A TANGIBLE INTERFACE FOR

A VIRTUAL REALITY PRESENTATION AND DESIGN

by Jun Oishi

Master of architecture 1993

Waseda University

Submitted to the Department of Urban Studies and Planning

in partial fulfillment of the requirements for the degree of

Master of Science

at the

Massachusetts Institute of Technology

September 2004

Signature of Author... [illegible]

Department of Urban Studies and Planning

August 13, 2004

Certified by .... [illegible]

Eran Ben-Joseph

Associate professor of landscape architecture and planning

Department of Urban Studies and Planning

Thesis Supervisor

Accepted by .... [illegible]

Dennis Frenchman

Professor of the Practice of Urban Design

Chairman, Committee for Graduate Students

A TANGIBLE INTERFACE FOR A VIRTUAL REALITY PRESENTATION AND DESIGN

by Jun Oishi

Submitted to the Department of Urban Studies and Planning in partial fulfillment of the requirements for the degree of Master of Science at the Massachusetts Institute of Technology May 2004

Abstract

It is important for city planning projects and large-scale building construction to involve stakeholders and feedback their opinion to the project.
This thesis introduces a Tangible Interface for Virtual Reality Presentation and Design (TIVR). TIVR is designed to establish an intuitive, interactive user interface that will enable better communication between architects, city planners, clients, neighborhoods, and stakeholders.
This thesis also evaluates the effectiveness by experimental conditions. The results suggests that TIVR provide a better condition for stakeholders to understand the project and state their opinion.

Contents

ABSTRACT 2

CONTENTS 3

LIST OF FIGURES 5

INTRODUCTION 7

- Demand for realistic presentation 7
- Virtual reality in city planning and architecture 8
- User interface for VR 10
- Image as the target of discussion 11
- Tangible media 11

RELATED WORK 13

- Tangible Geospace 13
- FURP (The Future of Urban Planning) 13
- Sensetable 15
- FishPong 16

DESIGN 18

- Tangible Interface for VR with target-camera method (TIVR 1) 18
- Multi-User Tabletop Urban Walkthrough presentation system (TIVR 2) 21

EVALUATION 28

- Control condition 28
- Tasks 28
- Experimental design 29
- Participants 33
- Procedure 34

RESULTS 36

Performance 36

Interviews 38

Observation 39

DISCUSSIONS 41

Performance advantage 41

Observations 41

CONCLUSIONS 43

FUTURE WORK 44

Design variation tool 44

Tangible user interface for animation camera control 44

Tangible user interface for section and layering 44

APPENDIXES 46

Appendix: A camera control Maxscript module for TIVR 1 and TIVR 2 46

Appendix: B Bridge module source code 51

ACKNOWLEDGEMENT 57

REFERENCE 58

List of figures

FIGURE 1 3D THEATER AT CONDOMINIUM SALES OFFICE, TOKYO 8
FIGURE 2 CAVE, VISIMAX, VISON-DOME 9
FIGURE 3 HEAD MOUNT DISPLAY (HMD) 9
FIGURE 4 TANGIBLE GEOSCAPE ON METADESK 13
FIGURE 5 FURP IN THE CLASSROOM, DUSP MIT 14
FIGURE 6 URP ON I/O BULB 14
FIGURE 7 MICRO CAMERA ON FURP AND PROJECTED IMAGE ON THE WALL 15
FIGURE 8 IP NETWORK WORKBENCH IN SENSETABLE 16
FIGURE 9 COFFEEHOUSE PATRONS AMUSE THEMSELVES WITH FISHPONG 17
FIGURE 10 TIVR 1 SYSTEM 19
FIGURE 11 TARGET PACK (LEFT) AND CAMERA PACK(RIGHT) 20
FIGURE 12 TIVR 2; FOUR CAMERA PHICONS WITH VIEWFINDER IMAGES 21
FIGURE 13 CAMERA PHICON WITH VIEWFINDER IMAGE 22
FIGURE 14 TEMPORARY PANORAMA CAMERA 23
FIGURE 15 MIMIO BAR 24
FIGURE 16 MIMIO PEN 24
FIGURE 17 MIMIO WITH REAR-PROJECTION SCREEN 25
FIGURE 18 TIVR 3 SENSOR 26
FIGURE 19 LE CORBUSIER'S CITROHAN DRAWING 29
FIGURE 20 JOYSTICK CONDITION WITH CITROHAN MODEL AND SITE 31
FIGURE 21 CAMERA ICON ON SITE 32
FIGURE 22 LOGITECH'S EXTREME 3D PRO 32
FIGURE 23 TIVR 2 CONDITION 33
FIGURE 24 TASK EXAMPLE IMAGE AND ANSWER TOLERANCES 35
FIGURE 25 USER'S PERFORMANCE (JOYSTICK CONDITION) 36
FIGURE 26 USER'S PERFORMANCE (TIVR1 CONDITION) 36
FIGURE 27 USER'S PERFORMANCE (TIVR2 CONDITION) 37
FIGURE 28 AVERAGE SCORE (SECOND) FOR CAMERA POINTING 37
FIGURE 29 APPRECIATION OF DEVICES (BEFORE THE TASKS) 38

FIGURE 30 APPRECIATION OF DEVICES (AFTER THE TASKS) 39
FIGURE 31 PARTICIPANT USES TWO PACKS IN ONE HAND 40
FIGURE 32 TUI TABLETOP SECTION VIEWER 45

Introduction

Leff works in the scheduling and logistics department at Bovis. Part of his job involves working with the marketing and project management teams to create proposals for potential clients.

"Often it is the case that construction methodology is not easily explained to project stakeholders, and therefore difficult for them to absorb," says Leff.

The best way for Bovis to communicate its strategy is through visuals – specifically 3D animation. Normal 2D plan views can communicate logistics, but 3D animation creates a greater understanding of the construction site, the equipment that will be used, and how the property will change over time.

By Jill R. Aitoro   www.cgarchitect.com

Demand for realistic presentation

These days, more and more stakeholders are concerned with city planning projects and large-scale building construction. Architects and city planners must spend more time presenting and explaining their concepts and design in order to accumulate outside opinions before finalizing their work.

Of course, computer graphics, movies and slides are useful presentation tools for those projects when the number of interested parties are limited. Plans and designs can be drawn up to satisfy the individual needs and tastes of clients, and presentations can be prepared that focus on explaining the most appealing points. The focus can also be tailored to the site neighbor's interests, as Clients and site neighbors are quite familiar with these presentations.

Many condominium promotion offices are likely to have a small theater equipped with three or four high-resolution projectors. Figure 1 below shows a good example of a presentation for a limited number of clients. In such a theater, computer-graphics presentation video clips can be shown. This sort of visible, easy-to-understand presentation method is also a must for general contractors and architects.

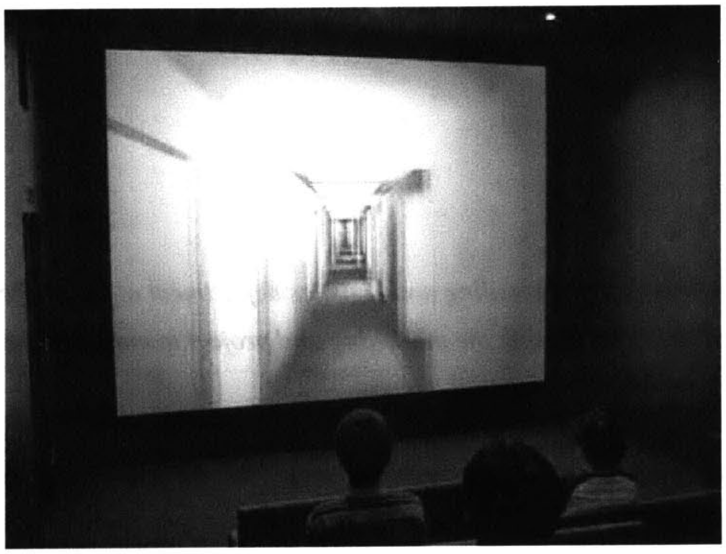

Figure 1 3D theater at condominium sales office, Tokyo

However, in larger-scale projects, there are more stakeholders and involved neighborhoods, and architects and planners have to explain their projects and accumulate opinions, while feeding public reaction back into the design. Since various stakeholders have their own interest in the project, it is architects and city planners obligation to establish a condition for stating public opinion. In order to satisfy these unpredictable demands, Virtual Reality (VR) is more useful than pre-rendered movies or PowerPoint type slides in large-scale project presentation. Once the VR space of the project has been prepared, it can be shown to stakeholders and neighborhoods .

Virtual reality in city planning and architecture

Virtual reality is a technique that was first invented by Ivan Sutherland, thirty-five years ago. "Basically, VR makes the artificial as realistic as the real" [i]. In the fields of architecture and city planning, VR makes the project as realistic as the real, and theoretically the artificial space can be navigated.

With the advances in modern computer workstations and graphic accelerator chips, today's VR systems show realistic virtual architectural and urban spaces. Precise design simulations of buildings and cityscapes can be illustrated. Several different virtual reality systems have been developed; three such systems are CAVE[ii] ,Vision-Dome[iii] and visiMax[iv].

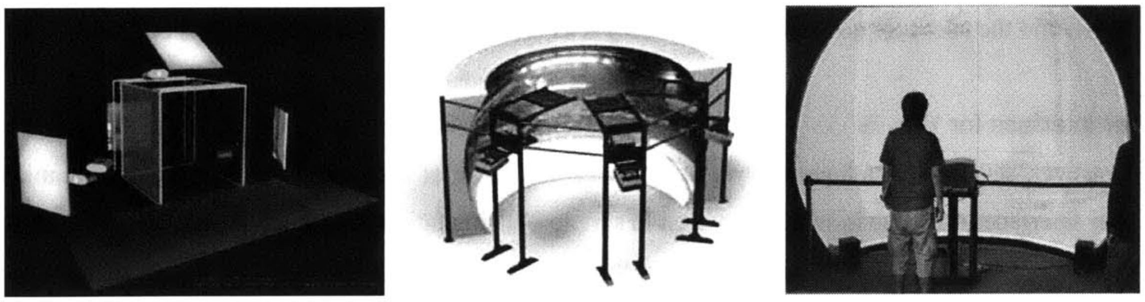
Figure 2 CAVE, visiMax, Vison-Dome

CAVE is a system with a cubic screen and five to six projectors, developed by the University of Illinois. Vision-Dome consists of a lightweight, small-hemisphere screen and fish-eye lens projector simulator system. VisiMax is a large dome-screen simulation theater with six high-resolution projectors. All these tools enable photo-realistic presentations as an immersive and manipulable virtual experience. They present not only a 360-degree view of the entire interior of a future building, but also the tactile qualities of the texture of wood, stone and concrete materials. Some of these systems have less than 1' Field Of View (FOV) / pixel resolution. This resolution is almost equivalent to 20/20 static visual acuity. It is safe to say that the visual quality of these VR systems is appropriate for these presentations.

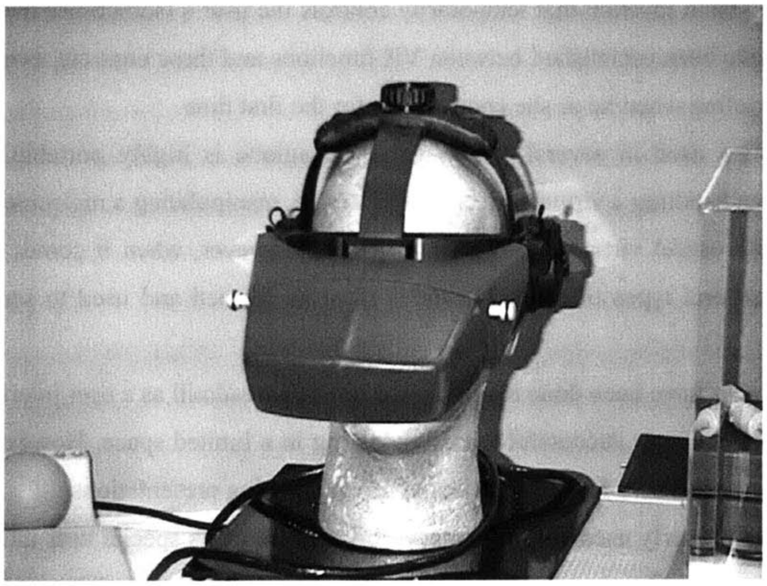
Figure 3 Head Mount Display (HMD)

The Head Mount Display (HMD) is also a popular device for VR systems. An HMD is a goggle-like apparatus with one or two small monitors in front of the eyes. The small monitor screen fills the user's vision and the user sees computer graphic images as if they exist in front of the user's eyes.

User interface for VR

However, several issues have not yet been discussed. One of these issues is the lack of an intuitive user interface. A mouse is sometimes used in simple VR simulations, such as VRML. When the mouse pointer is moved to the right or left edge of the screen, the camera will pan right or left in the virtual space. If the pointer is moved to the top or bottom of the screen, the camera will step forward or backward. The distance between the center and the mouse pointer determines the moving and turning speed. If the pointer is moved slightly upward, the camera will move slowly, and if moved to the top edge, the camera will run at top speed. Thus, the user must pay attention to both the position of the mouse pointer and the camera direction/speed. A mouse is a popular device in the daily operation of a computer, but it is a substitute device when used in VR.

