University of Amsterdam
Faculty of Science

Master Thesis Software Engineering

Interaction with 3D VTK widgets

A quantitative study on the influence of 2DOF and 6DOF input devices on user performance

René Wiegers

Host Organization:
Center for Mathematics and Computer Science (CWI)

Thesis Supervisor:	Prof. Dr. Jan van Eijck
Internship Supervisor:	Prof. Dr. Robert van Liere
Availability:	Unclassified
Date:	11 August 2006

Preface

This thesis describes the graduation project of René Wiegers, which concludes the Master program Software Engineering at the University of Amsterdam. The graduation project has been performed at the Center for Mathematics and Computer Science (CWI) located in Amsterdam, Netherlands.

CWI, founded in 1946, is the national research institute for mathematics and computer science in the Netherlands. CWI performs frontier research in mathematics and computer science and transfers new knowledge in these fields to society in general and trade and industry in particular. CWI comprises several domain specific research groups. The research described in this document will be conducted by order of the research group "Visualization and Virtual Reality". This group studies those mechanisms that support the development of visualization techniques at the systems and user levels. These techniques include those found in virtual reality, multi-dimension visualization and information visualization.

Acknowledgments

Several people contributed to the successful completion of the project and this resulting thesis. I would therefore like to express my sincere appreciation for the help and motivating words I received during the last few months. First of all, I would like to thank my supervisors Robert van Liere and Jan van Eijck for their excellent support and guidance during the project. Furthermore, I would like to express my appreciation to Chris Kruszynski for kindly assisting me with the design and implementation of the VTK software. I would also like to thank Arjen van Rhijn and Ferdi Smit for their pointers regarding the experimental setup and result analysis. Finally, I would like to thank my fellow student Maarten Pater for reviewing my thesis.

René Wiegers
Amsterdam, August 2006

Summary

3D graphics software like VTK is increasingly used in the scientific community in order to make the information behind enormous amounts of data more accessible. For interaction with the data, VTK provides 3D widgets, which are 3D objects that live in the same coordinate space as the 3D objects they manipulate. These widgets are manipulated by 2D input devices like standard PC mice, which traditionally have been designed with 2D interaction in mind. These devices can only move in the (x,y) plane and therefore only have two degrees of freedom (DOF) available for interaction. 3D input devices however, are especially designed for 3D interaction and have six DOF available for interaction. Three degrees describe their position and three degrees describe their orientation in 3D space. One of the main questions in 3D graphics systems is therefore related to the effectiveness of 3D interaction. In order to determine the pros and cons of using 3D input devices for the manipulation of 3D VTK widgets, a user experiment was conducted to investigate the following main research question:

What is the influence of the number of degrees of freedom of input devices on overall task completion time and overall task accuracy when manipulating 3D VTK widgets?

We hypothesized that the overall task completion time and the overall task accuracy will be improved when an input device with more degrees of freedom is used to manipulate the widgets. In order to determine the source of any differences in task completion time and task accuracy between 2DOF input and 6DOF input, we also investigated the following sub-question:

Is there a significant difference between 2DOF input and 6DOF input regarding both the average time spent on respectively translation, rotation and scaling, and the average accuracy by which each of these are performed?

We hypothesized that for widget rotation and translation, 6DOF input is more accurate and faster than 2DOF input. For widget scaling, we hypothesized that 2DOF input is more accurate and faster than 6DOF input.

For the experiment, subjects had to perform a docking task, requiring translation, rotation and scaling of a source object by manipulating a 3D widget, in order to match the position, orientation and size of a similarly shaped target object. Subjects performed the docking task both with 2D and 3D input. Both quantitative and qualitative results were recorded. The results confirm that overall task accuracy and overall task completion time are improved when 3D input is used to manipulate the widget. Inspection of the individual results for translation, rotation and scaling shows that average rotation time is improved when 3D input is used, without loss of accuracy. Regarding translation, 2D input turned out to be faster, while 3D input was more accurate. Regarding scaling, there was no significant difference between 2D and 3D input. Finally, on basis of the results, a combination of 2D and 3D input is suggested for widget manipulation, where translation and rotation are performed by the 3D input device and scaling by the 2D input device.

Contents

Chapter 1

Introduction

The research project '*A quantitative study on the influence of 2DOF and 6DOF input devices on user performance*', described in this work, is based on [15], which argues the need for a multi modal interface for the visualization library VTK. Goal of our research is determining the pros and cons of using 2D vs 3D input devices when manipulating 3D widgets in VTK.

1.1 Background

Today's computers generate enormous amounts of data. To make the information behind those data more accessible, visualization software can be used. Visualization software transforms the data into pictures, which can then be interactively transformed and explored in order to get insight into the information. An example of visualization software is the Visualization Toolkit (VTK) [12]. VTK is an open source, freely available software library for 3D computer graphics, image processing, and visualization, that is widely used for scientific visualization [15]. Currently, VTK is emerging as a standard in the scientific community for the visualization of scientific data and will therefore be the selected graphics library for this research.

1.1.1 3D Widgets

VTK defines all interaction in terms of '3D widgets' (e.g. point widget for point selection, plane widget for setting cutting planes, etc) [3]. A widget can be defined as an encapsulation of geometry and behavior used to control or display information about application objects [3]. 3D widgets are three dimensional objects that live in the same coordinate space as the 3D objects they manipulate, thereby simplifying interaction with 3D objects by making the interaction more intuitive by providing fast semantic feedback.

Figure 1.1 illustrates the concept of a 3D widget: The user wants to perform a certain task, say positioning an object. Therefore, he or she uses an input device to manipulate the 3D widget that is represented in the virtual world by its geometry. The widget responds to the input by a predefined behavior and produces output data describing the manipulation. In turn, the output the widget produces is used by the object to respond accordingly.

Figure 1.2 represent an example of a widget that can be used to position or orient an object the widget is bound to. This widget comprises several 'handles', represented by the vertical and horizontal blue bars. These bars can be used to translate(position) or rotate the widget, thereby translating or rotating the object.

After this short introduction into visualization and 3D widgets, the next section will discuss the problem definition.

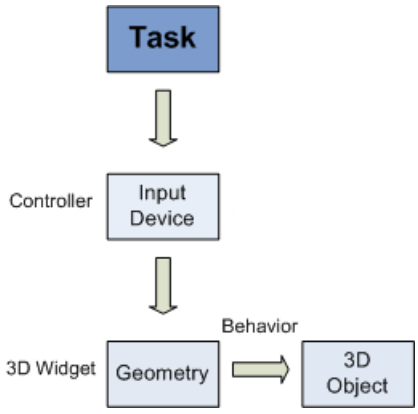

Figure 1.1: Concept of a widget

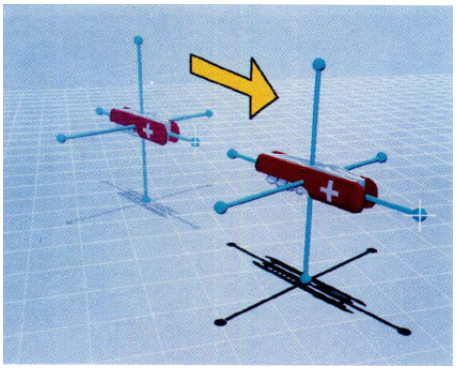

Figure 1.2: 3D Widget, adopted from [3]

1.2 Problem Definition

One of the major questions in 3D graphics systems is related to the effectiveness of 3D interaction. Interaction devices are characterized by their degrees of freedom (DOF), describing the possible interaction space. Because of the fact that 2D input devices[1] like standard PC mice are bound to the (x,y) plane, they only have two degrees of freedom available for interaction (figure 1.3). This means that two operations are required to manipulate three variables. Take for example the positioning of a point in 3D space, this would require two sequential actions: first a translation in the (x,y) plane, followed by a translation in the z direction. One commonly used design for mapping three variables (x,y,z) onto a mouse, allows two of the variables (x,y) to be input simultaneously in normal operation mode and the third (z) to be controlled through a mode change button that temporarily turns the mouse into a one-dimensional slider [7].

On the contrary, 3D input devices[2] have 6DOF. These describe translation of the device along any of the perpendicular axes (x,y,z), as well as rotation of the device around any of those axes (figure 1.4)

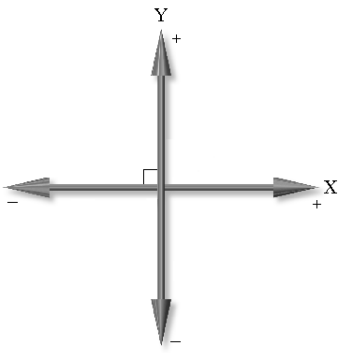

Figure 1.3: 2DOF interaction space

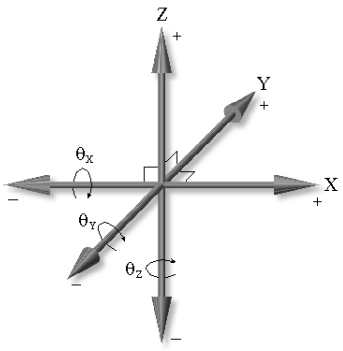

Figure 1.4: 6DOF interaction space

Thus, compared to 2D devices, 3D devices can potentially represent a broader range of input values simultaneously, requiring less sequential actions. For example, positioning of a point in 3D space would only require one atomic action with a 6DOF device.

