Interactive shape modelling for mathematics visualization

Lai, Danbo

2014

Lai, D. (2014). Interactive shape modelling for mathematics visualization. Doctoral thesis,
Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/61850>

<https://doi.org/10.32657/10356/61850>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**INTERACTIVE SHAPE MODELLING
FOR MATHEMATICS VISUALIZATION**

INTERACTIVE SHAPE MODELLING FOR MATHEMATICS VISUALIZATION

DANBO LAI

DANBO LAI

SCHOOL OF COMPUTER ENGINEERING

2014

2014

INTERACTIVE SHAPE MODELLING FOR MATHEMATICS VISUALIZATION

DANBO LAI

DANBO LAI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2014

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Alexei Sourin, for his invaluable guidance, support, and suggestions throughout my graduate study in Nanyang Technological University. His expertise, understanding, and patience added considerably to my graduate experience.

Thanks also go to my co-supervisor, Prof. Zhao Dongsheng of National Institute of Education, Singapore. His knowledge, suggestions, and discussions helped me with my research. His encouragement also helped me to overcome the difficulties encountered in my research.

I would also like to thank the Institute for Media Innovation (IMI) for providing me the opportunity and the scholarships to further my study and research.

I also wish to thank my parents and my husband for their unwavering moral support and endless love. Without them, I would never achieve what I have had now.

I also thank my colleagues at Fraunhofer IDM@NTU and at IMI for their friendship and support.

Last but not least, I would like to thank the School of Computer Engineering and Nanyang Technological University (Singapore) for providing me the opportunity to further my study and research.

Abstract

Existing educational virtual spaces simulate either traditional education environments (e.g., lecture theatres, seminar rooms, laboratories) or various phenomena in the real world (e.g., volcano eruptions, tornados, micro and macro scenes). However, subjects richly infused with mathematics and geometry, require from instructors principally new virtual educational spaces, which do not exist in real world. We propose a new approach called to mathematic visualization which emphasizes how complex geometry, colours, textures and other possible properties of virtual shapes can be defined with mathematical formulas and procedures. It immerses the learners into the world of mathematical definitions in explicit, implicit and parametric forms which can be rendered on any suitable graphics system. Since function definitions are small in size compared to traditionally used polygons, the educational environments can be easily used in shared virtual spaces on the internet.

To achieve interactivity while working with visual representation of complex mathematical definitions, special measures have to be proposed to accelerate rendering of the shapes defined by implicit functions. We proposed an algorithm for accelerating such rendering for shape modelling where some initial shape defined by an implicit function is gradually modified by other implicitly-defined shapes (tools) with relatively smaller sizes compared to the final shape. It is based on minimization of the volume of the bounding boxes of the tools. The algorithm adds additional mathematical operations to the final function script while the shape is being modelled interactively. Hence, the final definition of the shape is still a function script that can be rendered faster. The proposed method also permits conversion of any originally unstructured function definitions into the accelerated

function scripts.

As a proof of concept, an educational software tool was implemented for teaching computer graphics and virtual reality to undergraduate and graduate students of Nanyang Technological University.

Contents

Acknowledgements	1
Abstract	3
Contents.....	5
List of Figures	9
Chapter 1 Introduction	13
1.1 Background & Motivation	13
1.2 Objectives	15
1.3 Proposed approach.....	16
1.4 Thesis organization.....	17
Chapter 2 Visualization of mathematics in cyberspace.....	18
2.1 Electronic education	18
2.2 Problems of electronic education when teaching mathematics.....	20
2.3 Approaches to electronic education when teaching mathematics.....	22
2.4 Summary.....	31
Chapter 3 Mathematics Visualization.....	33
3.1 Hypothesis	33
3.2 The proposed concept	34
3.4 Design and implementation of the modelling software	36
3.4.1 Design principles	37
3.4.2 Implementation issues	38

3.4.3	Introduction to FVRML/FX3D.....	41
3.4.4	User interface examples.....	44
3.5	Necessity of acceleration	46
3.6	Summary.....	47
Chapter 4	Existing approaches to acceleration.....	49
4.1	Rendering implicitly defined objects	49
4.1.1	Implicit surface representation.....	49
4.1.2	Visualization methods	50
4.1.3	Recent research on implicit surfaces	52
4.2	Acceleration approaches	53
4.3	Objectives for the acceleration algorithm	55
4.4	Summary.....	56
Chapter 5	Proposed acceleration algorithm.....	57
5.1	Hypothesis of the algorithm.....	57
5.2	Proposed algorithm.....	58
5.3	Further improvement for special cases	67
5.4	Summary.....	69
Chapter 6	Algorithm implementation.....	71
6.1	Algorithm implementation.....	71
6.2	Clustering algorithm implementation	74
6.2.1	Data structure.....	74
6.2.2	Insertion.....	75

6.3	Integration with the interactive shape modelling tool.....	77
6.4	Summary.....	78
Chapter 7	Algorithm performance analysis	79
7.1	Experiments and results.....	79
7.2	Theoretical analysis	85
7.3	Summary.....	89
Chapter 8	User Testing of the Developed Software.....	90
8.1	Interactive shape modelling tool.....	90
8.2	Selected students work	99
8.3	Summary.....	101
Chapter 9	Rendering function-based shape models using POV-Ray.....	102
9.1	Rendering on POV-Ray.....	102
9.2	Experiments	105
9.3	Summary.....	108
Chapter 10	Conclusion and future work	109
10.1	Conclusion	109
10.2	Future Work.....	111
10.2.1	Immersive Interaction & Unified User Interface	111
10.2.2	Acceleration optimisation	113
10.2.3	Physical Properties and Haptic Interaction	113
10.2.4	Audible Mathematics.....	114

References.....116

Appendix List of Publications121

Journal Papers.....121

Conference Papers121

List of Figures

Figure 2.1	A virtual learning environment augmented with haptic device for 3D geometry education (from [45]).....	24
Figure 2.2	Snapshot of the web-enabled VRMath system (from [14]).....	24
Figure 2.3	A System uses CAVE virtual environment (from [48]).....	25
Figure 2.4	Augmented reality tool - Construct3D (from [49])	28
Figure 2.5	Shared virtual space application CyberMath (from http://www.nada.kth.se/~gustavt/cybermath/)	28
Figure 2.6	Snapshots of software from SpaceTime (from the website http://www.spacetime.us/).....	29
Figure 2.7	IMAGINARY exhibitions (from http://www.imaginary2008.de/).....	29
Figure 3.1	System architecture	40
Figure 3.2	Extended function-defined nodes of VRML and X3D.....	43
Figure 3.3	Interactively manipulating appearance of the initial shape	45
Figure 3.4	Exploring shapes used as tools	46
Figure 5.1	Function script for the basic shape and one tool-shape	59
Figure 5.2	The incremental shape-modelling pipeline	60
Figure 5.3	2D diagrams of the basic cases of clustering.....	62
Figure 5.4	Traversal of the cluster hierarchy	65
Figure 5.5	Algorithm flow-chart diagram.....	66
Figure 5.6	Different results while tool-shapes applied with different order.....	68
Figure 6.1	An example of the created tree clusters the tools	73
Figure 6.2	Data structures of object and tree node	75

Figure 6.3	Integrate the acceleration module to the system.....	77
Figure 7.1	Illustration of the tool-shape distributions (total number of tools in each test case was 50)	80
Figure 7.2	Experiment statistics for Figure 7.1	81
Figure 7.3	(a) Shape-tools with mutual self-intersection in the middle. (b) The respective tool bounding boxes	82
Figure 7.4	Function-based models used in testing (a) Plum with 89 tool-shapes (b) Orchid with 33 tool-shapes (c) Bamboo with 56 tool-shapes (d) Chrysanthemum with 82 tool-shapes	83
Figure 7.5	Experiment statistics for Figure 7.4.....	84
Figure 7.6	Free form shapes modelled in VRML and the respective tool bounding boxes.	85
Figure 7.7	(a) Simple 2D example of octree (b) The respective function script.....	88
Figure 8.1	Editing geometry of the initial shape.....	92
Figure 8.2	Modifying the initial shape with other shapes (tools)	94
Figure 8.3	Modifying shapes used as tools	94
Figure 8.4	Definition of the 3D colour space coinciding with the geometry (a) by interpolating colours using an explicit distance function (b) by defining three independent colour functions for red, green and blue colours. The shape was also truncated to display its inner colours.....	95
Figure 8.5	Modifying shapes used as tools	97
Figure 8.6	Accelerated function scripts	98
Figure 8.7	Examples of student's work	99

Figure 8.8	Parametrically defined animated (time-dependent) shapes interactively painted by implicitly defined 3D brush tools.....	100
Figure 8.9	Shapes defined by analytical functions are placed into an immersive shared VRML environment.....	101
Figure 9.1	Isosurface searching in POV-Ray	103
Figure 9.2	Process of rendering function-based model uses POV-Ray.....	104
Figure 9.3	Nested balls rendered with POV-Ray.....	105
Figure 9.4	Virtual wood-carving rendered with POV-Ray: Mosco nights	106
Figure 9.5	Virtual wood-carving rendered with POV-Ray: Sunset.....	106
Figure 9.6	Function scripts of Virtual wood-carving shown in Figure 9.4.....	107

Chapter 1 Introduction

This chapter gives a brief introduction to the thesis. In Section 1.1, the background of this project is presented, and the motivation for research is discussed. In Section 1.2, we define the objectives of the project. Section 1.3 gives a brief introduction of the proposed approach. Finally, in Section 1.4, the organization of the thesis is outlined.

1.1 Background & Motivation

The work presented in this thesis was initiated as a multidisciplinary project (computer science and pedagogy) targeting electronic education of math-related subjects in shared virtual spaces.

When teaching subjects richly infused with mathematics and geometry, there is a frequent problem that the learners are not able to “visualize” the theoretical concepts. It is important, therefore, to constantly illustrate the associated theories with practical visual exercises, which are preferably to be done in collaboration with other learners to allow them to discuss possible approaches to the problem and to consult the instructor, virtually or face-to-face. In this work, we aim at proposing an approach that would allow training learners in mathematical and geometry rich subjects, while they are immersed within virtual 3D environment as well as in 3D collaborative environments. We also target educators who are teaching at the secondary or tertiary level, or doing research on these topics.

The existing methods for cyber learning mathematics and geometry in virtual spaces are restricted to limited, mostly predefined types of geometric objects. In addition, no

existing methods have emphasized the geometric meaning of mathematical functions. However, we believe that investigating the correlation between abstract mathematical functions and intuitive properties of a shape in 3D space (e.g., visual and haptic) during learning will benefit the learner's spatial intelligence. Therefore, a novel method, allowing users to intuitively explore the correlation between abstract mathematical functions and visible/tangible properties of a virtual shape, is worthy of research.

Data transmission is one key issue to achieve high quality visual results, therefore, choosing a suitable method with a proper model will boost the whole learning experience while shared virtual spaces are used for education. There are basically three ways in which visualization can be performed in 3D shared virtual spaces: by image transmission, by model sharing, and by event transmission. Actual visualization can be a mixture of these three, but in this research we only consider model transmission. This is where the model of the shape has to be shared between the client computers as well as visualized interactively. It imposes serious restriction on the model, which has to be (1) small in size to exchange easily across the network, (2) computationally simple to provide for interactive updates (i.e. to require less than 3 seconds rendering time), and (3) able to support as high level of detail as possible.

Although model transmission is used in VRML [1] and its successor X3D [2], these are polygon-based tools and therefore interactive network exchange is limited by the size of the polygon meshes. One solution here can be to exchange only the updated parts of the models, but it may become problematic if the rendering pipeline does not support such an incremental update mode. Another restriction is that polygon meshes require third party software to make them. Therefore, there is still a need to construct efficient data

models supporting any level of detail and networked collaboration.

1.2 Objectives

Since this is a multidisciplinary project targeting shape modeling issues with the application to electronic education of math-related subjects, both computer science objectives in computer graphics and educational (pedagogical) objectives should be achieved.

The major goal of the project is to find ways how to ensure interactive modeling of the shapes defined by implicit functions. When implicit functions are used for defining shapes' geometry, the function evaluation is slowing down by the growing complexity of the function definition. Specifically, the research objective is to devise a new algorithm that will support interactive incremental modeling of the shapes defined by mixture of analytical functions in explicit, implicit and parametric forms. The challenge of achieving this research novelty is in making the acceleration independent to the polygonization algorithm and generating accelerated implicit model on the definition level.

The educational objectives require us to illustrate the usefulness of the developed algorithm on the example of the actual teaching math-related subjects while mathematics is visualized for the students in the form of different geometric shapes, colors, and animations. Hence, we will propose the virtual learning environment to allow the learner to define geometry, appearance, and any other possible properties of geometric shapes with mathematical formulas in various forms. Specifically, a mixture of implicit, explicit and parametric functions should be allowed to define the properties of a virtual shape while interactively modifying it. The challenge of achieving the educational novelty is in

making the learning environment intuitive for the learner, and to provide for immersion into the modeling space with maximum interaction with the individual components of the shape.

1.3 Proposed approach

To achieve interactive rendering of the shapes of which geometry is defined by implicit functions while visual appearance can be a mixture of explicit, implicit, and parametric functions, we have proposed an algorithm to accelerate it. The algorithm is devised for free-form shape modelling in cyberworlds where some initial shape is gradually modified by other implicitly-defined shapes with relatively smaller sizes compared to the final shape. The acceleration method is based on a constructive approach by adding additional function scripts to the function definition scripts so that when the function definition is evaluated during the rendering process, non-contributing elements to the resulting shape at a given point can be successfully bypassed. To further accelerate the rendering, we also proposed a method based on cluster volume minimization. This approach helps us to achieve an accelerated function evaluation by omitting the functions that do not contribute to the currently sampled point. As a research challenge, we consider here the worst possible scenario where the whole shape has to be redrawn, i.e. after receiving it at the client computer or modified by the client. It also permits to use such accelerated function scripts as a rendering platform independent notation language model format that can be then visualized with any suitable software.

To achieve the educational novelty, we have to propose and develop a concept of modelling with an emphasis given to the modelling process itself with an independent

definition of shape geometry, appearance and potentially physical properties by various types of analytical functions. We have proposed an approach to mathematic visualization where the learner uses only functions and algebraic function scripts as well as interactive manipulations of the virtual objects to visualize mathematics and to define virtual objects during the learning process. Representation in a form of algebraic function scripts is based on using explicit, implicit, and parametric functions for defining shapes' geometry, texture, colour, and potentially physical properties. The concept of mathematics visualization in virtual environments will be detailed in Chapter 3.

1.4 Thesis organization

This thesis is organized as follows. A survey of the relevant approaches to electronic education when teaching mathematics and geometry is presented in Chapter 2. In Chapter 3, we introduce the project hypothesis, the proposed concept of mathematics visualization, and the requirements for research and development of this project. The design and implementation issues of the modelling software and the necessity of acceleration are also discussed in Chapter 3. A survey of the relevant methods for accelerating the rendering of implicitly defined shapes is given in Chapter 4. In Chapter 5, the proposed acceleration algorithm is described in detail. Implementation issues of the proposed acceleration method are illustrated in Chapter 6. In Chapter 7, we discuss the algorithm control and analyze the performance of the proposed algorithm. User testing of the developed software is illustrated in Chapter 8, while applications for rendering on different platforms are demonstrated in Chapter 9. Finally in Chapter 10, the conclusion is drawn, and the future work is outlined.

Chapter 2 Visualization of mathematics in cyberspace

In this Chapter, a survey of electronic education when teaching mathematics and geometry is given. In Section 2.1 a brief introduction of electronic education is presented. Then, the problems of electronic education when teaching mathematics and geometry rich subjects are discussed in Section 2.2. Examples of approaches to visualize mathematics are described in Section 2.3. Finally, in Section 2.4 the research niche is identified.