A joystick is a popular navigation device for VR. An experienced Microsoft Flight-simulator pilot exhibits smooth acrobat maneuvering. However, most audience members at a presentation are not familiar with a joystick, and there is usually not enough time to train an audience. Some joysticks have complicated functions like twist control, and hat and speed control levers. If the joystick has a twist function, direction can be controlled by twisting the stick. A hat is a small additional joystick installed at the top of the main joystick that temporarily controls the user's (sometimes the pilot's) view. Since no standards have been established between VR functions and these controls, even a skilled simulator pilot needs a briefing when he or she controls VR for the first time.

Dataglobe is also used in several VR systems. Dataglobe is highly portable and can be carried anywhere. When handling a virtual object – for example, manipulating a molecule on a virtual desktop – Dataglobe is a useful virtual reality user interface. However, when it comes to navigation in the virtual space, several types of special gestures must be learned and used to send commands to the computer.

Several cite studies have been done regarding the use of a treadmill as a user interface for virtual space navigation. They are very successful when navigating in a limited space. However, these devices are too large and too expensive for the purposes of urban planning presentations.

Thus, VR is not properly used as a VR interaction because of its special user interface. VR is mainly used as a VR theater and audience can not interact with virtual space.

In most presentations using VR systems, the city planner and architects control the VR system by themselves or bring an operator to navigate the VR space and do not spend time on training audiences. Even with a VR system, the audience can see more variation in the images, but in most cases they sill remain as part of an audience and do not stand on the same ground that the architects and city planners are standing on.

Image as the target of discussion

There is another problem with regard to using a VR system. Most VR systems are designed for a single user. Only one person can wear one HMD at one time, and even with CAVE and Visimax, the VR space can be seen accurately only when one is standing at the center point of the system. During urban planning discussions, an image of the building or city is the target of discussion. The image must be shared with colleagues. However, with current VR systems, there is no easy way of sharing the image with other users.

These days, presentation quality shows great improvement with these technologies. Stakeholders might understand projects better than before with the help of these technology, but stakeholders still do not have means to explore the project and state opinion which architects and city planners. Stakeholders do not stand on the same ground thatarchitects and city planners do.

Tangible media

In order to satisfy these requirements of urban-planning and architecture presentations, this thesis introduces a new user interface, Tangible Interface for Virtual Reality (TIVR). This system is based on the tangible user interface. Tangible User Interface (TUI) is a growing research area in the domain of human computer interaction.

The theory of TUI and tangible media was first discussed by H. Ishii and B. Ullmer[v], and refers to a concept in which all data or information is manipulated and represented by physical, tangible objects, instead of by graphical user interface and keyboard / mouse-type input devices.

One of the most distinctive features of TUI is the phicon (Physical Icon). An icon represents in a 2D image the functioning of the computer as a metaphor of the usual world; for example, the recycling-bin icon means a file has been deleted, or the folder icon represents a repository. In contrast, a phicon represents digital information as a physical and graspable shape, and the user can control and feel digital information by moving, grabbing and twisting the phicon. With its physical shape, a phicon can unconsciously tell the user what its usage is. A dial-like phicon works as a dial, and the phicon that looks like a clock controls simulating time.

In this thesis, TUI is used for VR camera control in a 3D space. This interface enables people with no computer graphic skill to navigate the VR space.

Several implementations of TIVR are explained in this study. The advantage of this interface over an orthodox user interface is also tested from the perspectives of usability and expressiveness.

Related work

Tangible Geospace[vi]

Tangible Geospace is the first application on the Metadesk, which was one of the first platforms of Tangible interface study. Tangible Geospace allows the user to manipulate 2D and 3D geological and cityscape data by using a phicon (Physical Icon) and a semi-transparent table with a site map projected from behind the table. In this project, the user browses the MIT campus layout plan, and scrolls and rotates the layout plan by grasping the phicon. The user also observes a 3D view of the campus by using activeLens, a physically handy window frame.

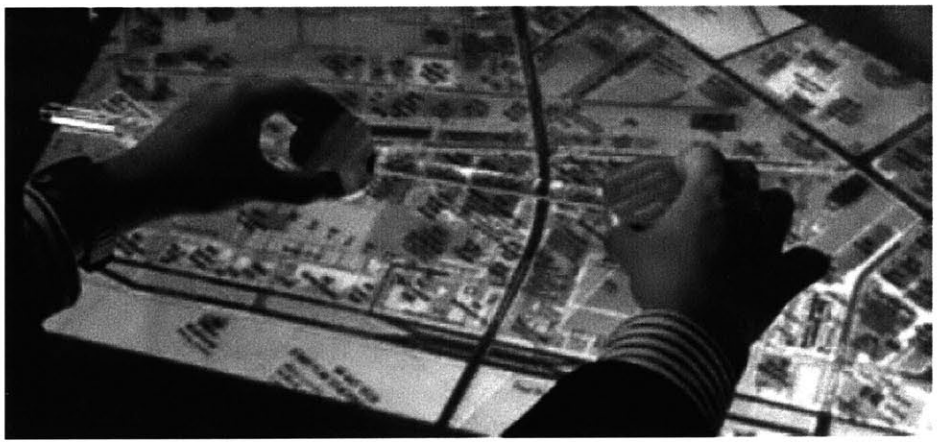

Figure 4 Tangible Geoscape on MetaDesk

Tangible Geospace and Metadesk introduce the idea and advantage of handling city-planning data on an interactive surface.

FURP (The Future of Urban Planning)[vii]

TIVR was also inspired by FURP. FURP is an urban planning support system that is based on the Luminous Planning Table (LPT). The LPT was also developed by the Tangible Media group. FURP focuses on the integration of conventional urban planning methods (drawings, physical models) with digital analysis (sun shade / traffic simulation).

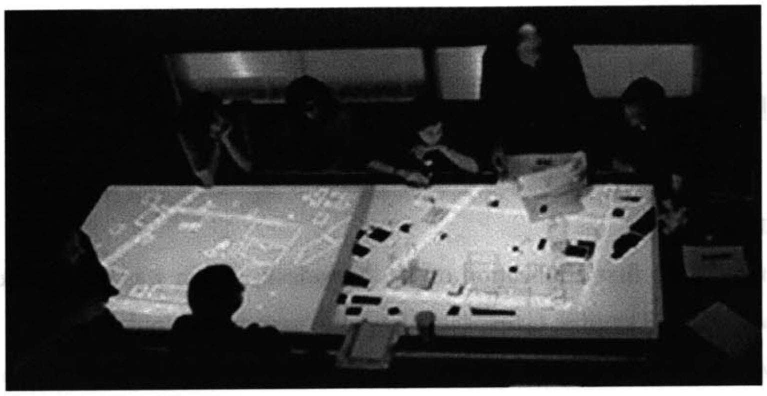

Figure 5 FURP in the classroom, DUSP MIT

LPT has its origin in the input and output (I/O) bulb workbench and urban planning (URP)[viii]. The I/O bulb workbench consists of a projector with a camera installed on the ceiling. The camera detects the movement of the phicons on the table. In URP, the phicons represent buildings. Based on the position data of the phicons, the computer calculates shadow simulation / airflow simulation and projects those results onto the table.

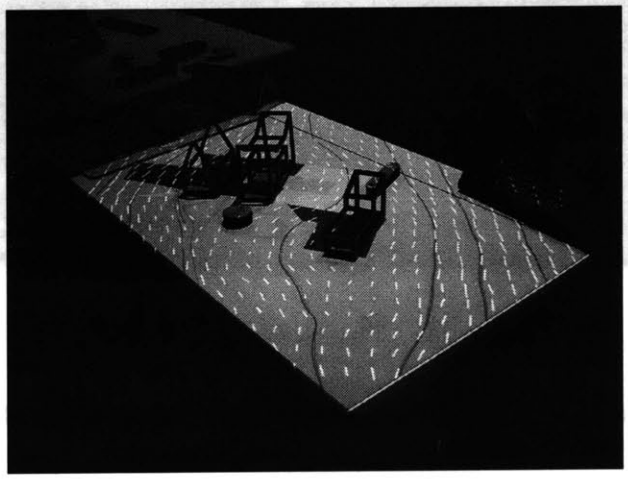

Figure 6 URP on I/O bulb

LPT is an expanded, evolved version of the I/O bulb with several practical simulation features for urban planning. FURP is installed in the classroom of the Department of Urban Studies and Planning at MIT and is also used for daily classwork.

For classwork, a new device, a small video camera that projects the area of focus onto a presentation screen, has been installed in FURP and added to the system. This small camera enables users and the audience to see the specific element that is currently being referred to. The concept of TVIR is based on this small camera.

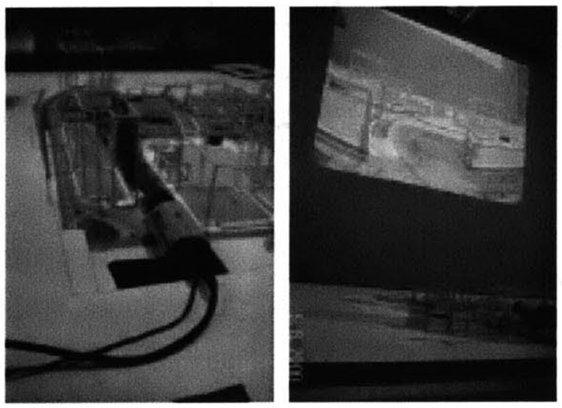

Figure 7 Micro camera on FURP and projected image on the wall

Sensetable[ix] [x]

Sensetable is a device that detects the positions of several tags on the table with high accuracy and high latency. With a projector image projected onto the tabletop, Sensetable gives users a tactile, graspable interface to manipulate the data displayed on the tabletop. There are several versions of Sensetable. The first Sensetable was developed by James Patten of the Tangible Media group in Media Lab. The second version of Sensetable was developed by NTT Comware (Nihon Telephone and Telegram Comware) in collaboration with TMG.

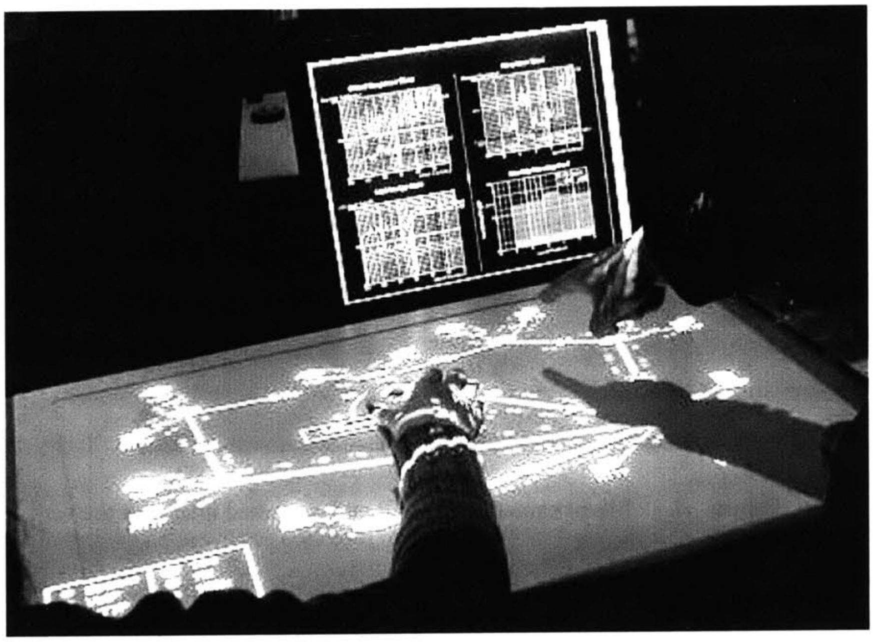

Figure 8 IP network workbench in Sensetable

A Sensetable consists of

An array of antennas

Control circuit

Driver software

Packs

The array of antennas is installed under the table and detects RFID tags in the packs. NTT Comware's Sensetable pack has three RFID tags: two of them are used for position tracking, and third tag is used for a small switch on the pack. The data is transmitted to a PC via a control circuit and USB interface, and the controller PC calculates the coordinates of the tags, and sends the data through the TCP/IP network.

FishPong[xi]

I was fortunate enough to be involved in the FishPong project that was carried out in the MIT MAS-834 class in 2003, and for that project, I wrote code for Sensetable. FishPong is an interactive system designed to stimulate informal, computer-supported, cooperative play (CSCP) in public spaces such as coffeehouses and cafes.

