate the mobile aspects of the project based on the predicted operational costs of development and testing of the system.

Simulating the mobile devices also meant that the spatial interaction part needed to be simulated in order to capture the positional aspect of the system. To achieve this, it was decided to use Virtual Reality (VR). Using VR, a 3D model of Dublin can be created to simulate the spatial interaction of the system.

3.1.1 Database Systems

In the literature review, a general study of the area of Database Management Systems (DBMS) was undertaken.

The primary motivation for researching database technologies is based on the functional requirements of the project, which includes the database systems' ability to store and process spatial data. Even though an RDBMS is capable of storing spatial data, it is not efficient as the mapping of spatial data using standard data types is not natural. Consequently, there is a need for additional data types to be integrated into the database system. Extensibility is one solution to this problem: to store and manage spatial data, a DBMS must be at least capable of extending the database schema to accommodate Abstract Data Types (ADT). ADTs, which are data types that comprise two or more pre-defined data types, can be defined and are capable of representing spatial data. One type of extended DBMS is the Object-Oriented Database Management System (OODBMS). These systems are based on the object-oriented paradigm and have good modelling capabilities making them extensible to the needs of spatial data. However, OOBDMSs have some standardisation issues. These systems do not have a standard data model and the essential algorithms that are required in a DBMS are not fully developed thus creating performance and security issues [Rigaux et al, 2002].

Following this, the use of Geographic Information Systems (GIS) was considered (Section 2.3.4). This loosely coupled approach was deemed unsuitable because the spatial and attribute data is separated into different modules and linked, which did not conform to the desired integrated design.

In contrast to the loosely coupled approach, there is the integrated architecture. This approach uses traits from two of the previously covered database systems, the OODBMS and the RDBMS. Object Relational Database Management System (ORDBMS) are DBMSs that possess a spatial extension to the database kernel that allow it to store and process spatial data within the database schema. These systems do not have as much functionality as GISs but have the ability to store and process spatial data with a limited amount of spatial functionality. The spatial data in the system are treated the same as the non-spatial data and can be created, deleted and manipulated using SQL. The data can also exploit the query engine and optimiser of the RDBMS. These software systems are databases integrated with spatial datatypes, spatial indexing methods and spatial query languages and are called Spatial Database Management Systems (SDBMS).

In selecting a suitable database system for the *CHI project*, the prerequisites are that the database system is capable of creating, storing and manipulating spatial data and that it has a user-friendly query interface. After careful consideration of these requirements and the availability of them in DBMSs, the preferred and most functionally suitable was the ORDBMS. This is because these systems offer a sound RDBMS architecture that incorporates ADTs and spatial indexes into their architecture to accommodate spatial data. There were a number of ORDBMS available at time of writing. An analysis of these systems is presented in Table 3.1, which are rated by the presence of eleven important factors relating to spatial databases.

	Spatial Index	User Interface	Spatial Datatypes	Spatial Joins	Scalability	Security	Extended SQL	Open Source	OGIS Compliant	Topology	Window Query
DB2	✓	×	✓	✓	✓	✓	✓	×	✓	✓	✓
Oracle 9i	✓	×	✓	✓	✓	✓	✓	×	✓	✓	✓
Postgres SQL	✓	×	✓	✓	×	×	✓	✓	×	×	✓
MySQL	✓	×	✓	×	×	×	✓	✓	×	✓	✓

Table 3.1: Comparison of ORDBMS

After careful consideration, Oracle 9*i* was the ORDBMS chosen for the *CHI project*. This database system has an extension to the RDBMS called Spatial Data Option (SDO) that offers spatial data types, spatial indexing methods and spatial join mechanisms. Postgres was not chosen because it was not OGIS compliant, and MySQL for the same reason and also because of the absence of spatial joins. The decision to use Oracle instead of DB2, which has a similar specification, was because it was available at the time.

3.1.2 Spatial Data

The next section of the literature review focuses on spatial data and how it is represented in the database. This includes a description of the operations used to manipulate and query spatial data. In a Spatial Database (SD), data are integrated into the system by using custom-built datatypes, which are capable of storing the data efficiently. This approach works well except when it comes to interoperability between database systems.

To avoid this issue a Spatial Data Model (SDM) is necessary. A SDM is a logical model that describes a standard structure for a spatial data hierarchy that database vendors can adhere to, so that the design of their spatial data components can be somewhat interoperable with other systems. There are two main types of SDM, namely, field-based and object-based models used to model different types of spatial data. The field-based approach is used to represent data that is continuous, such as rainfall distribution, where each point on a given layer has an associated value. The object-based approach defines spatial data as a set of discrete objects in a coordinated space, so each object is represented in its entirety by a set of coordinates and an associated coordinate system. These data models are used as a blue print for the overall design and representation of the spatial data structures within the system. The object-based approach is the most suitable for the *CHI system*, as each object in the CH dataset needs to be represented as a discrete, identifiable object that can be operated on independently. *Oracle Spatial* uses the object-based approach to define its spatial data.

The objects theoretically defined by the object-based model, are physically represented in the database system by special data types called Spatial Data Types (SDT). As there are many different types of geometric objects, there is a standard set of types of geometries that are defined by the OGC. Therefore, geometry is represented by one of a finite set of standard types. Oracle Spatial also conforms to this standard.

In the *CHI system*, the operations used to query the database will include Topological, Metric and Directional. The topological and metric operators (Discussed in Section 2.4.4) are available as standard in Oracle and therefore easily accessible. The directional operations required in the system will be developed as part of the application. This is because there are no currently available database systems with directional operations included in the data model.

The RDBMS for the *CHI system* needs spatial capabilities. All of these capabilities need to be integrated into the DBMS using a standard definition and manipulation notation. In normal DBMSs this is SQL and in Spatial DBMS it is extended SQL that enables SDTs to be created, modified and deleted from the database. The operations discussed in the previous section are all integrated into the extended SQL language based on the OGIS Simple Features Specification For SQL (Section 2.4.3.1) and can be used to query the database spatially by using special operators defined in the database schema. Extended SQL will be used to query the Oracle spatial database for spatial interaction between discrete spatial objects.

3.1.3 Spatial Indexing

Spatial indexing is a prerequisite when it comes to querying spatial data. Spatial indexes are used to index spatial data based on the geometric properties of the data as opposed to the position of the geometry in the table. In the *CHI system* spatial indexes are used to index Cultural Heritage (CH) data points. This improves the response time of a spatial query to the database by reducing the amount of records that are checked to satisfy a query.

From an analysis of the literature, it was decided that the linear Quadtree would be used to index the data points in the CH database. This is because the linear Quadtree is more efficient when it comes to indexing simple objects and additionally, this part of the database will be continuously updated and the Quadtree performs well here also. However, the analysis of the R*-tree revealed very good potential and was marked as an alternative structure, awaiting performance results of the overall system.

The following section is an analysis of query processing techniques.

3.1.4 Query Processing

Query processing is a central aspect of the project and the method used to retrieve data from a database. Spatial querying can be achieved by extending SQL with spatial data types and spatial operators. This extension should comply with some standards, the main one being the *OGIS Simple Features Specification for SQL* [OGIS, 1999].

In the literature, there are four key types of operation that are required in a spatial database system to process spatial queries. Update operations are standard operations that are required to *create, modify* and *delete* spatial objects. The spatial algebra required to perform these operations needs to be integrated into the query language. The selection operator is probably the most important of these operators when it comes to spatial querying with two main types, the point query and the window query.

Point queries are used to determine the objects in a scene that contains a specified point. The window query is a more complex selection operation, which returns all objects in a scene that interacts with a specified window. This type of selection operation uses a spatial index (Section 2.6) to determine the intersecting objects. It uses a two-step procedure where the first step (i.e. primary filter) uses the spatial index to efficiently determine the most likely candidate set of objects that intersect the query window. The candidate set is then passed on to the secondary filter, where computational geometry algorithms are used to determine the exact candidate set and identify false hits. This two-step approach substantially reduces the query time because the candidate set processed by the computationally expensive secondary filter is greatly reduced due to the initial reduction by the primary filter.

It was decided that the window query would be the method used to query the database in the *CHI project*. The window will represent the position of the user within the 3D environment, and as the user progresses through the space the window query position will be updated in real-time. The primary reason for selecting this method over the point query is because the window can represent a greater area than the point query and thus has greater opportunity to interact with the CH data.

The spatial join is the same operation as a regular join, in that it combines two tables, except the columns joined are spatial columns. An example of a spatial join is *"Give me all Cultural Heritage data points that are contained* ***inside*** *buildings"*. This query would involve joining the *"building"* and *"cultural heritage"* tables.

A spatial join is effectively the cross product of two tables followed by a selection condition. The cross-product operation is particularly expensive for spatial tables and there are many algorithms proposed to generate it. The most common ones used are the nested loop, tree matching and partition-based spatial merge.

The literature shows that, based on the three algorithms, the partition-based spatial-join merge is more efficient when neither of the input tables has a spatial index. When the smaller input table is indexed it also performs better than the other algorithms. It is also shown that the tree-matching algorithm performs better than the others in the cases where the larger input table is indexed and where both inputs tables have pre-existing indexes.

The final type of spatial operation discussed in the literature review is spatial aggregate. A common query that uses spatial aggregate operation is the nearest neighbour (NN) query. This is because the most common type of NN query uses a two-pass algorithm to process the query. For example, the first pass calculates all the distances of the objects in the scene and the second pass determines the closest.

In summary, the conclusions drawn from the analysis of the query processing section were that extended SQL is a suitable query language for the *CHI project* as it conforms to the OGIS specification for extending SQL with spatial data types and that most suitable selection operation is the window query because it will enable location-based queries to be carried out based on the area of the window. To carry out the spatial join queries, the most efficient algorithm, the tree-matching algorithm, will be used. This is because the datasets in the system will have pre-existing indexes making them optimal inputs for a tree matching spatial join algorithm. This algorithm is also suitable for the window query operation as the tree-matching approach also performs very well when only the larger input is indexed.

3.1.5 Directional Query Processing

Direction relations refer to the relative positions of objects in space. These relations are used to determine the directional properties that exist between groups of objects in a two-dimensional space.

The most pressing problem associated with direction relationships is the absence of defining standards; there is a general set of imprecise definitions that have spawned a myriad of different approaches. When defining directiontional relationships the foremost requirement is a definition of the type of directional relationships that are required, of which there are three. Absolute direction uses a global reference frame to define direction, e.g. *North, South, East and West* regardless of orientation. Object-based direction uses the concept of a target and reference object, where the target's direction is defined in relation to an oriented reference object, e.g. *front, back, left, right*. Viewer-based direction is direction that is defined from the viewer's perspective and is relative for each viewer of the scene.

In the literature it is shown that there are two common models used to define directional relationships; the projection-based model and the cone-based model. There are three projection-based approaches discussed in the literature review, the projected point approach, the direction relation matrix and a method of representing direction as a spatial object. The project requires a method of defining directional relationships from the viewer's perspective. After careful consideration, it was decided that for the purposes of this research, the most suitable approach for determining direction between the viewer and the CH data point was to use the approach taken by [Shekhar and Liu, 1998] where direction is modelled as a spatial object. This method was chosen because it is possible to define relations using all reference frames thus allowing queries to be formulated from different perspectives. Another deciding factor was because this approach uses MBRs to represent the spatial objects and this approximation method is available in Oracle.

3.2 Synthesis

The purpose of the previous section was to provide an analysis of the technologies that are available to meet the requirements of the proposed research project. The most suitable technologies were selected based on a set of criteria established as a result of an analysis of the literature review including the availability of these features in current systems. In the following sections, the proposed design of the *CHI system* is described based on the initial proposal and the analysis of the literature review.

3.2.1 Conceptual Design

The literature shows that the idea of publishing CH data on the web using virtual reality (VR) has become popular. The *Eirnet Project* [http://www.dmc.dit.ie/], which was developed as a starting point of the *CHI project*, is one such system. The *Gallery Project* [Mc Atamney, 2004] is another project that uses VR to enable users to navigate spatially around a 3D model of the National Gallery of Ireland (NGI). This application allows the curator of the gallery to virtually hang paintings in different locations inside the gallery to check if they are suitable before they are physically moved. This saves the curator a considerable amount of time when organising an exhibition.

The system described in this dissertation provides a framework that enables the user to navigate spatially in a 3D environment, while querying a CH dataset based on a contextual query where the primary parameter to the query is location. The results of such a query are subsequently exhibited to the user by publishing the relevant information in a separate pane of the system that represents the mobile devices being simulated (See Figure 3.1).

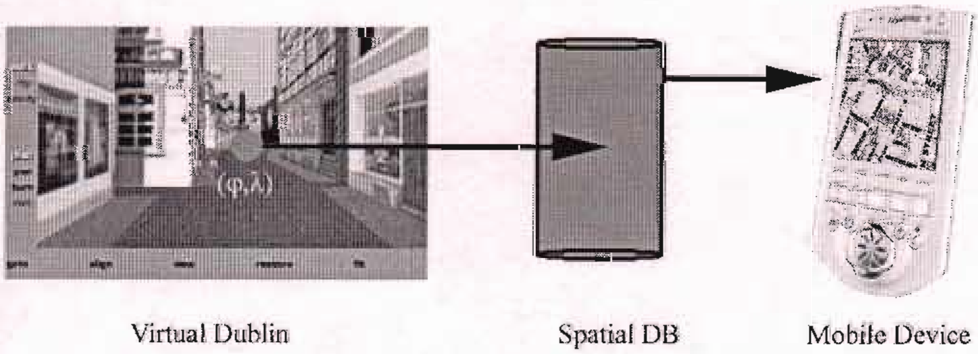


Figure 3.1: Conceptual Overview of the CHI Technology Demonstrator

The research aims to integrate the technologies in a novel and innovative way. The project integrates technologies from the area of computer graphics with technologies from the area of spatial database science to create a web-based spatial application that is capable of delivering a rich content to the user in the form of a narrative based on the user's context. The context in this case is the user's position and profile. The main aspects of the systems design are:

- Dynamic generation of VR model from database of geometries
- Real-time extraction and publishing of CH data stored in a database
- Integration of a VR model with a spatial data engine enabling spatial queries to be carried out within the VR model.

This section describes the conceptual design of the *Initial Technology Demonstrator* (Figure 3.2) based primarily on the "*Functional and Technical Specification*" for the project and was subject to the findings in the literature review. This includes a description of the client layer, application layer and database layer.

3.2.1.1 Client

The initial implementation of the system was based on the three-tier model, namely, the client layer, the middle layer and the database layer. The client consists of a standard web page with an embedded 3-D representation of Dublin that enables the user to navigate spatially through the model and retrieve information based on their location. It was decided that the client would also have a content pane to be used to display the information to the user.

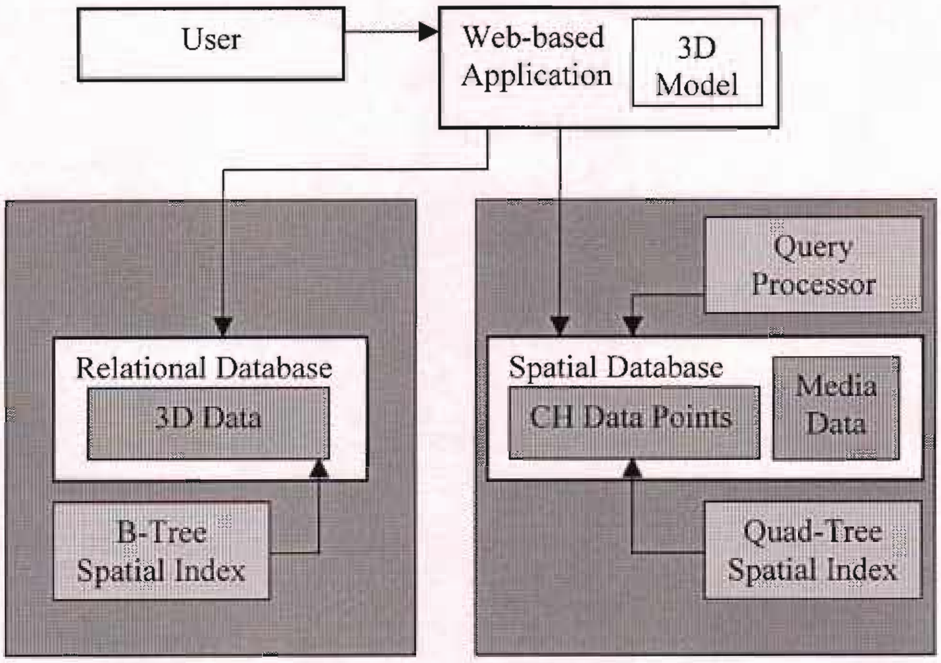


Figure 3.2: Conceptual Overview of CHI System

3.2.1.2 Application

The middle tier of the system's design comprises of an application that manages all the communication between the client and the database layers of the system. Using this approach, the load of the system will be focused on the middle tier, allowing a thin client to be developed.

The processing associated with the generation of a VR model from the database of geometries will be carried out in the 3D module of the middle tier and all spatial query formulation and CH data processing will be carried out in another part called the CH module of the middle tier.

The 3D module of the system will be responsible for extracting the geometries needed to build the 3D model from the database, rendering it into a specific format and displaying the model in a web-browser. The CH module of the system will be responsible for formulating the spatial queries based on the user's location, sending the queries to the database, and processing the results of these queries.

3.2.1.3 Database

As a result of the literature review analysis, it was decided that the database tier of the system would consist of an Object Relational Database Management System (ORDBMS). The database will contain Cultural Heritage (CH) data points that represent CH artefacts present in the system. Spatial data types will be used to represent each artefact and each will have a position associated with it.

The spatial data types will be indexed using a Quadtree spatial index to aid spatial query processing. Quadtrees index data with respect to its geographic location. The spatial queries will be processed using topological operators to query for spatial intersection and metric operations to determine distances within the model. All of the database tasks mentioned above will be carried out using an extended version of SQL for spatial data.

The ORDBMS will also contain all the multimedia data associated with the system. This will include images, text documents and video data. These objects will be read out of the database in real-time, and displayed to the user on the client pane.

A separate database will contain all the geometric data associated with the 3-dimensional model. This data will be stored in relational tables and extracted in real-time as needed. This data will be indexed using a regular B-tree index that indexes the data with respect to the position of the data in the table. An overview of the *CHI system* is illustrated in Figure 3.2 where all communication is via the http protocol.

This concludes the synthesis section of the analysis and synthesis Chapter. The result of this is the conceptual design of a system that will be capable of delivering a rich multimedia content to the user of the system based on their context, where context is primarily based on location. The following Chapter describes the design and implementation of the *Initial Technology Demonstrator.*

Chapter Four

Design and Implementation

4 Design and Implementation

4.1 Development Overview

The Objective of the *CHI project* is to research a methodology for publishing Cultural Heritage (CH) information on the web. The catalyst that triggers the querying and presentation of this cultural heritage data to the user is spatial navigation (movement) along with the historical temporal movements of the user.

The project builds on research into context-aware environments. Context-awareness is a challenge for mobile distributed computing to exploit the changing environment with a new class of applications that are aware of the context in which they are run. Context-aware systems change with the given environment and the location they occupy. Context encompasses more than just location. Factors also include such things as a user's profile, preferences and previous data retrieval history.

The *CHI project* is also concerned with the development of an ability to create narrative in a dynamic, interactive, hypermedia environment. As such, it researches areas as diverse as interactive narrative (Digital Storytelling) to explore new ways of creating narrative coherence and meaning in a Cultural Heritage environment.

Dynamic websites are a familiar concept to most web designers: - the content of the site is generated dynamically according to user demand or input. The web/hypertext paradigm, in the context of a spatial navigation interface, is one that is considered for this project- i.e. the fluidity of a dynamic website in terms of content, married to the narrative intelligence of a computer-game, navigated spatially [Carswell *et al*, 2001].

To navigate spatially and to receive CH data in real-time, which have relevance to the area the user occupies, the user is required to have some form of mobile device to receive the specific data. The user's device also requires a GPS (Global Positioning System) receiver that determines the position of the device. While it is technically feasible to meet these requirements such an implementation was not considered to be viable, in the context of the project, due to limited bandwidth availability and the cost of transmitting necessarily large amounts of data.

Instead, the *CHI system* uses simulated mobile devices to represent the actual devices. Using simulated devices made it possible to demonstrate how CH content could be adapted for real mobile devices in a spatial context, in particular the Personal Digital Assistant (PDA). The devices are embedded in the web-browser or application where all the parameters relevant to the device are available and are displayed one at a time. To represent the spatial navigation aspect of the system, VRML (Virtual Reality Modelling Language) is used. It was decided by the research team to create a 3D model of Dublin to represent the spatial dynamics of the physical space. The navigation through this space forms the basis for the queries that will be constructed to retrieve all the relevant data associated with any given point in the model.

The content that is dynamically compiled and presented to the user is stored in a spatial database. Unlike the traditional relational database approach of querying the data by comparing field values, the data is queried spatially. It is queried in relation to the location of the data relative to the user's viewpoint. Subsequently, each piece of data has some form of spatial attribute to identify it. This information is used in the specification and design of an egocentric (First Person) spatial interface. A demonstrator of the navigation capabilities of such a spatial model has been developed, based initially on the facts surrounding the *"Easter Rising 1916"*.

In this chapter, the development of the CHI framework is described in detail. This includes an overview of the datasets present in the system and the design of a three-tier architecture for the framework. This is followed by a description of the development of the content management aspects of the framework. Subsequently, an evaluation of the *Initial Technology Demonstrator* is performed, followed by the design and implementation of the Revised Technology Demonstrator. Finally, the development of the egocentric query processor is presented in detail.

4.2 Overview of Data Sets

The CHI database contains two main types of data, Spatial and Multimedia data. The spatial data are comprised of geometric and alphanumeric data. The geometric data consist of polygons, points and lines. Polygon data are used to represent the building blocks, vegetation and water areas. Point data are used to represent the positions of the cultural heritage data and the alphanumeric data are used to describe this data. The multimedia cultural heritage data are pictures, text, music and video files. This section describes the data sets that are used in the project.

4.2.1 Vector/Map data

This data comprises a set of vectors, also called geometries, which represent the building blocks, vegetation areas, and the water areas of the inner city area of Dublin. The vectors are obtained by vectorizing a street map using a raster to vector conversion program called R2V (Raster to Vector) by Ablesoft. Using this software a member of the research team was able to vectorize the map to a high quality and subsequently make any manual changes necessary buy using the manual editing tools that are also part of the program. The geometries are extracted into a standard flat file data format using the software and subsequently placed into a data file and loaded into the database with the use of a control file (a control file contains the target table and the rules for loading the data). There are two main types of data in this data set.

- *Polygon Data* – These data describe the geometries that can be represented by a closed polygon. The parts of the model that are represented by polygons are the buildings, water and vegetation areas.
- *Point Data* – These data describe the geometries that can be represented by a single point. Cultural Heritage (CH) information points in the model are represented by point data.

4.2.2 Multimedia data

These data are displayed to the user of the system when a data point is encountered. The data come from many sources including books, websites and photography archives. They are principallycomprised of four types of multimedia files in a variety of file formats.

- *Images* – The images in the *CHI system* are recorded under nemerous formats, principally jpg and bmp files.
- *Documents* – Text documents are used by the system to manage all types of text required to deliver the narrative.
- *Sound Files* – These files are used to supply the sound in the system; they are in mp3 format because of their economic size and high quality sound reproduction.
- *Video Files* – These are used to supply the users with CH footage from the *"Easter Rising 1916"* in Dublin; these are in mpeg video format.

4.3 Design of Initial Technology Demonstrator

The initial design of the CHI Technology Demonstrator was an aspect of the project that needed much consideration because it was a major decision point in the design of the entire project.

Conceptually, the Initial Technology Demonstrator allows a user to navigate through a virtual 3-Dimensional (3D) model of Dublin streets and view data relevant to their location in this space. The virtual streets contain the facades of buildings constructed and rendered using VRML (Virtual Reality Modeling Language). As the user moves within this 3D world, his location or virtual space coordinates are concurrently passed as parameters for the construction of a query to the spatial database. The query then retrieves all data from the database, within a certain tolerance, of the user's position in virtual 3D space, e.g. retrieve all data within +/-10m of these coordinates. The result of the query is returned to the user's hand-held (mobile) device, which for the purpose of the *CHI project*, is a series of simulated hand held devices, displayed in parallel with the VRML 3D world in a standard web-browser.

In this section, the design of a system that meets the requirements of the conceptual design is detailed. Initially, there is an overall outline of the design. Following this, each component of the system is discussed and the reasons for choosing these components are given. Finally, the communication paths between the components of the system are described.

4.3.1 Three-Tier Architecture

The *Initial Technology Demonstrator* uses a three-tier architecture, composed of the client layer, the application server layer and the database layer. This architecture focuses the load on the application server layer of the system, allowing for a thin client that is necessary in the simulation of a mobile context [Bertolotto *et al*, 2001]. All communications between the client layer and the database are conducted through the application server layer. In addition to this, it was decided to use this type of architecture because the processing load would be balanced, as each tier of the system will reside on a separate computer.

The overall design of the Initial Technology Demonstrator is illustrated in Figure 4.1.

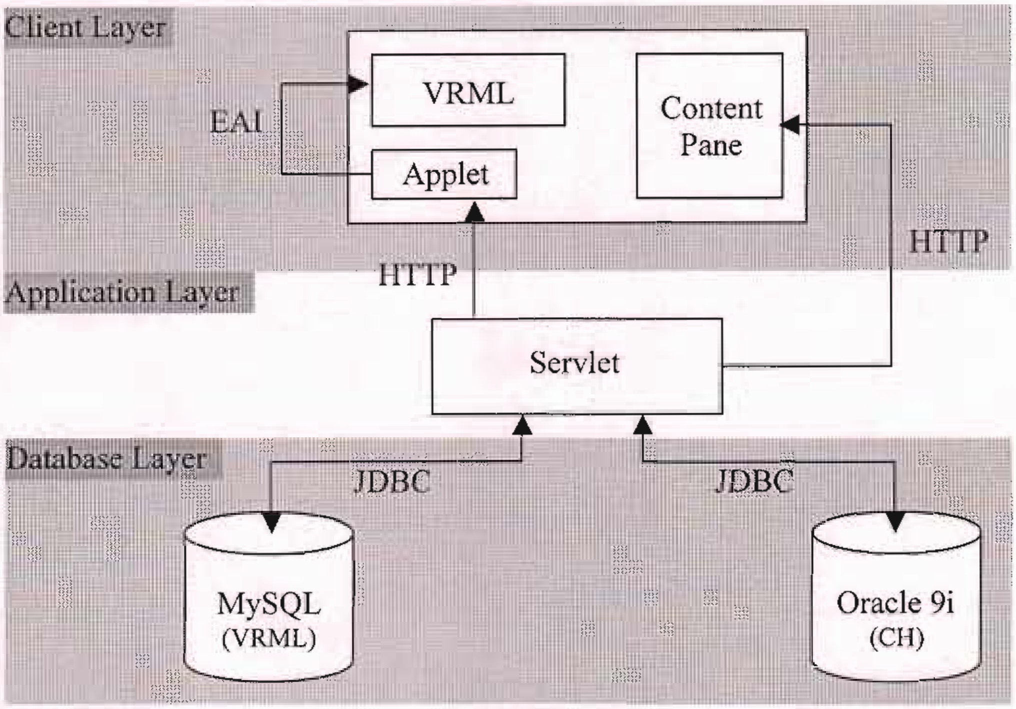


Figure 4.1: Design of Initial Technology Demonstrator

The three-tiers of the *Initial Technology Demonstrator* are discussed in the following sections and the components of each tier are detailed. The communication between components internally in each tier is discussed and following that the communication between each tier is described.

4.3.1.1 Client Layer

The client layer of the *Initial Technology Demonstrator* consists of three components. The first component is a VRML browser and below this is a java applet. The third component is a separate frame called the content pane and all are displayed in a single web page. This is illustrated in Figure 4.2.

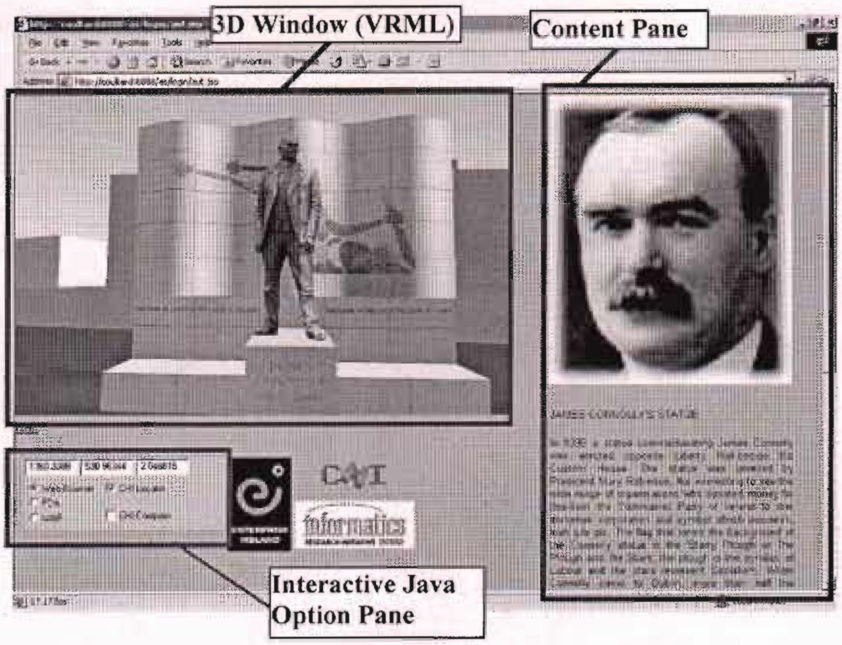


Figure 4.2: Client Layer of CHI System

4.3.1.1.1 3D Window

The Virtual Reality Modelling Language (VRML) is the language used as the implementation platform for the 3D spatial representation component of the *Initial Technology Demonstrator* because it is a scene description language that describes the geometry and behaviour of a 3D scene or world. In the case of the *CHI project*, the 3D scene models the streets of Dublin. The VRML interface is only available in the web browser. It is not available on any of the simulated mobile devices because of transmission speed restrictions. Also, to reduce transmission costs within the system, automatic LOD (Level of Detail) node functionality is used in the VRML model. This ensures that only objects that are within a certain distance of the user's viewpoint are displayed in the VRML browser [Diehl, 2001]. This reduces the transmission costs and loading time for the objects in the model.

4.3.1.1.1.1 Static VRML

The use of static VRML to model the 3D visualisation of the query space for the *CHI system* is one approach that was considered when compiling the *"Technical and Functional Specification"*[Carswell *et al*, 2001] for the system. This approach involves developing a single static 3D scene to represent the query space. This method is used widely in the area of 3D modelling and is useful if the required model does not need to be continuously updated to meet the needs of the project. In the case of the *CHI project*, using a single 3D scene was not suitable because the static model would be built in incremental steps and would need continuous updating. This was not suitable because one of the primary objectives of the project was to build an adaptable system that enabled the potential users of the system to insert their own geometric into the database (e.g. Cork, Limerick). From this, the VRML code generation would be automatic and the user would require little knowledge of VRML. Although the static model approach has the benefit of being less complicated because it involves managing one static file as opposed to a database and many associated files, using it would not achieve the objective.

This need for a fully adaptable and easily updateable 3D environment to represent the spatial aspect of the *CHI system* led to the concept of creating a system that dynamically builds the VRML model from a database of geometries. In this way, the VRML model could be updated on the database layer of the system and the changes would propagate through the system. This approach is discussed in the following section.

