Interactive Browsing of 3D Scenes in Hypermedia: The Hyper-G 3D Viewer

Diplomarbeit in Technischer Mathematik

Michael Pichler

Technische Universität Graz
Institut für Informationsverarbeitung
und Computergestützte neue Medien (IICM)

Fertigstellung:	6. Oktober 1993
Prüfungsfach:	Computerwissenschaft
Betreuer:	Dipl.-Ing. Keith Andrews
Begutachter:	O. Univ.-Prof. Dr. Dr. h.c. Hermann A. Maurer

Ich versichere, diese Arbeit selbständig verfaßt, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfsmittel bedient zu haben.

Die Diplomarbeit ist in englischer Sprache verfaßt.

Acknowledgements

I would like to thank everybody who has offered suggestions and support during writing of this thesis.
Especially I wish to thank a number of people at the IICM for their support during my work and preparation of this thesis:
Dipl.-Ing. Keith Andrews for his many suggestions and bringing a lot of ideas in discussions with him.
Dr. Frank M. Kappe for his help in finding literature.
Univ.-Prof. Dr. Hermann A. Maurer.
Last but not least I would like to thank all test persons who have "played" with various stages of the implementation for many suggestions and improvements.

Abstract

This thesis discusses methods for interactive navigation through virtual 3D scenes in a hypermedia system. Several methods for manipulating the scene and navigating through it have been developed as well as alternatives for highlighting pickable objects (information links). The implemented 3D viewer has been embedded into the Hyper-G hypmermedia system. Program evaluation was done with heuristic evaluation and usability tests.

Contents

1. Introduction **9**

2. Hypermedia **11**

3. Interactive Browsing of 3D Scenes **15**

3.1. Classification....15
3.2. Requirements....16
3.3. Input Devices....16
3.4. Flip Metaphor....17
3.5. Walk Metaphor....19
3.6. Fly Metaphor....20
3.7. Point of Interest-Movement....20
3.8. Links....21

4. Hyper-G **23**

4.1. Client-Server Architecture....23
4.2. Special Features....24
4.3. Document Types....28
4.4. Applications....29

5. The Hyper-G 3D Viewer **31**

5.1. About the Implementation....31
5.2. User Interface....32
5.3. Navigation....33
5.3.1. Flip Object....33
5.3.2. Walk....35
5.3.3. Fly....36
5.3.4. Fly to....38
5.3.5. Heads-up....38
5.4. Anchors....41
5.4.1. Bounding Cubes....42
5.4.2. Brightness....43
5.4.3. Colour Code....43
5.4.4. Colour Edges....43
5.5. Scene Description....44
5.6. Possible Applications....44

6. Selected Details of the Implementation **45**

6.1. Modularity....45
6.2. Portability....46
6.3. Embedding in Hyper-G....47
6.4. The Scene....48
6.5. 3D Objects....50

6.6. Camera Model 51
6.7. Light Sources 52
6.8. Geometric Objects 52
6.9. Polyhedra 53
6.10. Picking 3D Objects 54
6.11. Materials 57
6.12. Movement 57
6.13. Point of Interest Movement 57
6.14. Anchor Highlighting 59
6.15. Colour Chooser 60

7. Usability Evaluation **61**

7.1. Test Environment 61
7.2. Heuristic Evaluation 62
7.3. Navigation Modes 63
7.3.1. Heuristic Evaluation of Navigation Modes 63
7.3.2. Usability Tests of Navigation Modes 65
7.3.3. "Flip Object" 66
7.3.4. "Walk" 66
7.3.5. "Fly" 67
7.3.6. "Fly to" 68
7.3.7. "Heads-up" 69
7.3.8. Free Choice of Mode 70
7.3.9. Comparison of Modes 71
7.4. Anchor Highlighting 76
7.4.1. Heuristic Evaluation of Anchor Highlighting Modes 76
7.4.2. Usability Tests of Anchor Highlighting Modes 77
7.4.3. General Results 78
7.4.4. "Bounding Cubes" 78
7.4.5. "Brightness" 79
7.4.6. "Colour Code" 80
7.4.7. "Colour Edges" 81
7.4.8. Comparison of Modes 82
7.5. General Results 87

8. Conclusions **89**

App. A) Colour Plates **91**

App. B) GE3D "Graphics Engine 3D" **103**

B.1 Type Definitions 104
B.2 Functions 104
B.2.1 Opening the Graphics Device 104
B.2.2 Display Control 105
B.2.3 Drawing Modes and Attributes 105
B.2.4 The Transformation Matrix Stack 106
B.2.5 Building Transformation Matrices 109
B.2.6 Text 109
B.2.7 Line Primitives 109
B.2.8 Solid Primitives 111

B.2.9 Camera Definition....113
B.2.10 Light Sources....115
B.2.11 Closing the Graphics Device....115

App. C) SDF File Format **117**

C.1 Actor File (ACT)....118
C.2 Position File (POS)....119
C.3 Camera File (CAM)....119
C.4 Light File (LGT)....120
C.5 Material File (MTL)....120
C.6 Object File (OBJ)....121
C.7 Scene Description File (SDF)....122

Bibliography **129**

1. Introduction

As computer performance has increased, **3D programs** have become more and more widespread. The reason is that we live in a three dimensional world and are our senses are naturally accustomed to 3D space.

Currently 3D software is used mainly for two purposes: CAD (computer aided design) and computer animation. *Computer animation* programs create highly realistic pictures, but rendering takes a long time (several minutes per picture) and movements are predetermined. *CAD* programs, used by technical engineers and architects, allow real-time movement and changes to the scene objects, but are in general difficult to use and reserved to specialists.