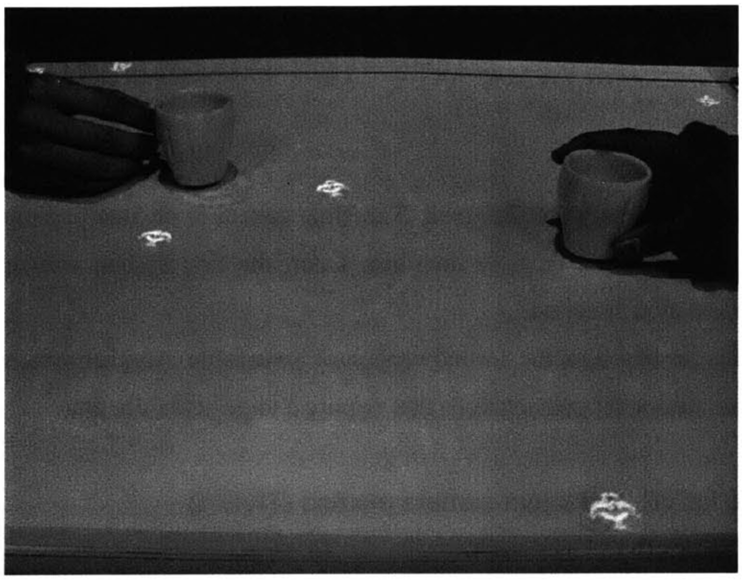

Figure 9 Coffeehouse patrons amuse themselves with FishPong

Coffeehouse patrons can participate in the FishPong game using their mugs and tables. When a coffeehouse customer sits down at a table and puts his/her mug on the table, a ripple appears under the mug and several fish approach the ripple, as if attracted by the mug. The fish changes its direction when it touches the ripple, but if the coffee drinker picks up his or her mug, the ripple disappears and the unfortunate fish falls off the edge of the table. The ripples produced by the mugs and the movements of the fish make a subtle, gentle-speed game of Pong, and coffeehouse patrons sitting at the table are unconsciously involved in the game.

RFID tags are embedded in each mug and a Sensetable antenna is installed on the surface of the table in the coffeehouse. Although the objective and approach is different, some part of TIVR's system had its origin in FishPong.

Design

In this study, three systems were designed. The first system is an interpretation of the most basic camera control method in the tangible interface. Later, the first system was optimized so as to be suitable for a presentation situation.

Both these systems are based on the second version of Sensetable. Another system was also developed, using an ultrasonic sensor for presentations that require a large-scale site plan.

Tangible Interface for VR with target-camera method (TIVR 1)

Target-camera method

Since the 1980s, the most orthodox method of camera control in CAD systems has been to click the camera position and target position on the plan window. "Camera path" and "target path" were used to control camera movement, even when computer graphics animation became popular. The target point and camera point method is still one of the most basic camera control methods in almost every CAD system.

Some graphics workstations have an array of dials (for example, Silicon Graphics workstation's Dialbox) to control camera position, direction and focal length. Other graphic workstations have special trackball devices to control the camera, but these devices are not the standard in CAD and CG systems.

Therefore, in this study the target-camera method was chosen because it is the most popular camera control, and it was decided to combine this method with the tangible user interface.

TIVR 1 Design

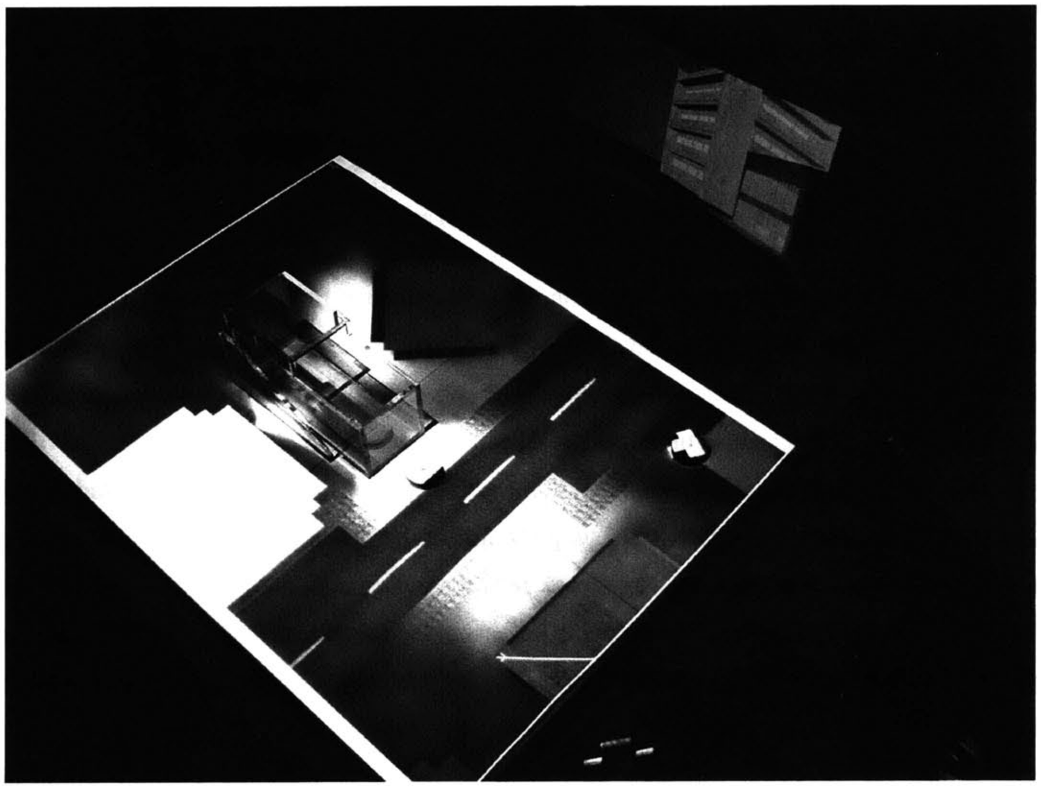

Figure 10 TIVR 1 system

TIVR 1 is based on NTT Comware's Sensetable. TIVR 1's table size is 814mm x 611mm and uses four Sensetable modules. A projector on the ceiling shows the site image on the table.

TIVR 1 also has a building model at the center of the site so that users can understand the relative position of the camera.

Sensetable can track the positions of multiple packs, so I used one pack as a camera position control and another pack for target control. The user moves these two packs to control the position and direction of the camera.

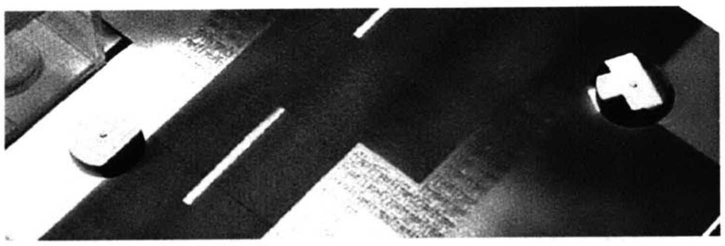

Figure 11 Target pack (left) and camera pack(right)

The image taken by the camera is displayed on the LCD screen in front of the user.

There are several user-interface advantages in this design. The user can see how the movement of these target-camera packs affects the camera image dynamically, instead of sequentially, as in most CAD software. The user can also move the target and the camera simultaneously and this will improve the usability. Of course, the user can enjoy all the advantages of the physical model and site. Another advantage is that if the user needs to simplify his or her task, he or she can move the packs sequentially. The user can place the target pack near the building he or she wants to see, and can then concentrate on the camera pack. This might be easier for the novice user. The user can move the camera first and the target later, or vice versa, or simultaneously. This modeless user interface is one of the great advantages of TUI.

TIVR 1 runs on 3ds-Max. 3ds-Max is a popular computer graphics software application for architecture and urban planning presentations. It also generates high quality computer graphics. Architects and city planners can use TIVR without having to bother with troublesome data-conversion. This is one of the advantages with the on-going project. 3ds-Max also uses the powerful macro language, Max script, and can be a good platform for 3D system development. Max script was used to develop camera control, camera image display and site plan projection.

TIVR 1 consists of two modules. One module of the system controls the camera in 3ds-Max and shows site images on the table and camera image on LCD display.

Another module is the bridge software. This module receives Sensetable's pack position from Sensetable's data server via a TCP/IP network, and sends these data to the camera control module in 3ds-Max via the OLE interface.

Multi-User Tabletop Urban Walkthrough presentation system (TIVR 2)

It is expected that there is an advantage to using the target-camera method in the tangible user interface, but several optimizations have been made.

Sensetable is basically a tabletop interface, and the user can use Sensetable from any direction. Thus, Sensetable has some potential in direction-free presentation systems, and is also likely to be able to accommodate multiple users.

This direction-free aspect has several advantages for the user. The user can walk around the table, always keeping his or body in the same direction as the camera. When a camera is used in the real world, the photographer always stays behind the camera and looks at the viewfinder or LCD screen. To control the camera in the tabletop system, it must be natural for the user to stay behind the camera.

TIVR2 Design

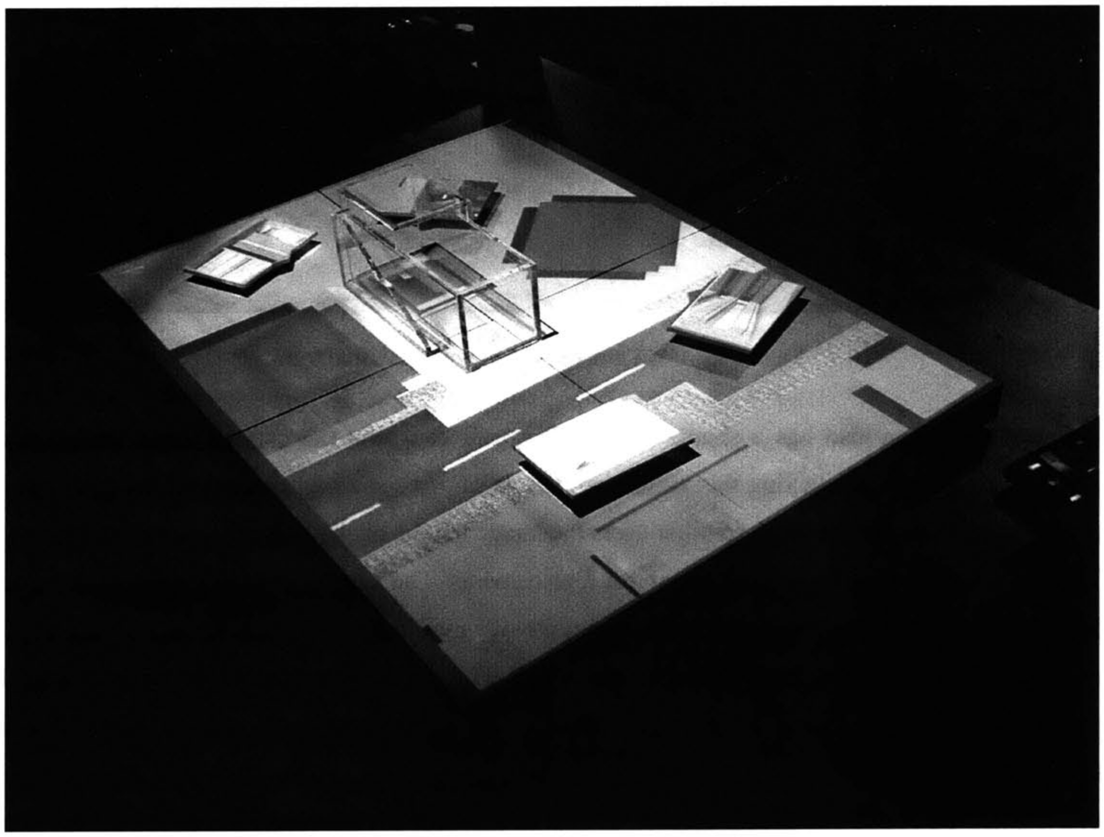

Figure 12 TIVR 2; four camera phicons with viewfinder images

The camera phicon with a viewfinder is the key development in TIVR 2. The user can control the camera in the 3D space by moving, picking up and placing the camera phicon on the table instead of using two target and camera packs, as was the case in TIVR 1. The camera position and direction control are integrated into one pack, so that users can control the camera more easily.

The camera image is projected onto the camera phicon in the same way as the viewfinder of a camera. This image moves and rotates dynamically, according to the camera phicon's movement. This viewfinder image helps the user to point his or her camera. Up to four users can use the camera phicon simultaneously.

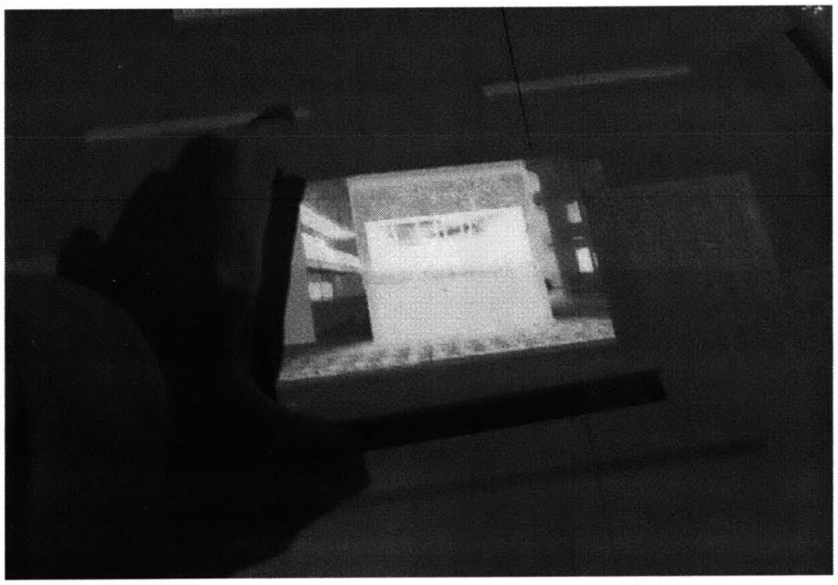

Figure 13 Camera phicon with viewfinder image

The camera phicon also has a shutter button. The user can send the camera image to an external projector or display by clicking this button, so that users can see a higher-resolution image. The user can utilize this function to show to other users the image that he or she would like to discuss .

TIVR 1's modules were modified to control four cameras. The camera control module was expanded in 3ds-Max to handle four cameras, and this module was modified to render four camera images.

