The Tinmith System - Demonstrating New Techniques for Mobile Augmented Reality Modelling

Wayne Piekarski and Bruce H. Thomas

Wearable Computer Laboratory
School of Computer and Information Science
University of South Australia
Mawson Lakes, SA, 5095, Australia

{wayne, thomas}@cs.unisa.edu.au

Abstract

This paper presents user interface technology, using a glove based menuing system and 3D interaction techniques. It is designed to support applications that allow users to construct simple models of outdoor structures. The construction of models is performed using various 3D virtual reality interaction techniques, as well as using real time constructive solid geometry, to allow users to build up shapes with no prior knowledge of the environment. Previous work in virtual environments has tended to focus mostly on selection and manipulation, but not starting from an empty world. We demonstrate our user interface with the Tinmith-Metro application, designed to capture in city models and street furniture.

Keywords: wearable computers, augmented reality, user interfaces, constructive solid geometry, 3D modelling.

1 Introduction

We have been investigating immersive 3D outdoor Augmented Reality (AR) architectures and applications. Our previous investigations have required 3D graphical models of buildings and land features, in order to render them for various applications. We believe that the construction of models interactively while outdoors, matching real physical features, is a convenient method that is efficient and visually verifiable. The authors know of no system that performs 3D outdoor AR graphical modelling; traditionally, models are designed in 2D desktop applications.

We have developed a user interface known as Tinmith-Hand, which uses pinch gloves and hand tracking to control a menu and manipulate 3D virtual objects. This has been implemented and used to develop a modelling system, known as Tinmith-Metro, which allows us to capture the designs of outdoor buildings and structures. The 3D modeller is based around the intuitive nature of constructive solid geometry (CSG) operations, which we include as an integral part of the user interface.

The remainder of this section will discuss the aims and objectives of our work. To place our work in context with that of other researchers, section 2 provides an overview of related work that forms a foundation of our ideas. We describe an example in section 3 showing from a user's perspective how Tinmith-Metro is used to model a building. The novel menuing system with pinch gloves is discussed in section 4, followed by the use of hand tracking and image plane techniques for selection and manipulation of models in section 5. The graphical modelling concepts (CSG) are explained in section 6, and how they are used to build complicated outdoor structures. The implementation details of the overall system are described in section 7. We conclude this paper discussing some of the problems solved, future directions, and the knowledge gained from this investigation.

1.1 Augmented Reality

Augmented Reality is the process of overlaying computer-generated images over the real world. By using a transparent head mounted display (HMD) placed on the head, combined with a wearable computer, (see Figure 1) it is possible for the user to walk outdoors and the computer to draw images to enhance their vision. Part (3) of Figure 2 shows how images from the real world are combined with computer generated images, using a Sony Glasstron HMD. This HMD uses a half-silvered mirror as an optical combiner, presenting the final image to the user. AR systems allow the user to have "X-ray vision", visualising objects which may not be visible to the user in the real world.

1.2 Why Is Outdoor AR Hard?

Performing outdoor AR research is harder than most other kinds of virtual environments, with the lack of lightweight, portable equipment that performs the tasks

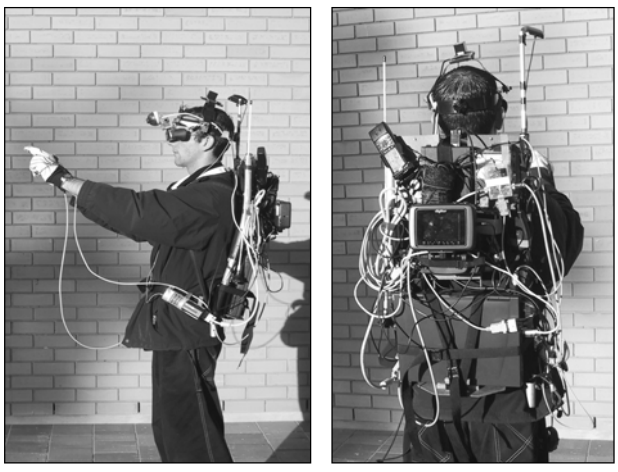

Figure 1 - Outdoor Tinmith Backpack Computer

desired. Requiring all the hardware to be portable places many restrictions on the equipment that can be used, as much of it is too large or cannot run from a battery.

The field of virtual reality (VR) also suffers from the lack of proper input devices and sub-optimal tracking systems. Therefore new input devices, interfaces, and trackers are continuing to be developed in an attempt to solve these problems. However, many of these devices require fixed infrastructure and are not useable in mobile outdoor environments. Two excellent papers - (Azuma 1997) and (Azuma 1999), explain the problems of working outdoors, and the various technologies that are currently available.

The problem of tracking and registering virtual images with the user's view of the physical world is a main focus of AR research, as can be seen in many current AR publications. However, there is little previous work in the area of user interfaces for controlling AR systems in an outdoor setting, which is the main focus of this paper.

1.3 Aims of this work

Previous AR and VR systems tend to be focused on the presentation of information with a standard set of interaction techniques. The construction of graphical models by in large is performed external to the virtual environment. When the construction of the models is in a virtual environment, this tends to be the manipulation of pre-existing graphical objects, such as houses, cars, or tanks. We wish to be able to construct and interact with the graphical models in the virtual environment at a deeper level. Furthermore, the construction of these new models from scratch would be performed outdoors, with minimal prior knowledge of existing physical structures, and with a small number of primitive tools.

We desired to implement a user interface and set of applications that would allow the user to construct new models, and to do this using natural head and hand gestures. Therefore, we wish to keep the hands free from holding input devices if possible.

1.4 Concepts implemented

The user operates the application with the Tinmith-Hand user interface, using head movement, hand tracking, pinch gloves, and a menu system to perform the following object manipulation tasks:

Object selection – the user can point at objects and select them, placing them into one of several clipboards.

