Implementation of Gesture Recognition in the Immersive Visualization Environment

Luis Catacora
Montgomery Blair High School
Silver Spring, Maryland

Under the direction of
Judith Terrill and
Terence Griffin
National Institute of Standards and Technology
Gaithersburg, Maryland
Abstract

The purpose of this research project was to study and implement gesture recognition for an immersive environment found at NIST. The Immersive Visualization Environment (IVE) is a virtual room that allows a user to manipulate and visualize 3D objects with a wand controller, and a headset respectively. Data analyzed in the Reconfigurable Automatic Virtual Environment (RAVE) could originate from a range of data sets; such data could be small enough to show a simple image or be large and complex enough to demonstrate complex interactions. The RAVE is the virtual room in the Information and Technology Lab at the National Institute of Technology that incorporates the immersive environment. Although the software found at NIST already included flexible and intuitive navigation methods, their use of pointer gestures has been limited. In order to establish a better interaction between the immersive environment and its user, game technology was researched to develop efficient gesture recognition for the RAVE. The procedure used consisted of a basic four-step plan for each gesture to be implemented in the RAVE. The results gathered dealt with the position, velocity, and acceleration of the RAVE wand while the user performed a specific gesture. In the end, the completed program was able to successfully identify a total of 15 gestures. The implemented methods were written in specific command scripts that could be easily replaced with any executable method, making the actions completely user definable.
1 Introduction

The IVE (Immersive Visualization Environment) is a fascinating three-dimensional environment. It is a virtual room that enables scientific visualizations of large amounts of data for further analysis and measurement [1], [3]. Data analyzed in the RAVE (Reconfigurable Automatic Virtual Environment) could originate from large parallel computations, biological, and chemical data sets of interest. The RAVE is the virtual room in the Information and Technology Lab at the National Institute of Technology that incorporates the immersive environment. The RAVE has various commands and capabilities for the user, such as onscreen menus and displays [2]. But the RAVE lacks the user-friendly connection that the more modern technologies possess in terms of their capabilities in gesture recognition. Thus this virtual room requires the enactment of gesture recognition in order to increase efficiency and compatibility between the user and the technology.

As visualizations become increasingly complex, better methods of human interaction are needed; consequently, detailed and accurate gestures should be investigated. The current technologies such as Natal and Kinect are modern game technologies for the Xbox that have abandoned the traditional use of a built controller, and have incorporated the human itself as the controller. The Wii console also provides a source for investigating common gestures used with a wireless Wii remote. This project focuses on the development of gesture recognition (i.e. flicks, stabs, lines, etc.) and the process that was followed to accomplish it.

Game technology was chosen for research because they contain the most up-to-date technology containing gesture recognition. These predefined recognized gaming gestures are surprisingly very useful in the field that the RAVE uses. Such already predefined gestures could be used in not only games, but also in data manipulation and visualization. Thus, by
implementing useful gestures into the IVE, the technology will benefit by containing an efficient method to visualize and manipulate visualizations.

For the main part of the research project, the procedures consisted of 2 major steps; the creation and research of the gestures to be implemented, and how well the gesture is recognized by the RAVE. These steps are essential for the optimal efficiency of gesture recognition within the RAVE. This entire research project was extremely fun to work with, particularly because it contained a chance for the user to work within 3D space.

2 Materials & Methods

1. Gesture Steps:

Steps taken for each gesture implemented in the RAVE ultimately followed a specific design format in designing and development. These gestures were created after researching various other consoles, such as the Wii and the Xbox Kinect, that also use a need of similar gestures.

2.1 Defining the gesture:

The first step was to specify what would happen once the gesture was performed, and to define what the gesture will recognize or what it would look for in the calculated variables. The coded arrays were designed to hold values for a detection window of approximately 2.5 seconds, give or take due to the speed of the processor. The main arrays created and used for gesture recognition were: x, y, z, yaw, pitch, and roll. Additional variables were then created to store the changes inside the arrays within the detection window. Specific changes in position, speed, and acceleration were used to define a gesture in the gesture-recognition program.
2.2 Collection of data:

After performing the gesture while in the RAVE a multiple number of times, position information of the wand was recorded. Parameters such as position, speed, and acceleration were written to the file “zout.txt” for further investigation and analysis for the appropriate algorithm that the computer would use for gesture recognition.

