

**CREDIT: TELE-IMMERSIVE CRANIAL IMPLANT DESIGN IN A  
HAPTIC AUGMENTED REALITY ENVIRONMENT**

BY

CHRISTOPHER SCOTT SCHARVER  
B.S., Computer Science, Furman University, 1998

THESIS

Submitted as partial fulfillment of the requirements  
for the degree of Master of Science in Computer Science  
in the Graduate College of the  
University of Illinois at Chicago, 2005

Chicago, Illinois

Copyright by  
Christopher Scott Scharver  
2005

To my parents,

Jeffrey and Candice Scharver

## ACKNOWLEDGMENTS

The efforts put into this thesis could not have been possible without a tremendous amount of support from my family and friends. I could never have completed this thesis without their never-wavering love and support. Additionally, I would like to express my gratitude to the others who have contributed their support.

In summer 1998, I contacted UIC swimming coach John Christie to inquire about the possibility of participating with UIC's team. I am an avid swimmer, but there was no team at my undergraduate institution. Unfortunate circumstances prevented me from competing in the 1994 North Carolina high school swimming and diving state finals.

John Christie lost his battle with cancer in summer 2000 at the age of 43. I never had the opportunity to thank him for allowing me the pleasure of joining UIC's swimming and diving team. That single year of participation answered for me a question that had been plaguing me during the previous four years. I questioned if circumstances had been different, would I have been able to compete for a NCAA Division I program? The answer, provided by the varsity letter that I earned as a member of the 1998-1999 UIC Men's Swimming and Diving team, is something I treasure very dearly.

The time required for swimming training and competition was significant, and I appreciate the support from the EVL staff. I cannot understate my appreciation for allowing me the chance to participate on the team. EVL provided a wonderful environment filled with amazing technology. Through the research in which I participated, I traveled to many places. Most notable were the workshop at SARA in Amsterdam, the Netherlands, and the iGrid 2000 conference in Yokohama, Japan. In collaborating

## **ACKNOWLEDGMENTS (Continued)**

with researchers around the world, EVL provided an amazing graduate experience that far exceeded my expectations.

The Virtual Reality in Medicine Laboratory at the University of Illinois at Chicago (UIC) is funded, in part, by the National Library of Medicine/National Institutes of Health contract N01-LM-3-3507. The Electronic Visualization Laboratory (EVL) at UIC receives major funding for virtual reality equipment acquisition and development and advanced networking research from the National Science Foundation (NSF), awards EIA-9871058, EIA-9802090, EIA-0224306 and ANI-0129527, as well as the NSF Partnerships for Advanced Computational Infrastructure (PACI) cooperative agreement (ACI-9619019) to the National Computational Science Alliance. Previously, EVL received funding for PARIS from the US Department of Energy (DOE) ASCI VIEWS program. PARIS is a trademark of the Board of Trustees of the University of Illinois.

CSS

## TABLE OF CONTENTS

<u>CHAPTER</u>		<u>PAGE</u>
<b>1</b>	<b>INTRODUCTION</b> . . . . .	1
	1.1 Background . . . . .	1
	1.2 Problem Statement . . . . .	2
	1.3 Purpose of the Work . . . . .	2
	1.4 Significance of the Problem . . . . .	3
	1.5 Solution Overview . . . . .	4
<b>2</b>	<b>CONCEPTS AND RELATED WORK</b> . . . . .	5
	2.1 UIC Cranial Implant Design Research . . . . .	5
	2.1.1 Personnel . . . . .	5
	2.1.2 Data Acquisition . . . . .	7
	2.1.3 Defect Specification . . . . .	8
	2.1.4 Implant Fabrication . . . . .	9
	2.1.5 Surgical Procedure . . . . .	11
	2.2 Related Literature . . . . .	12
	2.2.1 Cranial Implants . . . . .	12
	2.2.2 3D User Interaction . . . . .	13
	2.2.3 Immersive Modeling . . . . .	14
	2.2.4 Tele-Immersion . . . . .	17
	2.2.5 Augmented Reality . . . . .	18
<b>3</b>	<b>APPROACH</b> . . . . .	21
	3.1 Usage Scenario . . . . .	22
	3.2 Benefits of Virtual Reality . . . . .	23
	3.3 Materials . . . . .	24
	3.3.1 PARIS . . . . .	24
	3.3.2 PHANToM . . . . .	25
	3.3.3 Device Configuration . . . . .	26
	3.4 Software Libraries . . . . .	28
	3.5 Identifying Library Responsibilities . . . . .	32
<b>4</b>	<b>METHODOLOGY</b> . . . . .	35
	4.1 Software Integration . . . . .	35
	4.1.1 OIVAppShell . . . . .	36
	4.1.2 OIV-GHOST Integration . . . . .	37
	4.1.3 coinevents Library . . . . .	38
	4.1.4 vtkoiv Library . . . . .	39

## TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>		<u>PAGE</u>
	4.1.5 QAvatar Library . . . . .	41
	4.2 Application Design . . . . .	44
	4.3 Capabilities . . . . .	47
	4.3.1 Line Drawing Tool . . . . .	47
	4.3.2 Defect Specification Tool . . . . .	48
	4.3.3 Implant Generation Tool . . . . .	54
	4.3.4 Slice Tool . . . . .	56
<b>5</b>	<b>DISCUSSION . . . . .</b>	<b>58</b>
	5.1 Demonstrations . . . . .	58
	5.2 Domain Expert Evaluation . . . . .	59
	5.3 Rehabilitation . . . . .	61
	5.4 Other Potential Applications . . . . .	62
<b>6</b>	<b>CONCLUSIONS . . . . .</b>	<b>64</b>
	6.1 Lessons Learned . . . . .	64
	6.1.1 Hardware Issues . . . . .	64
	6.1.2 Software Techniques . . . . .	65
	6.1.3 Defect and Implant Specification . . . . .	66
	6.2 Implications for Future Research . . . . .	66
	6.3 Contributions . . . . .	68
	<b>APPENDICES . . . . .</b>	<b>71</b>
	<b>Appendix A . . . . .</b>	<b>72</b>
	<b>Appendix B . . . . .</b>	<b>74</b>
	<b>CITED LITERATURE . . . . .</b>	<b>78</b>
	<b>VITA . . . . .</b>	<b>85</b>

## LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	Personnel involved with the cranial implant design process. The referring clinician initiates communication with the neurosurgeon, who in turn collaborates with the medical sculptor prior to the surgical procedure. . . . .	6
2	Medical imaging software extracts a 3D model from the patient data. The software provides an interface for setting threshold values to segment the bone from surrounding tissues. . . . .	8
3	Stereolithography process generates a physical defect model from the digital model file. This physical model is used as a guide during the fabrication process. . . . .	10
4	After the stereolithography model forms a cast, the medical modeler sculpts the implant in a dental-grade wax. The meticulous nature requires using precision tools while sculpting the implant. . . . .	11
5	A medical modeler touches a virtual skull using PARIS. The stylus corresponds to a tool held in the right hand, while a virtual hand tracks the position of the left hand. A real model remains visible on the table. . . . .	22
6	PARIS configuration. . . . .	26
7	SensAble’s PHANToM Desktop is a commercially available haptic robot with force feedback capabilities. Image courtesy SensAble Technologies. . . . .	27
8	Data flow among software libraries . . . . .	30
9	The controller classes generate custom Coin3D user interaction events for uniform handling by other classes. . . . .	39
10	vtkActorToIV converts a vtkActor into an OIV SoShapeKit with geometry duplicating the actor’s points, lines, polygons, and triangle strips. . . . .	40
11	QAvatar library classes. . . . .	42
12	Sample CrEdit configuration file. . . . .	45
13	Controllers implement device communication for different tool implementations. . . . .	46
14	The modeler specifies the defect by interactively outlining its edges.. . . .	49
15	Defect geometry before and after filling in the edge using a vtkRuledSurfaceFilter. . . . .	51
16	The VTK defect construction pipeline processes the OIV lines and skull geometry. . . . .	52
17	The defect after interactive specification. . . . .	53
18	After the modeler draws the implant, CrEdit fills in the geometry. . . . .	54
19	The VTK implant generation pipeline. . . . .	55
20	The Slice Tool explores image-based medical slices. . . . .	57
21	CrEdit demonstration on PARIS at SC2002 in Baltimore, Maryland. . . . .	59
22	CrEdit environment with simulated patient information systems. . . . .	67



**LIST OF FIGURES (Continued)**

<b><u>FIGURE</u></b>		<b><u>PAGE</u></b>
23	Letter from Fady Charbel, MD, head and professor, UIC Department of Neurosurgery. . . . .	73

## LIST OF ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
3DUI	Three Dimensional User Interaction
API	Application Programming Interface
AR	Augmented Reality
CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAVE	CAVE Automatic Virtual Environment
CT	Computed Tomography
CVE	Collaborative Virtual Environment
DICOM	Digital Imaging and Communications in Medicine
DOF	Degrees of Freedom
EVL	Electronic Visualization Laboratory at UIC
GHOST	General Haptic Open Software Toolkit
GUI	Graphical User Interface
IP	Internet Protocol
OIV	Open Inventor

## **LIST OF ABBREVIATIONS (Continued)**

PARIS	Personal Augmented Reality Immersive System
SDK	Software Development Kit
TCP	Transmission Communication Protocol
UDP	Unreliable Data Protocol
UIC	University of Illinois at Chicago
VE	Virtual Environment
VR	Virtual Reality
VRML	Virtual Reality Modeling Language
VTK	Visualization Toolkit

## **SUMMARY**

Repairing severe human skull injuries requires customized cranial implants, and current visualization research aims to develop a new approach to create these implants. Following pre-surgical design techniques pioneered at the University of Illinois at Chicago (UIC) in 1996, researchers have developed an immersive cranial implant application incorporating haptic force feedback and augmented reality. The application runs on the Personal Augmented Reality Immersive System (PARIS(tm)), allowing the modeler to see clearly both his hands and the virtual workspace. The strengths of multiple software libraries are maximized to simplify development. This research lays the foundation to eventually replace the traditional modeling and evaluation processes.

# CHAPTER 1

## INTRODUCTION

### 1.1 Background

Today a medical sculptor utilizes anatomical modeling expertise to sculpt a prosthetic implant. However even with the aid of automated manufacturing techniques, the design process poses several problems. Techniques developed at the UIC in 1996 have greatly improved the practice. This thesis aims to augment these techniques using a new virtual reality display system and customized software tools.

Involving extensive planning during the pre-surgical design process, closing large cranial defects offers patients therapeutic benefits. These benefits include restoring the shape of the head, protecting vital brain tissue, minimizing pain, reducing operating and recovery times, and in some cases improving cognitive capabilities. Unfortunately, several factors limit cranial implant availability. Insurance companies often will not support this form of reconstruction due to the high labor and material costs. Because only neurosurgeons and medical modelers possess the specialized anatomical knowledge, assembling the necessary expertise is difficult. Travel expenses for both patients and specialists increase the overall cost. Currently acrylic polymer is the most commonly used implant material. When used intra-operatively, the material exudes extreme heat as it solidifies. Exposing the brain to these temperatures can cause tissue damage. For this reason, it is vital that implant design and fabrication takes place prior to surgery.

Traditional cranial implant fabrication and surgical placement methods are heavily dependent on subjective skills and procedures requiring specialized knowledge. The cranium's anatomical complexity impedes reconstruction without extensive planning. Additionally, the implant design must take into account features unique to the patient. Pre-surgical cranial implant design and fabrication alleviates many of these shortcomings.

## **1.2 Problem Statement**

The 1996 UIC approach incorporated a series of expensive manual manufacturing steps to produce a custom-fitting implant. These techniques serve as a guide for implementing a digital approach combining augmented reality and haptics. Medical sculptors are trained using their hands, and their abilities are heavily dependent on that fact. The awkward control devices used by traditional VR systems are not conducive to intricate sculpting techniques. Introducing force feedback allows the user to feel the virtual defect and implant models, giving the user the sense of touch that is important for 3D modeling (Massie, 1998). The design process also requires medical sculptors clearly see their hands while modeling an implant. Using augmented reality we can combine real and virtual information, allow real time interactivity, and manage 3D registration (Azuma, 1997).

## **1.3 Purpose of the Work**

This thesis centers around mutual interests between UIC's EVL and Virtual Reality in Medicine Laboratory. Force feedback robots often operate adjacent to associated displays, but this mode of operation is non-intuitive for medical sculptors accustomed to directly working with physical models. By combining graphics and haptics, this thesis presents a prototype environment for examining cranial implant

design. CrEdit presents an augmented reality environment combining sensory feedback, established by the following purposes:

- Integrate PHANToM haptics with AR
- Load medical data into 3D environment
- Examine tools required by medical sculptor
- Follow existing interaction metaphors
- Provide foundation for future research

#### **1.4 Significance of the Problem**

In 1999 alone, surgeons performed 220,065 cranial procedures (American Association of Neurological Surgeons, 1999). These operations are complex, requiring the medical skills of craniofacial specialists. Physically gathering these individuals can be challenging in terms of time and personnel. Moreover, the current state of the art cranial implant design techniques still occur in clay and wax materials. These substances do not lend themselves to long-distance collaboration for pre-surgical design and evaluation purposes (Ray Evenhouse, personal communication, Mar 2001). By taking advantage of advanced visualization facilities available at UIC, this research aims to provide a foundation for streamlining cranial implant creation, ultimately making it a more efficient and more readily-available procedure. As described by Parvati, the graphics and imaging will play increasing roles in medicine (Parvati, 2000). Exploring these roles requires collaboration between the technical personnel and the medical experts most familiar with the problem domain. Integrating complex graphics displays with haptic force

feedback poses interaction and computation challenges. Although CrEdit focuses on cranial implants, the entire system should eventually be applicable to other craniofacial surgical procedures.

## **1.5 Solution Overview**

Stereovision, wide angle-of-view, interactivity and viewer-centered perspective help the participants understand the depth relationships. Rather than looking at improperly scaled models on a computer monitor, the user perceives objects at an absolute scale. Combined with the sense of touch supplied by a PHANTOM haptic device (Salisbury and Srinivasan, 1997), this creates a rich sensory environment for developing a medical implant design application. Connecting these systems across high-speed networks enables users in different locations to collaborate among shared, virtual, and simulated environment (Park et al., 2000). Geographically scattered participants may discuss implant design, surgical pre-planning, and postoperative evaluation. This also provides educational opportunities, enabling instructors to interactively present methods and techniques. This thesis presents CrEdit, a prototype system that allows loading and manipulating data, marking areas of interest, extracting defect geometry, and feeling virtual bone surfaces.



## CHAPTER 2

### CONCEPTS AND RELATED WORK

This chapter describes the background concepts in detail. Previous UIC cranial implant techniques provide the basis for the virtual tools and capabilities. Additionally, the examination of related literature explores virtual environment concepts applicable for cranial implant design.

#### **2.1 UIC Cranial Implant Design Research**

Dr. Fady Charbel, Ray Evenhouse and their team at UIC pioneered a pre-surgical cranial implant design technique in 1996 (Dujovney et al., 1999). The process produces custom-fitting cranial implants prior to surgery using the patient's computed tomography (CT) data. The medical modeler loads the patient's CT data into medical computer-aided design (CAD) software and produces a digital model of the defect area. This polygonal model is exported to a rapid prototyping stereolithography machine. This computer-aided manufacturing (CAM) process fabricates a physical defect model. The medical modeler sculpts, molds, and casts the implant based on this model. Shaping the clay on the defect model, the modeler progressively sculpts the form of the implant's mold. The modeler then casts the implant by filling the mold with a medical-grade polymer. After casting, the implant is sterilized and prepared for surgery. To date, nine patients have received implants using this method.

##### **2.1.1 Personnel**

The cranial implant design process involves several personnel. Interaction between these personnel is currently sequential in nature. Three primary people share information about the patient. The flow of

information begins with the referring clinician. This person provides the initial patient evaluation, and he passes information in the form of a referral to the neurosurgeon. Because attaching a cranial implant to a person is an invasive surgical process, the neurosurgeon decides how to proceed considering the patient's situation. The neurosurgeon collects the patient's medical data and provides it to a medical sculptor. Figure 1 illustrates the flow of information among personnel involved with the cranial implant design process.

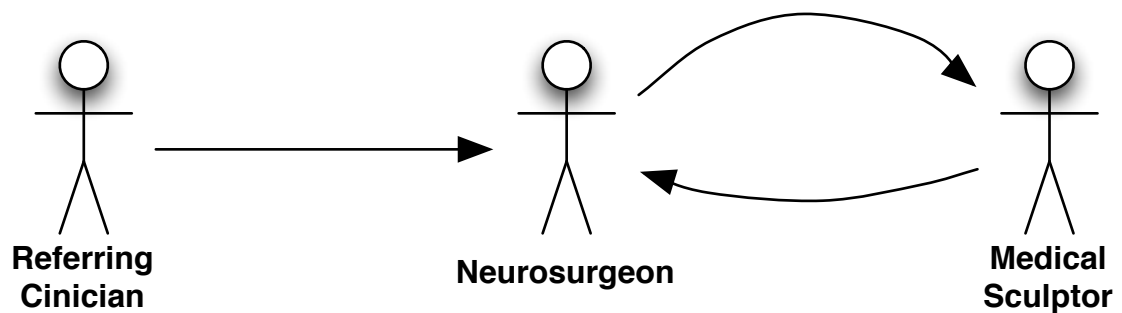


Figure 1: Personnel involved with the cranial implant design process. The referring clinician initiates communication with the neurosurgeon, who in turn collaborates with the medical sculptor prior to the surgical procedure.