This thesis discusses a viewer for 3D scenes with the following two properties:

- interactive navigation
- easy interface

Interactive navigation means that drawings are made in real-time and users are able to move the scene or change their view of it. This should give the impression of a "real" three-dimensional scene.

The *interface* has to be simple enough, such that also novice users should have no difficulties navigating through 3D space after a short introduction. On the other hand the program should also appeal to experienced users.

Several methods for *navigation* (movement) through 3D scenes - as easy and natural as possible - are discussed. The wide range of potential 3D models and the diversity of users is thereby considered. The most effective ones were implemented, allowing the user to choose which one to use, depending on the purpose. The term *navigation* can be understood in two ways: more widely taken it means any kind of movement or 3D manipulation, more strictly taken it means changing the view point in 3D space and excludes *object* manipulation. Where it is not clear by context it is specified which term is meant explicitly.

Current hardware needed to draw even relatively simple scenes in real time (at least about 20 to 25 frames per second) is still expensive, but prices are falling and the rendered drawings become more and more realistic.

Hypermedia systems are a way for organising large amounts of heterogeneous data in an easily accessible form. The document types supported in hypermedia usually include text, raster images, 2D drawings, digital audio and video, and sometimes animations.

The defining characteristic of hypermedia systems are the *links* or connections between documents. Such links can lead to documents of other types or to more detailed information, e.g. to a picture illustrating a text. The endpoints of links are called *anchors*. When an anchor is activated (e.g. by a mouse click) the corresponding link is activated.

Hyper-G is a large-scale, distributed hypermedia system currently being developed at the IICM in Graz. Hyper-G also supports many other features, like the clustering of documents, automatic consistency checks, a built-in conferencing system and advanced searching facilities.

This thesis takes the approach of establishing *3D scenes* as document type in the Hyper-G hypermedia system. Given a model of a 3D object the user can examine it from all sides, zoom in for details, and so on. Further, 3D environments can be watched, giving the user a feeling of being present in the scene, a kind of "virtual reality".

For example, in a text document about Graz there might be a link to a raster image of the clock tower (the most famous landmark of Graz). This picture is of high quality, but is still only a flat picture. So there might be an additional link to a 3D model of the clock tower. When the user turns it around and detects that it has a bay on only three out of four corners, this may be a more enlightening experience than reading it in a text.

Objects in the scene may serve as anchors for links to other hypermedia documents. To give a visual cue, which objects in the scene lead to information links, some alternatives for highlighting the anchors were developed.

The range of applications for the 3D viewer varies from technical illustrations (e.g. a model of a car), over models of real buildings (like a famous house) or environments (e.g. a city), up to artificial scenes (virtual realities).

In order to evaluate the various navigation methods and the anchor highlighting methods a number of 3D models were created. First, a *heuristic evaluation* was performed by some students, yielding to some improvements to the 3D viewer.

Particular attention was paid to a *usability test* to get to know which navigation metaphors and link highlights are preferred by novice and expert users and should be offered in a future implementation. Two studies of 32 test subjects were carried out for navigation techniques and anchor highlighting methods. Also some suggestions for further improvements or extensions of the current implementation were made.

2. Hypermedia

The term *hypermedia* is derived from the terms *hypertext* and *multimedia*. One definition of hypermedia can be given as "a system in which documents of various types or *media* are connected with *hyper*-links".

Hyper-text means non-linear text. *Links* exist as connections between documents. It is not necessary to read the documents (often also called nodes) in a predefined linear sequence. Instead when a link is activated the according document is opened. On-line help systems are a common example of hypertext. But also references in lexica, footnotes and literature references (e.g. [Kappe91]) can be seen as links.

Hyper-*media* documents are not restricted to texts. They may also be raster images (e.g. scanned pictures), audio data, 2D object-oriented pictures (e.g. from a drawing editor), video films, animations, and so on.

Hypermedia is influenced by many other research activities, and offers a wide range of research fields. Such related topics are information retrieval, computer aided instructions (CAI), human-computer-interaction, computer graphics, cognitive psychology, computer mediated communication, user interface design, and electronic publishing.

There are two different ways to combine documents of different media: either *frame-based* or *window-oriented*.

Frame-based hypermedia systems use *multimedia documents*, called frames. These are single documents consisting of several parts of information of different types, for example a picture on a sheet of text. The frames are designed and composed by the author including decoration, placement of texts, pictures, and buttons. This full control by the author can be seen as advantage, but is usually a disadvantage since not all authors are also good interface designers, and the frames are difficult to change. Standalone hypermedia systems like HyperCard or Toolbook use this model.

Window-oriented hypermedia systems use *document clusters* to form groups of related documents, but all document types stay separated in their own windows. The author connects the documents which belong together with so-called *cluster links*. This approach is taken by some larger hypermedia systems, like InterMedia, NoteCards, and also by Hyper-G (see colour plate 1 in App. A) as example Hyper-G session).

The window-oriented systems have several advantages. First of all, the hypermedia system is not bound to a particular *user interface* metaphor (like book, travel, library or desktop), because the user can arrange the windows freely. Further the single viewers can be implemented conveniently by different programmers.

Very important is the *extendibility* of the system. A new document type (e.g. 3D scenes) can be added to document clusters by implementing a new kind of *viewer* (or document manager) without changing the rest of the system.

Multilinguality can be supported in window-oriented systems. The same text (or voice) document can be stored in different languages and the system chooses the right one, when the user activates a link to it. Other documents like images are not affected by the selection of the language.

