TACTUS: A Hardware and Software Testbed for Research in Multi-Touch Interaction

Paul Varcholik, Joseph J. Laviola Jr., Denise Nicholson

Institute for Simulation & Training
University of Central Florida
Orlando, Florida
pvarchol@ist.ucf.edu, jjl@cs.ucf.edu, dnichols@ist.ucf.edu

Abstract. This paper presents the TACTUS Multi-Touch Research Testbed, a hardware and software system for enabling research in multi-touch interaction. A detailed discussion is provided on hardware construction, pitfalls, design options, and software architecture to bridge the gaps in the existing literature and inform the researcher on the practical requirements of a multi-touch research testbed. This includes a comprehensive description of the vision-based image processing pipeline, developed for the TACTUS software library, which makes surface interactions available to multi-touch applications. Furthermore, the paper explores the higher-level functionality and utility of the TACTUS software library and how researchers can leverage the system to investigate multi-touch interaction techniques.

Keywords: Multi-Touch, HCI, Touch Screen, Testbed, API

1 Introduction

The pending proliferation of multi-touch technology, which allows interaction with a surface through multiple simultaneous points of contact and from multiple concurrent users, has the potential to radically change Human-Computer Interaction. However, unless the research community can answer fundamental questions about optimizing interface designs, and provide empirically driven guidelines, the technology can become just another I/O modality looking for a purpose. Unfortunately, there is limited availability of hardware and software to support this research. Thus, investigators must overcome a number of technical challenges to develop a multi-touch platform before they can begin research in this area.

Through an extensive literature and state-of-the-art review, an analysis of the requirements for a multi-touch research platform revealed two categories of components: those essential for basic multi-touch research; and secondary components necessary for extensive investigation and longer-term research projects. These components, listed in Table 1, emphasize a low-cost, do-it-yourself approach.

A multi-touch platform is made up of two primary components: a physical interaction surface, and a software system for collecting and interpreting points of contact. The hardware and software systems each require significant investments of time and effort to construct. While advances in hardware and multi-touch software infrastructure provide interesting research opportunities themselves, they are a barrier to entry for researchers who want to focus on higher-level interface issues or the development of novel applications.

This paper presents the TACTUS Multi-Touch Research Testbed, a hardware and software system for enabling research in multi-touch interaction. A detailed discussion is provided on hardware construction, pitfalls, design options, and software architecture to bridge the gaps in the existing literature and inform the researcher on the practical requirements of a multi-touch research testbed. This includes a comprehensive description of the vision-based image processing pipeline developed for the TACTUS software library, which makes surface interactions available to multi-touch applications. Furthermore, the paper explores the higher-level functionality and utility of the TACTUS software library and how researchers can leverage the system to investigate multi-touch interaction techniques.

Table 1. Essential and Secondary Components for a multi-touch research platform

Essential Components	**Description**
Multi-touch surface	Constructed using commercial-off-the-shelf hardware and requiring near zero pressure to detect an interaction point
Software Hit-Testing	Ability to determine the presence and location of each point of surface contact; supporting at least four users
Software Point Tracking	Identifying a continuous point of contact and reporting its velocity and duration
Secondary Components	
Application Programming Interface (API)	A software system upon which multiple multi-touch applications can be developed
Multi-Platform Support	The ability to access multi-touch interaction data from different computing environments (e.g. languages, OS's, etc.)
Reconfiguration	Modifying the software system without recompilation
Software Service	Allowing multiple applications to access multi-touch interaction data simultaneously, including over a computer network
Presentation-Layer Independence	Isolating the multi-touch interaction data from the system to graphically present such data, allowing any GUI to be employed when developing multi-touch applications
Mouse Emulation	Support for controlling traditional 'Window, Icon, Menu, Pointing Device' interaction through a multi-touch surface
Tangible Interfaces	The ability to detect and interact with physical devices placed on or near the multi-touch surface
Customizable Gesture System	Support for training arbitrary multi-touch gestures and mapping them to software events

2 Related Work

In 2001, Deitz presented the Mitsubishi *DiamondTouch* [1], a front-projection, multi-touch system that uses an array of antennas to transmit identifying signals that are capacitively coupled through a user. The *DiamondTouch* is fairly expensive, compared to a do-it-yourself approach, and users will occlude the front-projection display as they interact with the system.

