GUIDED TRACE AND STITCH MODELING USING MULTIMODAL INTERACTION

by

Rajarathinam Arangarasan

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2007

Dedicated to my parents

Late. Mr. Arangarasan Rajarathinam and Mrs. Lakshmi Arangarasan

ACKNOWLEDGMENTS

I would like to thank my advisors Prof. Michael L. Gleicher and Prof. Vadim Shapiro for giving me the opportunity to pursue my Ph.D research under their guidance. Their tremendous support and valuable guidance had a major influence on this research. Prof. Michael L. Gleicher's friendly and congenial work environment significantly contributed to the positive outcome of my research, and I will always cherish the wonderful time spent together with him. I am greatly thankful to Prof. Vadim Shapiro for his invaluable help and guidance at a critical junction in my Ph.D program. Without their support the completion of this dissertation would not have been possible.

I would like to thank Prof. George N. Phillips Jr. for providing me an opportunity to work with him, and serving on my thesis committee. Though only for a couple of semesters, working with him and his lab members was one of my most memorable experiences during my Ph.D program.

I would like to thank Prof. Rajit Gadh for providing support and the opportunity during the early stages of my Ph.D program. I would like to thank Prof. Sanjay G. Dhande, Director of Indian Institute of Technology - Kanpur, for helping me with the first step of my long journey. I would like to thank Prof. Carl de Boor and Prof. Stephen Chenney for their valuable comments that greatly influenced my research. I would like to thank Prof. Amos Ron and Prof. Suresh Krishnan for serving on my committee and providing their valuable feedback in this research.

I would like to thank my colleagues, fellow Ph.D students, and friends: in UW Computer Graphics Lab (Dr. Lucas Kovar, Alex Mohr, Michael Wallick, and Eric McDaniel), in Spatial Automation Laboratory (Dr. Srinivas Raghothama, Arpan Biswas, and Vasu Ramaswamy), and in ICARVE Lab (Dr. Tushar Dani, Dr. Chi-Cheng Chu, Dr. Nuggehalli Shyamsundar, and Dr. Fan Zhao). They each helped make my time in the PhD program more fun and interesting.

I thank my parents, Late Mr. Arangarasan Rajarathinam and Mrs. Lakshmi Arangarasan, for instilling in me confidence and a drive for pursuing my PhD. I am forever indebted to them for their unending love and affection, and for their unconditional support in achieving my dreams. Last but not least, I thank my wife Andrea Rajarathinam for being a constant source of encouragement and love in my life.

Rajarathinam Arangarasan
2007

DISCARD THIS PAGE

TABLE OF CONTENTS

Appendix

Page

DISCARD THIS PAGE

LIST OF TABLES

DISCARD THIS PAGE

LIST OF FIGURES

Appendix
Figure Page

Figure Page

GUIDED TRACE AND STITCH MODELING USING MULTIMODAL INTERACTION

Rajarathinam Arangarasan

Under the supervision of Professors Michael L. Gleicher and Vadim Shapiro
At the University of Wisconsin-Madison

Freeform modeling is an integral part of the geometric modeling and design process. Existing freeform modeling systems expect the user to be familiar with the geometric representation and user interface in order to model precisely and rapidly. As a result, modeling complex freeform geometries precisely and rapidly can be quite difficult.

In this thesis, we address the problems manifested in current freeform modeling systems. We introduce a novel modeling approach called "Guided Trace and Stitch" (GuTS) which designs curves and surfaces precisely, fluently and rapidly. To achieve its ends, the GuTS modeling approach has multiple components - tracing, snapping, stitching, and a multimodal user interface. GuTS creates new curves and surfaces by tracing predefined geometries to create an accurate replica of the existing model, hence achieving precision. The GuTS modeling approach also allows multiple geometries to be snapped together. As a result, multiple geometries are easily and rapidly traced over and stitched together to create a complex geometry, hence achieving rapidity and precision.

The GuTS modeling approach raises several challenges for user interaction in interactive modeling. However, to address these challenges, we implemented the idea of a multimodal user interaction which uses two-handed input, 3-dimensional mouse input, pen and data tablet, voice input, and synthesized speech output. Basically, the goal was to create a user interaction which adapts natural interactions in the modeling realm. This multimodal interface is designed in such a way as to provide fluent and direct interaction. Further, the simultaneous use of multiple input and output modes through effective coordination provides rapid interaction.

We also implemented a software system based on this approach called the "GuTS System". The GuTS modeling approach is independent of the underlying data structure. To demonstrate, the GuTS approach is implemented using Subdivision curves and Bezier curves representation in two-dimension, and using tessellated geometry representation in three-dimension. This system demonstrates that the GuTS modeling approach is feasible. Using the GuTS system, we created several sample complex freeform geometries to demonstrate the practical usability of this system.

Michael L. Gleicher and Vadim Shapiro

ABSTRACT

Freeform modeling is an integral part of the geometric modeling and design process. Existing freeform modeling systems expect the user to be familiar with the geometric representation and user interface in order to model precisely and rapidly. As a result, modeling complex freeform geometries precisely and rapidly can be quite difficult.

In this thesis, we address the problems manifested in current freeform modeling systems. We introduce a novel modeling approach called “Guided Trace and Stitch” (GuTS) which designs curves and surfaces precisely, fluently and rapidly. To achieve its ends, the GuTS modeling approach has multiple components - tracing, snapping, stitching, and a multimodal user interface. GuTS creates new curves and surfaces by tracing predefined geometries to create an accurate replica of the existing model, hence achieving precision. The GuTS modeling approach also allows multiple geometries to be snapped together. As a result, multiple geometries are easily and rapidly traced over and stitched together to create a complex geometry, hence achieving rapidity and precision.

The GuTS modeling approach raises several challenges for user interaction in interactive modeling. However, to address these challenges, we implemented the idea of a multimodal user interaction which uses two-handed input, 3-dimensional mouse input, pen and data tablet, voice input, and synthesized speech output. Basically, the goal was to create a user interaction which adapts natural interactions in the modeling realm. This multimodal interface is designed in such a way as to provide fluent and direct interaction. Further, the simultaneous use of multiple input and output modes through effective coordination provides rapid interaction.

We also implemented a software system based on this approach called the “GuTS System”. The GuTS modeling approach is independent of the underlying data structure. To demonstrate, the GuTS approach is implemented using Subdivision curves and Bezier curves representation in two-dimension, and using tessellated geometry representation in three-dimension. This system demonstrates that the GuTS modeling approach is feasible. Using the GuTS system, we created several sample complex freeform geometries to demonstrate the practical usability of this system.

Chapter 1

Guided Trace and Stitch Modeling Using Multimodal Interaction

1.1 Introduction

Freeform modeling is integral to the design and modeling process. The ability to create precise freeform curves and surfaces is a vital part of applications such as illustration, design and analysis. The challenge of precise freeform curve and surface modeling arises out of the creative nature of freeform modeling. Complex geometries can be created by either building off of existing curves and surfaces or by building something completely from scratch. Freeform modeling is unlimited by a creator's ideas but is hindered by current technology. As a result, a model ahead of current technology can be difficult to specify and manipulate interactively. The current freeform modeling systems provide either precision or fluency, but not both simultaneously. Since both precision and fluency are important in the design process, the lack of either one can be quite frustrating to the user.

In this research we introduce a novel modeling approach called "Guided Trace and Stitch" (GuTS) modeling using multimodal user interaction which provides precision, rapidity, and fluency all at the same time. The term "multimodal" means "having more than one mode"[5], "two or more modes of operation and it is used to refer to a myriad of functions and conditions in which two or more different methods, processes or forms of delivery are used"[6]. As it relates to this research (i.e. human computer interaction), multimodal means the use of multiple different communication channels to extract and convey information between computer and user.

1.2 Motivation

Recent growth in computing power allows users to design complex geometries that were not possible earlier. As the complexity of geometry increased, issues surrounding performance and user interface increased considerably. Some significant approaches to address the above problems are: improvement of geometric representation, creation of efficient algorithms for real time geometric manipulations, and improvement of input/output (I/O) devices for effective user interface. Most modeling applications use a control point-based approach to precisely model complex geometries. To design complex geometries precisely and rapidly, the user needs knowledge of the geometric representations and the system's user interface. Limitations, such as indirect manipulation, lack of knowledge of geometric representations and user interface of the system, become significant when designing complex freeform shapes involving multiple geometries. To improve user interaction some research, discussed in Chapter 2, used higher degrees-of-freedom (DOF) I/O devices along with a simpler mesh structure to represent the geometries. These interaction approaches provide direct and fluent manipulation but lack precision. Some research focused on achieving precise modeling but lacked fluency and directness. Other research focused on fluency and directness but compromised precision. A trade off always existed between precision, fluency, and rapidity.

Many modeling situations benefit from precise, fluent and rapid freeform modeling. For example, in the final stages of conceptual design, the ability to create precise shapes validates the conceptual model directly. Another example is the ability of novice users to design complex geometries precisely and rapidly. In situations like these all three characteristics - precision, fluency and rapidity - play a vital role.

1.3 Goal and Overview

Our goal is to conceive, build, and implement a system to design complex curves and surfaces precisely, fluently and rapidly. To achieve this goal, we introduce a novel approach called "Guided Trace and Stitch" (GuTS) modeling using multimodal interaction.

In the GuTS modeling approach, pieces of new curves and surfaces are traced from guide shapes and stitched together to create complex shapes. Precise segments of geometries are created by tracing over the guide shapes. Automated snapping and user interaction mechanisms precisely place the geometries and rapidly manipulate multiple geometries. Thus precision is achieved through guided tracing over the guide geometries (for both curves and surfaces). Automated snapping mechanisms for the curves set the definition of precise conditions with minimal input from the user, eliminating additional input and time.

Achieving precision and rapidity using the GuTS modeling approach raises several issues for user interface. To address these issues we used multimodal user interaction, such as two handed input, 3-dimensional (3D) mouse input, pen and data tablet, voice input, and synthesized speech output. By effectively using multiple input and output modes simultaneously, the user interface is designed to achieve fluency, direct interaction, and rapidity.

1.4 Explanation of Precision, Rapidity, and Fluency

A brief explanation of the terms precision, fluency, and rapidity will be useful.

The definitions of precision and accuracy are defined as [1, 9]:

> "Precision is usually characterized in terms of the standard deviation of the measurements, sometimes called the measurement process's standard error. Precision is sometimes stratified into: Repeatability - the variation arising when all efforts are made to keep conditions constant by using the same instrument and operator, and repeating during a short time period; and, Reproducibility - the variation arising using the same measurement process among different instruments and operators, and over longer time periods.

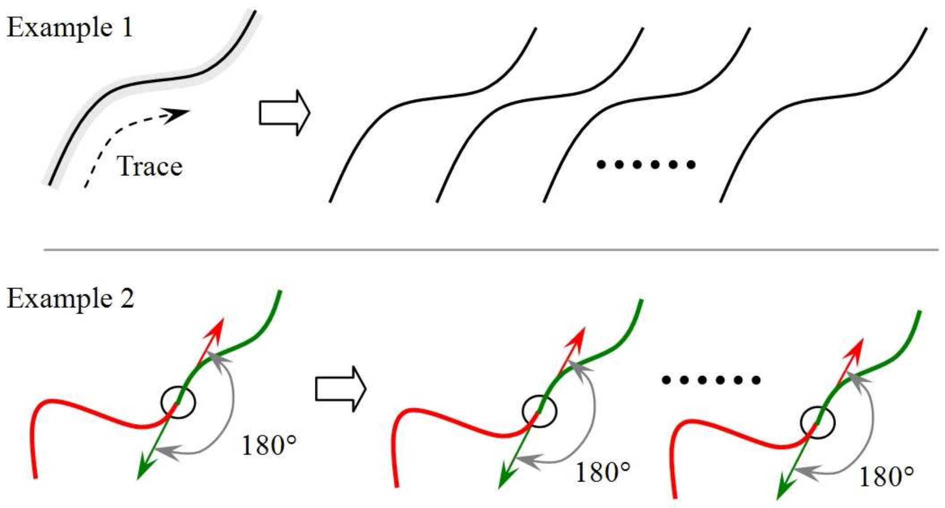


Figure 1.1 Geometric precision

Accuracy can be said to be the ‘correctness’ of a measurement, while precision could be identified as the ability to resolve smaller differences.”

For further clarification to the readers, the difference between precision and numerical accuracy is explicitly discussed with an example. In this thesis we use precision for geometric modeling, further referred to as geometric precision. Geometric precision represents the ability to consistently recreate or realign multiple geometries (curves and surfaces) with desired properties such as shape, continuity, and relationships. Geometric precision can be within a single geometry or as a geometric condition between multiple geometries. Figure 1.1 shows two examples of geometric precision. In Example 1, as the user traces over a guide curve, a new curve is created that is an exact replica of the input guide curve. The curve is exactly reproduced irrespective of the number of times it is traced, maintaining precision within a single geometry. In Example 2, as two geometries are brought closer, they snap such that the tangents are exactly in the opposite direction (i.e. 180°). Again, irrespective of how many times these two objects are brought closer, they will always snap such that the same tangency condition is produced. This ability to consistently recreate the properties of a geometry or between multiple geometries represents geometric precision.

On the other hand, numerical accuracy is the degree of conformity of a measured or calculated quantity to an actual value. For example, accuracy could be measured by the number of decimals places that are used to represent a numerical value. In the above case, when we mention 180° (actual value or target value), both 180.000001° and 179.999999° will be considered equal to 180° with up to 5 decimal places accurate. Unless explicitly mentioned otherwise, in this research the term precision refers to geometric precision, rather than numerical accuracy. Further details about precision and accuracy can be found elsewhere [1, 9].

Fluency is defined as “the property of a person or of a system that delivers information quickly and with expertise - fluency indicates a very good information processing speed, i.e. very low average time between successively generated messages” [97]. Fluency is not associated with any particular task. It is the state of being able to smoothly and easily perform a given function or task. With respect to this research, fluency is defined as a sequence of events or an approach the user is already familiar with and/or able to perform with ease. In addition to familiarity, the system also should respond as a user normally expects in order to maintain the ease of use. Simply stated, fluency shortens the learning curve and anticipates the user’s outlook. Thus fluency is a combination of familiar approach, ease of use, and expected natural response from the system.

Rapidity is the ability to perform a task or operation in a short period of time. It is measured directly in terms of time and indirectly by the number of operations needed to perform a certain task and ease of use. We measure the success of the GuTS modeling approach in terms of rapidity if the system allows the user to complete modeling operations faster than another similar modeling system.

1.5 Overview: Guided Trace and Stitch Interface (GuTS) Modeling Approach

The following section briefly discusses the GuTS modeling approach and is divided into three sections: curve modeling, surface modeling, and user interaction. The curve modeling and surface modeling sections focus on the modeling of curves and surfaces through the Guided Trace and Stitch (GuTS) modeling approach. The user interaction section focuses on the design of an effective user interface with multiple I/O modalities that suit the GuTS modeling approach.

1.5.1 Curve Modeling

GuTS creates complex curves by stitching together pieces of curves that are traced from a set of guide curves. The three basic steps in creating complex curves are:

1. Tracing and drawing pieces of curves from guide shapes
2. Positioning the guide shapes precisely to an existing curve
3. Stitching the pieces of curves together

In GuTS the new geometry is created by tracing over the guide shapes. By looking at the guide shapes the user will know exactly the shape s/he is going to create. This is the “What You See Is What You Get” (WYSIWYG) modeling approach. It also eliminates the user’s need to be familiar with geometric representation and the GuTS modeling approach.

The primary mechanism to create curves in GuTS is to draw them directly using a pen-based input device just as with the paper and pencil drafting process (refer Fig.1.4) . More precise shapes are created by using guide shapes and

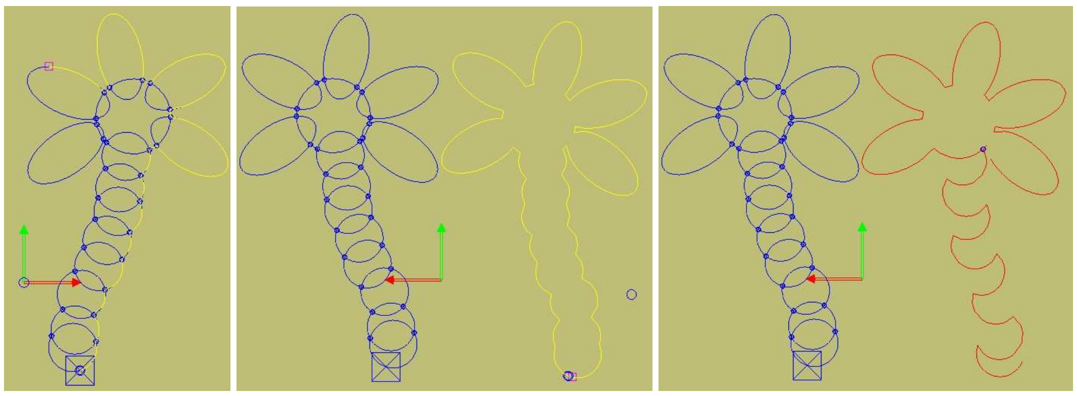


Figure 1.2 Tracing of 2D curves from guide shapes in GuTS

snap tools that help precisely position the cursor (drawing position). Guide curves for tracing can be of any shape or any type (lines, arcs, or freeform curves), including curves created in GuTS system. This flexibility allows GuTS to be easily adapted to create a wide range of complex shapes for different applications. Guide shapes are transformed as needed by the user so that curves can be placed where needed. Our preferred mechanism for positioning guides is to use a 3D mouse in the non-dominant hand, allowing the user to position, rotate, and scale a guide while tracing it with the dominant hand using a pen tablet. Since this directly mimics the drafting process, it provides fluency and a direct approach to drawing curves. Figure 1.2 shows three images using GuTS. The first image shows tracing over guide geometries, the second image shows a complex curve created after tracing over guide curves, and the third image shows a different resulting complex curve that is traced differently but from the same input guide curves.

In GuTS, snapping is performed at different levels. Snapping mechanisms establish precise relationships between multiple geometries with minimal user input or loss of time. Snapping the cursor is the mechanism by which precision is achieved during the tracing operation. Snapping the guides to existing curves creates the connection between the geometries for tracing. Relationships between curve segments are also precisely achieved by using snapping mechanisms. To assist in the creation of precise relationships between segments, the GuTS approach provides a set of snapping operations for positioning the guide curves. These snapping mechanisms are variants of gravity fields [93]. When a guide curve is close to a relationship with existing curve pieces, it snaps to a position that precisely establishes these relationships. Once snapped, this relationship is maintained as the user further manipulates the guide, until the guide is pulled away from the precise relationship. Though snapping mechanisms were used earlier [23, 24, 25] in the modeling process, geometries were snapped mainly to points and straight lines using mouse cursor position. In GuTS, several snapping mechanisms, such as one-point snap, two-point snap, slide snap and conditional snaps such as tangential and perpendicular snaps, are used to achieve relative alignment; and absolute position and orientation alignment is achieved using grid-based snapping. Other geometric operations, such as cutting, smoothing, joining, sweeping curves to create surfaces, and level of detail, also work in the GuTS system.

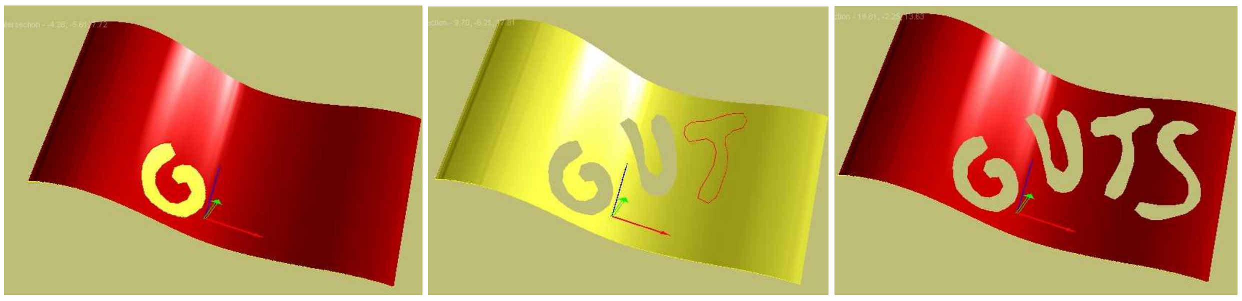


Figure 1.3 Tracing over a 3D surface in GuTS

The GuTS approach is independent of the underlying geometric representation and can be implemented using various geometric representations. To demonstrate the feasibility, we implemented the GuTS system using two different representations: Subdivision curves and Bezier curves representation in two-dimension (2D) and tesselated geometry representation in three-dimension (3D). From now on to distinguish between the GuTS modeling approach and the implemented software system, the approach is referred as the "GuTS modeling approach" or the "GuTS approach" or simply "GuTS", and the software system is referred as the "GuTS system".

1.5.2 Surface Modeling

In GuTS, surfaces are modeled using similar approaches as curve modeling. Several of the features – such as guided trace, stitch, intersect, and smooth – that are performed in curve modeling are extended to surface modeling.

In GuTS new surface patches (or pieces) are created by tracing over guide surfaces. A desired surface patch is produced by selecting a region on the guide surface through tracing. This tracing approach easily creates complex surface patches that cannot be created with ease using the conventional sweeping approach. For completion, the GuTS system also allows creation of surfaces through conventional operations, such as extrude, revolve, and sweep.

Tracing the surface patches from guide shapes is primarily done in two ways:

1. By using a point pen to draw a curve directly over the guide surface to specify an enclosed region

2. By using a thick brush to paint over the guide shapes to select portions of the guide surfaces

Both of these approaches produce the same resulting surface patch, but each approach serves its own purpose depending upon the tracing region. A detailed description of the tracing operations is presented in later chapters.

Figure1.3 shows three images. The first image shows a 3D surface geometry with a curve (letter 'G') traced over it. The second image shows a couple of characters trimmed out from the surface and a third letter being traced. The third image shows after the word 'GuTS' was traced out from the input surface.

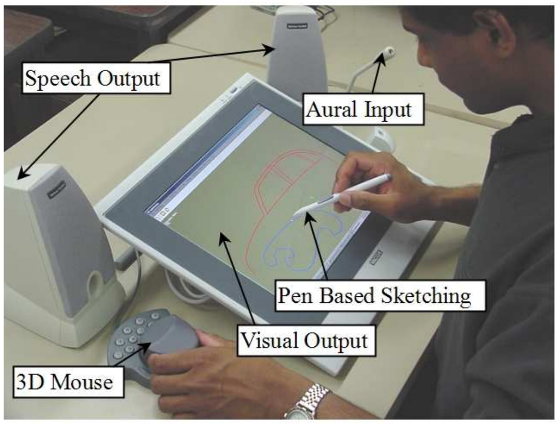


Figure 1.4 Setup of multimodal user interface

Often complex surfaces are produced by merging several surfaces together. Hence, once the pieces of surfaces are created as discussed above, they are stitched together to form the final desired complex surface. The GuTS system allows tracing over multiple guide shapes and produces a complex surface transparently.

1.5.3 User Interaction

The GuTS approach, like other modeling approaches, provides unique challenges for the interface to a design program. For example, in the GuTS approach a user must manipulate both the guide geometries as well as other (non-guide) geometries. Our implemented prototype system explores solutions to these challenges and attempts to maintain the direct approach and rapidity of the user interface. While the GuTS approach can be implemented with conventional keyboard, 2D mouse and monitor environments, we used multimodal interaction with multiple I/O devices to enhance the ease of use which then helps achieve rapidity and fluency.

We used a two-handed interface to achieve a natural and direct modeling interface, similar to the drafting process. The user holds a stylus-pen in his dominant (right) hand and a 6 degrees of freedom (DOF) 3D mouse in his non-dominant (left) hand (refer Fig.1.4). A key advantage to the 3D mouse over 2D devices is that it allows direct manipulation of 6 DOF in the 3D environment. A pen in the dominant hand is used to sketch and trace over the geometries. A flat panel display with integrated data tablet allows the user to draw the curves directly on the monitor as if drawing on a sheet of paper. This configuration of the data tablet and pen provides effective hand-eye coordination during the tracing operation. The non-dominant hand is used mainly to control imprecise manipulations, such as rotating the object. The automated snapping mechanisms achieve precision from imprecise input. Apart from controlling the guide shapes, other coarse level controls such as transforming the geometries, scaling, and panning are also performed by the 3D mouse in the non-dominant hand. This allows manipulating the geometries without diverting the user's current operation.

The GuTS system effectively uses input devices in both of the user's hands simultaneously. Hence, during mode switches and when invoking commands that are performed using menu selections in conventional systems, the current operation and workflow are not interrupted. We also use additional input mechanisms such as speech input to issue voice commands and perform mode switches without interrupting the current operations being performed using both the hands.

It is important to provide the user with a steady stream of feedback that includes the state of the system. In addition to the cues and visual output such as graphical displays, symbols and colors, we use auditory feedback with synthesized speech output to provide additional uninterrupted feedback. In some cases, voice output is a mechanism to guide the users actions. We believe this leads to interfaces that allow the user to keep his visual attention focused on the current operation, without the distraction of dialog boxes or text output windows.

Humans possess the natural ability to interact with each other through multiple modes concurrently in day-to-day communication. By utilizing this natural ability, our prototype of multimodal interaction provides the user with two-handed and aural inputs, good hand-eye coordination, and, visual and speech outputs. Since these I/O modes imitate the real-world interactions of humans, they present a familiar, effective and direct approach of interaction between the system and the user. Rapid interaction is achieved since the user interacts with multiple I/O modes simultaneously. A detailed discussion of the multimodal user interface in the GuTS modeling approach is presented in Chapter 5.

1.6 Thesis Statement

My thesis is that one can design complex curves and surfaces precisely, rapidly, and fluently using a Guided Trace and Stitch (GuTS) modeling approach with multimodal user interaction.

I support this thesis statement by presenting the concept of the GuTS approach and its novely; by showing that the approach is feasible by providing a set of algorithms sufficient for implementing it; and by demonstrating an implemented prototypical system to support that this approach is feasible in practice.

1.7 Contributions

1.7.1 Novel Approach

One of the primary contributions is the introduction of a novel approach called Guided Trace and Stitch (GuTS) modeling using multimodal interaction to model freeform geometries precisely, fluently and rapidly. In the GuTS approach geometries are created using three operations: guided tracing, automatic snapping, and stitching. Precision is achieved through guided tracing operations (for both the curves and surfaces) and automated snapping mechanisms (for curves). Since the tracing operation is direct and fluent, it eliminates the need for user knowledge of the underlying geometric representation. The GuTS approach also allows creation of complex geometries through tracing over multiple guide geometries.

Secondary contributions within the GuTS approach include: (a) new snapping mechanisms, such as pivot snapping, slide snapping, and two-point snapping, and (b) an algorithm to trace over curves and surfaces that allows pieces of curves and surfaces to be traced and stitched together to form complex shapes transparently.

I support these claims of a novel approach by:

- Showing no other similar system exists that does similar tasks with a similar multimodal user interface.
- Comparing with some of the existing software systems (and their features) and highlighting how the GuTS modeling approach is novel (for curve modeling, for surface modeling, and the multimodal user interface).

1.7.2 Multimodal User Interaction

The second primary contribution is the introduction of a multimodal user interface that provides a direct approach to manipulation and interaction, and enables rapid design through uninterrupted multiple I/O modes. This multimodal user interaction brings out the potential of the GuTS approach by expanding and exploring the way a user manipulates curves and surfaces.

I support these claims of the multimodal user interaction by:

- Comparing with other modeling systems how (and if) they use multimodal interface in geometric modeling.
- Highlighting how multimodal user interaction is uniquely integrated with the GuTS approach of geometric modeling.

1.7.3 Implementation

The concepts of the GuTS approach needed to be tested for practicality. As part of this research, we designed and developed an interactive prototype system, called the GuTS system, that demonstrates the GuTS approach with multimodal user interaction. The results of the prototype demonstration confirm the feasibility of this novel modeling approach. The implementation of the GuTS system itself is not a contribution, but the lessons learned through the implementation helped refine the GuTS approach iteratively.

I support these claims of implementation by:

- Demonstrating a working prototype system and highlight different features of the GuTS approach.
- Explaining how this implementation can be extended to other types of geometric modeling (i.e. using different geometry representations) and variations of user interface.

1.8 Structure of Dissertation

The rest of the dissertation is organized as follows:

Chapter 2 describes related previous research performed in the following areas: curve modeling, snapping mechanisms, surface modeling, multiple configuration of I/O devices, and analysis of 6 DOF input devices and user interaction approaches. A review of the previous work helps to understand the earlier geometric modeling and user interaction approaches. The review also clarifies how past research lacks precision, rapidity, and fluency at the same time which then becomes the starting point of the GuTS theory.

Chapter 3 explains the user's perspective of the GuTS modeling approach in detail. This chapter focuses on the user's viewpoint of the GuTS approach and its unique features - such as tracing, snapping, stitching, etc. This is followed by example geometries created using GuTS. This chapter visualizes the GuTS approach from the user's point of view.

Chapter 4 discusses in detail the geometric representation, underlying data structures, geometric algorithms for several features, and other technical details of the GuTS system for curve and surface modeling. The algorithmic details presented in this chapter serve as the mathematical validity of the GuTS modeling approach.

Chapter 5 describes the GuTS system architecture, multimodal user interface, multiple I/O devices, multiple modalities and its synchronization. Discussion in this chapter explains the concept of multimodal user interaction and its effective usability in the GuTS system.

Chapter 6 presents the sample models created using the currently implemented GuTS system. These examples serve as a proof of feasibility of the GuTS approach in practice. Also, discussion of how these models could not be produced by other software with the same results is presented.

Chapter 7 summarizes the dissertation.

Chapter 2

Previous Work

In this chapter, the related research is divided into two categories: geometric modeling and user interaction. In the geometric modeling section, we will discuss different types of modeling approaches while in the user interaction section we will discuss different types of input/output (I/O) devices and user interaction approaches.

2.1 Geometric Modeling Approaches

Several modeling approaches were implemented in the past to model curves and surfaces interactively. Some of the widely used freeform manipulation and modeling approaches are: control point and control polygon-based manipulation, snapping, drafting, tracing, pattern matching, gesture and sketch-based, and blob modeling.

2.1.1 Control Point and Control Polygon Manipulation Approach

Interactive drawing began with Ivan Sutherland's Sketchpad system [93], where a light pen is used to draw shapes. Rather than tracing the paths of shapes, as with paper and pencil, points were placed to define the geometries. To date, this is the dominant paradigm in drawing interfaces: geometry is controlled through a small set of points, such as the end points of lines. One advantage to the control point manipulation approach is that by placing the control points precisely, a precise geometry is produced. The control point manipulation approach is simple and direct for modeling simple geometries such as lines, arcs, circles, etc. However, the same cannot be said for modeling freeform geometries. Control point manipulation becomes tedious, time consuming, and requires expertise in geometric modeling in the case of freeform geometries - especially complex ones. Also, the manipulation of control points in freeform shapes is limiting as it requires the user to think in terms of control points rather than the geometry itself. Recent modeling software allows geometries to be edited directly, but still the user needs knowledge of the underlying geometric representation in order to model freeform geometries precisely and rapidly.

2.1.2 Snapping Approach

Methods such as gravity fields [93], grids, and Snap-Dragging [23, 24, 25] are used to precisely position control points. In this research, snapping mechanisms precisely position a point to construct curves in 2D and planar-faced objects in 3D. Since points can be placed precisely, this approach is significant and widely accepted in control point/polygon manipulation modeling approaches. Despite this, it is not suitable for freeform modeling. This research is a good starting point for the snapping idea and helps us extend new snapping mechanisms into our freeform geometric modeling research.

Baudelaire et al. [21] introduced planar maps to create new shapes by sketching planar shapes of regions, called “map sketching”. In this research multicolored, multi-contoured shapes are constructed through iteration of three basic steps: drawing, erasing, and coloring. First, multiple curves are drawn. Then, the unnecessary portions of the curves are erased. Finally the geometries and regions are colored to achieve the final shape. The geometric primitives used in the research are lines, arcs, circles, ellipses and Bezier curves. Recently, Asente and Schuster [16] extended this planar map concept to a system called “Live Paint” that is available as a feature in the commercial illustration software Adobe® Illustrator® CS2. While this research provided an unique approach of drawing and erasing multiple curves together to form a final shape, it did not focus on creating the initial geometries or on creating freeform surfaces. The GuTS approach, on the other hand, does provide a way to create, modify and manipulate both curves and surfaces using the same approach.

Honda, et al. [56] discussed an interaction technique called “integrated manipulation” that allows basic operations such as move, rotate and scale without switching mode. Based on the relative starting cursor position to the geometries, one of these basic operations is automatically chosen. In addition to this, based on proximity, a line segment snaps to other existing points or aligns with other existing line segments. While this approach provides a way to avoid switching mode, this is limited to only line segments. In the GuTS system, we apply different methods to avoid mode switching.

Igarshi, et al. [60, 59] showed a drawing system using prediction and cleanup for rapid creation of precise drawings. From the user’s free stroke the system identifies the appropriate constraints, creates a precise geometry progressively, and beautifies automatically. The user’s straight line strokes and corresponding constraints allow this system to draw line segments only. Unlike this technique, the tracing technique in the GuTS approach creates any forms, including curves and surfaces, rapidly and precisely - not just line segments.

2.1.3 Drafting Approach

Singh [91] presented a digital version of the drafting process through the use of French curves. In this research a methodology is provided to represent French curves in a digital format using elliptical arcs. Precision and fluency are achieved by mimicking the traditional drafting process. This research focused only on French curves and elaborated a way to create French curves digitally, which then uses them for tracing. Even though this research is limited to French curves, it provides some initial ideas about the tracing operation for curves and furthers our research directly by considering complex guide curves as well as guide surfaces. In the GuTS approach, defining and tracing is only a subset of the modeling approach.

2.1.4 Tracing Approach

In the commercial modeling system called Maya™ [73] from AutoDesk®, NURBS-based curves are drawn on top of the NURBS surface to select a region. While this approach is similar to our research, there are considerable differences. First, our research used curve modeling through a tracing operation which is not available in Maya™. Second, we used polygonal mesh instead of a NURBS or subdivision surface representation. Finally, our research showed multiple geometries can be used together directly during the tracing operation but in this commercial system only one surface geometry can be used at a time.

2.1.5 Pattern Matching Approach

Arvo et al. [15] described “Fluid Sketches” for recognizing predefined basic entities automatically from the user’s raw input strokes. Baudel [20] discussed shape matching, where the user sketches spline curves through direct strokes rather than control points. Yang et al. [99] used sketch-based modeling to create 3D parameterized objects from 2D sketches. In this system the input sketch is matched to existing 2D templates - requiring prior domain knowledge. In these approaches the user’s strokes identify the desired shape and easily create the geometries. Both the above approaches are very limiting since they recognize only a set of predefined curves and do not extend for any general freeform curve or surface modeling.

2.1.6 Gesture and Sketch-Based Approach

Some systems achieve fluency by constructing 3-dimensional (3D) shapes automatically from the input of 2-dimensional (2D) freeform strokes. Cohen et al. [38] described an approach of creating spatial curves from two 2D input drawings - one for the curve as it looks from the current viewpoint and the other for its shadow on the floor. While this approach allows the creation of 3D curves rapidly by inferring the input strokes, it does not help in modeling precise geometries. In “Sketch” [100] a gesture-based interface allowed rapid modeling of CSG-like models consisting of simple shapes. Although this system provides precision through constraints, it is suitable only for simple shapes rather than complex freeform geometries. In “Teddy” [61] plausible 3D polygonal surfaces are created directly from the input of freeform planar strokes. This approach is easy and rapidly constructs 3D shapes, but it creates approximate rather than precise geometries. In the GuTS approach precision, fluency, and rapitidy is achieved through guided trace, automated snapping, and stitching processes.

In recent years, as pen and tablet computing systems became widely and cheaply available, sketch-based modeling became more popular. Due to the increased popularity of sketch-based modeling in recent years, a dedicated full sketch-based modeling course [68] was given at Siggraph - one of the largest international graphics conferences. Several sketch-based systems were developed in recent years. Ijiri et. al. [62] used a sketch interface for modeling 3D flowers and leaves. Karpenko et al. [63] discussed the “SmoothSketch” system that creates 3D freeform shapes from complex sketches of visible contours of shapes. Kho and Garland [64] used sketching for interactive deformation of unstructured polygon meshes. Watanabe and Igarashi [95] used a sketch-based interface for terrain modeling. Nealen et al. [75] used a sketch-based interface for detailed mesh editing. Schmidt et al. [87] used sketch-based modeling with a blob tree to model hierarchical implicit models. All of these examples highlight the significance and popularity of sketch-based modeling, something we use in the GuTS approach.

Oh et al.[80] discussed a concept system called “SESAME” to sketch, extrude, sculpt, and manipulate geometries easily. This system only works with simple geometries and straightline extrusion. Therefore, it is not suitable for freeform surface modeling. Mizuno et al. [74] presented a user interface for a virtual sculpting system where a pressure sensitive pen carves a workpiece. This approach depicts the real world chiseling process in a virtual space, but it is not suitable for freeform modeling.

2.1.7 Freeform Modeling Approach

Some of the early research, such as Sederberg et al. [90], described a freeform deformation (FFD) approach to solid geometric models (represented in CSG or B-Rep). Terzopoulos et al. [94] described Dynamic-NURBS (D-NURBS) as a way to directly manipulate through simulated forces and local and global constraints. This is done by incorporating mass distributions, internal deformation energies and other physical quantities into the regular NURBS geometric substrate. Borrel et al. [32] described a new way of modeling freeform deformations on surfaces using constraints called “Simple Constrained Deformations”. Constraints in this approach are posed on a local region of a surface which is modified such that the defined constraints are satisfied. Biermann et al. [26] described a method of computing approximate results of boolean operations applied to freeform solids bounded by multiresolution subdivision surfaces. Biermann et al. [27] discussed a cut-and-paste tool to copy geometric features from one surface to another on multiresolution surfaces, but this has limited usefulness due to constraints on the type of shapes and the lack of real-time interaction. Zwicker et al. [108] presented a system called “Pointshop 3D” that allowed interactive shape and appearance editing of 3D point-sampled geometry by generalizing conventional 2D pixel editors. The above research - unlike the GuTS approach - requires a thorough understanding of geometric modeling and underlying geometric representation to create complex freeform geometries rapidly and precisely.

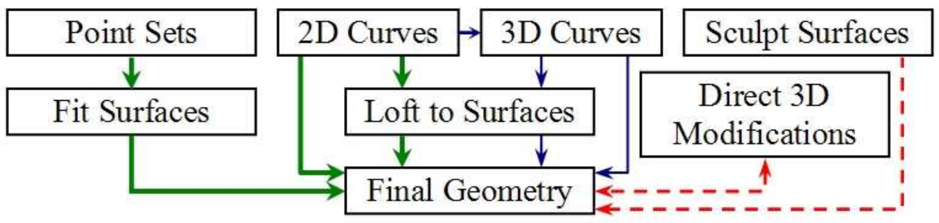


Figure 2.1 Curve and surface design

2.1.8 Blob Modeling Approach

Markosian et al. [72] described a system called "Skin" to create freeform shapes using a particle-based surface representation that resembled blob modeling. The system allowed a wide range of skeleton shapes, creases and multi-resolution through subdivision representation. Interesting smooth freeform surfaces are easily created from simple input shapes. Due to the nature of blob modeling, this approach - unlike the GuTS approach - produces only approximate geometries and is not suitable to create precise freeform shapes.

As a summary: Fig. 2.1 shows the current widely used methods for designing curves and surfaces. The common approaches are: to fit surfaces to point sets, to create curves and loft surfaces using curves, or to construct surfaces directly in 3D. In Fig. 2.1, the thick solid lines indicate precise design is possible; thin solid lines indicate precision can be achieved in some cases; and dashed lines indicate precision is difficult (and may not be possible) to achieve through current modeling approaches. Generating point sets through scanning requires real physical objects, which is often not possible in designing new shapes. The approach of creating curves and constructing the surfaces (by sweep, loft, blend, etc.) from these curves is the current well-known approach to achieve precise modeling. As discussed above, it is done using control point manipulation and it lacks the required fluency and direct approach. So far, direct 3D manipulation provides only approximate geometries and manipulations, despite its fluent and direct approach. Thus there is always a trade-off between precision, fluency, and rapidity.

Figure 2.2 shows the layout of designing curves and surfaces in GuTS, where it allows creation and manipulation of curves and surfaces directly with precision. Solid thick lines denote precision is maintained during these processes.

2.2 User Interaction Approaches

As computational power increased, the complexity of geometric modeling and the need for an effective user interface also increased. Hence, researchers started to look beyond once sufficient I/O devices, such as the keyboard, mouse and monitor that only provide 1-Dimensional (1D) and 2D interaction, to a new level of user interaction.

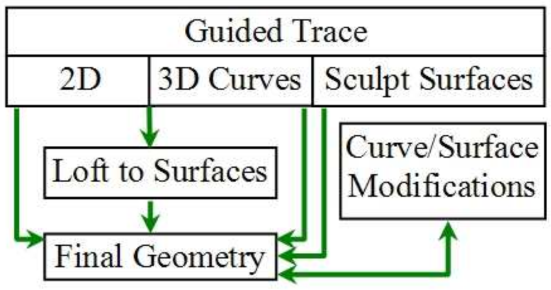


Figure 2.2 Modeling in GuTS

2.2.1 Freeform Modeling Using 6 DOF Input Devices

Since most geometric modeling is in a 3D environment and at least 6 DOF need to be controlled, some research focused on using 6 DOF 3D input devices (such as Polhemus [8], Ascension 3D trackers [2], Logitech 3D mouse [4], Space Ball) to directly interact in 3D. Some of these interfaces allow the user to define geometries directly in 3D using 6 DOF tracking input devices. Because 6 DOF input devices work considerably differently than a conventional 2D mouse and keyboard setup, they give researchers opportunities to think in different ways for modeling freeform geometries. The 3-Draw system of Sachs et al. [85] created 3D curves using a 6 DOF tracking device and constructed 3D surfaces by fitting a surface through the linked curves. Stork et al. [92] extended this approach to create and edit freeform NURBS surfaces directly using 6 DOF input devices. Some systems, such as “HoloSketch” of Deering [41], Schkolne et al’s [86] ‘surface drawing’ and Bill et al’s [28] research, allowed the user to sculpt the polygonal objects directly using 3D trackers, hand motions and virtual tools. Korida et al. [66, 78, 77] described the use of gesture input and 6 DOF input to simulate pottery modeling as in real-world ceramic art. Balakrishnan et al. [18] presented a computerized simulation of the tape drawing process typically used in the automotive industry. Badler et al. [17] used an absolute 3D device to manipulate and position 3D objects for kinematic structures. Due to the ability and nature of these 3D input devices, sculpting and modeling directly in 3D became feasible, direct and easier. Because of imprecise human input and the imprecise nature of these input devices, only abstract and approximate models are constructed using these approaches. Hence, such systems lack the required precision that is demonstrated in the GuTS approach.