The endpoints of links are called *anchors*. The *source anchor* is the place in a document that has to be clicked to activate a link. Some hypermedia systems also support *destination anchors*. That means that a link can lead to special part of the destination document, for example to a paragraph in an article - when that link is activated the text is scrolled to that paragraph.

A hypermedia system must also provide *searching facilities* for quick information retrieval. This can range from a simple boolean combination of search strings up to a *full text search* including weighting of word occurrences and automatic synonym translation.

Problems of hypermedia can be the "lost in hyper-space" syndrome - users do not know where they are, how to find information known to exist, determining how much information exists about a specific topic and how much of it has already been seen.

A graphical browser can be integrated to show the users their current position. But the documents shown in the map must be selected and arranged carefully, otherwise the map becomes cluttered with links.

An easy way to give an overview are *collections*. Collections group together related documents and/or sub-collections. Thus they form a hierarchy, which can be described as a tree or more general as a directed acyclic graph, when allowing a collection to belong to several other collections.

Current large hypermedia projects are [Kappe93]:

- *WAIS* - Wide Area Information Server. WAIS is based on *full text search* for information retrieval. The information is grouped into a number of databases, and there is one main database which contains keywords of the other databases, such to find the relevant databases in which to search the desired information. There are no links connecting documents.

- *Gopher*. Gopher is based on the *menu* user interface metaphor. It contains no hypertext and is therefore not a real hypermedia system. But it allows key word search and provides a connection of remote databases. It is used for connecting Campus-Wide Information Systems.

- *World Wide Web* - W³. W³ allows the documents to be spread all over the world. The documents are connected with *links*.

- *Hyper-G*. Hyper-G offers a variety of information searching facilities: *links* for connecting documents, *collections* for grouping documents, and *full text search*. It also offers special features like *annotations*, automatic *indexing* and link-generation, a *messaging* and computer conferencing system, animations, *dialogues* and so on, which are discussed in more detail in Chapter 4.

All the above systems are based on the client server model. The servers manage the documents and the clients are document viewers. Hyper-G includes a gateway to information held by the WAIS, Gopher and WorldWideWeb-servers *without* changing the user interface. Likewise the clients of Gopher and WorldWideWeb can access data of Hyper-G over a gateway.

3. Interactive Browsing of 3D Scenes

Navigation in 3D space is complicated by the need to control (at least) six degrees of freedom at the same time: three dimensions for position and three dimensions for orientation. A seventh parameter, the angular field of view could also be considered.

3.1. Classification

3D viewpoint movement can be done basically in two different ways: either moving the *object* (the scene as a whole) or moving the *viewpoint* through the scene. Which one is appropriate depends on the kind of the 3D scene:

1. 3D models of *objects*, which the user wants to move, to look at them from all sides (i.e. turn them around); examples would be a car, an engine or any other technical illustration.

2. 3D models of *worlds* or environments, where the user wishes to feel present in the scene, to walk (or fly) around in the scene; for example a room, a house, or a city.

However, in each of these two groups there are many possibilities for defining a *metaphor*. A metaphor means how the user can understand the behaviour of the interface [Ware90], a kind of an internal model. The user may feel like translating and rotating the object on screen or like walking or flying through the virtual scene.

Another aspect that influences the choice of a movement metaphor is the task it is used for. Characteristic types of viewpoint movements in 3D space are [Mack90]:

- *General movement*. Exploration of the environment, like walking through a building.

- *Targeted movement*. Movement with respect to a specific target point, such as examining a detail in the scene or an object of interest.

- *Specified coordinate movement*. Movement to a precise position and/or orientation, such as specifying that the line of sight should be along the z-axis from point (0, 0, 20) towards the origin.

- *Specified trajectory movement*. Movement along a specified trajectory (curve in 3D space), like specifying a camera movement for an animation.

A technique appropriate for general movement may be inefficient for another type of movement, say targeted movement, and an additional kind of movement had to be offered for that purpose. In general, each metaphor makes efforts to simplify some movement tasks, whereas other actions become rather difficult.

3.2. Requirements

A good viewpoint moving technique should fulfil the following requirements [Mack90, Robert93]:

- easy to use, including easy to learn, to understand, and to remember
- easy prediction of system behaviour in response to the input
- preventing user disorientation or at least an easy way to reverse the movement
- integration with the whole user interface and workspace

Other requirements depend on the purpose of the program. For example, in a CAD application it is important to position objects at an exact coordinate position. Animation software will have to offer a possibility for a smooth camera movement along a specified path. Movement techniques used for viewing 3D scenes, like the Hyper-G 3D viewer, do not have such special requirements.

3.3. Input Devices

A thorough discussion of Input Devices can be found in [Foley90], [Hill90] and [Pimen93]. This section intends to give a short overview over the most important and most common input devices.

The *keyboard*, being the most basic input device, is not suitable for graphic manipulation. The user would have to remember many commands and only the cursor keys offer an association to movement. Nevertheless it can well be used for additional functions and shortcuts for expert users.

- **2D Input Devices**

The input device most often used for computer graphics is a *mouse*. A mouse has two degrees of freedom, horizontal and vertical movement.

The advantages of a mouse are: A mouse is used with almost every computer, most users are very familiar with using a mouse, and novice users learn its usage in a few minutes.

The main disadvantage is that the mouse is a two dimensional input device (although 3D mice have been constructed) and therefore the movement metaphor must build a relationship between 2D input space and 3D virtual space.

