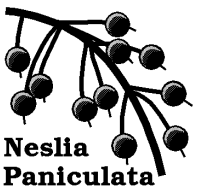
3D Character Centred Online Editing Modalities for VRML-based Virtual Environments

Ph.D. dissertation committee:
prof. dr. ir. A. Nijholt (promotor, supervisor)
prof. dr. J. Zwiers
prof. dr. J.J. van Wijk
prof. dr. J.C. Roberts
dr. Zs. Ruttkay
prof. dr. F.M.G. de Jong
prof. dr. ir. W.H.M. Zijm (chairman)



Taaluitgeverij Neslia Paniculata
Uitgeverij voor Lezers en Schrijvers
van Talige Boeken
Nieuwe Schoolweg 28, 7514 CG Enschede,
The Netherlands

CTIT Ph.D. Thesis series No. 03-51
Center for Telematics and Information Technology
P.O. Box 217, 7500 AE Enschede, The Netherlands

ISBN 90-75296-07-X

CTIT Ph.D. Thesis No. 03-51. ISSN 1381-3617

Enschede, The Netherlands.

The cover design shows a virtual environment populated by entities created with the editing systems described in this thesis.

Printed by: PrintPartners Ipskamp, Enschede.

3D CHARACTER CENTRED ONLINE EDITING MODALITIES FOR VRML-BASED VIRTUAL ENVIRONMENTS

DISSERTATION

to obtain
the doctor's degree at the University of Twente,
on the authority of the rector magnificus,
prof. dr. F.A. van Vught,
on account of the decision of the graduation committee,
to be publicly defended
on Wednesday June 18, at 15:00

by

Szilárd Kiss
born on December 11, 1974
in Kolozsvár, Romania.

This dissertation is approved by the promotor,
prof. dr. ir. A. Nijholt.

It is nice to know that the computer understands the problem. But I would like to understand it too.

Eugene Wigner (1902-1995)

Do your work for six years; but in the seventh, go into solitude or among strangers, so that the memory of your friends does not hinder you from being what you have become.

Leo Szilárd (1898-1964)

Acknowledgements

First of all I must thank Anton Nijholt for the opportunity to work in the Parlevink research group. He has been my promotor, supervisor and source of inspiration during the past few years. Without his guidance and support, none of this work would have been possible. Besides fruitful discussions, he encouraged me to visit and experience professional gatherings, conferences, all of them turning out to be invaluable sources of information.

I am grateful to the persons with whom I worked and who supported my work: Hendri Hondorp (also for being the best roommate), Job Zwiers (for always looking at the deep ends of things) and the former participants of the "graphics group meeting".

Many thanks goes to the conscious proofreaders of this thesis: Peter Asfeld and Mária Péter for a thorough analysis, Peter Asfeld, Hendri Hondorp, Anton Nijholt and Job Zwiers for the regular discussions and the valuable feedback.

I am very honoured to have Jack (Jarke J.) van Wijk, Jonathan Roberts, Zsófia Ruttkay, Franciska de Jong and dean Henk Zijm to complete the defence committee.

I would also like to greet my esteemed colleagues and friends whom made the stay at the university educational, eye-opening and fun. It is a privilege knowing all of you. The secretaries Charlotte Bijron and Alice Vissers are entitled to a special thanks for taking care of all things that needed to be taken care of.

Last but not least, I would like to thank my family. They supported me, even from far away. And thanks to my loving wife for keeping her promise of feeding me in this busy period. *Mária, Édesanya, Édesapa, Lóri, köszönöm hogy mellettem álltatok és segítettetek.*

Szilárd

27 May 2003
Enschede

Contents

Chapter 1

Introduction

In this thesis, we present our efforts towards researching visualisation and interaction modalities for the purpose of developing easy to use, three dimensional and integrated modelling systems aimed at character geometry and bone hierarchy creation. Although numerous and powerful graphics editing systems exist for this purpose, our system takes an approach that is more academically oriented. Similar to many academic visualisation, interaction and other types of projects, we make use of the open standard 3D (three dimensional) description language VRML (Virtual Reality Modelling Language). This content description language and its extensions for scripting and programming provide us with a 3D visualisation modality, an event-based interaction framework and last but not least, a platform that is web-enabled, allowing for a wide-spread web-based access and consequently interaction.

Our goal of virtual modelling, with the purpose of using the resulting characters as avatars or embodied agents in virtual environments, drives our exploration of modelling possibilities in VRML. After gaining knowledge of graphics modelling characteristics regarding primitives, selection and manipulation components, analysing HCI (Human-Computer Interaction) principles and taking in consideration the H-anim specification (Specification for a Standard Humanoid), we developed a modelling system with the following main characteristics:

- Single vertices and automatic collections of vertices are used as primitives. These collections are based on the natural ordering properties of regular meshes, which suggest a possible row (or ring if the mesh is closed) and column separation of the vertices. This hybrid approach allows us to handle the vertices using a low-level manipulation modality, while not excluding higher-level manipulation options that take advantage of the automatic vertex collections. Low-level operations applied on collections of vertices and higher-level operations directed to collections of vertices provide editing modalities that are powerful, provide a quick manipulation approach and are easy to comprehend by occasional or novice users.

- The interface is based on 3D editing and widget principles and it is embedded into the environment itself. This inclusion is possible by using an editing target that is not a whole scene: since we are using only a 3D character, the different controls can be embedded in the same environment as

the edited character. A single view is used instead of the historical multiple views present in almost all graphics modelling packages. By providing tools for positioning, orienting and scaling the edited entities in the form of precise manipulation widgets we eliminate the need of multiple, visually unattractive views.

1.1 The setting

When the research that concludes in this thesis was started, the title of the research proposal was "Multi-modal interaction in virtual environments". This title suggests the use of different input modalities in communication towards and from a virtual environment. Although we retained our interest in commonly available peripherals as input modalities (like a microphone or a web-cam), our research focus has shifted towards virtual reality, graphics and the interactive editing possibilities that virtual environments (VEs) could present.

Generally, VEs present surroundings that resemble different aspects of the real world, thus it is more intuitive to the user. In this setting, objects that present an interaction modality are also natural in appearance and functionality. Many content developers base their interactive objects on real-world examples, conventions or semantics: light switches, traffic signs, doors, value sliders (music volume, light intensity), etc.

Adequately combining VEs and 3D GUI (Graphical User Interface) technologies (although there are not yet standards, conventions in this area, especially at the interface level) is described as a promising research direction for testing 3D interface and interaction modalities. These technologies could be used for assessing the presence and immersion level of VE users.

The shift of our focus was determined by multiple factors. First of all, the need of our group for embodied avatars and agents, the need for animation and more evolved interaction possibilities has been the driving force besides our interest in VR and graphics. Observing the academic concern for inexistent 3D conventions, standards and specifications referring to the interface showed us that this is a territory that could be further exploited. By looking at the VRML creation tools, the general editing systems (graphics modellers), the general shift of applications towards web-enabled technologies and the increased power of current desktop computers, revealed the lack of specialised systems and also the technical feasibility of attempts to research on-line interactive modelling possibilities.

Besides the main language track our group is involved with virtual reality and virtual environments as an application platform for language research like vocal input, dialogue systems, natural language processing or lip synchronisation. To this extent, there are a number of students and researchers in our group who are pursuing research related to VEs.

The main product of the group and therefore the obvious choice for research is a virtual environment called "Virtual Music Centre", VMC for short. It portrays a theatre and performance hall in the home town of the university. It is a system which initially provided (besides the graphical environment) text dialogues for search within the art and music performances database of the VMC. This system nowadays is being extended with natural language recognition and interaction, experimental navigation systems, agent architectures, etc.

The environment provides a number of interaction possibilities. A playable piano, an exhibition area for different publications, an information board, a music wall among others, provide different options to the user. A 3D character has been also built from simple geometries in an attempt to make the environment more friendly, and perhaps to integrate the dialogue system with a human-like character. These latter requirements for humans and the possibility to include embodied agents into the environment provide an opportunity for the research of simple, eventually animation enabled tools aimed at the creation of 3D character hierarchies. Additionally, the integration of such tools themselves into virtual environments provide a setting for virtual interaction experimentations.

Simple and wide-spread VE usage also requires adequate means of access. We concentrate on environments that are usable by all web users, thus on environments that are usually visualised and acted upon in desktop hardware configurations. The primary requirement to explore these environments is a keyboard and mouse as input devices, all standard in desktop systems.

1.2 Requirements and technology

The requirements for our work can be already deduced from the previous section. We need a system that is an easy to use virtual environment offering the possibility to create virtual characters and possible animations without the use of complicated graphics packages. We also need a system that is easy to develop and to extend, it can be integrated into a VE and, besides the usability side, its interface, internal structure and interaction options gives the opportunity to effectuate experimentations in VR modelling. This can be done using novel graphics primitives, integrating the controls into the environment and with different manipulation options. We are also concentrating on an easy to use and natural interface that provides a low learning curve and thus it is suitable for inexperienced users, but still provides an acceptable alternative for creating characters aimed at interactive systems.

The technological basis for this work has been determined by the requirements and the previous technological solutions used in our group. The VRML language used to build the VMC environment is capable to provide a web-enabled visualisation engine via third party viewers, and extensions to the language allow the usage of Java as the programming language aimed at creating and managing the dynamic content necessary for modelling purposes.

The chosen language presents different properties that makes it attractive and it is easy to extend if needed. The possibility of interactive content, 3D graphics and real-world resemblance provide researchers with the tool of choice for alternative, experimental type of projects.

These choices at the present time have some drawbacks. Since major browsers try to get rid of Java compatibility or changed plugin architecture, current VRML viewers resort to older JVM (Java Virtual Machine) implementations or even web browsers to run properly. However, as VRML viewer makers update their products (which happens at the moment rather slowly because of lack of competition), the software platforms should return to an adequate level for advanced, real-time 3D support.

1.3 HanimPlus system

We present as the tangible result of our research and development efforts a virtual environment that provides an alternative to complex editing systems, which we call HanimPlus system. The environment is aimed at creating 3D characters based on the components of the H-anim humanoid hierarchy specification. To assess the feasibility of virtual reality systems and their interactivity possibilities for modelling, our approach uses the graphics engine, sensor capabilities of these types of systems with the purpose of handling the different properties (components) of embedded geometry objects. Data handling, communication management and custom, mostly experimental complexity handling provide a novice or occasional user friendly, web enabled 3D character and hierarchy modelling approach.

We also explore the possibilities to use integrated 3D controls and the modalities they provide for the different interaction properties: ease of use, visualisation, direct manipulation, visual feedback, etc. In the following subsections, we will introduce the components of our system, described later in detail in Chapter 4.

The HanimPlus system components can be divided into three main categories. The first category represents the geometry editor part of the system, and it is described chronologically, from a first prototype to the symmetrical second one, which have been combined and included later into the overall system. The second category constitutes the hierarchy editor, which has been developed to work in synchronisation with the geometry editor. The third category consists a number of alternative systems or prototype modules not yet integrated into the general HanimPlus system. Figure 1.1 represents the different components of the HanimPlus system, the target entities they handle and the relations between the target entities and the components.

1.3.1 Geometry editor system

Our first experimentations with a virtual environment aimed at graphics modelling resulted in a geometry editing prototype based on regular geometry meshes. This system has been the base for a symmetric geometry editor system, which has been extended to handle automatically symmetric meshes next to asymmetric meshes.

1.3.1.1 Geometry editor

The first step of our virtual editing experiment has been to specify geometry and manipulation properties. A classical vertex, edge and polygon primitives based low level approach seemed tedious to the user, while higher level approaches needed more powerful platforms, and were not suitable for novice users. Our aim to provide an easy to use approach for modelling motivated us to search for other approaches.

Once we noticed the natural segmentation of regular grid geometries into rows and columns of vertices, we quickly deduced the modelling possibilities hiding within the use of these as primitives. Automatic multiple vertex selections, multiple vertex oriented operations and the naturalness of the whole approach in favour of fast and easy geometry handling weight more than the

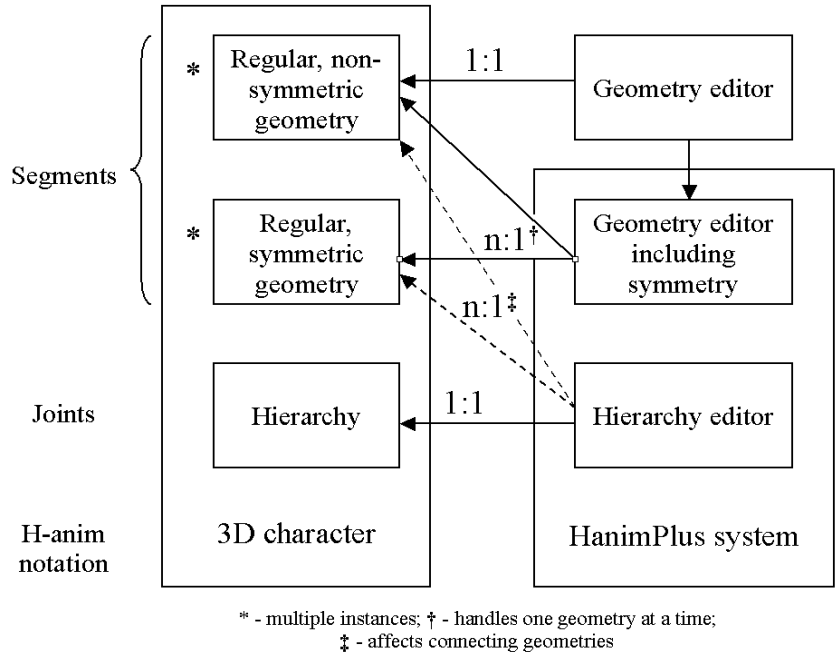


Figure 1.1: HanimPlus system overview.

limitations imposed by the regularity constraint, especially since a further objective was the modelling of 3D characters for virtual environments, in which case the regularity constraint is not particularly restrictive.

Using single vertices and collections of vertices as primitives, the basic vertex displacement operations applied to collections of vertices become handy editing shortcuts. These have been extended with operations directed at collections of vertices: scaling, smoothing, extending or reducing a geometry which altogether provided a higher level manipulation approach oriented to low-level graphics primitives.

The initial system was very much like a testbed for assessing the feasibility of VRML as a graphics modelling environment. From a technological point of view, it was a test of the dynamic managing possibilities necessary for handling the non-dynamic aspects of VRML. Due to the language restrictions, not all components could be handled directly, especially with geometry complexity changes, in which case the edited entity and its control structure are re-generated.

Because of the event-based communication modality of VRML was not enough for our geometry content handling needs, we used therefore an applet linked to our virtual environment through the EAI (External Authoring Interface), an extension mechanism defined for VRML. This applet serves as the database for all the positional, geometrical and control structure specific data, as well as a general event handler module that receives the interaction signals, handles the requested operations and updates or re-creates the dynamical, edited content that is displayed on the screen. This applet and EAI based communication method is also used in the HanimPlus system.

Since the geometry handling options are relatively few of number due to our choice of geometry primitives, our interface is based on a simplistic approach. All manipulation options are visible all the time, and they are activated or deactivated based on the selection context. Depending on their type, these operations can be classified as geometry appearance (surface) manipulators, view setting operations and complexity handling operations. Depending on their functionality, they can be classified as parameterless and parameter-supplying operations. The parameterless operations are transmitting a single activation status: thus their functionality can be satisfied with button-like sensor geometries, with textures that describe their functionalities in short text format. Operations that supply parameters are visualised as widgets, with geometries and sensors that suggest their functionality and provide to the user control to set the magnitude, the quantitative information of the operation performed. This latter approach is applied in case of displacement operations, rotation operations, scale operations, etc.

We found that this system worked well, with a few, mostly resource-related bottlenecks in performance, which have been eliminated later. It also showed us the types of sensor controls, visualisation, visual feedback shortcomings that we needed to address. We also recognised that precise sensory options, proper visualisation and control structure are essential for ease of use and proper geometry handling.

1.3.1.2 Symmetric geometry editor

The next phase of our modelling experimentations has been a version of the geometry modeller environment that incorporated symmetry. We observed that many character geometries are actually symmetrical, and generally geometry modellers handle these by providing operations for the users to symmetrically replicate geometries, operations that are usually effectuated manually. This implies that the user has to create and work with only half of a geometry most of the time, if the target character or geometry is required to be symmetric. We provide automatic symmetry handling capabilities to overcome the need for this awkward symmetric modelling approach.

The automatic symmetry handling method shortens the time needed to create symmetric geometries, while providing a consistent view of the edited geometry. The control grid is covering only half of the geometry, making the other, symmetric half available for preview purposes. The system code has been modified to maintain the geometry mesh symmetric in all circumstances. Due to symmetry, components of the geometry that fell onto the symmetry plane have been restricted to participate in operations that would result in asymmetrical geometries.

When combined with the hierarchy editor component, the geometry editor gained its present functionalities. Combining the previous options, it is now capable of handling both asymmetric and symmetric geometries. It has been also extended to include a new type of symmetry: external symmetry, which is applicable to pairs of symmetric geometries, e.g. human limb components. This has been a logical step, since the majority of the living or imagined beings that can be used as inspiration source for a character hierarchy contains body components that are either internally symmetric or have externally symmetric counterparts.

One minor drawback in current symmetry handling is the current pre-set behaviour. At the hierarchy level, the Joint (bone connection) components describe the symmetry properties of a character, and the geometries are regarded as asymmetric, self-symmetric or as having symmetric counterparts based on these descriptions. With hierarchy refinement operations, the local symmetry properties and symmetry rules defined by us decide over the symmetry properties of newly created Segment geometries (Joints and Segments are the hierarchy components we use). This does not allow the user to activate or deactivate symmetry preferences, which is at the moment only possible at data level (for turning self-symmetric geometries into asymmetric ones). We feel that for data consistency this is currently the best practice, but we do not exclude to provide symmetry options to the users in later versions of the system.

1.3.2 Hierarchy editor system

The H-anim human-like characters standard defines hierarchy components in form of Joints and Segments, for the benefit of interchangeability of geometry, animations, etc. Joints represent character bone connections (and provide a rotational mechanism), while Segments represent the skin geometries linked to character bones. We make use of these hierarchy components without applying the naming and hierarchical constraints depicted in the standard, for the simple reason to allow the construction of hierarchies that may or may not conform to the standard, and by this we extend the possibilities of our system beyond H-anim conformance, to a much broader set of characters.

To build the hierarchies, we make use of the Joint and Segment components, for which we developed different visualisation, selection and manipulation tools. Just like with the Segment editor systems, sensor boxes visualise Joint positions, while Joint hierarchy properties are visualised by line geometries. The whole character is also visualised, with sensors attached to the individual Segment geometries for target selection in case of the geometry editing mode.

The sensor boxes provide the selection functionality for the Joint hierarchy, where the selection is transcendent towards a whole character visualisation (through its Joint-Segment correspondence) and vice-versa. This allows the user to select geometries or Joints for editing purposes both via Joint sensors and the character Segments.

We define and use special objects for hierarchy ending points. Joint-like sensor boxes are pinpointing the end-coordinates of the different character geometry leaves. It is similar to having an extra Joint position, but without Joint complexity handling operations enabled. This allows the positional manipulation of these terminal coordinates and of the connecting geometries.

The most important aspect of the hierarchy editor subsystem is to handle hierarchy complexity. Based on a number of hierarchy properties (what sort of hierarchical connections are possible) and on considerations regarding hierarchy symmetry, we define several hierarchy complexity handling operations:

- “add terminal Joint” operation extends the hierarchy at selected terminal Joint level,
- “insert Joint” operation splits a Segment and inserts at the split point a new Joint,

- "ramificate Joint" operation creates new subtree starting point(s) in the hierarchy.

If there are symmetry properties to be maintained, these operations replicate the executed hierarchy and geometry handling steps also on the symmetrical entities. There are cases in which, depending on the Joint selection type (terminal or internal Joint) and the symmetry type (internal or external), the result of an operation would be either 1, 2 or 4 new Joint elements created (in case of the ramification operation).

The positional and complexity operations effectuated on Joints are reflected also on the geometries attached to these Joints. Displacing a Joint coordinate would deform the connecting geometries ring-wise (where the displacement percentage is based on the closeness of the respective ring to the Joint) to follow the Joint, making this operation an effective modelling option. In the case of hierarchy refinement, new Segment geometries are created in synchrony with Joints, either by dividing existing geometries or by generating completely new coordinates, based on the features of the vertices they are connected to in the hierarchy.

By using these automated geometry handling options, we actually apply a modelling approach to hierarchies based on the concept of local modelling. Normally, in a hierarchy when a component is positioned, all child components replicate the positioning, causing the displacement of many component. We inhibit this behaviour by introducing counter-displacement actions automatically, localising thus the effectuated operations. In this case, the overall shape of the character and its hierarchy are preserved.

1.3.3 Other editing components

By limiting the geometry construction to a regular mesh structure, we could observe the similarity between our vertices and the knot grid that defines a NURBS (Non Uniform Rational B-splines) surface. With the introduction of NURBS node extensions to VRML browsers (and in the next version of the standard) we had the opportunity to convert our geometry editor into a NURBS editor based on a vertex grid to knot grid mapping.

Although the actual mapping seems to be a direct, straightforward coordinate to knot conversion, this mapping is not enough for constructing NURBS surfaces which are the subdivided, smoothed, complex versions of the original polygon mesh. In reality, NURBS surfaces need knot repetitions to produce closed surfaces or to maintain similar boundaries as the original mesh. We found that a 4x2 order for the knot grid would allow us to maintain the vertical boundaries of the mesh, and using a single horizontal knot repetition the surface would be also closed. This order also provides a range of surfaces that is complex enough for our character editing purposes, resulting in smooth geometries.

Such an extension could provide high-level surface modelling possibilities, especially if other qualitative properties of these surfaces can be also handled by the user (like the knot weight properties of NURBS surfaces). Other higher level surface modelling approaches can be made available if they are to be included in the specification.

Textures are important components of geometries, especially if they are used to increase the realism, the qualitative level of an object. Since we are working

with characters consisting of simplistic, light geometries to be used in on-line virtual environments, textures are more than welcome.

We handle textures in our experimental texture applicator module based on a direct scaling of the vertex grid into the coordinates of a texture. Texture coordinates are depicted by user-controllable sensor geometries, that can be displaced freely on the texture surface. At this moment we do not allow texture repetitions (character geometries usually do not require tiled textures), but it is possible to use bigger textures containing more texture information than needed in which case the user can restrain the texture coordinates of the target geometry to a sub-area of the whole texture.

1.4 Outline of this thesis

In the present introductory chapter, we describe the thesis structure, providing insight into the circumstances, the requirements and the technologies that led to the development of the prototype systems described in Chapter 4. We introduce our system based on a natural segmentation according to the components of the system, segmentation which also shows the chronological order of our work.

Chapter 2 is an overview of the different research fields and their findings which regard or are applicable in the case of VEs, graphics, 3D characters, etc. These range from the field of information visualisation and user interfaces to interactive systems or human computer interaction, with a summary review of complex systems that tend to incorporate results from many of these fields.

We use these technologies for the different aspects of our modelling environments, when considering visualisation, interaction, or feedback choices. The review of complex systems is also an inspiration source to the possibilities offered by virtual reality systems.

Chapter 3 provides a description of general graphics concepts and modelling properties, from low-level graphics to high-level, or even experimental modelling approaches. The review of historical, classical and academic visualisation, modelling, interaction, interface approaches show us the strengths and weaknesses of the different systems, allowing us to deduct the right approaches for our modelling purposes.

The second topic of the chapter includes virtual environments and the technologies to enhance these environments by using built-in, knowledge-based or other types of enhancement possibilities in form of different additional functionalities present in these systems. These enhancements generally improve the quality of these systems, making exploration or other tasks easier for the user.

Chapter 4 contains the actual description of our work, which resulted in a number of prototype systems, aimed at modelling in virtual environments. A general guidelines section reveals the background, requirements, considerations and options that led to the present form of the system. The history of the HanimPlus system is presented, consisting of a geometry modeller which has been extended later into a symmetric geometry modeller, systems which at their turn have been included as components of the HanimPlus character modelling system, side by side with the hierarchy modelling component. We also present other modelling possibilities, for handling NURBS surfaces or for applying textures on geometries, modelling options to be included into the final

overall system. This chapter presents our contribution to the field of virtual reality, and especially to the interaction possibilities such systems provide the platform for.

Chapter 5 compares a number of general or specialised, high or low level, hierarchy enabled modellers to our system. For this purpose we have chosen modelling systems that are in some form of another generally available. This comparison shows us how our system performs, with respect to comparison topics such as selection, manipulation, indirection, hierarchy, etc. properties of these systems. As a result, we deduce the strength of our system, as well as the shortcomings and where and how our system can be improved.

Chapter 6 describes the general conclusions drawn from our work. We present the main line of arguments, the reasoning and novelty behind our modelling approach and the results we have achieved. The system properties are also characterised, based on different assessment options, to see whether our usability aims have been achieved.

We also explain in this chapter the research directions we would like to continue in our work. Our further efforts and investigations include enhanced controls, animation editing and control, integration of NURBS and texturing subsystems, as well as variations in the target capabilities of the system, to be able to eliminate the current geometry or hierarchy restrictions.

For a better overview of the thesis, we provide small summary subsections at chapter section levels. These subsections are meant to help the reader not only in his/her orientation in the presented material, but also to depict and follow easily the main arguments we try to present.

Chapter 2

Background

This chapter gives an insight into the available research topics that are relevant to our work. Investigations of information visualisation, interaction modalities, interactive interfaces, widget technologies and human computer interaction (HCI) were needed to assess the current state of the art for the respective technologies.

This review is used as a starting point for our investigations and developments. The knowledge of previous works in this area is needed to build upon and advance in our experiments. Visualisation, interface, widget techniques are useful for designing the virtual environment, while interaction, collision, HCI technologies are used to specify the behaviour and to assess the different aspects of the resulting interactive systems.

A number of complex systems are also reviewed that include different technologies for animation, behaviour or agents creation and the modalities to direct these systems. Such systems are eventually the target level towards which our systems are being developed, integrating and then testing novel, innovative technologies and concepts.

The general technology background on which this work concentrates resembles to a certain extent to the review of technologies described in [Walsh and Bourges-Sevenier, 2000]: both reviews are based on interaction and visualisation techniques aimed to be used in combination with web technologies.

2.1 Information visualisation

The relevant literature provides a number of information visualisation (IV) definitions. Although the field of IV is large and the handled data can be of various forms, the definitions always accentuate the process of better human cognition through proper visualisations.

[Card et al., 1999] defines IV as "computer-supported, interactive, visual representation of abstract data that amplifies cognition". Humans tend to comprehend visualisations better than data, and the purpose of IV is to represent these data quantities in a visual form, possibly with different options for easier data meaning associations and eventually even with built-in data analysis and filtering solutions. Depending on the data type, even multiple visualisation types can be used to convey the sought information to the user.

[Gershon et al., 1998] defines visualisation as the act or process of interpreting in visual terms or of putting into visual form. The IV process is described as the process of transforming *data*, *information*, and *knowledge* into visual form making use of the natural visual capabilities of humans. The goal is defined as the computer-assisted use of visual processing to gain understanding of the abstract data values.

[Schreiber et al., 2000] describes the components of the IV process as follows:

- data values are input signals to sensory and cognitive processes,
- information is data with an associated meaning,
- knowledge is "the whole body of data and information together with cognitive machinery that people are able to exploit to decide how to act, to carry out tasks and to create new information".

[Ware, 2000] defines the human visual perception system as a "massively parallel processor that provides the highest-bandwidth channel into human cognitive centres." It also notes that visualisation is culture or society dependent. To convey the visual information, stereotypes, commonplace labels and conventions have to be used that are set from early stages.

Recently, there have been a multitude of approaches and applications for the visualisation of abstract data in VEs (Virtual Environments). The visualised data include network analysis data, geographical data, statistical mathematical data, algorithms, etc. [Boyd et al., 1999] starts from this need for visualisation and proposes methods to integrate data visualisation tools into VEs. [Gausemeier et al., 2000] presents a tool that integrates CAD (Computer Aided Design) data and drawings into a virtual and web-enabled application, allowing the visualisation of large-scale plant constructions.

Renowned examples for IV research are:

- The cone trees in [Robertson et al., 1991] provide a modality to effectively visualise hierarchical structures. For instance, directory trees are visualised with labels arranged according to an interactive (revolvable) wheel shape, while the directory structure is shown by edges connecting the different parent-child wheel centres.
- The table lens and fisheye (focus and context) approach in [Rao and Card, 1994] for selective, interest-based and effective data visualisations. This technique enlarges the content that the user is focussing on, as well as the content that is arranged adjacently (and thus with related content). Similar is the mind map theory of [Buzan and Buzan, 1995] and the multitude of applications that span from this technique and provide a data management approach based on data item relations.
- Approaches like zoom techniques, "geon" representations (graph components represented with different graphical properties like orientation, colour, etc. for easy visual recognition and memorising), 2D construction planes, viewpoint markers, enhanced orthographic views etc. are described in [Grossman et al., 2001] as used for efficient display of data.

There are a number of research examples to attempt to use IV techniques within VEs for efficient visualisation. For example [Luttermann and Grauer, 1999] treats the visualised dynamic data (not the static geometry) as time-variant objects, e.g. objects that have a time dimension assigned to them. [Ressler et al., 1999] discusses VR as a visualisation and simulation alternative for manufacturing systems, even multiple connected manufacturing systems.

An interesting aspect of the IV research is the architectural, cultural and museum-oriented information visualisation projects:

- [Shiode and Kanoshima, 1999] provides user preferences based, personalised dynamic exhibition virtual environments. The users are presented with content that is most likely interesting to them after they had specified their preferences.

- [Roussos and Bizri, 1998] presents mediaeval stories in narrative environments. Here, the emphasis is on the presentation of historical settings, atmosphere and the historical events themselves (even if they are legends), with different story line paths towards a final piece of narrative.

- [Schaerf and Tessicini, 1999] discuss a virtual tour of Rome. It provides besides lists, informations and visualisations of different cultural, architectural, etc. sights also the possibility to plan routes between these sights based on sight preferences and time constraints.

There are also a number of cartographic IV initiatives. For instance GeoVrml [Reddy et al., 2000] has become part of the next generation VRML specification (X3D). [Dykes et al., 1999] presents the initial thoughts on such an extension. [Reddy et al., 2000] describes the system with more technical detail. The system provides a modality to visualise captured terrain measurement data, but also provides the possibility to create automatically terrain for VEs from measured data with tools that convert between the different data formats.

CAD-like furniture assembly is an approach to visualise furniture components and custom furniture that the user wants to build. The special purpose editing approach incorporates design knowledge with user preferences and visualises the resulting furniture concept [Nousch and Jung, 1999].

Another specialised visualisation approach is taken from biological examples. Using artificial intelligence and synthetic life form approaches, biological visualisation is achieved through the use of multi-cellular systems that use a common interface to communicate, indifferently from the functions they perform [Grand, 1998].

[Giorgetti and Palamidese, 1999] describes an approach to integrate speech and sound into animation modelling based on musical score paradigms and visualised accordingly. The speech, sound and animation components are treated as media objects and they are visualised for timing, synchronisation, modification, etc. purposes.

A number of interaction methods in a multi-modal interaction setting are discussed in [Waibel and Duchnowski, 1994]. The interaction methods and their hardware consist of: speech recognition (microphone), gesture recognition (tablets), handwriting recognition (tablet), lip-reading (camera), face tracking

(camera), eye tracking (special or normal camera) and acoustic localisation (microphone). Some of these techniques can be integrated into desktop interactive systems by using more conventional input devices like the mouse (for gestures) and a web-cam (for face tracking, lip-reading, eye tracking) and a microphone (speech recognition).

[Rickel and Johnson, 1997a] and [Rickel and Johnson, 1997b] suggest an interaction style based on autonomous agents (virtual teachers) that provide examples of how to handle different virtual equipment (replicas of real equipment) getting feedback from the virtual equipment and the virtual teachers (nodding, shaking head, tracking objects, etc.). The agent-environment interaction is also handled in an appropriate manner with visual attention, gestures and object manipulation.

Summary
IV techniques provide the tools to visualise and to prepare for manipulation different data sets. We use these techniques as inspiration sources for our visualisation choices. Although our approach is closer to graphics modelling concepts, we do use an approach of extracting hierarchy information instead of the manually-set skeleton and mesh connection used commonly by graphics modelling systems. We also make the logical connections within the hierarchies and geometries visible through relation handling methods, giving visual feedback for structure clarity. Furthermore, our work currently proceeds to incorporate animation handling and visualisation into the H-anim character editing system, making use of the reviewed techniques.

2.2 Interfaces and interaction

Interactive applications and interfaces allow users to create and/or manipulate objects in a visually portrayed environment. Although in the strict sense interaction covers all the mechanisms for exploration, we concentrate on interaction that is related to the graphics editing and VR fields where a user not only explores the program environment but also affects it in a visual and responsive manner.

2.2.1 3D user interfaces

The common 3D views present in modelling applications (3 side view and one perspective view) are not exact matching views of different applications, but they are all historically-based visualisations [Mealing, 1992]. They are grounded in the possibilities of input devices (mostly and historically limited to two dimensions) but the most limiting reason is the lack of 3D conventions and standards, forcing the 3D applications to use the existing 2D standards and the WIMP-based (Windows Icons Menus and Pointers) graphical user interfaces. However, there is an increasing number of research reports that point out the importance of direct, menu-less, integrated editing tools for 3D modelling in form of the "widget" concept [Bier, 1986] [Conner et al., 1992], a concept applicable also in the case of virtual environment based editing applications. Head Mounted Displays (HMDs) and other immersive display technologies also limit the usage

of 2D WIMP conventions. In these cases, the use of integrated, 3D widgets in case of modelling applications is also more than welcome.

There are a number of approaches to transform different Operating Systems (OS) by providing a 3D layer on top (desktop, windowing system) that presents the familiar OS desktop functions in a 3D environment: folders with different topics as different rooms, web browsers as screens, etc. Such systems are ROOMS 3D [ROOMS 3D, NA], TaskGallery, 3DWM [3DWM, NA], 3D Top [3D Top, NA] Win3D [Win3D, NA] or Icon3D [Icon3D, NA]. Some of these are simpler, while others provide possibilities to present application screens as textures for the 3D desktop environment objects. There are even approaches to present system processes as elements of 3D shooter games like Doom, resulting in an unorthodox process management approach [Chao, 2001].

A common assessment dimension for virtual interactive environments which relates mostly to hardware and visual quality is the feeling of immersion. There are special hardware approaches that can add to the immersiveness of a visualised VE. One of these is the artificial vestibular stimulation approach of [Campbell and Jacobson, 1999], where a medically applied technology is transformed to VR usage. This technology consists of the artificial stimulation (by applying small voltage electrical stimuli) of the human balance sensor centres, resulting in the sensation of movement. Another special hardware approach is the scent producing hardware initiative iSmell from DigiScents (the company is now defunct) or SENX Scent Device from TriSENX [SENX, NA] that could recreate scents or even taste which result in extra, olfactory or gustatory input to the user's experience, making the VE exploration more present, more immersive.

2.2.2 Interaction

Fitts' Law (see IV, GUI - Graphical User Interface or HCI textbooks like [Card et al., 1983]) is the result of an experiment effectuated to assess aimed movement and to predict human psychomotor behaviour in interactive 2D user interfaces. The technique is based on time and distance measurements toward a target object that yields in formulas (already suspected from common knowledge) asserting that movement time is a function of movement distance and that movement time is also a function of target size. The drawback of the method is that it turns non usable in 3D settings (it does not scale to higher dimensions) where the objects have both width and depth, both unaccounted in the original experiment.

A similar approach for dealing with the relations between the relative sizes of controller, cursor and target entities is presented in [Wang and MacKenzie, 1999]. It is shown how these entities affect object manipulation in a VE setting. The object manipulation speed increased when the controller and cursor had the same size. When the cursor and target were the same size, the object manipulation accuracy was facilitated. Both cases suggested these properties regardless of the absolute size of the entities.

The main source of interactivity in a VE is presented by the fact that the user is capable of navigating and exploring the environment (if navigation is enabled and there is something in the environment of course). The interactivity elements added to the basic navigation provide the real valuable interaction possibilities, which are in most cases environment-specific.

However, there are also general approaches to enhance the basic navigation facilities of VEs. [Bowman et al., 1997] presents viewpoint motion control techniques where the navigation is effectuated based on gaze and pointing. Somewhat similar, [Strauss et al., 1999] shows an interaction method where the real space is the interface to the VE. Real people are tracked with sensors, and the environment reacts to their collaborative body motion. The system is used to provide an interface to virtual environments that is usable for culture, performance, art and entertainment. [LaViola et al., 2001] proposes a gaze and footwear based navigation approach where the orientation is set by gaze and the navigation speed is controlled by special footwear, with the objective of freeing the hands for other types of interactions. [Grossman et al., 2001] also mentions the importance of two-handed interaction methods in case of effective handling of large data sets and the properties of different visualisation techniques.

Gaze information can present also a source of interaction. In the virtual conferencing system of [Vertegaal, 1999] gaze analysis was used to convey the focus of individual participants within a conferencing VE scenario. Participants were tracked for their eye gaze, and other participants could see whether their counterparts look at a shared document table, at another participant or present active listening properties. Other systems use gaze direction information to optimise the visual display in case of large-scale virtual environments by presenting detailed scenery in the display area where the user focusses and optimised (simplified) scenery on other areas of the display [Danforth et al., 2000]. This type of optimisation is particularly interesting in large-scale, detailed terrain visualisations where occlusion-based optimisation is missing.

Advanced systems for interaction such as shown in [Sannier et al., 1998] provide interactive control for directing multiple actors and cameras, with different levels of control for animation, visualisation, speech, etc. The described system is a complex tool used to orchestrate the performance of virtual actors, to capture it, to replay it, providing a sort of virtual performance.

Interactivity can even take the form of semi-autonomous avatars. This is a bread of avatars that are performing certain tasks automatically, while their owners monitor their performance and intervene by taking control over the avatars whenever necessary [Wilson, 1998].

A proposal to evolve VRML toward an object oriented (OO) construction (with prototype-instance object model and indirect constraints) is given in [Balaguer, 1999] for a more flexible, dynamic world manipulation approach. The constraints are presented as fields in nodes that contain formulas instead of the event routing mechanism currently used. Similar, object oriented concepts are observable from [Mesgari, 2000], where the 3D topological data structures are described in OO manner.

A special interaction case with VEs is treated in [Mauve, 1999], where the concepts of accessing and saving VR state information are presented regarding the whole environment, objects or events in an environment, with the possibility of later restoring them.

[Brogan et al., 1998] suggests interaction hardware (for VE interaction) based on a bicycle platform for cycling and herding applications. Other specialised interaction hardwares exist also, such as the:

- "Omni-directional Treadmill" of [Darken et al., 1997] or the "Infinite Floor" of [Iwata, 1999, Iwata et al., 2001]. These instruments are inspired

by the normal running treadmill, with a refined construction. The belt itself is segmented and it is composed of a number of miniature belts, which provide the sideways motion not present in normal running treadmill.

- A number of enhanced treadmill devices used with VR, where the additions consist of position tracking, slope simulation for greater walking effort accuracy or even terrain height simulation (with boxes or hexagonal pneumatic tube constructions).
- Different types of foot discs with different conventions for directing the navigation desires.

A "synthetic" approach is taken in [Bowman and Hodges, 1997b] by simulating the use of a real-world input and information device via a virtual replica. The pen-based tablet tool provides interfaces (menus) and maps within a VE, with the possibility of putting the tablet away. The system is implemented with the use of special hardware, a HMD and a tracked pen.

With regard to the interaction types possible in an interactive system, [Barrilleaux, 2000] uses a classification based on the "control personae" paradigm. In function of the 1st, 2nd and 3rd person manipulation modality options, the interaction can have different meanings:

- 1st personae interaction means navigation in VE. In this case, any pointer dragging in the window (eventually keyboard input) represents the turn, displacement, etc. desires of the user in the scope of movement and exploration within the environment.
- 2nd personae interaction represents the dragging of an object within the environment. This sort of manipulation is also called direct WYSIWYG (What You See Is What You Get) manipulation. Unfortunately, this sort of manipulation is not very effective in 3D with 2D input devices commonly used (like the mouse), due to the lack of proper 2D-3D mapping possibilities.
- 3rd personae interaction describes the manipulation of a target object through a widget. In this case, the user has to know exactly what the widgets do. Fortunately, at least for the basic translation, rotation and scale operations, the parameters to be learned are limited by number and also intuitive: local axes and planes, respectively the visual sensors that repeat themselves on the axes and planes. In Chapter 5 a number of representative widget constructions are presented as examples.

