**Kinect Control**

The Kinect control program is written in C# and uses XAML for the interface. The .NET framework is used to send and receive information from Windows applications, process speech recognition, play sounds, draw video and objects to the screen, and to ultimately interact with PowerPoint and Windows itself. The Microsoft Kinect SDK version 1.7 is used to interface with the Kinect. Code published under open licenses was used from various sources and links are provided in the .cs file header comments.



Figure – Microsoft Kinect

The software is implemented in a single window that is hosted in a XAML browser. The code starts by instantiating a *MainWindow*, declaring various variables, flags, and objects such as the *KinectSensor*, *SoundPlayer*, and the Microsoft *SpeechRecognitionEngine*. During initialization the system checks that a Kinect is attached and starts various subsystems on the Kinect that will be used. An event handler routine is provided to accept keyboard input and set flags/run code as appropriate. An exit routine is also provided to properly stop audio and video streams from the Kinects cameras and microphones and free memory associated with the sensor and to stop speech recognition as well.

Another program named AutoHotkey is used to ensure that the main window is always on top of other windows which works to integrate the visual feedback of the Kinect software with other programs such as PowerPoint. A script runs in the background that keeps a particular window on top when called. When the control key and tilde (~) are pressed simultaneously the script runs. The script toggles an “always on top” setting to keep the current window in focus continuously on top. This can be done manually with the keyboard or with the phrase “computer toggle lock” via the speech recognition functions of the program.



Figure – AutoHotkey script

Every frame the Kinect provides a video frame from the color camera. If a human is detected by the depth camera a skeleton object is created. The closest detected human in the Z axis (line of sight of the camera) is assigned a skeleton. A skeleton is a Kinect object that has a stick-figure-like body with joints and many other useful properties. Joint objects are then created that track the head, left hand and right hand and circles are overlaid onto these joint positions. The head and left hand circles are small, slightly transparent, and green. The right hand circle’s color, transparency, and size are determined by the relative distance between the right hand and the head compared to a certain set point distance.



Figure – User window

|  |  |
| --- | --- |
| **Phrase** | **Function** |
| computer show window | Shows main window |
| computer hide window | Hides main window |
| computer show circles | Shows circles on select joints |
| computer hide circles | Hides circles on select joints |
| computer close window | Shuts down program |
| computer turn on mousecomputer turn mouse oncomputer enable mouse | Enables mouse control |
| computer turn mouse offcomputer turn off mousecomputer disable mouse | Disables mouse control |
| computer switch window | Sends ALT-F4 to Windows  |
| powerpoint full screen | Sends F5 to Windows |
| powerpoint return | Sends ESC to Windows |
| computer undocomputer go back | Sends CTRL-Z to Windows (not currently working) |
| computer print screen | Sends ALT-PRTSCN to Windows  |
| computer toggle lock | Sends CTRL-SPACE to Windows |
| open paint | Sends WIN+`PAINT’+ENTER |
| computer maximize window | Sends WIN+Up |

The relative distance is calculated by the 3-space distance formula and this is normalized to a percentage of a certain set point. The default set point is 0.5 meters away but there are plans to include a calibration procedure that can set this distance according to body dimensions of the user. The closer the hand is to indicating a right-click, the larger, more opaque and more red the circle becomes. Once the limit is reached (0.5m) a flag is set that is sent via the Kinect method *SendMouseInput*. The position of the right hand in 3-space is converted via projection to a 2-space location and is scaled by the size of the skeleton and the resolution of the monitor being used. These raw values are added to an array that stores a number of previous inputs. A simple rolling average filter is applied before sending these coordinates via *SendMouseInput*. A clicking sound file is played upon a successful click.

The mouse-click gesture and mouse position gesture are processed in the same function. Two other gestures are processed that help to interact with PowerPoint. Moving the left hand far left will send a left arrow key command through Windows forms, and moving the right hand far right will send a right arrow key. Various other gestures have been developed for uses outside of PowerPoint but are not being used in this version of the software.

Table 1 – Speech commands

In addition to gestures, speech recognition is used to process human input. There are a number of phrases that are used to compare speech input to. There are also multiple versions of the same phrase/command that will perform the same action such as “computer enable mouse”, “computer turn mouse on”, and “computer turn on mouse”. Included is a table that lists all phrases currently in use and some information about their use. There are commands to open programs such as paint (crudely implemented), run scripts, send Windows hotkey commands, and interact with the Kinect software itself.

The speech recognition is implemented with Microsoft’s GrammarBuilder class. English is currently selected as the language used, but this could be changed as needed. The Kinect has 4 microphones that are set up as a linear array and their collective beam angle, ie where they are listening, can be set explicitly. This program uses an adaptive beam angle that actively tracks the loudest audio source and uses this for recognition. When speech recognition begins it starts a four second timer once a loud enough sound is detected. It attempt to find a match among candidate phrases and returns a confidence level of a match. The program will currently use any match that has a 70% or higher confidence rating from the engine. Figure 4 shows a debugging run of the speech recognizer with the speech events, their confidence levels and interpretations.

Figure 4 – Speech recognition