[1] In this work 2DOF input is also referred to as 2D input.
[2] In this work, 6DOF input is also referred to as 3D input.

On one hand, a 3D device seems to be very natural and intuitive to use in a 3D environment [9]. On the other hand, 2D devices are very easy to use.

To determine the effectiveness of 3D interaction, several experiments have been performed.

In [15], an experiment is conducted to prove the need for a multi modal interface for interaction with 3D applications. In the experiment, 6DOF input devices were used for picking and rotation. Manipulation of the visualization was provided by the use of 3D widgets. The results of the study indicate that direct 3D interaction with the application was much easier than interaction in 2D. Our research is a follow up of this study. Where this work describes global techniques for improving 3D interaction in VTK, our work represents a formal study comparing 2D and 3D input devices when manipulating VTK widgets.

In [2], a user experiment is conducted, comparing the user performance of a standard mouse interface with unimanual and bimanual versions of an interface that use 6DOF clastic rate-control devices for object and camera control in a 3D object docking task. They found that for novice users, the mouse interface performed better than either of the 6DOF interfaces. They noted that novice users found the 6DOF interfaces difficult to adjust to. However, expert users were more comfortable controlling the 6DOF interface. They found that the bimanual interface was most beneficial when the task was complex and required many epistemic actions.

In [6] a formal user experiment is conducted, comparing 3D input interfaces to a standard mouse interface. The experiment requires the subjects to manipulate a source object so that its orientation matches the orientation of a similarly shaped target object. The 3D input devices can rotate the object directly using absolute orientation, where the object matches the orientation of the input device. The mouse interface uses a virtual sphere and arcball to rotate the object. They found that for the orientation matching task, task completion time was shorter when a 3D input device was used, without any statistically detectable loss of accuracy.

In [7] it is suggested that selecting an appropriate input device for an interactive task requires looking at the perceptual structure of the task, the device and the interrelationship between the perceptual structure of the task and the control properties of the device. This structure can either be integral, showing Euclidean movement (i.e. diagonal movement) or separable, showing a city block pattern (i.e. movement is constrained along one dimension at a time). They hypothesize that user task performance is improved when the perceptual structure of the task matches the control structure of the device. An experiment was conducted, in which subjects performed two tasks with different perceptual structures, using two input devices with correspondingly different control structures, a three-dimensional tracker and a mouse. The integral task required changing the x,y location and size of a cube. The separable task required changing the x,y location and color of the cube. Both speed and accuracy were recorded. The results supported the hypothesis, showing that the mouse interface performed better in the separable task and that the tracker performed better in the integral task.

Since interaction with 3D widgets in VTK by means of 6DOF input devices is still an open research topic, this will be the subject of the research.

1.3 Research

In order to determine the impact of using 6DOF input devices for the manipulation of 3D widgets on user performance, an experiment will be conducted which compares 2DOF and 6DOF input devices for the manipulation of 3D VTK widgets.

The next section will describe the research goal and our hypotheses, followed by a description of the approach that has been taken.

1.3.1 Research goals

This study was conceived to investigate the following question:

- *What is the influence of the number of degrees of freedom of input devices on overall task completion time and overall task accuracy when manipulating 3D VTK widgets?*

Hypothesis H1: It is hypothesized that the overall task completion time and the overall task accuracy will be improved when an input device with more degrees of freedom is used for widget manipulation.

In order to determine the source of any differences in task completion time and task accuracy between 2DOF input and 6DOF input, we will investigate the influence of the basic geometric transformations underlying 3D interaction (translation, rotation and scaling) on these variables. Therefore, the following sub-question will be investigated:

- *Is there a significant difference between 2DOF input and 6DOF input regarding both the average time spent on respectively translation, rotation and scaling, and the average accuracy by which each of these are performed?*

Hypothesis H2: It is hypothesized that for widget rotation and translation, 6DOF input is more accurate and faster than 2DOF input. For widget scaling, it is hypothesized that 2DOF input is more accurate and faster than 6DOF input.

To find an answer to these questions, a formal user study comparing interactive tasks in VTK is conducted.

The next section will briefly describe the approach that has been taken during the project.

1.3.2 Approach

In order to get familiar with the problem background and context, a literature study was conducted at the start of the project.

The literature study focused on three main areas, namely Human Computer Interaction (HCI), 3D Widgets and user experiments involving input devices. The results of the literature study formed the basis for the remainder of the project.

After the literature study, the VTK toolkit was extended with support for 3D input devices. Based on the requirements and literature study, a design was made, which was thereafter implemented. Parallel to the VTK implementation, the experimental setup was designed, so that the application that was used during the experiment could be developed right after completion of the VTK implementation.

When construction of the application for the experiment was completed, the user experiment was conducted. Finally, the results of the experiment were analyzed.

The remainder of this document will describe these steps and their results in detail.

Chapter 2

Extending VTK

This chapter will describe how the VTK toolkit was extended with support for 3D input devices. To provide the new functionality, the VTK C++ class hierarchy was extended with new classes. The next section lists the requirements that formed the basis for the architecture and design of the new software, followed by a discussion of the resulting architecture and design.

2.1 Requirements

1. **The implementation should be upwards compatible with future versions of VTK**
   Migration to a new release of VTK should be possible without modification of the VTK source code.

2. **The design should allow multiple tracker implementations**
   The source code will be shared with the VTK community, so that other parties may implement their specific tracking system. Therefore, the design should not be specifically targeted towards the CWI tracking system.

3. **The design should allow multiple 3D input device implementations**
   Since there is a large variety of 3D input devices. The design should not be specifically targeted towards the CWI input devices, so that support for other 3D input devices can be easily implemented.

4. **Technical aspects should be hidden from the user**
   Technical aspects of Virtual Reality systems (support for input devices like trackers and pedals), should be hidden from the user.

The following sections will discuss how the resulting architecture and design satisfy these requirements.

2.2 Architecture

Since the implementation should be upwards compatible with future versions of VTK, modifying the VTK source code was not an option. Modifying existing VTK classes would make the implementation heavily susceptible to future changes to those classes. Therefore, the strategy was chosen to add new functionality to VTK by subclassing existing VTK classes. Although subclassing would make the implementation less susceptible to future changes to the VTK code, a possible drawback of using derived classes could be code duplication: private member data or methods that are inaccessible for derived classes and that therefore have to be reimplemented in the derived classes. However, since the design of VTK itself is also based on subclassing, most of the functionality of the superclass can be used by the derived class (i.e., most of the attributes and methods in VTK classes are declared as either public or protected), so this was not an issue and therefore considered the best alternative.

Figure 2.1 illustrates the integration of the VTK extension libraries with the standard VTK architecture [13]. The extension consists of C++ class libraries that are build on top of the standard VTK C++ core libraries. This implies that the extension code is not woven with the VTK code, making the code less susceptible to future changes to VTK.

The next section will discuss the design of the extension libraries in more detail.

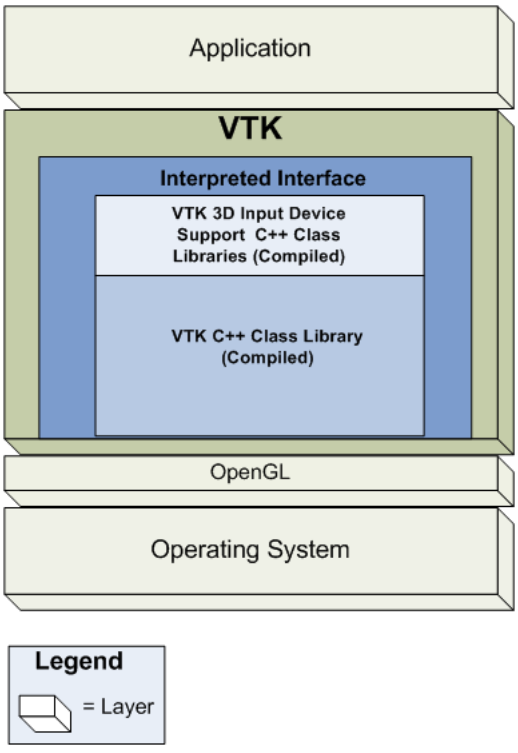

Figure 2.1: Integration with VTK architecture

2.3 Design

Several strategies were followed during the design and several trade offs were made in order to satisfy the requirements. These will be discussed in the following section, followed by a detailed discussion of the design.

2.3.1 General design strategies

As described above, the design should be flexible, so that multiple tracker implementations and 3D input devices can be used. To meet these requirements, abstract classes were used as strategy.

Abstract classes

As said, there are several types of trackers and 3D input devices, which all have their own characteristics. The consequence of these differences, is that it is not feasible to represent the various trackers or 3D input devices by one implementation.