2.2.2.1 Widgets

A branch of interaction modalities that are applicable to VEs as well as to general 3D graphics editing systems are the geometry-behaviour construction modalities expressed with the term widget [Matejic, 1993]. These constructions have different properties integrated by the developer which allows for direct interaction and visual feedback at the same time. Such a widget definition does not restrict these constructions in functionality: thus the potential they have is only limited by the creator's imagination, intentions or requirements. [Matejic, 1993] even looks at real widget building problems, which are defined

as graphics (how the widget looks), mathematics (what the widget calculates) and constraint (what are the limits of the controlled target entity or property) requirements that the widget builder has to specify.

Widgets are generally used to specify different modelling actions, operations, and not the user's interaction within a final, published VE. However, with the strong opinions about widget integration in 3D space [Bier, 1986] [Conner et al., 1992], we can define a new sort of virtual environment aimed at graphics modelling or virtual production. Widgets are regarded as superior control tools, since different geometries and sensors can be integrated in a relatively small space. This provides different direct access options targeting different geometrical axes, planes, and these possibilities can be exploited for "virtual modelling".

[Döllner and Hinrichs, 1998] considers widgets as higher level building blocks for geometry and communication. It goes one step further with visual inter-widget communication editing: they use behaviour graphs to specify their behaviour.

Concrete examples of widgets integrated in different systems are presented among others in [Keefe et al., 2001]. Timeline widgets, colour widgets are used for controlling a virtual painting system. [Cubaud and Topol, 2001] uses widgets to control the presentation of a digital library contents at collection or individual item level. "Skitter" widgets are described in [Bier, 1986] as 3D cursors that position different spatial features (target coordinates).

Recently, there have been efforts to build a "PROTO Repository" aimed to collect prototypes for the VRML language. The collected prototypes are published as part of the Web3D SDK (Software Development Kit), published quarterly by the Web3D Consortium. Some of the prototypes that can be encountered in the collection are prototypes of widgets. Simple slider widgets, colour widgets and even more complex, animation handling VCR-style widgets are available to the general public.

2.2.2.2 Collision detection

Collision detection methods are of great importance in VEs. First of all, the user (and its avatar) navigate between virtual objects usually with the collision detection enabled. If the user is close to a relatively high object (object cannot be stepped over), then the navigation is restricted as a result of simulated collision. Similarly, in the game industry, FPS (First Person Shooter) games use the same collision rules. Another widely spread use of collision detection in VEs is present in the form of terrain height following, although rarely regarded as a topic of interest [Steed, 1997]. Our work, presented in [Kiss and Nijholt, 2003] includes terrain height and collision information extraction in a framework that automatically adjusts the user's view for an optimal perception of the environment.

More interesting is the concept of inter-object collision, which gives additional possibilities for interaction specification. This technique is implemented in newer VR visualisation systems like Cortona for VRML [ParallelGraphics, NA] from Parallelgraphics and others. Allowing this way the implementation of basic physical properties, VEs can benefit from increased realism. It would also be a mean of implementation for proactive agents that possess the capability to manipulate objects or features of the VE they are situated in.

A multitude of methods exist for the actual user-object or object-object collision detection [Kitamura et al., 1995], [Thalmann et al., 1997], [Tu, 1999], [Steed, 1997] and [Volino and Magnenat-Thalmann, 2000]. These methods found their way into custom applications within VEs. [Bowman and Hodges, 1997a] uses some of these techniques (e.g. ray casting, arm lengthening) as a selection and manipulation tool for virtual objects, providing new types of interaction methods.

2.2.3 Human computer interaction

[Coomans and Timmermanns, 1997] defines interaction as "mutual response of computer and human to each other's actions". The human computer interaction (HCI for short) field specifies the interaction and interface properties of systems, and how the user relates to the system. HCI also treats the different measurements, guidelines that categorise interactive systems according to performance, usability, ergonomics, etc.

The HCI field provides the necessary specifications, guidelines and even standards for the measurement of effectiveness or usability of systems. However, these specifications are often aimed at 2D systems with emphasis on their menu-based or text based interfaces. We are interested in guidelines that are applicable or are specially defined for interactive 3D environments in general and for virtual reality in particular.

The 1998 ISO 9241 standard for usability: "Ergonomic requirements for office work with visual display terminals (VDTs), Part II, Guidance on usability" is a standard that defines acceptable work parameters for visual displays and for software that runs on these visual displays. [Smith, 1996] describes the properties of the standard, from which we extracted those concerning software. The dialogue principles, menu dialogues, command dialogues, direct manipulation dialogues or form filling dialogues are components of the specification that are targeting mainly 2D systems with dialogues and menus. The remaining usability guidance, presentation of information and user guidance descriptions are more interesting to us. They define the following system characteristics:

- system effectiveness - the extent to which the intended goals of using the overall system can be achieved,
- efficiency - the resources that have to be expended to achieve the intended usability goals,
- satisfaction - the extent to which the user finds the overall system acceptable.

The testing specification part of the same standard defines the usability tests as a combination of performance test (what the user can achieve, how he can achieve a target goal and how much time takes to achieve it). It also suggests the testing of physiological response: this however is only partly software-oriented, when the software implements ineffective input device handling, resulting in different physiological stresses.

Besides of specifying and quantifying interaction modalities, HCI also treats the outcome of interactive processes in order to gain insight into the efficiency

of a system. Execution time and ease, product range and applicability or interchange possibilities define next to other characteristics how well a system performs.

Direct/indirect manipulation is an HCI property which heavily influences the efficiency level of a system. Although the common knowledge of direct manipulation implies a single cycle of positioning, activating and displacing the cursor pointer, the direct manipulation concept is more fine-grained. Typically, we can speak of indirection in the cases of:

- Selection. There are two system properties that can cause indirection. The first one is switching to and from selection mode. This is an unjustified extra operation since selection sensor visualisation has no overlapping properties with a drag-type manipulation. As an example, a mouse click and drag fit perfectly the selection and manipulation intentions: thus a mode switch between them is unnecessary, effectively requiring a selection click, a mode switch (button click or keyboard shortcut) and a final click and drag. A second indirection source is requiring selection type specification. In the case of only partially overlapping primitive sensors (vertex, edge, polygon or other sensors), the users are freed by the selection type setting obligation if all sensors are active all the time, which can almost double the selection speed. This selection possibility is especially beneficial with inexperienced users who do not anticipate their modelling actions and group them according to selection type for fewer selection operations.
- Manipulation. The same indirection properties apply as previously with only small differences in their applicability. The mode switch induced indirection is the same as with selection, and it can be easily eliminated. However, with the different basic manipulation type options, although possible, it is impractical to visualise them at the same time. The necessary widget sensors would be then centred in the same position, aligned according to the same main axes and thus could easily overlap. Some systems overcome this by using one type of manipulation widget and visualising sensors outside the active widget volume, sensors which at their turn activate other manipulation widgets.

A particular component of usability is simplicity, although not widely discussed. However, it affects the learning curve of a system, the time required to complete a certain task and even the visualisation performance in the usually computationally intensive interactive systems. Although commercial modelling systems tend to be more and more complicated as time goes by, systems like [Igarashi et al., 1999] make inquiries into designs aimed at simplicity. Their sketch-based editor system uses different algorithms to extrapolate the user's gestures to commands and 2D sketches into 3D surfaces, resulting in a quick modelling approach, but unfortunately not providing tools for finer control over the generated geometry.

Besides the more common interaction devices, there are experimental approaches for other interaction paradigms via new input devices, which also provide a ground for HCI observations and evaluations. Some of these innovative interfaces are: two-hand interaction with navigation, selection, etc.; interaction transferred to foot; or even speech recognition hardware, described in papers like [LaViola et al., 2001], [Keefe et al., 2001] and [Darken, 1994], respectively.

Another aspect of efficiency is defined as on-line efficiency in [Naka et al., 1999]. A compression method for animation is discussed which is based on streaming, with the main concern for efficiency. Since animation data can be quite large, an efficient compression consumes less communication resources, making a system more responsive, avoiding congestion.

2.2.4 Summary

Historically, 3D editor systems used different views for editing: the 3+1 approach, representing the scene from top, front, side and 3D view [Mealing, 1992]. However, there are other approaches also, where the editor interface is inserted into the 3D environment, breaking away from the multiple viewpoints and also from the WIMP interface towards a fully integrated 3D interface [Gobbetti and Balaguer, 1995]. Following the footsteps of these latter approaches, we use the natural visualisation capabilities of VR systems and provide the necessary view handling tools that eliminate the need for multiple views.

Interactivity in its inherent form, as explored with the concept of widgets and with custom event handling methods, when used in combination with IV and HCI techniques it is a valuable addition to any modelling system with regard to naturalness, ease of use, simplicity or proper feedback in manipulation and visualisation. This is the path we have taken in our approach. All the functions implemented within the system interface that have more than one parameter (single parameter operations are portrayed as buttons), are depicted as widgets, 3D integrated controls that contain both geometry and behaviour specification.

Our work built on these observations, described in Chapter 4, explores the possibilities that such systems would offer, both from the perspective of the integrated GUI and from the perspective of a general interactive system possible to describe with the VRML file format and additional dynamic properties handling code.

Within interactive technologies, the VR type of environments and in particular the VRML format and its web-based viewers provide basic navigational and sensory bare-bone that can be used to specify the desired environment at a visual and control level. This makes the creation and utilisation of widgets both highly customisable and widely available.

2.3 Complex systems

Research efforts led by multidisciplinary, larger research groups try to integrate the different VR and VE related techniques to specify, produce and test complex systems. These systems integrate besides the normal static or animated VE components entities like embodied and animated agents that exhibit pro-active, intelligent behaviour or animated entities that simulate specific behaviours (typically animal or instinct based drives are simulated). Related techniques include path planning for movement, displacement, interaction, grasping and even gaze, which are topics discussed in many papers reviewed in this chapter.

Due to our interest in embodied avatars and agents, in the following sections we will look at character animation, behaviour, movement semantics and agent technologies, and how these are used complementary to each other for their integration in complex simulation systems.

2.3.1 Animation

We want to apply animation to articulated figures consisting of rigid links (Segments in H-anim [HAWG, 1999] terminology which is a standard for articulated human figures) connected by Joints with 1, 2 or 3 degrees of freedom (DOFs). The control methods use kinematics (time-based rotation values) or dynamics (force-based simulation of movement) to drive the articulated figures, methods for which the concepts and basics are presented in many animation textbooks [Mealing, 1992, Maestri, 2001]. The methods discussed differ not only in their motion control methods, but also in their animation possibilities like quality, speed, etc. and thus they provide possibilities for different target applications.

Besides the task of animating which could be quite complex by itself, some of the control methods presented take into account the context (environment) in which the character is being animated, such as the environment objects and obstacles or even the group and social behaviour [Thalmann, 2001], providing an integrated approach towards complex systems of Section 2.3, which integrate animation, behaviour (see Section 2.3.2) and agent technologies (see Section 2.3.3) into VEs.

The animation modelling components in classical editing packages have different tools for enabling easy foot positioning. [Maestri, NA] presents a number of these approaches. The first method is IK (Inverse Kinematics) with locks, when 3D character articulations are locked (pinned down) and IK calculations are used to calculate joint rotations when the character is moved (e.g. to eliminate foot skating). Some packages offer the same pinning functionality also for forward kinematics, when the user positions manually the unconstrained articulations to achieve the desired result. A similar animation possibility is to drag limb extremities, where internal joint rotations are resolved by IK.

An unusual method for animation is the use of inverse (or broken) hierarchies. This method is based on the forward kinematics animation of inverse limbs, typically the legs and arms. It is called broken because in order to achieve inverse hierarchies with non-repeating leaves, the torso has to be separated from the hierarchies and it has to be animated separately. This can be a little cumbersome, but the advantage is that the foot can be positioned without IK. Another approach to animation for which intuitive interfaces also start to emerge is the use of key-frame or mocap data and tools to test the manipulation possibilities on these data (combination, alteration techniques) as an interactive system would do in real-time. Such a system is presented in [Bruderlin and Williams, 1995] where filters can be used to alter animations.

2.3.1.1 Animation data and data capture

Traditionally, computer animation data is stored as key-frame data. This data structure has two components: the key which specifies relative time moments and the animation data values at these time moments such as rotation angle, displacement, etc. values [Maestri, 1999] [Mealing, 1992].

Creating these data values can be done manually (by talented animators) or can be captured automatically using different types of capture hardware [Magnenat-Thalmann and Thalmann, 1996] [Thalmann, 1996] or simpler mechanical devices like a bicycle [Brogan et al., 1998]. Mechanical, force-based measurements were used in the past for (medical) musculoskeletal measure-

ments and analysis such as the walking characteristics analysis presented in [Inman et al., 1981].

Newer, magnetic or optical commercial motion capture systems use different markers for tracking coordinates and even mechanical (or optical fibre) wearable systems exists for capturing movement, these latter being more cumbersome, but providing a greater area of movement freedom. Markerless, image-based motion capture systems are emerging, where the input is taken from one or more video cameras [Fua et al., 1998]. Motion capture systems usually store data as absolute coordinate values, but there are motion capture file formats that specify rotation data, which can be extracted from absolute coordinate values to be used for hierarchically articulated characters.

Next to body animation data, special motion capture systems are also used to capture facial animation data. [Ruttkay et al., 1998] uses this technique to identify MPEG4 (ISO standard for Coding of Moving Pictures and Audio) FAPs (Facial Animation Parameters) in captured data and reuse it with 3D and cartoon heads, consisting of a fixed skeleton and animated with spring simulations (muscle).

2.3.1.2 Generating animation

Table 2.1 enumerates and additionally roughly classifies the animation methods that are discussed in the rest of the paragraph. The classification dimensions are:

1. Interactivity/speed. The higher this value is, the more usable the system is in interactive virtual systems. Simple motion combination methods are the easiest and fastest form of animation.

2. Reusability/speed. This dimension appreciates motion control methods that reuse recorded movements, not requiring new motion data generated from scratch.

3. Generality. Physics based, IK motion systems score higher, since in theory they can produce an unlimited number of motion types, while procedural systems are limited by the motion data they possess (this is a qualitative and not quantitative limitation).

4. Quality. Physics based and qualitative variance methods score higher by producing unique and context-sensitive, adapted motions.

There are several characteristic properties that can be used to distinguish animation data. One of them is reusability. Some data can be reused in the sense of re-run cycles (like walking, running [Whittle, 1995] [Bruderlin and Calvert, 1996]) or by simply repeating a certain movement sequence when needed. There are also animation sequences that are dependent on the character's properties (position, orientation) relative to target entities and so they cannot be used in a pre-recorded manner, and these are animations like grasping or pointing [Huang, 1997] which are usually achieved using IK or similar techniques described further.

Physical simulation

A wide-spread method for animating articulated figures is kinematics. It is based on the motion properties time, position and velocity. The term forward kinematics is used when joint rotations are specified in function of time. The term inverse kinematics refers to the problem of specifying the forward kinematics values when the end-point (end effector) of the character articulation is known. There are many different methods to solve this problem. The most common examples of forward and inverse kinematics are the key-frame editors (which interpolate forward kinematics values from a few intermediate values) and the possibilities of fixing the extremities of articulated figures while moving the other components or dragging the extremities themselves (in this case IK methods resolve the joint rotations) present in many modelling/animation packages [Maestri, NA].

Dynamic simulations are force-based simulation methods that take into account the forces that occur in articulated characters as well as additional properties or events like velocities, mass, torques, collision [Huang, 1997]. This animation method is computationally expensive and it is used, for instance, in realistic simulations [Hodgins, 1996, Wooten and Hodgins, 1996, Tu, 1999] and tests for usability [Grosso et al., 1989, Badler et al., 1993].

Kinematics constraints are used in [Tu, 1999] for physics-based simulation of artificial animals (fish). Inverse dynamics is used through solving motion equations (Newtonian laws of motion) resulting in forces that are physically correct but not realistic. Constrained optimisation is used to find motions requiring less energy (open-loop controller) for naturalness.

[Boulic et al., 1996] and [Boulic et al., 1994] present a concept called "Inverse Kinetics", a variant of the IK incorporating centre of mass information. Another type of control system is the Proportional-Derivative (PD) controller (similar to a damped spring) shown in [Laszlo et al., 1996] and [Gritz, 1999]. PD is a proportional scale and deviation minimising derivative system, which means that it generates a force or torque proportional to differences in positions and velocities of the source and target coordinates.

A dynamics method is described in [Stewart and Cremer, 1992] with a musculoskeletal basis structure that uses a dynamic Newton equations system to resolve the animation. The system presented is defined as dynamic having the possibility to add or remove equations respectively to change attributes on the fly. Articulation relationships are modelled with constraint equations, dynamically also alterable.

A walking gait controller is presented in [Sun and Metaxas, 2001] that generates walking motion along a curved path and uneven terrain with high control path specification. The high-level parameters are step length, step height, heading direction and toe-out. A "sagittal elevation" motion algorithm is used to specify the foot-floor interaction model using sagittal elevation angle for realistic contact.

[Hodgins, 1996] and [Wooten and Hodgins, 1996] present human running, respectively human diving animation methods based on the same principles. They use mass data and moments of inertia in combination with a commercially available motion equations generating package. The movements are separated in phases, and a state machine is used to control the action selection mechanism.

Another category of controllers span from the field of artificial intelligence and neural networks (NN). [Grzeszczuk et al., 1998] and

Procedural animation (based on data)	Physical simulation (control systems)
- Fourier transformations to extract and alter qualitative features - image and signal processing techniques: filtering motion features - signal processing techniques for emotional transforms - learning and applying motion styles through HMMs and analogy - LMA properties based alterations (for effort and shape) - knowledge-driven alterations for specific subdomain - noise functions for qualitative variance - constraints and displacement maps - constraints based retargeting with spacetime approach - motion warping (time based alteration) - simulated annealing (scaling & fine-tuning) - path-based motion alteration	- dynamic Newton equations system - mass data and moments of inertia with motion equations - inverse dynamics with energy minimisation constraints - decentralised, agent-based approach (forces = agents) - inverse kinetics (IK with mass) - muscles as motion actuators with sensor feedback - proportional-derivative (damped spring) controllers (PD) - PD servos for dynamic simulation of joint torques - Genetic Programming method to create PD controllers - closed loop method for balancing in open loop walking - dynamics system with feedback control method - simplified inverse dynamics with mass (Newton-Euler eq.) - iterative numerical integration for IK - IK constraints with forward kinematics and feedback - inverting hierarchies (articulations) - "sagittal elevation" resolution method for IK - direct manipulation - Neural Networks based learning from examples/feedback - Machine Learning (adaptive state machines)
- motion prototypes annotated with behavioural tags - combination based on height (foot elevation) criteria - 2nd order derivative to extract and combine basic motions - priority (importance) levels and weights based combination - segmenting data and recombining with transitional motions - combining non-overlapping data based on spacetime constraints - timewarping to match motion sequences - fade-in, fade-out functions - curve blending functions - interpolation between motion sequences - grouping mutually exclusive motions for faster combination	

Table 2.1: Classification dimensions for different animation methods

[Grzeszczuk and Terzopoulos, 1995] use NNs for learning motion (from examples or positioning feedback), but apply it only to simple articulations. [Bellan et al., 1998] suggests a data-driven NN approach that is aimed to grasp the essence and variability of human movement.

The approach taken in [Gritz, 1999] to achieve physical simulation with the aid of genetic programming (GP) to produce controller programs is similar. Key marks (a subset of key frames and tracks) are used to specify the goals of an animation, compared to spline control points which have uncertainty incorporated through values and time.

A physics based animation which uses adaptive state machines as controllers is shown in [Faloutsos et al., 2001]. The state machines adapt by machine learning techniques, and balancing, pre and post states are used for optimal control. The technique is presented as a general, longer-term solution than data blending and warping, but it is rather a question of physical simulation versus data-oriented approach.

[Funge et al., 1999] treats high-level cognitive modelling for action planning. Domain knowledge is represented by preconditions, actions and effects, while characters are directed by goals. The paper defines an animation method hierarchy (pyramid hierarchy) as geometric, kinematic, physical, behavioural, cognitive modelling representing higher and higher levels of animation techniques. They use cognitive modelling to direct behavioural animation.

[Huang, 1997] presents another type of domain knowledge. Virtual touch sensors are used in the hands of virtual humans to simulate grasping behaviour. Small spheres are used as collision sensors in combination with grasping knowledge (deducted from the grasped object's properties) for physically correct grasping.

Procedural animation

We analyse the procedural animation methods from two aspects. First, we take a look at the combination techniques for motion data (left column, bottom in table 2.1, techniques which provide the easiest interactivity possibilities by simply reusing a motion library. In addition to combining motion sequences, there is another category of motion generating methods (left column, top) where the aim is to alter the motion data based on properties defined for the character, its mental model, behaviour, etc. The latter methods are generally used together with motion combination techniques, thus this category is possibly less interactive than the motion combining methods category alone but it also provides more qualitative variance in the generated motion sequences.

Combining animation data

Combining animation sequences (motion capture data) can be a tedious task when done by hand. Therefore, automatic motion combination techniques have emerged ranging from simple to complex methods, from fading functions to overlapping and blending techniques.

Common and simple combination methods are the use of fade-in and fade-out functions [Rose et al., 1998] (with additional refinements) [Lee and Shin, 1999] or the motion parameter curve blending method used by [Witkin and Popović, 1995]. The method presented in [Badler et al., 1999a] is

similar where motions are segmented into motion phases that can be named, stored and executed separately, and possibly connected via transitional motions.

[Cassell et al., 1994] and [Cassell et al., 2001] describe motion prototypes, a dictionary of gestures to create animations from text and dialogue theory. The gesture motions are annotated and synchronised for animating dialogues using parallel transition networks (PaT-Nets).

Resolving conflicts

The previously enumerated methods work satisfactory with non-conflicting motion sequences. When the motion data sets try to act upon the same target joints, the results can be unexpected. To cope with this problem, different methods for combining motion data were proposed, based mainly on different weighting and summing techniques.

[Huang, 1997] and [Sannier et al., 1999] discuss priority (importance) weights and weights-based combination of conflicting data applied in initiating and terminating movement phases, with an ease-in and ease-out technique based on cubic step functions, to achieve smoothness next to the meaningful combination of movements.

[Perlin and Goldberg, 1996] uses the traditional animators knowledge: "human motions are created from combinations of temporarily overlapping gestures and stances" but goes one step further and uses groups of mutually exclusive animations which compose levels with different importance weights to achieve next to motion combination a gain in speed by simplifying the search for conflicting motions.

Altering (varying) animation data

If motion sequences are reused frequently, then the repetitions lend a mechanical feeling to the animation. Since motion data (motion libraries) are limited, researchers tried to introduce variations in motion data to achieve non-repetitive movement sequences. This can be achieved in various ways, ranging from noise functions to B-splines and from IK to neural networks.

Qualitative variations

The first set of methods discussed use key-frame data and alter it qualitatively. Their goal is to introduce qualitative variations into the animation data, making non-repetitive reuse of the same data set possible.

[Unuma et al., 1995] applies Fourier transformations to motion data and uses frequency analysis to extract basic factors (like walk) and qualitative factors (like brisk, slow, etc.). Intrapolation and extrapolation in the frequency domain are then used to create new motion types from neutral data and emotional mood variations.

[Witkin and Popović, 1995] suggests motion warping techniques to alter key-frame data. By warping the motion parameter curves to animator-specified constraint key-frames, local variations are introduced in the motion data without changing the fine structure of the original motion. Motion sequences are combined by overlapping and blending the parameter curves.

Similar techniques from the image and signal processing domain is used in [Bruderlin and Williams, 1995] as complementary techniques to key-framing, motion capture, and procedural animation. The techniques are multiresolution

filtering, multitarget interpolation with dynamic timewarping, waveshaping and displacement mapping.

New motion from data

A second set of methods could be defined as methods using motion data as basis for new motion calculations, such as noise functions or knowledge-driven methods, and even physical simulation methods are used sometimes.

[Perlin and Goldberg, 1996] uses noise functions on existing key-frame data in combination with IK-based motion calculations for simulating vivid, natural movements from existing key-frame motion data. [Lee and Shin, 1999] defines constraints on the key-frame motion data, for which B-splines are fitted. These fittings result in hierarchical displacement maps that are used for motion alterations (via IK calculations).

[Bruderlin and Calvert, 1996] presents interactive, knowledge driven altering techniques for running data which are based on empirical (relations between parameters), physical (for body trajectory) and constraint knowledge (for flight/support states and stance/swing phases).

[Chi et al., 2000] discusses LMA (Laban Movement Analysis) properties based movement analysis and alteration. Effort and shape properties are used (the other three properties used by LMA are body, space and relationship) to introduce qualitative differences (mood, internal state) in the key-frame data.

Motion retargeting

A different set of methods concentrate on motion retargeting. The motion data is altered to be used for articulated characters of different sizes and types.

[Bindiganavale, 2000] inputs parameters into the motion capture data and applies them to other target articulations. The second-order derivative is used to search for significant changes in motion, by analysing zero-crossings to extract basic action segments, which are later interpolated using cubic splines to yield new motion sequences.

A two-step motion adapting method for new characters is described in [Hodgins and Pollard, 1997]. The first step consists of a geometric scaling enhanced with mass scaling and in a second step the parameters are fine-tuned using a "simulated annealing" technique (progressive adaptation of multivariate functions towards target states).

The method presented in [Gleicher, 2001] and [Gleicher, 1998] differs from the previous ones in its scope by the use of different paths to retarget the motion data. Paths represented as B-spline curves are altered (they are editable) and foot constraints are used to eliminate foot skating.

2.3.2 Behaviour

Semantic analysis and semantic generation of movement are important components in the behaviour (brain) specification for animated characters as stated in [Badler et al., 1999a] and [Badler et al., 1999b], where they use this semantics as a method to alter movement sequences. [Bruderlin and Calvert, 1996] uses a high-level motion control system based on dynamic parameters in combination with empirical, physical as well as constraints knowledge to procedurally generate walking/running animation data. Others like [Huang, 1997] use object grasping knowledge for sensor-based grasping simulation of different types of objects in virtual environments.

Classical cartoon creators use different animation sequences to emphasise the movement semantics in their cartoon characters. Computer character animators can use the same techniques when creating key-frame animations [Lasseter, 1987], to achieve expressiveness and believable behaviour. Film theory concepts could be used for animations [Lu and Zhang, 2002], mainly for staging, viewpoint animation, etc.

On the other hand, there are exact, medical and psychological-social measurements for the semantics of human movement. Regarding medical measurements, [Inman et al., 1981] presents a series of walking characteristics, while [Whittle, 1995] discusses phases of walking, forces and moments, muscle activity resulting from human gait analysis. Psychological textbooks analyse the different human poses, assigning meanings to them. [McNeill, 1992] analyses the meaning of gestures and the gesturing behaviour of humans during dialogs. Similarly, the human facial movements are analysed in [Ekman and Friesen, 1978] for extracting the expression of different psychological effects such as mood or mind state from face features.

These results found their way into computer animation. [Cassell et al., 1994] and [Cassell et al., 2001] incorporate gesture and dialogue semantics to generate the gesturing behaviour of autonomous characters. [Waters, 1987], [Lee et al., 1995], [Cassell et al., 1994] and other facial animation systems rely on human facial expression analysis for reproducing facial expressions.

Behavioural animation is also gaining momentum. Simple animal behaviours [Tu, 1999], complex human dialogue behaviours [Cassell et al., 2001] and some of the different motion alteration methods from Section 2.3.1.2 use moods, emotions to influence the created animation, making it more expressive, natural and meaningful.

2.3.3 Agents

Agenthood is defined in [Wooldridge and Ciancarini, 2000] as a system with the following properties:

- autonomy (encapsulated, private state),
- reactivity (situated in an environment, perceives environment and able to respond to changes),
- pro-activeness (goal-directed behaviour by taking the initiative),
- social ability (interaction using agent communication languages, cooperation).

Although there is no real, universally accepted definition for agenthood, there is a consensus that agent-oriented programming is an extension of object-orientation, including the above-mentioned and other "intelligent" properties which also act as hooks for interoperability. In case of VR systems, the emphasis is in most cases on automatic or autonomous embodied entities like avatars, agents, actors, etc. In the following we will enumerate a number of agent approaches.

[Badler et al., 1999a] and [Bindiganavale et al., 2000] describe parallel finite-state machine controllers called PaT-Nets (Parallel Transition Networks) used

together with additional parameters and information coded in a higher-level representation called PAR (Parameterised Action Representation). A PAR is defined as the description of a single action in terms of objects, agent, applicability conditions, preparatory specifications, execution steps, manner, termination conditions and post assertions. This is used as a sense-control-act architecture to animate "smart avatars", avatars whose reactive behaviour is manipulated in real-time to be locally adaptive. The architecture integrates autonomy control levels (to optimise lower-level reactivity or plan higher-level complex tasks), gesture control based on a object-specific relational table and non-verbal communication, attention control directed by visual sensing requirements and locomotion with anticipation for favourable positioning in interaction scenarios.

A personalised agent architecture is suggested in [Crabtree et al., 1998], where the agents act as personal assistants to the user based on a user profile. The agents present capabilities for information sharing, cooperation, for information privacy and are even capable to adapt by learning user habits and adjusting their profile.

[Webber, 1998] defines an agent architecture based on sense-control-act loops, where after the environment condition is registered and judgements are made, an appropriate action is effectuated. It uses also timed condition checking for querying environment states.

A similar, but more common architecture is the belief-desire-action architecture used in [Tu, 1999]. It treats the environment stimuli (beliefs) and internal states (desires) according to weighted priorities when making action decisions. The architecture is used to simulate both virtual animals and virtual humans.

A common concern in case of embodied agents is the task planning and path planning aspect of a system. Classical algorithms such as the A* search (widely used search algorithm), space subdivision techniques and other approaches are used as well.

The classical path planning algorithm A* is used in [James J. Kuffner, 1998]. It uses an orthogonal projection of the environment to a 2D plane, divided into cells. Collision detection and path planning is done based on the occupied/free cells of the plane. The free cells of the plane are transformed into a graph, for which the A* search algorithm is applied for checking free space availability. Variations exist on this method, for instance boundary cell skipping for collision avoidance or multilevel cell maps [Bandi and Thalmann, 1998].

[Thalmann et al., 1997] proposes an octree division of free and occupied spaces in a virtual environment to notate free space for path planning, incorporating dynamic possibilities for updating octree data due to animations or possibilities for adding or removing objects, in case of content change.

An autonomous planning method is shown in [Noser and Thalmann, 1996] which is based on a reactive and timed behavioural L-system that handles visual and tactile virtual sensors. The values of the sensors are used in production rules for determining the behaviour of the characters.

[Blumberg and Galyean, 1995] presents an autonomous virtual human directing system that is based on synthetic vision sensor (the viewpoint of the virtual human). The autonomous humans are directed by a behavioural system at motivational (disregarding the behaviour system), task (if the behavioural system allows it) and direct motor levels (influence the modality of the current action).

A reactive path planning method based on the possible successor state axiom is described in [Funge et al., 1999] instead of the common effect axiom (it uses the "what the world can turn into" metaphor). Their characters are directed by goals and domain knowledge, incorporating actions, preconditions and effects.

2.3.4 Complex system editors

A complex, high-level animation interface is presented in [Sannier et al., 1998] which allows the directing of multiple actors and cameras in real-time. Keywords are high-level controls, predefining actions, while text-based sentences are used for high-level actions. There are also tools for pre-programming actions, since there are many virtual humans to control. A scripting tool records all actions played by an actor and plays it back as requested. Actors can be programmed one by one.

[Gobbetti and Balaguer, 1995] uses a network of interrelated objects and constraints to visually edit the properties of virtual environments, including animation. The interesting approach is the fact that the edited environment and the controls are all three-dimensional, making the editing interface integrated into the environment.

With such complex systems it is possible to achieve even automatically the animation for text to scene conversion systems such as described in [Coyne and Sproat, 2001] or for text to animation systems as it is already being done in [Cassell et al., 2001].

2.3.5 Summary

We presented a review of the computer animation literature, mainly concentrating on articulated characters and at least to some degree on interactivity or real time simulation. Advances of different techniques such as key-frame, motion capture (also known as mocap), dynamics, inverse kinematics (IK), controller systems, and even neural networks were discussed.

We tried to analyse these methods from different aspects: semantics, data input, animation types, generation, combination and variation of animations, target type, and even the path planning perspective have been taken into account. By analysing the differences between these methods, we can specify our requirements for an animation system, capable of handling the animation of 3D characters in interactive environments.

This research will be integrated into our work via an animation editor, which is being currently developed. The animation editor completes a geometry editor [Kiss, 2001b] and an H-anim based bone hierarchy editor [Kiss, 2002], also described in Chapter 4 to provide a system capable of producing the avatar or agent bodies needed to populate virtual environments.

Since the aim is not just to provide a simple animation editor but a whole animation system which can be used for specifying basic movements and creating combinations of movements, the goal of this report was to find previous research dealing with animation in general, with animation combination, and qualitative assessment of animations in particular. Interactivity is the key requirement, which asks for speed in the system, and more importantly, for animation generation and combination possibilities that result in compelling animation se-

quences. To this extent, our approach towards animation is based on key-frame animation data with segmentation and reuse methods.

Chapter 3

VR and geometric modelling

This chapter expands upon the theories presented in the previous chapter, especially the theories relating to 3D modelling (classical, free form deformation - FFD, shaping tool approaches), virtual reality and most specifically virtual environments. These latter include knowledge about the environment, architectures (technologies) or 3D user interfaces.

For virtual reality, [Coomans and Timmermanns, 1997] presents a taxonomy of VR and enumerates a number of VR-related definitions. VR is regarded as a promising solution to visualisation and natural interaction. We look at these definitions rather as components or properties that can be used not to classify, but to specify the areas that a virtual environment is covering and the degrees of usability, efficiency, comfort, etc. The suggested properties or dimensions of a VE are:

- interaction: must be natural to the user, characterises comfort, ease of use, effectiveness;
- immersion: sensory immersion, presence feeling (full body/partial), characterises the visualisation hardware and the joy quality of an environment;
- visualisation: making non-visual data visually observable, shows levels of utility;
- real space: on-line communication and information retrieval, an application area for knowledge and socialising;
- autonomous agents: the computer program takes initiative, actions, providing qualitative services;
- simulation: visual, acoustic, haptic, scientific simulation, virtual prototyping are areas of deployment for virtual systems.

3.1 3D graphics and geometric modelling

An important aspect of our work is the specification of the geometrical modelling characteristics. In the following, we present a review of the classical, more

common modelling approaches as well as the state of the art of current systems. A comparison of different modelling systems is presented in Chapter 5.

3.1.1 Basics

For 3D graphics in general, we can talk about *primitives* that build up a shape, the *complexity* of the shape, and the different *purposes* the shapes are built for. The classical approach to primitives is based on vertices (coordinates, points) that are indexed, ordered to form edges (or lines) and polygons. A vertex specifies a distinct coordinate in 3D space, an edge connects two of these coordinates, while a polygon is a surface defined by three or more coordinates situated on the same plane. Textbooks such as [Maestri, 1999] provide a general description of these primitives and the operations that are usually performed on these primitives.

Several complex types of primitives exist which are used for parametric surfaces. Control points, knots and weights define the shape of Bézier, NURBS (also proposed for VRML in [Grahn et al., 2000], part of the X3D standard), Lattice (proposed for VRML in [Wakita et al., 2000], part of the X3D extended profiles) and other types of high-level, computationally determined surfaces. There are also other types of models that are completely out of the scope of this work, such as the point cloud or voxel-oriented graphics models.

There are also a number of other, simpler primitives that a modeller can use. One such construction that could qualify as a primitive is the so-called "closed loop", which consists of a number of continuous and closed ring-like vertices and edges [Maestri, 1999].

The complexity of shapes is determined by the number of participating vertices. In case of computational surfaces, this number is always changing or could be set according to user preferences, since the actual coordinates are calculated with a certain subdivision level. Subdivision means the recursive calculation of surface coordinates, where at each iteration, a number of neighbour vertices and their parameters specify the location of new vertices, introducing a smoothing effect into the mesh. These surfaces are classified mostly by their type, but also by their order, which gives the order of the equations that define the surface.

The purpose of models built for 3D graphics and especially for VEs consists of environment or scene objects, characters or special purpose objects that perform different functions, behave autonomously, etc. Typically, in these cases the model geometry is linked with different behaviour models.

3.1.2 Modelling

We look at the modelling types that are available for 3D objects. Traditionally, the first modelling approach is the classical, vertex, edge and polygon based modelling. This is also the lowest level of graphics visualisation systems: thus the interest in this type of modelling is understandable. Then there are a number of higher level modelling approaches for polygon-based and computational surface based objects as well, which at the visualisation level also resolve to vertex, edge and polygon representations.

3.1.2.1 Classical modelling

The most basic components of 3D models are the vertices. They are defined by a coordinate in 3D space. Positional manipulation of vertices, edges and polygons are therefore simple displacements of single or multiple coordinate values. In the following we concentrate on editing operations that handle the complexity of meshes constructed by polygons. The mesh complexity handling operations that are defined for vertices and are depicted in Figure 3.1 are:

- Extrude means to pull the selected vertex outward. As a result, the same number of polygons are added to the edited mesh as the number of polygons that the vertex was participating in before (connecting edges are cut, usually at the middle point).
- Bevel means to push the vertex inside. As a result, the vertex is expanded to a polygon along the connecting edges and the vertex is deleted (like cutting out a corner of a cube).

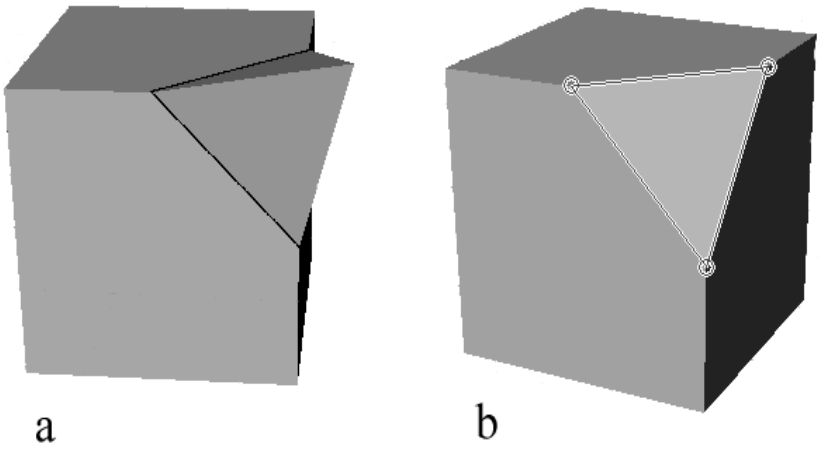


Figure 3.1: Extrude and bevel operations for vertices. The starting shape in all operations is a cube.

Two connecting vertices create an edge. The following mesh complexity operations can be effectuated on edges:

- Extrude is an operation which creates new polygons along the normal of the edge. The connecting edges are cut and reconnected with the selected edge, now in a slightly outward position.
- Bevel is an operation which creates a new face (4 points polygon) which connects the two polygons that the edge was participating in. Two new edges are created instead of the beveled edge by cutting the edges that are connected to the selected edge.

- Collapse is an operation which creates a single vertex from the edge. It is sort of a dimension shift: the edge becomes one vertex, and the triangle faces the edge is participating in become edges themselves (polygons with higher vertex count are only simplified by one vertex).

- Cut/connect is an operation which provides a mesh refinement method used on one or more, usually parallel edges to create meshes with higher polygon count, suitable for further manipulation.

Figure 3.2 visualises what happens in these operations:

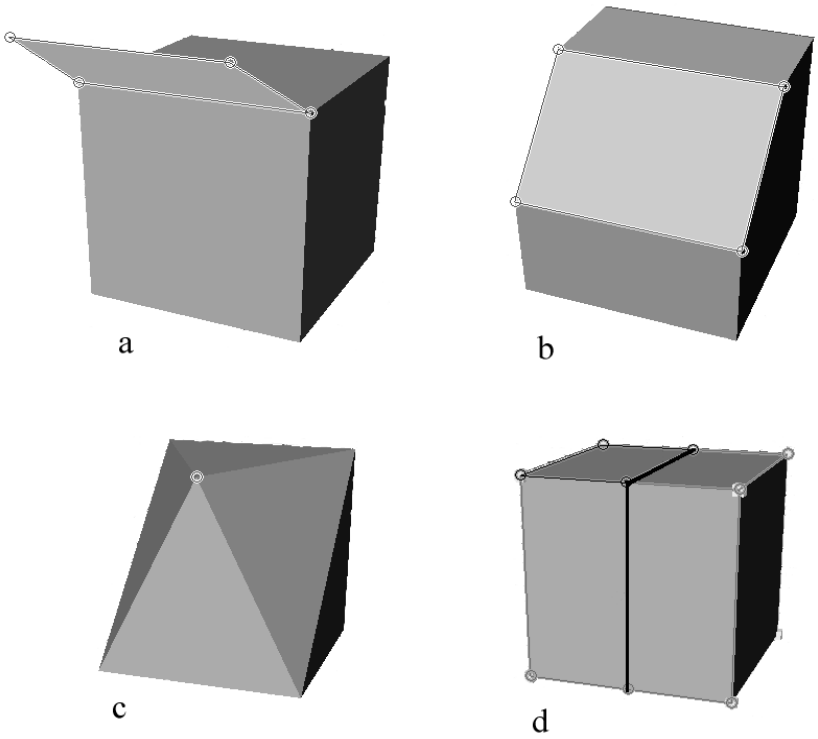


Figure 3.2: a) Extrude, b) bevel, c) collapse, d) cut/connect operations for single edges or several edges (all horizontal edges in the cut/connect case). The starting shape in all operations is a cube.

