

3D Input Devices for the GRAPECluster Project
Independent Study Report
By Andrew Bak

Sponsored by:
Hans-Peter Bischof
Department of Computer Science

Table of contents

- I. Introduction**
- II. The Data Glove**
- III. The Flock of Birds**
- IV. The Dual Analog Joystick**
- V. Mapping the input data to Camera position**
- VI. Conclusion**
- VII. Appendix I: Device setup**
 - A. Joystick setup**
 - B. 5DT Data Glove Setup**
 - C. Flock of Birds Setup**
- VIII. Appendix II: List of input device files.**
- IX. Appendix III: General Issues with Setup.**

I. Introduction

One of the important issues of the GRAPEcluster project is being able to maneuver around in space. A mouse provides a means of moving around in 2 dimensions; this still leaves a dimension unused. In this paper I will discuss the implementation of a data glove, the Flock of Birds 3D positioning device, and a dual analog joystick controller in an effort to provide a means to maneuver 3D space easily. In this independent study I have implemented these devices to talk in the visualization tool's language and send commands to move the camera to desired location based on user input. In the following sections I will talk about what I did over the course of the quarter, what I've learned and the outcomes of my efforts.

Most of the meetings of my group, have been documented on the website (www.cs.rit.edu/~apb3500/Grape). This website also contains links to the people involved in the project, and a link to the grape cluster web site.

II. The Data Glove

The first device I implemented was the 5DT data glove. There were many obstacles to overcome before successfully implementing it. The first this obstacle was that there were drivers needed. The Muppets project already used the data glove, and it was written in java. After finding the proper code in Muppets, the following files were taken and moved into the GRAPE root directory:

```
Spiegel/Muppets.jar  
Spiegle/MuppetGlove.dll
```

Besides these files, the following camera controlling plug-in was written:

Spiegel/viewcontrol/mapview/plugin/DataGlove/DataGlove. This class is initialized from *MapViewUI* when the plug-ins are loaded. When running this class spawns a polling thread. This thread polls glove data every tenth of a second and move the camera according to glove positions.

The main issue with doing this is that the data glove isn't a true 3d device. It has 5 analog finger position readouts, and two axes of leaning. This doesn't lend itself to the "Minority Report" type interface we were thinking of. If the data glove was chosen as the primary input in its current state I believe it would have a longer learning curve then other alternatives.

III. The Flock of Birds (3d position device)

The next input device implemented was the Flock of birds, or FOB. This is a 3D position device that returns the Cartesian coordinates of the position of a sensing device in relation to a main sensor cube. In order to tie this device into the project I had to use code from the Muppets project. The following files were used from Muppets for the FOB:

Spiegel/Muppets.jar
Spiegel/MuppetsFOB.dll
Spiegel/Bird.dll

FOB was easy for the first implementation since we mapped absolute coordinates from the raw input into the position of the camera. Peter Weisberg, another teammate, wrote code to make the flock of birds work in relative to where the device was located. This way the user could pause the FOB, put the sensing device down. Then pick it up and resume the FOB and move the camera relative to where it is located. Before this the sensing device would have to be brought back the same place before resuming. The biggest issue with relative movement is it's much easier to get lost in space. Regardless of this, the Flock of Birds holds the greatest potential and is intuitive to use. More on how to run the Flock of Birds is located in Appendix I.

IV. The Dual Analog Joystick.

One of my teammates at the beginning of the project, Mark Bryant, did some nice work in getting the aspect of the project going. He located *JInput* package, which mapped input devices. He also wrote a simple test program. The drawbacks were that the files only ran on PC. There are versions of *JInput* out there for other operating systems, but it requires more work to implement them. The only tricky thing about this input device is a joystick moves the camera relative to where it is. The only way we knew to quickly get the current position of the camera was by getting the value of the position from the GUI. Unfortunately this is coupling the input device to the GUI and when the new version of the GUI came to be the code was moved into plug-in classes. This meant all my input code had to be re-programmed a few times over the course of the quarter.

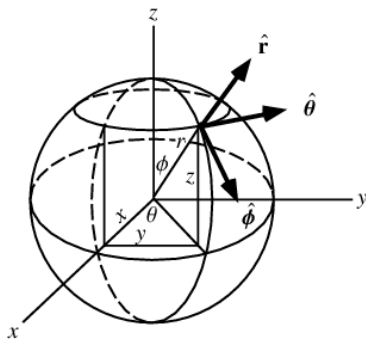
The device I used was the *Thrustmaster Firestorm Dual Power*. This is a dual analog controller with 12 buttons and a POV hat. I used this controller because it was the one I had at home. It was purchased at CompUSA. Currently the axes that are tied to the camera movements are hard coded to axes 14,15,16,17 in the *Controller AXIS* object array. These are the two analog joysticks on the controller. A mapping mechanism in the GUI would be a good next stage to support a lot more controllers.

The main issue with running this plug-in was the permission in the controller directory need to be set to let Java access it. Another problem is this code currently only runs on Windows machines. The GUI only activates the joystick code when the user requests it. This is a quick work around until other systems are supported. The polling thread polls the device at a fixed rate. This might not optimal and should be user definable in the future. More on how to run this is located in Appendix I.

V. Mapping the input data to Camera position

I used two different methods for mapping the input controls into camera positions. In the case of the Flock of birds, raw rectangular position is received from the device so they are mapped directly to rectangular position of the camera. This worked well for this type of input device.

The other way of mapping input into camera position was using spherical coordinates. The center of the sphere is the center of the cube in the universe. Two axes control θ, ϕ , while the third axis controls the distance from the center, or r .