The referring clinician initiates the process based on the patient's needs. This clinician may have little communication with the neurosurgeon and medical sculptor, but he is the one with the most direct contact with the patient. This history may provide valuable information essential to the design process. For example, the referring clinician may establish contact with other family members who may exhibit

similar physical features to the patient. The medical sculptor evaluates this information during the design process in order to create the custom implant. As the individual performing the actual surgery, the neurosurgeon must communicate with both the referring clinician and the medical sculptor during the planning process. Typically, the neurosurgeon does not see the implant prior to its arrival in the operating room. These communication restrictions complicate movement of information and ideas among personnel.

### **2.1.2 Data Acquisition**

A digital representation of the patient's skull is the basis of the implant design process. Using a General Electric High-Speed Advantage scanner, technicians generate computed tomography (CT) scans containing 1 mm thick slices. The process archives these images onto optical disk using the Digital Imaging and Communications in Medicine (DICOM) protocol. The system also sends the images through the university's data network to a graphics workstation. Data can then be stored for later processing or imported directly into other medical software as depicted in Figure 2.

The scanning system collects between 200-300 image slices at 1 mm intervals. Each image is 512x512 pixels with 4096 intensity levels. The gray intensity corresponds to the material density for the particular pixels. The dataset for a single patient is approximately 250 megabytes, although newer scanning systems output both higher resolution and thinner slices. As the scanning systems generate more data, the resulting images grow in size. The medical sculptors require a way to rapidly and interactively view a 3D reconstruction based on the patient's scanned data.

Medical sculptors already use CT data for pre-surgical planning. Incorporating this information into the virtual environment provides another resource available within the digital medium. The direct

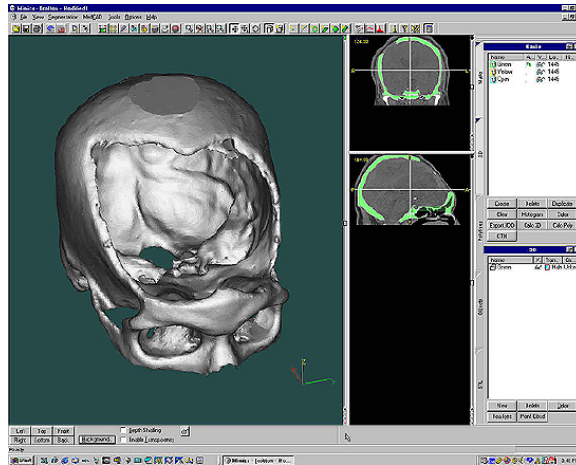


Figure 2: Medical imaging software extracts a 3D model from the patient data. The software provides an interface for setting threshold values to segment the bone from surrounding tissues.

manipulation of slices is extremely fast, as the application only adjusts texture coordinates without doing any texture processing. The resolution is limited by available graphics memory.

### 2.1.3 Defect Specification

Several processing steps must be completed to construct a file suitable for rapid prototyping. Pixel data from the CT slices undergoes segmentation to extract the skull information. Semiautomatic algorithms in commercial software initiate the segmentation scheme, but each slice is checked and manually corrected as necessary. The edges of the inner and outer tables of the cranium are identified, and their contours are converted from a pixel map representation to a vector map. The pixel to vector map conversion establishes the geometry for each skull outline from the CT slices.

Pixels and their volumetric equivalent, voxels, are image representations only and do not explicitly establish the geometry of the objects they represent. Conversely, vector maps are mathematical curves explicitly defining geometry. Vector maps from each CT slice are then connected to generate a surface model of the skull. To create an appropriate stereolithography output file for use in rapid prototyping, the software positions the surface model in the optimal orientation for the build cycle and re-samples the data into appropriate slice thickness. The re-sampled data now represents a tool path guide for the stereolithography machine's model generation process.

The defect geometry file is imported into a stereolithography machine. This system contains a laser positioned over a vat of liquid polymer. As the laser fires into the polymer, its energy converts the polymer to a solid mass. As each slice from the CAD file is traced and solidified, the platform on which the model is being built is lowered. The surface of the developing model is flooded, and polymerization by the laser beam is repeated. Slice-by-slice, a solid plastic form is gradually created. The finished model is removed from the vat, drained of excess polymer, and submitted to a final curing stage using ultraviolet radiation. The final model is a precise physical representation of the patient's skull and defect CT data. To save costs, only the portion of the skull immediately surrounding the defect is generated. Figure 3 depicts the resulting model.

The UIC pre-surgical implant process generates a defect model from the patient's CT data. This model serves as the basis for sculpting the implant with clay and wax in the next step of the process.

#### **2.1.4 Implant Fabrication**

The stereolithography model serves as a physical template for fabricating the final implant. Modeling clay is applied to the plastic model of the patient's defect. The clay is contoured to represent the



Figure 3: Stereolithography process generates a physical defect model from the digital model file. This physical model is used as a guide during the fabrication process.

inner prosthesis surface. A silicone rubber mold is made of this composite piece, and dental stone is poured into this mold. This produces a physical model of the defect that serves as the foundation upon which the implant is built.

A dental grade wax is used to fill the negative space representing the skull defect. Care is exercised in maintaining appropriate contour and thickness to assure the best cosmetic appearance following implantation. Figure 4 displays several tools and materials used by the sculptor during the modeling fabrication process.

When the wax pattern for the implant is completed, the remaining half of the mold can be poured. Registration notches are cut into the defect side of the mold, so that the two mold halves can be perfectly realigned. Dental stone is poured to form the top half of the mold.



Figure 4: After the stereolithography model forms a cast, the medical modeler sculpts the implant in a dental-grade wax. The meticulous nature requires using precision tools while sculpting the implant.

The mold halves are separated and the wax is removed, leaving a negative space to receive the implant material. Medical-grade acrylic monomer liquid and polymer powder are mixed to form a soft, pliable mass and carefully packed into the mold to avoid air entrapment. The mold is then closed, tightly clamped, and placed in a hot-water bath to slow-cure the resin. Following the prescribed cure time, the solid implant is removed from the mold. Additional curing in an oven releases any residual free monomer trapped in the implant. After trimming and polishing, the finished implant is sent to the operating suite for sterilization.

### **2.1.5 Surgical Procedure**

Surgical implantation is the final step of the process. Depending on the bone defect location, the surgeon positions the patient in the appropriate position. After shaving and cleansing the surgical site, the surgeon drapes the patient according to the cranial defect. The implant is fixed in place with a

minimum of three titanium plates and screws. The surgeon seals the incision using sutures and closes the skin with staples.

## **2.2 Related Literature**

Previous work explores both computer-aided cranial implant design and immersive modeling, but the challenges of unifying these concepts into an immersive cranial implant design application have not been examined. Any immersive modeling tools must be appropriate for the tasks, as the medical modeler requires specific tools and capabilities. Related literature explores computer-assisted cranial implant design, 3D user interaction, immersive modeling, tele-immersion, and finally, augmented reality.

### **2.2.1 Cranial Implants**

Key related work leverages computer-aided design during the planning process, and several methods successfully created cranial implants. Taha et al. combines rapid prototyping and commercial modeling software (Taha et al., 2001), as it uses implicit surfaces to create a filled implant. This commercial software is strictly designed for desktop interaction. There are no immersive capabilities. Another method by Burgert et al. utilizes skull symmetry and numerical analysis to provide solutions in certain cases (Burgert et al., 2003). If symmetry cannot be exploited, then this approach does not provide a solution. Irregular defects that cross the axis of symmetry would thus not be treatable.

While UIC uses medical modeling software to build the defect, these projects construct the implant entirely in software. In either case, such tools require complex interaction techniques. These tools are somewhat automated, as they rely on the computer to perform calculations rather than using the medical sculptor's specialized anatomical knowledge. Additionally, users operate only within 2D windows that lack immersive 3D user interaction. These 2D interactions are difficult to bring into a virtual environ-



ment (Mine, 1996). If anatomical training involves sculpting with clay and wax, as at UIC, the virtual environment demands interaction with natural techniques rather than flat 2D windows.

Using a commercial haptics modeling package, Bibb et al. sculpted cranial implants in a streamlined process that reduced total time from four days to three hours (Bibb et al., 2002). They used SensAble's commercial FreeForm modeling tool for the sculpting process. Digitally designing the cranial implant can provide a quicker and cheaper process.

In addition to the design, the surgical process may also benefit from pre-surgical planning. Keeve et al. (Keeve et al., 1996) used finite element method techniques to computationally simulate tissue deformations during maxillo-facial surgery. These calculations did not preserve real-time interactivity, however. INPRES (Salb et al., 2002) simulated cut trajectories to determine the optimal surgical solutions for selected craniofacial patient data sets. Integrating a HMD and camera-registered views of the surgical environment, INPRES investigated risk reduction through preoperative planning within a virtual environment.

The cranial implant design process pioneered at UIC serves as a guide for incorporating virtual and augmented reality enhancements. The inherent spatial and tactile nature of medical modeling requires strong cues to both depth and touch, and meeting these needs within a virtual environment requires specific 3D interaction tools. Defining these needs, examining how different technologies meet them, this research demonstrates a way to integrate them.

### **2.2.2 3D User Interaction**

Moving beyond a 2D display means that both rendering and user interaction must occur in a 3D workspace. Conducting user interaction within this 3D workspace presents several challenges.

A key aspect of 3D modeling is the use of 3D input during the process. (Nishino et al., 1998) provided users with two tracked gloves, and these gloves specified parametric objects displayed on a wall-sized projection screen. Using spatial and pictographic gestures, users defined parameters for quadratic surfaces.

Another issue is bimanual interaction. Real-world interaction often involves using both hands. The CAD application 3-draw (Sachs et al., 1991) provided bimanual interaction for specifying, deforming, and refining geometric shapes. Following a sequential series of steps, users specified geometry that was fit to splines and curves to form surfaces.

Finally, the lack of physical constraints on user input movements prevents a mechanism for stopping actions that may move beyond their bounds (Hinkley et al., 1994). The introduction of force feedback addresses this weakness. Although commercial products such as SensAble's FreeForm have been used for medical modeling (Kling-Petersen et al., 2000), these tools are closed to custom development. Researchers cannot add targeted features supporting tele-immersion to the existing software.

These dynamic input methods utilize 3D input for creating 3D objects. The addition of force feedback provides physical interaction that assists with the spatial input. Given the fact that medical sculptors work with real objects during the traditional sculpting process, the task of providing virtual tools requires 3D interaction techniques that support existing sensory experiences with sight and touch.

### **2.2.3 Immersive Modeling**

Modeling within an immersive environment presents unique challenges. Programming tools often provide easy creation of simple geometric shapes such as boxes, spheres, cylinders, and lines. How-

ever, the issues surrounding 3D input complicate interaction, as modeling techniques designed for 2D windowed interaction do not always carry over to an immersive environment.

Several techniques have explored creating geometry within immersive environments. HoloSketch (Deering, 1995), one of the first immersive modeling applications, presented a non-immersive stereo display to the user. The user sculpted geometry using basic 3D primitives. The head-mounted display modeler 3DM (Butterworth et al., 1992) followed CAD techniques like requiring specific axes and providing extensive 2D menus and windows. It is also important to load existing models into an environment. VIRUSES (Szymanski, 1995) provided an immersive environment for loading and manipulating model files within the CAVE. More recent work has examined free drawing through the 3D workspace. A system described by Schkolne et al. (Schkolne et al., 2002) created 3D models using gesture movements on an immersive workbench system. This last approach comes closer to the modeling approach most relevant cranial implant design.

Medical modelers at UIC sculpt using clay rather than sophisticated computer modeling packages, so any modeling techniques should follow similar interaction paradigms. With volume modeling, users sculpt by filling in space as they move through the environment. Recent experiments at Fraunhofer (Deisinger et al., 2000) found that surveyed experienced and inexperienced VR users preferred free-form volume modeling to geometric modeling.

Volume modeling builds upon implicit modeling techniques to create 3D objects defined as volumes rather than only surfaces. Blinn introduced a generalized method for algebraic surface drawing, also referred to as “blobby” objects (Blinn, 1982). By combining spheres and cylinders, he represented surfaces defined by implicit functions resulting from a combination of the objects. The surfaces were

defined by all points fulfilling the implicit function  $F(x, y, z) = 0$ . Among the implicit modeling techniques described by Bloomenthal (Bloomenthal, 1989), shape skeletons can define primitives resulting in blended surfaces. Using a series of base shapes, implicit surfaces are offset from those shapes by user-defined distance values. Galyean and Hughes incorporated interactivity with implicit functions to provide volume sculpting (Galyean and Hughes, 1991) by representing the results within volume data structures rather than surfaces. They defined volumes by a characteristic function representing clay, with a value of 1.0 representing the presence of clay and a value of 0.0 representing elsewhere. A surfacing algorithm presented the clay volume contour for rendering. Solid modeling interfaces may be extended for blends, sweeps, and revolutions as described by Grimm et al. (Grimm et al., 1995).

Due to the performance requirements, volume modeling poses extra challenges to immersive environments. Optimized data structures streamline calculations and maximize storage. Ferley, et al. utilized binary search trees to create dynamic sculpting tools (Ferley et al., 1999). An overview of other volume modeling representations is provided by Nielson (Nielson, 2000). Volumetric sculpting requires optimized algorithms to preserve interactivity. Ayasse and Mueller created voxel-based modeling tools by sacrificing precision for interactivity (Ayasse and Mueller, 2001). Other approaches are described in the literature (Ferley et al., 1999) (Wong et al., 2000) (Pyo and Shin, 2002) (Matsumiya et al., 2000).

Volumetric sculpting presents the user interactions most analogous to the medical sculptors' training. CAD techniques require more detailed knowledge of underlying mathematical primitives, whereas the direct manipulation presented by volume sculpting is similar to real-life solid modeling with materials such as clay.

#### **2.2.4 Tele-Immersion**

The different participants shown in Figure 1 might be located at different facilities. Networked collaboration might enhance their ability to communication while investigating a cranial implant design. (Wood et al., 1997) stated the need for multi-user collaboration as enhancing scientific visualization analysis. Similar benefits might hold advantages for different personnel consulting among each other during the cranial implant design process. A comprehensive review of networked virtual environments is available from Singhal and Zyda (Singhal and Zyda, 1999).

Tele-immersion results from connecting multiple immersed participants, although the diversity of visualization environments allows for non-immersed collaborators. Shastra (Anupam and Bajaj, 1994) was an early multimedia collaborative design system. Although Shastra did not provide immersion, it did utilize a layered architecture to mediate changes among participants. Architecture components handled transmission of audio, images, video, and geometry data. Due to the requirements of immersive displays, tele-immersion in EVL has utilized a client-server approach. With this approach, users interact through multiple clients connected to a central server. Blackboard (van Liere et al., 1998) stored simulation and visualization information in a central database, associating named variables with attributes and data. In EVL, CAVERNsoft (Park et al., 2000) provides services uniquely tailored for tele-immersive applications. It contains clients and servers for common network protocols along with enhanced modules providing tools for tele-immersion. Built upon CAVERNsoft, the Tele-Immersive Data Explorer (TIDE) (Sawant et al., 2000) was designed to address the separation of data access and rendering responsibilities for the real-time querying of publicly available data servers. TIDE relied on a central data

server to broker communication among the immersive environment, the remote data servers, and other participants.

The nanoManipulator (Hudson et al., 2003) used the Model-View-Controller paradigm to manage event-based triggers that resulted in data changes. This non-blocking communication allowed for a separation between the interactivity and the network latency. Even with latency and jitter complications resolved, the nature of tele-immersion still poses human problems. Mortensen et al. found collaboration involving the manipulation of shared objects difficult to implement (Mortensen et al., 2002). Rather than the network performance, reviewers indicated that the graphics and lack of tactile feedback were larger barriers to completing the tasks.

Within the larger scope of the medicine, medical laboratories (Rugelj and Svirgelj, 1997) will continue to build upon the needs for processing, transmitting, and archiving medical information. The cranial implant design process involves several participants, each with a different set of skills essential to the overall procedure. Providing a tele-immersive medium for pre-surgical planning, design, and evaluation is highly desirable.

### **2.2.5 Augmented Reality**

Visual information alone does not provide enough sufficient information for medical sculptors designing cranial implants. They are trained using their hands, so they must be able to interact directly with their hands. The sculptors also rely on the feel of surfaces, and touch feedback is an essential cue for their work. A virtual cranial implant design tool should combine the proprioceptive information sculptors expect by using their arms and hands, and haptic feedback should simulate the presence of virtual objects that do not materially exist. The virtual environment must be “augmented” to integrate

real interaction and haptic feedback. As defined by Azuma (Azuma, 1997), augmented reality systems display real and virtual information, allow real-time interactivity, and manage 3D registration.