2.2.2 Customized Special Purpose Higher DOF Input Devices

Rekimoto et al. [82] used a new type of input device called the “Tool Stone”, a cordless multiple DOF input device that enabled the issue of multiple commands by sensing physical manipulation of the device itself. Hinckley et al. [55] and Pierce et al. [81] used a small doll of the actual manipulation shape, attached to 6 DOF input devices, to interact naturally in a 3D environment directly. While focused on new input interaction mechanisms for issuing input commands, manipulating objects and viewpoint controls directly in 3D, they do not provide any solution for precisely manipulating objects in the 3D environment.

Balakrishnan et al. [19] used a novel bend and twist-sensitive input strip called ShapeTape™ attached to a 6 DOF input tracker to create and manipulate 3D shapes directly in 3D. Using both hands, the shape tape is manipulated in the real-world and by using this tape's shape and twists, freeform curves and surfaces are created. This approach does provide a direct and easy approach to create freeform geometries, but as in other higher DOF input devices, this too lacks precise input. It is this lack of precision that the GuTS approach addresses.

2.2.3 Experimental Evaluations of Input Devices

Several researchers, including Hinckley et al. [54, 52, 53], and Zhai et al. [102, 101, 103, 104, 105], presented a detailed study, experimental analysis, user performance and the importance of 6 DOF input devices in 3D interaction. The overview of multiple input devices lets us analyze the plethora of options for input devices. The studies and evaluation of multiple input devices and interaction mechanisms help us eliminate many input devices, based on their nature and unsuitability for precise manipulation in 3D. Some observations from these studies, such as manipulation of input devices that are controlled by fingers are more precise than other input devices that are manipulated through the palms or arms, aid us in choosing input devices accordingly since precision is an important factor in our research.

2.2.4 Two-Handed Approach

To further improve user performance and enable fluent interaction, Cutler et al. [40], Leganchuk et al. [69], Buxton et al. [36], and Balakrishnan et al. [19], used two-handed direct manipulation in a 3D environment. In these cases, the dominant hand provides precise detailed input while the non-dominant hand is used for coarse input. Their research highlights interesting and important aspects of the two-handed approach, such as the ability to complete a difficult interaction through coordinated and asymmetric bimanual inputs. From the experiment results they observed that bimanual techniques are significantly faster than uni-manual manipulation and bimanual interaction may be driven by factors other than simple time-motion performance advantages. Llamas et al. [70] discussed a system called "Twister" that allows two-handed editing of 3D shapes by applying orientation changes and rotational constraints at each displaced point. Grossman et al. [49] used two-handed and multi-finger approaches for gestural interaction with 3D volumetric displays. Yan [50] discussed the simultaneous use of two-handed input and multiple fingers through touch-sensitive screens based on the *Frustrated Total Internal Reflection* (FTIR) phenomenon. In the GuTS system, the two-handed approach makes use of the benefits observed in this research. Also, we extend the user interaction further by adding other input/output modes (multimodal interaction) that allow the GuTS system to be used fluently, directly and rapidly.

2.2.5 Multiple Input Modalities Approach

Bourguet et al. [33], Arangarasan et al. [12], Grasso et al. [48], Billinghurst et al. [29], and Houde [57] all used speech interface and other modes of input jointly so as to manipulate graphical objects directly and rapidly. Arsenault et al. [14] used force feedback mechanisms by using a force feedback device (such as the Phantom device [7]) to provide additional feedback to the user and to interact effectively in the complex 3D environment. All of this research emphasized the use of multiple modes and established the benefits of multimodal interaction. The research performed by Cohen et al. [39], Bowman et al. [34], Blattner et al. [30] and Brooks [35] created an understanding of the broad range of performance and characteristics of interaction mechanisms. Their research also displays the importance and effect of multimodal user computer interaction in 2D and 3D environments.

The above mentioned research not only applied a set of modalities for approximate 3D modeling and manipulation, but also for other different applications emphasizing the significance of multimodal interaction. However, none of them focused on the use of multiple modes simultaneously to enable directness and rapidity in precise freeform modeling. We apply multimodal interactions such as two-handed input, a 6 DOF input device, direct hand-eye coordination through a pen-based data tablet display, voice input and synthesized speech output for effective user interface in the GuTS system.

Chapter 3

GuTS - User's Perspective

In this chapter we present the user's perspective of the GuTS approach. The steps involved in the GuTS approach will be stated. The various operations will be discussed, including their contributions to precision, fluency, and rapidity.

Figure 3.1 shows the sequence of operations and several stages of geometries in the GuTS modeling approach. New pieces of geometries are created mainly by tracing along the guide geometries. Then, further tracing along the guide geometries extended these existing old geometries to form complex shapes. In addition, as an alternative approach, existing multiple pieces of geometries are directly stitched together to form complex shapes. By performing these operations multiple times, the desired final geometry is attained.

In GuTS, guide geometries are the geometries that are used for tracing and generating new geometries. The guide geometries and other regular geometries in the modeling session use the same geometric representation. The difference is guide geometries allow the user to trace over them and they also trigger certain snapping operations. Any regular geometry can be converted to a guide geometry and vice-versa. To differentiate between these two, in this thesis, we refer to guide geometries as 'guide geometries' or 'guide shapes' and regular (non-guide) geometries as simply 'geometries'. In curve modeling guide geometries are also referred to as 'guide curves' and in surface modeling as 'guide surfaces'.

The GuTS approach itself is independent of the underlying geometric representation. In our implemented GuTS system, we used interpolation subdivision curve and 3rd degree Bezier curve representations for curves. The user can select either one of these representations in a modeling session. Surfaces are represented by a tessellated mesh representation.

In short, the main steps are as follows:

1. Generation of guide geometries
2. Trace pieces of geometries from guide geometries
3. Position the guide geometries and other geometries precisely (absolute global position, relative position to other existing geometry - curve modeling only)
4. Automatically stitch the pieces of geometries together

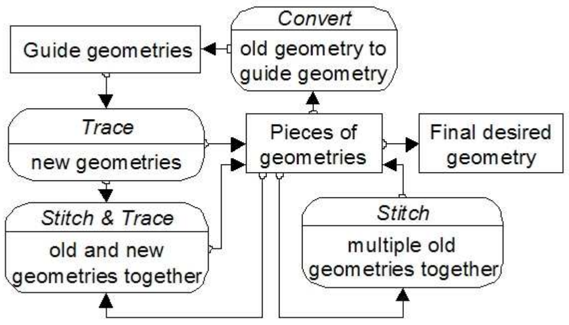


Figure 3.1 Sequence of operations and stages of geometries in GuTS

Both curve and surface modeling use the same basic paradigm – trace and stitch – for creating complex geometries. In the following sections we discuss each of these topics in detail that are specific to curve and surface modeling respectively.

3.1 Curve Modeling

3.1.1 Generation of Guide Curves

In GuTS, new curves are created by tracing over the guide curves. Guide curves can be of any shape. Any existing curve in the modeling session can also be a guide curve and used for tracing. This flexibility allows GuTS to easily adapt to create a wide range of complex shapes. In our current implementation of the GuTS system, the guide curves are generated mainly through scanning and converting existing geometries.

3.1.1.1 Scanning

The main goal in selecting guide curves is to choose a limited set of basic general-purpose curves that represent a wide variety of curves. One such example is a set of French curves and ship curves (as shown in Fig. 3.2) that covers a wide range of curves for specific application. These curves can be used as a template to produce several smooth curves (as performed in the conventional drafting process). In our GuTS system, French curves are scanned and the outlines are stored as a set of points. Since interpolation subdivision schemes (discussed in Chapter 4) are one of our geometric representations to present curves, limited sets of initial points are sufficient to easily attain the required smoothness of the curves. If required, generation of French curves digitally can be adopted as suggested in [91]. Currently, this scanning and storing of the set of points from the guide templates is performed manually to show that the guide geometries can be scanned and used effectively in the GuTS approach. This process can be automated with little effort in order to scan a large number of guide geometries. Figure 3.3 shows the actual process involved in scanning guide shapes. Figure 3.3(a) shows the scanned image of a real French curve. Figure 3.3(b) shows a small

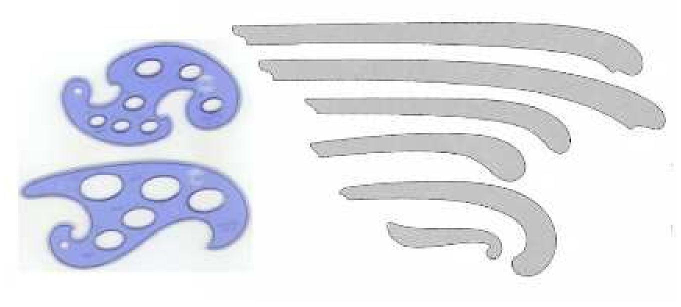


Figure 3.2 Generic guide templates - French curves and Ship curves

set of scanned points on the boundary of the French curve. Figure 3.3(c) shows the scanned Bezier curve (and its corresponding Bezier control points) on the boundary of the French curve. Figure 3.3(d) shows additional points are included for interpolation subdivision representation. Figure 3.3(e) shows the resulting French curve, and the final French guide-curve is shown in Fig. 3.3(f).

3.1.1.2 Converting existing geometries

Another way of populating the guide geometries is to convert the existing geometries to guide shapes. Since regular geometries and guide geometries use the same geometric representation, converting the geometries back and forth is straight forward. Using this approach, multiple geometries can be combined together to easily form complex guide shapes. Fig. 3.4 shows some of the sample guide curves that are used in the current GuTS system.

3.1.2 Drawing and Tracing Curves

The primary mechanism for the user to create curves in the GuTS system is to draw them, using a pen-based input device. Just as with paper and pencil drafting, more precise shapes are created by using guide shapes and tools that help position the pencil precisely. As haptic feedback provides precise guiding in the real world, precision is achieved using gravity fields [93] that attract the cursor towards the guide shape when the pen is near the guide shape. Then the user traces the desired portion of the guide shape, as shown in Fig. 3.5. Details about snapping mechanisms are discussed in the following section.

Guide curves can be of any shape and type (such as line, arcs, or freeform curves). The traced geometries inherit the properties of the guide shape and are presented in the same geometric representation. The complexity of the guide curve does not affect the interaction, performance or the approach of tracing. For example, the cursor also snaps to existing curves, making it easy to connect to them. Guide shapes are transformed as needed by the user so that curves can be placed as needed. Our preferred mechanism for positioning guides is to use a 3D mouse in the non-dominant

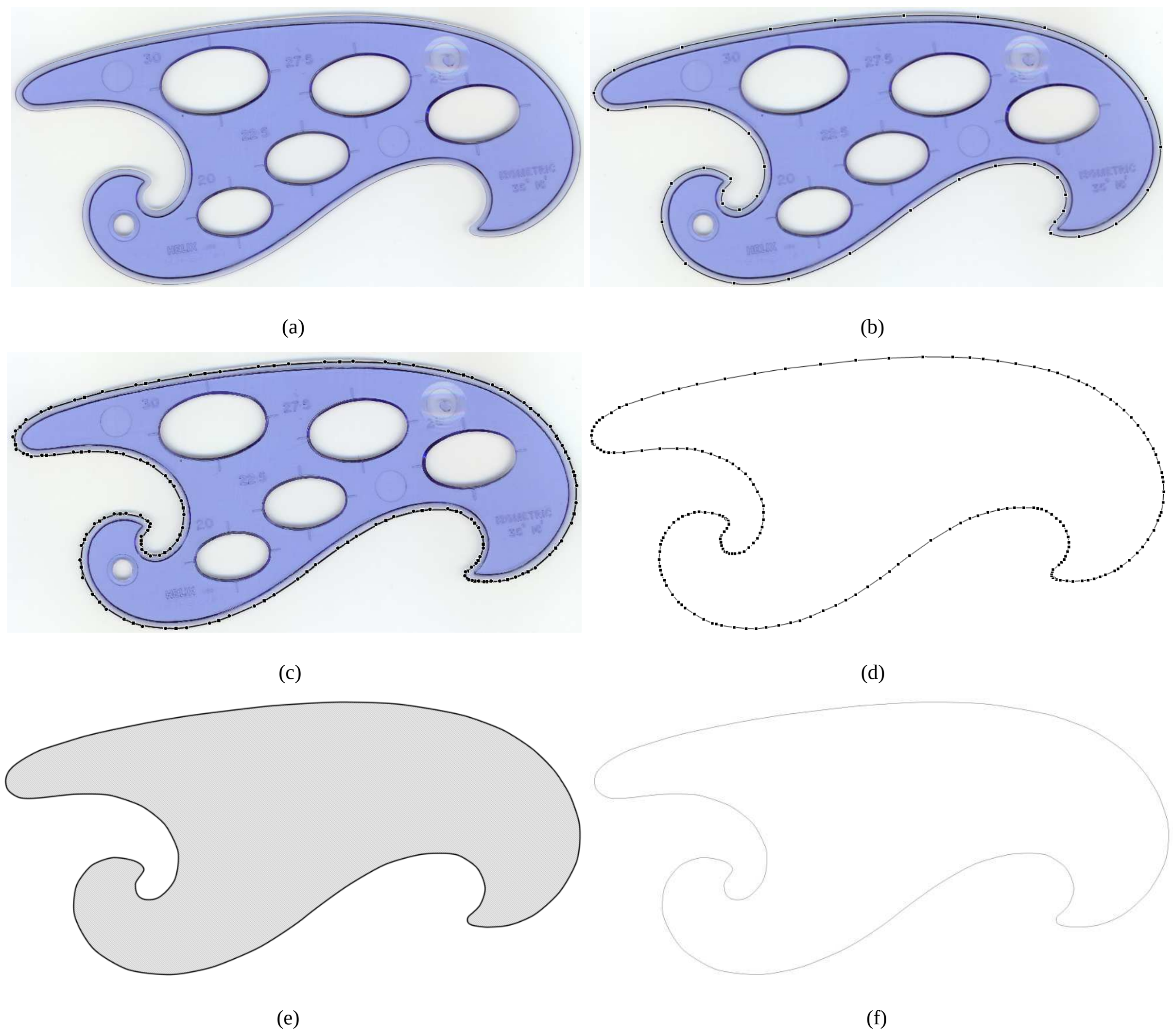


Figure 3.3 Scanning objects to create guide shapes

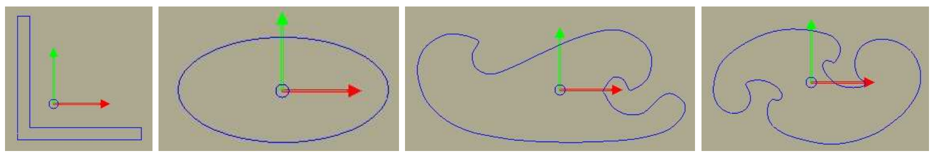


Figure 3.4 Sample guide shapes

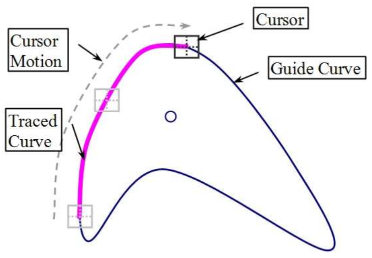


Figure 3.5 Tracing from guide curve

hand, allowing the user to position, rotate and scale the guide curve while tracing it with the dominant hand using a pen tablet. This directly mimics the drafting process.

3.1.3 Precise Positioning and Stitching

In the GuTS approach, snapping is used extensively. Several snapping mechanisms provide the required precision to perform several tasks. The snapping mechanisms in the existing modeling systems are primarily the mouse cursor snapping to geometries. In the GuTS approach, while cursor snapping is used, several other snapping mechanisms are also introduced. In the GuTS approach, the whole geometry becomes a snapping object and each guide geometry snaps to other regular geometries. This whole geometry snapping allows different types of snapping in the GuTS approach. They are: one-point snap (pivotal snap), two-point snap, fixed snap, and slide snap. Each of these snap mechanisms are discussed below in detail.

The primary tasks in snapping are: positioning the cursor location for tracing; positioning the guide curves and regular curves; and positioning the curves absolutely or relatively to other curves. As seen in the previous section, snapping the cursor to the guide shapes provides precision during the tracing operation.

To create precise relationships between segments, one must position the guides precisely. To assist with this, GuTS provides a set of snapping operations for positioning the guide curves. Because the end points of existing curves exert gravity, it is easy to extend a previously drawn curve piece without introducing a gap, as shown in Fig. 3.6. These snapping mechanisms are variants of gravity fields. When a guide curve is close to a relationship with existing curves, it snaps to a precise position that creates the desired relationship. Once snapped, this relationship is maintained as the user further manipulates the guide, until the guide is pulled away from the precise relationship. For example, after the user draws a segment of a curve, the user can position the guide curve at a different place to extend this curve. If a guide is moved close to an end of an existing segment, it snaps to connect to this point.

Figure 3.7 shows different snap mechanisms that are introduced in the GuTS approach. The snapping of a guide to a position is the basis for several types of snapping operations. The basic form, which we call a one-point snap, provides the starting point for more complex snapping operations. Once the guide is snapped using a one-point

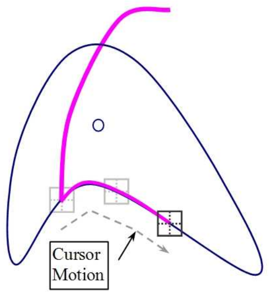


Figure 3.6 Tracing further

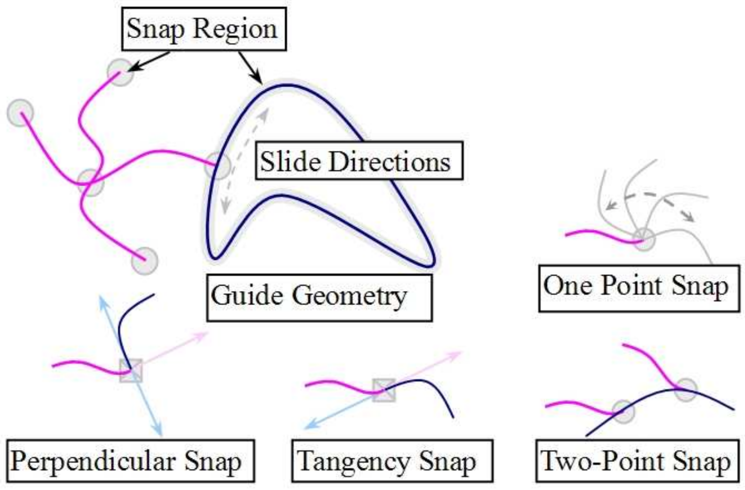


Figure 3.7 Gravity-based point, slide and fixed snaps

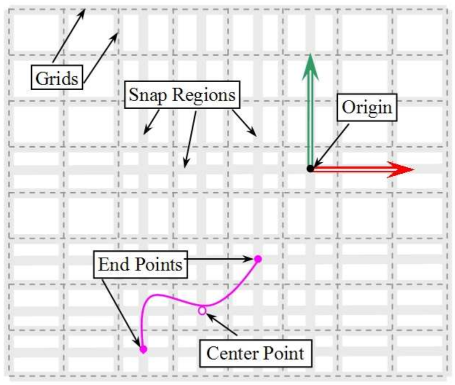


Figure 3.8 Grid based snapping

snapping operation, the guide can be further manipulated in ways that preserve the point connection. While one-point snapped, the guide can be rotated around the snap point (called a pivotal snap) or translated such that the connection is maintained by sliding along the guide curve direction. The point connection established by the one-point snap is maintained. Mechanisms for establishing precise orientation relationships add to the one-point snap. For example, as the user rotates the guide, if the orientation of the guide brings it close to either being tangent or perpendicular to the existing segment, the guide snaps to establish this relationship precisely.

The two-point snap also sets the orientation of a guide after it is snapped to an end of the existing geometry. As the user rotates the guide curve around the snapped point, if the guide comes close to another snap target point, the guide curve is rotated so that the second point snaps to the target point. This makes it extremely easy to position guides to connect two curves or two different target points. Different snapping mechanisms - pivotal snap, slide snap, and two-point snap - are novel snapping mechanisms and are secondary contributions of this research.

Traditional grid-based snapping as shown in Fig. 3.8, though not novel, is also used in the GuTS system. Grid-based snaps are used in the GuTS system to achieve absolute coordinate precision. With all of our snapping mechanisms, the stickiness of the snap is important. Snaps must be sticky enough to make it easy for the user to establish the relationship without positioning the guide precisely. The snaps must also be sticky enough to maintain these relationships, without being so sticky that the user cannot pull the guide away from the curve. For example, after the initial one-point snap, small motions tangent to the guide are interpreted as sliding the curve but still preserving the snap, while larger motions normal to the curve pull it away from its initial snap. The issues of engaging and disengaging snaps become even more important in cluttered drawings where an increased number of potential snap sites may cause an increased number of unwanted snaps. This clutter problem is an issue in any snapping interface and has been addressed in previous systems by locking mechanisms. Several other concepts, such as selective geometry manipulation

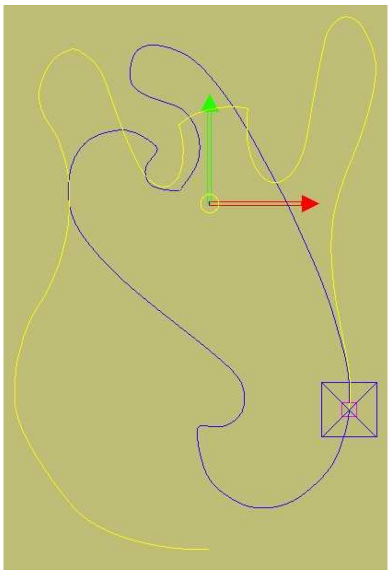


Figure 3.9 Digitized french curve in the GuTS system, as user tracing a guitar sketch

and layered geometries, can also be used to avoid the clutter problem. More details about this are discussed in Chapter 7.

Figure 3.9 shows the actual snapshot of the tracing operation in progress. The crossed box at the point that connects the guide curve and the regular curve represents the tangent snap established between these geometries. As a summary, some of the earlier research [93, 25, 24, 23] described simple snapping mechanisms such as gravity field and conditional snaps (tangential, perpendicular, etc.). In this research, we introduced new additional snapping mechanisms such as pivot snap, two-point snap and slide snap that provide the required precision during several geometric operations.

3.1.4 Automatic Continuity Detection

When multiple curves are joined together it is essential to provide the desired continuity between the curves. The snapping mechanisms provide an indication of the types of continuity that the user desires. In cases where snapping is not used, automatic continuity mechanisms are employed to establish relationships between segments.

The automatic continuity mechanism automatically detects the suitable continuity condition based on the angular difference between the two curves and their curve types. For example, with a subdivision curves representation, when a C^1 continuity curve is stitched to another C^0 continuity curve, it establishes the C^0 condition at the stitching point and the resultant curve also will have C^0 continuity. In another example, when a C^1 curve is stitched to another C^1 curve and the angular difference between these two curves is below a certain threshold level, it produces C^1 continuity at the stitch point as well as all along the resultant curve. On the other hand, when two C^1 curves are stitched together and

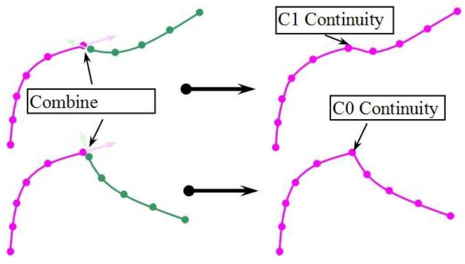


Figure 3.10 Auto continuity detection

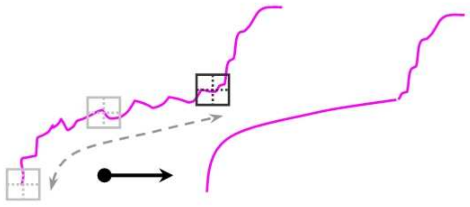


Figure 3.11 Smooth operation

the angular differences are above the threshold level, then it produces C^0 continuity at the stitch point but maintains C^1 continuity everywhere else on the curve. Figure 3.10 shows the process of automatic continuity detection when two curves are connected together by different scenarios. This insures that the desired continuity is maintained at the stitched region as other drawing operators, such as the refinement operators (discussed in a later part of this chapter) are applied. In a Bezier curve representation, at the connection points, depending upon the settings of the geometries, either C^0 or G^1 or C^1 continuity is created.

3.1.5 Other Geometric Features and Operations

The geometric operations and features described in the previous sections provide a way to create curves rapidly and precisely. For completeness of the GuTS system, we used several geometric features and operations. In this section we briefly discuss those geometric operations.

3.1.5.1 Smoothing

Freehand drawing is an easy solution when a user wants to fluently create some random freeform curve that cannot be easily and uniquely defined by a set of guide shapes. Drawing smooth, freehand curves without a guide is difficult: curves created by freehand drawing always contain wiggles due to imprecise human input and sampling issues. A smoothing tool is essential to make freehand curve sketching useful. Our smoothing tool applies to the curve locally by removing wiggles at any specified curve region (refer Fig. 3.11). In our current implementation, a weighted n-point smoothing algorithm is used for smoothing the curve. In this approach one vertex is adjusted at a time without changing element topology but improving element quality. In the interactive mode, as the user moves the cursor along the curve, the closest point is adjusted. Thus the fluency and smoothness of the freehand drawing are maintained.

3.1.5.2 Gestures

Though the tracing mechanism can be used to create basic quadratic shapes (such as circles, arcs, ellipses), it can be inconvenient as it requires the user to select the guide shape and then trace the entire shape. To facilitate the creation of commonly needed shapes, we have introduced a simple gesture-recognition feature to identify basic shapes such as a line, circle, ellipse, and arc, directly from freehand scribbled strokes. Our gesture-recognition algorithm subdivides a scribbled input into a set of regions and considers the distribution of points in each cell. Based on the distribution of the points, the final basic primitive is derived. The details about how the primitives are derived from the freeform sketch are provided in Chapter 4. Figure 3.12 shows simple scribbled drawings for a circle and a line, where the region is divided into basic 3x3 cells. The same logic can be extended to subdivide the cells further for more efficient and precise recognition of shapes. Though this method cannot be used to recognize any generic shapes, basic shapes such as a circle, line (horizontal, vertical, inclined), ellipse, and arc are well recognized. This feature is especially useful when creating basic shapes rapidly. In the GuTS system, the user selects the scribble mode to activate the gesture recognition feature. Once the gesture recognition feature is turned ‘on’, the user’s scribbled input is used to recognize the basic shapes rapidly. This ‘on/off’ feature helps to differentiate between regular tracing and gesture scribble input. The basic shapes created from the scribbled input are like any other geometries - those can be used as guide shapes and use them for further tracing and stitching operations.

3.1.5.3 Level of detail

Often, it is more convenient and computationally inexpensive to work with curves at low levels of detail. Such multi-resolution work is well supported within the GuTS system when subdivision representation is used. For one, the guide shapes can be provided at varying levels of resolution, allowing specification at any point. With subdivision schemes, different levels of detail are easily achieved. This feature enables the user to create a complex curve with different levels of detail at different portions of a curve. For example, when a guide shape at different levels of detail is used to trace a different portion of the curve, the resulting curve will contain these varying levels of detail. Though the level of detail can also be controlled automatically based on the curve properties, this feature provides the user with an additional control to modify the curve as desired.

Figure 3.13(a) shows a coarse curve that contains fewer vertices. As the curve is refined to the next level, as in Fig. 3.13(b), it becomes smoother and refining further produces the smooth curve shown in Fig. 3.13(c). After repeating the refining process several times, the changes between two different levels become negligible and the final

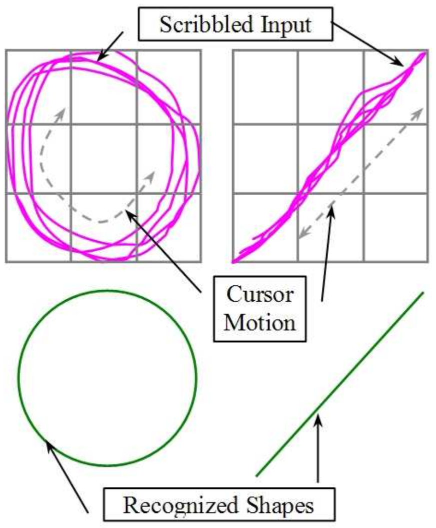


Figure 3.12 Basic gesture recognition

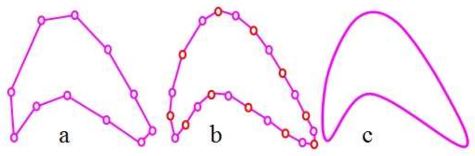


Figure 3.13 Course / refine operation

curve is called the limit curve, as shown in Fig. 3.13(c). The geometric operations and manipulations are easily performed at the coarse level because fewer vertices are involved. Once the operations are completed it can be refined to the required smoothness.

3.1.5.4 Cut / trim

In most cases the user can create a complex shape by only adding pieces of curves together. Sometimes just by removing a portion of a curve helps to get the desired shape easily and potentially avoids several stitching processes. In the GuTS system, two types of cuts are provided: ‘point cuts’ and ‘region cuts’. ‘Point cuts’ are done to split the curve and ‘region cuts’ are performed to split the enclosed region of a curve into pieces. Both cuts are illustrated in Fig. 3.14. When performing the region cut, guide shapes can also be used to precisely trace the path of the cut in the same way a curve is traced during creation.

3.1.5.5 Vertex-level operations

Inspired by the availability of the point-set given our subdivision implementation, there is a whole new range of operations that can be performed on the curves. We allow the user to perform operations on the individual points that make the curve. One example is vertex-level editing. Vertex-level editing allows some interesting operations such as drag and twist (vertices), erase (a vertex or edge), and change continuity of a vertex (to create a crease). These features are especially interesting since this allows control of the curve locally. Figure 3.15 shows a vertex-level twist and drag operation on a curve. User selects a portion of the curve and then the selected portion of the curve is twisted and dragged to modify the shape of the curve locally. Figure 3.16 shows another example of vertex level operation - vertex and edge erase. This feature allows to erase part of the curve locally.

3.1.5.6 Intersection

In the GuTS approach, one of the most commonly used operations is finding the intersection of curves. As the user moves the geometries, the intersecting points of geometries are calculated and displayed to user in real-time. To achieve interactive system performance, intersections are computed at two stages as shown in Fig. 3.17. Initially the approximate intersections are computed and displayed to the user. Since multiple curves are used in the GuTS

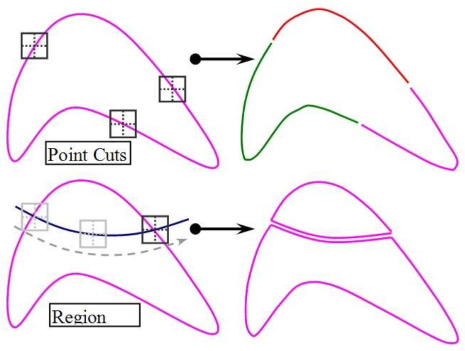


Figure 3.14 Cut operation (point cut and region cut)

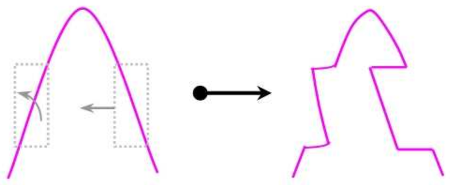


Figure 3.15 Vertex level twist and drag

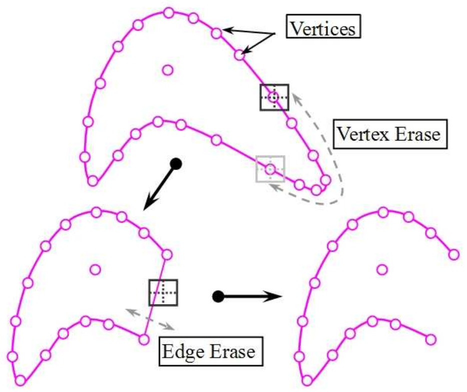


Figure 3.16 Erase (vertex and edge erase)

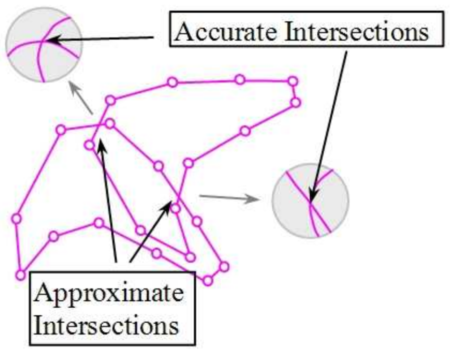


Figure 3.17 Multi-level intersection

system, these approximate intersection points are dynamically and automatically computed, and displayed to the user. When the user refers (or tries to use) any intersection point for any further operation, it computes the exact intersection by recursively refining up to the desired accuracy. This two-stage intersection process reduces the computation time considerably and computes all the intersection points, even with complex shapes, during real-time interactions. This two-stage intersection process is performed transparently to the user without any user intervention. The user will not notice this two-stage operation during the modeling session. The calculation and display of intersection points in the GuTS approach allows the user to trace between multiple geometries without interruption. This feature also provides a WYSIWYG (What You See Is What You Get) interface for tracing over multiple geometries.

3.1.6 Dimensional Tags

Dimensional tags are a set of origins (oriented points) that interactively display the dimensions between any user-preferred position and the current location of the guide geometries (and/or the current input position). This feature allows the user to interactively identify the dimensions with respect to several locations simultaneously. In Fig. 3.18, two dimensional tags - one at the origin of the drawing space and the second at an intersection point - are placed. As the cursor moves, the dimensions are displayed with reference to these two dimensional tags. These tags can be placed at any desired position. This feature allows simultaneous viewing of multiple dimensions measured from different points. Other than Cartesian coordinate dimensions, as shown in Fig. 3.18, other types of user-customized dimensions (such as angles and radial distances) can also be displayed.

For the completion of basic geometric and manipulation operations, several other generic operations such as copying, deleting, scaling, and mirroring are also performed in the same way as in any other modeling program.

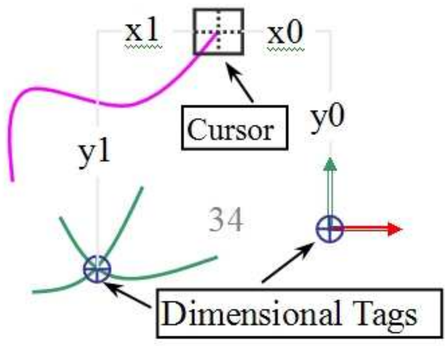


Figure 3.18 Dimensional tags

3.1.7 Example - Guitar Model

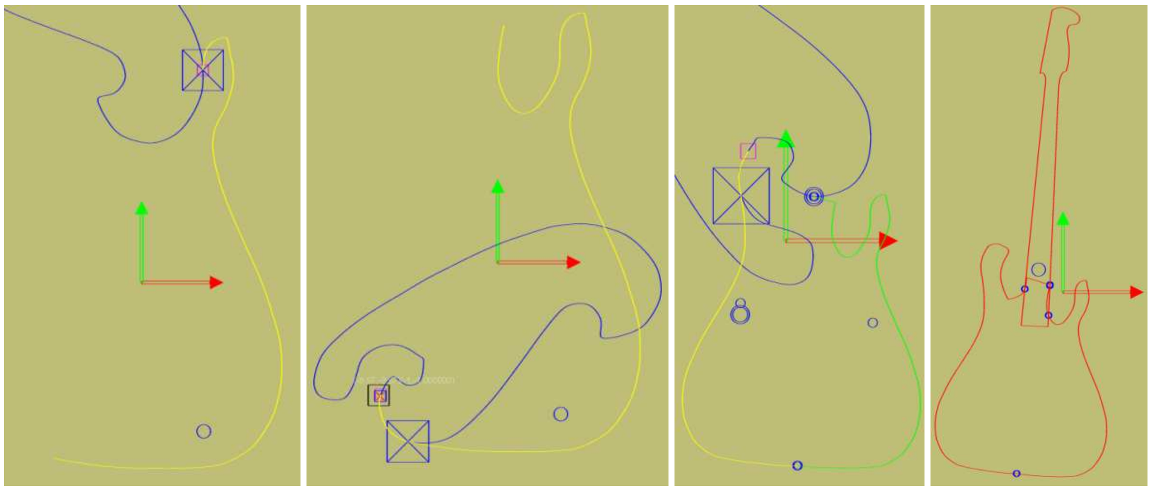
Figure 3.19 shows the process in creating a guitar model in the GuTS system. Figure 3.19(a) shows tracing the body of a guitar using a (French curve) guide geometry. Figure 3.19(b) shows tracing further the guitar body. Figure 3.19(c) shows after the right half mirrored to form the left half of the body, and then the top left part of the guitar body is being modified - part of the geometry is erased and new portion of the geometry is being traced. Figure 3.19(d) shows the completed guitar model. Several features that are discussed above are used to complete this guitar sketch. In several cases once the drawing of curves is completed, they can be used to create 3D geometries using conventional operations such as extrusion, revolution, and sweep. Thus the curves created in the GuTS system are effectively used to create freeform surface geometries.

3.2 Surface Modeling

In the GuTS approach, surfaces are modeled using similar approaches as described in the curve modeling section. Several of the features, such as guided trace, stitch, intersect, and smooth, performed in curve modeling are extended to surface modeling. In the GuTS system, surfaces are represented by tessellated mesh.

3.2.1 Generation of Guide Surfaces

Unlike French curves or ship curves that cover a wide range of curves, there are no generic surfaces that cover a wide variety of surfaces. So, in surface modeling, the guide surfaces are created primarily in three different ways. First, basic primitives such as block, sphere, cone, torus, etc. are used. Second, surfaces are used that are created from curves through geometric operations such as extrusion, revolution, sweep, etc. Third, surfaces that are imported as triangulated mesh models created in other modeling systems are used.



(a) (b) (c) (d)

Figure 3.19 Creating a guitar model

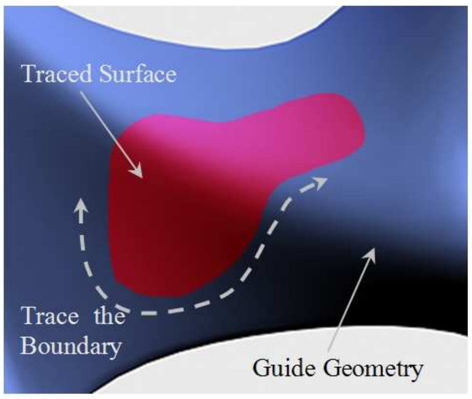


Figure 3.20 Point trace over guide surface

3.2.2 Tracing Surfaces

Surface patches are created in GuTS mainly by tracing over guide surfaces. There are two types of tracing: point tracing and thick-brush tracing.

3.2.2.1 Point tracing

Point tracing is performed as shown in Fig. 3.20. Using the pen tablet, a boundary is traced over the guide surface. As it is traced, a curve is drawn over the surface and which then selects a region of the surface geometry. The desired surface patch of a region on the guide surface is produced. This selection is primarily done in two steps: using a point pen to draw a curve directly over the guide surface and then creating the inner surface patch (bounded region) once the curve is closed.

3.2.2.2 Thick-brush tracing

In the second approach, a thick brush is used to paint over the guide shape to select regions of the guide surface, as shown in Fig. 3.21. Both these approaches produce the same type of resulting surface patch, but based on the shape of surface to be created, each approach serves its own purpose.

Though in both the above cases the guide surface is used for guiding, the boundary of the patch is not precise. The precision of the boundary as well as the inner surface region is achieved using the following two approaches. In addition to guide surfaces, guide curves are also used to achieve precise boundary. As shown in Fig. 3.22, the guide curve is placed on a plane (usually perpendicular to the view direction) and projected onto the guide surface. Then the user moves the guide curve as discussed in the previous section and traces along the guide curve. The resulting traced curve is projected onto the guide surface. A region of the guide surface is then traced which creates a surface patch with precise boundary.