An image-overlay module was also added in TIVR 2. This module captures four cameras images from the camera control module in 3ds-Max and then rotates, shrinks and moves the images to the position that matches the camera phicons on the table.

The camera phicon has a shutter release button. If the user presses this shutter button, the camera image is rendered in a high-resolution bitmap. This image is projected on a large screen or display so that others users can also see this image as a target of discussion.

TIVR 2 has four camera phicons, on the assumption that multiple users use this system at the same time, but multiple camera phicons are also effective if one user uses more than one camera phicon. User can leave one camera phicon in a certain place after he or she has positioned it, then the user can take another camera to look at another place. In this way, users can use camera phicons as a kind of 3D-camera sticky note.

If a user needs a wider view than one camera phicon can cover, he or she can place two camera phicons side by side and create a temporary panorama view camera.

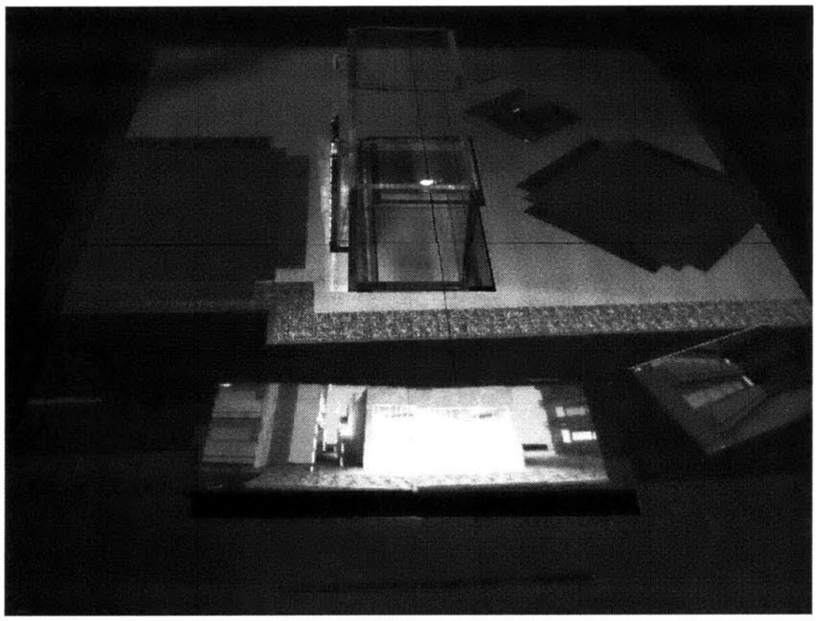

Figure 14 Temporary panorama camera

Camera phicon control system for a presentation with large-scale plan (TIVR 3)

Some urban development projects require the use of a large site-plan. In these cases, Sensetable is not an appropriate device because the table' s size is limited. To satisfy the requirements of these projects, a third system, TIVR 3, was developed.

TIVR 3 is based on modified mimio pens and a sensor bar. Mimio is a product of Virtual Ink Corp. Mimio was originally designed to record handwriting on a whiteboard in digital data. It consists of a sensor bar and marker pens. The sensor bar has two ultrasonic microphones at both ends of the bar, a phototransistor sensor, and a RS232C interface, which is normally attached to the side of the whiteboard.

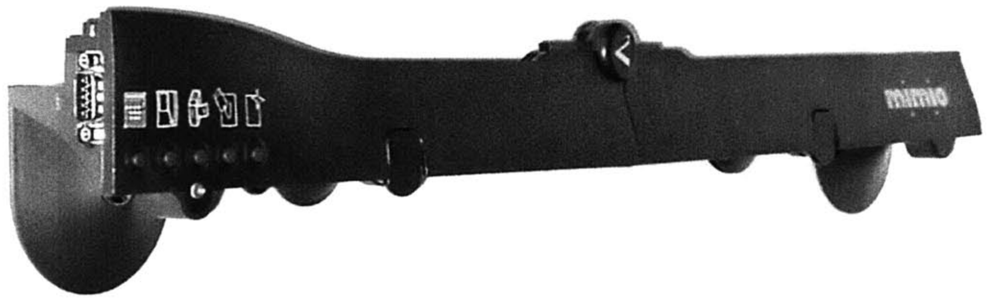

Figure 15 Mimio bar

The marker consists of the attachment cover of a dry-erasable marker pen and the pen itself. The pen's attachment cover has both an ultrasonic speaker and an infrared LED.

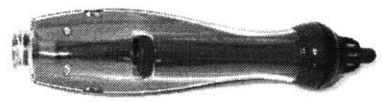

Figure 16 Mimio pen

When the pen's tip makes contact with the surface of the whiteboard, the pen's LED flashes and its ultrasonic speaker beeps at the same time. Once the sensor bar's phototransistor has detected the flash from the pen, the bar measures the delays of the ultrasonic beeps using the two microphones and calculates the distances from each microphone to the pen. The computer receives these distances and triangulates the marker pen's position. The most appealing technologies for the TIVR project were the mimio's high resolution and the large sensing range.

Mimio's resolution is almost 3mm, and it senses approximately 90 times in a second. Its recommended sensing range is eight feet. These capabilities well satisfy the requirements for TIVR 3.

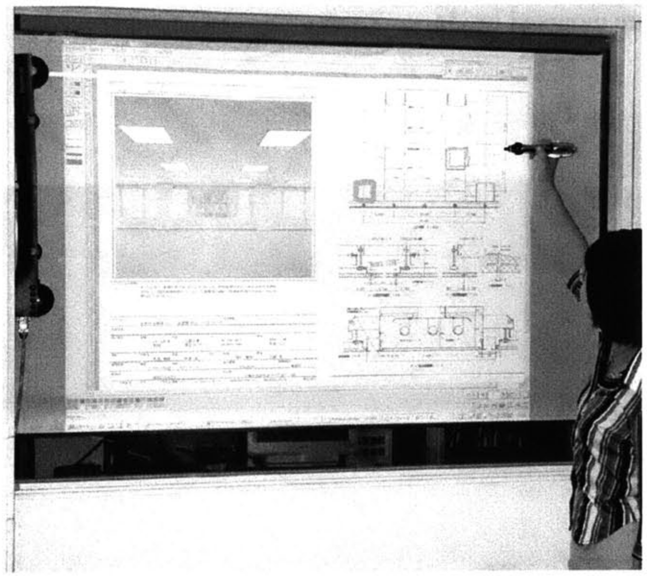

Figure 17 Mimio with rear-projection screen

Mimio has a mouse driver function so that users can use a whiteboard with the Mimio environment as a large touch-screen.

However, Mimio only detects one marker at a time, and its mouse driver does not recognize which pen is on the board. In interface system 1, in order to detect camera position and camera direction, two marker positions should be detected simultaneously and they should be distinguishable.

To solve this problem, the following two modifications were made:

Adjustable switching circuit

TIVR 3 driver

The adjustable switching circuit a) is a switching circuit that can flash two markers alternately. The switching interval of this circuit is adjustable, so that the interval of these two markers can be controlled, from 1 times/sec to 10 times/sec. Mimio also needs a small time gap between the two marker signals, otherwise it misunderstands these two signals as a signal from one pen, and outputs confused coordinate data. Thus, a four-cycle switching circuit was implemented, e.g. [start]->[pen 1 signal]->[gap]->[pen 2 signal] ->[gap]->[return to start]. The signal and gap duty rate is also

adjustable.
Due to the limitation of the Mimio sensors, the maximum interval frequency is approximately 3 times/sec. The frequency is also affected by the system's surroundings, because random reflections and background noise sometimes interfere with the system. Mimio's two pen tip modules are mounted on the top of a small box as a camera phicon. The circuit boards for the Mimio pen head, switching circuit and batteries are mounted inside.

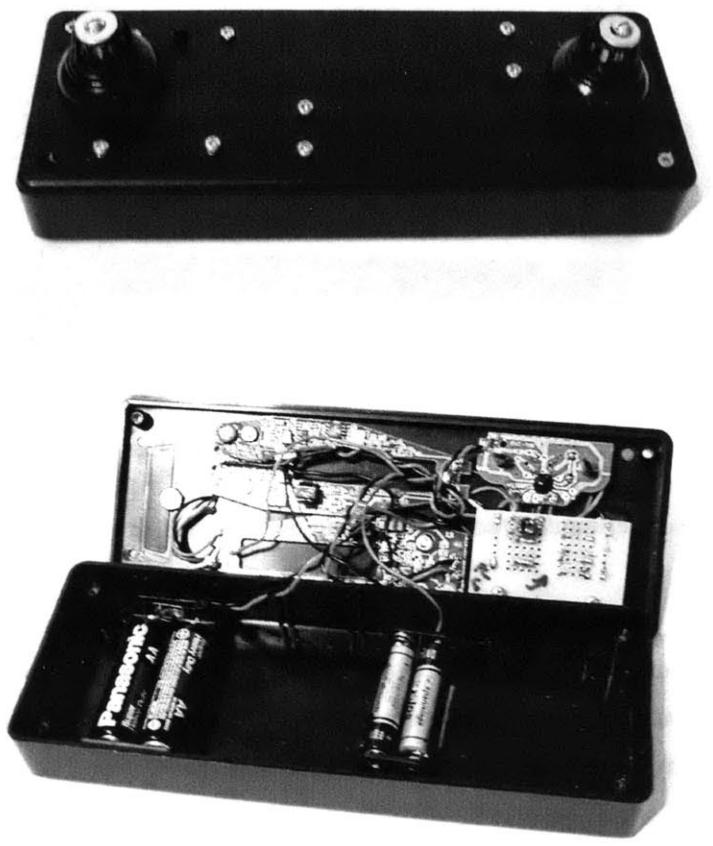
Figure 18 TIVR 3 sensor

The TIVR driver b) reads the RS232C data from the Mimio bar, divides them into two separate marker positions, and sends the data to the presentation system.
Mimio's data originally contains some spike-like data noise. The raw data of the coordinate sometimes jumps ten inches and return to the original position. This driver also checks the coherency of the data

and cancels the data noise, just like the original Mimio driver cancels it.
The data from the two pen tips is smooth when the user is moving the camera at a slow speed, but when the camera movement becomes quick, the data becomes jerky, because the data from the camera phicon is interleaved.
In order to smooth out the movement, the TIVR driver also compensates the coordinate data.

Evaluation

Control condition

An experiment was conducted to clarify the advantage of TIVR over orthodox user interfaces. TIVR 3 was designed for special circumstances, and the basic user interface is almost same as TIVR 2, so the experiment was planned using TIVR 1 , 2 and an orthodox VR system.

With the numerous advantages of tangible interfaces, such as intuitiveness and affordances, it was expected that both TIVR conditions would demonstrate some benefit during the VR experience, compared with orthodox interface conditions.

First, however, some background to the most orthodox interface for VR should be given.

Ivan Southerland developed virtual reality based on Bell Helicopter Company's helicopter flight simulator in the early 1960s. Since then, the flight simulator has been one of the most principal VR applications and the joystick has been the most popular input device for VR. As mentioned earlier, there have been several VR user interfaces before, but it is safe to say that the joystick is the most popular input device for 3D space navigation.

Therefore, the joystick user interface was chosen as a control condition for this experiment.

Tasks

At a practical presentation, the user has some knowledge of the existing site. The user also has a particular interest; for example, what his or her home looks like from the newly built building, what the new building looks like from his or her home, how the new building blocks existing view, how they cast shadow, and so on. When architects, city planners and audiences discuss the design of the project, each user tries to see the image in his or her own mind. For this experiment, the situation of the presentation was simplified. Participants were shown an image and asked to point the camera, while the time taken to move the camera was measured. The participants were then interviewed as to whether they liked interfaces.

Experimental design

Conditions in common

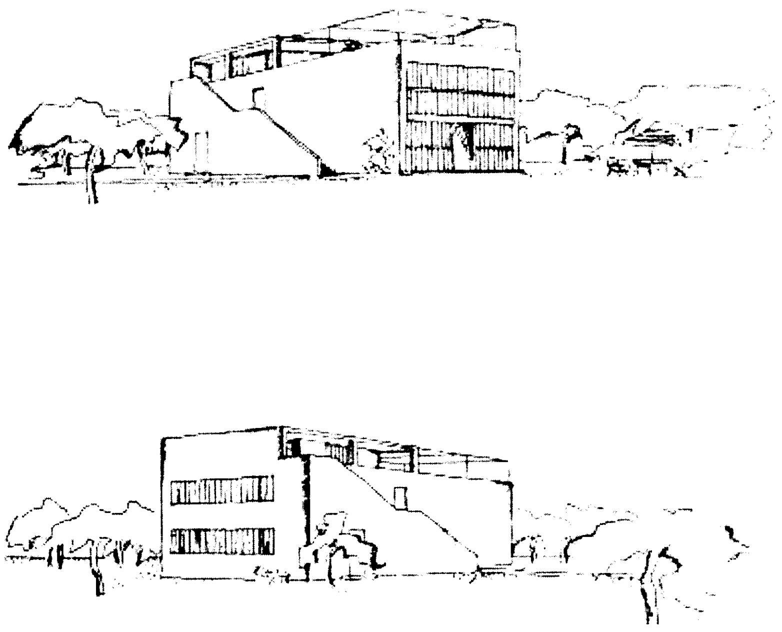

Figure 19 Le Corbusier's Citrohan drawing

The site and building for this experiment were prepared. The site is one of the narrow streets in the suburbs of Cannes, France. Le Corbusier's Citrohan 1[xii] stands at the center of the experimental site. Citrohan is Le Corbusier's mass-production house project with prototype houses. Citrohan 1 was never built, but Citrohan 2's prototype was built as the pavilion for the international exposition in Paris in 1925. Citrohan 2 was later destroyed and a replica was built at Bologna, Italy, in 1977.

