TOWARDS AFFORDABLE GESTURE BASED INTERFACES
An Exploration with Wii Remotes

Amy Ciavolino
CSEE, CoEIT, University of Maryland - Baltimore County
aci1@umbc.edu

Camille Marvin
Department of Computer Science, Harvey Mudd College
cmarvin@hmc.edu

Jason Creighton, James Coddington, Hans-Peter Bischof, Reynold Bailey
Rochester Institute of Technology
jtc6189@rit.edu, jxc6857@rit.edu, hpb@cs.rit.edu, rjb@cs.rit.edu

Keywords: Natural User Interface, Gesture Control, Wii Remote, User Interaction.

Abstract: The traditional keyboard and mouse computer interface is well suited for 2D applications such as document editing, but as 3D environments become more prevalent there arises a need for new methods of user input. An immersive 3D interface is preferable, but is often cost prohibitive. This paper presents the design and implementation of a Natural User Interface (NUI), named NuWii. NuWii was designed to be used with the Spiegel visualization framework for astrophysical data. The interface makes use of Nintendo Wii Remotes as infrared tracking cameras to detect 3D gestures made by the user. These gestures are interpreted and used to control the viewing camera's parameters throughout the course of a visualization. Our project provides a new, more intuitive way of manipulating the camera in the Spiegel visualization system. Furthermore, NuWii provides an expandable base that could be used for gesture control in many other applications.

1 INTRODUCTION

Spiegel is a visualization system that was developed to process and visualize large multidimensional data from simulations of galactic events such as black-hole mergers, event horizons, and gravity waves (Bischof, 2010; Bischof et al., 2006). Previous methods of user input for working with 3D models in the Spiegel visualization framework were not intuitive. Ideally, we want astrophysicists and other users to be able to view and interact with the simulations in a simple and natural way. To accomplish this, we created NuWii, a system that captures the user's motions in 3D and uses them to control the camera position in Spiegel. NuWii is designed to be easily expandable to other applications that require 3D gesture input. Our implementation uses a two level gesture hierarchy to accommodate custom gesture input. We designed NuWii to be portable, easy to set up, and affordable. NuWii uses two Nintendo Wii Remotes to capture the gesture input from the user. We have also developed an algorithm to extract the 3D point from the two images acquired by the Wii Remotes instead of using proprietary software in order to keep the cost down for anyone expanding upon our project.

2 PRIOR WORK IN THE FIELD

A significant amount of work has been done in the area of human-computer interaction, 3D point recognition, and natural user interfaces. A 3D hand recognition system was presented in 2009 (Wang and Popović, 2009) and efforts are under way to develop a 3D user interface similar to the one seen in the movie Minority Report (Underkoffler, 2010). There has even been some previous experimentation with 3D interaction in Spiegel (Bak, 2004). Several projects have explored the use of Wii Remotes for stereo-vision (Dehling, 2008), motion capture (Wang and Huang, 2008), and finger tracking (Lee, 2008). Most of the projects done with Wii Remotes use MATLAB for camera calibration and resolving points in 3D space. However, using MATLAB limits the audience, affordability, and flexibility of the software. Some previous papers tracked multiple points in 3D, but did so only in the context of head tracking (Cuypers et al., 2009). Head tracking assumes a limited range of motion of the points, making it easier to distinguish them. Other projects involved only minimal error checking (Hay et al., 2008). In contrast, NuWii is able to track two points under more general conditions.

3 HARDWARE

We elected to use Nintendo Wii Remote for its affordability, availability, and specialized camera hardware. In NuWii, the cameras on the front of the Wii Remotes pick up infrared light reflected off the user's finger tips. Two arrays of infrared LEDs supply the infrared light and finger slips, made from reflective tape, reflect it back to the cameras. We also machined a wooden board to hold the Wii Remotes in place. The notches in our board hold the Wii Remotes two feet apart and angled 22.5 degrees inward, but any reasonable angle with overlapping fields of view could be used. All the hardware used in NuWii is available to the average consumer. A photograph of the NuWii setup is shown in Figure 1.

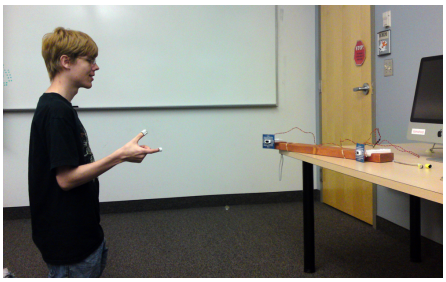


Figure 1: Photograph the NuWii setup. The user's gestures are being tracked by the two infrared cameras.