Previous AR implementations exhibit different features and consequences of their designs. Plesniak and Pappu presented haptics coincident with a holographic display (Plesniak and Pappu, 1998). Their holographic system ran at a low framerate, and there was no haptic cue to object interpenetration. Interestingly, a goal of their system was to provide AR tools to a skilled craftsperson, similar to the notions presented in this thesis of presenting virtual tools to a skilled medical sculptor. Integrating real and virtual imagery, Billingham and Kato created a two-user augmented puzzle-solving environment (Billinghurst and Kato, 2002) using video projected with HMDs. With a medical focus on volume visualization, MediDesk (Wohlfahrter and Encarnação, 2000) tracked a plastic pad for stylus interaction with the Visible Human dataset.

Haptics provides an additional sensory feedback contributor to the AR experience. By collocating graphics and haptics, Bouguila et al. noted a relationship between perceived depth and physical disparity (Bouguila et al., 2000). Adjusting the haptic feedback to correspond with the graphics display significantly improved perception. Fischer and Vance incorporated a PHANToM with the C6 CAVE-like display (Fisher and Vance, 2002). They scaled the PHANToM to operate within the extent of the C6 workspace, but the PHANToM and its stand occluded the projection displays. Reinig and Eskin (Reinig and Eskin, 2002) describe the requirements for accurate collocation of haptics and graphics, but they restrict the user's head to a fixed position.

By combining visual feedback with real-world information, AR systems provide extra information not available on other VR implementations. Registered interaction collocates the graphics with real-

world objects, enhancing interactivity. The addition to touch provides feedback to an additional element of the human sensory system. These improvements are all particularly advantageous to a system that aims to provide medical sculptors with the best available environment for cranial implant design.



## CHAPTER 3

### APPROACH

The proposed solution involves enhancing virtual reality so that the modeler may work efficiently. A user must see his hands, and a new prototype display systems fills this need. Interface tools have been designed considering the 1996 UIC process. The description of this system reveals areas for future development. Real environments are difficult to simulate (Hollerbach, 2000), so this research focuses strictly on presurgical design rather than surgery simulation.

Some existing approaches to cranial implant design were discussion in 2.2.1. However, a key motivation for CrEdit is the use of immersive projection technologies combined with force feedback. Force feedback provides improved situation awareness, and it serves as a better mode for manipulation tasks (Taylor, 1999). The unique capabilities required for haptics dictates custom software development (Potts, 2000), so the usage scenario is an important consideration.

Examining this scenario also requires identifying hardware choices and software capabilities. Esposito defined several essential questions for virtual reality user interface (Esposito, 1996). One question involved how any virtual body in the environment compares with user, and how that body moves. CrEdit relies on augmented reality, projecting the virtual environment over the real world. As a result, virtual representations align very well with the modeler's arms. The magnetic tracking system records movements and orientations; that information is also available from the robot. The virtual tools are controlled through direct interaction. Due to the tactile nature of cranial implant design, CrEdit addresses Espos-

ito's question of output modalities by generating force feedback as well as graphics. Collocating the 3D graphics with the haptic representation provides an illusion of touching virtual objects.

### 3.1 Usage Scenario



Figure 5: A medical modeler touches a virtual skull using PARIS. The stylus corresponds to a tool held in the right hand, while a virtual hand tracks the position of the left hand. A real model remains visible on the table.

Sitting at PARIS as shown in Figure 5, the modeler interacts with the system in a manner similar to the methods pioneered at UIC in 1996. Any patient or family photographs are scanned and saved to the computer. The modeler then converts the patient's skull CT data into polygon geometry. The application

loads all models and photographs into default positions. After adjusting positions as needed, the modeler may manipulate the skull data to obtain the best view of the defect. A pencil tool creates 3D annotations and highlights the edge of the defect. This defect outline serves as input for calculating a defect model separate from the entire skull. The user traces the defect edge interactively, so even irregularly shaped defect geometry may be extracted. Viewing the defect, the sculptor makes annotations indicating where to attach bolts during the surgery. Feeling the surface of the defect allows the modeler to determine how to sculpt the patient's implant. Referring to the digitized photographs, the sculptor builds material into the hole, slowly shaping the implant. When the work is completed, the application saves the model file to disk for evaluation and eventual fabrication.

### **3.2 Benefits of Virtual Reality**

With the current UIC traditional modeling approach, several steps remain labor-intensive and expensive. Although trained specialists create the implants, the process remains heavily dependent on subjective skills and procedures. Stereolithography is a relatively slow process, and the price increases directly with the volume and overall dimensions of the model being built. For example, an entire skull takes 16 hours and costs approximately \$1700 in time and materials. A separation of responsibilities exists between the modeler designing the implant and the doctor performing surgery. Assembling these specialists for consultation and evaluation can be difficult.

Medical sculptors work directly with their materials. Any translation of these skills into a virtual environment requires as much similarity as possible. Claymore (Mitsunobu et al., 1998) demonstrated direction manipulation of three-dimensional objects augmented with a sense of force feedback. This

sense of touch enhanced the feeling of working with a virtual object by providing the sensation of an actual object.

Using a commercial haptics modeling package, Richard Bibb sculpted cranial implants in a streamlined process that reduced total time from four days to three hours (Bibb et al., 2002). Digitally designing the cranial implant can provide a quicker and cheaper process.

Immersive technology provides three additional advantages over standard desktop-based user interaction. First, the viewer-centered perspective provided by head tracking presents a 3D view based on the user's current head position and orientation. Minor head movements easily reveal 3D cues to depth based on motion parallax. Second, a properly configured display system maintains size constancy. Displayed objects exhibit consistent scale with a 1:1 ratio to "real world" measurements. Third, the co-located graphics and haptics representations avoid a visual discrepancy. Typical haptics usage requires separating the graphics display and haptics feedback.

### **3.3 Materials**

This research relied upon two very specific materials. First, a see-through augmented reality display provides a large visible workspace. Second, a desktop-sized stylus rendered force feedback to simulate a sense of touch. These materials provide specific features essential to designing cranial implants within a virtual environment. This section describes the PARIS display, the PHANToM haptic device, and the configuration process required to integrate the two.

#### **3.3.1 PARIS**

One of the first VR systems designed within VR, the Personal Augmented Reality Immersive System (PARIS) incorporates significant improvements over previous projection-based virtual reality displays

(Johnson et al., 2000). Previous systems such as the CAVE and the ImmersaDesk support 3D vision well, managing separate stereo images for each eye and tracking head motion. There remain at least two important depth perception cues, occlusion and accommodation, that are not supported correctly in previous displays (Bouguila et al., 2000). For instance, in a conventional projection-based virtual reality display, an object in front of the hand is obstructed by the hand itself. This occlusion causes a visual conflict because the hand, which should be behind the object, appears in front of the object. The object should be visible in front of the hand. The half-silvered mirror in the PARIS display superimposes the displayed image over the hands. One does not occlude the other. In contrast to the PARIS mirror, Wloka and Anderson explored a video-based approach for resolving occlusion (Wloka and Anderson, 1995). The second depth cue, accommodation, refers to muscles controlling the eye to adjust sharpness. In a conventional VR display the eye will always focus on the display screen, which is typically significantly farther than arm's reach. The PARIS display is designed so that the hands and the virtual object are the same distance as the image of the screen. Unique to the PARIS, these projection improvements are particularly important for sculpting. The modeler's hands, the constructed implant, and the patient's data are all visible in the same working volume.

As shown in Figure 6, PARIS provides a large workspace that almost completely fills the user's field of vision. Space under the mirror is available for additional interaction. CrEdit uses a haptic device placed under the PARIS mirror to generate a sense of touch when interacting with the environment.

### **3.3.2 PHANToM**

The PHANToM (Salisbury and Srinivasan, 1997) is a commercially available haptic force feedback robot available from SensAble Technologies. It measures both position and orientation, thus providing

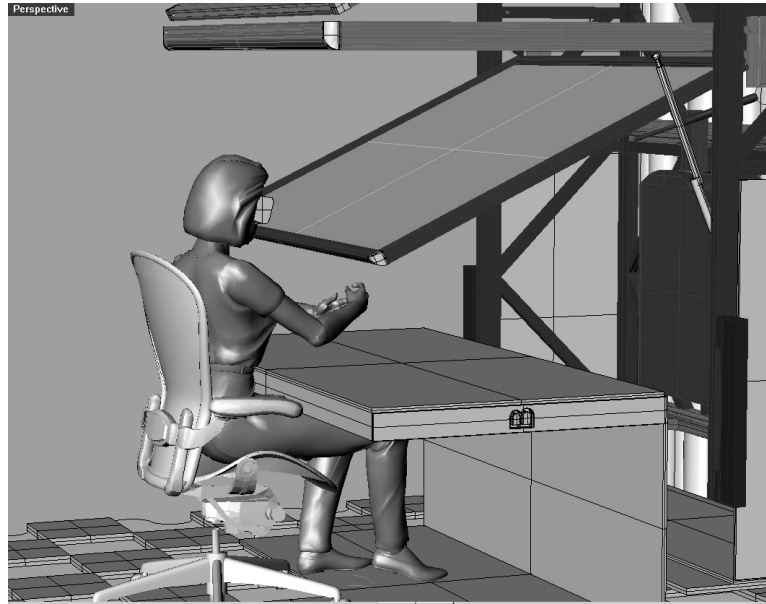


Figure 6: PARIS configuration.

six degrees of freedom. It generates only position-based force feedback, although other models can generate torque forces. As shown the Figure 7, the PHANToM's robotic arm sits atop the desk surface. The typical configuration places the device adjacent to a monitor. CrEdit locates the PHANToM under the PARIS screen, thus making placing it directly in the visual workspace.

### 3.3.3 Device Configuration

Configuring PARIS involves aligning the coordinate systems, thus enabling graphics and haptic software packages to maintain collocated model representations. Screen projection corner measurements establish the location of the window into the virtual environment. A world origin is chosen, and all measurements must be taken relative to that origin position.



Figure 7: SensAble's PHANTOM Desktop is a commercially available haptic robot with force feedback capabilities. Image courtesy SensAble Technologies.

Adding haptics to an augmented reality environment creates problems because these multiple coordinate systems must be measured and aligned (Vallino and Brown, 1999). Once aligned, the union of graphics and haptic representations presents a very compelling illusion to the user. Burdea noted the limitation of haptic devices' available workspaces (Burdea, 2000). PARIS presents a large workspace to the user, but the haptic workspace is limited only by the haptic device's available range. The PHANTOM Desktop moves through approximately 25% of the visible work area. Magnetic tracking provides additional work space throughout the remaining visible area.

### 3.4 Software Libraries

As defined by (Preston and Lever, 1996), scene graphs provide an object-oriented way to represent scene components as individual objects. Each node within a scene graph encapsulates state information. This state may describe properties such as lighting, coordinates, transformations, and material colors. The structural organization provides the relationships between different nodes within the scene graph. Scene graphs make graphics software easier to write, but there is a trade off between generalization and performance (Sowizral, 2000).

The evolution of scene graphics started with a language specifically for generating graphics with sophisticated hardware, GL (Silicon Graphics, Inc). Programmers desiring reuse and extensibility designed structured hierarchies for encapsulating graphics state information (Zelevnik et al., 1991). This scene graph approach represented graphics as programmable objects with easily modified properties. These capabilities provided a structured way to utilize objects, textures, and other graphical features. (Zelevnik et al., 1991) incorporated time and behavior as first-class notions, thus providing relationships beyond just shape descriptions and rendering characteristics. The Inventor framework (Strauss and Carey, 1992; Wernecke, 1994a) built upon these ideas and added mechanisms for interactivity, manipulation, and animation. Inventor became the basis for the original Virtual Reality and Modeling Language (VRML) format used for transmitting 3D models across the world wide web. Development of Performer (Rohlf and Helman, 1994) targeted scene graph optimization by taking advantage of multiple processors to more rapidly generate and display images. GL has evolved into OpenGL, a widely-supported graphics hardware language standard on numerous operating systems. SGI made Inventor available as open source in 2000, and Performer is available in a limited capacity as well. Ongoing scene graph



development projects, such as Coin3D, OpenSG, and OpenRM, continue to evolve with support for the latest OpenGL capabilities.

Generating usable models requires converting between the toolkits' data representations. Employing separate toolkits maximizes the strengths of each. Figure 8 illustrates the responsibilities and relationships among the three toolkits. CAVELib and GHOST provide input by sampling user positions from tracking and haptic hardware, respectively. VTK generates geometry using volume modeling methods. VTK data structures may be easily converted to Open Inventor scene graph nodes. Once VTK finishes calculations, it converts the results to an Open Inventor model. This scene graph is loaded into the main application and used for visualization and graphical interaction. The application also converts the resulting triangle geometry to GHOST's scene graph. GHOST communicates with the PHANToM hardware and generates force feedback that coincides with the environment's graphics. Figure 8 illustrates how data flows among the primary software libraries.

Coin3D serves as the central data representation, but CrEdit converts data structures among OIV, VTK, and GHOST as needed. Each library contributes a specific set of capabilities toward the overall system design.

First, the CAVE Library (Cruz-Neira et al., 1993) manages the graphics by providing the low-level connections between the operating system, tracking hardware, and projection display. The CAVELib creates multiple windows for each graphics display and synchronizes rendering into stereo buffers. Although PARIS is a single-display system, the CAVELib provides portability to multiple-display systems. In addition to managing each display, the CAVELib calculates the viewer-centered perspective projection based on data received from the tracking system.

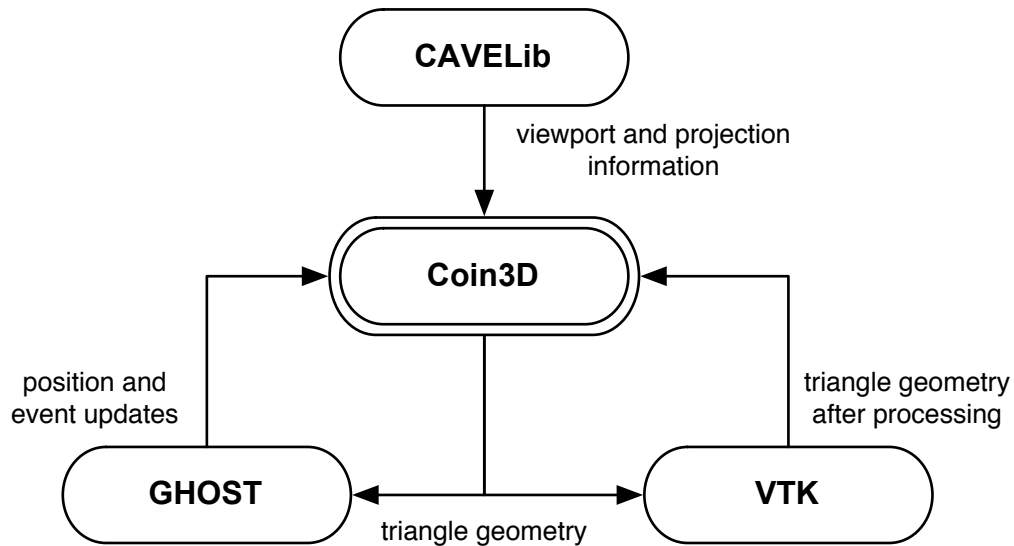


Figure 8: Data flow among software libraries

Second, Open Inventor provides a higher-level interface for programming graphics. Silicon Graphics, Inc. created the Open Inventor toolkit in 1992 (Strauss and Carey, 1992). This scene graph library included the notion of direct manipulation directly within the scene graph. Animation and interaction were included within different scene graph components, thus creating a comprehensive structure for managing nodes and their relationships with each other. OIV provides a number of common computer graphics structures including vectors and matrices. The OIV file format served as the basis for the VRML modeling specification (Bell et al., 1995). The flexible file format is human readable, and the robust file operations simplified verifying the scene graph structure during development.

SGI released OIV as open-source software in 2000 (Silicon Graphics Inc., ). OIV was also licensed by TGS, who has made numerous enhancements and provided additional capabilities to the scene graph. Simultronics in Motion released Coin3D (Simultronics in Motion, ) as an open-source library conforming to the OIV 2.1 API header files. CrEdit was originally implemented using TGS Inventor 3, but development switched to Coin3D in early 2002 due to developmental support and the ability to examine the library's source code. This flexibility was key when examining conversion from the OIV geometry to the haptics library.

PHANToM device control is available through the third essential library, the GHOST SDK (Sensible Technologies, 2001). This library handles all low-level device communication with the hardware, and it also provides a software programming interface for creating the haptic environment. A key GHOST feature is the fact that, like OIV, it stores geometry and behavior information in a scene graph.

Finally, the fourth library provides significant computational capabilities. The Visualization Toolkit (Schroeder et al., 2003) contains a comprehensive library of algorithms for converting data into visual representations. VTK does not use a scene graph, but instead relies on a modular data pipeline. Data originates in sources, and applications interconnect data visualization modules in order to provide a pipeline from the data into a resulting visualization.

