Wiimote Visualization Through Particles

Joshua Jacobson

1 Abstract

Since 2006, the Wii video game console has been found within homes throughout out the world. The Wiimote exists as the primary input method for the console. Most of the applications of the Wiimote have been for entertainment, but for this project I created visual representation of merely the pure functionality of the Wiimote.

2 Introduction

The Wiimote has been in the mass market for over four years. After its release in 2006, people discovered that the device can easily be connected to computers through the use of Bluetooth. Since then there have been many different libraries written to interact with the Wiimote. Much of the hacking that has been done from the Wiimote has been used for creating new and interesting methods of user input to other devices.

This paper introduces a program that gives pure graphical representations of the basic functionality offered by the Wiimote. These include the IR (Infrared) camera located on the front of the Wiimote and the accelerometer within the device. This program does not take into account any of the various peripherals that can be attached to the Wiimote: the Nunchuk, the Balance Board, or the Classic controller. Additionally, the Wii MotionPlus data was not examined either.

3 Related Works

There have been many different programs that have explored using the Wiimote before. The library that I used for this project came with example code that gave reports about what all of the current data being gathered from the Wiimote was doing. Also, Johnny Chung Lee has made several interesting Wiimote hacks. He has developed applications such as pointing the Wiimote at a projection screen or LCD display to effectively turn them into interactive whiteboards and the idea of placing the sensor bar on oneself and using the IR camera to create head tracking [2]. The Wiimote has also been used as an application controller in many cases. The Wiimote library that was used for this project gave a large list of different programs that were built using it. These includes such projects as using the accelerometer in order to create a point for boids to follow [3], to creating a Theremin from using IR positioning [1].

4 Technical Detail

I decided to use the Managed Library for Nintendo's Wiimote [4] as the library for communication with the Wiimote. Doing so allowed me to use the Microsoft XNA framework for my project. Since this project was done in 2D, I am far more experienced in using this library for those purposes. This library is not as complete as other libraries available because it does not give motion plus support. Additionally, this is a lower level library than others such as GlovePIE, which allows the user to map all the functionality of the Wiimote as if it were another device. For the Wiimote to work with the library, the Wiimote already needs to be considered paired with the computer. This is different than many of the other libraries that exist that have the ability to pair the Wiimote to the computer without going through the operating system.

4.1 IR Data

The IR data was the first data to be collected. The Wiimote has an IR camera on the front of it which allows it to constantly take pictures of IR positions. Then the Wiimote itself is able to keep track of up to four IR points at the same time. These points can then be asked from the library and used. I used the Wiimote sensor bar as my method of creating IR lights. The position of these lights was then used for three different methods of input: position, depth, and rotation. Position is obtained by picking the average position of the two points. If one of the points is not visible, then the system recognizes it as an invalid point and disregards it. Because I am just using the sensor bar, I decided to only look at the first two of the four reported IR points. This allows me to remove some of the problems of IR interference. The next step is depth. The depth data is created by measuring the reported distance between the two points on the sensor bar. The data is then used in the position data to give me a third axis for points. Rotational data is gathered by taking these two points and then finding the angle between them and the x-axis.