Important to mention are also *direct* 2D input devices, which are used directly on the screen. They include the *light pen* and *touch screen*. Although a touch screen may soon be full of finger prints and appears blurred, it can be operated by novice users even more easily than a mouse.

Other 2D input devices are *joy sticks*, *digitising tablets*, and the *trackball*. The trackball is like an upside-down mouse where the ball can be rotated in any direction. Essentially they do not differ from the mouse in the way they can be used as input device for 3D navigation.

- **3D Input Devices**

A 3D input device very similar to the trackball is the *spaceball*. The spaceball consists of a ball that the user can grasp with the hand. Like a trackball the device stays on the table, and only the ball is pushed. In contrast to the trackball the ball is not really rotated, instead the force applied to the ball is measured. Pushing the ball forwards/backwards or left/right and a torsion around its vertical axis gives three degrees of freedom.

Virtual Reality (VR) systems [Pimen93] use a variety of unconventional input devices. A *data glove* has sensors (usually optical sensors at fibre-optic cables) along the fingers to detect the current bending/flexing of the fingers. It can be used to recognise gestures. *Position trackers*, usually integrated in the data glove and the head mounted display (stereo output device) for tracking the *position* and the *orientation* of the hand and the head. So they are actually a 6D input device. *Eye trackers* can track the direction in which the looks - they are not widely in use.

The following chapters will concentrate on using a mouse as input device, because it is the most common (and an inexpensive) input device, and almost every computer user is familiar with using a mouse.

3.4. Flip Metaphor

This metaphor moves ("flips") the *object*, while the viewpoint remains constant.

The following movements are useful:

- translation,
- rotation,
- zooming.

Each of them can be seen in different ways. The first approach is based on the *coordinate axes*. Both translation and rotation are possible along each of the three axes, so the user has seven movement methods: translation along x, y, or z, rotation along x, y, or z, and zooming. Zooming means here scaling, i.e. to enlarge the object as a whole.

The choice of the axis for translation/rotation can be done explicitly be the user (e.g. by pressing a button) or implicitly by taking the one which is nearest to a horizontal or vertical line through the viewport. The latter alternative also allows to choose two axes at the same time for a 2D input device. (Such a coordinate based model is used by Wavefront's Advanced Visualizer [Wave91].)

The other possibility is to use axes parallel to a horizontal and vertical lines in the viewport as axes for translation and rotation. This is a very intuitive method and the user does not need to be aware of coordinate axes at all. The two translations (horizontally, vertically) together with zooming let the user position the object at any point in 3D space. This can be seen as translation in the camera coordinate system (see Section 6.6.).

Also rotation can be done in three directions: around a horizontal axis (rotate up/down), a vertical axis (rotate left/right), and the axis along the line of sight (tilting the object). The last one means to turn the object in a plane parallel to the view plane.

These three rotations can be mapped one-to-one to the possible rotations of a *spaceball*. But also when a 2D input device like a mouse is used, it is possible to simulate a spaceball [Chen88]. The idea is to think of three virtual sliders atop the window as can be seen in Figure 3.1.

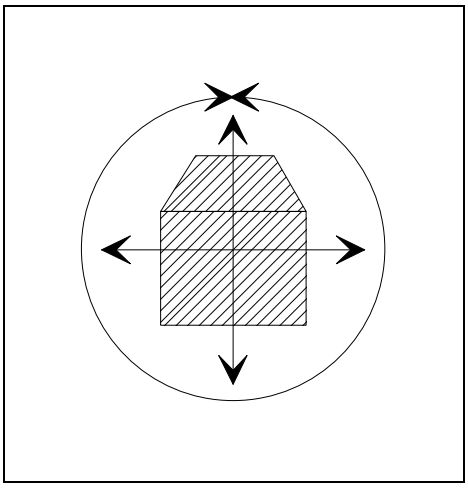

Figure 3.1: Virtual Sliders Atop an Object

Each mouse movement can be mapped onto a virtual sphere. Rotations near the window centre rotate the object left or right, up or down. Rotations near the window corner tilt the object around its centre.

3.5. Walk Metaphor

This metaphor simulates *walking* through 3D space. A similar model would be to drive a car. A movement of the mouse (or other input device) corresponds to a movement of the viewpoint in the scene.

The following movements are possible:

- walking forwards and backwards, and to the left and right side,
- turning the head for looking up and down, and to the left and right.

For more flexible movements this metaphor can be extended by

- movement straight upwards and downwards, like with a lift, or translation (panning).

For the realisation of this metaphor there are still some degrees of freedom. In "walking forwards ..." it has not been specified what *forwards* and backwards means. One possibility is to move the body always parallel to a ground plane (i.e. a plane y = const). This could be appropriate for "planar" scenes, where users rarely wishes to move upwards or downwards - and if so, they can use the *lift*-facility.

The other way is to "follow your nose", i.e. also considering the vertical orientation of the line of sight. This allows also to do a smooth walk along stairs. For movements that should remain on the same level (or height) an additional function for resetting the view horizontally can be offered ("level view").

A complete separation between looking direction and movement direction is not useful - for example when you look to the right side you keep walking in the previous direction unless you also turn your body. This would remove the correlation of the view and the resulting movement and require extra information to be displayed for orientation.

The walk metaphor is appropriate for smaller scenes. The user always has the feeling of controlling the system and any position in 3D space can be easily reached.

Some problems may be the movement over large distances, where the mouse has to be taken off the table, put back and movement has to be resumed, resulting in non continuous movement. Another disadvantage is that movements near objects are rather difficult. For example if the user wishes to look at a detail on a wall he/she will probably walk through the wall.