4.3.1.1.1.2 Dynamic VRML

This part of the design involves using the three-tier architecture to dynamically generate the VRML model on the middle-tier of the system from data on the database-tier and visualise it on the client-tier. This approach incorporates a completely dynamic design where the geometric aspect of the model is stored in the database [Carswell et al, 2001]. Obtaining the geometries is a semi-automatic process that involves vectorising a map of Dublin, cleaning the geometries using the editing software and loading them into the database (Section 4.2.1). At program execution time, the geometries are extracted from the database and read into a VRML structure. This VRML scene is then displayed in the VRML browser. This process is detailed in section 2.4.

Incorporating this type of design into the *CHI system* ensures that it is an adaptable system because the data in the database tier can be easily adapted to reflect current changes in the city layout. Also, the database tier could now be adapted to represent a different city altogether.

4.3.1.1.2 Interactive Java Option Pane

The *Interactive Java Option Pane (IJOP)* has two major functions as part of the client layer. Its primary function is to communicate with the VRML browser to obtain the coordinates of the user in the model. It then communicates the coordinates to the application server layer, where they are used as parameters to a spatial query (Illustrated in Figure 4.1). Its secondary function is to enable the user to interact with the system and allow him/her to activate parts of the system and input information to the application (See Figure 4.2). It also gives the user the option to select a series of simulated mobile devices for displaying the data. The simulated mobile devices include a WAP Enabled Mobile Phone and a Personal Digital Assistant (PDA).

4.3.1.1.3 Content Pane

The *Content Pane* has one function in the *Initial Technology Demonstrator* and that is to display data, based on the user's context. The user's location in space, the particular mobile device being modelled and the user profile and preferences (e.g. the user's interests, age, task, etc.) determine the user's *context*. As the user progresses through the VRML model he/she may interact with CH artefacts. In such a case, the relevant data are automatically extracted from the database and displayed in the content pane.

The content pane has the ability to present the CH data in three ways. The first is the standard web-browser. The second is a simulation of a WAP enabled mobile phone and the third a simulation of a Personal Digital Assistant (PDA). Consideration with regard to the mobile device being modelled determines the type and format of the data that are passed to the content pane. The user controls switching between the mobile devices via the interactive java option pane.

4.3.1.1.4 Location Sensor Communication

To simulate the presence of a GPS (Global Positioning System) receiver in the *Initial Technology Demonstrator*, a location sensor is used. The location sensor simulates the presence of the GPS receiver by determining the coordinates of the user within the VRML model. The position of the user is then used to build the complex spatial queries that are executed on the application-server layer. In VRML, this sensor is called a proximity sensor [Ames *et al*, 1997].

The VRML Proximity Sensor implements algorithms that derive the distance and direction from a given point to the user's viewpoint, hence their coordinates within a virtual space. The values of the variables are expressed in Cartesian *(x,y,z)* coordinates which increase in value with increasing distance of the viewpoint from the proximity sensor.

The communication between the VRML browser and the *IJOP* is accomplished by using a Java API called the VRML External Authoring Interface (EAI) [McCarthy and Descartes, 1998]. The EAI contains methods that allow the *IJOP* to detect and register with the VRML browser. Once this is achieved it is possible to pass the coordinate data to the *IJOP*. The applet then communicates the *x,y,and z* information to the application server layer via the HTTP protocol. The detail of this communication is covered in Section 4.4.

This concludes the design of the client layer of the *Initial Technology Demonstrator*. The following section describes the design of the application server layer.

4.3.1.2 Application Server Layer

The Application Server Layer of the *Initial Technology Demonstrator* consists of an application server and a series of Servlets. An application server is a piece of software that provides middleware services that allow deployment and management of personalised applications in a standardised environment on the web. This is where the dynamic VRML scene generation is carried out as well as the query building and result formatting. This tier of the system handles most of the communication between the components of the system.

4.3.1.2.1 Java Servlets

Java Servlets are used as the server side component of the *Initial Technology Demonstrator*. Servlets are a server-side technology that provide a wide range of advantages when compared to other approaches. Servlets are portable and very powerful because they are well defined and use the full power of the core Java APIs [Hunter and Crawford, 2001]. They are also very secure and are easily extensible. The *Initial Technology Demonstrator* utilises the Servlet technology to act as the server-side component that brokers as a hub for most of the communication in the system. This is shown in Figure 4.3.

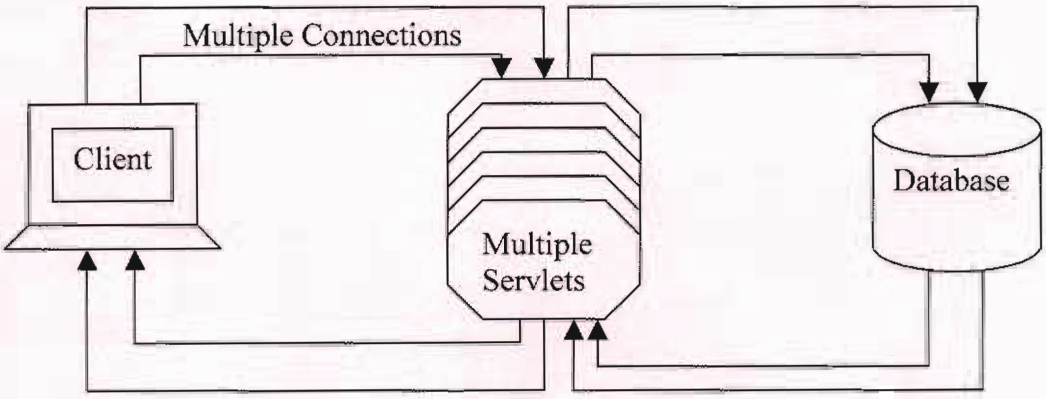


Figure 4.3: Communication using Servlets

Session Servlet

For each user of the *CHI system*, an instance of the *Session Servlet* is created. This Servlet has three main functions. The first function is to deal with all communication between the client and the database. Its second function is to monitor the position of the user within the VRML model and its third function is to formulate and process SQL queries to the spatial database.

Content Servlet

The function of the *Content Servlet* is to retrieve content from the database layer and display it on the client layer. When a query to the database returns *true*, meaning the user is interacting with CH data, the *Content Servlet* is requested to locate the data in the database, retrieve it and display it in the content pane. Theoretically, this sounds a trivial task but in practice it is complex. The detail of this is covered in section 4.4.

Apache Tomcat

To publish Servlets on the web a Servlet container is used. The most suitable Servlet container for the *Initial Technology Demonstrator* at time of development was *Apache Tomcat*. This was because it had a small installation footprint; it was open source (or free) and was known to be reliable, i.e. is developed in an open and participatory environment and is used to publish Servlets only [Apache Tomcat, 2004]. Therefore, for hosting the HTML pages in the system an *Apache HTTP Server* was also used. This server was selected for similar reasons.

This concludes the design of the application server layer of the *Initial Technology Demonstrator*. The following section describes the design of the database layer.

4.3.1.3 Database Layer

The Database Layer of the *Initial Technology Demonstrator* consists of two databases, the Cultural Heritage (CH) Database which is an Object Relational Database Management System (ORDBMS) and the VRML database, which is an RDBMS.

An ORDBMS is a database that incorporates Abstract Data Types (ADT) capable of storing objects. It also has a set of extended SQL functions that allow these objects to be inserted, deleted, modified and queried within the database. This is possible because ORDBMS have the ability to add and remove Abstract Data Types (ADTs) from the database schema without interfering with the database structure. The fact that it is a relational database means that it also has the added functionality of the traditional Relational Database Management Systems (RDBMS)(See section 2.3).

CH Database

The CH database has two functions. Its primary function is to store and manage all of the CH data in the system. This includes spatial data and CH (multimedia) data. Each piece of CH data in the database contains a spatial component. This spatial component identifies its location within the VRML model. The spatial component is either a point or a circle (buffer) around the point, to identify the area that the piece of data is relevant to.

As the user progresses through the model, the viewpoint of the user represents the query window. This window is then queried every five seconds against the CH data layer in the database to determine if data is available for this area of the model. The queries processed in the CH database are formulated on the application server layer, processed in the CH database and the results subsequently presented to the user on the client layer via the application server layer.

VRML Database

The VRML database has one function and this is to store the geometric data that is used to dynamically create the VRML model (virtual database). These geometric objects are stored in the database and rendered in real time to produce a VRML model on request. This data consists of mainly building boundary data (polygons) and is obtained by digitising physical maps and loading the data into the database. The process is described in section 4.4.1.

The following section gives an overview of the Oracle 9i database that is used by the *Initial Technology Demonstrator* to handle the spatial data requirements of the system.

4.3.1.3.1 Oracle 9i Spatial

It was decided that the *Content Database* component of the *Initial Technology Demonstrator* would be an Oracle 9i ORDBMS with the Spatial Data Option (SDO). The SDO is an add-on to the extendable Oracle DBMS that makes it possible to insert, store, manipulate and query spatial data in the database as they are represented in physical space [Sharma, 2001]. *Oracle Spatial* conforms to the OGC specification (See section 2.5.2.1) for spatial geometry building blocks [Rigaux *et al*, 2002].

This database was chosen because it is integrated into the extensible Object Relational Database Management System (ORDBMS), which gains access to the full functionality and security of the underlying DBMS.

4.3.1.3.1.1 Spatial Query Language (Extended SQL)

The SDO provides an extended SQL (See section 2.5.2) query language with a set of functions for querying and indexing spatial data. These functions are used to query data in relation to their position in space rather than their position in the table. *Oracle Spatial* uses z-ordering tree spatial indexing (see section 2.6.1.5) to index the data [Rigaux *et al*, 2002].

4.3.1.3.2 MySQL

It was decided that the VRML database component of the *Initial Technology Demonstrator* would be built using a MySQL Relational Database Management System. The decision to use this database was based on the knowledge that the VRML database would need constant updating and as MySQL has a user-friendly interface for inserting and updating tables it was considered to be the most suitable at the time.

This concludes the design of the database layer of the *Initial Technology Demonstrator*. It also concludes the section describing the overall design of the demonstrator. The following section describes in detail the implementation of the three tiers of the *Initial Technology Demonstrator*.

4.4 Initial Technology Demonstrator Implementation

This section describes the implementation of the *Initial Technology Demonstrator*. Initially, the steps involved in developing the client layer are described. This includes the process involved in digitising a map of Dublin to obtain the geometries needed for the VRML model. Following that, the development of the application server layer component is described. This includes the development of the Servlets that reside on the middle tier of the system. Finally, the implementation of the database components of the system is described. This includes the development of both the spatial and multimedia components of the system.

4.4.1 Maps to 'e-Maps'

The 3D elements of the *Initial Technology Demonstrator* are the result of a raster (discreet picture elements) to vector (visual objects represented as points and lines) conversion process. The two dimensional point and line data are used as a cross-section list for an extrusion function to add height to the 2D polygon elements. This approach is used to constitute the Dublin city centre.

To make the data accessible for further image processing, the analog image has to be converted, quantized and stored into a raster format. The process is automated and involves the use of several image-processing operations in order to extract relevant information from the image. A variety of, preferably unsupervised, algorithms must also be utilised to pre-process the pixel data.

The modelling of urban space requires a specialized set of work procedures. The acquisition of vectorized data and the necessary procedures for the Dublin 3D prototype are discussed in this section.

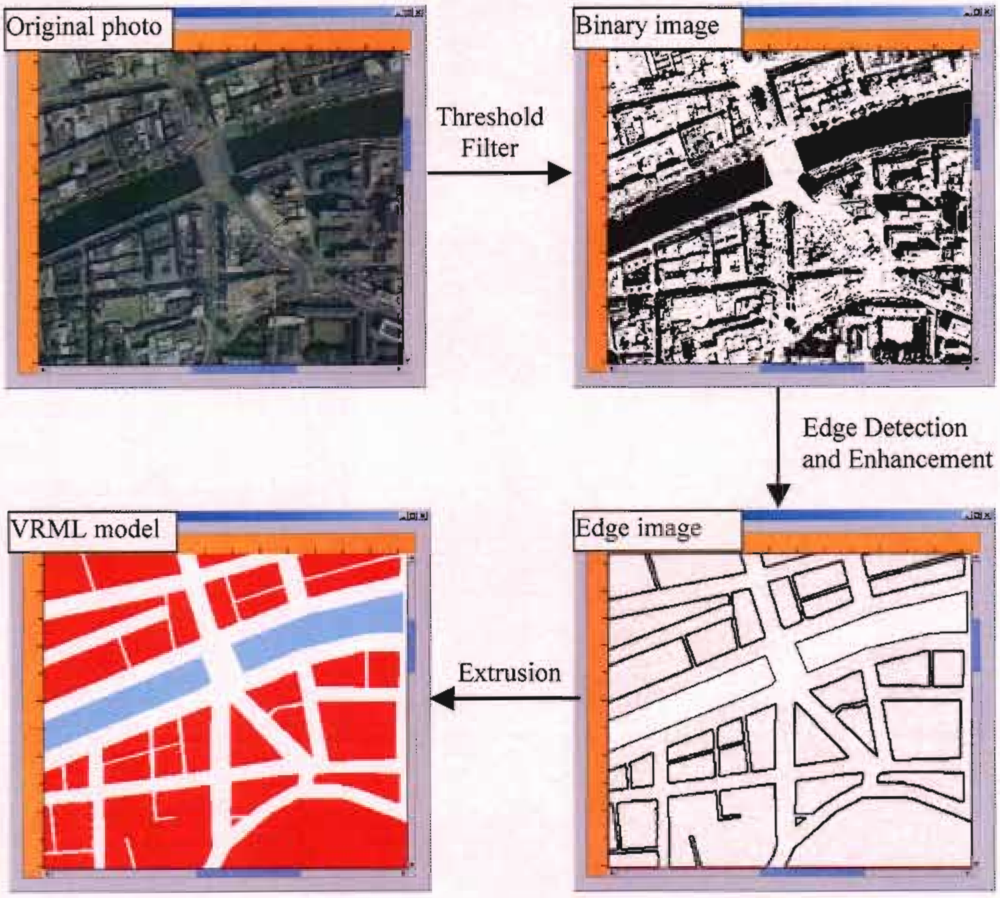


Figure 4.4: Four Main Stages of Map Digitisation

The pre-processing of the scanned samples removes unnecessary detail as well as dust and scratches, which were either present in the original source image or produced as an artefact of the scanning process. This work was carried out by another member of the research team (Marco Neumann).

The colour image is decomposed to separate the colours so that each colour layer can be vectorized independently. The filter function *"de-speckle"* is used to remove noise from the image and smooth and rough edges. This filter detects the edges in an image and blurs all of the selection except these edges. It has the effect of removing noise while preserving detail [Foley et al, 1990; Adobe, 2004]. During the conversion steps for the *Initial Technology*

Demonstrator, it was found that a further conversion into an 8-bit greyscale image is more effective for applying threshold and background removal to improve the image quality for vectorization. Even better was an additional conversion into 1-bit (binary) monochrome image layer, which respond best to automatic edge detection algorithms in best practice benchmarking tests [Suharto et al, 1996].

The raster-to-vector conversion itself consists of three basic operations: skeletonization or line thinning, line extraction or vectorization, and topology reconstruction.

Line thinning is the process that automatically thins the lines in a raster object until they are uniformly just one pixel wide. The line thinning process works from the edges of the line inward to the centre, successively peeling off outside layers of cells. Thus, the thinned raster line represents the centre-line of the wider lines of the original drawing and scanned image.

Line extraction is the automatic process of identifying a particular series of data entities or coordinates that constitute an individual line segment as portrayed on the input document. For input, it requires a raster object containing lines that are continuous and uniformly one pixel wide, such as the output from the raster line thinning process. It is necessary to be certain that the boundaries of the mapping units form closed polygons. It is also important that all unnecessary notations and text have been removed with the appropriate raster editing tools.

The line extraction process may create unwanted vector elements corresponding to any features that remain; the user may remove these later in the vector editing process. An automatic vector tolerance can be set as part of the vectorization steps. The tolerance can vary; the lower the tolerance, the more precise the vectorization will be, but also more vertices will be generated for the vector output file [Suharto et al, 1996].

Topology reconstruction is the process of determining the adjacency relationships among line segments. The individual line segments are joined into whole line features, and maps are built as continuous area representations. This procedure is performed using the editing tools available in Ablesofts R2V software [http://ablesw.com].

Following the digitisation of the map, the polygon point list extracted from the process is loaded into the MySQL database as a cross-section list. At program execution time the list is extracted from the database into a VRML format, where the two-dimensional vectors are extruded to represent the 3D model of Dublin City Centre.

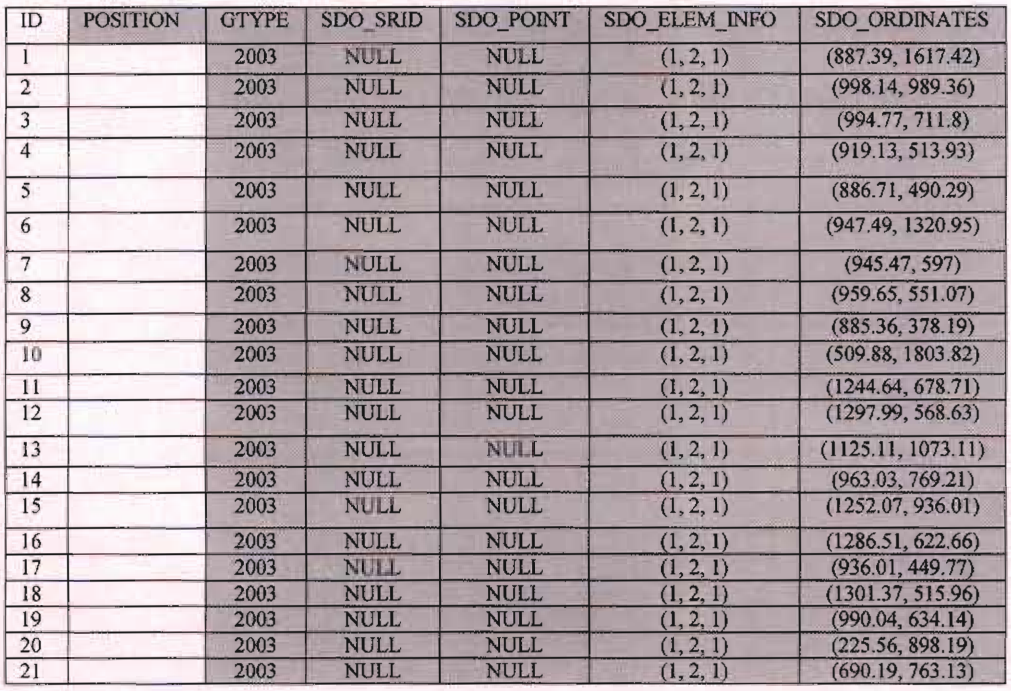
4.4.2 CH Database Development

In this section the procedures involved in the implementation of the CH database are described. Initially, there is an overview of *Oracle spatial* data option (SDO). Following that, the procedures involved in creating, loading and indexing spatial data are presented. The final part of this section gives details of the procedures involved in loading and extracting CH data to/from the CH database.

Spatial Data Option (SDO)

SDO is the spatial extension to the Oracle 9i database. All spatial data is created, stored and indexed using SDO. All spatial objects are stored using an ADT called SDO_GEOMETRY. The available geometry types include Points, LineStrings and Polygons or collections of each. SDO provides the functionality for Oracle to insert, retrieve, manipulate and query *Geometry* types in the database [Sharma, 2001].

The ADTs are contained in the database as MDSYS.SDO_GEOMETRY objects and are identified by the SDO_ELEM_INFO array that accepts three parameters. The SDO_GEOMETRY type also has a SDO_ORDINATE array. This array holds the spatial coordinates of the geometry that is defined in the SDO_ELEM_INFO array. This is illustrated in the Figure 4.5.



ID	POSITION	GTYPE	SDO_SRID	SDO_POINT	SDO_ELEM_INFO	SDO_ORDINATES
1		2003	NULL	NULL	(1, 2, 1)	(887.39, 1617.42)
2		2003	NULL	NULL	(1, 2, 1)	(998.14, 989.36)
3		2003	NULL	NULL	(1, 2, 1)	(994.77, 711.8)
4		2003	NULL	NULL	(1, 2, 1)	(919.13, 513.93)
5		2003	NULL	NULL	(1, 2, 1)	(886.71, 490.29)
6		2003	NULL	NULL	(1, 2, 1)	(947.49, 1320.95)
7		2003	NULL	NULL	(1, 2, 1)	(945.47, 597)
8		2003	NULL	NULL	(1, 2, 1)	(959.65, 551.07)
9		2003	NULL	NULL	(1, 2, 1)	(885.36, 378.19)
10		2003	NULL	NULL	(1, 2, 1)	(509.88, 1803.82)
11		2003	NULL	NULL	(1, 2, 1)	(1244.64, 678.71)
12		2003	NULL	NULL	(1, 2, 1)	(1297.99, 568.63)
13		2003	NULL	NULL	(1, 2, 1)	(1125.11, 1073.11)
14		2003	NULL	NULL	(1, 2, 1)	(963.03, 769.21)
15		2003	NULL	NULL	(1, 2, 1)	(1252.07, 936.01)
16		2003	NULL	NULL	(1, 2, 1)	(1286.51, 622.66)
17		2003	NULL	NULL	(1, 2, 1)	(936.01, 449.77)
18		2003	NULL	NULL	(1, 2, 1)	(1301.37, 515.96)
19		2003	NULL	NULL	(1, 2, 1)	(990.04, 634.14)
20		2003	NULL	NULL	(1, 2, 1)	(225.56, 898.19)
21		2003	NULL	NULL	(1, 2, 1)	(690.19, 763.13)

Figure 4.5: MDSYS.SDO_GEOMETRY column format

4.4.2.1 Creating CH Tables

Creating tables with a spatial column in Oracle is similar to creating any other type of column. There is no difference in the creation of tables for different types of geometries. This distinction is made when the actual geometries or objects are being inserted into the tables; this process is covered in section 4.4.2.2.

Oracle contains a number of predefined data types such as Number, Float, Char and Varchar2 [Mishra and Beaulieu, 2002]. During table creation, each column in the table is given a name, type and size. The capability for Oracle to facilitate spatial data is made possible by the addition of another separate schema into Oracle's extensible relational database schema. This schema is called MDSYS and it prescribes the storage, syntax, and semantics of supported geometric data types [Sharma, 2001]. This schema enables the user to create tables containing data types that are capable of holding spatial data. An example of a table that contains both standard data types and *Oracle Spatial* data types is shown in Figure 4.5. In this example, *ID* is a standard datatype and *Position* is a spatial datatype.

Tables that contain spatial datatypes or, as they are called in Oracle, *"Geometry objects"* are created in a similar way to standard data types. The only variation is the SQL syntax used to create the spatial column. Creating standard data types involves specifying the name and then the type along with the size as a parameter. Creating a geometry type requires that the name of the schema be specified (the MDSYS schema) and then dot notation is used to access the schema objects that define spatial datatypes. In Oracle this is SDO_GEOMETRY. The creation of the Cultural Heritage (CH) table is illustrated by the following SQL statement.

```
CREATE TABLE CULTURAL_HERITAGE (
      ID NUMBER (20) PRIMARY KEY NOT NULL,
      POSITION MDSYS.SDO_GEOMETRY NOT NULL,
      STREET VARCHAR2 (20) NOT NULL,
      BUILDING VARCHAR2 (20) NOT NULL);
```

Table 4.1: Create Table Procedure

The SQL syntax in Table 4.1 creates a table with a spatial column. The second column is the spatial column and is called POSITION and is of type MDSYS.SDO_GEOMETRY. There is no size specified because it is an

Abstract Data Type (ADT) and contains many standard data types in its implementation [Schneider, 1997].

4.4.2.2 Inserting Data

This section illustrates the process involved in inserting the spatial component of the CH data points. Inserting data into a spatial column in Oracle is more complex than the creation of the spatial column itself. This is because the parameters that define the type of geometry that is to be stored in the spatial field are specified at the insertion phase.

At the insertion phase the name of the spatial schema followed by the dot notation is used to access the spatial operators within the schema. The parameters are then passed into the operator. The MDSYS.SDO_GEOMETRY accepts five parameters. Two of the fields are of type NUMBER and the other three are object types. The fields are illustrated in Figure 4.5. The following section gives an example of inserting spatial data into a spatial table.

Inserting a Point

Inserting a Point into a spatial column is a special case. A point is an optimised geometry and is treated differently to any other type.

```
INSERT INTO CONTENT_INFO VALUES (1,
      MDSYS.SDO_GEOMETRY (3001, NULL,
      MDSYS.SDO_POINT_TYPE(234.3,34.4,876.2),
      NULL, NULL));
```

Table 4.2: Insert Point into Table

The code excerpt in Table 4.2 inserts a number into the non-spatial (ID) column and a *point* object into the spatial column. The SDO_GTYPE is set to 3001, indicating a 3D point, the SDO_POINT array contains the coordinates of the point and the SDO_ELEM_INFO_ARRAY and

SDO_ORDINATE_ARRAY fields are NULL. These are the required arrangement of parameters for defining an optimised point type.

The process of inserting multiple points into the table can be performed in two ways. A transactional insert approach can be used, which involves creating a file containing an individual INSERT statement for each point. This approach is tedious and would only be used to insert a small number of records. Alternatively, a bulk load program called *"SQL Loader"*[Geringer, 2001a] can be used. This is a piece of software supplied by Oracle and enables a large number of records to be inserted automatically. The program requires two files as inputs, namely the *Control File* and the *Data File*. The *Control File* describes the load to the program, what type of data is being loaded and specifies the path to the *Data File*. The *Data File* is a flat file that contains the coordinate data separated by a delimiter (CSV format), which tells the program where each point starts and ends. This method was used by the research team to load the point data and is the better approach.

Following the creation of a table with the spatial column and the insertion of the spatial data, the next step is to index the spatial data. The next section is an overview of the implementation steps involved to index the spatial column of the CUTURAL_HERITAGE table.

4.4.2.3 Spatial Indexing

This section describes the procedures involved in indexing the spatial column of the CH table. Initially, a description of the type of index that is used is given. Following this, the steps required to build this index in Oracle are explained.

Once spatial data has been loaded into the tables of the database, a spatial index is required if the tables are to be queried spatially. In the same way that regular indexes are utilised in databases to provide fast access to data, spatial indexes are required, for the same reason, to be specially constructed to provide an efficient data access method Spatial indexes allow users to treat data within a data-store as existing within a two-dimensional context. In the case of a spatial index, the index is built on the spatial column of the table to provide fast access to data in a particular region. *Oracle Spatial* uses a z-ordering tree (see section 2.6.1.5) spatial index [Rigaux et al, 2002] and the type of access method selected for indexing spatial data in the *Initial Technology Demonstrator* is the Quadtree.

- *Quadtree Indexing:* This type of index indexes data by successive decomposition of the coordinate space into tiles. The process of determining which tiles cover a given geometry is called tessellation [Geringer, 2001a]. Quadtree spatial indexes use the tiles that cover geometries as an approximation of that geometry. This is shown in Figure 4.6. Fixed tile indexing is recommended in preference to Hybrid indexing when using quadtree indexes. This is because hybrid indexing should only be used in rare circumstances e.g., when spatial joins are required between spatial layers whose optimal fixed level (SDO_LEVEL) are significantly different (4 levels or more) [Oracle, 2001]. The SDO_LEVEL refers to the level of decomposition of the space. For each addition (+1) to the SDO_LEVEL, the size of the fixed tiles is quartered. A comprehensive overview of quadtrees is given in section 2.6.1.1.

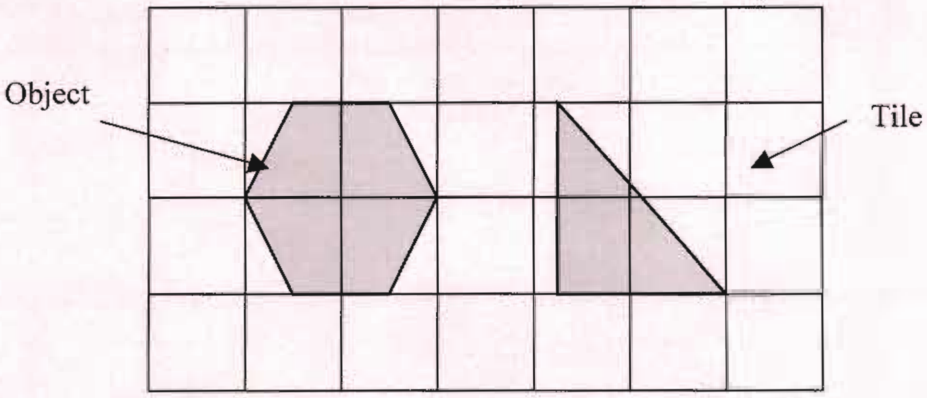


Figure 4.6: Quad-Tree Spatial Index

The *Initial Technology Demonstrator* uses a Quadtree index to index all the CH spatial data in the system [Gardiner and Carswell, 2003]. This index was selected because it has been shown in the literature that the Quadtree performs better when indexing simple objects (like points) and is more suited for data sets that are continuously updated as is the case of the CH data set. The following section details the process of indexing the spatial column of the CULTURAL_HERITAGE table in the CH database.

Creating a Spatial Index

In order for spatial queries to be executed efficiently, spatial indexes are created on the spatial column of the table. There is one prerequisite when creating spatial indexes in Oracle namely that is an entry must be added to the USER_SDO_GEOM_METADATA view for each layer or table that is to be spatially indexed. This entry is required and specifies the metadata associated with the data, such as the axis bounds of the grid that extends the layer. In the case of the CH database, there is an entry added for the CULTURAL HERITAGE table with an axis grid size of 3000m x 4000m and a tolerance of 0.005m, which is a measurement of accuracy that is used when querying the data. This transactional insert is illustrated in Table 4.3.

```
INSERT INTO USER_SDO_GEOM_METADATA VALUES (
       'CULTURAL_HERITAGE',
       'POSITION',
       MDSYS.SDO_DIM_ARRAY (
       MDSYS.SDO_DIM_ELEMENT ('X', 0, 3000,0.005),
       MDSYS.SDO_DIM_ELEMENT ('Y', 0,4000,0.005)),
       NULL);
```

Table 4.3: Inserting Geometry Metadata

Following the execution of this insert statement the CULTURAL_HERITAGE table can be spatially indexed using the *CREATE INDEX* operator in SQL. The syntax used is similar to any other Oracle *CREATE INDEX* statement, with some additional information. The additional syntax is *INDEXTYPE IS MDSYS.SPATIAL_ INDEX*. Optionally, many parameters can be specified but for the purposes of the CH database only the *SDO_LEVEL* parameter is necessary and is used to specify the level of tiling in the index. The spatial index is created as illustrated in Table 4.4:

```
CREATE INDEX CULTURAL_HERITAGE_INDEX
       ON CULTURAL_HERITAGE (POSITION)
       INDEXTYPE IS MDSYS.SPATIAL_INDEX
       parameters ('SDO_LEVEL=8');
```

Table 4.4: Creating Spatial Index

The SQL statement illustrated in Table 4.4 creates a Quadtree spatial index called CULTURAL_ HERITAGE_INDEX on the CULTURAL_HERITAGE table. It is created on the POSITION column of the table and the index type is specified.

After the SQL statement is executed, the query space is tessellated to a tiling level of eight; a spatial index table is created and populated. A B-tree index is then built on the table. The table is now ready to be queried spatially. At this point data can still be added to the table and the spatial index will expand to accommodate the new data.

