2013

INNOVATIVE HUMAN-COMPUTER INTERACTIONS TO SUPPORT COGNITIVE COLLABORATIVE GEOSPATIAL ENVIRONMENTS

Kevin Takala
Michigan Technological University

INNOVATIVE HUMAN-COMPUTER INTERACTIONS TO SUPPORT COGNITIVE COLLABORATIVE GEOSPATIAL ENVIROMENTS

By

Kevin Takala

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Integrated Geospatial Technology

MICHIGAN TECHNOLOGICAL UNIVERSITY

2013

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Integrated Geospatial Technology.

School of Technology

Report Advisor:	*Evgueny Levin*
Committee Member:	*Yushin Ahn*
Committee Member:	*Robert Pastel*
School Dean:	*James Frendewey*

Table of Contents

Abstract

In the current world geospatial information is being demanded in almost real time, which requires the speed at which this data is processed and made available to the user to be at an all-time high. In order to keep up with this ever increasing speed, analysts must find ways to increase their productivity. At the same time the demand for new analysts is high, and current methods of training are long and can be costly. Through the use of human computer interactions and basic networking systems, this paper explores new ways to increase efficiency in data processing and analyst training.

1. Introduction

In order to keep up with the ever increasing speed at which data is being demanded geospatial, analysts must find new and more efficient ways to process and output data. This data includes satellite, aerial and terrestrial images. These types of data can be applied in a variety of different fields including national security, public infrastructure, forest/crop health evaluation, and more. In order to generate results at a greater speed, analysts in the past have developed algorithms to automatically classify and process data. But while sophisticated and highly advanced algorithms can produce high accuracy results in some cases, there is still room for errors in the output data, and a human is required to identify and fix them. Another issue is the lack of people capable to analyze these types of data. The training of new analysts can be slow and expensive, especially in areas that lack capable teachers. It is therefore necessary that new systems be developed to maximize not only the accuracy and speed at which data is produced, but also the speed at which analysts are trained. This paper looks at a relatively new system, in which the analysts can interact with one another in an attempt to increase efficiency in these areas. The goal of this system is to improve the human-computer and human-human interaction to address the above issues. The methods explored in this paper involve the integration of an electroencephalograph (EEG) with Google Hangout (GH). The system was evaluated in terms of usability, efficiency, and its ability to increase productivity.

2. Background

An EEG is a device that measures neural impulses on the scalp. This is accomplished by measuring the voltage of the impulses at, in the case of the selected headset, 16 locations. The measured voltages are compared to a standard baseline provided by the manufacture's software. The baseline is then tailored to specific users through training sessions. For this study, the Emotiv Epoch was chosen (see figure 1). The Epoch allows users to develop interactive programs that can be run separately or in conjunction with other software programs and applications. Users have the option to develop programs in C++, C#, .NET, and Java formats. Within the Epoch software, users have the option to use the Cognitive, Expressive, and Affective Suites. This study is focused on the use of

the Cognitive Suite only, which is described below. For more information on how the Epoch works and other suites, see [1].

Figure 1. Image of the Emotiv Epoch. Image courtesy of http://www.emotiv.com/about/media/

The Cognitive suite (see figure 2 on the next page) detects a user's real time brainwave activity and discerns his/her mental intent to perform physical actions on objects. There are 13 individual commands the user may use. They are push, pull, left, right, up, down, rotate clockwise, rotate counter-clockwise, rotate left, rotate right, forward, backward, and disappear. Users improve the suites ability to detect the different commands through training. Training involves thinking of the selected action for eight seconds. After the eight seconds, users have the option to accept or reject that training session. After undergoing sufficient training the user may use the headset to preform basic tasks. It has been reported in [2] that the use of an EEG along with other tools is an effective method for increasing the productivity of GIS/Spatial analysts.