Object transform – perform translate, rotate, and scale operations, in a variety of different ways.

Create primitives – 3D primitives can be created in the virtual world, from infinite planes as the most primitive, to complex graphical models such as a water heater.

Combine primitives – previously constructed and manipulated primitives may be combined together using Constructive Solid Geometry (CSG) operations to produce higher level graphical objects.

The following components are used to implement the user interface and applications, used to construct large graphical objects outdoors:

Menu system and pinch gloves – the command interface to the system through the pinch action of our gloves. These gloves were custom built to integrate in with the rest of the system.

Four integrated pointing techniques – the system is capable of using four interchangeable pointing devices to supply input, depending on the requirements at the time and the suitability. The devices are one and two handed finger tracking, a head orientation eye cursor, and a track ball mouse.

Image plane interaction techniques – these techniques are where the objects are manipulated on a 2D plane perpendicular to the current view direction (Pierce 1997). By combining pointing with image plane techniques, it is possible to manipulate objects in a 3D environment, selecting an appropriate camera angle simply by walking.

Application tailored menus – to support the domain specific construction application, menu options are added that tailor the menu system to the domain specific tasks.

CSG operations – users intuitively understand operations such as carving and combining objects. We have leveraged this understanding by basing the interactive construction of complex real world shapes around the use of CSG operations.

2 Background

2.1 Previous Augmented Reality Systems

Most augmented and virtual reality systems are oriented toward information presentation, the user wearing a HMD, moving around the world, and experiencing the artificial reality. There have been a small number of systems for outdoor augmented reality such as the MARS Touring machine (Feiner 1997), NRL BARS system (Julier 2000), previous UniSA Tinmith navigation systems such as (Piekarski 1999) and (Thomas 1998), and UniSA ARquake (Thomas 2000). However, these systems only allow the user to control the presentation of the information, and not actually create new information, especially 3D models.

2.2 Previous Virtual Reality interaction work

There are a number of concepts we have leveraged from the area of VR interaction, although most of these are focused on the manipulation of existing objects. As a result, the construction of new objects from primitive building blocks is a fruitful area of investigation.

Our system builds on concepts from a number of researchers, including: Proprioception and the placement of objects relative to the body in (Mine 1997); The viewing and manipulations of objects using the Worlds-in-Miniature (Stoakley 1995) and Voodoo Dolls (Pierce 1999) techniques; Two handed 3D interaction techniques in (Hinckley 1994) and (Zeleznik 1997); selection and

manipulation techniques like the GoGo arm (Poupyrev 1996) and various others covered in (Bowman 1997).

A menuing system developed at a similar time as ours is an immersive VR system using Pinch Gloves (FakeSpace Labs 2001) recently described in (Bowman 2001). Although a similar concept as ours, it was different in that it was very much like traditional pull down menus. The top-level menu items were available on one hand, and the second level options on the other hand. Using the small finger it was possible to cycle through options if there were more than three options. The menus were limited to a depth of two, and it is not scalable to a large number of hierarchical commands. Our system is fundamentally different due to the way the user interacts with the menu.

2.3 Current outdoor capture techniques

There are a number of techniques to construct large outdoor graphical models. A simple technique to model an existing building is to measure up and record the dimensions of an entire building with a tape measure for example. This is very inefficient and time consuming because there may be inaccessible features, and the user must switch from indoors to outdoors at each iteration to enter the model and then verify its accuracy. Faster methods like laser scanning or multiple cameras (such as Façade in (Debevec 1996)) can be used to recover models, but tend to have large quantities (sometimes millions) of wasteful facets, and occluded objects will not be seen by the camera.

3 Outdoor model construction

We propose our novel CSG primitive based approach as a way of interactively allowing users to build models outdoors, without the limitations of the other discussed methods. Although there are other limitations introduced with this process, they are different from the other two methods outlined earlier, and so the user now has an extra choice when deciding how to capture 3D models. The method is designed to capture reasonably simple objects to the accuracy of the tracking devices, and the user can create highly detailed models as they see fit, while keeping others simple if desired.

3.1 Building construction example

Our first application example for Tinmith-Metro is the modelling of large outdoor structures. An example of how to construct an object model is detailed to examine this modelling technique. This example models a school building, which is a round shape, with an air conditioning tower on the roof, and large windows on the side. The building is neither a box nor a cylinder, and so requires different primitives for modelling.

System start up – The user dons the wearable computer, HMD, and pinch gloves. The user then starts Tinmith-Metro and performs the calibration of the trackers.

Create perimeter walls – The overall top down outline of a building must first be specified. In the example building, there are 32 facets, but the user will only define the outline of the building with 10 planes for simplicity. The building is approximately a cylinder, but not similar enough to use the real cylinder primitive. To create the outline, the user creates infinite planes to mark each wall. Each plane is created by the following: 1) the user positions themselves to look down the edge of a building wall, at any convenient distance, 2) the user places the eye cursor along the wall edge, 3) the user selecting the menu option with the gloves to create a right facing wall, and 4) the right facing wall (an infinite plane) is added to the virtual world intersecting the eye cursor and perpendicular to the image plane. This wall cuts the entire infinite world in half, in the same way as the real wall does, with the left being inside the building, and the right being outside. By walking around the building and marking each plane, the user is carving away sections of the infinite space and defining the volume of the building. Eventually, when the user has completely circled the building, the perimeter of the volume is no longer infinite, and is now closed. The final result is a 2D bounded perimeter as shown from the top down view in (1) of Figure 2. This figure shows the very long planes that created the bounding volume.

Create floor and roof – To complete the first solid shape of the model, the 2D perimeter is constrained with a roof and floor. These are created by the user looking toward the centre of the building and creating a default floor at zero metres and a default roof at three metres.