2.1 Electronic education

In book [5], the authors have classified the systems that use computer techniques to facilitate learning into the following categories:

- Tutorials (such as MathMedia Educational Software [6]).
- Drill and Practice (such as Project AutoMath [7]).
- Hypermedia (e.g. Learning Management Systems such as Moodle [8]).
- Simulation (e.g. Virtual Chemistry Lab presented in [9], Simulation for medical training presented in [10], and Marine Simulator for marine education in [11]).
- Educational Game (such as History of Biology [12] and Econ U [13] for economics education).
- Tools and Open-ended Learning Environment.
- Web-based Learning (e.g. Learning Management Systems and web-enabled virtual worlds [14]).
- Online Collaborative Environment (e.g. Whyville [15]).

This classification is based on different computer techniques as well as different

learning paradigms employed. However, these categories are not mutually exclusive: certain educational systems may belong to several types of them. For example, a Learning Management System could be classified into “Hypermedia”, “Web-based Learning”, and “Only Collaborative Environment”.

In recent years, various tools and systems have been proposed and developed for different educational purposes. Computational devices, including computers and mobile devices (such as iPad and iPhone), are used to supplement traditional classroom education. Electronic education based on using shared virtual spaces either as a recreation of a real world environment or as an imaginary world is attracting increasing research focus.

Shared virtual spaces in forms of virtual communities augment the way we interact, participate in business, and receive information throughout the world. These are social networks like Facebook [16], Twitter [17], and Myspace [18], as well as virtual worlds such as Second-Life [19] and OpenSimulator [20]. There are also multiplayer games (leisure games such as EverQuest [21] and World of Warcraft [22], as well as serious games such as EteRNA [23] and America’s Army [24]).

By taking advantages of easy information sharing, immersion, collaboration, and being capable of realistic simulation of a real world environment, such shared virtual spaces have attracted an increasing research focus on their applications for educational purposes. In recent years, 3D shared virtual spaces have been broadly investigated and employed by a wide range of educational applications. To name a few of them: Security Education using Second Life [25], CLEV-R for collaborative e-learning [26], virtual campuses such as the Virtual Design Studio of the University of Sydney [27-29], SciCentr and BioLearn [30] in AWEDU - The Active Worlds Educational Universe [31].

It has been universally accepted that educational applications could benefit from many features that shared virtual spaces are able to provide.

Therefore, we choose to propose our own approach suitable for application in shared virtual spaces. We believe that electronic education residing in shared virtual spaces is more valuable for education purposes than other kinds of electronic education systems.

2.2 Problems of electronic education when teaching mathematics

Mathematics and geometry intensive subjects are the kind of subjects that are traditionally difficult to be taught in the real learning environment and hence can benefit from computer created world. These subjects on one hand are very abstract and difficult to understand, on the other hand, they are very intuitive if the learners have an ability to visualize them. Normally, in the traditional classroom settings, instructors explain abstract concepts by providing two-dimensional visual hand-drawings or real three-dimensional object models, making the problem less abstract. The limitations of the traditional classroom settings are that hand-drawings are always time consuming and real three-dimensional object models are only limited to a few types, not always available, and not interactive (the real objects could not be dynamically changed after created). This can certainly be improved with the help of computer techniques. However, visual components required for these subjects naturally are not provided by existing tools. A novel system for electronic education, which is dedicated to the subjects richly infused with mathematics and geometry, is needed.

Subjects such as computer graphics and shape modelling, always require the

learner's high spatial intelligence. Spatial intelligence, which can be measured as spatial abilities, refers to the ability to identify, manipulate and transform three-dimensional objects. Research on cognitive abilities and psychologies also suggest that spatial ability, which is also known as spatial visualization skill, plays a critical role in engineering and scientific disciplines [32].

However, although it is emphasized as an intellectual competence that the learners gain from mathematics and geometry study in Singapore's primary and secondary mathematics syllabuses [33], spatial intelligence of the learners is not well trained in the current school mathematics education. The students entering universities always have very different backgrounds in geometry and possess different spatial intelligence. Most of the students find it difficult to deal with mathematics problems, especially those that require geometric reasoning and spatial abilities. This creates problems in teaching subjects with mathematics and geometry in tertiary courses.

The main problems for the learners include:

- Mental visualization of three-dimensional objects is difficult.
- Mental manipulation, like movement and rotation three-dimensional objects, is neither natural nor easy.
- It is difficult to develop mental mapping of mathematical concepts with three-dimensional objects.
- The learners lack the visual vocabulary pertaining to spatial geometry.
- There is a lack of attention paid to verbal processes involved in learning spatial geometry.
- The learners are unable to make accurate and time-consuming drawings of

spatial objects.

Another challenge is that, mathematics and geometry are thought to be boring and difficult to learn; therefore, the learners psychologically refuse to get deep into the world of mathematics.

We hypothesize that these problems can be solved while teaching in a computer generated mathematical world. Based on this hypothesis, we seek to find new ways for teaching mathematics and geometry in virtual spaces. To support and motivate our approach, we have surveyed the existing methods of electronic education when teaching mathematics related subjects. The survey is given in the following section.

2.3 Approaches to electronic education when teaching mathematics

The traditional method of understanding 2D and 3D geometric objects is by hand-drawing them, which is time consuming and requires developed drawing skills from the learners. A group of standalone interactive desktop tools, also known as dynamic geometry tools, emerged in 1990s. The tools display images of geometric objects that are defined and controlled by their parameters. The ability to explore geometry of objects by changing definitions and their parameters or by interactive moving individual defining elements encourages experimentation and enables these applications to comply better with pedagogical theories such as constructivism [34] which argues that humans generate knowledge and meaning from their experiences. We have come across of more than forty such applications. The most popular are Geometer's Sketchpad [35], Cabri Geometre and Cabri 3D [36], Cinderella [37], Euklid DynaGeo [38], Archimedes Geo3D [39], Euler 3D

[40], Geometria [41], Geomview [42], and to certain extent Mathematica [43], and Mathcad [44].

Interactive desktop geometry tools are suitable for explaining geometric concepts in common classes since they only display pictures of geometric objects without immersing the learners into the educational environment. The immersion accompanied by hands-on experience is required by the constructivism theory. It can be provided by using virtual reality systems as well as shared virtual reality environments.

Hence, in [45] a virtual reality learning environment, which also uses “dynamic geometry” paradigm for 3D geometry education, is presented. The highlight of this system is that it uses a haptic device called Cybertouch [46] to provide a tangible picture of complex three-dimensional objects and scenes. It also provides a stereoscopic view of the virtual learning environment. The proposed system contains three main components: GCA (Geometry Construction Agent) for the user to construct and modify geometric objects in the scene, HIA (Haptic Interaction Agent) that deals with the haptic feedback sent to the user and communication of data to the GCA, and a Collision Detection algorithm which calculates the collisions between the virtual hand and the dynamical geometric objects. The main advantage of this application is that it provides tangible picture of complex 3D objects and allows for some level of physical manipulation.

VRMath [14] is an online Virtual Reality Learning Environment that enables learning by making and designing personally meaningful artefacts. Different from “dynamic geometry” paradigm, VRMath employs rich semiotic resources that enable multiple perspectives and representations for mathematical meaning-making and uses a Logo-like programming language to construct 3D virtual worlds. The construction of 3D

geometry can either be done by mouse drawing on the display screen or by scripting a Logo-like programming language. After construction, the users can interactively manipulate the objects by using the mouse. The system also uses hypermedia and forum interface allowing for rich information and related concepts sharing and discussion for self-paced learning; however, the constructed 3D virtual world is not shared.

Figure 2.1 A virtual learning environment augmented with haptic device for 3D geometry education (from [45])

Figure 2.2 Snapshot of the web-enabled VRMath system (from [14])

In [47], a sketching user interface for teaching geometry, and a VR class integrated

with the corresponding adaptation support are proposed. By creating sketch-based interface and providing adaption mechanism that integrates a wide range of context related to user, environment and system, the user can enter information more easily while keeping a natural style of interaction and leveraging the power of computation.

Figure 2.3 A System uses CAVE virtual environment (from [48])

One more example of a system for learning abstract math concepts implemented in

CAVE (Cave Automatic Virtual Environment) virtual environment is presented in [48]. Learners in CAVE can be totally immersed into the virtual environment. However, the drawback of CAVE is its high cost and location restriction.

Besides pure virtual reality, mixture of real and virtual shapes can be used for education in the subject of geometry. The example of using augmented reality is Construct3D [49] that is a three-dimensional “dynamic geometry” construction tool specifically designed for mathematics and geometry education. It is based on a mobile collaborative augmented reality system "Studierstube" [50]. Construct3D uses a collaborative augmented reality setup, based on see-through head-mounted displays which allow a combination of virtual and real world. The main advantage of Construct3D is that its face-to-face setting of teachers and students allows for applying traditional pedagogical theories and face-to-face communication. In addition, 3D dynamic geometry construction allows for exploration of dynamic behaviour during construction of an object just as traditional desktop “dynamic geometry” tools do. However, the drawback of Construct3D is that it does not support distributed learning and moreover, only provides visual representation of geometry objects without tactile interaction.

Shared virtual reality allows the instructor to place in one educational space the learners who are physically located in different places. The participants of the shared session can be visually represented by their avatars and work on the same assignment constructing the same object. CyberMath [51, 52] is an extendable avatar-based shared virtual environment for teaching and exploring mathematics. It allows for guided collaborative learning as well as for self-learning. Besides, CyberMath allows the learners to interact verbally with each other using computer microphone. CyberMath is built like a

museum which has two types of halls: a virtual lecture hall in its centre and several surrounding exhibition halls. CyberMath provides for both overall theoretical learning in the virtual lecture hall (slide presentations) and hands-on practices in the exhibition halls. The original implementation of CyberMath was based on DIVE toolkit for building interactive shared distributed multi-user virtual environments. Due to drawbacks of DIVE such as limited visual quality, fixed feature set, and low execution speed, CyberMath was also made available as a desktop VR application as well as the six-walled CAVE virtual environment application.

Recent emergence of mobile devices with touch screens motivated development of new tools for teaching geometry. The notable example is SpaceTime [53] which is a cross-platform versatile mathematics tool. It has a set of sub-programs designed for different purposes and for different branches of mathematics, ranging from 2D geometry to high dimensional visualization, from algebra, calculus to fractals, and other areas. The software has been designed for different hardware and software platforms, including Windows Mobile devices, Apple iPhones and iPad devices as well as Microsoft Windows computers and Apple Mac computers. The software provides a significant number of functionalities including plotting 2D, 3D and 4D functions, and rendering parametric, polar and vector fields, as well as implicit contours and fractals.

Figure 2.4 Augmented reality tool - Construct3D (from [49])

Figure 2.5 Shared virtual space application CyberMath (from <http://www.nada.kth.se/~gustavt/cybermath/>)

Figure 2.6 Snapshots of software from SpaceTime (from the website <http://www.spacetime.us/>)

Image Gallery

Interactive Software: Surfer

Videos & Films

Surfaces as Sculptures

Figure 2.7 IMAGINARY exhibitions (from <http://www.imaginary2008.de/>)

Another recent notable development is a series of exhibitions named “Imaginary”

[54], which was organized in 2008-2011 by the German Mathematisches Forschungsinstitut Oberwolfach and the Royal Spanish Society of Mathematics. The exhibition is a collection of works from different research groups promoting geometry education. The exhibition consists of four main parts: image gallery, interactive software, films and 3D models. The image gallery parts showcase images of beautiful algebraic surfaces made by mathematicians and artists with text explaining their mathematical properties, how the image was generated and some important elements of the images. The interactive parts are presented by interactive software developed by different research groups and companies, specifically, Surfer [55], jReality [56], 3D-XplorMath [57], Morenaments [58], and previously mentioned Cinderella. Surfer is a program designed to calculate, indicate and change algebraic surfaces in real time. The users can enter and change polynomial equations on a large touch-screen and observe the resulting surface from different perspectives. jReality is a java library for visualization and sonification with application to mathematical and scientific content. 3D-XplorMath is a mathematical visualization program that visualizes different categories of interesting mathematical objects, ranging from planar and space curves to polyhedra and surfaces to ordinary and partial differential equations, and fractals. It also allows the user to further explore the mathematical objects by changing the parameters and viewing options. Morenaments is an application for drawing beautiful ornaments in the symmetry groups of the Euclidean plane.

By analyzing existing approaches to teaching geometry-based subjects with computers and in cyberspace in particular, one can notice that they emphasize the importance of visual perception and visual imagery, and concentrate on providing visual

representation of geometrical objects. However, some research works [59] suggest that spatial representations created in our conscious are the combination of both visual and linguistic perceptions. In addition, visual manipulation cannot totally replace physical manipulation. Gardner's multiple intelligence theory [60] and multi-sensory instructions also suggest that a good learning environment should provide learners the opportunity to receive multiple sensory learning information. Also, most of the existing computer-based geometry teaching methods are restricted to a limited class of predefined geometric objects. Besides, no existing method emphasizes the geometric meaning of mathematic functions.

We can therefore conclude from the state of the art of the methods and tools for teaching geometry-rich subjects with computers and in cyberspace in particular that this area is quite open for further research and development.

2.4 Summary

In this Chapter, the problems of electronic teaching mathematics have been discussed. A survey of existing methods of electronic education when teaching mathematics and geometry is also presented.

Existing electronic education systems provide learning content and guide the learners through the learning process. For better involvement of the learners, elements of virtual reality or complete immersion into virtual reality based environments can be used. Usually, the virtual replica of real classes and simulation of the real world phenomena are used as a virtual teaching stage. There are subjects, however, that are difficult to teach in real classes and thus they require additional teaching aids that only computer generated

virtual learning environment can provide. Geometry and other related subjects are examples of such applications where computers and cyberspace can help to create principally new teaching environments that do not have an equivalent in the real world.

Existing approaches to teaching geometry rich subjects with computers and in cyberspace emphasize the importance of visual perception and visual imagery, and concentrate on providing visual representation of geometric objects. Moreover, they are restricted to a limited class of predefined geometric objects. In addition, no existing method emphasizes the geometric meaning of mathematic functions.

Based on the survey presented in this Chapter, we can conclude that teaching subjects richly infused with mathematics and geometry by using computers and in cyberspace in particular is quite open for further research and development.

Chapter 3 Mathematics Visualization

In this Chapter, the principles of *mathematics visualization in virtual environments* are discussed. The hypothesis of this part of the project is given in Section 3.1. In Section 3.2, the concept of the proposed mathematics visualization is detailed. In Section 3.3, we outline the research and development objectives. Design and implementation issues of the shape modelling software are discussed in Section 3.4. In Section 3.5, we discuss the advantages of the proposed approach as well as the problem of the initial version of the shape modelling software. A discussion of the necessity of acceleration of the rendering to support the proposed approach then follows. Finally, Section 3.6 summarizes the whole chapter.

3.1 Hypothesis

Based on what have been reviewed in Chapter 2 about electronic education for teaching mathematics and geometry rich subjects, we *hypothesize* that a new approach to visualization and modelling of the shapes is required to develop geometric reasoning, spatial visualization skills and an ability to visualize mathematical definitions. The idea of the approach is that complex shapes can be constructed gradually by using individual components defined by analytical formulas in explicit, implicit and parametric forms, whichever is more suitable. Geometry, visual appearance, and potentially any other additional shape properties like physical tangible properties will be modelled in their own coordinate domains and eventually assembled into the shape. At each stage of modelling, only mathematical definitions and immersive interaction with the shape will be used.

Hence individual formulas are immediately rendered as well as any part of the shape being modelled can be seen as its defining function.

Eventually, both educational and research novelties will be achieved in this work. The educational novelty is that we psychologically immerse the learner into the world of mathematical definitions, where any shape created with mathematical functions can be visualized and explored. The research novelty in computer graphics and shape modelling is that we will create virtual shapes which are defined as function scripts or procedures using a mixture of explicit, implicit and parametric functions, and this final defining script or procedure will be independent of the visualization platform, thus it can be interactively rendered on any suitable graphics system.

3.2 The proposed concept

The main idea of the concept is to visualize mathematics in various ways while performing free-form shape modelling in a virtual space with individual components of the shape defined by analytical functions and scripts. The concept is detailed as follows:

1. Virtual shapes are defined by incremental construction of their components followed by assembling them into a final shape.
2. The components of the virtual shapes may include and are not limited to geometry, visual appearance, geometric textures, physical properties, sonic properties, etc.
3. The final shape can be rendered as a mixture of various stimuli: visual, haptic, sonic, etc. however only visual stimuli are considered in this thesis.
4. The virtual shape components are defined by analytical functions and function scripts consisting of explicit, implicit and parametric function definitions.