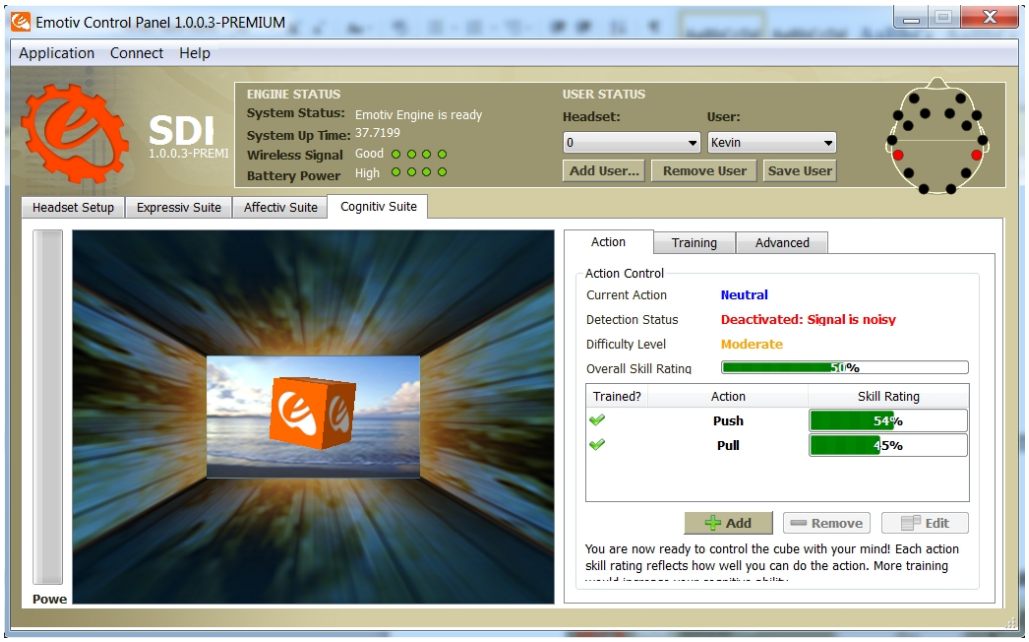

Figure 2. The basic layout of the Emotiv Control Panel for the Cognitive Suit.

In the case of this study, we investigated the use of the EEG in Photomod Lite. Photomod Lite is an open source photogrammetry software package that allows users to process satellite and aerial images. The system reviewed in this paper is designed to work with the stereoscopic viewing portion of Photomod, see figure 3. In this system, the user has the ability to adjust the parallax of the cursor for height extraction of features located in images.

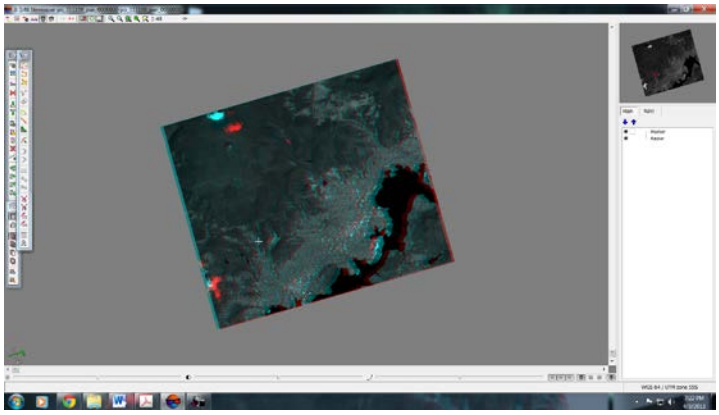

Figure 3. Image of Photomod when used for stereoscopic viewing of images (note that the Photomod image is not adjusted for parallax).

Google Hangout (GH) is an open source program which allows users to communicate via the internet with one another. Its uses include, communicating between friends, family and for business conference calls. GH has been shown in [3] and [4] to be an efficient tool for long distance learning and communication. Here, GH's ability to dual screen share is investigated as a method to allow for better collaboration between analysts working in different areas.

3. Methods

3.1 Interactions with Software

For the described research C# was chosen for the development of the application. This choice was based on its ease of use and the programmers existing knowledge of this language. The designed program creates a simple window allowing the user to select the program they wish to interact with (i.e. either Google Earth or Photomod, though only Photomod was tested in this research). See Appendix A for the source code.

Once the program is selected, a basic set of commands is displayed in the window. Upon clicking the start button the user can interact with the selected program. Interactions are currently done through the use of automated mouse scrolling and program specific hotkeys (see Appendix A: Epoch_Program.cs). The system is designed so that users may still work outside of the program without fear of the headset commands affecting other applications.