An improvement can be a non linear *motion function* for making movements over large distances faster, while small movements are fine and well to control.

3.6. Fly Metaphor

One disadvantage of "walking" over large distances is that the mouse has to be moved, taken off the table and moved again several times, which can become tiresome.

In the *fly* metaphor the user feels like sitting in an aeroplane. Input actions are only necessary to change the speed and/or direction of the flight. Even when the flight direction or speed is changed many times, the mouse (or other input device) does not have to be moved outside a small region around its original position.

The following parameters have to be controlled:

- speed: acceleration, deceleration
- direction of flight

In this metaphor the direction of movement will always be the current view direction - without a restriction to a ground plane. It seems plausible to walk through a room looking upwards, remaining at constant height, but the same situation would be unnatural for a flight.

A non-linear, over-proportional *acceleration function* is advantageous. It provides small steps as long the speed is low, allowing easy control of speed for detailed views. When the speed becomes higher, the steps also increase allowing to reach high velocity for movements over large distances.

The fly metaphor is appropriate for large-scale scenes, where it is tiresome to walk. On the other hand *fly* is harder to control than walk and the user may feel to have less control over the system. This can make it inappropriate for smaller scenes.

3.7. Point of Interest-Movement

Point of interest-movement [Robert93, Mack90], or targeted movement as it also can be called, is an approach for combining rapid movement towards a point of interest with controlled movement near the target object. Another advantage, common to the walk metaphor, is that the user need not look at any user interface component, neither real (like menus or displays atop the viewing area) nor virtual (like virtual sliders or scrollers).

First the point of interest (POI) is set by marking any visible point in the scene. The essence of POI-movement is the *logarithmic* moving function. It means in each time step the movement towards the POI is *not* constant (like in the walk metaphor), but proportional to the current distance between the viewpoint and the POI. This means the movement is fast

towards objects and slow near the surface of an object, and it is impossible to fly through faces.

This basic POI movement has the property that the marked point of interest during the movements towards it and away always projects to the same point on the window. Furthermore the target object appears to grow at a constant rate during approaching the viewpoint. This makes it very easy for the user to predict when the object will reach the desired size to look at it in detail.

Often it is also desirable to include an adjustment of the line of sight to be oriented to face the point of interest in a straight direction. For example to look at some text or picture on a wall. This is achieved by *orienting POI movement* [Robert93, Mack90].

The point of interest also gives a surface normal on the face it was hit. Additionally to the movement towards the target point, orienting POI movement includes a lateral movement of the viewpoint parallel to the view plane to put the POI near the centre of the viewport, and a rotation of the point looked at towards the point of interest.

Both movements should also be done logartithmically, but somewhat faster than the plain translation to ensure that the line of sight is near the POI-normal when the desired distance from the object is reached.

Point of interest movement as discussed here provides no means for performing rotations. If a turn is desired an appropriate oriented face must be found to perform the movement, which may be difficult or require an unnatural choice of the POI. Therefore, the point of interest movement is not suitable as a navigation metaphor in itself, but a useful addition to one of the other navigation metaphors, like walk or flip.

3.8. Links

Links, the connections between documents in Hypermedia (see Chapter 2.), can also be used as navigation tool, because a link can also lead to another part of logically the same "world". This is especially useful when the 3D environment is very large and consists of several independent parts.

For example consider a model of a building. When each room is finely modelled, drawing the entire scene would take a long time, resulting in slow, unsatisfying movements. However, its possible to design each room separately and to connect the rooms with links at the doors between them. This allows not only more detailed and realistic models of the rooms at satisfying drawing speed, it also relieves the user from tiresome walks from one room to another.

Another example for this metaphor would be a virtual city, where the user can walk through the streets and click on doors to walk into any building of interest.

4. Hyper-G

Hyper-G (the "G" stands for Graz) is the name of an ambitious hypermedia project currently under development at the Institute for Information Processing and Computer Supported New Media (IICM) at Graz University of Technology. Hyper-G is designed to be a *modular, large, general purpose, networked hypermedia system*. Indeed Hyper-G is going to be the major focus of the institute's work until about 1996.

4.1. Client-Server Architecture

Hyper-G is designed as a *distributed system*. This allows access to large amounts of data by many users simultaneously. The heart of the Hyper-G system, the *core system*, is divided into *clients* and *servers*. They are connected via a high speed local area network (LAN).

A *session manager* controls each Hyper-G session. The session manager starts *document viewers* to display documents and communicates via *ipc*. The session manager also controls link following (the various link types are discussed below). Further it offers on-line help and several navigation aids: an undo/redo function, navigation history, and a map (graph) of the current locality. Furthermore, a collection overview and guided tours (see below) are controlled by the session manager.

All database accesses (for the documents and the links) are done through *database managers*. There are several databases: for the documents themselves, for the links and anchors, for collections and tours. The concurrent server processes can run on different server machines to increase performance.

The core system is not isolated, but connected via a wide are network (WAN) to the rest of the world. This allows system access from remote terminals of different types, but is rather slow (typically 64 kbit/s).

The network also allows access to *remote databases* (e.g. time tables, phone books, satellite data). Thus Hyper-G provides access to millions of documents, which are as far as possible always up-to-date, and could not be kept in a single local system. These documents are like an integral part of the Hyper-G system: they can be *linked* to the local documents and are presented in the same user interface.

