using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.Xna.Framework;

using Microsoft.Xna.Framework.Audio;

using Microsoft.Xna.Framework.Content;

using Microsoft.Xna.Framework.GamerServices;

using Microsoft.Xna.Framework.Graphics;

using Microsoft.Xna.Framework.Input;

using Microsoft.Xna.Framework.Media;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Timers;

namespace SSVEP\_Summer\_Project

{

/// <summary>

/// This is the main type for your program. It is based off of the XNA game framework.

/// This is becasue this framework has a precisely timed program loop. This is important for SSVEP

/// stimulus flashing.

/// </summary>

public class SSVEPmain : Microsoft.Xna.Framework.Game

{

GraphicsDeviceManager graphics;

SpriteBatch spriteBatch;

// Paradigm

int[] paradigmArray = new int[54] { 9, 2, 1, 6, 11, 15, 17, 13, 10, 8, 14, 3, 7, 16, 18, 4, 12, 5, 15, 11, 4, 13, 7, 5, 12, 3, 9, 1, 2, 16, 17, 8, 18, 6, 14, 10, 1, 10, 14, 9, 7, 3, 8, 17, 2, 5, 18, 13, 12, 16, 4, 15, 11, 6 };

//int[] paradigmArray = new int[36] {9, 2, 1, 6, 11, 15, 17, 13, 10, 8, 14, 3, 7, 16, 18, 4, 12, 5, 15, 11, 4, 13, 7, 5, 12, 3, 9, 1, 2, 16, 17, 8, 18, 6, 14, 10};

int paradigmCntr = 0;

int paradigmIndx = 0;

/// <summary>

/// Main Variables. This is the area where you put all of your variables (i.e.

/// stuff that holds data

/// </summary>

int SCREENWIDTH = 1680;

int SCREENHEIGHT = 1050;

/// <summary>

/// UDP communication with BCI2000

/// </summary>

private Socket sending\_socket;

private IPAddress send\_to\_address;

private IPEndPoint sending\_end\_point;

private Byte[] send\_buffer;

string stateInfo;

// variables that hold stimulus textures and define the color and size.

Texture2D Cross, Solid1, Solid2;

Texture2D One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Ten, Eleven, Twelve, Thirteen, Fourteen, Fifteen, Sixteen;

Color StimulusColor = Color.White;

float stimScale = 1f;

TimeSpan SSVEPTimer;

//bool indicates the start

bool programStart = true;

int pauseCntr=0;

// stimulus tracker variables // you wont need to change these.

int State = 1; int Cntr = 0; int StimType = 1;

int Cntr1 = 0; int StimType1 = 1; int State1 = 1;

int StateFlash = 1; int StimFlash = 0; int CntrFlash = 0;

/// <summary>

/// This is the constructor for the SSVEP program. you generally wont need to

/// change or add anything here

/// </summary>

public SSVEPmain()

{

graphics = new GraphicsDeviceManager(this);

Content.RootDirectory = "Content";

IsMouseVisible = true;

IsFixedTimeStep = true;

TargetElapsedTime = new TimeSpan(10000000L / 60L);

graphics.PreferredBackBufferHeight = SCREENHEIGHT;

graphics.PreferredBackBufferWidth = SCREENWIDTH;

}

/// <summary>

// Put any code that is only ment to run once here.

/// </summary>

protected override void Initialize()

{

SSVEPTimer = TimeSpan.Zero;

// Create a new SpriteBatch, which can be used to draw textures.

spriteBatch = new SpriteBatch(GraphicsDevice);

// TODO: Add your initialization logic here

sending\_socket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

send\_to\_address = IPAddress.Parse("127.0.0.1");

//localhost

sending\_end\_point = new IPEndPoint(send\_to\_address, 20320);

// TODO: use this.Content to load your game content here

//Load Solid Stimuli

Solid1 = Content.Load<Texture2D>(@"SSVEP Textures/Solid1");

Solid2 = Content.Load<Texture2D>(@"SSVEP Textures/Solid2");

Cross = Content.Load<Texture2D>(@"SSVEP Textures/Cross");

One = Content.Load<Texture2D>(@"SSVEP Textures/1");

Two = Content.Load<Texture2D>(@"SSVEP Textures/2");

Three = Content.Load<Texture2D>(@"SSVEP Textures/3");

Four = Content.Load<Texture2D>(@"SSVEP Textures/4");

Five = Content.Load<Texture2D>(@"SSVEP Textures/5");

Six = Content.Load<Texture2D>(@"SSVEP Textures/6");

Seven = Content.Load<Texture2D>(@"SSVEP Textures/7");

Eight = Content.Load<Texture2D>(@"SSVEP Textures/8");

Nine = Content.Load<Texture2D>(@"SSVEP Textures/9");

Ten = Content.Load<Texture2D>(@"SSVEP Textures/10");

Eleven = Content.Load<Texture2D>(@"SSVEP Textures/11");

Twelve = Content.Load<Texture2D>(@"SSVEP Textures/12");

Thirteen = Content.Load<Texture2D>(@"SSVEP Textures/13");

Fourteen = Content.Load<Texture2D>(@"SSVEP Textures/14");

Fifteen = Content.Load<Texture2D>(@"SSVEP Textures/15");

Sixteen = Content.Load<Texture2D>(@"SSVEP Textures/16");

base.Initialize();

}

/// <summary>

// This is the main program loop. Put any code that runs many times here (i.e.