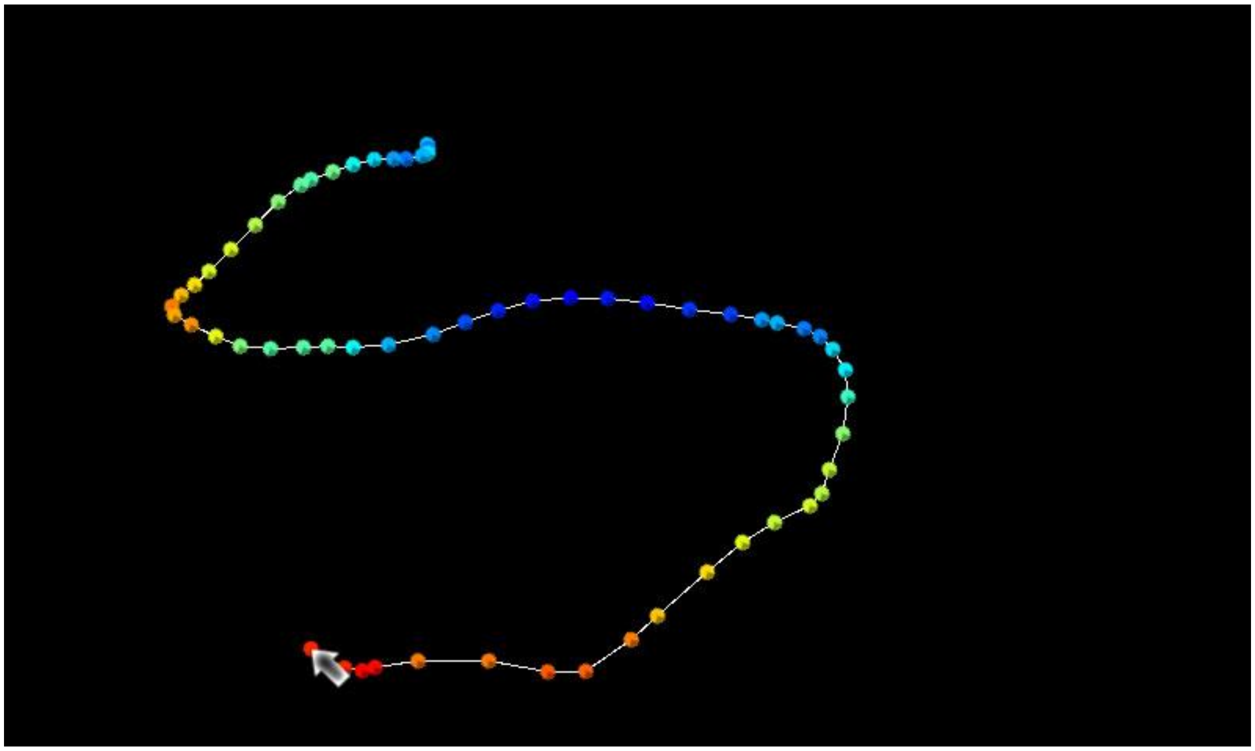

Figure 1Using IR data to create a path. The colors are a color map of the depth component

One of the main problems with the Wiimote itself is the many inaccuracies and ambiguities that the device creates. First, the IR data only works accurately if it is more than a yard away from the sensor bar, but still close enough for the points to be seen. While this does give an acceptable range to gather data in most locations, my room was not one of these and so I spent much time struggling with large data inaccuracies. Another problem that is presented with the Wiimote is that the depth sensing cannot distinguish between whether the remote is actually moving closer or farther away from the sensor bar or whether it is moving to the side of the sensor bar, because in each of these cases, the two light locations end up getting closer or farther away from each other.   Additionally, another problem occurs when the Wiimote chose to index the IR points in reverse order. This leaves rotational data to sometimes be flipped one hundred and eighty degrees because the Wiimote decided to track the two sensor bar points in the opposite positions than what they are usually in.

4.2 Accelerometer Data

The accelerometer is the next part of the Wiimote from which data was gathered. The accelerometer tracks the proper acceleration of the Wiimote in the three spatial dimensions. This means that it tracks the acceleration due to weight rather than actual changes in velocities. The accelerometer sensing along the Y axis is weighted against gravity because of this. When the accelerometer data is gathered from the Wiimote, it is then used to discover the current yaw, pitch, and roll of the object.

The accelerometer also poses problems. The accelerometer almost always has a natural drift in some direction. Additionally, this direction seems to change every time that the Wiimote has a major change in the accelerometer data. This makes the data unsuitable for full 3D position tracking, but can be used to track motions within small time frames. Additionally, the information is inaccurate when the Wiimote is in motion and so true special position cannot be obtained without the use of further input into the system.