In 2005, Han presented work on constructing low-cost multi-touch surfaces using Frustrated Total Internal Reflection [2]. Much of our hardware is a derivative of Han’s work. However, while Han’s paper presented a starting point for building low-cost multi-touch surfaces, it left out some of the details necessary to reliably construct a multi-touch platform.

In 2004, Wilson introduced *TouchLight* [3] and in 2007 Microsoft announced the Microsoft Surface [4] – an exciting development for bringing multi-touch technology closer to the consumer market. While a preliminary release of Microsoft Surface started in Spring 2008, there is still limited availability of this platform.

Both the *DiamondTouch* and the Microsoft Surface have commercial APIs for developing applications for their products. However, use of these software packages is largely dependent on the acquisition of their associated hardware. A few open-source multi-touch libraries exist, including: Touchlib [5], reacTIVision [6], and BBTouch [7]. These support various operating systems and programming environments, and vary in feature set and ease-of-use.

Much more work exists on the development of multi-touch hardware technology and associated software as in [8-10]. Our efforts build upon these achievements and offer a low-cost, full-featured multi-touch research testbed.

3 Multi-Touch Hardware

Although there are a variety of multi-touch designs, we believe that Frustrated Total Internal Reflection (FTIR) technology offers a robust and affordable hardware solution. FTIR surfaces share a set of common components: 1) An optical waveguide for conducting infrared (IR) light, 2) A supporting structure for holding the waveguide, 3) An IR sensing camera, 4) A projector and diffuser, 5) An IR emission source, 6) A computer. Our design is pictured in Figure 1. For the optical waveguide, we’ve chosen a 32”x24”, ½” thick sheet of clear acrylic. The dimensions of the surface match the 4:3 aspect ratio of most modern, short-throw projectors. The supporting structure for the acrylic is a 37” high table and includes a 6” wide border around the waveguide, for placing materials (or resting elbows) that will not interact with the surface. The IR camera and projector are placed below the acrylic, and are contained within the table. This design supports collaboration with up to four seated or standing users. The IR camera is a Microsoft LifeCam VX-6000, a commercial-off-the-shelf webcam that has been modified to allow IR light while filtering visible light. The projector is a Mitsubishi XD500U-ST short-throw projector, capable of producing a 60” diagonal from only 33” away. The diffuser is a sheet of Rosco Gray 7mm thick PVC, rear-projection material. For IR emission we chose a set of 32 Osram 485 LEDs. These are split into 4 chains (connected in parallel) of 8 LEDs (connected in serial) running to a common 12V power supply. Lastly, we’ve chosen a small-footprint MicroATX computer for operating the multi-touch surface software.

Several of these components are discussed in more detail below.

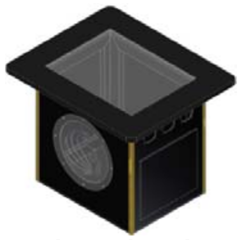

Figure 1. Multi-Touch Hardware Design

Acrylic Waveguide & Infrared LEDs. Acrylic has optical properties conducive to Total Internal Reflection [2]. It's also quite durable, inexpensive, and can be manufactured in a number of sizes. Thickness is a consideration at large dimension, as pressure on the surface causes the acrylic to noticeably deflect. At 32"x24" we found a ½" thick sheet of acrylic to be a nice compromise between rigidity and cost.

When placing the LEDs around the acrylic, the primary concerns are providing enough light and fully distributing that light within the waveguide. We initially drilled 5mm wide depressions, into the acrylic edges, to house the LEDs. This works quite well, and does not require polishing the acrylic edge to introduce light into the waveguide. A drawback to this approach is that the LEDs are semi-permanently affixed to the acrylic. Working with the acrylic (for example, to pour on a silicone rubber compliant surface) often requires removing the LEDs. In our final design, we chose to surround the acrylic with LEDs, equally spaced, and abutting the acrylic edges. Again, we found that polishing the acrylic edges was not required. The choice of 8 LEDs per side is more than sufficient for our surface size, given their 40-degree viewing angle. In fact, we found quite reasonable results with only two adjacent edges lit. To determine if enough light is being introduced into the waveguide, point the IR camera at an edge opposite to the illumination. The camera should detect a solid bar of IR – the light escaping the edge. If this bar of light is segmented, or significantly varies in intensity, then the illumination is not being fully distributed throughout the waveguide and additional LEDs are required.

