2003

Direct 3d interaction using a 2d locator device

Anees Ansari

University of South Florida

Follow this and additional works at: http://scholarcommons.usf.edu/etd

Part of the American Studies Commons

Scholar Commons Citation

Ansari, Anees, "Direct 3d interaction using a 2d locator device" (2003). Graduate Theses and Dissertations.
http://scholarcommons.usf.edu/etd/1325

This Thesis is brought to you for free and open access by the Graduate School at Scholar Commons. It has been accepted for inclusion in Graduate Theses and Dissertations by an authorized administrator of Scholar Commons. For more information, please contact scholarcommons@usf.edu.
Direct 3D Interaction Using A 2D Locator Device

by

Anees Ansari

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Science
Department of Computer Science And Engineering
College of Engineering
University of South Florida

Major Professor: Les Piegl, Ph.D.
Murali Varanasi, Ph.D.
Nagarajan Ranganathan, Ph.D.

Date of Approval:
July 1, 2003

Keywords: computer, graphics, input, mouse, direct3d

© Copyright 2003, Anees Ansari
DEDICATION

To Mom, Dad, Trishna, Nafees, Nani and Nana.
ACKNOWLEDGEMENTS

I would like to thank Dr. Piegl my Major Professor for his invaluable help and support during these two years at USF. He has been a big source of encouragement and has always unfailingly managed to inspire me to accomplish even the most difficult of tasks. Without his guidance and his consistent support this work would not have been possible. I want to thank Dr. Varanasi and Dr. Ranga for all the support they have extended to me both, as committee members and as faculty members at USF. They are two of the most wonderful people I have ever come across. I want to thank Dr. Perez for being an excellent graduate program director and for helping me immensely in every way he could. I wish to thank Daniel Prieto for helping me all along these years, as my boss and as a friend. I owe him a lot for giving me an opportunity and helping me learn. And finally I wish to thank my parents, my relatives and friends without whose love, prayers and support I would not be what I am today.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF FIGURES</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>viii</td>
</tr>
<tr>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Detailed problem statement</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Contribution of thesis</td>
<td>3</td>
</tr>
<tr>
<td>1.4 Outline of thesis</td>
<td>3</td>
</tr>
<tr>
<td>CHAPTER 2 PRIOR WORK</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Past research in 3D interaction using 6DOF input devices</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Past research in 3D interaction using 4DOF input devices</td>
<td>9</td>
</tr>
<tr>
<td>CHAPTER 3 INTERACTION DEVICES</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Characteristics</td>
<td>14</td>
</tr>
<tr>
<td>3.2 Devices</td>
<td>17</td>
</tr>
<tr>
<td>CHAPTER 4 3D INTERACTION</td>
<td>23</td>
</tr>
<tr>
<td>4.1 Direct and indirect 3D interaction</td>
<td>23</td>
</tr>
<tr>
<td>4.2 Disadvantages of indirect interaction</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Advantages of direct interaction</td>
<td>24</td>
</tr>
<tr>
<td>4.4 Disadvantages of true 3D devices</td>
<td>25</td>
</tr>
<tr>
<td>4.5 Advantages of the mouse and extending its use to 3D interaction</td>
<td>26</td>
</tr>
<tr>
<td>CHAPTER 5 CHALLENGES OF 3D INPUT USING THE MOUSE</td>
<td>29</td>
</tr>
<tr>
<td>CHAPTER 6 DIRECT 3D INTERACTION WITH A 2D LOCATOR DEVICE</td>
<td>33</td>
</tr>
<tr>
<td>6.1 Workspace and viewpoint</td>
<td>33</td>
</tr>
<tr>
<td>6.2 Input mapping</td>
<td>36</td>
</tr>
<tr>
<td>6.3 Transformations and operations</td>
<td>38</td>
</tr>
<tr>
<td>6.4 Feedback</td>
<td>46</td>
</tr>
<tr>
<td>6.5 Development system specifications</td>
<td>56</td>
</tr>
<tr>
<td>CHAPTER 7 APPLICATIONS</td>
<td>57</td>
</tr>
</tbody>
</table>
CHAPTER 8  EXAMPLE SESSION
  8.1 Modeling the nose
  8.2 Modeling the front fuselage
  8.3 Modeling the middle fuselage
  8.4 Modeling the back fuselage
  8.5 Modeling the left wing
  8.6 Modeling the right wing
  8.7 Modeling the tail
  8.8 Modeling the left horizontal stabilizer
  8.9 Modeling the right horizontal stabilizer

CHAPTER 9  CONCLUSIONS AND FUTURE RESEARCH

REFERENCES

APPENDICES
  APPENDIX A  MAPPING ALGORITHM
LIST OF FIGURES

Figure 1. A 3D ball device. 17
Figure 2. A tracker device. 17
Figure 3. A 3D mouse device. 18
Figure 4. A Rockin mouse. 18
Figure 5. A head mounted display. 19
Figure 6. A 3D glove device. 19
Figure 7. A bat device. 20
Figure 8. An elastic device. 21
Figure 9. A Mulit-DOF armature device. 21
Figure 10. A screenshot of Direct3D when it has just been started showing (a) The Tab view. (b) The Direct3D view. 35
Figure 11. A screenshot of Direct3D showing (a) The original line plotted. (b) The original line translated -10 units along the X axis. (c) The original line translated +20 units along the X axis and then rotated about the X axis. (d) A line-strip plotted. 40
Figure 12. A screenshot of Direct3D showing (a) A pentagon plotted. (b) The original pentagon translated along the X axis and then scaled along the Y axis. (c) A hexagon plotted. (d) The original hexagon translated along the X and Y axes and then rotated about the Z axis. 42
Figure 13. A screenshot of Direct3D showing (a) An edge-lift polyhedron i.e. a parallelepiped plotted. (b) A center-lift polyhedron i.e. a pyramid plotted and then rotated about the Y axis. 45
Figure 14. The button images used in Direct3D. (a) The button images displayed when the Point tab is selected. (b) The button images displayed when the Line tab is selected. (c) The button images displayed when the Polygon tab is selected. (d) The button images displayed when the Polyhedron tab is selected. 46
Figure 15. The tab images used in Direct3D.

Figure 16. The cursors used in Direct3D. (a) The cursor displayed when the movement is along the X axis. (b) The cursor displayed when the movement is along the Y axis. (c) The cursor displayed when the movement is along the Z axis. (d) The cursor displayed in the Point deletion mode.

Figure 17. A screenshot of Direct3D to illustrate various feedback techniques. (a) The selected button. (b) The tabs. (c) The radio buttons. (d) The status bar displaying the current state of the system and the help message corresponding to the current state of the system. (e) A part of the 3D environment showing a selected polyhedron being translated along the Y axis.

Figure 18. Magnified illustration of part (a) of Figure 17 showing the button-pushed effect.

Figure 19. Magnified illustration of part (b) of Figure 17 showing the tab-raised effect for the Polyhedron tab.

Figure 20. Magnified illustration of a part (c) of Figure 17 showing the radio buttons.

Figure 21. Magnified illustration of a part (d) of Figure 17 showing the status bar displaying the current state of the system and a help message corresponding to the current state of the system.

Figure 22. Magnified illustration of a part (e) of Figure 17 showing (a) The legs (three lines, white in color, converging at the orange point). (b) The projections (six lines, black in color, drawn on the XY, YZ and ZX planes). (c) The helper line for movement along the Y axis (the vertical black colored line) (d) Selection highlighting i.e. the selected polyhedron (magenta colored vertices). (e) The dynamic coordinate display.

Figure 23. A screenshot of Direct3D showing a pop-up message displayed when the user has just deleted the last point in the workspace.

Figure 24. A screenshot of 3D Home Architect.

Figure 25. A screenshot of mechanical CAD software.

Figure 26. An image of a person using simulation training software.

Figure 27. A screenshot of medical visualization software.

Figure 28. A screenshot of animation software.
Figure 29. A screenshot of geometric modeling software.

Figure 30. A 3D sketch.

Figure 31. A screenshot of molecular modeling software.

Figure 32. A screenshot of Direct3D illustrating the scene after the nose of the airplane has been added.

Figure 33. A screenshot of Direct3D illustrating the scene after the front fuselage of the airplane has been added.

Figure 34. A screenshot of Direct3D illustrating the scene after the middle fuselage of the airplane has been added.

Figure 35. A screenshot of Direct3D illustrating the scene after the back fuselage of the airplane has been added.

Figure 36. A screenshot of Direct3D illustrating the scene after the left wing of the airplane has been added.

Figure 37. A screenshot of Direct3D illustrating the scene after the right wing of the airplane has been added.

Figure 38. A screenshot of Direct3D illustrating the scene after the tail of the airplane has been added.

Figure 39. A screenshot of Direct3D illustrating the scene after the left stabilizer of the airplane has been added.

Figure 40. A screenshot of Direct3D illustrating the scene after the right stabilizer has been added and the finished model of the airplane.
Direct 3D Interaction Using A 2D Locator Device

Anees Ansari

ABSTRACT

Traditionally direct 3D interaction has always been limited to true 3D devices whereas 2D devices have always been used to achieve indirect 3D interaction. Till date no proper research has been done to try and extend the use of mouse to direct 3D interaction. In this research we explore the issues involved with using the mouse to accommodate the additional degrees of freedom required for 3D interaction. We put forth a unique and innovative design to achieve this objective and show that even a device as simple as the mouse can be highly effective for 3D interaction when supported by an appropriate underlying design. We also discuss in detail a software prototype “Direct3D” that we have developed based on our design and hope to take a step towards making direct 3D interaction easy, inexpensive and available to all computer users.
CHAPTER 1
INTRODUCTION

With the growing speed and dropping prices of computers, 3D graphics are becoming more and more prevalent in various areas. Research and educational tools and software, computer aided design, games, movies, animations, etc. all use 3D graphics. With the gaining popularity of 3D graphics there is a strong and noticeable need for 3D interaction.

1.1 Motivation

Interaction with computers has evolved immensely over the years. Currently 2D/2.5D graphics such as buttons, menus, dialog boxes, etc. combined with 2D interaction techniques, dominate the market. The most common and popular devices being used for 2D/2.5D interaction are the mouse and the keyboard.

3D graphics portrayed using 2D displays has also gained immense popularity from entertainment, commercial and research viewpoints and shows a lot of potential to be a successor of 2D/2.5D graphics. Unfortunately 3D interaction using 2D displays has not grown as fast or as better and till date involves many difficulties overshadowing the numerous advantages it has to offer.