However, the devices can be grouped on semantic level: The provided functionality is (nearly) identical, it only varies in the way it is implemented. To provide flexibility by supporting multiple tracker and 3D input device implementations, the design decision was made to define abstract base classes for both the trackers and 3D input devices. These abstract classes define (and partially implement) a device interface. General device characteristics are implemented in the abstract base class to avoid code duplication.

Device implementations deriving from the abstract class however, are forced to implement those methods that implement device-specific characteristics.

So, by providing an interface, any tracker and 3D input device may be used, as long as the implementation conforms to the interface.

Object factories

The decision was made to construct an object factory for the integration of the new code with the existing VTK code. Object factories allow an object to be replaced by the implementation of a specified object at runtime and can therefore be used to add the new functionality to VTK without modification of existing VTK code: instead of loading a standard VTK class, a new class can be loaded that is part of the new functionality.

Detailed design of the extension libraries is reflected in figure 2.2. The following sections will discuss the design rational of the libraries and the functionality they provide.

2.3.2 libvtkTracker

As can be seen from figure 2.2, libvtkTracker provides most of the functionality. It provides support for tracker event handling, support for 3D input devices and abstraction of 3D input device events. It also contains the object that is responsible for the integration of the new code with the standard VTK code by replacing its event loop.

Abstraction of Tracker

There are many tracking devices. To localize knowledge of the tracker used, the abstract base class vtkTracker has been defined, which must be derived by specific tracker implementations. These derived classes define how tracker events are received (e.g. through sockets or read from a

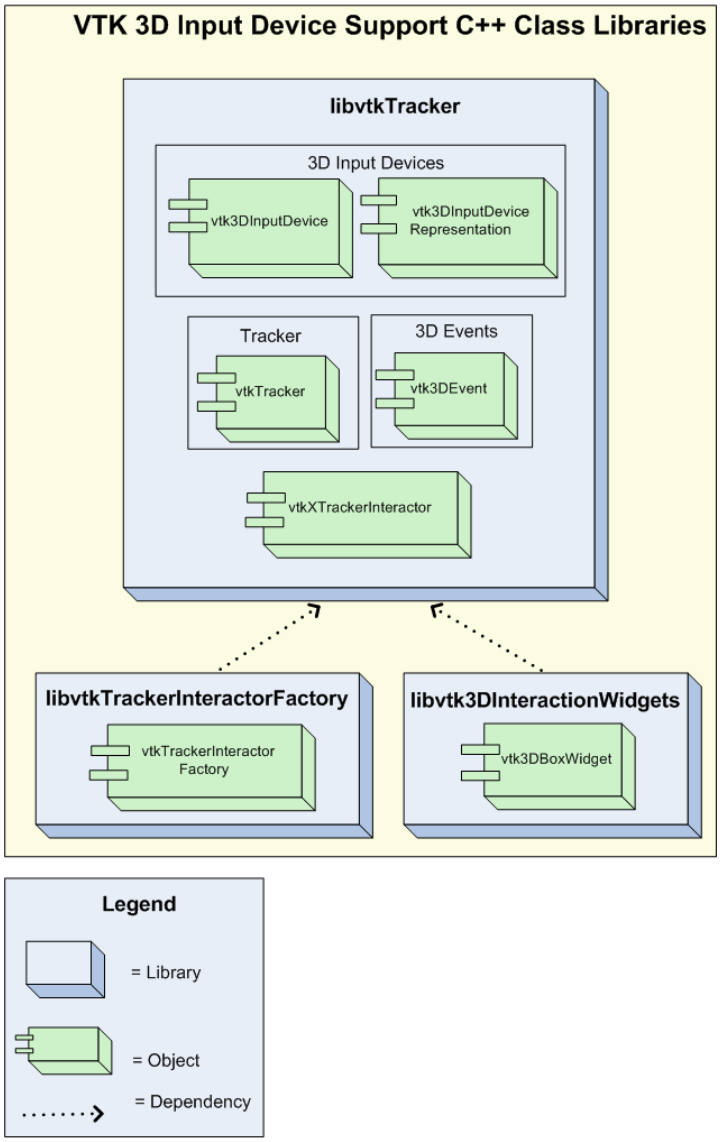

Figure 2.2: Extension Libraries

file on the file system) and how the received tracker data can be encapsulated into the appropriate vtk3DEvent (to be described shortly).

Abstraction of input devices

There are many variations of 3D input devices, ranging from commercially available 3D input devices, to custom made 3D input devices, like the ones used in the conducted experiment. All these input devices have their unique characteristics, like provided functionality or shape. To offer flexibility in the choice of 3D input device, two interfaces were defined, one for the behavior of the 3D input device and one for the visual representation of the input device. The advantage of separating the behavior from the appearance of the 3D input device, is that changes in the appearance are localized, so that these can be changed without knowing anything about the type of 3D input device the visual appearance is applied to. Furthermore, several devices can share the same appearance by setting the representation, so that code duplication is reduced to a minimum.

The two classes that make up the device abstraction are vtk3DInputDevice, which describes the behavioral part, and vtk3DInputDeviceRepresentation, which describes the device appearance. Currently, two device types have been defined. A 3D input device can either be an orientation device, used for widget rotation, or a selection device, used for general widget manipulation.

Abstraction of 3D input device events

For event dispatching, VTK follows the command/observer paradigm. When a VTK object is interested in receiving a specific event from an object, it can denote this interest by registering itself with the object that is responsible for invoking the event, along with a callback routine that handles the event. When the object of interest invokes the event, it calls the callback routine of each of the objects that have registered for the event. In turn, these callback functions dispatch the event to the appropriate methods for further handling. What makes these callback functions interesting for our tracker event implementation, is the fact that they are able to receive arbitrary objects, along with the event information.

This allows us to construct an object that encapsulates the technical details of tracker events, which can be sent along with the event. The advantage of this approach, is that the build-in VTK event dispatching mechanism can be used for event dispatching of 3D input device events. Therefore, the design decision was made to encapsulate all 3D input device event related information into a vtk3DEvent object.

For each 3D input device, the vtk3DEvent object provides a description of the device type (i.e. selection or orientation device), its current position, its current orientation, button state (i.e. pressed or not pressed), and several other attributes that are accessible to widget implementations. Figure 2.3 illustrates how these vtk3DEvent objects flow from source to destination:

1. Data is received from the tracker by a vtkTracker object.
2. The vtkTracker object encapsulates the data into a vtk3DEvent object and dispatches it to the vtk3DInputDevice objects.
3. The vtk3DInputDevice that belongs to the device that generated the event, appends additional attributes to the vtk3DEvent object, like a copy of its previous position and orientation. The vtk3DInputDevice objects then informs its vtk3DInputDeviceRepresentation object to update its visual representation on the screen accordingly.
4. Finally, the modified vtk3DEvent is then sent back to the vtkTracker object, which forwards the event to the vtkXTrackerInteractor (5) (see section 2.3.3), which in turn forwards the event to the widgets (6).

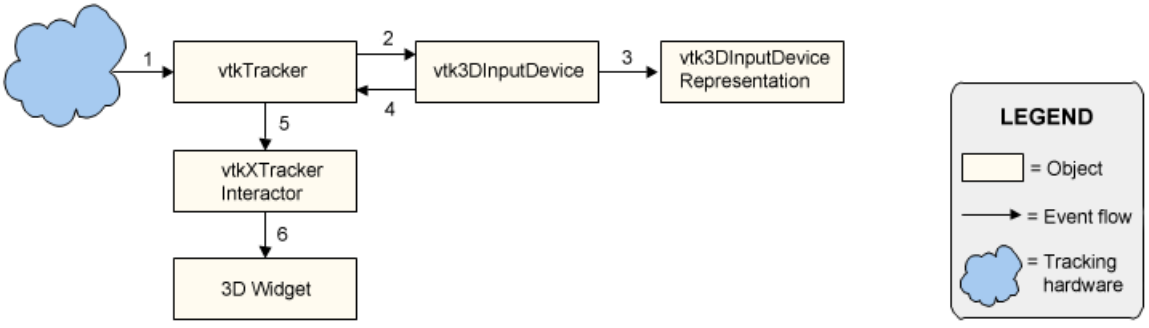

Figure 2.3: Event flow

2.3.3 libvtkTrackerInteractorFactory

The libvtkTrackerInteractorFactory library contains the factory that is responsible for the runtime replacement of the vtkXRenderWindowInteractor object by the vtkXTrackerInteractor object (which is part of the vtkTracker library). vtkXRenderWindowInteractor is an X Windows specific VTK class and is responsible for the translation of X Windows specific events (like mouse/keyboard events) to VTK Events. In turn, vtkXRenderWindowInteractor forwards these VTK events to vtkInteractorObserver objects (like widgets), which observe these events.

In order to be able to receive tracker events, vtkXRenderWindowInteractor is replaced by vtkXTrackerInteractor, which is a subclass. vtkXTrackerInteractor overrides the event loop of vtkXRenderWindowInteractor and checks for both X specific events and for tracker events.