Projector & Diffuser. Short throw projectors, capable of displaying large images from very close distances, have become increasingly available in recent years. A short throw is necessary for maintaining small table depth. If the multi-touch system has no depth requirement (e.g. a wall-display) then a traditional projector can help reduce cost. Strictly speaking, a traditional projector, with a system of mirrors for increasing focal length, can be used in a depth-limited multi-touch surface. However, this adds complexity to the hardware design.

The diffuser is the surface upon which the projected image will be displayed. Aside from the choice of materials, a chief concern for the diffuser is its placement – either above or below the waveguide. Placing the diffuser below the waveguide causes a slight disparity (of the thickness of the waveguide) between the interaction surface and the projected display. Furthermore, a pliable diffuser material, placed below the waveguide, will sag, deforming the projected image if not otherwise supported. Moreover, the diffuser may absorb some of the IR light passing through it. While this is a benefit for reducing ambient IR light, the diffuser will also absorb some of the light being reflected through FTIR when placed below the waveguide. For these reasons, we suggest placing the diffuser above the waveguide. This approach also protects the waveguide from scratches and oils from the users' fingers. Unfortunately, placing the diffuser above the waveguide negatively impacts the coupling between the users' fingers and the waveguide, decreasing the light reflected through FTIR. A material, placed between the diffuser and the waveguide, is required to improve coupling and thereby increase the amount of IR light directed to the camera while decreasing the force necessary to reflect it.

Compliant Surface. A finger makes an imperfect connection with acrylic. Micro air-gaps form between the user's fingers and the waveguide and maintain the original acrylic-to-air interface, thus supporting Total Internal Reflection. Moistening fingertips or pressing firmly on the surface can improve coupling, but these are not viable long-term solutions. A coupling material is required to permit low-force, high-quality surface interaction.

After much trial-and-error, we settled on a 1mm thick layer of SORTA-Clear 40 – a translucent silicone rubber – placed between the acrylic and the diffuser. SORTA-Clear 40 is a liquid that, when mixed with its catalyst, will cure at room temperature into a firm (40A Shore hardness) material that provides very good coupling with limited hysteresis. However, mixing the rubber with its catalyst creates air bubbles, which will cause considerable noise in the captured image. Placing the mixed rubber into a vacuum chamber can help remove these bubbles, but there is limited time before the material begins to cure, and the pouring and smoothing process will reintroduce some air. Placing the entire surface into a vacuum, after the material has been poured, may be the best option for removing bubbles – if a large enough vacuum chamber is available. We found good results, in mitigating air bubbles, simply by keeping the thickness of the rubber very small (e.g. <=1mm) and by pouring and smoothing slowly and deliberately. Applying a small amount of heat, from a hair dryer or heat gun, can help remove any stubborn bubbles before the rubber cures.

While pre-cured layers of rubber are available, we found them difficult to adhere to the acrylic without introducing a large number of air pockets. Pouring the silicone rubber directly onto the surface produced the best results.

4 Software Framework

The chief function of the TACTUS software library is to collect and interpret multi-touch surface input. The core components of the library do not specify how this data is used or displayed. Thus, the library is presentation-layer independent and graphical user interface (GUI) systems such as Windows Presentation Foundation (WPF), Windows Forms (WinForms), and Microsoft XNA can all be used to develop front-end applications that utilize the multi-touch framework. To support various presentation systems, the TACTUS software framework maintains two modes of data communication: polling and events. Traditional WinForms applications use events to communication object information; whereas polling is more common for simulations or video games, where input devices are continuously queried.

4.1 Image Processing

At the core of the software, is an image processing system that converts raw camera data into points of interaction. The image processing system runs in its own software thread, and captured frames are sent through the processing pipeline depicted in Figure 2.

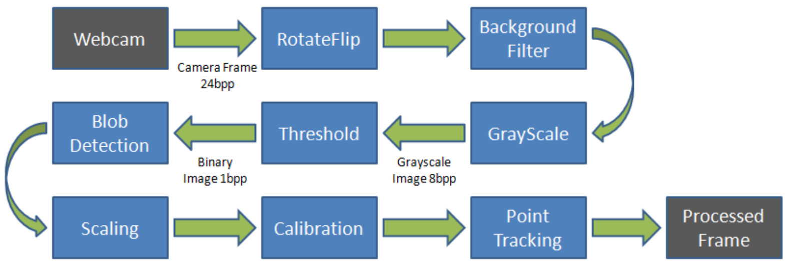

Figure 2. Image processing pipeline

Processing begins by capturing an image from a video source – a Microsoft DirectShow compatible device. The camera's device enumeration, frame rate, and resolution are specified through the framework's XML configuration file. The *RotateFlip* step transforms the image vertically and horizontally, as specified in the configuration file, orienting the image to match the projector.