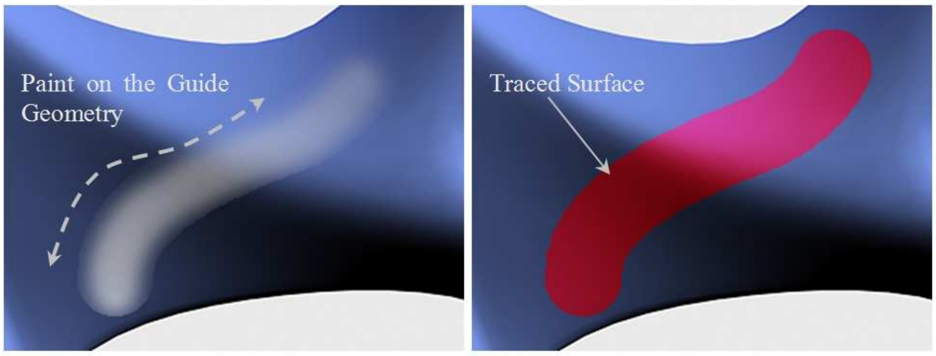


Figure 3.21 Thick-brush trace over guide surface

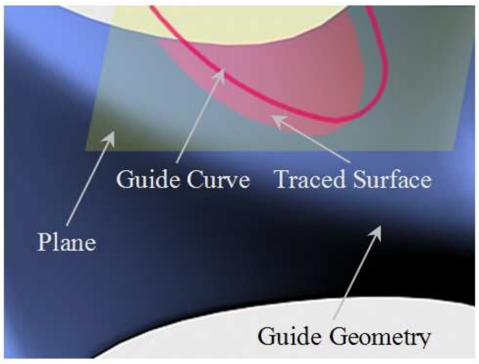
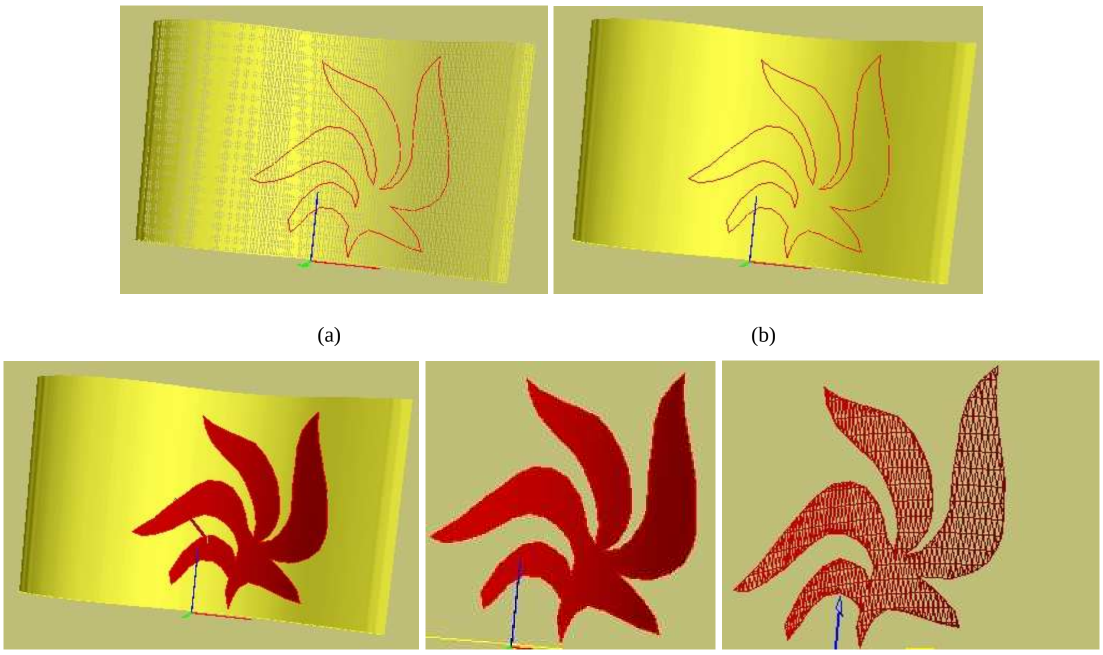


Figure 3.22 Tracing over guide surface using guide curves



(a) (b)

(c) (d) (e)

Figure 3.23 Creating a leaf model through point tracing

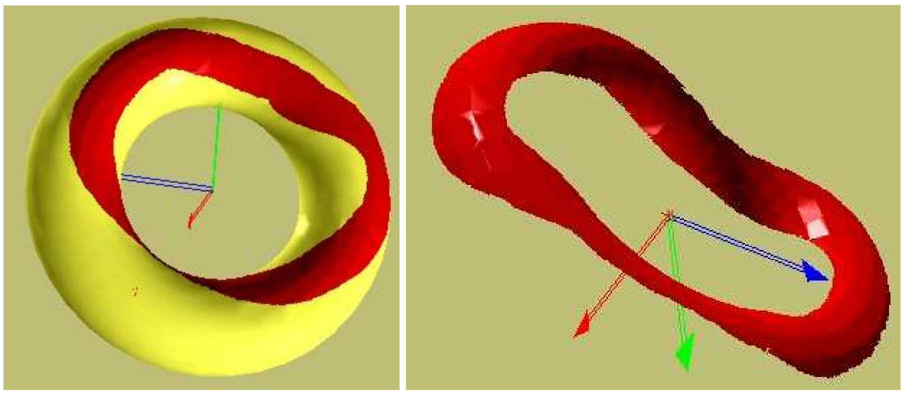


Figure 3.24 Thick-brush tracing

Figure 3.23 shows the actual snapshots of the step-by-step sequence of the point tracing operation in the GuTS system. The first picture shows an extruded surface and over that a traced curve is drawn in the shape of a leaf. The second picture shows the same in the shaded view. The third picture shows after the curve is inserted into the surface. The curve insertion algorithm is detailed in Chapter 4. This operation divides the surface along the traced boundary and the inner portion of the curve is highlighted. Then by trimming, the outside unnecessary surface is removed. The final desired leaf surface is shown in the fourth figure and the fifth figure shows the same in wire frame rendering.

The first picture in Fig. 3.24 shows the actual snapshots of thick-brush tracing over a simple torus object. The second picture shows after the traced portion of the surface is trimmed and the unwanted surface is removed. This example shows that a complex surface geometry can be created easily through a simple tracing operation using the GuTS approach. This is unlike other modeling programs where complex and time consuming operations are needed to create the same geometry.

Figure 3.25 shows the ability to trace using multiple guide shapes simultaneously and produces a single complex surface patch. As these guide surfaces are moved, the intersection between them is also computed in real-time. This is another example highlighting how simple tracing over multiple guide shapes creates a complex freeform surface in the GuTS approach. In other modeling operations it would take several time-consuming complex operations to create the same geometry. In addition to tracing the surface patches from guide shapes, basic operations, such as extrude, revolve, sweep, and loft operations, are also provided in the GuTS system.

3.2.3 Stitching Surfaces

Designing a complex freeform surface in a single step using a single patch is difficult and tedious. Often, merging several patches of surfaces together produces the required complex shapes easily. Hence, once the surface patches are created, as mentioned above, they are stitched together to form complex surfaces. It becomes important to stitch these surfaces together in a meaningful way to get the desired shape and properties.

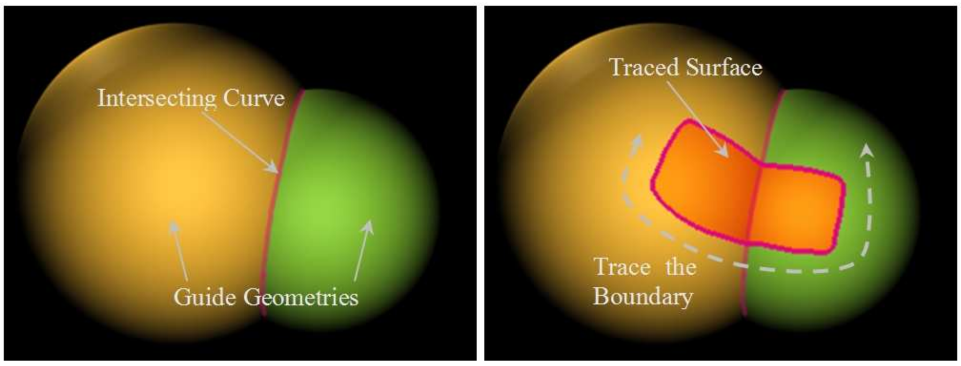


Figure 3.25 Tracing over multiple guide surfaces

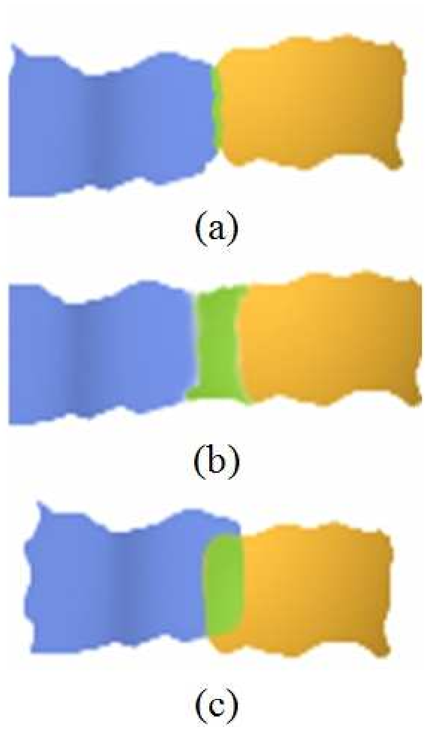


Figure 3.26 Surface stitch

It is very rare (or impossible) to have the boundaries of any two random freeform surfaces match exactly so that the surfaces can be stitched together directly. Therefore, the surfaces need to be processed at the boundary before stitching them together. Assuming S_1 and S_2 are two surfaces that need to be stitched, the following cases can happen.

1. S_1-S_2: boundary matches exactly

2. S_1-S_2: boundary does not match, then following cases will happen

    2.1 Overlap (modify - add or remove - vertices within the surface region)

    2.2 Extend (modify - add or remove - vertices outside the surface region)

These three cases, exact match, extend, and overlap, are shown in Fig. 3.26 (a, b, and c) respectively. In case 1, since both the boundaries match exactly it is straight forward and the surface patches are easily connected to form a single surface mesh. In case 2.1, the surface patches overlap. In this case, the intersection of these surface patches is computed and an exact boundary between the two surfaces along the intersection curve is created. This produces case 1 and these surfaces are easily stitched.

In case 2.2, when two surfaces do not overlap, a surface has to be created between the two surfaces based on the boundary conditions to connect them. In this case, no one solution exists and there is no clear definition of precise geometry for the extended surface between the two surface boundaries. At this stage, in the GuTS approach, we maintain shape precision by creating surfaces that are only a subset of the input geometries (i.e. subset of guide geometries). We have not implemented case 2 scenarios (modifying within the surface and extending the surfaces) in our current GuTS system, but this can be extended in the future as an add-on module with additional research.

3.2.4 Other Surface Operations

When designing surfaces it is not possible to get the desired shape just by creation itself. Often several other geometric operations are needed to refine the surface in order to achieve the desired shape and property.

3.2.4.1 Snapping at the boundary

To assist precise tracing over the boundaries and multiple surfaces when switching between surfaces, a snapping force is introduced along the boundary and intersecting curves (as shown in the Fig. 3.27). When the tracing point is within the threshold distance, it snaps to that boundary of the surface or to the intersecting curve forming a precise tracing operation.

3.2.4.2 Cutting and trimming operation

One important operation is cutting and trimming. As shown in Fig. 3.28, cutting can be done in several ways. It is similar to the creation of curves in the curve modeling approach. In creation, a boundary or a region is drawn on

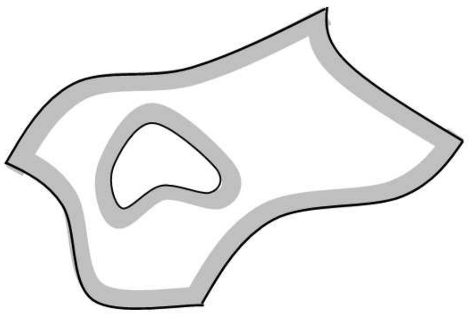


Figure 3.27 Snap region along the boundaries

a guide shape to create a new patch. In cutting, the boundary and the region are drawn on a surface, and the selected region is removed from the original geometry. All of the tracing mechanisms discussed in the surface tracing section can be used for trimming the surfaces.

3.2.4.3 Bump and dip features

As tracing is performed over a surface geometry, the traced portion can be raised to create a bump feature or lowered to create a dip feature. The raising and lowering can be performed easily using the local normal of the associated vertices. Also, when concave objects are raised, the need arises to check self-intersection and clear the geometry accordingly to create non-self-intersecting surfaces.

3.2.4.4 Local smoothing of surfaces

This feature is used to smooth the mesh locally. In this approach one vertex is adjusted at a time without changing the element topology but improving element quality. To make it interactive and simpler to use, thick-brush smoothing (similar to thick-brush tracing) is used to smooth the surfaces. When thick brush is used, the vertices within the brush radius are adjusted. Several techniques, such as Laplacian averaging, Optimized-based smoothing and Combined approaches, can be used. In the current version of the GuTS system, local smoothing is not implemented.

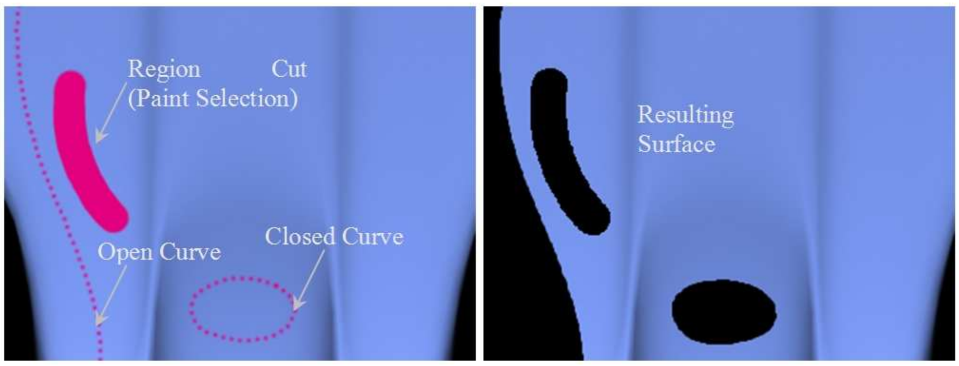


Figure 3.28 Cutting surface

Chapter 4

GuTS - Implementation and Technical Details

In this chapter we describe the technical details, geometric representation, high-level structure, and algorithms that are used in the GuTS modeling system. These detailed technical information will help the readers understand the underlying theoretical details and implementation of the GuTS system.

In GuTS, the underlying geometric representation and data structure is hidden from the user and provides direct manipulation of the geometries. The GuTS approach is not limited to any particular geometric representation and it can be implemented using different curve and surface representations. In our current implementation of the GuTS system, we used interpolation subdivision schemes and Bezier representations for curves and triangulated mesh structure to represent the surface geometries. We choose interpolation subdivision schemes for curve modeling for the following reasons: this scheme allows editing vertices of the curves directly; resulting limit curve interpolates the initial control points; allows multiple level-of-detail through interpolation subdivision; provides WYSIWYG approach; and it is easy to implement like a polyline. We chose the second representation - Bezier curves - for the following reasons: to demonstrate that the GuTS approach can be implemented using different geometric representations; Bezier curve representation is widely used in many computer graphics and CAD modeling systems enabling compatibility with those systems; and Bezier representation produces smoother curves compared to the interpolation subdivision scheme. Like the curve modeling, the surface modeling of the GuTS approach can also be implemented using different surface representations. In our current implementation of the GuTS system, we chose the mesh representation for the following reasons: multiple curve representations can create mesh representation directly; it is easy to represent and implement; and it can be easily extended to incorporate subdivision surface schemes (discussed in Section 4.4.7). Figure 4.1 shows a detailed list of the curve modeling functionalities and features in the GuTS system.

4.1 Curve Modeling - Subdivision Schemes

4.1.1 Interpolation vs. Approximation Subdivision Schemes

Subdivision schemes can be classified into many groups. One way to classify the subdivision schemes are: approximation (such as Loop [71], Catmull-Clark [37], and Doo-Sabin [42]) and interpolation (such as Four-Point [44], Butterfly [43], Modified Butterfly [107, 106], and Kobbelt [65]). In approximation schemes, both new nodal positions

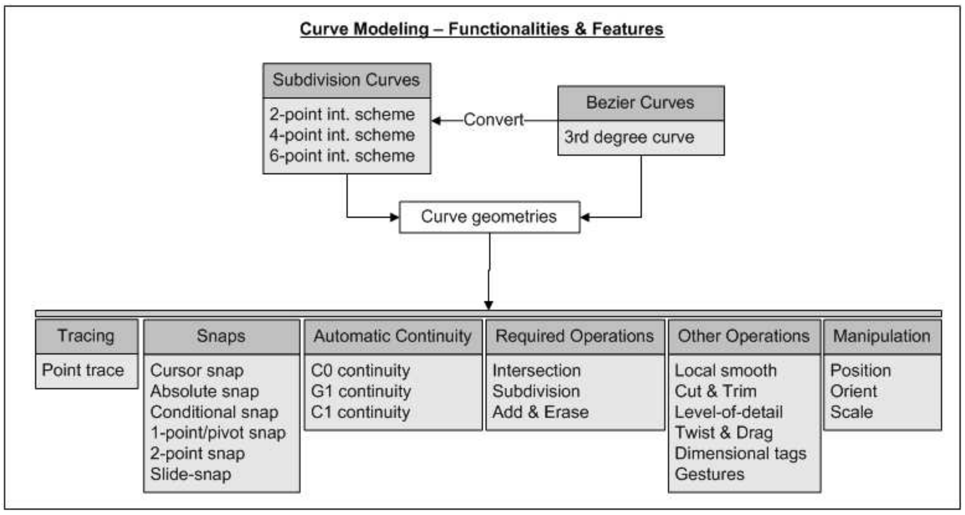


Figure 4.1 Curve modeling - Functionalities and features

- the newly created vertices and the vertices inherited from the coarser mesh - are computed. Consequently, the nodal positions of the initial mesh are not samples of the final surface. On the other hand, in interpolation schemes the nodal positions of the coarser mesh are fixed while only the nodal positions of new vertices are computed when going from a coarser to a finer mesh. Consequently, the nodal positions of the initial (or input) mesh, as well as any nodes produced during subdivision, interpolate the limit surface. Since the input points are part of the final geometry, it resembles editing the geometry directly rather than through control points.

In our current GuTS system, interpolation subdivision schemes are used. This scheme simplifies implementation of complex geometric manipulations such as tracing, cutting, stitching, and local editing with varying levels of details. One of the drawback of interpolation subdivision schemes is that they produce non-fair curves and surfaces. In the GuTS approach, we suppress this issue to an extent through the input points that are traced from guide shapes with desired smoothness and properties. Furthermore, several automated stitching and snapping mechanisms guarantee the required continuity and smoothness properties.

4.1.2 Properties of Interpolation Subdivision Schemes

Even though we did not introduce any new properties of the interpolation subdivision schemes in this research, a brief review of the properties of the interpolation subdivision schemes [44, 67] will help the readers understand the characteristics of different features in this chapter. Some of the important properties of interpolation subdivision schemes are:

- An interpolation subdivision algorithm is an insertion algorithm since all the points at stage k are carried over to stage $k + 1$ and new points are inserted between the old ones.
- The resulting limit curve interpolates the initial control points.
- It produces a straight line in R^d, whenever the control points lie on such a line
- The derivatives of the limit curve can be computed directly, without detailed refinement
- The resulting limit curves generated by 2-point, 4-point, and 6-point schemes are C^0, C^1,and C^2 continuous respectively.

Given the initial control points, based on the scheme, the intermediate points are inserted. In 4-point scheme four points are used to compute the new points for the next level. Similarly 2-point and 6-point schemes use two and six points respectively. Our implementation uses the subdivision scheme introduced by Dyn et al. [44] and Kuijt [67].

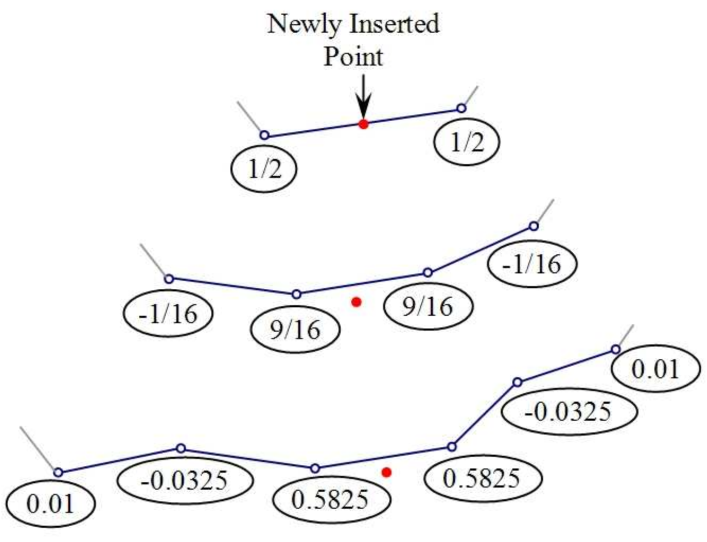


Figure 4.2 Masks for 2-point, 4-point, and 6-point interpolation schemes

4.1.3 Interpolation Subdivision Curve Representations

Figure 4.2 shows the commonly used masks for three types of interpolation schemes and its mathematical description is provided below. Given the initial control points $\{p_i\}_{i=-2}^{n+2}, p_i \in R^d$, intermediate points are added by several schemes as follows.

4.1.3.1 2-Point interpolation scheme

2-point scheme is a simple linear subdivision scheme that inserts new points at level $k+1$ by simply averaging two points from level k.

$$p_{2i+1}^{k+1} = \left(\frac{1}{2}\right)\left(p_i^k + p_{i+1}^k\right), -1 \leq i \leq 2^k n \qquad [4.1]$$

If the curve is closed, then the starting point and ending point are joined together and subdivided at each level.

4.1.3.2 4-Point interpolation scheme

Given control points $\{p_i\}_{i=-2}^{n+2}, p_i \in R^d$, intermediate points are added by the following scheme.

$$p_{i+\frac{1}{2}} = \left(\frac{1}{2} + w\right)(p_i + p_{i+1}) - w\,(p_{i-1} + p_{i-2}), -1 \leq i \leq n \qquad [4.2]$$

Let us denote the control points at the k-level set by $\left\{p_i^k\right\}_{i=-2}^{2^k n+2}$. Then the subdivision scheme defines the control points at level $k+1$ by

$$p_{2i}^{k+1} = p_i^k, -1 \leq i \leq 2^k n + 1, \qquad [4.3]$$

$$p_{2i+1}^{k+1} = \left(\frac{1}{2} + w\right)\left(p_i^k + p_{i+1}^k\right) - w\left(p_{i-1}^k + p_{i+2}^k\right), -1 \leq i \leq 2^k n \qquad [4.4]$$

where $p_i^0 = p_i, -2 \leq i \leq n+2$.

The role of w in the above scheme is a tension parameter. It has been observed [44] that when the values of w are $0 < w < 1/4$ the curve is continuous and when it is $0 < w < 1/8$, it has a continuous tangent vector. In 4-point scheme, $w = 1/16$ is most commonly used because at this value this scheme reproduces polynomials of a degree less than or equal to 3. The values of w which are relevant for application to curve design are $0 \leq w \leq 1/10$. Beyond this range, though the curve remains continuous, it tends to produce many loops and sharp bends.

The tangent at the i_{th} point is provided by

$$p^{'}(i) = \frac{1}{1-4w}\left\{\frac{1}{2}(p_{i+1} - p_{i-1}) - w\,(p_{i+2} - p_{i-2})\right\} \qquad [4.5]$$

For an open curve, two additional points in the beginning and two additional points at the end of the curve are needed, which affects the behavior of the curve near its end points. These extra points can be used to control the slope of the curve at the end points. For a closed curve, the additional points are derived from the existing points as follows: $p_{-2} = p_{n-1}, p_{-1} = p_n, p_{n+1} = p_0, p_{n+2} = p_1$.

4.1.3.3 6-Point interpolation scheme

In this scheme six points are used to subdivide the next level of points. This scheme produces continuous curves with continuous tangent and curvature.

$$p_{i+1/2} = \left(\frac{9}{16} + 2\theta\right)(p_i + p_{i+1}) - \left(\frac{1}{16} - 3\theta\right)(p_{i-1} + p_{i+2}) + \theta\,(p_{i-2} + p_{i+3}) \qquad [4.6]$$

For $\theta = 0$ and $w = 1/16$, this scheme corresponds to 4-point interpolation scheme. When the value of θ is $0 < \theta < 0.02$, then it guarantees the continuity of the curvature of the curve. The detailed description, proof of convergence and analysis of these schemes can be found from the research by Dyn et al. [44] and Kuijt [67].

4.2 Curve Modeling - Bezier Representation

The Bezier [22] form of the polynomial curve segment, named after Pierre Bezier, indirectly specifies the endpoint tangent vector by specifying intermediate points that are not on the curve. The Bezier curve interpolates the two end control points and approximates the remaining intermediate points.

For given $n+1$ control-point positions: $p_k = (x_k, y_k, z_k)$, with k varying from 0 to n. These coordinate points can be blended to produce the following position vector $P(u)$, which describes the path of a Bezier polynomial function between p_0 and p_n.

$$P(u) = \Sigma_{k=0}^{n} p_k B_{k,n}(u), 0 \leq u \leq 1 \qquad [4.7]$$

The Bezier blending function $B_{k,n}(u)$ are the *Bernstein polynomials*:

$$B_{k,n}(u) = C(n,k)u^k(1-u)^{n-k} \qquad [4.8]$$

where the $C(n,k)$ are the binomial coefficients:

$$C(n,k) = \frac{n!}{k!(n-k)!} \qquad [4.9]$$

$$P(u) = \Sigma_{k=0}^{n} p_k B_{k,n}(u), 0 \leq u \leq 1 \qquad [4.10]$$

4.2.1 Properties of Bezier curves

We did not introduce any new properties of the Bezier curves in this research, a brief review of the properties of the Bezier curve is presented here for completeness. Some of the important properties of Bezier curve representation are:

- A Bezier curve is a polynomial. The degree of the polynomial is always one less than the number of control points.
- The curve follows the shape of the control point polygon and is constrained within the convex hull formed by the control points.
- The control points do not exert 'local' control. Moving any control point affects all of the curve to some extent.
- The first and last points are the end points of the curve segment.
- The tangent vectors to the curve at the end points are coincident with the first and last edges of the control point polygon.
- Moving the control points alters the magnitude and direction of the tangent vectors.
- The curve does not oscillate about any straight line more often than the control point polygon - known as the variation diminishing property.
- The curve is transformed by applying any affine transformation to its control point, and the curve is invariant under such a transformation.

4.2.2 3rd degree Bezier Curve Representation

In computer graphics (CG) applications, 3_{rd} degree curves are commonly used. Quadratic curves are not flexible enough and anything above 3_{rd} degree gives rise to complications. While certain CAD applications require higher order curves, they loose 'local control' of the curves that is desirable for most CG applications. The best compromise for CG applications, and the GuTS system, are 3_{rd} degree curves which give reasonable design flexibility while avoiding the increased calculations needed with higher-order polynomials. Considering this and to be compatible with most CG systems, in our current implementation of the GuTS system we used 3_{rd} degree Bezier curves.

3_{rd} degree Bezier point function can be written in matrix form:

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \bullet M_B \bullet \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \qquad [4.11]$$

where the 'Bezier Matrix' M_B is

$$M_B = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad [4.12]$$

At the end positions of the 3_{rd} degree Bezier curve, the parametric first derivatives (slopes) are

$$P'(0) = 3(p_1 - p_0), P'(1) = 3(p_3 - p_2) \qquad [4.13]$$

The parametric second derivatives are

$$P''(0) = 6(p_0 - 2p_1 + p_2), P''(1) = 6(p_1 - 2p_2 + p_3) \qquad [4.14]$$

These expressions are used to construct piecewise curves with C^1 or C^2 continuity between sections.

Let us say two 3_{rd} degree Bezier curves with control points (p_0, p_1, p_2, p_3) and (p_4, p_5, p_6, p_7) are connected at the end points p_3 and p_4. Then, when $p_3 = p_4$, C^0 continuity is achieved. G^1 continuity is achieved when $(p_3 - p_2) = k(p4 - p5)$, and when $k = 1$ in the above equation, C^1 continuity is achieved. Using the above set of equations, automatic continuity between multiple Bezier curve segments is achieved in the GuTS system.

4.3 Algorithms for Curve Operations

In this section the algorithms and methodologies that are used to perform several curve based modeling operations are discussed. Primarily, we discuss novel curve snapping mechanisms such as point snap, one-point snap/pivot snap, slide snap, and two-point snap/fixed snap. Also we discuss the implementation details of finding the intersection of curves (for subdivision and Bezier representations), and a method to create primitives using gesture inputs.

4.3.1 Curve Snap Mechanisms

Curve snapping is one of the key components in curve modeling in the GuTS approach. While snapping (i.e. concept of gravity fields) itself is not new, most of the modeling systems use snapping the cursor to geometries. In this research we introduced novel snapping mechanisms called 'whole geometry' snapping, where one geometry snaps to another geometry. While there is a precedent of object snapping to a point, in this section we discuss different newly introduced snapping mechanisms, such as, one-point snap (pivot-snap), two-point snap (fixed snap), and slide snap. In addition to these novel snapping mechanisms, the GuTS system also allows traditional snapping mechanisms.

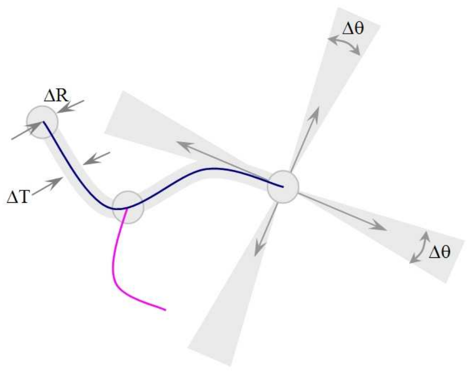


Figure 4.3 Gravity fields

Figure 4.3 shows the different gravity fields (i.e. snap regions - shown in grey color) of a curve in the GuTS system. First, at the end of each curve the end points exert a gravity field of small radius (ΔR). Second, a small thickness (ΔT) along the curve exerts a gravity field. With subdivision curves, the curve itself is stored as a series of point sets. Then the gravity field is computed for each line segment in the curve creating a thick gravity field all along the curve. With Bezier curves, the shortest distance to the curve from any given point is calculated dynamically, with certain thickness (ΔT) forming a thick gravity field along the curve. Third, a small angular portion $(\Delta\theta)$ from tangential and normal vectors at end points of the curve exert gravity fields. All these different types of gravity fields form various snapping mechanisms.

Figure 4.4 shows different types of snapping mechanisms that use the above gravity fields. End point gravity fields are used to snap two curves at end points. Then one of the geometries can be rotated on the snapped point - creating 1-point snap, also called ‘pivot snap’. The end point gravity field of a curve can also snap to the thick gravity field of another curve. Once a curve is snapped to an end point of another curve, the first curve can be moved (slid) such that the first curve slides while touching the end point of the second curve. The gravity field along the curve helps other curves to slide along this curve without detaching the curves. This snapping mechanism is called ‘slide snap’. Angular gravity fields at the end points snap other curves and maintain tangential or perpendicular conditions - called perpendicular snap and tangent snap. Outside these angular snap regions, a curve rotates by pivoting at the end point. The other snapping mechanism is the ‘two-point snap’. In this snapping mechanism, first a curve is snapped to an end point of another curve - forming pivot snap. Then one of the curves is rotated until the curve snaps to a second point. This second point can be from any curve in the modeling session. Once two points are snapped, then the geometry is fixed at that position - forming two-point snap (also called ‘fixed snap’).

In addition, gravity fields are introduced as grids in cartesian and/or polar coordinates (refer Fig. 3.8). These gravity fields position geometries precisely in a global coordinate system (i.e. absolute positioning). The cursor also exerts a gravity field around its center point. This gravity field provides precise position of the cursor with respect to

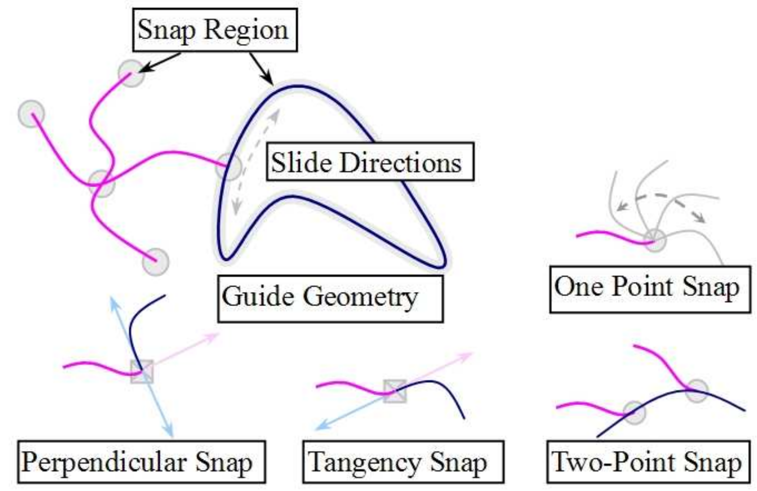


Figure 4.4 Snapping Mechanisms: point, slide, and fixed snaps

curve geometries, helping trace geometries easily, precisely and rapidly. This snapping mechanism is called ‘cursor snap’.

4.3.2 Intersection

In this section, we discuss the implementation details of finding intersection points between curves (for both Subdivision and Bezier representation). While the intersection algorithms itself are not novel, we present the implementation details to show how we used the subdivision and recursive properties of the geometries to find intersection points rapidly, still achieving interactive frame-rate. We also discuss the limitation of our implemented intersection algorithm for subdivision curves.

Intersection of the curves is performed in multiple stages and through local refinement. With subdivision curves, first the intersection between the two curves is performed at the coarse level. If the intersection is identified, then the intersecting segments are identified. Then three new vertices are computed (for 4-point scheme) and inserted at before, middle, and after the intersecting segments. Next the intersection between these two local regions is computed and again three new vertices are introduced. This process is continued until two successive intersection points lie within an acceptable tolerance limit. The first picture in Fig. 4.5 shows the intersection between two coarse curves. The second picture shows the next cycle, where three new points (shown as solid circles) are inserted using the 4-point scheme. The same approach is applied for 2-point and 6-point schemes. For 2-point scheme, only one new point is inserted locally in every cycle. For the 6-point scheme five new points are inserted in every cycle. This approach finds the intersection point in constant computational complexity in every cycle, irrespective of the size or the number of vertices in the curves. Due to the nature of the interpolation subdivision scheme, identifying the intersection point(s) between two curves at any given level of detail is as same as identifying the intersection point(s) between two polylines. Due to the same reason, if two curves do not intersect at any given level of detail then the GuTS system will not try to find the intersection point, even if those curves may intersect at a different level of detail. Similarly, even if two curves intersect at a given level of detail, they may not intersect when the level of detail changes. In our current implementation of the GuTS system, we address this issue as follows: if the curves do not intersect at a given level of detail then do not find the intersection point; if the curves intersect then find a more accurate intersecting point by recursively increasing the level of detail locally. The recursive process stops when a preset precision is achieved or when the intersection calculation is interrupted because the curves transition from intersecting to non-intersecting state.

With Bezier curves, the intersection can be calculated in various ways. Some of them include Bezier Clipping [89, 79], Recursive subdivision using De CastelJau’s algorithm [45, 31], and higher order (9th degree) polynomial equation. Root finding to a higher order polynomial equation is not a robust method and hence skipped. Both Bezier clipping and recursive subdivision methods are robust and easily implementable. In our current GuTS system, recursive subdivision using De CastelJau’s algorithm is implemented. This intersection algorithm is fast enough and computes intersection points dynamically as the user moves the curves interactively in the GuTS system.

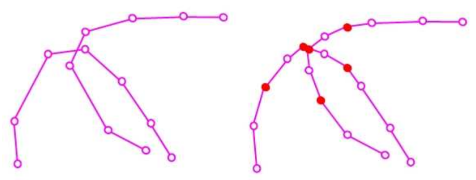


Figure 4.5 Multi-level intersection

4.3.3 Gesture-Based Primitive Creation

In this section, we discuss a method to create primitives from freehand input strokes. While the presented method itself is not novel, we present the implementation details to show how basic primitives, such as lines, circles, ellipses, and arcs, are created by recognizing the user's freehand strokes.

In the GuTS system, the user can turn 'on' or 'off' the gesture recognition feature. By default, gesture recognition feature is turned 'off', since the primary mode of input in the GuTS approach is guided tracing. When gesture recognition is 'on', as the user draws a freeform drawing, the region of the input sketch is scanned. Based on the distribution of the input strokes (the number of input points) within this region, the final primitive is recognized. Figure 4.6 shows how the input stroke is compared with the pattern and, with additional conditions, the final shape is recognized. Though the approach is simple, this approach identifies the shapes for basic primitives. Our implemented gesture recognition in the GuTS system is simple, based on the distribution of the input points in the 2D region, and is limited to only create simple primitives rapidly and fluently. As shown in Fig. 4.6, the scanned region is subdivided into a 3x3 matrix and the distribution of the number of points in each of these boxes is identified. Assuming N is the number of input points in the stroke, then 'approximately' N/8 points are distributed in the outer boxes, and 'approximately' zero points in the middle box. Along with this pattern, the additional condition of the height/width ratio is used to decide the final shape. Also, the major and minor radius of the ellipses is computed from the height and width parameters accordingly. Similarly for recognizing the lines, the diagonal columns are distributed with 'approximately' one-third of the total number of input points and the remaining boxes with 'approximately' zero points. Again the ratio of height/width is used to recognize the different types of line segments. In this approach the recognition is independent of both the direction of the input strokes, as well as the repetition of the input strokes. Similarly different patterns and conditions are used to match the arcs. Once the input gestures are recognized and basic shapes are created, those basic shapes are manipulated like any other geometry. This feature shows that the GuTS system creates simple shapes through simple gesture input. A complex gesture recognition system can be found elsewhere [83, 84], that uses a single gesture stroke to create simple shapes such as lines, texts, editing geometries, etc.

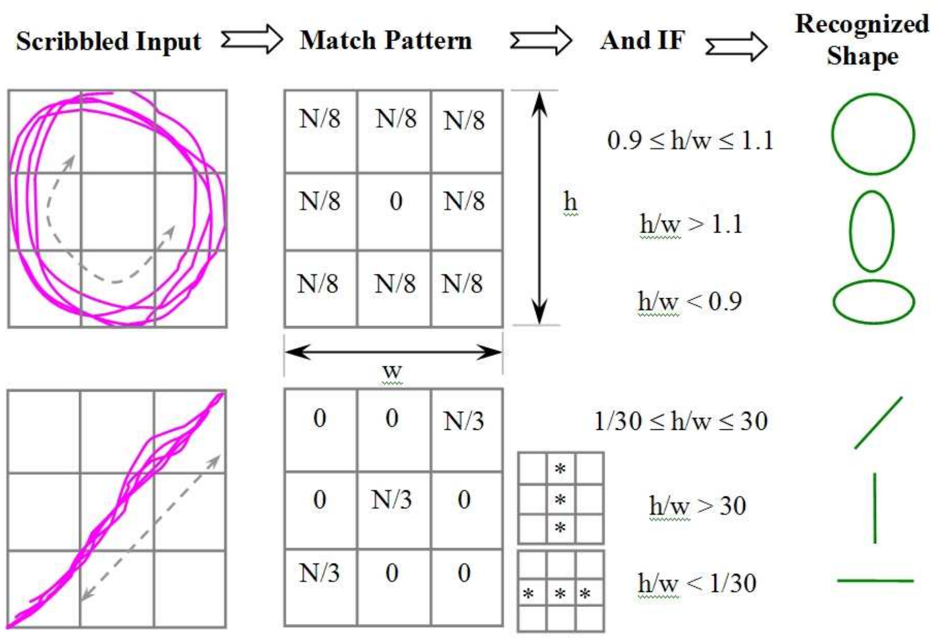


Figure 4.6 Gesture recognition algorithm

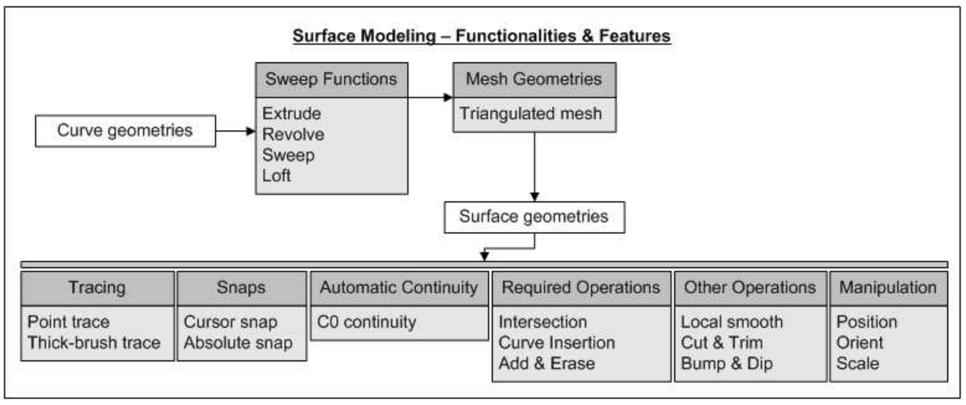


Figure 4.7 Surface modeling - Functionalities and features

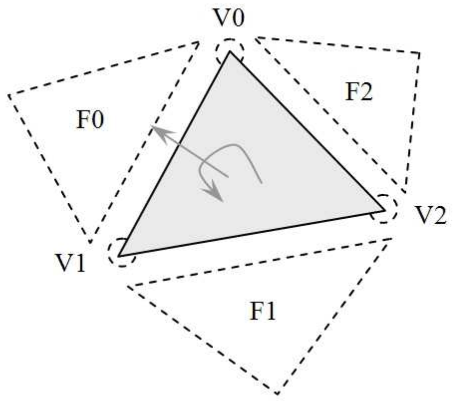


Figure 4.8 Triangular face structure

4.4 Surface Modeling - Triangulated Mesh Structure

Figure 4.7 shows a detailed list of surface modeling functions and features in the GuTS system. In this research we introduced new tracing mechanisms: zero-point tracing, thick-brush tracing, and tracing over multiple surfaces. As part of the tracing and stitching operation, we developed an algorithm to insert curves and split surfaces.

In the GuTS system, currently we use a triangulated mesh structure to represent surface geometries. Figure 4.8 shows a face structure. Each face contains information about its three vertices and three pointers to the adjacent faces, ordered counter-clockwise using a right-hand coordinate system. Adjacent face $F0$ is always adjacent to the vertices $V0$ and $V1$, and similarly faces $F1$ and $F2$ are always adjacent to the vertex pairs $V1$ and $V2$, and $V2$ and $V0$ respectively. Maintaining this order consistently helps to implicitly represent the edges $E0$, $E1$ and $E2$ of a face, between the vertices $V0$, $V1$ and $V2$ respectively.

4.4.1 Algorithms for Surface Modeling Operations

In this section the algorithms and methodologies that are used to perform several surface modeling operations are discussed. The important operations in surface modeling are zero-point tracing, and thick-brush tracing, tracing over multiple surfaces, inserting a traced curve, intersection of surfaces, and stitching surfaces. The algorithms for zero-point tracing, thick-brush tracing, and automated stitching operations are presented below.