Three or more edges (containing three or more distinct vertices) make up a polygon. Polygons with more than three vertices must have the vertices on the same plane in space. The polygon-oriented mesh complexity operations that are sketched in Figure 3.3 are as follows:

- Extrude is an operation which means pulling the selected polygon outwards. The polygon edges are replicated, with original and new edges constructing new faces (4 point polygons).

- Bevel is an operation which means pulling the selected polygon outwards while scaling it down. The edges, just like in the case of extrude, will be replicated and will form new 4-sided polygons.
- Collapse of a polygon means the reduction of the polygon to a single vertex. This results in a degradation of vertex number in all connecting polygons, since all the polygon edges are replaced by one (the same) vertex.

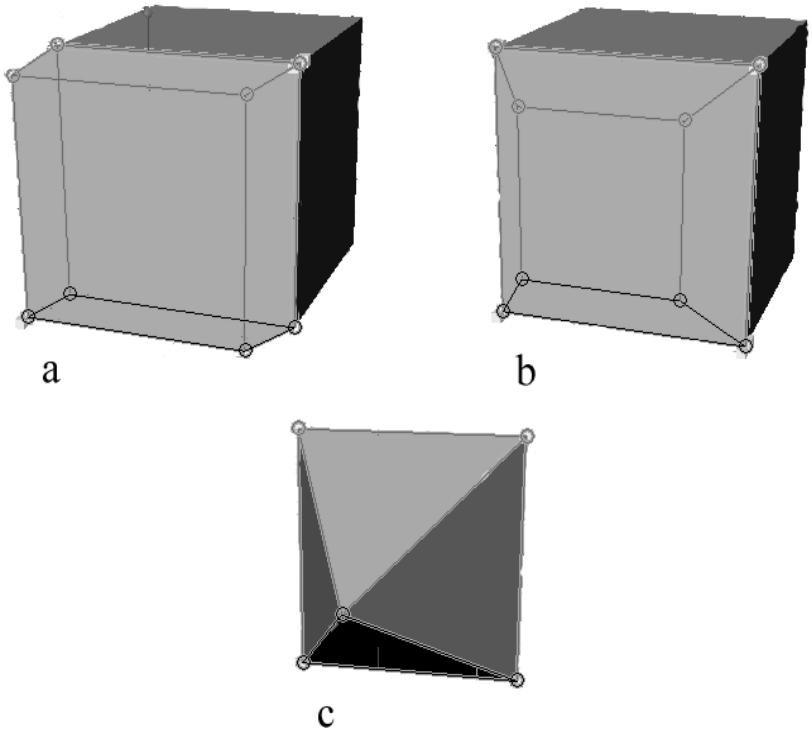


Figure 3.3: Polygon operations: a) extrude, b) bevel and c) collapse . The starting shape in all operations is a cube.

3.1.2.2 High-level positioning and inter-shape operations

Barrilleaux [Barrilleaux, 2000] provides a classification of visual aid and tool methods that provide assistance in shape positioning (arranging objects in an environment). Most of these methods are feedback methods that provide positioning information in an easily perceivable way for the user. These feedback types are:

- Feelers are visual aid tools (visualisations) that show the relations of the selected object to the target object. They can be passive and active (real-time geometric processing) feelers.

- Projections are feelers that project shadows on different objects to show proximity (position in space) for a greater positional awareness.
- Skirts are projections in the form of a translucent box under or besides an object or can serve as an idiom for lifted objects, where the intersection with other objects shows object proximity.
- Outlines provide a method of enclosing or delineating an object, component or group. They can have different functionalities:
  - as feelers, they show the spatial extent of objects,
  - as control feedback entities, they indicate selection or mouse-over,
  - as control widgets, they serve as drag handles.
- Tic marks are small symbols that are marking the spatial position, alignment of objects or they are marking graduating ranges for different measures. They can have different functionalities:
  - as feelers, they indicate proximity or alignment,
  - as control feedback entities, they indicate graduations over a range of data.
- Rulers are tools that provide quantitative measurement for object positioning. They can function as or in combination with feelers, as descriptive feedback, as active or passive measurement tool.
- Snaps provide a relational and control feedback hybrid tool for the object locations where an attachment can occur. Snaps can have properties like:
  - they can have a type for selective matching,
  - they can have an orientation for selective alignment.

The last type of feedback applies also to the control action taxonomy. This means that it is a technique that not only provides visual feedback, but also helps in positioning. The different implicit or explicit snap types are categorised according to the connection type between the objects:

- Snapping means a not permanent positioning of objects while the objects are touching.
- Gluing means a rigid physical attachment of objects while the objects are touching.
- Sticking means a non-rigid physical attachment of objects while the objects are touching, where the object may slide and twist.
- Linking means a logical attachment of objects while the objects are not touching, where the topology is maintained with object manipulation.
- Anchoring means fixing object geometry in space. This control action is used generally with sticky attachments.

These methods and control actions can be used to transform geometry (referred to in [Barrilleaux, 2000] as "dumb objects") into "smart objects" where the latter is a parametrically adjustable 3D entity that can have combinable subparts, it can change properties, etc.

[Bier, 1986] describes an object positioning method based on jacks and skittlers. It treats these entities as widgets integrated into the editing environment which represent Cartesian coordinate frames consisting of anchors and manipulation end conditions, respectively 3D cursors that are used to place the Cartesian coordinate frames (which describe local coordinate systems). These jacks and skittlers are used to specify features of the edited objects for different manipulations, not just geometrical, but also snap-coordinates. The feature selection mechanism is capable of following object surface, pointing to culled back-faces, or using the jack geometry (3 axes) as manipulative orientations (as axes and planes handled by 2D input devices).

3.1.2.3 Sketch-based modelling

There are approaches to higher-level modelling that concentrate on conveying two-dimensional pencil strokes into 3D objects or features of 3D objects. The Teddy system [Igarashi et al., 1999] and other, more recent systems such as described in [Karpenko et al., 2002] use this classical drawing method and gesture rules as the input modality for creating 3D geometries. With this sort of systems it is possible to have 2D looking graphics that are in fact three-dimensionally manipulable, but they offer usually methods to convert these objects into a more common 3D format. The advantage of such systems is the two-dimensional input modality used and the natural simplicity of the interface. However, it has also disadvantages: there is no control over the mesh complexity and lacks precision by not providing detail manipulation possibilities.

3.1.2.4 Higher-level modelling

FFD (Free Form Deformation) is a modelling approach which uses a control lattice to deform the space the model is in. It is a general modelling tool in the sense that it does not concern itself with the model placed inside the deformed space, it can use anything the user wants to deform. The whole approach is based on a space deformation formula given in [Sederberg and Parry, 1986] and refined in [Moccozet and Thalmann, 1997] for greater localised control and even animation.

An alternative to FFD is the combination of geometry models with physics-based deformations. Some prefer this method because it shows a more natural approach that is more easily understood by the users of a modelling system.

The systems described in [Sannier et al., 1998], [Sannier et al., 1998] and [Moccozet and Magnenat-Thalmann, 1997] use FFD for the simulation of the muscle on a virtual actor's face or the simulation of muscles within hand movements. In the latter case, the general FFD method is combined also with real-life hand movement observations for naturalness.

[Grahn et al., 2000] presents an implementation for the inclusion of NURBS support into VRML. This way, a higher-level surface modelling approach can be added to VRML VEs, where the surface is defined by a set of control points (sort of attractor points) and with a set of control point weights, which define the

extent of the attraction. Similar is the "lattice" initiative [Wakita et al., 2000], which is a formula-intensive, weightless but simple control point assemble for high level surfaces. With the manipulation of a single control point, a whole section of the target surface can be altered.

3.1.2.5 Shaping tools

There are approaches to modelling taken from real life in the form of carving, clay modelling, which all have in common the concept of a physical mass that is deformed or otherwise modified [McDonnell et al., 2001] (clay) [Keefe et al., 2001] (3D paint). They also tend to use haptic interfaces, which are natural extensions to these systems, providing tactile feedback for the modelled physical mass.

3.1.2.6 Specialised modelling

There are also specialised modelling tools, where the designers concentrate on a specific sub-domain of the possible 3D model domain, which usually incorporates shortened and easier modelling tools which span from the properties of the model domain it serves. Such systems are concentrating on L-systems [Lindenmayer and Prusinkiewicz, 1990] (with skeleton-like branches modelled algorithmically), H-anim compliant characters [HAWG, 1999], revolution surfaces, the Escher-inspired tileable sphere models of [Yen and Séquin, 2001], 3D paintings [Keefe et al., 2001], or even semi-automatic generation of head models from photographs and general meshes [Lee et al., 1998].

3.1.2.7 Peripherals

The most common modelling devices for 3D graphics are the mouse, joystick, tablet, etc. which use a 2D input metaphor. This is partially the reason why graphics modelling applications use different views of the 3D space to provide quick 2D-wise manipulation possibilities. However, nowadays these devices are extended to include the 3rd dimension: so we have different versions of 3D mouses, joysticks and other, similar devices.

Besides the more traditional and common 3D object manipulation hardware, new interaction devices have been designed: gloves, digitising tools (lasers), force feedback devices. The latter ones are pedals, steer wheels, joysticks, mainly used in the entertainment industry, but there are pen, gloves and other metaphor based interaction devices that also provide tactile feedback enabled interaction possibilities.

With the advent of the new tablet-like home computers, there may be a new opportunity for tablet-based interaction methods, with the difference of manipulating the objects directly on the screen, and not on a separate digitising tool. This allows for a very tight integration of the interaction and feedback methods.

3.1.3 H-anim specification

The H-anim Working Group of the Web3D Consortium created the H-anim specification for the purpose of a standardised animation approach of humanoid

figures. The specification is based on anthropological data and statistical measurements of humans, providing a physiologically based approach and a data set containing average human joint position values.

Although it is a) created under the auspices of the Web3D consortium responsible for the specification of the VRML language and b) contains examples with VRML notation, the specification is a general description that is implementable in different scene graph technologies. This approach can be used to create the basis for animation in case of 3D characters, especially characters used in VR systems or other scenegraphs.

The advantages that this specification presents are the animation compatibility that arises from the standard naming conventions and the standardised joint hierarchy types. Using these, it is theoretically possible to combine different animations that are targeted to standard joints with different characters that follow the same standard joint naming conventions. In reality, however, there are a number of possibilities with differing animation and hierarchy complexity levels, which make necessary some higher-level to lower-level mapping scheme (or even vice-versa) in case of animation data.

The internal components (or nodes, if we take in account the VRML implementation examples of the specification) of the H-anim specification are:

- Joints: these are the components that drive the animation. There is a root Joint that controls the displacement parameters of a character while internal and leaf Joints control the limb and body shapes via rotational parameters. The Joints are specified based on human anatomy, representing bone connections. They are defined with different degrees of freedom, based on anatomical measurements of real bone rotation constraints. Even standard Joint positions are suggested, based also on anatomical measurements, however these are only suggestions or data examples for a simple implementation, and content authors are free to use different measurements if needed.

- Segments define the geometry attached to Joints. In general graphics editing systems, animation is realised with the use of bones and attaching skin. In this case however, the Joint component represents bone connections and not actual bones, but the hierarchy properties allow to consider it as representing the bone itself, thus a geometry can be assigned as child to a Joint, which is then handled automatically by the Joint animating data.

- Displacer components define geometry (skin) deformation methods for more realistic animations which is especially usable for facial animations, since the face is an important animation component which lacks the presence of Joints and Joint-based animation possibilities. This interpolation and time-based method targets the coordinates of a Segment geometry.

- Sites are features of the H-anim hierarchies that define attachment points on the character geometry. These attachment points can be used for glasses, hats, tie and other accessories. These coordinates add the possibility of easy handling of accessories which can also be interchangeable with respect to accessories or even characters.

The specification leaves room for some variations. There is a set of Joints that represents the minimal requirements for H-anim compliance, which mainly reduces the number of spine Joints. There are also three different LODs (Levels Of Detail) specified, with varying Joint complexity. These versions can make the interchangeability of hierarchies and animations a bit problematic.

The hierarchical properties of the H-anim standard are:

- Joints can have (except the root) only one parent Joint, and several child Joints. Joints have also a Segment child. Joint operations are affecting the Joint, all child Joints, the Segment child and all their descendants.
- Segments have a single Joint parent, which handles their position in space, and a number of Displacer and Site children, which add deformation-type animation possibilities and provide attachment coordinates for accessories.
- Displacers have a single Segment parent.
- Sites have also a single Segment as parent.

The H-anim specification permits different coordinate grouping solutions. The stored geometry coordinates can be stored separately, locally in each Segment entity, or it can be stored in a global Coordinate node. The coordinate values are in both cases the same, only the indexing of the geometry surface triangles is different. The global storing scheme could provide extra consistency to a character with seamless skin geometry by the use of shared coordinates for different, touching Segment geometries. In this case, when the coordinate is updated, both geometries will contain the same vertex, assuring that the character remains seamless.

The new version of the H-anim specification (named H-anim 2001, while the previous one was version 1.1) specifies a seamless animation and deformation engine that makes use of a global coordinate storing scheme. With additional components, the newly proposed standard tries to handle the geometry collision and interpenetration automatically, relieving the content creator from this tedious task.

3.1.4 3D characters

The 3D character geometries are a special group of 3D models (objects). They are used in virtual environments as user-representing entities, scripted, environment-specific entities or as autonomous entities that perform certain tasks, sometimes called actors or agents. In these cases, we talk about embodiedness, where the term is applicable to avatars and agents alike. These characters are usually humans or resemble humans in their components, but this is not always a requirement. Characters that are built according to a hierarchy are called articulated characters. The most common types of animation systems tend to use articulated characters for their ease of manipulation, since hierarchies propagate the effectuated actions to child components.

[Badler et al., 1999a] proposes a classification system for 3D characters that is based on a number of different dimensions. These dimensions provide a classification system for 3D character properties like realism, believability, liveliness, etc.:

- Appearance dimension. Characters can vary from cartoon-like shapes to physiological models.
- Function dimension. Characters are classified based on their resemblance to unlimited cartoon actions or to human limitations.
- Time dimension. Characters can be generated off-line or on-line.

Sometimes the character geometries are not modelled directly. [Fua et al., 1998] describes a method where not the character skin (surface) is modelled, but the muscle-simulating and deforming "metaballs" are placed on the skeleton which represent the human body volume, and the actual skin is calculated after animation and the metaball deformation is effectuated. The drawback of this method is that it is a computationally intensive method, even in the simplest case when only one metaball is used for each of the skeleton segments. [Moccozet and Magnenat-Thalmann, 1997] and [Moccozet and Thalmann, 1997] take this approach and refine it to the hand level, along with other techniques.

Another higher-level character modelling approach, although only for the head, is presented in [Lee et al., 1998]. It uses a facial reconstruction model based on two images of a real person, and on a generic head model. The images provide the possibility to extract the facial features, and the general head model is deformed to match these features. The deformed model is then textured with the images of the real person, increasing the model's realism.

[Babski and Thalmann, 1999] presents seamless 3D characters and methods to adapt the shape according to animations. A contour ring based approach is used for the characters, making the handling of deformations easier. A similar approach of mesh deformation is applied in the skin deformation engine for seamless characters of [Smith, 2000], included in the H-anim 2001 specification [HAWG, 2001].

Sometimes the modelling process needs to specify the mass, limits, resistance of the articulated characters and their components (Segments and Joints), used later for physics-based animations like spring-mass systems, IK (Inverse Kinematics), etc. As a hybrid approach, edges can also be regarded as dampened springs, providing a physics-based geometry deformation model [Mealing, 1992].

Examples of characters modelled by different research initiatives are:

- Seamless human shape described in [Babski and Thalmann, 1999].
- Fish shape for the simulation system of [Tu, 1999].
- 3D skeleton of different articulations (17 Joints with 30 DOFs, 14 Joints with 15 Segments and 32 DOF) in the sport related systems of [Hodgins, 1996], [Wooten and Hodgins, 1996] and [Hodgins et al., 1995].

For creating new characters, some commercial systems use pre-built body parts that can be assembled together and eventually scaled to provide variations in size. Similarly, [Hodgins and Pollard, 1997] uses a method of specifying scaling properties for different character types (children, gender switch, alien) to create new characters from existing ones. In the latter case however, the scaling properties are specified according to statistical and other, artificially created

character data values, providing a natural approach over the common geometrical scale method. The character morphing approach of [Alexa et al., 2000] or [Sloan et al., 2001] could be used to acquire the intermediate states of the morphing process between two characters, resulting in new, possibly "genetically mixed" child characters (if scaled properly).

3.1.5 Summary

In this section we reviewed the main graphics modelling approaches used in the industry as well as a few of the research approaches for modelling, usually high-level manipulation possibilities. Our approach to these components, regarding the common and differing approaches is described below.

With regard to mesh modelling, in our approach we use the same graphics elements (vertices, edges and polygons), with the difference of collecting the most basic vertex elements into different groups. We use regular meshes for our modelling needs, which provide interesting properties: they define edge collections as rows (rings if the mesh is closed, and generally it is) and columns that we use as basis for our non-vertex (higher-level) manipulation options. The ring concept is similar to the edge loop and contour concepts described in [Maestri, 1999] and [Babski and Thalmann, 1999], respectively.

When it comes to mesh manipulation tools, we distinguish between two approaches. One approach is the usage of higher-level manipulation concepts for complex meshes. In this case the user has no detailed control over the mesh (always manipulating a multitude of vertices) or even the mesh complexity is being handled automatically. A contrary approach is concentrating on simpler models, for which a detailed and precise manipulation is possible through the direct manipulation of the graphics elements. These lower-level approaches concentrate on selected vertices (single, edge, polygon or multiple of these), resulting in less efficient manipulation possibilities when compared to higher-level systems. Our method is a hybrid approach which provides a higher-level selection mechanism in combination with lower-level and higher-level manipulation options.

With regard to character bone (hierarchy) modelling, we use an approach by which the geometry is linked to the hierarchy, and therefore positional operations are reflected on the geometry. The geometry itself is softly-linked, meaning that although the whole structure is arranged hierarchically, the hierarchy elements will keep their positions when parent elements are displaced. An example would be the displacement of the knee without displacing the angle and its children components, resulting in a different general shape of the legs (think of bow-leg or cross-leg types). The resulting process allows the local editing of the Joint positions, while providing consistency in the hierarchy.

3.2 Virtual environments

Many academic efforts deal with intelligent virtual environments as their object of research. Besides the higher-level intelligence conveyed through the use of agents, speech and other techniques discussed later in this chapter, there are lower-level approaches also to convey intelligence through compelling content, through realism or believability, through coordination or through user interaction [Álvaro Sánchez et al., 1998]. The content for these types of environments

depends on the type of tasks, type of participants derived from requirement specifications and target group analysis. Several of these VE content enhancements are described below.

Random facial movements are applied in [Lee et al., 1995] and also in [Perlin and Goldberg, 1996] to enhance the "liveliness", the believability of 3D characters within VEs. The latter paper applies randomness by means of noise functions, a technique used previously for compelling texture generation. Similar is the technology used in [Back and Stone, 1999] to provide a multitude of variate sound effects without repetitions. It adapts AL (Artificial Life) techniques to alter the behaviour of sound files (their properties) resulting in rich, non-looping sound effects as ocean waves, water drops, etc.

Sometimes the VEs are enhanced by intelligent video inclusion methods. For instance, [Carraro et al., 1998] shows an approach where a user participates in a bicycle race by video projections of other participants in front of as well as behind the user.

To increase the realism of virtual environments, [Summer et al., 1998] uses a ground simulation model which is capable of simulating interactions with the ground that leaves marks, imprints on the surface: foot steps, bicycle tracks, etc. The method consists of a height field simulation with different compression, material displacement and (high) viscosity values.

Some applications are specifically written for immersive technologies in order to make the experience more compelling. Such an application is the narrative maze of [Roussos and Bizri, 1998] which aims to produce a mystical, mediaeval appearance.

Regarding the display hardware, VEs can be categorised as non-immersive (desktop, personal digital assistant - PDA, other flat displays), immersive (Cave Automatic Virtual Environment - CAVE, 3D workbenches, other 3D displays and screens) or augmented (see-through displays). While in the first two cases the same types of environments can often be visualised but with different user perception modality, the third hardware approach consists of projecting virtual elements into the real world. [Azuma et al., 2001] describes systems that project information on real world objects such as buildings-related data, restaurant menus, etc. These systems are based on a see-through display used in combination with a GPS (Global Positioning System) for user localisation. The system described by [Reijers et al., 2002] projects game opponents (from 1st person shooter games) in a real-world augmented environment for a mixed reality game-play.

Regarding the VE system framework, we can differentiate between single-user and multi-user environments. The multi-user environments could be based on the same server-client architecture as with single-user environments such as in [Goddard and Sunderam, 1999], but they also can be based on distributed platforms (referred to as Distributed Virtual Environments - DVEs), important in ubiquitous setups. A special multi-user environment type is the CVE (Collaborative Virtual Environment) where the system contains logic to handle the environment-user interaction: locking of objects, communicating data (files, notes), etc.

A distributed and at the same time collaborative VE review is given by [Saar, 1999] and [Diehl, 2001], complete with design approaches, protocols that are or can be used. It also describes an architecture and implementation of a smaller scale DVE. Similar is the approach taken in

[Schönhage and Eliëns, 1999] and [Diehl, 2001] with the use of CORBA (Common Object Request Broker Architecture) and mobile object technologies for VRML gadgets and display agents for collaborative and multi-user applications. Another multi-user VE implementation is discussed in LivingSpace [Wray and Belrose, 1999]. This multi-user distributed platform specifies the requirements for distribution as: distribution protocol, updating changes and using dead-reckoning for avatars.

The main concern in MU environments is the synchronised positioning of the different users (e.g. their avatars). [Çapin et al., 1997] compiles an overview of the methods used (prediction methods, error corrections, synchronisations, etc.). Commercial MU environment systems all have their specific avatar handling techniques. A common method used is the "dead-reckoning" which means the anticipation of user movement based on previous speed and orientation, with synchronisation steps (or error correction steps) taken from time to time [Wray and Belrose, 1999], [Çapin et al., 1997].

An application area for multi-user VEs is education and presentation systems. [Fellner and Hopp, 1999] describes such a tele-learning and tele-education system with support for audio-visual communication on multiple channels and equipment with the proper control software.

Commercial multi-user environments include among others the Blaxxun platform [Blaxxun interactive, NAb], ActiveWorlds [ActiveWorlds Inc., NA] or Adobe Atmosphere [Adobe Inc., NA]. These systems use typically proprietary formats, although some of them are based on or even are extensions of open standards like VRML.

[Carson and Clark, 1999] proposes a system for authoring and executing shared VEs with the use of shared behaviours, avatars and shared objects. The system uses a hybrid approach, with a server that locks shared objects, storing persistent changes, and with peer-to-peer communication for updating avatar movements. The behaviours and interactions of the system are specified as a) autonomous behaviours (deterministic, independent behaviours: blinking light, windmill), b) synchronised behaviours (deterministic within a certain time: bouncing ball, re-synchronisation is required from time to time), c) independent user interactions (performed by different users at the same time without any inference, like ringing a bell), and d) shared interaction (only one user at a time can grab and move an object: immediate synchronisation and object locking is used).

An interesting extension to VEs is the inclusion of time-line information into the virtual environment [Luttermann and Grauer, 1999]. This time dimension is used to specify the behaviour of features at a given time and how these features evolve in time. A valid time property specifies the time when an event or a state was valid, in case of time-variant objects (visualised dynamic data). Due to the time dimension, time-navigation can be also performed: forward and backward navigation, speed adjusting.

3.2.1 Avatars

An avatar is by definition the personalisation of a user within a virtual world. Simpler, single user VE approaches do not always present a visual self of the user within an environment by default, while multiuser approaches give an appearance to each user by default.

For single user environments, the presence of an avatar can also prove to be beneficial. Besides being more realistic, the user seeing its own avatar (when tilting, looking in a mirror) enhances the presence experience of the user, especially with immersiveness-oriented hardware such as CAVEs, HMDs or other types of devices [Usoh et al., 1999]. This presence or immersiveness then could prove to be highly efficient with the different fobia-healing VEs that have been built for medical purposes such as height fear, flight fear, etc. [Schuemie and van der Mast, 1999].

3.2.2 Virtual sensors

[Noser and Thalmann, 1996] uses a virtual sensor technique in combination with autonomous actors. It uses a perceptual system consisting of a number of sensors and an organism system that applies rules in function of sensory information. For the perceptual system of autonomous actors, it uses virtual vision, virtual tactile sensors and virtual auditory sensors. [Thalmann et al., 1997] provides a detailed description for the techniques behind the virtual sensors: they use z-buffer information extraction, direct environment information query, proprietary acoustic renderer and sensor points on the hand for tactile information extraction.

[Tu, 1999] integrates the virtual sensor approach in a simulation framework for animals and humans. The virtual sensor used is a visual perception sensor which is simulated by data derived from graphical environment interrogations and from a graphical objects database built from these interrogations. This sensory information is the input to behavioural simulation modules.

An unusual virtual sensor approach is the one taken in [Delgado-Mata and Aylett, 2001]. It is based on the concept of virtual smell, and not without reason: there are devices (computer peripherals) in development that would be capable of producing a wide variety of scents by mixing a number of scent essences, just like colour printers work with a more limited number of basic colour substances.

3.2.3 Environment design

With regard to the design of VEs, we can look at two main characteristics, namely the quantity and the quality of VEs. While the qualitative properties are the more interesting ones, the quantitative considerations, besides the more common world building tools that are polygon-based and thus provide the possibility of building VEs from scratch, include also interesting topics.

Rapid development of VEs is an active topic both in research and commercial projects. [Paoluzzi et al., 1999] provides a geometric programming language to procedurally program VEs from basic building elements. Commercial approaches like I3, ISA, etc. (mainly for home design, furniture) use pre-built components for quick scene assembly, and even open initiatives exist to build extensive texture, sound and object libraries for easier content creation or even to provide a local static cache for quicker download times (as in case of the UniversalMedia initiative).

Regarding the qualitative properties of VEs, the following paragraphs contain a not exhaustive enumeration of these with regard to user experiences in a VE.

VE visualisation

There is an ongoing difference of opinion regarding the VE visualisation properties. The two concepts that are opposite to each other are "realism" and "believability". They are not opposing in the strict sense, since on one hand there are people who argue that realism does not mean necessarily believability [Imbert et al., 1998] and cartoons can deliver the same expressiveness [Nijholt and Hondorp, 2000], and on the other hand there are people who argue that to achieve believability, realism is necessary. What they all have in common is that for believability or realism, life signals are necessary, both for life forms and objects in a VE. A realistic environment is defined in [Bellman and Landauer, 2000] as needing situational realism (for instance collaboration without physical collocation), motivation (users will be more motivated in an engaging, enjoyable environment) and active participation (learning is more powerful when the user is engaged). There exists approaches to rendering engines that are either photo-realistic or engines that are exactly the opposite: cartoon-like, pen-like drawing, etc.

[Kardassevitch et al., 1999] proposes to enhance illumination quality in VEs by using textures that represent light. They describe a method of changing the illumination sources in the graphics pipeline with these multi-resolution light maps and calculating the final texture for display. This is a useful technique in real-time VEs, where the lighting options are limited due to performance requirements.

Entertainment value

[Bellman and Landauer, 2000] compares VEs with MUDs (Multi User Dungeons). It arrives at the conclusion that MUDs are far superior since they are more dynamical, adjustable, with a broader range of actions. Since MUDs are text-based games, this conclusion was expected. Recent developments try to include these rich worlds into visual form [Badler et al., 1999a] and commercial game companies also exploit the concept when creating the more and more popular multi-user on-line gaming environments.

Narratives are a special branch of VEs. Sometimes they are designed to be linear, but with enough care and consideration, they can be transformed into non-linear, compelling experiences such as the narrative environment presented in [Roussos and Bizri, 1998]. Their environment is composed of a number of modules from which only the first and the last has a fixed order, the others being situated at different maze locations, resulting in a very high number of actual paths that the user can take. As an artistic touch, the different modules are accessed by a scene fading method instead of the more common teleport method.

[Bowman and Hodges, 1997b] shows how to enhance the value of VEs by making the environments rich in content. Besides textual, cartographic information, an audio annotation toolset is introduced. The audio annotations can be set up to be triggered by different events (proximity, selection, orientation, time or other events), making the approach a valuable addition to the experience perceived by the user. [Murphy and Pitt, 2001] discusses the use of special, spatialised sound effects instead of plain sound enabled environments for an increased effect in visualising cultural heritage environments.

User preferences
VEs can be designed to adapt themselves as response to user preferences. [Kimoto, 1998] discusses a VE built for image retrieval, where the retrieved data is aligned with the user's position based on retrieval relevance. The same principle is exploited in exhibition-style VEs like virtual museums, where the paintings, pictures are rearranged in function of the user's demands based on multiple criteria searches [Shiode and Kanoshima, 1999]. The virtual tour environment in [Schaerf and Tessicini, 1999] displays a tour that is personalised according to the user's desires for target sites that he/she wishes to visit. Another important role of user preferences is within autonomous agents systems, where the agents act independently based on these preference definitions, which can be pre-set, learned or adapted by the agent AI [Crabtree et al., 1998]. Gender-based differences are also presented in [Cassell and Jenkins, 1998], and although they are analysed in a game playing context, the gender preferences related to game content are applicable also to VEs.

Wayfinding
Wayfinding in VEs is an active research topic aimed at enhancing the user experience. [Deol et al., 1999] suggests considerations for wayfinding that concentrate on the proper annotation, labelling and other techniques in the VEs that are supposed to convey information, or to signal to the user. [Elvins et al., 1997] describes an approach to enhanced wayfinding based on 3D landmarks called "worldlets". These landmarks capture the graphics in a frustum or spherical manner at viewpoints. Users then have the possibility to graphically recognise and to remember the viewpoints (it is an alternative to the standard viewpoint technique) based on these graphical representations acquired from the rendering pipeline. Navigation techniques are also analysed in [Usoh et al., 1999] to conclude that presence and wayfinding experiences are greatly increased with physical walking techniques and self (avatar) visualisation.

LOD technique
The LOD (Level Of Detail) technique is used in VEs to reduce the complexity of objects that are not in the close proximity of the user. In this way, complex objects can gradually be replaced with objects that look approximately the same from distance, and are constructed in a simpler, less resource-consuming manner. Some approaches even use pictures for lower levels of detail (an approach also used in the "impostor" technique of [Aubel et al., 1998]), reducing the polygon count to a minimum, but adding a texture component. Yet other approaches like [Carlson and Hodgins, 1997] give LOD techniques to manage system resources in complex animation scenarios: if the resource level is low, the level of animation detail for further situated animated entities are reduced for a more consistent frame rate by replacing the animation simulation method with a quicker, less precise one. [Perbet and Cani, 2001] proposes the LOD technique to specify 3D volumetric textures and 2D textures in large scale prairies simulation (ground and vegetation) for a compelling visualisation.

3.2.4 Intelligent objects

Elements of VEs can be enhanced with special information that helps the exploration, navigation or interactive processes the user effectuates in the environ-

ment. [Dautenhahn, 1998] describes for instance special "keepsake" objects in VEs which are database starting points for stories, memories. These objects or other participants (relatives, friends) are meaningful for the user, and present a way of storing autobiographical moments: they are a medium of narrative experiences. A network of stories and memories is able to constantly build conclusions, expertise, shading new light to old experiences, simulating knowledge gain and integration, assimilation. The implementation, functional properties of these intelligent objects are at the content creator's discretion, in function of the application requirements.

Some of the VE elements that may qualify as intelligent objects which are mentioned in other sections of this chapter are the following ones:

- meta-environment - an environment that evolves based on different rules,
- objects defined for story-telling applications,
- travel, steering, selection tools, widgets, objects,
- activity enforcing gadgets (e.g. replicas of real tools),
- landmark and worldlet techniques,
- geon objects [Biederman, 1987] which produce memorable, easily understood diagrams.

3.2.5 Environment knowledge

[Cavazza and I.J.Palmer, 1999] addresses the problem of knowledge representation in VEs. According to this paper, an IVE (Intelligent Virtual Environment) needs to use unified principles, by combining an abstract level of representation and a concrete model of the graphical world, with the world dynamics described with high-level concepts. In this context, simultaneous access is required to both concrete and abstract information, a shift from the command giving paradigm (to single entities) to the interpretation paradigm, where the information is processed in the global context.

Spatial partitioning
A rather distinct approach to environment knowledge is space discretisation. 2D or 3D space discretisation techniques are used to convey information about the environment in cases of obstacle avoidance, environment navigation, grasping, etc. A number of more common spatial partitioning methods like BSP trees, cells, etc. are described in [Diehl, 2001].

[Bandi and Thalmann, 1998] presents 2D space discretisation methods that are based on tiling (cells), free cells representing navigation opportunities and border cells providing proximity constraints. Multilevel tiling is also discussed by means of stairs in combination with the normal flood fill algorithm.

[Noser and Thalmann, 1996] and [Noser and Thalmann, 1998] propose three dimensional space discretisation in form of octree description. They describe the occupied space in an environment with dynamic octree methods, with a sort of "octree for objects" approach, which means object-wise segmentation of the environment octree description. A dynamic approach is necessary, since most

virtual environments contain moving elements, which require a local recalculation of the octrees. The octree information is used as input for simulating virtual sensors: collision, proximity, etc. It is also used in combination with L-systems to provide dynamic interaction capabilities.

[Bandi and Thalmann, 1997] discusses a space discretisation approach where the objective is 2D discretisation (the length is calculated from movement direction and height), for an animation method based on fixing a foothold and calculating the trajectory of foot to clear obstacle objects.

[Steed, 1997] suggests an approach to navigation which concerns the height of participant above the ground. Although scene graph systems offer this functionality for the user entity (avatar) automatically, the other entities like autonomous agents, avatars could use such an approach. It is based on a next viewpoint prediction (sort of a dead reckoning algorithm) and a cell structure to calculate (detect) surface heights and collision.

[Gillies and Dodgson, 1999] shows a 3D space monitoring system that uses time-line pictorial information to extract information about obstacles, moving objects, etc. It is a virtual sensor based on observation properties like: if a big part of the picture changes, the observer (user) is moving, otherwise an object is moving (invariance relationships) and a flat top surface suggest sitting possibility (affordability features).

3.2.6 Input/output devices

We have mentioned already a number if input/output devices, from simple and classical to complex and new. Mouse, trackball, keyboard, joystick, tablet are among the classical ones, although they can have newer cousins, with enhancements. These enhancements are usually in form of an additional dimension: different 3D mouse types have emerged, likewise 3D joysticks. Tablet input devices are gaining popularity, thanks to the PDAs (Personal Digital Assistants) and the emerging home tablet-PC solutions, where the digitising surface is integrated with the visualisation surface, making direct control and feedback possible.

Newer virtual devices that have been specially designed for virtual reality are stereo screens, synchronised projectors (for panoramic or stereo visualisation), tracking hardware (magnetic, optical, video sensors), force feedback or tactile simulation solutions. These are commonly assembled into visualisation products like HMDs (Head Mounted Displays), tracked gloves, CAVE systems, desk-like immersive systems with tactile input devices, (omni-directional) walking platforms or different movement controlling platforms (disk with pressure sensors, bicycle steer and pealing sensors). Older devices such as cameras that have been only media sources also tend to become useful as input devices for facial information capture, gesture recognition, motion capture, etc. [Nebel, 2001].

Examples of more non-conventional devices are the sensitive rod, smell devices, vestibular stimulation based motion simulation device. [Rheingold, 1991] describes also a line of older and not always commercialised non-orthodox VR devices, almost curiosities such as the Sensorama device based on film, panoramic view, smell (in a cabin with a gas-mask type of facial cover for precise smell impulses).

There is a category of complex and expensive systems based on motion simulator pods or suspended systems and a myriad of tools, used mainly as training

substitutes for the aerospace industry. These flight, military vehicle, etc. simulators add a realistic touch over software-only simulators (however realistic they might be), but they are available in a very limited way. However, regarding smaller pods (the base for simulation cabins, platforms) these are increasingly cost-effective and might be easily available for simulation purposes.

Finally, we can hardly wait to encounter the display techniques that are currently being developed, researched and commercialised: 3D screens without the need of glasses (autostereoscopic 3D Displays), foldable displays (with increased display sizes), on-retina laser projection devices, etc.

3.2.7 Summary

Our investigative experiments use VEs as editing systems. We include in spaces where the concepts of avatars, sensors, collision, navigation possibilities already exist, a number of custom elements that both provide a navigational space and a tool for the creation of further content.

The user of such an integrated system has the possibility to create their content in the same type of environment that the content will be deployed on. This is a design option that has an important advantage: it provides a realistic visualisation over the object of interest.

The widgets concept for "3D manipulators" connects different aspects of VE properties. Widgets can be used as "customiser objects" to enhance the compelling factor of an environment, according to the desires of the user. They provide extra layers of interactivity above the basic VE possibilities, which also put them in the category of intelligent objects. Even partial environment knowledge can be assigned to them, for instance if the constraints of a widget are set dynamically, according to different environment properties.

Multi-user availability is present in VEs with a little bit of modification. Specifying things like shared objects and locking properties gives the possibility to use these environments as multi-user worlds, collaborative or even distributed virtual environments.

An editor system being implemented as a VE has another advantage. The mapping between the interaction layer of the visualisation system and the available IO (Input/Output) devices is already taken care of, allowing the use of the different devices if necessary.

Chapter 4

The HanimPlus system

Based on our target goal of creating visual character editing utilities as an attempt to easily populate Virtual Environments (VEs), we present in this chapter the prototype systems built to achieve our goal. We do this systems presentation in a time-sequential manner, starting from the first geometry editing systems through the more evolved whole character editing system which is a combination of geometry editing and hierarchy editing subsystems. We base our systems on considerations drawn from the literature presented in Chapter 3.

The theoretical and technical considerations drawn from the relevant literature (previous chapter) led to the conception of the general guidelines presented in Section 4.1. These guidelines have been the basis for the development of the editor systems described in this chapter. Since the content of this work touches a number of research areas, these guidelines are sorted according to their relevance to these fields. The areas of research and their relevance to this work is thoroughly discussed in the subsections of Section 4.1. A short enumeration of these research fields and the connection to the presented work is as follows:

- Information Visualisation. Since the most probable target audience knows the VRML file format, we chose for our editing systems to visualise the underlying components, the structure of 3D objects (characters, geometry) as well as the interaction with these objects. Even if the users do not know the VRML file format but just the VRML interaction possibilities, the single concepts to be learned is IndexedFaceSet (IFS) construction, which we chose to visually represent in every detail, and the hierarchical properties of the edited character, which is not VRML specific, and which we chose to represent also in a visually detailed manner. An important aspect is the visualisation of user induced changes, which is to be done real-time, since VRML is a format that is aimed at real-time visualisation. The tools that visualise the interaction possibilities in a traditional 2D button and a more visually oriented, 3D control widget manner are chosen based on the function they perform. Functions that do not need parameter values to operate are visualised as simple buttons, while other functions that depend on user manipulation values (translation, rotation) need 3D widgets that convey the manipulation possibility to the users and record the actual manipulation values.

- 3D editing User Interface. By visualising the inner components and pro-

viding visual manipulators (buttons and widgets) the editing systems are capable of handling different 3D entities. The edited entities are 3D objects representing single or complex geometries, the latter containing a hierarchy, usable for animations. Other help tools (widgets) are also desirable for the UI (User Interface) to provide rotational, scaling operations for the edited entities to be manipulated into the desired orientation and size.

- 3D character geometry (with bone hierarchy). H-anim is relevant since it is an attempt of standardisation aimed at VRML (and generally 3D) humanoids, providing a basic hierarchy framework. However, this specification is limited in the sense of specifying humanoid hierarchies, which we would like to avoid, allowing the modification or extension of the H-anim specified humanoid Joints and Segments. Character hierarchy editing systems are also relevant, since they do provide hierarchy editing possibilities, but usually in a complex environment and with unsynchronised Joint and Segment correspondence, resulting in different editing possibilities that the H-anim construction suggests. To keep this synchronisation and the model consistent, we opt for a bound Joint and Segment editing approach, which requires the use of techniques that eliminate geometry loss and suppress the perpetuating hierarchy operations.