5. The final virtual shape definition is stored as a function script that can be further edited or rendered at any suitable rendering system with any desired precision.
6. At the level of the user interface, the shape defining functions are separated from the shape transformations as well as other parameters to highlight the meaning of the defining functions.
7. Each individual shape component can be retrieved as its function definition, edited, removed or added to the shape being constructed at any time of the construction process as well as after the shape is defined.
8. When constructing the shape, additional construction can be added to accelerate function evaluation that will eventually expedite the shape rendering. These constructions will eventually appear in the final function definition of the shape but will be transparent to the user during the construction process.
9. The user is immersed into the construction scene and performs interaction with a virtual shape in an intuitive manner without knowledge of any common programming languages or a specific command language.
10. Collaboration with other users can be provided when working on the same construction.

3.3 Research and development requirements

The following research has to be carried out:

1. To design a new interactive shape modelling tool based on the proposed concept.

This tool should be very intuitive for the user, and would provide for immersion into the modelling space with maximum interaction with the individual components of the shape.

2. To devise a new algorithm that will support interactive incremental modelling of the shapes defined by mixture of analytical functions in explicit, implicit and parametric forms. The algorithm should add to the function description of the shape additional accelerating components that will eventually constitute the final analytical definition of the shape.
3. To propose a new unified visualization algorithm for rendering parametrically defined curves, surfaces and solid objects hence allowing for using parametric and implicit definitions interchangeably.

The proposed tool should not just let the learners see a two dimensional images on the screen, but will immerse the learners into the mathematically defined virtual world, where the virtual objects, defined with mathematical functions, are three-dimensional, visual, tangible and even audible (tangible and audible parts are beyond the scope of this thesis and remain as future work). It also should permit deeper understanding of the virtual objects by allowing the learner to go inside and walk along the surface of the virtual objects.

The proposed tool has to be validated for various purposes ranging from teaching geometry, computer graphics and shape modelling to solving research problems in various application areas where mathematical definitions of shapes can be efficiently used.

3.4 Design and implementation of the modelling software

The educational objectives require us to illustrate the usefulness of the proposed approach and tool on the example of the actual teaching math-related subjects. Based on the survey

given in Chapter 2, developing virtual learning environment that would support the concept of virtual reality and potentially shared virtual reality is the current trends in this area. Thus, we select to implement the mathematics visualization tool using Virtual Reality Modelling Language (VRML) [12]. Although, a desktop virtual reality supported learning environment does not allow physical immersion, it still allows visual immersion [61] (visually immerse the learner into the world of mathematical definitions), which is consistent with the proposed concept.

3.4.1 Design principles

The proposed concept assumes that the learner will learn mathematics and geometry related knowledge by doing. It assumes that the learners would gain knowledge when they are actively creating shape models. It requires an interactive virtual learning environment that allows the learner to create complex shapes using only mathematical functions and interactive manipulations. The interactive shape modelling software uses a concept of incremental modelling where the initial basic shape can be subsequently modified by other tool-shapes with concurrent interactive visualization of the results.

Several features have been considered while designing the interactive shape modelling tool:

1. Firstly, it should not require any programming knowledge from the user. To support this, the software is designed so that the learner is able to create complex shapes by inputting mathematical functions and by intuitive interactive manipulations in 3D space.
2. Secondly, the learner has to be immersed into the visual representation of

mathematically defined objects to allow for maximum interaction with the created shape model. These objects must be available for interactive editing by various ways including using common mouse as well as advanced interaction tools such as 3-6 DOF haptic devices and new user interfaces based on depth sensing cameras like MS Kinect. Due to availability and ease of implementation, we support common mouse in the current version of software; however, advanced interaction tools (haptic) are already considered in the future work as well as in PhD project of another student. In addition, we separate each property of a shape to be defined in their own coordinate systems, while the system will then merge them together to define a shape.

3. Thirdly, the learner should have the ability of interactive editing the geometry and appearance of the shapes, and have an immediate access to mathematical definitions of all the parts contributing to the object. This should be implemented as electronic forms containing mathematical formulas, transformation parameters (matrices) and various rendering attributes. Electronic forms are chosen because they have several advantages in managing mathematical information. They are clear, accurate, and effective.

3.4.2 Implementation issues

To implement the above principles with using VRML and HTML web pages, the available data flows have to be used. Specifically, the VRML/X3D 3D modelling scene part should deal with the interactive shape modelling (interactive manipulations and modifications) and visualization issues. The data that VRML scene part should maintain include: the defining functions of the basic shape and each tool shapes, the transformation

parameters of the whole shape, as well as each tool shapes (which is derived from the interaction between the learner and VRML/X3D modelling scene), and other rendering attributes. The HTML/JS scripts should respond to the interactive editors, with which the user is able to explore and edit the function definitions of the basic shape, as well as each tool-shape applied, the operation used, the transformation parameters locating each tool-shape in the 3D space, as well as the rendering parameters used for visualization of the whole shape. Any interactive modification to the VRML/X3D scene should be synchronized to the HTML/JS based electronic forms and displayed immediately, and vice versa. This requires that there should be some mechanism that would support refreshing VRML/X3D scene from some trigger on HTML/JS files side, and vice versa. Figure 3.1 shows the system architecture design of the web-enabled incremental interactive shape modelling tool.

The current implementation uses the function-based extension FVRML/FX3D [62] [63] of the Virtual Reality Modelling Language and Extensible 3D, and Bitmanagement BS Contact VRML/X3D viewer [64] that is a plug-in for MS Internet Explorer. We choose this platform for implementation because it provides various flexibilities to create a shared virtual space based application. It supports 3D immersive rendering and interaction, allows us to use stereoscopic view (VRML supports stereo-visualization), allows learners to have an avatar, allows the avatar to walk through the virtual objects, allows the user to change the size of the virtual objects, allows us to use any level of detail of the virtual objects, allows us to write extensions which are defined as DLL libraries and are called from the VRML engine when the respective functions are requested (when advanced interaction methods are required, it also allows us to use

geometric collision detection or to use the previously proposed and developed extension [63] with function-based collision detection). All these features make possible the implementation of a shared virtual learning environment that could better immerse the learner into the world of mathematical definitions. VRML rather than its successor X3D was selected due to some technical issues with the licenses available.

Figure 3.1 System architecture

The following describes how we use this platform to realize the synchronization between VRML scene and HTML/JS scripts part.

Synchronization means that any change on one side must be reflected on another side. Firstly, it requires that when any change is done on the VRML side, the respective field on HTML side must also be changed. This is done by calling a function on HTML/JS scripts from internal VRMLScript. It is implemented with a function provided

by Bitmanagement BS Contact VRML/X3D viewer. On the VRML side, the syntax is as follows:

```
Browser.loadURL( OnEvent( string par1, string par2, string timestamp))
```

and on the HTML/JS scripts side there must be the respective function to respond, which looks like this:

```
function OnEvent( par1, par2, time){ ... }
```

Secondly, it requires that when any change is done on the HTML/JS side, the VRML scene will be refreshed with the respective changes. The following functions are provided by Bitmanagement BS Contact VRML/X3D viewer to do this:

```
bsContact.getNodeField( string nodeName, string fieldName)
```

```
bsContact.setNodeField( string nodeName, string fieldname, string value)
```

```
bsContact.setNodeEventIn( string nodeName, string eventInName, string value)
```

There are two ways to refresh the VRML scene:

1. To modify the field of one node directly.
2. To use ‘eventsOuts’. When the data which is used as ‘eventsOuts’ in VRMLScript is modified, it will trigger a certain event.

When the learner modifies certain fields in the HTML electronic forms, we either use ‘*bsContact.setNodeField*’ to change certain field value in the VRML scene directly or use ‘*bsContact.setNodeField*’ and ‘*bsContact.setNodeEventIn*’ together to change the value of certain ‘eventOut’ node and then trigger some event to happen.

3.4.3 Introduction to FVRML/FX3D

Here, we give a brief introduction of the function-based extension FVRML/FX3D of the

Virtual Reality Modelling Language and Extensible 3D, which were proposed and implemented previously [62][63].

In the VRML/X3D extension there are two new principal nodes—*FShape* and *FTransform*—that have been added to both VRML and X3D. The *FShape* node contains 10 more nodes in the same way as the standard X3D/VRML Shape node is organized. These nodes are *FGeometry*, *FAppearance* (with *FMaterial* and *FTexture3D*) and *FPhysics* which contains *FSurface* (with *FFriction*, *FTension*), *FDensity*, and *FForce*. These nodes can be used alone as well as together with the standard X3D and VRML nodes (See Figure 3.2).

The *FShape* node is a container for the *FGeometry* or any standard X3D/VRML geometry node, and the *FAppearance* or the standard *Appearance* node. These nodes define the geometry and the appearance of the shape, respectively. The *FGeometry* node is designed to define the geometry using functions typed straight in the code as individual formulas and java-style function scripts or stored as algorithmic procedures in DLL libraries.

The *FAppearance* node may contain the *FMaterial* or the standard X3D/VRML *Material* nodes, as well as the standard colour *Texture* and the function-based *FTexture3D* nodes. In the *FMaterial* node, the components of the illumination model are defined using functions in a way how it can be done in the *FGeometry* node, i.e. in analytical formulas, scripts or DLLs. The *FTexture3D* node contains a displacement function for the geometry defined in the *FGeometry* node.

For defining geometry and appearance, three types of function definitions can be used concurrently. They are implicit, explicit and parametric definitions.

Implicit functions are $f(x,y,z,t)=0$, where x , y , z are Cartesian coordinates and t is the time. The function equals to zero for the points located on the surface of a shape.

Explicit functions are $g=f(x,y,z,t)$. The function value g evaluated at any point of the 3D space can be used either as an argument of other functions (e.g. for defining colours or parameters of transformations) or as an indicator of the sampled point location. Thus, explicit functions can be used for defining bounded solid objects. In this case, the function equals to zero for the points located on the surface of a shape. Positive values indicate points inside the shape and negative values are for the points outside the shape.

Figure 3.2 Extended function-defined nodes of VRML and X3D

Parametric functions define curves, surfaces and solid objects as: $x=f_1(u,v,w,t)$;
 $y=f_2(u,v,w,t)$; $z=f_3(u,v,w,t)$ where x , y , z are Cartesian coordinates of the points,

while u , v and w are parametric coordinates, and t is the time. Parametric functions can also define r , g , b values of colour. The parameters u , v , w are then Cartesian coordinates of the modelling space.

The *FShape* node may be called from the *FTransform* node or from the standard *Transform* node. The *FTransform* node defines affine transformations which can be functions of time, as well as any other operations over the function-defined shapes. These are predefined set-theoretic (Boolean) intersection, union, difference, as well as any other function-defined set-theoretic and any other operations, e.g. r -functions, shape morphing, etc.

Based on the nodes provided by the function-based extension, the interactive modelling tool can be extended in a modular fashion. For example, if haptic interaction is required for better immersion, we add definition of physics in the same way as it is done with definition of the shape appearance. Stereo viewing is natively provided by the BS Contact VRML/X3D viewer that is used by the tool. Once time variable t is used in the function definition, time-dependent geometry, appearance and other properties will become available.

3.4.4 User interface examples

The web-enabled shape modelling tool implementation would allow the learner to interactively manipulate each component of the shape as well as the defining scripts, and the transformation parameters can be edited mathematically with input electronic forms (as HTML web pages). A menu bar is provided on the left side of the workspace (See Figure 3.3). The learner can switch between different manipulation types by clicking at

the respective menu icons. After the manipulation type is specified, the learner can interactively manipulate the respective property, e.g., appearance, as shown in Figure 3.3, by dragging the virtual shape in the middle window. A coordinate system for locating the coordinate domain, where the property being manipulated is defined, will be automatically toggled on. Thus the property coordinate system can be interactively manipulated with reference to the global coordinates. On the right side of the workspace, there is an input electronic form where the defining function, transformation parameters and other rendering parameters can be edited. Any interactive manipulation that can be done to the shape will be immediately reflected by the respective changes in the forms.

Figure 3.3 Interactively manipulating appearance of the initial shape

The implemented shape modelling tool also allows the learner to explore the mathematical definitions of each tool-shapes applied to the modeled shape by clicking at

any interested part. Figure 3.4 shows an example of a watermelon shape that is gradually modified by six tool-shapes which apply “watermelon seeds” to the shape. The learner can explore the mathematical definitions of these “seeds” by first selecting the “ShownHandle” button in the left menu bar, then by clicking at any of these "seeds" to see their defining functions, operations as well as transformation matrices which identify their location in the global coordinate system. An electronic form editor containing all this information will pop-up immediately after the respective tool-shape is clicked. Then, any changes done within the electronic form editor will be reflected immediately in the VRML/X3D scene.

Figure 3.4 Exploring shapes used as tools

3.5 Necessity of acceleration

While being very flexible and convenient for modelling complex free-form shapes, the proposed approach requires us to find ways to accelerate the evaluation of the shape’s

defining function that grows with the growing number of contributing functions. As a result, after a certain number of interactive operations the rendering process slows down below interactive rates (at most 3 sec for updating the image on the screen). The problem of accelerating such incremental shape modelling was previously addressed in [66], however, the methods proposed there were based on a special polygonization algorithm that was redrawing only the areas of the shape affected by the current interactive operation. In the current project, we have a very challenging requirement to *always re-render the whole model* rather than its part. This is due to the way the function-based definitions can be plugged into VRML/X3D pipeline that requires from the plugin to supply a polygon mesh of the whole shape rather than its part [62]. Also, this is a requirement of potential collaborative applications where the updated model has to be exchanged across the network and rendered at client computers. Therefore, we need to propose a method to accelerate rendering of the function-defined shapes so that the function evaluation is accelerated as much as possible to sustain interactivity. Also, the accelerating methods have to be implemented at the level of function definitions that can augment the final function script and thus make a new accelerated script that can be rendered on any suitable rendering platform.

3.6 Summary

In this Chapter, the hypothesis of the project is presented and the proposed concept of mathematics visualization in virtual environments is detailed. The research and development objectives are outlined. Design and implementation issues of the shape

modelling software are also discussed. The advantages of the proposed approach as well as the problem of the initial version of the shape modelling software are elicited. One challenging requirement of this project is that *the whole model* is always need to *re-rendered*. Therefore, it is necessary to accelerate the rendering of the function-defined shapes to sustain interactivity.

Chapter 4 Existing approaches to acceleration

In this Chapter, a survey of the relevant acceleration approaches for rendering of implicitly defined shapes is given. In Section 4.1, common rendering method for implicitly defined objects is surveyed and introduced. Section 4.2 discusses relevant accelerating approaches for implicitly defined shapes. The restrictions to this project and requirements for the acceleration method are explained in Section 4.3.

4.1 Rendering implicitly defined objects

4.1.1 Implicit surface representation

In computer graphics, an *implicit surface* is an isosurface of some scalar field in the three dimensional space. It is represented as a three variants function over the R^3 domain:

$$f: \mathbf{R}^3 \rightarrow \mathbf{R}$$

such that, for an isovalue v , the *implicit surface*, is defined where

$$f(x, y, z) = v, \text{ or}$$

$$f(x, y, z) - v = 0$$

Representations of implicit surface include analytical functions (e.g. *Superquadrics* [67], *Superformula* [68], etc.), basis function approach (Blobs [69] [70], radial basis function approach [71], moving least squares [72], etc.), and sampling grids.

Since the ultimate goal of the work presented in this thesis is to provide an electronic learning environment for mathematics and geometry rich subjects, thus, for implicitly

defined shapes we chose to use analytical functions and procedures to represent shape models (the proposed method and platform chosen support mixture of implicit, explicit and parametric functions to define a shape models). The representation of *implicit surface* in this thesis is originated from F-Rep representation [73], where *implicit surface* is the *boundary* of a *solid geometric object* which is represented by a function of three variables:

$$f(x, y, z) \geq 0$$

where the positive function values are for points inside the object, negative function values are for points outside the object, and the values equal to zero are for points on the surface.