4.4.2 Zero-Point Tracing

The steps involved in zero-point tracing are trace a curve over the surface, insert this curve into the surface and divide the surface accordingly, and stitch (combine) the surface patches together. In zero-point tracing, as the user draws on the screen, the 2D screen coordinates are converted into a 3D ray in the world coordinate system from the

3D point corresponding to the cursor position in the view direction. Then this ray is intersected with the guide surface and the closest intersecting point to the view point is computed. As the user moves the cursor, multiple intersecting points on the surface are computed and connected together to form a tracing curve. This tracing curve lies on the guide surface. Once this curve is traced, then the traced curve is inserted into the surface which is then divided along the inserted trace curve. Once the region is traced, several operations can be performed on this surface patch.

During tracing, depending upon the speed and movement of input pen-tablet creates varying curve quality - i.e. differing gap between input vertices and different total number of vertices in the curve. To eliminate this quality variation of the traced curve due to tracing speed, the GuTS system provides a filter with a certain minimum-maximum threshold value. This filter helps deciding whether a new vertex needs to be added to the trace curve or not depending upon the movement, location and difference of the pen-tablet from the previous point. So irrespective of the tracing speed, the quality of the traced curve remains the same. Before discussing different surface modeling operations, the algorithm for inserting a curve over the surface is presented.

4.4.2.1 Insert trace curve

The algorithm for the trace curve insertion is presented in pseudo code format as follows: (Input to this algorithm is a traced curve and a set of guide surfaces. Output from this algorithm is a set of surface patches after the traced curve is inserted into the input guide surfaces):

```
A  For the first vertex (current vertex) in the trace curve do the following
    A.1  If the current vertex is already a vertex in the surface mesh go to step A4
    A.2  If the current vertex is fully inside a face, then insert the current vertex using "face split" (discussed below)
         and go to step A4
    A.3  If the current vertex is on an edge, then insert the current vertex using "edge split" (discussed below) and go
         to step A4
    A.4  Mark and update the surrounding faces of the current vertex
    A.5  Make the next vertex in the trace curve as the current vertex and proceed to step B
B  For each remaining vertex in the trace curve do the following:
    B.1  If the current vertex is already a vertex in the surrounding faces, go to step B6
    B.2  If the current vertex is fully inside one of the surrounding faces, insert this point using "face split" (discussed
         below) and go to step B6
    B.3  If the current vertex is on an edge of one of the surrounding faces, insert the current vertex using "edge split"
         and go to step B6
```

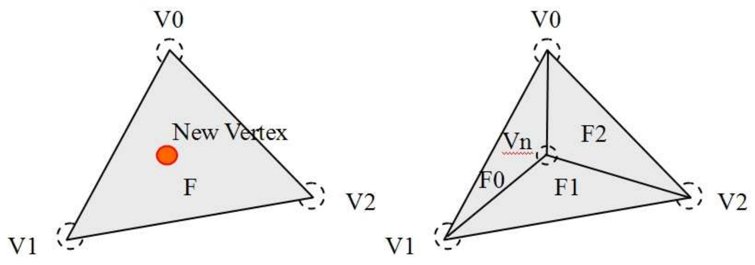


Figure 4.9 Face split

B.4 If the current vertex is not inside or on an edge of the surrounding faces, project the line segment that connects the last point and the current point

B.5 Depending upon the status of the projected line segment do one of the following:

B5.1 If the projected line bisects one of the surrounding faces completely, insert a new point at the bisection point and make this the current point. Refer Fig. 4.11 for an example - projected curve bisects the face F0, then a new vertex is inserted at the bisection point and face F0 is split to form two faces F01 and F02, and face F1 is split to form faces F11 and F12. Go to step B6

B5.2 If the projected line passes along one of the edges of the surrounding faces, insert the vertex on the face at the other end of the edge and make this vertex the current vertex (refer Fig. 4.12). Go to step B6

B5.3 If the projected line segment lies within one of a surrounding faces, extend this projected line further such that it fully bisects that face, and go to step B5.1

B5.4 If the projected line segment has zero length, then no unique solution exists - throw the exception and terminate the insertion process

B.6 Mark and update the surrounding faces of the current vertex

B.7 Make the next vertex in the trace curve the current vertex and continue from the beginning of step B1

C If the curve is closed, then insert the line segment from the last point to the first point in the trace curve, using step B

D From the first vertex of the trace curve, mark the left and right faces such that the trace curve divides the surface into two regions

4.4.2.2 Face split

'Face split' operation is needed to insert a new vertex inside an existing face. Figure 4.9 shows the face split operation. When the new vertex (Vn) is fully inside a face (F), then that face is split into 3 new faces (F0, F1, F2),

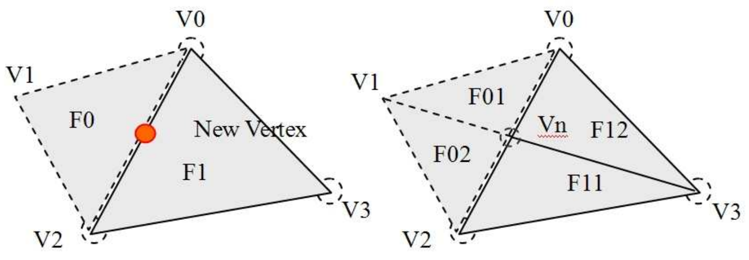


Figure 4.10 Edge split

with the order of vertices (V0, V1, Vn), (V1, V2, Vn) and (V2, V0, Vn) respectively. The old face (F) is replaced by the three new faces. Then these three new faces' neighboring face lists are updated. Then the neighboring faces' neighboring face lists are updated to complete the face split operation.

4.4.2.3 Edge split

'Edge split' operation is needed to insert a new vertex on an existing edge. The Figure 4.10 shows the edge split operation. When the new vertex (Vn) lies on an edge, that edge is split by dividing the adjacent two faces of the edge into four new faces. Faces F0 and F1 are split into F01 and F02, and F11 and F12 respectively with the order of vertices (V0, V1, Vn), (V1, V2, Vn), (V2, V3, Vn), and (V3, V0, Vn). Then these new four faces' neighboring face lists are updated. Then the neighboring faces' neighboring face lists are updated to complete the edge split operation.

The above mentioned face and edge operations does not have any special mechanism to create or maintain high quality mesh. Due to the same reason, in our currently implemented GuTS system, sometimes (especially during thick-brush tracing) the mesh results in poor tessellation. In this research, we did not focus on mesh quality explicitly and maintaining a high quality tessellated mesh is a separate research onto itself.

4.4.2.4 Projection curve

Figure 4.11 shows the state described in pseudo code B5.1. In this case the new vertex does not lie in one of the adjacent surrounding faces to the previous vertex so the trace curve is projected over the faces. Since the projected curve bisects the face F0 completely, edge split is used to insert a new point. Then, the previous vertex position is moved to the newly inserted vertex position. At this point as shown in the second picture in Fig. 4.11, the new vertex lies within one of the surrounding faces. Hence, by face split, the last vertex is inserted. This process is continued until all the projected curve segments are inserted.

Figure 4.12 shows the projected curve coinciding with one of the edges. This corresponds to the state described in pseudo code B5.2. In this case the projected curve coincides fully with an edge, so continue to the next vertex. As shown in the second picture in Fig. 4.12, no new faces or vertices are created - just the previous vertex status is

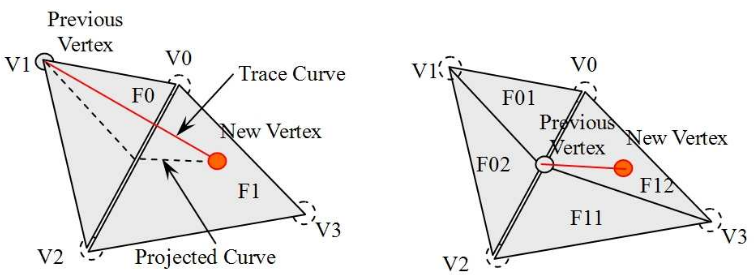


Figure 4.11 Projection curve - Bisects face

updated. At this stage since the new vertex lies within the surrounding faces, the new vertex is inserted through the face split process.

Figure 4.13 shows the projected curve lying within a surrounding face. This corresponds to the state described in the pseudo code B5.3. In this case the projected curve is extended further such that it bisects the face completely, as shown in the second picture in Fig. 4.13. Then the scenario boils down to the state described in B5.1, which can be addressed as mentioned in the section above corresponding to B5.1. In the current implementation of the GuTS system, this state is not included and it behaves the same way as described for the state B5.4, where the exception is thrown and the tracing process is terminated. Figure 4.14 shows a situation where no unique projection curve exists, and in this case the GuTS systems throws exception and the tracing process is terminated.

4.4.3 Thick-Brush Tracing

Thick-brush tracing is a process where a thick brush with a user specified radius is traced on the surface. As the tracing input point moves, the surface region along the traced path with thickness equal to the brush diameter is dynamically selected. In the GuTS system, similar to zero-point tracing, a filter with certain threshold value is used to maintain the quality of the trace curve irrespective of the tracing speed. Then this selected portion of the mesh is used for further geometric operations. In thick-brush tracing, similar to zero-point tracing, the 3D intersection point on the surface is computed. Instead of creating a trace curve, in this approach the faces are dynamically updated. This dynamic update of the mesh results in poor tessellation. Maintaining high quality tessellated mesh is a separate research onto itself and hence in the implemented GuTS system we did not address this issue.

The algorithm for thick-brush tracing is presented in pseudo code format as follows: (Input to this algorithm is the radius of the thick-brush, a series of input points of the traced curve, and a set of guide surfaces. Output from this algorithm is a set of surface patches, after the boundary of the traced surface region is computed):

```
C  For every input intersection point during the tracing do the following:

   C1  Find the face in the surface mesh in which the input intersection point lies
```

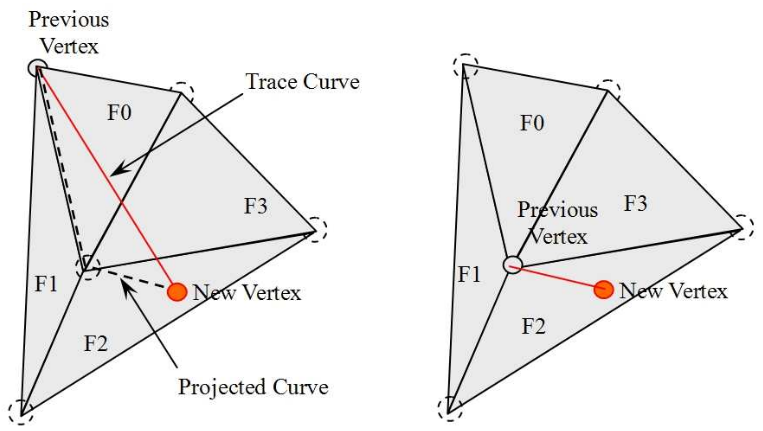


Figure 4.12 Projection curve - Coincides with an edge

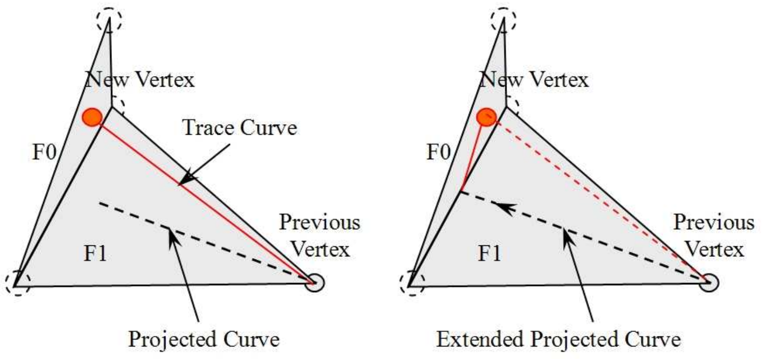


Figure 4.13 Projection curve - Lies within a face

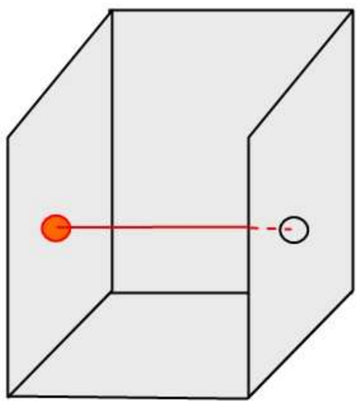


Figure 4.14 No unique projection curve

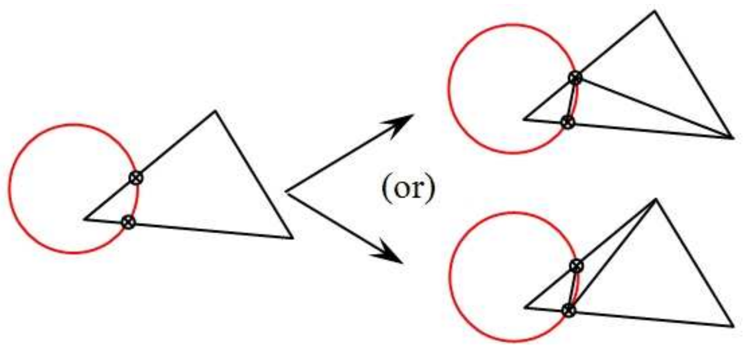


Figure 4.15 One vertex inside

C2 Check the current face with the sphere of the radius of the thick brush from the current input intersection point

C2.1 If all three vertices are fully outside and none of the edges intersects with the sphere, then mark this as fully outside, processed and go to step C3

C2.2 If all three vertices of the current face lie within the sphere, then mark this as fully inside, processed and go to step C3

C2.3 If one of the vertices is inside the sphere and the remaining two vertices are outside, then do the edge split twice (refer Fig. 4.15) and replace the current face with the three new faces. Make one of the new faces the current face and go to step C2

C2.4 If two of the vertices are inside the sphere and the remaining vertex is outside, do the edge split twice (refer Fig. 4.16), and replace the current face with the three new faces. Make one of the new faces the current face and go to step C2

C2.5 If two of the vertices are on the sphere and the third vertex is outside and if the chord distance between the inside edge and the surface of the sphere is beyond a certain tolerance limit, insert a new vertex through "face split" (refer Fig. 4.17). Replace the current face with the three new faces, make one of the new faces the current face and go to step C2

C2.6 If all three vertices are outside and an edge of the current face intersects the sphere and if the chord distance between the edge and the surface of the sphere is beyond a certain tolerance limit, do two edge splits (refer Fig. 4.18). Replace the current face with the three new faces, make one of the new faces the current face and go to step C2

C3 Make one of the unprocessed adjacent faces the current face and go to step C2.

Figure 4.15 shows one vertex inside and two vertices outside. In this case, in the current implementation of the GuTS system, either one of the shown ways splits the edges - randomly. Figure 4.16 shows two vertices inside and one vertex outside. In this case too as above, in the current implementation, in either one of the shown ways the edges are

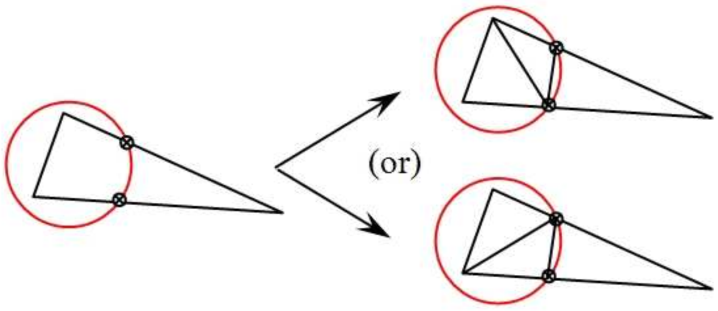


Figure 4.16 Two vertices inside

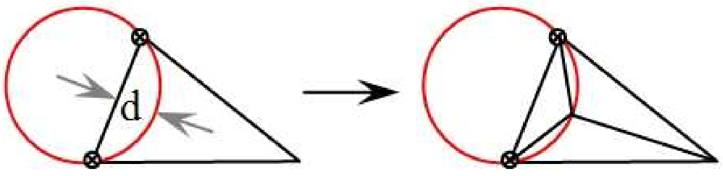


Figure 4.17 Two vertices ‘On’

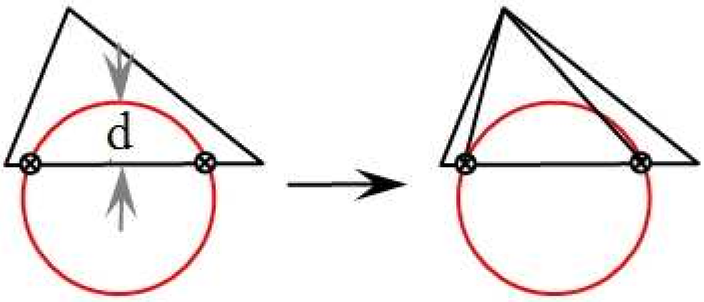


Figure 4.18 Only edge intersects

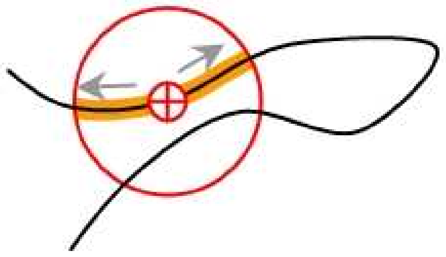


Figure 4.19 Trace around intersection point

split - randomly. Since the faces are traversed through the adjacent faces starting from the input intersection point, the faces that are processed and marked are only connected from the intersection point and within the brush radius. Even if other parts of the surface lie within the brush radius, they are not processed intentionally as shown in the Fig. 4.19.

4.4.4 Tracing over Multiple Surfaces

Tracing over multiple surfaces works the same way as tracing over a single surface. Even when multiple surfaces are used for tracing, at any given instance, only one point is inserted and over one surface. So making use of this, one surface, called the active surface, that is currently traced over is always maintained. When tracing shifts from the current surface to some other surface, the other surface is made the active surface. This is done transparently to the user. As multiple curves are traced, they are traced over multiple surfaces and are associated to that corresponding surface. Once the tracing process is completed, though it appears to the user that multiple curves are traced over multiple surfaces, internally each curve is attached to the corresponding surface. Then the computation of the traced surface boils down to (a) inserting the intersection curve, and (b) inserting the traced curve to the corresponding surface, as described in the single surface tracing section. Once the tracing is completed for all the necessary surfaces and multiple patches are created, they are merged together to form a single surface mesh.

Figure 4.20 shows the step-by-step pictorial view of tracing over multiple surfaces and stitching procedures to get the final desired geometry. Two spheres, A and B, are intersected and the intersecting curve is calculated dynamically in real-time. The computation of the intersection curve is described in the following section, and the intersection curve ensures the correct connectivity when the user switches tracing from one surface to another. This intersection curve is shown in grey. The traced curve which spreads over both the spheres is shown in green - the final shape that needs to be created. As mentioned above, the tracing procedure is performed one surface at a time internally, the same as in single surface tracing. First the intersection curve is inserted in the geometry. Then the corresponding portion of the traced curve is inserted in the appropriate surface. This produces the multiple pieces of the final geometry traced from multiple surfaces. In this example A’ and B’ are the traced patches from the surfaces A and B respectively. Then these two patches are combined together to form the final desired geometry.

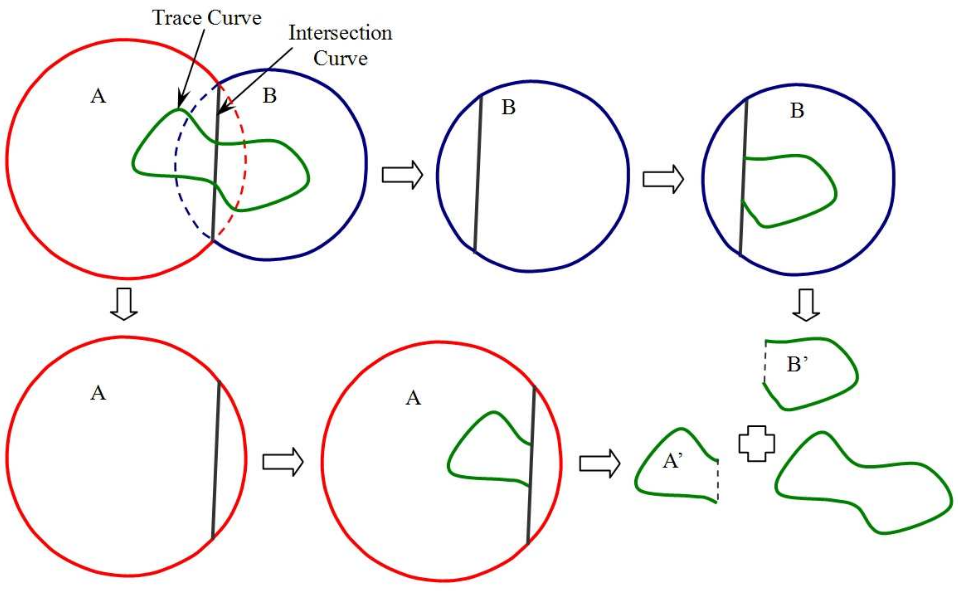


Figure 4.20 Tracing over multiple surfaces

4.4.5 Surface Intersection

In the current implementation, we used the V-Collide [58] collision detection package to identify the intersecting line segments between the surface meshes. V-Collide is a collision detection library designed to operate on a large number of arbitrary polygonal objects. It performs efficient and exact collision detection between triangulated polygonal models. It makes no assumptions about input structure and works on arbitrary models, also known as “polygon soups”. V-Collide uses three-stage collision detection architecture:

1. An N-body test finds possibly colliding pairs of objects,
2. A hierarchical-oriented bounding box test finds possibly colliding pairs of triangles, and
3. An exact test determines whether or not a pair of triangles actually overlaps.

The N-body routine uses coherence between successive time steps of a simulation to perform well in animations and moving simulations. The hierarchical Oriented Bounding Boxes (OBBs) and exact collision routines are taken from RAPID [47], a component of V-Collide which is also available as a stand-alone package.

The basic steps involved in using this library are creating objects, adding sets of triangles to these objects, choosing which pairs of objects should be tested for collisions, setting the positions of the objects, performing the collision test, and getting back reports of the test results. Based on these results and any other parameters of the simulation/interaction, the objects may be moved and the collisions tested again, etc. V-Collide is written in C++, but it also provides a C interface as well.

The V-Collide library takes the triangulated mesh and returns the list of the intersecting line segments between the faces in the input mesh. Figure 4.21 shows two simple meshes, each with two triangulated faces. In this case the V-Collide library returns three line segments in a random order and also the intersecting faces. Then in the GuTS system, these individual line segments are sorted and connected to form the intersecting curves. These intersecting curves also store the information about which two surfaces are intersected to form the curve. This information is used to insert the intersection curves in the appropriate surfaces correctly, as shown in the second and third picture in Fig. 4.21. An important point to observe here is that the intersecting curve is computed and inserted such that the inserted curve will match exactly between the two surface meshes. This gives a proper connectivity to stitch surface meshes along this intersection curve without many additional changes. While the complete intersection algorithm involves several steps and uses multiple libraries, all the necessary functionalities are implemented seamlessly in the GuTS system - resulting in the dynamic intersection of surfaces interactively and transparently to the user.

4.4.6 Stitching Surfaces Together

In our current implementation, stitching between surfaces happens when these geometries intersect over the traced portion of the region. The intersection curve is computed and intersected to the surfaces such that it ensures the proper

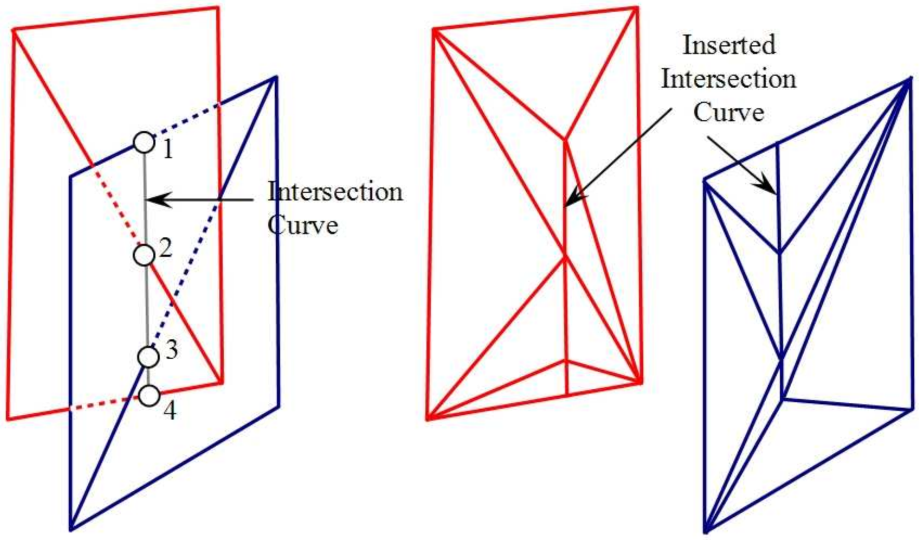


Figure 4.21 Inserting intersection curve

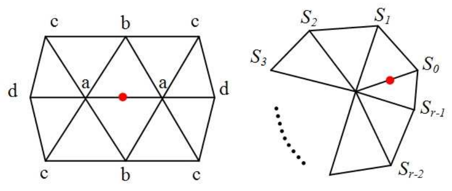


Figure 4.22 Masks for regular and extraordinary vertices

coincidence between the edges and vertices that confirms the proper connectivity between these multiple surface patches (as shown in Fig. 4.21). Since the vertices and edges match properly along the intersection curve, the stitching between the meshes boils down to creating the connectivity between the faces of the surface meshes accordingly, maintaining C^0 continuity.

4.4.7 Interpolation Subdivision Scheme (Modified Butterfly Scheme)

As mentioned earlier, the GuTS approach is independent of the underlying geometric representation. In our current implementation of the GuTS system, we used triangular mesh representation. In this section, we present how with minor modifications the surface geometries can be represented using interpolation subdivision surfaces. A basic interpolation subdivision scheme called 'Modified Butterfly Scheme' is presented here. The Modified Butterfly Scheme by Zorin et al. [107, 106] is a modified version of the original Butterfly Scheme by Dyn et al. [43], for triangular tilings. This interpolation scheme produces C^1 continuity surfaces and refines through a face split rule (or primal). A 'mask' is a picture showing how the old vertices are used to compute the new vertex for the successive level.

This scheme creates new vertices of valence 6 in the interior. On the boundary, the newly created vertices have valence 4. The vertices with valence 6 or 4 in triangular tilings are called regular vertices. Vertices with other valences are called irregular or extraordinary vertices.

Figure 4.22 shows the masks for regular and irregular vertices. For regular vertices, the values are $a = 1/2+w, b = 1/8 + 2w, c = -1/16 + w, d = -w$ (when $w = 0$ is used it represents the original Butterfly Scheme). For the case when an edge connects an r-vertex $(r \neq 6)$ and a 6-vertex, the neighbors of the r-vertex are used in the stencil as indicated in the second picture in Fig. 4.22. For $r \geq 5$ the weights are given by

$$S_j = \frac{\left(\frac{1}{4} + \cos(2\pi j/r) + \frac{1}{2}\cos(4\pi j/r)\right)}{r} \qquad [4.15]$$

with $j = 0, ..., r-1$. For $r = 3$ we take $S_0 = 5/12, S_1 = S2 = -1/12$, and for $r = 4, S_0 = 3/8, S_2 = -1/8, S_1 = S_3 = 0$. When the edge connects two extraordinary vertices, take the average of the values computed using the appropriate scheme of the previous paragraph for each endpoint. For boundary edges 1-dimensional four-point scheme is used. Hence, by adding the subdivision component as mentioned above to the current mesh geometry, surfaces can be easily represented as subdivision surfaces in the GuTS system.

Chapter 5

Multimodal User Interface

5.1 Interaction Challenges in the GuTS Modeling System

The keyboard, mouse and 2D monitor are the most commonly used input/output (I/O) devices in conventional modeling systems. These input devices are sufficient to provide discrete input, such as input points by clicking a mouse button or inputting a numerical value through a keyboard. These devices are not perfectly suitable for continuous operations, such as tracing that is performed in the GuTS approach. Also in the GuTS approach, most of the time multiple geometries are manipulated while other operations are performed simultaneously. One example is when multiple guide geometries are transformed (positioned and oriented), tracing is done simultaneously. Another example is when multiple operations are repeatedly input sequentially in a short period of time. Another example is interrupting an operation to perform another operation and returning back to the original operation, known as ‘mode switching’. To avoid confusion between the word mode (modality) in multimodal user-interface and in tasks performed, we use the word ‘task-mode’ to represent the latter. So the above ‘mode switching’ will be referred as ‘task-mode switching’. Thus the GuTS modeling approach provides some unique challenges for the interface to a modeling program.

The GuTS system can be implemented with a conventional mouse and keyboard user interface alone. Figure 5.1 shows the commonly used tasks in GuTS and the mapping of those tasks with the conventional I/O devices - if conventional I/O devices are used in the GuTS system. Figure 5.1 shows four columns: first left column shows the tasks list, second column shows the dimension in which the tasks are performed, third column shows the nature of the dimensions of the I/O devices, and fourth column shows the available I/O devices. ‘Mapping’ is a process to match a suitable I/O device and its available degrees-of-freedom (DOF) to perform tasks that require certain DOF. For example a 2D-mouse (2-DOF device) can be mapped to draw on the screen (a 2-DOF task). The same task can be mapped to perform using a keyboard (a 1-DOF device) using ‘up’, ‘down’, ‘left’, and ‘right’ keys. However, in this case 2 DOF drawing task is performed using a 1-DOF device, requiring additional mapping of 1-DOF key movements to perform the 2-DOF drawing task. So to achieve a best match between the list of tasks and available devices, we need to match the equivalent dimensions pair on the second and third column in Fig. 5.1. A significant amount of the tasks in GuTS require mapping with the available conventional I/O devices. This mapping process introduces indirect manipulation of tasks and operations in user interface. Also, limited I/O devices means increased ‘task-mode switching’ to perform tasks and hence increased completion time. While use of the traditional I/O devices may produce a workable interface, it does not address the above mentioned issues in the GuTS modeling approach.

To address these issues, we implemented a novel multimodal user interface in the GuTS system. The term multimodal user interface means use of multiple modes in I/O interaction. In the general sense, a multimodal system is defined [88, 76] as follows: a multimodal system supports communication with the user through different modalities such as voice, gesture, and typing. Literally, 'multi' refers to 'more than one' and the term 'modal' may cover the notion of 'modality' as well as that of 'mode'. 'Modality' refers to the type of communication channel used to convey or acquire information. It also covers the way an idea is expressed or perceived, or the manner an action is performed. 'Mode' refers to a state that determines the way information is interpreted to extract or convey meaning. In user interface design, a mode is "a distinct setting within a computer program or any physical machine interface, in which the same user input will produce perceived different results than it would in other settings" [98]. The modes (modalities) used in the currently implemented GuTS system are: 3D mouse input, pen-data tablet input, speech input, video (graphics and text rendering) output, and synthesized speech output. Our prototype of multimodal user interaction explores solutions to the above mentioned challenges and provides a fluent, direct, and rapid interface. From the author's perspective, multimodal interaction with multiple I/O devices enhances the usability of the GuTS system.

5.2 Humans Natural Multimodal Interaction Capability

Humans have a natural ability to use multiple modes in parallel and choose certain mode to perform certain task dynamically in real-time. Humans use this behavior in several ways for different reasons - such as, to perform multiple coordinated 'sub-tasks' in parallel to complete 'a task' rapidly, and to perform 'a complex task' by breaking it into multiple 'less complex sub-tasks' and complete the sub-tasks using multiple modes in parallel. For example, humans communicate through multiple modes in parallel - such as, hand gestures, speech, hearing, visual cues - with ease and effectively in their day-to-day activities. Another example, driving a car which is a complex task. This complex task involves several less complex sub-tasks, such as steering, accelerating, changing gears, braking, looking at the road, hearing noise from other vehicles, etc. Humans use their hands (to steer and change gears), legs (to accelerate and brake), eyes (to look at the road), ears (to listen to the surrounding noise), etc. Humans perform these multiple sub-tasks in a coordinated fashion to complete the complex task (i.e. driving).

Utilizing this human's natural interaction behavior, the GuTS system applies the phenomenon of multimodal interaction. Our current implementation simultaneously provides the user with two-handed input, speech recognized voice input, improved hand-eye coordination, visual output, and synthesized speech output. Since these modes of interaction imitate the normal communication modes of humans, they present a familiar, effective and direct approach of interaction between the system and the user. Also since the user interacts with multiple I/O modes simultaneously, multiple operations are performed simultaneously in a coordinated way which aid in the rapid completion of tasks.

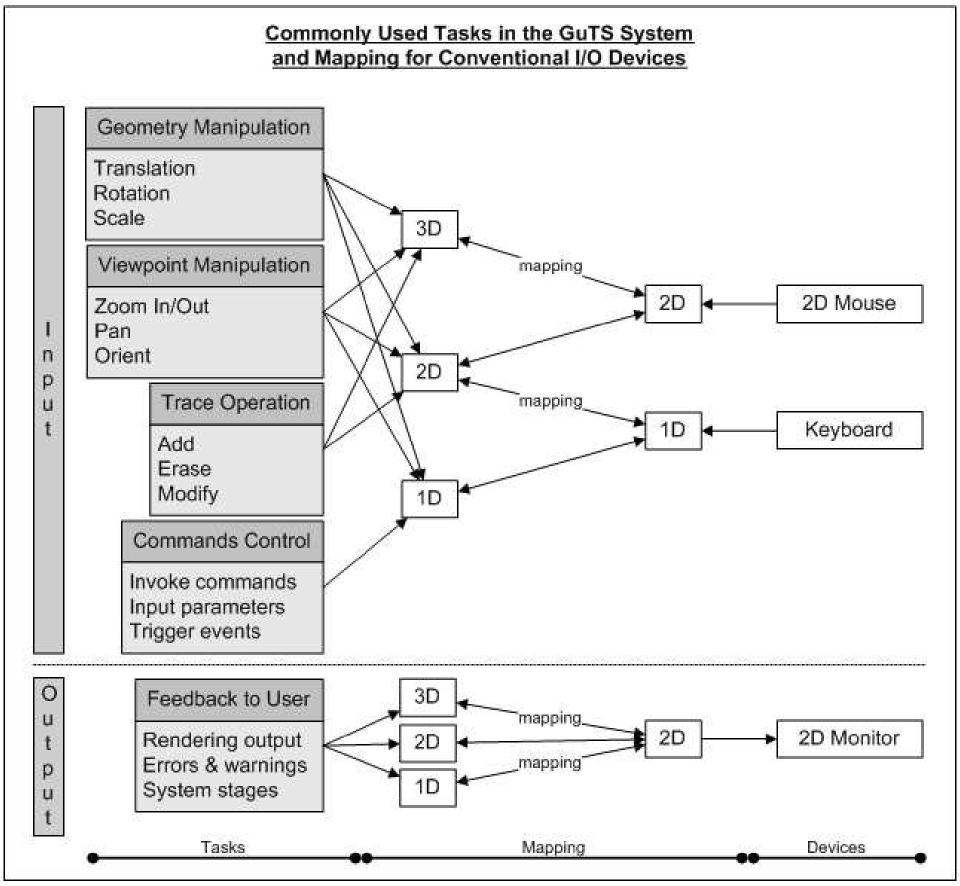


Figure 5.1 Mapping of tasks in the GuTS system for traditional I/O devices (if traditional I/O devices are used)

5.3 GuTS System User Interface Setup

In this section, we describe the GuTS system's user interface (UI) architecture, its multiple modules, pros and cons of each I/O mode and device, and the process of mapping different tasks with multiple I/O devices. Figure 5.2 shows the actual setup of the multiple I/O devices in the GuTS system.

5.3.1 GuTS System's User Interface Architecture

A traditional system's UI architecture with keyboard, mouse, and monitor is relatively straight forward. It requires two event-handlers for keyboard and mouse, and a rendering/windowing module for visual output. A multimodal UI system, like the GuTS system, requires multiple modules to manage multiple I/O devices and modes, and a sophisticated command parser to coordinate multiple commands from different modes simultaneously. Figure 5.3 shows the layout of the GuTS system architecture, its multiple modules, and interaction flow between these modules.

The 'Gesture Recognition' module tracks the input from the pen and processes it to the appropriate data depending upon the stage it is used. For example, when tracing over a surface in 3D, it takes the 2D screen coordinates of the pen and transforms it to the 3D geometric world coordinate system as a ray in the view direction. Then this 3D ray is used to compute the intersection point over the surface, which is further used to generate the trace curve. On the other hand, in the 2D environment to recognize patterns, such as circles, ellipses, lines, etc, from the scribbled input, the 2D distribution of points are used directly. The 'Voice Recognition' module recognizes the user's natural voice input commands processes it, converts it to a command, and sends it to the 'Command Parser'. The 'Speech synthesizer' in real-time dynamically converts text messages (such as errors, warnings, system stages, etc.) to human understandable language and plays them in audio form. The 'Graphics Engine' does the rendering of all 2D and 3D geometries. One of the complex modules is the 'Geometry Engine'. This module contains all the data structures and algorithms for both the curves and surfaces to perform all the necessary geometric computations. Another complex and core module is the 'Command Parser'. This module's primary task is to coordinate all the input and output command interactions between multiple modules. The 'Command Parser' receives multiple inputs and outputs concurrently, and it does the critical job of recognizing the necessary information, eliminating redundant information, and invoking the appropriate operation.

The most commonly performed tasks in the GuTS system are:

- Manipulating geometries (guide shapes and regular shapes)
- Manipulating viewpoint
- Tracing operation
- Invoking commands

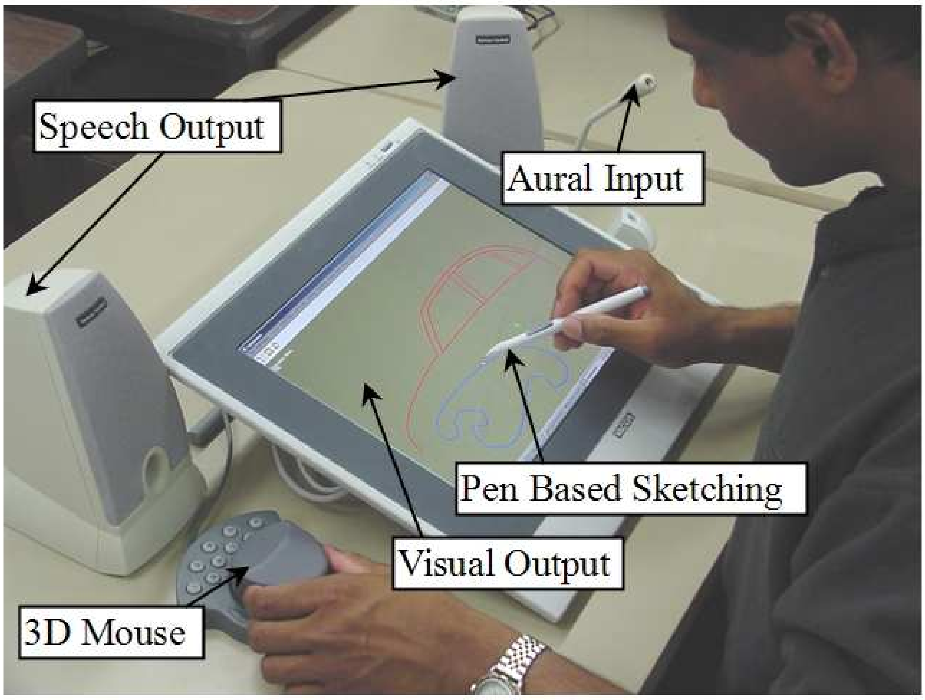


Figure 5.2 Configuration of I/O eevices in the GuTS system

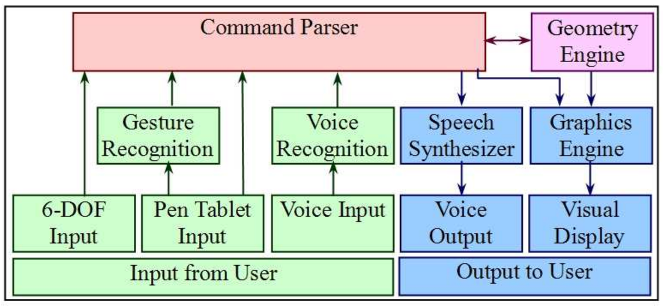


Figure 5.3 Layout of the GuTS system's user interface architecture

- Rendering visual output
- Providing other feedback (such as errors, system stage, etc.) to the user

Figure 5.4 shows the commonly used tasks in GuTS and the mapping of tasks with the I/O devices used. Figure 5.4 contains four columns: first left column shows the list of tasks, second column shows the dimensions in which each tasks are performed, third column shows the dimensions of the available devices, and the fourth column shows the available I/O devices. It can be observed from Fig. 5.4 that the list of tasks, I/O modes, and devices are matched to eliminate the mapping process. In our current implementation, we used a 2D display output (shown in gray color) but the GuTS system allows rendering the visual output in stereoscopic 3D rendering (shown in dotted line). In the GuTS system, fluency and rapidity are achieved - through mapping with right devices, by allowing direct manipulation of tasks, and the simultaneous use of multiple I/O devices which minimize 'task-mode switching' in the modeling process.

Manipulation of geometries and viewpoint controls are 6-DOF operations in the 3D environment. The 3D mouse provides direct 6-DOF input and is used for manipulating geometries and viewpoint controls. This provides direct manipulation rather than mapped (between 3D and 2D) interaction which is performed using a keyboard or regular mouse. In the real world, tracing or drawing is performed using a pen or pencil. The direct fluent nature of tracing is achieved by using the pen-data tablet in the GuTS system.

In the GuTS approach the user must manipulate both the current shape as well as the guide shape. To achieve natural interaction we use a two-handed interface that mimics the real world drafting process. As shown in Fig. 5.2, the user holds a stylus-pen in his dominant hand and a 6-DOF 3D mouse in his non-dominant hand. The pen in the dominant hand sketches and traces the curve or region of a surface. The dominant hand is used for precise input while the non-dominant hand controls imprecise manipulation. This allows the user to control a set of operations without interrupting the current operations in each hand. Though both the hands are used at the same time, the tasks performed are not totally independent. They are coordinated tasks that benefit from the use of the two-handed approach. Some of the earlier research [19, 40, 69, 36] also used a two-handed approach and their experimental results also highlight the benefits of a bi-manual approach as opposed to uni-manual interaction.