3.1 The Wii Remote

The Wii Remote has a built-in 128x96 monochrome camera. By using sub-pixel analysis, it can track up to four separate points of infrared light and return their coordinates in a 1024x768 range (Wii Brew, 2010). The field of view of the camera is 33.75 degrees vertical and 45 degrees horizontal. These values can be derived from the resolution of the camera (Lee, 2008).

The Wii Remotes use the Bluetooth HID protocol to communicate with their host. However, they do not use the standard data types and are meant only to communicate with a Wii gaming system. This lack of complete compliance makes connecting the Wii Remotes to a computer somewhat complicated, but once the connection has been made we have found it to be stable. A great number of libraries in a variety of high level languages have been written to facilitate easier communication with the Wii Remotes and their peripherals (Wii Brew, 2010). We chose the *motej* library because it is open source, allowing us a greater understanding of how it communicates with the Wii Remotes. Additionally, both Spiegel and *motej* are written in Java, making integration straightforward.

3.2 Infrared LED Arrays

Each Wii Remote is surrounded with infrared LEDs that supply light to be reflected back to the cameras. For each Wii Remote, we made an array of 48 LEDs. The wavelength of the LEDs is 940 nm which is optimal for the Wii Remotes' cameras (Wii Brew, 2010). The LEDs are arranged in eight groups wired in parallel. Each group contains 6 LEDs and a 75 Ohm resistor in series. The LED arrays are powered by a 12V, 1 amp power supply.

3.3 Finger Slips

We created two finger slips out of reflective tape that easily slide on and off the user's index finger and thumb (Figure 2). The finger slips completely cover the tips of the user's fingers to ensure that the light from the LEDs is reflected back to the cameras from any angle. We built the finger slips out of 3M 3000X Very High Gain Reflective Tape. In order to hold the reflective tape together we used small pieces of Nathan 3M Reflective tape. We also built finger slips completely out of Nathan 3M Reflective tape. These finger slips seemed to work just as well and were significantly less expensive than their high gain counterparts. Nathan 3M Reflective tape is also more flexible which made it easier to form the top of the finger slips.

Figure 2: Finger slips made of reflective tape. Nathan 3M tape (index finger) and Very High Gain tape (thumb).

4 STEREO-VISION

To locate a point in 3D, data from two or more Wii Remotes must be combined. NuWii uses two Wii Remotes. A third Wii Remote could be added to improve accuracy, but it would require a more advanced setup and increase the overall cost of the system. Our algorithm uses trigonometry to find the location in 3D space, given the angle of the Wii Remotes and their distance apart.

4.1 Algorithm

The algorithm takes input from two Wii Remotes. The output from the Wii Remotes is in the form of (X, Y) points in the range (0, 0) to (768, 1024). The algorithm assumes that the Wii Remotes are in the same y and z plane and are placed at known angles in the x plane. Deviating from the specified orientation will produce errors in the final result. It is possible to get usable output from the algorithm without knowing the distance between the Wii Remotes. However, if the distance is known then the final output will have the same unit as the distance. Steps 1-3 are repeated for the input from both the left and right cameras.

Step 1: The range of the camera input is altered to go from (-512, -384) to (512, 384). This is done by subtracting half the maximum values for the respective axis (Figure 3).

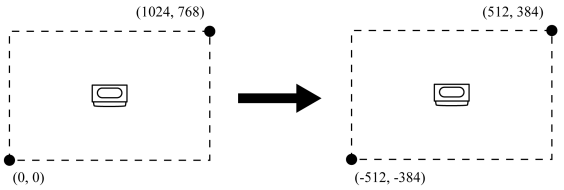


Figure 3: Transforming the setup coordinates.

Step 2: Using the altered input, the points are normalized by dividing the x value by 512, and the y value by 384. The x result is then multiplied by half the horizontal field of view of the camera and the y result is multiplied by half the vertical field of view in order to find the angles shown in Figures 4 and 5.

$$\theta = \frac{X}{512} \times \frac{45}{2} \quad (1)$$

$$\phi = \frac{Y}{384} \times \frac{33.75}{2} \quad (2)$$

Step 3: Trigonometry is used to find a normalized point one unit away from the cameras.

$$X' = tan(\theta) \quad (3)$$

$$Y' = tan(\phi) \quad (4)$$

$$Z' = 1 \quad (5)$$

Step 4: After the normalized points are found, they are rotated by α, the angle between the Wii Remotes and the y-z plane shown in Figure 6. Rotating the points is not required if the Wii Remotes are parallel. The following equations are used to rotate the points:

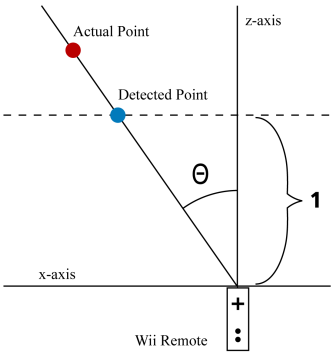


Figure 4: Top View

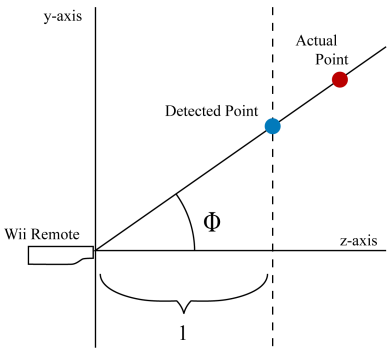


Figure 5: Side View

$$X' = X'cos(\alpha) + sin(\alpha) \quad (6)$$

$$Y' = Y' \quad (7)$$

$$Z' = -1 * X'sin(\alpha) + cos(\alpha) \quad (8)$$

Step 5: The distance between the Wii Remotes is added to the x value of the points from the right Wii Remote (the right Wii Remote will be on the left when looking into the cameras of the remotes). The units of the final output will be the same as the units used to measure this distance. If the distance between the Wii Remotes is unknown then 1 should be added instead.

$$X'_r = X'_r + d \quad (9)$$

This step moves the points read by the right camera into the correct coordinate system in relation to the left camera.

Rays originating from the Wii Remotes passing through these new left and right calculated points are created. Figure 6 shows that the left camera is at the origin (0,0,0) and the right camera is at (d,0,0) where

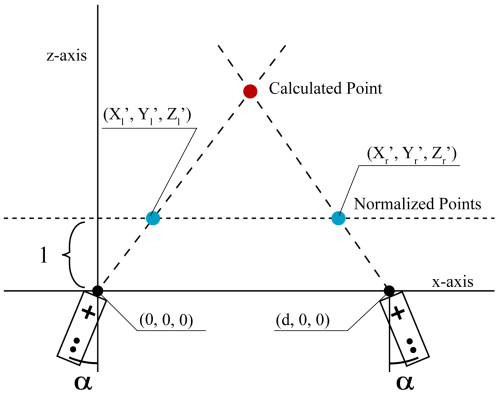


Figure 6: Finding the 3D point.

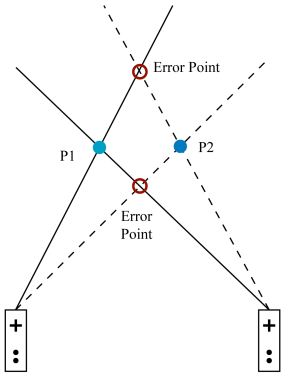


Figure 7: Lines intersecting incorrectly.

d is the distance between the Wii Remotes. A ray collision algorithm is used to find the location along the rays where they are closest to colliding. This location is the approximate position of the point. The exact location of the collision can not be calculated because the rays will not collide perfectly due to error in the data captured by the Wii Remotes.

4.2 Multiple Points

The algorithm described above senses one point. However, reading two points introduces ambiguity when both points lie in the same y plane. The problem occurs because the Wii Remotes transmit only the coordinates of the infrared points. This means that there are no surrounding visual aids to help distinguish between the two points. Because of this limited information, it can be impossible to tell which point from the second Wii Remote corresponds to the point seen by the first Wii Remote in certain situations.

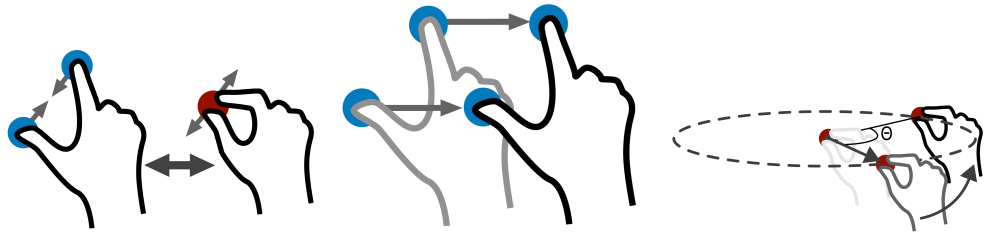
When the points are not in the same y plane the ray collision error can be used to match the correct points. To distinguish between these points both possibilities are tested and the pair with the smaller error is used. This method does not work when the points are in the same y plane because both pairs appear to be valid, as shown in Figure 7. We leverage the knowledge that the leftmost point on the first camera should be paired with the leftmost point on the second camera in most situations. When we cannot distinguish the points using the above methods, we use the fact that the Wii Remotes return the points that they detect in the same order throughout a session, and assume that they have not changed. This can be a problem if a camera stops sensing points and then flips the order it senses them in. In practice however, these issues with multiple points remain largely unnoticed because the points are constantly being updated.