Based on this representation, *implicitly defined shapes* in this thesis are as it is introduced in [62, 74], represented with implicit functions:

$$f(x, y, z, t) = 0$$

and procedures. Here, x , y , z are Cartesian coordinates and t is the time. The function equals to zero for the points located on the surface of a shape. In the function-based extension of VRML used in this project, a C-like syntax is used for defining analytical formulas. Arithmetic operators '+', '-', '*', '/', round brackets, and '^' which stands for power, can be used. Set-theoretic intersection '&' and union '|' are supported with any desired continuity.

4.1.2 Visualization methods

Common visualization methods for implicitly defined surfaces include *triangulation* and *ray tracing*.

Triangulation is a process of extracting polygonal mesh representation from other representation. Using *triangulation*, an implicit function representation is first converted to a polygonal mesh representation and then visualized and displayed on the screen using methods for mesh models. The *Marching Cubes* [70, 75] algorithm is the first triangulation algorithms for implicit surfaces. It extracts a triangular approximation of the isosurface by evaluating the defining function for every cell of a regular grid. After that, to overcome the limitations of the original marching cubes algorithm, some variants of the marching cubes algorithms have been proposed. Notable ones include *Marching Tetrahedra* [76] , *Dual Contouring* [77] and *Dual Marching Cubes* [78, 79]. Although, these algorithms are unique and work differently, they have one property in common that is the implicit function is evaluated for each sampled point in the space. However, since the original marching cubes algorithm is simple and easy to implement, many applications for visualization implicit surfaces are done with the original marching cubes. Therefore, although adaptive triangulation methods which sample points in semi-regular or none-regular grids do exists, we have to consider the worst case in which the implicitly defined model created will be visualized with algorithms such as the original marching cubes algorithm.

When using *ray tracing* methods for visualization, images are directly generated with the implicit representation by sampling the implicit surface at the intersection point of a viewing ray and the surface. Paper [80] is the first one that presented applications ray tracing implicit surfaces. In [81], ray tracing algorithm that guarantees that no intersection is missed was proposed with the assumption that the scalar function is Lipschitz continuous. Paper [82] proposed a recursive root bracketing algorithm which is employed

in the POV-Ray software. Recently, some ray tracing algorithms improve the performance by implementing it with GPU [83-85]. These algorithms have one aspect in common: the implicit surface being rendered has to be evaluated many times during the root finding process of ray tracing. Therefore, we assume that the accelerated model represented as analytical functions and procedures independent of the rendering algorithm will also be capable of working with ray tracing algorithms.

4.1.3 Recent research on implicit surfaces

In the paper [106], the authors proposed an approach called Multi-level Partition of Unity Implicit Surface for constructing implicit surface models from very large sets of points. It uses adaptive octree to partition the modeling space into cubes. The adaptive division is based on a local max-norm approximation error that is estimated according to the *Taubin distance* [107]. Within each cube, the points are approximated by one implicit function. If no further subdivision is needed the local approximation function Q_i will be added to the model according to some weighting function W_i .

In paper [108], the authors proposed an approach for building interpolating or approximating implicit surfaces from polygonal data using a moving least squares (MLS) formulation with constraints integrated over the polygons. Different from the MLS described above, which is with value constraints at points, the MLS with constraints integrated over the polygons scatter an infinite number of points continuously across the surface of each polygon. Despite of value constraints with polygons, they also impose normal constraints by forcing the interpolating function to behave like a prescribed function.

Paper [109] proposed a surface reconstruction approach with Poisson formulation. The input of the approach is a set of samples; each sample consists of a point and an inward-facing normal, assumed to lie on or near the surface being reconstructed. A 3D indicator function (defined as 1 at points inside the model, and 0 at points outside) is computed with the input data, and then it is used to extracting isosurface.

4.2 Acceleration approaches

The problem of accelerating rendering of implicitly defined shapes has been investigated for many years. The existing acceleration methods relevant to rendering implicitly defined shapes either use special types of implicit functions or come up with special rendering algorithms.

The existing rendering algorithms to accelerate implicitly defined shapes can be classified into two classes by their use of additional information: *approximation methods*, which work without using any information about the way the functions are computed, and *constructive methods* which aim to speed-up the functions by optimizing the way they are constructed.

The *Approximation methods* augment the functions with special data structures and function parameters, such as sampled values, surface position, and control points, to evaluate the actual function value at an arbitrary point. In [86], a method that uses a uniform grid is described. It samples the defining function at regular intervals with consequent interpolation between the grid points. The drawback of this method is that it takes a lot of memory space even if each cell contains a small amount of data. To reduce the number of cells, it was suggested in [87] to keep track of the points near the surface

only. However, the uniform grid method cannot represent shape details smaller than the sampling interval. Thus, several adaptive methods were proposed in [88, 89], which use octree structures to locate the required points as fast as possible. However, despite the problem of space requirements, the grid-based methods also have a problem of inflexibility – they are mostly useful to store models, rather than to modify them. Another approximation method proposed in [90] uses trivariate polynomials over convex hulls, which encloses the shape surface into a set of convex hulls and performs interpolation inside each of them separately.

The *Constructive methods* explicitly use the function's internal structure. Some acceleration methods are developed for gradual modifications of defining functions during the modelling process. Knowing which part of the function is being updated, it is possible to avoid re-processing the entire shape, and to update only the modified part of it, as proposed in [91]. Other constructive acceleration methods work by simplifying or even bypassing elements of functions that do not contribute to the resulting shape at a given point. BlobTree structure, described in [92], consists of a set of components which affect only a certain part of the final function. Each component is a function itself. By forcing these functions to conform to Lipschitz condition $f(r_1) - f(r_0) < \alpha|r_1 - r_0|$ [81], which holds for any r_1 , and r_0 , it becomes possible to determine the contributing components efficiently. This condition guarantees that for any point r , such as $|r_0 - r| < R$, the function will not reach some threshold value. This property of so called L-G surfaces [81] was combined with an octree containing contributing elements in its nodes [92]. Paper [93] proposed a method to use a level-of-detail when evaluating the function. Different kinds of skeletal representations, e.g., subdivision curves, were used to adjust to the required

level-of-detail. The subdivision skeleton was able to change its complexity to match the current detail settings. In [94], a method associating every BlobTree node with the level-of-detail information was proposed. It dynamically removes parts of the BlobTree which are not important for the current viewer's location. Nested bounding volumes were also used to cull away parts of space with no contributing elements or where no piece of the surface may be found [92, 94].

All the above algorithms are proposed with special rendering algorithms and some only work with special types of implicit functions.

4.3 Objectives for the acceleration algorithm

One objective of acceleration is that the accelerated function model can be transmitted through the web, and then it can be rendered on local client computer with any required level-of-detail. Such representation will provide for both small size for transmission and fast rendering speed. It will also support for collaborative networked modelling.

We assume the worst case in which the whole function representation will be sampled regularly with, for example, the marching cubes algorithm that samples the points at each corner of the regular grids, or ray-tracing that casts rays towards the shape for each pixel of the image.

One straight-forward method would be to add an additional condition checking script, so that during the evaluation step every element will be checked to determine if it contributes to the current sampled shape. To efficiently determine whether it contributes to the sampled point, bounding volumes can be used instead of the exact shape boundary. Since to check such condition adds additional calculation, the straight-forward method

may not be effective.

One better approach is to organize these bounding volumes into hierarchies to reduce the number of times to check these conditions. In order to choose a suitable data structure for organization of the bounding volumes, we have surveyed the original R-tree [95] and its variants [96-98]; however, the special restrictions added to this data structure and its respective algorithms would result in the clustering that is not optimized for our project. Thus, we have proposed our own data structure and its respective updating algorithm for clustering spatial objects according to the space occupied by them as well as by their insertion orders.

The proposed acceleration method and the clustering algorithm will be described in detail in the next Chapter.

4.4 Summary

In this Chapter, a comprehensive survey of the relevant rendering algorithms and acceleration methods for visualization of implicitly defined shapes is given. Though, existing acceleration methods are respectful and work well in their own modelling systems, all the existing acceleration methods are proposed with special rendering algorithms and some only work with special types of implicit functions. Since our project seeks accelerated function models that can be transmitted through the web, and then be rendered on local client computer with any required level-of-detail, our own acceleration method is proposed that meets the special requirements of this project. The proposed model represented as mathematical functions and function scripts will be both small in size for easy transmission and be computationally economical for fast rendering.

Chapter 5 Proposed acceleration algorithm

In this Chapter, a novel approach to accelerate the rendering process of the shape model defined with implicit functions and procedures is proposed. In Section 5.1, the hypothesis of the proposed approach is presented. In Section 5.2, the algorithm is introduced in detail. Section 5.3 gives further discussion and improvements of the proposed method and Section 5.4 summarizes this chapter.

5.1 Hypothesis of the algorithm

Since all the rendering methods (based on mesh extract) for implicit representations are based on sampling while the defining implicit functions have to be evaluated at each sampled point, an accelerated model has to reduce the expense of the function evaluation.

We *hypothesize* that we could accelerate rendering based on the constructive approach by adding additional function scripts to the function definition scripts so that when the function definition is evaluated during the rendering process, non-contributing elements to the resulting shape at a given point can be successfully bypassed.

We assume that each interactive modification will only influence a relatively small part of the whole shape. We also assume that the area of each modification is known and could be defined by the user as a respective bounding box. These bounding boxes will form clusters to accelerate rendering even further. A special hierarchical data structure will be needed to handle the clusters.

The proposed approach will help us to achieve an accelerated function evaluation by omitting the functions that do not contribute to the currently sampled point. The resulting

functional model of the shape will still be defined as a function script but with the additional parts responsible for acceleration. This extended function-script model will be independent of the rendering system.

We assume the worst case, that is, the whole function representation will be sampled regularly with, for example, the marching cubes algorithm, which samples the points at each corner of the regular grids, or ray-tracing, which casts rays towards the shape for each pixel of the image.

The proposed acceleration method as well as the clustering algorithm is described in detail in the following section.

5.2 Proposed algorithm

To illustrate how the proposed acceleration algorithm works, let us consider how an incremental function-based shape modelling is performed. First, the basic shape has to be set by typing an implicit function defining its surface. Hence, an origin-centred solid box can be defined as an intersection of six plane-bounded solid spaces as follows:

$$f_b(x,y,z) = \min(x+1, 1-x, y+1, 1-y, z+1, 1-z) \geq 0$$

The software uses the following symbols to denote set-theoretical operations: ‘|’ for the union, ‘&’ for the intersection, and ‘\’ for the subtraction. These set-theoretical operations will be implemented either using “min” and “max” functions or so-called ‘*r*-functions’[99], depending on the user settings.

After the function of the basic shape is defined, it will be displayed with some rendering parameters, such as the sampling resolution and the bounding box of the shape.

Next, a tool-shape has to be defined. It is done in the same way by defining the

tool-shape's function. For example, an origin-centered solid sphere with radius 0.1 can be defined as:

$$f_1(x,y,z)=0.1^2-x^2-y^2-z^2\geq 0$$

The tool-shape can be then positioned using interactive scaling, rotations, and translation operations. These operations form three affine transformation matrices for the final scaling, rotation and translation. The matrices can be edited that will interactively reflect in the respective changes of the tool-shape's geometry and positioning.

Finally, the tool-shape has to be applied to the basic shape. This can be done by using one of the set-theoretic operations such as union, subtraction, or any operation defined analytically by the user. For example, the subtraction of the above tool-shape from the basic shape is defined as:

$$f(x,y,z)=\min(f_b(x,y,z),-f_1(x,y,z))\geq 0$$

Figure 5.1 Function script for the basic shape and one tool-shape

The result of such operation will be displayed as the modified shape. To do it, the shape defining function script has to be generated first that will include the basic function definitions as well as coordinate transformations and set-theoretic operations. For the

basic shape and one tool-shape somehow placed on the surface of the basic shape, it will look like it is illustrated in Figure 5.1.

Then, the script has to be sent to the rendering software which will perform polygonization of the shape followed by sending the polygon mesh to the VRML or X3D rendering pipeline.

Figure 5.2 The incremental shape-modelling pipeline

If no acceleration is used, the process of defining new tool-shapes $\mathbf{T}$ and their applications to the basic shape $\mathbf{S}$ will continue in the same way while the script of the resulting shape will grow with each new step of modelling by the respective part $\mathbf{T}$, and hence the function evaluation during the shape polygonization will be performed slower.

The overall incremental shape modelling pipeline is illustrated in Figure 5.2.

To maintain interactivity, additional accelerating steps have to be added to the modelling pipeline (shaded block in Figure 5.2). The principles of the acceleration algorithm are the following:

1. Axis-aligned bounding boxes of the tool-shapes form initial cluster boxes.
2. Cluster boxes can be further clustered into higher-level cluster boxes.
3. Clustering is based on *minimization* of the cluster box volume at each level of nesting.

Each level of nesting is based on a respective maximum cluster box volume.

4. Clustering must take into account the order of set-theoretic operations to preserve the original design.

The clustering begins with the definition of the bounding box of the whole scene as the root cluster box. Then, with the definition of each new tool-shape T_i , the following basic cases of clustering (Figure 5.3) can occur:

1. Current tool cluster box T_i (Figure 5.3(a)) will cluster with the previous tool cluster box T_{i-1} if T_i is inside the parent cluster box C_k of the previous tool cluster box T_{i-1} and the volume V_{C_n} of the box clustering T_{i-1} and T_i is smaller than the maximum volume value δ_j allowed to form a new cluster inside C_k .

Figure 5.3 2D diagrams of the basic cases of clustering

2. T_i (Figure 5.3(b)) will nest into the parent cluster box C_k of the previous tool cluster box T_{i-1} if T_i is inside C_k but the volume of the box clustering T_i and T_{i-1} is larger than the maximum volume value δ_j allowed to form a new cluster inside C_k .
3. C_k (Figure 5.3(c)) will expand to cluster T_i that is outside C_k while the volume of the box clustering T_i and T_{i-1} is larger than the volume of C_k , but the volume of the box clustering T_i and C_k is still smaller than the maximum volume value δ_{j-1} allowed to form C_k .
4. T_i (Figure 5.3(d)) will nest into C_m , which is a parent cluster box of C_k , if all the conditions in case 3 are the same except that the volume of the box clustering T_i and C_k is larger than the maximum volume value δ_{j-1} allowed to form C_k .
5. In case of Figure 5.3(e), T_i is outside C_k while the volume of the box clustering T_i and T_{i-1} is smaller than the volume of C_k . When this happens, we will cluster T_i and T_{i-1} and proceed iteratively further within C_k to consider other tool-shapes in C_k as previous tools, i.e. T_{i-2} , T_{i-3} , etc.

The clustering results in nested clusters. By revisiting the clusters starting from the root cluster, the order of operations can be easily preserved. After one cluster is visited, there are two dimensions to go. One is inside the next cluster, which is in the same parent cluster of the current cluster, and another one is in the first cluster nested within the current cluster. In order to preserve the order of operations, the algorithm always visits the clusters nested in the current cluster first, and then visits the next cluster, which is in the same parent of the current one (See Figure 5.4). This process repeats recursively.

After each cycle of clustering following interactive changes of the shape, the scene function script has to be edited to reflect the defined clusters.

The proposed approach first ensures that the result of the clustering preserves the order of operations, and then optimizes the acceleration. The third principle, minimization of the cluster box volume, is achieved by always clustering bounding boxes which can form a smaller cluster. The aim of the volume minimization is to reduce the overall number of function samplings during the rendering process.

For example, for the case illustrated in Figure 5.3(a), the original non-accelerated script (left) will change to the accelerated script (right) as follows:

Basic shape S definition

.....

Tool T_{i-3} definition

Application of tool T_{i-3}

Tool T_{i-2} definition

Application of tool T_{i-2}

Tool T_{i-1} definition

Application of tool T_{i-1}

Tool T_i definition

Application of tool T_i

If in the root cluster then

{ Basic shape S definition

.....