Create solid object – The infinite planes are all separate and do not currently form a proper solid object. To form a solid object, the CSG intersection operation is selected from the Tinmith-Hand menu with the pinch glove. Once the operation has been initiated, the renderer draws a preview of current model. The user interactively corrects the height of the roof by lifting or lowering it to match the correct height of the building; and this is verified by using the registration of the virtual object against the physical building. When the roof is in the correct position, the

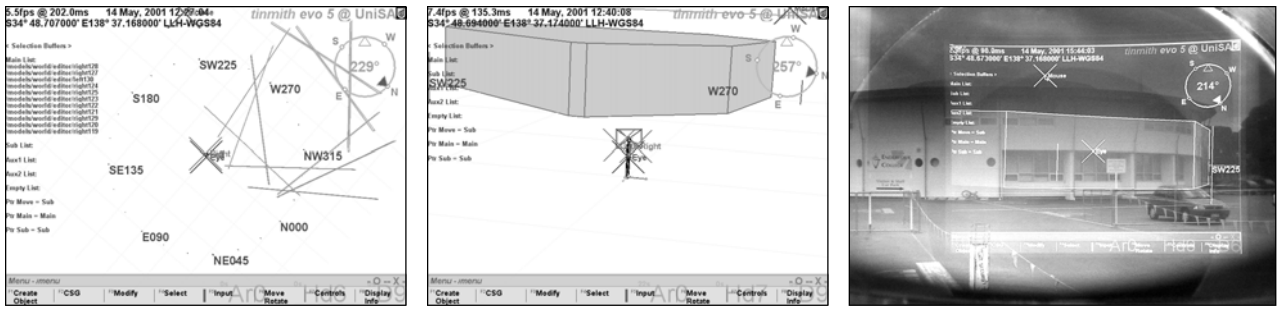

Figure 2 - Tinmith-Metro Demonstrating Various Stages of Construction of 3D School Model
(1) Satellite view of infinite planes defining building volume (2) Solid round shaped building, (3) Immersive HMD shot of school, facing south

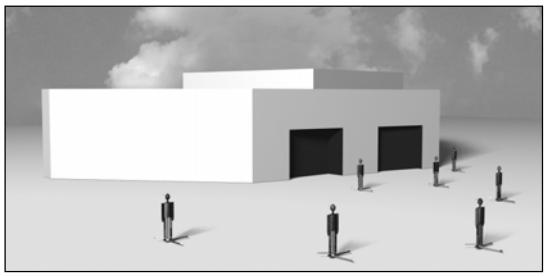

Figure 3 - Final rendered output of 3D school building model, captured using new techniques demonstrated in Tinmith-Metro

intersection operation is committed with another glove menu selection. The model of the building is now at the stage depicted in (2) and (3) of Figure 2.

Create air conditioner tower – The air conditioning tower may be added in a number of ways. One method uses a default cylinder primitive, scaled and moved into position. A second method is to use the infinite planes technique to create a completely new shape. In this example, the user will use the infinite planes technique and the same CSG intersect operation, but this time the tower roof is higher, with a reduced width.

Combine two objects together – Currently, portions of the air conditioner tower exist inside the building object. The tower contains internal facets that are not visible and wastes graphics resources. Using the CSG union operation, the system can merge the tower and building objects into one object, removing the internal facets and simplifying the model, producing a single object.

Create windows – The next task for the user is to add coloured depressed windows into the building, which is done by carving into the building. First the user creates a box shaped object to use as a tool, whose profile matches that of the window, and has a depth of at least the window's depression. The box object tool may be created using a prefabricated model or constructed by the user. The user positions themselves to be able to see the window easily, and the CSG difference operation is selected. Using the gloves, the user pushes the box object tool into the building, in the same way that a cookie cutter is used to remove portions of dough. The box object tool is pushed into the wall until it is the same depth as the window depression. The facets cut into the building are coloured blue like the window, and are also indented into the shape, they are not just surface modifications.

3.1.1 Results

The user has just created a model of a building that is approximately round, along with an air conditioning machinery tower, and carved in windows, shown as a ray traced image in Figure 3. The user may continue to model the building to any level of complexity that they desire, depending on the requirements for the model.

The accuracy of the objects created with this system is largely dependent on the tracking hardware used, and the amount of care taken by the user to accurately enter the information. For position tracking, the Trimble Ag132 has an accuracy around 0.5 metres. For orientation tracking, the IS-300 has a resolution of 1° static and 3° dynamic accuracy, with models measured as close to the building as possible (without degrading the GPS) are the most desirable.

3.2 Street furniture example

A second application example for Tinmith-Metro is to position models of typical community infrastructure, "street furniture", such as park benches, rubbish containers, and street lamps, located at our university campus. In this second example, the user operates a customised menu structure in Tinmith-Metro to position prefabricated models of smaller street furniture items. These models were created using NewTek LightWave and converted into the custom Tinmith file format.

Create grass area – The user creates a grass area by using the infinite planes technique to mark out a perimeter. The area the user is marking is between numerous campus buildings, and so the user approximates the area with several planes. A special "create grass" menu operation is selected, and then floor and roof are both created at default heights and intersected with the perimeter. The resulting object is a grass slab that is 5 cm thick. The purpose of the grass is to supply a background for the objects, and so accuracy or shape is not a concern.

Place down objects – The user moves around the area, standing near the real world objects. The user has previously modelled the required objects such as lamps, rubbish bins, benches, and trees on a desktop system, at the desired accuracy. By using the glove and menu, the object to place down is selected and then instantiated into the modelling environment.

Placement defaults and adjustment - By default, an object is placed one metre in front of the user, and oriented away from the image plane. This allows the user to immediately place an object at the correct orientation and position. The user may then manipulate the object if desired. Image plane techniques are inappropriate for the rotation of an object about the Z (heading) axis when in immersive view; therefore the top down map is used instead. If an object is not the correct size, it can be easily scaled to size using two handed manipulation techniques.