5 GESTURES

Our implementation of gesture recognition is designed to be easily expandable. This is accomplished by using two levels of gestures: basic gestures and composite gestures. Basic gestures are simple motions and movements of the user's hands that are detected within our stereo-vision algorithm. We created a gesture interface in Java that can be implemented by any class that needs to detect these basic gestures. The second level of gestures is composite gestures. Composite gestures are combinations of basic gestures that can be used for more complex input. These higher-level gestures are implemented by the class that uses our interface and are more application specific than basic gestures.

5.1 Basic Gestures

There are two categories of basic gestures implemented at this time: pinch and swipe. A pinch gesture (Figure 8(a)) is activated when the two points seen by the cameras move close enough together that they appear to be one point. There is also an unpinch gesture that is activated when a pinched point separates back into two points. An unpinch gesture can only be detected after a pinch has taken place, which keeps the cameras from falsely identifying two unrelated points as an unpinch gesture. The other basic gesture, swipe, is activated when the points move a set distance in any dimension (Figure 8(b)). The movement of either one or two points is tracked depending on how many the cameras see. Tracking any number of points allows a swipe to be detected regardless of the pinching state. Swipes can be detected in both positive and negative directions in all three dimensions for a total of six different swipes that can be detected and used for composite gestures.



(a) Pinch (b) Swipe (c) Rotate

Figure 8: Various Gestures.

5.2 Composite Gestures

By combining the basic gestures discussed above and the current position of the points, application specific gestures can be created. These composite gestures can be very simple, using just one basic gesture to activate some sort of onscreen movement, or as complicated as necessary, making use of several gestures in sequence. A rather conventional application would be to map the movement of the points to cursor position and the pinch/unpinch to a click. The gestures implemented for Spiegel, described in detail in the following sections, provide an example of more complex composite gestures.

6 NUWII AND SPIEGEL

The Spiegel framework was developed to visualize large multidimensional astrophysical data. It is designed according to the UNIX paradigm of pipes and small utilities that do one thing and do it well (Bischof et al., 2006). These small utilities are called ”functions” by Spiegel developers. Once the Spiegel GUI is loaded, the user chooses the functions that they want to use from the menu and the functions appear on screen as boxes with incoming and outgoing arrows. For example, in order to display a set of simulation data in 3D, the user would import five boxes - one to import the data file from the file system, one that extracts the stars from the data, one that converts the star data into a format that Java3D can understand, one that determines camera parameters, and finally a display window for the image. Our team wrote two new functions for Spiegel. One of these, named WiimoteControl, connects the Wii Remotes to the computer and interprets the data read from the Wii Remotes as camera coordinates. The other, 3DPointDisplay, shows the points from the Wii Remotes in 3D space and is used for debugging the system if problems arise.

6.1 Camera Control in Spiegel

Within Spiegel, three composite gestures were implemented to control the camera position. These gestures are used to enter different camera movement modes. Each gesture starts with a pinch followed by a swipe. The pinch sets a center point to be used as a reference for movement in each camera movement mode. The direction of the swipe determines which mode is activated. The user can easily exit each movement mode by unpinching their fingers. Using these gestures the user can view the simulation from any angle.

The best way for the user to remain oriented when viewing a simulation in Spiegel is to have the camera constantly pointing towards the origin. The simplest way to achieve this behavior is to use a spherical coordinate system for the camera position. Changes in the inclination and azimuth angles correspond to vertical and horizontal rotation respectively and changing the radial distance acts as a zooming function (see Figure 9). In NuWii, each camera movement mode changes the value of one of the spherical coordinates.