In the current setup the user is allowed to use two different thoughts to interact cognitively with the desired application. The thoughts are a basic push and pull command. In Photomod, a push thought will cause the cursor to appear to move toward the ground, and a pull thought will cause the cursor to appear to be moving away from the ground.

3.2 EEG - Google Hangout Integration

The goal of this research was to see if integrating an EEG with GH will allow for a more efficient work pace and accurate output data. In order to test this, two computers were linked via GH's dual screen share option. One user was designated as a theoretical "geospatial analyst expert" (see figure 4) and the other designated as a "trainee" or

“student”. The users then selected an area in a stereoscopic image to be measured for height. Users were only allowed to use the mouse to select the feature of interest, parallax adjustment had to be done using the EEG. Once the object was selected both users adjusted the parallax, and by viewing the trainee’s screen, the expert would then dictate whether the parallax had been adjusted correctly, see figure 5.

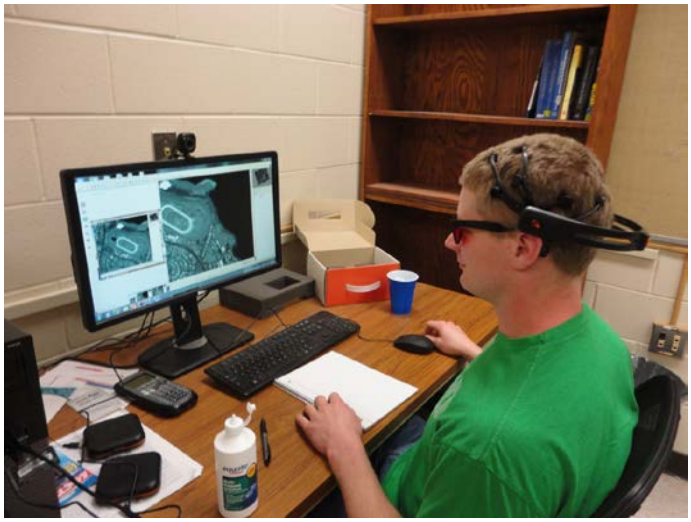

Figure 4. Image showing the expert working with the system.