The document *viewers* exploit the local intelligence of end-user terminals as far as possible. They retrieve the data over the network from the servers, but then run locally on the terminal. This is much faster than a conventional log-in to a server which is slow and

inefficient: every single keystroke having to be transmitted over a network to the host. In the client-server model most work is done locally (e.g. scrolling a text), only when new data are necessary (e.g. activation of a link) the network and server are accessed.

In general, a number of different terminals are connected to the system. When a Hyper-G session is started, the proper viewer for that terminal type is invoked to take advantage of the terminal capabilities. For example, a VT100-like terminal is able to display text only, in a fixed font, on a window of fixed size, and cursor keys are the only way for scrolling text; images can not be displayed. In an X-window a proportional font can be used, parts of the text can be italic, bold or underlined, and scrolling can be done with a scrollbar using a mouse.

The different viewers can also be used to support different - also unorthodox - input/output devices.

4.2. Special Features

This section can only give a short overview of the most important features of the Hyper-G system. A detailed discussion can be found in [Kappe91].

- **Window Based Model - Clusters**

The Hyper-G system is easy extensible to support new document types or user interfaces. This is possible as Hyper-G is *window-oriented* - in contrast to so-called frame based hypermedia systems, based on multimedia documents (see Chapter 2.).

Each type of document is displayed in a separate window by a *viewer* for that document type independently of the other documents. Documents that belong together are grouped in so-called *document clusters*. This allows Hyper-G to be not tied to a specific *user interface metaphor* for arranging document sequences like book, travel, library or desktop. Instead the user can have the windows arranged as desired and can store the preferences.

When introducing a new viewer (e.g. for 3D scenes) the rest of the system does not have to be changed. The new viewer can be implemented by a person or group who does not have to know details of the implementation of the other viewers. Also each document viewer can use its own file format - there is no need to design a single document standard which covers all types of documents.

- **Identification Levels**

It is possible to use the Hyper-G system *anonymously*. Therefore users need not have the feeling of being watched by a "big brother" and the system can be used from public terminals or in a museum.

Several operations like writing annotations require prior *identification*. Also for storing preferences of users some identification is necessary. Hyper-G supports several levels of identification:

Fully identified users are both known to the system (including name and address) and to other users. They must login with a password.

Semi-identified users are still known to the system, but may choose a (unique) pseudonym and a password. Other users only get to know the pseudonym, but do not know the identity of the user. This mode is suitable for the communication facilities: the user is not recognised by the other users and a misuse of the system is prevented.

Anonymously identified users choose a pseudonym and a password, but their identity is neither known to the system nor to other users. These users are only allowed to make private annotations, which are not visible to other users. In this mode, user monitoring is not possible because the system does not know the user. The semi-anonymous mode still allows the system to remember the preferences of a user between several sessions.

Anonymous users do not need any identification or password. No annotations can be made and user preferences can not be stored.

Identified users may belong to a *group* with certain access rights.

- **Multilinguality**

The user can choose the desired language for the system. Not only the user interface components (like menus, dialogues, help texts) are changed to that language, also the *documents* are provided in the desired language. This is achieved by clusters (see above) of documents with the same contents in different languages.

For example if the user sets his/her language preferences to (1) German and (2) English (in that order) the system will look first for a German text, but will present the English one if no German text is available (or any other if neither one is available).

- **Associative Links**

Historically, most hypermedia systems have used uni-directional links. That means they can be followed only from the source to the destination. In Hyper-G links are *bi-directional*. That means that links can also be followed in the reverse direction (of course the links are still directed).

This backward (also called *associative*) link following can be used to ask which documents contain a link to the current document. For example when reading a text about Graz, the user may ask what other texts to refer it, leading to other texts of interest (for example about the province of Styria or Austria) to which no link may have been provided.

Bi-directional links are also of advantage for link maintenance (see below). When a document is deleted, it can be determined which other documents have a link to it, which then have to be erased.

Associative or bi-directional links require that the links are stored in a separate database and not within the documents.

- **Anchors**

Hyper-G supports both source and destination anchors. All hypermedia systems have source anchors - it is simply the name for the parts of the documents that can be clicked to activate a link to another document or cluster.

A *destination anchor* is a part of the target object which is of interest when following the link. This can be a paragraph in a text, a region of an image or a piece of a sound clip (or others). When the link is activated the user gets a view already targeted on the part of interest. For example a text scrolled to the paragraph of interest or a zoomed view of a map.

A destination anchor need not explicitly be specified. There is always a default anchor covering the whole document.

- **Dynamic Links, Link Maintenance**

To manage the large amount of data it is necessary to include facilities for *automatic link maintenance*. An example is the deletion of all links from and to a document when the document is deleted. Also deletion of outdated documents should be supported by the system. A further step is *automatic link generation*. When a document is edited the system can be asked to propose or automatically generate links to related documents.

Many automatically generated links even do not need to be stored. They can be generated at runtime and are called *dynamic links*. A hypermedia system will include some encyclopaedias and dictionaries. For almost any word in any document (except maybe technical terms) there will be an automatic link into the dictionary explaining that word. It would be confusing to highlight each word and useless to store all these links in a database. Instead the user may click also on non-highlighted words, initiating a dynamic link generation. This is realised by a database query at runtime.

These dynamic links reduce the amount of storage for static links and also reduce the effort involved for link maintenance: In the example above, words deleted from the dictionary are no longer found and words deleted from a document cannot be clicked.

It is also possible to have *associative dynamic links*, which means to find a dynamic link that would point to the current document. For example the user wishes to find all documents about a certain topic. This requires *full text search* in all documents and therefore special data structures (inverted index) are necessary.