Each library has a unique role to play within CrEdit's design. The integration must allow these libraries to cooperate with each other. Relying on each library's strengths allows application development to focus on the communication among the libraries. Because each library handles different types of data, an early step in the design process was identifying how information would be passed among the libraries.

### 3.5 Identifying Library Responsibilities

Implementing communication among the libraries required categorizing data behavior. Each library has its own approach for combining separate data sets into a single result. In order to provide data to each library within its own format, CrEdit must convert among the various representations. The CAVELib is very light-weight, as it only manages windows and OpenGL contexts. Coin3D and GHOST have scene graphs, while VTK organizes data through a series of modules connected in a visualization pipeline. Analyzing these characteristics was the essential first step before creating the software implementation.

CAVELib creates the graphics window and OpenGL context based on the current operating system. CrEdit was initially prototyped on SGI IRIX, but primary development moved to Microsoft Windows 2000 once the PHANToM become available for introducing haptics. Additional development occurred using MacOS X. The CAVELib created the platform specific graphics window using the X Window System, Microsoft Foundation Classes, and Cocoa Framework. Beyond window management, the CAVELib also calculates and maintains the camera projection matrix based on magnetic tracking information. The CAVELib thus manages the following responsibilities:

- Create all operating system windows
- Create and maintain the OpenGL graphics context
- Maintain the OpenGL projection matrix
- Maintain the OpenGL model matrix
- Report head and hand tracker positions and orientations

While the CAVELib handles all projection and viewport information, OIV manages the graphics rendering. OIV also serves as the central data representation, as its scene graph provides an ordered structure that may be easily read to and from the hard disk. As the central data representation, OIV's duties are essential:

- Represent all data as scene graph geometry
- Load scene data to and from the disk
- Render the scene graph using the CAVELib projection matrices
- Unify the various coordinate systems visually
- Update to reflect to user interactions

GHOST controls the PHANToM, and in doing so creates its own thread for the haptic servo loop. This thread maintains a 1000 Hz update frequency to provide consistent haptic rendering. GHOST's responsibilities may be summarized as follows:

- Mediate low-level PHANToM hardware communication
- Manage haptic control thread
- Represent geometry from the haptic scene graph
- Report PHANToM position, orientation, and stylus button

VTK's primary responsibility is applying implicit functions to interactively created geometry. Making the OIV scene graph geometry available to the VTK pipeline requires a conversion process. This

conversion makes available the many visualization algorithms present within VTK, and it also lets VTK apply operations directly within its own pipeline.

- Convert OIV geometry for processing within the VTK pipeline
- Apply implicit functions to geometry
- Convert the VTK pipeline into OIV geometry after processing

Defining these libraries' responsibilities was the essential first step before designing CrEdit's software architecture. Because each library has a different set of capabilities, it was important to identify how each library would interact with the data.

## CHAPTER 4

### METHODOLOGY

The software techniques used in this research provide methods for integrating the different software components. This section describes the methodology integrating the different software libraries into the cranial implant application.

#### **4.1 Software Integration**

Several software components integrate the CAVE, OIV, GHOST, and VTK libraries. The libraries' different strengths cater to different responsibilities required by the application. Because each library has its own data representation, producing conversions among them requires examining how best to share information. Each library fills a unique role within CrEdit's overall design, and those roles must interplay with each other in order to provide the best combination of the capabilities.

As the library with the most comprehensive collection of classes and data structures, OIV provides the core representation. OIV provides classes for scene processing, dynamic hierarchy traversals, geometric primitives, user interaction, and event handling. None of the other libraries can offer such a comprehensive array of capabilities, but CAVELib, GHOST, and VTK do provide functionality unavailable in the standard OIV API. As CrEdit moves data among the CAVELib, GHOST, and VTK libraries, it relies on OIV to serve as the unifying library. Therefore, most of the additional classes that have been designed communicate only with OIV. The OIV scene graph is the central data repository.

#### 4.1.1 OIVAppShell

Open Inventor renders graphics using view information obtained from an SoCamera node placed within a scene. This node sets up the camera's location and frustum, but it is not well suited for use in CAVE-like immersive environments. The CAVE contains multiple screens that require their own projection parameters. OIV encapsulates scene graph traversals into actions. A SoGLRenderAction traverses the scene graph for OpenGL rendering. This action class assumes that certain structures are in place within the scene graph, and OIV provides a meta-class to manage a scene for rendering.

The OIV SoSceneManager class wholly manages a single scene graph. However, SoSceneManager relies on a single SoCamera node to generate the perspective projection. As an application applies its SoRenderAction, the SoCamera node provides the appropriate OpenGL viewport functions. The OIV scene graph may only contain a single active camera node. This restriction complicates rendering the same scene graph to multiple displays, since each display requires its own camera viewing the same scene graph. Because the CAVELib calculates the OpenGL matrices itself, the OIVAppShell class sets up that information in the SoGLRenderAction state. This implementation functions identically to the SoCamera node by setting up the appropriate state information. The TGS Open Inventor manual (TGS, Inc., 2002) demonstrated the general requirements for setting up state information. The key steps may be summarized as follows:

- Use CAVEGetViewport() to fetch the viewport parameters.
- Initialize the SoGLRenderAction with the results.
- Use CAVEGetPosition() for the eye position.



- Use `CAVEGetVector()` for the eye viewing direction.
- Use `CAVEGetProjection()` for the projection matrix.
- Use `glGetFloatv()` to fetch the current `GL_MODELVIEW_MATRIX`
- Use `glGetFloatv()` to fetch the current `GL_PROJECTION_MATRIX`
- Set the corresponding state variables.

As far as rendering is concerned, these steps complete the link between the CAVELib matrices and the OIV render traversal. The CAVELib provides the dynamically updating viewport and projection matrices, and OIV uses those parameters for rendering the structured scene graph. This approach takes advantage of OIV's ease of programming while maximizing the CAVELib's projection management.

#### **4.1.2 OIV-GHOST Integration**

Open Inventor and GHOST both store data in scene graphs containing varying node types, and making the two libraries interact demands careful management. Open Inventor includes textures, shapes, transformations, manipulators, and cameras. Although Open Inventor allows lateral state inheritance during scene traversal, GHOST only allows top-down inheritance. Open Inventor also allows the same node to appear multiple times in a scene graph. GHOST prohibits this repetition. As described by (Wang, 1999), conversion between OIV and GHOST requires maintaining separate scene graphs.

These structural differences are most important for transformations. Open Inventor stores transformations as matrices within nodes. Multiple `SoTransform` nodes result in accumulated transformation matrices. GHOST's `gstSeparator` nodes contain a single transform matrix. Storing multiple transform

matrices requires chaining `gstSeparators`. Software must manage changes in either scene graph and reflect those changes in the other representations.

#### **4.1.3 coinevents Library**

Coin3D implements most of the original OIV API. However, OIV targets 2D mouse-based user interaction within a window. It does not handle 3D events containing position and orientation information. The CAVE's tracking devices report 3D positions and orientations. GHOST reports the PHANToM's 3D position and orientation as well. Exploiting the similar representations allows programming for a single type of user interface event regardless of the originating device.

The Inventor Toolmaker (Wernecke, 1994b) details OIV's object-based event paradigm. Rather than functionally polling devices and responding to events, an OIV application generates events and passes them into the scene graph. Event-handling via `SoHandleEventAction` thus becomes another type of traversal similar to OpenGL rendering. The idea is that nodes within the scene graph respond to events rather than application specific source code.

Event generation responsibility falls upon the application. Typically OIV GUI widgets capture these events based on operating and window systems. The CAVELib creates `CrEdit`'s windows, so the application must manually generate events. Figure 9 illustrates how `CrEdit` generates events for passing throughout the rest of the application. `SbTrackerInfo` encapsulates tracker position and orientation information. To fit within the event paradigm, the coinevents library provides the `SoTrackerEvent` class. The library stores button press and release events using `SoControllerButtonEvent` objects.

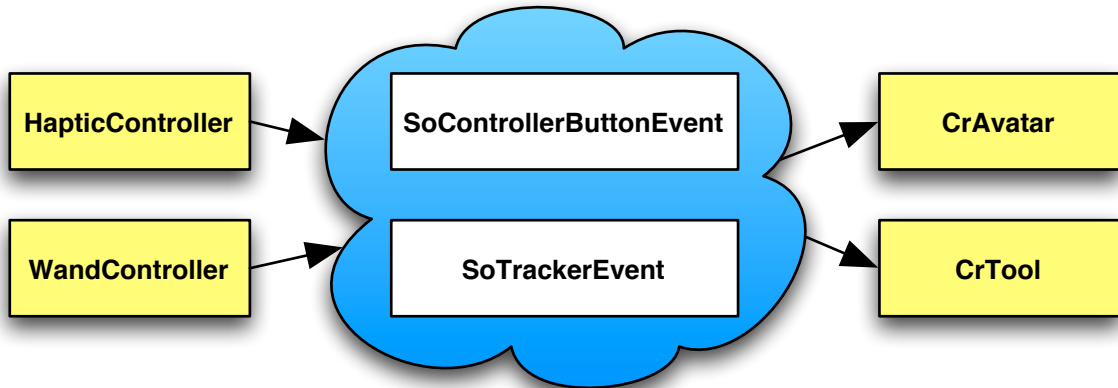


Figure 9: The controller classes generate custom Coin3D user interaction events for uniform handling by other classes.

#### 4.1.4 vtkoiv Library

The Visualization Toolkit executes a pipeline that performs a series of filters on data sources. VTK refers to its visualization result objects as actors. As VTK does not provide a scene graph, Open Inventor integration requires implementing collaboration classes that move data between the two libraries. Two classes, `vtkOIVSource` and `vtkActorToIV`, serve as the bridges between the VTK pipeline and the Open Inventor scene graph. Conversion from VTK to Open Inventor is based on Paul Rajlich's VTK to Performer (Rajlich, ) implementation. An application assembles VTK modules into a pipeline. The connections between these modules define the visualization parameters and behaviors. Once executed, the pipeline generates visual results into a `vtkActor` object. This object contains the visualization geometry and materials.

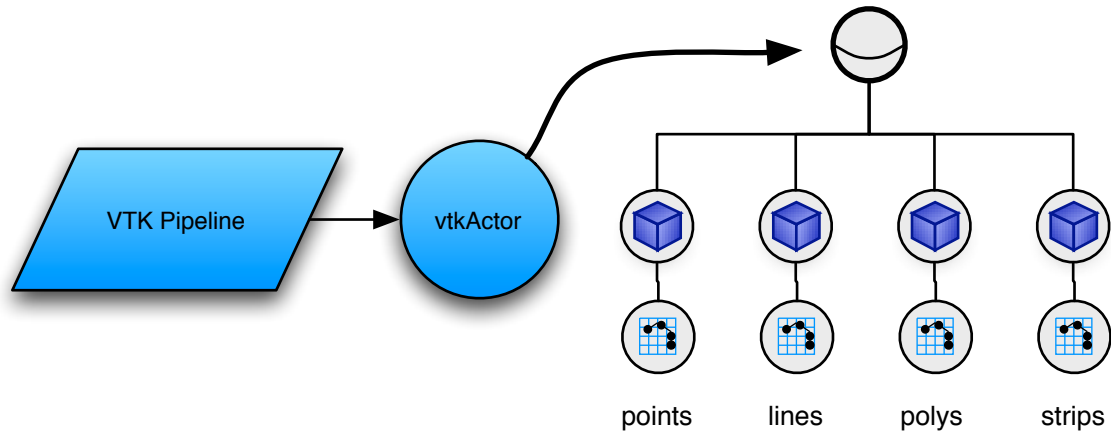


Figure 10: `vtkActorToIV` converts a `vtkActor` into an OIV `SoShapeKit` with geometry duplicating the actor's points, lines, polygons, and triangle strips.

The first class, `vtkOIVSource`, converts Open Inventor geometry to a data source at the beginning of VTK's pipeline. This conversion process makes any geometry contained in the OIV scene graph available as input for VTK operations. The defect and implant tools described in sections 4.3.2 and 4.3.3 rely heavily on this capability, as it leverages OIV's interactivity with VTK's geometry processing. `CrEdit` plugs vertex-based Open Inventor geometry directly into the VTK pipeline. When VTK executes the pipeline, it assembles the visual representation of the result into a `vtkActor`. The actor object encompasses all geometry and rendering options as configured in the pipeline.

Converting VTK's visualization geometry back to the OIV scene graph falls as a responsibility to the second class, `vtkActorToIV`. `vtkActorToIV` processes the resulting `vtkActor` to extract its geometry. It converts the resulting triangles and materials into an Open Inventor scene graph. This processing makes available VTK's comprehensive algorithms while retaining Open Inventor scene graph organization.

The conversion algorithm is applicable to integration between VTK other scene graph libraries (Patrick Muller, personal communication, Dec 2003).

#### **4.1.5 QAvatar Library**

In a tele-immersive application, users require visual representations of other participants in the shared environment. Avatars should convey essential information about the identity and actions of their users. EVL avatars typically include several key pieces of information. First, avatars have unique names identifying themselves to other users. Second, tracking information allows avatars to represent where their users are within the environment. For CrEdit, this tracker information includes the following: the head position and orientation in world coordinates; the wand position and orientation; and finally, the PHANToM position and orientation. QAvatar is designed to provide several key improvements over the previous avatar implementations:

- Cleanly separates networking implementation from the data requirements.
- User subclasses are not as tightly dependent upon each other.
- Drops notion of callbacks and instead notifies individual objects.
- Very effecient server uses threads and signals to reduce CPU load.
- Easily configures across platforms using CMake.

Once the avatar data packets are determined, it is essential to examine how they are sent. The QAvatarManager's self avatar stores the data buffers, but the application must fetch the appropriate data from their sources. Consider the approach used for previous Performer avatars, in which multiple files called CAVEGetPosition() whenever it was needed. There were a lot of local values stored in disjointed

source code locations. Using the `CAVEGetPosition()` calls from anywhere in the code is not a clean design. Although this function accesses shared memory maintained by the `CAVELib`, the application treats this data in a globally accessible manner. Also, the `PHANToM` does not operate through the `CAVELib`. `CrEdit` requires a consistent approach to reporting positions to other avatars. The wand and `PHANToM` already provide tracking information via `CrController` classes. The `SoTrackerEvent` contains all the required information, and all the `CrTools` already respond to those events for doing all interaction. The avatars update by responding to the same events.

Using the event model provides a consistent flow of information. Rather than having various parts of code querying tracker information, a better approach maintains a consistent data flow. The tracker information is grabbed once, and it's automatically sent to all pertinent components of an application on a per-frame basis. The head tracker is not attached to any classes in the source code, but it can still be easily packaged into a tracker event. The `QAvatar` library integrates an event-observation model with existing avatar concepts.

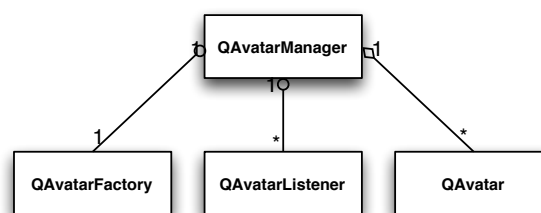


Figure 11: QAvatar library classes.

The main avatar class manages three buffers for aux, hello, and tracker data. It knows the specifics about the data that is inserted into those buffers, and subclasses must implement methods to pack and unpack any specific data to and from those buffers.

One way to remove coupling between concrete manager and avatar classes is to use the Factory Method design pattern. This pattern encapsulates instantiation within a separate object. By creating different factories, different concrete avatar classes may be created. The key advantage to this approach is that a base manager class can easily create instances of concrete avatar subclasses.

The old code triggered C callbacks when responding to network messages. I use a modification of the Observer Pattern that is a less generic. The Observer Pattern typically uses a “pull” approach in that Observers must request information from the Subject. This class “pushes” the data within specific notification methods. Listener classes respond to those messages without having to request state information from the notifier. Information may still be requested from the avatar object to extract buffer information.

This class manages the networking between multiple users. It creates the hash tables that map multiple users to local avatar objects. Messages are sent to a server by querying its own “self avatar.” This localizes all representation within avatar objects. The manager does nothing more than manage network communication. Any listeners attached to the manager are notified when new messages arrive.

The application’s QAvatarManager could be a CrControllerObserver. When the controllers are updated, their SoTrackerEvents will propagate to the QAvatarManager via CrControllerObserver::handleEvent(). However, using CrControllerObserver will not work for receiving avatar information. This class operates on a 1:1 ratio. There is only one CrController observed by a single CrControllerObserver. A

CrController is still a Subject, so the user's own CrAvatar instance can still be an Observer of the controller device. This allows the avatar to easily respond to movement events regardless of whether they originate from the wand or the PHANToM.

## **4.2 Application Design**

This section describes the developed CrEdit classes. Each application component had to provide specific methods for sharing information among the software libraries and their different data representations.

The user specifies the environment parameters into a text configuration file. The application's CrEditConfig class parses through these files and initializes the available models and images. The configuration file also provides the user name and data identification. These parameters associate the given environment options with a given set of data for a single user. This configuration allows creating separate configuration files for each user or each set of patient data.