- Animation guidelines. There are a number of animation and motion analysis methods that inspire the animation editor currently under development, the motion segmentation methods for basic motion extraction being especially interesting.

- Interactivity guidelines. The main (default) interactivity in VR is the possibilities provided by the browsers (navigation, examination, etc.). Next to these, there are custom examination and object handling interfaces, which replace or extend the default interface. There are also locally defined interaction possibilities in the form of different tools and widgets; these are defined in the content itself, just like in our case.

- HCI (Human Computer Interaction) guidelines (usability, ease of use, etc.). Interface simplicity, selection options, context sensitive operations and other aspects of HCI relevant to our systems are described.

Next, we describe our general guidelines, the geometry editors, the hierarchy editor, and a closing section for smaller experimental applications, like a NURBS (Non-Uniform Rational B-Splines) editor, texture editor, etc. These editor systems, in the case of geometry and hierarchy editors, are systems that are based upon the previous system, extending it to include other types of functionalities. Therefore their description would be pyramid-like, with the hierarchy editor described on top of the geometry editor. The smaller NURBS and texture editor experiments are at the moment separate systems that work with single geometries or replace the geometry editor system. These will be eventually integrated into the main character editing system.

4.1 General guidelines

The general guidelines are presented based on their belonging to the different research areas this work touches. They provided the focus path for the development of our prototype systems, from the approach needed to handle a simple coordinate to the handling of complex hierarchies and animations.

4.1.1 Information visualisation guidelines

Information visualisation is regarded mainly as visualisation of recorded data for the purpose of analysis. There are scientific, economic, technical and many other reasons for these visualisations. For our purpose of editing 3D characters, we visualise in our systems geometrical data, their connection properties, hierarchy data, available options for modelling, different widgets and screen estate managing. We distinguish between visualisation of pure data and other types of properties and between the visualisation of selection assignment or other types of sensors which are used to visualise target entities for the effectuated operations or editing (data manipulation) possibilities.

4.1.1.1 Data visualisation

For the geometry editor, the finished product is a mesh of polygons. The polygons depend entirely on a number of geometry coordinates. The coordinates are also the basis for the VRML file format and the internal storage format used. These are the entities the values of which the content creator sees in the source file. Since the coordinates are the basic elements of the mesh, every change on a coordinate is reflected on the polygons. This provides for the editing operations multiple options. For the basic editing operations, the vertices are the obvious choice. For higher-effectiveness editing, we can choose between the common edge and polygon based approach and between the possibility provided by the data construction particularities (discussed in Section 4.1.2). This latter option of handling sets of vertices provide a more natural, novice-user oriented approach, providing also a higher effectiveness level of manipulation than an edge- and polygon-based approach.

Our visualised data is therefore the geometry coordinates and the polygons that these coordinates construct. We must visualise also the conceptual connection of the regular mesh coordinates: their arrangement into sets of rings and columns. This arrangement is inherent from the regular grid data used for the geometries, and the visualisation is used also as a selection mechanism for single vertices or collections of vertices.

When the visualised geometry is a self-symmetric geometry, one side of the geometry is automatically mirroring the position of coordinates on the other side. In this case, the mirrored coordinates do not need selection and editing controls; this way they can be used as undisturbed visualisation surfaces. There are also differences in mesh manipulation: some of the coordinates due to the symmetry properties are not allowed to perform asymmetrical operations. In that case, with the different types of selections, the operations resulting in asymmetrical geometry must be disabled.

In the case of the hierarchy editor system, the underlying data components are the Joint positions, their connections (parent-child relationships) and the

Segment geometries. We must visualise these components and assign selection possibilities (sensors) to the editable entities: Joint positions and Segment geometries.

The hierarchy construction knowledge must be integrated in the rules for hierarchy modelling. These rules make it possible to maintain the "one parent" (an exception is made for the root Joint, since it has no parent Joint) and "one Segment per Joint" constraints of the H-anim specification. This also allows for translational selection properties in the system, such that selecting a Segment geometry also means selecting the Joint the Segment belongs to and vice-versa. This way, it is possible to even select a Segment geometry through the Joint selection mechanism, similar to selecting the individual Segments from the whole character geometry.

For ease of use, the symmetry concept of geometries can be used for characters as well, where the coordinates on one side of the body are mirrored on the another side. But since the symmetry in this case can take form as self-symmetrical Segment geometries and as pairs of symmetric geometries which require different handling, two cases of symmetries are distinguished: internal symmetry for self-symmetric Segments and external symmetry for pairs of symmetric Segments. Since the Joint operations also require differentiations in these cases (in case of symmetry there are Joints that have self-symmetric Segments and there are pairs of Joints that are symmetrical to each other), the same notation of internal and external symmetry is used in case of Joints, which influence the modelling possibilities in the context of different Joint selections.

Modifying Joint centre coordinates should modify the whole geometry, where the geometry follows Joint positions, even if the geometry deforms. Since Joint centre coordinates are conceptually the centre points or are close to the centre points of Segment beginning and ending coordinates, we can relate the general Segment positions (coordinates) to parent and child Joint coordinates. This enables us to convey simple Joint centre coordinate displacement operations into whole Segment geometries in the form of percentage-based successive ring displacements (percentage is based on geometry ring distances from displaced Joint centre), where the ring coordinates follow the Joint displacements.

There are a number of special coordinates visualised. They are leaf coordinates (Segment endpoints) and although they are not Joints, they are visualised similarly in the hierarchy editor. They are used to show the extent of the body and to provide higher level geometry operations for the Segment geometry endings. These operations are usually available only for Joint-based selections, such as Joint centre displacements with the Segment geometry following the displacement. However, leaf coordinates do not participate in hierarchy modification operations, since they are by definition not part of the hierarchy.

The character editing system should visualise the whole character, the character hierarchy and Segment geometries if they are selected and the system is in the right editing context. Their screen positioning and the reasoning behind the editing contexts separation are discussed in Section 4.1.1.3.

4.1.1.2 Operations (functions) visualisation

For the editing options, two types of controls are used: 2D buttons and 3D widgets. The 2D buttons contain no 3D information, representing functions (operations) that besides the selection context, do not need other information

for the effectuation of the requested operation. They can be regarded as parameterless functions. 3D widgets, on the other hand, represent operations where besides the selection context, additional information is necessary for the effectuation of the requested operation. Some examples of these are: movement(displacement) operation for coordinates, coordinate collections or Joints, rotational operations, etc., all requiring numerical values describing the extent of the operation.

The 2D buttons are visualised as common buttons with text as texture, describing the functionality of the respective button. A Billboard node can be used to position the buttons to always face the user. A little bit more detailed explanation for operation buttons is available through the status bar.

The operations that need to be visualised as 3D widgets are as follows:

- Unit selection: a range of selectable unit lengths. These unit lengths can be regarded as a list of selectable values, and the modality to choose one of them is possible through a PlaneSensor construction with the values as a list in form of texture attached to the plane sensor geometry (a simple panel). The selected value would form the basis for the editing operations extent: lower values would mean small, precise operations while higher values would mean operations with greater extent, making scale differences easily manageable.

- Rotation widgets: 3 Degrees Of Freedom (DOF) should be used for general geometry since precise but also arbitrary geometry positioning is needed, while only a single vertical DOF is enough for the hierarchy, since 3D character hierarchies are commonly built vertically (think of suspension points, centre of mass and gravity constraints). A single CylinderSensor allows the rotation around a single axis which means 1 DOF. To achieve 3 DOFs for Segment geometries, multiple and rotated CylinderSensor instances have to be combined and their recorded data handled accordingly to produce complex 3D rotations.

- Zoom widget: the widget does not imply changes in geometry, therefore an unlimited range of zoom values is not endangering the manipulation of geometries, even if the user is not yet accustomed to the precise 3D manipulation of VRML interaction.

- Displacement (positioning) widget: this would be the main editing widget, which positions the selected entities. The range should be controlled by requiring individual activations of the widget for unit based displacements, and the user's task eased by using appropriate custom-set unit values, reducing the necessity of a high number of repetitive edits operations or actions. The different integrated sensors separate the displacement DOFs, making precise editing possible.

- Ring rotation widget would imply rotational changes to selected rings. In such a case the distance unit does not count, since rotation data transformation is unnatural to be bound to distance type data. The manipulation range is unlimited, but since rotations are repetitive, a precise handling is easily achievable.

A detailed description of the actual implemented widgets is to be found in Section 4.3.1.1.

4.1.1.3 Screen estate management

With the integration of multiple editing components: Segment geometry and hierarchy, the screen real estate has to be managed, since not all components can be visualised at once without creating a cluttered environment. There is a hierarchy skeleton, a whole character and Segment geometry visualised, which do not have to be visualised at the same time. For instance, in case of hierarchy editing, the Segment selection is irrelevant and should not be visualised at all. In the case of Segment geometry editing, the hierarchy skeleton can be put aside but still visualised, since it can provide geometry information and it can serve also as a Segment geometry selection method. The whole mesh visualisation should be always present, providing information and selection possibility for both the hierarchy and Segment geometry editing modes.

The data visualisation options are closely related to the operation/function options visualised. The buttons and widgets must be visualised only if they apply to the selected editing mode (rotating a ring of coordinates makes no sense when a Joint is visualised) and being enabled only when the data selection context applies to them.

A user-handled aspect of screen estate management is a zoom tool that would allow the user to handle the scale size of edited geometries. Such a feature can be useful in cases of too small or too large geometries, where the detail or the overview is not satisfactory.

4.1.2 User interface guidelines for 3D editing

A 3D interface is already given since we use the VRML format where the whole environment is a virtual 3D environment. The user interface tools should be constructed of 2D buttons and 3D widgets. The 2D buttons are a WIMP legacy, and not really necessary, but their usage is justified by the expressive text display they can contain and by the fact that they represent functions without parameters. 3D versions can also be used as they provide the same functionality. Since there is no 3D button standard, a 3D geometrical representation is likely to look very similar to 2D buttons, especially if is set to always face the user.

The widgets are only few of number and their usage can be learned in a couple of minutes quite intuitively. Description texts for these are available in the status bar for easier start or fast remembering.

The main components for 3D geometry editing can be categorised into three steps:

- positioning of edited entities,
- selection of target entities or entity subcomponents,
- the effective data manipulation with knowledge about the internal construction of the edited entities.

For all of these editing phases, tools must be developed. These tools must depend on the actual geometry, hierarchy and animation properties of the edited characters. Due to these constraints, we can derive the following editing rules:

- positioning of hierarchies should be handled only around the vertical axis,

- positioning of individual Segments is fully 3 DOF oriented,
- the scale widget provides a pseudo-positioning approach to use as much screen estate as necessary for a particular action.

The selection process should be straightforward and totally unencumbered, all editable entities having a TouchSensor attached and activated at all times, effectively making selections the highest precedence actions.

The effective data manipulation requires tools that provide manipulation possibilities for different data types: positional, rotational and construction-specific data, including the rules from regular grid construction and hierarchical properties.

The actual implemented operations like mesh positioning and refining, hierarchy refining or extending, etc. are described in the subsections of sections 4.2 and 4.3, for the geometry editor respectively for the hierarchy editor systems.

4.1.3 3D geometry and characters guidelines

A single coordinate represents a single property: a position in space. Without context knowledge, the executable operation on a single coordinate is the movement (displacement) operation. However, when we include context knowledge, other operations are becoming available, like scale, smoothing, rotation, etc., all taking benefit of the extra information gained from coordinate relations or relative coordinate positions.

For the Segment geometries, we have chosen a regular grid for the mesh format. This was partly due to easier editing options and partly for easier code handling options. It allows the use of collections of coordinates which results in faster geometry handling. The data is parsed more rapidly and the code saving process is also shortened. VRML IFS code generation is also simple since the regular grid concept describes in itself the geometry triangles creating order (the indexing). Furthermore, this regular grid is also easily converted to NURBS control points, providing a higher level geometry. Although it is not yet a standard component of VRML, future versions of the format are about to adopt the NURBS geometries as standard components.

Our 3D characters are collections of regular meshes assembled together following the hierarchy constraints. The mesh geometries are handled generally separately, with the hierarchy editor concentrating on Joint hierarchy operations but also on the geometry operations which are the consequences of the Joint operations. An example of the latter is the geometry that follows the Joint displacements or the mesh segmentation method when a new Joint is inserted.

Data storage is achievable using a simple file format describing the basic properties for Segment geometries, hierarchies, etc. Segment geometry files need to contain grid dimensions, H-anim specified properties and the list of coordinates. Joint files need to contain positional and other H-anim properties. For both file types, the file name (without the ".segment" and ".joint" extensions) is their reference name as to be used for H-anim character construction, and to be used by the hierarchy file describing parent-child relationships and other properties. The Joint and Segment files belonging to a character should be stored in a subdirectory, with the subdirectory name being the character's name. The hierarchy file must also be stored under the character's name, but

with the ".hier" extension, also specifying the directory name the Joint and Segment components are located into.

We do not consider H-anim conformance as a criteria for all characters created by the system. We feel that if the user wants to create H-anim compliant characters, he/she is free to do that using the H-anim compliant templates or by own modelling efforts, which can be then saved in the H-anim specification suggested format, resulting in a H-anim character. However, the user has also the possibility to alter these H-anim hierarchies with Joints outside the pool of allowed Joints and Segments or use other types of template hierarchies, which then will be saved in an H-anim suggested manner, but since other types of Joint would be also present, the resulting characters will not be H-anim compliant. We provide different H-anim compliant templates, since the H-anim specification requires a number of minimum Joints and Segments, which then can be extended as wished until all the specified H-anim Joints and Segments are used up. Within the H-anim specification, there is a minimum and a maximum requirement for characters, which can be left out of consideration in our system at the user's discretion. This way the users have the possibility to manipulate the hierarchy as wished to achieve hierarchies that are different from humans: with more or less than 4 limbs, 1 head, 5 fingers, with tails, etc.

4.1.4 Animation guidelines

Our aim regarding animation is to apply automatic segmentation on motion data by analysis of motion data properties, like derivative calculation with motion sequence endpoint matching and to extract basic, parameterised motion sequences. We then need to visualise these basic motions and we need tools that help in testing the combinations of basic movements. We could use Joint-based motion data representation in combination with motion sequence-based visualisation to provide variate manipulation possibilities with regard to enabling/disabling data components, handling the timing options, etc.

4.1.5 Interactivity guidelines

Interactivity is inherited from the VRML visualising system. In such systems it is possible to at least walk and observe the environment as wished. Our systems provide on top of that interaction possibilities for geometry manipulation. The manipulation possibilities are aimed at the basic data components, collections of components or at custom data properties, but which at the end affect the visual appearance of the virtual VRML environment. The changes made are stored and saved in internal or visualizable VRML format, making the results of the interaction activities available to the user. Interactivity is enhanced by using templates for characters, since the underlying structure in many cases is at least similar. The template-based approach is a great advantage of specialised tools, making the creative process easier and less time-consuming.

4.1.6 Human computer interaction guidelines

Interactivity is sometimes considered as a subpart of HCI, making this the authoritative entity over it. However, our view is that not the interactive process itself, but closely related concepts like the outcome of interactive processes or the

interaction modalities can be included in the HCI field, respectively the usability and direct/indirect manipulation modalities (indirection considerations).

From the usability point of view, we looked first at the macro level, at the outcome of the editing process. We want to produce not very complicated 3D geometries and characters, which would be otherwise too cumbersome to program manually and too much of an overhead to do it with complex editing tools and conversions. Another macro aspect is the platform the systems run on: a web browser with a VRML plugin, providing an easy to access on-line environment which could be an incentive for a broader usage.

At the micro level, the editing process itself, we tried to decompose the editing options into a minimal number of operations, based on the common concepts of coordinates and coordinate collections (rings and columns of coordinates), and to provide in this way a minimalist user interface. We also base the look of the editing widgets themselves on the properties they manipulate, for a more natural and simple interaction.

Simplicity is also an important aspect. Besides the minimalist interface and property-based manipulators that result in a simplistic approach we also opt for a tool availability method based on the editing contexts and the selection contexts. We use separate editing modes (geometry and hierarchy editing mode) for screen space managing reasons, and therefore we separate the editing operations themselves, showing at once only the relevant operations. These operations should be then enabled or disabled based on the selection context, making the operation selection options even more easy to follow.

Regarding the UI, we adapt a hybrid approach. For the parameterless operations, based only on selections the 2D GUI (Graphical User Interface) Button element can be used; for operations that represent parameter-based manipulations which need some data like displacement, rotation, etc., custom 3D widgets are necessary to measure the user interaction. These widgets must be constructed in an easy to use, responsive manner, providing not only manipulation data, but also interaction feedback to the user. All possible operations should be available at once within an editing context, for a quick access UI.

In our systems, we use both direct and indirect manipulation. Much of the manipulation possibilities are based on direct manipulation, like the selection context, the Segment geometry zoom, ring rotation, 2D and 3D rotation (positioning) widgets, and of course all the function buttons, making the interface easier to use. The only component that does not satisfy the direct manipulation criteria is the displacement widget, requiring successive activations instead of drag operations. This approach is deliberately applied, since the displacement widget works on a number of different target entities (single or collections of coordinates, Joint positions) which also require precise handling since these operations which influence the geometry could be the weak point in the case of an unlimited range, imprecise drag-based manipulation tool.

4.2 Segment editors[1]

The editing system applications for single, regular meshes are an asymmetrical geometry editing system and a symmetrical geometry editing system, which is an extended version of the former. A combination of the two is included in the character editor system, complemented by a hierarchy editing subsystem. The extensions besides the symmetry handling are a number of widget modifications or the introduction of new widgets. These extensions are natural refinements in the system development process, aimed at improving the system and adapting it to the requirements of the hierarchy editing system it is evolved into.

4.2.1 General framework

Our visualisation method is to connect VRML and Java components (a VRML plugin respectively a Java applet) in a web page using the EAI (External Authoring Interface) specification which allows us to generate and edit VRML meshes interactively. The meshes used are based on regular grids, to provide an interaction and modelling approach that uses the internal semantics of such a mesh by linking the available modelling operations either to single vertices, the vertices of a ring or column, or all the vertices from the VRML mesh. Our method permits strict mesh complexity control and scales the operations according to the mesh properties. We describe the structure of the system and provide a few examples of the meshes that have been created.

Our aim is to create modelling systems that can be used to produce meshes for virtual environments based on the VRML (Virtual Reality Modelling Language [Web3D Consortium, 1997]) standard. The Java binding to the VRML world allows for a dynamic mesh handling approach. Instead of using conventional editing operations as described in [Maestri, 1999] and using automatically generated complex surfaces (see [Goto and Pasko, 2000], [Grahn et al., 2000] and [Wakita et al., 2000]) we use an approach based on vertices and sets of vertices grouped in rings and columns. We also try to provide an easy to use interface, for which an inspirational example would be the "Teddy" system [Igarashi et al., 1999].

4.2.2 Editor appearance

Figure 4.1 is a screenshot of the geometry editor in action. The system provides a simple, easy to use interface for all the operations possible, without any elements that must be looked up from menus or lists. The visual elements are grouped and marked according to their functionality in the editor system, and their functions are presented below.

1. Buttons of the Java applet for loading, saving the mesh data and exporting to VRML format. They use the familiar load/save window of the underlying platform, provided by the Java AWT classes, and they perform file readings and writings of the internal format and file writings to VRML file format.

2. Editing buttons in the VRML environment. From left to right and from top to bottom in order: add ring and add column (which increase the mesh

[1] This section is based on the [Kiss, 2001b] conference paper and the [Kiss, 2001a] technical report, with added information over the further developments regarding the Segment editor systems.

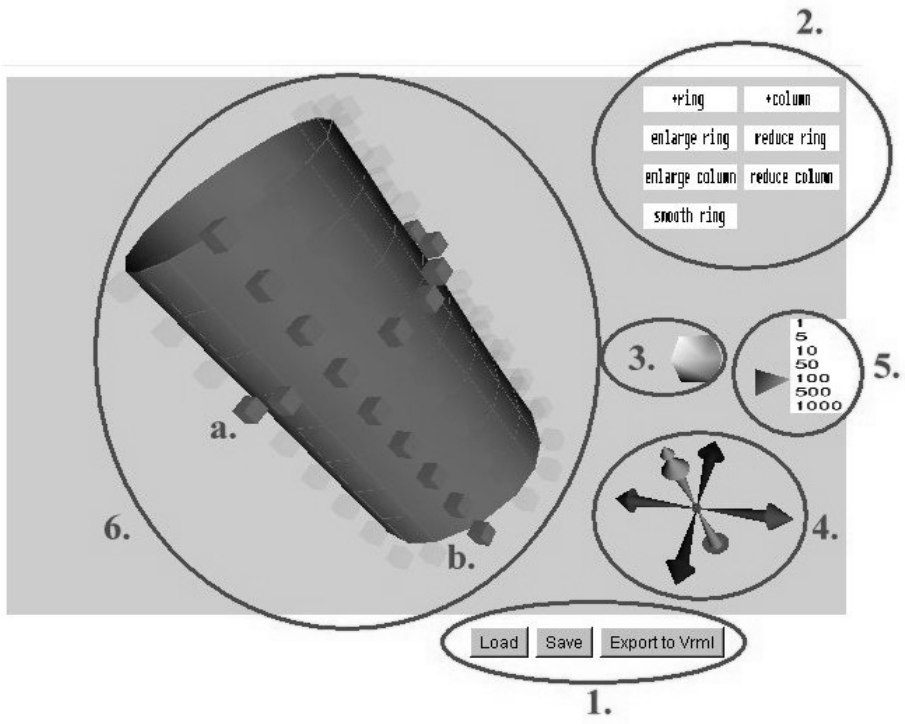


Figure 4.1: Annotated screenshot of the editor.

complexity), enlarge ring and reduce ring (for the ring-wise scaling of the mesh), enlarge column and reduce column (for the column-wise scaling, which has a global scope, affecting the whole mesh not just the selected column) and last the smooth ring button which is used for smoothing the mesh surface ring-wise, but trying to preserve the mesh curvature by applying a percentage-based smoothing approach.

3. Rotation widget (box). In the figure the widget has been already rotated, and the elements noted with 4. and 6., the editing widget and the edited mesh follow its rotation exactly, providing orientation and tool alignment feedback for editing. Used for positioning the mesh in a manner that allows to perform the desired selection of vertices and to preview the mesh from different positions. Later versions of the character editing system replaced this rotation sensor with a more precise one, since the mapping of 2D mouse movements to 3D rotations is not very intuitive. The new rotational widget separates the rotational DOFs, allowing for a more precise interaction. The new type of rotational widget can be seen in Figure 4.2.

4. Editing widget. The widget is composed of 6 arrows, with different colour for each pair of arrows that represent the positive and negative direction of the main axes. A click on an arrow will displace the selected vertices in the direction the arrow is pointing by the length selected using the unit selection widget (next widget).

5. Editing unit selection widget. The default value is set to 100 millimetres,

this can be changed by dragging the triangle on the left. A wide range of values (from one millimetre to one meter) gives the possibility to handle meshes at different scales, and to perform scale transitions easily. Selection feedback is given by setting the selection sensor(s) of the selected vertex(ices) to solid instead of semi-transparent.

6. Edited mesh. a) is the current selection, a ring, while b) is a column which is not yet selected, but the pointing device is over the column sensor, and the column is highlighted as a possible next selection.

There are two new widgets introduced as components of the Segment editing subsystem of the character editing system. These are a zoom widget and a ring rotation widget, depicted with 2 and 3 in Figure 4.2.

Figure 4.2: New and changed widgets in the character editing system: 1) changed rotational widget, 2) zoom widget, 3) ring rotation widget.

The Zoom widget has been introduced to provide an alternative instead of the automatic mesh scale size handling present in the initial versions in combination with a fixed-screen GUI, which has been left out from the character editing system. Instead, we implemented a widget which provides to the user the possibility of scaling the edited geometry to a comfortable scale, as the user sees fit.

The Ring Rotation widget provides the possibility to easily bend the Segment geometries ring-wise. The widget is similar to the normal geometry orientating widget, but it has a different functionality. The rotation widget rotates itself, the target entity and the editing widgets synchronised, to be always aligned; however, the ring coordinates rotation widget limits the rotation action to itself and the selected ring coordinates. Even the self-rotation is limited to single axes (which are represented by individual rotational sensor components) for achieving visual feedback, but without changing the widget's general orientation, which is synchronised with the other widgets and the edited mesh itself. This way, the rotation axes are not mixed up for different ring coordinate selections.

4.2.3 Data format

The editor uses an internal format to store the mesh data. This is done for two reasons. First, there is a data parsing reason, meaning that a simple data format would allow a more comprehensible parsing method. Second, an internal format ensures no complications regarding the use of a data file with possibly non-regular structure. Anyway, the latter might not be a good idea (to create the mesh elsewhere and import it in the application) as the goal is to use as simple meshes as possible. Users are not required to edit this data manually as it is done generally in VRML. This would be a low-level editing intervention,

and the application provides a visual and also low-level approach to editing. Although manual edit is certainly possible, it is more cumbersome than the possibilities offered by the editor, for instance to select a single vertex, and move it in any of 6 degrees of freedom even with a displacement unit of one millimetre. Of course, the same operation can be effectuated more globally, on a set of selected vertices.

For the separate geometry editors, the data format is more simplistic, containing only the most necessary data. A geometry file would contain a comment header, a Segment name, the number of coordinate rings in the Segment, the number of coordinates in each of the rings and the actual coordinate values in square brackets, separated by comma. The data file has the following structure:

```
Comment header until first hash sign
#
segment abc
rings 2 \r
points 4 \p
[x1 y1 z1, x2 y2 z2, ..., x8 y8 z8]
```

With the integration of the geometry editor within the hierarchy editor, the H-anim Segment specific properties have been also included and small changes have been made, but the simple file structure still remains. The included extra information specifies the mass of a Segment (used for physics-based simulations). The changes consist of: the exclusion of the Segments name from the file (the file name specifies the Segment name); the square brackets and commas are left out as well, with the coordinates being separated by placing them on new lines.

```
Comment header until first hash sign
#
rings 2
columns 4
mass 10
coordinates
x1 y1 z1
x2 y2 z2
...
x8 y8 z8
#
```

4.2.4 Structure of the editor

We have already mentioned that the editor uses the combination of VRML, Java and EAI technologies to provide its functionality. Since these are distinct technologies, and the EAI component handles all the interaction between the VRML and Java components, it is possible to discuss these components separately. Taking into account that the communication between the components is a two-way communication, it is best to show it in the flow diagram from Figure 4.3 and explain its mechanisms step by step.

The VRML component is a VRML file which at the beginning contains no mesh, just an anchor point for binding the generated mesh geometry into the scene graph (this will be loaded and generated at user request). The file does

contain beside the anchor point for geometry a set of "widgets" that are tools inside the virtual environment that aid in the editing process. Depending on their functionality, these widgets act directly upon the virtual environment, or indirectly by sending their events to be handled by the Java applet, which in turn passes modified data back to the VRML component. This second component (the Java applet) is used for the data handling necessary for opening, generating, modifying and saving meshes. This applet is also used for translating into the environment the effects of user actions performed through some of the widget tools.

These two components that work together using EAI are packed in an HTML file for presentation in a web browser. As they are side by side in a common environment (that of a web browser), it is possible to use code that enables the communication between the Java and VRML components, and this code is the EAI.

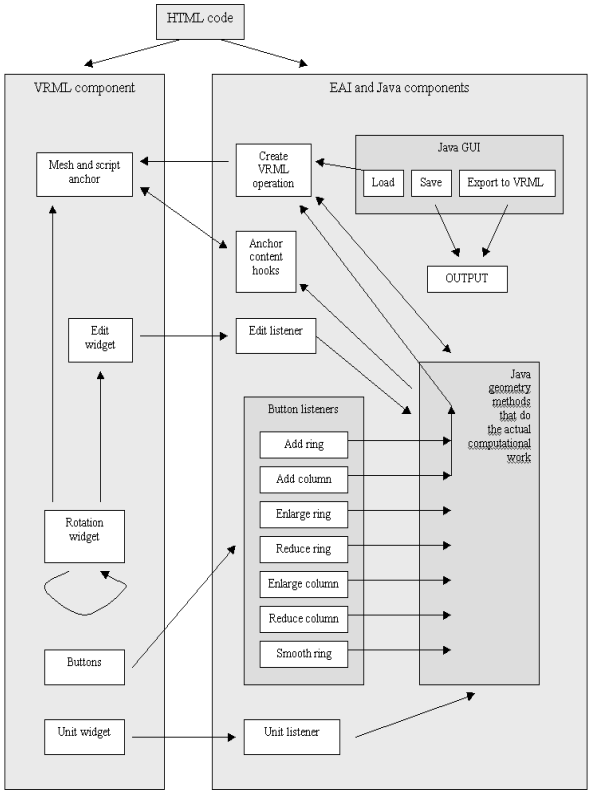


Figure 4.3: Flow graph.

4.2.4.1 External Authoring Interface (EAI) component

This is the simplest component of the structure: it connects the virtual environment and the applet that are inside the same web page. It gives the possibility for the Java methods to access the VRML content, and does this by providing

Java wrapper objects around the internal VRML browser, nodes and fields. As it can be observed from Figure 4.3 there are a multitude of connections between the VRML and Java components, and yet these are not all. These are in fact representing only conceptual connections, and the number of real connections is much higher, since the VRML "objects" (single nodes, grouped nodes or new prototypes) usually have multiple event sending/receiving possibilities, which have to be used to provide interaction possibilities to the system.

4.2.4.2 VRML component

The VRML component provides the "graphics engine" for the editor. It has the role of visualising the mesh and all the user controls in real-time. It allows the run-time linking of code into the system, which permits truly dynamical interaction to be built.

In the following we will present the elements of the virtual environment one by one, this will allow a more accurate view over the component. We start with the VRML prototypes, which are custom groups of nodes generally used to store reusable code (groups of nodes in this case), then we present the scene graph, how these elements build up the environment.

VRML "PROTO" prototypes

The first reusable code component inside the VRML file is a HUD (Heads Up Display) prototype. This prototype is used generally for creating custom display panels inside the virtual environment, for instance a custom navigation bar, or a tool-bar. Its purpose is to mirror the effect of user movement on these panels, in other words keeping the panel unchanged, in its initial position on the screen, indifferently how the user navigates in the environment. This particular HUD prototype accepts any subtree of nodes as parameter. Therefore it can be used for different purposes, as we will see when discussing the other elements of the VRML component. When integrated into the character editor, this component has been left out in favour of environment navigation and consequently user-handled visualisation options.

The second prototype is a geometrical assemble, representing a three dimensional arrow. Its functionality is related for specifying the DOF (or axis) operations, for geometry component displacements.

The last prototype of the application is a button prototype. It consists of a small rectangle, on top of which a picture is mapped: this picture explains in text (this is the method currently used) the functionality of that particular button. In addition the button has the functionality of displaying in the status bar a short description of the function it represents. Its main purpose is, however, the capability to track user actions with a sensor and the capability of switching the button on/off.

Scene graph

After the prototype declaration and the usual general settings code of a VRML file, the editing environment is created using the prototypes defined earlier and some additional nodes. First, an editing "widget" (or tool) is created. This is embedded in a HUD instance; thus user navigation does not affect it. The widget itself consists of 6 instantiations of the arrow widget, sensors attached to all of them, and a script that handles the events of the sensors. The 6 arrows

represent the 3 DOFs of a three dimensional space, the +X, -X, +Y, -Y, +Z, and -Z directions. As a visual aid, X directions are coloured red, Y directions green and Z directions blue, as a sort of direct mapping between the enumeration of the three dimensional coordinates and the enumeration of the three basic RGB colour components. Their purpose is to collect the user's displacement commands regarding the selected mesh vertices. As the user clicks on such an arrow, the sensor attached to the arrow sends a "clicked" message to the script. Each arrow has this functionality, and the script, depending on which arrow has been activated and depending on the current displacement unit, will issue a displacement event containing the value of displacement. This value is caught by an event listener defined in the Java component.

Other elements of the virtual environment are a number of instantiated buttons. Their functionality is to transmit their activation status to functions from the Java component that will effectuate the actions related to the buttons. The button functionalities are:

- adding a ring or a column. These buttons trigger Java methods that add a new ring or new column right after the current selection. The order of the rings is defined by the data enumeration order. The column order is calculated from the data using the right hand rule, with the thumb representing the ring data direction and the remaining bent fingers representing the vertex enumeration direction. This operation requires the creation of a new IFS mesh, since coordinates and indexes need to be recalculated.

- positive or negative ring scale. The selected ring is enlarged or reduced by adding to respectively subtracting from its vertex values. The extent of the change is determined by the displacement unit vector. Ring scaling introduces changes in the object thickness. The buttons are not active if a column is selected.

- positive or negative column scale. Unlike the ring scale operations that are acting on a subset of mesh vertices (rings), a column scale operation acts on every vertex of the mesh. A positive scale in this case means that the object will become a longer (taller) object and a negative scale that the object will become shorter, while the thickness does not change. Scale column operations are distributed between the mesh rings based on percentages calculated from the ring positions relative to the mesh centre, this way the mesh will retain its original shape.

- smooth ring. A fraction of the relative distances of vertices from the centre point of the mesh are used to smooth the contour line of the ring vertices. Eventually, with enough smoothing operations, the vertices will approximate the same distance from the centre point, as if they were on the same circle.

Another element embedded in a HUD is a container element, which is empty at the beginning, and is filled by the code generated by the Java component. The container is used only as a linking point, and contains no other functionality. The functionalities will be added at the same time with the generated geometry as the Java component generates the code simultaneously and links it to the VRML container element as one piece of code.

The next item is the rotating widget or tool. The tool was originally represented as a three dimensional box, with a rotation sensor linked to it. There is a drawback to this construction, which lies in the mapping of the two-dimensional input device (the mouse) to three-dimensional manipulation. However, as the user becomes acquainted with this particular manipulation possibility, it can provide enough control for editing purposes. The function of this widget is to enable the viewing of the mesh from all angles and also the favourable positioning of the mesh in the course of the editing process. As the user drags the rotating widget, the same rotation is conveyed to the edited mesh container (and consequently to the edited mesh itself) and to the editing arrows. This way, not only the mesh is positioned, but the axes of the editing arrows and the actual axes of the vertices remain aligned. A newer version of the rotation tool separates the rotation axes for a more intuitive and precise control over the rotation and thus the mesh positioning. It also provides a function of resetting the position of the mesh to the default position.

Another tool widget that is in appearance similar to the rotation tool is the ring rotation widget. It has almost the same geometry, but its target is not the whole mesh, but the currently selected ring or the ring the currently selected coordinate is located on. The widget provides the function to rotate a ring of coordinates around its centre and along one of the main axes. Since there are separate controls within the widget for each of the three main axes, a combination of rotation values can be achieved by sequentially applying and adjusting 1 DOF rotations.

The zoom widget provides control over the visualised Segment geometry's scale size. At the time the geometry editors were independent systems (not yet integrated into the hierarchy editor) this function was not present, since then the mesh was positioned using a dedicated algorithm, taking advantage of all the available screen space. Integrating the geometry editing system into a more cluttered environment resulted in another approach to screen space manipulation: a zoom widget, which gives the control into the user's hand with regard to the geometry's screen space. This also gives the possibility to handle small details that were not easily seen with an automatic geometry scaling approach. As a drawback for 3D objects, a geometry scaled too much can occlude some of the controls including the zoom widget. In this case, re-selecting the geometry from the whole character mesh disguises an operation of returning the edited geometry to its original scale.

The last element of the environment is an editing unit selection slider. The user has the possibility to select the edit unit between the 1 mm, 5 mm, 10 mm, 50 mm, 100 mm, 500 mm and 1000 mm values, allowing a wide scale of the meshes to be edited. A triangle is pointing to the current value, represented as a panel with the enumerated values written on it. By dragging the triangle, these edit values can be selected.

The vertex selection sensors created for all vertices and the ring and column sensors used originally the exact vertex positions. This made IndexedLineSet (ILS) sensors not optimally visible. A new approach uses now a scaled sensor grid, which makes the sensors raised slightly above the geometry surface, being clearly distinguishable.

4.2.4.3 Java component

The Java component is the part of the editor that essentially deals with all the interactions that occur inside the editor. It receives the events of the VRML component, effectuates the functions/commands associated with the particular events and it returns the result in some form to the environment.

Connections to the elements of the VRML component
To have access to the VRML component, the Java component uses code from the EAI packages. This way, it can get hold of the objects inside the virtual environment. It is not necessary to explain how exactly this has to be accomplished, a list of connections and their purpose should give enough insight to the event flow represented in Figure 4.3.

Not all the connections will be set up when the environment is created. To be able to communicate fully with the VRML component, the dynamic code created by the system must also be linked with the Java component, for updating purposes. This is done using the EAI in the same manner as used for the static code of the VRML component, after each VRML code generating operation.

Below is the list of connections made with the EAI:

- connection to the browser. Without this, the environment cannot be accessed at all by the Java component.
- connection to the editing buttons, to their generated events, and to their active/inactive properties.
- a mesh node and a coordinate setting event. The mesh node will always hold the VRML representation of the edited mesh, and the coordinate setting event is used to refresh the visualised geometry.
- selector setting event. As the coordinates change, the position of sensor boxes must also change to provide an accurate representation for the location of mesh vertices.
- events such as those for removing the generated geometry (before a new instance of the geometry is created for visualising the geometry complexity changes) and centering, scaling or translating the mesh (to provide a maximal view of the mesh).
- events to transfer vertex selection statuses (a single vertex, a ring of vertices or a column of vertices can be selected at a time).
- events to handle the editing unit selection process, which sets the magnitude of the editing operations.

Connections are made for all the nodes and events enumerated above. To each event there is a serial number assigned, which makes it possible for a callback method to identify the source event. Depending on what this event's function is, the callback method handles the event locally, or in case the event is more complicated, it triggers a number of methods responsible for handling that particular event and associated calculations.

File operations
The first file handling method is the file reading (loading) method. If the data file is opened successfully, its content is read into a string variable. A parsing method extracts the relevant data from the string and sets up the data parameters used by the Java component, like the number of rings and the number of columns. It also creates a dynamical parameter depending on the previous values to store the vertex coordinates (the number of vertex coordinates is seldomly the same, therefore the variable holding this data is always created dynamically). At the end, the method calls the VRML code generation method described in the next section, sets up initial values for a number of parameters for data consistency, and calls VRML-Java connection initialising methods for the newly generated VRML content.

The remaining two file operation methods deal with the storage of the edited mesh. The save method opens a window to specify the desired name and directory of the file to be saved, then it constructs the internal data file format and fills it with the required data extracted from the Java component variables. The last method, exporting onto VRML format resembles the previous method. The difference is that there is extra data to be generated beside the geometry (header, settings nodes), and the vertex data must be accompanied by mesh facets indexing data. This extra information is also extracted from data variables stored inside the Java component.

Generating VRML code
The VRML code creation method is responsible for creating the mesh geometry, the selection geometry and sensors, the script code that handles the selection process internally in the VRML component, and the events used for communicating with the Java component. It does all this by creating a single string of code, and sending it to the VRML component to be integrated in it. We call this VRML code dynamical because it is generated on the fly (multiple times during the editing process, when the mesh changes complexity) and it is modified constantly, with every operation committed. This dynamic code generation process has the following steps:

- Creating the indexes for the mesh geometry. These indexes specify the polygons of the mesh. Since a regular grid of vertices is used which provides a logical connection between the vertices, every item (cell) of a grid made from the vertices can be covered algorithmically by two triangle facets.

- Creating the mesh geometry string itself, an IFS VRML node, instantiating the data from the parsed vertex coordinates and using the previously created triangle indexes.

- The next step is the creation of sensors, handlers and a script with functions (events) to manage the properties and events of vertex operations. They will be created for *each* of the vertices from the mesh.

- Creating the code that handles the vertex grouping into rings or columns. It contains a line-set geometry, which connects the vertices of the ring respectively column together, reusing the same coordinates the mesh uses.

A touch sensor is connected to the line-set, which triggers the behaviours of the individual box sensors within the line-set.

Figure 4.4 shows the different components of the generated geometry and its control structure.