2.3 Programming implementation of gesture:

After analyzing the data for each gesture, algorithms were then created that would meet the basic key points that distinguished each gesture. The main recognition code was written in C++, the other programs displayed or conducted the appropriate action of the gesture (ex. flick right would spin object counterclockwise). Then the programs were modified so that gesture number would be written to a shared memory file. The resulting program was thoroughly commented through in-line documentation to aid in future modifications of the source code.

```
//checkLineLeft()//checks Line Left check
{cout<<"Gesture Recognized: Line Left \n"endl;myfile<<"Left";} //ends check Line Right

//checkLineRight()//checks Line Right check
{cout<<"Gesture Recognized: Line Right \n"endl;myfile<<"Right";} //ends check Line Right

//checkLineUp()//checks Line Up check
{cout<<"Gesture Recognized: Line Up \n"endl;myfile<<"UP";} //ends check Line Right

//checkLineDown()//checks Line Down check
{cout<<"Gesture Recognized: Line Down \n"endl;myfile<<"Down";} //ends check Line Right

//checkBox()//checks Box check
{cout<<"Gesture Recognized: Box \n"endl;myfile<<"Box";} //ends box check

//checkstab()//checks stab check
{cout<<"Gesture Recognized: Stab \n"endl;myfile<<"Stab";} //ends stab check

//checkFlickUp()//checks FlickUp check
{cout<<"Gesture Recognized: Flick Up \n"endl;myfile<<"FlickUp";} //ends Flick Up check

//checkFlickDown()//checks FlickDown check
{cout<<"Gesture Recognized: Flick Down \n"endl;myfile<<"FlickDown";} //ends Flick Up check

//checkFlickLeft()//checks FlickLeft check
{cout<<"Gesture Recognized: Flick Left \n"endl;myfile<<"FlickLeft";} //ends Flick Up check

//checkFlickRight()//checks FlickRight check
{cout<<"Gesture Recognized: Flick Right \n"endl;myfile<<"FlickRight";} //ends Flick Up check

//ends line gestures below.

//checkDiagonalLineLeftRight()
{cout<<"Gesture Recognized: Diagonal Line LEFT \n"endl;myfile<<"DiagonalLineL";} //ends Diagonal Line Left Right

//checkDiagonalLineUpRight()
{cout<<"Gesture Recognized: Diagonal Line Up Right \n"endl;myfile<<"DiagonalLineUR";} //ends Diagonal Line Up Right

//checkDiagonalLineDownRight()
{cout<<"Gesture Recognized: Diagonal Line Down Right \n"endl;myfile<<"DiagonalLineDR";} //ends Diagonal Line Down Right

//checkDiagonalLineDownLeft()
{cout<<"Gesture Recognized: Diagonal Line Down Left \n"endl;myfile<<"DiagonalLineDL";} //ends Diagonal Line Down Left

//checkTriangle()
{cout<<"Gesture Recognized: Triangle \n"endl;myfile<<"Triangle";} //ends check Triangle
```

Figure 2: Sample images used in RAVE to display when gesture was recognized.

Figure 1: Sample C++ Code with comments.
2.4 Testing of Gesture:

In order to run and test my new files, the C++ code had to be compiled and executed to see if the correct gesture output was performed. In order to test my code in the RAVE room, students needed to be accompanied by my mentor. Thus if it was possible, gesture tests were performed on both the Desktop version and the RAVE version. Some gestures that required movement in 3 dimensions could not be tested by the desktop version of the same 3D environment. The program initially displayed messages to the terminal from the original source code. Later on, a shared memory file was created containing the recognized gesture number in file “gestureNum” for external use and compatibility of the program. An external shell script was developed to display recognition images to the RAVE user since the user cannot see the computer screen when operating the RAVE. The process worked, but the display image covered a great deal of the 3D object. Thus a small arrow was created to highlight recognized gestures by having it in 3D space, along with the user’s object of interest.

3 Results and Analysis

The resulting program [Figure 3] was able to successfully recognize basic gestures from human movement. Raw data was stored into txt files [Figure 4] and reviewed for analysis and correction of gesture recognition. The various algorithms looked for changes in the data similar to those patterns shown [Figure 5]. The progress of displaying gesture detection progressed from onscreen terminal messages, to pop-up images to 3D arrows. Images displayed below [Figure 6].
Figure 3: start of the main section of the program that uses algorithms to recognize gestures. Used specific variables to determine changes in position for lines, and acceleration for flicks.

```
7111X:0.0192482 Y: -0.141275 Z: -0.123402 Y:102 PI: -9 RO: -5
71217X:0.1351245 Z: -0.1351245 Y:102 PI: -9 RO: -5
7131X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7141X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7151X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7161X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7171X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7181X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7191X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7201X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7211X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7221X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7231X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7241X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7251X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7261X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7271X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
7281X: -0.0254502 Y: -0.0562275 Z: -0.104200 Y:42 PI:3 RO:7
```

Figure 4: Sample raw data stored into txt file labeled “zout.txt” shows the main variables used. The wand’s position in the X, Y, and Z environment are displayed, along with the three Euler angles of the wand: yaw, pitch, and roll.
Figure 5: Patterns in the position, velocity, and acceleration (in order) of a gesture that were looked for in the coded algorithms. Examples shown represent the components for a line gesture.

Figure 6: The Desktop version of the immersive environment displaying the images that signals a recognized gesture.
4 Discussion/Conclusion

The basics of gesture recognition were added so that the RAVE users would be able to execute simple commands that would interact with their 3D object with just a flick of the wrist. The major importance of this project is the start of modernizing this technology to the 21st century and developing a strong preliminary interaction between the user and the technology. The only data collected was used to aid the gesture recognizing algorithms in determining gesture patterns, and to verify if the correct gesture was recognized upon completion. Gesture recognition will improve current analysis of data, measurements of data, and interactions between the user and the data within the IVE.

Current work in gesture recognition for the IVE was researched and compared to the other established game technology. The main game console of interest was the Wii because of the similar dependency of a wand controller in 3D space. Future works will also include a better representation of “relative” gestures, gesture tracing, and curve fitting for even more complex gestures. All future works will lead to quicker and effective interactions between the users and the IVE. This upgrade in gesture recognition could lead to user ability to redefine gestures in terms of how they are recognized and the action that follows the gesture.

References