If in cluster C_k then

{If in cluster T_{i-3} then {

Tool T_{i-3} definition

Application of tool T_{i-3} }

If in cluster T_{i-2} then {

Tool T_{i-2} definition

Application of tool T_{i-2} }

If in cluster C_n then

{If in cluster T_{i-1} then {

Tool T_{i-1} definition

Application of tool T_{i-1} }

If in cluster T_i then {

Tool T_i definition

Application of tool T_i }

} } }

The accelerated script shows how the clusters are used for preserving the order of operations (Please refer to Figure 5.4 for illustration, red lines with arrows show the traversal path). After the cluster C_k is visited, the tool cluster T_{i-3} will be visited, since it is the first cluster nested in C_k . Next, since there is no cluster nested in T_{i-3} , the algorithm visits the next cluster nested in C_k , which is T_{i-2} , and then the next cluster nested in C_k , which is C_n . Since there are two clusters nested in C_n (T_{i-1} , and T_i) they are then visited. Since C_n is the last cluster nested in C_k , all the clusters nested in C_k have been visited, and the algorithm then visits the next cluster nested in the same parent cluster of C_k (this part is not shown in the script to save the space).

Figure 5.4 Traversal of the cluster hierarchy

Figure 5.5 Algorithm flow-chart diagram

The complete flow-chart diagram of the updating algorithm for clustering is shown in Figure 5.5. The proposed method has been implemented as a part of the web-enabled modelling tool and a program which converts unstructured model to the accelerated model. Some implementation details are described in the next chapter. It was rigorously tested and the discussion on its performance follows in Chapter 7.

Three different data formats representing the modeled shape can be saved or exported:

1. Internal data formats that contain information of tool-shape defining functions, each transformation (translation, rotation, scaling) parameters, as well as operations applied, can be saved.
2. Non-accelerated function scripts that contain all information listed in 1, except that all transformation information is stored as one single matrix.
3. Accelerated function scripts that contain all information as non-accelerated function scripts.

5.3 Further improvement for special cases

The proposed method described in Section 5.2 works well for both flat data structure (order of applied operations are not essential to the result) and tree-like data structure (complex models need to consider the order of applied operations). In this Section, we will propose a method that is more dedicated to flat data structure.

One observation is that changing the order of a sequence of tool-shapes, which are applied with the same type of operation, will not affect the resulting shape. Figure 5.6 illustrates this observation. Figure 5.6(a) shows how the tool shapes T_1 and T_2 are placed

on the basic shape S . Figure 5.6(b) shows the resulting shape if T_1 and T_2 are both applied with ‘deposit’ (T_1 after T_2 and T_2 after T_1 will result in the same visual output). Figure 5.6(c) and Figure 5.6(d) show different results that T_1 applied with ‘deposit’ and T_2 applied with ‘remove’ but in a different order. This observation allows us to further accelerate the model by changing the order of applied tool-shapes so that the resulting function script is optimized.

Figure 5.6 Different results while tool-shapes applied with different order

Since changing the order of tool-shapes that were applied with different operations may change the original design, the method proposed in this section only works with some special types of designs, such as designs that with only one type of operation (e.g., 3D wood carving which only uses subtraction). To optimize such design, we can change the order of tool-shapes freely.

The difference between the methods proposed in this and the last Sections is that the clustering strategy is different. The principles of the clustering algorithm are as following:

1. Axes aligned bounding boxes of the tool-shapes form initial cluster boxes.
2. Cluster boxes can be further clustered into higher level cluster boxes.
3. Clustering is based on *minimization* of the cluster box volume at each level of nesting.

Each level of nesting is based on a respective maximum cluster box volume.

The principles here are the same as the principles in the prior section except that the fourth principle in the last Section is not here anymore. This means that the order of set-theoretic operations is not considered here. The insertion order of tool-shapes is not considered, instead, the existing tool-shapes are sorted according to the volume size of the bounding box which bounds it and the current applied tool-shape. Each time a new tool-shape is applied, this bounding box information will be updated. Hence, the clustering algorithm will trace back the tool-shapes according to this sorted list.

Accordingly, the clustering algorithm is also different from the algorithm introduced in the last section. The ‘last inserted tool-shape’ – T_{i-1} - is replaced with the ‘last nearest neighbour tool-shape’ – B (according to the sorted tool-shapes described in the last paragraph).

5.4 Summary

In this Chapter, a new approach to acceleration of the rendering process of a shape model defined with implicit functions and procedures is proposed. The algorithm is devised for free-form shape modelling on the web where some initial shape is gradually modified by other implicitly-defined shapes with relatively smaller sizes compared to the final shape.

The algorithm then adds additional functions to the final functions script, which make the rendering of the whole shape faster. To achieve interactive rendering, we have developed the acceleration algorithm based on cluster volume minimization. The method for clustering the bounding boxes of tool-shapes is also described in detail in this Chapter. The proposed algorithm will work on the web because the whole function definition will be exchanged between the client computers which will perform the actual visual rendering of the model.

Chapter 6 Algorithm implementation

In this Chapter, we present implementations of the proposed acceleration algorithm. In Section 6.1, the algorithm implementation details are illustrated. In Section 6.2 we demonstrate implementation details of the clustering algorithms proposed in Chapter 5. Finally in Section 6.3 we draw conclusions about the implementations.

6.1 Algorithm implementation

The algorithm is based on several independent modules:

- Model loading
- Bounding volumes creation
- Clusters construction
- Script generation

Model loading module extracts and stores information about each tool into an object structure. Each tool object consists of a unique *id* that identifies an implicit function defining it, its correspondent operation, its affine transformations defining the location and the size of the tool in the global coordinate system, and its correspondent local bounding box. For models created using the interactive shape modelling tool, this information is stored concurrently with the modelling process. The unique *id* is generated according to the order in which the tools are applied to the model. These *ids* are then used to index other information, which is stored in arrays. For unstructured models, for example, a model consists of simple primitives such as spheres and cylinders. Their

defining functions are $f_1(x, y, z) = r^2 - x^2 - y^2 - z^2 = 0$ and $f_2(x, y, z) = \min(\min(r^2 - y^2 - z^2, x + h), -x + h) = 0$, where r is the radius of the sphere or the base of the cylinder, h is the height of the cylinder. These local bounding boxes are explicitly defined by the user or calculated automatically where it is possible. In this instance, we define the bounding box as two extreme points:

$$\{P_1(-k \cdot 2 \cdot r, -k \cdot 2 \cdot r, -k \cdot 2 \cdot r), P_2(k \cdot 2 \cdot r, k \cdot 2 \cdot r, k \cdot 2 \cdot r)\};$$

$$\{P_1(-k \cdot 2 \cdot h, -k \cdot 2 \cdot r, -k \cdot 2 \cdot r), P_2(k \cdot 2 \cdot h, k \cdot 2 \cdot r, k \cdot 2 \cdot r)\}$$

for spheres and cylinders, respectively, where P_1 and P_2 are two extreme points of the bounding box. We scale up the bounding box 10% larger than the actual boundary to avoid errors at the boundaries.

Bounding volumes creation module transforms local bounding box of each tool into axis-aligned bounding boxes in the global coordinates. The bounding box is calculated by the following three steps.

1. Eight corner points of the local bounding box are derived from the two extreme points.
2. These eight corner points are transformed into the global coordinate system using the information of the transformations of their respective tool.
3. Then, the extreme values (maximum and minimum) of these eight transformed points along each coordinate are used to define the two extreme points of the axis-aligned bounding box of this tool.

Clusters construction module organizes the bounding volumes/clusters into hierarchies. The hierarchy is implemented as a multi-array tree structure. No restrictions are added to the degree of the tree, which means that the internal nodes of the tree may

have any number of children (at least two, since one child node is redundant). Each *leaf node* of the tree is associated with one tool object as well as the axis-aligned bounding box of this tool, while each *internal node* is associated with the axis-aligned bounding box which bounds all its children. Each *node* of the tree is also assigned an integer number to indicate the level of clustering that this node is in. This integer number is assigned when the node is created and it is updated when the bounding volume of the node is updated. Figure 6.1 shows an example of the created tree. Each *node* is regarded as a cluster, as described in the previous Chapter.

Figure 6.1 An example of the created tree clusters the tools

The data structures and the updating algorithms for the cluster construction module will be described in detail in the next section.

Script generation module is responsible for the generation of the final accelerated function scripts. The algorithm performs a pre-ordered traversal of the created clustering tree to visit each node of the tree exactly once. For the case when an *internal node* is

visited, the algorithm inserts a condition check statement into the final scripts for the node being traversed using the information about its bounding box. In the case when a *leaf node* is visited, besides inserting a condition check statement according to its bounding box information, the algorithm also inserts the applications and definition statements for the respective tool into the final scripts.

The accelerated function scripts is then sent back to the viewing pipeline for either rendering during the interactive modelling process using the web-enabled shape modelling tool, or storing it in a file to be rendered later using other rendering tools as it is further described in Chapter 8.

6.2 Clustering algorithm implementation

6.2.1 Data structure

We store information about the applied tool-shapes in an array. When a new tool-shape is applied, a unique *id* will be arranged to identify it, and this *id* is used as array indexes to look-up information of each tool-shape.

To do clustering, we define an *object* to represent each tool-shape, and insert it into a tree hierarchy. Each tree *node* is considered as a cluster. Each *leaf node* of the created tree is associated with exactly one *object*. An *object* contains all the necessary information for clustering of the tool-shape that it represents. The data structures for an *object* and a tree *node* are as shown in Figure 6.2.

Figure 6.2 Data structures of object and tree node

6.2.2 Insertion

When a new tool-shape is applied, the algorithm inserts a new *object* that represents this new tool-shape into the tree hierarchy. Let us consider an example of inserting a new object **E** into the current tree **T**. We first calculate the volume v_E of this new object **E**, and determine the lowest level ll_E that this new object could be inserted in. If ll_E equals 0 (the highest level), then we associate **E** to the root node and the insertion process is done; otherwise, we obtain a list *l* of objects that already inserted into the tree **T**. If *l* is empty, this means that **E** is the first inserted object, we create a new tree node **F_n** for **E**

and point F_n as the child of T . If I is not empty, then we continue to do the next step: to find a proper position to insert E . To find this proper position, we traverse back of the list I . According to the basic cases illustrated in Section 5.2, we determine the proper position that E should occupy.

When a proper position for the new object is found, and the new object has been inserted into the tree, we have to adjust the tree (we do ‘Adjust_Sibling’ and ‘Adjust_Tree’) to avoid redundancy. The purpose of the function “Adjust_Sibling(F)” is to check the relations of tree F and its siblings S . If S is within F and S meets the requirements of F ’s next lower level, then point S as a child of F . If S is within F but S cannot be the child of F , then we merge S and F to be one node. If S is not within F , but the bounding volume of S and F still under their father (that means $S+F$ still meets the requirements of the next lower level of their father), then we merge S and F to create a new node, say H . Here we have four possible conditions. First, if H ’s level is smaller than that of both S and F , then we keep S and F but point them as children of H . Second, if H ’s level is smaller than that of S but equal to F , then we just do point S as a child of F , and adjust F , H is the new F . Third, if H ’s level is only smaller than that of F , then we point F as a child of S , adjust S , H is the new S . Fourth, H ’s level is not smaller than that of any of them, then we just merge S and F .

6.3 Integration with the interactive shape modelling tool

Figure 6.3 Integrate the acceleration module to the system

Considering the complexity of the clustering algorithm and the length of the code, we chose to implement it with JavaScript in HTML files. The clustering algorithm needs bounding box information of each tool shapes. Thus, in the VRML side, it maintains the bounding box information of all the tool shapes applied. While there is any change in the VRML scene, it will call the subroutine:

```
function OnEvent(par1, par2, time)
```

in HTML/JS scripts by the subroutine:

```
Browser.loadURL( OnEvent (string par1, string par2, string timestamp ))
```

in VRMLScript, 'par1' is assigned with certain string to tell the HTML/JS scripts that the acceleration module is called, while 'par2' is some internal format that contain the

bounding box information of all the tool shapes. After the clustering algorithm is run at the HTML/JS scripts side, it will send back the clustering result to the VRML by the subroutine:

```
bsContact.setNodeField(string nodeName, string fieldname, string value),
```

```
and bsContact.setNodeEventIn(string nodeName, string eventName, string value)
```

Then, the VRML side will regenerate the accelerated function scripts with the clustering results.

This process is triggered on every time there is a change to the operator structure. The result of the acceleration will be applied to the definition of the modeled shape which is transparent to the user.

6.4 Summary

The technical details of the implementation of the proposed acceleration method have been described thoroughly in this Chapter. Four core modules of the acceleration method: model loading, bounding volumes creation, clusters construction, and script generation have been introduced and illustrated. The implementation detail and the pseudo code of the clustering algorithm (insertion and deletion of a tool-shape) are also presented in this Chapter.

Chapter 7 Algorithm performance analysis

In this Chapter we discuss the algorithm control and performance analysis. In Section 7.1, examples are given to illustrate the performance of the proposed acceleration algorithm. In Section 7.2, we analyze the performance of the proposed algorithm.

7.1 Experiments and results

We have done several experiments to compare the performance of the proposed algorithm with rendering of non-accelerated models defined in the function scripts. As it is introduced in Section 5.2, we assume the volume of the whole modelling space is 1 unit, and δ_i is the maximum cluster box volume in level j . We tested shape modelling with the volume of the bounding box of the final shape $\delta_0 = 1$, and the volume of the tool bounding boxes $\mathbf{T}_i = 0.001$. We have also used the following maximum values of the cluster volumes: $\delta_i = k_i \cdot \delta_{i-1}$, $k_i = 0.25$. Here, δ_i is the respective maximum cluster box volume of clusters allowed to be created in level i as is introduced in Chapter 5. These settings allowed us to select maximum 4 levels of clustering, since $\delta_4 = 0.003 > \mathbf{T}_i$, while $\delta_5 = 0.0008 < \mathbf{T}_i$. Three different distributions of 50 tool-shapes were used for the experiment, specifically, randomly distributed, placed in a specific order (side by side), and placed not in any specific order and far away from each other (Figure 7.1). The performance statistics on the three experiments is shown in Figure 7.2. Here, the blue lines (with stars) show the rendering time required for not accelerated models, and the red lines (with squares) shows the rendering time for accelerated models.

Figure 7.1 Illustration of the tool-shape distributions (total number of tools in each test case was 50)

Based on the experiment results shown in Figure 7.2, we can conclude that:

1. Accelerated rendering is essentially faster than rendering of non-accelerated model.
2. With the increasing number of the operations applied to the initial shape, the rendering time for the accelerated model is sustained at the interactive rate that is less than 3 sec for the commonly used rendering parameters.

Figure 7.2 Experiment statistics for Figure 7.1

Next, we tested the case where all tool-shapes intersect at one common part, i.e. their cluster boxes intersect (Figure 7.3). Even in the worst case, the rendering was performed at the interactive rate or 3 sec while the non-accelerated rendering took 25 sec. The line boxes shown in the bottom image of Figure 7.3 are the cluster boxes created for acceleration.

For further testing we created four free-form shape models (Figure 7.4) which were carved with various 3D tools on the surface of a thin box. The rendering statistics of them is shown in Figure 7.5, respectively, and it is consistent with the previous tests.

Figure 7.3 (a) Shape-tools with mutual self-intersection in the middle. (b) The respective tool bounding boxes

Figure 7.4 Function-based models used in testing (a) Plum with 89 tool-shapes (b) Orchid with 33 tool-shapes (c) Bamboo with 56 tool-shapes (d) Chrysanthemum with 82 tool-shapes

Figure 7.5 Experiment statistics for Figure 7.4

Figure 7.6 Free form shapes modelled in VRML and the respective tool bounding boxes.

We then tested two free-form shapes displayed in Figure 7.6 with their bounding boxes. The achieved acceleration rates are the same as in the previous tests, i.e. 6-8 compared with the non-accelerated models.

7.2 Theoretical analysis

The performance of acceleration with different parameters (the number of levels of clustering and maximum clustering volume values for each level) can be analyzed analytically when a model is given and all the tool-shape bounding boxes are known.