In conventional systems, invoking commands are usually performed through menus. While the GuTS system provides menus, it also uses voice inputs to invoke commands. If the commands are invoked through menus, it disrupts the operation that is performed in the dominant hand, causing task-mode switching and wasting of time. Geometries are rendered in the data tablet display. More about the display, window, and organization of several controls is discussed later in this chapter. Synthesized voice output is used in the GuTS system, which is considerably different than prerecorded audio output. In synthesized speech output the dialogues are generated in run-time. This provides the user proper feedback of the system and error messages while guiding the user on what to do further in corresponding to real-time interaction, rather than with some pre-recorded audio messages.

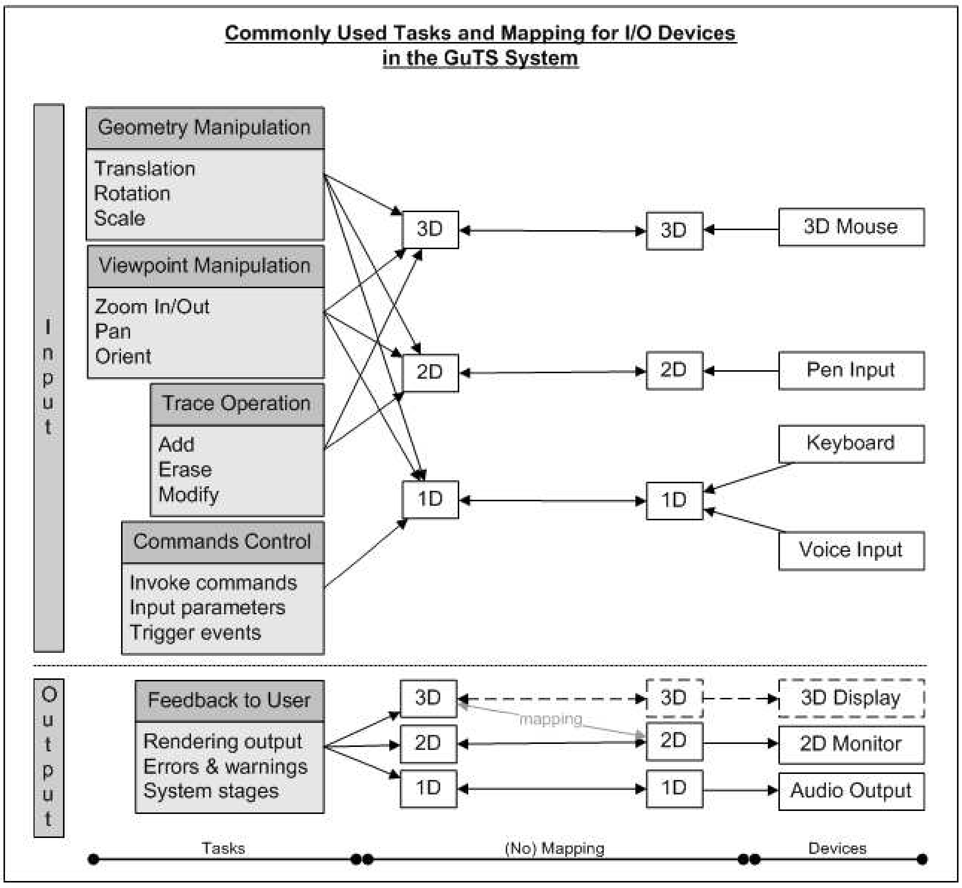


Figure 5.4 Mapping of tasks with I/O devices in the GuTS system

Thus the major tasks in the GuTS system are divided and distributed to the several available I/O modes, to perform each task directly, fluently and rapidly. Each of these modes' pros and cons are analyzed and and its applicability in the GuTS system are discussed in detail in the following sections.

5.3.2 Pen-Based Input

- Pros:
  - Provides WYSIWYG input interface and direct interaction to draw and sketch (like pen-paper interaction).
  - User is already familiar with this type of interaction (like pen-paper interaction) - little or no learning curve.
  - Provides additional functionalities, such as 2D mouse mouse button clicks, pressure sensitive tip, and erase tip at the back of the pen.
- Cons:
  - Only 2D interface - while good for direct interaction for 2D tracing, requires mapping (though minimal) to trace in 3D.
  - Due to the human nature, the draw input using a pen always contains some noise (wiggles). We eliminate this noise through guided tracing.

A pen in the dominant hand is used to sketch and trace a curve over the surface. Using a flat panel display with an integrated data tablet allows the user to draw curves directly on the monitor as if drawing on a sheet of paper. This configuration enhances hand-eye coordination, since the hand pen input position and the visual geometry output is at the same point, unlike regular mouse/monitor interface, where hand input positions are at a different position from the visual display.

Figure 5.5 shows several parts of the pen and its functionalities. These types of pens are becoming standard as tablet input has become widely used in recent years. In this research, we did not modify or change any of the hardware of the pen but applied them effectively for drawing and tracing operations that provide fluent and direct interaction. The two buttons are similar to the left and right buttons on the regular mouse. The front and back end of the pen is pressure sensitive. As the user applies certain pressure, the tip is activated for input. The front tip is used for tracing/drawing and the back tip is used for erasing the curve. This avoids task-mode switching and uses both drawing and erasing operations interchangeably by just flipping the pen.

5.3.3 6-DOF 3D Mouse Input

- Pros:

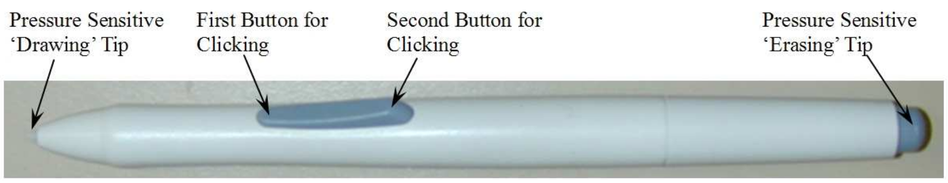

Pressure Sensitive 'Drawing' Tip
First Button for Clicking
Second Button for Clicking
Pressure Sensitive 'Erasing' Tip

Figure 5.5 Pen data tablet and its functionalities

  - Provides direct 3D interaction (6-dof).
  - Works as an additional input mode, enables the use of non-dominant hand and two-handed interaction.
  - A good 6-dof device for a desktop based application - requires minimal physical movement. Unlike 3D-trackers [such as ascension, polhemus trackers] [2, 8] that requires large hand movement in 3D space - causing fatigue to user preventing use for a prolonged period.

- Cons:

  - Imprecise input - difficult to control in a precise manner. We use different snapping mechanisms to achieve precision from imprecise input.
  - While it provides 6-dof control, it does not provide direct 3D manipulation as in real-world. This is unlike real 3D trackers [ascension, polhemus trackers] [2, 8] that allows moving in real 3D space to input 3D transformation.

A key advantage of the 3D mouse over 2D devices is that it allows direct control of manipulating the geometries in 6-DOF in the 3D environment. The 3D mouse is operated using the non-dominant hand. The non-dominant hand is mainly used to control imprecise manipulations such as the guide shapes where automated snapping mechanisms enable precision from the imprecise input. Apart from controlling the guide shapes, other coarse level controls such as transforming the geometries, zooming and scrolling are also performed using the 3D mouse in the non-dominant hand. This allows the user to move around the geometries without diverting from the current operation that is performed using the dominant hand. Since the non-dominant hand operates at an imprecise level, mechanisms such as automatic snapping are used to achieve the required precision rapidly. The two input devices on the dominant hand and non-dominant hand are synchronized to achieve two-handed bi-manual interaction. This allows the user to perform a complex task using the coordination of both the hands simultaneously, instead of operating single handedly as in the conventional mouse system.

Figure 5.6 shows the multiple parts of the 3D mouse in the 2D setup. Again in our current implementation, we did not modify or change any of the hardware of the 3D mouse, but applied it effectively for performing 3D operations that provides fluent and direct interaction. Eleven different buttons are used as short-cut keys to invoke different functions.

The functions of these buttons are assigned such that they assist the tasks that are performed using the 3D mouse. These functionalities provide additional control to the non-dominant hand and help perform additional tasks without interrupting other operations performed using other modes. Some of the functions assigned to these buttons are:

- Toggle the manipulation control to guide geometries
- Toggle the manipulation control to regular geometries
- Toggle the manipulation control to viewpoint controls
- Scale up the size of geometries
- Scale down the size of geometries
- Lock the translation component
- Lock the orientation component
- Reset the scene and viewpoint

Figure 5.7 shows additional operations that are performed in the 3D setup. In 3D, three translational and three rotational components are used directly. As the user switches between the 2D and 3D environment, the 3D mouse changes its configuration internally and automatically. Also the mouse buttons reassign their functions accordingly without any additional user input.

5.3.4 Speech Input

- Pros:
  - Works as an additional input mode without interrupting other input modes.
  - User's natural mode of communication with little or no learning curve for the user.
  - Fast to input the commands through speech.
  - Speech recognition and speech input are slowly becoming a mainstream feature in the user interface. For example, the latest Windows operating system 'Vista' comes with enhanced speech recognition capabilities built right into the application. This capability allows several applications and users to use speech input as one of the primary user interaction mode.
- Cons:

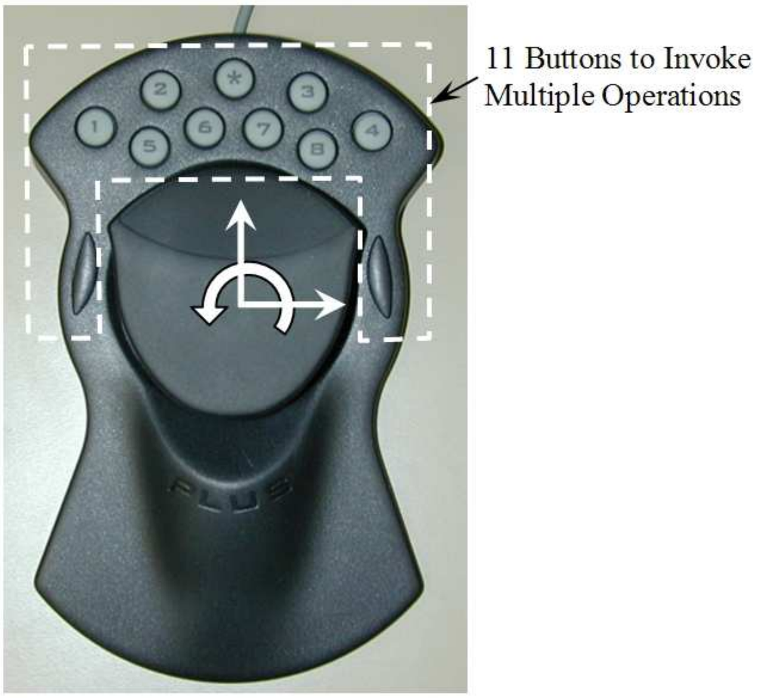


Figure 5.6 3D mouse and its functionalities in the 2D mode

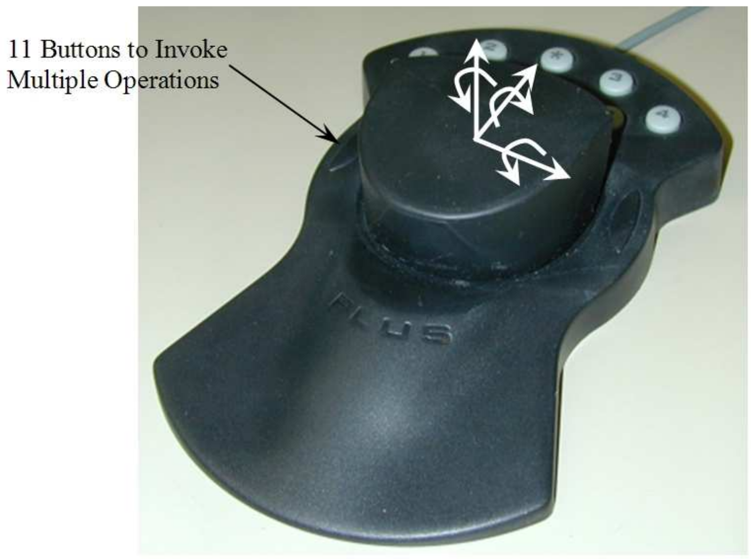


Figure 5.7 3D mouse and its functionalities in the 3D mode

- Though speech-recognition has improved a lot in recent years, still it is not 100% accurate. Making it not suitable for mission-critical applications.
- User dependent - though not mandatory, still requires user training to achieve good speech recognition rate, and very sensitive to each user's accent, speech clarity, speaking style, vocabulary, rate of speech, etc.
- Susceptible to the ambient noise, and causes interruption to other users in the surrounding.
- User has to remember (all) the input commands, and the list of commands may become long making it impossible to remember all the commands.
- Invoke unwanted commands due to false-recognition. False-recogntion and confusion in recognition between 'similar sounding' words or phrases.
- Not suitable to recognize natural spoken sentences and extracting commands from the recognized speech (full sentences) is complex. For example, the following two sentences - "I would like to draw a semi-circle of radius 5 centimeters" and "sketch an half circle that is 100 millimeters in diameter" - refers to the same task, but it is algorithmically difficult to parse and process the commands correctly.

As seen above, the GuTS system effectively uses both the user's hands with pen and 3D mouse input devices. While both hands are engaged in an operation, to invoke any additional commands such as the menu operation, the current operation has to be paused or interrupted. Then once the menu operation is invoked, the previous operation is resumed or started again ('task-mode switching'). Task-mode switching interrupts the current operation and workflow of the user and introduces additional unnecessary gaps in the design process. Additional speech input mechanisms provide the additional mode of input without interrupting the tasks performed by both hands and eliminates task-mode switching.

For example when the left hand is used to transform the geometry and the right hand is used for tracing, voice input can be used to issue commands such as 'display wireframe rendering' (or) 'display shaded rendering' to change the type of display required at that instance. This eliminates the need for the user to browse through the menus or toolbars to invoke that command and resume the tracing operation.

5.3.5 Flat Panel Tablet Display

- Pros:
  - Provides WYSIWYG visual output.
  - Enables hand-eye coordination, because user sees the pen and the visual display directly and at the same spot.
- Cons:

  - Limited resolution and screen display area.
  - Not suitable for stereoscopic display.

A flat panel display with an integrated data tablet allows the user to draw curves directly on the monitor as if drawing on paper (refer Fig. 5.2). This setup works as both the input and output mechanism, and enhances the effective hand-eye coordination of the user.

Figure 5.8 shows the configuration of windows in the GuTS system. Menu commands can be invoked through pen tablet or voice input. The text display shows the system statistics and I/O device status of multiple modes, such as which I/O modes are on/off, 3D mouse control status, rendering frame rate, etc.

The controls at the right-hand side are grouped into curves, surfaces, and view/render controls. Based on the working task-mode, these controls are brought to the foreground. Each of these control panels is shown in Fig. 5.9. These control panels are shown here to specify the arrangement of additional controls in the windowing system in the GuTS system.

5.3.6 Synthesized Speech Output

- Pros:
  - Provides an uninterrupted additional mode of output.
  - Does not distract (or) deviate the user's visual focus from the current task - unlike error messages or dialog boxes that disrupt the user's visual focus.
  - Natural language feedback to the user - easy and clear to understand.
  - Synthesized speech output slowly becoming a mainstream capability. For example, the operating system Windows Vista comes with a new human-sounding speech synthesizer. This enhanced capability, will allow several applications and users to use synthesized speech output in their day-to-day use.
- Cons:
  - Ears can get overwhelmed and sensitive after hearing auditory feedback for a prolonged period of time.

It is important to provide the user with a steady stream of feedback and unambiguously inform the state of the system. In addition to the visual output and cues such as graphical displays, symbols and colors, we use auditory feedback with synthesized speech output to provide additional uninterrupted feedback. Speech output also becomes important when the graphical display of the feedback is not sufficient.

For example, when a user tries to revolve a curve without defining the centerline, the system speaks out saying 'no centerline exists, create centerline'. In some cases, voice output is used as a mechanism to guide the user on

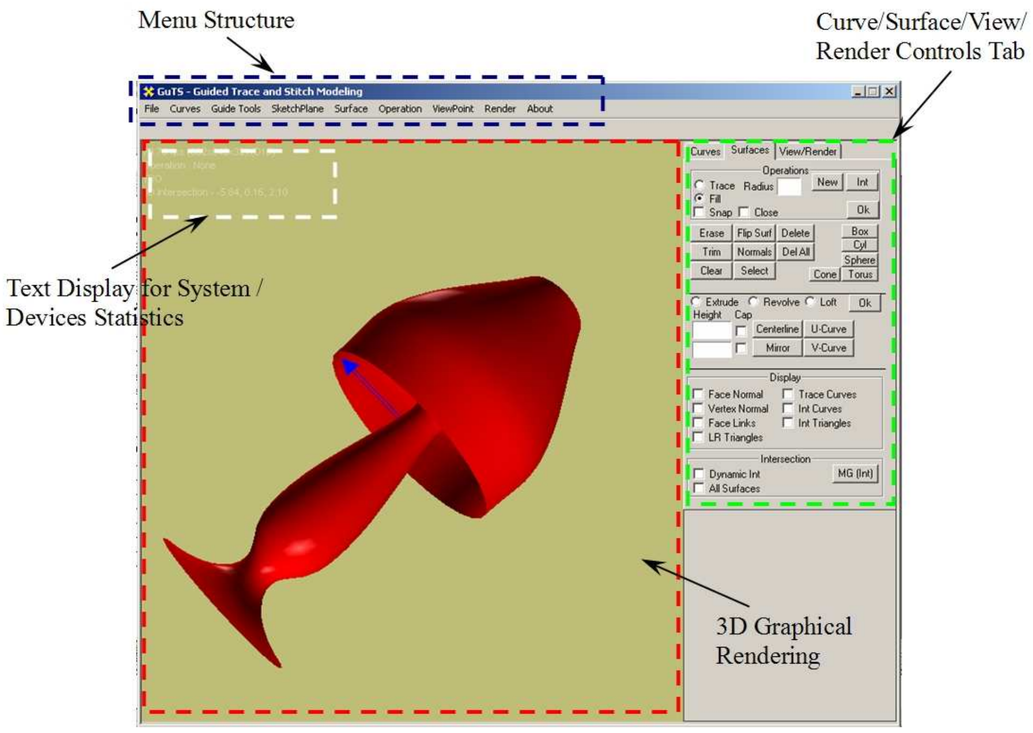


Figure 5.8 Windowing in the GuTS system

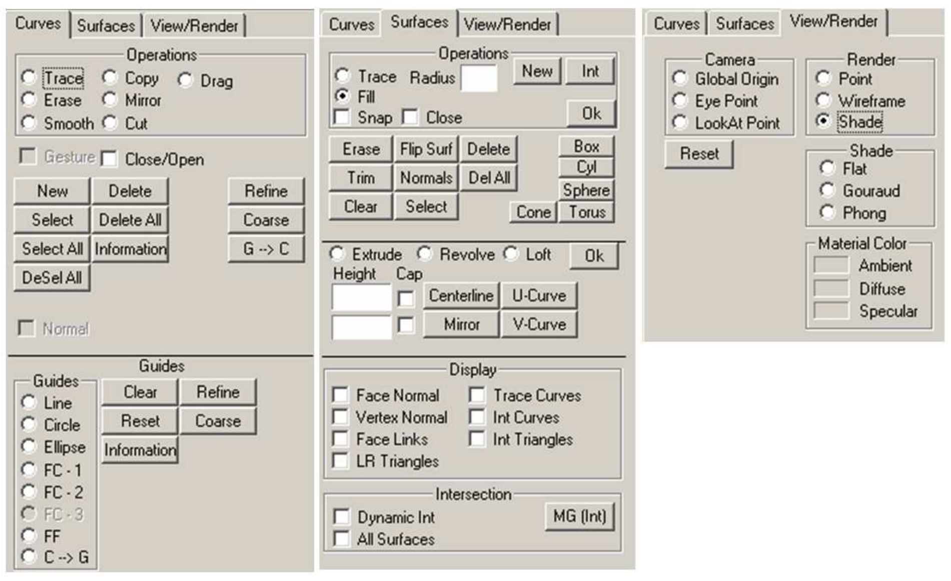


Figure 5.9 Control tabs in the GuTS system

what to do further. This assisting mechanism helps the user learn the system faster without the need to remember the sequence of operations. Also, this leads to interfaces that allow the user to keep his visual attention focused on the current operation, without the distraction of dialog boxes or text output windows as in the conventional application. For example every time there is an error, it is shown in a text window and the user shifts his attention from the graphics window to the text window to read the message and then have to focus back to the graphics window. The additional auditory output eliminates switching the user's attention.

5.4 Technical Specifications

The GuTS system is written in Visual C++. It uses DirectX's Direct3D [3] as the underlying graphics library. The current implementation runs on Microsoft Windows-based systems. The following table shows the hardware and software components of the input and output devices of the GuTS system. This information will be helpful for the readers in recreating the GuTS system.

I/O Device	**Software components**	**Hardware components**	**Company information**
Pen Data Tablet	Data Tablet Driver from Wacom	Wacom	www.wacom.com
3D Mouse	3D Mouse Driver from Logicad	Magellan	www.logicad3d.com
Speech Input	Microsoft speech recognition API (or) IBM Via voice speech recognition system	Generic Microphone	www.microsoft.com www.ibm.com
Synthesized Speech Output	Microsoft speech synthesizer API	Generic speaker set	www.microsoft.com

Table 5.1 Specifications of multimodal user interaction components

5.5 Variations of Modeling Applications with Multimodal User Interface

The above discussed I/O devices configuration in this chapter is one example of how multimodal user interface can be used in a modeling system (i.e. the GuTS modeling approach). The above discussed (geometric) modeling and multimodal user interface concepts can be extended to a variety of different applications. For example, we present two different applications - 'Molecular Modeling System' and 'Detailed Virtual Design System' (DVDS) - that were developed by the author that demonstrate how the above discussed modeling and multimodal user interface concepts can

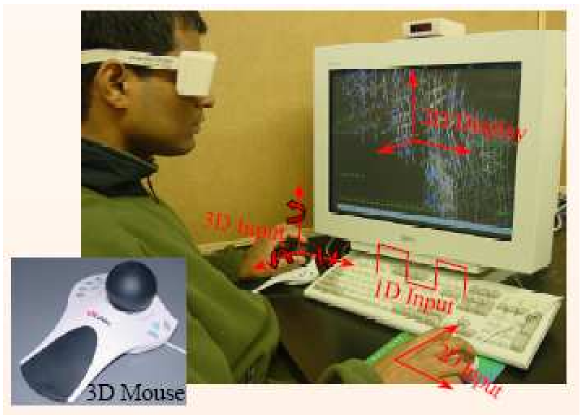


Figure 5.10 Molecular modeling using ‘Fishtank VR’ interface

be adopted for different applications. These two examples demonstrate the applicability and similarity of multimodal user interface design in this research for various applications.

5.5.1 Molecular Modeling System

First application is a molecular modeling system with multimodal user interaction in an immersive virtual environment. In this application [13], we developed an environment that visualizes and models complex protein structures, 1,000s to 10,000s (or more) of atoms per molecule. This framework allows modeling and manipulating molecular structures through intuitive natural interaction, direct manipulation in 3D, and a time-efficient interactive environment.

Figure 5.10 shows an user interface setup very similar to the GuTS system, but with a couple of differences - use of a 3D stereoscopic display and use of a regular 2D mouse instead of a pen tablet. Extending this research to a large immersive scale, we developed a detailed complex multimodal environment for molecular modeling. Since the stereoscopic display provides depth perception (compared to a 2D display), it helps the user to visualize, analyze, and understand complex molecular and protein structures in 3D with ease.

Figure 5.11 shows different components that are used in this molecular modeling system. The readers will notice the similarity of the different user interface components between this system, the DVDS system (refer Section 5.5.2), and the GuTS system. Some of the differences are use of head tracking and additional controls for secondary users allowing collaborative modeling and visualization between multiple users.

Figure 5.12 shows the detailed system architecture of the molecular modeling system. In this case three dedicated computers were used for different tasks. First, a dedicated graphics hardware for 3D stereoscopic rendering. Second, a dedicated computer for processing input 3D trackers and gestures from the gloves. Third, a dedicated sound server for processing the audio input and output. All three computers are connected through a network and work synchronously and seamlessly with the user’s interaction. While the overall concept of the multimodal user interface is similar to

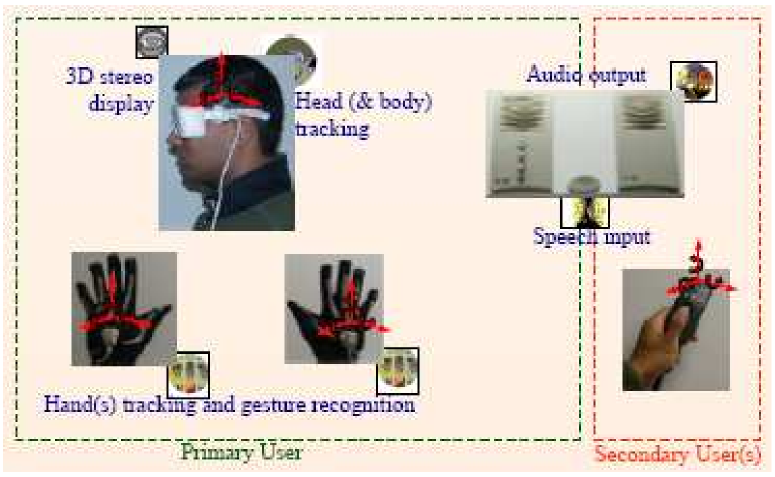


Figure 5.11 Different components in the molecular modeling system

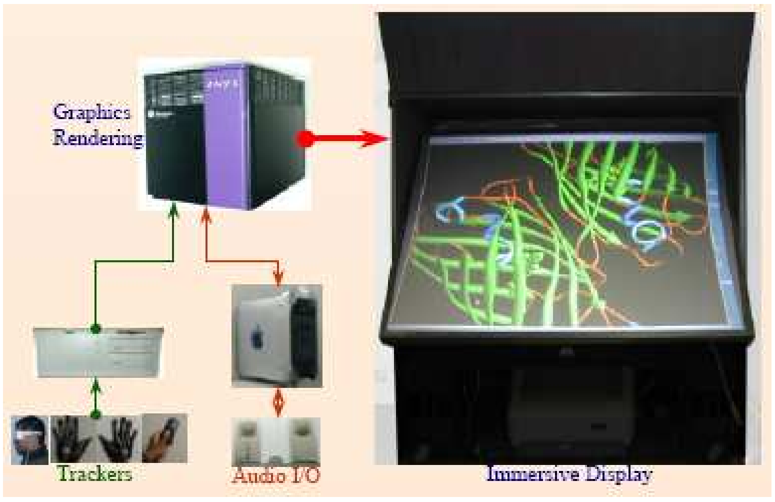


Figure 5.12 System architecture of the molecular modeling system

the DVDS and GuTS systems, this application setup demonstrates the scalability of this concept to achieve a high performing distributed environment. More importantly, in all the applications (the GuTS System, DVDS, and the molecular modeling system) the underlying analysis and mapping of different available I/O devices to the tasks that need to be performed remains the same. This demonstrates that the concept of this multimodal user interface design is scalable and it can be applied, modified, extended to any number of applications with varying I/O devices.

Figure 5.13(a) shows the user modeling in front of the immersive display. Figures 5.13(b) and 5.13(c) show the virtual hand corresponding to the user's real hand. Readers can observe the virtual fingers in the virtual hand exactly replicate the real fingers of the user. In the following sections, we present how the tasks and I/O devices are mapped for this application.

5.5.1.1 Set of tasks

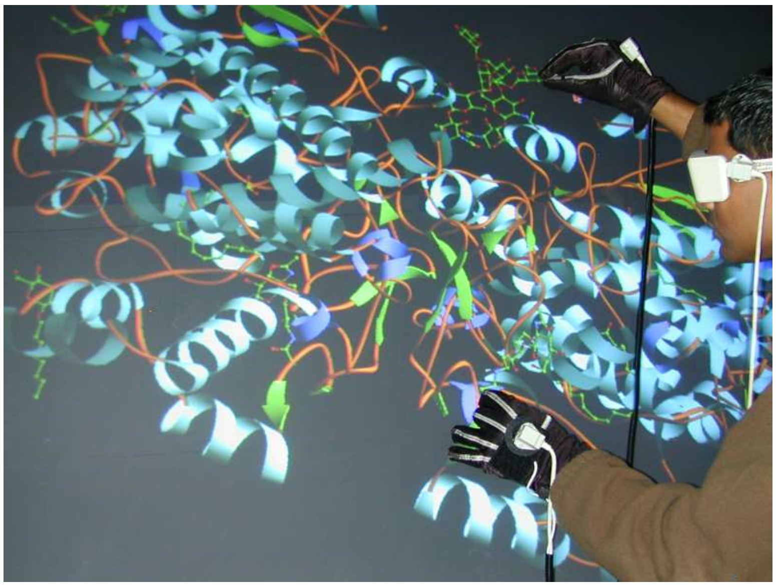
The first column in Table 5.2 lists the major tasks that are commonly performed in a molecular structure visualization, modeling and fitting program. The word (F) in column 1, represents that the particular task is a filtered task of some other task. For e.g. 3D position is a filtered operation of 3D position and orientation, where orientation parameters are filtered out. Similarly, "Zoom In" task is a filtered operation of 3D position, where the translation of camera viewpoint is allowed in only one dimension. It is important to mention that in this section we focus on the task, rather than how or what technique is used to perform that particular task. For example, we focus on task "select an object", and not how it can be selected such as ray casting, collision detection, or spatial proximity.

Second column shows the required degrees of freedom for each task. Third column shows the nature of task, i.e. continuous (C), discrete (D), repetitive (M), and need visual guidance (V). The Tag (V) means, during that particular task the user need to visually see the changes to perform that task correctly. In short, they are visually dependent tasks. Forward slash, for e.g. C/D means that task can be either continuous or discrete. Having grouped the tasks, the next step is to identify the best suitable I/O modes to perform each of these tasks that provides natural, intuitive and effective interaction to the user.

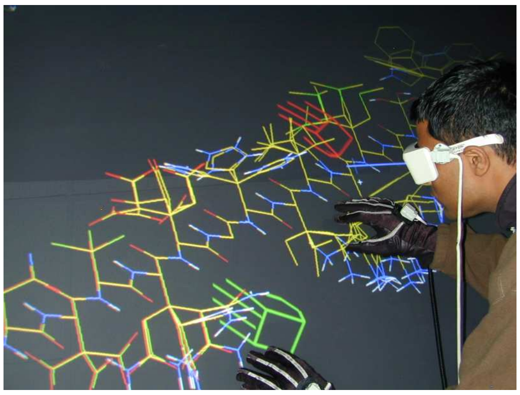
5.5.1.2 Matching tasks with I/O modes

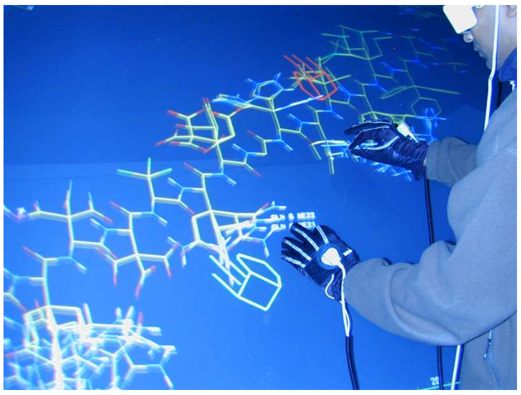
Our approach to identifying the best suitable mode is to assign a similar mode that a user will commonly use in a real world scenario. Matching tasks with multiple I/O modes is totally different than mapping the keyboard and mouse events in a desktop application. It is mainly because in multimodal interaction, the number of I/O devices increases several fold, dimensionality of the I/O devices also changes considerably and the I/O modes vary dramatically. The available input devices and its DOF and its nature can be listed as follows:

Once the DOF, the nature of the devices, and tasks are found, it is easy to find the most suitable I/O modes for the task. The fourth column in Table 5.2 lists the most suitable I/O modes for each task. Table 5.4 provides the physical connection between the human part, I/O devices, and the assigned tasks.



(a)





(b) (c)

Figure 5.13 User modeling molecular structure in front of the immersive display

Tasks	**DOF**	**Nature**	**I/O Mode**
Camera Navigation			
3D Position, Orientation	6	CV	TG
3D Position (F)	3	CV	TG
3D Orientation (F)	3	CV	TG
Zoom In/Zoom Out (F)	1	C/D,V	TG/A
Look Along X Axis	1	D	A
Center and Zoom Atom H	1	DM	N
Geometry Navigation			
Go to Atom H	1	D	A
Go to Previous Atom	1	D	A
Go to Next Residue	1	D	A
Modeling/Editing			
3D Position, Orientation	6	CV	TG
3D Position (F)	3	CV	TG
3D Orientation (F)	3	CV	TG
Torsion Angles (F)	1	CV	TG
Delete Atom/Residue	1	D	A
Apply/Cancel Changes	1	D	A
Undo Last Change	1	D	A
Geometry Selection			
Pick Atom (by pointing)	1	DV	TG
Pick Atom (by name)	1	D	A
Geometry Query			
Get Distance	1	D	A
Get Atom Information	1	D	A
Feedback To User			
Error Report	1/2/3	D/C,V	S/I
Statistics	1	DV	S/I
Visual Cues	1/2/3	CV	I

Table 5.2 List of commonly performed tasks

C: Continuous, D: Discrete, V: Visual, F: Filtered, T: Spatial Trackers, G: Glove Gestures, A: Audio Input, N: Natural Language Input, S: Speech Output, I: Immersive Display

I/O Devices	**DOF**	**Nature**
Spatial Trackers (T)	6	CV
Gloves (G)	1	D
Audio Input (A)	1	D
Natural Language Input (N)	1	DM
Immersive Display (I)	3	CV
Synthesized Speech Output (S)	1	D

Table 5.3 Available I/O devices, its DOF and nature

Human Part	**I/O Devices**	**Tasks**
Left hand	Tracker, Glove	Camera navigation
Right hand	Tracker, Glove	Geometry edit/select
Head/body	Tracker	Camera navigation
Mouth	Voice input	Audio commands
Eyes	Visual display	Visual feedback
Ears	Speech output	Auditory feedback

Table 5.4 Physical connection between human parts and I/O Devices

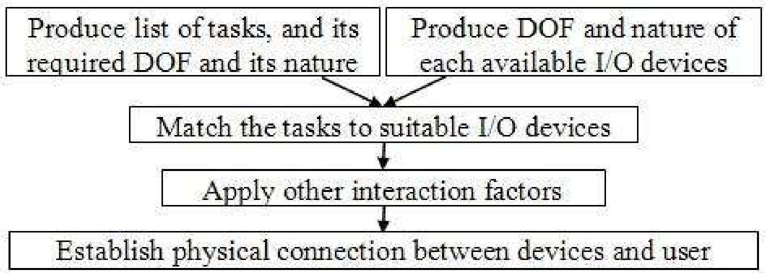

Produce list of tasks, and its required DOF and its nature
Produce DOF and nature of each available I/O devices
Match the tasks to suitable I/O devices
Apply other interaction factors
Establish physical connection between devices and user

Figure 5.14 Matching tasks and modes

In short, camera navigation operations are grouped as ‘viewmode’, geometry editing and selection operations as ‘editmode’. Then by default, left hand for ‘viewmode’, right hand for ‘editmode’, head/body for the fine control of ‘viewmode’, and wand in the secondary user can be used either in ‘viewmode’ or ‘editmode’. The mode of each hand can be easily changed from ‘viewmode’ to ‘editmode’ and vice versa. When both the hands are in the ‘editmode’, both hands can be used to pick two different atoms simultaneously. In this mode, it is easy to pick two atoms and issue a voice command such as “get distance”, to get the distance. Similarly, as the user feels appropriate, different configurations can be set dynamically in real time. Also a particular mode can be turned on or off. For example, by issuing voice commands such as ‘start head tracking’ or ‘stop head tracking’, head tracking can be started or stopped.

Figure 5.14 shows the overall procedure of matching the tasks with I/O modes and establishing the physical connection with the user. Currently these steps are configured manually. This can be automated by knowing the list of tasks, availability of the number of modes and their functionalities, and its nature. Once these are known the appropriate I/O mode can be assigned automatically for each task. Other parameters such as relative/absolute, precise/coarse, natural human practices, etc., play a role in assigning the tasks. For example coarse actions are assigned to non-dominant hand and precise actions are assigned to dominant hand.

The current approach works well and is easy to customize for several operations and configurations. The integration of multiple modes for a very complex interaction environment needs a sophisticated failure handling system. For example if any of the modes fail, without much disturbance to the user, the system should be able to infer the input from the other available modes.

5.5.2 Detailed Virtual Design System (DVDS)

The second example application is ‘Detailed Virtual Design System’ (DVDS) [12]. This application contains a framework for detailed geometric modeling in a multimodal virtual environment. Detailed design involves the development of a detailed model of the product. This includes defining the features, determining their dimensions, tolerances, etc. Almost all of the CAD systems are developed for detailed design. But due to the interface and interaction techniques, 1D and 2D I/O devices, they make shape modeling a time-consuming and tedious task. The

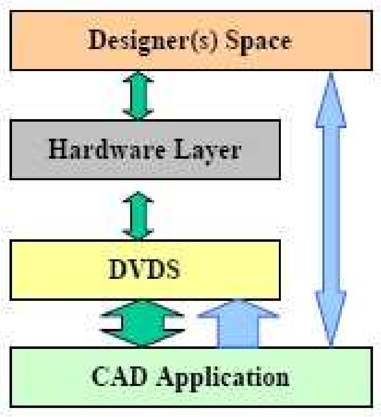


Figure 5.15 Layout of DVDS architecture

nature and architectural development of the DVDS provide higher dimensional multimodal interaction between the designer and the system, thus making the design process faster and easier.

The DVDS system supports multiple input-modes (hand motions, gestures, and voice commands), and multiple output-modes (stereoscopic rendering, and synthesized speech/sound output). Some of the important features of this framework include: multimodal interaction between the designer (human-being) and system (Benefits: Increased communication between the designer and the system), multiple designers simultaneously collaborate in the same environment (Benefits: Effective collaboration between designers, avoids ambiguity and reduces design time), and seamless integration with the existing industry standard CAD systems (Benefits: Avoids data translation and paves the way to merge novel VR technologies to embrace well developed industry standard CAD knowledge).

Figure 5.15 and Fig. 5.16 show the system architecture of DVDS. DVDS is an intermediate software layer, which resides between the hardware and the commercial CAD system. DVDS' command parser module synchronizes the multi-modal inputs, which arrive simultaneously from different kinds of input devices. It parses the input commands and redirects them accordingly to the CAD system to activate any geometric manipulations and/or to the graphics engine for display and navigation operations.

You will notice the similarity between the GuTS system, molecular modeling system, and the DVDS system architecture. Some of the key differences are: in DVDS, use of a commercial CAD system as the underlying geometric engine, use of an immersive stereoscopic 3D display, and fingers-gesture recognition. Also, in DVDS the modeling is done differently than in the GuTS system. In DVDS, the user manipulates the control points and parametric values directly to simulate the CAD modeling system, but using direct multimodal user interface.

Table 5.5 shows the generic sketch entities that are supported in DVDS. Each sketch entity has a set of control points, which help to change the dimensions and modify the shape of the sketch, and a handle, which is used to transform the sketch entity. The handle is mostly at the geometric center of the sketch entity, as shown in Table 5.5.

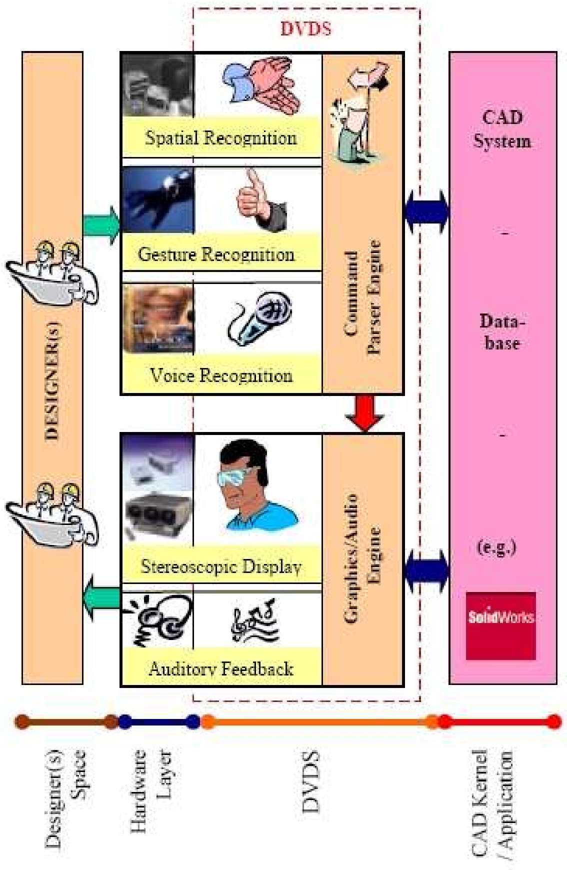
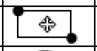
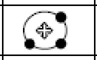
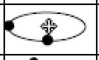
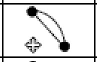
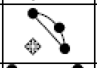
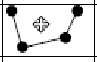
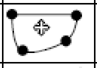
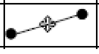


Figure 5.16 DVDS - System architecture

Sketch Primitives	Control points	Description
Rectangle		Two extreme points
Circle		Center and radius
Circle		3 points on circumference
Ellipse		Center, point along major and minor axis respectively
Arc		Center, starting point and ending point on arc
Arc		Start, End and another point on arc
Polyline		End points of each line segment
Spline		Spline along the control points
Centerline		End points of line segment

● Control Points ✥ Moving handle

Table 5.5 Generic sketch entities and its control points and handle

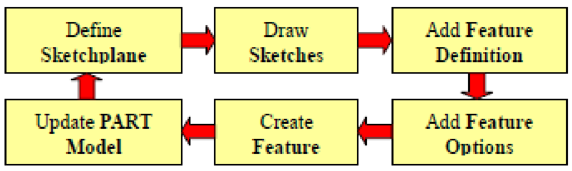


Figure 5.17 Sequence of feature manipulation operations in conventional CAD systems

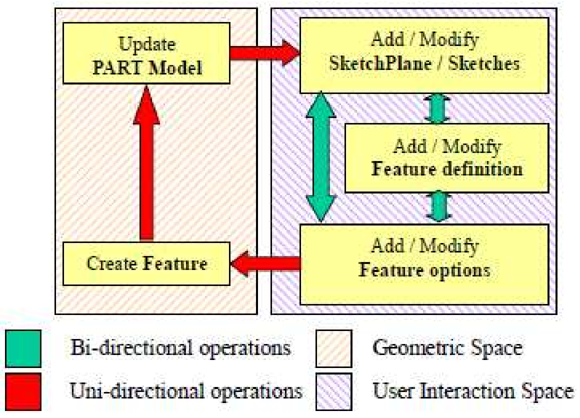

Update PART Model
Add / Modify SketchPlane / Sketches
Add / Modify Feature definition
Add / Modify Feature options
Create Feature
Bi-directional operations
Uni-directional operations
Geometric Space
User Interaction Space

Figure 5.18 Sequence of feature manipulation operations in DVDS

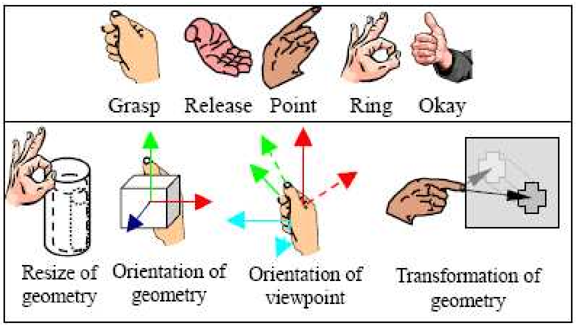


Figure 5.19 Gestures in DVDS

Due to the architectural design of the conventional CAD systems' interface, the sequence of operations in CAD systems is unidirectional, as shown in Fig. 5.17. On the other hand, in DVDS the sequence of operations is bi-directional (refer Fig. 5.18) taking advantage of the multiple modes. For example this allows the designer to dynamically change the feature definition and sketch parameters simultaneously using both his/her hands, while changing the viewpoint orientation using a different mode of input.