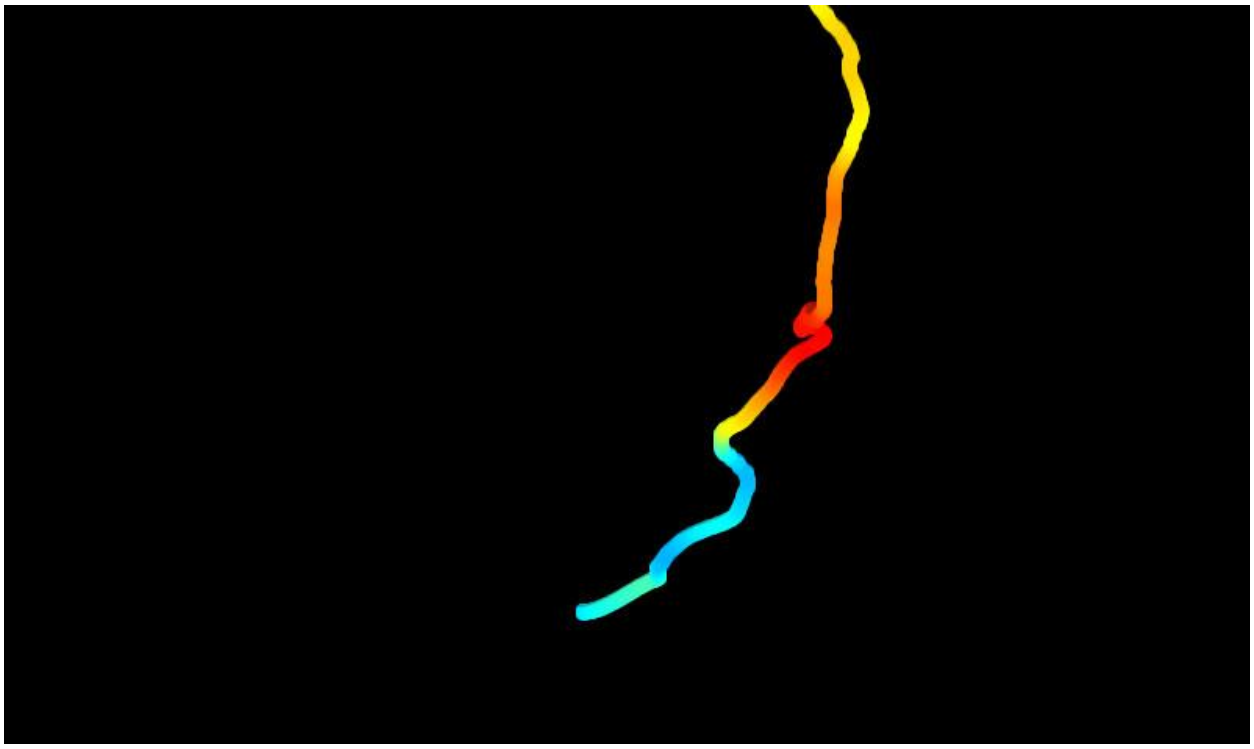

Figure 2 Using the accelerometer data to create the curve.

4.3 Visualizing the data

After the data is received by the program, then it can be interpreted in different ways. First, the IR data is used for positioning on the screen, using one of two methods. One is the single point version in which depth data is used and only a single point is generated. This allows the point to include depth data as it is a combination of more than one point. These positions are then placed within a Catmull-Rom curve in order to create a smooth representation of the path by which the Wiimote was dragged along. In multipoint mode, instead of creating a single curve along the screen, it instead creates curves for every IR point that is tracked. Each time one of these points stops being tracked the curve is ended. This allows multiple curves to be created on the screen, each of them detailing individual sources. This multipoint method does not use any depth data because the data cannot be generated by only looking at a single point using the Wiimote.