Most of the research done till date in the area of 3D interaction focuses either on achieving direct 3D interaction using specialized true 3D devices or on attaining an indirect 3D interaction using 2D devices. With this research we want to draw attention to an area that has been widely neglected in the past i.e. achieving direct 3D interaction using 2D locator devices.
We attempt to arouse interest towards this new promising approach to 3D interaction by putting forth a novel design to demonstrate the practicality and the advantages of the approach. Our design will also inherently promote and facilitate the use of 2D locator devices for direct 3D interaction. These devices are inexpensive, widely available and popular and will thus avoid the need to buy and use expensive 3D devices.

1.2 Detailed problem statement

Many tools such as the ones used for CAD (computer aided design) and CAM (computer aided manufacturing) as well as the ones used in numerous research areas such as geometric modeling rely heavily on 3D interaction. However true 3D displays cannot yet compete with the high quality and low cost combination offered by 2D displays. Hence 3D graphics and 3D interaction still depend on 2D displays.

2D devices such as the mouse and keyboard still remain the most common and preferred devices yet 3D interaction achieved using these have always been with a view to achieve precision by limiting the user to indirect interaction. On the other hand 3D interaction achieved using true 3D devices such as gloves and trackers have always been focused on achieving direct 3D interaction at the cost of precision.

In this research we want to achieve a win-win situation between these two opposite ends. We want to achieve “Direct 3D interaction using a 2D locator device”. We want to allow users to directly interact with the 3D environment without having to compromise on precision.

One additional issue of concern is that all the direct 3D interaction today relies heavily on specialized 3D devices. Not only are these devices expensive and difficult to obtain, but also difficult to use and learn. The high cost of these devices limits their usage to well established research and commercial organizations.

With the help of this research we aim to get rid of the cost factor involved in experiencing and/or using direct 3D interaction and want to make it available to everyone at no extra cost and without having to buy any specialized devices.
1.3 Contribution of thesis

We propose a novel approach to achieve direct 3D interaction using a commonly available 2D locator device, namely the mouse. We can safely assume that everyone who owns a computer has a mouse, and is familiar and confident using one. Thus using our approach neither will the user have to incur additional costs to buy a special device nor will he/she have to spend time familiarizing with the device.

We have formulated a design that shifts the burden from the user to the software. The only area where the user will have to spend a little amount of time will be familiarizing with the mappings and the corresponding mouse movements.

Finally we have actually implemented prototype generic software using which the users can actually get a feel of what our design aims to achieve. The software is also aimed at building a strong base, highlighting the practicality of the design and promoting future research on achieving direct 3D interaction using 2D locator devices.

1.4 Outline of thesis

The initial part of our research gives a brief overview of all the work done in the field of 3D interaction. Here we discuss the various approaches that have been employed in the past to achieve both direct and indirect 3D interaction. Also we review the various techniques that have been employed to make 3D interaction easier for the end-user.

We then shift our focus on the various interaction devices, their characteristics, advantages and disadvantages. This gives us an insight on how the different characteristics of the devices gain and lose importance depending on the application they are being used for and the environment they are being used in.

This is followed by a detailed overview on 3D interaction. Here we talk about direct and indirect 3D interaction and the advantages and disadvantages of each. We then justify our intent to combine both approaches to get a unique style to achieve 3D interaction using 2D locator devices and the various advantages it will have to offer.
Before formulating a methodology to achieve our desired goal of achieving 3D interaction using the mouse, we enumerate the various challenges such as mapping of the device, visualization of workspace, etc. that we had to overcome.

Next we explain the design we have come up with to achieve direct 3D interaction using the mouse and briefly discusses the software implementation of the same. Towards the end we mention some of the applications where such a design might prove useful followed by some possible example sessions.

Finally we draw inferences from our research work and suggest possible areas and enhancements for future research work on the topic.
CHAPTER 2
PRIOR WORK

3D interaction has always been a challenging topic for computer graphics. However due to its immense importance in areas such as CAD, 3D geometric modeling, etc. it remains one of the most popular areas for research.

Current advancements in technology have enabled graphic output far better than could be imagined ten years ago. Graphic input however, which is of equal importance, has lagged far behind.

To improve the quality of input in graphic interfaces Buxton [1] suggests that we should not look at the input device as an independent entity. The device should be examined in a more global and holistic view as a part of the system. He also suggests that we should examine input devices as closely as possible and at various levels of detail to uncover all its characteristics and then use the ones which will be of advantage to us. An additional advantage of such analysis is that most of the times problems which arise at a particular level of detail are solved or eliminated by making minor adjustments or changes at a different level.

A lot of research has been done in the past on ways to achieve 3D interaction. Some use true 3D devices, which provide six ‘Degrees of Freedom’ (DOF) to achieve direct interaction and some use 2D devices, which provide four DOF, to achieve an indirect interaction. Hand [2] in his paper has surveyed the techniques used to perform 3D object manipulation and navigation. According to him most of the 3D interaction application programs have three common domains:
1. Object manipulation.
2. Viewpoint manipulation.
3. Application control.

He also states that the research in this area can be divided into 2 generic phases:
1. Evolution of techniques based on the use of the 2D devices such as the mouse.
2. New ideas generated when true 3D input devices came into picture.

Nielsen and Olsen [3] have used a technique called ‘triad mouse’. They have devised a technique to directly manipulate 3D objects using 2D locator devices. They use a three pronged cursor which they call ‘triad’. Using a 2D device the user can manipulate the triad making the system function like a 3D mouse. They use a framing cube to serve as a frame of reference to help the user estimate the actual cursor position.

Hinckley, Tullio, Pausch et. al. in their paper [4] advocate, that we should focus on finding novel ways to achieve 3D interaction and not argue on the advantages and disadvantages of each approach since each approach will excel for some applications and prove futile for some others.

A few important basic factors that are helpful when designing any interface to achieve 3D interaction are mentioned below. [2][5]
1. The environment should be as transparent as possible. The user should feel as though he is interacting with real world objects and not with an interface.
2. The priority of the interface should be to allow the user to work using tasks which seem appropriate and natural and not the ones which will simplify things for the computer.
3. The interface should be easy to learn and to use. The gap between ‘Gulf of Execution’ (knowing what to do) and ‘Gulf of Evaluation’ (knowing how to do it) should be bridged.
4. The interface should promote speed and accuracy.
5. Tactile and/or force and/or kinesthetic feedback should be used if possible. Kinesthetic feedback enables the user to know the position of his/her limbs relative to the rest of the body i.e. spatial awareness.

6. The user should be able to feel his/her presence in the virtual environment and should be able to gather information from the surroundings.

7. Viewpoint manipulation design should be given a lot of attention.

8. If possible viewpoint manipulation should also provide feedback.

9. The user may be allowed to choose between two views.
   a. Egocentric: Where the user is at the center of the space.
   b. Exocentric: Where the user gets a feel of looking at space from outside.

10. A virtual representation of the actual input device being used can be employed.

   It is not necessary to incorporate all of the above factors in order to design a good interface. The designers, depending on their applications, can assign appropriate weights to each of the factors.

2.1 Past research in 3D interaction using 6DOF input devices

6DOF devices help make the interaction more natural and intuitive for the user [6]. Some of the significant researches done in this area will be discussed in this section.

Boritz and Booth [7] have studied the user’s ability to locate a 3D point, using a 6 DOF input device, in a computer simulated virtual environment. In their research users had to perform two tasks, namely:

1. Point location: Moving a 3D pointer to a specific fixed point in the virtual 3D environment.

2. Interactive path tracing: Following a path in the virtual 3D environment.

Using four visual feedback modes:

1. Fixed viewpoint monoscopic perspective.
2. Fixed viewpoint stereoscopic perspective.
3. Head-tracked monoscopic perspective.
4. Head-tracked stereoscopic perspective
Liang and Green [6] have researched to achieve geometric modeling using a 6 DOF input device. Most of their experiments have been performed using a hand held Isotрак sensor, namely the ‘Bat’ [8]. The position and orientation of the bat is monitored in real time and then displayed on the screen in the form of a ‘Jack’ [9], a 3D cursor.

Kaufman, Yagel and Bakalash [10] have implemented an interface for direct 3D interaction with objects and their visceral exploration. Their interface uses a 3SPACE Polhemus Isotрак and VPL DataGlove and a corresponding 6D cursor, namely the ‘Jack’. The workspace view is a 3D rectilinear space in perspective view and the user movements are restricted to this frame. To help in generating, manipulating and viewing sampled and/or synthetic volumetric objects a volume editor ‘edvol’ is provided as an integral part of the environment.

A three dimensional surface modeling program ‘3DM’ that uses a head-mounted display to simplify 3D manipulation and understanding has been developed by Butterworth [11]. The program is based on techniques used in CAD and drawing programs and applies those techniques to modeling in a true 3D environment in an intuitive way.

Some researchers have even proposed using both hands in combination with 6DOF devices to enhance productivity and naturalness. Cutler, Frohlich and Hanrahan [12] use a tabletop virtual reality device called ‘Responsive Workbench’ in combination with a system that allows users to manipulate virtual 3D model with both hands. They found the coordinated and asymmetric two-handed interactions interesting and have concluded that in such a system both the hands perform distinct small subtasks in a synergistic way to accomplish a bigger complex task.

In a similar research Sachs, Roberts and Stoops [13] have studied direct 3D interaction by using a pair of hand held 6DOF devices. They deduce that the simultaneous use of two hands has a sort of an inbuilt S-R feedback, since the users know
the relative position of their hands. Also such a design increases the speed and quality of
the design.

2.2 Past research in 3D interaction using 4DOF input devices

One of the best researches in this area has been done by Branco, Costa and Ferreira [14]. In their paper they state that sketching in the conceptual phase allows one to explore high-level design decisions at low cost. However the creation of 3D shapes using CAD tools is difficult and time consuming and hence these tools are kept away from the conceptual phases of design. They want to propose a solution for this issue by providing to the designers a tool which will be simpler and faster to use but will be as powerful as a CAD system.

The authors mention that most of the industrial products used originated from pencil and paper that are nothing but 2D input devices. Therefore the system they have designed is intended to work with 2D devices and aims to combine simplicity and intuition with the useful features of the modelers. They call it ‘IDeS: Intuitive Design System’.

IDeS aims at providing an as simple an interface as possible. With IDeS the user performs three tasks:

1. Drawing: The user draws as he would in a conventional drawing package the only exception being that if a trivalent junction appears then the system employs perceptual analysis to store some information about it to use in 3D reconstruction.
2. Picking a modeling tool: The user picks the tool he wants to use, but the system decides if it has enough information to execute the command associated with the appropriate tool. If not the execution is postponed till the user draws the information that is missing.
3. Explaining: The user has to provide some information about the drawing to the system e.g. when the drawing is finished or to convert a free hand drawn line to a straight-line segment, etc.
When drawing in *IDeS* the objects must be drawn without hidden lines. When the drawing is finished the user must explicitly tell the system that it is a 3D representation. The system then calculates the fully visible, partially visible faces and hidden faces.