During initialization, the software library captures a set of frames, while the surface is quiescent, and combines them to form a background image. The *Background Filter* step subtracts this background image from the active frame, thus removing noise from the image. Noise originates from ambient infrared light, "hot-spots" produced by the projector, oils and debris on the waveguide and compliant surface, and from light unintentionally escaping the waveguide. The TACTUS software library allows the user to recapture the background image at any time and does not force the restart of the image processing system to compensate for a dynamic lighting environment.

Most webcams capture images in color, typically at 24 bits per pixel (bpp). The *Grayscale* action converts a color frame to grayscale (8bpp). This step can be removed if the camera natively captures images in grayscale – the format required for the subsequent *Threshold* filter, which further isolates pixel values to black or white (fully *on* or fully *off*). Pixels below the configurable threshold value are treated as *off* and pixels above as *on*. The resulting 1bpp black & white image is sent to the *Blob Detection* process, which groups neighboring *on* pixels into *blobs*. Blobs are the regions of the image that we consider for potential points of surface interaction. The blob detector filters out blobs below a minimum width and height, as specified in the configuration file.

Scaling adjusts the dimensions of the image to correspond to the resolution of the image projected onto the multi-touch surface. The *Calibration* step then adjusts for differences between the interaction surface, projector, and the camera that create discrepancies between the points touched on the surface and the location of those points in the camera frame. By sampling points at the four corners of the surface, we can construct a transformation matrix and generate a lookup table with the corrected location for every point of interaction. This table can be serialized, and the resulting file specified in the XML configuration for automatic loading by the framework.

The final phase of the image processing pipeline is *Point Tracking*, which takes the detected, scaled, and calibrated blobs and abstracts them into *FtirPoint* objects. An FtirPoint object has attributes including: a globally unique identifier (GUID), location, timestamp, bounding box, speed, direction, and duration. Each FtirPoint represents a single point of interaction with the surface, and it is this data that is most useful for multi-touch applications. With the location and bounds of an FtirPoint we can perform hit testing – testing an area on the screen for an interaction point – through simple rectangle intersection. The point tracking process also labels each FtirPoint with a GUID that is maintained as long as the interaction point is present. To track a point across frames, we again perform rectangle intersection between the previous and current frame's FtirPoints. Points that intersect are considered the same point, and differences in location and time are used to calculate the point's speed, direction, and presence duration. Detected points that do not intersect with previous points are assigned a new GUID. This process allows points to split and merge, and enables gestural interaction with the surface. However, the performance of this technique is tied to the quality of the camera and the compliant surface.

The purpose of the compliant surface is to improve coupling between the user's fingers and the waveguide. If that coupling is poor, the user's fingers will "stutter" across the surface, and will not continuously reflect IR to the camera. The gaps between images would cause the point tracking system to re-label what would otherwise be the same point. The framework provides a stutter-correction system that tracks points within a time-window for label reuse. Tracked points that become absent from a camera frame are transferred to a "pending disposal" collection for a user-configurable time (250 millisecond default). Newly detected points are matched against this collection, again through rectangle intersection, before they are assigned a new GUID. In this fashion, the framework will reconstitute a point that becomes briefly disconnected from the surface. Stutter mitigation, however, does not address a camera with a slow frame rate. If the user's fingers move across the surface faster than the camera can track, the software library will label the points as disconnected. Future work on the library will attempt to address this through point prediction.

Exiting the image processing pipeline is the set of currently detected FtirPoints. Figure 3 shows a camera frame as it is passed through the image processing pipeline. Specifically, the figure displays: the raw camera frame (a), the background filtered image (b), the threshold image (c), and the fully processed FtirPoints displayed on the multi-touch surface (d).