2.3.4 libvtk3DInteractionWidgets

libvtk3DInteractionWidgets contains 3D widgets that support 3D input devices. Currently, the library only contains the vtk3DBoxWidget, which is a standard vtkBoxWidget which has been extended with support for 3D interaction.

The vtkBoxWidget defines a region of interest that is represented by an arbitrarily oriented hexa- hedron with interior face angles of 90 degrees. The object creates seven handles that can be selected and manipulated. The first six correspond to the six faces and can be used for scaling, the seventh is in the center of the hexahedron and can be used for translation. Figures 2.4-2.7 illustrate the functionality the vtkBoxWidget provides. Figure 2.4 shows the state of the cone before widget manipulation. Figures 2.5-2.7 show the cone after positioning, rotation and scaling respectively.

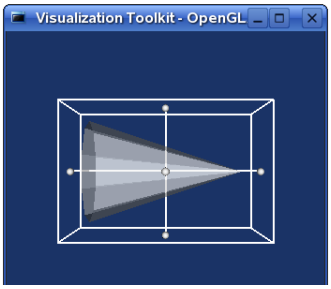

Figure 2.4: BoxWidget

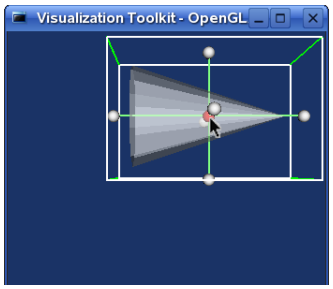

Figure 2.5: Positioning

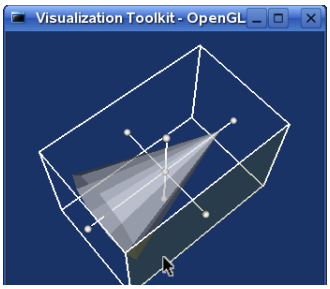

Figure 2.6: Rotation

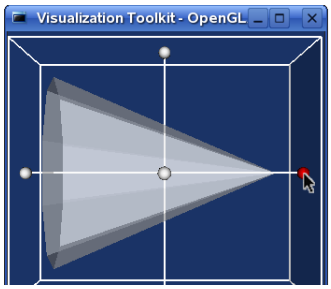

Figure 2.7: Scaling

The handles of the vtk3DBoxWidget can be manipulated by a selection device. The orientation of the vtk3DBoxWidget can be controlled by an orientation device. As long as a (predefined) button is pressed on the orientation device, the orientation of the widget will be identical to that of the orientation device.

To address the difficulties that users have with point location in 3D [15], the widget highlights the cursor of the corresponding 3D input device when it is within selection range. For the selection device, the cursor is highlighted when it is near one of the handles of the widget (figures 2.8 and 2.9). For the orientation device, the cursor is highlighted when it is within a certain distance of the widget (figures 2.10 and 2.11).

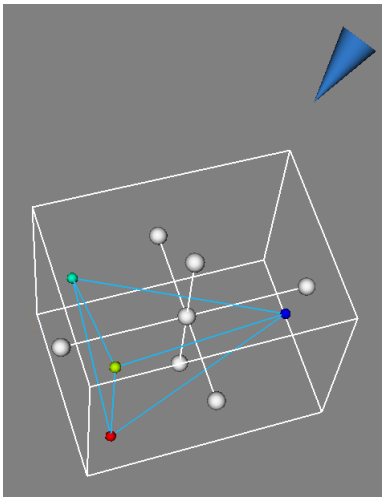

Figure 2.8: No selection possible

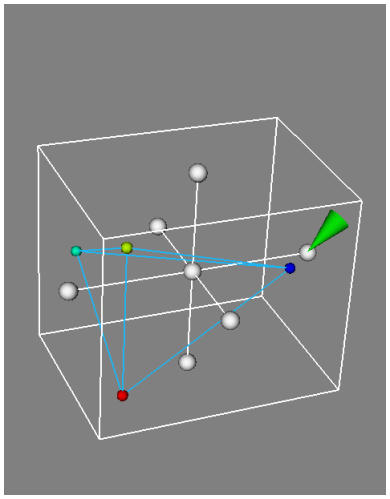

Figure 2.9: Selection possible

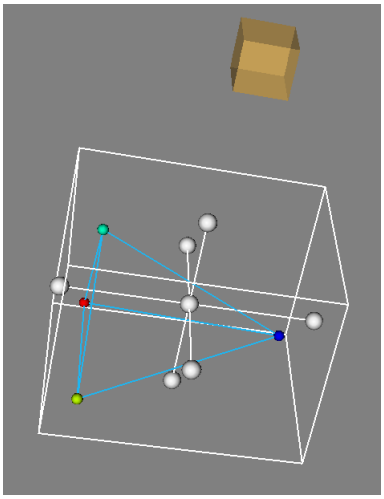

Figure 2.10: No selection possible

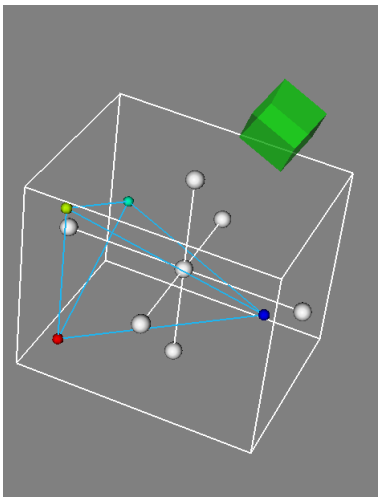

Figure 2.11: Selection possible

2.4 Evaluation

A great deal of the project was spent on design and implementation of both the new VTK libraries and the application that was needed for the experiment. Therefore, this section will discuss the state of the software at the end of the project and the software process that was followed during the project.

2.4.1 State of the software

Since it is the intention that the code will eventually be shared with the VTK community, great care was taken to follow the VTK standards with respect to coding style and documentation.

To give an impression of the size of both the code and documentation, the extension libraries comprise about 1653 effective lines of code (eLOC), measuring all lines that are not comments, blanks or standalone braces or parenthesis. In addition, the header files contain about 438 lines of documentation, which is used to automatically generate extensive browsable class documentation. In addition to the extension libraries, the code of the application that was constructed for the experiment comprises about 688 eLOC.

The current state of the software provides rudimentary support for 3D input devices for VTK and allows 3^{rd} parties to extend it with their own tracker and input device specific implementations. However, despite the fact that great care was taken during construction of the software to insure its quality, the software still needs extensive quality checking before proposing to incorporate the code into the standard VTK code by its maintainers. Due to time considerations, the software was tested only sufficiently enough to assure its suitability for the experiment.

2.4.2 Software process

At the start of the project, an attempt was made to construct the entire architecture and design of the software to serve as basis for the construction of the software. However, it proved to be rather complex and overwhelming to conceive the entire architecture and design at the beginning of the project.

Therefore the decision was made to adopt an approach that was more incremental in nature. Functionality was added to the software step by step. At the beginning of each increment, a design was made for the new functionality. When necessary, adjustments were made to the old design in order to incorporate the new functionality.

The advantage of this approach for this particular project, was that the project could be divided into sub-problems. During each increment, the focus lay on the problem at hand, making it a lot easier to handle. This led amongst others to greater productivity and helped to achieve project success.

Chapter 3

The experiment

3.1 Test environment

As tracking environment, the Personal Space Station (PSS) was used [10]. The PSS is a near-field virtual environment, where all interactive 3D tasks are realized directly with the hands or by using task specific graspable input devices. The PSS consists of a mirror in which stereoscopic images are reflected. The user reaches under the mirror to interact with the virtual world. The PSS uses optical tracking for interaction. Retro-reflective markers under infra-red (IR) lighting conditions are used to track objects located under the mirror. The graphics engine is a standard PC equipped with an ATI FireGL3 graphics board and a high resolution 22 inch Iiyma CRT monitor. The display resolution is set at 1280x1024 @ 120 Hz. The tracking engine is a PC equipped with two Leutron Vision PictPort H4D dual channel frame grabbers and two Leutron Vision LV-7500 progressive scan CCD-camera's. The cameras operate at a frequency of 60 Hz.

Figure 3.1: The Personal Space Station, adopted from [11]

3.2 Task description

For the experiment, subjects performed a 3D docking task, using stereoscopic vision. Given a source object, the docking task requires the subject to match the orientation, size and position of a similarly shaped target object. Object docking is an established task, allowing us to relate the results to previous work. Furthermore, it allows us to separately study translation, rotation and scaling performance [4]. In the setting of this experiment, subjects had to manipulate a 3D widget in order to accomplish the docking task.

As docking object, a tetrahedron was used (see figure 3.3). Each vertex of the tetrahedron was uniquely identified by its color. The docking objects could be distinguished by the color of the lines connecting the vertices. These were blue for the source object and red for the target object. For the scaling, translation and rotation of the source object, a modified version of the vtkBoxWidget with support for 3D input was used (see section 2.3.4).