3.2.1 Results

After the previous process is complete, the user has captured a model depicting the grass area, with various

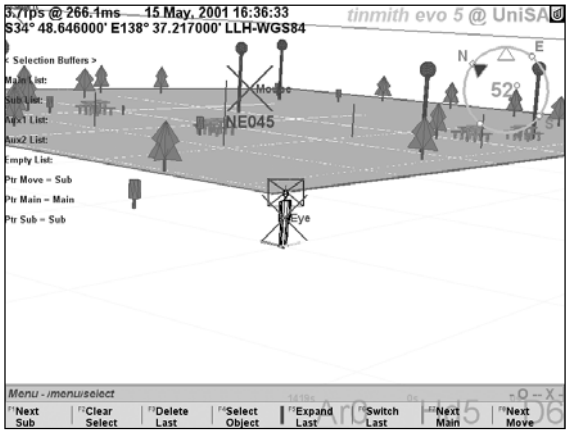

Figure 4 - Outdoor Furniture Placement Screenshot

items of street furniture laid out on top. This model can be viewed immersively or with orbital view in the system (as in Figure 4), or displayed on a separate VR or desktop system indoors.

4 Glove based menu system

The menuing system of Tinmith-Hand provides the user interface to a fully functional 3D modelling system, supporting object hierarchies, CSG, and editable transformations, without the use of a keyboard or traditional mouse. Although the system has a trackball attached, it is preferable to avoid using this device, as the hands are used for the object manipulation tasks. We found that while the trackball was functional and used to start the system up from X Windows, it was more difficult and less intuitive to use during modelling.

The menu options, shown in Figure 5, are presented in a transparent green dialog box at the bottom of the display, which can be repositioned if desired. We used transparency, allowing the user to see through the menu in order to reduce visual clutter caused by the menu boxes. The menu colours and transparency are dynamically changeable.

Each menu option is assigned to a finger on the gloves. To select an option, the user touches the matching fingertip with the thumb tip. For example the CSG option would be selected if the ring finger (LF3) and thumb of the left hand were pressed together. To indicate a selection, the user must hold the press for a short period of time to eliminate key bounce problems or accidental brushing of the glove. When the press is complete, the system beeps and then moves to the selected node in the menu hierarchy. The system then can execute an action at this node if required. In addition, Tinmith-Hand may present a new set of options or return back to the top level of the menu structure if the operation is complete. By pressing any finger on the palm of the glove, the menu returns back to the top level.

The menus do not float in the 3D world like other VR menus such as (Bowman 2001) and (Mine 1997), since we feel that these menu options should always be visible during the graphical object creation task. The menus are fixed to the screen, and designed to aid and focus the user on the task of graphical object construction.

Currently, the menu contains over 100 nodes arranged into a hierarchy, with a maximum of 8 choices per level. Some operations in the system require multiple steps to complete, and hence go deeper into the menu at each step, but most can be started within 2 clicks of the root menu. The 8 choices per level could potentially be expanded with multi-modal interfaces employing both gestures and voice recognition.

4.1 Tinmith-Glove

Wearable computers supporting interactive augmented reality applications require input devices to issue commands to the computer, and 3D tracking to manipulate graphical objects. Tinmith-Glove is our custom built, low cost set of pinch gloves to support command entry and hand tracking. We present the design and implementation issues for our gloves to help others construct inexpensive wearable input technology.

4.2 Current technology

Currently, there are a small number of products on the market that allow users to interact with virtual environments. Each of these have different uses, and many have a cost measured in thousands of dollars per unit.

Two gloves were considered before the start of this project, as they performed similar operations to those desired. The FakeSpace PinchGlove (FakeSpace Labs 2001) contains electrical sensors at each fingertip to measure touching, while the VTi CyberGlove (Virtual Technologies 2001) uses bend sensors to measure finger positions, designed mostly for motion capture. For our application, finger tip to thumb tip touching detection is required, and so the PinchGlove with conductive sensors would be the most appropriate.

The Tinmith-Glove was designed to allow the use of glove based input technology to support virtual and augmented reality applications. These gloves would perform similar pinching tasks as the PinchGlove, while at the same time allow us to customise the placement of the sensors, add new ones to increase the functionality, and have a low price that makes the technology available to anyone. Our research environment requires a flexible hardware implementation that makes changes simple.

4.3 Construction

The glove is based on a typical gardening glove that loosely fits the hand. A correct fit is important; the glove may become damaged during removal if it is too tight.

Detection of finger presses is by the completion of an electric circuit; a conductive surface is required on the tips of the thumbs and fingers. Special flexible metallic tape was acquired from a hardware store, which is normally used to adhere reflective insulation inside the roofs of houses. This tape is conductive on one side and sticky on the other. Pieces were cut out and placed over the fingertips. In the first version of the glove, the tape was wrapped all the way around the fingertips, but on the

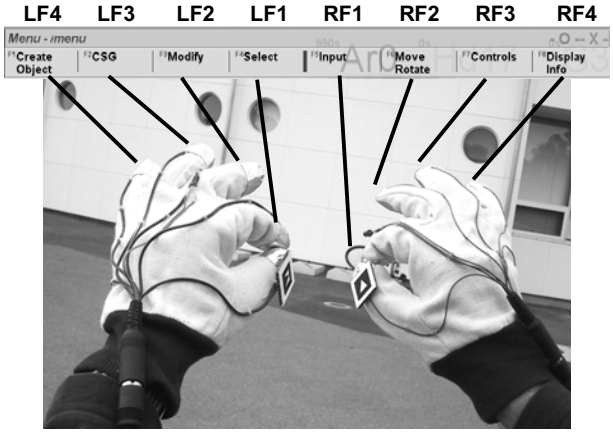

Figure 5 - Tinmith-Gloves with metal pads, tracking targets, and top level menu mappings used with the Tinmith-Hand user interface

second design, only the places where pressure is applied had metallic surfaces. This was done to minimise the amount of area that was conductive, preventing problems where fingers would falsely contact each other.