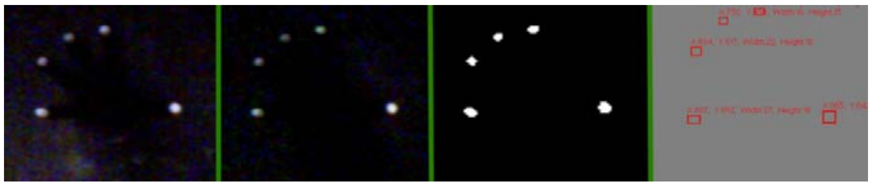
(a) (b) (c) (d)

Figure 3. A camera frame passed through the image processing pipeline: raw camera frame (a), background filtered (b), threshold (c), processed points (d)

4.2 Framework Features

While the image processing system forms the heart of the TACTUS software framework, there are a number of additional features that can aid in the creation of multi-touch applications including: multi-platform communication, 2D/3D graphics, pen/writing-style interaction, and gesture recognition.

The TACTUS software library is built on two open-source libraries: the Bespoke Open Sound Control Library (OSC) [11] and the Bespoke 3DUI XNA Framework [12]. OSC is an open, lightweight, message-based protocol that enables, for example, multi-touch data to be transmitted over a network. The Bespoke 3DUI XNA Framework is a software library for enabling research in game development and 3D user interaction (3DUI). The TACTUS software framework employs this library as a presentation layer for multi-touch applications; allowing games and simulations to be constructed with multi-touch input.

Another interesting feature of the TACTUS software system is its support of pen/writing-style interaction. Pen-style computing refers to human-computer interaction through the digital representation of ink or writing, typically input through a computer stylus [13]. Ordinarily, pen-computing is single-touch – where input is collected from only one location at a time. This is a degenerate case of multi-touch, where we constrain the input and treat interaction points as digital ink. The TACTUS software library collects ink data into *Stroke* objects which can be used for 2D recognition. TACTUS provides a machine-learning system for training and classifying stroke data based on work by Rubine [14].

5 Case Studies

The TACTUS software library has been utilized in the creation of many multi-touch applications, and across a variety of domains. This section discusses four projects built with the framework, pictured in Figure 4: SurfaceCommand (a), InkDemo (b), Waterfall (c), and the TACTUS mouse emulator (d).

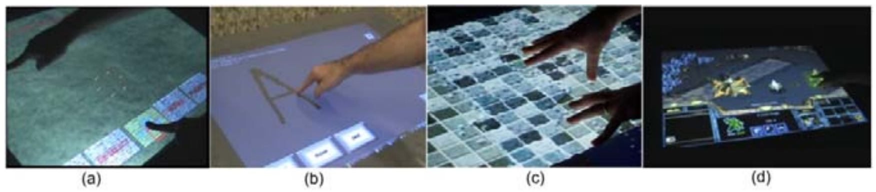

Figure 4. Multi-Touch applications: SurfaceCommand (a), InkDemo (b), Waterfall (c), and the TACTUS mouse emulator (d)

SurfaceCommand is a multi-touch demonstration styled after real-time strategy games. Built using the Bespoke 3DUI XNA Framework, with the TACTUS multi-touch extensions, SurfaceCommand presents a 3D battlefield viewed through an orthographic virtual camera. The user can pan around the battlefield by sliding two or more fingers across the display and zoom into and out of the map with "pinch" gestures – a motion whereby two interaction points are moving in roughly opposite direction, either toward or away from each other. Spaceships within the battlefield can be selected and moved about the map with single interaction points; and multiple ships can be selected and deselected with mode buttons along the bottom of the display. This simple application explores techniques that could be used within a real-time strategy video game and was developed in just four days.

InkDemo, pictured in Figure 4b, demonstrates the pen-style interaction of the TACTUS framework. Stroke data is collected when the user provides only a single point of interaction. As the user slides a finger across the display, simple block-style lines are generated to visualize the underlying stroke data. A set of strokes can be labeled and committed to the TACTUS symbol recognition system. With a sufficient number of training samples, the user can then classify an unlabeled set of strokes.

Our third application, Waterfall, is an example of multi-platform communication and the rapid development capability of the TACTUS software system. The application is a fluid-dynamics demonstration, where simulated water flows down an inclined surface and can be perturbed by multi-touch surface interaction. Users interact with the water to form "dams" with their hands and fingers. The simulation was developed in C++ and rendered with the OGRE game development platform [15]. The multi-touch input was serialized via Open Sound Control, as described in section 4.2, and received by the simulation using an open-source C++ OSC implementation. The integration effort took only three days from start-to-finish.