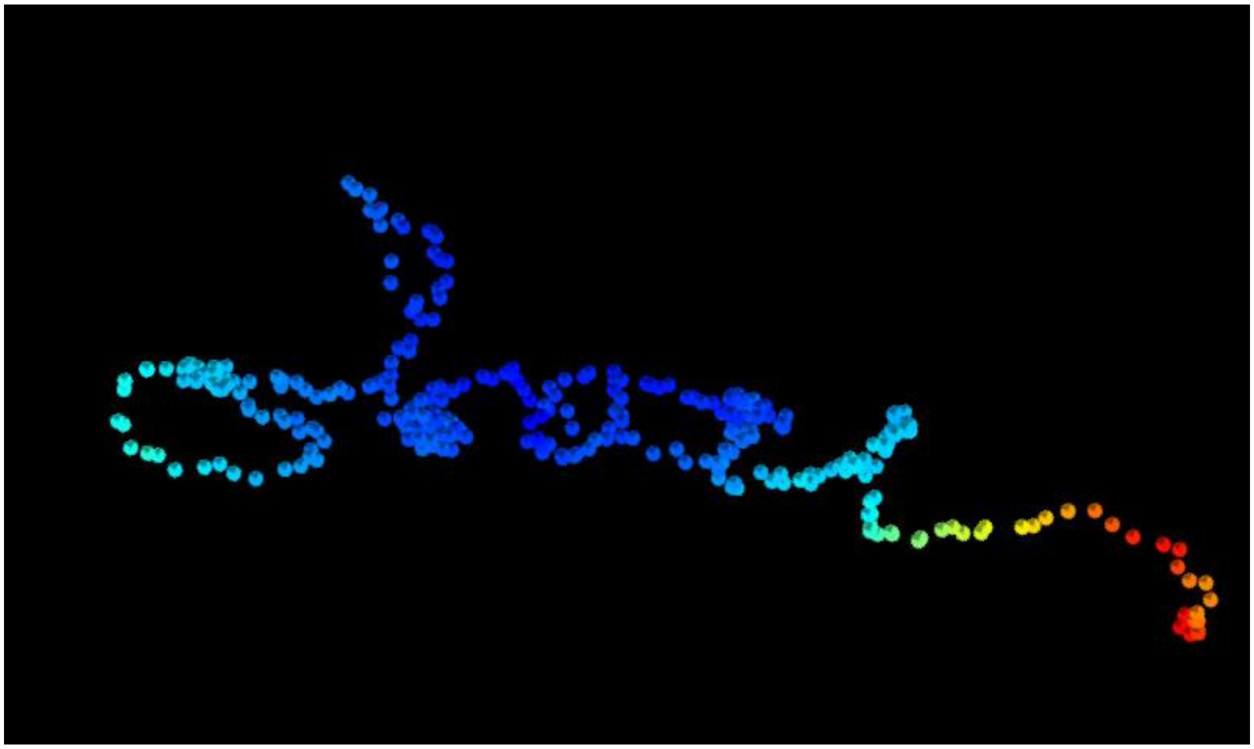

Figure 3 Particles flowing along a created path. The little grey spot on each particle points in the direction that the Wiimote was oriented at that time.

After the path data is created, the system generates particles that flow along each of the different curves generated by the program. Each of these particles contains different data about the curve at any given point. The multipoint version uses simple points that are color coded for each of the different curves. The single point version instead uses larger circles to move along the path. These little circles have a color which is based on the current depth of the object. The color map that is generated for the depth field is based on the minimum and maximum possible values in the current dataset rather than creating an absolute mapping for depth to color. This means that even datasets with very little depth variation will contain the full possible range of color values. I used this method instead of creating an absolute scale as it allows the method to be adapted to the different distance situations that can come from Wiimote. In addition to using color to portray the depth of the particle at any given point, the particles also contain a small arrow in which the rotation of the Wiimote at any given point is also seen. As the rotational data is highly susceptible to being flipped one hundred and eighty degrees many times these arrows do point in the opposite direction.