This concludes the implementation section on spatial indexing. All spatial aspects of the CH database implementation have been described. The next section is concerned with the multimedia elements of the CH database.

4.4.2.4 Streaming Data Types

In this section, the implementation of the multimedia components of the CH database is described. Initially, there is an explanation of JDBC. This includes an example of a database connection using JDBC. Following this, the structures used to store multimedia objects in Oracle are described. This includes a description of the procedures used to insert and extract Binary Large Objects (BLOBs) and Character Large Objects (CLOBs) to/from the database.

All the multimedia files referred to in section 4.2 are stored in the Oracle database and include images, text documents, sound files, and movie files. These files have to be inserted into the database efficiently and stored in tables. On demand, the files have to be read out of the database and displayed on the client layer of the system, i.e. presented to the user.

The method chosen to perform these tasks is data streaming. Data streaming involves splitting up a large (binary) file into bytes and transmitting them one at a time until all have been streamed into the database. In the database, the bytes are reassembled into the original file. This method was used because the files need to be stored as complex binary data types in the database [Bales, 2002] and also because transmitting large binary files intact is not efficient.

In this section, the insertion and selection of large object data types is covered. There are two ways of achieving this depending on the type of file being processed. These are binary large object (BLOB), and character large object

(CLOB) respectivly. The difference between CLOB and BLOB is that the CLOB is subject to character set translation as part of Oracle's National Language Support (NLS), whereas a BLOB's data is taken verbatim; within Oracle the contents of a text document is transmitted one character at a time and each character is checked to correspond with the character set used. However, this is not the case with binary data. The binary data is transmitted exactly as it appears in the binary file and will not be changed due to its transmission.

To stream data, a connection to the Oracle database is required. This involves opening a connection on the database from the application and streaming data through it. This connection is achieved using JDBC. JDBC is a Java Application Programming Interface (API). The classes in the API are custom built for connecting to the database and include classes for a wide range of tasks. The next section is an overview of JDBC and the procedures used to connect to the database.

4.4.2.4.1 JDBC

The previous section on spatial data was concerned with creating tables, inserting data and indexing spatial data. This is achieved by using Oracle SQL plus, which is an application provided by Oracle to execute SQL and PL/SQL (Oracle's native scripting language) scripts [Mishra and Beaulieu, 2002] and therefore does not require the development of any secondary applications.

This section concentrates on the multimedia aspects of the system. This involves inserting images and other documents into the database and requires the development of secondary applications. These applications are developed in Java, and require a connection to the Oracle database.

To connect to the Oracle database a JDBC driver is used. This driver acts as an intermediary between the database and the Java application. Different JDBC drivers can be used to connect to the database depending on the architecture of the system. In the *Initial Technology Demonstrator,* a type 4 client-side thin driver is used. Type 4 thin drivers are pure java drivers and communicate directly between the database and the application, thus making the driver totally platform independent [Bales, 2002].

JDBC Connection

Due to the need for many connections to the database from the CHI application a utility class was developed and used to handle all connections. This class has a method called *connect* which is shown below. It illustrates the steps taken to make a connection to the database using a thin (type 4) JDBC driver.

```
Public Connection connect (String host, int port, String
sid, String username, String password)
            throws SQLException {
DriverManager.registerDriver (new
oracle.jdbc.driver.OracleDriver());
String database = "jdbc:oracle: thin: @" + host + ":" + port
+ ":" + sid;
OracleConnection conn =
(OracleConnection)DriverManager.getConnection(database,
username, password);
return conn;
}
```

Table 4.4: JDBC Connection Method

The example in Table 4.4, shows a method called *connect* that takes five parameters, the *host name*, *port number*, *sid (database name)* of the database, and the *username* and *password* for the database. It then registers the JDBC driver with *DriverManager*, gets a connection from the *DriverManager* and returns this connection.

4.4.2.4.2 Large Objects (LOBs)

In this section, the use of Large Objects (LOBs) for storing image files and text documents is explained. Initially, the use of BLOBs in the CH database is described. This includes the procedures involved in inserting and selecting BLOB data. Following this, the use of CLOBs in the CH database is described and the procedures involved in inserting and selecting CLOB data are presented.

In the CH database BLOBs are used to store images, sound and movie files. This data is streamed into the database and is taken verbatim. CLOBs are used to store the text files in the CH database. The text data is streamed into the database as characters and is subject to character set translation (text), which is part of Oracle's National Language Support (NLS).

LOB Locator

Accessing LOB data in the database involves using a locator that points to the actual data in a table. A locator is an internal database pointer and is created by inserting a new row into the table or by updating an existing row. Once the locator is created it is retrieved using *SELECT FOR UPDATE* SQL syntax to establish a lock on the record. Once a lock is secured, data can be inserted or selected as required. This process is common to all LOBs and is used in the following examples.

4.4.2.4.2.1 Binary LOBs (BLOBs)

In this section, the use of Binary LOBs in the *Initial Technology Demonstrator* is discussed. This includes a description of the procedures used for inserting and selecting BLOB data to/from the CH database.

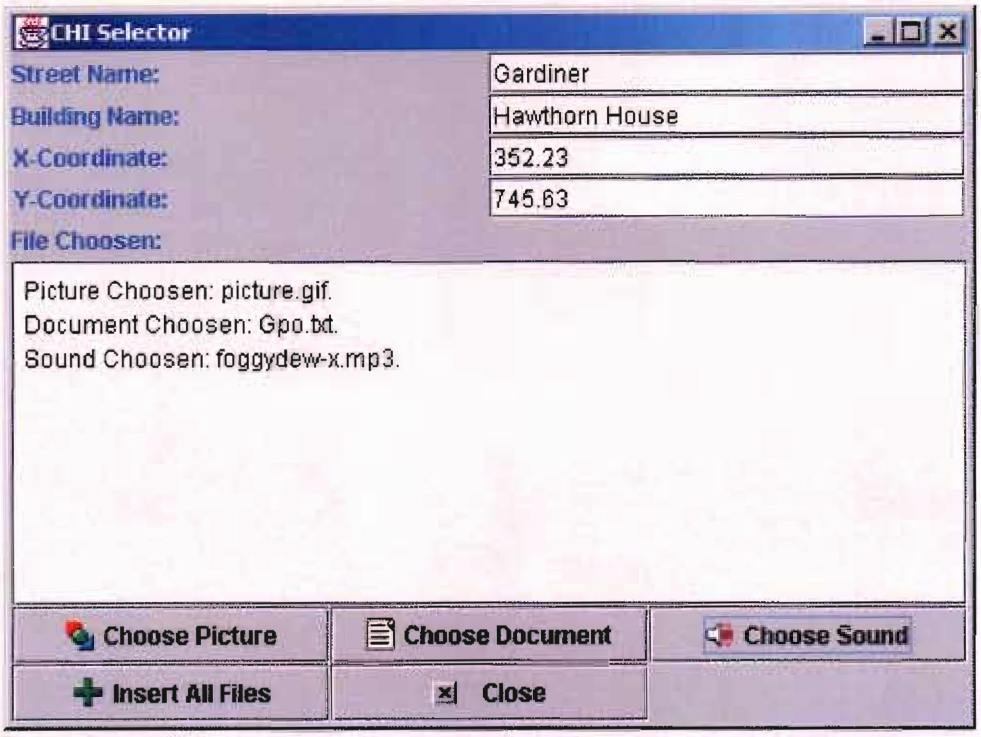
Binary LOBs are large data types that can be used to store any type of data file as long as it is less than 4GB in size. The idea behind BLOB objects is that the content of the file does not need to be known by the user in order to use them.

The following section explains the process of inserting BLOBs into the database. This can be achieved in two ways, the *PreparedStatement* objects *setObject()* and *setBinaryStream()* methods may be used, but this involves using the OCI driver, which is a client side driver. Instead the *oracle.sql.BLOB* class will be used to manipulate BLOBs because it is necessary to use a server side driver with Oracle 9i and because this approach works for both drivers.

Inserting BLOBs

In the *Initial Technology Demonstrator*, BLOBs are inserted into the database using the *CHI Selector* application. The processes involved with inserting BLOBs into the *CH database* are explained in this section. These processes are carried out in the background of the application and are not visible to the user. The complete code for the *CHI Selector* can be found in Appendix A.

The *CHI Selector* application is written in Java and enables the user to select media files from the file system. Each CH data point in the database can have three files associated with it, an image file, a text document and a sound file. The *CHI Selector* enables the user to select an image, a document and an audio or any combination of them to be loaded into the CH database. In addition to this, the user is required to specify the street name, building name and the position of the CH data point. The *CHI Selector* interface is shown in Figure 4.7.



Street Name: Gardiner

Building Name: Hawthorn House

X-Coordinate: 352.23

Y-Coordinate: 745.63

File Choosen:

Picture Choosen: picture.gif.
Document Choosen: Gpo.txt.
Sound Choosen: foggydew-x.mp3.

Figure 4.7: CHI Selector application

Behind the scenes, inserting a BLOB into the database requires that a table with a BLOB column be created. BLOBs are part of Oracle's native data types and require no special treatment. The table created in the following is the *CHI_CONTENT_INFO* table. This table contains two BLOB columns and a CLOB column and is used to store all three LOB objects in the CH database. This table is also used in the CLOB implementation.

```
CREATE TABLE CHI_CONTENT_INFO (
       ID NUMBER (20) PRIMARY KEY NOT NULL,
       PHOTO BLOB NULL,
       DESCRIPTION CLOB NULL,
       SOUND BLOB NULL);
```

Table 4.5: Create Table Procedure for Content Table

The SQL syntax in Table 4.5 illustrates how a table with a BLOB column is created. Following this the table is ready for the process of inserting an image into the Photo column. This process is different from a regular transactional insert that uses the *statement.execute(sql)* statement. This is because a locator object, not the actual data, is stored in the table's column. Therefore, the locator needs to be retrieved before inserting the actual LOB data. This involves a three-step process that is described in the following:

1. Create a locator by inserting a new row (empty blob) into the table.
2. Lock the row by retrieving the locator from the inserted row using a SELECT statement with the FOR UPDATE clause, which manually locks the row.
3. Use the locator to insert the BLOB data into the row in the database.

Creating a locator

A locator is an object that points to the BLOB data in the database and is required for manipulating the BLOB. Only the database can create a locator, because it points to a location in the database address space [Bales, 2002]. This is accomplished by formulating an SQL statement to create the locator. The statement is subsequently executed in the *CHI Selector* application to create the locator (See Appendix A).

The new BLOB locator is created by using the *empty_blob()* database function. This function inserts an *empty_blob* into the BLOB column of the table. The INSERT statement is shown in Table 4.6.

```
INSERT INTO CHI_CONTENT_INFO
       (ID, PHOTO, DESCRIPTION, SOUND)
       VALUES(1,empty_blob(),
       empty_clob(),empty_blob());
```

Table 4.6: Insert Procedure for LOB data

Execution of this SQL statement creates three new locators for the *CHI_CONTENT_INFO* table, one for the PHOTO column, one for the DESCRIPTION column (discussed in the next section) and another for the SOUND column (identical to the PHOTO column). The three locators are stored in a single row with ID 1. Initially the locator points to a location that contains no data. To insert data into the BLOB, the locator has to be retrieved back out of the table. The next section details how this is accomplished.

Retrieving a locator

As mentioned previously, the locator needs to be retrieved in order to secure a lock on the column. This gives the *CHI Selector* application exclusive access to the locator. This process needs to be conducted every time the application wants to insert data into the BLOB column.

To retrieve the BLOB locator for the PHOTO column of the *CHI_CONTENT_INFO* table, a *SELECT* statement is executed using a *Statement* object. To lock the locator the *FOR UPDATE*, or *FOR UPDATE NOWAIT* clause must be included in the SQL statement; this manually locks the locator so that data can be inserted. The SELECT statement is shown in Table 4.7.

```
SELECT PHOTO FROM CHI_CONTENT_INFO
       WHERE ID = ‘1’
       FOR UPDATE NOWAIT;
```

Table 4.7: Procedure to Retrieve BLOB Locator

The SELECT statement retrieves the locator from the database and the locator is stored in a *ResultSet* object. The locator is passed into an *oracle.sql.BLOB* object making it available within the application so that the BLOB data can be inserted. The following code fragment (Table 4.8) illustrates this.

```
rslt= (OracleResultSet)stmt.executeQuery(
                    "SELECT PHOTO FROM
                    "CHI_CONTENT_INFO
                    "WHERE ID = '1'"
                    "FOR UPDATE NOWAIT");
BLOB photo = ((OracleResultSet)rslt).getBLOB(1);
```

Table 4.8: Acquisition of BLOB Locator into Java Application

The locator is now retrieved and available within the application, finally the actual BLOB data can be inserted into the row in the database. Insertion of the data into the database using the locator is covered in the next section.

Using the locator to insert the BLOB data

Following the retrieval of the locator from the database, the locator is used to insert binary data into the database. This is accomplished using a series of steps. In the first step, the image file is placed in a *File* object. Following this the *File* object is placed in a *FileInputStream* object. After this the binary output stream is obtained from the BLOB object by using the *getBinaryOutputStream()* method. The next step obtains the optimal buffer size by calling the BLOB objects *getBufferSize()* method. This ensures that the optimal buffer size is used when writing the binary data to the database. Following this, the output stream's *write()* method is used to write the binary data to the database. The code fragment in Table 4.9 shows the series of actions. The full code for this program can be found in Appendix A.

```
File picture = new File(“picture.gif”);
FileInputStream in = new FileInputStream(picture);
OutputStream out = photo.getBinaryOutputStream();
// Get the optimal buffer size from the BLOB
byte[] buffer = new byte[photo.getBufferSize()];
int length = 0;
      while ((length = in.read(buffer)) != -1) {
      out.write(buffer, 0, length);
      out.close();
in.close();
```

Table 4.9: Writing BLOB Data into Database

After writing the data to the database using the output stream, effectively inserting the multimedia data, the program continues by closing the output stream with a call to the *close()* method. This ensures the stream is closed and that there is no loss of data. The input stream is also closed and the data is committed.

In practice, streaming data into the database using LOBs is found to be the best approach as opposed to loading objects in one at a time. The data is streamed into the database as byte chunks of data rather than all at once. This reduces the amount of resources that the program consumes at any one time when inserting the data into the database.

This concludes the section on inserting BLOBs. The following section describes the process of selecting the BLOB data from the database and into the client layer (Content Pane) of the *Initial Technology Demonstrator*. This is achieved using the *Content Servlet*. The complete code for the Servlet can be found in Appendix A.

Selecting BLOBs

As the user of the *Initial Technology Demonstrator* navigates through the VRML model of the system, Cultural Heritage (CH) artefacts are encountered and the relevant information is displayed to the user in the *Content Pane* of the application. This encounter is the catalyst for extracting the media files from the database. Extracting the media files from the BLOB is achieved by reading the binary data out of the database and into the *Content Servlet* (See Appendix A), which parses the byte stream into a device specific format for presentation to the user of the system. This is illustrated in Figure 4.8.

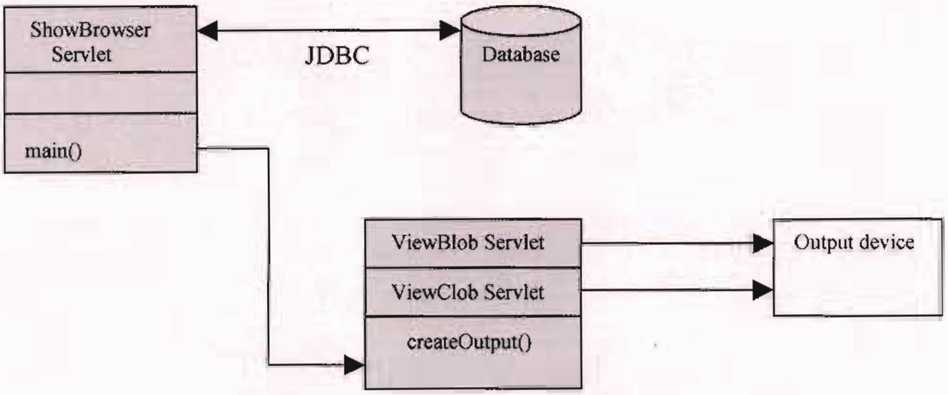


Figure 4.8: BLOB Data Extraction

This section describes the process of selecting the BLOB objects (these are either images or movie files) and streaming them to the *Content Pane* of the *CHI application.*

The process of selecting BLOBs from the database is different to selecting regular data from the database. This operation requires that the data is streamed, in the same manner it is inserted. Selecting BLOB data involves a two-step process that is described in the following:

1. Select the BLOB locator from the database using the record ID.
2. Use the getBinaryStream() or getBytes() methods in the Servlet to access the binary data. In this case the getBinaryStream() method is used.

Selecting the BLOB locator from the database, involves using a SELECT statement that accesses the BLOB column from a particular row. For example, the following SELECT statement (Table 4.10) selects the PHOTO column from the CHI_CONTENT_INFO table.

```
SELECT PHOTO
      FROM CHI_CONTENT_INFO
      WHERE ID = '1';
```

Table 4.10: Retrieving BLOB Locator for Data Extraction

Using this SQL statement, the locator is placed into a result set. The *ResultSet* object's *getBlob()* method is then used to place the locator into a local Blob variable within the Servlet. This is shown in Table 4.11.

```
//Retrieve the locator
Blob photo = resultSet.getBlob(1);
//Set the Content type
response.setContentType("image/jpeg");
//Use locator to get the binary input stream
InputStream in = photo.getBinaryStream();
//Create Stream Util object
StreamUtil su = new StreamUtil();
//Place input stream into byte buffer
final byte[] buffer = su.toByteArray(in);
//Create image out of buffer
Image image =
Toolkit.getDefaultToolkit().createImage(buffer);
```

Table 4.11: Extracting BLOB Image from Database

The code fragment in Table 4.11 begins by placing the BLOB locator into a local variable by using the *getBlob()* method. Following this, the content type is set to image/jpeg, to indicate the output format of the BLOB. The locator is then used to get the binary input stream and the BLOB data is read from the photo (Blob) object. The Blob data is subsequently read into an image object using a buffer array. This image is displayed on the client layer (Content Pane). The *Content Pane* is shown in Figure 4.9 with the BLOB Object highlighted.

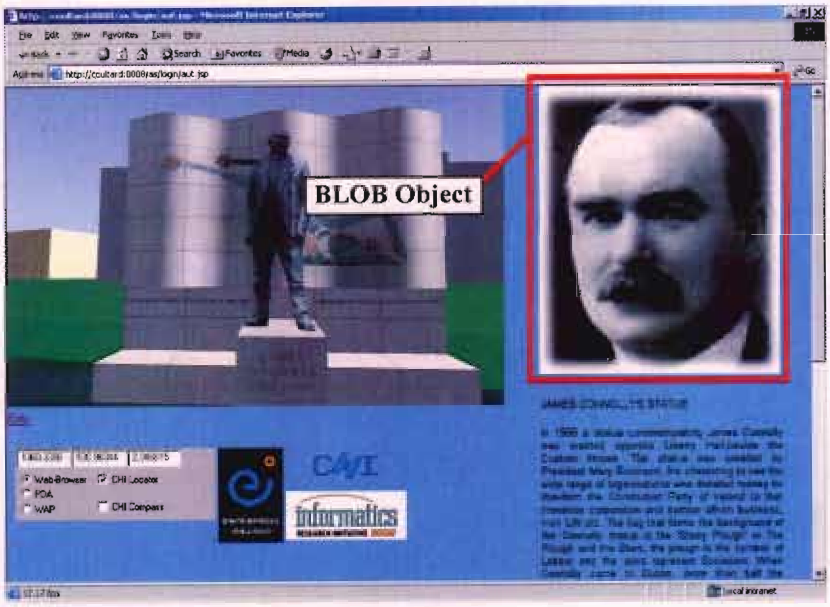


Figure 4.9: BLOB Object Displayed in Content Pane

This concludes the section on Binary LOBs. All programs mentioned in this section can be found in Appendix A. The following section is on Character Large Objects (CLOBs).

4.4.2.4.2.2 Character LOBs (CLOBs)

In this section, the use of Character LOBs in the *Initial Technology Demonstrator* is discussed. This includes a description of the procedures used for inserting and selecting CLOB data to/from the *CH database*.

Character Large Objects (CLOBs) are used to store text files in the *CH database*. CLOBs are treated almost the same as BLOBs except that they are specifically used for storing character data that is subject to National Language Support (NLS) character conversion. In comparison to the BLOB objects methods for handling binary data, CLOBs have methods for reading and writing both ASCII and Uni-code (character) data.

Inserting CLOBs

The main difference between inserting BLOBs and CLOBs into the database is that the former streams data one byte at a time and the latter streams character data, which may be more than one byte in length. Another difference is that a CLOB requires a CLOB column to be created. Creating a CLOB column in the database table is similar to creating a BLOB column and is covered in Section 4.4.2.4.2.

Inserting a text file into a CLOB in the *CH database* involves reading the file into an *oracle.sql.CLOB* object in the *CHI Selector* application. Using this approach imposes the need for the three-step method used to insert the BLOB data in the previous section and is as follows. First a locator is created by inserting an *empty_clob* into the column using the *empty_clob()* database function. The locator is then retrieved by selecting it to create a lock on the column, and then utilised by the *oracle.sql.CLOB* object to write the CLOB data to the database.

When the locator is retrieved the *getCharacterOutputStream()* method writes the data into the CLOB field in the database table as Unicode characters using streams. The following code excerpt illustrates how the CLOB data is inserted into the database by the *CHI Selector* application (See Table 4.12). The complete code for this program can be found in Appendix A.

```
CLOB doc = Resultset.getCLOB(2);
File document = new File("documant.txt");
FileReader fin = new FileReader(document);
char[] buffer = new
char[doc.getBufferSize()];
Writer out =
doc.getCharacterOutputStream();
int length = 0;
while ((length = fin.read(buffer)) != -1) {
      out.write(buffer, 0, length);
```

Table 4.12: Inserting CLOB Object into Database

In the example, the CLOB locator is stored in an *oracle.sql.CLOB* object. The file *"document.txt"* is read using a reader object. A character buffer is then created using the optimal buffer size from the locator. Subsequently, a writer is obtained from the CLOB object by calling its *getCharacterOutputStream()* method and the contents of the text file are streamed into the CLOB in the database using a *while loop*.

Selecting CLOBs

As mentioned in the previous section, the LOB data in the database is extracted when the user of the system encounters a CH data point in the VRML model. This encounter is the trigger for extracting the media files from the database. Extracting the media files from the CLOB column is accomplished by reading the Unicode data out of the database and into the *Content Servlet*, which parses the data into a character format for presentation to the user of the system. This process is illustrated in Figure 4.8.

The sequence of events involved in extracting the CLOB data is as follows. Only the methods in the *java.sql.Clob* interface allow us to retrieve CLOB data from the database. The *oracle.sql.CLOB* implements this interface, thus allowing it to access the required methods. The *getCharacterStream()* method is then used to read the Unicode data from the CLOB using a byte stream. This involves a two-step process to extract the data from a CLOB and is as follows:

1. The CLOB locator is selected from the database.

2. The *java.sql.Clob* objects *getCharacterStream()* or *getAsciiStream()* methods are used to access the data; in this case the *getCharacterStream()* method is invoked.

Initially, the CLOB locator is selected from the CLOB column to lock the column. This is because the column only holds the locator and not the actual data. This is illustrated in Table 4.13.

```
SELECT DESCRIPTION
FROM CHI_CONTENT_INFO
WHERE ID = ‘1’;
```

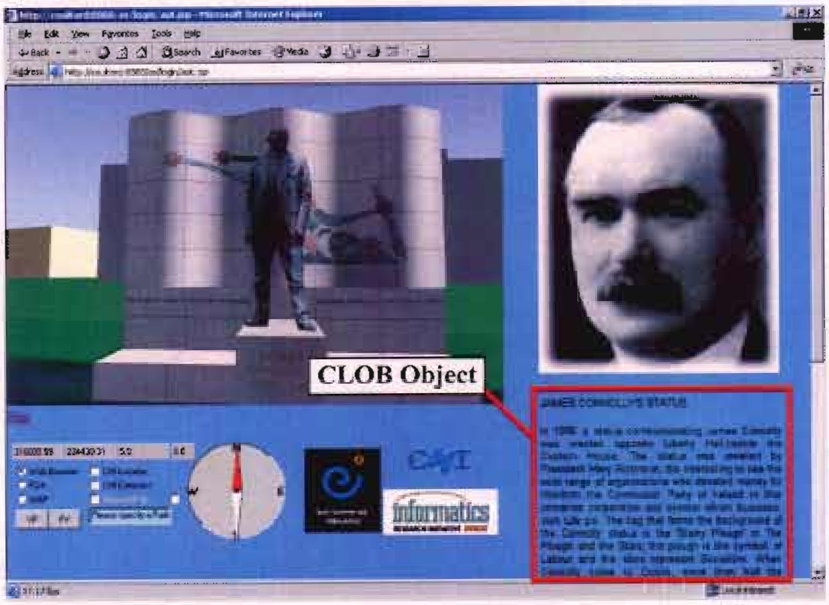
Table 4.13: Retrieving CLOB Locator from Table

The locator is placed into a local *Clob* object by using the result sets *getClob()* method. The data is then read out of the CLOB column using the locator. The following code excerpt shows the process of reading the data from the database and into the *Content Servlet*.

```
//Retrieve the locator
Clob description = resultSet.getClob(1);
//Set the Content Type
response.setContentType("text/plain");
//Obtain a Reader Object to read the CLOB
Reader in = description.getCharacterStream();
//Get the length of the CLOB
int length = (int)description.length();
//Create a new Character Buffer
char[] buffer = new char[1024];
//Read the data out into the Servlet
while ((length = in.read(buffer)) != -1) {
            out.write(buffer, 0, length);
                  }
//Close the Reader Object
in.close();
//Flush the output
out.flush();
```

Table 4.14: Extracting CLOB Document from Table

The code in Table 4.14 illustrates the process of selecting data from a CLOB column. The locator is stored in a *java.sql.Clob* object within the Servlet. When the locator is successfully stored in the Clob object the *getCharacterStream()* method retrieves the Unicode data stream. This data is read into the Servlet using the Servlets *PrintWriter* object and the text document is displayed on the client layer (Content Pane) of the *Initial Technology Demonstrator*. This is illustrated in Figure 4.10.



CLOB Object

JAMES CONNOLLY'S STATUE

[illegible]

Figure 4.10: CLOB Object Displayed in Content Pane

This concludes the section on streaming data types. All standard multimedia files that are used in the *Initial Technology Demonstrator* are processed using the methods covered in this section. All binary files (gif, jpg, mpg, avi) are processed using BLOBs and all character files (txt, doc, rtf) are processed using CLOBs.

The causality for the CH data to be read from the database is the spatial query. As the user progresses through the VRML model of Dublin, the location is continuously sent to the application server to check for any interaction between the present location and the location of the CH data. If there is an interaction, the program reads the CH data from the *CH database* using the techniques covered in section 4.4.2.4.

The following section is a description of the techniques used to query the *CH database*. This includes an initial description of spatial querying and the methods employed in the *CHI system*. Following this is a detailed explanation of user-based queries and the approach taken to deliver CH data to the user based on the concept of an egocentric point-of-view.

4.4.2.5 Querying Spatial Data

In this section, spatial queries are presented and, in particular, window queries. Spatial queries are database queries that query spatial data based on geographic location as opposed to tabular position (See Section 2.7.2). The queries are formulated and processed using spatial operators. These operators are native to specific database systems and are covered in section 2.7.2.1. In the case of the *Initial Technology Demonstrator,* the operators available in *Oracle spatial* are used.

Oracle spatial uses a two-tier model that uses a primary and secondary filter technique in resolving spatial queries. Brief explanations of each are given below and for a detailed description, see section 2.7.2.1.2.

Primary Filter: This filter uses geometry approximations (MBRs) for fast selection of candidate records that are passed to the secondary filter. This filter reduces the candidate set, minimising the number of objects that need to be inspected by complex computationally expensive geometric algorithms.

Secondary Filter: This filter applies exact computational geometry to the candidate set supplied by the primary filter and identifies geometries that satisfy the spatial query. These operations are computationally more expensive but are applied to a smaller data set.

4.4.2.5.1 Window Queries

A window query is the recognized standard operation to query the *CH database* for any CH artefacts that may be present in the location of the query window. The window is of a specified width and height centred on the user's location and is represented in *Oracle Spatial* as an optimised rectangle and is defined by a minimum of two points *p1* and *p2*. An advantage of using a rectangle is that it is an optimised shape used for query processing (See Figure 4.11). The code fragment in Table 4.15 shows how the *sdo_relate* operator is used to query the database using an optimised rectangle.

```
SELECT      A.ID, STREET, BUILDING
FROM  CHI.CHI_CONTENT_BUFFER A
WHERE       SDO_RELATE (A.POSITION,
            MDSYS.SDO_GEOMETRY (2003,NULL, NULL,
            MDSYS.SDO_ELEM_INFO_ARRAY (1,1003,3),
            MDSYS.SDO_ORDINATE_ARRAY(X1,Y1,X2,Y2)),'mask=a
            nyinteract querytype=window') = 'TRUE' ORDER
            BY ID";
```

Table 4.15: Window Query using Optimised Rectangle

This example shows how the *sdo_relate* operator is used to compare the query window (optimised rectangle) with the CH dataset to determine if they satisfy any topological relations (e.g. Intersect, contains, covers, etc.). The *sdo_relate* operator accepts three parameters, the *sdo_geometry* column of the layer being queried, which must be spatially indexed, the query window *sdo_geometry* and a *param* list that determines the behaviour of the operator. The query window geometry is defined in the SQL string and contains two arrays. The *sdo_elem_info_array* contains the values that define the type of geometry that is to be queried against the layer. In this instance, it is an optimised rectangle that is defined by a triplet value (1 (offset), 1003 (outer polygon), 3 (optimised rectangle)). The *sdo_ordinate_array* contains the coordinate values of the rectangle. The mask is set to *"anyinteract"* which means if any of the topological Boolean predicates return true (or interact) then the geometry is added to the resultset. A detailed description of all Oracle operators can be found in [Geringer, 2001a]

The SQL statement in Table 4.15 queries the spatial database for interaction between the query geometry and the CH data layer. The query is processed every five seconds, if the user is on the move. This time was selected to reduce the amount of queries to the database because of the operational load that more frequent queries put on the Server machine. If the position of the user is the same or less than 5m away from the position that the last query was processed, the query is not run again. When the user's position is outside this threshold, the query is run again. This significantly reduces the computational cost of redundant queries to the database.

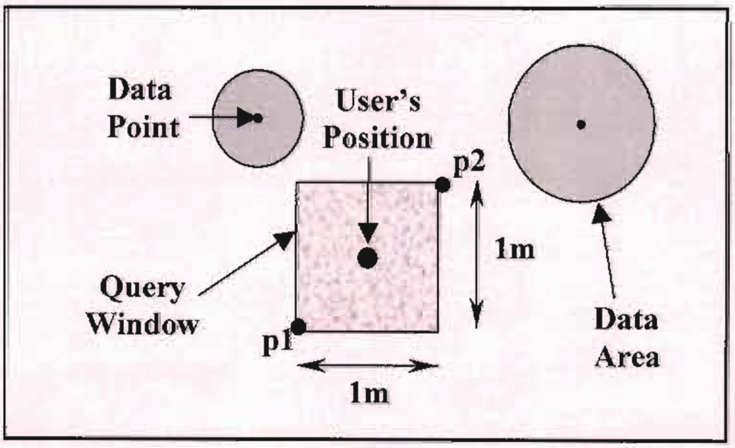


Figure 4.11: Optimised Rectangle (Window) Query

In the *Initial Technology Demonstrator*, each CH *data point* in the database represents a CH artefact and is surrounded by a buffer (the extents of which are also explicitly stored by *Oracle Spatial*) of varied radius depending on the size and/or significance of the artefact. Taken together, the data point and surrounding buffer represent a *data area*. Justification for placing a buffer around the individual data points rather than around the viewer's dynamic location in space was one of maximising query optimisation, because one of the most important aspects of the query process is the speed of the query execution. Therefore, an optimised rectangle was the most favourable geometry to query against the database (See section 2.7.2.1.2).