Wires were placed onto the edge of the metal pads, with another layer of tape placed on top to secure it. The tape layers were fused together by running a hot soldering iron over the surface, melting the sticky backing and bonding the metallic layers together. These wires were run to an 8-pin connector on the wrist, with hot glue securing the wires and connector to the glove. During extensive outdoor use there have been no breakages so far.

Cables are used to connect the gloves up to a processing box, which interfaces to the laptop via a serial port. A Parallax (Parallax 2001) Basic Stamp BS2 microcontroller performs this processing. The BS2 features 16 I/O pins, a serial port, EEPROM, and voltage regulator. Hardware development time is low because the MCU comes fully integrated and only requires a power supply to start using it. A simple interface prototype board was built in order to connect the glove cables up to MCU, with some resistors added to protect against short circuits.

The control program is written in a special high level language for the BS2. This language allows the developer to write programs in a Basic-like language to use the I/O pins and the serial port. The control loop applies voltage to each finger one at a time, and polls for which pad the voltage is detected on (thumb, palm, or none). If this value is different from the last check, then a single byte is sent to the serial port indicating that the finger has changed state, along with the new location of the finger. This polling process is performed 30 times per second for each finger, ensuring that quick presses are accurately captured.

In the laptop, the Tinmith system reads the serial message and then converts it into an internal event that is made available to other objects. These events are similar to those emitted by a keyboard, and so Tinmith code previous written for desktop devices can be modified easily to support the Tinmith-Glove. Tinmith performs debouncing to remove small false contacts that are generated as the fingers and thumbs are pressed together.

Currently, the glove supports both finger and palm pressing, although others could be added easily. We envisage a variety of different user gestures. For example, in the ARquake system (Thomas 2000), which used a plastic gun as a prop, it would be possible to replace this with a gun or fist gesture, thereby alleviating the need to carry any extra hardware.

4.4 3D hand tracking

Hand tracking is the traditional method for 3D object manipulation in immersive virtual environment, but this tracking is customarily expensive and non-portable. With the advent of fast processors and inexpensive video cameras, we have used pattern recognition to track fiducial markers. We utilised the freely available ARtoolkit system (Kato 1999), which is a set of generic computer vision libraries. Using a single camera, the ARtoolkit library is able to resolve a full six-degree of freedom tracking solution for multiple targets present in the video frame. The fiducial markers mounted on the thumbs are simple 2x2 cm cardboard squares with a black outline and a pattern in the centre.

With our P2-450 laptop and USB camera mounted on the head, we achieve capture rates of around 5-10 fps with only 20% CPU usage, so the system is feasible on most modern machines. The quality of the tracking was also excellent under natural light, with minimal false detections caused by the environment. While the overall position of the tracking is good, the orientation values are quite inaccurate (jittering over 10°-20° angles), and hence not a complete solution, but one which is workable for simple cases.

The placement of the fiducial markers on the hands is important as we need to ensure they are visible at all times to the camera. We first thought to place the targets on the back of the user's hand, but we quickly realised that being large, the hands easily fill the field of view of the camera. It was decided that fingers would allow finer motor control and more movement. The ends of the index fingers were dismissed as they moved too much during the selection of the menu option with the index finger and thumb. We noticed that the thumb did not move much during a pinching gesture however. People tended to bring their fingers down to meet their thumb as opposed to bringing the thumb up to meet the fingers. As a result, the targets were placed on the top of the thumb. As another option, the index finger could be used for one handed cursor movements, with the cursor attached to the dominant hand index finger, and the selection of the menus with the other.

5 Pointing and selection techniques

With Tinmith-Hand, the user has a choice of four input devices for pointing and selecting. Some are 2D, like the trackball and eye cursor, while the finger tracking is 3D based. The user chooses whichever device is appropriate at the time for the particular control or selection task. Traditional desktop applications only use one device, or merge the devices to all be the same.

5.1 Hand based finger tracking

Tinmith-Hand is designed to support applications that interact with graphical objects and enter spatial information. We chose hand gestures to be the main interaction method for the user interface as they seemed to fit well with the natural operations of the CSG modelling system.

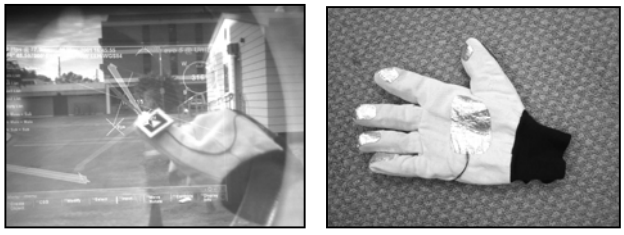

Figure 6 - (1) 3D Cursor Mapped On To Hand
(2) Rear view of glove showing metal pads and wiring

Given the location of the hands from the markers, the system overlays registered 3D axes, as shown in (1) of Figure 6. At the same time, a 2D flat cursor (similar to a mouse cursor) is overlaid on top. The cursor is placed in a desired location by movement of the user's hand. When the user activates selection using the menu and gloves, a ray is fired into the scene from the 2D cursor, and the first object hit is selected. Although 3D coordinates for the hands are available, no 3D based interaction techniques have been implemented at present.

5.2 Eye cursor

The eye cursor is fixed to the centre of the display, and is controlled by the user rotating their head to point to different objects in the world. We have found the eye cursor to be very useful during the construction mode. In particular for when looking down a wall, it is used to specify the direction of the infinite plane, being coplanar with the wall. Objects may also be selected with this mode, the user aims their head at the object, and the ray fired is tested for intersecting objects.