This seemed like good choice for moving around since; when a camera is moved to a new position it always faces the middle of the universe. So moving with respect to the middle seemed to make sense. I used the standard equations to convert Spherical Coordinates into Cartesian points and moved camera to the specified points.

(Picture from www.mathworld.com)

VI. Conclusion:

Overall I am pleased that I was able to interface so many devices in such a short period without any prior experience in this field. The next stage of this project will involve working with the users of the program to determine how they like these types of devices and determine which one to pursue further. Though each device has been interfaced and runs, these are quick implementations for rapid prototyping. A good example of this is the joystick setup. A more complete interface would allow the user to assign axes to each camera axis and buttons to special tasks. Another challenge is being able to port these devices to non-Windows machines. There is clearly much more work that can be done in this area.

I have learned how to do many things throughout the quarter. Using *JInput*, and working with windows native dll's mapped to Java are a few big things. I also learned a great deal about the GRAPE code. Besides technical skills, I managed a team of two volunteers which was an interesting experience. I've learned it's hard to keep people motivated when other classes give them *real* work. One of my group members had to leave as soon as the third week of the quarter due to workload. It is much more difficult to assign tasks if you don't know how soon they can be done by. Working with the GUI team I noticed that their volunteers were more involved and kept each other going. Perhaps this is due to the much larger initial group size. When leading volunteers, more is better. Overall this has been a good learning experience and I accomplished what I set out to do.

VII. Appendix I: Device setup

A. Joystick setup



1. Choose to scan for Joystick devices

2. *Thrustmaster Firestorm Dual Power*



2. Joystick GUI after Scanning.

To control the visualization tool using the dual analog joystick, you must first be running under Microsoft Windows with *DirectX* installed. Scan for devices is the button, as pictured above. If a permission error occurs you must change the permission on the *DirectX* files inside *spiegel/controller* so that java can access them. Once devices are scanned, a combo box appears with all controllers that *JInput* supports. This includes mouse, keyboard, and other such devices. To use the game pad, select it from the list, and click start. Before continuing to the next step you must ensure the display is initialized and a camera is created. This is done by loading data file from *File Control* tab, and clicking *init* from the *Display Control* tab. To move in space, hold button 8(right hand trigger, the lower one), and move the joysticks. The right joysticks Y-axis moves the camera up and down. The right joystick X axis moves the camera left to right, around the center. The distance from the center is controlled by the left joystick, X-axis. Moving the axes slightly creates nice slow movements which gives a good sense of star positions.

B. 5DT Data Glove Setup.



In order to use the 5DT Data Glove you must first type in the COM port that the glove is connected to. Then hit the start button. Because the data glove only works in windows, there are unexpected errors if it is started on a non-windows system. Another issue is that if the wrong com port is specified, or the glove isn't plugged in, then the program crashes. All hardware should be checked to ensure the glove is plugged into the proper com port and it is plugged into a power supply.

After hitting start the glove using spherical coordinates, or absolute coordinates. To use spherical coordinates bend your thumb in. In this mode the distance from the center in this mode remains constant. The rotation around the center is controlled by the gloves y-axis (move your hand in a slow throwing motion to see this movement). The movement of the index finger is tied to the ϕ angle.

The other way to move the camera using the data glove is through sliding along each axis. With you thumb straight, bend down your index finger and move the glove up and down. This controls the camera's X-axis. The same motion with the middle finger controls the Y-axis. The ring finger is tied to the Z-axis. If multiple fingers are bent down at a time then you slide on multiple axes. To turn off the Glove press *stop*.

C. Flock of Birds setup



The Flock of Birds can only be controlled in a windows environment. To ensure proper control of the Flock of Birds be sure the FOB is plugged in turned on, and not in stand-by mode. Specify which com port it is plugged into in the text field as pictured above. Click start to initialize the FOB. The wait button pauses and resumes the polling thread. When the device is initialized it is in wait mode. Click the wait button to start polling data. At this point you should be able to move the device around. Be aware of where the sensing device is when you come out of wait mode. Clicking wait again stops the poling thread and saves the location of the camera. So when the polling thread is resumed again it doesn't jump the camera to the sensing device's absolute location, instead it moves it relative to where the camera was last. The stop button closes the connection to the FOB properly.

VIII. Appendix II: List of input device files:

General/overhead

Spiegel/viewcontrol/mapview/plugins/Makefile
 Spiegel/viewcontrol/mapview/plugins/Inputs3d/*
 Spiegel/Makefile.opts

Flock of Birds*

Spiegel/viewcontrol/mapview/plugins/FOB/*
 Spiegel/MuppetsFOB.dll
 Spiegel/Bird.dll

Joystick

Spiegel/viewcontrol/mapview/plugins/Joystick3d/*
 Spiegel/controller/*

Data Glove*

Spiegel/viewcontrol/mapview/plugins/DataGlove/*
 Spiegel/MuppetGlove.dll

* Data Glove and Flock of Birds:

Spiegel/Muppets.jar

IX. Appendix III: General Issues with Setup:

Upon checking the *grapecluster* code out of the CVS repository a couple things need to be done. First the Makefile.opts needs to be setup for the windows machine. This means you must use the *classpath* with the semi-colons instead of colons. A windows host setup is already in the Makefile.opts, all that is needed is to specify the ROOT directory. Also before compiling erase the empty *Spiegel* directory in the *Spiegel* directory. This is an empty directory which prevents *spiegel.java* from building when “make all” is run. Also be sure to make the controller DirectX files are accessible using:

```
Spiegel/controller> chmod 755 *
```

Build using:

```
Spiegel> make all
```

Run using:

```
Spiegel> make runSD
```