The accelerometer data is used to actually move the position of the various particles on the screen. These particles can be moved in two different ways. The first way is by gathering data from the instantaneous pitch, yaw, and roll data that is being gathered from the accelerometers and then applying them as a force in the direction of the last known position on the screen. This makes it so the user can fan and shake the Wiimote around and have it affect the force pushing on the particles on the screen. The second method is to simply modify the data based on the Wiimote data that is gathered directly from the accelerometer. In both of these cases, it is possible to see the effect of the natural drift of the accelerometer. This second method also has a second function in which the program creates a path directly based on the data gathered from the accelerometer, rather than with IR data. This data can be acted on in both of the other ways.

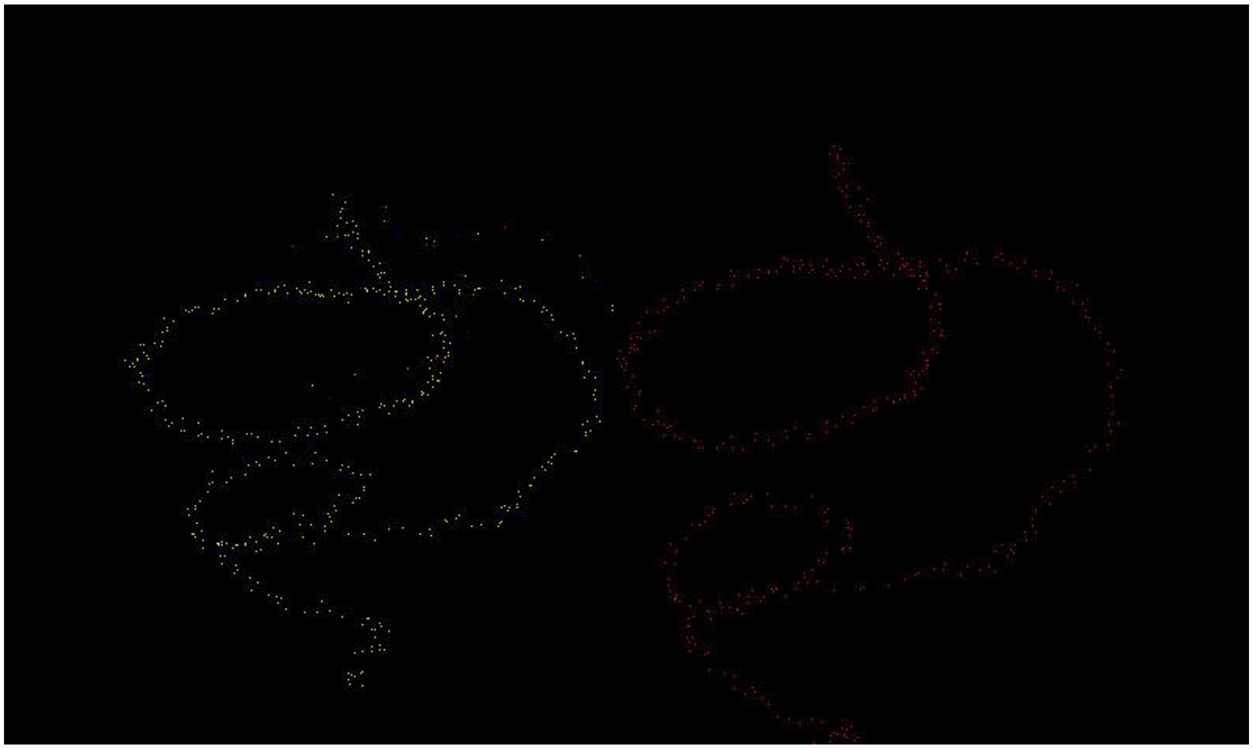
Figure 4 Multipoint Tracking. It is possible to see some little blips next to the yellow path that were caused by IR Interference.