When editing in *IDeS* the system employs ‘Gluing’. Using gluing a straight-line becomes a poly-line after transformation and a closed poly-line becomes a polygonal mesh. Boolean operation can further be performed using the polygonal mesh so obtained. i.e. a union or difference operation can be done on the object and the polygonal mesh. To obtain complex objects.

The drawing engine employed by *IDeS* performs four main tasks:

1. **Drawing Graph Management:** This module manages the graph that describes at each moment the 2D drawing.
2. **Perceptual Analysis:** This is used to manage all the accesses to the junction dictionary and to classify junctions depending on the angles between the intersecting lines.
3. **3D Reconstruction:** This component warns the user if the drawing cannot be interpreted as a 3D model else it attempts to reconstruct the solid using an algorithm which has four basic steps:
   a. Virtual camera positioning.
   b. Gluing of the first junction.
   c. Visible part reconstruction.
   d. Hidden part reconstruction.
4. **Drawing Events Generation:** Module which accepts feedback from the above three modules and gives rise to drawing events depending on the feedback.

The authors conclude their paper by saying that though *IDeS* is still a prototype it has received considerable amount of praise by architects and designers who tested it.

Likewise ‘*SKETCH*’ the interface developed by Zeleznik, Herndon and Hughes [15] allows users to rapidly conceptualize and edit approximate 3D scenes. It uses a
simple non-photorealistic rendering mechanism and a purely gesture-based interface with pre-defined gestures that accepts simple line drawings as input. All the operations are performed in the 3D scene i.e. a single orthographic view with the help of a three-button mouse with occasional use of one modifier key on the keyboard.

The user has to simply sketch the salient features of any of a variety of 3D primitives. SKETCH then uses four simple placement rules and draws the corresponding 3D primitive in the 3D scene.

A few more important functionalities in SKETCH are mentioned below:

1. The camera can be manipulated using gestures.
2. Automatic grouping mechanism can be used to help apply aggregate transformations.
3. Since less semantic information is stored the user may be required to explicitly sketch constraints also.

The authors conclude by agreeing that SKETCH is just in its early stages and a lot more study needs to be done in order to make it better and to help increase the range of it’s applications without compromising on simplicity.

Shoemake [16][17] has devised an input technique called the ‘Arcball’ to adjust the spatial orientation of an object in a 3D environment. Arcball uses the mouse as the input device and achieves a kinesthetic agreement between the mouse movement and object rotation by constant interpretation of the mouse motion and association with the corresponding mapping. It blends human factors and mathematical fundamentals well and provides consistency and rich feedback. However the disadvantage is that Arcball cannot control translation and scaling as is usually required in any 3D interaction software. Theoretically with a single drag the user can rotate an object 360 degrees, around any axis, using the Arcball [4]. Practically however the users find it complex to achieve such a rotation and compose a 360 degree rotation of multiple small rotations.
Chen, Mountford and Sellen [18] describe a technique called ‘Virtual Sphere’ which is very similar to the Arcball. The Virtual Sphere is a mouse driven 2D interface which can be thought of as a virtual trackball. The user clicks and drags on the object shown on screen and the computer interprets these to rotate the object correspondingly. The third degree of freedom is provided by enclosing the object in a circle and detecting clicks and drags outside the circle. These clicks and drags cause rotation of the object about an axis perpendicular to the screen.

Hinckley, Tullio and Pausch [4] compare the Arcball and Virtual Sphere and conclude that the Arcball is more mathematically sound and avoids ‘hysteresis’ effect. Hysteresis is the effect of not producing closed loops of rotation by corresponding closed loops of mouse motion, in simple words it means that reversing the sequence of drags will not return the object to its original position.

Chen, Mountford and Sellen in their research [18] aim to achieve an optimum solution to direct manipulation and positioning of 3D objects in real time using 2D control devices. In their paper they say that various controllers are used to manipulate the 3D objects dynamically and go on to discuss four such controllers primarily used for rotation.

1. Sliders.
2. Overlapping Sliders.
3. Continuous XY+Z.
4. Virtual Sphere

All of these achieve to be as simple as possible both in appearance and in use, to enable the user to focus on the task rather than the interface.

They have even studied and compared the effectiveness of each of the controllers in diverse situations. Their studies indicate that simple, single axis rotations were performed faster by using the sliders (both conventional and overlapping) and complex tasks were performed faster using the XY+Z and Virtual Sphere interfaces. They
conclude based on the studies that the Virtual Sphere was clearly superior in terms of speed when complex rotations were needed and was also reported by the subjects as the interface providing the most natural feel.

Zeleznik, Herndon, Robbins, et al. [19] have implemented a toolkit to construct 3D widgets. This toolkit makes 3D object generation faster and easier for non-technical users. Also since construction of 3D widgets is inherently geometric, this toolkit imparts a natural feel to the construction of these widgets by employing direct manipulation of primitives to create the desired widget. It also provides the user with the power to link two or more primitives thus easing the construction of the more complex 3D widgets.

Researchers [20] have even tried the simultaneous use of two 6DOF devices in the two hands. They suggest that one-handed input is less natural and less efficient and two handed input has the potential for implementing interfaces that are more natural and simpler thus enhancing efficiency. Two handed input can split a compound task into two possibly parallel tasks controlled by both hands [21].

3D navigation is required in numerous interactive graphics and virtual reality applications and a lot of research focuses on the issue. One of the best papers is the by Hanson and Wernert [22] in which they discuss 3D navigation using a mouse.

Thus we can see that a lot of research has been done in the field of 3D interaction but hardly any focuses on making direct 3D interaction possible with 2D devices. This is the reason that makes our work fresh and innovative.
CHAPTER 3
INTERACTION DEVICES

Over the years the devices that we use to interact with computers have changed drastically [23]. Input devices set, constrain and bring out numerous actions and responses from the user. To be able to design good 3D interaction software we must be aware of the devices that may be used with the software, their characteristics, advantages and disadvantages. In this chapter we shall first take a look at the characteristics of the devices followed by a brief overview of few important, uncommon devices.

3.1 Characteristics

Characteristics are very important and influence the way the user will employ a device [4][23][24][25][26][27].

1. Affordance/Form Factor: The device should inherently suggest to the user how it is supposed to be used. Users have often reported diametrically opposite impressions of devices that differed only in their physical housing

2. Tactile cues: The device should have strong tactile cues which give the user the perception of the preferred way of holding it. In the absence of such cues the user may be unsure of the correct way of using the device

3. Grasp: There are two types of grasp. ‘Power grasp’ is when the device is held against the palm in a fixed orientation. The word ‘power’ is used because the posture emphasizes strength and security of grip. The second type of grasp known as ‘Precision grasp’ involves pads and tips of the fingers. The word ‘precision’ is used because it emphasizes dexterity and free tumbling of the device being used.

4. Device acquisition time / Time to grasp: This signifies the amount of time it takes to engage the device if the hands were currently being used for some other device or some other task.
5. **Clutching**: It is a property of relative positioning devices. It involves, disengaging, adjusting and re-engaging the device to extend its field of control.

6. **Resolution**: It is the smallest incremental change in the device position that can be perceived.

7. **Sampling rate**: It is the number of times per second the position of the device is recorded. This factor is very important for real-time 3D environments because the devices used in such a situation have to respond to natural and fast movements of humans.

8. **Lag**: Amount of time taken to update the display in response to pre-defined events. The most difficult part is detecting the source of the lag.

9. **Control-Display Gain**: This represents the ratio of the motion of the device (control) to the corresponding movement of the cursor on the display. It represents trade-off between rough positioning and fine positioning.

10. **Input-Output mappings**: This determines which movements of the device cause corresponding movements on-screen of the cursor.

11. **Gestures**: Gestures are the most common interaction paradigm. The best way to use gestures is to map them directly to user intention.

12. **Tactile and force feedback**: This implies the property of a device to provide a force in response to pre-defined movements/interactions.

13. **Multi-modal input**: This implies the property of a device to merge two or more modes of input such as speech, touch, etc.

14. **Pointing speed/ Bandwidth**: This property represents the speed of target selection using the device.

15. **Pointing precision**: This property represents the smallest target that can be easily selected using the device.

16. **Time to learn**: This represents the amount of time it takes to learn the device operation.

17. **Desk footprint**: This represents the amount of physical space the device takes on the desk.

18. **Cost**: The price of the device.

19. **Fatigue**: This corresponds to the tiring of the user when using the device.
20. **Sticky/Free**: Sticky devices are the ones which have some sort of a mechanism to prevent changes along other axes when one axis is used. Free devices on the other hand have no such mechanism.

21. **Orthogonal/Nested**: Orthogonal devices have fixed frames of reference, whereas nested devices do not.

22. **Rotation/Translation**: Rotation devices are the ones that operate by rotation, e.g. trackball and translation devices are the ones that operate by translation e.g. mouse.

23. **Unbounded/Bounded**: Devices that have no physical limits on the field of control are called unbounded devices, whereas the ones that do are called bounded devices.

24. **Homogenous/Distinguished position**: Homogenous devices cannot be set to a remembered physical position, whereas distinguished position devices can.

25. **Volatile/Non-volatile**: Volatile devices cannot retain their physical position when released, but non-volatile devices can.

26. **Inertial/Inertia-less**: Inertial devices keep moving for a short distance and time when released. Inertia-less devices do not exhibit this tendency.

27. **Held-up/Body-mounted**: Held-up devices need to be held using the hand and cannot stand on their own. Body-mounted devices do not need external support and their body helps them stand on their own.

28. **Sense**: This is the property of a device to sense certain characteristics. There are three common types of sensing devices. Position sensing devices sense their position and orientation. Motion sensing devices sense the distance/angle they have moved from a particular position. Force sensing devices sense the amount of force being applied on them.
3.2 Devices

![3D Ball](image1)

Figure 1. A 3D ball device. Image taken from [4].

3D Ball [4]: A 3D ball is a small spherical plastic device about 2 inches in diameter. It is basically used to rotate objects. It encloses a tracker and provides 6DOF. The form factor of the ball is very good, since humans inherently tend to rotate spherical objects. Its surface however offers very few tactile landmarks preventing users from getting a clear conception of the device. The cord of the ball is also another major hindering factor which the users find annoying, since it is very heavy and often gets in the way of a rotation.

![Tracker](image2)

Figure 2. A tracker device. Image taken from [4].