5.3 Handheld trackball

The handheld trackball device has been logically attached to a 2D cursor. The trackball is operated in a traditional manner for cursor movement and selection, and is also required to start the software from the X window manager. We have found the trackball useful for debugging purposes, but it prevents the user from using the pinch gloves, and therefore the least effective.

5.4 Object selection

When a user performs a pick operation on a graphical object, the system determines the closest polygon under the cursor. When a polygon is selected, the simplest object is chosen, but the user can traverse up the hierarchy to select more of the model if desired. Every polygon and object in the scene exists in the world model hierarchy, many objects are also children of other objects, and are represented using a file system notation. For example, the hand of the human avatar stored in the scene graph is represented as /human/left_arm/lower/wrist-/hand.

5.5 Selection buffers

For CSG operations, users are required to select multiple objects, operate on them independently, and then combine them together to produce a final object. One solution is to repeatedly select and deselect objects as required, but selection is tedious and error prone. As a result, rather than having the ability to select just one object and operate on it, the user operates on collections of objects in the selection buffers, which are very similar to traditional clipboards. In any particular selection buffer, the user may place multiple selected objects. The user is able to switch between selection buffers and put different objects into different buffers. CSG operations are performed between two selection buffers at a time. For example, a user may intersect all the planes in buffers A and B while moving only the planes in buffer B, the result going into A. Later, the user can place some different planes into buffer C, rotate them, and then intersect them with the previously existing result (this is how the construction of the building was performed in the example).

5.6 Object transform techniques

Tinmith-Hand supports a number of image plane techniques for manipulating (move, rotate, and scale) objects in the environment. These image plane techniques require one or two input cursors via the four input devices (one and two handed finger tracking, eye cursor, or hand trackball) and a selection buffer to operate on. The input device and the selection buffer are selected by the user via the menus. It is important to realise that these operations are only 2D based, and so operations are performed only in a direction perpendicular to the camera direction.

6 CSG modelling system

At the centre of the modelling system is the CSG engine. CSG allows the construction of complex 3D graphical shapes using only a small number of primitives. It uses the same CSG concepts from graphical software like ray tracers (such as POV-Ray (POV-Team 2000)) that traditionally support only mathematically simple objects such as infinite planes, spheres, and objects that can be described by an equation. This limits the complexity of models that are possible, as things like triangles are not possible to define with a surface equation. As a result, by combining these primitives together using CSG, it is possible to describe new ones.

Each primitive has an outside and inside, and can be tested if an object, or part of it, is inside or outside another object. The three fundamental CSG operations (based on set theory) are shown in Figure 7.

As our fundamental primitive, planes are infinite objects that have a front and back face, (defined by the surface normal) and the space behind the plane is defined to be 'inside'. Hence, if 6 planes, all perpendicular to each other, are intersected, they form a closed 3D box with an 'inside'.

Complex shapes may be built up using infinite planes, as was shown in the house example. The example only dealt with the case of a convex shape however. A concave shape is one in which there are holes or other indents in the surface. Using a set of infinite planes, it is not possible to model a concave shape such as a T, L, or donut shaped building (as shown in Figure 8) using a single CSG operation.

A concave building can be created by breaking the problem down into stages. The following is an example showing how to construct an L shaped building:

1. Create a bounding box representing the entire space used by the building. Use infinite planes and CSG

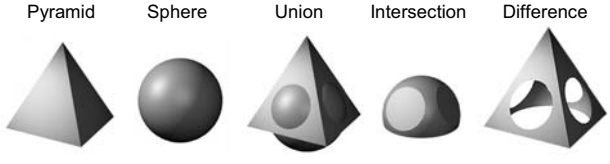

Figure 7 - CSG Primitive Operations

intersect to produce this box, and store it in a selection buffer.

2. Create a second box representing the space used by the hole we want to remove. Use the same methods as previous, and store in a different selection buffer.
3. Activate the CSG difference between the two buffers created previously. The CSG engine will show the resulting object and allow the user to fine tune the locations of the objects until everything is correct. The user commits the result, and the L shaped building is now modeled.

This process can be repeated as many times as necessary to carve out other parts of the building such as windows, tunnels, garages, bridges, and donuts.

6.1 CSG engine implementation

In order to support applications such as Tinmith-Metro, a complete CSG engine was implemented. This engine takes in a series of object meshes from the object hierarchy, and calculates the result in real time to allow the user to interactively view and modify the CSG operations.

As an example, given two solid cubes A and B, they will each contain 6 facets, and be a convex shape. A difference operation A – B (carving one away from the other) requires the objects be subdivided, with each polygon being used to cut every other polygon in half. This produces a complex mesh that is then processed, throwing out facets that do not fit the set operation. The subdivision is then reversed to join back facets that were not altered, and the result is ready to be rendered.

This process is computationally expensive, as whenever one of the source objects moves, the models need to be newly subdivided, processed, and simplified. Given simple cubes, cylinders, and other objects, the system runs in real time at interactive frame rates. However, given two complex spheres with hundreds of varying facets each, this process can slow down to the point where it is unusable.

There are many methods for resolving CSG operations, using rendering hardware, ray-tracing techniques, or voxels, but the required hardware cannot be used outdoors, does not run at real time rates, or does not preserve the original facet mesh structure.

7 Tinmith system

The Tinmith-Hand interface and Tinmith-Metro application are part of a larger AR system, Tinmith-evo5. The Tinmith system is built up of both hardware and software, using off the shelf products and custom built components for our research, as some of our needs can not be met with existing technology.