Figure 12 contains a sample configuration file excerpt. The PATIENTID identifies the data, and the application uses its value for networking management. The file also provides a string USERID for sharing the user's name with other participants within the virtual environment. When collaborating, CrEdit connects to the remote server and port specified by the SERVER parameter. Providing these options through a text file make it easy to configure the basic parameters for the environment. An essential parameter is the PHANTOMOFFSET parameter. This setting contains the translation offset, described in 3.3.3, that provides the coordinate system transformation from the CAVELib to the PHANToM device.

Force feedback is a key feature for using PARIS, but remote users may not have the required hardware. Interaction must still be available with only traditional tracking devices. CrEdit abstracts the



```

PATIENTID VisibleHuman
USERID Chris
SERVER laurel.evl.uic.edu 7000
PENCILCOLOR 0.1 0.8 0.7

# In cradle: -0.0767584 -3.94836 0.00686898
# From floor: 0 29.75 26.25
PHANTOMOFFSET 0.0 33.63 -21.19

SKULLFILE data/VisibleHuman_skull.iv

REFIMAGE data/implantplacement.sgi
REFIMAGE data/sculpttools.sgi
REFIMAGE data/stloutput.sgi

SLICEVOLUME head-ct 268.76 268.76 1
SLICEIMAGE rsna-data/new-png/GEImages-25000.png
...

```

Figure 12: Sample CrEdit configuration file.

CAVELib and GHOST control devices. A combination of the Observer and Strategy patterns (Gamma et al., 1995) separates controllers and user interface tools. This facilitated development during instances when only one controller was available, and it is important to provide a standard definition for device configuration (Potts, 2000). CrEdit implements interaction within Tools that respond to Controllers.

Figure 13 illustrates the relationships between the Controller and Tool classes. The WandController and HapticController classes communicate with the CAVELib and GHOST toolkits, respectively. As devices' states change, Controllers notify their Observers. Every Tool is a separate Strategy, and each

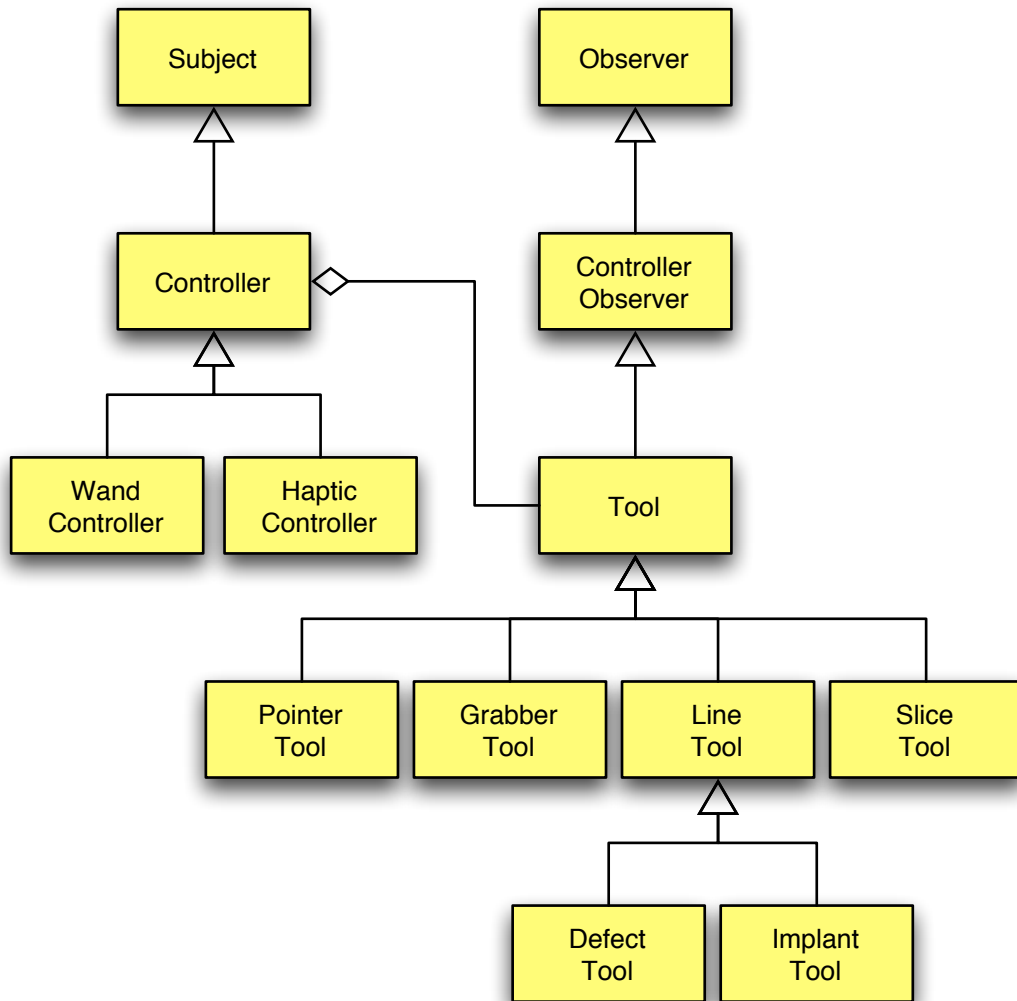


Figure 13: Controllers implement device communication for different tool implementations.

responds uniquely to Controller events. Force feedback responses may be customized within a Tool, but such behaviors only apply when observing a HapticController.

### **4.3 Capabilities**

Traditionally, the medical modeler utilizes patient and family photographs for reference. These digital photographs provide visual cues for guiding the implant design. Rather than cluttering the workspace with numerous photographs, the modeler may instead load digital images directly into the virtual environment. These images may be moved freely through the environment. Transparent images may be positioned directly in relation to the CT data. Because it is not important to feel the images as planar surfaces, these images are manipulated using the tracked pointer rather than the haptic device.

The design process requires an anatomically correct model of the patient's skull. Preprocessing converts the patient's skull CT data into Open Inventor geometry. Decimation reduces the detail so that the model will not impede interactivity within the virtual environment. This model is subsequently converted into the haptic library's scene graph for haptic rendering and tool interaction.

Due to the smaller size of the PHANToM workspace in comparison to the PARIS visual display, the Wanda provides interaction with other objects within the environment. By casting a 3D ray from the Wanda position into the environment, the grabber tool can select and move other objects. This functionality is secondary to the other tools, all of which focus more directly on the needs for cranial implant design.

#### **4.3.1 Line Drawing Tool**

The first basic tool is a 3D pencil with which the user may mark points of interest. The conversion of CT data into geometry can result in threshold errors, and the resulting geometry may not accurately

reflect the structure of the skull. The pencil tool allows users to draw and write freely, indicating those scanning errors or where to place clamps when attaching the implant to the bone during surgery. A simple erasure mode deletes the last line drawn by the user.

The line drawing algorithm works within the state encapsulation paradigm provided by OIV. CrEdit creates the line shape, but user interactions continue to refine the vertices stored as the shape's coordinate attributes. The OIV SoLineSet class represents a set of 3D line strips. The line drawing tool tracks the current line number as well as the current vertex number within that line. By representing the drawing within the scene graph, the lines become a persistent state easily accessible in the same manner as any other geometry.

When executing the VTK pipeline, it's important for lines to have the same coordinate reference as the skull geometry. This alignment simplifies setting up calculations involving the lines and skull. A SoGetMatrixAction calculates the transformation matrix to convert from the PHANToM's coordinates into the skull's coordinates. This transformation is applied to each vertex as the user draws lines. The entire scene graph under the PHANToM offset is already affected by a transform. Therefore, the action must be applied to the root of the sub-graph rather than the root of the entire scene.

With the correct values for the points on the line, CrEdit also integrates the lines into two key steps of the implant design process. These lines are used for both defect and implant generation.

### **4.3.2 Defect Specification Tool**

As described in 2.1.3, defect specification is another important part of the planning process. The modeler must indicate the boundary of the cranial defect. Using the haptic feedback as a guide, the modeler can trace along that edge of the bone as depicted in Figure 14. The modeler specifies the defect

by pressing the PHANToM stylus button to outline the edge. Semi-transparent spheres indicate the volume included as part of the defect. Portions of the skull within the blue spheres will be extracted. This process calculates the resulting data based on where the spheres were drawn. Depending on the complexity of the defect, this process could take only a few minutes. Contrasted with the stereolithography fabrication, extracting the defect in the virtual environment takes less time. The modeler may then begin designing the implant.

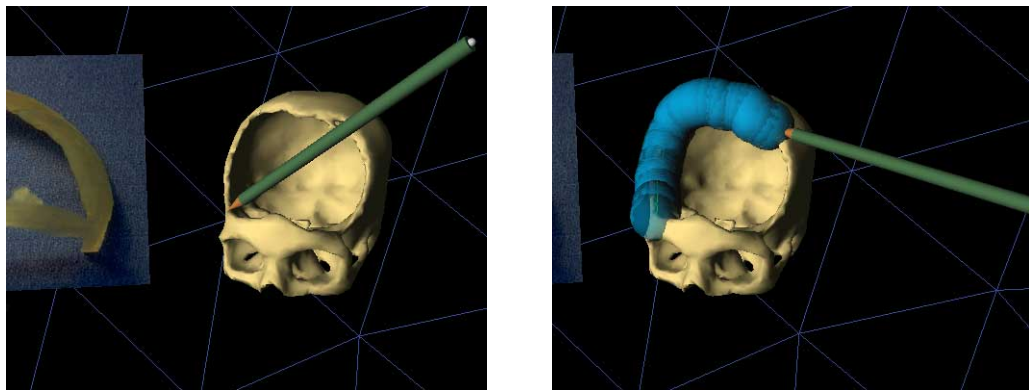


Figure 14: The modeler specifies the defect by interactively outlining its edges..

#### Defect Specification Steps

- Interactively indicate the edge of the defect.
- Select any additional surrounding areas.
- Apply the boolean geometry operation.

A key feature of the application design goals was the ability to interactively select the defect area with a cutting operation. It turns out that VTK does not implement implicit functions with polygon data. In order to create the behavior desired, the ideal solution involves creating a VTK method for converting any geometry into an implicit function. That is a gargantuan task requiring extensive mathematical details. For the cutting operation, there is a feasible workaround.

I have spent a lot of time trying to figure out a way to add implicit distance functions to VTK. It's really not trivial to do so. I have gathered quite a bit of documentation about the process, but trying to get things to work was proving to be quite the complex and time-consuming task. I decided to instead stick with the VTK implicit primitives.

My first approach involved trying to connect the points of the line with cylinders. One caveat is that `vtkCylinder` creates infinite cylinders. There are no caps. The documentation states that if caps are needed then `vtkImplicitBoolean` can be used. That makes the creation a little more complex. Each line segment must then be evaluated against a cylinder, two planes, and two spheres. To make matters worse, the transformation matrix which can be applied with an implicit function operations on the points, not the function itself. This means that I have to determine the transformation matrix to get the correct transform, and then I have to calculate its inverse to apply to the points!

In process of assembling the code for the above approach, I noticed that the radii for the spheres were almost always much larger than the distance between any two points in a drawn line. Rather than going through the trouble of positioning planes and cylinders, I could get the desired effect by using only spheres. The surface results were similar to the desired defect area.

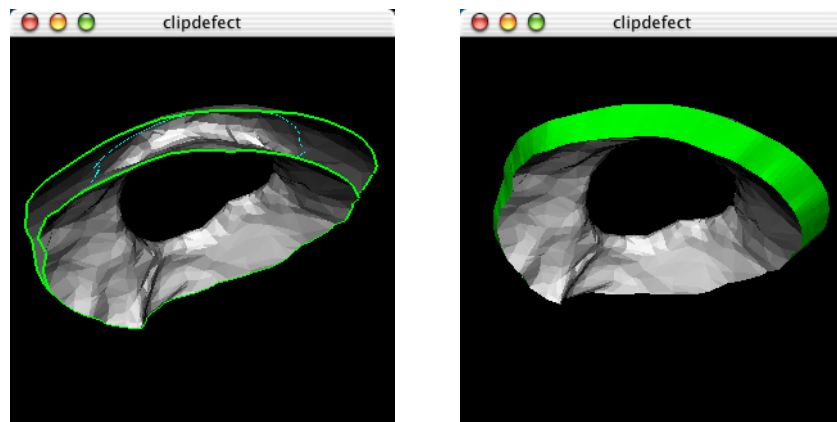


Figure 15: Defect geometry before and after filling in the edge using a `vtkRuledSurfaceFilter`.

There was still an obvious problem with the clipping operation. Because CrEdit uses a polygon-based representation of the geometry, it does not create a solid defect. These problems become most clearly evident after the clipping operation. Without the sense of an object's interior, the clipping leaves open edges as shown in Figure 15. After ordering the vertices located along the edges, CrEdit orders them using a `vtkStripper`. It subsequently uses a `vtkRuledSurfaceFilter` to connect the edges and create the appearance of a solid defect model.

With the addition of the extra steps, the pipeline must execute some separate components. This branching results in two different geometries. `vtkAppendPolyData` combines the two pipeline branches, thus creating the end result as a single geometry object.

Because the VTK-OIV conversion code is written to stand on its own, it returns its result packaged within a `SoShapeKit`. Using this collection of nodes would require minor processing to traverse the shape kit and extract the geometry. CrEdit directly retrieves lines, triangles, or polygons from the

output. This specialization allows easy extraction of the geometry that has been output from the VTK pipeline.

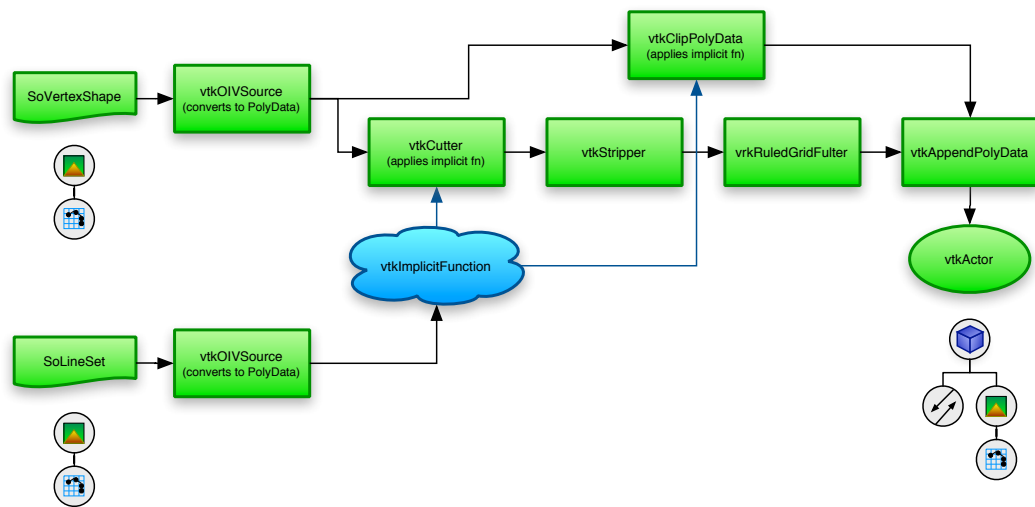


Figure 16: The VTK defect construction pipeline processes the OIV lines and skull geometry.

The VTK defect pipeline integrates OIV and VTK geometry to generate a focused area of the original model. As shown in Figure 16, the pipeline begins with two conversions from OIV geometry. First, CrEdit converts the cranium model into `vtkPolyData` using the `vtkOIVSource` class described in 4.1.4. This process converts the OIV geometry coordinates into `vtkPolyData`. Second, any lines drawn by the user with the defect tool must also be converted into `vtkPolyData`. `vtkOIVSource` handles the line



conversion as well. These conversions are the first step in the defect generation process, as they prepare the data for passing through the VTK pipeline.

The VTK pipeline starts with the two converted geometries. As shown on the left side of Figure 16, one `vtkOIVSource` contains the skull's triangle geometry, and the other `vtkOIVSource` contains the lines drawn by the modeler. A `vtkImplicitFunction` creates collective spheres along each point within the lines. These spheres are used for constructive solid geometry, basically defining boundaries for performing addition and subtraction on the geometry. After VTK finishes processing the pipeline, then CrEdit converts the resulting `vtkActor` into OIV geometry containing the defect.

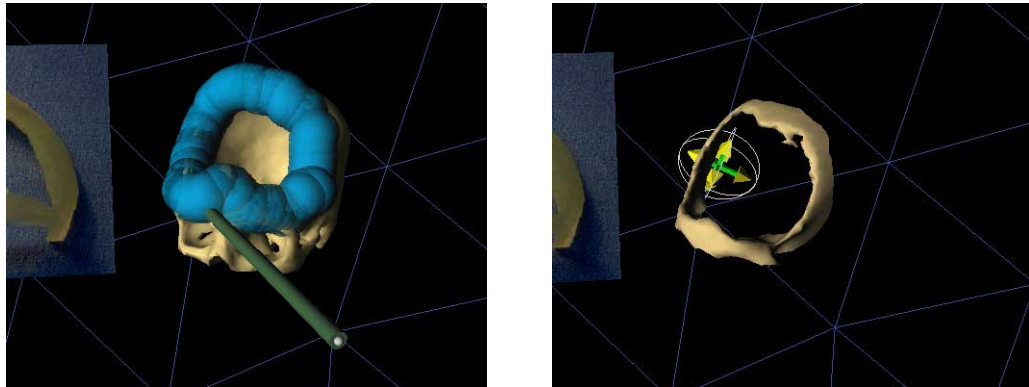


Figure 17: The defect after interactive specification.