The last example demonstrates the TACTUS mouse emulator – an application that allows the use of a multi-touch surface to control traditional, mouse-driven Windows applications. The mouse emulator associates a set of gestures with common mouse commands, as listed in Table 2.

Table 2. Mouse emulator gesture mappings

Function	Gesture Description
Left Click	Quick tap on the surface with one finger.
Left Click (alternate)	While holding down a finger, tap another finger to the left side of the first.
Drag	Perform a Left Click (alternate) but do not release the left side press. Drag both fingers to the destination and release.
Right Click	While holding down a finger, tap another finger to the right side of the first.
Double Click	Tap two fingers at the same time.
Mouse Wheel Scroll	While holding down a finger, drag another finger vertically and to the right side of the first. Dragging up scrolls the mouse wheel up and vice versa.
Alt-Tab	While not a mouse command, the Alt-Tab command is a useful Windows feature that switches between applications. To perform an Alt-Tab, hold down a finger and drag another finger horizontally above the first. Dragging to the left moves backward through the list of active applications and dragging to the right moves forward.

Figure 4d shows the emulator in use with the commercial video game *Starcraft* by Blizzard Entertainment. This popular real-time strategy game is controlled through the mouse and keyboard; but using the TACTUS mouse emulator, one can play Starcraft through a multi-touch surface. Videos of these, and other TACTUS demonstrations, can be found at http://www.bespokesoftware.org/.

6 Conclusions

In summary, while multi-touch technology has generated considerable excitement and offers the potential for powerful new interaction techniques, the researcher must overcome a significant obstacle for entry into this field – obtaining a multi-touch hardware and software research platform. Few commercial options exist, and there are deficiencies in academic literature on constructing such a platform. This paper describes the requirements of a multi-touch research system and presents TACTUS, a hardware and software testbed that enables research in multi-touch interaction. The testbed discussed offers insight into the construction of a robust, low-cost multi-touch surface and the development of an extensible software system for the rapid creation of multi-touch applications.

This work is supported in part by the National Science Foundation under award number DRL0638977.

References

1. Dietz, P., Leigh, D.: DiamondTouch: A Multi-user Touch Technology. In: 14th ACM Symposium on User Interface Software and Technology, pp. 219--226. ACM, New York (2001)
2. Han, Y.J.: Low-cost Multi-touch Sensing through Frustrated Total Internal Reflection. In: 18th ACM Symposium on User Interface Software and Technology, pp. 115--118. ACM, New York (2005)
3. Wilson, A.: TouchLight: An Imaging Touch Screen and Display for Gesture-based Interaction. In: 6th International Conference on Multimodal Interfaces, pp. 69--76. ACM, New York (2004)
4. Microsoft Surface, http://www.microsoft.com/surface/
5. TouchLib, http://nuigroup.com/touchlib/
6. Martin, K., Ross, B.: reacTIVision: A Computer-vision Framework for Table-Based Tangible Interaction. In: 1st International Conference on Tangible and Embedded Interaction, pp. 69--74. ACM, New York (2007)
7. BBTouch, http://code.google.com/p/opentouch/
8. Davidson, P. and J. Han: Synthesis and Control on Large Scale Multi-touch Sensing Displays. In: The 2006 Conference on New Interfaces for Musical Expression, pp. 216--219. IRCAM, Paris, (2006)
9. Kim, J., Park, J., Kim, H., Lee, C.: HCI(Human Computer Interaction) Using Multi-touch Tabletop Display. In PacRim Conference on Communications, Computers and Signal Processing, pp. 391--394. IEEE Press, New York (2007)
10. Tse, E., et al.: Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In: Proceedings of the working conference on Advanced visual interfaces, ACM (2006)
11. The Bespoke Open Sound Control Library, http://www.bespokesoftware.org/osc/
12. The Bespoke 3DUI XNA Framework, http://www.bespokesoftware.org/3dui/
13. Bowman, D., Kruijff, E., LaViola, J., and Poupyrev, I.: 3D User Interfaces: Theory and Practice, (2004): Addison Wesley
14. Rubine, D.: Specifying gestures by example. in International Conference on Computer Graphics and Interactive Techniques. (1991)
15. Torus Knot Software. OGRE - Open Source 3D Graphics Engine, http://www.ogre3d.org/