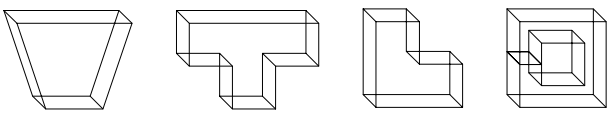

Figure 8 - Convex Trapezoid, and Concave T, L, and Donut Shapes

7.1 Hardware

The wearable computer system as shown in Figure 1 is based on a Gateway Solo P2-450 laptop (64 mb RAM, ATI Rage OpenGL) mounted on a hiking backpack. An Intersense IS-300 hybrid tracker performs orientation sensing. Position information is gained from a Trimble Ag132 GPS, with an accuracy of 50 cm, varying with conditions. The display is a Sony Glasstron PLM-700e monocular SVGA display. A large 12V battery powers the various trackers, as well as the small LCD television on the back for debugging and spectators to view. A SuperCam WonderEye USB video camera is used to provide images for the hand tracking system.

The laptop runs RedHat Linux 7.0 with kernel 2.4 as its operating system, including the standard GNU development environment. XFree86 v3.3.6 is used for graphics, as it does hardware accelerated OpenGL using Utah-GLX. The performance of the older ATI Rage chipset is adequate for our current needs. Currently, we use USB for our camera as there is no other way to capture video with a Linux laptop.

7.2 Tinmith-evo5 software architecture

The menus and interaction techniques are only the application level interfaces, those that communicate directly with the user. To implement a system of this magnitude, a modular architecture with an appropriate framework must be provided.

Although sharing the same name as its predecessors (Piekarski 1999, Thomas 1998), Tinmith-evo5 is a completely new design and implementation, written in C++ to maximise speed and efficiency. The complete details of the system are not presented in this paper, but basically, the overall goal is to process data from input devices, make changes to the internal state of the system, and then render images to the user's display. At the same time, speed and practicality of implementation were major goals, as we wanted to use the system in real world applications. See (Piekarski 2001) for more information.

Input devices and trackers are firstly abstracted away into objects, which are then made available for other objects to read in and process. When new tracker information arrives, the class notifies listening objects using a callback mechanism, which then allows the listener to recalculate its internal values based on this new information. The changes are propagated throughout the system until all objects have been updated, and then the display is rendered when the system is idle. This flow of data through the system allows us to write small component based objects which handle one task, and then glue them all together. By default, communication is done using fast function calls in a single process, and there is no threading, shared memory, network, system call, or RPC overhead. As a result, the propagation of values through the system has little effect on the CPU, leaving resources available for rendering and processing tracker updates. If desired however, network serialisation objects can be plugged in to distribute values over a network, but the default is to run locally, maximising performance.

A key feature of the system is the object repository, which is a place that objects can be stored to allow other objects to access them easily. A standard way of retrieving objects is making all objects accessible through many global pointers, but it is well known that this method becomes unworkable for hundreds of objects connected together in the system. Our solution is the creation of an object storage system that stores pointers to all the objects in the system, as they are created. Each class that can be stored implements methods to perform serialisation and callbacks, using special preprocessor macros and code generators. To reference an object in the repository, a string resembling a file system path name is used. The forward slash is used as a delimiter, and this allows us to organise the classes into categories in a tree structure. Other objects wanting a reference to an object simply call a method that retrieves the object pointer and returns it. Our store is as efficient as a global variable as it is mostly accessed only at object creation, but it is also a dynamic run time system that can be changed as the system is running.

The object repository contains the menu system, the tracking system definition, the graphical objects, and all other system objects. At system start up, the object repository firstly reads through the physical disk file system (which matches the object storage paths) and reads in object definition files. These files contain start up values for various classes, and are created into instantiated objects, which are used to configure the environment for the rest of the system. Once completed, other portions of the system can then execute to configure the particular application and task, creating the necessary objects. Since each object may be serialised, we may snapshot the state of the system and reload it, or distribute it over a network.

The menu state objects are created based on the definitions contained on the disk, which are then read in by the menu object to control the state of the application. Since the menu is decoupled from the application, a translation layer is used to convert actions into method calls. Our trackers are stored in the system, and by using a feature called an object symbolic link, it is possible to switch the objects supplying tracker data to other objects completely transparently. Using this, it is possible to implement a patch board of tracking devices and switch them for a variety of tasks. We believe that the ability to transparently change input devices and coordinate systems is a key to making the interaction techniques of Tinmith-Hand feasible.

The rendering system also uses the object repository, and is a full hierarchical modelling system similar to SGI's Inventor, supporting a scene graph, as well as transformation nodes controlling the movement of all the child objects. Each object in the scene graph is stored in the object repository, allowing polygons and objects to be easily referenced using path names by other code, so they can be controlled. Also, a tracking device may be attached to any node in the scene graph, and the device will automatically apply its movement to the node via callbacks, moving all children. Apart from rendering objects, we also use the hierarchical renderer to resolve tracker data so that it can be rendered accurately. The user is modeled as a human 2 metres tall, with 15 separately movable parts, and we attach our trackers to this model. The ARtoolkit camera is modelled relative to the head of the body, and by traversing through the scene graph we can determine the targets relative to world or head coordinates easily.

Another use of this human model would be to implement a GoGo arm using simple orientation sensors. Imagine attaching orientation trackers (such as the TCM2) to the upper and lower portions of the user's arm, the system would then apply the tracker's orientations to the shoulder and elbow joints of the human model. From these angles, and the known length of the user's upper and lower arm, the system would be able to determine the location of the base of the user's hand.

It is important to understand that the architecture used for this system was originally designed to support AR applications, although it could be used to support a wide variety of systems where data flow and objects are used, such as: VR systems, 2D GUI applications, and constraint systems. There is a large amount of code present in the system to handle 3D rendering, user interface components such as transparent dialog boxes, and transformations for the Earth's various coordinate systems.