When the user's query window intersects a data area in any way, e.g. touch, overlap, etc. (See section 2.4.4.3), the relevant data is placed into a resultset and displayed to the user in the form of text, images, audio, and video files (See Figures 4.9 & 4.10). In the initial implementation, it was sufficient to collect data in this manner.

This concludes the implementation of the *Initial Technology Demonstrator*. A prototype was built that demonstrates the synergy created by combining these technologies in an innovative way to produce a system to deliver content to the user based on their context, their context being their spatial location.

During the development of the *Initial Technology Demonstrator* it was realised that there were a number of features that needed to be further developed in the system. Certain aspects of the prototype that were not satisfactory needed addressing. The following section is an evaluation of the initial prototype, in an effort to determine what aspects need to be addressed to improve the system, and how these issues should be addressed.

4.5 Initial Technology Demonstrator Evaluation

In this section, the details of an evaluation of the *Initial Technology Demonstrator* are presented. This includes an assessment that was carried out in order to identify the elements of the prototype that need addressing. Following this, there is a description of the problems that were identified by this process. Finally in this section, the possible solutions to these problems are discussed.

4.5.1 ITD Evaluation Methodology

In this section the evaluation methodology used to assess the Initial Technology Demonstrator (ITD) implementation of the CHI project is described. Experts carried out the evaluation of the ITD exclusively. In this case experts are are understood to have sufficient knowledge of currently available web-based systems which utilise technologies similar to those employed in the ITD.

This evaluation is based on the functionality of the system, which can be directly related to its design. Therefore, the purpose of this evaluation is to determine if the prototype functions as expected, based on the design.

The ITD evaluation used a method whereby a small number of individuals, with knowledge of location-based media development, compared the prototype against a list of usability factors. This helped to determine what features of the ITD were sufficiently suitable and which were not.

For this evaluation, four experts tested the ITD and judged it against the list of usability principles and a list of technical principals, outlined by the project team. This proved to be the most effective way to determine whether the ITD was suitable for the users of the system and if not, what exactly needed to be changed. In this case, the expert group was composed of two computer science researchers and two digital media researchers.

Usability factors

This is the list of factors, determined by the research team, that the evaluator used to assess the prototypes usability potential. These factors can be used to ascertain how easy a system is to use. For the ITD these are:

1. **Navigation** – this refers to the ease of use of the system and how easy it is to navigate through the system. Navigation is a very important issue in this case because navigating in a 3D space is unnatural for humans. It is much easier to navigate on a surface than in a volume because 3D interfaces are projections rather that true 3D causing the movements to be indirect and awkward [Nielsen, 2000, p.222]. The evaluators of the system were asked to focus on two aspects of navigation. The primary concern relating to navigation within 3D environments is the absence of familiarity with the navigation tools available in VRML browsers. The evaluators were asked to comment on this aspect of navigation and also the navigational help supplied within the model (e.g. Exits and help).

2. **Data Relevance –** this refers to the relevance of the data displayed to the user of the system. One of the primary purposes of the research is to deliver information that is relevant to the user's position in the model. Therefore, the evaluators were asked to monitor carefully the results of the spatial queries that were being displayed, in particular, the relevance of the data in relation to their position and also the relevance of the data in relation to their direction. For example, when looking at the *Liberty Hall*, is the data being displayed to the user relevant to that building or area?

3. **Help and Documentation** – this refers to the availability of help and documentation in the ITD. Even though it is better if the system can be used without documentation, it is necessary. The evaluators were asked

to communicate any problems encountered when in need of help. For example, how easy was it to find the help facility if needed? And if so, how easy was it use and understand?

Technical factors

This is the list of technical factors, also determined by the research team, that the evaluators were required to use to determine the prototype's technical potential. These factors can be used to pertain how effective and efficient a system may be. For the ITD these are:

1. **Architectural Design** – this refers to the overall design of the ITD. The evaluators were asked to access the architecture of the system, from general design, to individual technologies used. They were encouraged to voice their own opinions on the choices made by the research team. In this way only could the evaluation be fully successful.

2. **Data Extraction Speed** – this refers to the time it takes to determine if the user of the system is interacting with CH data, and if so how long it takes to extract the relevant data and display it. The method used here was to employ a qualitative approach to measure the latency of the spatial interaction. This was achieved by recording the exact time the user interacted with a data point and also the exact time the content is displayed to the user. By subtracting the interaction time from the display time, it is possible to determine what the time interval is. These times were recorded for each evaluator of the system navigating independently.

3. **Multi-Tasking** – this refers to the ability of the ITD to carry out multitasking operations. This includes the system's ability to operate when many users are accessing the system. In this case, the units of

measurement were the CPU and Memory usage percentages. The tests carried out involved the four expert evaluators logging into the system at the same time and navigating independently of each other. Each user was using a Pentium 4 (1.7 Mhz) computer with 512MB of memory and an nVidia graphics card with 16MB memory. The server system configuration is the one described in Section 5.2.1.3.2.

4. **Adaptability** – this refers to the ability of the ITD to be adaptable. This includes the system's ability to deliver adapted content data to different simulated devices and how the system could accept heterogeneous sets of data from alternative sources. For example, how would the system deal with geometric data from another type of database and how it would deliver data to an alternative VR component?

4.5.2 ITD Evaluation Results

In this section, the evaluation results of the ITD are presented. This includes a description of the usability issues that were determined by the usability study carried out by the research team. Following this the technical issues that surface during the technical evaluation procedure are detailed.

The evaluation of the ITD proved to be very successful in all areas. In the evaluation, the research team used experts in the technological field, to assess the ITD.

Usability Results

In this section, the usability evaluation results are outlined. These results are based on the list of usability factors outlined in section 4.5.1. The following is a list of issues that were identified:

1. In relation to the navigational aspects of the prototype, a number of issues arose from discussions with the evaluators. Since the emphasis of the ITD is not on the photo-realistic aspects of the system, the navigation within the parts of the VRML model that did not have facades proved to be difficult. It was not clear to the users what direction they were travelling in when trying to navigate back to the 'beaten track'. The obvious solution to this problem was to employ some kind of *Compass* device into the system. In principle, this would help the users of the system to navigate. In addition to this, the use of *Director Guides* was considered. These are large arrows placed strategically within the model to direct the user to the next point of interest.

2. There were a number of comments made in relation to the relevance of data being exhibited to the user. In many cases the data that were presented to the user were not relevant to the current position. It was reported that the data relevant to a particular point in the model would be displayed when the user has passed on to another point.

   This prompted a serious investigation into the reason for this happening. The objective of this investigation was to determine what aspect of the system was causing the problem. There were three parts of the system that were suspected as potential problem areas. The speed of the application server to process the Servlet was one, the ability of the database to read out the data (quickly) was another, and the type of spatial index used was also considered. The approach taken was to test these components of the system separately, to determine which one component or combination of components was causing the problem. It was discovered early on in the investigation that the problem was with the type of spatial index. This was achieved by reading some data directly from the database into a Servlet. These data were displayed without any delay. In principle, the solution to this problem was to use an alternative spatial index.

3. In addition to this recurring problem, there were a number of comments on the relevance of the data being displayed to the user based on the current direction of the user. It was pointed out that when the user was facing one particular monument, data about something else was being delivered to the user. For example, when the user was facing the General Post Office (GPO), data about the Daniel O'Connell statue was being delivered to the user. This information led to a study that looked at the relevance of the data based on the direction of the user in the model. It was realised that in several instances the data displayed to the user related to an artefact that was behind or out of the user's sight.

   These results prompted research into orientation and direction in the context of spatial queries. The problem with the system at this stage was that the window used to the query against the database did not take into consideration the user's point-of-view but instead just their location. This meant that changes in the users direction had no effect on the resulting data. The solution to this problem was to design a component that would respond to changes in the users orientation and modify the query window accordingly.

4. In regard to the help and documentation section of the usability evaluation, there was a comment on the absence of help for the navigational aspects of the VRML browser. For example, what keys enable the user to look upwards within the model? To familiarise the user with the navigation controls, it was decided to add some kind of help facility to the prototype to familiarise them with the environment.

It has been found that users more familiar with VR applications perform better when using similar applications [Dalgarno and Scott, 1999].

Technical Results

The following is a description of the feedback that was obtained from the evaluators relating to the technical issues of the ITD. This includes some repetition of related problems mentioned in the usability evaluation as problems were directly related. This part of the evaluation uncovered some major problems with the ITD. These are described in the following:

1. The architecture of the system was a topic of major discussion. It was pointed out that having two separate databases, one for the VRML data and another for the CH data was an efficient approach but not effective. It was efficient because reading the geometries out of the MySQL database was very fast and consequently the model loaded very quickly. It was not effective because storing the geometries in a relational database meant that the spatial properties of the data were not captured, therefore it was not possible to perform spatial operations on it. It was also considered ineffective to manage two databases when one would suffice. It was recommended that it would be an advantage to use one database for all data because it would be a centralised repository for the data. Reducing the amount of work needed to maintain a system is a crucial concern in system development and using two databases for the ITD increases the amount of work required. It was also brought to the attention of the researcher, that considering the type of data that is present in the system, it is also very important that the integrity of the data be preserved. In this case, the security of the data and versioning control needs to be tight, where a master user owns the data and any

changes to the data must be sanctioned. Some potential issues with the approach of using one central database were identified; for example, the load is focussed on one database and this potentially reduces the number of users the system can handle. Also in the event of a database crash, all data are rendered unavailable.

2. It was mentioned in the usability evaluation that the relevance of the data was inconsistent with the area currently being displayed. Tests were performed to determine the cause of this problem (explained in the Usability Results) and revealed that the root of the problem was the process of determining the interaction between the user and the data points in the system.

   Previously, it was thought that the problem was with the extraction of the data from the database, but this was not the case, and was proven by a test that extracted the data directly from the database using a Servlet. The coordinates of a known CH data point were passed into the Servlet ensuring data would be displayed. In this case, the data was displayed almost instantly. By bypassing the spatial query in the system using a Servlet to extract the data enabled us to determine that the problem was not with the database or the Servlet container. As a result of this test, it was believed that the problem was with the spatial index used. In this case, the intended solution was to remove the current spatial index and use one that was more suited to the type of data, to rectify the problem.

3. Another test undertaken by the evaluation team was carried involved testing how the ITD performed when multiple users subjected it to multiple accesses to the databases (details of which are explained in technical factors section). This test revealed a major problem within the middle layer of the system. The system performed well during logon

and initial loading of data. It also performed well when only one person was using the *CHI locator* to locate data. The problem arose when more than one user was using the *CHI locator*. As one user of the system encountered a piece of data, it would be displayed to them, but also to the other users of the system. This indicated that there may have been a problem with variables in the ITD being replaced with other variables and that session tracking would need to be implemented to determine the root of the problem. The approach in this case was to monitor the CH data IDs for each user. After a long bug tracking test, which involved tracking over ten variables for each user as they used the system, it was finally discovered that the *'CH Data ID'* for the users was defined as a global variable in the system. This meant that each time a user encountered a CH data point, the variable was set globally thus causing the same data to be displayed to all users of the system.

4. In addition to the comments about the design of the database layer of the ITD, it was suggested that, to reduce the amount of work required to adapt the data to each separate device and to adapt the VR data to a different format, a content adaptability approach be taken. Instead of creating a separate file for each device format, it would be more productive to use a standard data format that adapted the output based on the request from the particular device.

   The obvious approach to take here was to introduce XML into the system. Using this technology, it is possible to extract from the database into a standard XML file format. The XML file can then be used for all devices after a device specific XSLT stylesheet is applied to the data, to tailor it for each device format.

This concludes the evaluation of the ITD. After a final discussion with the experts the design of the *Revised Technology Demonstrator (RTD)* was developed. The following section describes the revised design.

4.6 Design of Revised Technology Demonstrator

The revised design of the CHI technology demonstrator is a more streamlined design in many ways. It is the direct result of a comprehensive evaluation carried out by a group of experts in the area of digital media. The overall design is illustrated in Figure 4.12.

Conceptually, the *Revised Technology Demonstrator* allows a user to navigate through a virtual three-dimensional model of Dublin streets and view data relevant to their location in this space. The virtual streets contain the facades of buildings constructed and rendered using XML and XSLT into VRML (Virtual Reality Modeling Language) (Section 4.7.3). As the user moves within this 3D world, the location or virtual space coordinates are concurrently passed as parameters to an Enterprise Java Bean (EJB) that constructs a query to the spatial database (Section 4.7.2). The query then retrieves all data from the database contained in the users field-of-view (FOV) within the virtual 3D space, i.e. retrieve all data that the user can see (Section 4.7.4). The result of the query is returned in XML and rendered into the user's device specific format using XSLT.

In this section, the design of the *Revised Technology Demonstrator* is presented. This includes a description of the information workflow through the RTD. Following this, a description of the modifications made to the ITD due to the evaluation of the system is given.

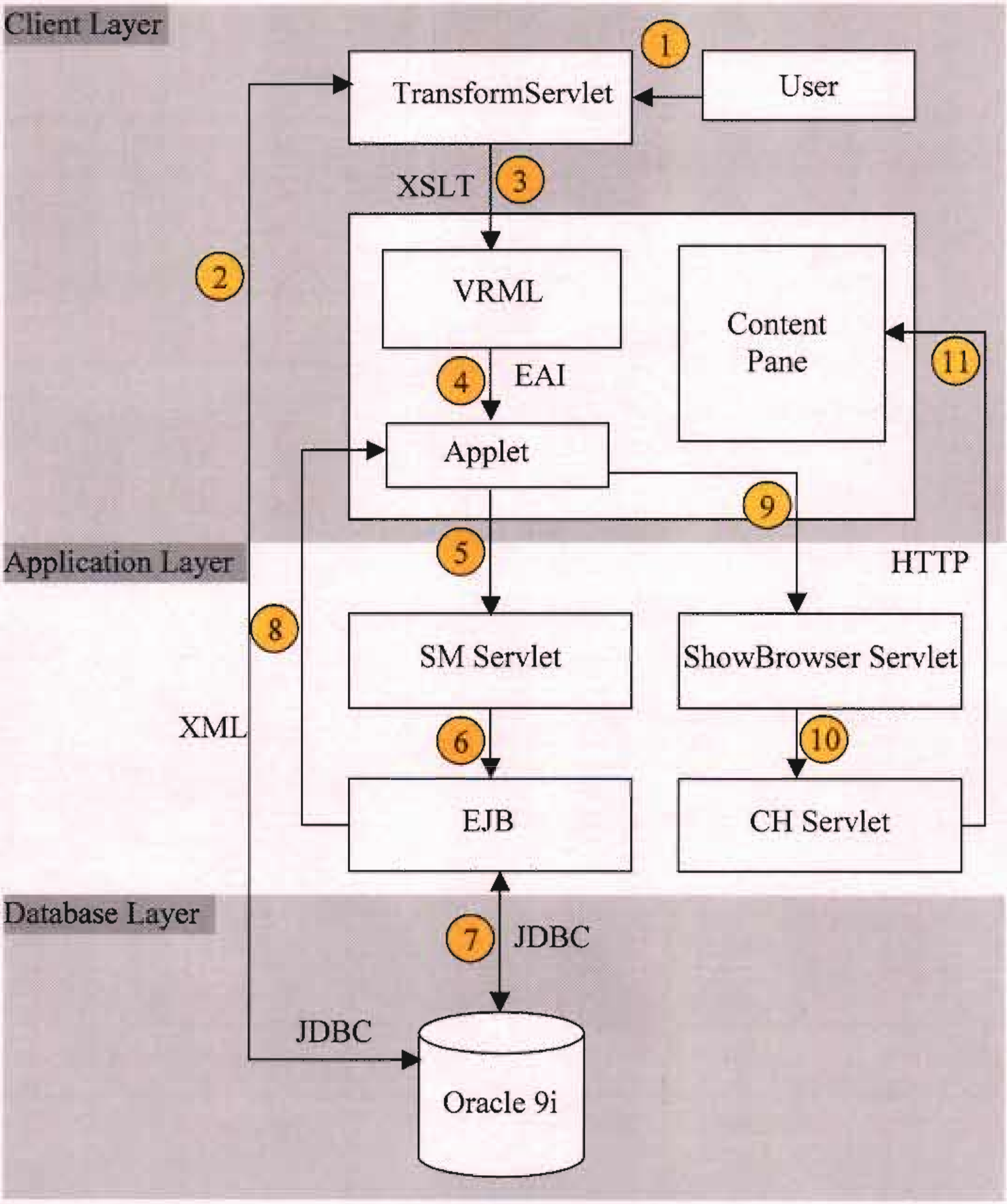


Figure 4.12: Design of Revised Technology Demonstrator

In the following, the data flow in the RTD is described. The numbers in Figure 4.12 correspond to the numbers in the description:

1. The user logs into the system using a username/password and a users profile is extracted from the database using *JDBC* and stored in memory using a bean.

2. The *VRML* geometry data is extracted from the database using *JDBC* and rendered into an *XML* file.

3. An *XSLT* style sheet is applied to the *XML* data to create a *VRML* output. This is displayed in a *Cortona VRML* browser.

4. The External Authoring Interface *(EAI)* is used to obtain the user's position and orientation within the *VRML* model and passes it into the *Interactive Java Option Pane* (Applet).

5. The applet invokes a *SM Servlet* and passes it the position and orientation parameters. The *Directional View-Port* is constructed using the parameters.

6. This *SM Servlet* then invokes a *CHI EJB* for this user and passes on the viewport parameters to it. At this point the spatial queries to the database are formulated.

7. The *CHI EJB* connects to the database using *JDBC* and sends the queries to the *Oracle Spatial* database engine. A result set of all geometries that interact the *View-Port* is returned.

8. The ID's of the *Cultural Heritage* artefacts that are associated with the interacting spatial objects are passed back to the applet.

9. The applet invokes the *ShowBrowserServlet* and passes the artefact IDs to it. The artefact's text is then rendered into XML.

10. The *ShowBrowseServlet* invokes the *CH Servlet* and passes the artefact IDs to it. An *XSLT* style sheet is applied to the *XML* and the data is rendered into the device specific format.

11. This data is then displayed to the user in the *Content Pane*.

Enhancements to the ITD were made on all layers of the system. They are described in relation to the layer they belong to, so any changes that are visible in the diagram are described in the following.

4.6.1 Client Layer

There were a number of enhancements made to the client layer of the system. They are mainly concerned with the navigational aspects of the VRML model. From the discussions with the evaluators of the system, it was decided to include two compasses in the system.

4.6.1.1 Compass

The VRML model of the RTD contains a virtual compass which allows the user to interact with the location information in a more natural way while navigating in the 3D model of Dublin. The view orientation is visualised as letters facing the camera in relation to the geographical orientation of the camera view. For example, if the user faces *North-West*, the direction and degree of rotation from *North* is displayed within the model (Figure 4.13).

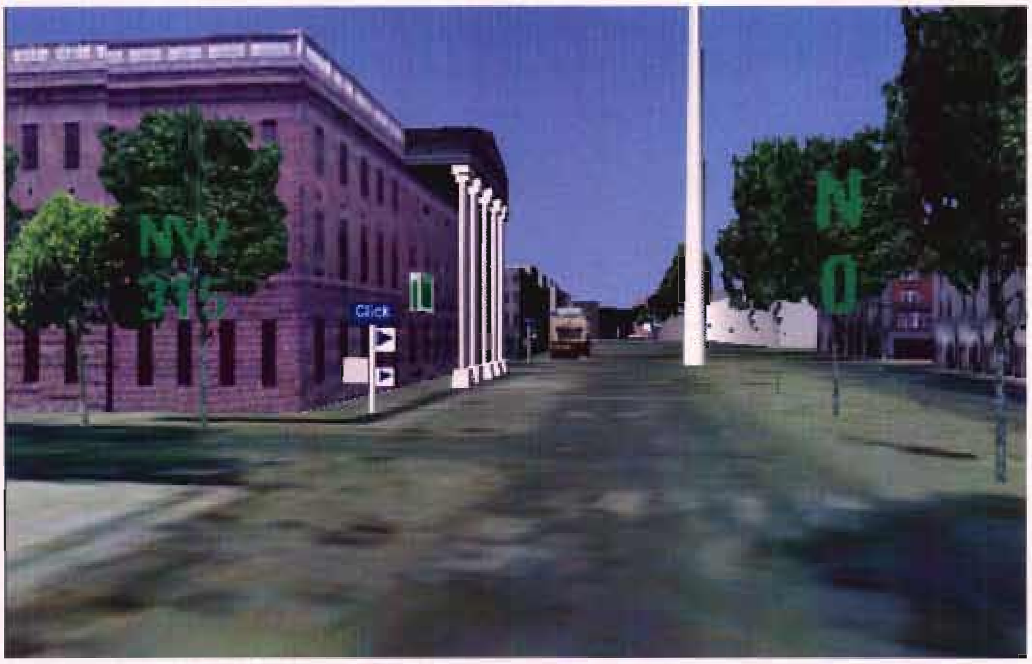


Figure 4.13: Virtual Compass in RTD

The intereractive (internal) compass can sometimes obstruct the user's view within the model, so it was decided that an external compass would be added to the *Interactive Java Option Pane* for use when the internal compass is causing an obstruction. This compass is similar to a compass that would be used in conjuntion with a map.

4.6.1.2 Help Function

The absence of a help file was another aspect of the client layer that surfaced from the evaluation. As it is not clear to the novice user of the system how to navigate within the VRML model, a *Users Guide* was added to the RTD design. The guide informs the user of what the RTD does, how to navigate in the system and how to set the options. This valuable addition to the system helps familiarise the user with the *Cortona VRML browser* navigation tools.

4.6.1.3 Information Towers

When the user is navigating through the 3D model, it is unclear where there is and isn't data. It was shown from the evaluation that sometimes the user wants to know exactly where there is data present so that they can view it. It was suggested that billboards be displayed with a topic for selection. The research team did not agree with this, as it was intrusive and obstructive. Instead, it was decided to use information towers to indicate where there was data present in the model. Large semi-transparent towers identify the areas occupied by data. These towers do not obstruct the user's view in any way and can be removed by deselecting them on the *Interactive Java Option Pane*.

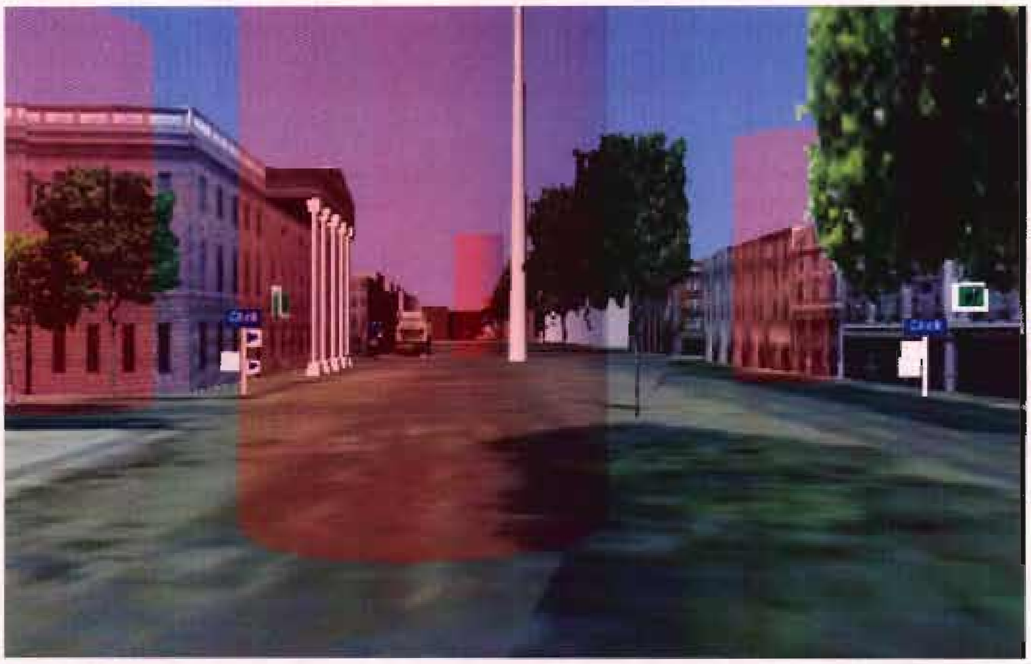


Figure 4.14: Information Towers in RTD

This completes the section on design changes made on the client layer of the ITD. The following section describes the changes that were made to the application server layer of the ITD.

4.6.2 Application Server Layer

The application server layer of the ITD saw major changes to its design following the expert evaluation. The main design changes to this layer include the introduction of EJB technology to replace the middleware Servlet of the ITD, and the use of XML and XSLT to adapt the CH data and VRML data into device specific formats (Section 4.7.3). The type of application server used to deploy the application was also changed. In an attempt to streamline the ITD design, the RTD uses an Oracle 9iAS application server as opposed to the *Apache HTTP Server* and *Tomcat Servlet Container* combination. This is mainly because the application server offers a more integrated software bundle and, as it is also an Oracle product, it integrates well with the database.

4.6.2.1 Enterprise Java Beans

The multi-user problem discussed in the technical evaluation results section presented major difficulties for the ITDs design. The client of the ITD obtains all the information it requires from the system by interacting with the *Interactive Java Option Pane*, which hides the complexity of all the other connections in the system. The ITD implementation of the applet requests updates from the database every five seconds by passing requests to the Servlet, which in turn passes on the commands until it reaches the database where the query is processed. Two results are obtained on each request, which indicate if the information in the *Content Pane* needs to be updated.

Once these results are obtained they are then passed back to the Servlet, which passes the information to its Java Virtual Machine (JVM). This is where the problem was manifesting itself. While each applet that is accessed by individual clients is a standalone, and therefore independent, object, the Servlet is a static object, i.e. only a single instance of it is created and it is then accessed by all the applet instances. The Servlet then deals with each applet request individually using a multithreaded approach, by opening a connection and maintaining that connection until the request has been completed.

However, in the ITD Applet/Servlet connection, the results are printed to the console and then read in by the applet(s). Thus, whenever a single user makes a change, or no change, all the applets read the same information and therefore will change their output accordingly, hence the applets are no longer standalone as they are directly influencing each other's output criteria.

One possible solution to this problem was to adapt the Servlet technologies within the system to use 'object-passing' to send and retrieve information from each individual user. This would ensure that each user in the system has a unique connection to the Servlet and would not be passed any irrelevant or erroneous information from other clients. However, this approach would create a large amount of unnecessary traffic within the system caused by 'object passing'. It was decided by the research team to use Enterprise Java Beans (EJBs) for this part of the system.

EJB technology is a powerful tool in the development of client-server technologies. EJB defines a server-side component model that allows business objects to be developed and moved from one container to another. The EJB server is responsible for making the component a distributed object and for managing services such as transactions, persistence, concurrency and security. The main advantage of the integration of this technology, is the ability of the programmer to concentrate on the business logic of the application being developed, while the EJB container abstracts away the implementation detail [Monson-Haefel, 2001].

The main Servlet component of the ITD was replaced with an EJB component to improve the performance and extensibility of the application. The addition of the EJB was incorporated to facilitate the multi-user element of the RTD.

The incorporation of the EJB into the project is also a major advantage in terms of providing a more reliable and robust application. The pooling function adopted by EJBs is a major advantage in making the RTD a powerful multithreaded application. This is because EJB containers have the ability to manage all sessions automatically.

The RTD now operates on a multi-tiered level, where the user interacts with the client, that interacts with a Servlet, which doubles up as the client to the EJB; the EJB interacts with the database to retrieve all the relevant information. The EJB approach is illustrated in Figure 4.15.

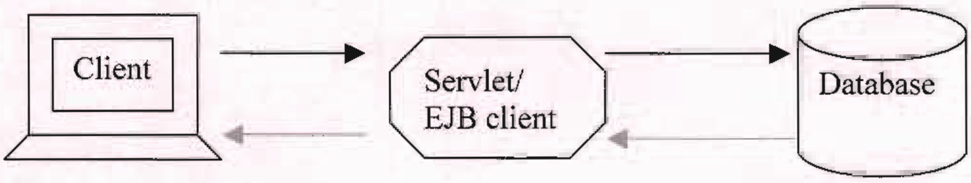


Figure 4.15: EJB implementation of CHI application

This approach required that the choice of application server be reconsidered. The ITD approach using Servlets used the *Apache Tomcat Servlet Container* and was adequate for these purposes. With the addition of the EJB technology and the need to streamline the system, it was necessary to change to *Oracle 9i Application Server (9iAS)*. This was because *Oracle 9iAS* is fully configured for EJBs and enables all web-based aspects of the project to be hosted on a server that integrates all component containers in one.

4.6.2.2 Oracle 9i Application Server (9iAS)

Oracle 9iAS is an application server developed by Oracle [Oracle, 2003] based on open standards such as Java 2 Enterprise Edition (J2EE). The Enterprise Integration framework with pre-built adapters and Web-Services allow applications to be easily integrated with legacy systems and databases. It was selected because it is the most integrated web-server available and is especially configured for the *Oracle 9i Spatial Database*.

4.6.2.3 Content Adaptation

The evaluation process outcomes indicated that a more efficient and adaptable process was needed with regard to content delivery in the ITD. The data being displayed in the browser of the ITD was retrieved from the database and converted into a HTML file for display in the *Content Pane*. However, this approach needed to be revised to enable the data to be viewable on a number of devices, hence instead of converting the data to HTML when retrieved from the database, it would be more useful if the data was returned in XML format. This XML file could then be transformed using an XSLT stylesheet to render the data into a format that complied with each specific output device capabilities. These changes were adopted and the necessary modifications were implemented

This completes the section on changes made to the design of the application server layer of the ITD. The following section is a description of the design changes that were necessary on the database layer.

4.6.3 Database Layer

Following the evaluation of the ITD, it was necessary to make some changes to the database layer of the system. The most important of these was the migration of the VRML vector data from the MySQL database into the Oracle database. This decision was made in an attempt to centralise all the data in the system. This decision was also influenced by the need for better security and versioning control of the map data. For example, if all users have open access to the data, it is very difficult to monitor changes to the data and maintain data integrity and consistency. Since Oracle has the ability to modify data and record the changes to the data in a separate table without changing the master set, it makes it a suitable database for the RTD.

4.6.3.1 R-Tree Indexing

During the evaluation of the ITD it was discovered that there was a delay between the interactions of the data and the data actually being displayed to the user. Following an investigation of the whole process, it was determined that the problem was with the type of spatial index being used. It was mentioned during the analysis and synthesis of the literature review (Section 3.1.3) that the best alternative to the current Quadtree index is the R-tree index. The R-tree is easier to create, faster at nearest neighbour queries, and can index geodetic data as well. Also, with the migration of the VRML data into the Oracle database, it is necessary to index this data as well. According to the literature, the R-tree is better for indexing polygon data and a recent performance test [Ravi et al, 2002] shows that in many cases the R-tree outperforms the Quadtree when performing large window queries. R-tree indexes use Minimum Bounding Rectangles (MBR) to approximate the geometries. This is shown in Figure 4.16.