In order to compare 2DOF input against 6DOF input, subjects performed the docking task with a standard 3-button PC mouse and with two 3D devices respectively.

The experimental setup for 2D input is depicted in figure 3.2. The docking task with 2D input is depicted in figure 3.3. To scale the widget in a particular direction, subjects had to manipulate one of the handles located at the sides of the widget. To position the widget in the (x,y) plane, the handle located at the middle of the widget had to be manipulated. Handle manipulation starts when the cursor is located over a handle and the left mouse button is pressed. By moving the mouse in a particular direction, the handle is manipulated. Manipulation ends when the mouse button is released. To position the widget in the z direction, a mode switch was required. The mode switch turned the mouse into a one dimensional slider, allowing the subject to translate the widget along the z axis. To perform the translation, the cursor had to be located within the boundaries of the widget. By pressing the right mouse button and moving the mouse up or down, the widget was translated along the z axis. Finally, rotation was performed by moving the mouse while holding the left mouse button when the cursor was within the boundaries of the widget.

Figure 3.2: 2DOF input setup

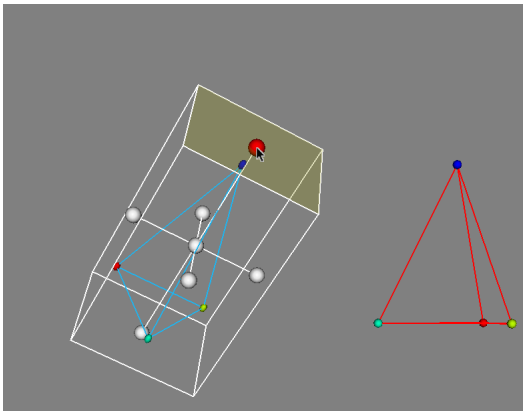

Figure 3.3: Docking with 2DOF input

For the docking with 6DOF input, a cube (figure 3.4) and a thimble (figure 3.5) input device were used. These devices could be used simultaneously. The cube was used for widget rotation and the thimble was used for translation and scaling. Participants held the cube in their non-dominant hand and slit the thimble over one of the fingers of their dominant hand. For the 6DOF setup, two foot pedals served as input device buttons.

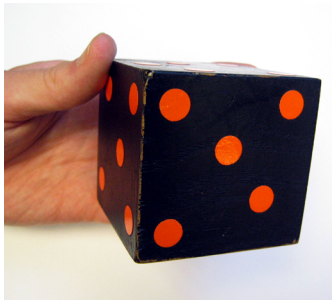

Figure 3.4: Cube 3D input device

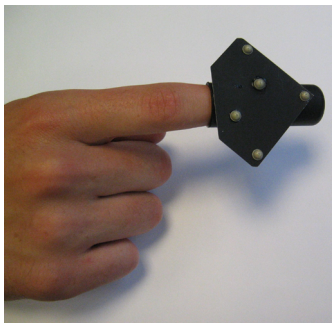

Figure 3.5: Thimble 3D input device

Rotation was possible when the cursor of the cube was within a certain distance of the widget. Rotation starts when the left pedal is pressed. When rotating, the widget takes the absolute orientation of the input device. Rotations ends when the left pedal is released. For translation and scaling, the cursor of the thimble had to be within a certain distance of the handles. Manipulation starts when the right pedal is pressed and ends when the right pedal is released. In contrast to the mouse interface, no mode switch was required for translation by a (x,y,z) vector.

The experimental setup for 6DOF input is depicted in figure 3.6. The docking task with 6DOF input is depicted in figure 3.7.

Figure 3.6: 6DOF input setup

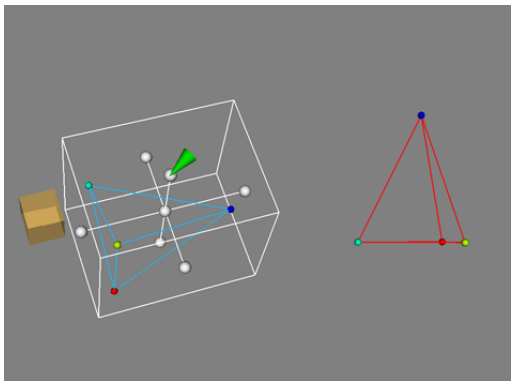

Figure 3.7: Docking with 6DOF input

3.3 Participants

Eight unpaid volunteers participated as test subjects during the experiment. All participants were required to be male and right handed. Furthermore, since unexperienced mouse users are hard to find, participants were required to be both experienced mouse users as well as experienced PSS users.

3.4 Procedure

The docking task consisted of a series of twelve trials for each interaction mode (i.e. 2DOF input vs 6DOF input) , resulting in a total of twenty four trials per subject. To counterbalance any learning effect, half of the subjects started the experiment with the 2DOF input device, and the other half of the subjects started the experiment with the 6DOF input. All subjects matched the same sequence of randomly generated target position, orientation and scale, thereby counterbalancing these variables for all users and input devices.

Before beginning the trials, the subjects received verbal instructions and a short demonstration regarding the task and interface. Before each trial series, subjects were given four training sessions to get familiar with the task and interface.

During the training session, subjects were encouraged to ask questions about the task or interface. During the trials however, the subjects performed the task unaccompanied.

The subjects were instructed that precision and speed were of equal importance and they were not given any feedback regarding proximity during the docking procedure. The subjects were not given any direction as to what was fast or accurate enough, so the precise definition of accuracy and speed was left to their own judgment. When satisfied with the docking, subjects pressed the spacebar to advance to the next trial.

After each session, participants completed a short questionnaire that interviewed the user about his impressions of the suitability of the input device for the particular task. This questionnaire also served as a short brake to minimize fatigue. At the end of the experiment, candidates were asked to indicate their preferred device for the docking tasks.

3.4.1 Logging

During each trial series, a main log file was kept that for each of the twelve trials recorded the duration of the docking task and the accuracy by which the docking task had been performed. How these metrics were defined is stated in the next paragraph.

Furthermore, a separate log file was kept for each interaction device during each trial series. Each time a button/pedal was pressed, the log file recorded in what way the subject was manipulating the widget (i.e. translation, rotation or scaling), along with the time the manipulation started/ended and the total duration of that particular manipulation.

These log files were used for analysis after the experiment was conducted. The main log file was the primary source for the analysis. The device specific log files were used to explain and underpin the results that analysis of the main log produced.

3.5 Performance Metrics

There were two performance metrics defined: task completion time and task accuracy.

For each trial, the task completion time was measured. For each trial series, these task completion times were averaged. Differences in task completion time between series belonging to the same docking task (i.e. 2DOF vs 6DOF input) would indicate that a subject could dock the object faster with one of the two input devices.

Task completion time was defined as the time between the moment the widget was manipulated for the first time and the moment the widget was manipulated for the last time before the subject pressed the spacebar to indicate satisfactory docking.

For each trial, the task accuracy was measured. For each trial series, these task accuracies were averaged. Differences in task accuracy between series belonging to the same docking task would

indicate that a subject could dock the object with more precision using one of the two devices.

Task accuracy was defined by three metrics. The first describes the positioning accuracy, the second describes the orientation error and the third describes the scaling error. The positioning accuracy is defined as the ratio between the initial distance between the source and target object and the distance between the two objects at the last moment the widget was manipulated before the subject pressed the spacebar to indicate satisfactory docking. Distance was defined as the total Euclidean distance across all three dimensions of the stimulus space, ranging from the center of the source object to the center of the target object.

The orientation error was defined as the shortest arc rotation between the orientation of the source object and the orientation of the target object, at the last moment the widget was manipulated before the subject pressed the spacebar to indicate satisfactory docking.

The scaling error was defined as the ratio between the x, y, z scale of the target object and the x, y, z scale of the source object respectively, at the last moment the widget was manipulated before the subject pressed the spacebar to indicate satisfactory docking.

3.6 Results

Eight candidates participated. All candidates were male and right-handed. All candidates had prior experience with the PSS and with the mouse. Three candidates reported their PSS experience level to be 'very experienced' and three candidates reported it to be 'experienced'. Three candidates reported their mouse experience level to be 'very experienced' and three candidates reported it to be 'experienced'.

It should be noted that the results of only six candidates were used for analysis. In one case the candidate only participated in the PSS experiment and not in the mouse experiment, causing the comparison between the two interfaces to become unbalanced. In the other case the candidate produced results that on inspection proved to be unrepresentative and that were therefore removed from the dataset.

An analysis of variance (ANOVA) was ran on the data. ANOVA is a statistical procedure, testing the null hypotheses that group means do not differ. The procedure returns a p-value, indicating the probability that the two sample means do not differ. The lower the p-value, the higher the probability that the two means differ. During analysis of the results, a p-value of 0.05 or less was chosen as statistically significant. This means that in that case there can be concluded with 95% confidence that the means do differ. ANOVA also returns an F value, indicating the ratio between the difference between groups and the difference within groups. A large F indicates that there is more difference between groups than there is within groups, which supports the hypothesis that the means do differ. So, the higher the F value, the lower the p value.