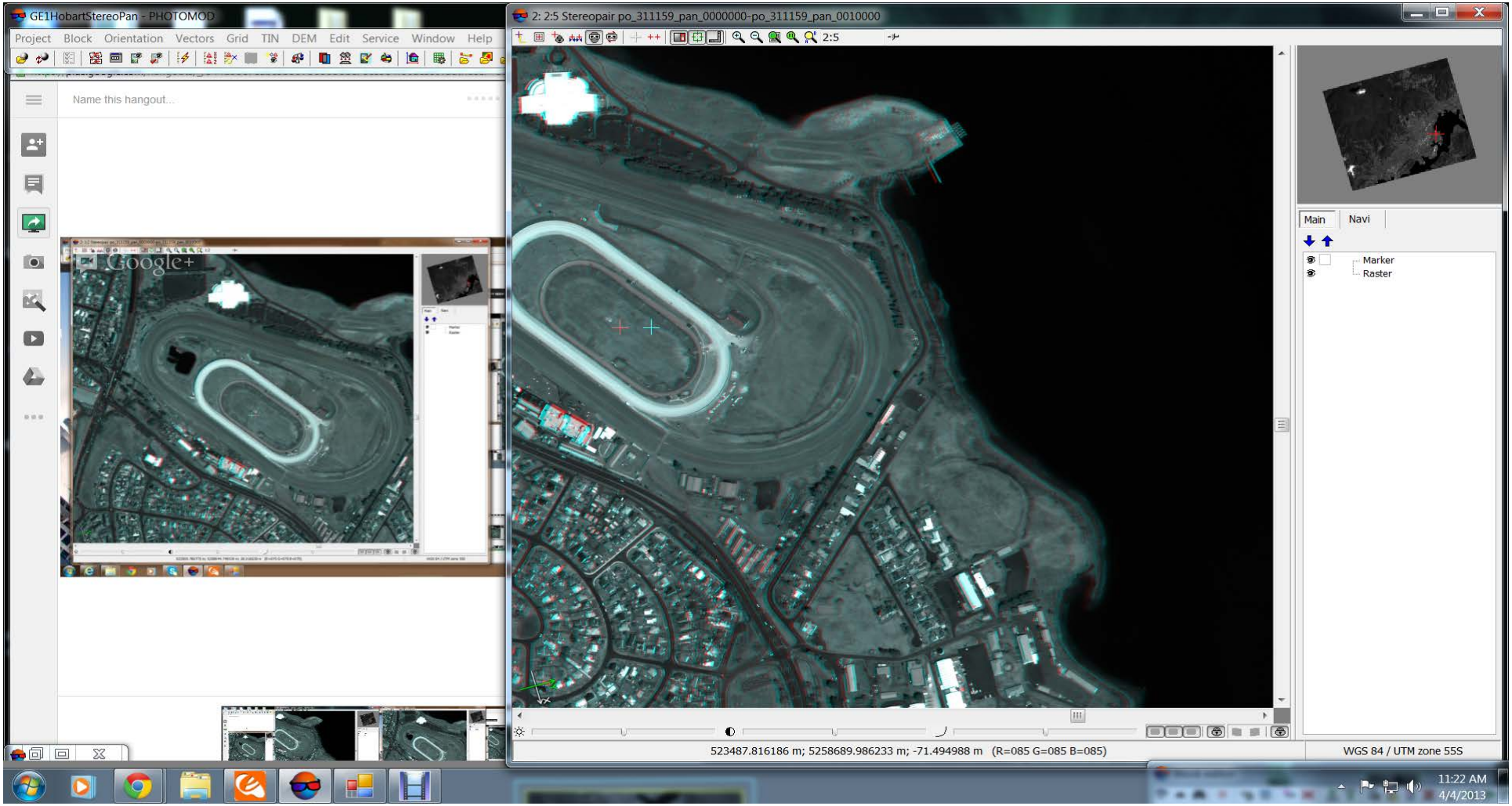

Figure 5. Image showing the basic layout of the system. GH window showing the trainee's screen (Left). Photomod window controlled by the expert (Right).

4. Results and Discussion

The overall results show that combining the Emotiv Epoch with GH is an efficient tool for increasing productivity, output data accuracy, and analyst training speed. It is especially useful in the training of new analyst. No longer would trainees and experts need to be in the same place at the same time; they could be located anywhere where internet access is available. While the described system has only been used in two software systems, it is possible to create such a system for many different software programs and applications. It is also possible to expand the number of commands the user can perform using the headset. However, as the number of commands the user uses via the headset increases, so does the difficulty in training the different commands. The number of users in the system is also not limited to two, but can be increased to as many as needed. Although the more users there are, the smaller the screens of each, and therefore the resulting accuracy of measurements may be decreased.

The biggest flaw in the current system is the disproportional size of the windows. If the shared screen of the trainee was of equal size to the expert's Photomod window the resulting accuracy of measurements would be improved. The system may also be prone to mistakes when the user inadvertently has a thought similar to one of the trained actions, resulting in a false command. For example, if the user should become excited and the Epoch mistakes the resulting electrical impulses for a push command, the cursors parallax would be erroneously adjusted. In order to mitigate this problem, users should use the system routinely. Isolating users from areas and situations that could cause unwanted, external influences on their thoughts or emotions may also be helpful. Increased training and use of the headset does allow for a more accurate distinction between different thoughts. The designed interaction program also has a flaw in that any window displaying certain key words in its title bar will be affected by the user's thoughts.

5. Conclusions

In conclusion, the developed system was found to be effective for preforming basic geospatial processing tasks. It has shown itself to be capable of increasing the productivity and accuracy of the output data. Though, it is more through the systems ability to allow collaboration than through its increase in human computer interactions that these results were produced. Through daily and repeated use the system became easier to use, which is most likely a combination of actual training and the analyst becoming more familiar and comfortable with the system. The benefits of using this system include but are not limited to, higher levels of productivity, more accurate output data, and a greater ability to help in the processing of data in areas all over the world. It has also been shown to be an effective tool in the training of new analysts. However, there is still room for improvement in it. By making the windows equal in size the system would be greatly improved. A program which is capable of only working within the desired software program, and a headset which is capable of better distinguishing individual thoughts, would also help improve the system. In future studies it is recommended that these items be improved upon, and in doing so, the system will be a more effective tool for users.