5 Results

During the development of the project I ran into many problems. The first was that my work area is not suitable to development on the Wiimote in this way. My work area is somewhat small and therefore much of the time my Wiimote ended up too close to the sensor bar. This caused data collection problems because one of the IR lights would be flickering in and out of the sensor. This is a problem that I also have with playing the Wii itself and so I expected to run into this. Normally I need to move my sitting position one or two feet away from my computer screen, but in this case I needed to stay closer to the screen in order to test the application along with looking at the code for bugs. In addition, for some reason my room tends to have a lot of extraneous IR light and so the Wiimote would pick up lots of IR interference while I attempted to gather clean data. This would mean that the pointer would end up moving all around the screen, with little that I could do to try and fix the problem. I eventually needed to rearrange my room in order to successfully complete the project.

Apart from the location issues that I faced when creating this project, most of the problems were caused by the inaccuracies of the Wiimote and its accelerometer data. The natural drift, constant yet ever-changing, that the accelerometer produced made all of the different applications of the accelerometer hard to process. I eventually had to hardcode in some basic movement fixes that lessened the natural shift quite a lot, because although the shift did change as actions were performed, there seemed to always be a constant downwards force of one that seemed to exist no matter how the shift changed within the program.

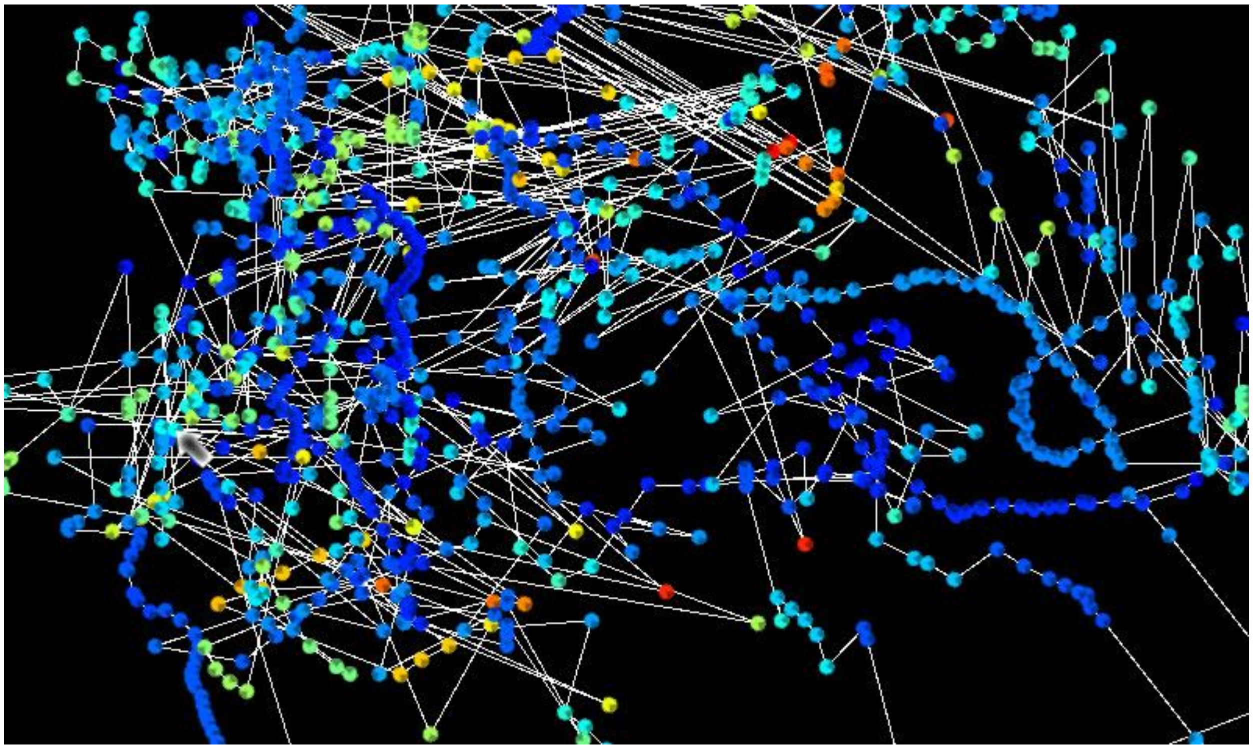
Figure 5 IR Interference.