In DVDS, a data glove is used to capture detailed movements of fingers and recognize finger gestures dynamically in real-time. Figure 5.19 shows different gestures and their mapping to geometric and viewpoint manipulations. The gestures are grasp, release, point, ring and okay. Grasp is used to grab the object or viewpoint to orient in 6 degrees of freedom (DOF). Release is a notification of end of grasping gesture. Point gesture is used to relocate the model or select a small point by pointing at it. This gesture is useful in manipulating tiny features. Ring gesture is used to reshape, modify the feature by pulling, pushing, twisting the control points. Gesture okay is to confirm the action. The primary operations are freeform transformation and constrained transformation of the part or viewpoint. Usually the viewpoint is changed so that the accuracy of the geometry is maintained. Another kind of operation is zoom in and zoom out. These operations are manipulated through direct hand motion and gestures. Apart from gestures, voice input is also used for invoking commands.

Figure 5.20 shows the user in front of the immersive large display performing modeling. In this figure you will notice floating 3D menus (to replace 2D menus on the desktop systems) on the right side of the picture, providing menu controls to the user in the virtual environment.

These two example applications demonstrate that the concepts of multimodal user interaction used in the GuTS system can be adopted to (a) a wide variety of application areas, (b) with a wide variety of application-specific I/O modules, (c) with a wide variety of application-specific custom hardware devices, and (d) with a varying degree of complex distributed environments. In fact, the GuTS system itself can be extended to use the following components: large immersive display systems that use 3D stereoscopic display, gloves for finger gesture, 3D trackers for hands and

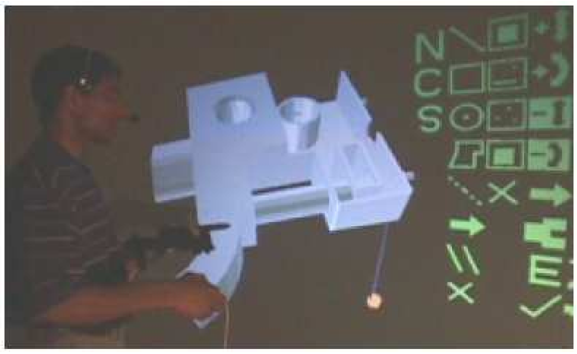


Figure 5.20 User in front of DVDS immersive display

head tracking, and use of large-scale touch-enabled two-handed immersive desks for replicating life-size drafting and modeling platforms.

Chapter 6

Results

In this chapter we present examples of curve geometries and 3D surface models that we created using the GuTS system. These examples demonstrate several features of the GuTS approach that are implemented in the GuTS system. Also, readers can observe that in addition to GuTS-specific operations, traditional modeling operations, such as geometry mirroring, extruding, revolving, and lofting operations, are extensively used in creating these examples. Some of the key points to observe from these examples are:

1. Use of guide geometries to create new geometries through tracing

2. Create new geometries easily without knowledge of the underlying geometric representation

3. “What You See Is What You Get” (WYSIWYG) modeling approach

4. Create complex geometries easily and rapidly through multiple guide geometries

5. Create precise geometries that are the exact replica (or portion) of the guide geometries

6. Create geometries the same way regardless of the underlying geometric representation

6.1 Examples of Curve Geometries

Figure 6.1 was created using circular guide shapes and one elliptical guide shape. This drawing was completed in less than 30 seconds. As this example demonstrates, in the GuTS approach, even a complex shape can be created rapidly and easily by using only simple guide shapes. In this drawing only half of the diagram is sketched while the other half is mirrored. Also, most of the circular arc segments are connected such that they maintain C^1 continuity precisely. Since the GuTS approach provides several snapping mechanisms, these aligning operations are done precisely, rapidly and easily.

Figure 6.2 shows an illustrative sketch of a conceptual car model. All the freeform curves - except the arcs and ellipses - are traced from French curves. Each curve is the result of stitching several small pieces of curves together. Yet, it is impossible to locate the stitch regions. Smoothness of the curves is quite evident. Due to the use of several

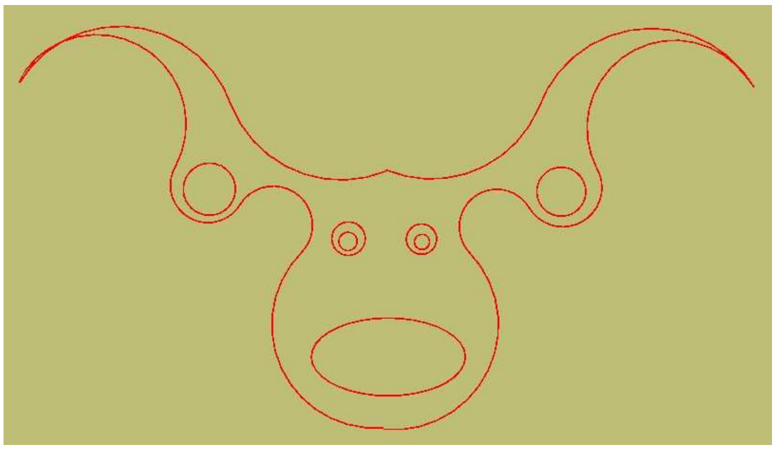


Figure 6.1 Illustrative sketch of a moose head

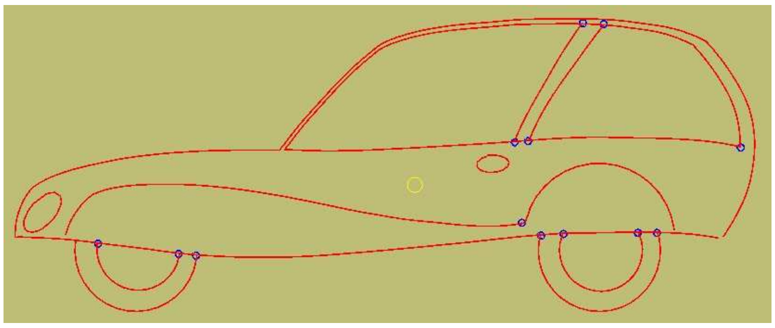


Figure 6.2 Illustrative sketch of a car

automated stitching mechanisms, as discussed in previous chapters, the smooth C^1 connectivity is maintained between multiple pieces. The small circular dots in this picture are the intersection (connection) points between the curves. While the sketch of the moose head (Fig. 6.1) and the illustrative sketch of the car (Fig. 6.2) are concept shapes, the geometries used to create these shapes are precisely traced from the guide shapes (such as circles, ellipses, and French curves), and the traced curves are aligned precisely using several snapping mechanisms.

Figure 6.3 shows another example of a curve geometry. In addition to French curves, several other guide shapes, such as an ellipse, circle, and line segments, were also used to sketch this geometry. When designing ship hulls, instead of French curves, the standard “ship-curves” can be used to design the hulls precisely. The small circular dots in this picture are the intersection (connection) points between several curves.

Figure 6.4 shows an illustrative sketch of a sword. All the freeform curves were drawn using a French curve. Fig. 6.4(a),(b),(c), and (d) show the progress of the diagram. The actual French curve that is used can be seen in these figures. Fig. 6.4(e) shows the completed sketch. A one-point pivotal snap is represented by the circle drawn with double lines (Fig. 6.4(a)) at the point of connection between the guide shape and the regular curve. Similarly, the crossed box in Fig. 6.4 (b) and (c) shows that a tangential condition snap exists between the guide shape and the regular curve. These visual cues dynamically provide direct feedback of the snapping conditions to the user in real-time.

Figure 6.5 is an illustrative sketch of a flower and plant which shows several features of the GuTS approach. First, as shown in Fig. 6.5(a), a circular guide is positioned at different places and copied fully to create new regular curves. Yellow represents the guide curve being copied to form a regular curve. Green lines represent regular curves but are not selected. Figure 6.5(b) shows elliptical guide curves placed at different locations and copied to create new regular curves which form the flower petals. The blue ellipse is the guide curve and the yellow curve is the recently copied curve. Since all these circles and ellipses are created in full, they are copied in a single step instead of tracing all over the geometry. Figure 6.5(c) shows the completed illustrative sketch of the flower and plant diagram in red. Red represents the curves that are selected for further operation. In this case, as shown in Fig. 6.5(c), the selected regular (red) curves are converted to guide curves (shown in blue). Now the whole flower and plant sketch itself is a guide

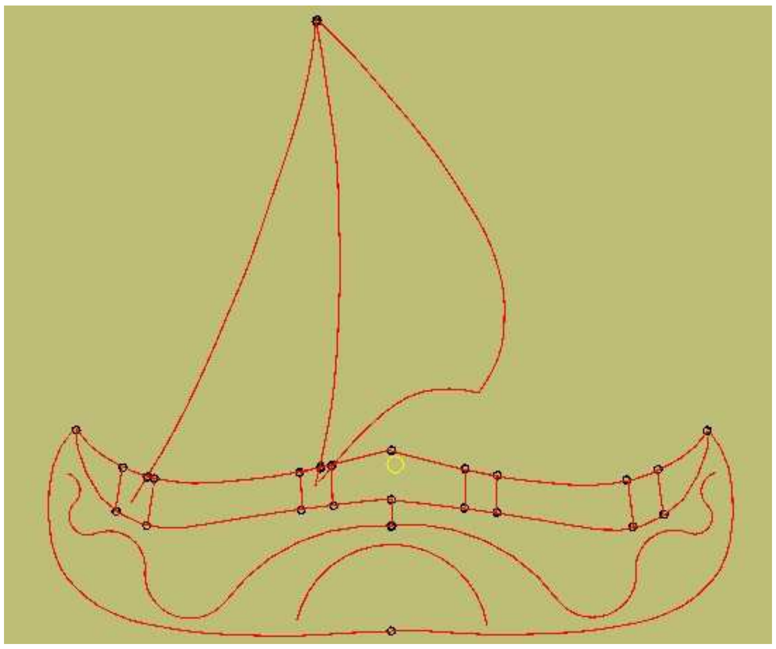
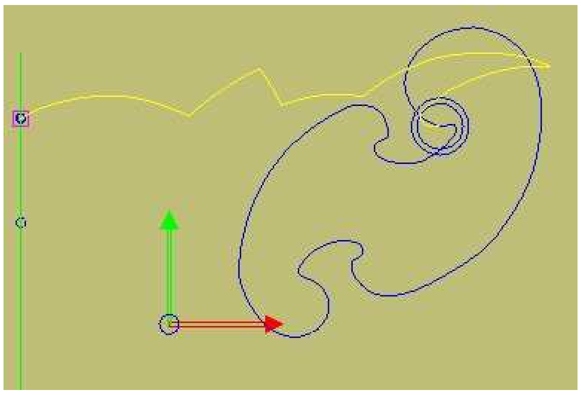
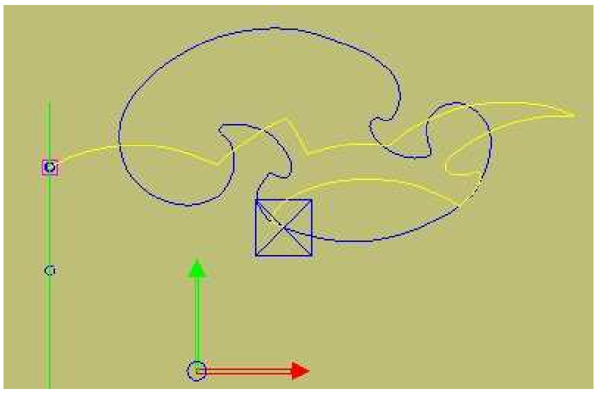


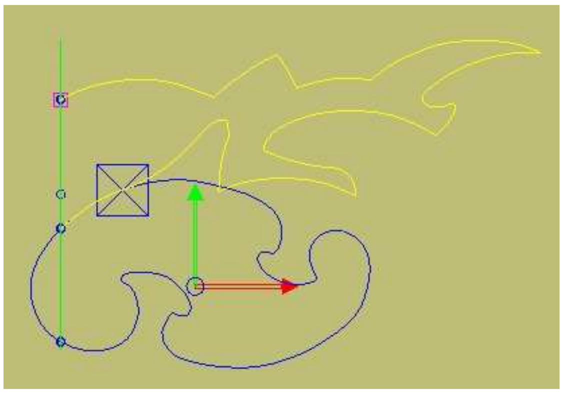
Figure 6.3 Illustrative sketch of a boat



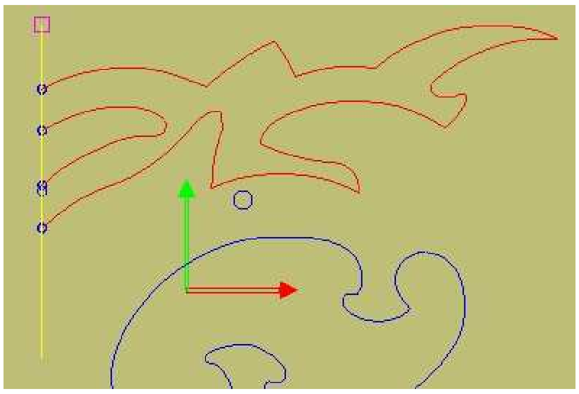
(a) curve geometry is traced from a French curve



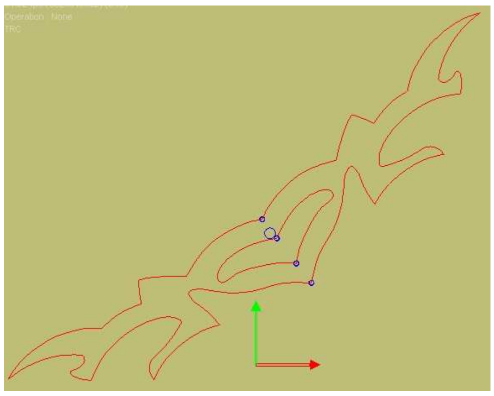
(b) ...tracing further...



(c) ...tracing further...



(d) compled half part of the sword



(e) after mirroring the half, the resulting completed sword

Figure 6.4 Illustrative sketch of a sword

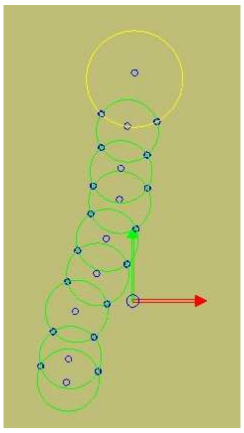
geometry. It can now be used as a guide shape to create additional geometries. Figure 6.5(d) shows this whole set of guide geometries used for tracing. The tracing portion is shown in yellow as it progresses.

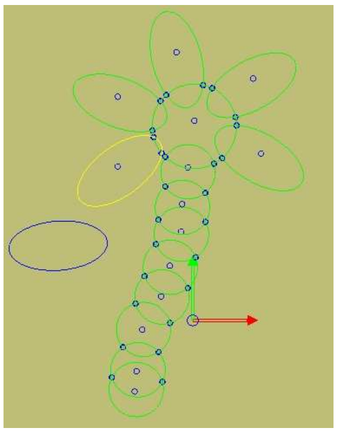
Figure 6.5(e), the sketch to the right, shows the completed sketch that was created by tracing the outer boundary of the guide shapes (shown in blue). Figure 6.5(f) shows a totally different and artistic form of the flower and plant which is drawn using the same guide shapes. These drastically different sketches illustrate the versatility of this approach. By simply using these guide shapes, several varieties of sketches can be easily created from imagined ideas. This example also shows that these arcs and curves are precisely copied, although in pieces, to retain the precision of the geometries during several stages.

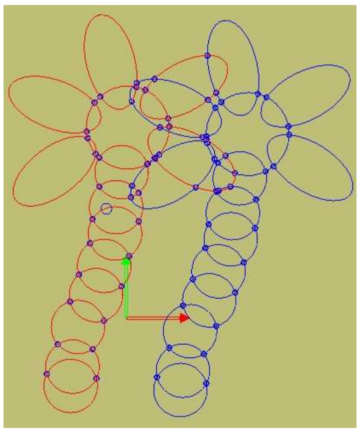
Figure 6.6 shows the diagram of a guitar model. Fig. 6.6(a) shows as it was drawn. A French curve was used to draw the body of the guitar. A straight line tool and other curved tools were used for the neck of the guitar. As mentioned earlier, the blue crossed box represents the establishment of a tangential snap condition between the guide curve and the regular curve. Fig. 6.6(b) shows the completed guitar diagram. The small circular dots show the intersection points between multiple curves. As we will see later in this chapter, this diagram is used as a base to create a 3D guitar model.

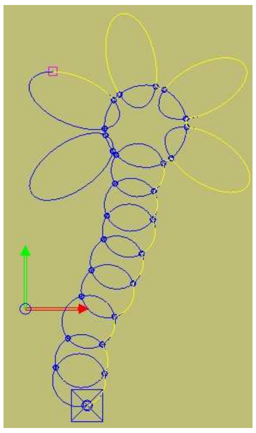
Figure 6.7 shows the diagram of a pen model. Fig. 6.7(a) shows as it was drawn. A French curve was used to draw the curved body parts of the pen model. A straight line tool was used to draw the clip portion. As mentioned earlier, the small circular dots show the intersection points between multiple curves. Fig. 6.7(d) shows the completed pen diagram. We will see later in this chapter how this diagram is used as a base to create a 3D pen model.

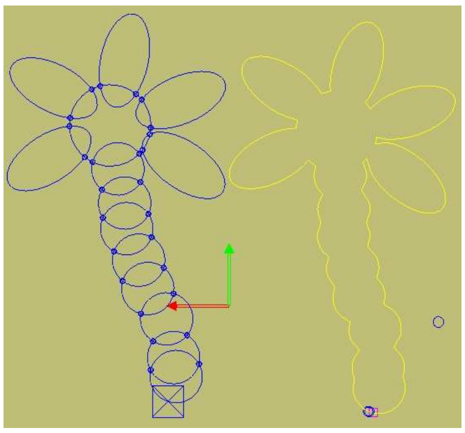
Figure 6.8 shows examples of creating curves geometries in the GuTS system using Bezier representation. Figure 6.8(a) shows two Bezier curves represented as a single curve segment. It also shows the curve segment's corresponding control polygon. Figure 6.8(b) shows after the user interactively traced partly over the two Bezier curves (yellow segment). In this case, the starting point on the first curve and the ending point on the second curve are completely random. The user traces over this Bezier curve segment interactively as if it is a single curve - exactly like Subdivision representation. In these diagrams, the control polygons are made visible to demonstrate tracing is performed over Bezier curve segments. In the GuTS system the user can turn off the visibility of the control polygons, rendering only the resulting Bezier curve. The user will not notice any difference and s/he will perform the tasks exactly the same way regardless of the underlying geometric representation. This example demonstrates that the GuTS approach allows the same user interface to be transparently used regardless of the underlying geometric representation. Figure 6.8(c) shows the newly created Bezier curve segment from the traced region shown in Fig. 6.8(b). The newly created curve contains two new Bezier curves that are represented as a single curve segment with its corresponding new control polygons. Figure 6.8(d) shows another example that contains multiple, in this case four, Bezier curves represented as a single curve segment. As the user traces over the curves randomly (from a random starting point to a random ending point), the resulting new traced curve is shown in Fig. 6.8(e). Note that the newly formed Bezier curves still maintain C^0 continuity from the original guide curve.


(a) multiple circular guides are used to create the stem


(b) elliptical guides are used to create the petals


(c) whole sketch is copied and converted into a complex guide (shown in blue)


(d) tracing along the guide to create new geometry (shown in yellow)


(e) the completed new traced geometry (shown in yellow)

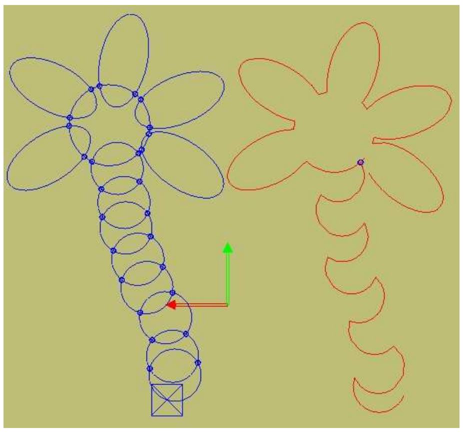
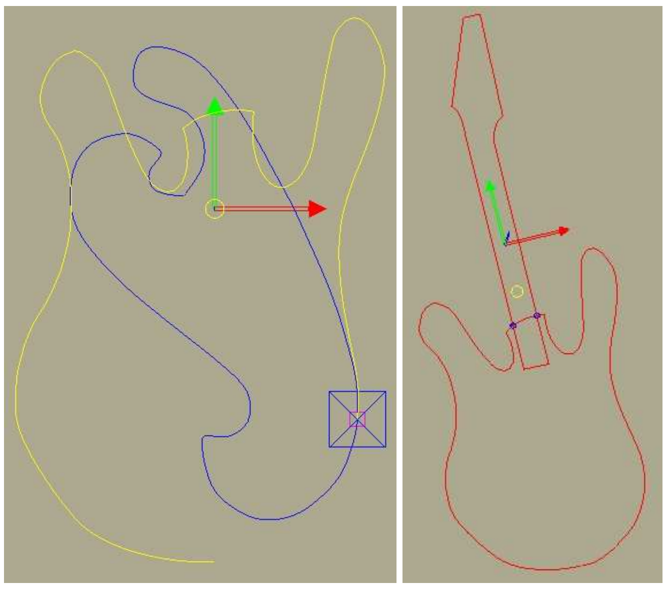
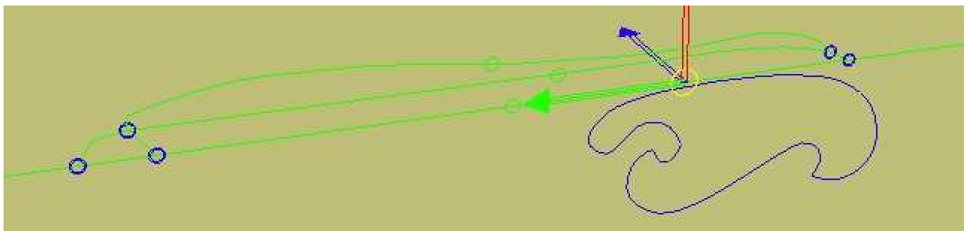

(f) using the same guide, but traced differently produces totally different resulting geometry (shown in red)

Figure 6.5 Illustrative sketch of a flower and plant

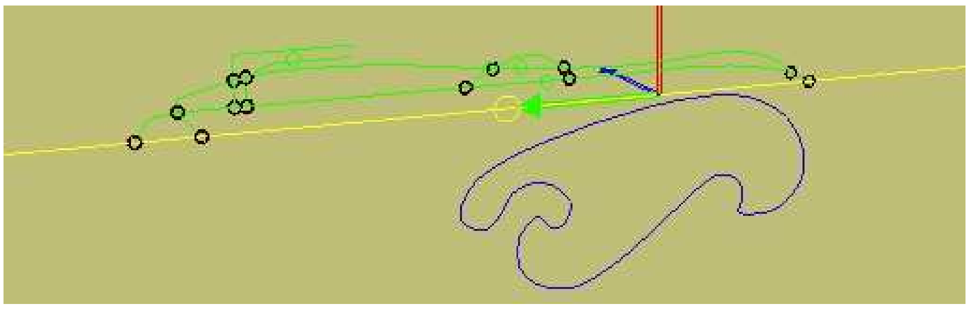


(a) (b)

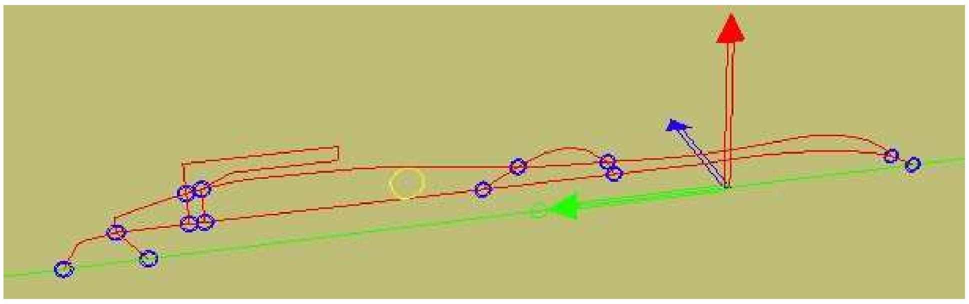
Figure 6.6 Diagram of a guitar



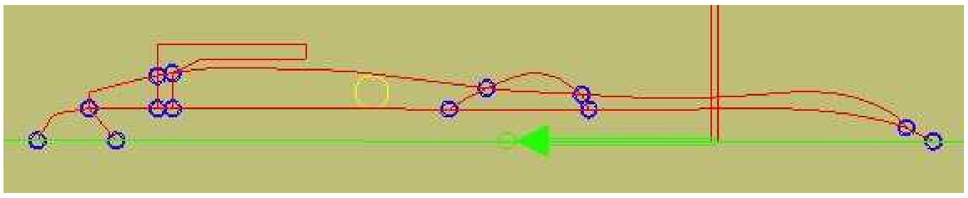
(a) curves forming the body of the pen traced from a French curve



(b) pen handle and other parts of the pen are traced

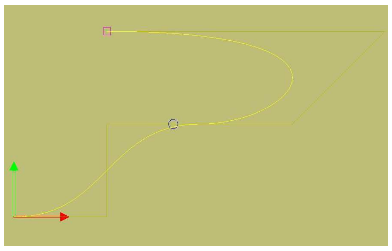


(c) fully traced diagram of the pen aligned along the center-axis line (shown in green) - small blue circles represent the intersection of the curves, and the yellow circle represents the geometric center of the selected curves

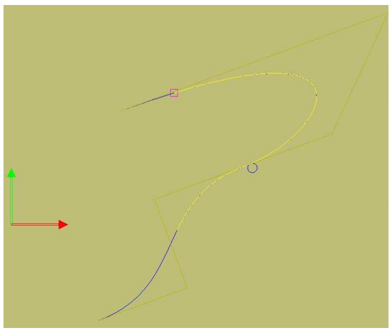


(d) completed diagram of the pen

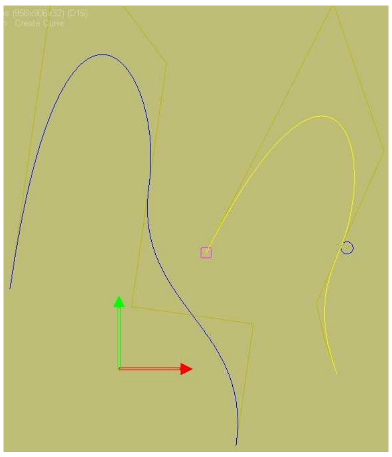
Figure 6.7 Diagram of a pen



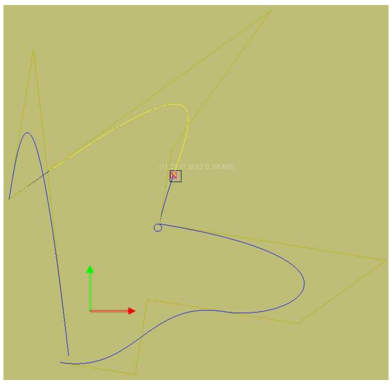
(a) two Bezier curves represented as a single curve segment and its corresponding control polygon



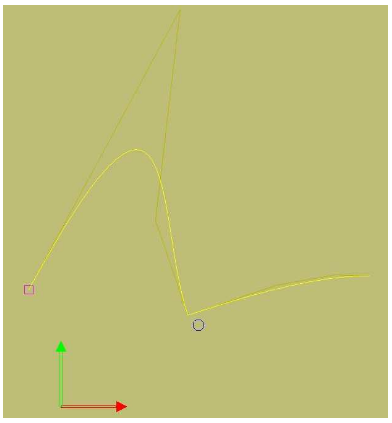
(b) user interactively traced over two Bezier curves with random start and end points as if it is a single curve (shown in yellow)



(c) newly traced curve segment consists of two curves (shown in yellow), and its corresponding new control polygon

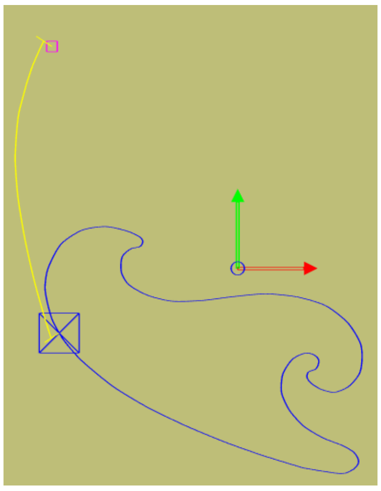


(d) another example where four Bezier curves are represented as a single curve segment and traced randomly

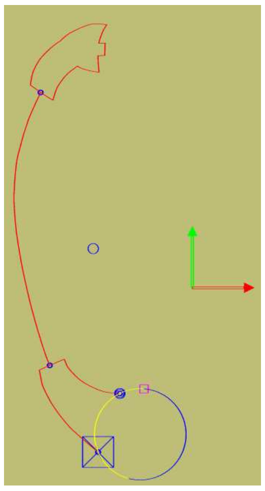


(e) newly traced curve segment consists of multiple curves - note the resulting curve maintain C_0 continuity from the guide curve

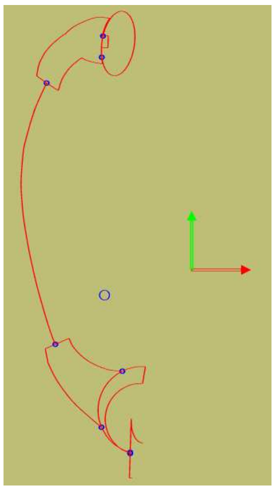
Figure 6.8 Creating curve geometries (using Bezier representation)



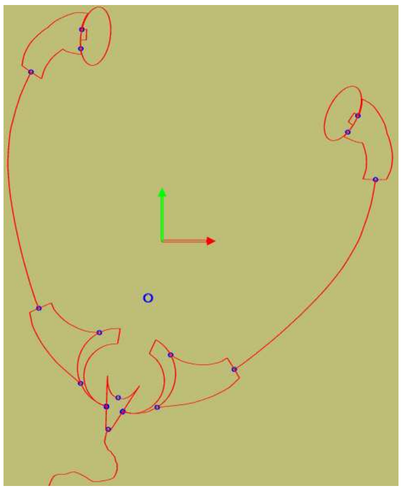
(a) a French curve is being used to trace part of the earphone



(b) base of the earphone is being traced from a circular guide shape



(c) shows half-completed ear-phone sketch



(d) mirrored to complete earphone model and then a wire is added to the base using freeform tracing and a smoothing operation was applied

Figure 6.9 Creating an headset/earphone using various guide shapes

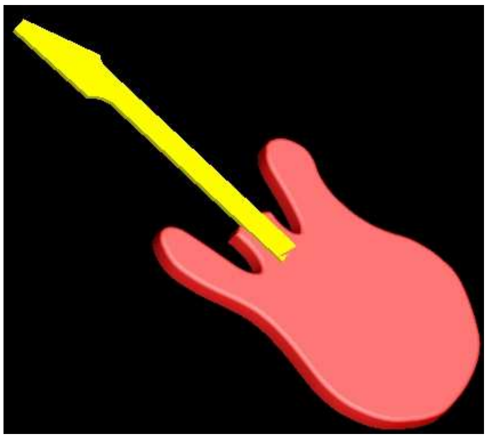


Figure 6.10 Guitar model

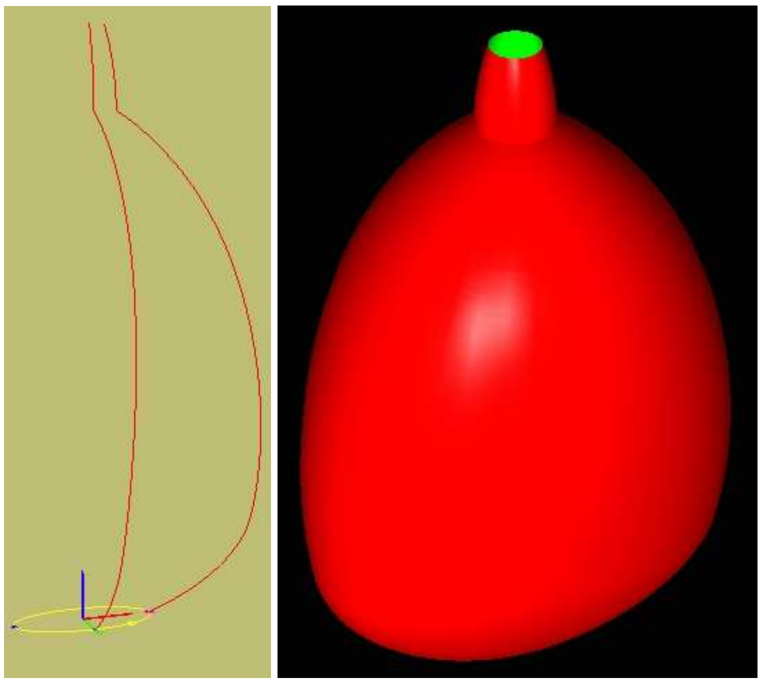
Figure 6.9 shows creating an headset/earphone model using different guide shapes - French curve, circular, elliptical, line, guide shapes, and freeform tracing. Figure 6.9(a) shows a French curve is being used to trace the geometry that connect the earbuds and the base. Figure 6.9(b) shows a circular guide shape is being used to create the base of the earphone. Figure 6.9(c) shows after the earbud and base feature are created using elliptical and line guide curves. Figure 6.9(d) shows the completed earphone model after mirroring and adding the wire to the base using freeform tracing.

6.2 Examples of Surface Geometries

The first sets of surface models in this section show how surface geometries are created using curve diagrams designed in the GuTS system. These examples demonstrate that curves created by the GuTS approach can be further easily used to design surface geometries. These examples also demonstrate that the conventional modeling operations, such as extruding, revolving, lofting, and mirroring operations, work well along with the GuTS operations. Figure 6.10 shows a guitar model which is an extruded object of the guitar diagram shown in Fig. 6.6(b). The top and bottom faces of the guitar are tessellated and rounded along the edges to give the completed model a smooth finish. Though creating models through extrusion is not new, the curves designed in the GuTS approach can be used to create 3D surfaces by using the conventional modeling approaches.

Figure 6.11(a) shows three curves that are drawn using the GuTS system. Fig. 6.11(b) shows the lofted surface of a bottle model, which is created by lofting through the curves shown in Fig. 6.11(a). This example shows another way of using a traditional approach - lofting - to create 3D surface models from the curves designed in the GuTS system.

Figure 6.12(a) shows a pen model that is created using the curve diagram shown in Fig. 6.7. Several curves are used to create the various pieces of surface geometries to form the final shape of this pen model. Conventional modeling



(a) (b)

Figure 6.11 Bottle model

operations, such as extrusion and revolution operations, create these surface geometries. Fig. 6.12(b) shows the pen model with transparent rendering to display the internal components of the model.

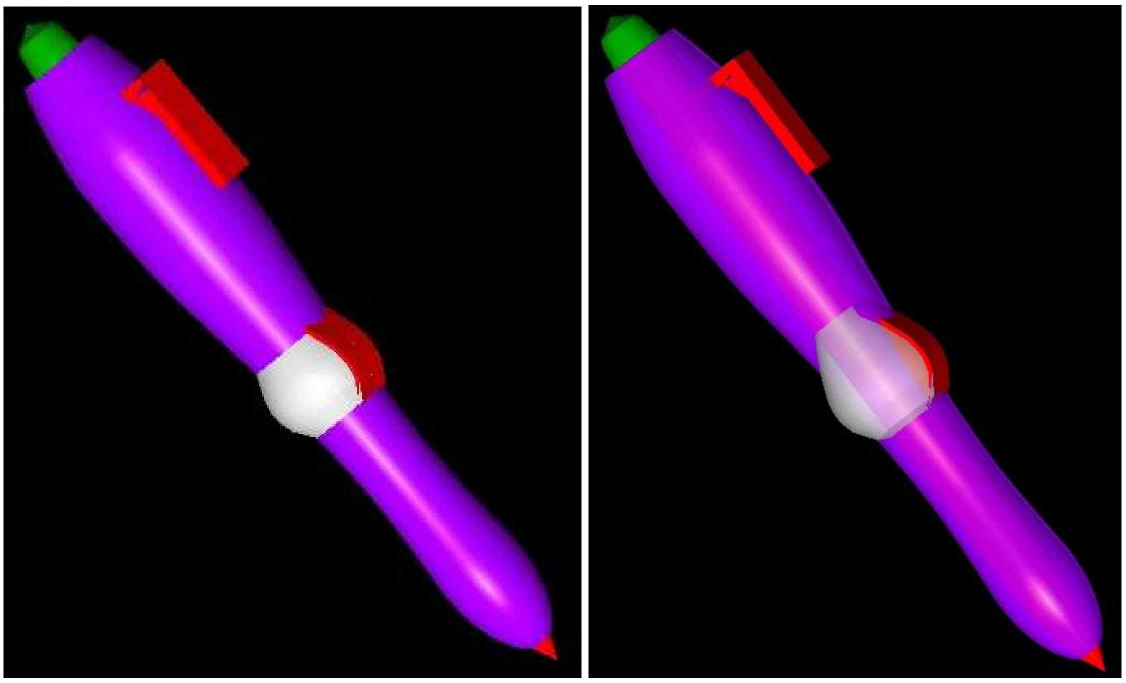
The previous three figures showed how curve geometries created in the GuTS system can further be used to create surface geometries. The following figures show the two types of tracing operations that are performed in the GuTS system: zero-point tracing, and thick-brush tracing. These examples also demonstrate how these two types of tracing are interchangeably used over single, as well as multiple, guide surfaces to create complex geometries easily, directly and rapidly.

The series of figures in Fig. 6.13 illustrates the use of zero-point tracing to trace a freeform leaf model in a single step. Figure 6.13(a) shows the extruded freeform surface (yellow) in wire-frame rendering while the traced curve is shown in red. Figure 6.13(b) shows the same in shaded rendering. Figure 6.13(c) displays the model once the traced curve is inserted into the surface geometry. Once this traced curve is inserted, the actual surface geometry is divided into two portions, inside and outside. Then the inner portion of the leaf is highlighted in red. Figure 6.13(d) shows only the required leaf portion, after the unwanted outside surface geometry is removed (deleted). Figure 6.13(e) is the same surface but in wire-frame rendering to provide the details along the boundary.

The series of figures in Fig. 6.14 is another example of zero-point tracing over an extruded surface. Figure 6.14(a) is the result of the first character being traced and inserted. Figure 6.14(b) displays the two characters already traced, inserted and removed while the third character is just traced. Figure 6.14(c) is the result of the third character being inserted and erased. Figure 6.14(d) shows the surface after all the four characters are cut out. Figure 6.14(e) shows a freeform curve traced from one surface boundary to another surface boundary and inserted to split the surface geometry into two pieces. Then, the highlighted (red) portion of surface is removed to achieve the final surface geometry, as shown in Fig. 6.14(f). The sequence of these images shows that the user can trace directly over the surface, without any knowledge of surface geometries or the underlying geometry representation, and create complex shapes easily and directly.

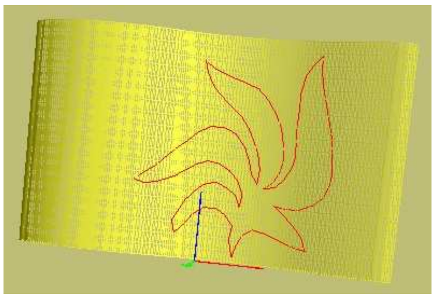
Figure 6.15 shows the sequence of images that describe the creation of a pair of eyeglasses using zero-point tracing. This is another example of how zero-point tracing can be used to design geometries. In Fig. 6.15(a) the rim portion of the eyeglasses is drawn over a freeform surface. For Fig. 6.15(b) the curve is inserted and divided into two regions, highlighting the inner surface region in red. Figures 6.15(c) and (d) show what remains after the unwanted outer surface portion is removed, in wire-frame and shaded rendering respectively. Mirroring this one side of the eye-glass along the middle portion of the frame produces the other side of the eyeglasses. Figures 6.15(e) and (f) show the final eyeglasses after the mirror operation is completed and rendered in wire-frame and shaded mode respectively.