The common conditions for these three experimental designs are Citrohan's acrylic model at the center of the site, the plan projected onto the surface of the table, and the LCD display for the camera image. The presentation system is used in all three conditions, so the image quality of the plan projected on the table and the image on the LCD display are always the same in all conditions.

The specifications of the equipment used are as follows:

LCD Projector: Infocus LP130

Resolution:	1024x768 @70Hz
Panel:	DLP single panel
Light:	1100 ANSI lumens

LCD display: NEC LCD1715

Resolution:	1280x1024 (Using 1024x768 for experiment)
Backlight:	260 cd/m2

PC: Pentium 4 2.53Ghz with 512Mbyte memory
Nvidia GeForce5200 with 128Mbyte memory
Windows 2000 SP4 + 3D studio MAX 5 +Delphi 5

Joystick condition

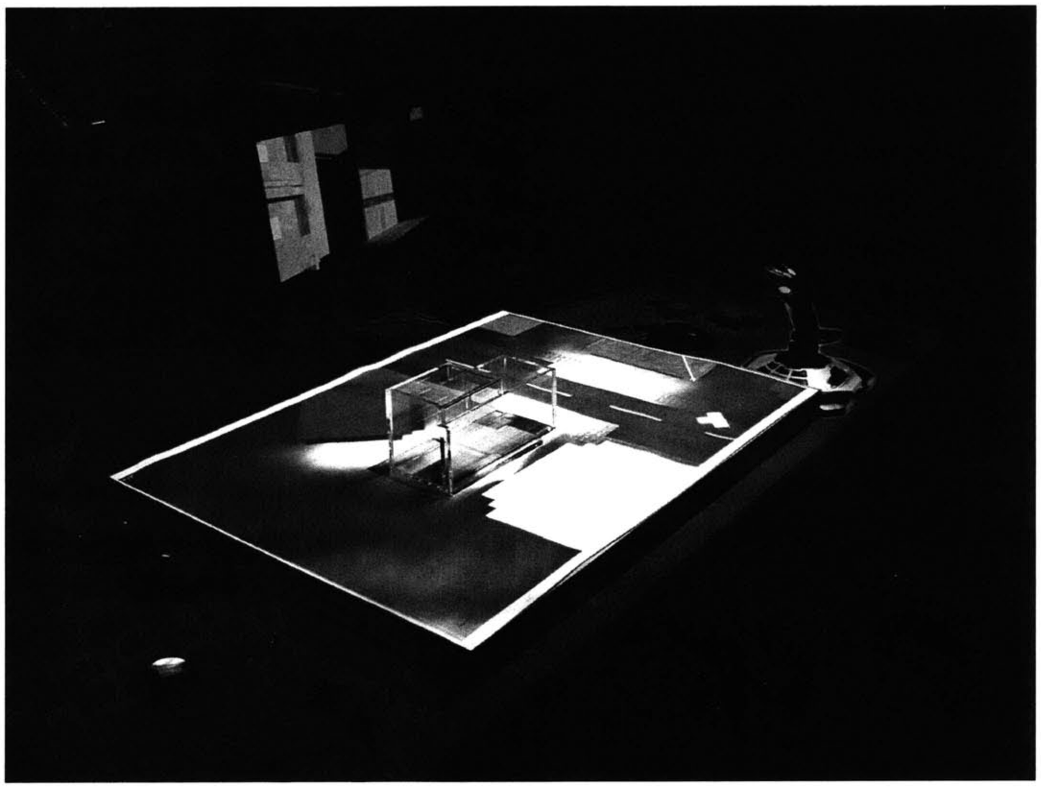

Figure 20 Joystick condition with Citrohan model and site

To compare the tangible user interface with a joystick-operated VR, an experimental system was designed, in which participants can navigate a virtual 3D architectural space with joystick.
An LCD monitor was installed on the side of the table opposite to the user. The site plan image was projected onto the table and the acrylic Citrohan model was placed at the center of the table. The camera icon was also projected onto the table. These site images, the Citrohan model and the camera icon were used only for display; the user could not touch or manipulate them directly.

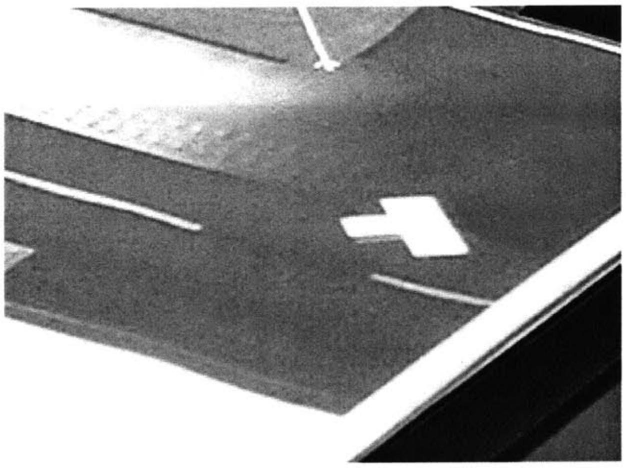

Figure 21 Camera icon on site

The user controlled the camera via the joystick, a which was a Logitech Extreme 3D Pro. This joystick has one analog stick with a twist sensor, eight-way hat (small joystick on top of main stick), twelve buttons and one ladder. The joystick's X-Y and twist data were used to navigate the camera.

Figure 22 Logitech's Extreme 3D pro

The joystick's Y-axis movement controls the camera's forward and backward movement, and the X-axis controls the right and left sidestep. A twist controls the camera's horizontal rotation. All movement and rotations are in analog control. The user can control the camera's speed via the stick's tilt angle. The camera's maximum speed is 8m/s (scale speed), and the maximum rotation speed is 180 degree/s.

TIVR 1 condition

The condition for TIVR 1 was the same as the original implementation of TIVR 1. The LCD monitor, site image and model conditions were the same as the joystick condition.

TIVR 2 condition

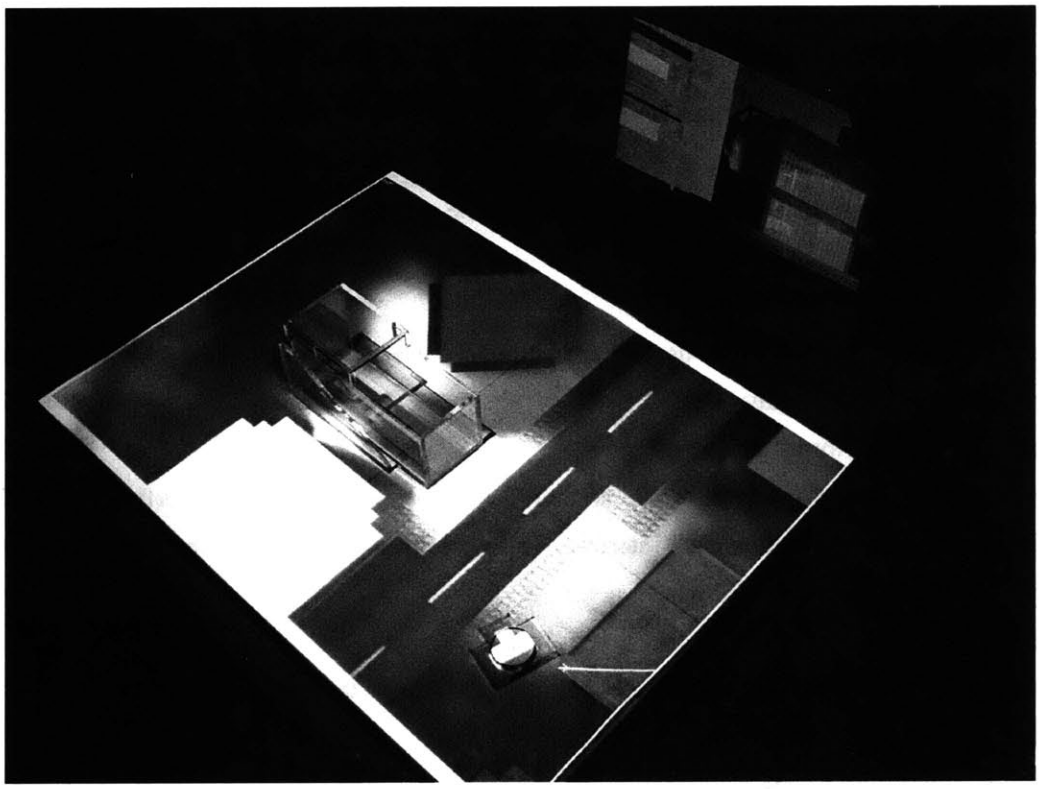

Figure 23 TIVR 2 condition

In order to compare TIVR 2 with the joystick interface, the TIVR 2 system was simplified for the experiment. Each participants participated in the experiment alone, so TIVR 2 was modified to be a single user interface. The viewfinder image was moved to the LCD screen so that the participants could see the same image as they saw at the TIVR 1 condition and the Joystick condition. The rest of the condition, site image and model were the same as the other conditions.

Participants

There were eight participants, four female and four male. Participants range in 28 to 38.
All participants are recruited through e-mail posted on Lawrence elementary school PTO mailing list in Brookline. Participants with high 3D computer-graphic, flight simulator and video game skills were asked not to participate this experiment. This participant pool is meant to represent the novice users of VR navigation. All the participants were volunteers.

Procedure

The procedure was as follows:

Explanation

The purpose of the experiment was explained, as well as how it would be carried out.

Site and building briefing

The site and building were explained briefly, as was the size of the building and the site shown on the table.

User interface briefing

The three-user interface and their usage were introduced. Users were shown how the camera image is displayed on the LCD display.

Site strolling using three user interfaces

The participants were asked to walk around the site, using the three interfaces and taking a three-minute walk to examine each interface. The participant was given a chance to get acquainted with the interfaces and site buildings.

First interview

The participants were then asked to rate their appreciation of these devices on the five-point Likert scale.

Camera movement tasks

The participant was shown one of the images and asked to point the camera to the position of the image, using the joystick, TIVR 1 and TIVR 2 conditions randomly. Each participant used all three interfaces three times in total.

The camera was always placed at the same position at the start, then the image was shown and the participant asked to move the camera. They were also asked to state whether they felt that the camera was pointing in the right position and direction. The participant was then instructed to point the camera, and the time taken to complete the task was measured.

The position was judged correct if it was located within the tolerance of 20 mm radius from the exact position, and its direction was within the tolerance of 5 degrees from the correct angle. Images matched 90% if the camera was located within these tolerances.

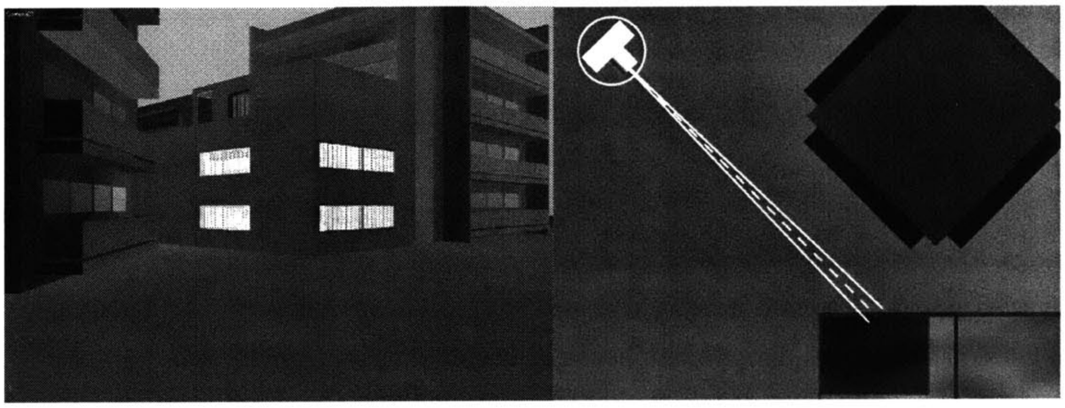

Figure 24 Task example image and answer tolerances

Three times were measured for each of the three conditions. The camera's movement was also recorded.

Second Interview

After these trials, the participants were asked to rate on a five-point Likert scale whether they felt it was easy to use these devices or not.

The participants were also asked to make comments and talk freely during the experiment. These comments were collected.

Results

Performance

The participant's performance (time) was collected for each of the nine tasks, and their interview answers. The participants were asked to move their camera into the correct position and all of them completed the task. Therefore, time could be used as a score to measure performance in the test.

The following histograms show the distribution of the time the participants.