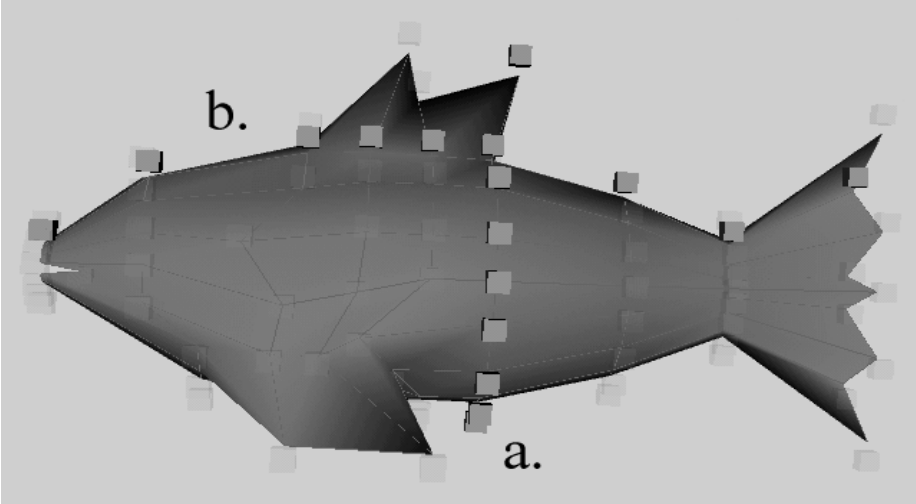


Figure 4.4: Geometry coordinates visualisation with sensors and the underlying mesh; a) ring selection visualisation; b) column selection visualisation.

Geometry operations

The most frequently used method is a method which catches the displacement event transmitted from the VRML component. In function of the current selection, the method applies the displacement to the data stored in the Java component and sends that data back to the VRML component. It also sends the same displacements to the sensor box(es) assigned to the selected vertex or vertices.

The size of a sensor box assigned to a certain vertex depends on the distance of the vertex from its neighbours. This way, sensors do not overlap, and remain scaled according to the complexity and the size of the edited mesh geometry.

A mesh positioning method provides an interface where the visibility of the edited mesh is maximal, being entirely visible and without occluding by rotation the other elements of the interface. It scales and translates the geometry to a convenient position.

The ring adding method inserts a new ring in order after the selected ring (or after the ring containing the selected vertex), except when the selection is the last ring, when no action is performed. The data for the new vertices are calculated as the mean values of the vertices from the neighbouring rings, and the dimension of the vertex data variable is changed to be able to hold the new values along the old values.

The column addition method is essentially the same. The only difference between the two methods lies in the structure of the vertices, and there is no

restriction in adding a column after the last column like in the case of adding rings.

If the ring smoothing method is activated, it calculates the centre of the current ring and depending on the distance of vertices from the centre and from the mean centre distance, the vertex values are increased or decreased to achieve smoother surfaces. The new data is transmitted then to the VRML component.

The ring scaling method affects the vertices of the current ring. For every vertex, distances from the centre position are calculated in the horizontal (XZ) plane, and the scale value (the current editing unit is used as the scaling value) is distributed to the vertex X and Y values proportional to the previous values.

Column scaling is different in functionality from ring scaling. While ring scaling acts locally, only affecting one ring of vertices, column scaling affects all vertices of the mesh. It can be used for enlarging the distance between the ring vertices, thus longer/taller meshes can be obtained. A column centerpoint is calculated (from the current column) and depending on the distances of the current column vertices from the centre point, each ring is displaced proportionally by a certain calculated fraction of the scale value.

4.2.5 Symmetry

The editor was continuously tested and modified during the developing process to eliminate undesirable effects or to enhance its usability. At first the symmetry which is present in many real-world objects had to be introduced by hand to the mesh, making the user's task unnecessarily complicated. To eliminate this overhead with symmetric object modelling, the system has been modified to provide this symmetric editing functionality. The mesh, according to this approach, is edited only on one side of the symmetry plane, the other side follows the changes automatically. This method yields a few advantages. With fewer control points, the size of the generated VRML code drops also: thus a gain in speed can be achieved. By using one side of the object for editing, the other side can be used for preview purposes, as selections are not visible on that side. And as a separate preview side is available, the edited side can use a semi-transparent appearance for non-selected controls, giving more comprehensibility to the mesh structure. The difference between the two approaches is shown in Figure 4.5.

4.2.6 Conclusions

The editor system in this state gives a high level of control over a loaded mesh, allowing precise editing, new mesh construction and instant visibility. In addition, it is also web deployable. However, the system can model only single meshes. Figure 4.6 shows a few models created with our system. Our intent was to extend it to include the ability to model hierarchies of meshes, thus being able to model complex meshes such as H-anim [HAWG, 1999] hierarchies or other hierarchies based on bone structures [Badler et al., 1993] to produce the avatar and agent meshes needed for inhabited virtual environments. The next section provides the description of such a system, capable also of editing character hierarchies.

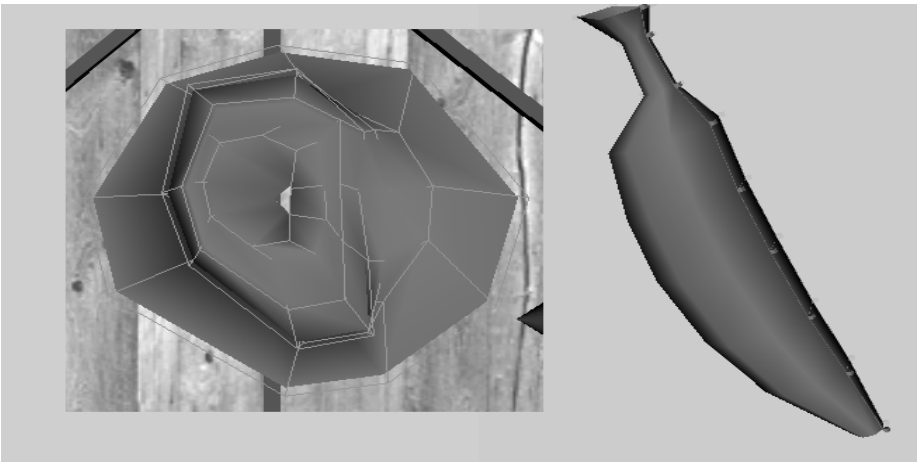


Figure 4.5: Asymmetric and symmetric mesh control structure visualisation. a) Asymmetric mesh with full control structure; b) Symmetric mesh with half-side control structure, mesh geometry is easily observable on the side without controllers.

4.3 Hierarchy editor

This section describes the hierarchy editor system, which integrates the properties of the two geometry editing systems (symmetrical and asymmetrical). This description is based on the [Kiss, 2002] IV2002 paper, which is extended to describe the system in detail. Especially the hierarchy handling methods are explained more minutiously.

The goal of the system is to provide a user-friendly virtual environment for the development of 3D characters with an articulated structure, by transforming the geometry and hierarchy of articulated characters into visible, visually editable entities. The interface allows the modelling of both the Joint structure of the character (the hierarchy) and its Segment geometry (the skin). The main system characteristics are as follows:

- the combination of VRML, Java and EAI web technologies used provide the possibility of on-line modelling,
- the rules and constraints based operations and thus interface elements yield easier and faster modelling operations,
- the system uses vertices and sets of vertices as graphics primitives forming a non-conventional and novice-friendly editing approach,
- provides the possibility to handle and extend hierarchies based on the H-anim structure elements.

For this purpose we continue to use the VRML language and we make use also of the H-anim [HAWG, 1999] specification. The H-anim structure elements “Joint” (for the hierarchy) and “Segment” (for the geometry) are used in the

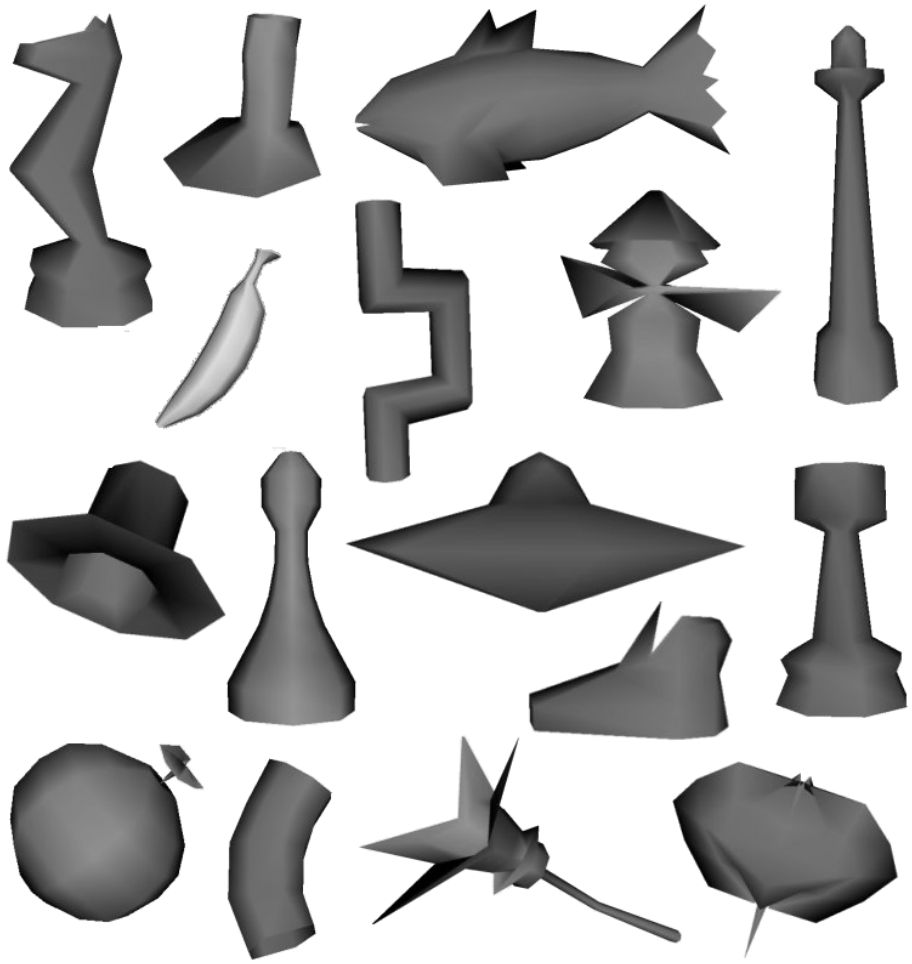


Figure 4.6: Geometry examples.

system as basic components of the articulated characters. The Segment geometries are based on a regular grid of vertices, as discussed in and taken over from Section 4.2.

The system is aimed to be simple, such that users would be able to learn to use it quickly. The complexity of the geometry and hierarchy is user-handled. Since creating 3D characters is a specialised modelling task and our aim is to provide operations on a low level of detail that is still suited to the casual user, an expressive and minimal user interface has been built on top of the modelling operations.

Just like the previously described geometry editing system, this system consists of a VRML virtual environment, a Java applet and the EAI component put together and used as an interactive system that is able to dynamically create and update the different visualisations of the edited character (hierarchy, geometry), usable on-line and off-line within a web browser.

Complex modelling systems like Maya Personal Learning Edition (PLE)

[Alias—Wavefront, NA], GMax (simpler version of Studio Max) [Discreet, NA], Blender (now open source) [Blender Foundation, NA], CosmoWorlds (VRML modeller) [SGI, NA], etc. tend to be very general and thus present an overwhelming interface to the novice or casual user. They certainly exceed in possibilities our system, which aims among other to eliminate the steep learning curve usually associated with state of the art modelling tools. Our system specialises in building articulated hierarchies and the related geometries which result in simpler, web environment friendly 3D characters, and the constraints of this task allow the use of a simplified interface.

The system breaks the interface - environment duality: since we model a character at a time without modelling its surrounding environment, the entire interface, with the exception of a few file-oriented buttons, is situated inside the virtual environment. This allows camera freedom (the user can move around freely) which is enhanced by positioning widgets (researched in [Bier, 1986], [Conner et al., 1992], [Döllner and Hinrichs, 1998] among others) and different data visualisations, making multiple views unnecessary (Multiple views are also not very appropriate for the limitations of current on-line technologies and Internet access speed limitations). For the data visualisations, we separate the hierarchy from geometry, and we use a semantically linked and rule-based hierarchy and modelling approach.

We try to achieve selection indirection. Our system, in contrary with GMax and MilkShape 3D [chUmbaLum sOft, NA] for instance, provides direct selection functionality: the type of selection is handled automatically by using non-overlapping selection sensors instead of explicit selection type buttons.

Specialised avatar creation tools (for on-line environments) like Avatar Studio [Blaxxun interactive, NAa] and Avatar Lab [Curious Labs, NA] are based on pre-built body components which can be parametrically altered for longer/shorter, thinner/thicker, etc. results. Geometry complexity is fixed, as is the number of Segments. They do not provide flexibility in hierarchy construction and geometry complexity.

Since we work with web-based technologies, this allows modelling to be performed on-line, and the whole system could be transformed into a collaborative editing environment if it is connected to a central database.

At last but not least, modelling systems have high computing requirements that are usually accessible only for professional users. Our system is much lighter: it is a little over 200KB of code that runs on much more modest requirements.

4.3.1 Framework

The system consists of two interacting components embedded in a HTML hypertext file. The 3D environment (VRML world) holds the static environment, the widgets (buttons and other controls) and the containers for the dynamically created VRML content that visualise the different aspects of the edited character: hierarchy, whole character or selected Segment, all equipped with sensors for selection handling.

Next to the VRML component there is a Java applet which stores all the data that is assigned to the modelled character, and specifies the methods to create and update the dynamic content of the VRML component using the

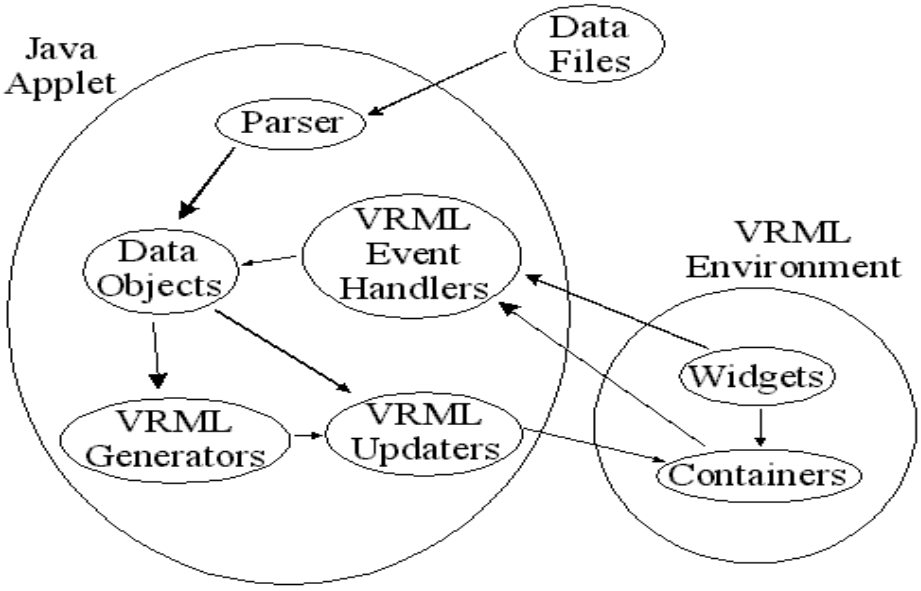


Figure 4.7: System components.

EAI communication channels. A simpler diagram of the connections between the elements of the two components is shown in Figure 4.7. The multitude of individual connections would be impractical to visualise, since it is a two to threefold increase over the Segment editor flowgraph visualisation in Figure 4.3.

4.3.1.1 VRML component

Besides the static virtual environment, there are three VRML containers that hold the dynamic scene components. Their functionalities are:

- "Character Container" - a container for the whole character which holds all the Segment geometries of the current character. The Segment geometries are selectable.
- "Hierarchy Container" - a container for the hierarchy which holds a visual representation of the Joint positions, terminal points and connections in the current character hierarchy. Each Joint and terminal point is selectable.
- "Segment Container" - a container for a single Segment which represents the geometry of the currently selected Segment. All primitives are selectable.

The content placed in these containers is created and updated by the Java component (see Figure 4.7), which also listens to the events produced by the control structures from the container. These updates are initiated with the widgets/buttons of the VRML component. In addition, widgets from the VRML component can also change properties of the containers directly (the position and orientation, used for preview and interface managing purposes).

The system is functionally divided in two editors. The first one is the hierarchy editor which visualises the position of Joints in the character and the connection between the Joints (the Joint hierarchy) with selection sensor boxes respectively Joint relation lines. This editor permits the creation and modification of the underlying hierarchy of the character and influences the content of the hierarchy container and the content of the whole character container.

The second editor is a geometry editor which is used to visualise and edit the surface of single Segment geometries by representing the surface coordinates with sensor boxes and coordinate rings and columns with sensor line-sets. It is an instantiation of the geometry editor with selected character Segment geometries. This editor influences the container of the selected Segment and the whole character container: thus every change in a single Segment of a character is repeated on the whole character visualisation for data consistency.

4.3.1.2 Java applet component

To enable a high level of interactivity, the system is capable of reading in the data, creating data objects, visualising them, listening to the visualised environment events and consequently modifying the data. This needs behaviour specification (links between the Java and VRML components) that are dynamically created, changed and updated during the modelling process. One important task is managing the communication between the dynamic VRML content and the Java component. This communication is also dynamically defined, since it must link to dynamically (run-time) created VRML entities (nodes or events). This is achieved using node and event retrieval based on indexes calculated from the data quantities stored in the Java component.

4.3.1.3 Primitives

The primitives used for geometry editing are described in Section 4.2. To show that the primitives used are enough for the purpose of character modelling, the following geometry examples have been modelled with our system (see Figure 4.8): a human torso and an ear.

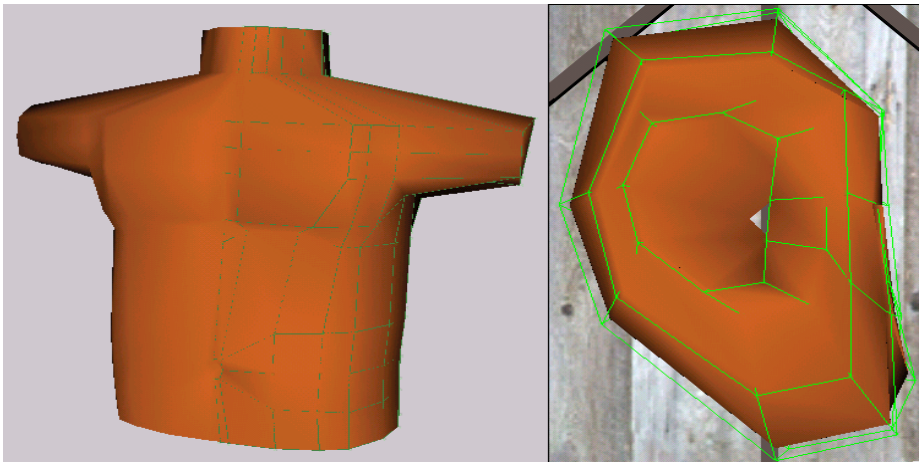


Figure 4.8: Geometry examples.

The primitives for the hierarchy editing are simple coordinates in space, specifying Joint centre positions. The sensor handling of these coordinates are also simple, taken over from the geometry vertices handling approach. The difference lies in the effects of actions effectuated on selected Joints: these have an extended scope, they act on Joint centre position and whole Segment geometries, providing a higher level manipulation possibility that a single coordinate sensor might suggest.

4.3.1.4 VRML constraints

The system is built upon the open VRML format. This enables 3D interactivity and web-based interaction. VRML, although the "Language" word is part of its name, is more a file definition format than a programming language. This file format uses the graphics primitives coordinate, (multi)line and polygon, the latter ones building on coordinate arrays and indexes. This also influenced the ring and column primitives used in the system, since these are collections of coordinates that can be applied directly in geometry composition.

There are no VRML constraints for hierarchy, as it was in the case of the geometry editor. The hierarchy visualisation uses also the basic vertex construction to represent the hierarchy Joint positions, but without the limitations that arose from the geometry construction properties.

VRML is interpreted, and its interpreters are freely available browser plugins. Stand-alone viewers are also available, but those do not always support the Java platform for programming the interaction and most certainly they do not support the EAI method for programming, since that is based upon inter-plugin communication. In this context, the hardware considerations are addressed within the browser companies, and our system relies on them for on-screen display.

4.3.1.5 Direct manipulation

Most modern graphics packages use direct manipulation methods to eliminate indirection: the target and the operation are situated in the same spot. This is fine in a modelling package where there are a few two-dimensional views (GMax, Maya PLE, Blender, MilkShape 3D, etc.), where the mouse two dimensional handling possibilities can be directly mapped. However, our manipulation approach is slightly different since we solely use a three dimensional view. We apply direct manipulation in the case of selections and positioning/viewing widgets, which are in turn indirected in the mentioned modelling packages. Most of our widgets provide also a direct manipulation approach.

4.3.1.6 H-anim(+)

The H-anim specification provides a standard way to describe human-type characters (avatars). It specifies a number of 93 (in version 1.1) Joints with their corresponding Segments. The Joints and Segments have standardised names and there is measured data available about Joint positions and flexibility. For different purposes, anchor coordinates can be defined for Segments (e.g. for glasses, tie, watch, etc.) in the form of Sites. Displacers can be used to deform the Segments, when additional animation is required besides the Joint rotation method.

There are variations of hierarchy complexity that are suggested by the H-anim standard for H-anim type characters. They are called Level Of Articulation (LOA), and there are a number of variations of them. There are Joints that are required by the specification for a minimal H-anim conformance, but other than that different variations can be built.

In our system there is no restriction on the names, positions and hierarchy that could be built: thus there is no H-anim validity checking. However, if one uses an existing H-anim model just to adjust the 3D character's geometry or to extend it (if not already a full H-anim hierarchy) by naming the new Joints and Segments in a H-anim compliant way, then the result will be a H-anim compliant character. This approach gives the possibility to extend the H-anim character space to non-human characters with more or less than two legs, two arms, one head or to add different extensions: tails, fins, wings, etc.

In our system, we only use the rules between the H-anim components defined by the specification, and disregard the rules of defined Joint names and connections. We do this for the obvious reason that we do not want to restrict the user to human models. Users might want to build avatars in the form of animals and other entities (TelePresence's multi-user software had an avatar in form of a wooden board).

The hierarchy rules are:

- a Joint can have multiple Joint children,
- every Joint has one and only one Segment child,
- all (except the root) Joints have one and only one parent Joint.

4.3.1.7 Hierarchy

The user has the possibility to use and modify predefined H-anim and other hierarchy variants or to create new types of hierarchies by adding or removing Joints. A few example hierarchies are presented in Figure 4.9.

The hierarchy operations mentioned (visible in Figure 4.11) bear the marks "Joint" and none of them uses "Segment". This is a consequence of the rules stated in Section 4.3.1.6. Since each Joint has one and only one Segment child, they do not influence at all the hierarchy if their parent Joint is already a part of the hierarchy. Thus they are handled automatically in the background.

Hierarchy changes are reflected on the Segment geometries. Hierarchy expansion creates new Segments and their geometries. Hierarchy refinement divides Segment geometries. Collapsing hierarchy Joints means concatenation of geometries if the mesh complexity allows it.

Modifying the Joint positions results in the automatic modification of the geometry in cases when the parent-child relationships makes it possible. The exception appears at multiple and non-symmetric children Segments, when there is no clear way of choosing appropriate geometry changes for all child relations without the danger of interfering with the user's preferences. When the relationships allow the automatic geometry modifications, the geometry of the selected Joint (also the symmetric geometry if exists) and the geometry of the Joint's parent Joint deform to follow the new Joint position. This is very advantageous

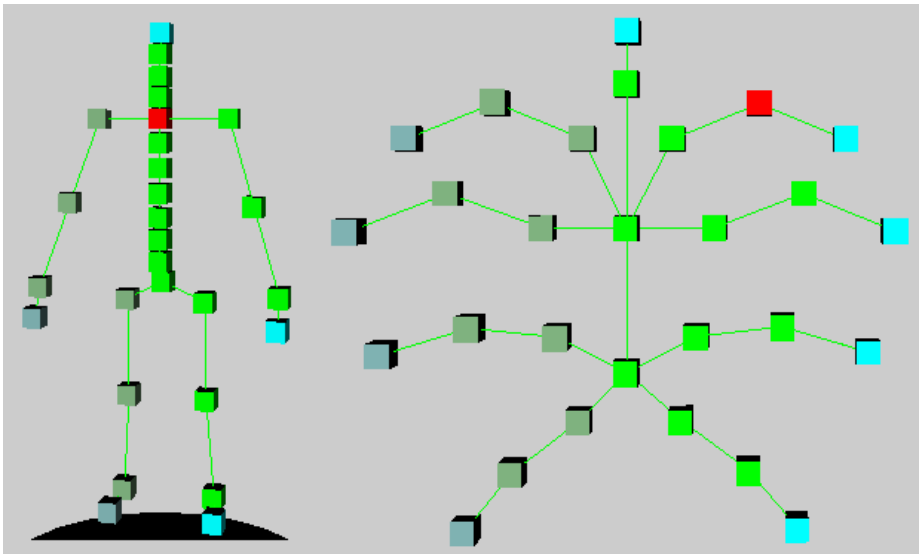


Figure 4.9: Hierarchy examples: human and spider.

in case of human ankles, knees, elbows or when the torso is segmented, by automatically keeping the relation between the affected Segment geometries. In this way, different body characteristics can be produced quickly if needed.

4.3.1.8 Geometry

Each hierarchy Segment defines a piece of geometry, and the character mesh is the sum of these Segment geometries. Currently in our system the coordinates are stored separately in each Segment geometry, but they can be also stored in a unified "Coordinate" VRML node (the H-anim specification allows both methods, but recommends the latter one). The Segment geometries of the hierarchies of Figure 4.9 are presented in Figure 4.10.

4.3.2 User interface

The interface consists of two components: a Java part (only a few Buttons) and the VRML interface (see Figure 4.11). The Java Buttons are used simply to trigger file opening and saving operations, while the VRML interface is used for all the editing operations. It is our intention to integrate the Java Buttons functionality into the VRML environment next to the editing functionalities for adding consistency to the system and eliminating a double interface. Since the goal is to create a single character at a time without modelling the target environment, it is possible to integrate the edited character and all the widgets directly into a VRML world where a real-time rendered preview of the character can be obtained at any moment.

To aid simplicity, only relevant widgets/buttons are visible in the interface, and they are visible all at once (even those deactivated due to different selection contexts). This approach helps by revealing the available modelling options, as well as the selection consequences.

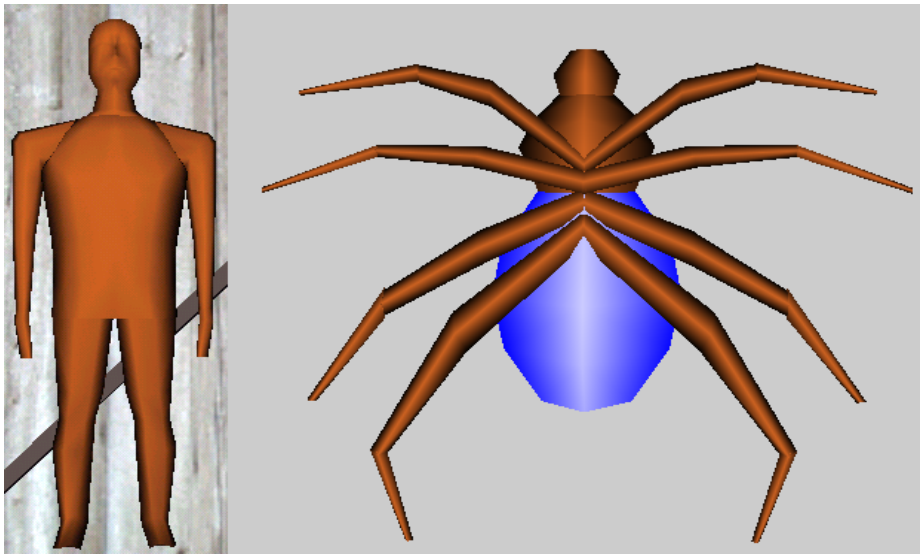


Figure 4.10: Character examples: human and spider.

4.3.2.1 VRML interface

The VRML environment contains the main editing interface. In this environment there are three different visualisations of the edited character:

- a full character visualisation (the collection of Segments, noted with 1 in Figure 4.11) which serves for preview and for Segment selections,
- a hierarchy visualisation (the Joint structure, noted with 2) that serves for editing the hierarchy,
- a Segment geometry visualisation (a single Segment geometry, noted with 3) that serves for editing the geometry of the selected Segment.

These three visualisations are displayed simultaneously in the environment when a single Segment geometry is edited (with the hierarchy visualisation set aside, but still viewable for reference), but when the hierarchy is edited, the Segment geometry visualisation is discarded, since the geometry changes introduced by the hierarchy editing process can be viewed on the whole character as well. The Segment and Joint selections are interlinked (equivalent) due to the 1:1 relationship (see the rules in Section 4.3.1.6) and the editing mode determines the active selection. This type of visualisation is used as a compromise between displaying relevant information and overwhelming a casual user with too much information.

There are two editing modes that can be switched by a single button click (no.4 in Figure 4.11): editing the hierarchy or editing the geometry of a selected Segment. Each of them have specific interface components (described in Section 4.3.2.1), but there are also universal widgets. These universal widgets also apply for the hierarchy editor, although they have been introduced for single geometries. The first of the two universal widgets is the editing widget (no.6 in

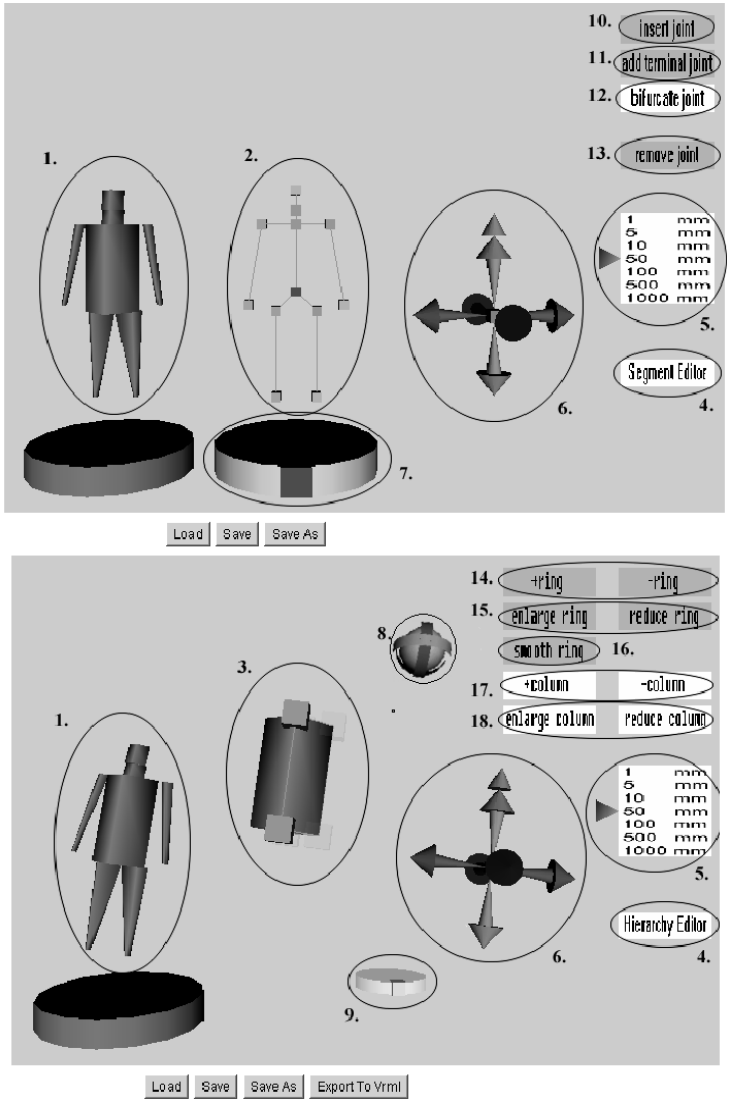


Figure 4.11: The editing environment in two stages, on top with the Hierarchy editor active, on the bottom with the Segment editor active. The elements are: 1.Whole character, 2.Hierarchy representation, 3.Selected Segment, 4.Editor selection buttons (superimposed), 5.Unit widget, 6.Edit widget, 7.Hierarchy rotation widget, 8.Segment rotation widget, 9.Zoom widget, 10-13.Function buttons: Insert Joint, Add Terminal Joint, Bifurcate Joint, Remove Joint; 14-18.Function buttons: Add/Remove Ring, Positive/Negative Ring Scale, Smooth Ring, Add/Remove Column, Positive/Negative Column Scale.

Figure 4.11). It consists of six arrows representing the six degrees of freedom (colour coded). They are sensor enabled, and clicking on them would mean that the currently selected coordinate(s), whether they represent a Joint position, a single point or a set of points from a Segment geometry, will be displaced in the direction the clicked arrow points into. The other universal widget, the unit selection widget (no.5 in Figure 4.11) allows the selection of values for the magnitude of all the modelling operations. The bigger this value, the coarser the modelling operations will be. The edit widget and the different functions of the two editors use this value for coordinate calculations.

Hierarchy editing mode interface
When a hierarchy file is loaded, a visualisation of the Joint hierarchy (skeleton) is generated from the information found within the file. The Joints are represented by green-red (red when selected) sensor boxes and their parent-child relationship is represented by IndexedLineSet line geometries connecting the related Joints. There are also terminal sensors and terminal lines, which represent the Segment lengths of terminal Joints and the endpoints of terminal Segments. These sensor boxes are coloured turquoise-yellow (yellow when selected) and function also as visualisation aid while they provide the displacement functionality for the modelled character's endpoints. These sensors do not participate in hierarchy modifying functions.

The only local widget of the hierarchy editor is a pedestal (the rotation widget) on which the hierarchy representation is placed. By dragging the pedestal, the user can position and preview the skeleton (and at the same time the whole character) from any desired position along the Y rotation axis.

The hierarchy building functionalities of the editor are accessible using the function buttons. There are four function buttons defined:

- "Insert Joint" button. If this button is activated, it adds to the hierarchy a Joint as the parent of the selected Joint and as the child of the selected Joint's former parent. The insert operation inserts a new Joint between a Joint child-parent relationship (and not between a parent-Joint relationship) to take advantage of the specific property of one Joint having one parent versus the ambiguous property of one Joint having possible multiple Joint children. The functionality is enabled for all Joints except the root Joint, which does not have a parent to which to add the new Joint. According to the symmetry rules, if the new Joint's parent is an internally symmetrical Joint, then the new Joint will be also internally symmetrical, else if the parent Joint is an externally symmetrical Joint, then another Joint is added to the parent Joint's symmetrical Joint, and the two new Joints will be the externally symmetrical Joints of each other.

- "Add Terminal Joint" button. This function button is enabled only if terminal Joints are selected. It extends the hierarchy with one additional Joint using from the former terminal Joint and Segment their direct properties (coordinates) and their indirect properties (length, orientation) to generate new properties. The same symmetry rules apply as with the previous function button.

- "Bifurcate Joint" button. Clicking on the button will add two Joints as the children of the currently selected Joint. These new Joints will split

the hierarchy line at the selected Joint, allowing more diversity in the hierarchy. The function is enabled with every Joint selected, thus this function can be used for hierarchy refinement and for extending the hierarchy as well (by applying the function to internal respectively terminal Joints). The same symmetry rules apply with this function also, but with the slight change that in case of external symmetry, two pairs of Joints are added, one pair for each of the selected Joint respectively for its externally symmetrical Joint.

- "Remove Joint" button. This function performs Joint removal operations, with the scope of reducing the complexity of hierarchies. It is enabled with every Joint selection except for the root Joint and removes the Joint from the hierarchy, but preserves the geometry associated with the Joint removed and appends that geometry to the parent Joint's geometry (Segment concatenation is performed).

Segment editing mode interface
There are widgets that are local to the Segment geometry editor, and these are a geometry rotation widget, a ring of coordinates rotating widget and a zoom widget. The rotation widgets are represented as an ensemble of three colour-coded circular stripes which have rotation sensors linked to them. By dragging these stripes, the user can view the geometry from all angles and can position the geometry as desired in the course of the editing process or the user can rotate whole rings of coordinates for a quick mesh editing option. Since the rotation axes are separated in the widget, a precise handling is possible. The zoom tool eliminates the difficulties of editing meshes that are too large or too small to be managed by zooming them to a comfortable size level.

The remaining elements of the geometry editor are a number of function buttons. Their functionality is to transmit their activation status to methods in the Java applet that will calculate and effectuate the actions related to the buttons.

4.3.3 Data handling

The system uses a simple text format to store hierarchy, Joint, respectively Segment information. The main hierarchy file is a text file representing hierarchy relationships for one single character. In this file, for each Joint there is a line record specifying the Joint's name, the name of its child Segment, the name of its parent Joint and for later use lists for child Site and Displacer names. The internal hierarchy information stored in Joint and Segment objects is extracted from this main file. An example hierarchy file is as follows:

```
This is the file header, until the first hash mark
Format:                              |--> optional
jointname     parent     segment    site[multiple] disp[multiple]
where multiple means multiple instance names, separated by a
comma, semicolon and without white spaces, to ease the parsing
#
RootJoint[@int]         NULL            torso
l_hip[r_hip]            RootJoint       l_leg
```

```
r_hip                    RootJoint       r_leg
vt1[@int]                RootJoint       t1
skullbase[@int]          vt1             skull
l_shoulder[r_shoulder]   vt1             l_arm
r_shoulder               vt1             r_arm
#
```

The Joint and Segment files (there is one separate file for each Joint and Segment) are in a subdirectory which is specified by the hierarchy file's name. This method of file handling has been used for the possibility to integrate at a later stage other entities of the H-anim standard: Sites and Displacers, as well as other extensions, for instance animation data files. File segmentation could also provide the possibility to interchange the different Segments/Joints to compose new characters from existing components. The Joint and Segment files are simple text files which store the H-anim based properties of their owners. We already presented the Segment file format in 4.2.3. The Joint equivalent is even more simple, since less information is stored, as the following example also suggests:

```
header until hash mark
File contains: centre, rotation, scale, scaleOrientation,
translation, ulimit, llimit, limitOrientation and stiffness
values of the H-anim specification
#
centre -0.25 1.5 -0.05
rotation 0 1 0 0
scale 1 1 1
translation 0 0 0
...
#
```

Finally, the system has the option to export the edited character in H-anim style into a VRML file specified by the user. It uses the Joint and Segment nodes style specified by the H-anim specification. Java3D export can also be included, since the H-anim specification is file format independent and our Java3D geometry export functions are already used in a separate geometry editor tool.

4.3.4 Symmetry

The system handles symmetries automatically to make the modelling task easier. The types of symmetries used is internal and external symmetry, applicable for both Joints and Segments. Since the H-anim standard specifies the default character position as facing the viewer (It is looking into +Z direction.) the symmetry plane can be considered the YZ plane where X=0 (defined by the root Joint and other internally symmetric Joints). Examples for symmetry are shown in Figure 4.12.

4.3.4.1 Internal symmetry

In case of internal symmetry, the Joint's Segment child has geometry coordinates that are the symmetric values of each other with respect to the symmetry

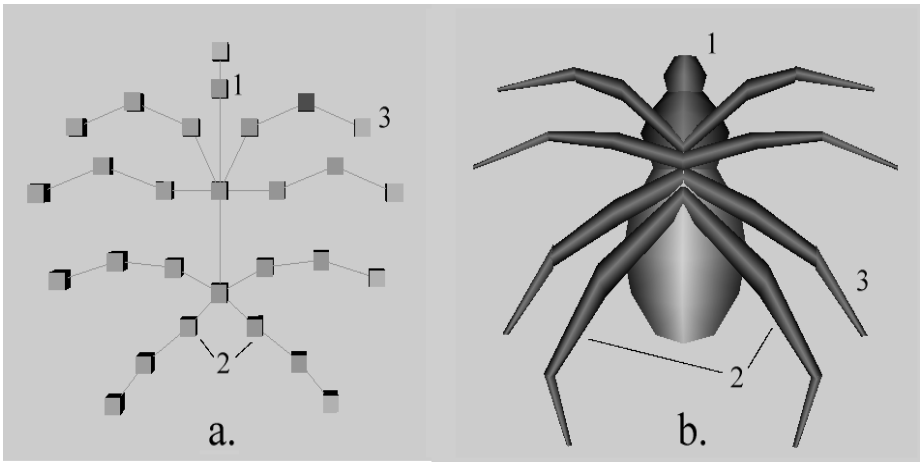


Figure 4.12: System visualisations. a) Joint hierarchy visualisation with sensors and parent-child connections; a_1) Internally symmetric Joint; a_2) Externally symmetric Joints; a_3) Leaf coordinate (centre of Segment's last ring coordinates), externally symmetric; b) Character geometry visualisation: all of the Segment geometries belonging to a Joint hierarchy; b_1) Internally symmetric Segment; b_1) Externally symmetric Segments; b_1) Terminal Segment, externally symmetric.

plane. This is relevant to the geometry editing: when an internally symmetric Segment is edited, its control structure is built only for half of the geometry coordinates and the other coordinates are automatically adjusted to maintain the symmetry. Additionally, Joint centerpoints and Segment coordinates that are on the symmetry plane are not allowed to "escape" from that plane. Regarding the hierarchy editing operations, internal symmetry influences the created new Joints as follows:

- if a single Joint is added and the parent Joint is an internally symmetric Joint, the new Joint will be also internally symmetric,
- if two Joints are inserted to an internally symmetric parent Joint, the new Joints will be the externally symmetric Joints of each other.