The three camera movement modes are horizontal rotation, vertical rotation, and zoom. Horizontal rotation is activated by pinching and swiping to the right. To rotate the model, the user rotates the pinched point around the reference point to move the camera around the model (see Figure 8(c)). Swiping backwards (towards the user) after pinching triggers the vertical rotation mode. After swiping, the user can move his hand up or down to rotate the camera vertically around the model. Horizontally, the camera can be rotated around the model indefinitely, but vertical rotation is capped at positive and negative 90 degrees. This restriction keeps the user from moving the camera over the model which would make the view upside-down. The zoom control is activated by pinching and then swiping down. Once in the zoom mode, the distance between the center point and the current point is used to scale the zoom speed. If the current point is at the center point set by the pinch, then the

camera will be stationary. When the current point is in front of the center point, then the camera will zoom in. Similarly, if the current point is behind the center point, then the camera will zoom out.

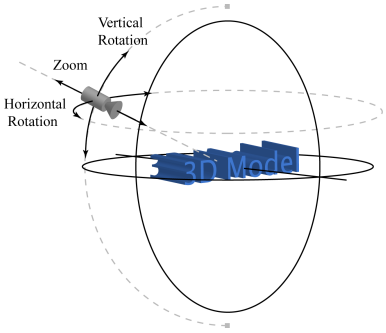


Figure 9: Camera in relation to the scene

7 CONCLUSIONS AND FUTURE WORK

There are several possible directions for future work. Currently, our software recognizes two basic gestures and three composite gestures, which are identified using two points of IR input. Future contributors could design and implement more gestures, which would expand control over Spiegel significantly. The expansion of the gesture library could be aided by tracking more than two points at a time. This would require more advanced trigonometry, additional Wii Remotes, and/or different wavelengths of IR light. Currently the Wii Remote cameras must be placed in a close approximation to the orientation specified by the user in software in order for the gesture recognition code to work correctly. A camera calibration method could be written, allowing the Wii Remotes to be placed at any angle and any distance apart. Research is also needed in order to quantify the differences and advantages to using a 3D gesture system over a traditional system.

In developing our system, we experimented with different gestures and hand motions however more formal investigations are necessary to determine which gestures are more natural to our users. The NuWii system is intended to be a starting point, where more gestures can be added to the library so that more end user applications can be supported. While our gestures work acceptably for the described applications, other applications will have their own needs for specific gestures and settings.

Further research into the way that the Wii Remote connects to the computer via Bluetooth would also be helpful, since we noted that other Bluetooth devices occasionally caused interference. Additionally, the error in sensing the points could be reduced by the addition of another Wii Remote placed above the first two. This would help reduce the error in sensing correct points as well as eliminating error when the wrong points are matched.

In this paper we have introduced NuWii, a working gesture-based interface for the Spiegel visualization framework. We have explained our tracking algorithm, and described the gestures that we have implemented thus far. Our system is capable of tracking gestures in 3D, our source code is available to the public under the GNU Public License at `nuwii.googlecode.com`. The input device can be replicated using less than $150 worth of hardware.

REFERENCES

Bak, A. (2004). 3d input devices for the grapecluster project. Independent study report, Rochester Institute of Technology.

Bischof, H.-P. (2010). The Spiegel Project. `http://spiegel.cs.rit.edu/~hpb/grapecluster/Spiegel/index.html`.

Bischof, H.-P., Dale, E., and Peterson, T. (2006). Spiegel - a visualization framework for large and small scale systems. In Arabnia, H. R., editor, *Proceedings of the 2006 International Conference on Modeling, Simulation and Visualization Methods*, pages 199–205. CSREA Press.

Cuypers, T., Van den Eede, T., Ligot, S., and Francken, Y. (2009). Stereowiision: Stereo vision with wiimotes. In *Proceedings of 3D Stereo MEDIA 2009*.

Dehling, E. (2008). Using multiple wiimote cameras to control a game. Master's thesis, University of Twente.

Hay, S., Newman, J., and Harle, R. (2008). Optical tracking using commodity hardware. In *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*, ISMAR '08, pages 159–160, Washington, DC, USA. IEEE Computer Society.

Lee, J. C. (2008). Hacking the nintendo wii remote. *IEEE Pervasive Computing*, 7:39–45.

Underkoffler, J. (2010). John underkoffler points to the future of ui. Talk at TED 2010.

Wang, D. and Huang, D. (2008). Low-cost motion capturing using nintendo wii remote controllers. Csc2228 project report, University of Toronto, Toronto, Ontario M5S 3G4.

Wang, R. Y. and Popović, J. (2009). Real-time hand-tracking with a color glove. In *SIGGRAPH '09: ACM SIGGRAPH 2009 papers*, pages 1–8, New York, NY, USA. ACM.

Wii Brew, C. (2010). Wiimote. `http://wiibrew.org/wiki/Wiimote`.