6. References

[1] Levin, E.; Carter, J.; Sergeyev, A. V. “Human-Computer Symbiosis in Cyberspace Environments”. *SPIE Defense and Security Symposium 2012, pp 21-27.*

[2] Carter, J. “Improving the Performance or GIS/Spatial Analysts Through Nobel Applications of the Emotiv Epoch EEG Headset”. *Masters Degree Report, Michigan Technological University 2012.*

[3] Damian, D.; Lassenius, C.; Paasivaara, M.; Borici, A.; Schroter, A. "Teaching a globally distributed project course using Scrum practices". *Collaborative Teaching of Globally Distributed Software Development Workshop (CTGDSD), 2012* , vol., no., pp.30,34, 9-9 June 2012.

[4] Heines, Jesse M.; Jassem, Krzysztof. “Teaching Internationalization – Internationally”. *Innovation and Technology in Computer Science Education Conference 2013, Canterbury, U.K.*

7. Appendix A

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Gyro_EEG
{
   static class Program
   {
      /// <summary>
      /// The main entry point for the application.
      /// </summary>
      [STAThread]
      static void Main()
      {
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         Application.Run(new Epoch_Program());
      }
   }
}
```

Eopch_Program.Designer.cs

```
namespace Gyro_EEG
{
   partial class Epoch_Program
   {
      /// <summary>
      /// Required designer variable.
      /// </summary>
      private System.ComponentModel.IContainer components = null;

      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
      protected override void Dispose(bool disposing)
      {
         if (disposing && (components != null))
         {
            components.Dispose();
         }
         base.Dispose(disposing);
      }
```

```
        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.appSelect = new System.Windows.Forms.ComboBox();
            this.startButton = new System.Windows.Forms.Button();
            this.stopButton = new System.Windows.Forms.Button();
            this.HotkeyBox = new System.Windows.Forms.TextBox();
            this.BGEmoLoop = new System.ComponentModel.BackgroundWorker();
            this.label1 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // appSelect
            //
            this.appSelect.FormattingEnabled = true;
            this.appSelect.Items.AddRange(new object[] {
            "Google Earth",
            "Photomod"});
            this.appSelect.Location = new System.Drawing.Point(13, 28);
            this.appSelect.Margin = new System.Windows.Forms.Padding(4);
            this.appSelect.Name = "appSelect";
            this.appSelect.Size = new System.Drawing.Size(233, 24);
            this.appSelect.TabIndex = 0;
            this.appSelect.SelectedIndexChanged += new
System.EventHandler(this.appSelect_SelectedIndexChanged_1);
            //
            // startButton
            //
            this.startButton.Location = new System.Drawing.Point(17, 131);
            this.startButton.Margin = new System.Windows.Forms.Padding(4);
            this.startButton.Name = "startButton";
            this.startButton.Size = new System.Drawing.Size(100, 28);
            this.startButton.TabIndex = 2;
            this.startButton.Text = "Start";
            this.startButton.UseVisualStyleBackColor = true;
            this.startButton.Click += new System.EventHandler(this.startButton_Click);
            //
            // stopButton
            //
            this.stopButton.Enabled = false;
            this.stopButton.Location = new System.Drawing.Point(136, 131);
            this.stopButton.Margin = new System.Windows.Forms.Padding(4);
            this.stopButton.Name = "stopButton";
            this.stopButton.Size = new System.Drawing.Size(97, 28);
            this.stopButton.TabIndex = 3;
            this.stopButton.Text = "Stop";
            this.stopButton.UseVisualStyleBackColor = true;
            this.stopButton.Click += new System.EventHandler(this.stopButton_Click);
            //
            // HotkeyBox
            //
```

