J.S. Peirce

Edward Sayers

ECE 478: Homework 1

**Introduction**

The goal of this project was to develop an application that allows a PowerPoint slideshow to be controlled with a Kinect. This was done using a WPF application to create an overlay on top of PowerPoint. The application displays a cursor which is controlled by hand movements. It allows regions of the slide to be pressed in order to preform actions such as changing slides. An XML document is used to define the controls available to each slide.

**Application Use**

A WPF application was chosen because of its support for the Kinect API and the availability of example programs. Connection of a Kinect sensor is handled by the sensor chooser object provided by the API. This allows visual feedback indicating when the Kinect sensor is connected properly as well as feedback indicating when there is a problem. When a sensor is connected, an icon at the top of the screen will indicate that it is connected properly and then fade out. If there is a problem with the connection a message will appear indicating the source of the problem.

Once a sensor is properly connected, a user can gain control of the application by standing in front of it and raising a hand. A cursor shaped like a hand will appear on the screen and will follow the motion of the users raised hand. When the cursor hovers over a selectable area of the presentation it will change to indicate as such. While the cursor is hovered over one of these areas, the user may “push” the button by moving their hand toward the sensor. The cursor will fill until the button is fully pushed after which time the action triggered by that button will happen.

The buttons themselves are hidden in static locations on the screen. There are five total buttons:

* A button along the left edge of the screen to move to the previous slide.
* A button along the right edge of the screen to move to the next slide.
* A button in the bottom center of the screen to quit the application.
* Two buttons in the center of the screen to trigger actions.

Each of these buttons are disabled by default but can be enabled on a slide-by-slide basis in the XML document that accompanies the presentation. In this way each slide can have different actions available.

When the application is first started, it will prompt the user to open the XML file that defines the presentation. The XML document is defined as follows:

<?xml version="1.0" encoding="utf-8" ?>

<Presentation filename="presentation.pptx">

<Slide id="1">

<Button id="ButtonName"/>

</Slide>

</Presentation>

The outer “Presentation” node must include the filename of the actual PowerPoint presentation that is to be opened. For every slide in that presentation, the XML file must have a “Slide” node. An id property may be defined for each slide node but is not currently used. Inside each slide node there must be a “Button” node for each button that is enabled for that slide. The “id” property for each button node should be the name of the button which is to be enabled.

**Design**

The application is split into three main objects: *PowerpointControl*, *KinectSlideShow*, and *MainWindow*. The *PowerPointControl* object uses the office interop classes to control the PowerPoint slideshow and track its state. The *KinectSlideShow* object parses the XML file to build a list of slides along with the properties of those slides. The *MainWindow* object brings these both together, along with the Kinect API to define the behavior of the application.

The *PowerPointControl* object has the following methods and properties:

* Properties
  + Filename – The name of the file containing the PowerPoint slideshow.
  + CurrentSlide – The index of the currently displayed slide.
  + SlideCount – The total number of slides in the current presentation.
  + IsOpen – Returns true if the slideshow is open, false otherwise.
* Methods
  + GotoSlide – Change to the slide at the given index.
  + NextSlide – Change to the next slide.
  + PreviousSlide – change to the previous slide.
  + Close – Close the current presentation.

It also defines an event handler that is triggered whenever the current slide is changed.

The *KinectSlideShow* object has the following methods and properties:

* Properties
  + Slides – List of slides in the slideshow
  + IsOpen – Returns true if the slideshow has been loaded from XML, false otherwise
* Methods
  + Open – Open XML file and make a list of the slides
  + Close – Delete the list of slides

The *MainWindow* object has the following properties:

* Properties
  + sensorChooser – The *KinectSensorChooser* object.
  + slideShow – The *KinectSlideShow* object.
  + ppControl – The *PowerPointControl* object.

This object also contains event handlers for various events for which a reaction is necessary. There is an event handler for each of the five Kinect buttons and there is an event handler for when a slide changes. The slide change event looks for all of the buttons in the window and enables those that were defined in the XML document.

The final source code for this project is located at the following web address: https://github.com/eulerworksinc/REDMOON/tree/xml\_parse/Kinect-PP-WPF