// the flashing of the visual stimuli.

/// </summary>

/// <param name="gameTime">Provides a snapshot of timing values.</param>

protected override void Update(GameTime gameTime)

{

if (programStart)

{

// wait for a bit

pauseCntr++;

if (pauseCntr == (60 \* 10))

{

programStart = false;

StimFlash = 9;

}

}

else

{

//Increment through the randomly permuted trials

paradigmCntr++;

if (paradigmCntr == (2100))

{

paradigmIndx++;

paradigmCntr = 0;

}

//End program when paradigm reaches the end

if (paradigmIndx == 54)

{

this.Exit();

}

//update timer

SSVEPTimer += gameTime.ElapsedGameTime;

// Allows the game to exit

if (Keyboard.GetState().IsKeyDown(Keys.Escape))

this.Exit();

/// Update the state of the stimuli (i.e. flash white or black)

#region UpdateStimuli

//update the counters

Cntr++;

Cntr1++;

CntrFlash++;

//Update the 6 hz stimulus

if (Cntr == 5 && State == 1)

{

StimType = 1;

Cntr = 0;

State = 0;

}

else if (Cntr == 5 && State == 0)

{

StimType = 2;

Cntr = 0;

State = 1;

}

//Update the 10 Hz stimulus

if (Cntr1 == 3 && State == 1)

{

StimType1 = 1;

Cntr1 = 0;

State1 = 0;

}

else if (Cntr1 == 3 && State == 0)

{

StimType1 = 2;

Cntr1 = 0;

State1 = 1;

}

//Class Field

//256x256 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 1)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 1;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//32x32 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 2)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 2;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//128x128 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 3)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 3;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//4x4 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 4)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 4;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//2x2 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 5)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 5;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//32x32 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 6)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 6;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//8x8 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 7)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 7;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//2x2 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 8)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 8;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//128x128 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 9)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 9;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//16x16 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 10)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 10;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//64x64 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 11)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 11;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//256x256 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 12)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 12;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//16x16 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 13)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 13;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//1x1 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 14)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 14;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//8x8 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 15)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 15;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//64x64 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 16)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 16;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//4x4 10Hz SSVEP

if (paradigmArray[paradigmIndx] == 17)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 17;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

//1x1 6Hz SSVEP

if (paradigmArray[paradigmIndx] == 18)

{

if (CntrFlash == 300 && StateFlash == 0)

{

StimFlash = 18;

CntrFlash = 0;

StateFlash = 1;

}

else if (CntrFlash == 1800 && StateFlash == 1)

{

StimFlash = 0;

CntrFlash = 0;

StateFlash = 0;

}

}

// Update BCI2000 state

stateInfo = "StimCode " + Convert.ToString(StimFlash + "\n");

send\_buffer = Encoding.ASCII.GetBytes(stateInfo);

sending\_socket.SendTo(send\_buffer, sending\_end\_point);

#endregion UpdateStimuli

}

base.Update(gameTime);

}

/// <summary>

// This is the main drawing function. It is automatically called after the update function.

// This is where you draw the SSVEP stimuli (squares, shapes, color, textures).

/// </summary>

/// <param name="gameTime">Provides a snapshot of timing values.</param>

protected override void Draw(GameTime gameTime)

{

//Set background color

GraphicsDevice.Clear(Color.Gray);

spriteBatch.Begin();

// spriteBatch.Draw() is the main function you use to draw any type of texture to the screen. Here is how you use it:

// Note: only need to change the parts that are in \*\* \*\*

// spriteBacth.Draw( \*\*your Texture\*\* ,\*\*x-y screen coordinates of where you want to draw it\*\*, null, \*\*Color of texture\*\*, 0, Vector2.Zero, \*\*size of texture\*\*, SpriteEffects.None, 0);

#region DrawStimuli

if (1 == 1)

{

//256x256 6Hz SSVEP

if (StimFlash == 1)

{

if (StimType == 1)

{

spriteBatch.Draw(Fifteen, new Vector2(SCREENWIDTH - ((Fifteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Fifteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Sixteen, new Vector2(SCREENWIDTH - ((Sixteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Sixteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//32x32 6Hz SSVEP

else if (StimFlash == 2)

{

if (StimType == 1)

{

spriteBatch.Draw(Nine, new Vector2(SCREENWIDTH - ((Nine.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Nine.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Ten, new Vector2(SCREENWIDTH - ((Ten.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Ten.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//128x128 10Hz SSVEP

else if (StimFlash == 3)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Thirteen, new Vector2(SCREENWIDTH - ((Thirteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Thirteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Fourteen, new Vector2(SCREENWIDTH - ((Fourteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Fourteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//4x4 6Hz SSVEP

else if (StimFlash == 4)

{

if (StimType == 1)