As quick overview, the results of the experiment are depicted in table 3.1 and table 3.2. Table 3.1 shows the average task completion time and average task accuracy for both 2D and 3D input. Table 3.2 shows for each input device the average time that was spent on translation, rotation ans scaling during the tasks. We discuss the results in detail in the following sections.

	μ	σ	M	F	p
2D input					
Task completion time (s)	89.5702	28.06	94.5359	5.63	<0.03
Translation accuracy (%)	99.0256	0.2502	99.0819	19.9	0.0002
Rotation offset (degrees)	4.7245	2.9358	4.1247	1.94	>0.17
Scaling offset x (%)	4.1719	3.6002	3.6373	0.11	>0.74
Scaling offset y (%)	7.6791	9.6548	2.977	0.32	>0.57
Scaling offset z (%)	8.9332	10.835	5.2042	0.82	>0.37
3D input					
Task completion time (s)	65.2648	21.7288	61.0613	5.63	<0.03
Translation accuracy (%)	99.44	0.2023	99.4637	19.9	0.0002
Rotation offset (degrees)	3.4395	1.2691	3.0885	1.94	>0.17
Scaling offset (x) (%)	3.6028	4.6512	2.9785	0.11	>0.74
Scaling offset (y) (%)	5.6235	8.025	1.697	0.32	>0.57
Scaling offset (z) (%)	5.4665	7.6817	3.1881	0.82	>0.37
μ = mean, σ = standard deviation, M = median					

Table 3.1: Average task completion time and average task accuracy

	μ	σ	M	F	p
2D input					
Translation time (s)	15.9521	3.1885	15.8216	6.32	<0.02
Rotation time (s)	38.3306	18.3437	35.0087	7.94	0.01
Scaling time (s)	10.8261	4.4168	11.8825	0.82	0.375
3D input					
Translation time (s)	20.0906	4.7281	20.1582	6.32	<0.02
Rotation time (s)	21.9634	8.2615	21.8083	7.94	0.01
Scaling time (s)	8.7362	6.6643	9.1223	0.82	0.375
μ = mean, σ = standard deviation, M = median					

Table 3.2: Average manipulation time for translation, rotation and scaling

3.6.1 Manipulation time

Figure 3.8 shows a box and whisker plot of the average docking duration. The box shows the middle 50% of the data, ranging from the 25th to the 75th percentile of the data set. The red bar indicates the median value of the data set. The horizontal bars at the end of the vertical lines indicate the extend of the data values. Possible outlyers are indicated with a cross. The notches in the box plot represent a robust estimate of the uncertainty about the medians for box-to-box comparison. Boxes whose notches do not overlap indicate that the medians of the two groups differ at the 5% significance level [14] [16].

The results of the ANOVA indicate that the docking task was performed significantly faster with the 3D input devices, $F_{1,22}$=5.63, p<0.03 (figure 3.8).

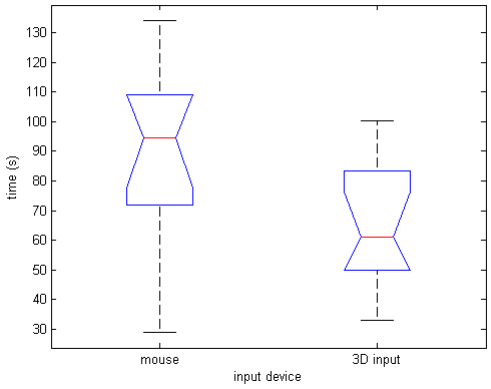

Figure 3.8: average docking duration

Figures 3.9-3.11 show the average time that subjects spent on positioning, rotation and scaling.

Translation time

An ANOVA for the average translation time reveals that the mouse is significantly faster, $F_{1,22}$=6.32, p<0.02 (figure 3.9).

Rotation time

An ANOVA for the average rotation time reveals that 3D input is significantly faster, $F_{1,22}$=7.94, p=0.01 (figure 3.10).

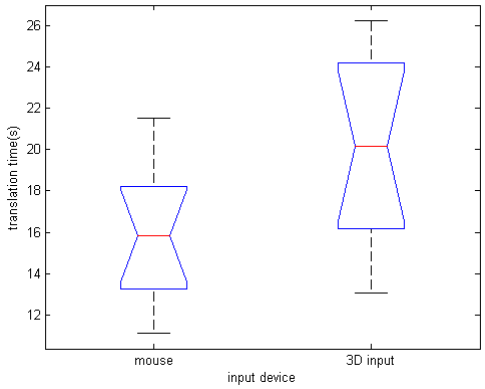

Figure 3.9: average translation time

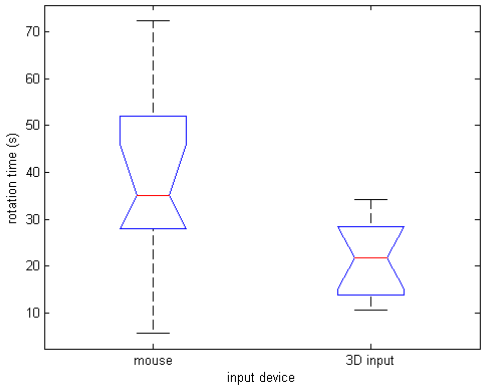

Figure 3.10: average rotation time

Scaling time

An ANOVA for the average scaling time reveals no significant difference between the input devices, $F_{1,22}$=0.82, p=0,375 (figure 3.11).

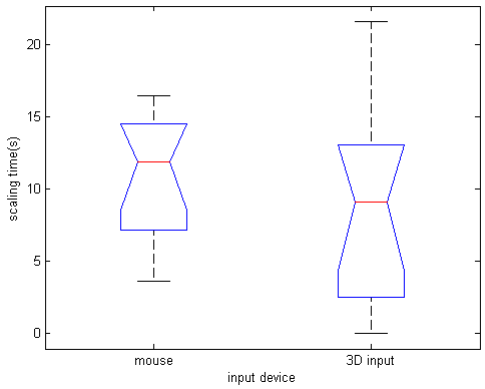

Figure 3.11: average scaling time

3.6.2 Accuracy

The accuracy of the docking task consists of positioning, orientation and scaling accuracy.

Positioning

Figure 3.12 shows a box and whisker plot of the positioning accuracy.
The results of the ANOVA indicate that the positioning accuracy of the docking task was significantly higher with the 3D input devices, $F_{1,22}$=19.9, p=0.0002

Orientation

Figure 3.13 shows a box and whisker plot of the orientation error.
The results of the ANOVA indicate that there was no significant effect for the number of degrees of freedom on orientation accuracy, $F_{1,22}$=1.94, p>0.17.

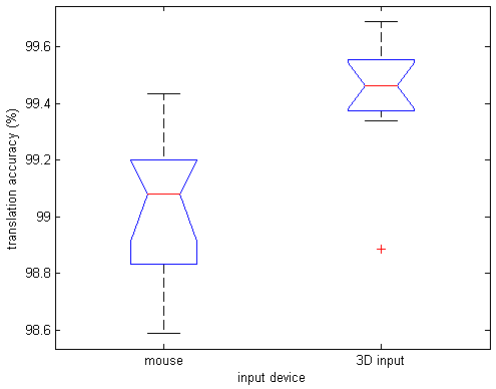

Figure 3.12: average positioning accuracy

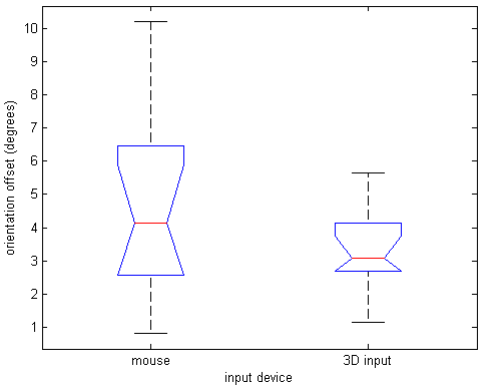

Figure 3.13: average orientation error

Scaling

Figures 3.14 - 3.16 show box and whisker plots for the scaling error in respectively the x, y and z direction. The figures suggest that there is no significant effect for the number of degrees of freedom on scaling accuracy. For the x direction, performing an ANOVA resulted in p>0.74 and $F_{1,22}$=0.11. For the y direction, performing an ANOVA resulted in p>0.57= and $F_{1,22}$=0.32, and finally performing an ANOVA for the z direction resulted in p>0.37 and $F_{1,22}$=0.82.