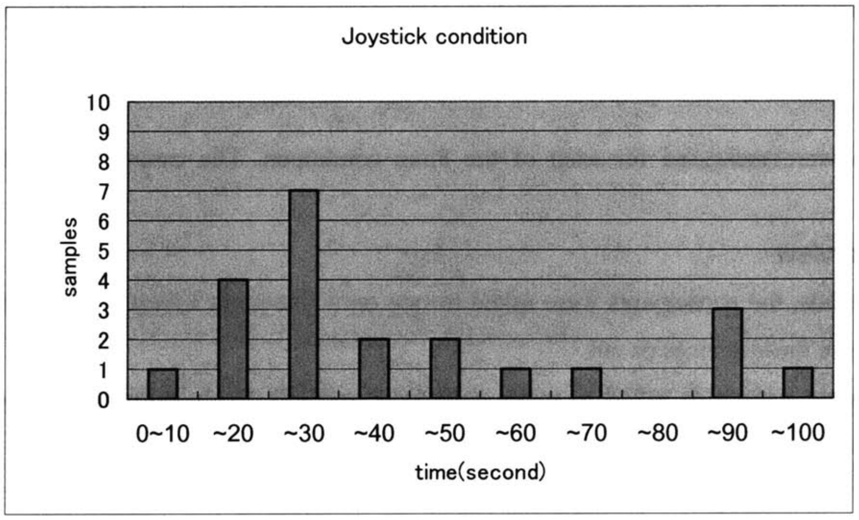

Figure 25 User's performance (joystick condition)

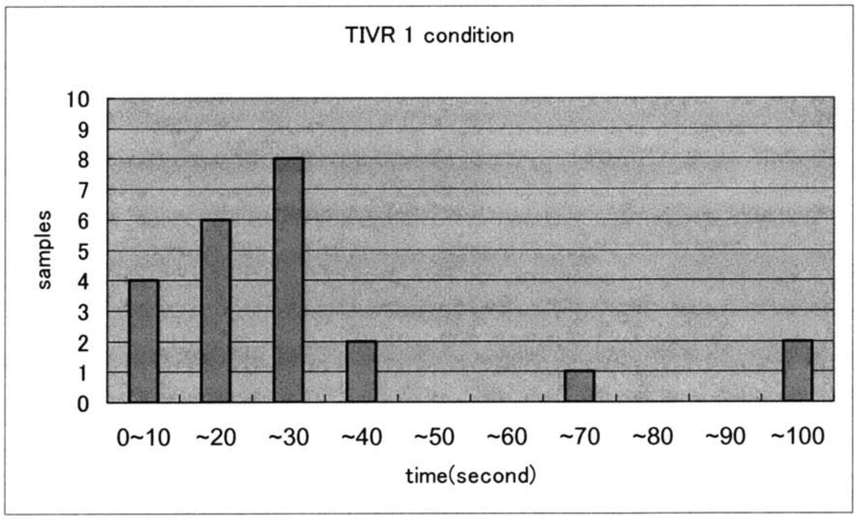

Figure 26 User's performance (TIVR1 condition)

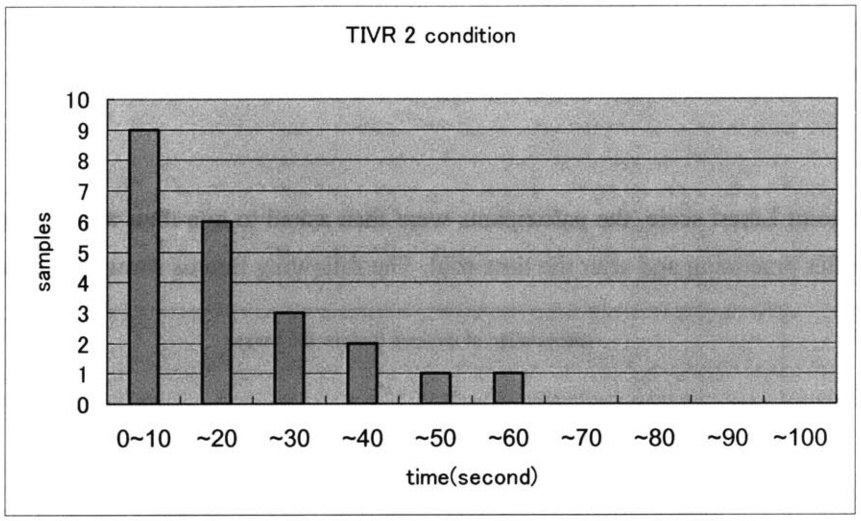

Figure 27 User's performance (TIVR2 condition)

These figures show that the participants scored highly using TIVR 2. The histogram for the joystick condition shows that some participants took an extraordinarily long time to complete the tasks. From the observation, these participants almost got lost on the table during the experiment, and took a very long time to complete the task.

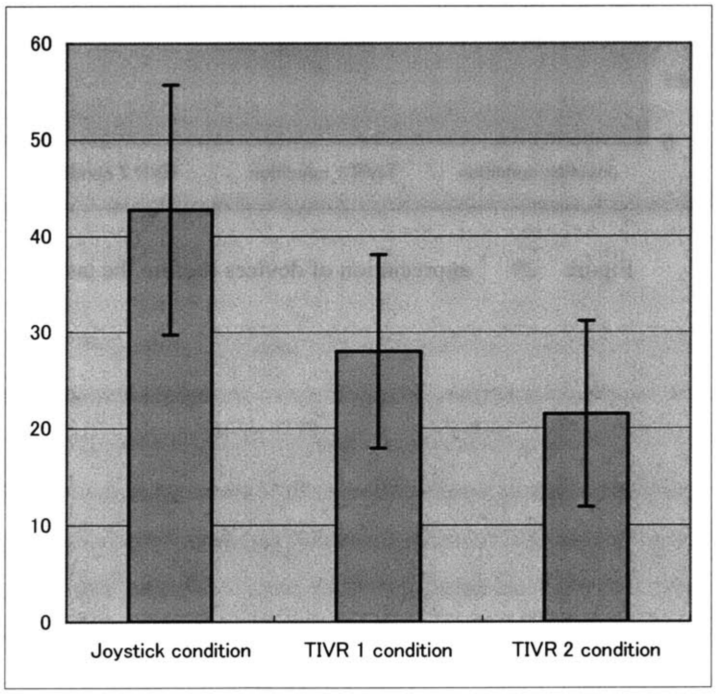

Figure 28 average score (second) for camera pointing

The expected trend was supported by this data, and the results suggest a satisfactory significance (ANOVA for condition effect gives F(3,24)=3.75 p= 0.028).

Interviews

Using a five-point Likert scale, the participants were then asked to rate their appreciation of the three devices just after practicing and after the time-trial. The following figures shows the users' preference

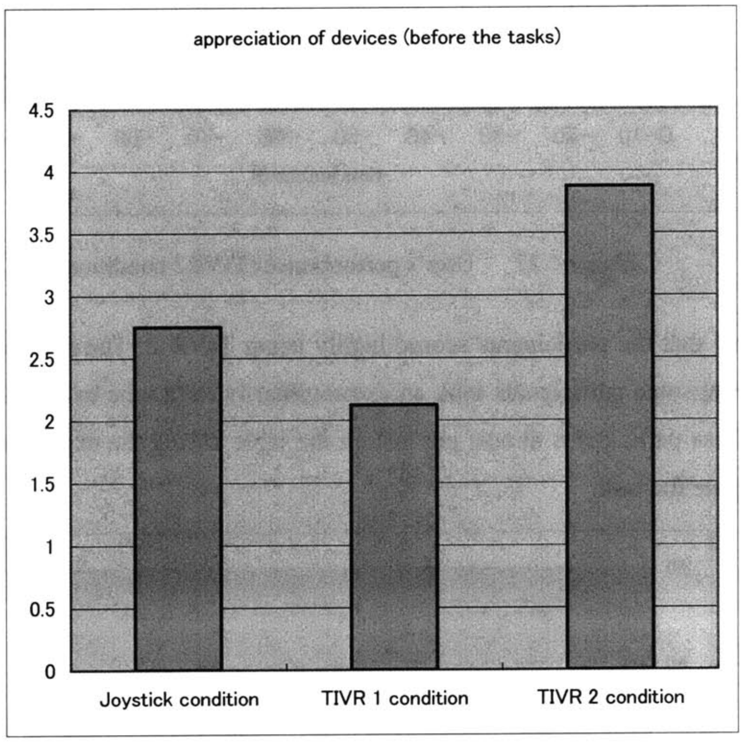

Figure 29 appreciation of devices (before the tasks)

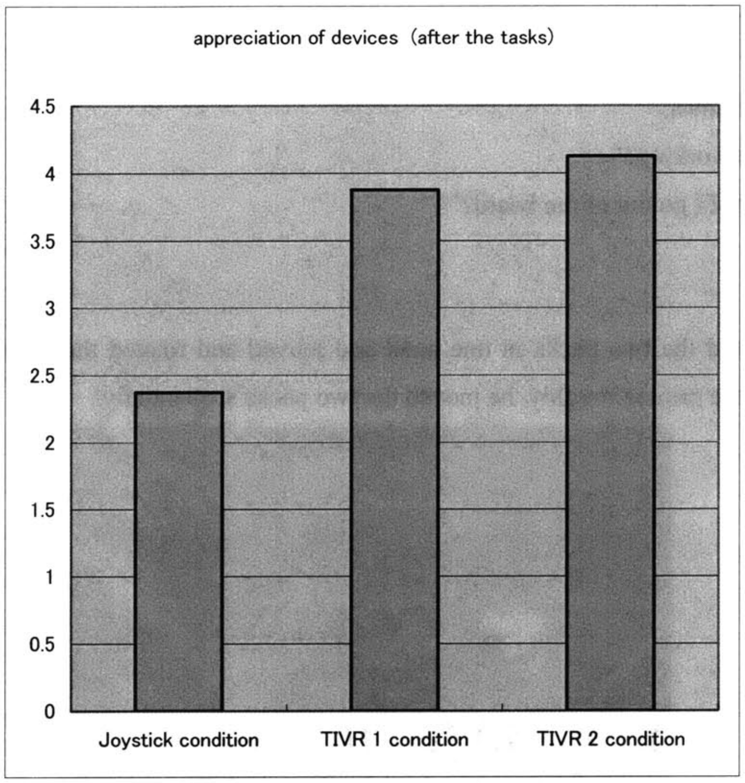

Figure 30 appreciation of devices (after the tasks)

The participants preferred TIVR 2 most in both interviews (p=1.75E-06). The second preference changed from the joystick to TIVR 1 after the time-trials.

I also asked about the strong points and weak points of each condition. Many participants commented on the smoothness of the joystick as a strong point, and the jerkiness, instability and vibration of the TIVR condition as a weak point.

Regarding the joystick, several participants also commented that it was hard to control the camera when its direction was toward the participants.

Observation

Joystick condition

Some participants seemed to be confused, as if they had lost control of the camera. They seemed to lose orientation, and rotated the camera excessively.

Some other participants seemed confused regarding the direction control when camera was moving toward them. If they were confused, the participants looked back and tried to match their orientation with the camera's orientation.

Most participants remained silent, except for the questions regarding the usage of the joystick. Their remarks were as follows:

'Are these buttons working?'

'What will happen if I go out of the board?'

TIVR1 condition

One participant held the two packs in one hand and moved and rotated them just like one camera. When he pointed the camera roughly, he moved the two packs sequentially.

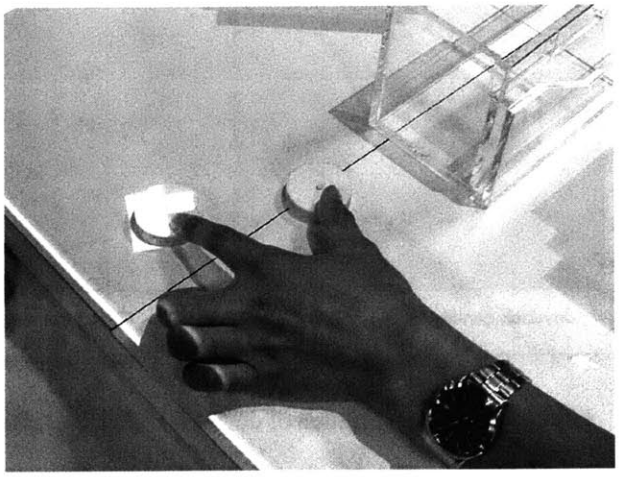

Figure 31 Participant uses two packs in one hand

Most participants hold the packs in both hands, but moved the camera pack and the target pack sequentially, not simultaneously.

TIVR 2 condition

Some participants stood up and moved or leaned over the table and model, and peeked into the model, then moved the camera to match their viewing direction.

Most participants picked up the camera from the table and placed it at the destination, and then adjusted the camera position and direction by sliding the camera phicon on the table.

Two participants picked up the camera with their fingers to make a fine adjustment. One of these participants tried to adjust the camera precisely, even after he declared that he had finished.

Discussions

Performance advantage

Due to the limitation of small number of the samples, all that can be safely stated is that the data of the experiment suggests slight trend of TIVR advantage over joystick interface. Figure 25,26 and 27 shows that TIVR's performance statistics are more convergence than joystick condition. Reason for this convergence is obvious because participants control their camera by their hands in TIVR condition, so they did not loose control compared to joystick conditions in which some users lost sense of orientation and lost control of their camera.

Though participants pointed out that TIVR conditions have some jerkiness and instability of TIVR, TIVR conditions showed better performance.

The result of the interview also shows user's preference for TIVR 2, but the preference for TIVR 1 condition

Observations

I could observe several participants' interesting behaviors. It is too early to conclude these observations as distinctive futures of TIVR, but these observations imply the potential of TIVR technology.