Tracker [4][28] A tracker is basically a 3D ball without the spherical encasing. It comes many shapes but the most popular one is the rectangular one. It suffers from the same advantages and disadvantages as a 3D ball. There are various types of trackers
available depending on the technology such as optical tracker, acoustic tracker and magnetic tracker. Optical tracker uses light waves to detect the position and orientation. Similarly acoustic tracker employs sound waves and magnetic tracker uses magnetic waves.

![Image of 3D mouse](image3)

Figure 3. A 3D mouse device. Image taken from [28].

3D Mouse [28]: A 3D mouse is very similar to a 2D mouse the only difference is that it has a roller to move the cursor farther from and closer to display.

![Image of Rockin mouse](image4)

Figure 4. A Rockin mouse. Image taken from [29].

Rockin Mouse [29]: A Rockin mouse is an extension of the 2D mouse to allow the user to work using 6DOF if desired. It has curves at the side and can be tilted to achieve
manipulation along an extra dimension. Since it is backward compatible to a mouse it is a very practical 3D input device.

Figure 5. A head mounted display. Image taken from [47].

Head Mounted Display [11]: A Head mounted display is used to give the user the feeling of being within three dimensional space. It helps the user to better understand the relationships between the 3D objects. Its main disadvantages are its weight and the fact that it literally cuts the user off from the real world.

Figure 6. A 3D glove device. Image taken from [48].

19
3D Glove [30]: A Data glove is a basically a tracker in the form of a glove. Besides detecting position and orientation it also senses gestures input using fingers. It aims at utilizing the dexterity and skill of the user’s hand. There have been various popular glove technologies a few of which are Sayre glove, MIT LED glove, Digital data entry glove, etc. Glove based input can be best used in applications such as sign language interpretation, computer based puppetry, musical performance, etc.

![Figure 7. A bat device. Image taken from [45].](image)

Flying Mice [26][8]: These are devices based on the mouse but which are operated by holding, moving and rotating them in air. The most popular “flying mice” device is the *Bat*. These devices are easy to learn because of the natural and direct mapping and also have fast speed of operation. However their disadvantages are limited movement range, lack of coordination, fatigue and difficulty in device acquisition.
Elastic devices [26]: These devices are force operated. They move in proportion to the force applied on them but have a self centering mechanism which causes them to return to their original position once released. They have the advantage of providing both displacement and force feedback.

Multi-DOF armatures [26]: These are mechanical armatures pivoted at a one end point and calculate the relative position of the other end point, which is then mapped onto
a corresponding cursor movement on the screen. The major disadvantage with these
devices is their limited applicability and constrained operation.

In this chapter we have discussed numerous input devices and their
characteristics. There are still a lot of unexploited scenarios which will probably be used
in the future. With the emergence of every new technique old problems will disappear
and new ones will surface. However the aim should always be to use technology to make
interaction better and easier and [4] we should keep in mind that the choice of a device
will always depend on the application it is required to support and the intended users.
CHAPTER 4
3D INTERACTION

3D interaction is a complex and challenging task. Various techniques have been used in the past to make this interaction process comfortable and easy for the user. In this chapter we will first define ‘direct’ and ‘indirect’ 3D interaction. We will then discuss in brief about the major disadvantages of using the indirect approach and the corresponding advantages when using the direct approach. After justifying the need for a direct approach to 3D interaction we will list the significant disadvantages of using true 3D devices followed by the advantages of using the mouse.

4.1 Direct and indirect 3D interaction
If the interaction process requires the user to model, manipulate or orient an object by making changes to the object itself, the interaction is classified as ‘direct’ interaction. On the other hand if the interaction process requires the user to model, manipulate or orient an object by making changes to one or many controllers linked to the object, the interaction is classified as ‘indirect’ interaction. Currently direct 3D interaction is mostly achieved using true 3D devices and indirect 3D interaction is achieved using 2D devices.

4.2 Disadvantages of indirect interaction
Indirect interaction although easier to achieve has certain distinct disadvantages. Venolia in his paper [28] states that although everyday more and more 3D tools surface they are not gaining popularity because of their complicated indirect interface. Below we mention the main disadvantages that users face when using the indirect approach.
1. **Unnatural:** It is natural human tendency to directly touch any object which he/she desires to manipulate or orient. This makes indirect interaction unnatural since the user has to manipulate the object by making changes to a controller and not the object itself.

2. **Poor feedback:** Using indirect interaction, the user interacts with a controller and the corresponding changes are reflected on the object. This forces him to divide his attention between two areas, namely the controller which he is manipulating and the object which is being indirectly affected. This causes the feedback from the controller to the user and the feedback from the object to the user to interfere with each other rendering the overall feedback poor.

3. **Complex to understand:** In indirect interaction changes to a controller placed somewhere far from the actual object causes modifications to the object. This indirection often becomes complex to understand.

### 4.3 Advantages of direct interaction

Direct interaction may be difficult to achieve but holds noteworthy advantages. Chen, Mountford and Sellen in their paper [18] emphasize the strong need for simple direct manipulation in numerous areas such as engineering design and architecture. Similarly Sachs, Roberts and Stoops [13] conclude that the easiest way to develop 3D models on the computer is by directly drawing in the 3D environment. The significant advantages of using the direct approach are mentioned below.

1. **Natural:** Using direct interaction, changes to the object are made directly to the object itself. This is in conformance with natural human interaction tendency.

2. **Rich feedback:** Using direct interaction the user has to focus his attention solely on the object he is currently modifying. There is no interference in the flow of feedback from the object to the user. Furthermore a variety of feedbacks such as tactile, pressure, etc. can be used depending on the system resources; the application and the user requirements leading to a significant enhancement and ease in the interaction experience of the user.
3. *Easier to understand:* In direct interaction changes made to the object affect the object itself and there is no indirection involved. This makes it much easier to understand as compared to indirect interaction.

4. *Reduced burden on user:* Direct interaction is aimed at making the interaction process easy and comfortable for the user mostly at the expense of increasing the complexity of the underlying system which facilitates the interaction. This helps in increasing user performance as well as enhancing his/her productivity and efficiency.

### 4.4 Disadvantages of true 3D devices

True 3D devices are designed to facilitate interaction in a 3D environment. However, almost all the 3D interaction today takes place using 2D displays [6], and in a two-dimensional environment [7]. This often makes them a poor choice for most 3D interaction applications being used today. The major disadvantages of true 3D devices are listed below.

1. *Precision lost due to instability:* Although using a true 3D device enhances the speed with which the user can work, precision is often sacrificed in the bargain [4]. These devices are tough to control due to the instability of moving them in air. On the other hand, the overall speed may in fact be reduced if we try to achieve precision using these devices.

2. *Expensive [4]:* Most true 3D devices are expensive and that makes it difficult for the common users to own them. Their use is hence limited to research or applications such as medical visualization, virtual reality systems, etc.

3. *Complex to understand [18]:* True 3D devices are uncommon and queer; hence most users find it difficult to learn their use.

4. *No versatility [26]:* Most of the true 3D devices have limited applicability and cannot be applied to a wide range of applications. This lack of versatility is a major hindering factor for these devices.

5. *Rate control [26]:* Since true 3D devices offer 6DOF and are generally held in air when used, controlling the rate of these devices is a difficult task both, to learn and to achieve.
6. **Lack of control feel [26]:** Often the 6DOF offered by these devices gives the users a ‘lack of control’ feeling which hinders their productivity and efficiency.

7. **Fatigue:** Most of the true 3D devices are either worn on the body/head or held in air to operate them effectively. This significantly increases the fatigue level of the user.

8. **Device acquisition time:** The time required to engage most true 3D devices is so high that users can often accomplish tasks faster without using them.

9. **High sampling rate requirement [23]:** A very high sampling rate is required when using true 3D devices in a virtual 3D environment to achieve smooth viewing.

10. **Lag [23]:** In a 3D environment, real time rendering of the virtual environment is a processor intensive task and may take significant amounts of time, hence to reduce the overall lag, the lag from the true 3D devices themselves should be as less as possible.

11. **Poor perceptual structure [23]:** Most true 3D devices have strange structures which makes them unintuitive and non-metaphoric. This makes it difficult for the user to construct a mental model of these devices.

12. **Other ergonomic considerations [31][6]:** A lot of other ergonomic considerations such as the accuracy, pleasure, constraints offered by the human body, etc. prevent effective use of true 3D devices.

**4.5 Advantages of the mouse and extending its use to 3D interaction**

Currently the mouse is extremely popular for use in 2D applications. It possesses numerous advantageous characteristics which contribute to its popularity. We propose to extend the use of the mouse to 3D interaction. Listed below are the major advantages which support our proposal.

1. **Popular [18][24][25][32]:** Currently a mouse is almost a must buy with a computer and is one of the most popular and dominant devices.

2. **Integration of 2D with 3D [28]:** The mouse functions excellently in a 2D environment. If it can be extended to be used in a 3D environment, we can achieve integration of 2D and 3D task domains which in turn will enhance the productivity and efficiency of the user.
3. **Ease of use**: The mouse being very easy to use, we can safely assume that almost all the computer users have used a mouse extensively and have a highly reliable mental model of the mouse.

4. **Zero device acquisition time**: Hinckley, Tullio and Pausch [4] state that mouse based techniques are slower in a work routine but if we consider the time required for switching over to a 3D device and back to the mouse, the mouse based technique becomes faster. If the use of the mouse is extended to 3D, switching between 2D and 3D tasks will not require change of device making the device acquisition time zero.

5. **Good control in 3D environment** [28]/[32]: Studies done in the past have indicated that users can easily control a 3D cursor using a mouse and may even overcome the complexity involved with using the mouse in a 3D environment.

6. **Good form factor** [29]: The physical shape of the mouse is well suited to be held in the human hand and does not restrict the user to any particular grip.

7. **High stability** [29]: The mouse rests on the surface of the desk, is quite heavy and has a large area of contact with the surface of the desk. All these factors make it very stable.

8. **Less fatigue** [29]/[32]: The mouse is not required to be worn or held in air and does not considerably increase the fatigue level of the user. The forearm can easily rest on the table when operating a mouse and since it is a relative device it needs very little arm movement.

9. **Clutching easily solved**: Since most of the users have a clear mental model of the mouse, they can easily understand and execute clutching with respect to the mouse.

10. **Device to cursor mapping** [29]: The mapping of the mouse to the cursor is very natural and thus reduces the cognitive load on the user.

11. **Button positions** [29]: The button positions on the mouse are well suited to human operators. The directions of buttons are orthogonal to the sensing dimension of the mouse which makes it easy to use the buttons without unintentionally moving the mouse.