{

spriteBatch.Draw(Three, new Vector2(SCREENWIDTH - ((Three.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Three.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Four, new Vector2(SCREENWIDTH - ((Four.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Four.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//2x2 10Hz SSVEP

else if (StimFlash == 5)

{

if (StimType1 == 1)

{

spriteBatch.Draw(One, new Vector2(SCREENWIDTH - ((One.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((One.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Two, new Vector2(SCREENWIDTH - ((Two.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Two.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//32x32 10Hz SSVEP

else if (StimFlash == 6)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Nine, new Vector2(SCREENWIDTH - ((Nine.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Nine.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Ten, new Vector2(SCREENWIDTH - ((Ten.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Ten.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//8x8 6Hz SSVEP

else if (StimFlash == 7)

{

if (StimType == 1)

{

spriteBatch.Draw(Five, new Vector2(SCREENWIDTH - ((Five.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Five.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Six, new Vector2(SCREENWIDTH - ((Six.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Six.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//2x2 6Hz SSVEP

else if (StimFlash == 8)

{

if (StimType == 1)

{

spriteBatch.Draw(One, new Vector2(SCREENWIDTH - ((One.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((One.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Two, new Vector2(SCREENWIDTH - ((Two.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Two.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//128x128 6Hz SSVEP

else if (StimFlash == 9)

{

if (StimType == 1)

{

spriteBatch.Draw(Thirteen, new Vector2(SCREENWIDTH - ((Thirteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Thirteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Fourteen, new Vector2(SCREENWIDTH - ((Fourteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Fourteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//16x16 6Hz SSVEP

else if (StimFlash == 10)

{

if (StimType == 1)

{

spriteBatch.Draw(Seven, new Vector2(SCREENWIDTH - ((Seven.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Seven.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Eight, new Vector2(SCREENWIDTH - ((Eight.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Eight.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//64x64 10Hz SSVEP

else if (StimFlash == 11)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Eleven, new Vector2(SCREENWIDTH - ((Eleven.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Eleven.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Twelve, new Vector2(SCREENWIDTH - ((Twelve.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Twelve.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//256x256 10Hz SSVEP

else if (StimFlash == 12)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Fifteen, new Vector2(SCREENWIDTH - ((Fifteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Fifteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Sixteen, new Vector2(SCREENWIDTH - ((Sixteen.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Sixteen.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//16x16 10Hz SSVEP

else if (StimFlash == 13)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Seven, new Vector2(SCREENWIDTH - ((Seven.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Seven.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Eight, new Vector2(SCREENWIDTH - ((Eight.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Eight.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//1x1 10Hz SSVEP

else if (StimFlash == 14)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Solid1, new Vector2(SCREENWIDTH - ((Solid1.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid1.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Solid2, new Vector2(SCREENWIDTH - ((Solid2.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid2.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//8x8 10Hz SSVEP

else if (StimFlash == 15)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Five, new Vector2(SCREENWIDTH - ((Five.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Five.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Six, new Vector2(SCREENWIDTH - ((Six.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Six.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//64x64 6Hz SSVEP

else if (StimFlash == 16)

{

if (StimType == 1)

{

spriteBatch.Draw(Eleven, new Vector2(SCREENWIDTH - ((Eleven.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Eleven.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Twelve, new Vector2(SCREENWIDTH - ((Twelve.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Twelve.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//4x4 10Hz SSVEP

else if (StimFlash == 17)

{

if (StimType1 == 1)

{

spriteBatch.Draw(Three, new Vector2(SCREENWIDTH - ((Three.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Three.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType1 == 2)

{

spriteBatch.Draw(Four, new Vector2(SCREENWIDTH - ((Four.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Four.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//1x1 6Hz SSVEP

else if (StimFlash == 18)

{

if (StimType == 1)

{

spriteBatch.Draw(Solid1, new Vector2(SCREENWIDTH - ((Solid1.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid1.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Solid2, new Vector2(SCREENWIDTH - ((Solid2.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid2.Height + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

// Black Screen

else if (StimFlash == 0)

{

if (StimType == 1)

{

spriteBatch.Draw(Solid1, new Vector2(SCREENWIDTH - ((Solid1.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid1.Height + SCREENHEIGHT) / 2)), null, Color.Black, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

if (StimType == 2)

{

spriteBatch.Draw(Solid2, new Vector2(SCREENWIDTH - ((Solid2.Width + SCREENWIDTH) / 2), SCREENHEIGHT - ((Solid2.Height + SCREENHEIGHT) / 2)), null, Color.Black, 0, Vector2.Zero, stimScale, SpriteEffects.None, 0);

}

}

//Cross

spriteBatch.Draw(Cross, new Vector2(SCREENWIDTH - (((Cross.Width \* 0.08f) + SCREENWIDTH) / 2), SCREENHEIGHT - (((Cross.Height \* 0.08f) + SCREENHEIGHT) / 2)), null, Color.White, 0, Vector2.Zero, 0.08f, SpriteEffects.None, 0);

}

spriteBatch.End();

#endregion DrawStimuli

base.Draw(gameTime);

}

}

}