One of the most unexpected observations was the user's innovative attitude regarding the TIVR. It was expected that the participants would use packs and phicons, as explained above. However, several users tried to use them by their own way.

There may be several reasons why they might have tried to improve the interface. One hypothesis is that users can utilize their knowledge and experience of daily life to improve usability in TIVR condition.

In TIVR condition, some participants showed their willingness to use the system for more than required for the tasks. In contrast, most participants released the joystick as soon as the task was finished. The number of samples is limited and it is too early to say that TIVR draw out user's participation.

Future study will probe the effect of TIVR for user's motivation, but I expect that TIVR have some potential to attract users.

Conclusions

In conclusion, it is proposed that using this tangible user interface-based system for architectural and urban-planning discussions will help the audience to show their desired image in the 3D space. Both person with computer-graphics skill and without it will enjoy strolling and experiencing in the future city and building.

I can also expect that TIVR will motivate users to participate discussions, but the effect will be measured in the future study. This study's evaluation was individual performance test, but the experiment with discussion-condition will probe the true effect of TIVR, and may depicts more appropriate user interface for the mutual understanding.

Finally, It is hoped that TIVR will be one of the steps of improvement of the tool in mutual understanding and trust between architects, urban planners, neighborhoods and stakeholders, and to prevent misunderstanding and conflict.

Future work

Design variation tool

Design modification is one obvious omission from the TIVR tool. Design modifications, such as changing the height of the building or widening a street, need a great deal of engineering and cost simulations and checking of legal issues. However, architects and planners can prepare variations in the design. Therefore, a function for choosing design variations will satisfy the audience's needs and assist architects and planners with their work.

Tangible user interface for animation camera control

TIVR is being developed on the premise that a user of the system is a non-professional. However, control of an animation camera is difficult, even for professional CG creators. Partly because of failures in camera pointing, up to half of the rendered bitmaps are not used in the final animation movie.

A camera movement tracking function was added to the TIVR for experimental use, but this function will be expanded as a tangible animation camera controller.

Tangible user interface for section and layering

The TIVR focuses mainly on a simulation of the exterior view. However, interior and section view simulations are also important in several situations.

There is a great need for a combination of the TUI Tabletop Section Viewer and 4D CAD (spatial 3-dimensions with time axis CAD) in construction management. 4D CAD simulates construction process, such as how pre-fabricated beams are hoisted to the adequate floor, or how door frame is carried into the proper room. In these situations, section viewing function in real-time animation is really effective.

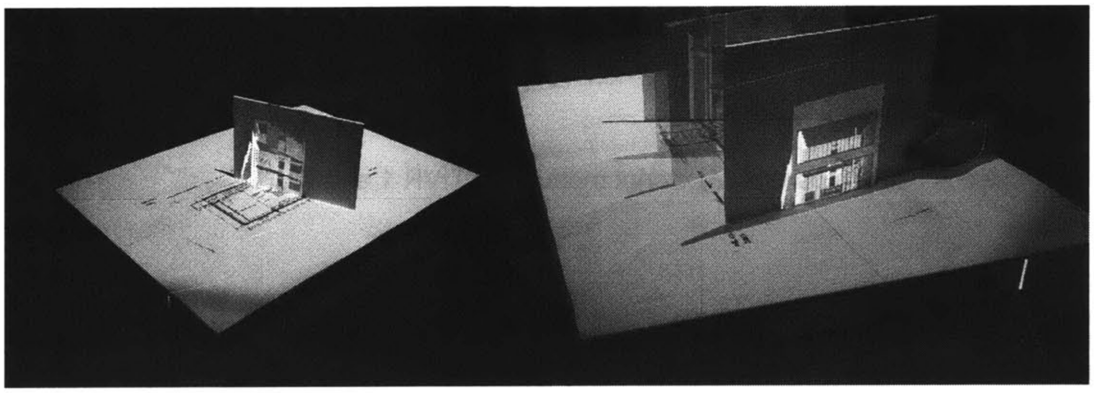
Figure 32 TUI Tabletop Section Viewer

Appendixes

Appendix: A camera control Maxscript module for TIVR 1 and TIVR 2

```
global
x1,y1,x2,y2,x3,y3,x4,y4,v1,v2,v3,v4,w1,w2,w3,w4,l,k,wd,vd,calibState,xcalib,ycalib,views,pc,pt,log
file,log_open

function logopen =
(
	if (log_open == 1) then (closelog)
	log_open=1
	local log_file
	log_file = "*.log"
	log_file = getSaveFileName caption:"Save log to:"
	logfile=createfile log_file
)
function logclose=
(
	close logfile
	log_open=0
)


function calibration1 penx peny =
(
	x1=penx
	y1=peny
	messagebox "calibration left-top complete"
)
function calibration2 penx peny =
(
	x2=penx
	y2=peny
	messagebox "calibration right-top complete"
)
function calibration3 penx peny =
(
	x3=penx
	y3=peny
	messagebox "calibration right-bottom complete"
)
function calibration4 penx peny =
(
	x4=penx
	y4=peny
	  v1=x2-x1
	  w1=y2-y1
	  v4=x4-x1
	  w4=y4-y1
	  v3=x3-x4
	  w3=y3-y4
	  vd=v1-v3
```

```
          wd=w1-w3
          l=v1*wd-w1*vd
          k=v4*wd-w4*vd
        viewS = getViewsize()
        messagebox "calibration left-bottom complete"
)
function pen penx peny penstate =
(

        if (calibstate==4) then (penmain penx peny)
        if (calibState==3 and penstate==1) then
        (calibration4 penx peny
        calibstate=4)
        if (calibState==2 and penstate==1) then
        (calibration3 penx peny
        calibstate=3)
        if (calibState==1 and penstate==1) then
        (calibration2 penx peny
        calibstate=2)
        if (calibState==0 and penstate==1) then
        (calibration1 penx peny
        calibstate=1)
)
function penmain penx peny=
(
                local a=penx-x1
                local b=peny-y1
                local aa=k*v1/l-(a*wd-b*vd)/l-v4
                local bb=(a*wd-b*vd)*v4/l-k*a/l
                local xx1=(-aa+sqrt(aa*aa-4*bb))/2
                local xx2=(-aa-sqrt(aa*aa-4*bb))/2
                local dd1=abs(xx1-0.5)
                local dd2=abs(xx2-0.5)
                if (dd1<dd2) then (xx=xx1 )
                        else (xx=xx2)

                if ((0<=xx1) and (xx1<=1))          then (xx=xx1)
                                                    else (xx=xx2)
                yy=((a*wd-b*vd)-l*xx)/k
                local viewS = getViewsize()
                local newCameraPoint = mapScreenToCP [xx*viewS.x+xcalib,yy*viewS.y+ycalib]
                Local                           NewCameraPoint3                             =
[newCameraPoint.x,newCameraPoint.y,$camera01.position.z]
                if                  (length(newCameraPoint3-$camera01.Position)             <
length(newCameraPoint3-$camera01.target.Position))
                then ($camera01.position =newCameraPoint3)
                else ($camera01.target.position = newCameraPoint3)
)

function cameraMove1 tx ty cx cy=
(
                local a=cx-x1
                local b=cy-y1
                local p=[a,b]
                local l= length(pc-p)
                pc = [a,b]
                local aa=k*v1/l-(a*wd-b*vd)/l-v4
                local bb=(a*wd-b*vd)*v4/l-k*a/l
                local xx1=(-aa+sqrt(aa*aa-4*bb))/2
                local xx2=(-aa-sqrt(aa*aa-4*bb))/2
```

```
                    local dd1=abs(xx1-0.5)
                    local dd2=abs(xx2-0.5)
                    if (dd1<dd2) then (xx=xx1 )
                              else (xx=xx2)
                    if ((0<=xx1) and (xx1<=1))          then (xx=xx1)
                                                       else (xx=xx2)
                    yy=((a*wd-b*vd)-l*xx)/k
                    local viewS = getViewsize()
                    local newCameraPoint = mapScreenToCP [xx*viewS.x+xcalib,yy*viewS.y+ycalib]
                    a=tx-x1
                    b=ty-y1

                    l= length(pt-p)
                    pt = [a,b]

                    aa=k*v1/l-(a*wd-b*vd)/l-v4
                    bb=(a*wd-b*vd)*v4/l-k*a/l
                    xx1=(-aa+sqrt(aa*aa-4*bb))/2
                    xx2=(-aa-sqrt(aa*aa-4*bb))/2
                    local dd1=abs(xx1-0.5)
                    local dd2=abs(xx2-0.5)
                    if (dd1<dd2) then (xx=xx1 )
                              else (xx=xx2)

                    if ((0<=xx1) and (xx1<=1))          then (xx=xx1)
                                                       else (xx=xx2)
                    yy=((a*wd-b*vd)-l*xx)/k
                    local newTargetPoint = mapScreenToCP [xx*viewS.x+xcalib,yy*viewS.y+ycalib]
                    local centerpoint=(newCameraPoint+ newTargetPoint) /2
                    local cv=newTargetPoint-centerpoint
                    cv=[(cv.x*cos 90)-(cv.y*sin 90),(cv.x*sin 90)+(cv.y*cos 90),cv.z]
                    newTargetPoint=centerPoint-cv
                    newCameraPoint=centerpoint+cv
                    $camera01.position                                                    =
[newCameraPoint.x,newCameraPoint.y,$camera01.position.z]
                    $camera01.target.position                                             =
[newTargetPoint.x,newTargetPoint.y,$camera01.target.position.z]
                    if (log_open==1) then (print $camera01.position to:logfile)
)

function cameraMove2 tx ty cx cy=
(
                    local a=cx-x1
                    local b=cy-y1
                    local p=[a,b]
                    local l= length(pc-p)
                    local message="camera delta "+(l as string)
                    local aa=k*v1/l-(a*wd-b*vd)/l-v4
                    local bb=(a*wd-b*vd)*v4/l-k*a/l
                    local xx1=(-aa+sqrt(aa*aa-4*bb))/2
                    local xx2=(-aa-sqrt(aa*aa-4*bb))/2
                    local dd1=abs(xx1-0.5)
                    local dd2=abs(xx2-0.5)
                    if (dd1<dd2) then (xx=xx1 )
                              else (xx=xx2)
          if ((0<=xx1) and (xx1<=1))          then (xx=xx1)
                                               else (xx=xx2)
                    yy=((a*wd-b*vd)-l*xx)/k
                    local viewS = getViewsize()
                    local newCameraPoint = mapScreenToCP [xx*viewS.x+xcalib,yy*viewS.y+ycalib]
```

```
                        a=tx-x1
                        b=ty-y1
                        l= length(pt-p)
                        pt = [a,b]
                        aa=k*v1/l-(a*wd-b*vd)/l-v4
                        bb=(a*wd-b*vd)*v4/l-k*a/l
                        xx1=(-aa+sqrt(aa*aa-4*bb))/2
                        xx2=(-aa-sqrt(aa*aa-4*bb))/2
                        local dd1=abs(xx1-0.5)
                        local dd2=abs(xx2-0.5)
                        if (dd1<dd2) then (xx=xx1 )
                               else (xx=xx2)

                       if ((0<=xx1) and (xx1<=1))            then (xx=xx1)
                                                         else (xx=xx2)
                        yy=((a*wd-b*vd)-l*xx)/k
                        local newTargetPoint = mapScreenToCP [xx*viewS.x+xcalib,yy*viewS.y+ycalib]
                        $camera01.position                                                    =
[newCameraPoint.x,newCameraPoint.y,$camera01.position.z]
                        $camera01.target.position                                             =
[newTargetPoint.x,newTargetPoint.y,$camera01.target.position.z]
                        local tv = $camera01.target.pos - $camera01.pos

                        local d = (atan2 tv.x tv.y)
                        $co.rotation=(eulerangles 0 0 (d+90)) as quat
                        $co.position=[$camera01.position.x,$camera01.position.y,$co.position.z]
                        if (log_open==1) then (print $camera01.position to:logfile;print "logout")
)

function joystick y x rz =
(
        local V=$camera01.target.pos-$camera01.pos
        local angle=-rz*4/joyrate
        v=normalize[(v.x*cos angle)-(v.y*sin angle),(v.x*sin angle)+(v.y*cos angle),0]
        local v2=[(v.x*cos 90)-(v.y*sin 90),(v.x*sin 90)+(v.y*cos 90),0]
        local v3=[-x*v.x/joyrate,-x*v.y/joyrate,0]
        local v4=[-y*v2.x/joyrate,-y*v2.y/joyrate,0]
        $camera01.pos=$camera01.pos+v3+v4
        local d = (atan2 v.x v.y)
        $co.position=[$camera01.position.x,$camera01.position.y,$co.position.z]
        $co.rotation=(eulerangles 0 0 (d+90)) as quat
        $camera01.target.pos=$camera01.pos+v
        if (log_open==1) then (print $camera01.position to:logfile)
)


function cameraMove3 cx cy d=
(
        d=-d
        local a=cx-x1
        local b=cy-y1
        local aa=k*v1/l-(a*wd-b*vd)/l-v4
        local bb=(a*wd-b*vd)*v4/l-k*a/l
        local xx1=(-aa+sqrt(aa*aa-4*bb))/2
        local xx2=(-aa-sqrt(aa*aa-4*bb))/2
        local dd1=abs(xx1-0.5)
        local dd2=abs(xx2-0.5)
        if (dd1<dd2) then (xx=xx1 )
             else (xx=xx2)
        yy=((a*wd-b*vd)-l*xx)/k
        local newCameraPoint = mapScreenToCP [xx*viewS.x,yy*viewS.y]
        local nv=$camera01.target.position-$camera01.position
        local n = length nv
        nv=[(cos d)*n,(sin d)*n,0]
        $camera01.position = [newCameraPoint.x,newCameraPoint.y,$camera01.position.z]
        $camera01.target.position=$camera01.position+nv
        $co.rotation=(eulerangles 0 0 (180-d)) as quat
        $co.position=[$camera01.position.x,$camera01.position.y,$co.position.z]

        if (log_open==1) then (print $camera01.position to:logfile)
)


function init =
(
calibState = 0
)

        joyrate=30000
        calibState = 0
        xcalib = -117
        ycalib = 40

        xcalib = 0
        ycalib = 0
        pc = [0,0]
        pt = [0,0]
        log_open=0
        registerOLEInterface
#(pen,cameraMove1,cameraMove2,cameraMove3,joystickR,JoystickL,JoystickF,JoystickB,Joysti
ck,init,logopen,logclose)
```