12. **Familiarity** [29]: Since the users are already familiar to the mouse and its operations there is a high probability that their productivity and efficiency will be high even when using the mouse in a 3D environment.
Foley, Wallace and Chan [31] in their paper mention that any system designed for interaction must minimize the work required by three types of basic human processes; *perception*, *cognition* and *motor activity*. Perception is the process in which incomprehensible stimuli is received, transmitted to the brain and then received by the receptor organs. Cognition is the process using which we acquire, organize and retrieve information. Motor activity can be defined as the physical response to stimuli after perception and cognition have taken place. Our proposal of using the mouse for direct 3D interaction aims at facilitating these three processes.
CHAPTER 5
CHALLENGES OF 3D INPUT USING THE MOUSE

A mouse is inherently a 2D device. Using it for 3D input involves overcoming numerous challenges. This chapter discusses these challenges and mentions some of the design strategies we have devised to overcome them.

Coordinated movement in 3D space which involves all the three axes at the same time should be avoided [26] and in order to effect 3D transformations using a mouse, the user will have to decompose the 3D task into a series of 1D or 2D tasks [6]. Thus the user will be forced to think in terms of one or two dimensions. Additionally the curved geometry of 3D space is very different from the flat workspace of the mouse and since a 2D device inherently allows only for 4DOF, using it for 6DOF may seem awkward and uncomfortable to the user [16][17][28].

The mapping will have to be from 2D movements of the mouse to 3D space [20][32]. Though this is possible, such mappings are usually hard to predict and to a certain extent unnatural. This also implies that the system will have some degree of complexity and unnaturalness and hence the user will have to spend some time learning to be able to use the system effectively and efficiently. In our design we provide a functionality known as ‘helpers’ aimed at assisting the user in learning and preferably mastering the mouse movements.

Since we aim to achieve direct interaction, the interface should be as transparent as possible and the user should at all times feel as if he is interacting directly with the object and not with the interface [6]. To achieve this, the mapping of the device to the cursor needs to be as simple and natural as possible. As far as possible, stimulus-response
correspondence needs to be achieved between the hand movement and the on-screen cursor. At the same time we should take care to see that we do not incorporate any movement that will be ergonomically worse than the normal mouse operation [6].

The user movement should be filtered and kept track of, to cause corresponding changes on-screen. Also the interpretation needs to be consistent in order to provide positive feedback to the user to help him create a reliable mental model of the system. For example; if the user is moving the mouse in the ‘X’ direction, the system should consistently recognize this as movement in the ‘X’ direction. In order to do so, we frequently sample the mouse movement and based on the last two samples we interpret the direction of his/her movement.

To help the user’s perception, we have to increase the quality and the quantity of information displayed for him/her on-screen [32]. At the same time to avoid cluttering and obscuring important areas of the virtual environment there should not be too much information displayed on the screen. It is very important to achieve an optimum level between these two factors; namely quality and quantity.

The user should be provided with an overall frame of reference and numerous cues to help him know the position of the 3D cursor in relation to the 3D environment. To facilitate this, the system provides a bounding cube which serves as an overall frame of reference. Additionally there are two major cues which we call ‘legs’ and ‘projections’ which help the user to be aware of the cursor’s position in 3D space.

If we display numerous views they occupy a big area of the screen. At the same time the size of each view becomes smaller thus reducing clarity and increasing complexity. To remedy this, we use only a single orthographic view. Also since there is only one view to be displayed it can occupy a major area of the screen, thereby enhancing clarity.
Another similar issue is to reduce the screen space occupied by the buttons. If all the buttons are displayed all the time they eat-up on the screen space that can be used to display the 3D environment. We chose to use tabs for this purpose. Tabs not only reduce the screen space occupied by the buttons but also provide classification and structure to the button organization.

Visibility is another factor that has to be given utmost importance. To enhance visibility and to prevent obscurity, all the objects are rendered as wire-frame models. Also all of the important areas such as the XY, YZ and ZX planes and X, Y and Z axes, etc. are rendered using colors that are not used anywhere else in the environment making them distinct and easily noticeable.

The user must at all times be aware of the state of the system i.e. he/she should know what action the system is expecting from him/her. This is facilitated by displaying a message for him/her on the status bar about the current state of the system and making the button corresponding to the current state prominent.

The complexity involved with the modeling of advanced entities such as a cube, pyramid etc. should be reduced. Our software employs the ‘divide and conquer’ strategy wherein advanced entities are developed by building on smaller ones, which are easy to model. For example to plot a square the user has to simply plot four points. Similarly, to plot a cube he/she has to plot a square (i.e. one face of the cube) and the program automatically provides him with the rest of the faces.

Constraints should be designed and implemented to make the tasks easier for both the user and the computer [3] and since we will not be able to use tactile or force feedback, the visual feedback should be as strong and rich as possible. Also the interface should place low mental burden on the user, i.e. the users must not be required to remember the interface, and it should come naturally to them.
In our design utmost importance was given to all the factors mentioned above. Failure to take care of any of the above criteria would have rendered the interface unnatural, unintuitive, and unproductive.
CHAPTER 6
DIRECT 3D INTERACTION WITH A 2D LOCATOR DEVICE

In this chapter we put forth a design in which we try to incorporate all the issues we talked about in the earlier chapters. We also discuss in detail the prototype software, which we call ‘Direct3D’, which was implemented based on the design. The aim of the prototype is to help everyone understand the practicality and the advantages of the approach. At the same time we expect the prototype to serve as a basic building block for any future research in this area.

6.1 Workspace and viewpoint

Today most 3D modeling and interaction applications designed to be used with 2D devices present the user with numerous views such as top view, bottom view, left hand side view, right hand side view, etc. Such a design forces the user to interact and work in a 2D environment to build 3D objects. In our design we have tried to overcome this drawback.

The user is provided with a single orthographic view of the coordinate system, which we will henceforth call ‘Direct3D view’, as shown in Figure 10(b). All objects are drawn in the Direct3D view using the mouse. The user viewpoint is located on the line whose equation is given as \( x = y = z \) in the first octant (where \( x \) is positive, \( y \) is positive and \( z \) is positive). The distance of the viewpoint, on this line, from the origin depends on the scale of the coordinate system being employed.

A cut-away view of cube (i.e. an octant) is provided as a frame of reference. All movements and transformations are limited to be within this cube. If a transformation or
operation causes a point to move outside this cube, the transformation or operation is not applied.

All the three axes are displayed in white color and are clearly marked with the corresponding letters ‘X’, ‘Y’ and ‘Z’. The three planes XY, YZ and ZX are drawn in the form of grids, using unique colors.

The advantages of all the above design decisions are listed below.
1. The user is allowed to work and interact in a 3D environment.
2. Since there are lesser views, complexity is reduced.
3. The view area can be significantly magnified since there are lesser number of views to display.
4. The time wasted in switching from one view to another is eliminated.
5. In an orthographic view, the length of a line displayed on screen does not change with the depth of the line; hence it is very suitable for modeling applications where it is important to know the true length of a displayed line at all times.
6. The grid form of the XY, YZ and ZX planes makes them stand out and at the same time does not hamper visibility.
7. The viewpoint is in the middle of the screen as well as in the middle of the coordinate system and gives the user an ‘exocentric’ view.
8. The X, Y and Z axes are clearly drawn and marked and the origin is in the center of the screen.
9. The framing cube serves as an excellent frame of reference, using which plotting entities becomes much easier.

Besides the main view, Direct3D view where the entire drawing takes place there is also another small view to the left of Direct3D view, which we will henceforth call ‘Tab View’ as shown in Figure 10(a). This view contains buttons which allow the user to use the various functions built in the software.
Figure 10. A screenshot of Direct3D when it has just been started showing (a) The Tab view. (b) The Direct3D view.
At the bottom of the Tab view there are four tabs labeled as mentioned below:

1. Point
2. Line
3. Polygon
4. Polyhedron

Clicking on each of these tabs causes the corresponding buttons to be displayed and the other buttons to be hidden. For example, if the current selected tab is ‘Point’, only buttons related to transformations and operations for points (Plot Points, Move Points, etc.) are displayed.

The tabbed view has the advantage of saving on-screen space and providing a classification and structure to the button organization. Also it prevents the user from committing mistakes by hiding the irrelevant buttons. Each of the tabs and buttons has images signifying their functionality. The selected tab is raised compared to the other tabs and the selected button is displayed using the pushed-down effect. This functions to avoid confusion and ambiguity since the user can explicitly see which tab and which button are currently selected.

Besides the buttons, the Tab view contains three pairs of radio buttons. These correspond to functionalities we call ‘Legs’, ‘Projections’ and ‘Helpers’. These functionalities will be explained in the later sections. These radio buttons serve to enable/disable the corresponding functionality.

6.2 Input mapping

A significant challenge was to design a novel and simple way to map the 2D mouse movements onto 3D space. The mapping we propose in this research relies to a significant extent on gestures, the most well known way of communicating with the users. The mapping is summarized below.

1. A horizontal movement of the mouse towards the right with the left mouse button held down causes increase in the X coordinates of the current selection.
2. A horizontal movement of the mouse towards the left with the left mouse button held down causes decrease in the X coordinates of the current selection.
3. A vertical movement of the mouse upwards with the left mouse button held down causes increase in the Y coordinates of the current selection.
4. A vertical movement of the mouse downwards with the left mouse button held down causes decrease in the Y coordinates of the current selection.
5. A diagonal movement upwards and towards the right with the left mouse button held down causes decrease in the Z coordinates of the current selection.
6. A diagonal movement downwards and towards the left with the left mouse button held down causes increase in the Z coordinates of the current selection.

The direction of the intended movement is estimated by repeatedly sampling the 2D movement of the mouse and comparing it with the last two samples and then using that as the basis for mapping to a 3D space movement.

Pseudo-code for the above mapping is outlined in Appendix A. The mapping of the mouse to the 3D space described above has the following significant advantages:
1. It is very simple to understand. There are absolutely no complex or ambiguous movements.
2. The movements are all straight line movements, each controlling only one axis, and hence are easy to control.
3. The control area for each axis are well isolated and there in no confusion due to a movement translating into changes to more than one axis.
4. The general assumption and mental model of most users are that the X axis is horizontal, positive towards right and negative towards left. The Y axis is vertical positive upwards and negative downwards. The Z axis is perpendicular to the screen with the negative part going into the screen and the positive part coming out of the screen. The mappings described above very closely reinforce the user’s assumption. Hence the movements seem very natural to the users.
5. Since the last two samples (rather than just the last one) are used it is an effective estimate of the user’s intention.
6.3 Transformations and operations

We treat transformations and operations as modes. The system can be in only one mode at any given point of time. By default the system is in the ‘Point plotting’ mode. To switch to a different mode the user has to simply click on the corresponding button.