```
    this.HotkeyBox.Location = new System.Drawing.Point(259, 27);
    this.HotkeyBox.Margin = new System.Windows.Forms.Padding(4);
    this.HotkeyBox.Multiline = true;
    this.HotkeyBox.Name = "HotkeyBox";
    this.HotkeyBox.ReadOnly = true;
    this.HotkeyBox.Size = new System.Drawing.Size(240, 155);
    this.HotkeyBox.TabIndex = 4;
    //
    // label1
    //
    this.label1.AutoSize = true;
    this.label1.Location = new System.Drawing.Point(16, 7);
    this.label1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
    this.label1.Name = "label1";
    this.label1.Size = new System.Drawing.Size(77, 17);
    this.label1.TabIndex = 5;
    this.label1.Text = "Application";
    //
    // label2
    //
    this.label2.AutoSize = true;
    this.label2.Location = new System.Drawing.Point(259, 6);
    this.label2.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);
    this.label2.Name = "label2";
    this.label2.Size = new System.Drawing.Size(57, 17);
    this.label2.TabIndex = 6;
    this.label2.Text = "Options";
    //
    // Form1
    //
    this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);
    this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
    this.ClientSize = new System.Drawing.Size(508, 187);
    this.Controls.Add(this.label2);
    this.Controls.Add(this.label1);
    this.Controls.Add(this.HotkeyBox);
    this.Controls.Add(this.stopButton);
    this.Controls.Add(this.startButton);
    this.Controls.Add(this.appSelect);
    this.Margin = new System.Windows.Forms.Padding(4);
    this.Name = "Form1";
    this.Text = "Gyro EEG";
    this.TopMost = true;
    this.ResumeLayout(false);
    this.PerformLayout();

}//end initializecomponent method

#endregion

private System.Windows.Forms.ComboBox appSelect;

private System.Windows.Forms.Button startButton;
private System.Windows.Forms.Button stopButton;
private System.Windows.Forms.TextBox HotkeyBox;
private System.ComponentModel.BackgroundWorker BGEmoLoop;
```

```
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;


        public System.EventHandler label1_Click { get; set; }
    }
}
```

Epoch_Program.cs

```
using System;
using System.Runtime.InteropServices;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows;
using System.Windows.Forms;
using Emotiv;
using MouseKeyboardLibrary;
using EARTHLib;

namespace Gyro_EEG
{
    public partial class Epoch_Program : Form
    {
        // Access to the EDK is via the EmoEngine
        EmoEngine engine;

        // userID is used to uniquely identify a user's headset
        int userID = -1;

        // Event Handler for emotive actions
        EmoEngine.EmoStateUpdatedEventHandler KeyEvent;

        ApplicationGE googleEarth;

        String activeWindow;
        // Used to find the user application selection
        int SelectedIndex = -1;

        IntPtr hWnd = GetForegroundWindow();
```

```
/// <summary>
/// The constructor
/// Creates a new winows form with the specified layout
/// </summary>
public Epoch_Program()
{
   // Create EmoEngine
   engine = EmoEngine.Instance;

   //add a user to the new engine
   engine.UserAdded += new EmoEngine.UserAddedEventHandler(engine_UserAdded_Event);

   // Create windows form1
   InitializeComponent();

   // Allows cancelation of backgroundWorker BGEmoLoop
   BGEmoLoop.WorkerSupportsCancellation = true;

   // Event Handler for Background Worker
   BGEmoLoop.DoWork += new DoWorkEventHandler(BGEmoLoop_DoWork);


   // Sets initialization text for Hoy Key Infor Box
   HotkeyBox.AppendText("Please Select Application");
   startButton.Enabled = false;


}//end form1 constructor

#region Functions

/// <summary>
/// Gets the current cognitive action from the control panel and translates it to an action
/// 1 = neutral, 2 = push, 4 = pull, etc.
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void ReadEmo_GE(object sender, EmoStateUpdatedEventArgs e)
{
   // Access to user's Emo State
   EmoState state = e.emoState;

   int action = (int)state.CognitivGetCurrentAction();


   if (GetActiveWindowTitle() == "Google Earth")
   {
       switch (action)
       {
         //neutral
         case 1:
           break;