Therefore, the optimal parameters for clustering can be estimated if a model is given.

Let us consider a model with volume value of V , the number of tool-shapes of N , and the volume size of the bounding boxes for the tool-shapes of v on average. We assume that a uniform grid sampling (a common method used because of its simplicity to implement) is used during the rendering process. We also assume that the complexities of the functions defining each tool-shape are the same or at least similar. For the analysis, we estimate the performance by the multiplication of time needed for one function evaluation and the total number of function evaluations needed for a given rendering resolution, for example, $m_x \cdot m_y \cdot m_z$, which means that m_x , m_y , m_z numbers of points are sampled along X , Y , Z axis, respectively.

For the non-accelerated case, the total evaluation time can be estimated as:

$$(1 + N) \cdot M \cdot T \quad (1)$$

Here, M is the multiplication of m_x , m_y , and m_z , while T is the time needed to evaluate the defining function of each tool-shape.

For the case when for each tool-shape only its bounding box is used in the accelerated function scripts and no further clustering is done, the overall evaluation time can be estimated as follows:

$$(1 + N \cdot v/V) \cdot M \cdot T + N \cdot M \cdot t \quad (2)$$

Here, t is the time required for evaluation of one function in the condition check statements. Since v is relatively smaller compared to V , and t is relatively smaller compared to T , the overhead introduced by condition checks can be counter-balanced and the model will be rendered faster than the original model.

The aim of the proposed approach to clustering of the tool-shapes into nested

clusters is to further accelerate the function evaluation process by reducing the overall number of condition check statements, which is number N in Equation 2.

If the proposed clustering method is applied and assuming that the total number of levels for clustering is l , the average volume size of the clusters in level i is v_i (where i are integers from 1 to l), and the average number of clusters in level i is n_i . Then, the overall evaluation time can be estimated as:

$$(1+N \cdot v/V) \cdot M \cdot T + (\sum(n_i \cdot v_i)/V) \cdot M \cdot t \quad (3)$$

From Equation 3, it can be seen that the aim for selecting the most suitable number of levels and the maximum volume value for each level for each shape becomes a problem of minimization of the following for the accelerated shape:

$$\sum(n_i \cdot v_i) \quad (4)$$

Most of the existing acceleration methods are either proposed with special rendering algorithms or they only work with special types of implicit functions (e.g., distance functions). Since our algorithm is devised for any type of implicit functions and any rendering platform capable of visualizing implicitly defined shapes, we had to find a suitable benchmark. Hence, we selected octree algorithm since it has wide application including acceleration of implicit function-based shape modelling where finite bounding boxes are considered as volume elements. If octree is used, the accelerated script will look like it is illustrated in Figure 7.7(b). There, the octree has two levels. In level one, 3 cells have volume elements in it, while in level two, 8 cells have volume elements in it. For each cell with volume elements, the algorithm should add one if-condition checking to the accelerated model.

(a)

```

If in the root cell then
{ Basic shape S definition
If in cell  $G_{00}$  then {
If in cell  $G_{0001}$  then {
Tool  $T_1$  definition, Application of tool  $T_1$ 
Tool  $T_2$  definition, Application of tool  $T_2$ }
If in cell  $G_{0010}$  then {
Tool  $T_1$  definition, Application of tool  $T_1$ 
Tool  $T_2$  definition, Application of tool  $T_2$ }
If in cell  $G_{01}$  then {
If in cell  $G_{0100}$  then {
Tool  $T_1$  definition, Application of tool  $T_1$ 
Tool  $T_2$  definition, Application of tool  $T_2$ }
If in cell  $G_{0110}$  then {
Tool  $T_1$  definition, Application of tool  $T_1$ 
Tool  $T_2$  definition, Application of tool  $T_2$ }
... .. }

```

(b)

Figure 7.7 (a) Simple 2D example of octree (b) The respective function script

The overall evaluation time needed by this methods can be given as follows:

$$(1+N \cdot n/L^3) \cdot M \cdot T + (m_1 + m_2 + \dots + m_L) \cdot M \cdot t \quad (5)$$

where L is the total level of the octree, m_i is the number of cells that have volume elements in level i , n is the average number of the cells in level L that each tool-shape is occupying.

To further compare spatial subdivision methods (including octree) with the proposed clustering method, we can analyze Equation 3 and Equation 5. For $(1+N \cdot v/V) \cdot M \cdot T$ in Equation 3 and $(1+N \cdot n/L^3) \cdot M \cdot T$ in Equation 5, $(1+N \cdot v/V) \cdot M \cdot T$ will always give a smaller

value than $(1+N \cdot n/L^3) \cdot M \cdot T$. Because bounding volumes will be tighter than the grids, thus the inequality $N \cdot v/V < N \cdot n/L^3$ is almost always true. For $(\sum(n_i \cdot v_i)/V) \cdot M \cdot t$ in Equation 3 and $(m_1+m_2+\dots+m_L) \cdot M \cdot t$ in Equation 5 (corresponding to calculations for condition check statements), different parameters will cause different results. However, if the goal is to achieve good subdivision results (by reducing $(1+N \cdot n/L^3) \cdot M \cdot T$ in Equation 5), larger L will be needed which results in the increase of $(m_1+m_2+\dots+m_L) \cdot M \cdot t$ in Equation 5. In summary, the result predicted by Equation 3 will be achieved faster than that predicted by Equation 5.

Equation 4 can be used for optimized acceleration of the existing finalized models rather than during the incremental shape modelling process. During the shape modelling process, it is not always possible to predict what tool-shapes will be used. Therefore, it may be not possible to always select optimized parameters for clustering in advance of the modelling process. However, after each incremental step, certain strategies can be applied to adjust the parameters to optimize the acceleration. In general, two properties of a shape can affect the selection of the number of levels of cluster nesting. One is the property relation between the size of the tool-shapes used and the size of the whole model. Another one is how these tool-shapes are distributed in space.

7.3 Summary

In this Chapter, experiments are given to illustrate the algorithm performance. The statistics show that the result of our accelerating method performs quite well. Theoretical analysis is also given to measure the performance as well as comparing with other commonly used method such as octree.

Chapter 8 User Testing of the Developed Software

In this Chapter, the educational application of the developed software is illustrated to verify the proposed concept. Section 8.1 gives an example of how the interactive shape modelling tool is used during a typical shape modelling session for the student assignment in an undergraduate course. In Section 8.2 some students' works are given.

8.1 Interactive shape modelling tool

We used the function-based extension FVRML/FX3D of the Virtual Reality Modelling Language and Extensible 3D [62] [63] and implemented with it an interactive immersive web-enabled shape modelling tool based on the proposed concept. The software is primarily designed as a teaching aid for various university courses including computer graphics, shape modelling and virtual reality at the School of Computer Engineering of the Nanyang Technological University in Singapore. Specifically, in the course Computer Graphics and Visualization, there is a coursework "Visual Mathematics" that requires the students to define, using implicit and parametric functions, a fancy solid shape which geometry should be constructed from a few simpler shapes using set-theoretic operations (union, intersection and subtraction) and the colour has to be defined by functions of the Cartesian coordinates used in the modelling space. The geometry of each individual solid shape has to be defined by functions $g=f(x,y,z,t)\geq 0$. The function value g equals to zero for the points located on the surface of a shape. Positive values indicate points inside the shape and negative values are for the points outside the shape. Union of two shapes,

defined by functions $g_1=f_1(x,y,z,t)\geq 0$ and $g_2=f_2(x,y,z,t)\geq 0$, can then be implemented as function $g_3=\max(f_1(x,y,z,t), f_2(x,y,z,t))\geq 0$, intersection as $g_3=\min(f_1(x,y,z,t), f_2(x,y,z,t))\geq 0$, and subtraction as $g_3=\min(f_1(x,y,z,t), -f_2(x,y,z,t))\geq 0$. Hence, the construction of a complex shape is a process of constructing a long defining function with additional inverse affine transformations of coordinates for positioning individual shapes with reference to each other. Though quite simple by its nature, the process requires understanding of how each individual shape function will be rendered as a 3D shape, and it also requires advanced spatial visualization abilities to imagine how individual shapes have to be placed in space to eventually produce the resulting shape. Last but not least, when the geometry is defined, it has to be coloured using 3D colour functions $red= q_1(x,y,z)$; $green= q_2(x,y,z)$; $blue= q_3(x,y,z)$ where the value of each colour is between 0 and 1.

The interactive modelling tool proposes a concept of incremental modelling where the initial basic shape can be subsequently modified by other shapes with concurrent interactive visualization of the results. The tool does not require any programming knowledge and is based on intuitive interactive manipulations in 3D space and filling in electronic forms with mathematical definitions and parameters. Let's consider an example of a modelling session.

The session begins by running the modelling tool that is web-based. It displays as a window with 3D immersive space where the user can go with or without displaying an avatar. In front of the user, a placeholder of an initial shape will be displayed. A variable set of interactive options will be also displayed on the sides of the modelling space (Figure 8.1).

Figure 8.1 Editing geometry of the initial shape

By selecting ‘Geometry’ and ‘Edit’ buttons, a new window with interactive forms will be displayed. In this window, the analytical function defining the geometry can be typed and edited. Also the parameters of three affine transformations (translation, rotation, scaling) and rendering parameters (bounding box and resolution) can be set or edited any time the geometry editing window is displayed. Each change to the geometry, its transformation or rendering parameters will be immediately reflected in the 3D scene. The defining analytical function can be either implicit, or explicit or parametric. Therefore, the shapes to be modeled can be curves, surfaces and solid objects. In the example shown in Figure 8.1, a solid super-ellipsoid is defined by an inequality $1-x^6-y^6-z^6 \geq 0$. Some default rendering parameters are also set that are the bounding volume defining the modelling space and the respective rendering resolution. Each time any of the fields in the editing window change, the shape will be re-rendered accordingly. Also, the shape can be interactively manipulated in the modelling space. It can be moved, scaled or rotated with reference to the modelling coordinate system. The object

representing the 3D modelling coordinate system can be toggled on and off. Each interactive manipulation with the shape will be immediately reflected in the shape geometry editing window hence converting visual interaction into numbers and formulas.

Next, a new shape that will be used as a modelling tool has to be selected from the tool menu on the right side of the modelling space. It offers a few predefined shapes as well as an ability to create any new tool merely by typing its defining analytical formula. The selected tool will be then displayed in the 3D environment. The user can interactively move the tool, as well as change its size and orientation. The shape of the tool can be also modified by editing its defining function with an immediate visual feedback in the modelling space. When the user is satisfied with the tool shape, size, location, and orientation, a set-theoretic operation can be selected or defined analytically to add the tool to the current shape. See Figure 8.2, where instances of a solid sphere defined by $0.1 - x^2 - y^2 - z^2 \geq 0$ were interactively scaled and positioned on the surface of the initial object, and then used as cutting tools to leave semi-spherical cavities on the surface of the super-ellipsoid. The resulting object now looks like a die.

During the modelling session, the user can always modify the geometry and parameters of the previously applied tools by selecting the shape handles. In this ‘*ShowHandle*’ mode, all the shapes previously applied as tools will be displayed together with the initial shape, and clicking at any of them will display the window with the shape defining function and parameters that is similar to the window used for the initial shape definition (See Figure 8.3). After the parameters of the tools are changed, including its defining function and transformation parameters locating it in the modelling space, the resulting shape will be updated accordingly. Hence, the shape can be replaced with any

other shape while its location in space will remain unchanged, or the location of the shape is modified with or without changing the shape's geometry. Separation of the transformation parameters from the mathematical definition of the shape allows the user to visualize mathematics involved in an intuitive and interactive way.

Figure 8.2 Modifying the initial shape with other shapes (tools)

Figure 8.3 Modifying shapes used as tools

(a)

(b)

Figure 8.4 Definition of the 3D colour space coinciding with the geometry
 (a) by interpolating colours using an explicit distance function
 (b) by defining three independent colour functions for red, green and blue colours. The shape was also truncated to display its inner colours.

In our example modelling session, the geometry construction process continues till the user is satisfied with the look of the shape. Next the visual appearance design begins; however, this could be done first before designing the geometry. By selecting 'Appearance' button, a window similar to geometry definition will be displayed. Here,

the user has to type or edit the analytical formulas defining red, blue and green colours. These are functions of the Cartesian coordinates of the modelling space and their values must stay between 0 and 1 to define the colours properly. The 3D colour field can be defined anywhere in the modelling space and it becomes visible for the user when any geometry is placed into the space. In other words, the geometry will sample the colour defined in the modelling space. Both the geometry and the appearance coordinate systems can be displayed as a reference. Both, the geometry and the colour space can be also interactively moved, scaled and rotated with reference to each other thus producing different colours on the curves, surfaces and inside solid objects. The parameters of all the interactive operations will be immediately reflected in the respective geometry or appearance editing windows (See Figure 8.4).

In Figure 8.4, the colours are defined by two different ways. In Figure 8.4-a, the colour is defined by an explicit function identical to the original shape definition function $distance=1-x^6-y^6-z^6$. The value of *distance* variable, which can range from $-\infty$ to 1, are then linearly mapped to the red, green and blue colour values, which are from 0 to 1, using a key distance values 0, 0.2, 0.2, 1 and the corresponding colour values 0 1 1, 0 1 1, 1 1 1, 1 1 1. For example, when $distance=0$, which means the sampled point is on the surface of the shape, the colour is *cyan*, i.e. $r=0, g=1, b=1$. It remain cyan within a certain distance inside the shape between $distance=0$ and $distance=0.2$ and then changes to red, i.e. $r=1, g=0, b=0$. In Figure 8.5-b, a different approach to colour definition is used. Here, the same function $distance=1-x^6-y^6-z^6$ is directly used in the three analytical colour functions defining red, green and blue colours: $r=abs(distance)$; $g=abs(0.9-distance)$; $b=abs(1-distance)$; that results in a similar visual appearance of the

shape. In this example, the shape was further edited by cutting away half of it to show how the defined colours distribute inside the shape.

After modelling is completed, the shape can be either saved into a proprietary data format for further editing in future, or exported to VRML or X3D files. After exporting the shape, the user can see how many additional operations and coordinate transformations have been added to the source code which were hidden from the user as nonessential for coming up with the design. See Figure 8.5.

Figure 8.5 Modifying shapes used as tools

In the meantime, the acceleration module is running automatically to maintain the interactively. The system collects information (bounding box) of each tool shape automatically, and then generates accelerated function scripts instead of the original function-scripts. The learner will not see any difference between the original model and the accelerated model except that the second one will render faster. The respective accelerated function script is shown in Figure 8.6.

Figure 8.6 Accelerated function scripts

8.2 Selected students work

Figure 8.7 Examples of student's work

Figure 8.7 shows some students' work of an undergraduate course (CZ2003 Computer Graphics and Visualization). All these models are defined with implicit functions (ranging from 10 to 45 tool shapes). For example, the 'Car' model in Figure 8.8, is defined with one basic shape and 33 tool shapes.

Besides the illustrated student assignment, the tool can be used for modelling with parametrically defined geometries, as well as time-dependent shapes and visual appearances (Figure 8.8). It also can be used in collaborative environments where several users can simultaneously work on the same design. We are also including an ability of haptic interaction with the shapes where their tangible properties will be defined in the same way as geometry and visual appearance functions and parameters. This option is currently supported in the haptic version of the function-based extension of VRML and X3D [63].

Figure 8.8 Parametrically defined animated (time-dependent) shapes interactively painted by implicitly defined 3D brush tools

The exported VRML or X3D codes of the created shapes can be used for visualization, edited as needed, as well as used in shared VRML or X3D virtual environments (See Figure 8.9).

Figure 8.9 Shapes defined by analytical functions are placed into an immersive shared VRML environment

8.3 Summary

In this Chapter, educational application is illustrated with an example of a typical modelling session using the proposed interactive shape modelling tool. Using this example, we have shown how a shape is created with implicit functions and interactive manipulations, and how interactivity is achieved with the proposed acceleration method while the learner is not supposed to be aware of it. In addition, we also showed some examples of student works on an assignment in an undergraduate course and shapes defined with parametric functions as well as animated shapes.