Figure 6.16(a) shows a traced Bezier curve segment (from Fig. 6.8(c)) was extruded to create a 3D surface, which then had a zero-point tracing performed on it (Fig. 6.16(b)). Figure 6.16(c) shows the traced region erased from the surface. In Fig. 6.16(d) another zero-point tracing operation was performed on the surface. After erasing the traced region, the resulting final surface geometry is shown in Fig. 6.16(e).

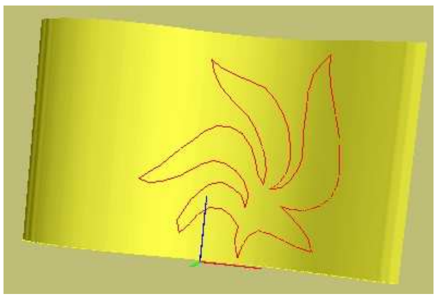


(a) (b)

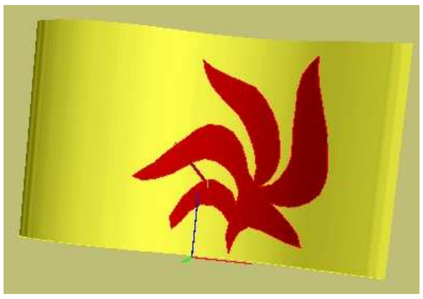
Figure 6.12 Pen model



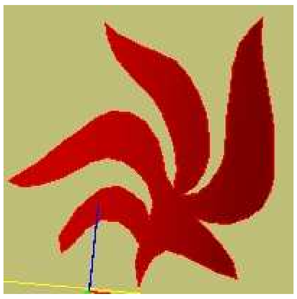
(a) shows zero-point tracing of a leaf sketch on a surface



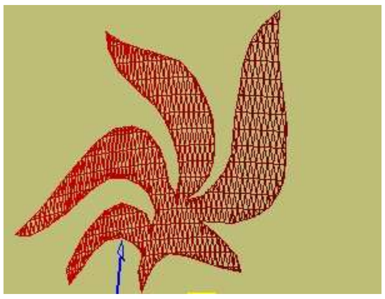
(b) same as in Fig. 6.13(a), rendered in shaded mode



(c) traced curve is inserted into the surface and the corresponding region is selected (shown in red)

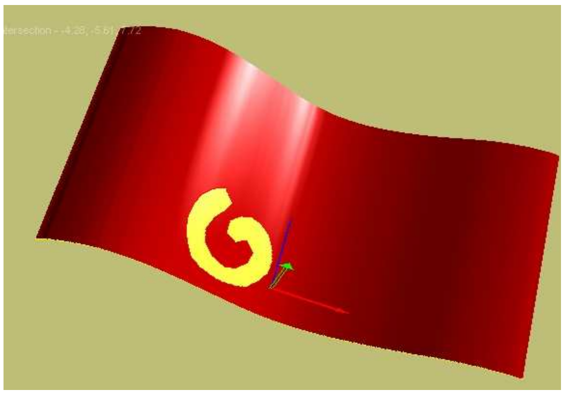


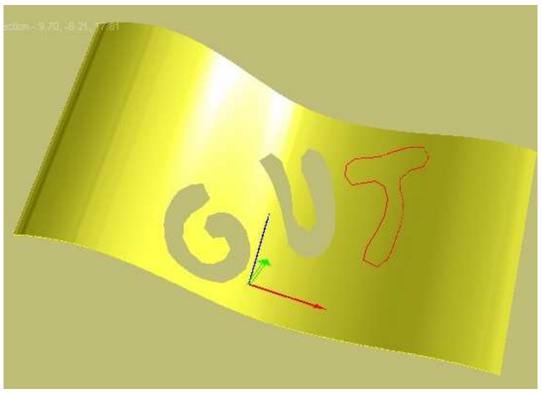
(d) shows the required leaf portion after the unwanted surface geometry is removed

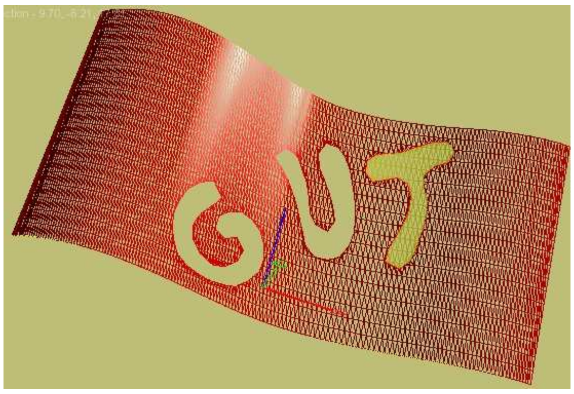


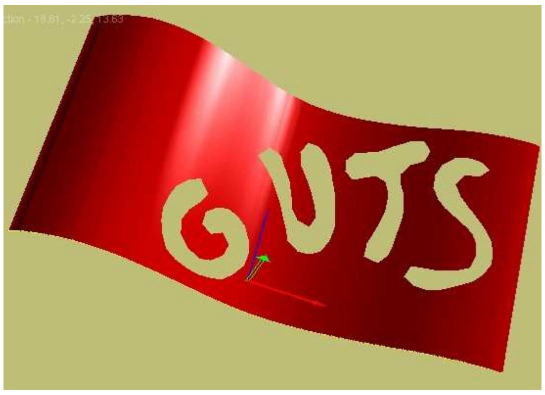
(e) same geometry as in Fig. 6.13(d) but rendered in wire-frame mode to show the details

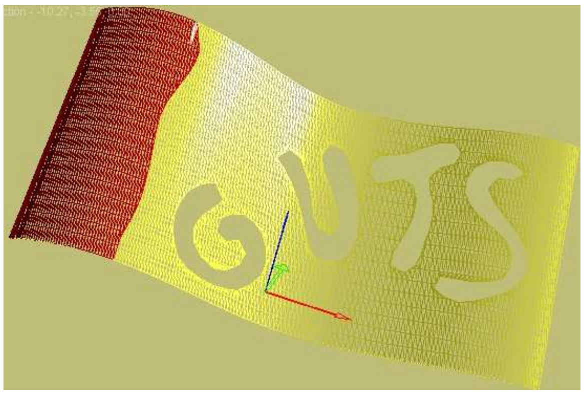
Figure 6.13 Creating a leaf model through zero-point tracing


(a) shows character ‘G’ is traced and inserted into an extruded surface


(b) shows characters ‘G’ and ‘u’ trimmed from the surface, and character ‘T’ is just traced


(c) shows after the character ‘T’ is trimmed from the surface


(d) shows the word ‘GuTS’ trimmed from the surface


(e) shows part of the freeform surface is being trimmed through a zero-point traced curve (shown in red on the left side)

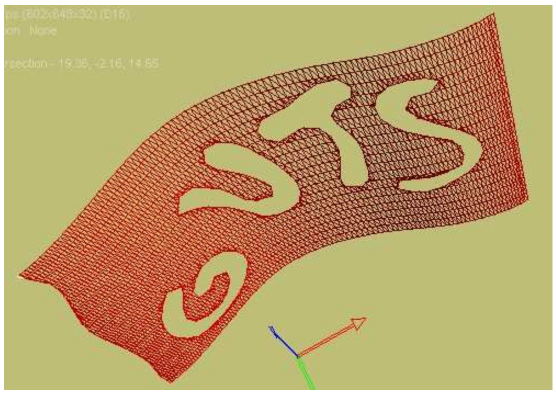
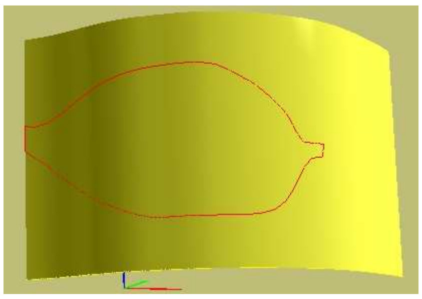
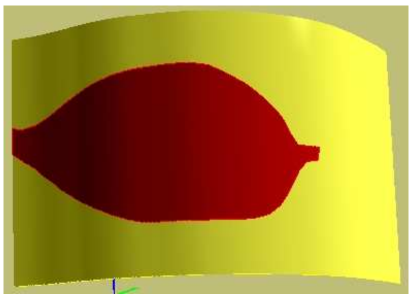

(f) shows the final surface geometry

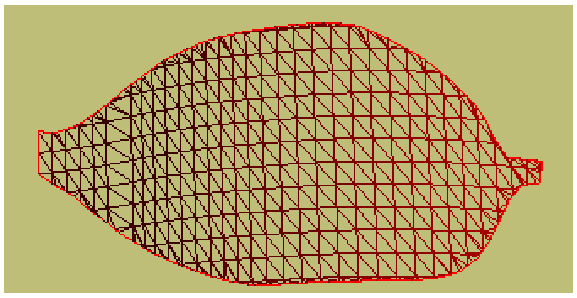
Figure 6.14 Freeform zero-point tracing of characters



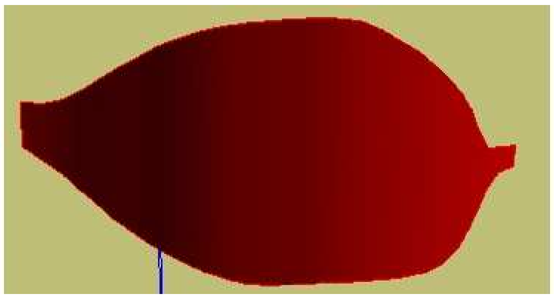
(a) the rim portion of the eye-glass is traced over a surface



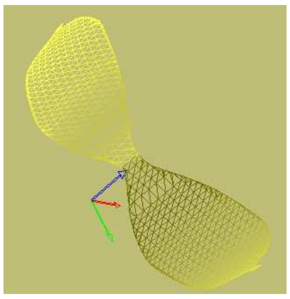
(b) the traced curve is inserted into the surface (shown in red)



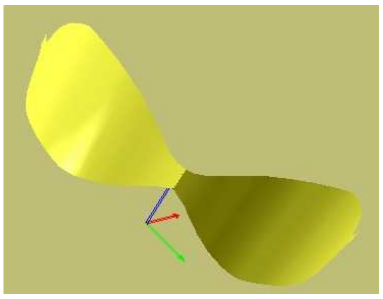
(c) the rim portion after the unwanted outer part is removed



(d) same as Fig. 6.15(c) rendered in shaded mode

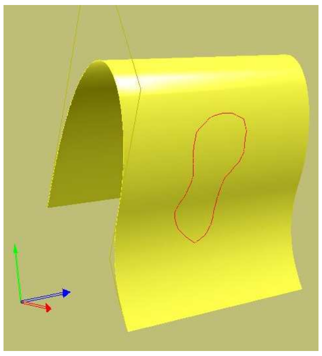


(e) mirror the half portion to form the full eye-glass model

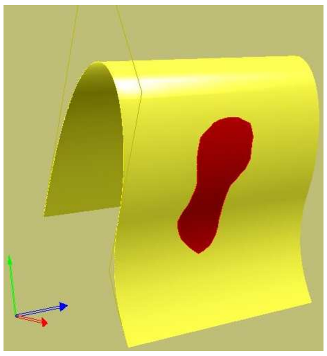


(f) completed eye-glass model rendered in shaded mode

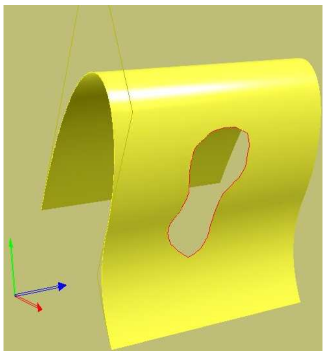
Figure 6.15 Sun-glass model through zero-point tracing



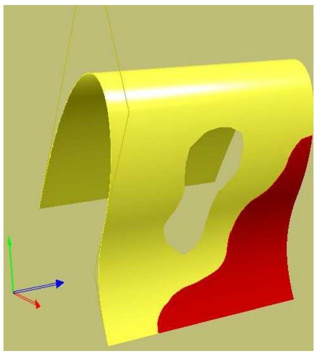
(a) Bezier curve is used to create an extruded surface, and a curve is traced over this surface



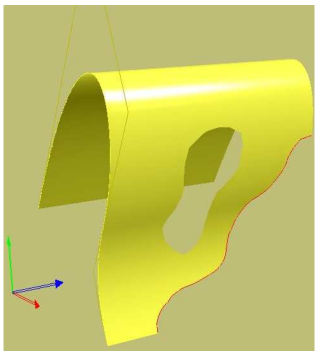
(b) traced curve is inserted into the surface and the traced region is highlighted (shown in red)



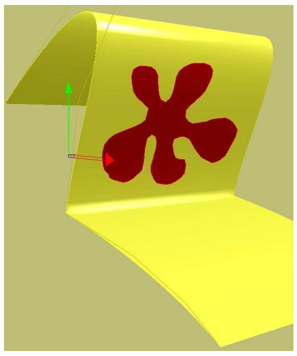
(c) the traced region is erased from the surface



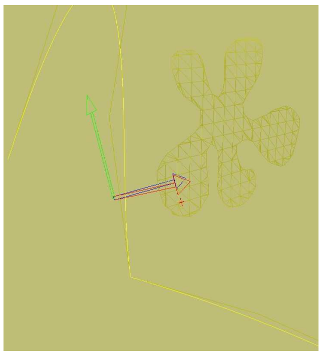
(d) another curve is traced and inserted into the surface and the region is highlighted (shown in red)



(e) the resulting surface after the traced region is erased



(f) shows another example, while the surface is same, but a different curve in the shape of a flower is traced and inserted into the surface



(g) resulting flower shaped surface after the unwanted surface region is removed

Figure 6.16 Creating surface geometries (from Bezier curves)

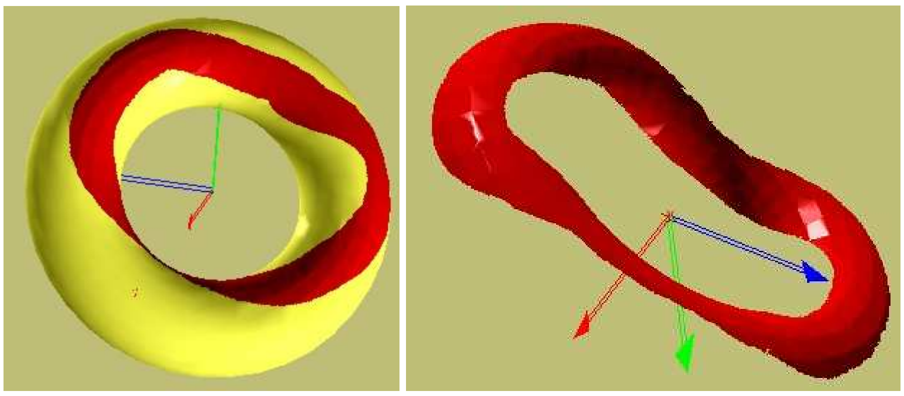

(a) (b)

Figure 6.17 Tracing through thick-brush

Figure 6.16(f) shows another example of a surface extruded from a Bezier curve segment (from Fig. 6.8(e)). A flower-shaped zero-point trace was performed on this 3D surface and the traced region is shown in red. Figure 6.16(g) is the final surface in wireframe rendering after the traced region is trimmed from the extruded surface. The above two examples demonstrate that 3D surfaces are created from Bezier curve segments exactly the same way as if interpolation subdivision curves were used. Regardless of the curve representation, the resulting surface geometries and surface operations remain the same in GuTS approach.

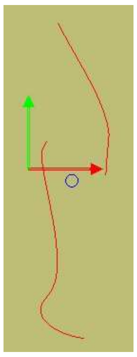
So far, the examples showed different models created by using the zero-point tracing operation. Figure 6.17 shows a thick-brush tracing operation where a thick brush with the required thickness is used to trace over a surface. As the user traces, a region of the surface is traced, precisely selecting the boundary at every cycle of the drawing process. In Fig. 6.17(a) a thick-brush was traced over a simple torus surface and the selected portion of the surface is shown in red. Figure 6.17(b) shows the unwanted outer portion of the surface was removed. While the geometry torus itself is simple and precise, the haphazard tracing operation over this simple object creates a more complex surface. This complex piece of surface is easily traced from simple torus geometry, highlighting the advantages of the GuTS modeling approach. If this same model was modeled using conventional modeling systems, the user would have to draw multiple complex curves that form the boundary UV curves of the final geometry and then loft these curves to get the final geometry. On the other hand, in the GuTS approach, all these operations are performed in a simple single tracing operation directly over the surface.

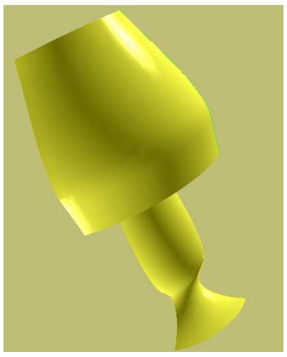
Figure 6.18 shows the sequence of images in designing a lamp model. Figure 6.18(a) displays the initial curves that were used to revolve and create the surfaces that are shown in Fig. 6.18(b). Only one quadrant (i.e. 90 degrees) of the lamp surface was revolved and the detailed editing was done on this part. At a later stage, by mirroring this quadrant, the full symmetric model was created. Figure 6.18(c) shows the traced portion, highlighted in red. In this case both zero-point and thick-brush tracing were used. Figure 6.18(d) shows the same but in shaded mode. Just by looking at both pictures, it is not easy to identify which portion of the region is traced using which tracing option. Figure 6.18(e) shows the surface after the unwanted selected region was removed. Figures 6.18(f) and (g) are the results of the mirroring operation and the erection of the final lamp model. They are rendered in shaded and wire-frame mode respectively to highlight the details.

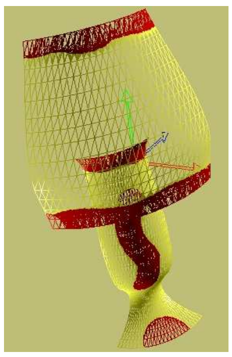
Figures 6.18(h), (i), and (j) are the same quadrant of the lamp model, shown in Fig. 6.18(b) but used to create a different design of the final geometry. This emphasizes that by using the same initial surface guide geometry, several different detailed models can be created just through the use of different tracing operations. This provides the user flexibility to design multiple geometries from the initial base geometry in a short period of time.

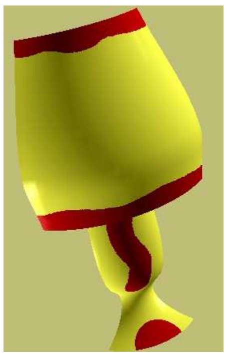
The above presented surface examples use only tracing operations over disjointed surfaces. Figure 6.19 shows a new, interesting, useful and time-saving tracing operation in the GuTS approach. The sequence of pictures in Fig. 6.19 shows the ability to trace, both zero-point and thick-brush tracing, over multiple guide shapes, as if tracing over a single surface. In Fig. 6.19(a) three geometries - two hemispherical and one semi-cylindrical surfaces - are intersecting each other. The dynamic intersection algorithm finds the intersecting triangles and intersecting curves between these geometries in real-time. Once these multiple geometries are made into guide geometries for tracing, these intersecting curves are inserted into the geometries. Figure 6.19(a) shows the geometries in wire-frame rendering with two regions traced and selected while the third region is just traced. Figure 6.19(b) displays after the third region on the right and the fourth region on the left (not visible in picture), and a thick-brush tracing in the middle is inserted over multiple geometries. The selected regions of the surfaces are shown in red. After the unwanted selected surface regions and other side of the hemispherical surfaces are removed, the final geometry is shown in Fig. 6.19(c). Such a complex final surface geometry is easily and directly modeled using the GuTS approach but it cannot be modeled with the same ease using other geometric modeling programs.

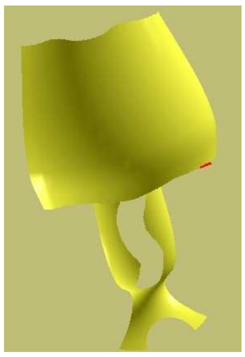
Figure 6.20 shows the sequence of pictures during the creation of a cooking spatula. In this example, one extruded and one revolved surface were used. Both these surfaces were made into guide geometries. Then, zero-point tracing was used to trace the desired shape of the spatula over both geometries. Figure 6.20(b) shows the stage where one region of the surface was traced but before the second traced curve is inserted. Figures 6.20(c) and (d) are the result of both surface regions being traced and inserted. They are rendered in shaded and wire-frame mode to highlight the details of the geometry. Figure 6.20(e) shows the final geometry after the unwanted surface regions were removed and mirrored to create the other half. Figure 6.20(f) shows the completed final geometry in wire-frame rendering mode. This example also demonstrates the capability that multiple guide shapes can be positioned and viewed by the user before the surface regions are traced. This allows the user to visualize partly how the final shape will look before the surface regions are traced - providing the user with additional visual feedback during different stages of the modeling process.

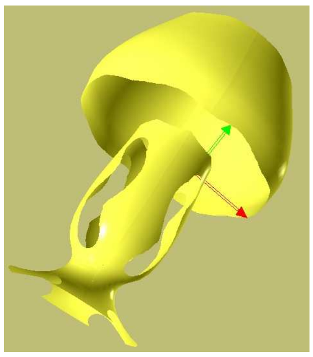

(a) the initial input curves

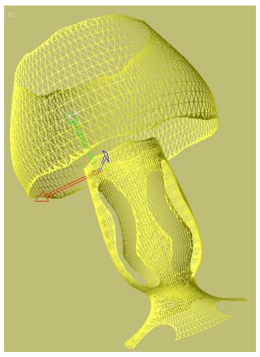

(b) curves are revolved to form one quadrant of a lamp model

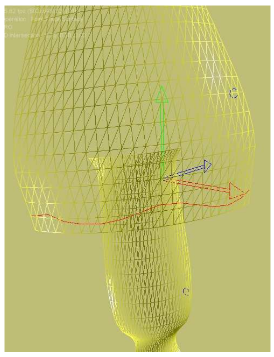

(c) different parts of the surfaces are traced using both zero-point and thick-brush tracing (showin in red)

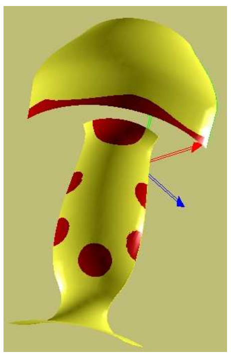

(d) same as in Fig. 6.18(c), rendered in shaded mode


(e) resulting surface after the traced regions are erased


(f) mirrored the quadrant surface to form the complete lamp model


(g) same as Fig. 6.18(f), rendered in wire-frame


(h) starting with the same initial model as in Fig. 6.18(b)


(i) uses different zero-point and thick-brush tracing operations

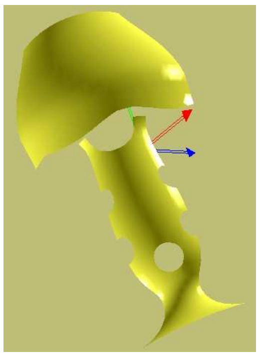
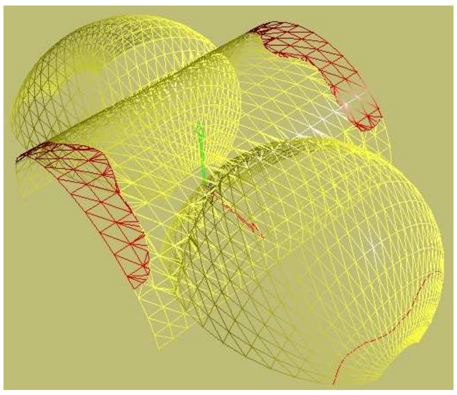
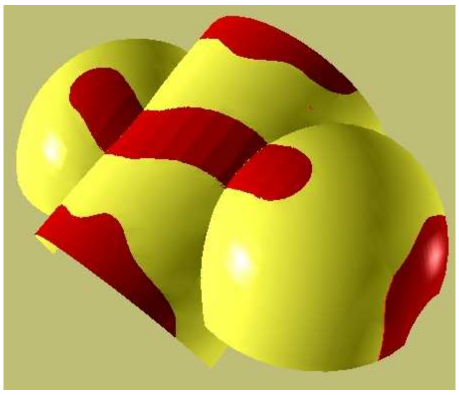

(j) shows the resulting totally different lamp model

Figure 6.18 Lamp models


(a) shows multiple geometries and two regions are traced and inserted into the semi-cylindrical surface


(b) shows additional two regions traced on the hemi-spherical surfaces and a thick-brush traced over all three surfaces in the middle (shown in red)

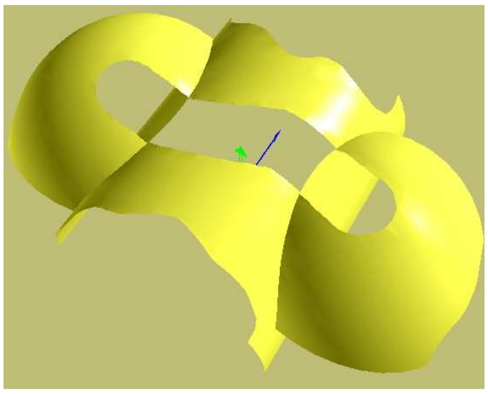
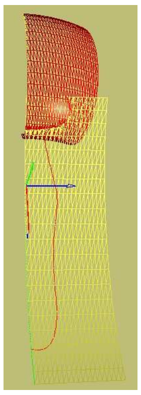
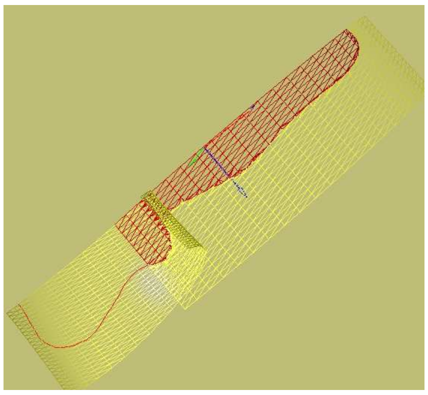

(c) shows the resulting complex abstract surface after all the traced regions are erased

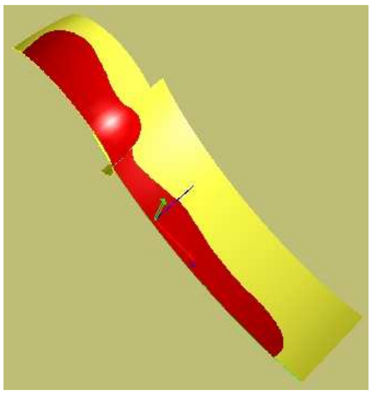
Figure 6.19 Abstract model - Designed through multiple guide shapes



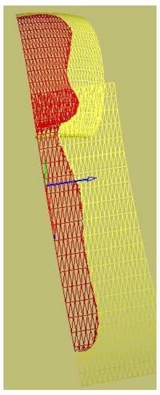
(a) handle is traced over one surface



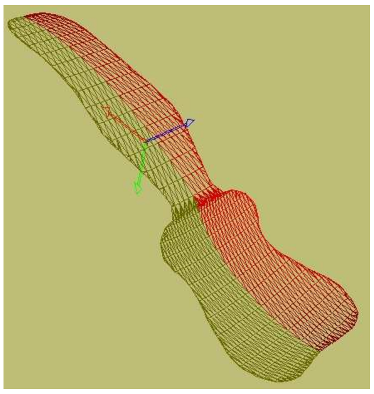
(b) handle is inserted into a surface and the bottom part of the spatula is drawn over the second surface



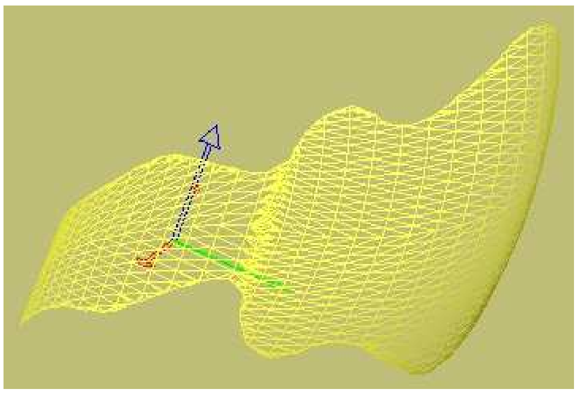
(c) after both handle and bottom part of the spatula inserted into both the surfaces (shown in red)



(d) same as Fig. 6.20(c) in wire-frame rendering



(e) Unwanted surface regions are erased and mirrored to form the complete spatula model



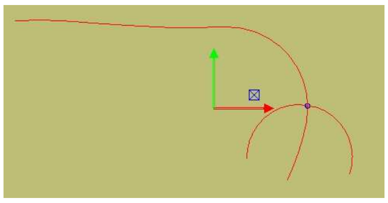
(f) completed spatula model rendered in wire-frame mode

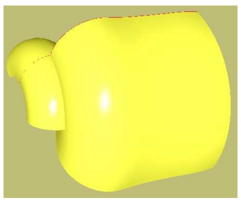
Figure 6.20 Spatula (cooking utensil) model

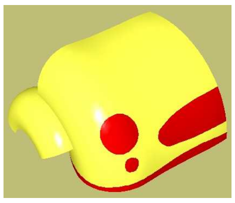
Figure 6.21 shows the sequence of pictures in creating a car bumper model. Figure 6.21(a) shows the curves that created the surfaces shown in Fig. 6.21(b). Since this is a symmetric object, only half of the final geometry was designed and later the other half was mirrored. Figure 6.21(c) shows the portions of the surfaces were traced. As mentioned earlier, it is difficult to identify which type of tracing operation was performed on these surfaces. Figure 6.21(d) shows the remains after the unwanted portions of the surfaces were removed. Figures 6.21(e) and (f) show the model after the other half was mirrored, in wire-frame and shaded rendering respectively. Figure 6.21(h) shows the complete geometry after the top portion was extruded. Figures 6.21(g) and (i) are the same as the models shown in Fig. 6.21(f) and (h), but only in red.

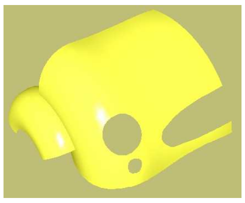
Figure 6.22 shows the sequence of pictures in creating a car wheel. Figure 6.22(a) shows the curve used to create the surfaces shown in Fig. 6.22(b). Figure 6.22(b) shows the traced region in red. Since this is a symmetric object, only one quadrant of the final geometry was designed and later the remaining portion was mirrored. Figure 6.22(c) shows the model after the traced portions were removed. Figure 6.22(d) was the result of one mirror operation while Fig. 6.22(e) shows all four quadrants were mirrored. Figures 6.22(f) and (g) display the completed wheel model in wire-frame and shaded rendering respectively.

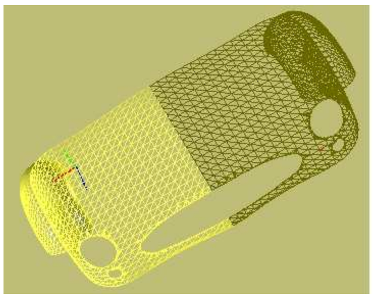
The GuTS system also edits models that are designed in other existing (commercial or non-commercial) modeling systems through a set of standard file formats. Figure 6.23(a) shows the head model that was imported into the GuTS system as a standard mesh file format called ‘object file format (off)’. In Figure 6.23(b) the ears were removed and a new portion of the head was traced and selected. Figures 6.23(c) and (d) show the final face mask after the selected surface region was removed and rendered in shaded and wire-frame rendering mode.

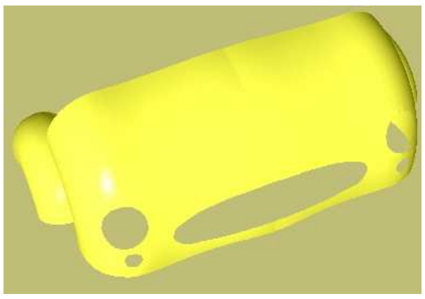

(a) initial input curves

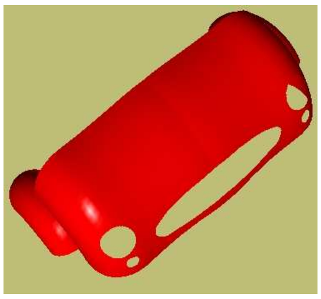

(b) curves are revolved and extruded to form half of the bumper model

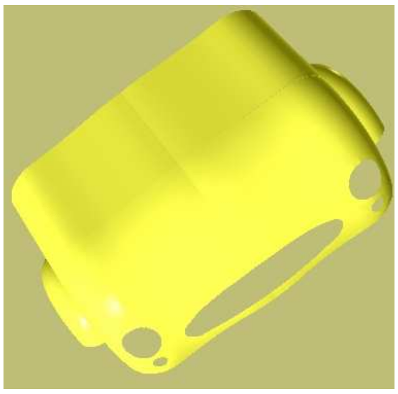

(c) additional features are traced over the surface (shown in red)


(d) traced regions are erased from the surface geometries


(e) surfaces are mirrored to create other half of the bumper model


(f) resulting bumper model rendered in shaded mode


(g) same as in Fig. 6.21(f) shown in red at a slightly different angle


(h) mid top portion of the bumper is extended to form the complete bumber model

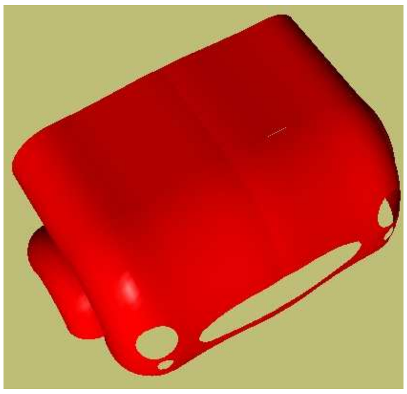
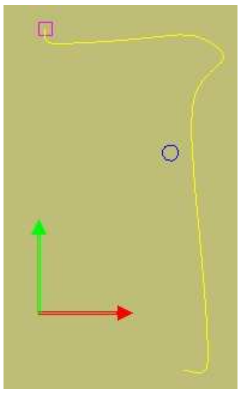
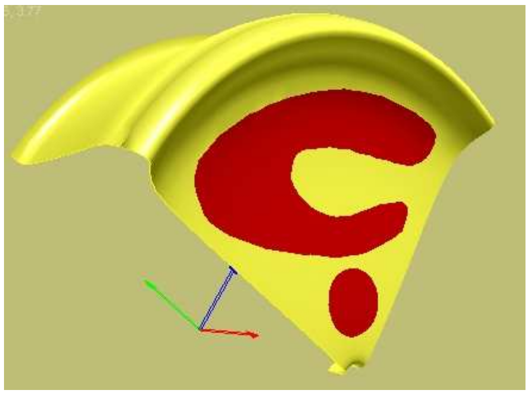
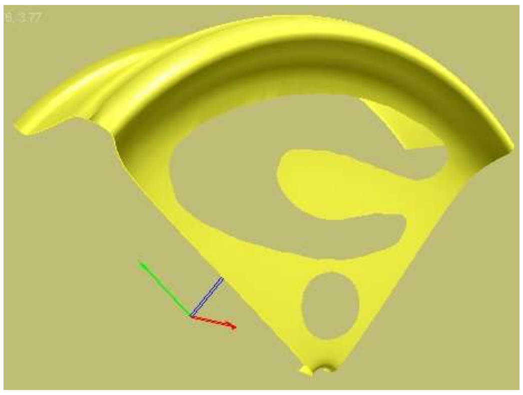
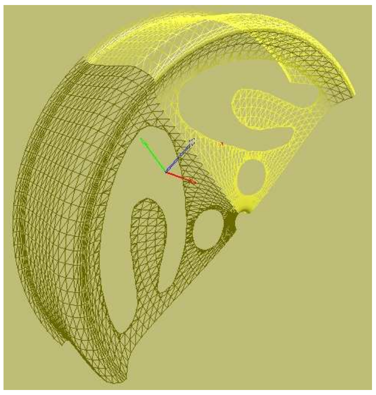

(i) same as in Fig. 6.21(h) rendered in red at a slightly different angle

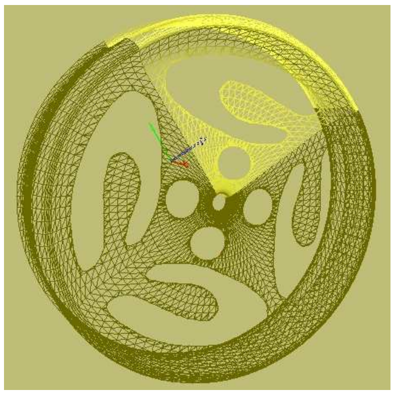
Figure 6.21 Car bumper model

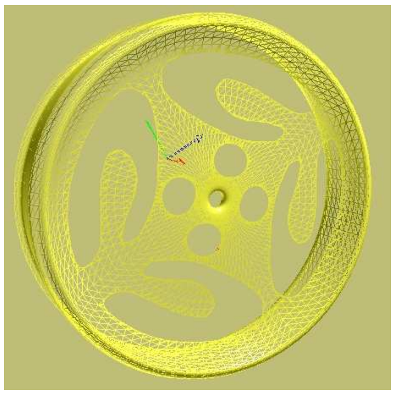

(a) initial input curve


(b) input curves are revolved one quandrant and different regions are traced using zero-point and thick-brush tracing (shown in red)


(c) traced regions are erased from the surface


(d) mirrored once to form half of the wheel


(e) mirrored again to form the full wheel model


(f) completed full wheel model in wire-frame rendering

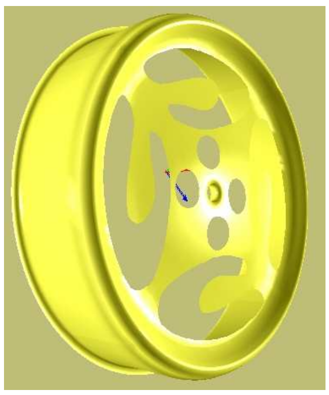
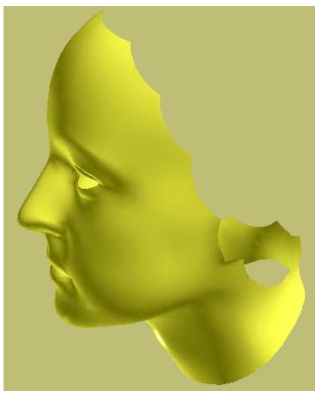

(g) completed full wheel model rendered in shaded mode

Figure 6.22 Car wheel model

(a) imported surface model that was modeled in another modeling system

(b) thick-brush tracing was done to select a region of the model (shown in red)


(c) traced region is erased from the surface geometry

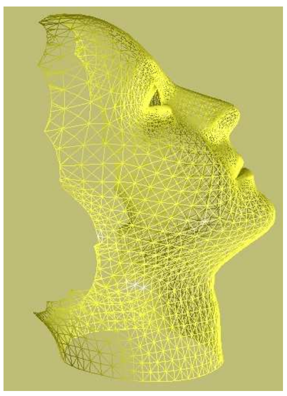

(d) resulting final geometry rendered in wire-frame mode

Figure 6.23 Face mask model

Chapter 7

Conclusions

In this chapter, we summarize the GuTS modeling approach, the GuTS system and the contributions of this research. This will be followed by a discussion of the general limitations of the GuTS approach and our implementation as well as opportunities for future research.

Freeform modeling is a core part of geometric modeling, but freeform models are difficult to specify and manipulate interactively. Hence, working with freeform curves and surfaces precisely is challenging. To manipulate complex freeform geometries precisely with an interactive system, the user must communicate to the system how the vertices, edges, faces, curves, surfaces, and the entire geometries are manipulated in real-time dynamically. Modeling complex curves and surfaces using an interactive system is difficult, since it requires the user to have knowledge of geometric representation and the system's user interface. This is due in part to the limitations posed by conventional modeling approaches and partly due to the limited set of input and output (I/O) devices (such as the keyboard, 2D mouse and monitor).

In this research, we addressed these issues in order to overcome the above difficulties in designing complex curves and surfaces. We presented a novel approach called "Guided Trace and Stitch" (GuTS) modeling using multimodal interaction. The GuTS approach addresses an important issue in designing complex curves and surfaces: to provide precision, fluency, and rapidity at the same time. In the GuTS approach, complex pieces of curves and surfaces are created by guided tracing and stitching together pieces of geometries traced from guide shapes. The GuTS interface provides the fluency of a pen-based sketching interface, as in paper-pencil drawing, and at the same time provides precision through guiding, automated snapping, and stitching mechanisms. Productivity is also achieved when communication and interaction between the system and the user is direct, fluent and rapid.

Precise segments of geometries are created by tracing along guide shapes. Automated snapping and other interaction mechanisms for precisely placing the curves and surfaces permit precise and rapid manipulation of geometries. Precision is achieved through the use of guide geometries and automated snapping mechanisms.

Multimodal user interaction (such as two handed input, 3-dimensional (3D) mouse input, pen and data-tablet for hand-eye coordination, voice input, and synthesized speech output) addresses the user interface challenges raised in the GuTS approach. Using these multiple modes, the user interface is designed effectively to provide the user with a fluent and direct approach of interaction.

Rapidity is achieved through snapping mechanisms and the multimodal user interface. The automated snapping mechanisms define precise conditions with minimal input from the user, eliminating additional input and time. Using multiple input and output modes simultaneously and effective coordination provides rapid interaction.

7.1 Review of Contributions

The primary contributions of this research are:

7.1.1 Novel Approach

This research introduces a novel approach called “Guided Trace and Stitch” (GuTS) modeling using multimodal interaction to model freeform geometries precisely, fluently and rapidly. While reviewing several related modeling systems and research, we found no other system exists that does similar tasks (for curve and surface modeling) with a multimodal user interface. In the GuTS approach geometries are created using three operations: guided tracing, automated snapping, and stitching. Precision is achieved through guided tracing operations (for both curves and surfaces) and automated snapping mechanisms (for curves). Since the tracing operation is direct and fluent, it eliminates the need for the user to be knowledgeable of the underlying geometric representation. The GuTS approach also creates complex geometries by tracing over multiple guide geometries. Secondary contributions of the GuTS approach include: (a) new snapping mechanisms, such as pivot snapping, slide snapping, and two-point snapping, and (b) automated stitching mechanisms for tracing over curves and surfaces that allow pieces of curves and surfaces to be traced and stitched together to form complex shapes transparently.