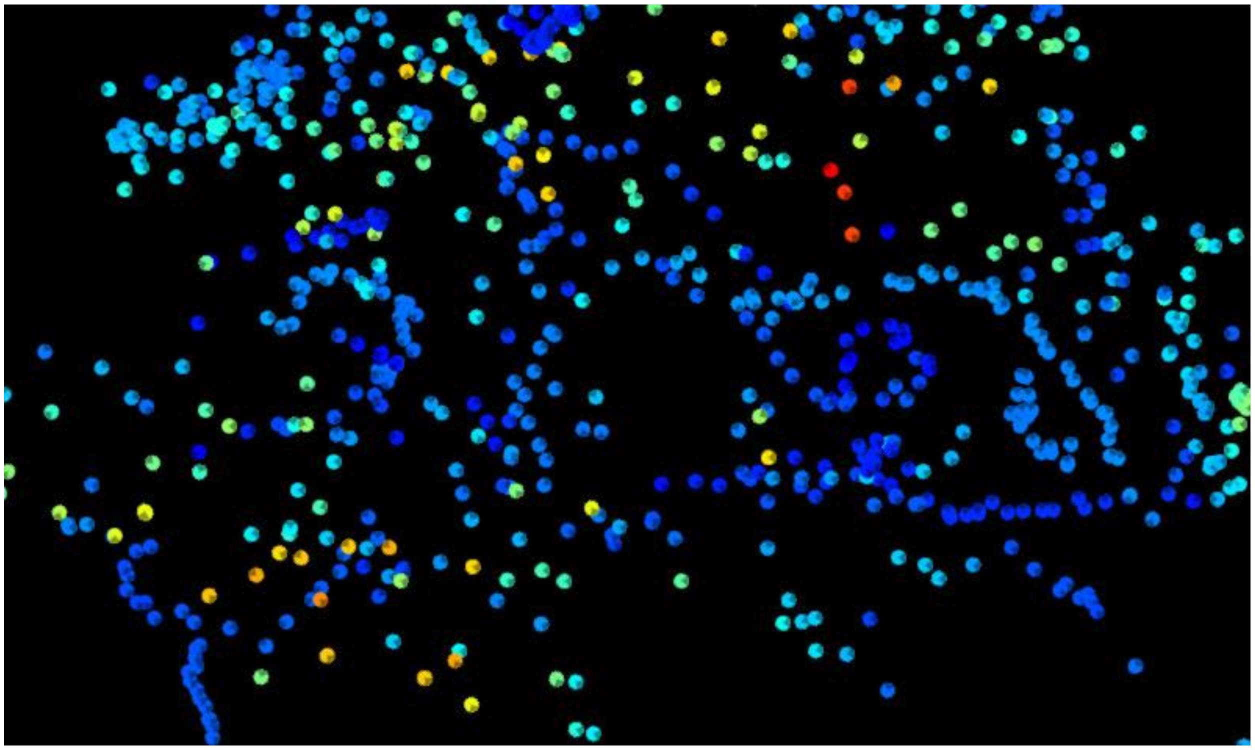

Figure 6 Resulting particles from IR Interference.

Despite these problems, the program does what it was created to do: visualize the input data from the Wiimote through the usage of a particle field.

6 Conclusion

Overall, I created a small program that allows the visualization of the Wiimote's different basic non-button data input. It provides visual representations of the IR camera data and accelerometer data in both real time simulations and playback through the use of particles.

7 References

[1] R. Burke. A Wii Flock of Boids: Integrating the Nintendo Wii Controller with XNA http://robburke.net/2007/06/a-wii-flock-of-boids-integrating-the-nintendo-wii-controller-with-xna/

[2] J. Lee. http://johnnylee.net/projects/wii/

[3] K. Moore. Wii Theremin - How It Works http://kenmooredesign.blogspot.com/2008/11/wii-theremin.html

[4] B. Peek. Managed Library for Nintendo's Wiimote http://wiimotelib.codeplex.com/