Chapter 9 Rendering function-based shape models using POV-Ray

This Chapter discusses how the accelerated function defined models can be rendered using a different rendering platform—POV-Ray ray tracing software.

9.1 Rendering on POV-Ray

The proposed accelerated function-based model is suitable for rendering not only using function-based extension of VRML and X3D. It can be used with any rendering software capable of displaying surfaces and solids defined by implicit and FRep [73] functions. To prove it, we used POV-Ray [100]—de facto standard ray tracing software. POV-Ray has a function "isosurface" capable of displaying surfaces defined by implicit functions.

Since POV-Ray is open-source software, we have edited the respective code in C++ and made it possible to read models defined by analytic function scripts. As we have mentioned, although different polygonization methods and ray-tracing have been proposed to render implicit functions, one common thing is that they are based on sampling as does the method used in POV-Ray for ray-tracing "isosurface". The isosurface patch of POV-Ray was written by Ryoichi Suzuki. According to Ryoichi Suzuki's description, the method of isosurface searching in the used version of POV-Ray (Version 3.5) is as follows (See Figure 9.1). It uses a recursive subdivision equipotential method for ray tracing implicit surfaces. In the first step, POV-Ray calculates the function values $F(d_1)$ and $F(d_2)$ on the ray, where d is the distance from the initial point and $d_1 < d_2$. If there is a possibility of isosurface presence between d_1 and d_2 , POV-Ray will calculate

the function value at another point d_3 , on the ray between the two points d_1 and d_2 . The possibility is evaluated with the values $F(d_1), F(d_2)$, the length from d_1 to d_2 , and MAX_GRADIENT (one parameter used to define an isosurface using POV-Ray). Then, if there is a possibility of isosurface between d_1 and d_3 , POV-Ray calculates another point between d_1 and d_3 . If there is no possibility between d_1 and d_3 , POV-Ray looks for another point between d_3 and d_2 , and so on. This process is carried out recursively until the length of the two sampled distance is smaller than the 'ACCURACY'. We can see that during the surface searching process, at each sampled point function evaluation is required, thus, we can imagine that the rendering speed will be accelerated a lot if we speed-up the function evaluation process.

Figure 9.1 Isosurface searching in POV-Ray

In our experiment, we compared non-accelerated isosurface models and accelerated isosurface models. Although we used an unstructured implicitly defined shape model, any model that has the following feature will be suitable for our experiment:

1. It is defined by graduate modification of an initial implicit defined shape or at least we can identify all the 'elements' that constitutes the final shape.

2. Each ‘element’ is relatively small compared to the whole shape. This would also require that the surface is finite.
3. The bounding box of each ‘element’ is known or at least we can derive it from the definition.

To create a non-accelerated isosurface model, in the first step, we identify all these ‘elements’ represented with their defining implicit functions and operations used to combine them. Then, in the second step, we generate the function scripts that defines it. Thirdly, we replace the function named ‘Evaluate_Function(FUNCTION func, VECTOR fnvec)’ with this function scripts in the source code file named ‘*isosurf.cpp*’ of POV-Ray. Finally, we rebuild the POV-Ray executable file and run it to display the isosurface model. For the accelerated version, we add an extra step that does clustering using the bounding box information of each ‘element’ and then generate accelerated function scripts as we have done when using FVRML. Then we use this accelerated function scripts to replace the function - Evaluate_Function(FUNCTION func, VECTOR fnvec). See Figure 9.2.

Figure 9.2 Process of rendering function-based model uses POV-Ray

9.2 Experiments

Using the method described in the previous section, we have performed rendering of a few models (Figures 9.3-9.5).

The nested balls model was the same as used in the previous chapter but when using POV-Ray it could be rendered with much higher precision. Other two models simulate wood carving which is performed interactively by subtracting materials from a model of a wooden block [66, 101]. At the end, these models are defined by hundreds of subtractions of cylinders and ellipsoids resulting in hundreds of defining functions. Each model was then converted to non-accelerated and accelerated function scripts to compare the rendering performance. Figure 9.6 shows parts of the respective accelerated and non-accelerated functions scripts of the carving model shown in Figure 9.4. The acceleration factor achieved was between 6 and 11 for different ray tracing parameters.

Figure 9.3 Nested balls rendered with POV-Ray

Figure 9.4 Virtual wood-carving rendered with POV-Ray: Mosco nights

Figure 9.5 Virtual wood-carving rendered with POV-Ray: Sunset

```

DBL Evaluate_Function(FUNCTION funct, VECTOR fnvec)
{
    DBL x, y, z, fun;
    DBL x1, y1, z1;

    x1=fnvec[X];
    y1=fnvec[Y];
    z1=fnvec[Z];

    x=x1*(-0.266880)+y1*(-0.963730)+z1*(0.000000)+(0.670359);
    y=x1*(0.963730)+y1*(-0.266880)+z1*(0.000000)+(0.351572);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = min(min(0.003000*0.003000-y*y-z*z, x+(0.143809)), -x+(0.143809));
    x=x1*(-0.141422)+y1*(-0.989949)+z1*(0.000000)+(0.539150);
    y=x1*(0.989949)+y1*(-0.141422)+z1*(0.000000)+(0.281637);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = max(fun, min(min(0.003000*0.003000-y*y-z*z, x+(0.030154)), -x+(0.030154)));
    x=x1*(-0.000001)+y1*(-1.000000)+z1*(0.000000)+(0.492537);
    y=x1*(1.000000)+y1*(-0.000001)+z1*(0.000000)+(0.202559);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = max(fun, min(min(0.003000*0.003000-y*y-z*z, x+(0.053305)), -x+(0.053305)));
    x=x1*(1.127375)+y1*(0.000000)+z1*(-0.007926)+(1.040084);
    y=x1*(0.002827)+y1*(4.983803)+z1*(0.402119)+(0.731657);
    z=x1*(0.070073)+y1*(-0.804258)+z1*(9.967360)+(0.903239);
    fun = max(fun, 0.062500*0.062500-x*x-y*y-z*z);

    return -fun;
}

```

(a) Non-accelerated function scripts

```

DBL Evaluate_Function(FUNCTION funct, VECTOR fnvec)
{
    DBL x, y, z, fun;
    DBL x1, y1, z1;

    x1=fnvec[X];
    y1=fnvec[Y];
    z1=fnvec[Z];

    x=x1*(-0.266880)+y1*(-0.963730)+z1*(0.000000)+(0.670359);
    y=x1*(0.963730)+y1*(-0.266880)+z1*(0.000000)+(0.351572);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = min(min(0.003000*0.003000-y*y-z*z, x+(0.143809)), -x+(0.143809));

    if (fabs(x1-(-0.4676))<1.1225&&fabs(y1-(-0.0507))<1.8879&&fabs(z1-(-0.0921))<0.0500) {
    if (fabs(x1-(-0.1625))<0.0960&&fabs(y1-(-0.4396))<0.9070&&fabs(z1-(-0.1000))<0.0066) {

    if (fabs(x1-(-0.2025))<0.0159&&fabs(y1-(-0.5735))<0.0666&&fabs(z1-(-0.1000))<0.0066) {
    x=x1*(-0.141422)+y1*(-0.989949)+z1*(0.000000)+(0.539150);
    y=x1*(0.989949)+y1*(-0.141422)+z1*(0.000000)+(0.281637);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = max(fun, min(min(0.003000*0.003000-y*y-z*z, x+(0.030154)), -x+(0.030154)));
    }
    if (fabs(x1-(-0.2025))<0.0066&&fabs(y1-(-0.4925))<0.1172&&fabs(z1-(-0.1000))<0.0066) {
    x=x1*(-0.000001)+y1*(-1.000000)+z1*(0.000000)+(0.492537);
    y=x1*(1.000000)+y1*(-0.000001)+z1*(0.000000)+(0.202559);
    z=x1*(0.000000)+y1*(0.000000)+z1*(1.000000)+(0.100006);
    fun = max(fun, min(min(0.003000*0.003000-y*y-z*z, x+(0.053305)), -x+(0.053305)));
    }
    if (fabs(x1-(-0.92327))<0.1220&&fabs(y1-(-0.13637))<0.0285&&fabs(z1-(-0.0953))<0.0167) {
    x=x1*(1.127375)+y1*(0.000000)+z1*(-0.007926)+(1.040084);
    y=x1*(0.002827)+y1*(4.983803)+z1*(0.402119)+(0.731657);
    z=x1*(0.070073)+y1*(-0.804258)+z1*(9.967360)+(0.903239);
    fun = max(fun, 0.062500*0.062500-x*x-y*y-z*z);
    }}}}}}}}}}}}}}}}}}}}}
    return -fun;
}

```

(b) Accelerated function scripts

Figure 9.6 Function scripts of Virtual wood-carving shown in Figure 9.4

9.3 Summary

In this Chapter, we used POV-Ray as an example to show how the acceleration algorithm can be applied to different platforms. Experiments are also given to verify the effectiveness of the acceleration algorithm when it is applied to POV-Ray.

Chapter 10 Conclusion and future work

This Chapter summarizes the work presented in this thesis. In Section 10.1, the conclusion is made, and future work is discussed in Section 10.2.

10.1 Conclusion

Based on the literature review of the existing electronic learning systems and systems for cyber learning mathematics and geometry rich subjects, we have proposed a new approach to visualization and modelling of geometric shapes with the aim to develop geometric reasoning, spatial visualization skills and an ability to visualize mathematical definitions.

The idea of the proposed approach is that complex shapes can be constructed gradually by using individual components defined by a mixture of analytical formulas in explicit, implicit and parametric forms, whichever is more suitable. Geometry, visual appearance and any other additional shape properties, like physical tangible properties, will be modelled in their own coordinate domains and eventually assembled into the shape. At each stage of modelling, only mathematical definitions and immersive interaction with the shape will be used. Hence individual formulas are immediately rendered as well as any part of the shape being modeled can be seen as its defining function. The final shape will be defined as a function script or procedure that is independent on the visualization platform and can be efficiently rendered with such different software tools as, for example, Virtual Reality Modelling Language (VRML), OpenGL, POV-Ray, Visualization Toolkit (VTK).

A new algorithm to accelerate the rendering speed of implicitly defined shape is proposed for supporting interactive incremental modelling of the shapes. We have developed the acceleration algorithms based on cluster volume minimization. The algorithm adds additional accelerating structures to the function model of the shape. These structures are, in turn, additional functions and elements of the script that eventually allow for bypassing during the function evaluation the parts of the model which do not contribute to a particular point of the object being tested. The final accelerated function script can be then rendered on different platforms. We have illustrated the algorithm performance while rendering using VRML/X3D as well as POV-Ray. The acceleration factor achieved was between 6-11 for the typical resolutions and complexities of the objects used in the project.

To verify the usefulness of the developed algorithm, a new virtual learning environment for learning subjects richly infused with mathematics and geometry, specifically, shape modelling is proposed. It allows for immersing the learner into the unlimited world of mathematical definitions. A new concept of the user interface for supporting such immersion is also proposed.

The interactive immersive web-enabled shape modelling tool, which is developed using the function-based extension FVRML/FX3D of the Virtual Reality Modelling Language and Extensible 3D [88], has been used as a teaching aid for the course of Computer Graphics in School of Computer Engineering of the Nanyang Technological University in Singapore (CZ2003/CSC204).

The results achieved were published as two conference and one journal papers [102-104] as well as presented at four PhD seminars and several demo sessions at the

10.2 Future Work

Several directions of future work have been identified: to support more advanced user computer interaction methods, to further improve the acceleration algorithm, and to further expand the properties of the virtual shape in the virtual learning environment.

10.2.1 Immersive Interaction & Unified User Interface

The proposed approach assumes to immerse the learner into the unlimited world of mathematical definitions. More intuitive and immersive interaction method could be proposed and applied to the interactive shape modelling tool. Complex navigation method could be developed. For example, it may allow the learner to go inside the shape, walk along its surface, as well as fly through the shape. One of the methods that can be used here is to provide several view navigation options to the learner. After a certain navigation mode is chosen, the system would automatically plan the path for the learner. For the option to go inside the shape, the system may locate the learner at the middle of the bounding box of the modeled shape. For the option to walk along the surface of the modeled shape, the system may first locate the learner at certain point on the surface of the model, and then allow the learner to go forward, backward, or sideways following the keyboard inputs. For the option to fly through the virtual space, the system may use the bounding box of the modeled shape to derive a bounding sphere or other bounding volumes. Then, using the derived bounding volume, the system will allow the learner to walk along the surface of the bounding volume. Another method which would provide for

more flexibility and can add educational value to the learning environment is to allow the learner to define the navigation path with mathematical functions. The path can be defined in a similar way as curves are defined in the system. Then, the system would allow the learner to fly using the route specified by this curve.

A comprehensive, unified, and stable framework of the user interface could be further investigated, proposed and implemented. With it, all the properties of the shapes defined with mathematical functions will work in a similar way. The system should be very easy to use, and would require no or very few instructions for a new learner. Besides, the user interface should reduce the required steps to complete a task and provide more interactive interactions. For example, small windows could be added to the current modelling space. One is for visualizing the whole shape as the current modelling window does. Another one is for a property of the modelled shape. The learner can specify which property is to be visualized in this window. During the modelling process, the learner can click to maximize one of these two windows, and then the other one will be shown as a small window at the corner of the bigger one. Secondly, the operators used to modify the modelled shape could be organized and represented as a tree like structure which will pop-up when the learner wants to see it. Each operator will be represented as an element in the tree. Then, the learner can select any of them. The system will highlight the influenced area of this operator in the modelling space. Thirdly, in the window which visualizes each property of the modelled shape, the learner can click at any part to see its defining function and other parameters that define it.

Further research to measure the degree to which the interface is intuitive should be done to verify the effectiveness of the proposed approach.

10.2.2 Acceleration optimisation

The acceleration algorithm could be further improved by automatic selection of the parameters used for clustering. As it was mentioned in Chapter 7, after each incremental step, certain strategies can be applied to adjust the parameters to optimize the acceleration. In general, two properties of a shape can affect the selection of the number of levels of cluster nesting. One is the property relation between the size of the tool-shapes used and the size of the whole model. Another one is how these tool-shapes are distributed in space.

One more consideration is that the proposed algorithm uses axis-aligned bounding boxes for clustering. However, we believe it is possible to extend them to any geometry and orientation as soon as /1/ the tool-shapes are clustered into nested clusters, /2/ the clusters at each level will not exceed some maximum value, and /3/ the cluster volume is minimized. Two influencing factors for acceleration should be considered when choosing a bounding shape. One is the tightness of the bounding shape. The tighter bounding shape will result in a fewer condition checks. Another one is the cost of each condition checks. Normally, tighter bounding shapes correspond to more complex formulas, which will eventually require more calculations for each condition check. We will consider it in our future work.

10.2.3 Physical Properties and Haptic Interaction

Currently, the interactive shape modelling tool only allows for defining and manipulating the appearance and geometry properties of the modelled shape. We will add physical properties of a virtual shape into the virtual learning environment in the future. It will

allow the learner to define physical properties of a virtual shape by using methods similar to those that are used to define the geometry and appearance properties. This work will use the haptic version of the function-based extension of VRML and X3D [63].

After incorporating physical properties into the virtual learning environment, the physical properties of a shape can be defined with explicit functions. Then we will support tactile manipulations with haptic devices for virtual shapes whose physical properties are defined. Specifically, the following interaction method will be provided.

1. Allow the learner to touch the surface of a virtual shape as well as feel the internal physical properties of a virtual shape.
2. Allow the learner to move, rotate, and scale the virtual shape with a haptic device.

10.2.4 Audible Mathematics

As we have mentioned, some literature, such as Gardner's multiple intelligence theory [60] and multi-sensory instructions, suggest that a good learning environment should provide the learners the opportunity to receive learning information from multiple-sensory experiences. Currently, we are able to allow the user to define the geometry and visual appearance of a virtual object with mathematical functions, and we also plan to support haptic interaction with the haptic version of the function-based extension of VRML and X3D. However, audible information is not provided yet. If sound also can be defined mathematically, specifically, by analytical functions in a form of implicit, explicit and parametric functions, then, it will complement our approach.