4.3.4.2 External symmetry

External symmetry means that a Joint and its Segment child has a symmetrical Joint-Segment pair whose centre point and coordinate values are symmetric with respect to their stored values. With external symmetry, the hierarchy controls are enabled only for one side of the hierarchy and the operations effectuated on them are mirrored to their externally symmetric Joints. In case of Segments, the control structure now covers all the geometry and as in case of Joints, the operations effectuated on them are mirrored on the externally symmetric Segment's coordinates. The general hierarchy expansion rule for external symmetry is: all the inserted Joints will have an externally symmetric Joint regardless of the parent Joint's symmetry type.

4.3.5 Conclusions

In our environment, the user does not have to handle the complexity of commercial [professional] graphics modelling packages for shape control, nor the limits of predefined body components present in specialised modelling packages in case the user needs a customised human character or types of characters other than human. However, next to the direct manipulation possibilities, our system presents a degree of indirection for Joint centre and geometry coordinate displacements (a separate, non-drag edit tool), which can be remedied by using specialised widgets.

Our approach states a useful goal and provides an interesting and useful solution. While it is not the most powerful tool available, it meets the goal of easy usability and provides a unified and consistent design. The design works within the context of VRML and EAI, exploiting the possibilities supplied by these technologies.

As an advantage from using regular geometries, the system can be easily converted into using NURBS-based geometry [Grahn et al., 2000] instead of polygon based geometry and even lattice-based geometries [Wakita et al., 2000] could be used. Since the regular grid coordinates satisfy the control point requirements of NURBS surfaces, they can be easily converted into control points (using appropriate dimensions, the surface will even pass through the control points as in case of polygons, retaining the general shape of the geometry). Then the same primitive operations (vertex, ring and column operations) apply to the NURBS control points, providing a high level manipulation possibility (both for complexity and surface).

One aspect of the system which can be refined is the manipulation aspect. Currently, editing is done with indirect manipulation, using a clickable editor widget targeted to the selected primitive. The editor widget and unit widget combination can be replaced with a directly manipulated editing widget similar to the one presented in Section 4.4.3. Such a widget should then operate on a number of different levels, covering a multitude of selection options: single vertices, ring/column collections of vertices, Joints, or even terminal points of Segment geometries.

On a more technical note, the visualisation of parent-child relations could be improved to show the direction of the relations (which is the parent and which is the child) for instance with a traditional diamond shape used in graphics packages. The current line representation has been chosen for simplicity.

The system is being extended to include more functionalities. These planned extensions will provide animation possibilities (including deformations via Displacers), Sites (additional elements of the H-anim specification) and textures. In the case of texturing a per-coordinate texture manipulation approach is used in a separate geometry editing module that is being integrated into the system.

4.4 Other editors

We have built a number of smaller environments for testing different editing methods. We built a NURBS editor based on the geometry editor system, a texture applying environment for texturing geometries and we tested the usage of drag edit buttons on a NURBS geometries and editing controls generating

application. In the following, we present shortly these applications.

4.4.1 NURBS version of the Segment editor

NURBS geometry possibilities are introduced as extensions in current VRML browser plugins by the plugin creator companies. Due to their usefulness and probably also due to increased computational and graphics power of today's computers, these and other types of extensions are being specified by the next version of the VRML standard, with code name X3D.

Our interest in NURBS lies in the fact that these graphics entities provide smooth surfaces out of a relatively small number of coordinate values (NURBS control points), and since these coordinate values are organised following a regular grid pattern, our geometry coordinates are perfect data sources for NURBS geometries.

With small modifications, our geometry editing system can function as a NURBS geometry editor. These modifications include the different generated geometry (NURBS Surface instead of IndexedFaceSet) and the different channels of communication with the applet that handles the interaction events. All other components of the geometry editing system can be used without further delay. The NURBS geometry surface is constructed with an order of (4x2) that constrains the geometry to pass through its control points. This is certainly, together with the fact that the weight property of the NURBS geometry is also not modifiable, a heavy constraint on the NURBS surface possibilities. Still, the result is a smooth surface that is much more compelling that the polygon based geometries, while this modelling approach requires no extra effort from the user.

Figure 4.13 provides a polygon and a NURBS version of the simplest coordinate grid (4x2 coordinates, defining a rectangular block), and the NURBS geometry surface of more complex geometries.

4.4.2 Texture editor

The texture editor is a test application to show a geometry texturing approach based on the individual visualisation and direct manipulation of texture coordinates on a texture panel, while the geometry is textured based on the user set texture coordinates. The user sees the whole texture available in a separate texture panel, the texture coordinates the geometry uses within this texture panel, and has the possibility to drag the texture coordinate sensors, updating this way the geometry texture positions as wished. Figure 4.14 shows a simple geometry and a texture, as well as the connections between the geometry and texture: the texture coordinates of individual geometry vertices. In this particular picture, the tile numbers have been reduced on the left side of the geometry and the tiles have been made slanted on the right-hand side of the geometry.

4.4.3 NURBS surface editing interface generation

To test a directly manipulated edit widget that could eventually replace the click-based editing widget now in use, we created a NURBS surface and edit controls generating application. This application generates NURBS surfaces with the order and dimension specified by the parameters. It also generates for

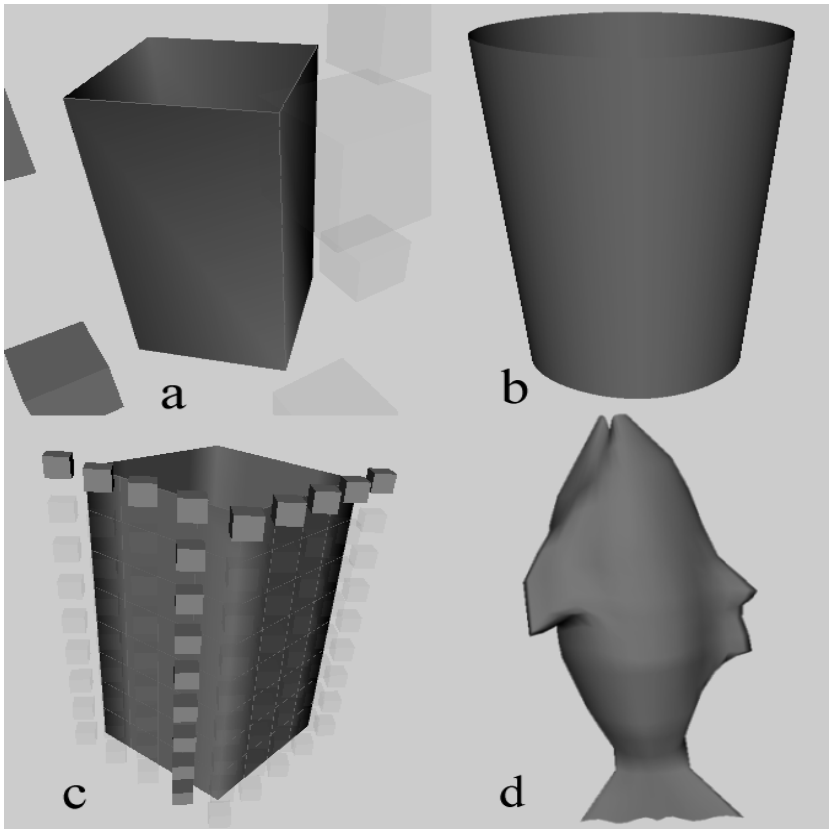


Figure 4.13: NURBS examples. a) Simplest IndexedFaceSet geometry (a box); b) the NURBS surface with the coordinates of the previous IFS as control points; c) NURBS geometry converted from the coordinates of a refined (detailed) box; d) a more complex NURBS geometry.

each of the control points a separate edit widget, consisting of three drag sensors in form of small slider sticks, visible in Figure 4.15. The widget drag sensors convey displacement actions along the three DOFs they represent to the individual control points of the NURBS geometry, providing a direct manipulation tool for the control points, which can be applied also for IFS geometry coordinates, Joint centres and with some positional and visualisation considerations it could be applied even to collections of IFS coordinates.

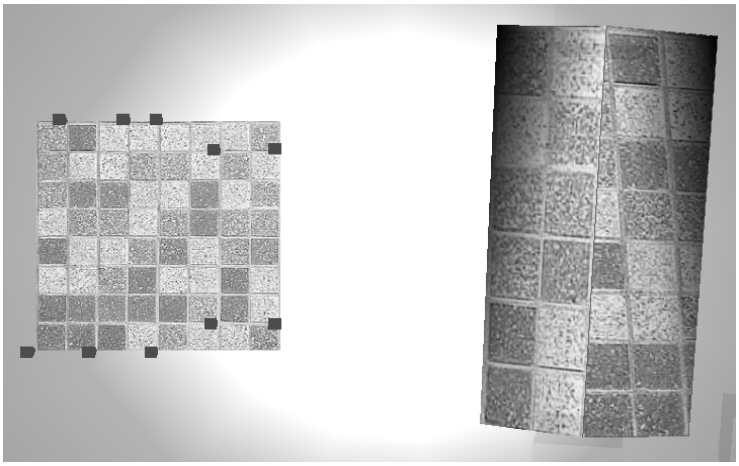


Figure 4.14: Visual texturing application

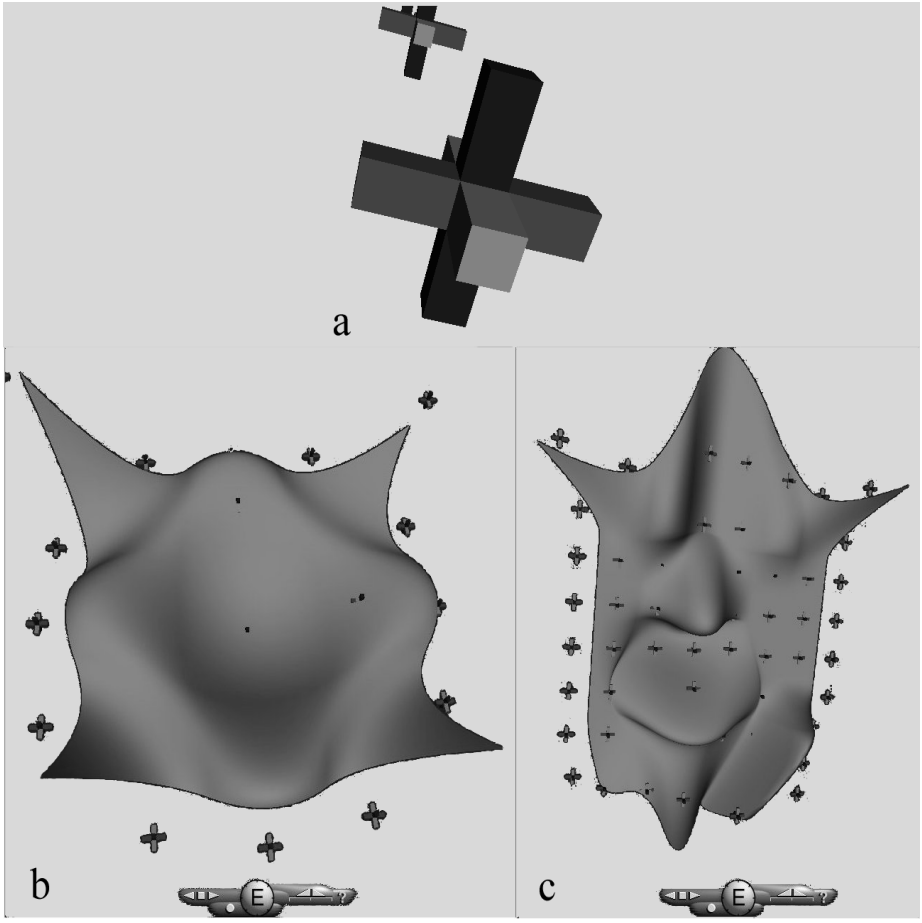


Figure 4.15: NURBS geometries and control widgets. a) Edit widget for the control points; b) edited NURBS geometry, with dimension 5x5 and order 3x3; c) edited NURBS geometry, with dimension 8x8 and order 3x3.

Chapter 5

Assessment

In this chapter we present a comparison between our system and different current desktop modelling systems. In our case we opted for a properties and features based comparison method instead of a more formal one. We did this due mainly to the lack of 3D assessment methods, standards and conventions (noted also in [Conner et al., 1992]), but also due to the inapplicability of general 2D methods developed for WIMP or other interaction paradigms. The latter inapplicability issues are presented in [MacKenzie and Buxton, 1992], regarding Fitt's law of screen target acquisition. Recently, there have been similar assessment experiments, regarding manipulation object, cursor and target object size relations [Wang and MacKenzie, 1999]. However, this is not applicable to modelling interfaces where view manipulation and zoom operations take commonly place.

We have chosen the modelling systems to represent both specialised modellers like CosmoWorlds [SGI, NA], commercially viable packages like Maya Personal Learning Edition [Alias—Wavefront, NA] or GMax [Discreet, NA] and even open source or freely usable tools like Blender [Blender Foundation, NA] or MilkShape 3D [chUmbaLum sOft, NA], all having in common the ability to handle VRML models (at least at the import/export level) and the ability to handle hierarchies. The main aspects of the comparison are as follows:

- graphics primitives and components used,
- selection mechanism,
- mesh manipulation mechanism,
- positioning mechanism (viewpoint selection),
- indirection occurring in the above three mechanisms,
- manipulation diversity, speed (inherent from indirection) and precision (which expresses quality),
- bone hierarchy properties: linked, unlinked, etc.,
- individual properties, characteristics.

These aspects describe the usability and interactivity degrees of a system. However, they do not characterise the simplicity and the learning curve needed for these systems, which also depend on the task complexity that such systems are capable to handle. From this point of view, our system is closer to the character modelling packages MilkShape or GMax, both oriented to lower level character editing.

Note that we do not treat the regular grid criterion as an impediment. On the contrary, we try to analyse and pinpoint the advantages this approach yields in the interaction and modelling process, since this approach gives the possibility of pre-set vertex collections that are handled as primitives, both in the selection and manipulation context.

5.1 System reviews

The following sections describe the characteristics of different modelling approaches, both VRML and generally oriented. Each system is analysed for its possibilities, shortcomings and peculiarities. By analysing primitive, selection, manipulation and other modelling options, we can gain information to the level of indirection, ease of use or learning curve properties of these systems. The following is a list of pro and contra behaviours that we associate to the different properties that a modelling system presents:

- Graphics primitives. For web-based or game interactive systems, a low-level primitive approach is more appropriate. Parametric meshes, where the mesh complexity is automatically calculated are too expensive for such systems. Instead, these systems tend to use textures to enhance rendering quality. Vertex, edge, polygon or even vertex collections as primitives provide different levels of manipulation with a low-level mesh complexity handling.

- Selection mechanism. The easier is to select the different primitives, the better a system performs. Requiring a separate selection mode or even selection type specification slow down the user, and thus its efficiency. Selectable entities are conceptually and visually separate, so assigning selection sensors to them has no technical obstacle.

- Mesh manipulation. Again, a separated manipulation mode provides some degree of indirection. However, contrary to the selection mechanism, in this case the manipulation types are acting on the same visual (selected) entity, thus they are mutually exclusive, introducing a necessary degree of indirection. An extra modality which adds value to a system by reducing the manipulation type indirection is the usage of widgets that provide quick, visual manipulation options with different parameters.

- Positioning mechanism. This component helps the user to view the mesh from a favourable angle, for effortless selection and manipulation possibilities. Any positioning option that goes beyond the usual, historical 3+1 view and still provides precise handling, is a noteworthy extension.

- Indirection. This property looks at the succession of operations that have to be effectuated by the user to achieve a manipulation task. As shorter

this list of operations is, as less indirected and better an interface becomes. Eliminating selection type, selection mode, manipulation mode, manipulation parameter setting operations results in direct manipulation possibilities, which usually also leads to a more natural interface.

- Manipulation quality. Diversity, speed and precision are key properties for the manipulation options. More diversity means a wider range of models, speed means shorter execution time, while precision helps in accurate modelling. The higher these values are, the better a system performs. There is also one drawback with diversity: if a certain, subjective complexity level is exceeded, the learning curve becomes too high to justify the effort of occasional or non-professional users.

- Hierarchy. The possibility to edit some sort of hierarchies is present in each of the analysed systems. This is not an internal property of the systems, rather it is more a task that can be executed on the particular systems. How a user executes this task (what options, tools are available and how they perform) is the assessment modality in this case.

Section 5.2 provides an overview of the system properties and a quick comparison modality with a summary table containing the most important system characteristics.

5.1.1 MilkShape 3D

General description. MilkShape 3D is a low-level polygon modelling, 3D character and animation editing package. It is capable of handling different game character formats and reportedly H-anim plugins have been also created. The system has been built with the purpose of modifying and creating 3D game characters of different formats. It uses the historical 3+1, 2+1, etc. views, although it is customisable by the user. The system does not concern itself with 3D scene modelling.

Primitives. The basic components of the system are the classic vertex and face elements. The edge primitive is not used in modelling. An additional primitive of the system is a joint component, used to specify character hierarchies and animation. The possibility to create a limited number of groups from primitives is also provided. This requires manual setup from the user's side, but at least the created groups are persistent, ready to be reused at a later time.

Selection. The selection mechanism of the system exhibits lack of naturalness. For a selection type that differs from the previous selection type, the new selection type has to be specified, whether it is a joint, a vertex or a face. This is a strong indirection factor, since normal modelling operations require many selection type setting operations, especially at the casual user level. More advanced users would try to eliminate the unnecessary indirection by planning and effectuating operations grouped or by hiding vertices. However, this approach does not eliminate the indirection, only reduces it to a small extent.

Manipulation. The manipulation mechanism is built on the same principles as the selection mechanism, showing the same indirection behaviour. The manipulation mode and selection mode are even mutually exclusive: only one mode is active at a time, increasing the indirection factor. Manipulations are

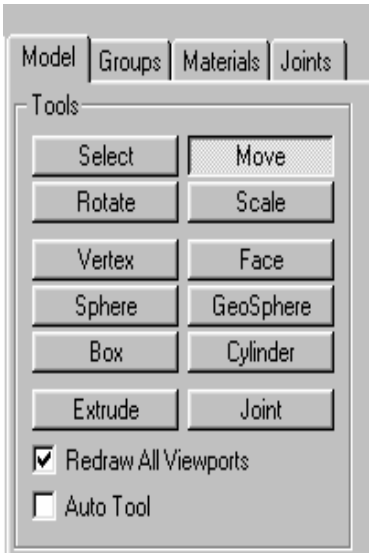


Figure 5.1: MilkShape 3D selection and manipulation toolbox. All of the options are mutually exclusive. The mouse input is rigidly linked to the current selected option, providing unnecessary indirection.

mouse-linked and if a manipulation mode is selected, the selection mode is eliminated and has to be set again if another selection is necessary. The visualisation only includes one type of sensors at a time, and this means extra mouse clicks or keyboard shortcuts in comparison with an approach that provides at the same time the selection sensors and manipulation widgets for the basic entities.

Shape complexity is manipulated manually by adding vertices or removing vertices and building the polygons for the desired complexity, with the advantage that the complexity is controlled by the user, but with the disadvantage of a manually performed low-level approach. The only direct complexity manipulation option is the extrude operation.

Shape positioning is either performed relative to the local coordinate frame by selecting the whole character and rotating it, or by using the customisable 3+1 views available in the system. The latter possibility is the most generally used method, since in case of characters, they are assumed to be in a neutral position by systems that incorporate them (mainly 3D games in this case).

Hierarchy. The system provides the tools necessary for editing bone-like hierarchies. These hierarchies are hard-wired, behaving like a static hierarchy: if a joint is displaced, all joints below in the hierarchy will move the same amount. This makes the hierarchy editing process global. The geometry is assigned to joints by selecting the target vertices and effectuating the assignment operations. However, this process is described as being a lengthy and error-prone process, just like in the case of GMax.

The system presents a number of enhancements for ease of use, one of them being the possibility to use background pictures as reference for character outline, bending or other properties. Vertices can be hidden or made visible to enhance the selection properties, and there are also selection shortcuts besides

groupings via a joint-assigned vertices selection mechanism.

5.1.2 CosmoWorlds

General description. CosmoWorlds is an editing package based on VRML as its main file format, capable of handling complete scenes or scene components (such as characters). Besides the possibility to create (add) the normal VRML elements into the edited scene, it can designate graphics objects as Points, Edges, Polygons (PEP) objects. This is an approach to convert predefined objects into IndexedFaceSet objects for the purpose of manipulation. After entering in PEP mode, it provides a direct manipulation approach, with regard to selection and manipulation of graphics components, but not without shortcomings. The common 3+1 views is used by default, although customisable by the user.

Primitives. The standard vertex, edge and face components are extended in this case with 4-vertex faces (a separate object of type IndexedFaceSet) and non-persistent selection groups. The latter ones are achievable through drag selection with a selection pointer, which makes them imprecise, since all the vertices, edges and faces that the pointer touches are selected. It is a sort of bulk selection, and although it is not persistent, successive manipulation steps can use the same or modified selections.

Selection. The selection possibilities are direct and intuitive in this case. Since the system handles both scene and individual objects, there is a selection switch required for the graphics editing mode PEP. However, in the latter case, the selection options (vertices, edges and individual polygons) are visible only if the pointer is travelling above the object (and completely visible only in wireframe mode). For complex shapes, this could mean longer searches for target entities that need to be selected.

Apart from the fact that it is only two-dimensional (facing the user), the vertex selection sensor is also divided between the polygons that the vertex participates in, resulting in a pie-slice type of partitioning of the sensor surface. This means that for marginal vertices that participate only in one polygon, the selection area is diminished. This is especially cumbersome in cases when the polygon is positioned nearly perpendicular on the screen, when the polygon is not visible and thus the vertex sensor is also non-selectable. Multiple selections are handled via a rectangle selection tool or drag selection, both inappropriate for precise, distinct group selections.

The selection status of the primitives is colour coded for a more valuable visual feedback. The colours change depending on the cursor position: when the selection tool is over the feature, when a selection has been made and displacement is possible or if the pointer wanders off, the selection becomes passive but still available to button-based operations (mainly aimed at mesh complexity changes).

Manipulation. The simplest manipulation modality, directly integrated into the selection tool is a displacement modality, when the selected entities (vertex, edge or polygon) can be positioned by dragging, restricted to the plane the selected entities participate (this is why the selection tool is segmented between the different polygons). To overcome the limitations from this displacement method (e.g. this way the vertices of a vertical wall cannot be displaced perpendicular to the wall), a general displacement, scale and rotation widget is used (both in case of whole objects for scene arrangement and in case of prim-

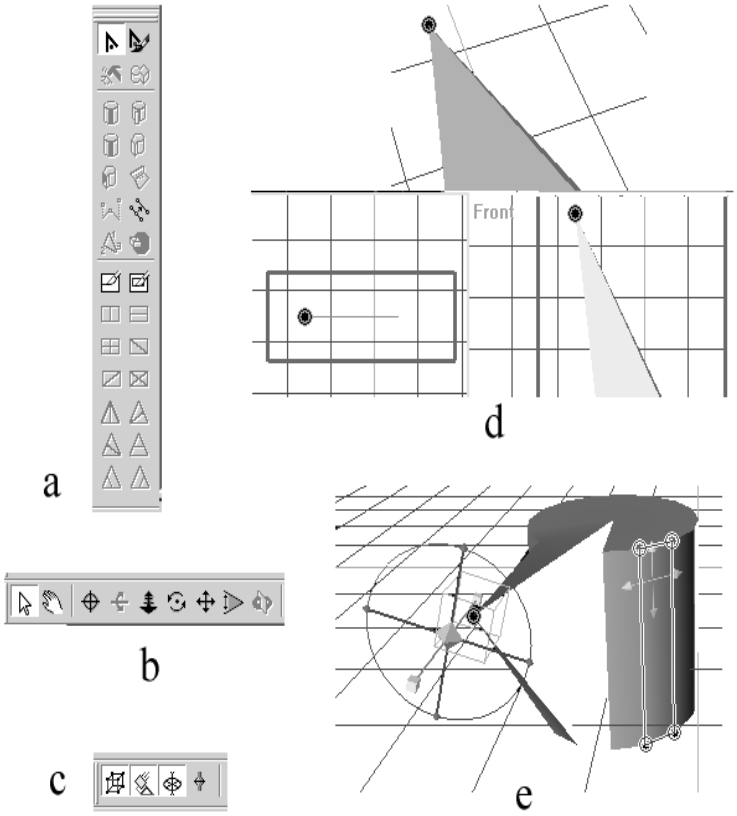


Figure 5.2: CosmoWorlds properties. a) PEP toolbox, with many vertex, edge and polygon handling operations, b) View handling toolbar, c) Manipulation widget on/off switches (for translation, plane picking, rotation, alignment widgets), d) marginal vertex with normal (top), smaller (lower right) polygon area for XY plane translation and without (lower left) polygon area for a z-axis translation, e) Circle represented rotation widget and box represented translation widget, in this case for a single vertex.

itives). This widget uses three plane displacement drag sensors, three double axis (only one is usable at a time) rotation drag sensors (except in case of single vertices), and sensors for one-dimensional scale at the corners of the selection's bounding box, a box that also represents the local coordinate system in which the scale operation is effectuated.

The change between different editing modalities happens with the use of interface buttons. There are two cases depending on the editing context and user preferences:

- switching between geometry manipulation and view handling (which is a mode where each different view handling operation is indirected: rotation, sliding, panning, zooming, etc.),
- switching between object manipulation and primitive manipulation (PEP), with an added step of indirection in the case of drag selections.

The possible mesh refinement operations include the classical extrude, concatenate, etc. operations, with a wide range of subdivision options in case of 3-sided and 4-sided polygons (middle, diagonal, single, double slicing, even in different orientations). However, this latter operation is not as precise as it should be due to multiple selection bulkiness.

Hierarchy. The scene graph or complex object hierarchies can be handled via menu options or via an interactive hierarchy graph (combined with a property editor), similar to IDE (Integrated Developmet Environment) approaches. Groupings, parent addition and sibling settings can be adjusted, but without a detailed control of the hierarchy. Each Transform node containing an object cannot be extended with other objects (after loading such a file the system converts it automatically to one object per Transform node).

The texture handling capability of the system uses a direct vertex-texture coordinate matching. The approach provides a detailed texture control, with the exception of texture edges, where great care must be exercised to use the whole texture without going over the last texture pixel, in which case the system repeats automatically the texture, what is not always necessary. A snap-edge would be more than welcome in this case. The display of all sides of a basic object on the texture map at once is also a rather confusing and unintuitive effect, especially if the visualised texture coordinates occlude each other.

5.1.3 Maya PLE

General description. The Personal Learning Edition of Maya is a free version of the commercially available package. It is a complex tool, used by professionals in the graphics and picture industry, providing many possibilities in 3D scene creation, rendering, animation, etc. The traditional 3+1 views and combinations are used, with the possibility to create persistent custom views for easy view manipulation and previews.

Primitives. At the scene level, the system works with a multitude of object types. NURBS objects, polygon objects and volume objects are the basic ones. Curve creation tools and different curve-based operations enables the creation of extruded, revolved or lofted surfaces, where one, two or more curves specify the surface properties. At object level, the basic elements are the classic vertices,

edges and faces in case of polygon objects and different types of control points in case of parametric curves and surfaces. The additional joint primitive aids in creating bone hierarchies and animations.

Selection. Selection is hierarchical and type-oriented. First of all, to access the properties of an object, the object must be selected with a left mouse click, eventually with name selection. The right mouse click gives an aura-style menu for the component selection options, where depending on the object type, vertices, edges, faces, control vertices, surface points, etc. can be set up as selection types. It is possible to "prepare" for selection multiple objects, but only one type of selection can be specified at a time, thus the selected objects must possess compatible selection properties.

Grouping is an important aspect of selection in this system. Many variations are achievable (regarding to objects, object components, memorising selections - called sets), which are further extended by the parenting options and operations. Through these operations, although menu-oriented, the system provides a scene hierarchy oriented approach to modelling with an ease and richfullness that is missing in other systems.

Due to the complexity of the system and the fast-paced analysis we effectuated (similar to the behaviour of a novice user), there may be other selection modalities that we left unexplored. We concentrated mainly on the direct interaction, without exploring the depths of the menu system.

Manipulation. The manipulator tools, similar to the selection modalities, have to be specified separately via option buttons. The basic move, rotate and scale operations are available, in form of widgets activated for the selected vertex, edge, polygon, control vertex, object, etc., allowing an unified approach for all selection types. The widgets work separately on the 3D axes respectively, on all the axes at once in 3D view or pairs of axes in the front, top or side views. It is a plus that the different basic manipulation tools are combined with the selection mechanism.

When it comes to editing meshes, indifferently of the type of mesh used, the editing options provide a large scale of operations, from low-level vertex editing to high level parametric surface modelling. In the case of polygon meshes, the low level operations control the mesh complexity, with operations like beveling, extruding, merging, splitting, collapsing or subdividing. The control vertices of parametric meshes behave the same as polygon vertices when it comes to the move, rotate and scale operations and they have special editing options with regard to shape complexity, options inherent from the parameters they represent.

Again, we did not explore all the menu options available for the editing the scene and especially for editing the scene objects. Although the basic operations in case of some graphics modelling knowledge are relatively easy to learn, efficiency in such a complex system requires more learning and practice.

Hierarchy. The system incorporates the tools for hierarchical object creations, and particularly for character creation. Grouping and parenting operations enable the user to create character hierarchies while the joint primitives (objects) make simple and complex animations possible. The system animation tools include deformers (affecting skin shape), skeletons (joint hierarchies), skinning (joint-skin linking) and constraints (for extra objects on the character). All these tools take time to be accustomed to, but they provide great flexibility and possibilities for advanced users.

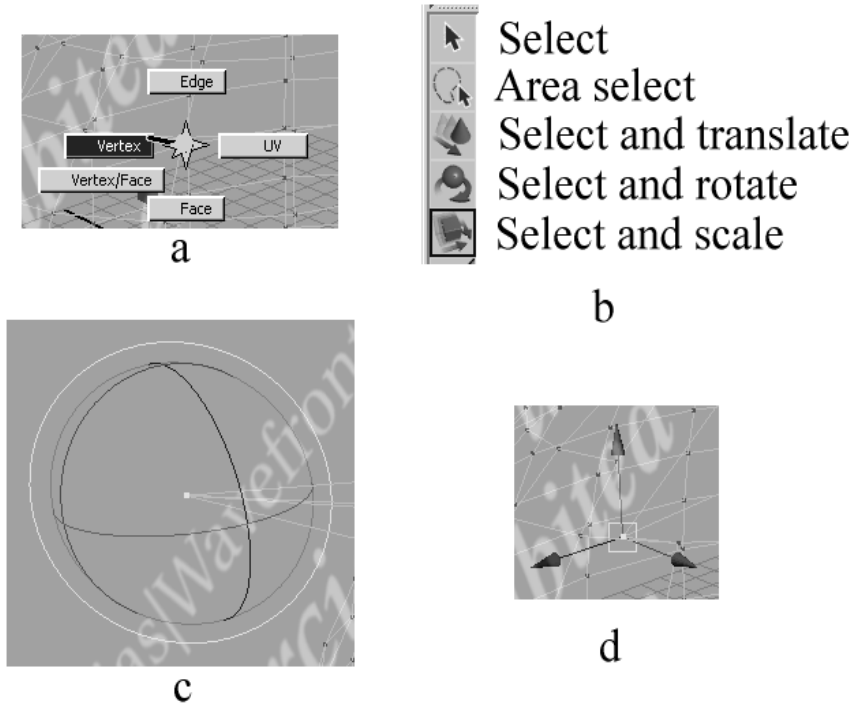


Figure 5.3: Maya PLE properties: a) Right-click menu for selection type specification, b) Selection toolbar provides options for separate selection operations or combined selection and manipulation operations, c) 3-axis rotation widget, d) 3-axis translation widget.

A particularity of this system is the possibility to use the embedded script-like language (MEL). This can be used to create custom settings, geometries or interfaces. It is used also as a general plugin creation modality. With enough expertise it could be used to include the modelling properties of our HanimPlus system into the Maya interface as a plug-in or as extra modelling options.

Precision handling is qualitatively enhanced by using visual feedback in the form of grayed-out former positions of the manipulated entities. This way, the user can see both the initial position and the new position of the same entity, which provide location pointers in space, resulting in increased spatial awareness.

5.1.4 GMax

General description. This graphics tool is a slightly trimmed version of 3D Studio Max, aimed at game character modelling, including meshes, bones, etc. However, it also contains higher-level components like parametric surfaces, space deformation tools and many others, usable for a multitude of tasks. The system also inherits a rendering option next to animation, lights and camera position handling, showing that (at least for the big brother Studio Max) the area of

application is not only game characters, but also scene editing and rendering for different, in many cases professional uses. The visualisation is by default done in 3+1 views, although customisable by the user.

Primitives. A number of basic and complex shapes form the "primitives" for the 3D scene being edited. For single objects, the handled primitives are vertex, edge, triangle and polygon. However, the basic objects turn to polygonal constructions only after explicitly setting so. Even if a 4-sided polygon is not planar, it is treated as a single polygon, and an explicit property setting is necessary to turn it into triangles usable in lower-level geometric formats or in graphics engines like VRML or OpenGL.

Selection. At the scene level, objects can be selected intuitively, by simple point-and-activate actions. The switch to geometry editing is done via menus or via a list of hierarchical (successive) modifier operations, after the target object is selected. The user has to set up selection "modifiers" to be able to access the primitives of an object, and within this mode, it also has to specify the selection type. This method of specifying the selection ability, selection type and then the hierarchical modifiers is not very intuitive, requiring in-depth knowledge of the modifier relations.

Manipulation. The manipulation tools work both at scene level and at geometry level as selection pointers, allowing instantaneous access to the manipulation possibilities after a selection has already been made. The tool is represented as similar-looking widgets with three colour coded axes in form of arrows that switch their colour if activated (pointer is close to arrow head). In function of the selected manipulation modality, the arrows provide a direct, drag input modality for moving objects along the axes, scaling objects along axes or rotating objects along the axes. Two axes of manipulation are selectable at once by using small sensors on the surface of the planes which are defined by pairs of axes.

Hierarchy. In this case, the hierarchy is created separately from the mesh, usually close to the mesh to follow the mesh directions at least. Afterwards the bones have to be positioned according to the mesh; this is a little bit eased with the possibility to select successive bones with a keyboard shortcut (PGDN), but this requires system knowledge. The system also adds an automatic short bone as end of the chain, to visualise hierarchy endpoints (leaves), and uses a dummy pelvis object instead of a root joint. The bone hierarchy is rigid, operations effectuated on a bone are perpetuated to the child bones, resulting in global changes. It also provides the possibility to turn off global changes when positioning the pivot of a bone; however this results in a confusing situation where a move action is in fact a bone lengthening action (affecting parent bone). The different (not continuously created) bone chains have to be linked manually together (eventually through a dummy object to preserve the individual hierarchies).

The system provides helper actions such as bone dimension setting to resemble a mesh more accurately or creating fins that pertrude the mesh for the easier selection of bones. This seems not to be enough in all possible cases, since during the creation and positioning process, the user is advised to manipulate partial/ total hiding, transparency, wireframe settings for better visibility and selection reasons.

After all the bones are created, they must also be linked to the geometry. For this purpose, a "skin modifier" widget is used. This tool specifies areas of

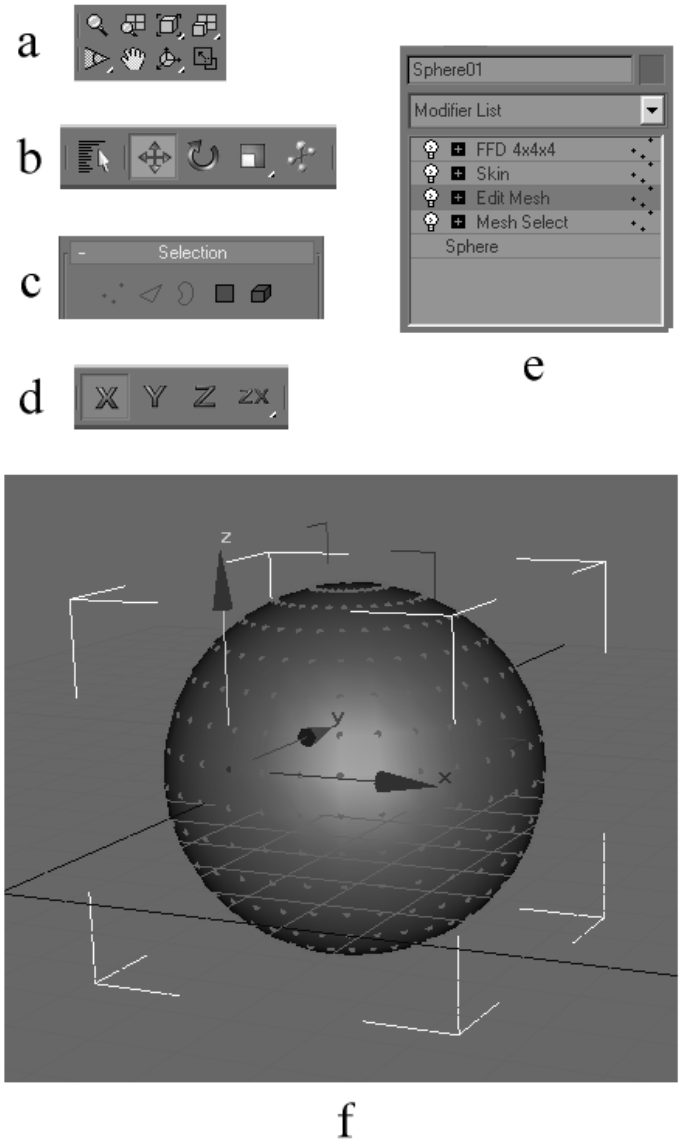


Figure 5.4: Some of the GMax system characteristics are: a) View manipulation toolbox, b) Selection and manipulation toolbar (selection by name, selection and displacement, selection and rotation, selection and scale), c) Selection type specification options (vertices, polygons, area, object selection types), d) Manipulation plane specification (also selectable via widgets), e) Modifier list needed to model an object (selection, editing modifiers), f) An object with a selected vertex and active widget, where the widget itself does not suggest the active editing mode (it is the same for all widget types, representing the main axes and planes. By looking at it, does not suggest that the actual selected editing mode is rotation).

action for bones, meaning that the coordinates that fall within the specified area are deformed when the bone is rotated. This operation can be time-consuming and renegade vertices could also represent a problem solvable only by adding them individually to the skin modifier. Comparing to this case, the H-anim specification provides a hierarchy specification usable to avoid these problems.

This particular system has a wide range of features, but has a steep learning curve for occasional or novice users due to the possibility to combine features in many ways. The indirection in this case comes from the many options between which the user has to switch to get to the desired function. Instead of a simpler, overviewable graphics architecture, the user has to assimilate complex graphics knowledge and an option-packed interface. For instance, one has to know the behaviour of different tools such as right click, enter, etc. necessary to end certain editing modes (selecting other tools may not be enough), otherwise unexpected behaviour might happen.

5.1.5 Blender

General description. Blender is a complex system, aimed mainly at rendering (movie, television, etc. production), and not at interactive publishing. However, it is based on 3D scene and objects, and it is also capable of importing and exporting scene files in different format.

Primitives. The system provides the possibility to handle vertices and complex objects. Edges, planes, and the related graphics operations are left out from the system. The object primitives consist of common scene graph meshes (cone, cylinder, plane to name a few), on curves (Bézier and NURBS curve types) and on surfaces (different types of NURBS surfaces). Emphasis is on the customisation of both the underlying vertices, control points, and on the different appearance tools, which are the most valuable for rendering purposes.

Selection. The selection mechanism is unconventional. Where other tools use the mouse left click for selections and right click for quick menu access, here a left click positions the graphics cursor (which may be used as a reference coordinate for different operations) whereas the right click provides the selection possibility. In vertex edit mode, single, multiple or area vertices can be selected, while in object edit mode, single or multiple objects can be selected (depending whether the objects were created in the same object edit session when they are grouped together).

Manipulation. The application enables mouse gestures to specify the basic manipulation type. A line gesture means translation, a round gesture means rotation and a zigzag gesture means a scale operation, in all cases using the graphics pointer or the local centre as the centre coordinate. After a gesture was registered (mouse drag), the succeeding mouse movement (which starts at the point when the mouse button was released) specifies the extent to which the selected operation is effectuated. The gesture-based manipulation is available to objects, multiple objects as well as vertex selections (single, multiple or area selections).