- **Collections**

Collections are groups of related documents and/or sub-collections. They form a hierarchical structure. Collections help find information about a specific topic and keep an overview. Especially for novice users, following a hierarchy makes it easier to keep orientation in large amounts of data.

In Hyper-G collections need not be trees: a collection may be a sub-collection of more than one other collection. This results in a directed acyclic graph. For example a collection Biochemistry will belong both to collection Biology and Chemistry, which are sub-collections of Natural Sciences.

As any document in Hyper-G belongs to at least one collection, it is possible to show the users their current position in the collection hierarchy at any time, and allow searching for more general or more specialised documents about a specific topic.

- **Guided Tours and CAI**

A "Guided Tours" is a path (sequence of links) through several documents. The path may or may not be linear - all facilities of hypermedia can be used. The documents are somehow related, but are independent of the collection hierarchy.

Potentially every (identified) user may create a tour (e.g. "my favourite impressions of Australia" or "amazing physical phenomena").

An important application of tours are lessons, or *computer aided instruction* (CAI). Such lessons with links to other lessons and well-designed logical structure are usually authored by experts.

- **Annotations**

Any user (unless anonymous) may become an author by writing *annotations*. Annotations can concern documents and/or links. Either the user writes a new document and links it to existing ones or the user creates a link between existing documents (by specifying source and destination anchor).

The newly created documents may be of any type (there is no special document type annotation). The author of the annotation can be determined by other users. Annotations can be made private, visible to a user group or visible for all.

There are no special editors integrated in Hyper-G. The documents are edited using existing software packages. Converters are available to convert the documents of diverse file formats into the formats supported by Hyper-G.

- **Communication**

One way to communicate in Hyper-G is by annotations (see above). Other users have access to one's own opinion.

The Hyper-G system also includes a build-in messaging system (electronic mail) and a *computer conferencing* system. This computer conferencing system can be used for all kinds of discussions, ordered by a hierarchy of topics. Of course, there can be links to and from conferences to other hypermedia documents. To avoid a misuse of the conferencing system the authors must be at least semi-identified, which does not mean that other users know their real name (see above). Several topics can be only accessible for certain user groups.

4.3. Document Types

A paradigm of hypermedia is to allow access to heterogeneous types of information. Hyper-G is an open system allowing addition of new document types. Currently Viewers are being developed for the following document types [Kappe91]:

- text
- 2D raster images, like digitised pictures
- 2D object oriented drawings, usually used for technical drawings or illustrations
- sound (digital) and speech (synthesised)
- digital movie clips
- maps for orientation
- electronic mail and computer conferencing
- courseware, computer aided lessons (CAI)
- dialogues

... and the *3D viewer* discussed in this thesis.

The incorporation of 3D scenes in hypermedia systems is unusual - to our knowledge Hyper-G is the first hypermedia system to integrate 3D documents. Chapter 5. discusses the Hyper-G 3D viewer in detail.

4.4. Applications

Hyper-G is designed to be usable by general, untrained users as well as expert users. Hyper-G should be adaptive to the various needs of different user groups to allow a wide range of possible applications.

The main application area of Hyper-G will be *information* and *communication systems* for larger institutions, universities and companies.

The current version of Hyper-G forms the basis for the *university information system* University of Technology Graz. It is accessible by all students and members of the university. A wide variety of different users on different hardware are using the system to obtain information about the various institutes and departments, for administrative purposes, as a tool for computer aided instruction, and as a medium for computer supported communication.

The European Space Agency (ESA) will offer the "Genius GDS" service based on Hyper-G from the beginning of 1994. "Genius GDS" stands for "Global Environmental Network Information and User System: Guide and Directory Service" and will give a large number of people access to information about satellites and the obtained data of the earth (mainly images) over Internet. As the information is not hierarchical and contains many potential cross references (user manuals, contact addresses, textual descriptions of image series, satellites, and sensors) a hypermedia system offers many advantages over a conventional database.

A number of smaller *spin-off applications* are surfacing which are mainly pursued by IMMIS - the Institute for Multimedial Information Systems of the Joanneum Research [Kappe93].

One such by-product of Hyper-G was an electronic presentation of Austria at the Expo'92 in Seville, Spain. Here, visitors are presented with a map of Austria ("*AMAPA*"), can zoom in into any region of interest to obtain a detailed map, and can scroll that map arbitrarily. By clicking some of over 1500 icons, they can look at texts, pictures and video clips from all over Austria. All information, including the video clips, is stored in digital form. An improved version of the program was in use at the Expo'93 in Tae Jon, South-Korea.

5. The Hyper-G 3D Viewer

The Hyper-G 3D Viewer enables the integration of 3D scenes as Hypermedia documents and provides interactive navigation facilities. Some objects in the scene can be designated as *anchors*. These objects are highlighted and can be picked to activate a link. Also destination anchors are possible: when arriving at a 3D scene, the camera is positioned as desired.

5.1. About the Implementation

The Hyper-G 3D viewer was implemented using C++ [Strous91] on a Silicon Graphics Personal Iris 4D/35 under the UNIX operating system [Bourne82] with the X 11 window system [Reilly88, Scheif86]. The user interface was built with the InterViews tool kit [Linton89, Linton87], release 3.1.

Three-dimensional graphical output is done with the device independent *GE3D* graphics library - see App. B) - which was implemented in C [Kern88] for GL (Silicon Graphics' native graphics library).

The integration into Hyper-G, concerning communication between viewers and the *session manager*, is discussed in detail in Section 6.3. The communication is done via IPC (inter process communication): the viewer is started as demon process, waiting for activation from the session manager; when active, messages can also be sent in the other direction. The *look and feel* issues of the viewer are separated from the data maintenance of the viewer.