The resulting `vtkActor`, once loaded into the environment as shown in Figure 17, closely resembles the starting defect model shown in Figure 3.

### 4.3.3 Implant Generation Tool

Translating the sculptors' dexterous work with clay into the virtual environment is a challenging task. Following previous work demonstrating users' preferences for volumetric sculpting (Deisinger et al., 2000), CrEdit implements implant generation through the use of implicit modeling. This approach is analogous to squirting a tube of toothpaste. For this proof of concept implementation, the implant generation tool generates polygonal models representing volumetric sculpting techniques.

CrEdit's implant generation tool builds upon the both the line drawing and defect specification tools. The tool maintains its own line as the modeler draws, indicating the area that must be filled in with the implant. The tool's representation is similar to the other line tools, so the consistency reinforces the fact that interaction is the same as well. As shown in Figure 18, the white-color implant generation tool shows an approximate of the implant volume while the user draws in the hole.

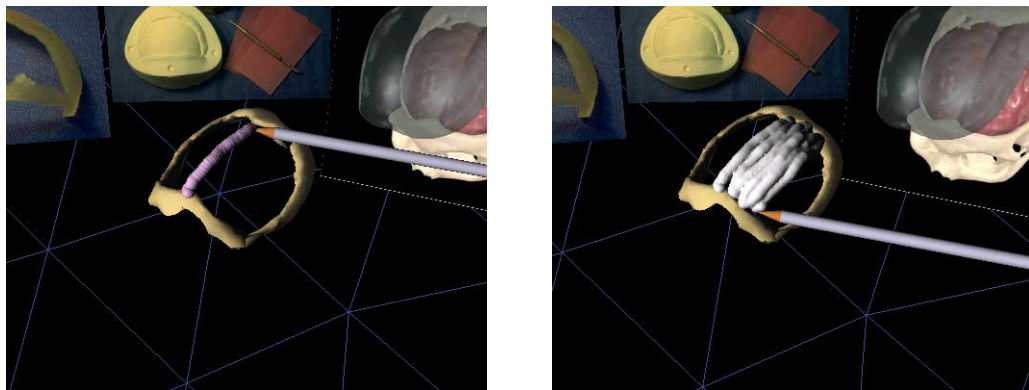


Figure 18: After the modeler draws the implant, CrEdit fills in the geometry.

In order to create the implant geometry, the tool requires lines serving as a skeletal frame for the volumetric representation. The tool creates a 3D voxel grid within a static boundary volume. This grid is a 3D lattice structure, and each point in the lattice serves as an implicit function evaluation point. Each point within the lattice represents a summation of the distance from the closest line. Below the determined distance radius, the point is considered “inside” the volume; outside the threshold, the point is considered “outside” the volume. Through the VTK pipeline, CrEdit provides the user with tools for constructing the skeletal frame for the volumetric representation.

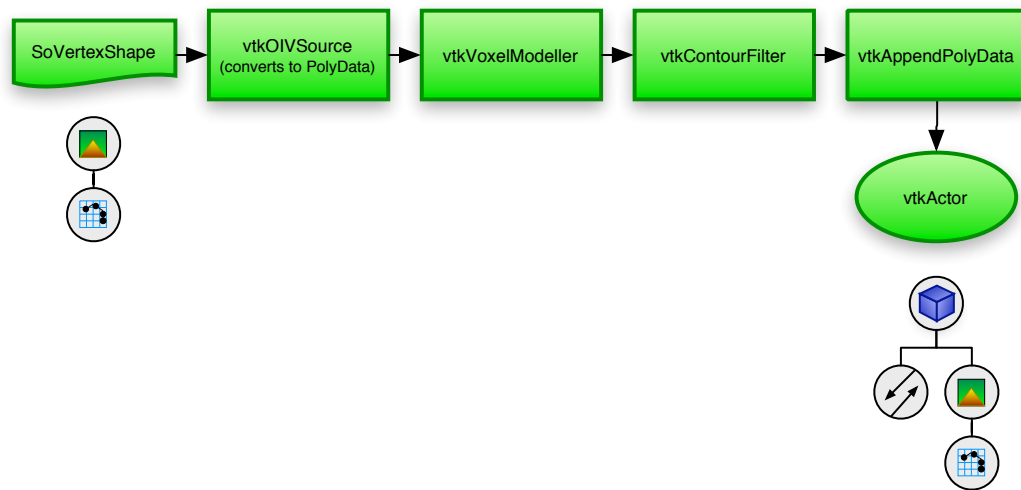


Figure 19: The VTK implant generation pipeline.

The CrEdit implant generation tool executes a VTK pipeline similar to the defect specification tool. As shown in Figure 19, the pipeline uses the lines as input for distance calculations performed within the `vtkVoxelModeller`. The voxel modeller calculates each voxel's distance from the input geometry. If the voxel is within a given threshold distance, then the voxel is marked as binary on. If the voxel is not within the threshold, then the voxel is marked as binary off. This more compact binary storage is the `vtkVoxelModeller`'s difference from the `vtkImplicitModeller`. The use of binary values rather than distances leads to slight increases in performance as well as reduced data size.

#### **4.3.4 Slice Tool**

Following demonstrations of CrEdit and PARIS to GE Medical Systems in 2003 (GE Medical Systems, personal communication, 2003), development included the ability to view volumetric medical slice data. The algorithm simply maps a sequence of 2D CT images into a 3D OpenGL texture. Three slices interactively move through the image data in the axial, sagittal, and coronal plane as shown in Figure 20.

CrEdit updates the slices' alignment by implementing the Observer pattern. Each time the modeler touches the skull surface, a touch notification event triggers updating the slices. This notification process is tied directly into the slice tool, so that tool becomes a mechanism for interactively exploring the 3D images originally used to construct the skull geometry.

Medical sculptors already use CT data for presurgical planning. Incorporating this information into the virtual environment provides another resource available within the digital medium. The direct manipulation of slices is extremely fast, as CrEdit adjusts only the texture coordinates without doing

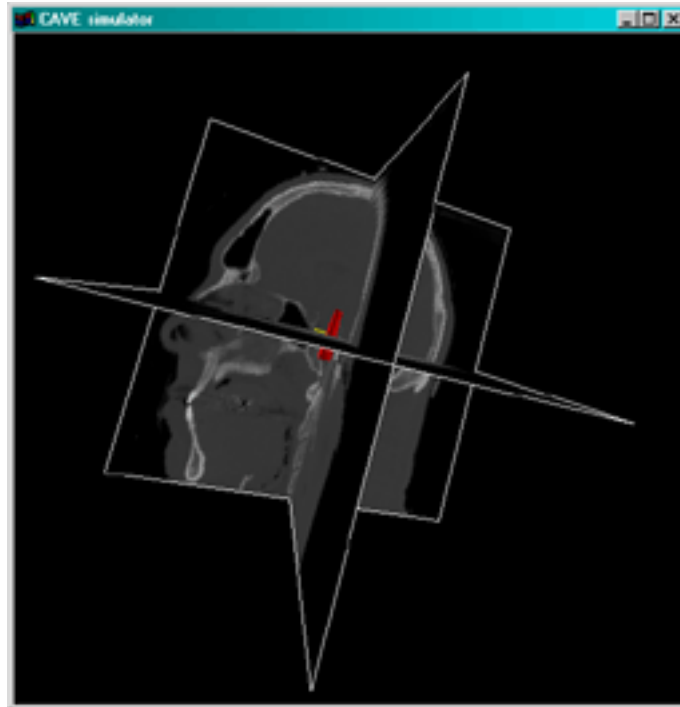


Figure 20: The Slice Tool explores image-based medical slices.

any processing on the CPU. The resolution is limited by available graphics memory, but the projected workspace only provides 1280x1024 pixels.

## **CHAPTER 5**

### **DISCUSSION**

This chapter discusses system evaluation by examining comments from medical experts. This feedback provides insight to the needs and desires of the specialists to whom the problem domain is directly relevant. This chapter also explores several other potential applications of the integrated technology and software techniques. Through the course of CrEdit development, over forty demonstrations have been provided. Visitors came from diverse specialties and countries, and their interests indicate a strong appeal for the haptic augmented environments possible using PARIS.

#### **5.1 Demonstrations**

PARIS demonstrations at conferences yielded feedback from professionals in computer science, radiology, rehabilitation, and other fields. An early version of CrEdit was demonstrated on PARIS at SuperComputing 2002 in Baltimore, Maryland as shown in Figure 21. Located in the FakeSpace booth, PARIS drew visualization and supercomputing experts from laboratories and companies including NASA, the United States Department of Energy, Apple Computer, SARA, and TGS. Visitors expressed surprise when they encountered the force feed co-located with the virtual skull graphic, and many noted how intuitive the system felt.

Following the positive feedback at SC2002, Fakespace requested another demonstration several months later at the Medicine Meets Virtual Reality 2003 conference in Newport Beach, California. Attendees included computer scientists, surgeons, medical researchers, and other medical professionals.



Figure 21: CrEdit demonstration on PARIS at SC2002 in Baltimore, Maryland.

This conference showcased cutting-edge applications of virtual reality and simulation technologies for medicine. Visitors especially liked the large screen, as the enveloping field of view immersed them into the environment. The head tracking provided a unique capability not available on other systems including the Reach-In configuration demonstrated in the adjacent booth.

General Electric Medical Systems visited EVL in early 2003, and their interest in CrEdit led to a demonstration in GE's Technology Pavilion at the Radiological Society of North America 2003 in Chicago, Illinois. Over 40,000 attendees visited this conference, and the CrEdit demonstration on PARIS was a highlight of GE's demonstrations of innovative technologies.

## **5.2 Domain Expert Evaluation**

CrEdit development took place in tight collaboration with Ray Evenhouse, who was the medical sculptor in the implant process described in Section 2.1. He noted the advantages of CrEdit as particu-

larly related to the 3D view, sensory perception, and collaboration potential. (Ray Evenhouse, personal communication, Mar 2005).

While medical sculptors work with clay and wax, neurosurgeons are more familiar with imaging technologies such as CT and MRI. They have a higher familiarity with 2D computer images and 3D reconstructions from those image, and PARIS provides a natural extension of those displays within its inclusion of stereoscopic presentation. Stereo brings the graphics one step closer to the 3D of the real world. Head tracking enables taking advantage of motion parallax as a depth cue. The large workspace is extremely valuable, as it allows placing auxiliary patient or procedure information elsewhere in the environment. The workspace provides ample room for visual cues that might aid the implant design for a specific patient.

CrEdit's haptic force feedback creates a sense of touch that approximates working with physical objects. The PHANToM stylus shape is similar to many of the sculpting tools used during cranial implant fabrication. Tactile evaluation is essential to the design process. Although sculptors use numerous photographs as references, the most important evaluation is done by physically touching a model to examine how it feels. Evenhouse says that students are instructed that, "When it feels right, then it's right" (Ray Evenhouse, personal communication, Mar 2005).

Networking could significantly enhance collaboration among all participants in the design process, including the sculptor, physician, surgeon, as well as the patient. Rather than the technical decisions made by the sculptor, the patient and surgeon should be responsible for making the critical aesthetic decisions (Ray Evenhouse, personal communication, Mar 2005). The patient and relatives have more



direct knowledge of the ideal implant shape. This opportunity for including patients in the process is unique, and it would more directly involve them throughout the design and evaluation procedures.

Fady Charbel, MD is the head of the UIC's Department of Neurosurgery. He visited EVL for a CrEdit demonstration in spring 2004 Figure 23, Appendix A. During his visit, Dr. Charbel noted the intuitive ease with which he could determine bone thickness. The tactile sensation conveyed the bone properties in a matter of seconds (Fady Charbel, personal communication, Apr 2004).

Pravin Patel, MD, Chief of Plastic and Maxillofacial Surgery at the Shriners' Hospital for Children in Chicago examined PARIS during a visit in April 2003. He noted the unique blend of technologies as a potential for future work in facial surgery in addition to cranial implants (Pravin Patel, personal communication, Apr 2003).

Using a digital process allows for easier corrections and changes in course during the design process. As changes are made to the model, CrEdit could maintain a full history. Working in clay, mistakes can force sculptors to start over. A digital design process incorporating undo and redo history would allow faster exploration, and it would eliminate the more time-consuming and expensive steps of the implant process. If the tools are sufficiently refined, then the collaborators may explore variants of a particular implant prior to surgery.

### **5.3 Rehabilitation**

The second PARIS was delivered to the Rehabilitation Institute of Chicago's (RIC) Sensory Motor Performance Program in May 2003. The Robotics Laboratory investigates physical motor performance, examining how the human body performs movements to accomplish certain tasks. Integrating PARIS with a PHANTOM 3.0 robot, RIC is applying the techniques described in this thesis to stroke reha-

bilitation by examining functional skills for daily living (Patton et al., 2004; Scharver et al., 2005). PARIS provides a large workspace otherwise unavailable, and the PHANToM 3.0 provides a larger haptic workspace than the PHANToM Desktop used for CrEdit development.

Beyond using a larger PHANToM, the RIC development also includes work integrating PARIS with the Whole Arm Manipulator (WAM) and HapticMASTER (van der Linde et al., 2002) robotic devices. These devices are able to generate significantly stronger forces than the PHANToM, an important feature when working with subjects who might not have full arm strength. The stronger robots can compensate for the effects of gravity, reducing the muscular effort required to move through the virtual workspace. These devices have different coordinate systems than the PHANToM, but the procedures developed using CrEdit still apply.

Explorations of applying virtual reality to rehabilitation are very recent (Sveistrup, 2004), but the combinations of visual, auditory, and tactile feedback yield diverse opportunities. With the second PARIS, RIC possesses the unique equipment for pursuing this research using the very same hardware and software techniques developed for CrEdit.

#### **5.4 Other Potential Applications**

CrEdit demonstrations have been shown to David Byrne of the Talking Heads (David Byrne, personal communication, 2004) and science fiction author Bruce Sterling during his visit to EVL (Goldfayn, 2004). Their positive reactions highlight the interest in applying augmented reality systems like PARIS to other problem domains.

National Geographic research fellow and paleontologist Paul Sereno examined PARIS and CrEdit in February 2004 (Paul Sereno, personal communication, Feb 2004). He has been excavating dinosaur

fossils encased in limestone. The fragile nature of the very thin cranial bones requires investigating virtual methods. The bone is too delicate to manually separate from the rock. Digital processing would simplify visually separating the bone from the surrounding rock, thus providing the opportunity for virtual analysis.

The CT images slices described in Section 4.3.4 were originally developed as part of a demonstration for GE Medical Systems at the Radiological Society of North America 2003 conference. The company expressed an interest in applying PARIS as a potential 3D radiology workstation of the future. Rather than scrolling through 2D images on a display, a radiologist could directly interact in 3D with a patient's image data. (GE Medical Systems, personal communication, Mar 2004)

The application of the development techniques used while creating CrEdit to other disciplines demonstrates the technology's usefulness. Collocated graphics and haptics within an immersive environment open avenues beyond CrEdit's capabilities.

## CHAPTER 6

### CONCLUSIONS

Developing this application on PARIS revealed several issues with both the hardware and software. Lessons learned from these experiences are already being incorporated into future research plans.

#### **6.1 Lessons Learned**

Through the course of this research, it became clear that additional improvements would be needed. Several hardware issues revealed themselves during demonstrations and subsequent development. The first PARIS was a production prototype, and the available hardware revealed several areas for future improvement. The software techniques utilized in realizing CrEdit's implementation worked well enough to extend to similar projects using different hardware.

##### **6.1.1 Hardware Issues**

The PARIS demonstration at Supercomputing 2002 intended to utilize acoustic tracking. The confined area under the mirror created significant feedback to the audio hardware. Software driver limitations prevented solving these complications. Magnetic tracking has been used as a fallback method while development addresses the acoustic tracking problems.

Magnetic tracking requires a base sensor unit, and that unit is currently mounted under the table at which the modeler sits. The head and hand are both tracked using magnetic trackers. However, if these trackers move too close to the sensor unit, then interference negates tracking. As a result, the wand

sensor cannot be used directly in front of the modeler. The PHANToM is available in this area, but this limitation remains something to address.

Another drawback to this sensor position is that any table movement will offset the measurements for the entire environment. By definition, augmented reality depends on registration of real and virtual objects (Azuma, 1997). Any table or mirror movement compromises calibration. Future hardware designs must minimize any interference and maximize stability.