Creating the objects is managed through an object creation menu, categorised by object types. Mesh refinement is controllable via subdivision operations on the edges belonging to selected vertices. A vertex deletion means also the deletion of all edges and faces the edge is participating in (although there

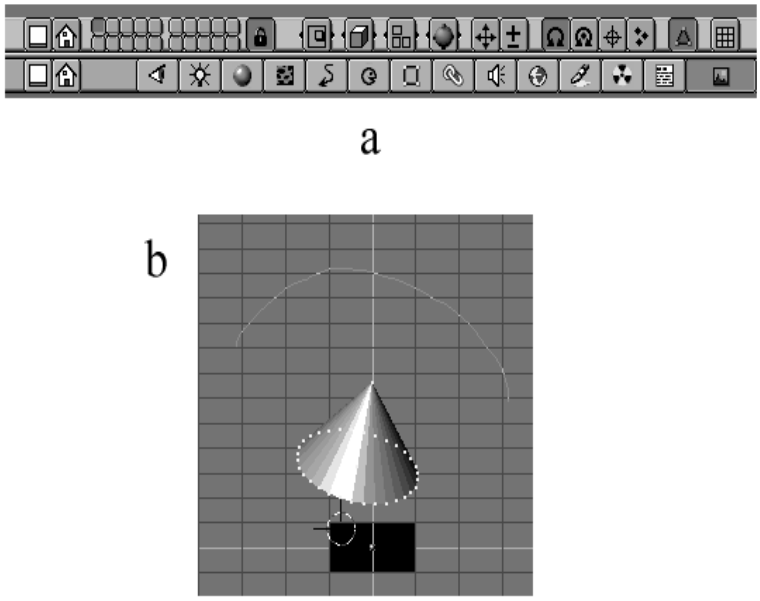


Figure 5.5: The Blender interface. a) Each of the buttons in this toolbar handles visualisation options (top row) or opens up toolboxes for the different scene settings, object properties, etc. b) Arc-gesture requiring a rotation of the selected entities.

are some light variations): thus the method is limitedly usable for direct mesh simplification. Noteworthy is the absence of the undo operation, present in most commercial packages as a means to quickly restore the previous state(s) if the result is unwanted.

Hierarchy. There is no explicit way to create hierarchies. The desired partial objects (body segments) can be created separately and then different constraints can be assigned to create hierarchical behaviour. The constraint (and other) linking and assignment operations are created by typing in the associated object names, contrary to the visual or list object approaches usually available in other systems.

An interesting aspect of the system is the adaptation of the layers technique used generally in imaging applications to a 3D modelling application. The rationale behind this is to provide a methodology to easily create different effects. The method strengthens the image production, rendering aspects of the system. Besides the object and vertex edit mode, different edit modalities like views, lights, textures, etc. can be used to create renderable or 3D scene content.

The system is complex and heavily shortcut-key oriented, suggesting a steep learning curve. Some of the shortcut keys can be learned only consulting the manual or tutorials. The views are separated and quickly accessible via key shortcuts. They do make the break from the usual 3+1 view by using only one view at a time, but still provide the separated 2D views for positioning the

edited objects for selections and operations.

Object names (and other property names like materials, textures, etc.) are used in the system to create different links like constraints, property assignments, etc. This requires that the user has at least a little bit of overview over the names he utilises in the modelling process, just like programmers in the code creation process.

5.1.6 Our HanimPlus hierarchy editor

General description. This system concentrates on the usage of H-anim or H-anim like hierarchy templates to quickly modify the complexity and position of the Joint structure and the connecting geometry surface. Complex templates allow to concentrate mainly on the alteration of the different features, while simple templates allow custom hierarchies to be built.

Primitives. Vertices are the only basic primitives used. Since the geometry is constrained to be regular, the edge and polygon construction/destruction is automatically handled by the system. The regular geometry criteria also allow us to treat vertex ring and column collections as distinct entities, resulting in a new approach for primitives: collections of vertices arranged in a ring or in a column fashion. The primitive status of these collections is justified by the selection modality and the editing operations aimed at these collections.

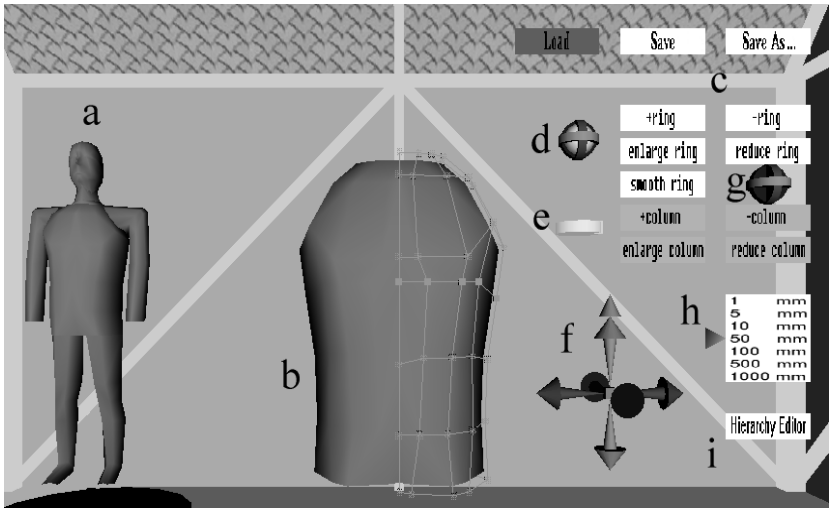


Figure 5.6: The interface components: a) Whole character visualisation, with the different Segment geometries selectable; b) Selected Segment geometry, internally symmetric, with half control structure and a selected row primitive; c) A number of active button-like sensors; d) 3-axis rotation widget, controls the geometry orientation (view handling); e) Scale widget (the geometry was already scaled); f) Displacement widget (not drag-widget), works in combination with unit widget; g) Ring primitive rotation widget, also separated to different axes; h) Unit widget for setting the extent of the displacement operations; i) Switch into hierarchy mode. In the system, all selection sensors are active at once, making selections a direct operation. Widgets are also available at all times, shortening manipulation selection time.

Selection. All of the primitives have selection sensors assigned to them. Vertices are selectable with sensor boxes, collections of vertices by ring or column "line" sensors, which go through all the participating vertex coordinates. Joint positions are treated in the same way as vertex coordinates, with a sensor box as the selection actuator. The selection presents colour and transparency change based visual feedback for the mouse-over and activation states.

Manipulation. The two editing modalities (hierarchy and geometry editing) are switchable with a single button push. After a selection is made, unavailable operations for the selection type are grayed out. Operations are either button-based for single, logical, parameterless operations (hierarchy editing operations, smoothing operations, etc.) or are based on widgets, providing direct drag or click manipulation options (displacement, rotation).

The character complexity is customisable at Joint level and at geometry level. Joint insertion or adding operations have been defined for hierarchy refinement and extension. The geometry complexity is alterable, with the restriction that the geometry must retain its regularity properties: this requires the complexity operations to be ring or column oriented. Single vertices cannot, but ring or column collections of vertices can be inserted or removed from a geometry to refine or simplify the edited character's skin. The automatic symmetry handling is achieved by allowing editing on one side of the character, and replicating the operations on the symmetrical half.

The positioning of the character and the edited structures differ substantially from the positioning solutions of the previously presented systems. The fact that the editing tools are integrated into the environment as well as that a single structure is edited rather than the whole environment, gives the possibility to provide tools for easy view manipulation. The visualised hierarchy structure can be drag-rotated along the vertical axis, while the visualised geometry segment can be drag-rotated around all axes and scaled to a comfortable size level.

Hierarchy. The system is designed specifically to provide the hierarchy editing possibility, for the creation of characters, avatars that can be animated according to the H-anim modalities, but without the constraints of that specification. The advantage of our approach is the automatic linking of geometry segments to the different Joints (which translates Joint operations also to the connecting geometries) and the soft-linking of Joints, enabling thus local hierarchy adjustment operations.

Symmetry is elevated in our system to an important feature, which is handled automatically. Self-symmetric geometries, pairs of symmetric geometries and Joints are differentiated and their behaviours are reflected or transmitted in case of hierarchy modelling. The automatic symmetry handling option is turned on or off by loading two different hierarchy types. More detailed, mixed symmetry handling (symmetrical and asymmetrical entities in the same hierarchy at the same time) are technically feasible (by manually editing the input hierarchies), but no such options are integrated yet in the user interface.

5.2 Cumulative comparison

Generally speaking, graphics editing packages use the classical graphics primitives vertices, edges and polygons together with joint (character bone) primitives

for hierarchy constructions and animations. They may also present high-level graphics primitives like parametric surfaces, objects or curves, depending on the task area these modellers aim to cover. Some of them also present geometric variations regarding the primitives used or they provide a greater manipulation flexibility by allowing persistent selections.

Our approach provides a unified approach to primitives constructed from sets of vertices. The regular geometry constraint is used to provide a persistent approach to multiple vertices as automatic logical primitives, both in the selection and manipulation possibilities. We concentrate on speeding up the editing process by using natural, logical primitives and operations, and scaling them to a higher manipulation level.

The regular grid option that allow us to provide column and row primitives, does however exclude non-regular geometries from the pool of geometries that our system can handle. This is a drawback that can be characterised as favouring naturalness, easiness, quickness over the ability to handle all types of geometries.

If the modelling packages provide a hierarchy modelling possibility, often this possibility is limited to a separate joint/bone editing process with later connection to the geometry, usually a process called "skinning". Our system, by concentrating on hierarchies, provides an approach that is hierarchy-controlled right from the beginning.

Regarding the widget construction, there is some room for improvement towards greater direct manipulation integration. The displacement widget could be transformed into a drag widget and the unit widget could be eliminated. This would be a case of direct manipulation versus greater precision (and more secure position handling) with the balance slightly in favour of the direct manipulation which even results in speed gain, more overall naturalness.

Tables 5.1 and 5.2 provide in a short format the principal properties of the reviewed systems, allowing a quick comparison between the different systems.

5.3 Summary

Ease of use is an important aspect of the system we built. Through the simple, but powerful interface and simple basic concepts that are also intuitive the system is easy to comprehend and to use for unexperienced users. All the basic components of the system are visualised and it can be easily learned by simply trying them out. The relations between selection and operation types, whether button-based or widget-based become rapidly obvious. Text-based button descriptions and easy to use widgets with either cyclic behaviour (in case of drag sensors) or limited range but successively applicable operations help in the precise manipulation and positioning of the edited character or its components.

The system efficiency lies typically between parameter based pre-built component modification systems like [Blaxxun interactive, NAa] and general modelling systems, but closer to the general modelling systems. Regarding the geometry models that the system is able to produce, these are comparable to the geometries produced by general modelling tools. At this point, the efficiency gain can be regarded as a pure advantage.

At the data storing level, our system does not encourage interoperability. The internal format used ensures the regular grid criteria for our system. However, the exporting options output VRML, X3D or Java3D format files, which

Table 5.1: Characteristics of the different editor systems

Systems Properties	MilkShape 3D	CosmoWorlds	Maya PLE
Vertex primitive	Yes	Yes	Yes
Edge primitive	No	Yes	Yes
Polygon primitive	Yes	Yes	Yes
Ring/column primitives	No	No	No
Groupings	Yes, persistent	Yes, not persistent	Yes, persistent
Selection type has to be specified	Yes	No	Yes
Manipulation type has to be specified	Yes	Yes	Only at type changes
Positioning mechanism	Views	Views, persistent object rotation	Views, persistent object rotation
Indirection (from the above three properties)	Very high	Low-medium (scene handling)	Medium
Manipulation diversity	Low level	Low, medium level	Low, medium, high level
Manipulation modality	Direct (drag)	Direct (drag)	Direct (drag)
Manipulation quality (precision)	Limited to the different views	Capable drag widgets	Capable drag widgets
Hierarchy properties	Hard-linked, manual bone-skin assignment	General scene graph modalities	Hard-linked, manual bone-skin assignment
Complexity handling	At low level	At low level, multi-selection level	At low, multi-selection and high level
Dedicated joint or bone objects	Yes	No (general object hierarchy)	Yes

Table 5.2: Characteristics of the different editor systems *(continued)*

Systems Properties	GMax	Blender	HanimPlus
Vertex primitive	Yes	Yes	Yes
Edge primitive	Yes	No	No
Polygon primitive	Yes	No	No
Ring/column primitives	No	No	Yes
Groupings	Yes, persistent	Yes, not persistent	Implied
Selection type has to be specified	Yes	Single option	No
Manipulation type has to be specified	Only at type changes	Yes	No
Positioning mechanism	Views, persistent object rotation	Mutually exclusive views, object rotation	Dedicated rotation widgets, geometrically non-intrusive
Indirection (from the above three properties)	Medium-high (selection methods)	Medium-high (views, gestures)	Low-medium (click-displacements)
Manipulation diversity	Low, medium, high level	Low, high level	Low, medium level
Manipulation modality	Direct (drag)	Direct (drag)	Direct (mixed, drag and click)
Manipulation quality (precision)	Capable drag widgets	Limited to the different views	Axis-limited precise operations
Hierarchy properties	Hard-linked, manual bone-skin assignment	Hard-linked, manual constraints setup	Soft-linked, automatic skin assignment
Complexity handling	At low, multi-selection and high level	At low, multi-selection and high level	At multi-vertex level (constraints)
Dedicated joint or bone objects	Yes	No (only constraints)	Yes

can be used by more advanced modellers if needed.

An important advantage of our system is the integrated interface. This allows for a better view handling (positioning, size control) since the edited geometry is a self-contained embedded object. It also provides a higher immersiveness degree, especially if the full-screen option is used within certain web browsers.

MilkShape 3D may be the system closest to our HanimPlus character modeller. Its target is to modify/construct low polygon models for interactive games, where character complexity is a frame-rate influencing factor. It does just that, but with a highly indirected interface, where frequent selection type and manipulation type options have to be set. Our system, while targeting itself to interactive, web-based VR systems, has the same goal of low-level editing and, in addition, to make the editing more natural and to speed up the whole process. Integrating the interface into the VE and providing natural graphics elements and operations provides an efficient, directly manipulated system.

Chapter 6

Conclusions and future work

In the previous chapters, we presented our goals, a literature review, graphics and virtual environment technologies, a detailed description of our system and a system analysis based on properties and behaviour. This chapter presents the main arguments we elaborated during the course of our work, the results we achieved and the conclusions we have drawn, as well as a description of the future direction we would like to take in our research efforts concerning modelling in virtual environments.

6.1 Conclusions

The result of our activity is a 3D character editing system described in Chapter 4, aimed at creating lightweight characters for virtual environments. We handle both the geometry and hierarchy of these characters at a level comparable with commonplace modelling systems (see the reviews of Chapter 5). The system is aimed at novice or inexperienced users occasionally creating characters, observable from the use of natural interaction paradigms and simplistic interface. However, this does not exclude more experienced users, when the target goal is to create quickly virtual characters with detailed control over the hierarchy and geometry structure.

Our system is capable of producing a wide range of character hierarchies. Although we use the H-anim components as building blocks, we do not limit ourselves to the character range constraints imposed by the H-anim specification. For the character geometry itself, we use a limitation to regular coordinate grids, which have interesting properties that we can exploit. However, this limitation does not affect too much the range of geometries we can handle, and the gain from effectiveness, natural approach, etc. weights more than the limitations.

We make use of network-based and virtual technology platforms for our virtual modelling environments. VRML and Java give the possibility to visualise and handle the system components dynamically, whether they are the modelled entities or the manipulation tools. A big advantage of our system is being web-enabled, representing a new breed of on-line, request-based applications that users can access whenever needed.

The novelty aspects of the system can be observed in many characteristics. Besides being an on-line modelling system, the technology we use enable us to push the limits of VR by integrating visualisation, interaction, data management options into a virtual environment, modelling options that have been previously considered as inappropriate for VR due to resource constraints or the rather static nature of the VR content systems such as VRML.

Another novel aspect of the system is the integration of the modelling controls into the virtual environment. This is a possibility to use the space provided by the virtual environment not only for the edited entities themselves, but for all other tools, since the environment itself is not linked by any means with the character being edited. Together with 3D modelling tools (widget) recommendations observable in graphics research papers, this integration made possible the effective, direct manipulation enabled interface we have now. All of the geometry, hierarchy handling operations, positioning tools and data management options are handled similarly, providing consistent interface.

By using regular grids of vertices for our graphics objects, we could define alternative graphics primitives around which we built a natural editing environment. By using the row and column division of the graphics grid as graphics primitives (sets of vertices arranged in row or column form), we provide a higher-level modelling possibility to the low-level vertices. This materialises by applying the low-level modelling options to multiple target vertices at once or by using natural editing options defined for collections of vertices (rotation, smoothing, etc.).

The logic integrated into the system in form of self-symmetry, symmetric object pairs, hierarchy construction properties, etc. handles automatically many tedious tasks that a user otherwise would have to perform. This leads to increased effectiveness while the user can concentrate on other, more important tasks.

An important achievement of the system is its usability properties. The simplistic interface, natural editing options, visual interaction possibilities, directly selectable entities, direct manipulation are all properties that led to an easy to use system, which shows manipulation properties that are generally above the properties of many editing systems.

The limitation occuring due to the regular grid requirement is diminished by the options we gain using this criteria. Another limitation of our system is the usage of click-sensors in widgets (although we only use it to a limited extent) instead of a more user-friendly drag-sensor. Our reason to this is the need for precision, since some drag-sensors have a limitless range, and increased carefulness is required to handle these. Instead, we used the safe click-sensor method to limit the widget range, but we provided options for different unit magnitudes for greater effectiveness.

6.2 Future work

One thing can be said for sure about graphics modelling systems: they are always far from completed since they always can be extended with new possibilities, features or techniques. This is a natural process, as new interaction modalities, input devices or general techniques become available.

Our system currently is in the phase of lacking some of the general capabili-

ties of general modelling tools to be really useful. For a more complete system, aimed at creating the most obvious components and functions of characters, avatars and embodied agents, we also need on top of the current system:

- The separate modules or possibilities like the NURBS format handling or texture editor integrated into the system. This gives the possibility of creating smooth, detailed models and to set the texture and appearance of models, providing skin information.
- An animation editor component, which is currently under development, also to be integrated into the system with the purpose of motion capture or manual keyframe data mapping, editing, combining. This would give a tool that provides the possibility to interactively, behaviourally direct embodied agents, autonomous avatars, etc. The presented information visualisation techniques may be considered as an interesting modality to integrate them as animation editing, targeting and general setup tools. File handling that uses cone tree-like visualisation, animation sequence panels managing options, the visualisation of Joint rotation values are the visualisations tools that we are concentrating on. Further tools for analysing animation sequences, extracting basic animations and creating new combinations of animations with different parameterised qualitative variations, all this from rich animation data sources like video footage present exciting research opportunities we would like to pursue.

Regarding the graphics capabilities and the direct manipulation principles, we would like to convert all remaining indirected components into direct manipulation options by using only drag sensors for parameter based editing operations. There can be several construction options for direct manipulation widgets, from which we currently explored the construction where the widgets are placed separately from the geometry, to increase edited object visibility and to be always accessible for manipulating the selected entities. In this case, a displacement drag-widget which changes its position during manipulation does not fit well with the concept of widgets positioned separately from the geometry. A different feedback method can be used, for instance a drag manipulation could be illustrated by texture displacements on the widgets.

With the advent of new elements in the VRML/X3D language, our system could be extended to take advantage of new interaction modalities, of new standard geometry types (NURBS and Lattice geometries, H-anim 2001 hierarchies) and of new levels of interconnection possibilities that make the handling of dynamic content easier.

As a more ambitious goal, we would also like to provide manipulation tools at the whole VRML/X3D language level. If we define manipulation widgets for all data types of the language, it could be possible to build systems that read in complete virtual environments and make them visually, naturally adjustable. It would be an interesting research topic to investigate the possibilities of an integrated editing modality for non-visible components of an environment, for the event flow and for the scripting possibilities offered by these environments.

Appendix A

List of acronyms

API	Application Programming Interface
AWT	Abstract Window Toolkit
CAD	Computer Aided Design
CAVE	Cave Automatic Virtual Environment
CVE	Collaborative Virtual Environment
DOF	Degrees Of Freedom
DVE	Distributed Virtual Environment
EAI	External Authoring Interface
GPS	Global Positioning System
GUI	Graphical User Interface
H-anim	Human Animation Specification
HCI	Human Computer Interaction
HMD	Head Mounted Display
HUD	Heads Up Display
IDE	Integrated Developmet Environment
IFS	IndexedFaceSet
IK	Inverse Kinematics
ILS	IndexedLineSet
IV	Information Visualisation
IVE	Intelligent Virtual Environment
JVM	Java Virtual Machine
LOA	Level Of Articulation
LOD	Level Of Detail
MU	Multi User
MUD	Multi User Dungeon
NURBS	Non Uniform Rational B-splines
PDA	Personal Digital Assistant
UI	User Interface
VDT	Visual Display Terminal
VE	Virtual Environment
VMC	Virtual Music Centrum
VR	Virtual Reality
VRML	Virtual Reality Modelling Language
VRUI	Virtual Reality User Interface
WIMP	Windows Icons Menus and Pointers
WYSIWYG	What You See Is What You Get

Appendix B

VRML nodes used in the HanimPlus system

The extent to which the different nodes available in VRML are used in our systems is determined by our needs for visualisation and interaction. The data types we visualise is the set of Segment geometry coordinates, the geometries they define, Joint hierarchy coordinates and Joint relations. The following is the list of components used for visualisation:

- for storing the geometry vertices data, we used a Coordinate node per Segment component,
- for the position of individual vertices, we used Box geometries, positioned according to the individual geometry vertices data,
- for the visualisation of vertex grouping relations (rings or columns of vertices) we used IndexedLineSet geometries using a subset of the geometry vertices data,
- for the visualisation of the geometries themselves, we used IndexedFaceSet nodes using all of the vertex data from a Coordinate node,
- for the storing of Joint hierarchy position data, we used properties of the Joint component,
- for visualising the Joint position, we used box geometries,
- for visualising the Joint relations (hierarchical connections), we used IndexedLineSet geometries and the positional data of the related Joints.

With the exception of Joint hierarchy connections and the symmetric components that are automatically updated, all above mentioned components have sensors attached, sensors used for selection purposes. While the TouchSensor is enough to provide the selection functionality, the different widgets and controls require different sensors to provide quantitative information about the extent of the intended operation. In this case CylinderSensors, PlaneSensors or SphereSensors may be used instead to provide one dimensional rotation, two dimensional displacement, respectively, three dimensional rotation information.

As you can see, the depth of the system is not coming from the limited number of VRML components used, but from the manner how these are used. Handling the data, dynamically creating and managing the visualisation, observing the interaction and executing the user requests is a complex process, which widens the possibilities of the built-in components of VRML to a considerable extent. The following subsections present the used geometry and sensor nodes respectively their components and default properties.

B.1 Geometry nodes

The **Box** node is a light, simple node that has a single property, representing the size values of the box along the three main axes. The geometry positioning is achievable through wrapping Transform node. The size component can be used to adjust the visibility of the box in different circumstances. Direct size adjustment is not possible, enclosing Transform node scaling, eventually dynamic code handling is needed. We use this geometry type for specifying Segment vertex or Joint positions, and we assign sensors (TouchSensors) to them for selection purposes.

```
Box { field SFVec3f size 2 2 2 }
```

Cone nodes are used in the construction of displacement widgets, to provide directional information (we constructed an arrow PROTO from two cones). We also assign sensors (TouchSensor) to these arrows, for signalling displacement operations.

```
Cone {
        bottomRadius 1
        height 2
        side TRUE
        bottom TRUE
}
```

The **Coordinate** node is a data node, containing a set of vertices. It is used in combination with nodes that are based on vertex coordinate data: IndexedFaceSet and IndexedLineSet nodes. The coordinate values can be handled dynamically.

```
Coordinate { point [] }
```

Cylinders are simple geometry nodes used by us as components of rotational widgets, in combination with rotational sensors (CylinderSensor). These constructions handle rotations around a single axis. We used them as geometry rotation widgets, positioning widgets and even scale widgets.

```
Cylinder {
        radius 1
        height 2
        bottom TRUE
        side TRUE
        top TRUE
}
```

IndexedFaceSet nodes provide an approach to custom geometries where the creator specifies all the coordinates and how these coordinates are used to form polygons. Different properties of the node can be set to provide various appearance options. It is one of the main components of our system, visualising Segment geometries.

```
IndexedFaceSet {
        convex TRUE
        ccw TRUE
        solid TRUE
        colorPerVertex TRUE
        normalPerVertex TRUE
        creaseAngle 0
        coord NULL
        coordIndex []
        color NULL
        colorIndex []
        normal NULL
        normalIndex []
        texCoord NULL
        texCoordIndex []
}
```

IndexedLineSet nodes are similar in construction to IndexedFaceSet nodes, except they provide means to create line geometries instead of polygon faces. We use this type of node to visualise Joint hierarchy connections and the relations between vertices that are belonging to the same groups.

```
IndexedLineSet {
        colorPerVertex TRUE
        coord NULL
        coordIndex []
        color NULL
        colorIndex []
}
```

The **Transform** node provides general geometry handling capabilities to its children nodes. It is possible to position, rotate and scale all nodes below a Transform node, even dynamically if needed. We use this node throughout the environment for positioning the different interface and environment components, for different visualisation, feedback and update purposes.

```
Transform {
        center 0 0 0
        translation 0 0 0
        rotation 0 0 1 0
        scale 1 1 1
        scaleOrientation 0 0 1 0
        bboxCenter 0 0 0
        bboxSize -1 -1 -1
        children []
}
```

B.2 Sensor nodes

When attached to geometries and handling their produced events, sensors become tools that enable user interaction and provide quantitative measurements for the extent of user interactions. In the following we describe the sensors used in the HanimPlus system.

CylinderSensor. This sensor is a rotational drag sensor which produces vertical axis (Y) rotational information as a consequence of user dragging (activating). A different axis rotation can be simulated by using a Script node, separating the actual rotation angle information and using it in combination with other axis settings as a new rotation data field. We use separated axis rotation sensors instead of a single SphereSensor to increase handling accuracy.

```
CylinderSensor {
        diskAngle 0.262
        maxAngle -1
        minAngle 0
        offset 0
        enabled TRUE
        autoOffset TRUE
}
```

The **PlaneSensor** node is a plane-based drag sensor which produces vertical and/or horizontal axis (XY) displacement information, without depth values. Currently we use it only in the unit setting widget, but other uses are also possible.

```
PlaneSensor {
        maxPosition -1 -1
        minPosition 0 0
        offset 0 0 0
        autoOffset TRUE
        enabled TRUE
}
```

SphereSensor nodes provide 3D rotation information from the 2D input modality present in desktop environment: the mouse. It is our observation that this 2D to 3D mapping is not transparent to the user, and a precise rotation control cannot be exercised. We removed SphereSensor rotation widgets from newer versions of our systems, in favour of axis-separated CylinderSensor based rotational widgets.

```
SphereSensor {
        offset 0 1 0  0
        autoOffset TRUE
        enabled TRUE
}
```

The **TouchSensor** node produces information events with pointing device hovering and activation over a target geometry. Signalling when the pointing device is over a sensor-enabled geometry allows the use of feedback methods,

increasing the dynamics of the environment. We use button-like geometries in combination with this sensor to produce push (or click) events, signalling that a parameterless operation was requested.

```
TouchSensor { enabled TRUE }
```

Bibliography

[3D Top, NA] 3D Top (N.A.). Virtual Desktop. http://www.3dtop.com.

[3DWM, NA] 3DWM (N.A.). Window Manager. http://www.3dwm.org.

[ActiveWorlds Inc., NA] ActiveWorlds Inc. (N.A.). ActiveWorlds Servers. http://www.activeworlds.com.

[Adobe Inc., NA] Adobe Inc. (N.A.). Atmosphere. http://www.adobe.com/products/atmosphere/.

[Alexa et al., 2000] Alexa, M., Behr, J., and Müller, W. (2000). The Morph node. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 29–34. ACM Press.

[Alias—Wavefront, NA] Alias—Wavefront (N.A.). Maya Personal Learning Edition. http://www.aliaswavefront.com/en/products/maya/ple/.

[Álvaro Sánchez et al., 1998] Álvaro Sánchez, Aguado, G., de Antonio, A., and Segovia, J. (1998). Designing Verbal Communication for Intelligent Virtual Environments: A Case Study. In *ECAI98 Workshop on Intelligent Virtual Environments*. Detailed Abstract.

[Aubel et al., 1998] Aubel, A., Boulic, R., and Thalmann, D. (1998). Animated impostors for real-time display of numerous virtual humans. In *Virtual Worlds '98*, volume LNAI 1434 of *Lecture Notes in Artificial Intelligence*, pages 14–28. Springer Verlag.

[Azuma et al., 2001] Azuma, R. T., Baillot, Y., Behringer, R., Feiner, S. K., Julier, S., and MacIntyre, B. (2001). Recent advances in augmented reality. *IEEE Computer Graphics and Applications*, 21(6):34–47.

[Babski and Thalmann, 1999] Babski, C. and Thalmann, D. (1999). A seamless shape for HANIM compliant bodies. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 21–28. ACM Press.

[Back and Stone, 1999] Back, M. and Stone, M. (1999). The Kinetic Mandala: Audio A-life in a web-based 3d environment. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 85–91. ACM Press.

[Badler et al., 1999a] Badler, N. I., Bindiganavale, R., Bourne, J., Allbeck, J., Shi, J., and Palmer, M. (1999a). Real time virtual humans. In *Proceedings of International Conference on Digital Media Futures '99*. British Computer Society.

[Badler et al., 1999b] Badler, N. I., Chi, D., and Chopra, S. (1999b). Virtual human animation based on movement observation and cognitive behavior models. In *Computer Animation '99*, pages 128–137.

[Badler et al., 1993] Badler, N. I., Phillips, C. B., and Webber, B. L. (1993). *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press.

[Balaguer, 1999] Balaguer, J.-F. (1999). Less is more: The power of simplicity. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 39–45. ACM Press.

[Bandi and Thalmann, 1997] Bandi, S. and Thalmann, D. (1997). A configuration space approach for efficient animation of human figures. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects (NAM '97)*, pages 38–45. IEEE Computer Society.

[Bandi and Thalmann, 1998] Bandi, S. and Thalmann, D. (1998). Space discretization for efficient human navigation. In *Proc. Eurographics '98 in Computer Graphics Forum Conference Issue*, volume 17(3), pages 195–206.

[Barrilleaux, 2000] Barrilleaux, J. (2000). *3D User Interfaces with Java3D*. Manning.

[Bellan et al., 1998] Bellan, Y., Costa, M., Ferrigno, G., Lombardi, F., Macchiarulo, L., Montuori, A., Pasero, E., and Rigotti, C. (1998). Artificial neural networks for motion emulation in virtual environments. In *CAPTECH'98*, pages 83–99. Springer Verlag.

[Bellman and Landauer, 2000] Bellman, K. L. and Landauer, C. (2000). Playing in the MUD: Virtual worlds are real places. *Applied Artificial Intelligence*, 14(1):93–123.

[Biederman, 1987] Biederman, I. (1987). Recognition-by-components: A theory of human image understanding. *Psychological Review*, (94):115–147.

[Bier, 1986] Bier, E. A. (1986). Skitters and Jacks: Interactive 3D positioning tools. In *Proceedings 1986 ACM Workshop on Interactive 3D Graphics*, pages 183–196. ACM Press.

[Bindiganavale et al., 2000] Bindiganavale, R., Schuler, W., Allbeck, J. M., Badler, N. I., Joshi, A. K., and Palmer, M. (2000). Dynamically Altering Agent Behaviors Using Natural Language Instructions. In *Autonomous Agents 2000*, pages 293–300.

[Bindiganavale, 2000] Bindiganavale, R. N. (2000). *Building Parameterized Action Representations from Observation*. PhD thesis, University of Pennsylvania.

[Blaxxun interactive, NAa] Blaxxun interactive (N.A.a). Avatar Studio. http://www.blaxxun.com/.

[Blaxxun interactive, NAb] Blaxxun interactive (N.A.b). Blaxxun Platform. http://www.blaxxun.com.

[Blender Foundation, NA] Blender Foundation (N.A.). Blender 3D graphics creation suite. http://www.blender3d.org/.

[Blumberg and Galyean, 1995] Blumberg, B. M. and Galyean, T. A. (1995). Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 47–54.

[Boulic et al., 1994] Boulic, R., Mas, R., and Thalmann, D. (1994). Inverse Kinetics for Center of Mass Position Control and Posture Optimization. In *European Workshop on Combined Real and Synthetic Image Processing for Broadcast and Video Production* . Springer Verlag.

[Boulic et al., 1996] Boulic, R., Mas, R., and Thalmann, D. (1996). A robust approach for the control of the center of mass with inverse kinetics. *Computers & Graphics Journal*, 20(5):693–701.

[Bowman and Hodges, 1997a] Bowman, D. A. and Hodges, L. F. (1997a). An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 35–38. ACM Press.

[Bowman and Hodges, 1997b] Bowman, D. A. and Hodges, L. F. (1997b). Toolsets for the Development of Highly Interactive and Information-Rich Virtual Environments. *International Journal of Virtual Reality*, 3(2):12–20.

[Bowman et al., 1997] Bowman, D. A., Koller, D., and Hodges, L. F. (1997). Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control techniques. In *IEEE Proceedings of VRAIS'97*, pages 45–52. IEEE Computer Society.

[Boyd et al., 1999] Boyd, D. R. S., Gallop, J. R., Palmen, K. E. V., Platon, R. T., and Seelig, C. D. (1999). Using Virtual Environments to Enhance Visualization. In *Virtual Environments '99*. Springer Verlag.

[Brogan et al., 1998] Brogan, D. C., Metoyer, R. A., and Hodgins, J. K. (1998). Dynamically Simulated Characters in Virtual Environments. In *IEEE Computer Graphics and Applications*, volume 15(5), pages 58–69. IEEE Computer Society.

[Bruderlin and Calvert, 1996] Bruderlin, A. and Calvert, T. (1996). Knowledge-Driven, Interactive Animation of Human Running. In *Graphics Interface '96*, pages 213–221. Morgan Kaufmann.

[Bruderlin and Williams, 1995] Bruderlin, A. and Williams, L. (1995). Motion Signal Processing. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 97–104.

[Buzan and Buzan, 1995] Buzan, T. and Buzan, B. (1995). *The mind map book*. BBC Books.

[Campbell and Jacobson, 1999] Campbell, C. and Jacobson, J. (1999). Artificial Vestibular Stimulation for Immersive Environments; A Working Peripheral. White paper.

[Card et al., 1999] Card, S. K., Mackinlay, J. D., and Shneiderman, B. (1999). Information visualization. pages 1–34.

[Card et al., 1983] Card, S. K., Moran, T. P., and Newell, A. (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates.

[Carlson and Hodgins, 1997] Carlson, D. A. and Hodgins, J. K. (1997). Simulation Levels of Detail for Real-time Animation. In *Graphics Interface '97*, pages 1–8.

[Carraro et al., 1998] Carraro, G. U., Edmark, J. T., and Ensor, J. R. (1998). Pop-Out Videos. In *Virtual Worlds*, pages 123–128.

[Carson and Clark, 1999] Carson, J. A. and Clark, A. F. (1999). Multicast Shared Virtual Worlds Using VRML97. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 133–140. ACM Press.

[Cassell and Jenkins, 1998] Cassell, J. and Jenkins, H. (1998). *From Barbie to Mortal Kombat: Gender and Computer Games*. MIT Press.

[Cassell et al., 1994] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., and Stone, M. (1994). ANIMATED CONVERSATION: Rule-based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversational Agents. In *Computer Graphics (Siggraph '94 Proceedings)*, volume 28, pages 413–420. ACM Press.

[Cassell et al., 2001] Cassell, J., Vilhjálmsson, H. H., and Bickmore, T. (2001). BEAT: the Behavior Expression Animation Toolkit. In *Computer Graphics (Siggraph '01 Proceedings)*, pages 477–486. ACM Press.

[Cavazza and I.J.Palmer, 1999] Cavazza, M. and I.J.Palmer (1999). High-Level Interpretation in Dynamic Virtual Environments. *Applied Artificial Intelligence, Special Issue on Intelligent Virtual Environments*, 14(1):125–144.

[Çapin et al., 1997] Çapin, T. K., Pandzic, I. S., Magnenat-Thalmann, N., and Thalmann, D. (1997). A Dead-Reckoning Algorithm for Virtual Human Figures. In *VRAIS'97*, pages 161–168. IEEE Computer Society.

[Chao, 2001] Chao, D. (2001). Doom as an Interface for Process Management. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 152–157. ACM Press.

[Chi et al., 2000] Chi, D., Costa, M., Zhao, L., and Badler, N. (2000). The EMOTE Model for Effort and Shape. In *Computer Graphics (Siggraph '00 Proceedings)*, pages 173–182.

[chUmbaLum sOft, NA] chUmbaLum sOft (N.A.). MilkShape 3D. http://www.swissquake.ch/chumbalum-soft/ms3d/.

[Conner et al., 1992] Conner, D. B., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C., and van Dam, A. (1992). Three-dimensional widgets. In *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25(2), pages 183–188. ACM Press.

[Coomans and Timmermanns, 1997] Coomans, M. and Timmermanns, H. (1997). Towards a Taxonomy of Virtual Reality User Interfaces. In *International Conference on Information Visualisation IV'97*. IEEE Computer Society Press.

[Coyne and Sproat, 2001] Coyne, B. and Sproat, R. (2001). WorldsEye: An Automatic Text-to-Scene Conversion System. In *Computer Graphics (Siggraph '01 Proceedings)*, pages 487–496. ACM Press.

[Crabtree et al., 1998] Crabtree, I., Soltysiak, S., and Thint, M. (1998). Adaptive Personal Agents. *Personal Technologies Journal*, 3:141–151.

[Cubaud and Topol, 2001] Cubaud, P. and Topol, A. (2001). A VRML-based User Interface for an Online Digitalized Antiquarian Collection. In *Web3D 2001, Proceedings of the sixth International Conference on 3D Web Technology*, pages 51–59. ACM Press.

[Curious Labs, NA] Curious Labs (N.A.). Avatar Lab. http://www.curiouslabs.com/go/avlab.

[Danforth et al., 2000] Danforth, R., Duchowski, A., Geist, R., and McAliley, E. (2000). A platform for gaze-contingent virtual environments. In *In Smart Graphics (Papers from the 2000 AAAI Spring Symposium, Technical Report SS-00-04)*, volume 2177, pages 66–70. AAAI.

[Darken, 1994] Darken, R. (1994). Hands-off interaction with menus in virtual spaces. In *Proceedings of SPIE 1994, Stereoscopic Displays and Virtual Reality Systems*, volume 2177, pages 365–371.

[Darken et al., 1997] Darken, R. P., Cockayne, W. R., and Carmein, D. (1997). The Omni-Directional Treadmill: A Locomotion Device for Virtual Worlds. In *Proceedings of UIST'97*, pages 213–221. ACM Press.

[Dautenhahn, 1998] Dautenhahn, K. (1998). Story-telling in Virtual Environments. In *Working Notes Intelligent Virtual Environments, Workshop at the 13th European Conference on Artificial Intelligence*.

[Delgado-Mata and Aylett, 2001] Delgado-Mata, C. and Aylett, R. (2001). Communicating Emotion in Virtual Environments through Artificial Scents. In *Intelligent Virtual Agents, Third International Workshop, IVA*, volume 2190 of *LNCS*, pages 36–46.

[Deol et al., 1999] Deol, K. K., Sutcliffe, A., and Maiden, N. (1999). Modelling Interaction To Inform Information Requirements In Virtual Environments. In *TWLT15 Proceedings Twente Workshop on Language Technology 15: Interactions in Virtual Worlds*, pages 107–114.

[Diehl, 2001] Diehl, S. (2001). *Distributed Virtual Worlds: Foundations and Implementation Techniques Using VRML, Java and CORBA*. Springer Verlag.

[Discreet, NA] Discreet (N.A.). GMax.
http://www.discreet.com/products/gmax/.

[Döllner and Hinrichs, 1998] Döllner, J. and Hinrichs, K. (1998). Interactive, Animated 3D Widgets. In *Proceedings of Computer Graphics International CGI'98*. IEEE Computer Society Press.

[Dykes et al., 1999] Dykes, J. A., Moore, K. E., and Fairbairn, D. (1999). From Chernoff to Imhof and Beyond: VRML and Cartography. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*. ACM Press.

[Ekman and Friesen, 1978] Ekman, P. and Friesen, W. V. (1978). *The Facial Action Coding System (FACS): A Technique for the Measurement of Facial Action*. Consulting Psychologists Press.

[Elvins et al., 1997] Elvins, T. T., Nadeau, D. R., and Kirsh, D. (1997). Worldlets - 3D Thumbnails for Wayfinding in Virtual Environments. In *ACM Symposium on User Interface Software and Technology*, pages 21–30. ACM Press.