The physical nature of sound is vibrations of the air. The four main attributes which affect the way the sound is perceived are: amplitude, frequency, spectrum, and duration.

This attributes can be represented with rational numbers, thus, each of them can be certainly represented as a mathematical function. Sound information can be encoded. The sound of mathematics is not a new issue, and has been studied by many researches. A comprehensive introduction of mathematics for defining the sound of music, which can be regarded as “beautiful” sound, as well as their relations can be found in David J. Benson’s book [105]. However, to allow the user to define each attributes of sound with analytical functions is not studied yet. One challenge of this work is how to make this task educational. Moreover, how to merge the sound property into the definition of the virtual object is also a challenge.

References

- [1] <http://www.web3d.org/x3d/vrml/>
- [2] <http://www.web3d.org/x3d/>
- [3] [http://en.wikipedia.org/wiki/Constructivism_\(learning_theory\)#cite_note-0](http://en.wikipedia.org/wiki/Constructivism_(learning_theory)#cite_note-0)
- [4] http://en.wikipedia.org/wiki/Constructionist_learning
- [5] S. M. Alessi and S. R. Trollip, *Multimedia for Learning: Methods and Development*, 3rd ed.: Allyn & Bacon, 2001.
- [6] <http://www.mathmedia.com/index.html>
- [7] <http://www.eric.ed.gov/contentdelivery/servlet/ERICServlet?accno=ED385964>
- [8] <http://moodle.org/>
- [9] B. Zayas, "Learning from 3D VR representations: learner-centred design, realism and interactivity," in *AI-ED 2001 Workshop-External Representations in AIED: Multiple Forms and Multiple Roles*, San Antonio, Texas, 2001.
- [10] K. Johnsen, R. Dickerson, A. Raij, B. Lok, J. Jackson, M. Shin, J. Hernandez, A. Stevens, and D. S. Lind, "Experiences in Using Immersive Virtual Characters to Educate Medical Communication Skills," in *IEEE Conference on Virtual Reality*, 2005, pp. 179-186,324.
- [11] L. Xiuwen, X. Cui, and J. Yicheng, "Multi-Level Virtual Reality System for Marine Education and Training," in *First International Workshop on Education Technology and Computer Science*, 2009, pp. 1047-1050.
- [12] http://www.spongelab.com/game_pages/hob.cfm
- [13] <http://gaming.psu.edu/games/econ-102/>
- [14] A. Yeh and R. Nason, "VRMath: A 3D Microworld for Learning 3D Geometry," in *World Conference on educational multimedia, hypermedia and telecommunications*, Chesapeake, VA, 2004, pp. 2183-2194.
- [15] <http://www.whyville.net/smmk/nice>
- [16] www.facebook.com
- [17] <https://twitter.com/>
- [18] www.myspace.com/
- [19] <http://secondlife.com/>
- [20] http://opensimulator.org/wiki/Main_Page
- [21] <https://www.everquest.com/>
- [22] <http://us.battle.net/wow/en/>
- [23] <http://eterna.cmu.edu/web/>
- [24] <http://www.americasarmy.com/>
- [25] J. Ryoo, A. Techatassanasoontorn, and D. Lee, "Security Education Using Second Life," *Security & Privacy, IEEE*, vol. 7, pp. 71-74, March-April 2009 2009.
- [26] T. Monahan, G. McArdle, and M. Bertolotto, "Virtual reality for collaborative e-learning," *Computers & Education*, vol. 50, pp. 1339-1353, May 2008.
- [27] S. Clark and M. L. Maher, "The role of place in designing a learner centred virtual learning environment," in *Computer Aided Architextual Design Futures*

- 2001, Kluwer, Dordrecht, 2001, pp. 187-2001.
- [28] M. L. Maher, "Designing the virtual campus as virtual world," in *the 1999 conference on computer support for collaborative learning*, Palo Alto, California, 1999, pp. 376-382.
 - [29] M. L. Maher and S. J. Simoff, "Collaboratively designing within the design," in *CoDesigning 2000*, 2000, pp. 391-399.
 - [30] M. Corbit and B. DeVarco, "SciCentr and BioLearn: two 3D implementations," in *the third international conference on collaborative virtual environments*, Menlo Park, CA, 2000, pp. 65-71.
 - [31] <http://www.activeworlds.com/edu>
 - [32] Cite "Recent research on cognitive abilities is reinforcing what some psychologists suggested decades ago: spatial ability, also known as spatial visualization, plays a critical role in engineering and scientific disciplines. "
 - [33] <http://www.moe.gov.sg/education/syllabuses/sciences/files/maths-secondary.pdf>
 - [34] Article about constructivism
 - [35] <http://dynamicgeometry.com>
 - [36] <http://www.cabri.com>
 - [37] <http://www.cinderella.de>
 - [38] <http://www.dynageo.com>
 - [39] <http://www.raumgeometrie.de>
 - [40] <http://euler3d.software.informer.com>
 - [41] <http://www.geocentral.net/geometria>
 - [42] <http://www.geomview.org>
 - [43] <http://www.wolfram.com/mathematica>
 - [44] <http://www.ptc.com/products/mathcad>
 - [45] G. Nikolakis, G. Fergadis, D. Tzovaras, and M. G. Strintzis, "A Mixed Reality Learning Environment for Geometry Education " in *Methods and Applications of Artificial Intelligence*. vol. Volume 3025/2004, G. A. Vouros and T. Panayiopoulos, Eds.: Springer Berlin/ Heidelberg, 2004, pp. 93-102.
 - [46] Immersion Technologies Inc. <http://www.immersion.com/index.html>
 - [47] M. Cuixia, D. Yi, T. Dongxing, C. Jia, W. Hongan, and D. Guozhong, "An adaptive sketching user interface for education system in virtual reality," in *IEEE International Symposium on IT in Medicine & Education*, 2009, pp. 796-802.
 - [48] M. Roussou, "A VR playground for learning abstract mathematics concepts," *IEEE Computer Graphics and Applications*, vol. 29, pp. 82-85, January/February 2009 2009.
 - [49] H. Kaufmann and D. Schmalstieg, "Designing Immersive Virtual Reality for Geometry Education," in *IEEE virtual reality conference*, 2006, pp. 51-58.
 - [50] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. M. Encarnacao, M. Gervautz, and W. Purgathofer, "The Studierstube Augmented Reality Project," *Presence: Teleoperators and Virtual Environments*, vol. 11, pp. 33-54, 2002.
 - [51] G. Taxen and A. Naeve, "CyberMath: A System for Exploring Open Issues in VR-based Education," in *SIGGRAPH- Educator's Program*, 2001.
 - [52] G. Taxen and A. Naeve, "A System for Exploring Open Issues in VR-based Education," *Computers and Graphics*, vol. 26, pp. 593-598, 2002.

- [53] <http://www.spacetime.us>
- [54] <http://www.imaginary2008.de>
- [55] <http://www.imaginary-exhibition.com>
- [56] <http://www3.math.tu-berlin.de/jreality>
- [57] <http://3d-xplormath.org>
- [58] <http://www.morenaments.de>
- [59] D. J. Bryant, "A Spatial Representation System in Humans," *Psychology*, vol. 3, pp. 7-9, 1992.
- [60] H. Gardner, *Frames of Mind: The Theory of Multiple Intelligences*: Basic Books, 1983.
- [61] M. A. Gutierrez, F. Vexo, and D. Thalmann, *Stepping into Virtual Reality*: Springer, 2008.
- [62] Q. Liu and A. Sourin, "Function-based Shape Modeling Extension of the Virtual Reality Modeling Language," *Computers & Graphics*, vol. 30, pp. 629-645, 2006.
- [63] L. Wei and A. Sourin, "Function-based Approach to Mixed Haptic Effects Rendering," *The Visual Computer*, vol. 27, pp. 321-332, 2011.
- [64] <http://www.bitmanagement.com>
- [65] <http://www.mozilla.org/en-US/firefox/new/>
- [66] K. Levinski and A. Sourin, "Interactive Function-based Shape Modelling," *Computers & Graphics*, vol. 31, pp. 66-76, 2007.
- [67] A. H. Barr, "Superquadrics and Angle-Preserving Transformations," *Computer Graphics and Applications, IEEE*, vol. 1, pp. 11-23, Jan. 1981 1981.
- [68] J. Gielis, B. Beirinckx, and E. Bastiaens, "Superquadrics with rational and irrational symmetry," in *the eighth ACM symposium on solid modeling and applications*, New York, 2003, pp. 262-265.
- [69] G. Graves, "The magic of metaballs," *Computer Graphics World*, vol. 16, pp. 27-32, 1993.
- [70] G. Wyvil, C. McPheeters, and B. Wyvil, "Data Structure for Soft Objects," *The Visual Computer*, vol. 2, pp. 227-234, 1986.
- [71] M. D.Buhmann, *Radial Basis Functions: Theory and Implementations*: Cambridge University Press, 2003.
- [72] D. Levin, "The Approximation Power of Moving Least-squares," *Mathematics of Computation*, vol. 67, pp. 1517-1531, October 1998 1998.
- [73] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko, "Function representation in geometric modeling: concepts, implementation and applications," *The Visual Computer*, vol. 11, pp. 429-446, 1995.
- [74] Q. Liu and S. Alexei, "Function-based representation of complex geometry appearance," in *2005 ACM Web3D*, 2005, pp. 123-134.
- [75] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics (Proceedings of ACM SIGGRAPH)*, vol. 21, pp. 163-169, 1987.
- [76] G. Nielson and B. Hamann, "The Asymptotic Decider: Removing the Ambiguity in Marching Cubes," in *200t the conference on Visualization*, Washington, DC, 1991, pp. 489-496.
- [77] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual Contouring of Hermite

- Data," *ACM Transactions on Graphics*, vol. 21, pp. 339-346, 2002.
- [78] S. Schaefer and J. Warren, "Dual Marching Cubes: Primal Contouring of Dual Grids," in *12th Pacific Conference of the Computer Graphics and Applications*, 2004, pp. 70-76.
 - [79] G. M. Nielson, "Dual Marching Cubes," in *the conference on Visualization '04*, Washington, DC, USA, 2004, pp. 489-496.
 - [80] J. F. Blinn, "A Generalization of Algebraic Surface Drawing," *ACM Transactions on Graphics*, vol. 1, pp. 235-256, July 1982.
 - [81] D. Kalra and A. H. Barr, "Guaranteed ray intersections with implicit surfaces," *Computer Graphics*, vol. 23, pp. 297-306, July 1989.
 - [82] J. J. van Wijk, "Ray tracing objects defined by sweeping a sphere," *Computers & Graphics*, vol. 9, pp. 283-290, 1985.
 - [83] R. d. Toledo, B. Levy, and J.-C. Paul, "Iterative methods for visualization of implicit surfaces on GPU," in *the 3rd international conference on Advances in visual computing 2007*, pp. 598-609.
 - [84] O. Fryazinov and A. Pasko, "GPU-based real time FRep ray casting," in *Graphicon*, Russia, Moscow, 2007, pp. 69-74.
 - [85] C. Loop and J. Blinn, "Real-Time GPU Rendering of Piecewise Algebraic Surfaces," *ACM Transactions on Graphics*, vol. 25, pp. 664-670, July 2006.
 - [86] T. A. Galyean and J. F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," in *SIGGRAPH 91*, 1991, pp. 267-274.
 - [87] E. Ferley, M.-P. Cani, and J.-D. Gascuel, "Practical volumetric sculpting," *The Visual Computer*, vol. 16, pp. 469-480, 2000.
 - [88] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones, "Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics," in *SIGGRAPH*, 2000, pp. 249-254.
 - [89] E. Ferley, M.-P. Cani, and J.-D. Gascuel, "Resolution Adaptive Volume Sculpting," *Graphical Models*, vol. 63, pp. 459-478, 2001.
 - [90] C. L. Bajaj, J. Chen, and G. Xu, "Modeling with Cubic A-Patches," *ACM Transactions on Graphics*, vol. 14, pp. 103-133, 1995.
 - [91] E. Galin and S. Akkouche, "Incremental Polygonization of Implicit Surfaces," *Graphics Models*, vol. 62, pp. 19-39, 2000.
 - [92] B. Wyvill, A. Guy, and E. Galin, "Extending the CSG Tree. Warping, Blending and Boolean Operations in an Implicit Surface Modeling System," *Computer Graphics Forum*, vol. 18, pp. 149-158, June 1999.
 - [93] M.-P. Cani and S. Hornus, "Subdivision-curve primitives: a new solution for interactive implicit modeling," in *International Conference on Shape Modeling and Applications*, Genova, Italy, 2001, pp. 82-88.
 - [94] A. Barbier, E. Galin, and S. Akkouche, "Complex Skeletal Implicit Surfaces with Levels of Detail," *Journal of WSCG*, vol. 12, pp. 35-42, 2004.
 - [95] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," in *ACM SIGMOD International Conference on Management of Data*, 1984, pp. 47-57.
 - [96] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: an Efficient and Robust Access Method for Points and Rectangles," in *the 1990 ACM SIGMOD International Conference on Management of Data*, 1990, pp.

322-331.

- [97] T. K. Sellis, N. Roussopoulos, and C. Faloutsos, "The R+Tree: A Dynamic Index for Multi-Dimensional Objects," in *the 13th International Conference on Very Large Data Bases*, 1987, pp. 507-518.
- [98] L. Arge, M. d. Berg, H. J. Haverkort, and K. Yi, "The Priority R-tree: a practically efficient and worst-case optimal R-tree," in *the ACM SIGMOD International Conference on Management of Data*, Paris, France, 2004, pp. 347-358.
- [99] V. L. Rvachev, "On the analytical description of some geometric objects," *Reports of Ukrainian Academy of Sciences*, vol. 153, pp. 765-767, 1963.
- [100] <http://www.povray.org>
- [101] A. Sourin, "Functionally based virtual embossing," *The Visual Computer*, vol. 17, pp. 258-271, 2001.
- [102] D. Lai and A. Sourin, "Visual Immersive Mathematics in 3D Web," in *the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, 2011, pp. 519-525.
- [103] D. Lai and A. Sourin, "Interactive Visualization of Mathematics in 3D Web," in *the 2012 International Conference on Cyberworlds*, Darmstadt, Germany, 2012, pp. 122-129.
- [104] D. Lai and A. Sourin, "Interactive Free-form Shape Modeling in Cyberworlds," *The Visual Computer*, pp. 1-11, 07 May 2013.
- [105] <http://www.abdn.ac.uk/~mth192/html/music.pdf>
- [106] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H. Seidel, "Multi-level partition of unity iplicits" *ACM Transactions on Graphics*, vol. 22, issue 3, pp. 463-470, Jul. 2003.
- [107] G. Taubin, "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, Nov. 1991.
- [108] C. Shen, J. F. O'Brien, J. R. Shewchuk, "Interpolating and Approximating Implicit Surface from Polygon Soup", *ACM Transactions on Graphics*, vol. 23, issue 3, pp. 896-904, Aug. 2004.
- [109] M. Kazhdan, M. Bolitho and H. Hoppe, "Poisson Surface Reconstruction", *Eurographics Symposium on Geometry Processing*, pp. 61-70, 2006.

Appendix List of Publications

Journal Papers

1. **Danbo Lai**, Alexei Sourin, “[Interactive Free-form Shape Modeling in Cyberworlds](#),”
The Visual Computer, Springer, 2013.

Conference Papers

1. **Danbo Lai**, Alexei Sourin, “[Interactive Visualization of Mathematics in 3D Web](#)”, in
Proceedings of the 2012 International Conference on Cyberworlds (CW 12), 25-27
Sep, 2012, Darmstadt, Germany. pp. 122-129.
2. **Danbo Lai**, Alexei Sourin, “[Visual Immersive Mathematics in 3D Web](#)”, in
Proceedings of the 10th ACM SIGGRAPH International Conference on
Virtual-Reality Continuum and its Applications in Industry (VRCAI 2011), 12 Dec,
2011, Hong Kong, China. pp. 519-526.