Appendix: B Bridge module source code

```
unit sensetable;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ScktComp, StdCtrls,comobj, AppEvnts, ExtCtrls;
type
  TForm1 = class(TForm)
    ServerSocket1: TServerSocket;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Button1: TButton;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Button2: TButton;
    Button3: TButton;
    procedure Button1Click(Sender: TObject);
    procedure ServerSocket1ClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocket1ClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
  private
    { Private }
  public
    { Public }
  end;
  TpackInfo = record
        packNo:smallint;
        x,y:word;
        theta:smallint;
  end;
  TMyFiLo = class(TObject)
  private
        pointer:integer;
        member:array[0..3] of double;
        full:boolean;
  protected
  Public
        procedure init;
        procedure add(data:double);
        function average:double;
        function last:double;
```

```
            function norm:double;
   end;
var
   Form1: TForm1;
   ppX,ppY,cx,cy,tx,ty:word;
   objmax:variant;
   cmove,tmove:integer;
   X1A,Y1A,X2A,Y2A,X3A,Y3A,TA:TMyFiLo;
   p1far,p2far,p3far:integer;
   moved:boolean;
   bt:integer;
   direction:integer;
const
            packnorm=0;
            packbutton=1;
            packButtonR=3;
            packoff=2;
            noizeNorm=2.0;
            noizeTH=100;
implementation
{$R *.DFM}
procedure TMyFiLo.init;
begin
            pointer:=0;
end;
procedure TMyFiLo.add(data:double);
begin
            case pointer of
            0:        begin
                                member[0]:=data;
                                pointer:=1;
                      end;
            1:        begin
                                member[1]:=data;
                                pointer:=2;
                      end;
            2:        begin
                                member[2]:=data;
                                pointer:=3;
                      end;
            3:        begin
                                member[3]:=data;
                                pointer:=0;
                      end;
            end;
end;
function TMyFiLo.last:double;
begin
            case pointer of
            0:        begin
                                result:=member[3];
                      end;
            1:        begin
                                result:=member[0];
                      end;
            2:        begin
                                result:=member[1];
                      end;
            3:        begin
                                result:=member[2];
```

```
                end;
        end;
end;
function TMyFiLo.average:double;
begin
        result:=(member[0]+member[1]+member[2]+member[3])/4;
end;
function TMyFiLo.norm:double;
var
        ave:double;
begin
        ave:=self.average;

result:=(sqr(member[0]-ave)+sqr(member[1]-ave)+sqr(member[2]-ave)+sqr(member[3]-ave))/4;
end;
function packStatus(pack:Tpackinfo):integer;
begin
        packstatus:=packnorm;
        if (pack.x>=9999) or (pack.y>=9999) or (pack.theta>=9999) then packstatus:=packoff;
        if    (pack.x=10001)    and    (pack.y=10001)    and    (pack.theta=10001)    then
packstatus:=packbutton;
end;
function rd(x:integer):integer;
begin
        rd:=round(x/1)*1;
end;
function sqr(a:double):double;
begin
        sqr:=a*a;
end;
function dist(x1,y1,x2,y2:double):double;
begin
        dist:=sqrt(sqr(x1-x2)+sqr(y1-y2));
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
        objmax.init;
end;
procedure TForm1.ServerSocket1ClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
end;
procedure TForm1.ServerSocket1ClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
var
        packs:array[0..100] of TPackinfo;
        i,j,dataSize:integer;
        packinfo:TPackinfo;
        d:integer;
        x,y,dd,ddd:double;
        moved:integer;
 begin
        i:=-1;
        datasize:=8;
        while datasize=8 do
        begin
        inc(i);
        dataSize:=serversocket1.Socket.connections[0].ReceiveBuf(packs[i],8);
        end;
        label10.caption:='data no'+inttostr(i);
```

```
          moved:=0;
          for j:=0 to i do
          begin
                    packinfo:=packs[j];
                    label1.caption:='x='+inttostr(packinfo.x);
                    label2.caption:='y='+inttostr(packinfo.y);
                    label3.caption:='theta='+inttostr(packinfo.theta);
                    case packinfo.packNo of
                    9:
                              begin
                                        label13.caption:=inttostr(bt);
                                        if packstatus(packinfo)=packbutton then
                                        begin
                                                  objmax.pen(ppx,ppy,1);
                                        end else
                                        begin
                                                  ppx:=packinfo.x;
                                                  ppy:=packinfo.y;
                                        end;
                                                  x3a.add(packinfo.x);
                                                  y3a.add(packinfo.y);
                                                  if packinfo.theta<=60000 then
                                                  begin
                                                  ta.add(packinfo.theta);
                                                  label11.caption:=floattostr(ta.norm);
                                                  end;
                                                  label4.caption:=inttostr(ppx);
                                                  label5.caption:=inttostr(ppy);
                              end;
                    7:
                              if packstatus(packinfo)=packnorm then
                                        begin
                                                  x1a.add(packinfo.x);
                                                  y1a.add(packinfo.y);
                                                 if  (x1a.norm>noizenorm)  or  (y1a.norm>noizenorm)
then
                                                  begin
                                                            cx:=rd(packinfo.x);
                                                            cy:=rd(packinfo.y);
                                                            label6.caption:=inttostr(cx);
                                                            label7.caption:=inttostr(cy);
                                                            label14.caption:=inttostr(packinfo.theta);
                                                            moved:=2;
                                                  end;
                                        end;
                    10:
                              if packstatus(packinfo)=packnorm then
                              begin
                                        x1a.add(packinfo.x);
                                        y1a.add(packinfo.y);
                                        if (x1a.norm>noizenorm) or (y1a.norm>noizenorm) then
                                        begin
                                                  cx:=rd(packinfo.x);
                                                  cy:=rd(packinfo.y);
                                                  label6.caption:=inttostr(cx);
                                                  label7.caption:=inttostr(cy);
                                                  moved:=1;
                                        end;
                              end;
                    12:
```

```
                    if packstatus(packinfo)=packnorm then
                    begin
                            x2a.add(packinfo.x);
                            y2a.add(packinfo.y);
                            if (x2a.norm>noizenorm) or (y2a.norm>noizenorm) then
                            begin
                                    tx:=rd(packinfo.x);
                                    ty:=rd(packinfo.y);
                                    label8.caption:=inttostr(tx);
                                    label9.caption:=inttostr(ty);
                                    moved:=1;
                            end;
                    end;
            15:
                    if packstatus(packinfo)=packnorm then
                    begin
                            x2a.add(packinfo.x);
                            y2a.add(packinfo.y);
                            if (x2a.norm>noizenorm) or (y2a.norm>noizenorm) then
                            begin
                                    tx:=rd(packinfo.x);
                                    ty:=rd(packinfo.y);
                                    label8.caption:=inttostr(tx);
                                    label9.caption:=inttostr(ty);
                                    moved:=2;
                            end;
                    end;
            16:
                    if packstatus(packinfo)=packnorm then
                    begin
                            x1a.add(packinfo.x);
                            y1a.add(packinfo.y);
                            direction:=direction+packinfo.theta;
                            x3a.add(direction);
                            label12.caption:=floattostr(x3a.norm);
                            if   (x1a.norm>noizenorm)   or   (y1a.norm>noizenorm)   or
(x3a.norm>noizenorm*50) then
                            begin
                                    cx:=rd(packinfo.x);
                                    cy:=rd(packinfo.y);
                                    ddd:=x3a.last
                                    dd:=round(x3a.last);
                                    objmax.cameramove3(cx,cy,dd);
                            end;
                    end;
            end;
    end;
    case moved of
            1:objmax.cameraMove1(x1a.last,y1a.last,x2a.last,y2a.last);
            2:objmax.cameraMove2(x1a.last,y1a.last,x2a.last,y2a.last);
    end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
    serversocket1.port:=9998;
    serversocket1.Open;
    ppx:=0;
    ppy:=0;
    cmove:=0;
    tmove:=0;
```

```
          bt:=0;
                    objMax    :=    CreateOleObject('Max.Application.4');
                              x1a:=TMyFiLo.create;
          y1a:=TMyFiLo.create;
          x2a:=TMyFiLo.create;
          y2a:=TMyFiLo.create;
          x3a:=TMyFiLo.create;
          y3a:=TMyFiLo.create;
          TA:=TMyFiLo.create;
          x1a.init;
          y1a.init;
          x2a.init;
          y2a.init;
          x3a.init;
          y3a.init;
          TA.init;
          p1far:=0;
          p2far:=0;
          p3far:=0;
          objmax.init;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
          objmax.logopen;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
          objmax.logclose;
end;
end.
```

Acknowledgement

This thesis was made possible by the assistance and support of a large number of people.

First, I would like to thank Professor Eran Ben-Joseph and DR.Carlo Ratti of the Department of Urban Studies and Planning, as well as Professor Hiroshi Ishii of MIT Media Lab's Tangible Media Group for their advise and guidance throughout the development of this thesis.

Thanks should also go to the parents of Lawrence elementary school, those who participated in my evaluation experiment.

TMG and NTT Comware generously lent me all the equipment and space I used for the entire thesis.

I would like to thank J. Patten, J. Hines , G. Pangaro, M. Hirano, A. Narita, members of TMG and NTT Comware, and those who developed Sensetable. I believe your development will create the future and it is my great honor that I could join create the future with you.

I thank K. Kobayashi, K. Ryokai, for their great help, advise and suggestion.

I also thank Takenaka Corporation, who funded me for two years of my study in MIT.

Lastly, I would like to thank my family, who supported such a busy student father, and my grand mother Fumie Taguchi who had been to the sky since May 17, 2004.

Reference

[i] N. Negroponte, “Virtual Reality: Oxymoron or Pleonasm? “ Wired magazine 1.06 1993

[ii] Room-sized virtual reality (VR) device. Developed by Electronic Visualization Laboratory, the University of Illinois at Chicago.

iii Elumens Corporation (1996)

iv J. Oishi et al., (2002), Takenaka Corporation.

[v] H. Ishii and B. Ullmer, “Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms,” Proc. ACM CHI'97 Conference on Human Factors in Computing Systems, pp. 234-241,Addison-Wesley/ACM Press, 1997

vi H. Ishii and B. Ullmer, “Tangible Bits: Towards Seamless Interfaces between People, Bits, and Atoms,” Proc. ACM CHI'97 Conference on Human Factors in Computing Systems, pp. 234-241,Addison-Wesley/ACM Press, 1997

[vii] H. Ishii, J. Underkoffler, D. Chak, B. Piper, E. Ben-Joseph, L. Yeung, and Z. Kanji, “Augmented Urban Planning Workbench: Overlaying Drawings, Physical Models and Digital Simulation”, Proc. IEEE and ACM International Symposium on Mixed and Augmented Reality ’02 pp. 203-211

[viii] J. Underkoffler and H. Ishii. "Urp: A luminous-tangible workbench for urban planning and design" Proc. ACM CHI'99 Human Factors in Computing Systems Conference, pp. 386-393, Addison-Wesley/ACM Press, 1999..

[ix] J. Patten, H. Ishii, J. Hines and G. Pangaro,"Sensetable: A Wireless Object Tracking Platform for Tangible User Interfaces," Proc. ACM CHI' 01 Conference on Human Factors in Computing Systems, pp.253-260 Addison-Wesley/ACM Press, 2001

[x] M. Hirano, A. Narita, K. Kobayashi, H. Ishii,"A Tangible Interface for IP Network Simulation", Proc. ACM CHI'03 extended abstracts on Human factors in computing systems, pp. 800-801,Addison-Wesley/ACM Press, 2003

[xi] J. Yoon, J. Oishi, J. Nawyn, K. Kobayashi, N. Gupta "FishPong: Encouraging Human-to-Human Interaction in Informal Social Environments" ACM CSCW‘04

[xii] The geometric data and texture mapping are part of the Unbuilt project by Prof. Takehiko Nagakura of the School of Architecture, MIT.

Figure 4,6 are the intellectual property of Tangible Media group, Media Laboratory, MIT.
Figure 5,7 are the intellectual property of Tangible Media group, Media Laboratory, MIT and Department of urban studies and planning, MIT
Figure 8 is the intellectual property of Tangible Media group, Media Laboratory, MIT and NTT Comware

Mimio is a registered trademark of VirtualInk Corporation.
3DS Max is a registered trademark of Discreet inc.

3130-95