[Faloutsos et al., 2001] Faloutsos, P., van de Panne, M., and Terzopoulos, D. (2001). Composable Controllers for Physics-Based Character Animation. In *Computer Graphics (Siggraph '01 Proceedings)*, pages 251–260.

[Fellner and Hopp, 1999] Fellner, D. W. and Hopp, A. (1999). VR-LAB - A Distributed Multi-User Environment For Educational Purposes And Presentations. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 121–132. ACM Press.

[Fua et al., 1998] Fua, P., Plänkers, R., and Thalmann, D. (1998). Realistic Human Body Modeling. In *Fifth International Symposium on the 3-D Analysis of Human Movement*.

[Funge et al., 1999] Funge, J., Tu, X., and Terzopoulos, D. (1999). Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. In *Computer Graphics (Siggraph '99 Proceedings)*, pages 29–38.

[Gausemeier et al., 2000] Gausemeier, J., Krumm, H., Molt, T., Ebbesmeyer, P., and Gehrmann, P. (2000). A database driven server for an Internet based plant layout presentation system. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 17–22.

[Gershon et al., 1998] Gershon, N., Eick, S. G., and Card, S. K. (1998). Information Visualization. *ACM Interactions*, 5(2):9–15.

[Gillies and Dodgson, 1999] Gillies, M. F. P. and Dodgson, N. A. (1999). Psychologically-based Walking in a Cluttered Environment. In *Eurographics Short Papers*.

[Giorgetti and Palamidese, 1999] Giorgetti, D. and Palamidese, P. (1999). Manipulating 3D character Animation Media Objects. In *EG 99 Multimedia Workshop*. Springer Verlag.

[Gleicher, 1998] Gleicher, M. (1998). Retargetting Motion to New Characters. In *Computer Graphics (Siggraph '98 Proceedings)*, pages 33–42. ACM Press.

[Gleicher, 2001] Gleicher, M. (2001). Motion path editing. In *Symposium on Interactive 3D Graphics*, pages 195–202. ACM Press.

[Gobbetti and Balaguer, 1995] Gobbetti, E. and Balaguer, J.-F. (1995). An Integrated Environment to Visually Construct 3D Animations. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 395–398. ACM Press.

[Goddard and Sunderam, 1999] Goddard, T. and Sunderam, V. S. (1999). ToolSpace: Web Based 3D Collaboration. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 161–166. ACM Press.

[Goto and Pasko, 2000] Goto, Y. and Pasko, A. (2000). Interactive Modeling of Convolution Surfaces with an Extendable User Interface. In de Sousa, A. and Torres, J. C., editors, *EuroGraphics 2000 - Short presentations*, pages 37–42. EuroGraphics.

[Grahn et al., 2000] Grahn, H., Volk, T., and Wolters, H. J. (2000). NURBS in VRML. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 35–43. ACM Press.

[Grand, 1998] Grand, S. (1998). Bubbles in cyberspace: a cellular approach to virtual environments and intelligent synthetic life forms. In *ECAI 98 workshop on Intelligent Virtual Environments*.

[Gritz, 1999] Gritz, L. I. (1999). *Evolutionary Controller Synthesis for 3-D Character Animation*. PhD thesis, The George Washington University.

[Grossman et al., 2001] Grossman, T., Balakrishnan, R., Kurtenbach, G., Fitzmaurice, G., Khan, A., and Buxton, B. (2001). Interaction techniques for 3D modeling on large displays. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 17–23.

[Grosso et al., 1989] Grosso, M., Quach, R., Otani, E., Zhao, J., Wei, S., Ho, P. H., Lu, J., and Badler, N. I. (1989). Anthropometry for Computer Graphics Human Figures. Technical Report Technical Report MS-CIS-89-71, University of Pennsylvania.

[Grzeszczuk and Terzopoulos, 1995] Grzeszczuk, R. and Terzopoulos, D. (1995). Automated Learning of Muscle-Actuated Locomotion Through Control Abstraction. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 63–70.

[Grzeszczuk et al., 1998] Grzeszczuk, R., Terzopoulos, D., and Hinton, G. (1998). NeuroAnimator: Fast Neural Network Emulation and Control of Physics-Based Models. In *Computer Graphics (Siggraph '98 Proceedings)*, pages 9–20.

[HAWG, 1999] HAWG (1999). H-Anim: Specification for a Standard VRML Humanoid, version 1.1. Web3D Consortium, Humanoid Animation Working Group. http://www.h-anim.org/Specifications/H-Anim1.1/.

[HAWG, 2001] HAWG (2001). The Humanoid Animation Specification, version 2001. Web3D Consortium, Humanoid Animation Working Group. http://www.h-anim.org/Specifications/H-Anim2001/.

[Hodgins, 1996] Hodgins, J. K. (1996). Three-Dimensional Human Running. In *Proceedings of the IEEE Conference on Robotics and Automation*. IEEE Computer Society.

[Hodgins and Pollard, 1997] Hodgins, J. K. and Pollard, N. S. (1997). Adapting Simulated Behavios for New Characters. In *Computer Graphics (Siggraph '97 Proceedings)*, pages 153–162.

[Hodgins et al., 1995] Hodgins, J. K., Wooten, W. L., Brogan, D. C., and O'Brien, J. F. (1995). Animating Human Athletics. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 71–78. ACM Press.

[Huang, 1997] Huang, Z. (1997). *Motion control for human animation*. PhD thesis, Swiss Federal Institute of Technology.

[Icon3D, NA] Icon3D (N.A.). 3D Desktop Icons Launcher. http://www.icon3d.com.

[Igarashi et al., 1999] Igarashi, T., Matsuoka, S., and Tanaka, H. (1999). Teddy: A Sketching Interface for 3D Freeform Design. In *Computer Graphics (Siggraph '99 Proceedings)*, pages 409–416. ACM Press.

[Imbert et al., 1998] Imbert, R., Sanchez, M. I., de Antonio, A., and Segovia, J. (1998). The Amusement Internal Modelling for Believable Behaviour of Avatars in an Intelligent Virtual Environment. In *Workshop in Intelligent Virtual Environments, ECAI European Conference on Artificial Intelligence*.

[Inman et al., 1981] Inman, V. T., Ralston, H. J., Todd, F., and Lieberman, J. C. (1981). *Human Walking*. Williams & Wilkins.

[Iwata, 1999] Iwata, H. (1999). Walking about Virtual Environments on an Infinite Floor. In *IEEE Virtual Reality VR'99*, pages 286–293. IEEE Computer Society.

[Iwata et al., 2001] Iwata, H., Yano, H., and Nakaizumi, F. (2001). Gait Master: A Versatile Locomotion Interface for Uneven Virtual Terrain. In *IEEE Virtual Reality VR2001*, pages 131–137. IEEE Computer Society.

[James J. Kuffner, 1998] James J. Kuffner, J. (1998). Goal-Directed Navigation for Animated Characters Using Real-Time Path Planning and Control. In Magnenat-Thalmann, N. and Thalmann, D., editors, *CAPTECH'98*, number 1537 in LNAI, pages 171–186. Springer Verlag.

[Kardassevitch et al., 1999] Kardassevitch, C., Jessel, J. P., Paulin, M., and Caubet, R. (1999). Improving the Illumination Quality of VRML97 Walkthrough via Intensive Texture Usage. In *Virtual Environments '99*. Springer Verlag.

[Karpenko et al., 2002] Karpenko, O., Hughes, J. F., and Raskar, R. (2002). Free-form sketching with variational implicit surfaces. In *Eurographics (EG2002)*, pages 585–594. Blackwell Publishing.

[Keefe et al., 2001] Keefe, D. F., Feliz, D. A., Moscovich, T., Laidlaw, D. H., and LaViola, J. J. (2001). CavePainting: a fully immersive 3D artistic medium and interactive experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 85–93. ACM Press.

[Kimoto, 1998] Kimoto, H. (1998). Human Centered Virtual Interactive Image World for Image Retrieval. In *Virtual Worlds, First International Conference, VW'98*, volume 1434 of *LNCS*, pages 315–322. Springer Verlag.

[Kiss, 2001a] Kiss, S. (2001a). A simple, visual mesh editor. CTIT Technical Report series No. 01-27 (TR-CTIT-01-27), ISSN 1381-3625. Technical Report, University of Twente.

[Kiss, 2001b] Kiss, S. (2001b). Web Based VRML Modeling. In *IV2001 Information Visualisation*, pages 612–617. IEEE Computer Society Press.

[Kiss, 2002] Kiss, S. (2002). 3D Character Modeling in Virtual Reality. In *IV2002 Information Visualisation*, pages 541–548. IEEE Computer Society Press.

[Kiss and Nijholt, 2003] Kiss, S. and Nijholt, A. (2003). Viewpoint Adaptation during Navigation based on Stimuli from the Virtual Environment. In *Web3D 2003, Proceedings of the seventh International Conference on 3D Web Technology*, pages 19–26. ACM Press.

[Kitamura et al., 1995] Kitamura, Y., Smith, A., Takemura, H., and Kishino, F. (1995). Parallel Algorithms for Real-time Colliding Face Detection. In *International Workshop on Robot and Human Communication*, pages 211–218. IEEE.

[Lasseter, 1987] Lasseter, J. (1987). Principles of Traditional Animation Applied to 3D Computer Animation. pages 35–44. ACM Press.

[Laszlo et al., 1996] Laszlo, J., van de Panne, M., and Fiume, E. (1996). Limit Cycle Control and its Application to the Animation of Balancing and Walking. In *Computer Graphics (Siggraph '96 Proceedings)*, pages 155–162.

[LaViola et al., 2001] LaViola, J. J., Feliz, D. A., Keefe, D. F., and Zeleznik, R. C. (2001). Hands-Free Multi-Scale Navigation in Virtual Environments. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 9–15. ACM Press.

[Lee and Shin, 1999] Lee, J. and Shin, S. Y. (1999). A Hierarchical approach to Interactive Motion Editing for Human-like Figures. In *Computer Graphics (Siggraph '99 Proceedings)*, pages 39–48. ACM Press.

[Lee et al., 1998] Lee, W.-S., Lee, E., and Magnenat-Thalmann, N. (1998). Real Face Communication in a Virtual World. In *Proc. Virtual Worlds 98*, LNAI, pages 1–13. Springer Verlag.

[Lee et al., 1995] Lee, Y., Terzopoulos, D., and Waters, K. (1995). Realistic Modeling for Facial Animation. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 55–62.

[Lindenmayer and Prusinkiewicz, 1990] Lindenmayer, A. and Prusinkiewicz, P. (1990). *Algorithmic Beauty Of Plants*. Springer Verlag.

[Lu and Zhang, 2002] Lu, R. and Zhang, S. (2002). *Automatic generation of Computer Animation: Using AI for Movie Animation*, volume LNAI 2160 of *Lecture Notes in Artificial Intelligence*. Springer Verlag.

[Luttermann and Grauer, 1999] Luttermann, H. and Grauer, M. (1999). VRML History: Storing And Browsing Temporal 3D-Worlds. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 153–160. ACM Press.

[MacKenzie and Buxton, 1992] MacKenzie, I. and Buxton, W. (1992). Extending Fitts' law to two dimensional tasks. In *Proceedings of the CHI '92 Conference on Human Factors in Computing Systems*, pages 219–226. ACM Press.

[Maestri, 1999] Maestri, G. (1999). *Digital Character Animation 2*, volume 1 - Essential Techniques. New Riders Publishing.

[Maestri, 2001] Maestri, G. (2001). *Digital Character Animation 2*, volume 2 - Advanced Techniques. New Riders Publishing.

[Maestri, NA] Maestri, G. (N.A.). Learning to Walk: The Theory and Practice of 3D Character Motion.
http://www.siggraph.org/education/materials/HyperGraph/animation/character_animation/walking/learning_to_walk.htm.

[Magnenat-Thalmann and Thalmann, 1996] Magnenat-Thalmann, N. and Thalmann, D. (1996). Computer Animation. In *Handbook of Computer Science*, pages 1300–1318. CRC Press.

[Matejic, 1993] Matejic, L. (1993). LOG: Building 3D User Interface Widgets by Demonstration. Technical Reports, Brown University, CS-93-22. Technical Report.

[Mauve, 1999] Mauve, M. (1999). Transparent Access To And Encoding Of VRML State Information. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 29–38. ACM Press.

[McDonnell et al., 2001] McDonnell, K. T., Qin, H., and Wlodarczyk, R. A. (2001). Virtual clay: a real-time sculpting system with haptic toolkits. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 179–190. ACM Press.

[McNeill, 1992] McNeill, D. (1992). *Hand and Mind: What Gestures Reveal About Thought*. University of Chicago. Characterizes gestures.

[Mealing, 1992] Mealing, S. (1992). *The art and science of computer animation*. Oxford.

[Mesgari, 2000] Mesgari, S. M. (2000). *Topological cell-tuple structures for three-dimensional spatial data*. ITC dissertation number 74, ITC, The Netherlands.

[Moccozet and Magnenat-Thalmann, 1997] Moccozet, L. and Magnenat-Thalmann, N. (1997). Dirichlet Free-Form Deformations and their Application to Hand Simulation. In *Proc. Computer Animation '97*, pages 93–102. IEEE Computer Society.

[Moccozet and Thalmann, 1997] Moccozet, L. and Thalmann, N. M. (1997). Multilevel Deformation Model Applied to Hand Simulation for Virtual Actors. In *International Conference on Virtual Systems and MultiMedia (VSMM'97)*, pages 119–128. IEEE Computer Society.

[Murphy and Pitt, 2001] Murphy, D. and Pitt, I. (2001). Spatial Sound Enhancing Virtual Storytelling. In *International Conference on Virtual Storytelling 2001: Using Virtual Reality Technologies for Storytelling*, volume 2197 of *LNCS*, pages 20–29. Springer Verlag.

[Naka et al., 1999] Naka, T., Mochizuki, Y., Hijiri, T., Cornish, T., and Asahara, S. (1999). A Compression/Decompression Method for Streaming Based Humanoid Animation. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 63–70. ACM Press.

[Nebel, 2001] Nebel, J.-C. (2001). Generation of True 3D Films. In *International Conference on Virtual Storytelling 2001: Using Virtual Reality Technologies for Storytelling*, pages 10–19. Springer Verlag.

[Nijholt and Hondorp, 2000] Nijholt, A. and Hondorp, H. (2000). Towards Communicating Agents and Avatars in Virtual Worlds. In de Sousa, A. and Torres, J. C., editors, *EuroGraphics 2000 - Short presentations*, pages 91–95. EuroGraphics.

[Noser and Thalmann, 1996] Noser, H. and Thalmann, D. (1996). The Animation of Autonomous Actors Based on Production Rules. In *Proc. Computer Animation '96*, pages 47–57. IEEE Computer Society Press.

[Noser and Thalmann, 1998] Noser, H. and Thalmann, D. (1998). Towards Autonomous Synthetic Actors. In T. L. Kunii, A. L., editor, *Chapter 9 in Cyberworlds*, pages 143–158. Springer Verlag.

[Nousch and Jung, 1999] Nousch, M. and Jung, B. (1999). CAD on the World Wide Web: Virtual Assembly of Furniture with BEAVER. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 113–119. ACM Press.

[Paoluzzi et al., 1999] Paoluzzi, A., Francesi, S., Portuesi, S., and Vicentino, M. (1999). Rapid Development of VRML Content via Geometric Programming. In *Virtual Environments '99*. Springer Verlag.

[ParallelGraphics, NA] ParallelGraphics (N.A.). Cortona VRML Viewer. http://www.parallelgraphics.com/.

[Perbet and Cani, 2001] Perbet, F. and Cani, M.-P. (2001). Animating prairies in real-time. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 103 – 110. ACM Press.

[Perlin and Goldberg, 1996] Perlin, K. and Goldberg, A. (1996). Improv: A System for Scripting Interactive Actors in Virtual Worlds. In *Computer Graphics (Siggraph '96 Proceedings)*, pages 205–216.

[Rao and Card, 1994] Rao, R. and Card, S. (1994). The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. In *ACM Conference on Human Factors in Software (CHI '94)*, pages 318–322. ACM Press.

[Reddy et al., 2000] Reddy, M., Iverson, L., and Leclere, Y. G. (2000). Under the hood of GeoVRML 1.0. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 23–28. ACM Press.

[Reijers et al., 2002] Reijers, N., Cunningham, R., Meier, R., Hughes, B., Gaertner, G., and Cahill, V. (2002). Using Group Communication to Support Mobile Augmented Reality Applications. In *Fifth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, page 297. IEEE Computer Society.

[Ressler et al., 1999] Ressler, S., Godil, A., Wang, Q., and Seidman, G. (1999). A VRML Integration Methodology for Manufacturing Applications. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 167–172. ACM Press.

[Rheingold, 1991] Rheingold, H. (1991). *Virtual Reality*. New York: Touchstone.

[Rickel and Johnson, 1997a] Rickel, J. and Johnson, W. L. (1997a). Integrating Pedagogical Capabilities in a Virtual Environment Agent. In *Proc. of First International Conference on Autonomous Agents*, pages 30–38. ACM Press.

[Rickel and Johnson, 1997b] Rickel, J. and Johnson, W. L. (1997b). Task-Oriented Dialogs with Animated Agents in Virtual Reality. In *Proc. of the First Workshop on Embodied Conversational Characters*, pages 39–46. ACM Press.

[Robertson et al., 1991] Robertson, G., Mackinlay, J., and Card, S. (1991). Animated 3D visualizations of hierarchical information. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 189–194. ACM Press.

[ROOMS 3D, NA] ROOMS 3D (N.A.). 3d desktops. http://www.rooms3d.com.sg.

[Rose et al., 1998] Rose, C., Bodenheimer, B., and Cohen, M. F. (1998). Verbs and Adverbs: Multidimensional Motion Interpolation using Radial Basis Functions. *IEEE Journal of Computer Graphics and Applications*, 18(5):32–41.

[Roussos and Bizri, 1998] Roussos, M. and Bizri, H. (1998). Mitologies: Medieval Labyrinth Narratives in Virtual Reality. In *Proceedings of 1st International Conference on Virtual Worlds*, pages 373–383. Springer Verlag.

[Ruttkay et al., 1998] Ruttkay, Z., ten Hagen, P., Noot, H., and Savenije, M. (1998). Facial Animation by Synthesis of Captured and Artificial Data. In *CAPTECH '98 Proceedings*, pages 268–271.

[Saar, 1999] Saar, K. (1999). VIRTUS: A Collaborative Multi-User Platform. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*. ACM Press.

[Sannier et al., 1998] Sannier, G., Balcisoy, S., Magnenat-Thalmann, N., and Thalmann, D. (1998). An Interactive Interface for Directing Virtual Humans. In *Proc. ISCIS'98*. IOS Press.

[Sannier et al., 1999] Sannier, G., Balcisoy, S., Magnenat-Thalmann, N., and Thalmann, D. (1999). VHD: A System for Directing Real-Time Virtual Actors. In *The Visual Computer*, volume 15, No 7/8, pages 320–329. Springer Verlag.

[Schaerf and Tessicini, 1999] Schaerf, M. and Tessicini, A. (1999). JubilEasy: Build a personalized 3D visit of Rome. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 105–112. ACM Press.

[Schönhage and Eliëns, 1999] Schönhage, B. and Eliëns, A. (1999). Dynamic and Mobile VRML Gadgets. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 47–52. ACM Press.

[Schreiber et al., 2000] Schreiber, A. T., Akkermans, J. M., Anjewierden, A. A., de Hoog, R., Shadbolt, N. R., de Velde, W. V., and Wielinga, B. J. (2000). *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press.

[Schuemie and van der Mast, 1999] Schuemie, M. J. and van der Mast, C. A. (1999). Presence: Interacting in Virtual Reality? In *TWLT15 Proceedings Twente Workshop on Language Technology 15: Interactions in Virtual Worlds*, pages 213–217. University of Twente.

[Sederberg and Parry, 1986] Sederberg, T. W. and Parry, S. R. (1986). Free-Form Deformation of Solid Geometric Models. In *Proceedings of SIGGRAPH '86, Computer Graphics 20, 4*, pages 151–159.

[SENX, NA] SENX (N.A.). Scent and taste device. http://www.trisenx.com/.

[SGI, NA] SGI (N.A.). CosmoWorlds. . Commercialization of PC version ceased; software available for SGI Iris only.

[Shiode and Kanoshima, 1999] Shiode, N. and Kanoshima, T. (1999). Utilising the Spatial Features of Cyberspace for Generating a Dynamic Museum Environment. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 79–84. ACM Press.

[Sloan et al., 2001] Sloan, P.-P. J., Rose, C. F., and Cohen, M. F. (2001). Shape by example. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 135–143. ACM Press.

[Smith, 2000] Smith, J. (May 2000). Seamless VRML Humans. In *Scanning 2000 Conference*.

[Smith, 1996] Smith, W. J. (1996). *ISO and ANSI Ergonomic Standards for Computer Products: A Guide to Implementation and Compliance*. Prentice Hall.

[Steed, 1997] Steed, A. (1997). Efficient Navigation Around Complex Virtual Environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 173–180. ACM Press.

[Stewart and Cremer, 1992] Stewart, A. J. and Cremer, J. F. (1992). Beyond Keyframing: An Algorithmic Approach to Animation. In *Graphics Interface '92*, pages 273–281. Morgan Kaufmann.

[Strauss et al., 1999] Strauss, W., Fleischmann, M., Thomsen, M., Novak, J., Zlender, U., Kulessa, T., and Pragasky, F. (1999). Staging the space of mixed reality Reconsidering the concept of a multi user environment. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 93–98. ACM Press.

[Summer et al., 1998] Summer, R. W., O'Brien, J. F., and Hodgins, J. K. (1998). Animating sand, mud, and snow. In *Proceedings of Graphics Interface '98*, pages 125–132. Morgan Kaufmann.

[Sun and Metaxas, 2001] Sun, H. C. and Metaxas, D. N. (2001). Automatic Gait Generation. In *Computer Graphics (Siggraph '01 Proceedings)*, pages 261–270. ACM Press.

[Thalmann, 1996] Thalmann, D. (1996). Physical, Behavioral, and Sensor-Based Animation. In *Graphicon'96*. GRAFO Computer Graphics Society.

[Thalmann, 2001] Thalmann, D. (2001). The Foundations to Build a Virtual Human Society. In *Intelligent Virtual Agents Workshop (IVA2001)*, volume 2190 of *LNAI*. Springer Verlag.

[Thalmann et al., 1997] Thalmann, D., Noser, H., and Huang, Z. (1997). Autonomous Virtual Actors based on Virtual Sensors. In *Creating Personalities for Synthetic Actors: Towards Autonomous Personality Agents*, volume 1195 of *LNAI*, pages 25–42. Springer Verlag.

[Tu, 1999] Tu, X. (1999). *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*, volume 1635 of *LNCS*. Springer Verlag.

[Unuma et al., 1995] Unuma, M., Anjyo, K., and Takeuchi, R. (1995). Fourier Principles for Emotion-based Human Figure Animation. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 91–96.

[Usoh et al., 1999] Usoh, M., Arthur, K., Whitton, M. C., Bastos, R., Steed, A., Slater, M., and Frederick P. Brooks, J. (1999). Walking → Walking-in-Place → Flying, In Virtual Environments. In *Computer Graphics (Siggraph '99 Proceedings)*, pages 359–364. ACM Press.

[Vertegaal, 1999] Vertegaal, R. (1999). The GAZE Groupware System: Mediating Joint Attention in Multiparty Communication and Collaboration. In *CHI*, pages 294–301.

[Volino and Magnenat-Thalmann, 2000] Volino, P. and Magnenat-Thalmann, N. (2000). *Virtual Clothing : Theory and Practice*. Springer Verlag.

[Waibel and Duchnowski, 1994] Waibel, A. and Duchnowski, P. (1994). Connectionist Models in Multimodal Human-Computer Interaction. In *Proceedings of the 1994 Government Microcircuit Applications Conference (GOMAC 94)*.

[Wakita et al., 2000] Wakita, A., Yajima, M., Harada, T., Toriya, H., and Chiyokura, H. (2000). XVL: A Compact And Qualified 3D Representation With Lattice Mesh and Surface for the Internet. In *Proceedings of the Web3D-VRML 2000 Fifth Symposium on Virtual Reality Modeling Language*, pages 45–51. ACM Press.

[Walsh and Bourges-Sevenier, 2000] Walsh, A. E. and Bourges-Sevenier, M. (2000). *Core Web3D*. Prentice Hall.

[Wang and MacKenzie, 1999] Wang, Y. and MacKenzie, C. L. (1999). Object Manipulation in Virtual Environments. In *ACM CHI'99*, pages 48–55.

[Ware, 2000] Ware, C. (2000). *Information Visualization: Perception for Design*. Academic Press.

[Waters, 1987] Waters, K. (1987). A muscle model for animation three-dimensional facial expression. pages 17–24. ACM Press.

[Web3D Consortium, 1997] Web3D Consortium (1997). VRML97: The Virtual Reality Modeling Language.
http://www.web3d.org/technicalinfo/specifications/vrml97/.

[Webber, 1998] Webber, B. (1998). Instructing Animated Agents: Viewing Language in Behavioral Terms. In *Multimodal Human-Computer Communication*, volume 1374 of *Lecture Notes in Computer Science*, pages 89–100. Springer Verlag.

[Whittle, 1995] Whittle, M. W. (1995). Musculoskeletal Applications of Three-Dimensional Analysis. In Allard, P. and Stokes, I. A., editors, *Three-Dimensional Analysis of Human Movement*, chapter 13, pages 295–309. Human Kinetics Publishers.

[Wilson, 1998] Wilson, M. (1998). Position Paper: Agents and Activities in the Intelligent Virtual Environment Brightown. In *ECAI 98 workshop on Intelligent Virtual Environments*.

[Win3D, NA] Win3D (N.A.). 3D Desktop.
http://www.clockwise3d.com/products/products.html.

[Witkin and Popović, 1995] Witkin, A. and Popović, Z. (1995). Motion Warping. In *Computer Graphics (Siggraph '95 Proceedings)*, pages 105–108. ACM Press.

[Wooldridge and Ciancarini, 2000] Wooldridge, M. J. and Ciancarini, P. (2000). Agent-Oriented Software Engineering: The State of the Art. In *First International Workshop on Agent-Oriented Software Engineering (AOSE)*, pages 1–28. Springer Verlag.

[Wooten and Hodgins, 1996] Wooten, W. L. and Hodgins, J. K. (1996). Animation of Human Diving. 15(1):3–14.

[Wray and Belrose, 1999] Wray, M. and Belrose, V. (1999). Avatars in LivingSpace. In *VRML '99, Proceedings of the Fourth Symposium on the Virtual Reality Modeling Language*, pages 13–19. ACM Press.

[Yen and Séquin, 2001] Yen, J. and Séquin, C. (2001). Escher sphere construction kit. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pages 95–98. ACM Press.

Summary

The subject of this thesis belongs to the area of computer graphics, particularly, to the domain of graphics modelling. This thesis presents experiments in graphics modelling, namely in the application of graphics modelling principles to virtual reality (VR), with an interaction approach that is typical to VR systems. Our approach takes advantage of novel graphics primitives that take the form of sets of vertices, of non-standard approaches to interface construction with the use of controls integrated into the 3D environment and also of the advantages that the widget technologies are offering. A number of information visualisation, interaction and manipulation options have been adapted to our case, to provide a 3D character editing virtual environment that is easy to use, aimed at novice users and beyond. The resulting system is an implementation based on the Virtual Reality Modelling Language (VRML), which means we are using a visualisation tool that is at the same time the target environment for the resulting 3D characters. This leaves room for experimentations in 3D visualisation and interaction techniques, a field that currently lacks in widely accepted standards, specifications or recommendations.

The target audience we have in mind is users inexperienced in graphics modelling, occasional users or even novice users. This is the reason for our simplified, natural approach to graphics components, effective visualisation, simple interface, direct manipulation and natural graphics operations, resulting in a minimal learning overhead.

Creating 3D models to be used in virtual environments with graphics tools that follow the traditional WIMP (Windows Icons Menus and Pointers) interfaces does not seem to be inspired. Observing that we target 3D character objects (hierarchical constructions) and not whole scenes, it is possible to integrate the controls in the environment itself. This VR and interactive modelling combination takes advantage of the innate visualisation possibilities, and our main task becomes the content handling task, which unfortunately with current VR systems like the VRML technology we use requires considerable efforts due to the still limited dynamical properties of such systems regarding the updating and handling of graphical components.

Graphics modelling systems are constructed around a set of graphics primitives. Unlike the classical vertex, edge and polygon solution, we use a novel approach based on single vertices and collections of vertices, collections deduced from the properties of the graphics meshes we use. Deciding to take advantage of the different properties of regular meshes like possibility of automatic polygon handling, decreased data size and, most importantly, the natural classification of component vertices, we did set up primitives that represent ring and column collections of vertices. The restriction that occurs from using only regular

meshes is compensated in this case with faster handling, increased effectiveness and a natural manipulation paradigm.

This approach results in a hybrid manipulation possibility. A single vertex provides low-level manipulation possibilities, which are elevated to an increased extent if the target entity is an entire set of vertices. We also define operations that are oriented towards entire sets of vertices, which takes our approach above the low-level manipulation paradigm. The effective, direct selection and manipulation operations also reinforce our attempt for a higher-level, effective manipulation interface.

Regarding geometry manipulation, we have two main objectives. The first one is to allow the user to control the geometry complexity, with the restriction imposed by the regular mesh criteria. Users can add or remove sets of vertices to retain regularity. The second objective is the automatic handling of symmetry, since for the majority of 3D characters this is a common property. However, for the purpose of not being too restrictive, we also allow asymmetric meshes in our system. Symmetry is handled at two levels, viz. internally when a geometric object is symmetric in itself, respectively externally, when there are two geometry objects on the two sides of the symmetry plane, mirroring each other. Symmetry properties are specified in the character hierarchies, and the geometries inherit these properties. Both symmetries are common in 3D characters.

The underlying structure of a 3D character is not visible. We extract this structure information and visualise it for editing purposes. The structure itself is based on the H-anim humanoid guidelines specification, consisting of Joint and Segment objects representing bone connections respectively geometries. We use these as basis for our character hierarchies, but we do not impose naming and component limits, allowing the construction of characters that are not based on a humanoid-like skeleton. The geometries are handled one by one by the geometry modelling subsystem, whereas the hierarchy is handled in its entirety by the hierarchy modelling subsystem, where the emphasis is on the hierarchy extension and refinement modalities. The hierarchy contains at each Joint object a geometry which is a separate component of the whole character hierarchy. Local hierarchy editing principles are applied (child Joints are not affected in the subsequent hierarchy), and the geometry deforms according to the hierarchy operations effectuated.

For the interface, a simplistic approach has been used. For visualisation purposes, the system has been divided into two editing modes: a geometry editing mode and a hierarchy editing mode. In both cases, the complete character is visualised next to the edited single geometry or next to the hierarchy visualisation. All possible operations are visualised in these modes all the time, but the ones that are not applicable in the current selection context are disabled. Since the provided geometrical and constructional operators are limited in number, the interface operations are easily accessible and easy to remember.

Integrating the interface into the environment and the use of 3D in the interface gives the possibility to apply widget construction principles for the tools representing our interface operations. We use the widget construction recommendations (geometry and linked behaviour) in case of operations that require the handling of quantitative information, not only an activation signal. Displacement, unit setting, rotational, zoom, etc. widgets have been created with this method for natural and precise interaction capabilities, increasing the

usability of the system.

Our ultimate goal is to provide interactive editing capabilities in virtual environments, especially for 3D characters, whether it is the geometry, the hierarchy, the materials (textures), the animation, or other, related modelling possibilities. Within this process, we can specify, design and evaluate visualisation solutions, interactive modelling technologies, and we can evaluate the feasibility, usability of these solutions. Doing this, we also assess the applicability of running 3D editor style systems on-line, which becomes increasingly plausible with today's hardware advances and increased broadband connection availability. This allows for the occasional use of on-line modelling systems, where the system itself is small in size (for easy download) and all data handling, modelling operations are effectuated locally.

Samenvatting

Het onderwerp van dit proefschrift behoort tot de zogenaamde "Computer Grafiek", beter bekend onder de Engelse naam "Computer Graphics". Dit proefschrift beschrijft experimenten in het grafisch modelleren. In het bijzonder wordt gekeken naar het toepassen van concepten uit Virtual Reality (VR). Gekozen is voor een benadering van de modelleringinteracties die typisch is voor VR systemen. Onze benadering maakt gebruik van nieuwe grafische primitieven zoals verzamelingen van coördinaten, niet-conventionele methoden van de interface door gebruik te maken van in een 3D-wereld geïntegreerde besturing- en manipulatiemogelijkheid en ook van voordelen die geboden worden door de zogenaamde 'widget' technologieën. Een aantal opties voor informatievisualisatie, interacties en manipulaties werden aangepast aan onze wens om tot een systeem te komen voor het modelleren van 3D-objecten in een virtuele wereld dat zich leent voor gebruik door nieuwelingen en andere gebruikers. Het verkregen systeem is gebaseerd op Virtual Reality Modeling Language (VRML). Dat betekent dat wij een gereedschap voor visualisatie gebruiken die tegelijkertijd doelomgeving voor de resulterende 3D-objecten is. Dit laat het experimenteren met 3D-visualisatie en interactietechnieken toe. Dit is een onderzoeksgebied dat op dit moment een gebrek aan algemeen geaccepteerde standaarden, specificaties of aanbevelingen toont.

Het publiek waar wij ons op richten zijn incidentele gebruikers en gebruikers die niet ervaren zijn in het grafisch modelleren. Dit is de reden dat we gekozen hebben voor een directe en natuurlijke benadering van grafische componenten, voor effectieve visualisatie, voor een eenvoudige interface, en voor directe manipulatie en grafische operaties, resulterend in minimale overhead wat betreft het leren omgaan met het systeem.

Het creëren van 3D-figuren voor virtuele omgevingen met behulp van grafische gereedschappen die gebaseerd zijn op de traditionele WIMP (Windows Icons Menus and Pointers) interface is weinig inspirerend. We merken op dat ons doel zich beperkt tot de constructie van (hiërarchisch opgebouwde) 3D-objecten, niet van gehele virtuele werelden. We kunnen daarom de besturingsstructuren in de omgeving zelf integreren. Deze combinatie van virtual reality en interactieve modellering gebruikt de ingebouwde visualisatiemogelijkheden van een VR-systeem, en de gebruiker kan zich beperken tot zijn hoofdtaak: het omgaan met inhoud. Bij de huidige VR-systemen, zoals bijvoorbeeld VRML-technologie, vraagt dit helaas behoorlijke inspanningen vanwege de nog steeds beperkte dynamische eigenschappen van deze systemen wat betreft het bijwerken en manipuleren van grafische componenten.

Systemen voor grafisch modelleren zijn georganiseerd rond een aantal grafische primitieven zoals knopen, lijnen en veelhoeken (polygonen). Afwijkend van

dit traditionele model gebruiken wij een benadering die gebaseerd is op afzonderlijke coördinaten en verzamelingen van coördinaten die afgeleid zijn uit de eigenschappen van de grafische componenten ('meshes') die we gebruiken. Wij maken daarbij gebruik van de verschillende voordelen van reguliere grafische objecten, zoals automatische polygoonverwerking, reductie van data en, uiterst belangrijk, een natuurlijke classificatie van de coördinaten van de componenten. Bij ons leidt dit tot primitieven bestaande uit verzamelingen van knooppunten in de vorm van ringen en kolommen. Tegenover de nadelen die voortvloeien uit het gebruik van slechts reguliere objecten staan de voordelen van het snel kunnen manipuleren, de toegenomen effectiviteit en de natuurlijke manipulatie.

Deze benadering maakt hybride manipulatie mogelijk. Een individueel knooppunt biedt manipulatiemogelijkheden op een laag niveau, maar die mogelijkheden worden opgetrokken tot een hoger niveau aangezien het doel een verzameling van knooppunten kan zijn. Op een hoger niveau worden verzamelingen van knooppunten gemanipuleerd, waardoor onze benadering uitstijgt boven die van manipulaties op een laag niveau. De effectieve, directe keuze- en manipulatieoperaties versterken ook ons streven om te komen tot een interface voor effectieve hoogniveau manipulaties.

Wat betreft geometriemanipulatie hebben we twee doelen. Het eerste is de gebruiker de gelegenheid te bieden de geometriecomplexiteit te beheersen, gegeven de restricties die voortvloeien uit de criteria horend bij de reguliere objecten. Gebruikers kunnen verzamelingen knooppunten toevoegen en verwijderen en daarbij de regulariteit behouden. Het tweede doel is het automatische omgaan met symmetrie, aangezien dit voor de meerderheid van 3D-objecten een algemene eigenschap is. Echter, om niet al te restrictief te zijn, staan we ook niet-symmetrische objecten toe in ons systeem. Symmetrie wordt op twee niveau's gebruikt: in de eerste plaats wanneer de geometrie van afzondelijke objecten zelf interne symmetrie bezit,en in de tweede plaats wanneer twee verschillende geometrische objecten elkaars spiegelbeeld zijn in een hierarchie van objecten. Deze laatste vorm van symmetrie wordt gespecificeerd in de figuurhierarchiëen, en de geometrie van de afzonderlijke delen erft deze eigenschappen. Beide vormen van symmetrie komen algemeen voor bij 3D-figuren.

De onderliggende structuur van een 3D-figuur is niet zichtbaar. Wij extraheren deze structuurinformatie en visualiseren deze voor 'editing'-doeleinden. De structuur, gebaseerd op de H-anim humanoide richtlijnspecificatie, bestaat uit 'joint'- en 'segment'-onderdelen die als het ware leiden tot een gewrichtenledematen geometrie. We gebruiken deze als basis voor onze objecthiërarchieën, maar we houden niet vast aan de naamgeving of de beperkingen van de componenten, dat wil zeggen, we laten de constructie van objecten toe die niet gebaseerd zijn op mensachtige skeletten. De onderdelen van een hiërarchie kunnen afzonderlijk worden bewerkt in een subsysteem voor geometriemodellering, terwijl de hiërarchie in zijn geheel kan worden bewerkt in het subsysteem voor hiërarchiemodellering. Daar ligt de nadruk op modaliteiten voor uitbreiding en verfijning van de hiërarchie. Bij ieder 'joint'-object heeft de hiërarchie een geometrie die een aparte component is van de gehele objecthiërarchie. Lokale principes voor hiërarchiebewerking worden toegepast (onderdelen op een lager niveau worden niet beïnvloed), en de geometrie vervormt in overeenstemming met de uitgevoerde hiërarchieoperaties.

Een eenvoudige aanpak is gevolgd voor de interface. Voor visualisatiedoeleinden is het systeem opgedeeld in twee bewerkingstoestanden: een geometrie-

editing mode en een hiërarchie-editing mode. In beide gevallen wordt het volledige object gevisualiseerd naast de visualisatie van het te bewerken onderdeel of de te bewerken hiërarchie. Alle mogelijke operaties voor beide modes zijn steeds gevisualiseerd, maar die operaties die niet toepasbaar zijn in de gekozen mode zijn 'uitgezet'. Het aantal operaties voor aanpassing van de geometrie en constructie van de hiërarchie is tamelijk beperkt, waardoor de interfaceoperators gemakkelijk toegankelijk zijn en eenvoudig te onthouden.

De integratie van de interface in de omgeving en het gebruik van 3D in de interface maakt het mogelijk om enkele principes voor 'widget'-constructie toe te passen voor de gereedschappen die onze interfaceoperaties representeren. Bij operaties die niet alleen een activeringssignaal vergen maar ook kwantitatieve informatie, tonen we gebruiksaanbevelingen voor geometrie en eraan gekoppelde 'widgets'. Voor verplaatsingen, het instellen van eenheden, rotaties, zoominstellingen, etc., zijn 'widgets' gecreëerd die vanwege hun natuurlijke en nauwkeurige interactiemogelijkheden de bruikbaarheid van het systeem vergroten.

Ons uiteindelijke doel is het aanbieden van interactieve bewerkingsmogelijkheden in virtuele omgevingen, in het bijzonder voor 3D-objecten, waarbij het kan gaan om het modelleren van de geometrie, de hiërarchie, de materialen ('textures'), de animaties of andere, gerelateerde modelleermogelijkheden. Tijdens dit proces kunnen we voorstellen voor visualisatie en technieken voor interactief modelleren toetsen op maakbaarheid en bruikbaarheid. Ook kan de toepasbaarheid van on-line 3D-editor-systemen beschouwd worden. Deze mogelijkheid wordt steeds meer aantrekkelijk vanwege hardwareontwikkeling en toegenomen beschikbaarheid van breedbandverbindingen. Dit laat het incidenteel gebruik toe van on-line modelleringsystemen, waarbij het systeem zelf niet al te groot is in afmetingen terwijl alle datamanipulaties en modelleeracties lokaal geëffectueerd worden.