The software makes the following three basic kinds of transformations explicitly available to the user.
1. Translation
2. Rotation
3. Scaling

To enable ease of use and to prevent complexities the mapping described earlier is used for these transformations as well. Horizontal movement causes Translation/Rotation/Scaling of the selection in the X direction. Vertical movement causes Translation/Rotation/Scaling of the selection in the Y direction. Diagonal movement causes Translation/Rotation/Scaling of the selection in the Z direction. Building on these basic transformations the user can achieve even advanced complex transformations such as through point constraint, planar constraint, etc.

The software also provides various operations to the user. These operations range from plotting simple points to plotting complex polyhedrons. Described below are each of these operations in detail.

Plot points: In this mode the user can move the cursor in the required direction using the generic mouse operations. Once he/she is satisfied with the location he/she can use the right mouse button to plot a point at that location.

Move points: In this mode the user can select an already plotted point. Once a point is selected he/she can move it to the desired location using the generic mouse operations. No right click is necessary to end this operation.
Delete points: In the point deletion mode the user can delete any of the already plotted points. If the point is a part of a line, the corresponding line is deleted but the other end point of the line remains untouched. Similar is the situation if the point is a part of a polygon or a polyhedron.

If there are no points in the drawing area and the user clicks in the drawing area, a message is displayed informing him that there are no points to delete. Similar is the case if the last point in the drawing area is deleted. In such a situation a message pops-up informing him that there are no more points to delete.

Plot lines: To plot lines, the user has to simply plot two points one after the other and the program automatically draws a line between them.

Plot line-strips: To plot line-strips, the user can plot as many points as he wants consecutively, the program automatically draws lines between every two consecutive points plotted.

Move lines: To move lines, the user clicks on the desired line. The program then highlights the mid-point of the line and the user can use this mid-point as a handle to move the line to the desired location.

Rotate lines: To rotate lines, the user selects the desired line. After the mid-point is highlighted he/she can use it as a handle to rotate the line about the desired axes to the desired orientation.

Scale lines: To scale lines, the user selects the desired line. Then using the highlighted mid-point he/she can scale it along the desired axes.
Figure 11. A screenshot of Direct3D showing (a) The original line plotted. (b) The original line translated -10 units along the X axis. (c) The original line translated +20 units along the X axis and then rotated about the X axis. (d) A line-strip plotted.
Plot polygons: To plot polygons, the user plots points, one less than the number of vertices he wants the polygon to be made up of. The program draws a line between every two consecutive points plotted. To plot the last vertex, instead of a right-click, he/she has to double click the left mouse button to signal that the vertex just plotted was the last vertex. The program then automatically draws a line between the last and the first point to close the polygon.

Move polygons: To move a polygon, the user selects the desired polygon. The program highlights the centroid of the program. The user can use this as a handle to move the polygon to the desired location.

Rotate polygons: To rotate a polygon, the user selects a polygon and uses the highlighted centroid to rotate it about the desired axes to the desired orientation.

Scale polygons: To scale a polygon, the user selects the polygon and uses the highlighted centroid to scale it along the desired axes.
Figure 12. A screenshot of Direct3D showing (a) A pentagon plotted. (b) The original pentagon translated along the X axis and then scaled along the Y axis. (c) A hexagon plotted. (d) The original hexagon translated along the X and Y axes and then rotated about the Z axis.
To decrease the number of buttons and to reduce complexity we have incorporated two generic polyhedron plotting operations. Using these operations we can plot numerous types of regular polygons. These operations are mentioned below

Plot center lift polyhedrons: By center lift we mean that after the user has finished plotting the base polygon, all the vertices of the base are connected automatically by the program to the centroid of the base. The user can then move the centroid to the desired location to obtain a polyhedron.

To plot a center lift polyhedron, the user plots the base polygon. After that the program highlights the centroid of the polygon and the user can move it to the desired location.

Plot edge lift polyhedrons: By edge lift we mean that after the user has finished plotting the base polygon the program automatically duplicates the base polygon. All the vertices of the base are connected automatically by the program to the corresponding vertices of the duplicate polygon. The user can then move the duplicated polygon to the desired location to obtain a polyhedron.

To plot an edge lift polyhedron, the user plots the base polygon. After that the program duplicates the entire base polygon and highlights the centroid of the duplicate polygon. The user can move this polygon to the desired location using the centroid as a handle.

Move polyhedrons: To move a polyhedron, the user selects the desired polyhedron. The program automatically highlights the centroid of the polyhedron. The user can use the centroid as a handle to move the polyhedron to the desired location.

Rotate polyhedrons: To rotate a polyhedron, the user selects a polyhedron. Then using the highlighted centroid he can rotate it about the desired axes to the desired orientation.
Scale polyhedrons: To scale a polyhedron, the user selects a polyhedron and then uses the highlighted centroid to scale it along the desired axes.

Legs: When plotting/moving a point or an entity (line, polygon or polyhedron) using the centroid, the program draws three straight lines from the current selected point to the three planes XY, YZ and ZX. These lines are what we call ‘Legs’. These legs have bases at the point where they meet the XY, YZ and ZX planes. Legs can be turned on or off by using the pair of radio buttons provided in the Tab view.

Projections: When plotting/moving a point or an entity (line, polygon and polyhedron) using the centroid the program draws shadows of the legs corresponding to the current selected point onto the three planes XY, YZ and ZX. These shadow lines are what we call ‘Projections’. These projections pass through the base of the legs and are parallel to the corresponding axes. Projections can be set to on or off state by using the radio button pair in the Tab view.

Helpers: ‘Helpers’ are used only to help beginners get used to the mouse movements. These are lines drawn on screen and which pass through the current cursor position. A horizontal helper line is drawn if the user movement is along the X axis. A movement along the Y causes a vertical helper line to be displayed. Similarly a diagonal helper line is displayed for movements along the Z axis. They can be turned on or off using the radio buttons provided in the Tab view.
Figure 13. A screenshot of Direct3D showing (a) An edge-lift polyhedron i.e. a parallelepiped plotted. (b) A center-lift polyhedron i.e. a pyramid plotted and then rotated about the Y axis.
6.4 Feedback

The design incorporates numerous, primarily visual feedback techniques. The significant ones are listed below.

Button images: Each button has an image on it which signifies the function it performs. These images are used to help the user understand the functionality of the button by just looking at them. Button images are illustrated in Figure 14.

![Figure 14](image1)

(a) The button images displayed when the Point tab is selected. (b) The button images displayed when the Line tab is selected. (c) The button images displayed when the Polygon tab is selected. (d) The button images displayed when the Polyhedron tab is selected.

Figure 14. The button images used in Direct3D. (a) The button images displayed when the Point tab is selected. (b) The button images displayed when the Line tab is selected. (c) The button images displayed when the Polygon tab is selected. (d) The button images displayed when the Polyhedron tab is selected.
Tab images: Similar to the buttons, each of the tabs has an image on it. The image signifies the four entities namely point, line, polygon and polyhedron. Clicking on each tab causes the corresponding buttons to be displayed and the others to be hidden. The image on the tabs helps the user know the functionality of each just by looking at it. Tab images are illustrated in Figure 15.

![Tab images](image1.png)

Figure 15. The tab images used in Direct3D.

Cursor change: There are four cursors used in the program. Three of these are the X, Y and Z cursors.

![Cursors](image2.png)

(a) The cursor displayed when the movement is along the X axis. (b) The cursor displayed when the movement is along the Y axis. (c) The cursor displayed when the movement is along the Z axis. (d) The cursor displayed in the Point deletion mode.

Figure 16. The cursors used in Direct3D.
The X cursor, as shown in Figure 16(a), is a horizontal line with arrows at both ends and appears whenever the mouse moves along the X direction with the left mouse button held down. The Y cursor, as shown in Figure 16(b), is a vertical line with arrows at both ends and appears whenever the mouse moves along the Y direction with the left mouse button held down. The Z cursor, as shown in Figure 16(c), is a diagonal line with arrows at both ends and appears whenever the mouse moves along the Z direction with the left mouse button held down. These three cursors give feedback to the user to help him determine the axis along which he is moving.

The fourth and the last type of cursor used is the hand cursor, as shown in Figure 16(d). It is used when in point deletion mode. The finger can be used to exactly pin-point to the point to be deleted. At the same time a hand cursor immediately conveys to the user that he is in point deletion mode.
Figure 17. A screenshot of Direct3D to illustrate various feedback techniques. (a) The selected button. (b) The tabs. (c) The radio buttons. (d) The status bar displaying the current state of the system and the help message corresponding to the current state of the system. (e) A part of the 3D environment showing a selected polyhedron being translated along the Y axis.
Button pushed effect: A button when selected is displayed using a *pushed effect* as illustrated in Figure 18. All the other buttons are comparatively raised. This helps the user be aware at all times of the selected button by just looking at it. He/She is thus also aware of the current mode.

![Figure 18. Magnified illustration of part (a) of Figure 17 showing the button-pushed effect.](image)

Tab raised effect: The selected tab is raised as compared to the other three as illustrated in Figure 19. This gives feedback to the user and helps him know the entity which the system expects him to interact with.

![Figure 19. Magnified illustration of part (b) of Figure 17 showing the tab-raised effect for the Polyhedron tab.](image)
Radio buttons: The radio buttons which are visible at all times are clearly marked on/off, as illustrated in Figure 20. Each of the radio buttons unambiguously indicates the state of the corresponding functionality.

![Radio buttons illustration](image)

Figure 20. Magnified illustration of a part (c) of Figure 17 showing the radio buttons.

Help message display: The status bar displays the current mode i.e. system state and a help message about the current mode as illustrated in Figure 21. The message explains to the user the methodology for working in the current mode.

![Status bar illustration](image)

Figure 21. Magnified illustration of a part (d) of Figure 17 showing the status bar displaying the current state of the system and a help message corresponding to the current state of the system.

Legs: The legs are used to give the user a visual estimate of the distance of the current point from the XY, YZ and ZX planes. Legs are illustrated in Figure 22(a).

Projections: Although the projections can be used alone, they function best in combination with the legs. They add to the information provided by the legs and give the user a visual estimate of the distance of the current point from the X, Y and Z axes. Projections are illustrated in Figure 22(b).
Helpers: Helpers are designed to be used only by novice users when learning the mapping of the mouse movements to the 3D space. The program interpolates the mouse movement, calculates the direction of the mouse movement and displays the corresponding helper. This gives positive feedback to the user and helps him learn the technique of confining the mouse movement to a horizontal, vertical or diagonal line. Helpers are illustrated in Figure 22(c).

Selection highlighting: Any entity selected by the user is highlighted with a color (i.e. magenta) that is not used anywhere else in the workspace. This clearly displays information about the current selection. Selection highlighting is illustrated in Figure 22(d).