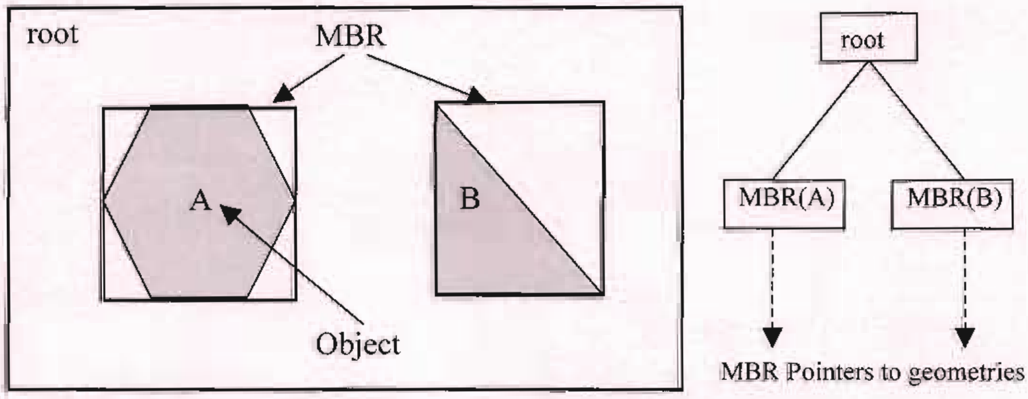


Figure 4.16: R-Tree Spatial Index

R-tree indexes are based on an approximation technique that involves the creation of object MBRs in a 2-Dimensional (2D) space. When querying the data it is the MBR set that is queried and not the actual geometries. This makes querying more efficient by eliminating the majority of geometries from the candidate set and reducing the number complex geometric calculations needed to answer the query.

In the RTD, the geometries that represent the locations of the CH data and the VRML data also use the R-tree index. In the case of geometric data, this is important because when converting spatial data from a *Cartesian* to a *Geodetic* coordinate system, the R-tree index is capable of indexing the geodetic as well. An overview of R-trees can be found in section 2.6.2.1.

One disadvantage of the R-tree index is that it may become inefficient after a large amount of updates and it is recommended that the REBUILD command be used occasionally to rebuild the structure.

4.6.3.2 Directional Querying

This section describes changes to the method of querying the database of the ITD. It was in response to a problem that surfaced on the client layer of the ITD but the actual source of the problem was on the database layer. The problem was noticed in the evaluation, when the data being displayed to the user was not entirely relevant to the direction the user was facing. The discovery of this problem led to the design of a totally new component to the ITD.

An initial enhancement of the query processor extended the ITD by adding functionality that makes it possible to adjust the query window's shape. Unlike the initial approach of using an optimised rectangle, the user was able to specify the parameters to construct a query window of any desired shape.

Although this added flexibility to the dimensions of the user's relevant query space, the problem with this approach was that the query window's orientation was static, i.e. by changing the orientation of the viewer, simply by rotating about the z-axis, the query window's position was not affected. This was not an optimal solution as the user was still receiving information about data that was behind them and not what was in their direct field-of-view, i.e. along their line-of-sight.

It was as a direct result of this problem that research was carried out on user-based and egocentric query methods. In the following paragraphs, an explanation of directional querying is given. The focus is on egocentric query formulation in the context of the RTD. This includes a description of the various frames of reference than can be used to define direction relations. Following this, a description of the design of the directional querying is given.

There has been a substantial amount of research carried out on directional queries by the research community [Goyal and Egenhofer, 2001; Liu et al, 2003; Liu et al, 2000; Papadias et al, 1994; Pfoser et al, 2000; Shekhar et al, 1999a; Shekhar et al, 1999b; Theodoridis et al, 1996]. It is clear that these type of queries have become important and only now with the emergence of mobile services are they being noticed by a wider audience as a necessity in spatial applications. To make sense of direction, a reference frame must first be established, where in general there are three possible options, *Intrinsic*, *Deictic* and *Extrinsic* as discussed in Section 2.7.1.

For configurations of spatial objects, in a GIS or digital image, that represent real positions and orientations of the environment, it has been customary to use extrinsic reference systems [Goyal et al, 2001]. However, although in [Shekhar et al, 1999b] the direction from fixed objects in 2D space has a profound but highly static effect on the objects relevance in a context-aware environment (i.e. relevance does not change), for our purposes, where each individual has their own personal line-of-sight and therefore dynamic, personalised search space, a deictic reference frame is what is considered [Gardiner and Carswell, 2003].

For example, when working within an intrinsic reference frame, queries like *"Are there any CH artefacts in front of the post office?"* can be answered, where the post office is the object. In contrast, an example of a query within a viewer-based or deictic reference frame would be *"Are there any CH artefacts contained within my view-port in the direction that I am facing?"* i.e. a view-port virtually constructed along my line-of-sight.

Therefore, for this purpose the direction that the viewer is facing serves as the selection condition for queries to the *CHI database*. This section discusses how the position of the viewer, combined with the direction of his/her line-of-sight, is used to develop a viewer-based directional query processor that utilises an oriented, bounded object together with standard *Oracle Spatial* topological and metric operations (See section 2.4.4). This approach uses the Line-of-Sight (LoS) direction vector to represent orientation and constructs a view-port of varying, user-defined dimensions as the primary filter when querying the database. As the direction of the user changes, the view-port is reconstructed in real-time to reflect the user's new line-of-sight search space. This method of querying the database differs from the ITD window query in that that the shape of the view-port window is user-defined and has orientation. This approach does not include new indexing data structures or access methods, but instead utilises the already well-known R-tree index data-structure (See Section 2.6.2.1) to perform the spatial queries [Papadias et al, 1994; Guttman, 1984; Ravi et al, 2002].

Following the results of this research, along with the evaluation of the ITD, it was decided that a component that processed user-based directional queries would be a valuable addition to the RTD.

This concludes the section on the *Revised Technology Demonstrator's* enhanced design. The following section describes the implementation details of the modifications made to the ITD.

4.7 Revised Technology Demonstrator Implementation

In this section, the implementation details of the redesigned and additional components of the RTD are described. This includes the construction, population and indexing of the new *Block, Vegetation* and *Water* layers in the CHI database. Following that, the implementation of the CHI EJB is described. Following this, the use of XML and XSLT for content adaptation in the RTD is explained. Finally, an in-depth description of the implementation of the directional query processor is given.

4.7.1 Database Migration

This section describes the implementation of creating, inserting and indexing the Block, Vegetation and Water layers of the RTD. It also shows how the *CH database* was re-indexed using an R-tree index.

The method used to migrate the VRML data from the MySQL database was to dump the data into a Comma Separated Values (CSV) flat-file format and then load it into Oracle. This operation is a standard operation in MySQL and results in a set of flat files, one for each layer, with ID value pairs in them. These files were then loaded into the Oracle database. The implementation procedures for creating the tables, inserting and indexing the data are described in the following sections.

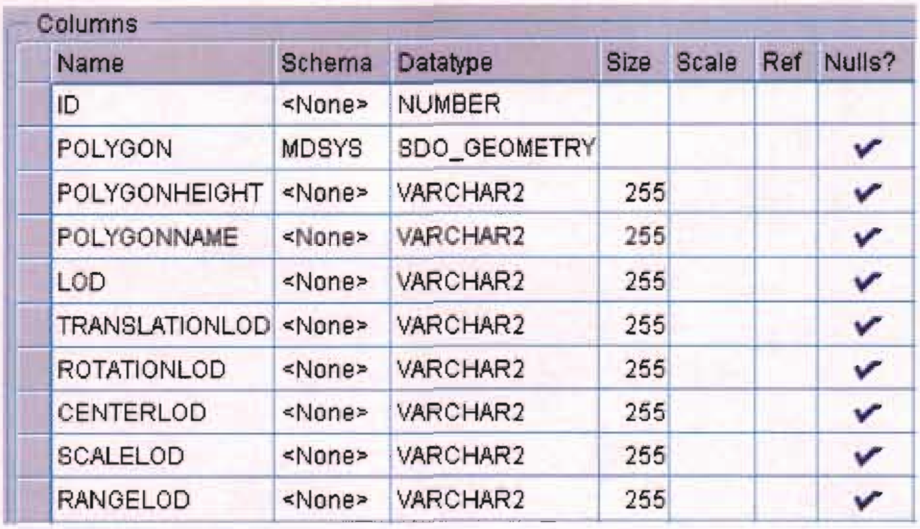
4.7.1.1 Creating Polygon Table

In this section, the design and implementation of the polygon geometry tables are described. One of the major changes to the database design was the method in which the polygon coordinates were stored. In the MySQL database, the coordinates were stored in a normal text field. In the RTD design the spatial coordinates are stored in a table that is similar to the table of the MySQL database except that the coordinates of the geometries are stored in the table as a spatial data type. The geometric description of each polygon in the *Block*, *Vegetation* and *Water* layers are stored in a single row, in a table that has one column of object type SDO_GEOMETRY and at least one other column that defines a unique primary key. The creation of the *Block* table is illustrated in Table 4.16.

```
CREATE TABLE CHI.CHI_GEOM_BLOCK (
      ID NUMBER PRIMARY KEY,
      POLYGON MDSYS.SDO_GEOMETRY,
      POLYGONHEIGHT VARCHAR2(255) NULL,
      POLYGONNAME VARCHAR2(255) NULL,
      LOD VARCHAR2(255) NULL,
      TRANSLATIONLOD VARCHAR2(255) NULL,
      ROTATIONLOD VARCHAR2(255) NULL,
      CENTERLOD VARCHAR2(255) NULL,
      SCALELOD VARCHAR2(255) NULL,
      RANGELOD VARCHAR2(255) NULL);
```

Table 4.16: Create Geometry Table Procedure

This SQL syntax creates the *Block* table. The first column is the polygon ID of type NUMBER, the second column is the spatial column and is called POLYGON and is of type MDSYS.SDO_GEOMETRY. The remainder of the columns are metadata columns. The table is shown in Figure 4.17.



Columns

Name	Schema	Datatype	Size	Scale	Ref	Nulls?
ID	<None>	NUMBER				
POLYGON	MDSYS	SDO_GEOMETRY				✓
POLYGONHEIGHT	<None>	VARCHAR2	255			✓
POLYGONNAME	<None>	VARCHAR2	255			✓
LOD	<None>	VARCHAR2	255			✓
TRANSLATIONLOD	<None>	VARCHAR2	255			✓
ROTATIONLOD	<None>	VARCHAR2	255			✓
CENTERLOD	<None>	VARCHAR2	255			✓
SCALELOD	<None>	VARCHAR2	255			✓
RANGELOD	<None>	VARCHAR2	255			✓

Figure 4.17: Database table for Geometries

4.7.1.2 Inserting a Polygon

In this section, the method used to insert the polygon data into the *Block* table is described. Inserting a Polygon into a spatial column is somewhat more complex than a regular transactional insert. This is because the element type is specified on insertion of the polygon. In the following illustration a 2D polygon is inserted into the spatial column of the *Block* table. This includes an explanation of the element types and parameters required to define the polygon. A complete list of element types can be found in [Geringer, 2001a]

The SDO_GTYPE field is the first of five fields in the SDO_GEOMETRY object (shown in Table 4.17). This four-digit parameter defines the type of geometry being inserted and each digit represents a different aspect of the geometry.

In the code example in Table 4.17, the SDO_GTYPE is 2003. The first digit of this defines the dimensionality. The dimensionality determines the number of ordinates that a point contains. In this case the dimensionality is 2, which means that 2 numbers represent each point in the geometry i.e., a 2D geometry.

```
INSERT INTO BLOCK VALUES (1,
       MDSYS.SDO_GEOMETRY (2003,NULL, NULL,
       MDSYS.SDO_ELEM_INFO_ARRAY (1, 1003, 1),
       MDSYS.SDO_ORDINATE_ARRAY (1435.0, 6.9, 1517.3,
       8.0, 1444.6, 33.4, 1435.3, 6.9)));
```

Table 4.17: Insert Procedure for Polygon Geometry

The second field in the SDO_GEOMETRY object is the SDO_SRID. This parameter associates the geometry with a particular coordinate system. Oracle contains a spatial dictionary called MDSYS.CS_SRS that lists all supported coordinate systems [Geringer, 2001a]. The parameter that is entered into the SDO_SRID field must be listed in the SRID column of this dictionary, or be NULL. All spatial columns in a table must have the same SRID. In this example the SRID is NULL because the coordinates are Cartesian and do not require any specific coordinate system.

The third field is the SDO_POINT field. This field is used in the creation of optimised point data in an SDO_GEOMETRY. Using the SDO_POINT field to create a point provides for optimal storage and performance [Geringer, 2001b]. This field is ignored if the fourth and fifth fields, namely the SDO_ELEM_INFO and SDO_ORDINATES fields are not NULL.

The SDO_ELEM_INFO is the fourth entry in the SDO_GEOMETRY parameter list. If the geometry is not a single point a description of each element contained in the geometry is specified in the SDO_ELEM_INFO_ARRAY. This array is a VARRAY (varying sized array) of type NUMBER [Oracle, 2001]. Entries to this array are specified in groups of three.

1. *Ordinate offset:* Geometries may be made up of one or more elements. This requires an index on the SDO_ORDINATES_ARRAY. The

ordinate offset index value specifies the position of the first ordinate of a particular element in the array that represents a group of elements. The first elements ordinate offset in a group of elements is always 1.

2. *Element Type:* This parameter describes the type of element that is to be inserted (point, line, polygon and others). This is also called the *etype*. The *etype* can be either a one digit or four-digit parameter. The complete list of element types can be found in [Geringer, 2001a]

3. *Interpretation:* The interpretation can take on different meanings. Its value can define whether straight lines or circular arcs connect the elements or the value might mean that a particular element is a header for a compound element.

The SDO_ORDINATES field is the fifth and final field. It is also a VARRAY (varying sized array) of type NUMBER. It contains the coordinates of a geometrical element described by the SDO_ELEM_INFO_ARRAY field. Using a specific combination of these parameters a 2D polygon is inserted into the spatial column of the BLOCK table created above.

Notice how in this example the SDO_POINT type field is NULL and both the SDO_ELEM_INFO and SDO_ORDINATE_ARRAY fields contain values. This is required when inserting any geometry other than the optimised point type [Oracle, 2001].

4.7.1.3 Indexing Polygon Table

It is stated in the literature that the R-tree index is the most efficient type of index for indexing polygon data. In accordance with this assertion and the evaluation of the ITD, it was decided to use the R-tree to index the VRML data in the system.

```
CREATE INDEX CHI_BLOCK_INDEX
      ON CHI_BLOCK (POLYGON)
      INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Table 4.18: Create Spatial Index Procedure (Polygon)

The SQL statement illustrated in Table 4.18 creates an R-tree spatial index called CHI_BLOCK_INDEX on the CHI_BLOCK table. It is created on the POLYGON column of the table and the index type is specified.

4.7.1.4 Indexing CH table with R-tree

It was discovered in the evaluation of the ITD that the Quadtree index used on the CH table was inefficient for the type and frequency of the queries being processed. The decision was subsequently made to use an R-tree index as discussed in section 4.3.2.1. The method used to index the CH table is identical to the method used on the BLOCK table and is illustrated in the following:

```
CREATE INDEX CULTURAL_HERITAGE_INDEX
      ON CULTURAL_HERITAGE (POSITION)
      INDEXTYPE IS MDSYS.SPATIAL_INDEX;
```

Table 4.19: Create Spatial Index Procedure (Point)

The SQL statement in Table 4.19 creates an R-tree spatial index on the CULTURAL_HERITAGE table called CULTURAL_HERITAGE _INDEX. It is created on the POSITION column of the table and the index type is specified.

4.7.2 CHI Session EJB

In this section, a comprehensive description of the role that the *CHI Session EJB* plays in the RTD is given. This includes a brief explanation of its function within the system followed by a description of a path through the system.

The *CHI Session EJB* is used to handle the communication between the Client Layer and the Application Server Layer of the system. There is an instance of this object created for each user session that is active in the system. This EJB is also responsible for monitoring the context of the user, for example, if the location of the user within the 3D model changes, this value must be recalculated and sent to the *Session EJB*. Other relevant context data are passed at this time including the type of mobile device currently being simulated on the client and the user's profile.

SQL statements are used to retrieve, from the spatial database, the data to be displayed on the content pane. For example, when the *Session EJB* receives the users context from the *Interactive Java Option Pane*, a query is constructed and sent to the Oracle database engine for execution. The *Session EJB* is responsible for the construction, validation and execution of SQL statements on the database layer. The query results are returned to the *Interactive Java Option Pane* via the *Session EJB*, which organizes the results into (simulated) device specific format using XML (Section 4.7.3). If the same query is run again (i.e. no change in user context), the IJOP will re-invoke the *Session EJB* that was previously instantiated. The *Session EJB* thereby eliminates the need to re-interpret the query and will cache the parsed version of the previously executed SQL command. This is illustrated in Figure 4.18.

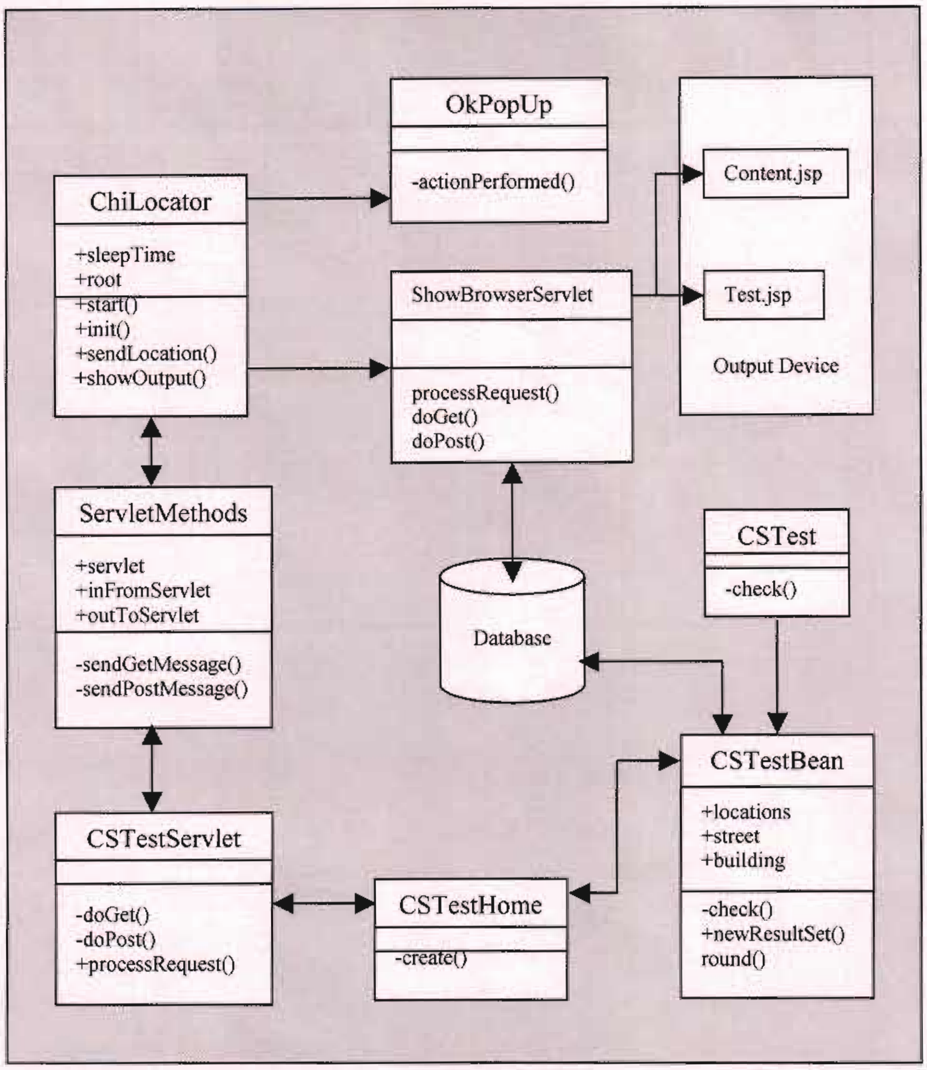


Figure 4.18: Application Server Layer EJBs and Communications

The following is a description of the program invocations in the RTD. This includes a description of a full cycle path through the RTD implementation of the system.

All requests made by the system are made by the *sendLocation* method within the *ChiLocator* class, which is called consecutively at an iteration cycle that corresponds to the *sleepTime* variable specified within the program. The method retrieves all the information required from the applet and sends it to the *CSTestServlet* object via the ServletMethods object. The ServletMethods class is an abstract class for accessing Servlets and is independent of the Servlet and its location.

Once the *ServletMethods* class has been initialised, the information obtained from the applet is passed to the *CSTestServlet* class that resides on the application server. This class is the client component of the *Session EJB* application that performs all the server-side processing. An instance of the *CSTestBean* is created through the *CSTestHome* interface and a remote connection is made to the *CSTestBean* using the *CSTest* interface. This remote connection enables access to the methods contained within the bean. However, it should be noted that at no time is the Servlet connected directly to the bean. All connections to the bean are made via the two interfaces discussed above.

Once the coordinate information is passed to the bean a connection is made to the database and the query results are extracted from the database and sent back to the ChiLocator applet along the same path (Note: the servlet and EJB technologies all exchange information by opening connections that allow information to be passed in both directions).

The data, which consists of record Ids, is then passed to the *ShowBrowserServlet*, which extracts the CH data from the database and displays it in the *Content Pane* of the RTD.

This concludes the section on Enterprise Java Beans in the *Revised Technology Demonstrator*. The following section details the introduction of the content adaptation component of the RTD.

4.7.3 XML & XSLT Processing

In this section, a description of the changes made to the ITD that incorporates a content adaptability approach are described. The main focus of this section is the processes involved in adapting the data into a device specific format using

XML conversion and XSL Transformations. This process is also used to deliver the text data in the system.

The main motivation for incorporating the XML standard in the RTD is its ability to manipulate the content to suit a variety of output devices. One of the main focuses of the *CHI project* is the rendering of the content to a range of mobile devices including WAP enabled mobile phones and handheld computers.

The transformation of the data must be performed entirely on the bottom two tiers of the system i.e. in the application server or the database. This is because the functionality of the current handheld technologies restricts the amount of processing that can occur on the client side of the system and therefore, any transformation must be completed before the content is sent to the client.

Another approach that was considered for the content adaptation component of the RTD was to convert all the content data in the database into XML and store it in this format. However, this was considered to be in conflict with the issue of keeping the database content unchanged. Another factor that was considered in relation to this approach was the lack of support in XML for binary data types. Currently, the CH database stores all of the non-text data as binary data, changing all the content in the database to an XML format does not support these elements.

The main focus of the content adaptation is the definition of the type of content that the device is capable of displaying i.e. if the user is using a mobile phone to retrieve data from the system then it would not be possible to display video content. Hence the XML and XSLT will build documents based on device specific criteria.

4.7.3.1 The XML Parser

The parser selected to parse the data into XML files is the Oracle XML parser. One of the most important layers in any XML oriented RTD is the XML parser. The parser is responsible for taking the raw XML document as input and making sense of the document. What results from an XML document being parsed is typically a data structure that can be manipulated and handled by other XML tools or Java API's. Two of the main factors in choosing a XML parser are speed and conformity. Speed becomes an issue when implementing documents with a high degree of complexity, while conformity becomes an issue when certain parsers omit functionality to gain additional speed. The researcher needed to decide on the balance required between these issues based on the application needs.

The Oracle XML parser conforms to all the above criteria and also gives the option of validating XML files with a DTD or XML Schema. DTDs and XML Schema are technologies that define the structure of an XML document, which is a very important factor in making the content reusable between applications. Both technologies provide constraints that give meaning to the data being represented by the XML.

Another factor that influenced the decision to use the Oracle parser was the compatibility with the Oracle XML SQL Utility (XSU), which allows XML documents to be created directly from SQL commands to the database. This increases the simplicity of the database content retrieval while also reducing the amount of code required in building well-formed XML documents.

For a detailed overview of all the features supported by Oracle XML Parser, see the *Oracle9i XML Developers Kit Guide* [Oracle, 2003].

4.7.3.2 XML SQL Utility (XSU)

The XSU retrieves data from an object-relational database and transforms it into XML. This conversion is required as the first step of the rendering process in the RTD. The package is compatible with Oracle's XML parser and the resulting XML documents can be passed to the parser where they can then be manipulated to a specific format. XSU also supports the generation of an XML Schema given a SQL query, which is an important factor that ensures the resulting XML is reusable by different implementations.

4.7.3.3 The Transformation Process

This section describes the classes implemented on the server side of the RTD to create the XML documents. The classes required to implement the XML parsers are stored in the *xmlparserV2* zip file, which is part of the full Oracle9i application server download and is also part of the Oracle XDK package. The XDK package also contains the XSU classes in the *XSU12* archive. The following illustration of the XML process also contains two classes generated within the project to explore the features and functionality of the two previously discussed archives.

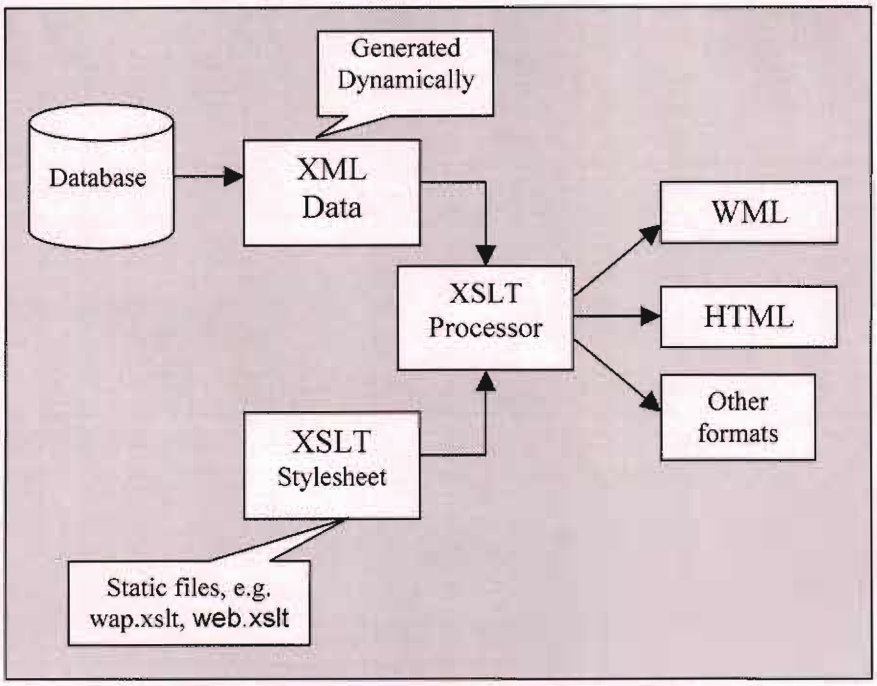


Figure 4.19: Dynamic Content Generation

The first step in the transformation process is the retrieval of the data from the database and takes place in the *XMLGeneration* class. This is achieved using the parser class *OracleXMLQuery* from the *xmlparserV2* library, which stores the query result in XML format. The result can then be converted into a *String* type, *Document* type, etc., depending on the requirements of the user. This generated document is based on the *W3C* DOM standard, which represents the XML document in a parsed tree-like form.

In the *XMLGeneration* implementation, the data retrieved from the database is placed in an *XMLDocument* and then passed to the *XMLXSLOutput* class. This class combines the XML document generated with an XSLT stylesheet to generate the required output for the device implementing the system. This process is illustrated in Figure 4.19.

The XSLT file used in the transformation process is a static file on the server that is read in and converted into an *XSLStyleSheet* object where it is processed with the XML file to generate the output. Each output device has a particular stylesheet based on the individual requirements, although a large number of the devices can implement the same stylesheet. The choice of stylesheet is determined by looking at the mime-types of each device. The mime-type contains information about each device and this data can be used to select the particular stylesheet by storing information on each supported device in the RTD. The complete code for all XML related classes is listed in Appendix B.

4.7.3.4 The XSLT Stylesheet

In this section, the structure of the XSLT stylesheet is discussed and its use in the development of the final content document is described. The XSL Transformation (XSLT) Processor applies XSLT stylesheets to XML documents transforming them into XML, HTML, or virtually any other text-based document.

The CHI XSLT stylesheet implements a multiple template approach to build the output document i.e. it creates a template for each element in the source document that it encounters and hence each template is responsible for a small part of the transformation task. Each template uses an *<xsl:apply-templates>* action to tell the processor to carry on processing the children nodes, allowing recursive traversing of the tree to continue. The following (Table 4.20) is an example taken from the *chi.xsl* stylesheet:

```
<xsl:template match="CHI">
        <table border="0"CELLSPACING="0" WIDTH="300">
                        <xsl:apply-templates/>
        </table>
</xsl:template>
```

Table 4.20: XSLT Stylesheet

The segment of code in Table 4.20 represents a single template. The template is implemented when any element value within the source XML file contains the value CHI. When a match is found a table is constructed with the criteria detailed above and the *<xsl:apply-templates/>* tells the parser to continue on to the other children nodes in the source document. A full version of the XSLT stylesheet implemented in the CHI transformation process can be found in Appendix B.

The method of implementing multiple templates as opposed to using a single template is used because these allow a specific template to be assigned for each individual element in the source document and changing or adding templates will not affect the other templates in the stylesheet.

This concludes the section on XML processing. The processes described in this section are implemented into the RTD prototype. The procedures are used to extract the VRML geometries and all text content from the CHI database. The device specific XSLT stylesheets are then used to render the data. It was not possible to apply this technique to the BLOB data in the system because this type of data cannot be structured using XML. This is because XML is used to represent text-based data and, as BLOB data is extracted from the database in a binary format, this makes it impossible to structure it in an XML format. The complete code for all examples in this section is listed in Appendix B.

4.7.4 Directional Queries

In this section, a description of the implementation of the viewer-based directional query processor component that operates on the application server layer of the RTD and processes Line-of-Sight (LoS) queries on the CHI database is given. This includes the implementation details of a Field-of-View (FOV) algorithm that determines the user's FOV based on the orientation of the user within the VRML model. Following this, the implementation details of a further enhancement to a directional query processor that determines the user's LoS with regard to all other layers in the system, for example, the building block layer.

In the enhanced implementation of the RTD, the method of querying the Cultural Heritage database is more advanced than in the ITD. In the RTD approach, orientation is a necessary parameter so that the user's view-port can be dynamically constructed, resulting in only data contained within the viewer's field-of-view (FOV) being returned. The spatial position and orientation are taken from the viewer's perspective (See Section 2.8.1.3). Using this frame of reference a view-port is defined in real time as the viewer progresses through the virtual space and is used as the primary filter to query an R-tree spatial index (R-tree indexes are detailed in section 2.6.2.1). This makes querying the data more adaptive; as the user progresses through the VRML world the view-port is being continuously updated with respect to the direction he/she is facing. The query that is formulated in this manner is similar to a standard range query with an optimised shape. Similar to the standard range query described previously (Section 4.6.1), the query window is compared to the data represented by an R-tree index. If the query window interacts with any Minimum Bounding Rectangles (MBRs) of the data areas, the window satisfies a topological relation and the result of the query is true.

In the Directional Query Model (DQM), the triangular query window is defined by three points, which represent the extents of the user's field-of-view. The user's viewpoint *p1* is always one of the vertices of the triangle. The points *p3* & *p4* are calculated by first attaining the azimuth from the 0° North direction to the line-of-sight (LoS) of the user, i.e. to point *p2*. The view-port extrapolation is illustrated in Figure 4.20.

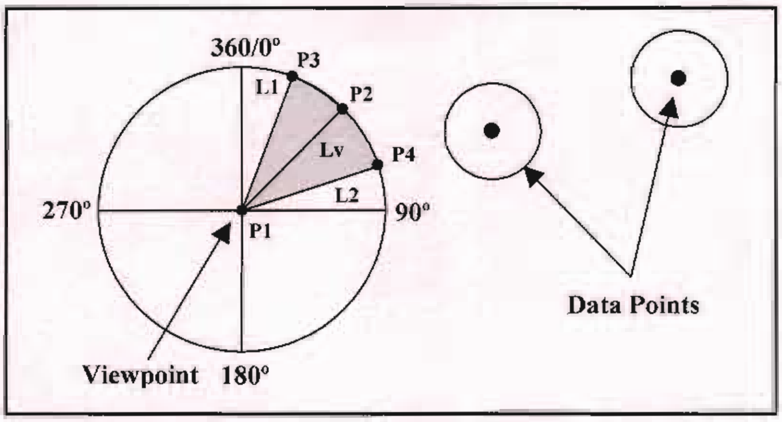


Figure 4.20: View-Port Extrapolation

The azimuth of the line-of-sight (*Lv*) and the position is obtained from the VRML browser (See section 4.3.1.1.4), which simulates obtaining the values from a digital compass, embedded within a spatially-enabled PDA. To determine the azimuths to point's *p3* and *p4,* a specified fraction of an angular FOV value is subtracted from the user's azimuth *Lv* to get the azimuth to point *p3*; addition of the same fraction of the FOV value to *Lv* provides the azimuth of point *p4*. These FOV extents (*L1 & L2*) are then used to calculate the positions of vertices *p3* and *p4* on a query buffer of specified radius, giving the view-port a finite distance. Together, the three vertices produce an optimised spatial query shape that is used as a query window to identify data that lies along the user's line-of-sight.

The position and orientation coordinates that are used by the Java application are obtained by using the External Authoring Interface (EAI) Java Application Programming Interface (API). The EAI is a programming interface for communication between VRML and external programs and allows the developer to register VRML events and properties to the Java programming environment [McCarthy and Descartes, 1998].

The location of the user's viewpoint, while navigating within the VRML model, is used to simulate the user's position in the real-world streets of Dublin. The virtual 3D coordinates (x,y,z) are transformed into geographic coordinates (ϕ,λ), the initial interest of the context-based query to the spatial database. In addition to the position, the orientation of the user's line-of-sight is also obtained in the same way. The orientation field values provide a rotation axis about which to rotate the viewpoint and a rotation angle specifying the degree to rotate around the axis. The initial three values in the field are the X, Y and Z components of the 3D direction vector and the fourth value in the orientation field specifies the positive or negative rotation angle measured in radians [Ames et al, 1997].

In VRML, the z-axis is the vertical axis and is used to specify the orientation of the user. *R* is the angle (in radians) of orientation of the user around any given axis. The *R*-values scale from 0 to +/-3.14 and are subsequently converted into degrees. In the case of the z-axis, this angle is the azimuth and is used to calculate the intersection of the FOV bounding lines and the query buffer. In the VRML data model 0° is *North*, in Java 0° is east (This is shown in Figure 4.21). This discrepancy is adjusted in the application so that all azimuths are synchronised both in the VRML implementation and in the Java implementation.

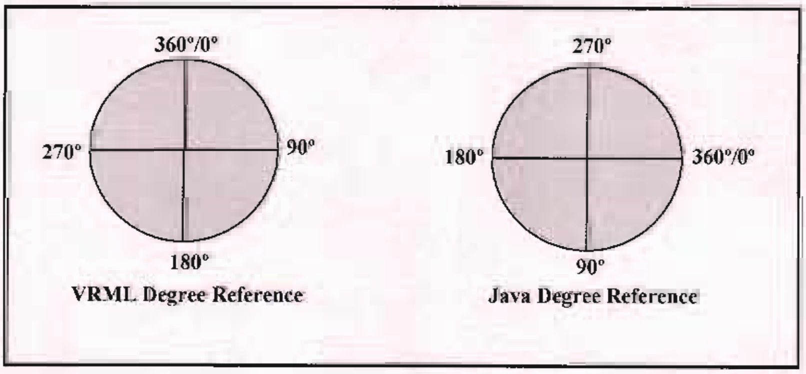


Figure 4.21: Degree Reference Systems

The construction of the query window is performed by the *CreateWindow* method of the *QueryWindow* class, which accepts five parameters and returns the three points that represent the query window. These five parameters are those obtained from the position and orientation nodes of the EAI. The pseudo-code for the *CreateWindow* operation is described as Algorithm 1.

Algorithm 1: Construct Oriented Query Window

```
Input:
        X, Y            //The position of the user.
        Ox, Oy, Oz      //The components of the 3D direction vector.
        Or              //The rotation angle around the vector measured in
                        radians.
Output:
        p1, p2, p3      //Three points

Class QueryWindow {
        CreateWindow (X, Y, Ox, Oy, Oz, Or){
                Angle a = A;
                Radius r  = R;
                toDegrees(Or);
                //adjustment for reference frame
                Vector L1 angle = ((Or - A) - 90);
                //adjustment for reference frame
                Vector L2 angle = ((Or + A) - 90);
                Point p1= X, Y;
                Point p2 = CalcIntersection (L1, R);
                Point p3 = CalcIntersection (L2, R);
                Return p1, p2, p3;
}
CalcIntersction (angle, radius){
                Return point;
        }
}
```

Following execution the three points returned from the *CreateWindow* method that represent the view-port are used in an SQL query string to query the spatial database for CH artefacts. In the SQL statement the *sdo_elem_info_array* is modified to represent a polygon, i.e. the query window.

```
SELECT A.ID, STREET, BUILDING
       FROM  CHI.CHI_CONTENT_DATA A
       WHERE SDO_RELATE (A.POSITION,
       MDSYS.SDO_GEOMETRY (2003, NULL, NULL,
       MDSYS.SDO_ELEM_INFO_ARAY (1,1003,1),
       MDSYS.SDO_ORDINATE_ARRAY (X1, Y1, X2, Y2, X3,
       Y3, X1, Y1)),
       'mask=anyinteract querytype=window')= 'TRUE';
```

Table 4.21: Directional Query Window Procedure

The *sdo_elem_info_array* contains the values that define the type of geometry to be queried against the CH layer. In this instance, it is a polygon and is defined by a triplet value (1 (offset), 1003 (outer polygon), 1 (points are connected by straight lines)). The three points obtained from the *QueryWindow* class are specified in the *sdo_ordinate_array* and the first is repeated to close the polygon.

This completes this section on the directional query aspects of the query processor. All programs referred to and relating to this section can be found in Appendix A.

4.7.5 Line-of-Sight (LoS)

In this section, the concept of the Line-of-Sight (LoS) algorithm that is a component of the directional query processor is explained. This includes a detailed description of the LoS problem and following that, the integration details of the algorithm into the processor. This includes point-to-point LoS determination and point-to polygon LoS determination.

Following the development of the directional view-port, it became apparent that some of the data returned from the spatial queries were not entirely realistic because a portion of it came from areas inside buildings in the VRML model. This irregularity led to an investigation into hidden surface determination algorithms [Foley et al, 1990] and ultimately the development of a LoS algorithm to determine if the data points situated in the view-port are, in fact, in the viewer's line-of-sight. This problem is illustrated in Figure 4.22. The large triangular area in the diagram represents the user's view-port in 2 dimensions. The brick patterned shapes B1, B2, B3, and B4 represent building blocks and the black points D1, D2, D3, and D4 represent CH data points. The enclosed white space in the triangle highlights the desired shape that the LoS algorithm should identify as the query area. The light grey sections in the triangle represent the areas that should be excluded from the query space, as they are not visible from the user's viewpoint.

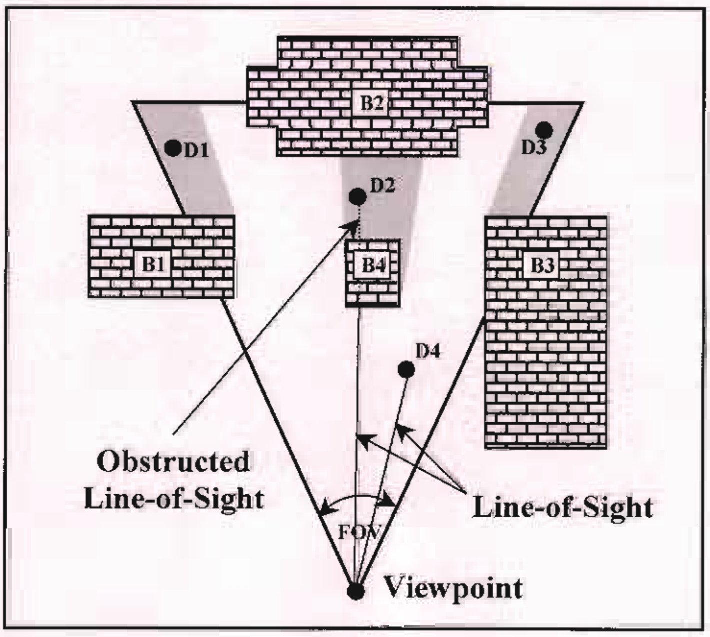


Figure 4.22: Optimised View-Port with LoS

When the *CHI database* is queried using an oriented view-port, the query simply checks to determine if the triangular view-port intersects with the CH data layer in any way. If there is a topological relationship is detected the data are placed in a result-set and the relating multimedia content is subsequently displayed to the user. Theses queries do not take into account the fact that the query window is also interacting with the other layers in the database (e.g. the building block layer).

In actuality, if the user is looking into a building, he/she cannot see what is inside. In our initial implementation of the query processor, the user could indeed retrieve data about an area that was not visible. As an initial filter, this is unacceptable. Therefore, the query processor now checks to determine if the view-port interacts with any of the building blocks in the block layer in the database. If so, the sub-areas of the triangular-shaped view-port that overlaps the building layer are removed from the query window. This is illustrated in Figure 4.22, where the shape of the triangular query window is reduced to the enclosed white and grey space only and excludes B4 and the intersecting parts of B1, B2 and B3.

Performing this check eliminates the building blocks from the view-port while supplying data on what buildings the view-port is intersecting with. This data can later be used to determine if the building blocks involved in the intersection are obstructing the user's line-of-sight to other data points that may not be inside the buildings but behind them.

Additionally, in comparison with the CH layer, the building block layer also contains attribute information about the individual polygon objects (buildings) included in the layer. Therefore, in the case of the block layer, attributes like the name, address, purpose and associated history of the buildings are linked to each building object. This metadata is just as useful to the user as any other CH artefact data and consequently the LoS algorithm is applied to the building block layer also to determine the building blocks that are in view.

However, the two problems mentioned above have slightly different solutions. In the case of the CH layer, the solution to the LoS determination is less complex than LoS determination for the block layer. This is because the query necessary to determine the LoS between the viewpoint and a data point only requires that a single line is queried against the block layer. The LoS determination between the viewpoint and a polygon is more complicated because the number of possible intersections is greater, thus an increased number of lines are required in the LoS query.

Determining the LoS for the CH and block layer involves using a combination of *Oracle Spatial* operators to execute the LoS algorithm. The approach taken was to take the *Scanline* LoS algorithm [Foley et al, 1990, p.680] in *Computer Graphics* and apply it to the area of spatial database query processing. The *Scanline* algorithm, the *z-Buffer* algorithm [Foley et al, 1990, p.668] and the *Binary Space-Partitioning Tree (BSP)* algorithm [Foley et al, 1990, p.675] were considered but the *Scanline* approach was chosen because the topological and boolean operations needed to process the algorithm are already inherent in Oracle Spatial.

- *Scanline Algorithm* – The scan-line algorithm was developed to determine the visible surfaces of objects [Foley et al, 1990] and operates by using a sweep-line algorithm to scan the area in question (field-of-view) to determine if there are any objects in the sweep path and, if so, identifies the visible surfaces. This is achieved by maintaining an *edge table* that stores the polygon edges while scanning the area. The visible surfaces are then identified using a complex algorithm.

The sweep-line algorithm utilised by the directional query processor in the *CHI system* takes a vector-based approach to scanning the query space. In contrast to the raster-based implementation of the algorithm the query processor employs a vector-based implementation.

The vector-based technique works on the same principal except the scan-line and the query image (layer) are both vector objects. The algorithm uses a series of vector scan-lines to scan the query area to determine where they intersect objects. This data is subsequently used to determine line-of-sight.

4.7.5.1 Point-Based Line-of-Sight

Implementing the scan-line algorithm for the CH layer is accomplished using Algorithm 2 and the *sdo_intersection* operator in Oracle. First, a series of lines is created between the viewpoint of the user and the data points inside the view-port. These lines are the scan-lines. In turn, *Algorithm 2* accepts each scan-line as an input parameter to determine if it interacts with any of the objects in the block layer. If there is any interaction between the objects in the block layer and the scan-line, the CH artefact in question is not visible to the user from that viewpoint and is not placed in the direct LoS resultset. If the scan-line does not intersect with the block layer, the data point is considered to be visible and is placed in the direct LoS result set.

4.7.5.2 Polygon-Based Line-of-Sight

Determining the line-of-sight between the viewpoint and polygons in the block layer using the scan-line algorithm is more complex and therefore more computationally expensive than the point-to-point approach used on the CH layer. In this case, the LoS between a point and a polygon has to be determined, meaning there may be many possible LoS occurrences. The algorithm is repeated as before and, until there is a positive LoS is detected, every possibility has to be checked.

The process involves initially testing if the view-port interacts with one or more of the building objects in the block layer by using the *sdo_intersection* operator. If so, the IDs of these objects are placed in an array. The spatial difference between the view-port and the intersecting block object is then calculated using *sdo_difference* operator. The *sdo_difference* operator returns an *sdo_geometry* object that represents the difference polygon between the two geometries. If there are multiple intersecting objects, the next block object in the array is compared with the result of the previous *sdo_difference* operation to determine the new difference polygon. This procedure continues until all objects in the array have been processed. A result of this process is the view-port (triangular area) in Figure 4.22 with building blocks B1, B2, B3 and B4 removed, i.e. the shape of the enclosed white and grey space only.

The next step in the Point-to-Polygon LoS algorithm is to connect each point of the difference polygon object (each having an associated intersecting object ID) to the viewpoint. In turn, each of these scan-lines is tested until at least one scan proves negative for intersection with another object. If this is the case it is evident that there is indeed an unobstructed line-of-sight to that block object and therefore it is visible in some way. Processing of the remaining scan-lines relating to that particular object is stopped and the ID of the object is placed in the direct LoS result set and the next object in the array is tested. The ID of the block is added to the list of objects in the user's line-of-sight and the metadata related to it is available to the user.

In the case where no points of the intersecting object are in the user's LoS an additional scan test of the object is made. This ensures that the object is not currently in the user's FOV. This test is as follows: a series of points (e.g. 1m apart) is calculated around the perimeter of the block object. Each point is connected to the viewpoint with a straight line. The scan-line algorithm (Algorithm 2) is used to determine if there is a direct line of sight. If all scan-lines intersect with other objects, the block object is clearly not visible to the user. However, if one or more scan-lines don't intersect another block object, the current building object is in view and is added to the user's direct LoS result set.

Following the processing of all data points and block objects for LoS, the list of objects in the user's line-of-sight is supplied to the narrative engine of the system for metadata processing. The narrative or storytelling engine uses a semantic filter to create a hyper-linked digital story based on events that occurred in the area surrounding the user [Carswell et al, 2002]. A comprehensive overview of the semantic aspects of the *CHI project* can be found in [Neumann, 2003].

Algorithm 2: Scan Line Algorithm

```
Input:      Scanline 1 → N
            Buildings 1 → N

Output:     Intersection points

Class ScanLine{

        For (each Scanline){
                CheckIntersection (scanline, layer);
        }

CheckIntersection (vector scanline, layer lay){
        If (scanline interacts with polygon){
                return point of intersection
        }else{
                return initial point
        }
    }
}
```

This completes the section on the LoS determination aspects of the directional query processor. All programs referred to and relating to this section can be found in Appendix A.

4.7.6 3 Dimensional Queries

In this section the idea of 3-D queries is introduced. It is focused around the fact that the human Field-of-View is measured in 3-Dimensions and that height is an important factor when performing spatial queries. There is a description of the inhibiting factors that make it impossible to develop such a query and following this there are the implementation details of a 2.5D query that uses height as a parameter in the spatial query but does not include vertical angular LoS.

The enhanced implementation of a directional query processor with 2D orientation and LoS functionality greatly increases the relevance of the data that is returned by the query.

In computer graphics, querying in 3-Dimensions is a trivial task because the algorithms to perform such queries are deeply embedded in the technology. However, this is not the case in the area of spatial databases. The need for these algorithms in spatial database systems has only become apparent recently and is still not integrated into the systems [Gardiner and Carswell, 2003]. Even though *Oracle Spatial* can store 3D spatial data, as of yet (2004) it cannot perform 3D queries on the data. It would be possible to partially implement these algorithms as part of the directional query processor but it is beyond the scope of the *CHI project*. This shortcoming of spatial database technology led to the development of a compromised 2.5D query.

Previously in the ITD, each CH data point contained in the database had a 2D coordinate associated with it, x and y. This was sufficient because the queries being generated required only 2D point data sets when querying the database. This meant that any data that was present within the view-port was relevant regardless of user's vertical field-of-view.

The human Field-of-View (FOV) (Figure 4.23) spans approximately 200° horizontally, taking into account monocular peripheral vision, and 135° vertically [Arthur, 2000]. This limits what can be seen at any one time. The normal binocular FOV is 120° with and extra 70° of monocular vision (35° each side). The human field-of-view also has an angle of 60° above the direct line of sight and 75° below it. This means that the height of data in the model should be taken into account as well as the vertical area that is being searched. This can be partially achieved by adding an additional coordinate (z) to each data point in the database giving it height. The z-value is then used as a height parameter in the directional query to determine the relevant data based on position and height. The view-port can also have on *offset* height value from the ground giving the view-port a floor and roof. This enables queries like *"Are there any cultural heritage artefacts contained within the view-port in front of the viewer up to a height of 10 meters off the ground?"* and, where the view-port has a height offset, queries like *"Are there any cultural heritage artefacts contained within the view-port in front of the viewer that are between 10 and 15 meters off the ground?"* This is illustrated in Figure 4.24.

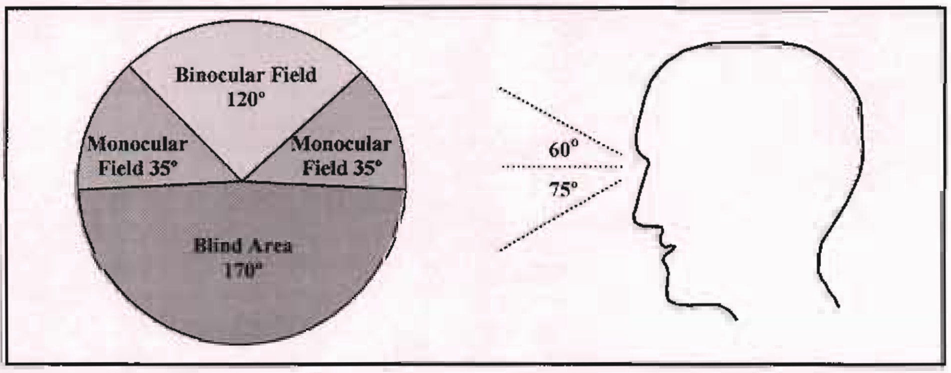


Figure 4.23: Human Field-of-View [Arthur, 2000]

An example of the 2.5D directional view-port is shown in Figure 4.24. Point's *p1*, *p3* and *p3* represent the 2D dimensions the view-port. The height *h* is the required height of the query space and is specified by the user. It has a *min* value for the height offset and a *max* value for the height extents of the query space. The difference between 2.5D and 3D in this context is that a 3D query block would be a pyramid shape as opposed to a triangular block (Figure 4.24).

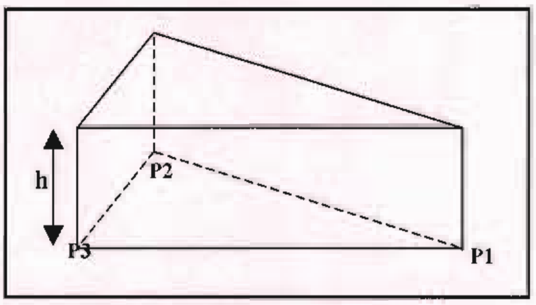


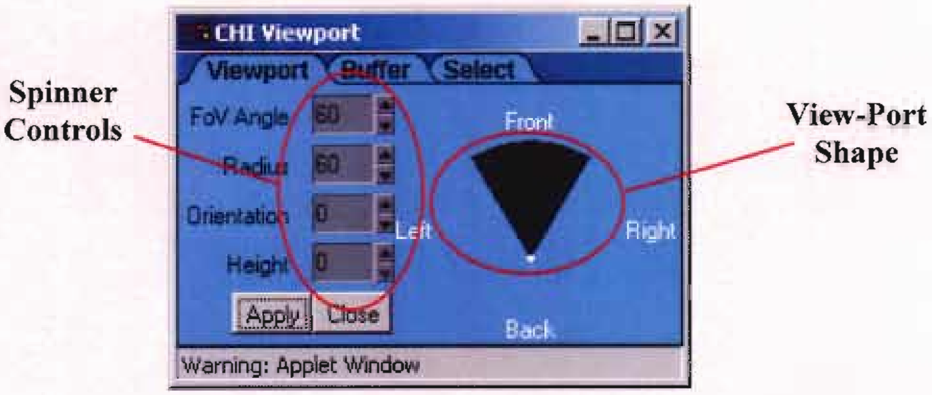
Figure 4.24: 2.5D Query Block

This approach extends the query model by adding the ability to construct essentially a 3D viewer-based directional query for the search space. The query is processed initially using topological and metric operations (See Section 2.4.4). Following this the specified constraint is used to check the height and consequently further reduce the result set to data that satisfies the constraint.

4.7.7 View-Port Modification

In this section, the view port controller is presented. This is a tool developed as part of the *CHI project* to give the users of the system control over the dimensions of the view-port. The user has the option to change the horizontal FOV angle of the view-port and the radius or distance it extends. The ability to modify the orientation of the view-port with reference to the user's orientation is also available. This functionality enables queries like *"Are there any cultural heritage artefacts contained within the view-port at 30° East of the viewer's*

LoS that are between 10 and 15 meters off the ground?". The view-port controller is shown in Figure 4.25.



FoV Angle 60

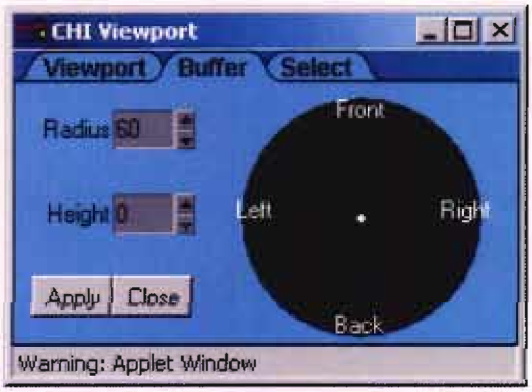
Radius 60

Orientation 0

Height 0

Figure 4.25: View-port Controller

It became apparent during the development of the View-Port controller that the directional view-port might not always be the preferred option for the user. For example, a user might want to query the area surrounding them and not just in their FOV. This led to the addition of a series of tabs to the View-Port controller. A buffer tab was added and gives the user the option to query the 360° surrounding area, the dimensions of which are controlled by two spinner controls that adjust the radius, size and height of the view-port. This is illustrated in Figure 4.26.



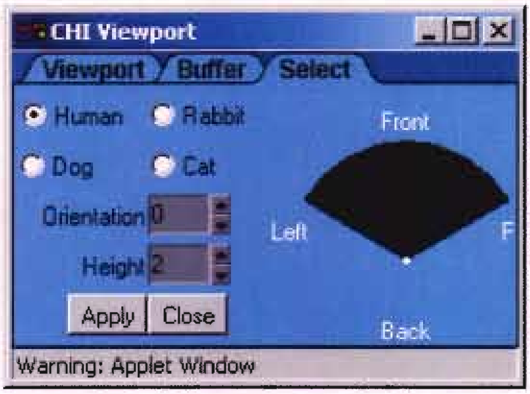
Radius 60

Height 0

Figure 4.26: Buffer Control

The next tab on the controller is the static selection control that gives the user a selection of predefined view-port configurations that mimic four different

FOVs. The default is the human FOV at a height of 2m. There is also a choice between various alternate preset FOV configurations (Rabbit (0,2m), Cat (0,3m) and Dog (0,5m) as well as a user defined FOV where orientation and height can be specified. This addition is used to demonstrate the different FOV shapes that exist. This is illustrated in Figure 4.27.



● Human ○ Rabbit
○ Dog ○ Cat
Orientation 0
Height 2

Figure 4.27: Select Control

Depending on the user's personal choice the dataset can be queried over a large area or the query can be narrowed down to a very small sub-region. This concludes the section on view-port modification. The following is a summary of the CHI framework implementation.

4.8 Summary

In this chapter, the design and development of the CHI Framework has been described. This includes the design and development of the *Initial Technology Demonstrator*. Following this an evaluation of the prototype was performed. This evaluation was very beneficial and prompted a number of design changes that led to the design and implementation of the *Revised Technology Demonstrator*.

The result of the successfully completed project is a dynamic system that enables the users to navigate spatially through a 3D environment while information based on the spatial location, orientation and semantic meaning of a data set is presented. This is illustrated in Figure 4.28.

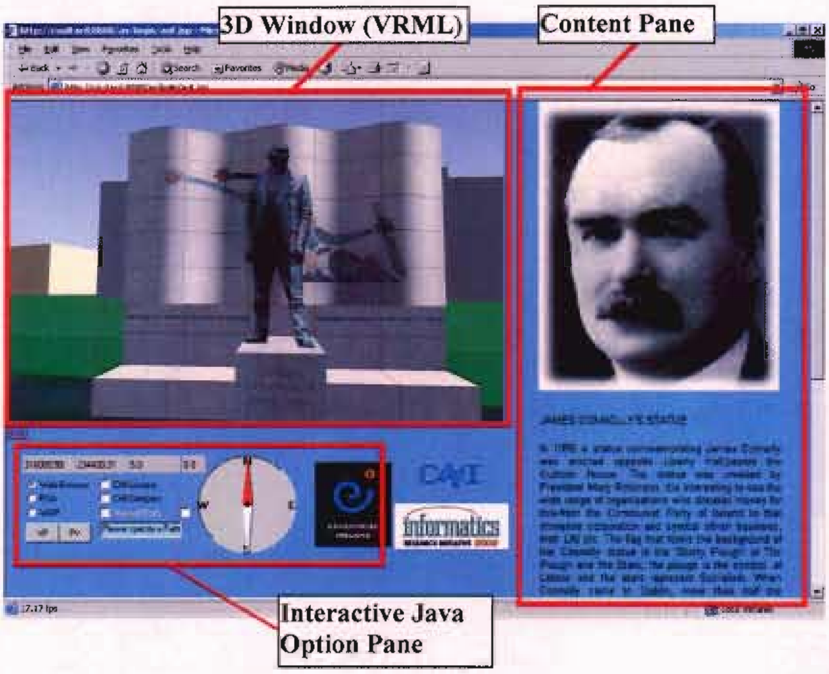
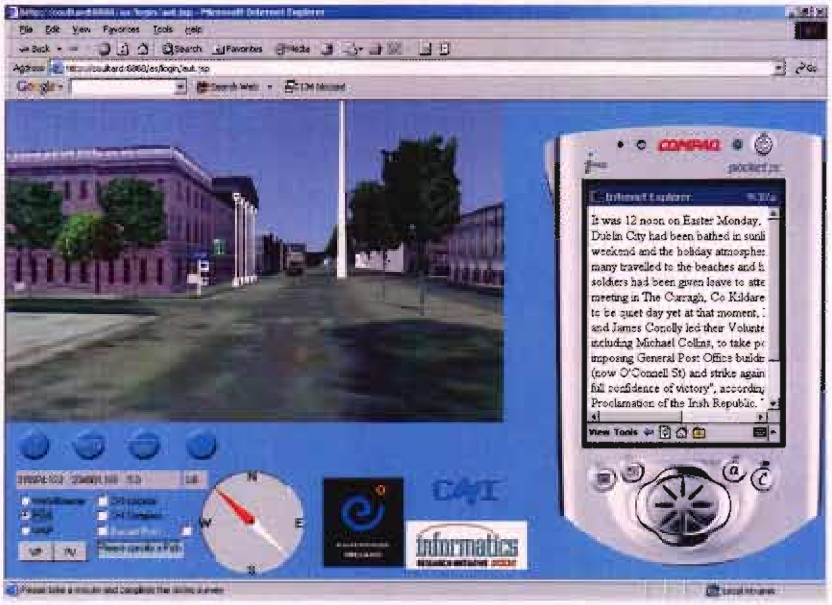


Figure 4.28: Revised Technology Demonstrator Interface

The RTD consists of a *3D window*, *Content Pane* and the *Interactive Java Option Pane.* The *Content Pane* may be replaced with simulated mobile devices that are capable of displaying the type and amount of data that corresponds with the display characteristics, transmission and reception capacity of the actual devices. This is illustrated in Figure 4.29.

Once the RTD implementation was completed, the prototype was tested to verify that everything functioned properly. When this was completed, the next phase was to undertake the evaluation procedure and this is discussed in the following Chapter.



It was 12 noon on Easter Monday.
Dublin City had been bathed in sunli
weekend and the holiday atmosphe
many travelled to the beaches and h
soldiers had been given leave to atte
meeting in The Curragh, Co Kildare
to be quiet day yet at that moment,
and James Conolly led their Volunte
including Michael Collins, to take p
imposing General Post Office buildi
(now O'Connell St) and strike again
full confidence of victory", accordin
Proclamation of the Irish Republic.

Figure 2.29: Simulated Personal Digital Assistant (PDA)

Chapter Five

Evaluation and Results

5 Evaluation and Results

5.1 Introduction

The purpose of this Chapter is to consider the evaluation of the *Revised Technology Demonstrator* from three different perspectives. The evaluation follows previously agreed evaluation strategies set out for the *CHI project* [CHID7.1, 2002].

The overall objective of the evaluation was to test the system to ensure that the *Revised Technology Demonstrator* addressed the user requirements. The evaluation objectives are as follows:

- To test the technical robustness of the system and the conformity of the system to the user requirements.
- To ensure that the system content and functionality corresponds to the specification.
- To demonstrate that the system (i.e. architectures, interfaces) is accepted by potential end-users.
- To report problems and recommendations to the technical team in order to enhance the system.

The evaluation process was designed to assess system conformity and usability and content under field conditions and was divided into three sections.

1. Internal Evaluation
2. External Evaluation
3. Scalability Evaluation

The internal evaluation was a process that involved a technical evaluation of the system by experts within the research group. This was achieved by drawing up a set of system criteria. The external evaluation involved gathering information from potential end-users. The scalability evaluation was an evaluation of the changes that would be necessary to bring the system from a simulated environment to a real-world environment.

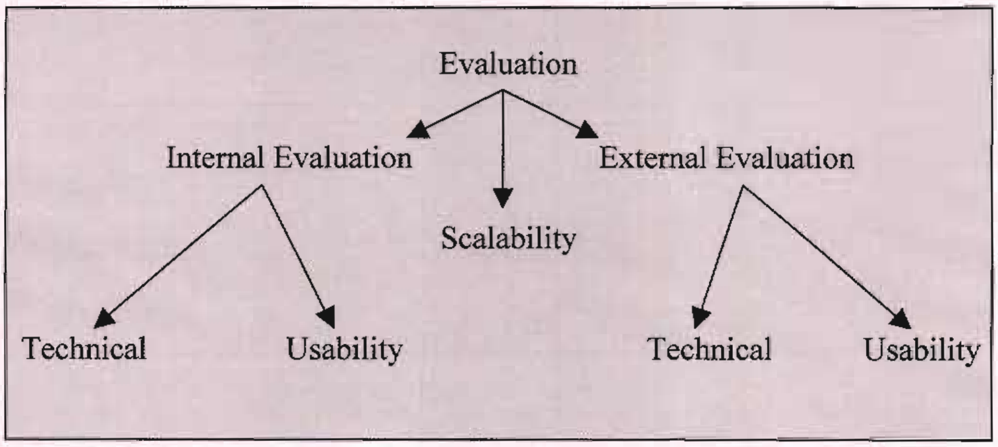


Figure 5.1: The Evaluation Process

5.2 Internal Evaluation

The objective of the internal evaluation was to compare and contrast the *Revised Technology Demonstrator* with the user requirement study [CHID2.1, 2001] and the project specification [CHID3.1, 2001]. This internal evaluation consisted of an *Internal Technical Evaluation* and an *Internal Usability Evaluation*.

5.2.1 Internal Technical Evaluation

Individuals who have knowledge of web-based applications and architectures performed the *Internal Technical Evaluation*. This evaluation was based on a set of criteria formulated by the research team.

5.2.1.1 Correspondence to Specification

The internal technical evaluation was used to verify that the user requirements outlined in the [CHI2.1, 2001], were met by the *Revised Technology Demonstrator*. This task was performed by the research team, who carried out the evaluation by accessing the system, and checking that all the functionality elements listed in Section 3.1 of [CHID3.1] were met. The results of this evaluation are outlined in the following description of the completed system.

5.2.1.1.1 System Architecture

For the *Revised Technology Demonstrator*, the main components are implemented in a three-tier web-based architecture, which consists of three layers, namely the Client Layer, Application Server Layer, and the Database Layer. The system has been implemented on a Windows 2000 Server platform with a particular focus on spatial query generation on the middle tier of the system.

All communication between the client layer and the database layer are conducted through the application server layer. The application, in this case, the query building and result formatting, is run on the application server layer. The client communicates with the application server using the HTTP networking protocol.

5.2.1.2 Interoperability

Interoperability was a key aspect of the RTD. Following the evaluation of the ITD, it was decided that it was essential to increase system modularity to enable individual components to be updated or replaced without affecting the system as a whole.

One of the first parts of the RTD to be enhanced using this approach was the view-port component. This aspect was developed as a separate component to ensure that it was in effect plug-and-play. It was required that this component could be passed to another research team member who had no prior knowledge of the component for use. It was regarded as interoperable if the team member could successfully integrate it into the system without any knowledge of the internal workings of the component.

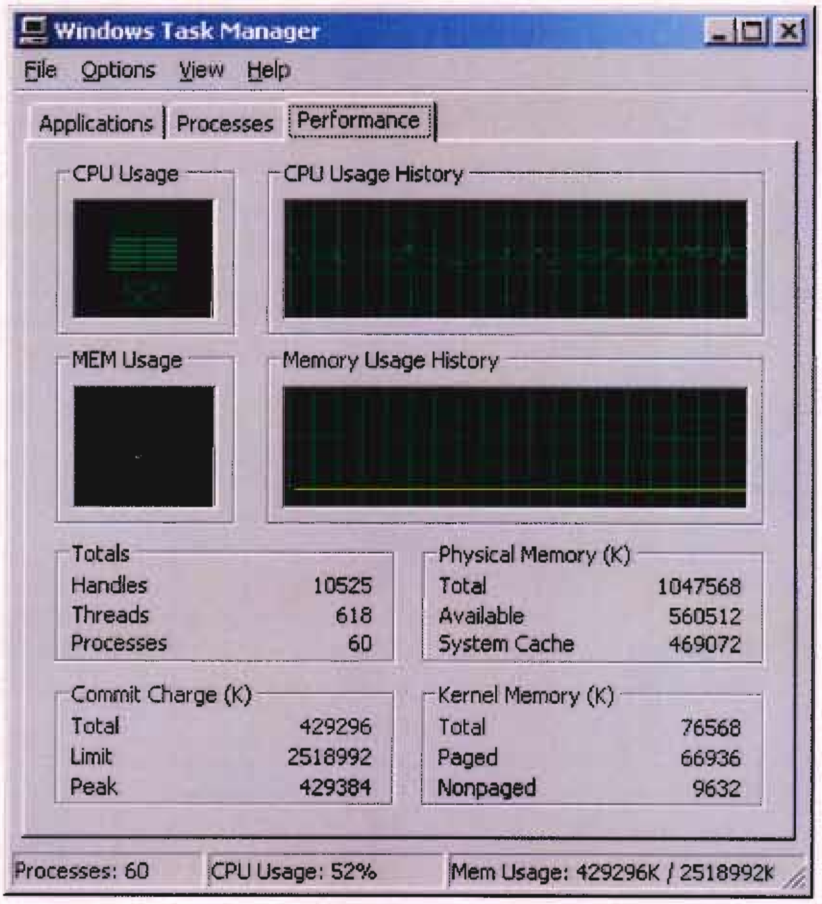
5.2.1.3 Performance

To evaluate the performance of the *Revised Technology Demonstrator* the research team carried out a performance analysis test. This was achieved by analysing the CPU and Memory usage of the Client and Server machines involved in the evaluation.

5.2.1.3.1 Client Side Performance

The client machine for this test was a standard PC computer with 512MB memory and a 1.7GHz Intel Pentium 4 processor. In addition it was equipped with a 3D accelerator graphics board (GeForceMX4200), which unburdened the CPU floating point unit from calculating the 3D data transformations. The client test involved one user who was instructed to "Explore Dublin City Centre" for 20 minutes. To evaluate the performance, the CPU usage and Memory usage for the client machine was monitored during the test by a researcher.

The average CPU usage for the test was 48%. This is a relatively high but acceptable figure for the client. The memory performance for the client test was continuously in the lower third of the available memory and proves a high-quality implementation of components in regards to memory usage. The demanding component for the client application is the calculation of real-time 3D computer graphics objects. For the *CHI Project* it was of vital importance to minimize the 3D display hardware resource prerequisites, so that it could be used on a wide range of average computer platforms for end users. The low memory usage recorded as part of the test was as good as expected since the design and implementation of 3D processing software components was geared towards performance on a minimum 3D hardware requirement. The CPU and Memory usage for the test is shown in Figure 5.2.



Totals
Handles 10525
Threads 618
Processes 60

Physical Memory (K)
Total 1047568
Available 560512
System Cache 469072

Commit Charge (K)
Total 429296
Limit 2518992
Peak 429384

Kernel Memory (K)
Total 76568
Paged 66936
Nonpaged 9632

Processes: 60 | CPU Usage: 52% | Mem Usage: 429296K / 2518992K

Figure 5.2: Client Side Performance Monitor

5.2.1.3.2 Server Side Performance

The server computer was a Microsoft Windows Server machine with a 1.4Ghz Intel Pentium 3 processor and 256MB of memory. The machine was responsible for processing incoming requests from clients wishing to query the machine for Cultural Heritage (CH) information. Beyond the query processing, the machine also serves as host for user interface components and a spatial (virtual) database. The complete user interface is built dynamically on user request. This test was run in conjunction with the client performance test and lasted for 20 minutes.

The general server performance test derives from simple Memory and CPU activity. CPU performance remained well within an acceptable range of 10-90% with an average of 50% load for one client. In particular, the client user interface loading, and query building and processing stress the CPU. While CPU performance may be considered as acceptable, memory performance must be regarded as unacceptable for future system development. Total memory was observed to be continuously at maximum (100%) (See Figure 5.4). This can be explained by the fact that the Oracle database takes upfront a substantial memory allocation. With 256MB RAM the machine is not suitable for Oracle 9i database components. However, the system was acceptable for evaluation purposes since tests run with more than five clients could process requests in an acceptable time with the aid of Java garbage collection enhancement.

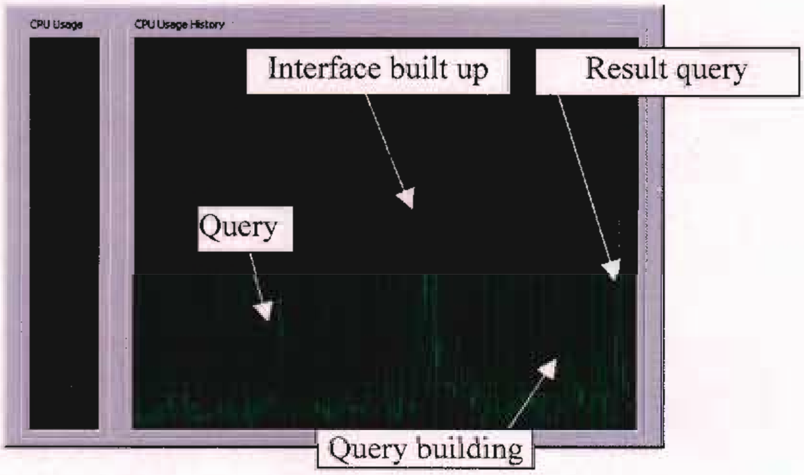
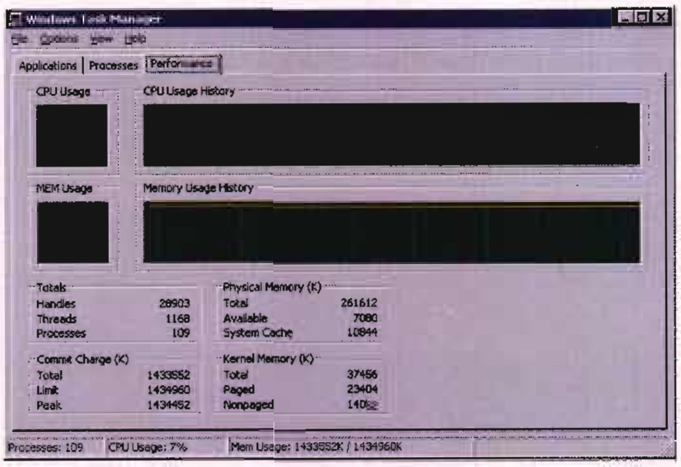


Figure 5.3: Server Side CPU performance for one single user session