The project image resolution remains an issue. PARIS is built to include a high-end projector, yet it only exhibits a maximum 1280x1024 pixel resolution. For image data including CT scans, significantly higher resolution would be ideal. Higher resolution allows seeing more information on the projected display, and that's especially important for image-based medical information. The cost of the projector also presents a barrier to the initial cost of the system. Projector capabilities must improve in terms of resolution and affordability.

### **6.1.2 Software Techniques**

Identifying the common capabilities of the multiple libraries facilitated application development. Managing separately the haptic and graphic representations reduced development by relying on existing libraries.

These software techniques are being applied by the Rehabilitation Institute of Chicago to design stroke rehabilitation software using the second manufactured PARIS(tm) (Patton et al., 2004). As described in 5.3, the RIC research integrates the larger PHANToM 3.0 for a tremendously increased haptic work area. Other devices such as the WAM and HapticMaster do not have the high-level capabilities provided by GHOST, but they do provide lower-level communication. The low-level device communi-

cation required by these additional devices is not available in the current version of GHOST, but it is available in SensAble's OpenHaptics toolkit. Future development taking place at this lower-level will require similar abstractions between the haptics and graphics representations.

The tool abstraction described in 4.2 provides a high-level way for accepting 3D user interaction. CrEdit used this abstraction excessively during development and testing. It was also essential when the PHANToM was unavailable on a particular development machine.

### **6.1.3 Defect and Implant Specification**

Using the traditional process, manufacturing a stereolithography defect model can take several hours and cost almost \$2000. Using PARIS, the user may rapidly specify the defect interactively. Depending on the complexity of the injury, the medical modeler can outline a defect in less than one minute.

Implant modeling requires further development. Using VTK is convenient, but it is slow and does not fully integrate with the notion of a scene graph

## **6.2 Implications for Future Research**

Full effectiveness cannot be determined without testing with real patient data. Volume sculpting requires further attention now that developers have obtained familiarity with the hardware. As part of a three-year research grant, evaluation will compare digital models against traditionally constructed models. The digital tools will be refined continuously. Over the next two years, EVL and the VRMedLab will implement and profile the networking between PARIS and other immersive display systems. A new PARIS with tracking and hardware improvements will also be built as part of this research.

Separating computational resources, we will explore how best to take advantage of clustered computing. Only Open Inventor and the CAVELib are tied directly to the graphics hardware. VTK and the

GHOST SDK, both of which primarily perform calculations, do not require graphics processing. These libraries may run on high-performance processors. Clusters may allow partitioning the computational tasks across multiple machines. Intelligently managing computer resources could enable multiple users to securely share medical data, analysis material, evaluation sessions, and more actively steer implant construction.

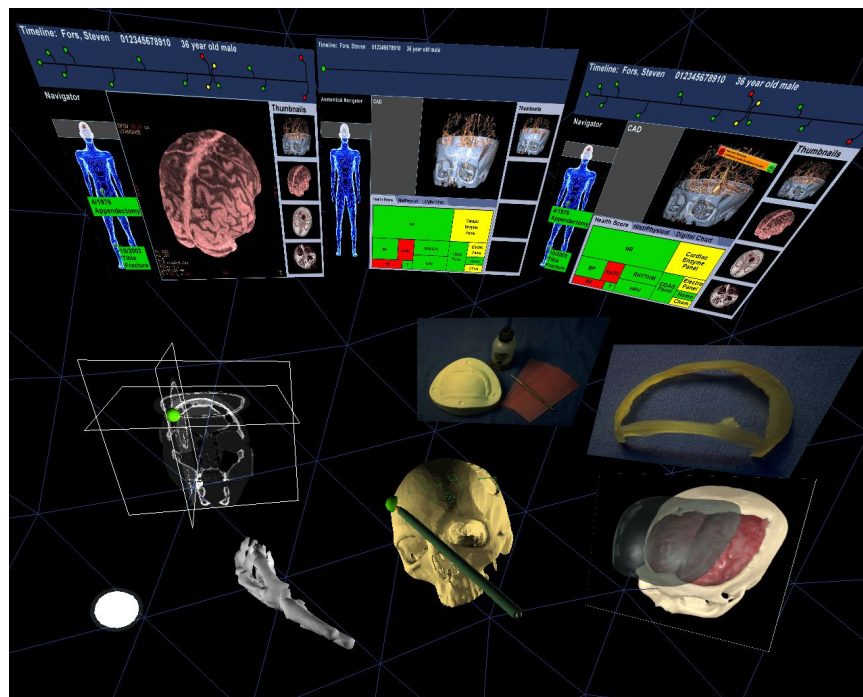


Figure 22: CrEdit environment with simulated patient information systems.

The consultation components can be applied to any area where medical professionals would benefit from a rich visualization environment combined with superior teleconferencing tools. We will explore these other areas in the third year of the grant. Figure 22 depicts the environment enhanced with potential patient information and medical CT slices. These capabilities were demonstrated to radiologists at the Radiological Society of North America's 2003 conference. There is interest in utilizing this system for the pre-operative consultation and planning related to reconstructive surgery. As network capabilities improve, the application can transmit more data to more clients. Future research hopes to determine definitively that utilizing a networked virtual reality system will minimize the consultation time and cost to the patient, surgeon, and manufacturer.

### **6.3 Contributions**

CrEdit's development streamlined the process of integrating a haptic force feedback device with an projected-based augmented reality display. Focusing on cranial implant design, the development process identified needs and capabilities. After identifying how available libraries addressed these issues, CrEdit incorporated intercommunication among these different libraries. As each library tended to its own responsibilities, their strengths contributed to the overall design.

The collated graphics and haptics are an essential combination. PARIS is a unique display due to its large, see-through mirror. Integrated haptics required some trial and error, but the process was eventually refined to a specific series of steps. After calibrating the screen, measure the device origin, and storing configuration values, the system presents a convincing illusion of graphics and haptics within the workspace (Scharver et al., 2004a; Scharver et al., 2004b).



CrEdit's design separates the hardware devices from the software tools. The controller and tool implementation allows using the software with both the PHANToM and the Wanda. This abstraction proved extremely useful during development, as the PHANToM hardware was not required for quick testing. Additionally, this abstraction significantly eased the integration of the HapticMASTER with RIC's PARIS, a process urgently completed within three days. This flexibility would not have been so easily accomplished without CrEdit's foundation.

This research establishes the foundation for a grant from the National Institutes of Health. NIH grant N01-LM-3-3507 "A Tele-Immersive System for Surgical Consultation and Implant Modeling," defines goals to integrate graphics, haptics, and networking to create a distributed volumetric modeling application for tele-immersive cranial implant design. CrEdit directly led to this grant award.

In summary, this thesis presents the following contributions:

- Collocates the PHANToM's haptics with immersive 3D graphics
- Abstracts controller devices, thus allowing interaction and programming flexibility
- Integrates CAVELib, OIV, GHOST, and VTK libraries following an organized data flow among their different geometry representations
- Loads patient medical data into the 3D environment
- Provides VTK pipelines for cranial defect and implant generation
- Implements prototype tools according to the needs of medical sculptors designing cranial implants
- Establishes a foundation for future research including tool refinement and enhanced tele-collaboration

As described in the previous chapter, the contributions contained within this thesis extend beyond the problem domain of cranial implant design. The development paradigm is applicable at sites possessing the other PARIS installations. The prototype system at EVL serves as the main development system for the NIH research grant, and it will also be used to explore alternative tracking methods. The second PARIS is actively involved with stroke rehabilitation research at the Rehabilitation Institute of Chicago, and researchers are integrating that system with other powerful robotic devices. Finally, a third system at the Illinois Technology, Research, Education, and Communications Center (TRECC) runs CrEdit as a demonstration application. These systems all include the software developed for this thesis, and the techniques proposed herein establish an example for future application development.

## **APPENDICES**

## **Appendix A**

### **LETTER FROM FADY CHARBEL, MD**

The attached letter shown in Figure 23 was received following a visit to EVL including a CrEdit demonstration. As the surgeon who performed implants in the original UIC implant process, his reactions to the system are particularly encouraging.

**Appendix A (Continued)**UNIVERSITY OF ILLINOIS  
AT CHICAGO

Department of Neurosurgery (MC 799)  
College of Medicine  
Neuropsychiatric Institute  
912 South Wood Street  
Chicago, Illinois 60612-7329

April 26, 2004

Ray Evenhouse  
School of Biomedical Health Information  
250 AHSB  
MC 520

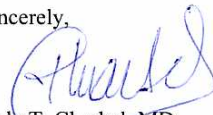
Dear Ray:

The tour today of EBL Laboratories was very exciting and I saw many possibilities for us working together. I was particularly impressed with the ability to manipulate the image in real time, as well as, the work on it with the haptic feedback.

This interaction is very timely because of the increasing shortage of neurosurgeons especially in remote areas, together with the paucity of highly technical subspecialists in this field.

Again, many thanks for the tour. I'm looking forward to working with you on this.

Sincerely,



Fady T. Charbel, MD  
Head and Professor

**UIC**

Phone (312) 996-4842 • Fax (312) 996-9018 • Direct # Line(800) 597-5970

Figure 23: Letter from Fady Charbel, MD, head and professor, UIC Department of Neurosurgery.

## Appendix B

### BUILDING AND RUNNING CREDIT

CrEdit has been built using several different libraries that provide graphics, haptics, and networking capabilities. Building CrEdit from source code requires several toolkits, but the configuration process is one that works in a cross-platform manner. This approach made the transition from Windows to Linux very easy. The libraries and their versions are as follows:

- CAVELib 3.1.1
- Coin3D 2.3.0 with thread-safety enabled
- GHOST 4.0
- QUANTA 0.4
- VTK 4.5

These libraries should be installed and available to the compiler. The source has been tested on Windows 2000 with Visual Studio 6, Red Hat 9 with GCC 3.3, and MacOS X 10.3 with GCC 3.3. The CrEdit source code configures using CMake from Kitware.

#### **B.1 Source Organization**

The source code is organized into several directories. This organization makes it easier to keep source code separate from application models and other data. The libraries described in section 3.4 are designed for inclusion with other projects, so they may be built independently from the main source code.

## Appendix B (Continued)

**coinevents** contains the 3DUI events library

**qavatar** contains the QUANTA avatar library

**vtkoiv** contains the VTK-OIV integration library

**CrEdit** contains the application sources and data files

**data** stores all medical skull and image data used during development

**models** contains 3D models used by the application source code

**rsna-data** contains image slides originally added for demonstration at the RSNA conference

**src** contains the CrEdit source code

Everything configures relative to the very top directory containing the contents listed above. This organization was required due to the way CMake handled subdirectories. This limitation is no longer in place, and recent development versions of CMake allow for better structuring of the source code.

Written by the same authors as the Visualization Toolkit, CMake is a cross-platform make utility. Its strength is that it uses text files to generate platform-specific build files. CMake creates makefiles for UNIX systems, and it creates Visual Studio Project files for Windows. The GUI allows changing paths and variables very easily. CMake also supports out-of-source build directories, thus preventing compiled object and build files from polluting the source tree. The first step of the configuration process is to specify those two directories to CMake. The CMake GUI provides a convenient interface for selecting these two locations.

CMake uses a single CMakeLists.txt file to configure the entire source tree. CrEdit includes several additional CMake modules for finding OpenGL, CAVELib, GHOST, and Inventor libraries. One must

## Appendix B (Continued)

specify to CMake the appropriate include and link locations if the libraries are not available in their default locations. CMake then generates the appropriate files for compiling the CrEdit and its required libraries. Two specific options are unique to the software interactions. First, `COIN_DLL` must be set on Windows systems if Coin3D is compiled as a dynamic link library. Second, `USE_STD_NAMESPACE` manages the use of ANSI namespaces with C++ headers. GHOST includes its own C++ Standard Template Library, and namespaces cause conflicts when compiling CrEdit on Windows. However, building CrEdit on Linux does require the use of namespaces.

### **B.2 Running CrEdit**

After configuring and compiling the source code, one needs to make sure that a configuration file is available. This configuration file must include the required information discussed in Section 4.2.

The build process generates the executable, but the data files must be available in the current working directory path. In addition to the configuration file, any associated data must be available. Typically, this requirement is satisfied by making certain that all application models and data files are available from the same working path. One may then execute CrEdit from within that directory. These steps will ensure that CrEdit may load everything properly into the virtual environment.

1. Start trackd to run the tracking system.
2. Change to the main CrEdit directory.
3. Execute CrEdit by specifying the path to the built executable along with the desired configuration file: `(buildPath)/credit.exe VisibleHuman.config`



## Appendix B (Continued)

With tracking up and running, CrEdit will display with the correct registration on PARIS. The haptics and graphics alignment might need periodic adjustment, but resolving any errors requires recalibrating the display system. Refer to the trackd and CAVELib documentation for proper configuration of those software components.

While running CrEdit, one may use the wand buttons to cycle through the available tools. The PHANToM stylus activates tools which use the stylus for interaction, but several key commands trigger defect generation. These extra interactions are summarized as follows:

**Wand button1** cycle backward through PHANToM tools

**Wand button3** cycle forward through PHANToM tools

**Wand button2 + button3** reset all object locations

**Keyboard S key** show the skull model, sometimes hidden after defect generation

**Keyboard F1 key** generate the defect mesh by calculating defect tool geometry calculations

One manipulates the primary skull geometry using the PHANToM, but the Wand provides the ability to select and move other objects in the scene. The light sources, reference images, and other models may be moved freely through the environment. The PHANToM remains the primary interaction device for the skull and modeling tools.

## CITED LITERATURE

- [American Association of Neurological Surgeons, 1999]American Association of Neurological Surgeons: Top reported cranial procedures performed in 1999, 1999.
- [Anupam and Bajaj, 1994]Anupam, V. and Bajaj, C.: Shastra: a multimedia collaborative design environment. IEEE Multimedia, 1(2):39–49, 1994.
- [Ayasse and Mueller, 2001]Ayasse, J. and Mueller, H.: Interactive manipulation of voxel volumes with free-formed voxel tools. In Vision Modeling and Visualization, pages 359–366, Stuttgart, Germany, Nov 21-23 2001.
- [Azuma, 1997]Azuma, R.: A survey of augmented reality. Presence: Teleoperators and Virtual Environments, 6(4):355–385, Aug 1997.
- [Bell et al. , 1995]Bell, G., Parisi, A., and Pesce, M.: The virtual reality modeling language version 1.0 specification, 1995.
- [Bibb et al. , 2002]Bibb, R., Bocca, A., and Evans, P.: An appropriate approach to computer aided design and manufacture of cranioplasty plates. Journal of Maxillofacial Prosthetics & Technology, 5(129), 2002.
- [Billinghurst and Kato, 2002]Billinghurst, M. and Kato, H.: Collaborative augmented reality. Communications of the ACM, 45(7):64–70, Jul 2002.
- [Blinn, 1982]Blinn, J. F.: A generalization of algebraic surface drawing. ACM Transactions on Graphics, 1(3), Jul 1982.
- [Bloomenthal, 1989]Bloomenthal, J.: Techniques for implicit modeling. Technical Report P89-00106, Xerox PARC, Palo Alto, California, 1989.
- [Bouguila et al. , 2000]Bouguila, L., Ishii, M., and Sato, M.: Effect of coupling haptics and stereopsis on depth perception in virtual environment. In Proceedings of the Haptic Human-Computer Interaction Workshop, pages 54–62, Aug 2000.
- [Burdea, 2000]Burdea, G.: Haptic issues in virtual environments. In IEEE Computer Graphics International, pages 295–303, Geneva, Switzerland, Jun 19-24 2000.

- [Burgert et al. , 2003]Burgert, O., Seifert, S., Salb, T., Gockel, T., Dillmann, R., Hassfeld, S., and Muehling, J.: A vr-system supporting symmetry related cranio-maxillofacial surgery. In Medicine Meets Virtual Reality, volume 94, pages 33–35, Newport Beach, California, 2003. IOS Press.
- [Butterworth et al. , 1992]Butterworth, J., Davidson, A., Hench, S., and Olano, T.: 3dm: A three dimensional modeler using a head-mounted display. In ACM Symposium on Interactive 3D Graphics, volume 25, pages 135–138, Cambridge, MA, Mar 30 - Apr 1 1992.
- [Cruz-Neira et al. , 1993]Cruz-Neira, C., Sandin, D., and DeFanti, T.: Virtual reality: The design and implementation of the cave. In Proceedings of the SIGGRAPH Computer Graphics Conference, pages 135–142, Aug 1993 1993.
- [Deering, 1995]Deering, M.: Holosketch a virtual reality sketching/animation tool. ACM Transactions on Computer-Human Interaction, 2(3):220–238, 1995.
- [Deisinger et al. , 2000]Deisinger, J., Blach, R., Wesche, G., Breining, R., and Simon, A.: Towards immersive modeling - challenges and recommendations: A workshop analyzing the needs of designers. In EuroGraphics Conference of Virtual Environments, Amsterdam, the Netherlands., 2000.
- [Dujovney et al. , 1999]Dujovney, M., Evenhouse, R., Agner, C., Charbel, L., Sadler, L., and McConathy, D.: Performed prosthesis from computer tomography data: Repair of large calvarial defects. In Calvarial and Dural Reconstruction, eds. S. Rengachary and E. Benzel, pages 77–88. Park Ridge, Illinois, American Association of Neurological Surgeons, 1999.
- [Esposito, 1996]Esposito, C.: User interface issues for virtual reality systems. In ACM Conference on Computer-Human Interaction, pages 340–341, Apr 13-18 1996.
- [Ferley et al. , 1999]Ferley, E., Cani, M., and Gascuel, J.: Practical volumetric sculpting. In Implicit Surface '99, 1999.
- [Fisher and Vance, 2002]Fisher, A. and Vance, J.: Implementing haptics feedback in a projection screen virtual environment. In Proceedings of the PHANToM Users Group Workshop, Santa Fe, New Mexico, 2002.
- [Galyean and Hughes, 1991]Galyean, T. and Hughes, J.: Sculpting: an interactive volumetric modeling technique. ACM Transactions on Computer Graphics, 25(4):267–274, Jul 1991.
- [Gamma et al. , 1995]Gamma, E., Helm, R., Johnson, R., and Vlissedes, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley Professional Computing Series. Reading, Massachusetts, Addison-Wesley, 1995.