Dynamic coordinate display: The coordinates of the current selected point are clearly displayed upwards and to the right of the point in the drawing area. These coordinates are updated dynamically as the point undergoes transformations. Dynamic coordinate display is illustrated in Figure 22(e).

Colors for planes: The XY, YZ and ZX planes are all drawn using unique and different colors to help them stand out in the drawing area.

Colors for axes: The three axes X, Y and Z are drawn and marked using the white color which is not used elsewhere in the drawing area.

Grid for visibility: The XY, YZ and ZX planes are drawn as grids to enhance the visibility of the drawing area.

Pop up messages: Some feedback is also provided using pop-up messages. For example; when the last point in the drawing area is deleted a message pops up as illustrated in Figure 23 which informs the user that there are no more points to delete.
Background: The background color is gray which is not used anywhere else in the drawing area. Gray is used in almost all 3D viewing software as it is soothing to the eyes and at the same time has good visibility.
Figure 22. Magnified illustration of a part (e) of Figure 17 showing (a) The legs (three lines, white in color, converging at the orange point). (b) The projections (six lines, black in color, drawn on the XY, YZ and ZX planes). (c) The helper line for movement along the Y axis (the vertical black colored line) (d) Selection highlighting i.e. the selected polyhedron (magenta colored vertices). (e) The dynamic coordinate display.
Figure 23. A screenshot of *Direct3D* showing a pop-up message displayed when the user has just deleted the last point in the workspace.
Natural mapping: The horizontal, vertical and diagonal movement of the mouse is how most users inherently perceive the directions of the X, Y and Z axes respectively to be. Hence this gives a very good mental feedback to the users.

True measurement: The orthographic view helps us preserve line lengths at varying depths. Hence at all times, the user can see and compare the true length of the lines drawn, which is very useful for modeling objects.

Window size: The window size of the program varies with the screen resolution. However for any given resolution the window size is fixed and cannot be changed so as to preserve the orthographic view and utilize the maximum screen space possible. To ensure this the maximize/restore-down button and the resizing operations of the window are disabled.

Instant display update: The drawing area is updated immediately and there is unnoticeable lag between an operation/transformation and the screen refresh.

Direct interaction: The best feedback of the software is that the users can directly interact with the entities. They can directly select the entity and apply various transformations to it. There is no indirection associated with any of the operations or transformations.

6.5 Development system specifications

Direct3D has been developed to work in the Windows environment. We have used a combination of Visual C++ 6.0 and OpenGL to develop the software. The system we have used for the development of the software has a Pentium III processor with 128 MB RAM. The Operating system used is Windows XP and the recommended screen resolution is “1280x1024”. However the software has been tested on a range of machines slower and faster and has functioned successfully on all of them.
CHAPTER 7
APPLICATIONS

The software we have developed is a prototype with limited utility. However the design offers a lot of potential and with the incorporation of additional functionality, it could be used in a range of applications. A few of the probable application areas are listed below.

Architecture: The design can be used to handle the input to software which allows architects to build virtual building models, preview them and show them to the builders so as give them a clear idea of what they want. Any changes that may be requested can be easily and quickly incorporated. Figure 24 shows the snapshot from one of the popular architectural software, *3D Home Architect*.

![Figure 24. A screenshot of 3D Home Architect. Image taken from [49]](image-url)
Manufacturing: The design can be of great use to mechanical part designers when used in combination with CAD software. The combined software will be capable of handling input from the mouse and allow interaction and modeling in a 3D environment. Thus the designers will also be able to build virtual parts quickly and in a single view without the use of any specialized devices.

![Image of mechanical CAD software](image.png)

Figure 25. A screenshot of mechanical CAD software. Image taken from [50]

Simulation training: The design can be used in combination with training software to simulate various environments. Such combined systems could be made widely available since the only requirements will be a computer and a mouse. Some of the probable areas that fall under this category are military training, areas of manufacturing industry that are hazard prone, etc. Since this design uses simple 2D locator devices such as the mouse, it will allow the users to focus on the environment rather than the device they are using.
Medical visualizations: The design can benefit software used in medical visualization. For example: Novice doctors and surgeons can explore the human body and interact with it using a specialization of our design and the mouse as an input device. This will help make the software cheaper and easier to use.
Animation: Interactive animation software can be augmented with our design. The combined system can help animation artists work faster and more productively to develop animations that can be used in numerous areas such as movies, etc. The design can also be used in conjunction with educational software and games to allow users to explore a 3D environment using just the mouse and a few simple movement techniques.

![Animation Software Screenshot](image1)

Figure 28. A screenshot of animation software. Image taken from [53]

Geometric modeling: 3D Geometric modeling can be made simpler and faster to achieve using our design. This in turn may have a cascade effect and may benefit many areas which depend heavily on geometric modeling.

![Geometric Modeling Software Screenshot](image2)

Figure 29. A screenshot of geometric modeling software. Image taken from [54]
Drawing/Sketching in 3D: Using our design, drawing and sketching in a 3D environment can be made available to novice computer artists where they can practice and enhance their skills at no cost. The professional artists can enhance their productivity and efficiency since they will be relieved from the burden of having to use specialized devices.

Figure 30. A 3D sketch. Image taken from [55]

Other research areas: Various areas of research such as genetics and molecular modeling can benefit from the design. Using the design, the software currently being used for these applications can be enhanced to allow 3D interaction using inexpensive and popular 2D locator devices such as the mouse.
Thus our design can be used as an *Input/Output device* to supplement software used in numerous important areas. New software can be custom built to be compatible with our design and existing software can be reverse engineered to incorporate the design.
CHAPTER 8
EXAMPLE SESSION

In this chapter we will be modeling an object using the prototype we have developed. Through this we aim to highlight the ease, the practicality, the advantages and the capabilities of our design. The object we will be modeling is an airplane. The reason behind the choice is that, an airplane is not very complex to model. Hence, with a little practice, even novice users can easily model it. At the same time the model demonstrates the potential held by our design and allows us to exhibit a variety of functionalities supported by the prototype.

8.1 Modeling the nose
1. Start the Direct3D software.
2. The initial state of the system will show the Point tab selected i.e. raised with Point plotting as the current mode as indicated by the pushed ‘Plot points’ button. The legs and projections are turned on and the helpers are turned off by default as reflected by the group of radio buttons. The current location of the cursor is at (100,100,100) as represented by the orange colored point.
3. We will be plotting a pyramid i.e. a center lift polyhedron, as the nose of the airplane. To do so click on the polyhedron tab to display the buttons related to polyhedron modeling and manipulation.
4. Observe that the polyhedron tab becomes raised indicating that it is selected.
5. Click on the plot center lift polyhedron button. Observe that the button remains pushed indicating that it is currently selected.
6. The status bar also displays the current mode i.e. ‘Center Lift Polyhedron Plotting Mode’. It also displays a help message to the user explaining to him to the
methodology of working in the center lift polyhedron plotting mode i.e. ‘Plot the base polygon and then move the center point to form a polyhedron’.

7. The cursor will be at the location (100,100,100) by default. As a first step, we want to plot the base of the pyramid i.e. a square, on the plane whose equation is \( z = 90 \) and we want the coordinates of the base vertices to be (100,100,90), (90,100,90), (90,90,90), (100,90,90).

8. To begin we need to move the cursor to (100,100,90). To do so move the mouse diagonally upwards and to the left i.e. in the negative Z direction.

9. Observe that the cursor changes to the Z cursor during the movement along the Z axis and the legs and projections move correspondingly as we move the cursor.

10. Keep a check on the dynamic coordinate display and once it shows (100,100,90) stop moving the mouse.

11. Observe that the leg on the XY plane has its base at (100,100,0), the leg on the YZ plane has its base at (0,100,90) and the leg on the ZX plane has its base at (100,0,90) clearly indicating the position of the cursor in 3D space.

12. The projections of the legs on the XY, YZ and ZX planes also correspond to the current position of the legs.

13. Right click to plot a point at the current cursor location. The color of the point just plotted is set to yellow i.e. there is a color change from orange to yellow indicating that the point was successfully plotted.

14. Now move the cursor in the X direction to reach the location (90,100,90). Again observe that the cursor changes to X cursor during the movement along the X axis and that the legs and projections move correspondingly. Also notice that the program automatically draws a line between the current cursor location and the previous point plotted i.e. (100,100,90).

15. Once you have reached the location (90,100,90) plot a point using the right click.

16. Similarly plot the point (90,90,90).

17. For the last point move the cursor to the location (100,90,90) by a procedure similar to the one followed to plot the earlier three points. However to indicate that this is the last point of the base of the polyhedron, use a left double click, instead of a right click, to plot the point.
18. The program on sensing the double click plots the point and closes the polygon i.e. draws a line between the first and the last point of the square base.

19. It also calculates the centroid of the base polygon, plots a point using the calculated centroid coordinates and highlights the point as the currently selected point.

20. The *dynamic coordinate display*, the *legs* and the *projections* are automatically updated to correspond to the currently selected point i.e. the centroid.

21. The program also automatically changes the mode to *point translation* mode.

22. Move the current point 10 units in the positive Z direction i.e. diagonally downwards and to the right. The point will now have coordinates as (95,95,100).

23. Finally we need to position the nose in the context of the entire scene.

24. Click on the *move polyhedrons* button to enter the *polyhedron translation* mode.

   Select and move the pyramid so that its centroid reflects coordinates of (50,50,92).

25. This is the final position of the nose and is illustrated in Figure 32.
Figure 32. A screenshot of Direct3D illustrating the scene after the nose of the airplane has been added.
8.2 Modeling the front fuselage

1. The entire fuselage will be designed as an edge lift polyhedron.
2. Click on the *plot edge lift polyhedron* button. Observe that the status bar reflects the changed state i.e. ‘Edge Lift Polyhedron Plotting Mode’ and the help message also changes accordingly i.e. ‘Plot the base polygon and then move the copy of the polygon to form a polyhedron’.
3. Plot a square for the base polygon, just as we did for the nose, but using the coordinates (100,100,100), (86,100,100), (86,86,100), (100,86,100) for the vertices.
4. When you have plotted the last vertex you will observe that the program automatically closes the polygon and provides you with a copy of the currently plotted base polygon.
5. It also calculates the centroid of the copied polygon and enters *polygon translation* mode.
6. Move the polygon 20 units in the negative Z direction using the centroid as a handle i.e. till the Z coordinate of the centroid becomes 80.
7. We have just finished plotting a parallelepiped whose dimensions are 14x14x20.
8. Now similar to the nose, we have to position the front fuselage in the context of the scene.
9. Click on the *move polyhedron button* to enter the *polyhedron translation mode* and move the front fuselage till the centroid coordinates reflect (50,50,80).
10. We are basically going to align the nose and all the parts of the fuselage such that the centroids of each lie on the line whose equation is \( x = y = 50 \) i.e all the centroids have X coordinate as 50 and Y coordinate as 50 with different Z coordinates.
11. Also notice that after we have positioned the front fuselage as described above, the front-face of the front fuselage and the base of the nose (i.e. the pyramid plotted earlier) both lay on the plane whose equation is \( z = 90 \).
12. Now we need to modify the front-face of the front fuselage so as to coincide its vertices with the vertices of the base of the nose.
13. To achieve this click on the *polygon* tab and then on the *scale polygon button*. 
14. Observe the changes in the tab and the button i.e. *polygon* tab will be *raised* and the *scale polygon* button will be *pushed*. The status bar will also reflect the changed state of the system.