          //push
          case 2:
```

```
                MouseSimulator.MouseWheel(-1);
                break;

            //pull
            case 4:

              MouseSimulator.MouseWheel(1);
              break;

            //default (do nothing)
            default:
              break;

      }//end acton test for google earth

   }//end google earth is selected app


   //Photomod is selected app
   else if (GetActiveWindowTitle() == "Photomod")
   {

         switch (action)
         {
            //neutral
            case 1:
               break;

            //push
            case 2:
               Console.WriteLine("in switch");
               MouseSimulator.MouseWheel(1);
               break;

            //pull
            case 4:
               Console.WriteLine("in switch");
               MouseSimulator.MouseWheel(-1);
               break;

            //default (do nothing)
            default:
               break;

         }//end Photomod commands

   }//end Photomod test


}//end ReadEmo_GE


/// <summary>
/// Returns a handle to the window with which the user is currently working
/// More or less gets the active windows "ID number"
```

```
        /// </summary>
        /// <returns></returns>
        [DllImport("user32.dll")]
        static extern IntPtr GetForegroundWindow();


        /// <summary>
        /// Gets the text from the active windows title bar
        /// and copies it into a buffer
        /// </summary>
        /// <param name="hWnd"></param>
        /// <param name="text"></param>
        /// <param name="count"></param>
        /// <returns></returns>
        [DllImport("user32.dll")]
        static extern int GetWindowText(IntPtr hWnd, StringBuilder text, int count);

        /// <summary>
        /// Using GetForegroundWindow and GetWindowText
        /// this method gets the active window's title
        /// </summary>
        /// <returns></returns>
        private string GetActiveWindowTitle()
        {
            const int nChars = 256;
            IntPtr handle = IntPtr.Zero;
            StringBuilder Buff = new StringBuilder(nChars);
            handle = GetForegroundWindow();

            if (GetWindowText(handle, Buff, nChars) > 0)
            {
                //active window is photomod
                //stereopair part of code is necessary to detect if stereoviewing extension in Photomod is being
used
                if (Buff.ToString().Contains("PHOTOMOD") || Buff.ToString().Contains("Stereopair"))
                {
                    activeWindow = "Photomod";

                }
                //active window is Google Earth
                else if (Buff.ToString() == "Google Earth")
                {
                    activeWindow = "Google Earth";
                }

                // if neither is the active window return an empty string
                else
                {
                    activeWindow = "";
                }
                return activeWindow;
            }
            return null;
        }

       /// <summary>
```

```
/// Calculates the new mouse position based off of the headset movement
/// </summary>
/// <param name="x"> the movement in the x direction of the headset</param>
/// <param name="y">the movement in the y direction of the headset</param>
/// <param name="elapsed">the amount of time the headset was in motion</param>

 public void TransitionMouseTo(double x, double y, double elapsed)
 {
    double seconds = elapsed;
    double frames = seconds * 10000;

    PointF vector = new PointF();
    PointF mousePos = Cursor.Position;


    vector.X = (float)(x / frames);
    vector.Y = (float)(y / frames);

    if (Math.Abs(vector.X) < 4 && Math.Abs(vector.Y) < 4)
    {
       // Do nothing, vectors smaller than logic are noise
    }
    else
    {
       for (int i = 0; i < frames; i += 1)
       {

          MouseSimulator.X += (int)vector.X/500;
          MouseSimulator.Y -= (int)vector.Y/500;

       }//end set mouse position for loop
     }//end set mouse position
}// end TransitionMouseTo method

#endregion


#region Events


/// <summary>
/// Recognizes a user event, and allows the engine to begin the data acquisition
/// </summary>
/// <param name="sender"> </param>
/// <param name="e"></param>

void engine_UserAdded_Event(object sender, EmoEngineEventArgs e)
{

    // record the user
    userID = (int)e.userId;

    // enable data aquisition for the user.
    engine.DataAcquisitionEnable((uint)userID, true);

    // ask for up to 1 second of buffered data
```

```
    engine.EE_DataSetBufferSizeInSec(1);