```
Totals                         Physical Memory (K)
Handles          28903         Total            261612
Threads           1168         Available          7080
Processes          109         System Cache      10844

Commit Charge (K)              Kernel Memory (K)
Total          1433552         Total             37456
Limit          1434960         Paged             23404
Peak           1434452         Nonpaged          14052

Processes: 109   CPU Usage: 7%   Mem Usage: 1433552K / 1434960K
```

Figure 5.4: Server Side Memory and CPU Performance

5.2.1.4 Latency

In real-time systems, latency – the time delay between (input) stimulus and (output) response – is regarded as a critical factor in system performance and is clearly of importance in the context of the *Revised Technology Demonstrator (RTD).* In the case of the *CHI system,* the latency is the time interval between an initial view-port interaction with the CH data set and the final display of the required data.

A latency test performed on the *Revised Technology Demonstrator (RTD)* produced poor evaluation results and this prompted a number of system design changes which resulted in a significant reduction in system latency. The changes introduced included the use of a different spatial index (R-tree) and an increased rate of query processing (from five to two seconds). Following these changes, it was necessary to quantitatively assess system latency. To achieve this the following measurements were employed:

5.2.1.4.1 Measurement Method

The measurement method used to calculate the latency of the RTD was very simple. It involved navigating through the VRML model in *Locator Mode* while searching for CH data points. Locator mode refers to the system state when querying the view-port against the database.

The data collection step involved setting up two timers within the RTD application. The first timer captured the time in milliseconds at which the users view-port intercepted a CH data point and the second timer captured the time in milliseconds that the corresponding data was displayed on the *Content Pane*. Using these results, the time it took to extract and display the data to the user was calculated. All tests were performed on the client machine configuration detailed in Section 5.2.1.3.1.

Twenty tests were carried out on the system. these indicated an average latency of 1.1 seconds, a figure which is acceptably close to the a-priori target figure of 1 second. This result is satisfactory and clearly shows that the earlier latency problem of the ITD had been solved.

It is worth mentioning that the results were obtained while the user was travelling rapidly at a (simulated) constant speed of 10m/s through the model. In reality, a normal pedestrian user would be unable to move at this rate. The results of the tests are illustrated in Figure 5.5.

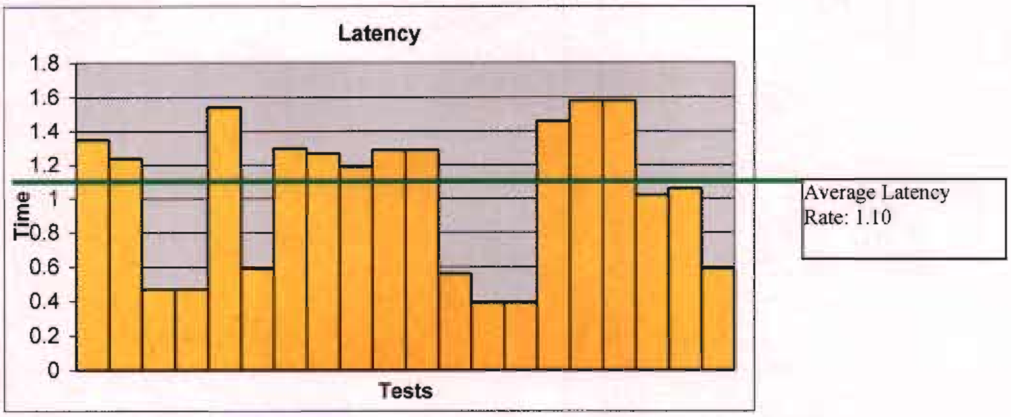


Figure 5.5: Latency Test Results

5.2.2 Internal Usability Evaluation

Individuals who have knowledge of usability methods performed the internal usability evaluation. This evaluation was a summative evaluation, which aims to determine the overall quality of the interface [Nielson, 2003]. This is achieved by creating a set of criteria that the evaluators of the system will follow to test the usability of the system. The criteria used to test the system are detailed in the following sections.

5.2.2.1 Ergonomics (user-friendliness)

During the design of the RTD the basis for the user-friendliness aspects of the system came from previous research carried out by [McAtamney, 2004]. The ergonomics of the RTD were tested using a collection of usability guidelines put forward by [Nielsen, 1993]. Using this approach, Nielsen compiled a set of 10 guidelines from the large number of currently accepted ones. These are explained in the following:

- *Simple and natural dialogue* – Dialogues should not contain information that is irrelevant or rarely needed. Every extra unit of information competes with the relevant units of information and dismisses their relative visibility. All information should appear in a natural and logical order.

- *Speak the user's language* – The dialog should be expressed clearly in words, phrases, and concepts familiar to the user, rather than in system-oriented terms.

- *Minimise the user's memory load* – The user should not have to remember information from one part of the dialog to another. Instructions for the use of the system should be visible or easily retrievable whenever appropriate.

- *Consistency* – Users should not have to wonder whether different words, situations, or actions mean the same things.

- *Feedback* – The system should always keep the users informed about what is going on, through appropriate feedback within reasonable time.

- *Clearly marked exits* – Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.

- *Shortcuts* – Accelerators (unseen by the novice user) may often speed up the interaction for the expert user such that the system can cater for both inexperienced and experienced users.

- *Good error messages* – They should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

- *Prevent errors* – Even better than good error messages is a careful design that prevents a problem form occurring in the first place.

- *Help and Documentation* – Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, be focused on the users task, list concrete steps to be carried out, and not be too large.

The number of users that participated in this test was six. These users were asked to compare the system against the list of usability principles to clarify whether the prototype is suitable for end users. This proved the most effective way of clarifying whether the prototype was suitable and if not, what needed to be changed.

The evaluators of the system were given the list of principles and asked to use the system, independent from any help from the author, and then comment on it, based on the list. The system was well received and some extremely valuable comments were given. The main findings to emerge from this test were:

1. In relation to the initial pages, it was pointed out that a more user-friendly and intuitive entry point is needed. This was because the links that allow the user to proceed to the next page were situated at the bottom of each page and were not visible to the user, causing confusion.

2. After the login procedure, a more comprehensive introduction relating to the different aspects of the system was needed. It was found that the users were unaware of many of the features such as the view-port control simply because they were not informed of them. For this reason an introduction to the actual elements of the interface was provided.

3. The information towers that represent the information areas within the model were received well, but it was found that the check box that initiated them was not clearly labelled.

4. To keep the VRML browser clutter-free, it was decided early on that the onscreen toolbars were obstructive and that the default setting would be "off". However, during this test it was suggested by 4 members of the group (over 50%) that the interface was better with the toolbars "on". Therefore the decision was made to make them available with instructions on how to turn them off.

5. With regard to the navigational aspects of the system, in addition to the compasses added after the initial evaluation, it was suggested that a 2D map of the model would be useful. In this case, the map was not added as it was seen to be too obstructive to the user's view.

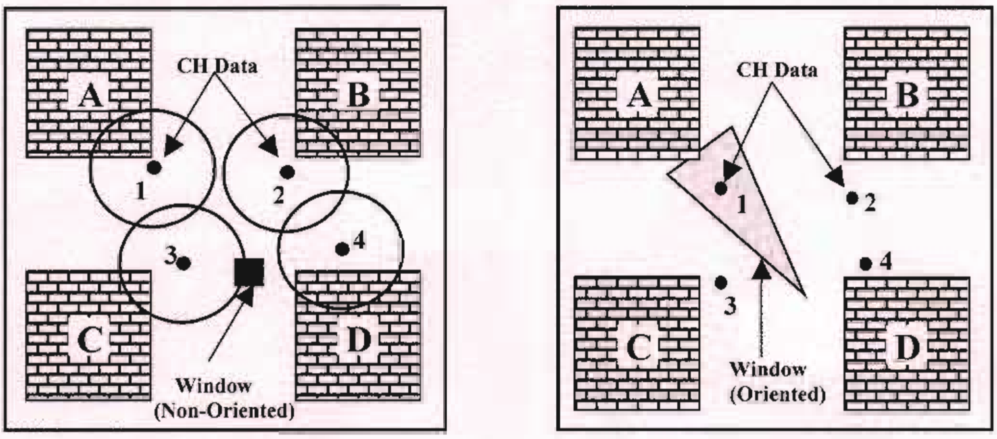
6. Another suggestion was to provide a logout button to the application. In the main application page it was pointed out that there was no exit.

These proposals were implemented prior to any other evaluation procedures being performed. This test yielded some very valuable qualitative results for the system.

In addition, during the course of the user evaluations, it was observed that younger users were more confident when it came to using the system. A similar problem is also identified by [Dalgarno and Scott, 1999] in a study of the usability problems. This strengthened the need for technical help and documentation to try and eliminate the fear of technology.

5.2.2.2 Multimedia Data Relevance

This section is a description of tests that were carried out to evaluate the relevance of the data displayed in the RTD. Tests were performed to investigate the accuracy of the results of the spatial queries, in particular the accuracy of the data returned by the directional queries. A simple experiment (Figure 5.5) illustrates the advantage of using a view-port directional query approach in comparison to the classical range query strategy.



(a) *(b)*

Figure 5.5: Comparison of Query Methods

Figure 5.5(a) illustrates how the *CH database* was organised in the ITD implementation of the *CHI system*. The example shows how the data was queried using a range query. Objects A, B, C and D represent street building blocks in the database. The query window represents the position of the user as he/she moves through the model. A problem with this method is as follows: if the query window is situated in data area '3' and the orientation of the user is northwest, the returned data is that relating data point '3', when the actual data should be that relating to data point '1'. Similarly, if the user is facing east the returned data is that relating to data point '3' when it should be that relating to data point '4'.

The improved query approach developed as part of the RTD is illustrated in Figure 5.5(b). In this example, the query window is an oriented, dynamically generated triangle based on the user's position. In addition to the buffer around the CH data point, the new query can be used to maximise the accuracy of view-port interactions by querying exact CH artefact and monument positions as illustrated in Figure 5.5(b). If the user's viewpoint is situated in the same position as the previous example and is facing northwest, the data that is returned is that relating to data point '1' together with Building A metadata (if any), while an orientation change to the east results in data relating to data point '4' and building D is obtained.

Another investigation considered how accurate the queries are when querying for height combined with horizontal intersection. One user "Exploring Dublin City Centre" performed this test for 20 minutes in search of CH data comprised of images, text and video. The SQL query tested for data that was contained within the view-port window and had a height of less than 20m and greater than 10m. The results of this test were acceptable and showed that data could be queried using height also. This addition to the system means that layers of data can be added to the database with the same X and Y coordinates but relating to different Z coordinate levels to distinguish it from data that represents, for example, information about different floors of a building.

This concludes the Internal Evaluation of the RTD. This evaluation helped identify many problems with the system before the external evaluations were carried out. All updates to the system were made before potential end-users tested it.

5.3 External Evaluation

The objective of the External Evaluation was to evaluate the *Revised Technology Demonstrator* using individuals outside the research group. This includes an *External Technical Evaluation* and an *External Usability Evaluation*.

5.3.1 External Technical Evaluation

This part of the evaluation was carried out by a smell number of potential end-users of the system who had specialist knowledge and who were given exclusive access to all system documentation and code. These experts were asked to review and comment on the overall architecture of the system and give a professional opinion on the design and performance of the system on the basis of the following criteria:

5.3.1.1 System Architecture

The architecture of the system is a typical three-tier web-based architecture. This type of architecture simplified the development and increased the scalability of the system by separating the three main functions of the application. Since the evaluators made no significant negative comments about the choice of system architecture it was assumed to be satisfactory.

5.3.1.2 Robustness

The robustness of the system refers to the completness of the system in terms of how easy it would be to completely install the system in another location. This was an aspect of the system that had been neglected. During this phase, it was pointed out by the evaluators that there were a number of problems with the system in this respect. In particular, there were many absolute references made to files and directories throughout the system that would cause problems in installing it. It was suggested as a solution that an installation procedure be developed that would allow the system to be installed semi-automatically.

5.3.2 External Usability Evaluation

This part of the system evaluation was undertaken by a group of potential end-users who had minimal relevant technical knowledge.

5.3.2.1 Peer Reviews

The purpose of the peer reviews was to ensure that the quality and utility of the system was adequate for end-users and to obtain comments from the evaluators by means of open discussion rather than responses to a set premature structured questions. This approach would subsequently aid the process of designing an effective questionnaire. The first step was to provide the evaluators with the relevant information about the *Revised Technology Demonstrator* before the discussions began. This was achieved by delivering a PowerPoint presentation followed by a live demo of the system.

The peer review process was performed from two different perspectives. The first peer review was conducted with the funding agency, Enterprise Ireland (EI). This peer review was held on Monday the 17th of November 2003 in the Digital Media Centre, Dublin Institute of Technology. This group was comprised of Dr. Carol Gibbins, EI, Dr. Joseph Curtis, EI and Mr. John Fagan, EI. The main aim of this peer review was to present the application to the funding agency and to gain as much feedback as possible. The second peer review held on 25th of November 2003, also took place in the Digital Media Centre, Dublin Institute of Technology. On this occasion the group of members were historians, Mr. Lorcan Collins, author of the book "Easter Rising 1916" and Mr Conor Kostic. The main aim of this presentation was to gain relevant feedback about the application, this time from a tourism perspective. In an attempt to acquire as much information as possible from each peer review, the discussions lasted in excess of one hour each.

In the first peer review, the prototype was checked against the list of outlined objectives. In this case, the project was successful in all aspects. Some useful suggestions were obtained from the second meeting in relation to the uses of the system and the accuracy of the information. It was suggested that the simulated system would be a useful tool to allow tourists to identify and plan city routes before arrival or before they leave their hotel

5.3.2.2 Evaluation Questionnaire

The second part of the *External Usability Evaluation*, the *User Form* evaluation and system demonstration evaluation, took place in the Digital Media Centre of the Dublin Institute of Technology. It involved the participation of both direct users of the project as well as those not directly affiliated with it. Both groups participated in testing the *CHI system* by using it in realistic situation as part of the Evaluation.

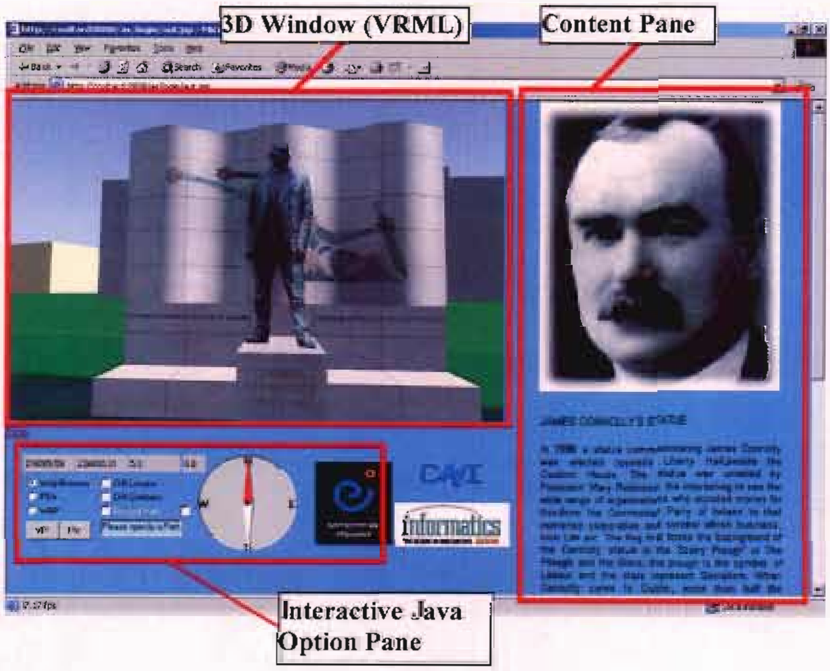


Figure 5.6: Revised Technology Demonstrator – Client view

The first part of the online evaluation form consists of questions with regard to the user interface design and the human-computer interaction. The detailed methodology is outlined in [CHID7.1, 2003].

The interface is used to simulate the users "walk" through the virtual Dublin. The visitor uses the mouse or the keyboard to navigate, while the position is continuously registered and used as an identifier to query the cultural heritage database. The initial graphical detail of the 3D scene is reduced to visually significant objects to indicate the city structure. At present the CHI 3D prototype reference implementation allows for dynamic construction of building blocks, parks and vegetation, rivers and lakes, and a street network on demand (The street network was developed but was not included in the final implementation).

In addition, the simulator introduces an easy-to-use approach to include external higher detailed 3D models. This enables the CHI 3D prototype to keep to a minimum the amount of data to be transferred from the server to the client and prepares for future extension of higher visual detail or other demands, like temporal characteristics or information displays, of the city model if requested. External models can be loaded via the Internet protocol as static data structures or dynamic object generation processes with the help of VRML Level Of Detail (LOD) switch Nodes. To emphasize the potential of the *Revised Technology Demonstrator* implementation, the system uses dynamic object generation processes for loading visually higher detailed scene elements.

5.3.2.3 Questionnaire Design

The questionnaire was comprised of two main sections, the interface design questions and the interface navigation questions. Broadly speaking, two main types of questions are used in questionnaires [Oppenheim, 1992]. 'Open' or free response questions are questions that require the user to specify an answer freely usually by using text and 'closed' questions are questions that offer the user a choice of alternative replies where the user is required to pick the most suitable choice. The majority of the questions in both modules are 'closed' questions. This was mainly because it made quantification and manipulation of the data easier. At the end of each section an 'open' text area is available, to allow the users the freedom to add comments about the particular module. The purpose of the questions in each module is described in the following, and the questionnaire itself can be found in Appendix C.

5.3.2.3.1 Interface Module Questionnaire

The interface module questionnaire was used to assess the effectiveness of and ease of access to each part of the system. The questions were designed and used to quantify the ease-of-use and graphic attractiveness of the 2D web-based aspects of the RTD. This module used a Linkert Scale[2] of 3 to rate the questions.

5.3.2.3.2 3D Navigation Module Questionnaire

The interface navigation questionnaire was used to assess the functional elements of the 3D navigation interface i.e. Pan, Fly, Zoom, Readability and Linkability. Initially, a yes/no question was used to enquire if the functions were used. If so, a Linkert Scale[2] of 3 was used to rate them.

5.3.2.4 Evaluation Form Results

The CHI research team supervised the evaluation and asked the following user groups to conduct the evaluation during the *Revised Technology Demonstrator* phase of the project. These evaluation groups were composed of students, researchers, specialists and project directors.

[2] A Linkert Scale is a numbered scale with verbal anchors at either end. In this case, for example, there is a three-point scale, where 3 represents excellent and 1 represents poor.

5.3.2.4.1 User group: Students

This group consists of current MPhil or PhD students, at the Digital Media Centre. 20 students filled in the online evaluation form. They were asked to test the CHI Homepage, VRML Navigation, and Hypermedia Navigation Components. In addition to the online questionnaire, some of the students were interviewed on-site to get first hand information from users. All records of participants were made anonymous and gave good insight into the projected behaviour of a general public audience.

5.3.2.4.2 User group: Researchers

This user group consisted of research staff at DMC sites not directly related to the project. They were responsible for testing the CHI interfaces in the evaluation and were involved in testing of the CHI Homepage, VRML Navigation and Hypermedia Navigation components. The number of participating researchers was 5.

5.3.2.4.3 User group: Specialists

Invited external experts in multimedia were asked to give a professional view of the system. Three specialists participated and these were particulary helpful in site interviews, were they provided significant feedback on the system components from an expert perspective.

5.3.2.4.4 User group: Project Directors

This group represents the CHI research team responsible for the successful implementation of the overall project. They evaluated the CHI Home page, VRML Navigation, Hypermedia Navigation, Client Layer, Server Layer, and Database Layer. The purpose of this evaluation was to verify that the *CHI system* adheres to the functionality outlined in the *Technical and Functional Specification* [CHID3.1, 2001] and the *User Requirements* [CHID2.1, 2001].

The research team completed the online questionnaires after having used the CHI sub-systems.

5.3.2.5 Evaluation Results Analysis

The purpose of this section is to analyse the results obtained from the online questionnaire and interview sessions.

5.3.2.5.1 Sub-system Evaluation: CHI Interface Design

The CHI Interface evaluation focused on functionality and consequently on elements of system access and interfacing. The objective was to ensure ease of user access and to achieve successful user interface interaction performance overall.

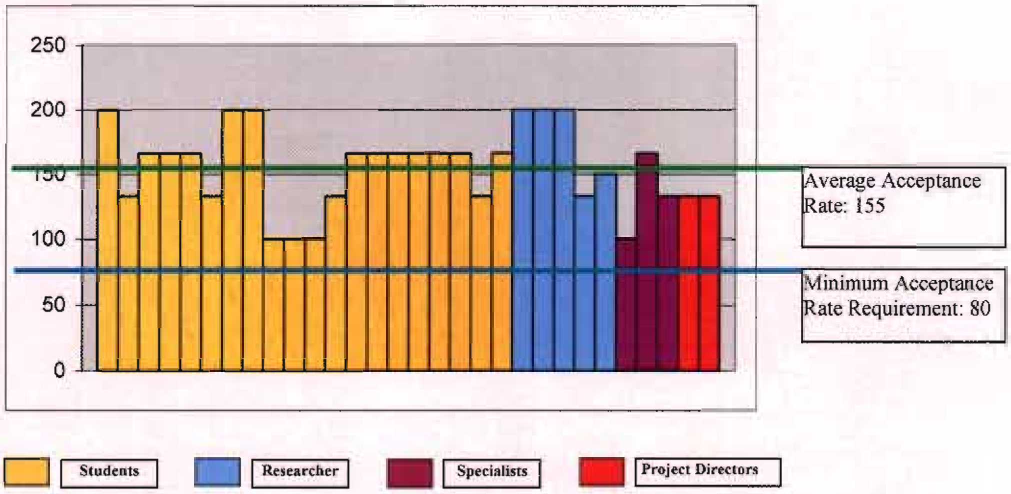


Figure 5.7: CHI Interface Design Results Analysis for all users

The research team decided on a minimum user acceptance rating requirement of 80 (See Figure 5.7), a figure which was exceeded by all participants. Amongst the individual evaluation groups, *Students* and *Researchers* demonstrated the highest levels of average acceptance (with scores of 159.6 and 163.2 respectivly) wheras the, possibly more demanding, *Specialist* and

Project Director groups were somewhat more critical (both recording scores of 133)

5.3.2.5.2 Sub-system Evaluation: CHI Interface Navigation

The principal objectives in relation to the evaluation of the VRML Navigation Sub-system were to analyse the acceptability of Pan, Zoom and Fly-through functions as well as the readability, linkability of dynamic and static web documents. Reliability, linkability and ease-of-use are of particular importance. Initially, navigation in an unfamiliar 3D environment is a confusing experience for most users and this issue requires particular attention in order to use the *CHI system* to its full potential. Navigation in virtual environments is not predefined and very much conditioned to the users experience with 3D computer graphics [Ousland and Turcato, 1999].

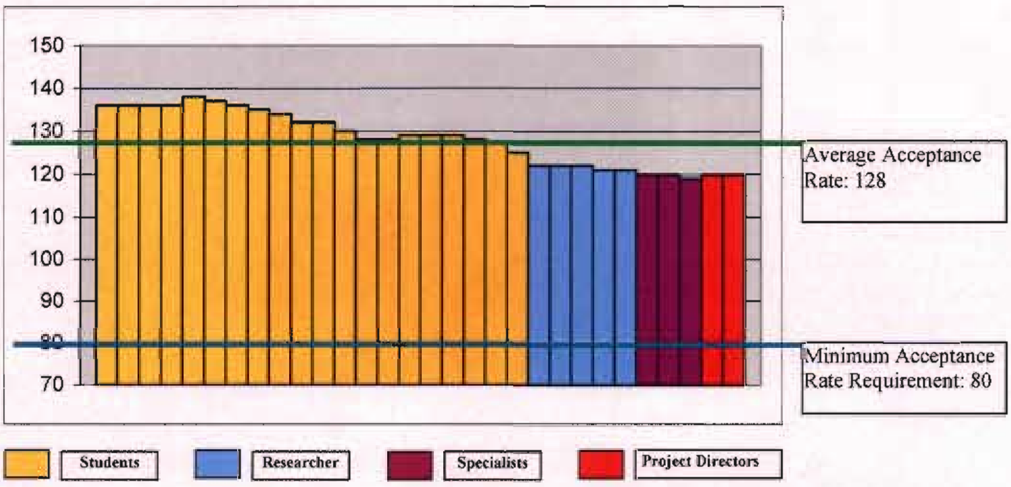


Figure: 5.8: CHI VRML Navigation Results Analysis for all users

The results of the CHI Interface Navigation evaluation (See Figure 5.8) indicate an overall acceptance rate of 128, which significantly exceeds the minimum target figure of 80. As in the previous test, undergraduate *Students* provided the highest acceptance score (131.95). There was significant rating agreement among the older, more experienced, and possibly more critical, groups, with members already familiar with the difficulties of 3D navigation, which comprised postgraduate *Researchers* (122), *Project Directors* (120) and Specialists (119.7) The emergence of this distinction is clearly an interesting and potentially significant finding.

5.4 Scalability Evaluation

The objective of the Scalability Evaluation was to evaluate the potential for (upward) scalability in relation to the *Revised Technology Demonstrator* and to identify any salient systematic issues that would arise by scaling the system. Five researchers in the Digital Media Centre undertook this evaluation.

This evaluation involved looking at the overall design of the system to determine any critical system components that would be an issue if the system were scaled up i.e. to handle a large number of simultaneous users.

5.4.1 Scalability Issues

A number of issues were identified by this evaluation. These issues are described in the following:

1. **Performance** - the most obvious scalability issue that arose was the speed and memory of the server machine. To work well under pressure, the 1.4 Ghz Intel Pentium 3 machine with 256MB memory specification is not suitable for this application.

2. **Interfaces** – the idea of a standardised interface to the project is a major a major issue in regard to enabling other content sources to be used by the system. There is the need for a standard set of interfaces that enable other content sources to be seamlessly utilised by the system.

3. **Web-Based Content Loading** – the current implementation of the system uses an application to load content data into the database. There is a need for a secure web-based application that enables data to be loaded into the database from any location via the web.

5.5 Conclusion

The *CHI project* investigated the applicability of these technologies for context-aware mobile computing applications. The project takes advantage of metadata-standards to enable user and device adapted services in the field of *Tourism* and *Cultural Heritage (CH)* data management and presentation to be utilised.