8 Conclusion

This paper has introduced the Tinmith family of systems, designed to allow complex outdoor user interaction in augmented reality environments. Using the Tinmith-evo5 software architecture, we have implemented the Tinmith-Hand AR user interface, and used this as a foundation for the Tinmith-Metro city modelling and capture application.

Using the techniques described in this paper, we can control the real-time CSG engine and produce models of arbitrary complexity easily, and verify them as they are being created. These models can then be saved and viewed later by others on desktop or virtual reality systems.

The ability to capture models outdoors has a wide number of uses, which we are only now beginning to explore with our system. These have applications for a variety of different areas, such as the GIS, environmental, surveying, and architectural fields.

9 Acknowledgments

The authors would like to especially acknowledge the work of Arron and Spishek Piekarski, who both helped in the construction and design of the glove and HMD. Thanks also to the Division of ITEE and Defence Science Technology Organisation (DSTO).

10 References

Azuma, R. (1997): *A Survey of Augmented Reality.* Presence: Teleoperators and Virtual Environments, Vol. 6, No. 4, 1997.

Azuma, R., Hoff, B., Neely, H., and Sarfaty, R. (1999): A Motion-Stablized Outdoor Augmented Reality System. In *IEEE Virtual Reality,* pp 252-259, Houston, Tx, 1999

Bowman, D. A. and Hodges, L. F. (1997): An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments. In *1997 Symposium on Interactive 3D Graphics,* pp 35-38, Providence, RI, Apr 1997.

Bowman, D. A. and Wingrave, C. A. (2001): Design and Evaluation of Menu Systems for Immersive Virtual Environments. In *IEEE Virtual Reality 2001,* pp 149-156, Yokohama, Japan, Mar 2001.

Debevec, P. E., Taylor, C. J., and Malik, J. (1996): Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. In *23rd Annual Conference on Computer Graphics,* pp 11-20, New Orleans, LA, Aug 1996.

FakeSpace Labs (2001): *Pinch Gloves.* URL - www.fakespacelabs.com/products/pinch.html

Feiner, S., MacIntyre, B., and Hollerer, T. (1997): A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In *1st Int'l Symposium on Wearable Computers,* pp 74-81, Cambridge, Ma, Oct 1997.

Hinckley, K., Pausch, R., Goble, J. C., and Kassell, N. F. (1994): A Survey of Design Issues in Spatial Input. In *7th Int'l Symposium on User Interface Software Technology,* pp 213-222, Marina del Rey, Ca, Nov 1994.

Julier, S., Lanzagorta, M., Baillot, Y., Rosenblum, L., Feiner, S., and Hollerer, T. (2000): Information Filtering for Mobile Augmented Reality. In *3rd IEEE and ACM International Symposium on Augmented Reality,* pp 1-10, Munich, Germany, Oct 2000.

Kato, H. and Billinghurst, M. (1999): Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *2nd IEEE and ACM International Workshop on Augmented Reality,* pp 85-94, San Francisco, Ca, Oct 1999.

Mine, M., Brooks, F. P., and Sequin, C. H. (1997): Moving Objects In Space: Exploiting Proprioception In Virtual-Environment Interaction. In *ACM SIGGRAPH 1997,* pp 19-26, Los Angeles, Ca, Aug 1997.

Parallax (2001): *Basic Stamp BS2.* URL - www.parallaxinc.com

Piekarski, W., Gunther, B., and Thomas, B. (1999): Integrating Virtual and Augmented Realities in an Outdoor Application. In *2nd Int'l Workshop on Augmented Reality,* pp 45-54, San Francisco, Ca, Oct 1999.

Piekarski, W. and Thomas, B. (2001): Tinmith-evo5 - An Architecture for Supporting Mobile Augmented Reality Environments. In *2nd Int'l Symposium on Augmented Reality,* New York, NY, Oct 2001.

Pierce, J., Forsberg, A., Conway, M., Hong, S., Zeleznik, R., and Mine, M. (1997): Image Plane Interaction Techniques in 3D Immersive Environments. In *1997 Symposium on Interactive 3D Graphics,* pp 39-43, Providence, RI, Apr 1997.

Pierce, J. S., Steams, B. C., and Pausch, R. (1999): Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments. In *1999 ACM Symposium on Interactive 3D Graphics,* pp 141-145, Atlanta, Ga, Apr 1999.

Poupyrev, I., Billinghurst, M., Weghorst, S., and Ichikawa, T. (1996): The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. In *9th Int'l Symposium on User Interface Software Technology,* pp 79-80, Seattle, WA, Nov 1996.

POV-Team (2000): *The Persistence Of Vision Raytracer.* URL - http://www.povray.org

Stoakley, R., Conway, M. J., and Pausch, R. (1995): Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Conference on Human Factors in Computing Systems - CHI95,* pp 265-272, Denver, Co, May 1995.

Thomas, B., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., and Piekarski, W. (2000): ARQuake: An Outdoor/Indoor Augmented Reality First Person Application. In *4th Int'l Symposium on Wearable Computers,* pp 139-146, Atlanta, Ga, USA, Oct 2000.

Thomas, B. H., Demczuk, V., Piekarski, W., Hepworth, D., and Gunther, D. (1998): A Wearable Computer System With Augmented Reality to Support Terrestrial Navigation. In *2nd Int'l Symposium on Wearable Computers,* pp 168-171, Pittsburg, Pa, Oct 1998.

Virtual Technologies (2001): *CyberGlove.* URL - www.virtex.com/products/hw_products/cyberglove.html

Zeleznik, R. C., Forsberg, A. S., and Strauss, P. S. (1997): Two Pointer Input For 3D Interaction. In *1997 Symposium on Interactive 3D Graphics,* pp 115-120, Providence, RI, Apr 1997.