15. Now select the front-face of the front fuselage and scale it down along the Y axis such that the lines between its vertices overlap the lines between the vertices of the base of the nose.

16. Similarly scale-down the front fuselage front-face along the X axis.

17. The nose and the front fuselage should now look like a composite part as illustrated in Figure 33.
Figure 33. A screenshot of Direct3D illustrating the scene after the front fuselage of the airplane has been added.
8.3 Modeling the middle fuselage

1. This is the easiest part to model.
2. Select the \textit{polyhedron} tab and enter the \textit{edge lift polyhedron plotting} mode.
3. Plot a parallelepiped exactly similar to the one described for the front fuselage. The only change being, this time move the copied polygon 50 units in the negative Z direction i.e. till the Z coordinate of the centroid becomes 50.
4. After doing so we will have a parallelepiped with dimensions of 14x14x50.
5. Enter the \textit{polyhedron translation} mode and move this polyhedron till its centroid reflects coordinates of (50,50,45).
6. After doing so the back-face of the front fuselage and the front-face of the middle fuselage should overlap and the scene should look as shown in Figure 34.
Figure 34. A screenshot of Direct3D illustrating the scene after the middle fuselage of the airplane has been added.
8.4 Modeling the back fuselage

1. Enter the *edge lift polyhedron plotting* mode and model a parallelepiped similar to the front fuselage. Use the vertices (100,100,100), (90,100,100), (90,90,100) and (100,90,100) for the base polygon.

2. Move the copied polygon 20 units in the negative Z direction to achieve a polyhedron having dimensions 10x10x20.

3. Enter the *polyhedron translation mode* and move this polyhedron such that its centroid reflects the coordinates as (50,50,10).

4. After doing so the back-face of the middle fuselage and the front-face of the back fuselage should lie on the same plane whose equation is \( z = 20 \).

5. Click on the *polygon* tab and enter the *polygon scaling* mode and scale-up the front-face of the back fuselage along the X and Y axes such that its vertices coincide with the vertices of the back-face of the middle fuselage.

6. The scene should now be as shown in Figure 35.
Figure 35. A screenshot of Direct3D illustrating the scene after the back fuselage of the airplane has been added.
8.5 Modeling the left wing

1. Enter the edge lift polyhedron plotting mode and model a parallelepiped using the coordinates (100,100,100), (100,100,85), (60,100,85) and (60,100,100) for the base polygon.

2. Move the copied polygon 2 units in the negative Y direction to get a polyhedron of dimensions 10x40x2.

3. Move the polyhedron just plotted so that its centroid reflects the coordinates (77,50,62). This will align it at the left side of the middle fuselage.

4. Now click on the polygon tab and enter the polygon translation mode.

5. Select the face of the left wing which is parallel to the YZ plane and is away from the body of the plane.

6. Move this face such that its centroid reflects coordinates of (97,50,42) i.e. 20 units in the negative Z direction.

7. Now we shall use the line translation mode to fine tune the shape of the wing. Click on the line tab and enter the line translation mode.

8. We will be moving an edge of the face we have just translated.

9. The edge to be moved is the one that is parallel and nearest to the XY plane.

10. Move this line 5 units in the positive Z direction.

11. The scene should now look as illustrated in Figure 36.
Figure 36. A screenshot of Direct3D illustrating the scene after the left wing of the airplane has been added.
8.6 Modeling the right wing

1. This will be an exact mirror image of the left wing. Enter the *edge lift polyhedron plotting* mode and model a parallelepiped exactly similar to the one we modeled for the left wing.

2. Move the polyhedron just plotted so that its centroid reflects the coordinates (23,50,62). This will align it at the right side of the middle fuselage.

3. Now click on the *polygon* tab and enter the *polygon translation* mode.

4. Select the face of the right wing which is parallel to the YZ plane and is away from the body of the plane.

5. Move this face such that its centroid reflects coordinates of (3,50,42) i.e. 20 units in the negative Z direction.

6. Again we shall use the *line translation* mode to fine tune the shape of the wing. Click on the *line* tab and enter the *line translation* mode.

7. We will be moving an edge of the face we have just translated.

8. The edge to be moved is the one that is parallel and nearest to the XY plane.

9. Move this line 5 units in the positive Z direction.

10. The scene should now look as illustrated in Figure 37.
Figure 37. A screenshot of Direct3D illustrating the scene after the right wing of the airplane has been added.
8.7 Modeling the tail

1. Enter the *edge lift polyhedron plotting* mode and model a parallelepiped using the coordinates (100,100,80), (100,70,80), (100,70,100) and (100,100,90) for the base.

2. Move the copied polygon 3 units in the negative X direction to get a polyhedron of width 3.

3. Enter *polyhedron translation* mode and move the tail till its centroid reflects the coordinates as (50,70,7).

4. Two of the vertices of the tail base i.e. the face parallel and closest to the XZ plane, should now lie on an edge of the back fuselage.

5. We need to make the other two vertices of the tail base i.e. the vertices of the tail farthest from the XY plane, to lie on an edge of the back fuselage.

6. We could achieve this using *line translation* mode, but since we have demonstrated that mode earlier, let us use *point translation* mode this time.

7. Click on the point tab and then on the *Move points* button to enter *point translation* mode. Select the point (49,55,20) and move it 2 units in the positive Y direction. Repeat the procedure for the other point (52,55,20) also.

8. The scene should now reflect Figure 38.
Figure 38. A screenshot of Direct3D illustrating the scene after the tail of the airplane has been added.
8.8 Modeling the left horizontal stabilizer

1. Enter the *edge lift polyhedron plotting* mode and model a parallelepiped using the coordinates (100,100,80), (100,100,90), (80,100,100) and (80,100,80) for the base.
2. Move the copied polygon 2 units in the negative Y direction to get a polyhedron of width 2.
3. Move the polyhedron using the *polyhedron translation* mode till the centroid coordinates reflect (65,50,6).
4. Two of the vertices of the face of left horizontal stabilizer, which is closest to the back fuselage, should now lie on an edge of the back fuselage.
5. To make the other two vertices lie on an edge we need to use the *point translation mode*.
6. Click on the *point* tab and enter the *point translation* mode.
7. Select the point (55,51,20) and move it 2 units in the positive X direction. Repeat the procedure for the other point (55,49,20).
8. The scene should now reflect Figure 39.
Figure 39. A screenshot of Direct3D illustrating the scene after the left stabilizer of the airplane has been added.
8.9 Modeling the right horizontal stabilizer

1. This will be an exact mirror image of the left stabilizer. Enter the *edge lift polyhedron plotting* mode and model a parallelepiped using the coordinates (100,100,100), (100,100,80), (80,100,80) and (80,100,90) for the base.

2. Move the copied polygon 2 units in the negative Y direction to get a polyhedron of width 2.

3. Move the polyhedron using the *polyhedron translation* mode till the centroid coordinates reflect (35,50,6).

4. Two of the vertices of the face of right horizontal stabilizer, which is closest to the back fuselage, should now lie on an edge of the back fuselage.

5. To make the other two vertices lie on an edge we need to use the *point translation mode*.

6. Click on the *point* tab and enter the *point translation* mode.

7. Select the point (45,51,20) and move it 2 units in the negative X direction. Repeat the procedure for the other point (45,49,20).

8. With this step the model of the airplane is complete and should look as illustrated in Figure 40.

Through this example session we aim to help users to work with the prototype we have developed. At the same time we intend to highlight the potential of the design and portray its advantages.
Figure 40. A screenshot of Direct3D illustrating the scene after the right stabilizer has been added and the finished model of the airplane.
CHAPTER 9
CONCLUSIONS AND FUTURE RESEARCH

Though this research work done to achieve ‘direct 3D interaction using a 2D locator device’, may only scratch the surface, we believe that we have put forth a novel and important concept.

The design parameters outlined by us as well as the software prototype are significant achievements, which we hope, will generate more interest in this direction. The prototype and the example session successfully demonstrate the immense potential held by the approach.

We wish to mention that the prototype developed by us, supports only straight line primitives. In the future we want to extend it to support curved primitives as well. We would like to develop large scale specialized software, on the lines of the prototype, with support for many more functionalities.

Another major task that demands our consideration is to test the user performance when using software developed using our design. Research needs to be done to ascertain user needs as well as to find newer, better and quicker ways of accomplishing fundamental tasks, as perceived by the users.

A related area that attracts our attention and in which not much research has been done is exploring direct 3D interaction using two 2D locator devices. The idea is to use the device in the dominant hand for precision and the device in the non-dominant hand for speed.
Although our interface has been evaluated only informally, it holds considerable significance as studies have shown that informal evaluation has often proven to be very enlightening.
REFERENCES


[47]. Simulation and Visualisation Research Group, Department of Computer Science at the University of Hull. http://www2.dcs.hull.ac.uk/simmod/Technology/HMD.htm

[48]. Simulation and Visualisation Research Group, Department of Computer Science at the University of Hull. http://www2.dcs.hull.ac.uk/simmod/Technology/data_gloves.htm


APPENDICES
APPENDIX A
MAPPING ALGORITHM

while left mouse button is pushed and held down
{
    record mouse position
    if we have we recorded three or more positions
        calculate the slope of the line from the current position to the last position
        and store as a

        calculate the slope of the line from the current position to the second last
        position and store as b

        calculate the slope of the line from the last position to the second last
        position and store as c

    based on the values of a,b,c determine
        if movement is horizontal
            if movement is towards right
                increase X coordinates
            else if movement is towards left
                decrease X coordinates
        else if movement is vertical
            if movement is upwards
                increase Y coordinates
            else if movement is downwards
                decrease Y coordinates
        else if movement is diagonal
            if movement is downwards and towards the left
                increase Z coordinates
            else if movement is upwards and towards the right
                decrease Z coordinates
    store the last position as the second last position recorded
    store the current position as the last position recorded
}