Stephen Paladines

12/09/18

Project Reflection

Tackling a new coding project is always a challenging, especially one in which you don’t know where to begin. With the Kinect Project, I had to learn where to start and how to tackle the idea within the next few months. The first goal I created was learning about the C# language and how it works. After setting aside a few hours a week, I learned that the language is used for all Microsoft applications and is similar to C++ and C, two common academic programming languages. Looking at the syntax and writing a few examples of code made it easy to pick up.

Once I felt comfortable writing in C#, the next step was to learn about the XAML component of the project. XMAL stands for (Extensible Application Markup Language) and is used to create visuals known as “GIUs’ for applications also referred to as Windows Presentation Foundation or WPF for short. Although creating and modifying the WPF’s were an issue down the line, I managed to understand the fundamentals.

Now that I learned about the languages behind the Kinect, it was now time to find resources with documentation on how to work with the sensor and utilize all its functionalities. One of the common issues I faced early on was relying on code and information form users that was outdated. With the release of the Kinect SDK v2, several modifications were made to the code and rendered previous projects broken. What helped tremendously was a book called “***Beginning Microsoft Kinect Windows SDK 2.0***” by author Mansib Rahman. Mansib managed to break down each line of code and explained its role in the grand scheme of things.

From following the textbook, I was able to discover that Microsoft provided templates for different types of projects with the Kinect. I decided to use “***Control Basics***” as my template. Having the necessary resources, I was able to tackle the issue of creating an application that would render objects for users and provide them with a measure of accuracy. Right away, I was able to start implementing the window size, buttons, and logic behind the scene just to name a few. One of the most challenging parts of the project was getting the Kinect cursor to call an event whenever the cursor was hovering over the test object.

The current version lacked the hovering method implemented in the previous SDK and instead choose to use a push functionality for objects. After iterating though different ideas, the most practical approach was to use the built-in functions for the mouse. These functions, known as “***mouse.enter and mouse.leave***” will trigger whenever the mouse is within an object. My strategy was to update the mouse to a location close to the Kinect cursor. That way whenever the courser is hovering on an object, the mouse will also display at the location and call the methods needed.

Another challenging issue with this project was measuring the accuracy between the cursor and object. The current method implemented works by waiting a 10 millisecond whenever it is detected that the cursor has enter the area of the object. This delay is needed in order to get the position based on the location after entering the area and not the position when entering for the first time. All in all, this project taught me the inner works of the Kinect and how to create exciting projects with a device that is rich in features: IR, Depth, Tracking and much more.