The results of *CHI project* evaluation lead to the conclusion that the initial requirements and technical specification have been met and that the system has been successfully implemented. Furthermore, the user survey made it possible to anonymously test the CHI interface and interacting components and revealed valuable opinions and comments on the overall *CHI system* design and performance from an independent group of people. The required acceptance rate of 80 was met for all users of the system and shows the overall satisfaction of the users with the *CHI system*. In future evaluations, it is appropriate to extend the range of questions to investigate in more detail any key discrepancies between user expectations and their experience of the system performance. The implementation in a three-tier architecture, as recommended in the technical specification has been shown to be a very successful way of developing the *CHI system* and enabled developers to work simultaneously on different aspects of the implementation. The software development documentation and complexity are two issues that need particular attention in such an environment.

Chapter Six

Conclusions and Recommendations

6 Conclusions and Recommendations

6.1 Introduction

This Chapter presents the conclusions of the research carried out on the development of the *Cultural Heritages Interfaces* (CHI) project. These conclusions relate to the internal, external and scalability evaluation of the Revised Technology Demonstrator (RTD). As a result of the expressed views and perceived overall attitude of respondents to the use of the prototype, it is apparent that the project has significant potential as a cultural portal for tourism or as a test framework for location-based services.

6.2 Conclusions

On the basis of research outcomes and experience gained during the course of this development project a number of conclusions may be drawn regarding the use of VR and Spatial Databases technologies together for the delivery of Cultural Heritage content over the World Wide Web:

1. The integration of VR and Spatial Databases can provide a valuable tool for the promotion of tourism and cultural heritage. However, it is clear that there remain a number of technological and other factors that currently limit the likelihood of successful adoption.

2. On the basis of related research findings, it is clear that, given the significant extent of the data download volume requirements to support practical, mobile system operation, access to an efficient and effective broadband service is an essential determinant of future successful system adoption. However, in the absence of such transmission guarantee, it is likely that, even in simulated (transmission) form, implementations of the project concept would be capable of providing potential or current tourists with a useful way to experience a novel and

informative view of Dublin (or potentially any other target location), typically within a hotel environment.

3. The use of VR to provide a user interface represents a relatively new approach to the provision of a *Virtual Desktop Environment* (VDE). Currently, the majority of computer users can be considered to be unfamiliar with 3D navigation and this unfamiliarity gives rise to system adoption hesitancy.

4. A primary research objective of the project was to integrate Virtual Reality (VR) with Spatial Database (SD) technology. The computer graphics algorithms used in VR are very efficient in their domain, however there is a need for further research to determine how best to integrate these algorithms into spatial database systems. For example, querying a 3D space with a 3D object is a common task in computer graphics. To perform the same query using spatial databases is, currently, not possible; most currently available spatial databases have the ability to store 3D data, but are unable to query it in 3D.

5. A number of issues remain to be resolved in order to extend system use to the area of mobile communications. While the use of potentially suitable mobile user interface devices such as *Personal Digital Assistants* (PDAs) is increasing, the cost of using such devices for volume data transfer remains prohibitive. In addition, in order to function correctly, the system requires user location data to be provided by individual *Global Positioning Systems* (GPS) receivers. Currently, each user would minimally require a PDA, a SIM pack (for mobile network access) and a GPS reveiver. Clearly, future mobile systems will be dependent on manufacturer's integration of these functions.

6. Finally, research findings highlight a need for the addition of directional relations in spatial databases. There has been some research carried out but as of yet there is no standard definition.

6.3 Recommendations

In the light of these research findings and conclusions, a number of recommendations for future work, relating to future system development, are now made.

6.3.1 CHI II

CHI II is a proposed second phase of the present project, the principal objective being to introduce the project system concept onto the streets. In order to operate effectively the user would be required to have a PDA with GPS mobile network connectivity. As the user actually progresses through the streets of Dublin, CH content would be delivered to the PDA.

The system framework is ideal for building and testing spatial applications. Using this framework, developers can build deploy and test spatial applications without having the overheads of mobile network costs and time-consuming field research. The 3D model acts as the spatial interaction component and the simulated mobile devices display any information necessary for testing purposes.

The application technology developed as part of the project could be used commercially in many other information domains. The fully mobile implementation could be used to deliver a host of different services such as stick-e-notes. The stick-e-notes concept allows system users to post e-notes wherever they want by electronically attaching the note to a specific location. Any other person using the system will receive the e-note message as they interact with this tagged location. This system could be used to locate friends in busy locations or leave messages for them.

6.4 Summary

Further research and development should be undertaken in relation to an extension of the *CHI project*, based on the knowledge gained from the process of development and the favourable results obtained from the RTD system evaluation. The conclusions and recommendations presented in this Chapter briefly summarise the project results and outcome.

The *CHI project* may be considered to provide a potential framework for the low-cost development and testing of mobile applications. The adoption and use of the core technologies represented in the project is becoming more widespread and, with the insight gained in the course of application development, the members of the associated research team gained expert knowledge in the area and intend to expand this knowledge by extending the scope of development effort to addressing the issue of mobile applications.

References

References

[1] Abel, D. J. and Smith, J. L.: A data structure and algorithm based on a linear key for a rectangle retrieval problem. Computer Vision, Graphics and Image Processing, 24(1):1-13, 1983.

[2] Aboulnaga, A. and Aref, W. G.: Window Query Processing in Linear Quadtrees. Distributed and Parallel Databases 10(2): 111-126 :2001.

[3] Ames, A. L., Nadeau, D. R., and Moreland, J.L.: "VRML 2.0 Sourcebook": 1997: John Wiley and Sons, Inc. pp. 519-531

[4] Apache Tomcat Homepage: 2004: URL: http://jakarta.apache.org/tomcat/index.html

[5] Aref, W. G., and Samet, H., "An Algorithm for Perspective Viewing of Objects Represented by Octrees", Computer Graphics Forum, 14(1), 1995, pp. 59--66.

[6] Athur, K.W.: "Effects of Field of View on Performance with Head-Mounted Displays": PhD thesis, University of North Carolina, 2000.

[7] Atzeni, P., Ceri, S., Parabosci, S. and Torlone, R.: "Database Systems": McGraw-Hill Publishing Company: 1999.

[8] Bales, D.: "Java Programming with Oracle JDBC": O'Reilly Press: 2002.

[9] Bancilhon, F. and Buneman, P.: "Advances in Database Programming Languages": ACM Press: 1990.

[10] Barfield, W., Hendrix, C., Bjorneseth, O., Kaczmarek, K. A., and Lotens, W.: "Comparison of human sensory capabilities with technical specifications of virtual environment equipment," Presence: Teleoperators and Virtual Environments, 1995.

[11] Bayer, R. and McCreight, E.: "Organization and Maintenance of Large Ordered Indexes," Acta Informatica 1 (3): 173-189, 1972.

[12] Beckmann, N., Kriegel, H.P., Schneider, R. and Seeger, B.: The R* tree: an efficient and robust index method for points and rectangles. In Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '90), pages 322-331, Atlantic City, NJ, USA, May 1990.

[13] Bertolotto, M., Carswell, J. D., McGeown, L., and McMahon, J., "iSmart+iSIS: Deploying Integrated Web-Based Spatial Applications Within an Oracle Database Environment", International Workshop on Web Geographical Information Systems (WGIS2001), IEEE CS Press, Kyoto, Japan. 2001.

[14] Bowon, K. and Kyhyun, U.: 2D+ String: A Spatial Metadata to Reason Topological and Directional Relationships. SSDBM 1999.

[15] Brinkhoff, T., Kriegel, H.P. and Schneider, R.: "Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems": In Proceedings of the Ninth International Conference on Data Engineering, pages 40-49, 1993b.

[16] Brinkhoff, T., Kriegel, H.P. and Seeger, B.: "Efficient processing of spatial joins using R-Trees" In Proceedings of the ACM-SIGMOD, 1993a.

[17] Brinkhoff, T., Kriegel, H.P., Schneider, R. and Seeger, B.: "Multi-Step Processing of Spatial Joins": SIGMOD Conference: 197-208: 1994.

[18] Burrough, P.A.; and McDonnell, R.A.: "Principles of Geographic Information Systems": 1998: Oxford University Press, NY.

[19] Carswell, J. D., Eustace, A., Gardiner, K., Kilfeather, E., and Neumann, M.: "An Environment for Mobile Context-Based Hypermedia Retrieval." DEXA Workshops 2002: 532-536.

[20] Carswell, J. D., Neumann, M., Kilfeather, E., Redfern, T., Eustace, A. and Gardiner, K.: "Technical and Functional Specification": 2001: Digital Media Centre.

[21] Chamberlin, D. and Boyce, R.: "SEQUEL: A structured English query language": In Proc. 1976 ACM SIGFIDET workshop on Data description, access and control, pp. 249-264: 1976.

[22] Chamberlin, D. D., Astrahan, M. M., Eswaran, K. P., Griffiths, P. P., Lorie, R. A., Mehl, J. W., Reisner, P., Wade, B.W.: SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control. IBM Journal of Research and Development 20(6): 560-575 : 1976.

[23] Chang, S. K.: Principles of Pictorial Information Systems Design. Prentice Hall Intern. Editions, 1989.

[24] Chung, C-W., McCloskey, K. E.: "Access to Indexed Hierarchical Databases Using a Relational Query Language". IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 1, February 1993.

[25] Codd, E. F., "A Relational Model of Data for Large Shared Data Banks": 1970: CACM 13(6): 377-387

[26] Codd, E. F.: "Does your DBMS run by the rules?" : 21st October 1985b, 49-64.

[27] Codd, E. F.: "Is your DBMS really relational?" : 14th October 1985a, 1-9.

[28] Collins, L. and Kostick, C.: "The Easter Rising: A Guide to Dublin in 1916": O'Brien Press: 2000.

[29] Comer, D.: "The Ubiquitous B-Tree". ACM Computing Surveys, 11(2):121--128, June 1979.

[30] Connolly, T. and Begg, C.: "Database Systems": 1998: Addison-Wesley

[31] Dalgarno, B. and Scott, J.: "Usability Problems in Desktop Virtual Environments": In J. Scott & B. Dalgarno (eds) Interfaces for the Global Community, proceedings of the 1999 conference of the Computer Human Interaction Special Interest Group of the Ergonomics Society of Australia (OzCHI), Wagga Wagga: Charles Sturt University. 1999.

[32] Dalgarno, B.: "The Potential of 3D Virtual Learning Environments: A Constructivist Analysis" Electronic Journal of Instructional Science and Technology, 5(2): 2002.

[33] Delis, V., Hadzilacos, Th., Tryfona, N.: 1994: "An Introduction To Layer Algebra", in "Advances in GIS Research", by Waugh and Healey (eds.), Taylor and Francis

[34] Diehl, S.: Distributed virtual worlds: foundations and implementation techniques using VRML, Java, and CORBA. Berlin. New York: Springer, 2001.

[35] Egenhofer, M. and Franzosa, R.: "Point-Set Topological Spatial Relations": International Journal of Geographical Information Systems, 5(2): 161-174: 1991.

[36] Egenhofer, M. J.; Herring, J. R.: Categorizing Binary Topological Relationships Between Regions, Lines, and Points in Geographic Databases. Technical report, Department of Surveying Engineering, University of Maine, Orono, ME, 1991.

[37] Egenhofer, M., and Herring, J.:"A Mathematical Framework for the Definition of Topological Relationships," Fourth International Symposium on Spatial Data Handling, K. Brassel and H. Kishimoto, eds., Zurich, Switzerland, 1990.

[38] Egenhofer, M., Sharma, J. and Mark, D.:"A Critical Comparison of the 4-Intersection and 9-Intersection Models for Spatial Relations: Formal Analysis" Autocarto 11, R. McMaster and M. Armstrong, eds., Minneapolis, MN, 1993, pp. 1-11.

[39] Egenhofer, M.: Spatial SQL: A query and presentation language. IEEE Transactions on Knowledge and Data Engineering, 6:86-95, 1994.

[40] Eisenberg, A. and Melton, J., SQL: 1999, formerly known as SQL3, SIGMOD Record, 28 (1), March 1999.

[41] Elmasri, R. and Navathe, S. B.: "Fundamentals of Database Systems": 2d Ed.: Benjamin/Cummings, Redwood City, CA, 1994.

[42] ESRI, Inc. URL: www.esri.com/software/arcgis/arcinfo/index.html: 2003

[43] Ezeigbo, C.U.: "Spatial Data Model For Regional Integration in Africa": 2001: International Conference on Spatial Information for Sustainable Development, Nairobi, Kenya

[44] Finkel, R.A., and Bentley, J.L.: Quad-trees; a data structure for retrieval on composite keys, Acta Informatica 4 : 1974.

[45] Foley, J. D., van Dam, A., Feiner, S. K. and Hughes, J. F.: "Computer Graphics: Principles and Practice": Second Edition: Addison-Wesley: 1990.

[46] Gaede, V., Gunther, O. Multidimensional Access Methods. ACM Computing Surveys, 30(2), 1998.

[47] Gardiner, K., Carswell J. D., "Viewer-Based Directional Querying for Mobile Applications", Third International Workshop on Web and Wireless Geographical Information Systems W2GIS, Roma, Italy 13th December 2003.

[48] Gargantini, I.: An Effective Way to Represent Quadtrees. CACM 25(12): 905-910 (1982)

[49] Geringer, D.: "Oracle9i: Spatial" Student Guide, Volume 1: 2001a.

[50] Geringer, D.: "Oracle9i: Spatial" Student Guide, Volume 2: 2001b.

[51] Goyal, K. G., and Egenhofer, M. J.: "Similarity of Cardinal Diractions" in: Jensen, C., Schneider, B. S., Seeger. B., Tsotras, V. (eds.): Seventh International Symposium on Spatial and Temporal Databases Lecture notes in Computer Science Vol. 2121, Springer-Verlag, pp. 36-55, July 2001

[52] Guting, R.H.: "An Introduction to Spatial Database Systems": VLDB, 3:357-399: 1994.

[53] Guttman A.: "R-trees: a dynamic index structure for spatial searching", In Proc. ACM SIGMOD Int. Conf. on Management of Data, Boston, MA, 1984.

[54] Hinze, A.; Voisard, A.: "Location- and Time-based Information Delivery in Tourism". In: Hadzilacos, T., Manolopoulos, Y., Roddick, J. F., Theodoridis, Y. (Eds.): "Advances in Spatial and Temporal Databases". 8th International Symposium, SSTD 2003. Springer Verlag, Series: Lecture Notes in Computer Science. Vol. 2750. Santorini Island, Greece, July 24 - 27, 2003.

[55] Hunter, J. and Crawford, W.: "Java Servlet Programming": O' Reilly: 2001.

[56] Jackins, C., Tanimoto, S. L.: "Oct-Trees and Their Use in Representing Three-Dimensional Objects," CGIP, 14(3), November 1980, 249-270.

[57] Kamel, I. and Faloutsos, C.: "On Packing R-trees": CIKM: 1993.

[58] Kelty, L., Beckett, P., and Zalcman, L.: "Desktop Simulation in Advancing Simulation Technology and Training", Proceedings of SimTecT99, The Simulation Technology and Training Conference, Simulation Industry Association of Australia: 1999.

[59] Kilfeather, E., "Hypertext, Narrative and Coherence: The role of the reader and writer in the practice of hypertext": 1998, M. Phil Thesis. DIT. http://www.dmc.dit.ie/eoin/interview/Hypertext_and_Text61.pdf

[60] Kilfeather, E.: "CHI User Requirements Document": 2001: Digital Media Centre, DIT.

[61] Kriegel H.-P., Brinkhoff T., Schneider R.: "Efficient Spatial Query Processing in Geographic Database Systems", IEEE Bulletin on Data Engineering, Vol. 16, No. 3, 1993, 10-15.

[62] Kroenke, D. M.: "Database Concepts": Prentice Hall: 2003.

[63] Larijani, L. Casey.: The Virtual Reality Primer, McGraw-Hill, New York, 1994.

[64] Leach, G., Al-Qainmari, G., Grieve, M., Jinks, N., McKay, C.: "Elements of a Three-Dimensional Graphical User Interface" In S. Howard, J. Hammond, and G. Lindgaard (eds) *Human Computer Interaction, INTERACT'97* London: Chapman and Hall: 1997.

[65] Li, J., Jing, N., and Sun, M.: "Spatial Database Techniques Oriented to Visualisation in 3D GIS." School of Electronic Science and Engineering, National University of DefenceTechnology http://www.digitalearth.ca/ html_papers/DE_A_064.htm

[66] Liu, X., Shekhar, S., and Chawla, S.: "Object-based Directional Query Processing in Spatial Databases" IEEE Transactions of Knowledge and Data Engineering: 2003.

[67] Liu, X., Shekhar, S., and Chawla, S.: "Processing Object-orientation-based Direction Queries" in the proceeding of the Eighth International Symposium on Advances in Geographic Information Systems, 69-76, Washington D.C., November 2000.

[68] Liu, X.: "Modeling and Processing Directional Queries in Spatial Databases": Ph.D. thesis, University of Minnesota, August 2000.

[69] Lui, John C.S., Chan, M.F., So, K.Y., Tam, T.S.: "Dynamic Partitioning for a Distributed Virtual Environment": Third High Performance Computing Asia Conference, 23 – 25: September, 1998.

[70] Mc Atamney, H.: "If You Build It, Will They Come? – An Investigation Into The Use Of The Virtual Reality Modelling Language As A Tool To Promote Tourism On The Internet"": Mphil Thesis: Dublin Institute of Technology: 2004.

[71] McCarthy, M., Descartes, A.: "Reality Architecture: Building 3D worlds with Java and VRML": Prentice Hall Europe: 1998.

[72] Melton, J. and Eisenberg, A.: SQL Multimedia and Application Packages (SQL/MM). SIGMOD Record 30(4): 97-102 (2001).

[73] Mishra, S. and Beaulieu, A.: "Mastering Oracle SQL": 2002: O'Reilly Press.

[74] Monson-Haefel, R.: "Enterprise Java Beans": 2001: O'Reilly.

[75] Morton, G. M.: A computer oriented geodetic data base and a new technique in file sequencing. Technical report, IBM Ltd., Ottawa, Ontario, Canada, 1966.

[76] Neumann, M.: "Spatially Navigating the Semantic Web for User Adapted Presentations of Cultural Heritage Information in Mobile Environments": First International Workshop on Semantic Web and Databases". Berlin 2003

[77] Nielsen, J.: "Designing Web Usability": New Riders Publishing: 2000.

[78] Nielsen, J.: "Usability Engineering": Boston Academic Press: 1993.

[79] OGIS: "Open GIS consortium: Open GIS simple features specification for SQL (Revision 1.1)": 1999: URL: http://www.opengis.org/techno/specs.htm

[80] Ooi, B. C., Sacks-Davis, R., Han, J.: "Indexing in Spatial Databases" (Make Corrections)

[81] Oosterom, P.V. and Claassen, E.: "Orientation Insensitive Indexing Methods for Geometric Objects" Proc. 4th Int'l Symp. on Spatial Data Handling, Zürich, Switzerland, 1990.

[82] Oppenheim, A.N.: "Questionaire design, interview and attitude measurement": Printer Publishers: 1992.

[83] Oracle 9iAS White Paper: URL: http://otn.oracle.com/products/ias/pdf/9iasr2newfeature.pdf: 2002.

[84] Oracle Spatial Java Library User's Guide: http://otn.oracle.com/docs/products/spatial/pdf/sdoapi.pdf: 2003.

[85] Oracle Spatial User's Guide and Reference: 2001: URL: http://download-west.oracle.com/otndoc/oracle9i/901_doc/appdev.901/a88805/toc.htm

[86] Oracle9i XML Developer's Kit Guide: 2003: URL:

[87] Orenstein, J.A.: Spatial query processing in an object-oriented database system. In SIGMOD, Washington, D.C., 1986.

[88] Ousland, A. R. and Turcato, H.: "Navigating 3D Environments": JUPITER2 - Joint Usability, Performability and Interoperability Trials in Europe. EURESCOM. 1999.

[89] Papadias, D., Egenhofer, M. J., Sharma, J.: Hierarchical Reasoning about Direction Relations. ACM-GIS 1996.

[90] Papadias, D., Theodoridis, Y., and Sellis, T.: The Retrieval of Direction Relations Using R-trees. Proc. 5th Int'l Conference on Database and Expert Systems Applications, DEXA'94, Athens, Greece, Springer - Verlag LNCS #856, September 1994.

[91] Papadias, D., Theodoridis, Y., Sellis, T. And Egenhofer, M.: "Topological Relations in the World of Minimum Bounding Rectangles: a Study with R-trees": *SIGMOD Conference*, San Jose, CA, 1995.

[92] Papadopoulos, A., Rigaux, P., and Scholl, M.: "A performance evaluation of spatial join processing strategies": In SSD, pages 286-307, 1999.

[93] Patel, J. M. and DeWitt, D. J.: "Partition based spatial-merge join": In Proc. ACM SIGMOD Conf. on Management of Data, 259 - 270, Montreal, Canada, 1996.

[94] Pequet, D. and Ci-Xiang, Z.: "An Algorithm to Determine the Directional Relationships between Arbitrarily Shaped Polygons in the Plane": Pattern Recognition, Vol. 20, No. 1, pp.65-74: 1987.

[95] Pfoser, D., Jensen, C. S., and Theodoridis, Y.: "Novel Approaches in Query Processing of Moving Objects." In proc. Intl. Conf. on Very Large Databases (VLDB), 2000.

[96] Ravi, K. V. K., Ravada, S., Abugov, D.: "Quadtree and R-tree Indexes in Oracle Spatial: A Comparison using GIS Data": Proceedings of the 2002 ACM SIGMOD international conference on Management of data: 2002.

[97] Rigaux, P., Scholl, M., Voisard, A.: "Spatial Databases": 2002: Academic Press, Morgan Kaufmann Publishers.

[98] Robertson, G.G., Card, S.K. and MacKinlay, J.D.: "Nonimmersive Virtual Reality": Computer, 26(2), 81-83: 1993.

[99] Robinson, J.T.: The K-D-B-Tree: A Search Structure For Large Multidimensional Dynamic Indexes. SIGMOD Conference 1981.

[100] Roussopoulos, N. and Leifker, D.: Direct Spatial Search on Pictorial Databases Using Packed R-Trees. SIGMOD Conference 1985.

[101] Roussopoulos, N., Kelley, S., and Vincent, F.: "Nearest neighbour queries": In Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, ACM Press. 1995.

[102] Safar, M. and Shahabi, C.: 2D Topological and Direction Relations in the World of Minimum Bounding Circles. IDEAS 1999.

[103] Samet, H. "Application of Spatial Data Structures". Addison-Wesley, 1990.

[104] Samet, H. and Webber, R. E.: "Storing a Collection of Polygons Using Quadtrees": ACM Transactions on Graphics, Vol. 4, No. 3, 1985.

[105] Samet, H., Aref, W. G.: "Spatial Data Models and Query Processing": Modern Database Systems: 338-360: 1995a.

[106] Samet, H., Webber, R. E.: "Storing a Collection of Polygons Using Quadtrees": ACM Transactions on Graphics 4(3): 182-222: 1985.

[107] Samet, H.: "Spatial Data Structures": Modern Database Systems: The Object Model, Interoperability, and Beyond: 361-385: 1995b.

[108] Schiller, J.: "Mobile Communications": Addison Wesley Pub Co.: 2000.

[109] Schnieder, M.: "Spatial Data Types for Database Systems": Springer Press: 1997.

[110] Selinger, P., Astrahan, M., Chamberlin, D., Lorie, R.A., and Price, T.G.: "Access Path Selection in a Relational Database Management System": In Proc. ACM SIGMOD Conf. On Management of Data, Boston, MA: 1979.

[111] Sellis, T., Roussopoulos, N. and Faloutsos, C. "The R+-tree: A dynamic index for multidimensional objects", Proceedings of the 13th International Conference on Very Large Databases, pp507-518, 1987

[112] Sharma, J.: "Oracle Spatial." An Oracle technical white paper, Oracle Corp., Redwood City, CA, May 2001.

[113] Shekhar, S. and Chawla, S.: "Spatial Databases: A Tour": Prentice Hall Publishers: 2003.

[114] Shekhar, S., and Liu, X.: "Direction as A Spatial Object: A Summary of Results": In the proceeding of the sixth International Symposium on

Advances in Geographic Information Systems, 69-75, Maryland, ACM, 1998.

[115] Shekhar, S., Chawla, S., Ravada, S., Fetterer, A., X. Lui, and Lu, C.T: “Spatial Databases: Accomplishments and Research Needs”: IEEE Trans. on Knowledge and Data Engineering: 1999(a).

[116] Shekhar, S., Coyle, M., Goyal, B., Lui, D-R., Sarker, S.: “Data Models in Geographic Information Systems”: CACM, 40(4), 103-111:1997.

[117] Shekhar, S., Liu, X., and Chawla, S.: “An Object Model of Direction And Its Implications”: GeoInformatica, 3(4), 357-379, Kluwer Academic Publishers: 1999(b).

[118] Stonebraker, M., Rowe, L. A.: The Design of Postgres. SIGMOD Conference, 340-355: 1986.

[119] Suharto, S., Man Vu, D.: “Computerized cartographic work for censuses and surveys”: Paper prepared for The Expert Group Meeting on Innovative Techniques for Population Censuses and Large Scale Demographic Surveys Netherlands Interdisciplinary Demographic Institute (NIDI) The Hague: 22-26 April, 1996.

[120] Theodoridis, Y., Papadias, D., and Stefanakis, E.: “Supporting Direction Relations in Spatial Database Systems.” Proc. 7th Int'l Symposium on Spatial Data Handling, SDH'96, Delft, The Netherlands, August 1996.

[121] Theodoridis, Y., Papadias, D., Stefanakis, E., Sellis, T.: “Direction Relations and Two-Dimensional Range Queries: Optimisation Techniques”: Data and Knowledge Engineering, Vol. 27, No. 3, pp. 313-336,1998.

[122] Vince, J: “Virtual Reality Systems”: Addison-Wesley: 1995

[123] VRML Consortium: http://www.vrml.org/

[124] Worboys, M.F.: “GIS: A computing perspective”: 1995: Taylor & Francis Ltd, London

[125] Worsley, J. and Drake, J.: “Practical PostgresSQL”: 2001: O’Reilly.

[126] Yu, C. and Meng, W.: Principles of database query processing for advanced applications. San Francisco, CA, USA: Morgan Kaufmann, 1998.

[127] Yu, C., Meng, W.: “Confronting Database Complexities”: 2002

Appendices on CD