In curve modeling, the curves are rapidly aligned and positioned precisely using several snapping mechanisms. Some of the snapping mechanisms, such as absolute positioning, point-to-point snapping and the gravity field, are introduced earlier [93, 25, 24, 23]. In this research we introduced ‘whole object snapping’, that includes different snapping mechanisms, such as one-point snap/pivotal snap, slide snap, and two-point snap/fixed snap, which help the user interactively position the curves precisely and rapidly during the modeling process. One-point/pivotal snap creates a (pivotal) point contact and the curves orient themselves about this point. Slide snap creates a point contact and then further slides the curves along another curve. Two-point snap/fixed snap connects a curve between two points of another curve or two points of two different curves. Based on the relative positioning of the geometries, these snap mechanisms are automatically activated and allow the user to rapidly define precise relationships between the geometries.

The automated stitching and continuity detection feature - based upon the alignment and orientation of the geometries - automatically stitches the curves together during the tracing operation. The continuity between multiple guide curves is automatically detected and passed to the traced geometry. This automated continuity detection achieves different C^0, C^1, and C^2 continuities automatically, depending upon the properties of the guide curves and snapping conditions. This feature eliminates the user’s need to explicitly specify the continuities at every stitching region and allows the user to trace over multiple guides rapidly in order to form complex geometries. In the GuTS approach, two types of tracing methods are used. They are zero-point tracing and thick-brush tracing over a triangulated mesh structure. Zero-point tracing is used to insert a curve that is drawn on the triangulated mesh structure and splits the surface into regions. Thick-brush tracing selects a region of surface by tracing the path over a surface with a user-specified brush-thickness.

7.1.2 Multimodal User Interaction

The GuTS modeling approach created unique challenges for user interface. To address these issues we introduced a multimodal user interaction that not only provided a direct approach of manipulation and interaction but also enabled rapid design through multiple parallel uninterrupted I/O modes. This multimodal user interaction brings out the potential of the GuTS approach by expanding and exploring the way a user manipulates curves and surfaces. Compared with other modeling systems that use multimodal interface for creating approximate abstract geometries, the GuTS approach focused on achieving precision, fluency, and rapidity at the same time. In this research we also demonstrated how multimodal user interaction was uniquely integrated with the GuTS modeling approach.

7.1.3 Implementation

As part of this research, we designed and developed an interactive prototype system, called the GuTS system, that demonstrates the GuTS approach with multimodal user interaction. The GuTS system highlighted different features of the GuTS approach. We also presented several examples of curve and surface models that are created using the GuTS system. These examples from the prototype system also proved the feasibility of this novel modeling approach in real-world applications. We implemented the GuTS system using multiple geometric representations, i.e. interpolation subdivision and Bezier representations for curves, and tessellated mesh representation for surface geometries. By using multiple geometric representations in the GuTS system, we demonstrated that the GuTS approach can be applied to different geometric representations; the underlying geometric representation is transparent to the user; and the same multimodal user interface can be used regardless of the underlying geometric representation.

7.2 Pros & Cons of the GuTS Approach

In this section, we highlight the key pros and cons of the GuTS approach.

7.2.1 Pros

The pros of the GuTS approach are:

- Precision, fluency, and rapidity is achieved at the same time in freeform geometric modeling.
- Suitable for modeling complex curves and surfaces that can be created from one or multiple geometries.

- Helps the user to visualize the resulting geometry before creating them - i.e. by placing the guide shapes in place, the user will know exactly how the resulting geometry will look, even before any tracing operation is done.

- WYSIWYG (What You See is What You Get) modeling approach.

- Does not require the user to be knowledgeable of the underlying geometric representation.

- Independent of the underlying geometric representation - providing an unified front-end user interface to the user. This approach can be implemented using different geometric representations.

- Makes use of the user's natural multimodal interaction behavior - providing direct coordinated multimodal user interface.

7.2.2 Cons

The cons of the GuTS approach are:

- Not efficient when modeling certain types of geometries - such as, when most of the modeled geometries involve lines, polylines and planar faces.

- Not suitable for modeling geometries that cannot be represented as guide shapes.

- Large number of geometries in the modeling session will interfere with the automated snapping mechanisms. This can be addressed using different methods, such as layering concepts discussed in section 7.3.1.

- Since the geometric tracing and stitching are partly continuous processes, implementing editing features (such as maintaining history of modifications, undo feature, etc.) would become difficult.

- While the GuTS approach can be implemented using a conventional keyboard, 2D mouse and monitor interface, it will make the user interface indirect and complex, making it difficult to perform the required operations.

- For interpolation subdivision curves and surfaces, precision is limited by the sampling of the input guide geometries. This is mainly a limitation of the implementation, rather than the GuTS approach itself.

7.3 Limitations and Future Work

In this section, we discuss the limitations in detail, and discuss its potential solutions and future work.

7.3.1 Large Number of Curves

One of the insights from the observation of the implemented curve modeling program is an issue with the snapping mechanisms. When a large number of curves exist in a session, guide shapes often stick to unwanted curves that disrupt the modeling process. This created unwanted snaps and forced the user to pull away these snaps until the desired snap point or curve was reached. This can be avoided by using some of the following approaches. As mentioned in the earlier chapter, 'skitters and jacks' can be used, i.e. by explicitly specifying the points that are for snapping, or by making only a few active curves so that only these curves are considered for the snapping operation. Another way is to use layers in which a small set of curves in the same group are put in a layer. By having multiple layers, at any given time the user will be interacting with a limited set of curves. Though these methods are not implemented in our current system, they are theoretically well-defined and can easily be implemented in practice to handle a large number of curves.

7.3.2 History of Modifications and Undo Feature

In the current implementation of the GuTS system, there is no mechanism to keep track of the sequence of operations that are performed on geometries. This prevents rolling back ('undo') to the previous step in the design process, as well as changing or modifying the previously performed geometric operations. While discrete operations can be easily tracked and task history can be easily maintained, in the GuTS approach several tasks (such as tracing and stitching) are performed simultaneously and continuously, making a history list quite tricky. In addition, with the multimodal user interface, tracking all the inputs from the different devices adds another layer of complexity. In this research we did not address this issue but it would be interesting research to pursue in the future.

7.3.3 Subdivision Schemes for the Surface Mesh

Curve modeling is implemented using the interpolation subdivision scheme. This allows the user to refine the curves and to change the level of detail of the curves dynamically and interactively. The implementation of surface modeling does not use any subdivision schemes. This prevents controlling the level-of-detail of the surface mesh dynamically. As presented in Chapter 4, the modified butterfly interpolation subdivision scheme can be used to subdivide the surface meshes dynamically and interactively. Also, the ability to dynamically control the level-of-detail, both locally and globally, will also assist in new ways to perform other geometric operations, such as tracing, stitching, smoothing, etc., in the GuTS modeling approach. While the algorithms will be complex to construct, it will considerably reduce the complexity and increase the flexibility of the geometric operations.

7.3.4 Extending Surface Stitching Operations

In our current implementation, the surfaces are stitched along the intersection curve. This is explicitly done to eliminate the introduction of new geometric shapes to the final geometry that are not from the guide shapes or the initial input geometries. This restriction connects the surfaces easily without any changes along the surface intersection boundaries. This limitation poses some restrictions, such as two surfaces can be stitched only when the surface patches completely intersect. When these surfaces do not intersect, the current implementation does not do any stitching operation. This can be overcome by implementing surface lofting algorithms. Through lofting operations, a new surface patch is created that connects the two input surfaces along the boundaries. This will stitch two or more surface patches by bringing them closer such that an intermediate surface is created. Then, the two surfaces can be stitched together. It will be the same as in curve modeling: when two curves are brought closer, they are snapped and stitched together to form the desired connectivity.

7.4 Summary

In this research we presented a novel approach called the Guided Trace and Stitch (GuTS) modeling approach using multimodal user interaction for freeform geometric modeling that simultaneously provides precision, fluency and rapidity. In the GuTS approach, complex curves and surfaces are created by stitching pieces of geometries that are traced from guide shapes. The GuTS interface provides the fluency of a sketching interface as in paper-pencil drawing and at the same time provides precision through automated guiding and stitching mechanisms. Multiple I/O modes imitate natural human interaction and design processes while providing the advantage of controlling the shapes digitally.

LIST OF REFERENCES

[1] Accuracy and precision. (n.d.). wikipedia, the free encyclopedia. retrieved january, 2007, from reference.com website: `http://www.reference.com/browse/wiki/Accuracy_and_precision`.

[2] Ascension, january 2007, `http://www.ascension-tech.com/`.

[3] Directx's direct3d, january 2007, `http://www.microsoft.com/windows/directx/default.mspx`.

[4] Logitech 3d mouse, january 2007, `http://www.3dconnexion.com/`.

[5] Multimodal. (n.d.). dictionary.com unabridged (v 1.1). retrieved january, 2007, from dictionary.com website: `http://dictionary.reference.com/browse/multimodal`.

[6] Multimodal. (n.d.) wikipedia.org. (2005). retrieved january, 2007 from `http://encyclopedia.thefreedictionary.com/multimodal`.

[7] Phantom haptic device, january 2007, `http://www.sensable.com/`.

[8] Polhemus, january 2007, `http://www.polhemus.com/`.

[9] Precision. (n.d.). wikipedia, the free encyclopedia. retrieved january, 2007, from reference.com website: `http://www.reference.com/browse/wiki/Precision`.

[10] United states department of health and human services, january 2007, `http://www.hhs.gov/ohrp/policy/index.html`.

[11] Uw institutional review board (irb), january 2007, `http://info.gradsch.wisc.edu/research/compliance/humansubjects/index.htm`.

[12] R. Arangarasan and R. Gadh. Geometric modeling and collaborative design in a multi-modal, multi-sensory virtual environment. In *Proceedings of the ASME 2000 IDETC/CIE Conference*, Maryland, USA, September 10-13 2000.

[13] R. Arangarasan and G. N. Phillips Jr. Modular approach of multimodal integration in a virtual environment. In *IEEE Fourth International Conference on Multimodal Interfaces*, USA, October 14-16 2002.

[14] R. Arsenault and C. Ware. Eye-hand co-ordination with force feedback. In *ACM Computer-Human Interaction Letters (CHI 1-6)*, volume 2, pages 408–414, April 2000.

[15] J. Arvo and K. Novins. Fluid sketches: Continuous recognition and morphing of simple hand-drawn shapes. In *Proceedings of the 13th Annual ACM Symphosium on User Interface Software and Technology*, pages 73–80, San Diego, CA, USA, November 6-8 2000.

[16] P. Asente and M. Schuster. Dynamic planar map lllustration. In *SIGGRAPH ’05: ACM SIGGRAPH 2005 Sketches*, page 92, New York, NY, USA, 2005. ACM Press.

[17] N. Badler, K. Manoochehri, and D. Baraff. Multi-dimensional input techniques and articulated figure positioning by multiple constraints. In *Proceedings of Workshop on Interactive 3D Graphics*, pages 151–169. ACM, 1986.

[18] R. Balakrishnan, G. Fitzmaurice, G. Kurtenbach, and W. Buxton. Digital tape drawing. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, pages 161–169, Asheville, USA, November 7-10 1999.

[19] R. Balakrishnan, G. Fitzmaurice, G. Kurtenbach, and K. Singh. Exploring interactive curve and surface manipulation using a bend and twist sensitive input strip. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 111–118, 1999.

[20] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proceedings of the ACM symposium on User Interface Software and Technology*, pages 185–192, Marina del Rey, CA USA, November 2-4 1994.

[21] P. Baudelaire and M. Gangnet. Planar maps: An interaction paradigm for graphic design. In *Proceedings of the SIGCHI Conference on Wings for the Mind*, pages 313–318, Austin, TX, USA, April 30 - May 4 1989.

[22] P. Bezier. Mathematical and practical possibilities of unisurf. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 127–152, New York, USA, 1974. Academic Press.

[23] E. Bier. Snap-dragging in three dimensions. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 193–204. ACM, 1990.

[24] E. A. Bier. Skitters and jacks: Interactive 3d positioning tools. In *Proceedings of Workshop on Interactive 3D Graphics*, pages 183–196, Chapel Hill, NC, USA, October 23-24 1986.

[25] E. A. Bier and M. C. Stone. Snap-dragging. In *Proceedings of the 13th Annual Conference on Computer Graphics*, volume 20, pages 233–240, Dallas, TX, USA, August 18-22 1986.

[26] H. Biermann, D. Kristjansson, and D. Zorin. Approximate boolean operations on free-form solids. In *SIGGRAPH ’01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 185–194, New York, NY, USA, 2001. ACM Press.

[27] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH ’02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 312–321, New York, NY, USA, 2002. ACM Press.

[28] J. R. Bill and S. K. Lodha. Computer sculpting of polygonal models using virtual tools. In *Technical notes: UCSC-CRL-94-27, Computer Engineering & Information Sciences*, Santa Cruz, CA, USA, July 22 1994. University of California.

[29] M. Billinghurst, S. Baldis, L. Matheson, and M. Phillips. 3d pallette, a virtual reality content creation tool. In *Proceedings of Virtual Reality and Software Technology, (VRST 97)*, pages 155–156. ACM, 1997.

[30] M. M. Blattner and E. P. Glinert. Multimodal integration. In *IEEE Multimedia*, volume 3, pages 14–24, 1996.

[31] W. Boehm and A. Mller. On de casteljau’s algorithm. In *Computer Aided Geometric Design*, volume 16, pages 587–605. Elsevier Science, August 1999.

[32] P. Borrel and A. Rappoport. Simple constrained deformations for geometric modeling and interactive design. In *ACM Transactions on Graphics*, volume 13, pages 137–155, April 1994.

[33] M. Bourguet and A. Ando. Synchronization of speech and hand gestures during multimodal human-computer interaction. In *ACM Computer-Human Interaction (CHI 18-23)*, pages 241–242, 1998.

[34] D. Bowman, D. Johnson, and L. Hodges. Testbed evaluation of ve interaction techniques. In *Proceedings of Virtual Reality and Software Technology (VRST '99)*, pages 26–33. ACM, 1999.

[35] F. Brooks. Grasping reality through illusion: Interactive graphics serving science. In *Proceedings of CHI'88*, pages 1–11. ACM.

[36] W. Buxton and B. Myers. A study in two-handed input. In *Proceedings of CHI'86*, pages 321–326. ACM.

[37] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. In *Computer Aided Design*, volume 10, pages 350–355, 1978.

[38] J. M. Cohen, L. Markosian, R.C. Zeleznik, J.H. Hughes, and R. Barzel. An interface for sketching 3d curves. In *Proceedings of 1999 Symposium on Interactive 3D Graphics*, pages 17–21. ACM SIGGRAPH, 1999.

[39] P. Cohen, D. McGee, S. Oviatt, L. Wu, J. Clow, R. King, S. Julier, and Rosenblum. Multimodal interaction for 2d and 3d environments. In *IEEE Computer Graphics and Applications*, volume 19, pages 10–13, July-August 1999.

[40] L. D. Cutler, B. Frohlich, and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Symposium on Interactive 3D Graphics*, pages 107–114, 1997.

[41] M. F. Deering. Holosketch: A virtual reality sketching/animation tool. In *ACM Transactions on Computer-Human Interaction*, volume 2, pages 220–238, 1995.

[42] D. Doo and M. Sabin. Analysis of the behaviour of recursive division surfaces near extraordinary points. In *Computer Aided Design*, volume 10, pages 356–360, 1978.

[43] N. Dyn, Levin. D., and J. A. Greogory. A butterfly subdivision scheme for surface interpolation with tension control. In *ACM Transactions on Graphics*, volume 9, pages 160–169, 1990.

[44] N. Dyn, D. Levin, and J. A. Gregory. A 4-point interpolatory subdivision scheme for curve design. In *Computer Aided Geometric Design*, volume 4, pages 257–268, 1987.

[45] G. Farin and D. Hansford. *The Essentials of CAGD*. A K Peters, Ltd., Natic, MA, 2000. ISBN 1-56881-123-3.

[46] J. D. Foley, A. van Dam, K. S. Feiner, and J. F. Hughes. *Computer Graphics - Principles and Practice*. Addison-Wesley, second edition, 1990. ISBN 0-201-12110-7.

[47] S. Gottschalk, M. C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical report, University of N. Carolina, Department of Computer Science, Chapel Hill, 1996. Technical report TR96-013. Proceedings of ACM Siggraph'96.

[48] M. A. Grasso, D. S. Ebert, and T. W. Finin. The integrality of speech in multimodal interfaces. In *ACM Transactions on Computer-Human Interaction*, volume 5, pages 308–325, December 1998.

[49] T. Grossman, D. Wigdor, and R. Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 931–931, New York, NY, USA, 2005. ACM Press.

[50] J. Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press.

[51] M. D. Hearn and P. Baker. *Computer Graphics - C Version*. Prentice Hall, second edition, 1997. ISBN 0-13-530924-7.

[52] K. Hinckley, R. Pausch, J. Goble, and N. Kassell. A survey of design issues in spatial input. In *Proceedings of User Interface Software and Technology*, pages 213–222. ACM, 1994.

[53] K. Hinckley, R. Pausch, D. Proffitt, J. Patten, and N. Kassell. Cooperative bimanual action. In *Proceedings of CHI'97*, pages 27–34. ACM, 1997.

[54] K. Hinckley and M. Sinclair. Touch-sensing input devices. In *ACM Computer-Human Interaction (CHI 15-20)*, pages 223–230, May 1999.

[55] K. Hinckley, J. Tullio, and Pausch R. Usability analysis of 3d rotation techniques. In *Proceedings of User Interface Software and Technology*, pages 1–10, 1997.

[56] M. Honda, T. Igarashi, H. Tanaka, and S. Sakai. Integrated manipulation: Context-aware manipulation of 2d diagrams. In *Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, pages 159–160, Asheville, USA, November 7-10 1999.

[57] S. Houde. Iterative design of an interface for easy 3-d direct manipulation. In *Proceedings of CHI'92*, pages 135–142. ACM, 1992.

[58] T. Hudson, M. Lin, J. Cohen, S. Gottschalk, and D. Manocha. V-collide: Accelerated collision detection for vrml. In *The Proceedings of VRML97, ACM Press*, pages 119–125, Monterey, CA, USA, February 24-26 1997.

[59] T. Igarashi, S. Kawachiya, H. Tanaka, and S. Matsuoka. Pegasus: A drawing system for rapid geometric design. In *Proceedings of the Conference on CHI 98 Summary: Human Factors in Computing Systems*, pages 24–25, Los Angeles, CA, USA, April 18-23 1998.

[60] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: A technique for rapid geometric design. In *Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology*, pages 105–114, Banff, Canada, October 14-17 1997.

[61] T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3d freeform design. In *ACM SIGGRAPH 99 Conference Proceedings*, pages 409–416, 1999.

[62] T. Ijiri, Igarashi T., S. Takahashi, and E. Shibayama. Sketch interface for 3d modeling of flowers. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Sketches*, page 6, New York, NY, USA, 2004. ACM Press.

[63] O. A. Karpenko and J. F. Hughes. Smoothsketch: 3d free-form shapes from complex sketches. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 589–598, New York, NY, USA, 2006. ACM Press.

[64] Y. Kho and M. Garland. Sketching mesh deformations. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 934–934, New York, NY, USA, 2005. ACM Press.

[65] L. Kobbelt. Interpolatory subdivision on open quadrilateral nets with arbitrary topology. In *Proceedings of Eurographics 96, Computer Graphics Forum*, pages 409–420, 1996.

[66] K. Korida, H. Nishino, and K. Utsumiya. An interactive 3d interface for a virtual ceramic art work environment. In *Proceedings of Virtual Systems and Multimedia (VSMM '97) Conference*, pages 227–234, 1997.

[67] F. Kuijt. *Convexity preserving Interpolation Stationary nonlinear subdivision and splines*. PhD thesis, University of Twente, Faculty of Applied Mathematics, 1998.

[68] Jr. J. J. LaViola. Welcome and introduction. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 1, New York, NY, USA, 2006. ACM Press.

[69] A. Leganchuk, S. Zhai, and W. Buxton. Manual and cognitive benefits of two-handed input: An experimental study. In *ACM Transactions on Computer-Human Interaction*, volume 5, pages 326–359, December 1998.

[70] I. Llamas, B. Kim, J. Gargus, J. Rossignac, and C. D. Shaw. Twister: a space-warp operator for the two-handed editing of 3d shapes. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 663–668, New York, NY, USA, 2003. ACM Press.

[71] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

[72] L. Markosian, J. M. Cohen, T. Crulli, and J. Hughes. Skin: A constructive approach to modeling free-form shapes. In *SIGGRAPH 99 Conference Proceedings*, pages 393–400, 1999.

[73] Autodesk Maya. `http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018`, September 22 2006.

[74] S. Mizuno, D. Kobayashi, M. Okada, J. Toriwaki, and S. Yamamoto. Virtual sculpting with a pressure sensitive pen. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1, New York, NY, USA, 2003. ACM Press.

[75] A. Nealen, O. Sorkine, M. Alexa, and D. Cohen-Or. A sketch-based interface for detail-preserving mesh editing. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1142–1147, New York, NY, USA, 2005. ACM Press.

[76] L. Nigay and J. Coutaz. A design space for multimodal systems - concurrent processing and data fusion. In *INTERCHI '93 - Conference on Human Factors in Computing Systems*, pages 172–178. Addison Wesley, Amsterdam, 1993.

[77] H. Nishino, M. Fushimi, K. Utsumiya, and K. Korida. A virtual environment for modeling 3d objects through spatial interaction. In *Systems, Man, and Cybernetics, IEEE SMC '99 Conference Proceedings*, volume 6, pages 81–86, 1999.

[78] H. Nishino, D. Nariman, K. Utsumiya, and K. Korida. Making 3d objects through bimanual actions. In *Systems, Man, and Cybernetics, IEEE International Conference*, volume 4, pages 3590–3595, 1998.

[79] T. Nishita. Applications of bezier clipping method and their java applets. In *Proceedings of Spring conference on computer graphics*, pages 3–15, 1998.

[80] J. Oh and W. Stuerzlinger. Sesame: 3d conceptual design system. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 39, New York, NY, USA, 2004. ACM Press.

[81] J. Pierce, B. Stearns, and R. Pausch. Two handed manipulation of voodoo dolls in virtual environments. In *Proceedings of Symposium on Interactive 3D Graphics*, pages 141–145, 1999.

[82] J. Rekimoto and E. Sciammarella. Toolstone: Effective use of the physical manipulation vocabularies of input devices. In *Proceedings of User Interface Software and Technology*, volume 2, pages 109–117, 2000.

[83] D. Rubine. Specifying gestures by example. In *ACM SIGGRAPH Computer Graphics*, volume 25, pages 329–337, July 1991.

[84] D. Rubine. Combining gestures and direct manipulation. In *Conference proceedings on Human factors in computing systems*, pages 659–660, May 3-7 1992.

[85] E. Sachs, A. Roberts, and D. Stoops. 3-draw: A tool for designing 3d shapes. In *IEEE Computer Graphics and Applications*, volume 11, pages 18–26, 1991.

[86] S. Schkolne, M. Pruett, and P. Schroder. Surface drawing: Creating organic 3d shapes with the hand and tangible tools. In *CHI 2001*, 2001.

[87] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: sketch-based solid modeling with blobtrees. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 14, New York, NY, USA, 2006. ACM Press.

[88] L. Schomaker, J. Nijtmans, A. Camurri, F. Lavagetto, P. Morasso, C. Benot, T. Guiard-Marigny, B. Le Goff, J. Robert-Ribes, A. Adjoudani, I. Defe, S. Mnch, K. Hartung, and J. Blauert. A taxonomy of multimodal interaction in the human information processing system - a report of the esprit project 8579. Technical report, February 1995. `http://hwr.nici.ru.nl/~miami/taxonomy/taxonomy.html`.

[89] T. W. Sederberg and T. Nishita. Curve intersection using bezier clipping. In *CAD*, volume 22, pages 337–345, 1990.

[90] W. T. Sederberg and R. S. Parry. Free-form deformation of solid geometric models. In *ACM SIGGRAPH 1986*, volume 20, pages 151–160, August 18-22 1986.

[91] K. Singh. Interactive curve design using digital french curves. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 23–30, Atlanta, GA, USA, April 26-29 1999.

[92] A. Stork, O. Schimpke, and R. D. Amicis. Sketching free-forms in seme-immersive virtual environments. In *Proceedings of DETC'00, 2000 ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages 5–7, Baltimore, MD, USA, September 10-13 2000.

[93] I. E. Sutherland. I. e. sketchpad - a man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, Detroit, MI, USA, 1963.

[94] D. Terzopoulos and H. Qin. Dynamic nurbs with geometric constraints for interactive sculpting. 13(2):103–136, April 1994.

[95] N. Watanabe and T. Igarashi. A sketching interface for terrain modeling. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 73, New York, NY, USA, 2004. ACM Press.

[96] A. Watt. *3D Computer Graphics*. Addison-Wesley, third edition, 2000. ISBN 0-201-39855-9.

[97] Wikipedia. Fluency — wikipedia, the free encyclopedia, 2007. [Online; accessed 28-March-2007] `http://en.wikipedia.org/w/index.php?title=Fluency&oldid=116179674`.

[98] Wikipedia. Mode (computer interface) — wikipedia, the free encyclopedia, 2007. [Online; accessed 29-March-2007] `http://en.wikipedia.org/w/index.php?title=Mode_%28computer_interface%29&oldid=111636206`.

[99] C. Yang, D. Sharon, and M. van de Panne. Sketch-based modeling of parameterized objects. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Sketches*, page 89, New York, NY, USA, 2005. ACM Press.

[100] R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. Sketch: An interface for sketching 3d scenes. In *Computer Graphics Proceedings (SIGGRAPH 1996)*, pages 163–170, 1996.

[101] S. Zhai. User performance in relation to 3d input device design. In *Computer Graphics*, volume 32, pages 50–54, November 1998.

[102] S. Zhai and P. Milgram. Quantifying coordination in multiple dof movement and its application to evaluating 6dof input devices. In *Proceedings of Computer-Human Interaction*, pages 320–327, Los Angeles, CA, USA, April 18-23 1998.

[103] S. Zhai, P. Milgram, and W. Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of CHI96: ACM Conference on Human Factors in Computing Systems*, pages 308–315, Vancouver, BC, Canada, April 13-18 1996.

[104] S. Zhai and J. W. Senders. Investigating coordination in multidegree of freedom control i: Correlation analysis in 6 dof tracking. In *Proceedings of 41st Annual Meeting of Human Factors and Ergonomics Society*, pages 1254–1258, September 22-26 1997.

[105] S. Zhai and J. W. Senders. Investigating coordination in multidegree of freedom control ii: Correlation analysis in 6 dof tracking. In *Proceedings of 41st Annual Meeting of Human Factors and Ergonomics Society*, pages 1254–1258, September 22-26 1997.

[106] D. Zorin. *Stationary Subdivision and Multiresolution Surface Representations*. PhD thesis, Caltech, Pasadena, CA, 1997.

[107] D. Zorin, P. Schrder, and W. Sweldens. Interpolation subdivision for meshes with arbitrary topology. In *Computer Graphics Proceedings, (SIGGRAPH 1996)*, pages 189–192, 1996.

[108] M. Zwicker, M. Pauly, O. Knoll, and M. Gross. Pointshop 3d: an interactive system for point-based surface editing. *ACM Trans. Graph.*, 21(3):322–329, 2002.

APPENDIX
Design and Evaluation of the GuTS System and User Interface

A.1 Introduction

In this chapter we discuss the steps involved in designing and evaluating the GuTS system's user interface. Design and evaluation involve the following steps:

- Design goals: defining the goals of the study
- User analysis: identifying and describing the intented users of the system
- Task analysis: identifying and describing the essence of the tasks to be supported by the system
- Design decisions: decisions about the design of the system
- System description: Overview of the system and a prototype of the interface
- Evaluation-I: Usability inspection of the prototype system
- Evaluation-II: Usability testing of the revised system with the user's response

Figure A.1 shows the overall layout of the design and evaluation study of the GuTS system's user interface. Studies that involve human subjects (users) require the approval of the UW's 'Institutional Review Board' (IRB) [11] and are

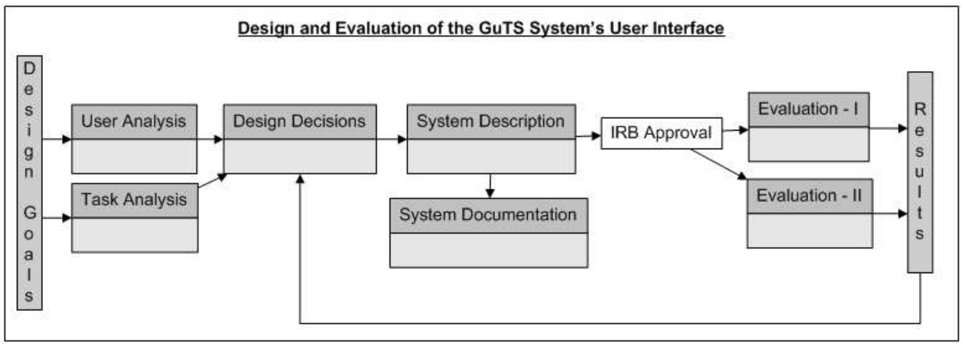


Figure A.1 Design and study of the GuTS system user interface

monitored by United States Department of Health and Human Services [10]. In this research, due to limited resources - shortage of time, ability to recruit a wide variety and group of test subjects, physical facility to build the experimental setup, gathering the subject's feedback, recursively implement the feedback into the system - we decided not to perform the actual human subjects study, but design and present the evaluation process. This procedure allowed us to think in terms of refining important design aspects that would be required during the design of the GuTS system.

A.2 Goals of this Study

The goal of this study is to measure the success rate of the GuTS approach using the GuTS system to achieve precise, fluent, and rapid modeling through an experimental user study.

The above goal can be divided into sub-goals and each sub-task's individual characteristics can be measured as follows:

- Feasibility - The GuTS system can be used to model freeform curves and surfaces
- Guided Trace and Stitch Paradigm - Eliminates the need to possess a complex knowledge of geometric modeling and provides a WYSIWYG user interface
- Pen-based input - Allows for easy, direct, and fluent tracing operation
- 3D Mouse input - Eliminates mapping between 2D and 3D input, and provides direct manipulation in 3D
- Speech input - Allows the user to input without interrupting current tasks and decreases ‘mode switching’
- Auditory output - Eliminates distraction of the user's visual attention from the current focus and acts as a guiding mechanism
- Snapping mechanisms - Provides rapid alignment of geometries precisely and decreases error produced by imprecise user input
- Two-handed input - Increases productivity (compared to uni-manual input) and mimics the real-world drafting process with two-hands
- Visual data tablet - Provides better hand-eye coordination (compared to the regular 2D mouse and monitor setup) and mimics real-world paper-pencil drafting
- Multimodal user interface - Eliminates mode switching and allows users to perform multiple coordinated tasks simultaneously

A.3 User Analysis

User analysis includes the definition of the target audience. Some of the key parameters that need to be considered are:

- Physical characteristics (age, sex, nationality or ethnicity, demographic information, perceptual abilities, e.g., perceptual handicaps, motor skills and physical disabilities)
- Knowledge and experience (level of education, reading level, native language, knowledge of particular domain-specific terminology, etc.)
- Computer / IT experience or knowledge (computer literacy, level of experience with similar systems, level of experience with other systems, etc.)
- Level of experience with the task
- Psychological characteristics (attitudes, motivation to use the system, etc.)
- Quantitative characteristics (how many subjects will be needed, any need to maintain proportion of the subjects, etc.)
- Financial and compensatory aspects (free or need to pay the subjects, credit them as part of their course, etc.)

For studying the GuTS system, the requirements of the users are:

- Physical characteristics: No restrictions on age, sex, nationality or ethnicity. Users must have no perceptual or physical disabilities.
- Knowledge and experience: Users should have English language proficiency in reading and writing.
- Computer / IT experience: Users should have general knowledge of the personal computer (PC) and the Windows operating system, basic knowledge in using interactive graphics application(s) - not necessarily CAD modeling systems.
- Level of experience with the task: None
- Psychological characteristics: None
- Quantitative characteristics: Number of users will be 20 to 30
- Financial aspects: Ideal candidates would be a group of users that are interested in participating in this study for free.

A.4 Task Analysis

Task analysis focuses on the definition of the task from the user's perspective. The key parameters that need to be considered are divided into two parts: tasks to be performed and characteristics of the tasks.

The tasks to be performed:

- Hierarchical description of activity (structure charts, GOMS models, goals and subgoals, etc.)
- Essential use cases (As 'leaves' of the hierarchical tree)
- Scenarios of use (typically at least one for each essential use case, multiple for important use cases, etc.)

The tasks to be performed that are specific to the GuTS system are:

- Hierarchical description of activity: Depending upon the tasks, structure charts, GOMS models, and goals and subgoals will be used.
- Essential use cases: Core tasks include tracing, stitching, manipulating geometries and viewpoint, pen-based input, 3D mouse input for manipulation, usability of the GuTS windowing system, speech input, sound output, and two-handed coordinated manipulation.
- Scenarios of use: Create specific tasks that require the user to perform each of the above tasks, so the completion and success rate of each task can be independently measured and quantified.

General description of task characteristics:

- What are the tasks (What are the exact tasks the users perform? What to observe for each task? What are the benchmarks? What determines success or failure?)
- The frequency or timing of the task (How frequently do users perform the task? What are the time constraints on the task? etc.)
- The complexity and difficulty of the task (How complex is the task? How difficult is the task? How structured is the task? etc.)
- The relationship of the task to other user tasks (Is system use mandatory or discretionary? How important is the task? What is the relationship between the users and the data? etc.)
- The physical environment of task performance (Where is the task performed? What other tools does the user have? etc.)

- The social, organizational and cultural environment of the task (What kind of relationships subjects have with other people in the workplace or on the work team? What are the effects of organizational, national, or ethnic/cultural differences? etc.)
- Planning for learning and breakdowns (What training will be provided? How is the task learned? What can be said to the users without contaminating the experiment? What are the necessary steps needed to eliminate bias? What happens when things go wrong? etc.)

Specific to the GuTS system, the general description of the task characteristics are:

- What are the tasks?
  - Exact tasks users will perform include: creating complex guide shapes, tracing over the guide shapes to create new geometries, performing automatic stitching operations, manipulating the geometries using other geometric operations, converting 2D curves to 3D surfaces, tracing and stitching 3D surfaces, using speech input for inputting commands, using a 3D mouse for geometric and viewpoint manipulation.
  - Following are some of the things that will be observed for each task: is the task completed successfully? Is task completed in allotted time? How much time is taken for each task? How many times the user has to try to finish a task successfully? If a task can be done through multiple modes, which mode was used to complete the task?
  - Benchmarks: benchmarks include correctness and completion of the task and, in some cases, within the allotted time.
  - Success or failure: this will be determined based on whether a task is completed or not. For some tasks, it will be based on whether or not the task is performed within a certain period of time.
- The frequency or timing of the task: at this stage, how frequently users perform certain operations will be measured - this identifies the most frequently used tasks, how much time is being spent on these tasks, and determines the number of ‘mode switches’. When users perform continuous operations such as tracing, stitching and geometric manipulation in the GuTS system, it will be difficult to monitor these operations visually. In those circumstances, it will be helpful to develop a custom feature that tracks the usage of GuTS operations and the subsequent time-stamps. Once these operations and time-stamps are recorded for each user and task, the required information can be retrieved automatically from the recorded data with little effort and great accuracy.
- The complexity and difficulty of the task: complexity of a task in the GuTS system can be measured through different parameters - such as number of sub-tasks required to finish a task, total time required to finish a task, different types of sub-tasks involved to complete a task, user’s analytical and motor skills required to complete a task, etc.

- The physical environment of task performance: the study will be performed in an academic setting in a computer lab - the tools that will be provided are as shown in Fig. 5.2.
- Planning for learning and breakdowns: users will be debriefed on the usage of the input/output devices; how to navigate and control the interface in the GuTS system; one or two examples will be demonstrated to the user; the user may then spend a few minutes using the GuTS system before the experiment starts. Users will know how the GuTS system can be used but will have no details as to why a certain feature is implemented in certain way so as to eliminate bias and avoid contaminating the experimental results.

A.5 Design Decisions

At this stage, decisions are made on how the system will be implemented. Key design decisions are documented formally. This documentation will contain, for each design decision, the decision itself, alternatives considered and the rationale for selecting a particular alternative. The rationale may draw on published usability guidelines, prior relevant research studies or data gathered from the target audience.

Some of the key design decisions and reasons to choose them are listed in Table A.1.

A.6 System Description

In this stage, design decisions are made concrete. These will be documented in two forms: (a) an overall layout of the system architecture and (b) a prototype of all or a major portion of the system. A detailed description of the GuTS system's user interface is presented in Chapter 5.

A.7 Evaluation

Evaluation of the user interface can be classified in several ways. One method of classification is summative evalution and formative evaluation.

In summative evalution:

- Evaluation of the user interface is done after the system has been developed
- Typically performed once at the end of development
- Not a formal procedure
- Evaluation data is used in the next major version of the software

In formative evaluation:

- Evaluation of the user interface is done while the system is being developed

Design decisions	**Reasons**
Pen-based input	Draw, sketch, and trace mimics the process as in the paper-pencil drawing paradigm; Easy, direct, and fluent to trace geometries
3D-mouse input	Allows direct manipulation of 6-DOF in 3D; Eliminates mapping between 2D and 3D input
Speech input	Provides an additional mode of input that allows the user to input without interrupting the current tasks; Decreases ‘mode switching’
Auditory output	Provides an additional mode of output and does not distract the user’s visual attention from the current task; Acts as a user guiding mechanism
Guided trace and stitch paradigm	Eliminates the need to possess a complex knowledge of geometric modeling; Enables precise modeling; Provides a WYSIWYG user interface and short learning curve
Snapping mechanisms	Provides rapid alignment of geometries precisely; Eliminates the need to possess a complex knowledge of geometric modeling; Provides a WYSIWYG user interface
Two-handed input	Increases productivity (compared to uni-manual input); Mimics the real-world drafting process with two hands
Visual data tablet	Provides better hand-eye coordination (compared to the regular 2D mouse and monitor setup); Direct sketching over the geometries as in paper-pencil drawing
Multimodal user interface	Provides multiple coordinated I/O interface; Eliminates mode switching; Allows manipulating geometries while performing additional tasks simultaneously

Table A.1 Key design decisions and reasons

- Evaluation starts as soon as the system development cycle starts
- Evaluation appears as part of the prototyping
- Very formal and well organized procedure
- Evaluation is performed several times during each development cycle

Data collected through formative evaluation is classified as follows:

- Objective Data: Data that is observed directly, and these are facts.
- Subjective Data: Data that is generally the opinions of the users. Sometimes, this is a hypothesis that leads to additional experiments.
- Quantitative Data: Data that is numeric, performance metrics, opinion ratings, statistical analysis, etc. This data tells that something is right or wrong.
- Qualitative Data: Data that is non-numeric, user opinions, observations, etc. This data tells what is right or wrong.

The steps in formative evaluation include: design the experiment and user study; conduct the experiment and user study; collect the data; analyze the data; draw conclusions and establish a hypothesis; incorporate the results and conclusions into the system; and repeat the whole cycle again.

Formative evaluation consists of different methods. They are:

- Inspection methods: in this method, usability experts inspect the system during formative evaluation.
- Testing methods: in this method, usability tests are conducted with real users under observation by experts.
- Inquiry methods: in this method, usability evaluators collect information about the user's likes, dislikes and understanding of the interface.

It is quite common to do a 'pilot study' before performing a large user study. A 'Pilot study' is an initial run of a study for the purpose of verifying that the test itself is well-formulated. For example, a user takes the test to check whether the test script is clear, the tasks are not too simple or too hard, the time allotted to perform the task is reasonable for the user, and that the data collected can be meaningfully analyzed.

A.7.1 Evaluation-I

Before the experiment starts, first debrief the user. During debriefing the user is told at least the following: purpose of the study; what they will be doing; explain they can quit at anytime without any consequences; describe the equipment and software setup; tell them what will be recorded; and how long the session will take, etc. Mostly, design the experiment such that the whole session lasts about an hour, but no longer than 2 hours (unless more than 2 hours are required for the experimental study).

Then give the user a quick demo or overview of the system itself. If necessary, provide the user a hands-on experience before the experiment starts. Then the user will start the experiment. Data will be collected in multiple forms as discussed above (inspection, testing, and inquiry methods). For the inquiry method, an evaluation/survey form (a questionnaire) is a widely used method to collect data from the user. An evaluation form is a thoroughly prepared questionnaire that will be completed by each subject before, during and after the test. Things to consider while preparing the questionnaire are: ask the right questions; you only get one opportunity to ask; and questions need to be clear consistent and enable quick responses. The questionnaire itself is not presented here but this questionnaire will be thorough and covers all the key aspects that were discussed in this chapter.

Once evaluation-I is complete, results from the subjects' evaluations are analyzed and compared to the design goals of the system. Data can be analyzed in several ways - some commonly used statistical methods are ANOVAs and Chi-Squared tests. The data and results from the evaluation study should support the designer's goals and conclusions. Any shortcomings need to be addressed and incorporated into the system's design. If necessary, new hypotheses will be established based upon the data and the system redesigned.

A.7.2 Evaluation-II

Once the changes based on evaluation-I are incorporated into the system, the next stage of evaluation is performed. At this second evaluation stage, the subjects can be the same from the evaluation-I or they can be totally different. Having the same subjects would be helpful in comparing the results between evaluation-I and evaluation-II. Sometimes it is best to keep the experiment same but sometimes it is good to change the experiment depending upon the amount of changes that happened since the previous experiment. Then, the whole cycle continues - performing the experiment, gathering data, analyzing results, and incorporating changes to the system.