The *InterViews* tool kit offers classes for menus, buttons, labels, boxes, input handlers and so on. They are all abstracted from the underlying window system (like X windows). It is very easy to combine these elements to create a user interface and experiment with different arrangements.

InterViews implements three kinds of UI *"look and feel"*: Motif, SGI Motif, and Open Look. The desired one can be chosen via command line. InterViews also manages the different types of X windows *(visuals)*. Graphic workstations we use support pseudo-colour and true-colour visuals.

Like any other X application, it is possible to define style values (X attributes) outside the program. The program specifies default values for font, cursor colour etc., which can be overridden in the command line or loaded from a configuration file, where the users can specify their preferences. This technique is well supported by InterViews and allows a consistent look and feel for all Hyper-G applications.

5.2. User Interface

The *viewing area* is embedded into the application window (see the colour plates in App. A). The window can be resized without distorting of the view (the aspect ratio of the scene is preserved). At the top there is a menu bar with pull-down menus and below the viewing area there is a row of buttons. The button "show anchors" at the window bottom (see Figure 5.1) is used to switch anchor highlighting on and off (see Section 5.4.).

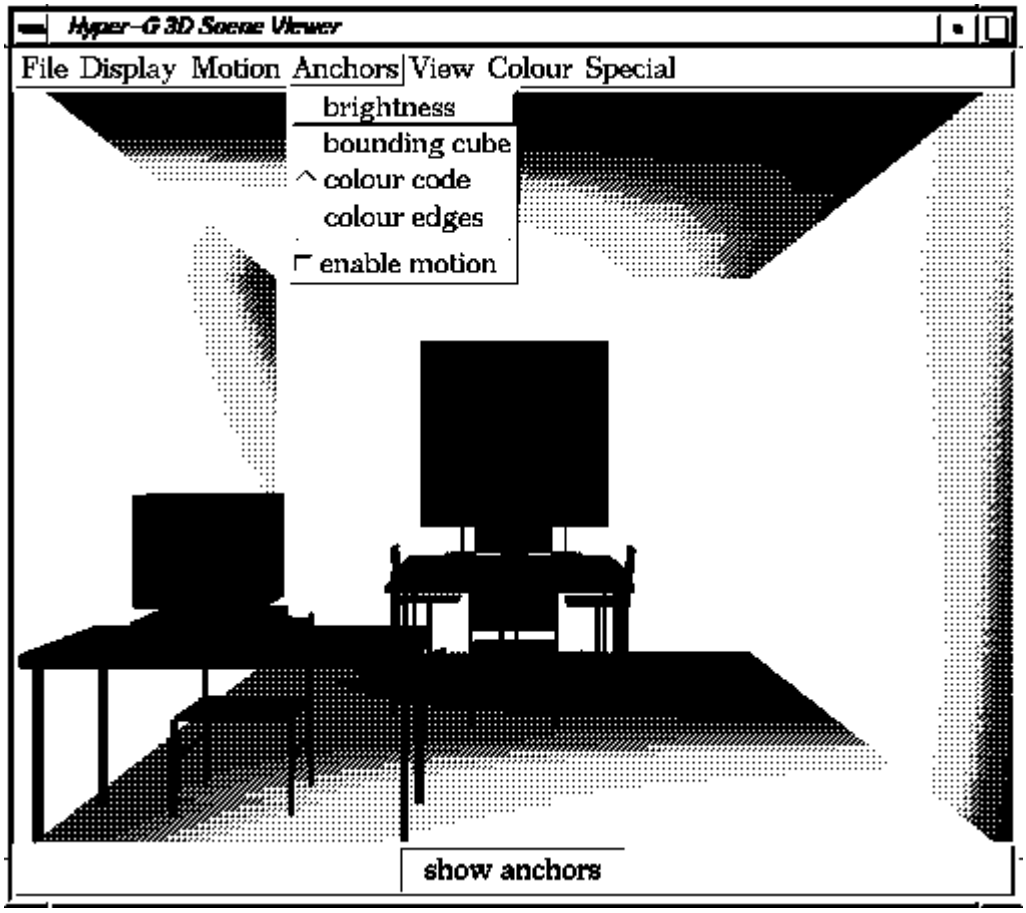

Figure 5.1: Office Scene

Several commands are activated with pull-down menus. The include selection of the display mode, the movement (navigation) metaphor, and the anchor highlighting method. Further the colours for background and UI components can be selected and changed at run time via a colour chooser dialogue.

The pointing device used for navigation input is a three-button mouse as it is available on all computer systems. Some commands need a modifier key on the keyboard to be pressed (SHIFT, CONTROL).

5.3. Navigation

The Hyper-G 3D viewer offers five navigation modes: *flip object*, *walk*, *fly*, *fly to*, and *heads-up*. When *navigation* is more strictly taken it means a movement of the view point, which is achieved by the latter four modes. In contrast, *flip object* is a technique for object movement. The desired motion mode can be selected from a pull down-menu with radio-menu items (see colour plates 2 to 4 in App. A).

Additionally there are two menu items for controlling the *view*:

- *reset view* restores the default viewpoint with which the scene was loaded. Useful when the user totally looses the orientation.

- *level view* rotates the view such that the line of sight is horizontal. This function can be used after looking up and down to continue movement horizontally.

5.3.1. Flip Object

As its name suggests, *flip object* moves the object, while the point of view remains unchanged. The interface can be seen from Figure 5.2 (or colour plate 2 in App. A).