- [Goldfayn, 2004]Goldfayn, A.: Wi-fi opens web possibilities – and the outdoors. Chicago Tribune, (129):Section 2, 4, May 8 2004.
- [Grimm et al. , 1995]Grimm, C., Pugmire, D., Bloomenthal, M., Hughes, J., and Cohen, E.: Visual interfaces for solid modeling. In ACM Conference on UserInterface Software and Technology, pages 51–60, 1995.
- [Hinkley et al. , 1994]Hinkley, K., Pausch, R., Goble, J., and Kassell, N.: A survey of design issues in spatial input. In ACM Symposium on User Interface Software and Technology, pages 213–220, Marina del Ray, California, Nov 2-4 1994.
- [Hollerbach, 2000]Hollerbach, J.: Some current issues in haptics research. In IEEE Conference on Robotics and Automation, pages 757–762, San Francisco, California, Apr 24-28 2000.
- [Hudson et al. , 2003]Hudson, T., Helser, A., Sonnenwald, D., and Whitton, M.: Managing collaboration in the nanomanipulator. In IEEE Virtual Reality, pages 180–187, Los Angeles, CA, Mar 22-26 2003.
- [Johnson et al. , 2000]Johnson, A., Sandin, D., Dawe, G., DeFanti, T., Pape, D., Qiu, Z., Thongrong, S., and Plepys, D.: Developing the paris: Using the cave to prototype a new vr display. In ACM Symposium on Immersive Projection Technology, Ames, Iowa, Jun 19-20 2000.
- [Keeve et al. , 1996]Keeve, E., Girod, S., Pfeifle, P., and Girod, B.: Anatomy-based facial tissue modeling using the finite element method. In Proceedings of the 7th IEEE Conference on Visualization, pages 21–ff, San Francisco, California, Oct 1996. IEEE Computer Society Press.
- [Kling-Petersen et al. , 2000]Kling-Petersen, T., , and Rydmark, M.: Modeling and modification of medical 3d objects. the benefits of using a haptic modeling tool. In Medicine Meets Virtual Reality, Newport Beach, California, Jan 27-30 2000.
- [Massie, 1998]Massie, T.: A tangible goal for 3d modeling. IEEE Computer Graphics and Applications, 18(3):62–65, 1998.
- [Matsumiya et al. , 2000]Matsumiya, M., Takemura, H., and Yokoya, N.: An immersive modeling system for 3d free-form design using implicit surfaces. In ACM Symposium on Virtual Reality Software and Technology, pages 67–74. ACM Press, 2000.
- [Mine, 1996]Mine, M.: Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. Technical report, University of North Carolina at Chapel Hill, 1996.

- [Mitsunobu et al. , 1998]Mitsunobu, H., Oshiba, T., and Tanaka, J.: Claymore: Augmented direct manipulation of three-dimensional objects. In Third Asian Pacific Computer and Human Interaction, pages 210–217, Kangawa, Japan, Jul 15-17 1998.
- [Mortensen et al. , 2002]Mortensen, J., Vinayagamoorthy, V., Slater, M., Steed, A., Lok, B., and Whitton, M. C.: Collaboration in tele-immersive environments. In Proceedings of the Eighth Eurographics Workshop on Virtual Environments, pages 93–101. Eurographics Association, 2002.
- [Nielsen, 2000]Nielsen, G.: Volume modelling. In Volume Graphics, eds. M. Chen, A. Kaufman, and R. Yagel, pages 28–48. London, England, Springer, 2000.
- [Nishino et al. , 1998]Nishino, H., Utsumiya, K., Korida, K., Sakamoto, A., and Yoshida, K.: 3d object modeling using spatial and pictographic gestures. In ACM Symposium on Virtual Reality Software and Technology, pages 51–58, Taipei, Taiwan, Nov 2-5 1998.
- [Park et al. , 2000]Park, K., Cho, Y., Krishnaprasad, N., Scharver, C., Lewis, M., and Leigh, J.: Cavernsoft g2: A toolkit for high performance tele-immersive collaboration. In ACM Symposium on Virtual Reality Software and Technology, pages 8–15, Seoul, Korea, Oct 22-25 2000.
- [Parvati, 2000]Parvati, D.: Graphics and imaging in medicine. IEEE Computer Graphics and Applications, 20(1):24–25, 2000.
- [Patton et al. , 2004]Patton, J., Dawe, G., Scharver, C., and Kenyon, R.: The development of a life-sized 3-d system for the rehabilitation of motor function. In Proceedings of 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Sep 5 2004.
- [Plesniak and Pappu, 1998]Plesniak, W. and Pappu, R.: Coincident display using haptics and holographic video. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 304–311, Los Angeles, CA, Apr 1998. ACM Press.
- [Potts, 2000]Potts, A.: Phantom-based haptic interaction, 2000.
- [Preston and Lever, 1996]Preston, M. and Lever, P.: Scene description issues for computer graphics. In Eurographics UK, pages 49–60, London, United Kingdom, 1996.
- [Pyo and Shin, 2002]Pyo, S. and Shin, Y.: Fast volume carving. In EuroGraphics, Saarbrueken, Germany, Sep 2-6 2002.
- [Rajlich, ]Rajlich, P.: vtkactortopf.

- [Reinig and Eskin, 2002]Reinig, K. and Eskin, J.: A system for accurate collocation of haptics and graphics. In Proceedings of the Seventh PHANToM Users Group Workshop, 2002.
- [Rohlf and Helman, 1994]Rohlf, J. and Helman, J.: Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In ACM SIGGRAPH Special Interest Group on Computer Graphics and Interactive Techniques, pages 381–394. ACM Press, 1994.
- [Rugelj and Svelj, 1997]Rugelj, J. and Svelj, V.: Computer supported multimedia environment for the support of long-distance collaboration in medicine. In IEEE Symposium on Computer-Based Medical Systems, pages 215–220, 1997.
- [Sachs et al. , 1991]Sachs, E., Roberts, A., and Stoops, D.: 3-draw: A tool for designing 3d shapes. IEEE Computer Graphics and Applications, 11(6):18–26, 1991.
- [Salb et al. , 2002]Salb, T., Burgert, O., Gockel, T., Brief, J., Hassfeld, S., Muehling, J., and Dillmann, R.: Risk reduction in crniofacial surgery using computer-based modeling and intraoperative immersion. In Medicine Meets Virtual Reality, Newport Beach, California, Jan 2002.
- [Salisbury and Srinivasan, 1997]Salisbury, J. and Srinivasan, M.: Phantom-based haptic interaction with virtual graphics. IEEE Computer Graphics and Applications, 17(5):6–10, 1997.
- [Sawant et al. , 2000]Sawant, N., Scharver, C., Leigh, J., Johnson, A., Reinhart, G., Creel, E., Batchu, S., Bailey, S., and Grossman, R.: The tele-immersive data explorer: A distributed architecture for collaborative interactive visualization of large data-sets. In International Immersive Projection Technology Workshop, Ames, Iowa, Jun 19-20 2000.
- [Scharver et al. , 2004a]Scharver, C., Evenhouse, R., Johnson, A., and Leigh, J.: Designing cranial implants in a haptic augmented reality environment. Communications of the ACM, 47(8):32–38, Aug 2004.
- [Scharver et al. , 2004b]Scharver, C., Evenhouse, R., Johnson, A., and Leigh, J.: Pre-surgical cranial implant design using the paris prototype. In Proceedings of the IEEE Conference on Virtual Reality, pages 199–206, Mar 27-31 2004.
- [Scharver et al. , 2005]Scharver, C., Patton, J., Kenyon, R., and Kersten, E.: Comparing adaption of constrained and unconstrained movements in three dimensions. In Proceedings of the IEEE International Conference on Rehabilitation Robotics, Chicago, Illinois, Jun 2005.
- [Schkolne et al. , 2002]Schkolne, S., Pruett, M., and Schröder, P.: Drawing with the hand in free space. Leonardo, 35(4), Aug 2002.

- [Schroeder et al. , 2003]Schroeder, W., Martin, K., and Lorensen, B.: The Visualization Toolkit An Object-Oriented Approach To 3D Graphics. Kitware, Inc., 3 edition, 2003.
- [SensAble Technologies, 2001]SensAble Technologies: Ghost sdk, 2001.
- [Silicon Graphics Inc., ]Silicon Graphics Inc.: Open inventor.
- [Simultronics in Motion, ]Simultronics in Motion: Coin3d.
- [Singhal and Zyda, 1999]Singhal, S. and Zyda, M.: Networked Virtual Environments. New York, New York, ACM Press, 1999.
- [Sowizral, 2000]Sowizral, H.: Scene graphs in the new millennium. IEEE Computer Graphics and Applications, 20(1):56–57, Jan 2000.
- [Strauss and Carey, 1992]Strauss, P. and Carey, R.: An object-oriented 3d graphics toolkit. ACM SIGGRAPH Computer Graphics, 26(2):341–349, Jul 1992.
- [Sveistrup, 2004]Sveistrup, H.: Motor rehabilitation using virtual reality. Journal of NeuroEngineering and Rehabilitation, 1(10), Dec 2004.
- [Szymanski, 1995]Szymanski, M.: VIRUSES: A Virtual Reality Shape Editing System, and the Complexities of Editing in a Virtual Environment. Doctoral dissertation, University of Illinois at Chicago, 1995.
- [Taha et al. , 2001]Taha, F., Testelin, S., Lengele, B., and Boscherini, D.: Modeling and design of a custom made cranium implant for large skull reconstruction before a tumor removal, Jun 2001.
- [Taylor, 1999]Taylor, R.: Scientific applications of force feedback: Molecular simulation and microscope control. In ACM SIGGRAPH, 1999.
- [TGS, Inc., 2002]TGS, Inc.: TGS Open Inventor 3.1 User's Guide. San Diego, California, 2002.
- [Vallino and Brown, 1999]Vallino, J. and Brown, C.: Haptics in augmented reality. In IEEE International Conference on Multimedia computing and Systems, volume 1, page 9195, Florence, Italy, Jun 7-11 1999.
- [van der Linde et al. , 2002]van der Linde, R., Lammertse, P., Frederiksen, E., and Rulter, B.: The haptic-master, a new high-performance haptic interface. In Proceedings of EuroHaptics 2002, Edinburgh, United Kingdom, Mar 2002.

- [van Liere et al. , 1998]van Liere, R., Harkes, J., and de Leeuw, W.: A distributed blackboard architecture for interactive data visualization. In Proceedings of the IEEE Conference on Visualization, eds. E. D., H. Rushmeier, and H. Hagen. IEEE Computer Society Press, 1998.
- [Wang, 1999]Wang, H.: A haptic viewer of three dimensional model, 1999.
- [Wernecke, 1994a]Wernecke, J.: The Inventor Mentor: Programming Object-Oriented Three Dimensional Graphics with Open Inventor, Release 2. Reading, Massachusetts, Addison-Wesley Publishing, reading, massachusetts edition, 1994.
- [Wernecke, 1994b]Wernecke, J.: The Inventor Toolmaker: Extending Open Inventor, Release 2. Reading, Massachusetts, Addison-Wesley Publishing, 1994.
- [Wloka and Anderson, 1995]Wloka, M. M. and Anderson, B. G.: Resolving occlusion in augmented reality. In Proceedings of the ACM Symposium on Interactive 3D Graphics. ACM Press, 1995.
- [Wohlfahrter and Encarnação, 2000]Wohlfahrter, W. and Encarnação, M.: Medidesk: Interactive volume exploration on the studydesk. In Proceedings of the Fourth International Immersive Projection Technology Workshop, Ames, Iowa, Jun 2000.
- [Wong et al. , 2000]Wong, J., Lau, R., and Ma, L.: Virtual 3d sculpting. Journal of Visualization and Computer Animation, 11(3):155–166, 2000.
- [Wood et al. , 1997]Wood, J., Wright, H., and Brodlie, K.: Collaborative visualization. In Proceedings of the IEEE Conference on Visualization, pages 253–259, Phoenix, Arizona, 1997. IEEE Computer Society Press.
- [Zelevnik et al. , 1991]Zelevnik, R., van Dam, A., Conner, D., Wloka, M., Aliaga, D., Huang, N., Hubbard, P., Knep, B., Kaufman, H., and Hughes, H.: An object-oriented framework for the integration of interactive animation techniques. In ACM SIGGRAPH Special Interest Group on Computer Graphics and Interactive Techniques, pages 105–112. ACM Press, 1991.



## VITA

### CHRISTOPHER SCOTT SCHARVER

#### EDUCATION

- B.S., Computer Science, Furman University, Greenville, South Carolina, USA, 1998
- M.S, Computer Science, University of Illinois at Chicago, Illinois, USA, 2005

#### EXPERIENCE

- Research Engineer, Rehabilitation Institute of Chicago, Spring 2004 - present
- Software Developer, VRCO, Inc., Summer 2001 - Summer 2003
- Research Assistant, Electronic Visualization Laboratory, University of Illinois at Chicago, Fall 1998 - Spring 2004

#### PUBLICATIONS

- Scharver, C., Patton, J., Kenyon, R., and Kersten, E.: Comparing adaption of constrained and unconstrained movements in three dimensions. In Proceedings of the IEEE International Conference on Rehabilitation Robotics, Jun 2005.
- Patton, J., Dawe, G., Scharver, C., and Kenyon, R.: The development of a life-sized 3-d system for the rehabilitation of motor function. In Proceedings of 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Sep 5 2004.
- Scharver, C., Evenhouse, R., Johnson, A., and Leigh, J.: Designing cranial implants in a haptic augmented reality environment. Communications of the ACM, 47(8):32-38, Aug 2004.
- Scharver, C., Evenhouse, R., Johnson, A., and Leigh, J.: Pre-surgical cranial implant design using the paris prototype. In Proceedings of the IEEE Conference on Virtual Reality, pages 199-206, Mar 27-31 2004
- Park, K., Cho, Y., Krishnaprasad, N., Scharver, C., Lewis, M., and Leigh, J.: Cavernsoft g2: A toolkit for high performance tele-immersive collaboration. In ACM Symposium on Virtual Reality Software and Technology, pages 8-15, Seoul, Korea, Oct 22-25 2000.
- Sawant, N., Scharver, C., Leigh, J., Johnson, A., Reinhart, G., Creel, E., Batchu, S., Bailey, S., and Grossman, R.: The tele-immersive data explorer: A distributed architecture for collaborative interactive visualization of large data-sets. In International Immersive Projection Technology Workshop, Ames, Iowa, Jun 19-20 2000.

#### PROJECTS

- Inner ear Performer visualization
- Tele-Immersive Data Explorer (TIDE)

- Collaborative Micro-Electric Machine Simulation (MEMS)
- Immersaview passive stereo viewer
- Immersive waterfall builder

#### CONFERENCES

- IEEE Virtual Reality 2004, Chicago, Illinois, March 29-31, 2004
- Radiological Society of North America 2003, Chicago, Illinois, December 1-5, 2003
- Medicine Meets Virtual Reality, Newport Beach, California, January 22-24, 2003
- Supercomputing 2002, Baltimore, Maryland, 2002
- American Geophysists Union 2004, San Francisco, California, December 10-14, 2001
- Supercomputing 2000, Dallas, Texas, USA, 2000
- Internet and iGrid 2000, Yokohoma, Japan, August 2000
- Eurographics Workshop on Virtual Environment, Amsterdam, the Netherlands, 2000
- Immersive Projection Technology Workshop, Ames, Iowa, 1999
- Workshop on Large Data Visualization, Salt Lake City, Utah, 1999
- Supercomputing 1999, Portland, Oregon, 1999

#### OPEN SOURCE PROJECT PARTICIPATION

- CMake, <http://www.cmake.org/>
- Coin3D, <http://www.coin3d.org/>
- DarwinPorts, <http://darwinports.opendarwin.org/>
- SGI Open Inventor, <http://oss.sgi.com/projects/inventor/>
- Visualization Toolkit, <http://www.vtk.org/>

REFERENCES AVAILABLE ON REQUEST