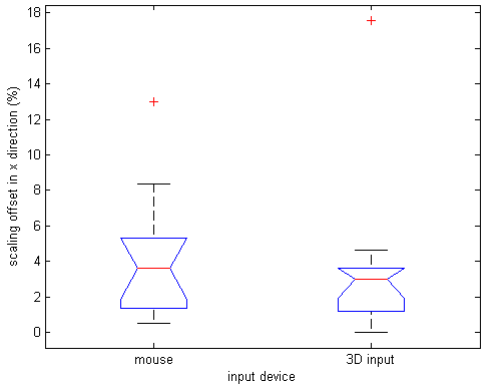

Figure 3.14: average scaling offset in x direction

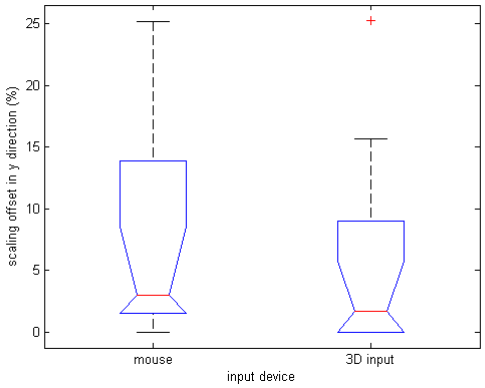

Figure 3.15: average scaling offset in y direction

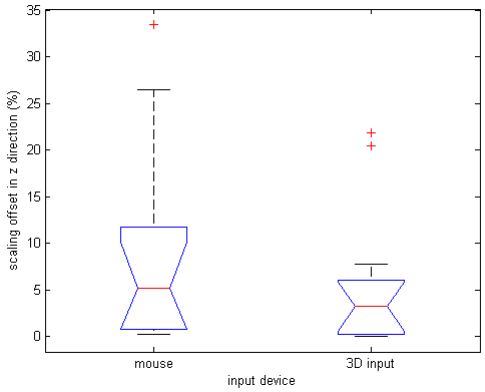

Figure 3.16: average scaling offset in z direction

3.6.3 Subjective Ratings

Figure 3.17 shows the ratings subjects gave on a 1 to 5 scale to input device comfort, input device suitability for positioning, rotation and resizing of the widget and to overall input device suitability for performing the docking task. The worst possible score is represented by a rating of 1, the best possible score is represented by a rating of 5.

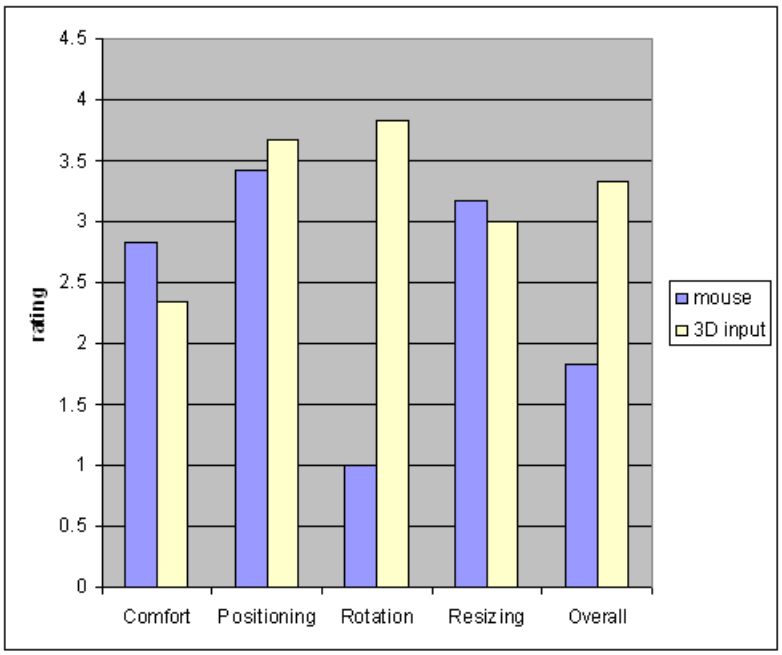

Figure 3.17: Input device ratings

Input device comfort

The figure shows that subjects found the mouse more comfortable during the experiment. Most subjects stated that using 3D input, their arms got fatigued after a while because they had to stretch too far and because their arm couldn't rest comfortably. This was not an issue with the mouse. Furthermore, some subjects indicated that they had to maintain a high level of concentration using the 3D input devices. When asking for clarification, subjects stated that sometimes they knew how they wanted to manipulate the widget, but didn't always know how to accomplish this using the 3D input devices. Some subjects indicated that they found it difficult to use four input devices simultaneously (i.e. two 3D input devices and a pedal for each input device). In general, using 3D input devices, subjects had to pay more attention to the input devices during the trails than was the case with the mouse.

Positioning

Figure 3.17 shows that the 3D input device scored slightly better than the mouse when it comes to widget positioning. Most subjects found the 3D input device reasonably suitable for positioning, though some subjects found it laborious that they had to position and rotate with separate devices. They'd rather seen a combination of positioning and rotation in the cube input device. Half of the subjects found the mouse suitable to very suitable for positioning. The other half found the mouse reasonably suitable to less suitable for positioning. When asking for clarification, these subjects indicated that they found it easy to position the widget in the (x,y)

plane, but harder when they also had to position the widget in the z direction. They found positioning in the z direction more intuitive using 3D input. One candidate indicated that the positioning functionality of the widget could not always be selected because it was occluded by an other part of the widget, this was no problem with 3D input.

Rotation

When it comes to rotation, figure 3.17 clearly indicates that 3D input is preferred to the mouse. Many subjects indicated that they found the way the widget rotates to be very unintuitive when the mouse is used as input device. They found it hard to tell how the widget was rotating and therefore how to accomplish the rotation they wanted to perform. As one subjects put it: "most of the time, I was just guessing". Subjects indicated that they missed the axis of rotation perpendicular to the screen. They found it confusing and very frustrating that they first had to rotate the widget by 90 degrees to be able to rotate about this axis. On the contrary, subjects were very positive when it comes to rotation with the 3D device. Most subjects found the rotation with the 3D device to be very intuitive, since the widget takes the same orientation as the 3D input device. However, some subjects indicated that they would prefer that the widget takes a relative orientation and not the absolute orientation of the input device. They indicated that they sometimes performed a rotation that was undone the next time they used the orientation device. This was because they hadn't paid any attention to the orientation of the device while using the selection device. Some users indicated they found it unintuitive that the widget first had to be selected by the 3D orientation device when performing a rotation. They would prefer to see a coupling between the selection device and the orientation device to be able to use the selection device to indicate the widget that has to change its orientation.

Resizing

Figure 3.17 suggests that the mouse performed slightly better for resizing. In general, subjects found the mouse reasonably suitable for resizing the widget. Subjects found 3D input also reasonably suitable for resizing. However, some subjects indicated that when using 3D input, they sometimes found it hard to locate the handles of the widget. They did not have this problem when they used the mouse to manipulate the handles.

Overall

The overall rating of the 3D input devices is considerably higher than the overall mouse rating. The intuitive and easy way in which the widget could be rotated was one of the most appreciated properties of 3D input. Some subjects indicated that it was harder to manipulate the handles of the widget when using 3D input. When using a mouse, subjects only needed to position the mouse over the handle to select it. When using 3D input, the z coordinate of the handle also had to be taken into account. Furthermore, subjects disliked the fact that more (physical) actions were required for simple tasks using 3D input. Some subjects indicated that they felt less accurate using 3D input. They attribute this to the shaking of their hands influencing the positioning or rotation.

Most subjects indicated that the unintuitive way the widget rotated played a big role in their mouse rating. As discussed earlier, they found rotation to be very unintuitive and frustrating. Some subjects noted that they got fatigued during the mouse trials because they had to concentrate on rotation. One subject felt that accuracy was feasible using the mouse, but very time consuming. However, most subjects did find the mouse suitable for resizing and to some extend, for positioning. Overall, most subjects preferred 3D input for the docking task.

3.7 Discussion

3.7.1 Task completion time

Figure 3.8 shows that the average duration of the docking task is shorter when performed with 3D input, which suggests that 3D input devices are preferable to 2D input when task completion time is an issue. When looking for the source of difference in task completion time, we initially only saw a significant difference in favor of the mouse when it came to average translation time. Average rotation and scaling time showed no significant difference between 2D and 3D input. This was very peculiar, since average task completion time was in favor of 3D input. Inspection of the log files revealed however, that in the case of 3D input, some of the subjects pressed the rotation pedal when they were not actually rotating, causing very long rotation times. This may be explained by the fact that the widget took the absolute orientation of the input device. Several subject stated that they found this very annoying, since the prior widget orientation was lost when performing a new rotation and no attention was given to the device orientation between two successive rotations. Therefore, some subjects kept the pedal of the orientation device pressed for the duration of the trial in order to prevent this from happening. Since these rotation times caused a distorted view of the average rotation time, they were considered as outliers and therefore removed from the dataset. The average rotation times for the adjusted dataset are depicted in figure 3.10. The figure clearly shows that the average rotation time is shorter for 3D input, which conforms to the observations made during inspection of the questionnaires. This suggests that for rotation, 3D input devices are preferable to 2D input when task completion time is an issue.

Figure 3.9 shows a significant difference in average translation time in favor of 2D input, suggesting that 2D input is preferable for translation, when task completion time is an issue.

Figure 3.11 shows no significant difference between 2D and 3D input for average scaling time. This corresponds reasonably well with the results of the subjects' preferred input device for scaling, which showed only a marginal difference in favor of 2D input.