}//end engine_UserAdded_Event


/// <summary>
/// Reads which application has been selected from the form window
/// Displays controls for that application in the form window
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void appSelect_SelectedIndexChanged_1(object sender, EventArgs e)
{
    //gets the selected app from Gyro EEG window
    SelectedIndex = appSelect.SelectedIndex;

    //enable the start button
    startButton.Enabled = true;

    //show the options depending on what app is selected
    if (SelectedIndex == 0)
    {
        //clears the options box
        HotkeyBox.Clear();

        //inputs the string below into the options box
        HotkeyBox.AppendText("Push to Zoom Out\r\nPull to Zoom In");

    }//end Google Earth options

    //selected app is Photomod
    else  if (SelectedIndex == 1)
    {
        HotkeyBox.Clear();
        HotkeyBox.AppendText("Push to move the cursor down\r\nPull to move the cursor up");

    }//end Photomod options

}//end appSelect_SelectedIndexChanged_1 method


#endregion

#region Background Thread

/// <summary>
/// Reads that the start button was clicked in the form window
/// Calls the background worker to start executing
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void startButton_Click(object sender, EventArgs e)
{
    //disallow new app selection until stop button is pressed
    appSelect.Enabled = false;
```

```
            //test to see whether the background worker is busy
            if (BGEmoLoop.IsBusy != true)
            {

                if (SelectedIndex == 0)
                {
                    googleEarth = new ApplicationGE();
                }


                //gets the new events from the headset
                KeyEvent = new EmoEngine.EmoStateUpdatedEventHandler(ReadEmo_GE);

                //udates the engine with the new event
                engine.EmoStateUpdated += KeyEvent;

                //start running the background worker
                BGEmoLoop.RunWorkerAsync();

                //enable the stop button
                stopButton.Enabled = true;

                //disable the start button
                startButton.Enabled = false;

                //minimize the Gyro EEG window once the start button is clicked (because its annoying if its
always open)
                this.WindowState = FormWindowState.Minimized;
            }//end test to see if background worker is busy

        }//end startButton_Click method

        /// <summary>
        /// Reads that the stop button was clicked in the form window
        /// Stops the background worker
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void stopButton_Click(object sender, EventArgs e)
        {
            //cancels the background worker, do this first or has problems restarting
            BGEmoLoop.CancelAsync();


            //stop getting updates from the emotive
            engine.EmoStateUpdated -= KeyEvent;

            //disconnect emoengine
            engine.Disconnect();

            //reallow new application selecting
            appSelect.Enabled = true;

            //disable the stop button
            stopButton.Enabled = false;
```

```
            //enable the start button
            startButton.Enabled = true;

        }//end stopButton_Click method


        /// <summary>
        /// The background worker
        /// Connects the engine to the control pannel
        /// Handles the headset movements (to be passed to the TransitionMouseTo method) in a background
thread
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BGEmoLoop_DoWork(object sender, DoWorkEventArgs e)
        {
            // connect to control panel
            engine.RemoteConnect("127.0.0.1", 3008);

                Stopwatch sw = new Stopwatch();
                int gyroX;
                int gyroY;

                engine.ProcessEvents();


                Thread.Sleep(1000);

                BackgroundWorker worker = sender as BackgroundWorker;
                while (true)
                {
                    if (worker.CancellationPending == true)
                    {
                        e.Cancel = true;
                        break;
                    }
                    else
                    {
                        try
                        {
                            sw.Reset();
                            sw.Start();
                            engine.ProcessEvents(1000);

                            sw.Stop();

                            // Get gyro data and put in assigned variable

                            engine.HeadsetGetGyroDelta(0, out gyroX, out gyroY);

                            TransitionMouseTo(gyroX, gyroY, sw.Elapsed.TotalSeconds);
                        }//end try

                        catch (EmoEngineException EmoE)
                        {
                            System.Windows.Forms.MessageBox.Show(EmoE.ToString());
```

```
                }//end catch1

                catch (Exception EmoE)
                {
                    System.Windows.Forms.MessageBox.Show(EmoE.ToString());
                }//end catch2
            }//end worker is not cancelled else statment
        }//end worker while loop


    }//end BGEmoLoop_DoWork method

    #endregion
  }//end partial class form1
}//end Gyro_EEG namespace
```