These differences is average manipulation times for positioning, rotation and scaling can be better explained in light of the respective accuracies that were achieved, which will be the subject of the next section.

3.7.2 Task accuracy

Positioning

Figure 3.12 shows that there is a significant difference between 2D input and 3D input regarding positioning accuracy. This suggests that 3D input devices are preferable when positioning accuracy is an issue. Figure 3.9 reveals that subjects spent more time on translation with 3D input than they did with 2D input. Subjects indicated that they found positioning easier with 3D input. A possible explanation could therefore be that they tried harder to position the object with 3D input, which resulted in a longer positioning time and in a higher accuracy.

Rotation

Figure 3.13 shows that there is no significant difference regarding accuracy, when rotation is performed with a 2D input device or 3D input device. This suggests that both devices will perform equally well when accuracy is an issue. However, figure 3.10 shows that the average time spent on rotation was shorter in favor of 3D input. This suggests that for rotation, 3D input is as precise as 2D input, but requires less manipulation time.

Scaling

Figures 3.14-3.16 show that there is no significant difference regarding accuracy, when scaling is performed with a 2D input device or 3D input device. This suggests that both devices will perform equally well when accuracy is an issue. Figure 3.11 shows that the average time spent on the actual scaling does not differ between the two devices. However, it should be noted that subjects indicated that the handles of the widget were easier to locate using 2D input.

3.7.3 Observations

Results of the experiment indicate that there is room for improvement regarding the way the widget is manipulated. This holds for the mouse interface, as well as for the 6DOF interface.

Is is obvious that the way the widget rotates can be improved when manipulated by the mouse. Using the mouse, subjects spent a lot of time rotating the widget during the trials. A more intuitive rotation may provide better results for the mouse interface, regarding the rotation accuracy, as well as the mean trial time.

Regarding 3D input, interaction may be improved by decreasing the number of actions required to perform a simple handling. Some subjects indicated that they found it confusing to have two devices in their hand and two pedals at their feet. Removal of one of the pedals or different device function bindings may reduce interaction complexity.

Furthermore, overall docking performance may be improved by letting the software auto-dock the object when the source object is within a certain range of the target object. This holds for both 2D input as well as for 3D input.

3.7.4 Relation to previous work

Comparing to the work of Cline [2], our results show less improvement in average task completion time when using 3D input. Since our experimental setup allowed for bimanual 3D input, we use the bimanual results of Cline as basis for the comparison. The complex docking task of Cline, requiring translation and rotation of a cube, showed an improvement of approximately 41% in favor of 3D bimanual input. Compared to 2D input, our results show an improvement of approximately 27% when performing the docking task with 3D input. Comparison of both average manipulation time and accuracy for translation and rotation is not possible, since Cline reported these results to be erratic and inconclusive.

Comparing to the work of Hinckley et al [6], our results show more improvement in average rotation time when 3D input is used for rotation. Hinckley et al reported that 3D devices were up to 36% faster compared to 2D input. Our results show an even greater improvement of 45% in average rotation time in favor of 3D input. Furthermore, as was reported by Hinckley et al, there was no detectable loss in accuracy when using 3D input for rotation.

Chapter 4

Conclusion and future work

We studied the influence of the number of degrees of freedom of interaction devices on task completion time and task accuracy when manipulating 3D VTK widgets. We performed a user experiment where subjects had to perform a docking task, requiring translation, rotation and scaling of a source object by manipulation of a 3D widget, in order to match the position, orientation and size of a similarly shaped target object. Subjects performed the experiment with both 2DOF input and 6DOF input.

We hypothesized that the overall task completion time and the overall task accuracy would be improved when an input device with more degrees of freedom is used for widget manipulation. Furthermore, we hypothesized that for widget rotation and translation, 6DOF input is more accurate and faster than 2DOF input. For widget scaling, we hypothesized that 2DOF input is more accurate and faster than 6DOF input.

The results show that the overall task completion time and the overall task accuracy are improved when 3D input devices are used to manipulate the widget. This rejects the null hypothesis and supports our hypothesis H1.

For our hypothesis H2 we can not provide a conclusive answer. For rotation, 3D input was significantly faster compared to 2D input, however there was no significant difference regarding rotation accuracy between 2D and 3D input. For translation, 3D input was significantly more accurate, while 2D input was significantly faster. Finally, for scaling the results show no significant difference regarding manipulation time and accuracy between 2D and 3D input.

Based on the current results, we propose a combination of 2DOF input and 6DOF input for VTK 3D widget manipulation. We believe that widget translation and rotation should be combined into one 6DOF input device, since the results clearly indicate that 3D input is superior to 2D input regarding widget rotation time and translation accuracy. For widget scaling, we recommend 2D input, since subjects indicated that it is easier to locate the widget handles, has a higher comfort level and is less tiring.

This work presents findings of a first exploratory study. Since the software developed during our project is reusable, additional studies can be performed relatively quickly and may therefore be considered to give more conclusive results. For future studies, we suggest improving the mouse interface, since rotation was clearly very cumbersome. Rotation could be made more intuitive by rotating about two fixed axes (x,y) and by providing a modifier key on the keyboard for rotation about a fixed z axis. Furthermore, it may be interesting to consider a trackball as alternative to the three button mouse, where the ball may be used for rotation. Finally, for the 6DOF interface it may be considered to have the widget follow the relative orientation of the input device, instead of the absolute orientation in the current situation, since this proved to be very confusing.

Bibliography

[1] Boritz, J., and Booth, K. S. A study of interactive 3D point location in a computer simulated virtual environment. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology* (New York, NY, USA, 1997), ACM Press, pp. 181–187.

[2] Cline, M. Higher degree-of-freedom bimanual user interfaces for 3-D computer graphics. In *Proceedings of Human Interface Technologies* (2000), pp. 41–46.

[3] Conner, B. D., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C., and van Dam, A. Three-dimensional widgets. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics* (New York, NY, USA, 1992), ACM Press, pp. 183–188.

[4] Froehlich, B., Hochstrate, J., Skuk, V., and Huckauf, A. The globefish and the globemouse: two new six degree of freedom input devices for graphics applications. In *CHI '06: Proceedings of the SIGCHI conference on Human Factors in computing systems* (New York, NY, USA, 2006), ACM Press, pp. 191–199.

[5] Herndon, K. P., and Meyer, T. 3D widgets for exploratory scientific visualization. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1994), ACM Press, pp. 69–70.

[6] Hinckley, K., Tullio, J., Pausch, R., Proffitt, D., and Kassell, N. Usability analysis of 3D rotation techniques. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology* (New York, NY, USA, 1997), ACM Press, pp. 1–10.

[7] Jacob, R. J. K., Sibert, L. E., McFarlane, D. C., and M. Preston Mullen, J. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact. 1*, 1 (1994), 3–26.

[8] Kosara, R., Healey, C. G., Interrante, V., Laidlaw, D. H., and Ware, C. User studies: Why, how, and when? *IEEE Comput. Graph. Appl. 23*, 4 (2003), 20–25.

[9] Martens, J.-B., Qi, W., Aliakseyeu, D., Kok, A. J. F., and van Liere, R. Experiencing 3D interactions in Virtual Reality and Augmented Reality. In *EUSAI '04: Proceedings of the 2nd European Union symposium on Ambient intelligence* (New York, NY, USA, 2004), ACM Press, pp. 25–28.

[10] Mulder, J., and van Liere, R. The Personal Space Station: Bringing interaction within reach. In *Proceedings of VRIC* (2002), pp. 73–81.

[11] Personal Space Technologies.
http://www.personalspacetechnologies.com.

[12] Schroeder, W., Martin, K., and Lorensen, B. *The Visualisation ToolKit.* Kitware, 2002. ISBN 1-930934-07-6.

[13] Schroeder, W. J., Martin, K. M., and Lorensen, W. E. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *VIS '96: Proceedings of the 7th conference on Visualization '96* (Los Alamitos, CA, USA, 1996), IEEE Computer Society Press, pp. 93–100.

[14] The Mathworks.
http://www.mathworks.com/access/helpdesk/help/toolbox/stats/boxplot.html.

[15] van Liere, R., and Kok, A. J. A multimodal Virtual Reality interface for VTK. *ICMI'05 Workshop on Multimodal Interaction for the Visualization and Exploration of Scientific Data* (October 2005).

[16] van Rhijn, A. Spatial input device structure and bimanual object manipulation in virtual environments. *Submitted for VRST 2006* (May 2006).

[17] Wang, Y., MacKenzie, C. L., Summers, V. A., and Booth, K. S. The structure of object transportation and orientation in human-computer interaction. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1998), ACM Press/Addison-Wesley Publishing Co., pp. 312–319.

[18] Ware, C., and Jessome, D. R. Using the bat: a six dimensional mouse for object placement. In *Proceedings on Graphics interface '88* (Toronto, Ont., Canada, Canada, 1988), Canadian Information Processing Society, pp. 119–124.