**PhaseChange.cs**

using UnityEngine;

using System.Collections;

//This method PhaseChange gets data from KinectManager.cs and set up different phase values, which is used by other scripts.

public class PhaseChange : MonoBehaviour {

//PhaseInput1 and PhaseInput2 are used to determine if the Kinect is getting data that switch phases

public int PhaseInput1;

public int PhaseInput2;

//Phase is the actual value that determine the phase

public int Phase;

//Not used now

public float test;

//timer1 and timer2 are used to make sure that phase changes under the condition that the data getting from Kinect are consistent for over a certain amount of time

private float timer1;

private float timer2;

// Use this for initialization

void Start () {

}

// Update is called once per frame

void FixedUpdate() {

//Start the timer when getting data from Kinect

if (PhaseInput1 == 1)

{

timer1 = timer1 + Time.deltaTime;

}

if (PhaseInput2 == 1)

{

timer2 = timer2 + Time.deltaTime;

}

//Dump the timer when no phase changing is happening

if (PhaseInput1 == 0)

{

timer1 = 0;

}

if (PhaseInput2 == 0)

{

timer2 = 0;

}

//Trigger change on value of Phase if the Kinect streams consistent data over 1 second

if (timer1 > 1)

{

Phase = 1;

}

if (timer2 > 1)

{

Phase = 2;

}

}

}

**Transformation.cs**

using UnityEngine;

using System.Collections;

//The method Transformation extract data from PhaseChange.cs and gets data from KinectManager.cs, determine whether it's going to transform from male to female or female to male or stay the same, then stream result into DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs

public class Transformation : MonoBehaviour

{

//MaleGesture and FemaleGesture are used to gather data from KinectManager.cs to determine whether the Kinect is recognizing male gesture or female gesture

public bool MaleGesture;

public bool FemaleGesture;

//Access the two scripts DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs

//I do not remember why I defined two different variables for the same scripts, I am not sure whether if it's necessary or if it's actually doing something

public DoubleSideShaderMorph DoubleSideShaderMorph\_lower;

public DoubleSideShaderMorph DoubleSideShaderMorph\_upper;

public DoubleSideShaderMorph1 DoubleSideShaderMorph1\_lower;

public DoubleSideShaderMorph1 DoubleSideShaderMorph1\_upper;

//Access PhhaseChange.cs

public PhaseChange PhaseChange;

//KeyCheck is used in DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs to control model transformation between male version and female version

private float KeyCheck;

//A set of timers are used to determine whether a pose is hold long enough to trigger the event. Thus important events won't be triggered by random poses

public float timer1;

public float dumptimer1;

public float timer2;

public float dumptimer2;

//Phase is the local variable that matches PhaseChange.cs's Phase value

private int Phase;

// Use this for initialization

void Start()

{

}

// Update is called once per frame

void FixedUpdate()

{

//Extract Phase value from PhaseChange.cs and apply it to local variable Phase

Phase = PhaseChange.Phase;

//Phase 1 events

if (Phase == 1)

{

//If Kinect regonizes a pose as malegesture, timer1 starts and reset dumptimer1

if (MaleGesture)

{

timer1 = timer1 + Time.deltaTime;

dumptimer1 = 0;

}

//If Kinect records a non-malegesture pose the dumptimer starts, this was create to prevent Kinect give random false value which result restarting timer1

if (MaleGesture != true)

{

dumptimer1 = dumptimer1 + Time.deltaTime;

}

//If the the false value is streamed from Kinect for more than half seconds also it does not get dumped cause there is no male gesture being recorded then it's considered a "real" false value, when that happens reset timer1

if (dumptimer1 > 0.5)

{

timer1 = 0;

}

//If the male gesture is held for more than 1 second, then proceed to next step

if (timer1 > 1)

{

//Checking if the model is already in "male state", if yes, do nothing. If no, access DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs to change the KeyCheck value to 1 to initiate transformation from female state to male state

if ((DoubleSideShaderMorph\_lower.blendshape < 100) & (DoubleSideShaderMorph\_lower.blendshape > 0) & (DoubleSideShaderMorph\_upper.blendshape < 100) & (DoubleSideShaderMorph\_upper.blendshape > 0))

{

return;

}

else

{

DoubleSideShaderMorph\_lower.KeyCheck = 1;

DoubleSideShaderMorph\_upper.KeyCheck = 1;

DoubleSideShaderMorph1\_lower.KeyCheck = 1;

DoubleSideShaderMorph1\_upper.KeyCheck = 1;

}

}

//If Kinect regonizes a pose as malegesture, timer2 starts and reset dumptimer2

if (FemaleGesture)

{

timer2 = timer2 + Time.deltaTime;

dumptimer2 = 0;

}

//If Kinect records a non-malegesture pose the dumptimer starts, this was create to prevent Kinect give random false value which result restarting timer2

if (FemaleGesture != true)

{

dumptimer2 = dumptimer2 + Time.deltaTime;

}

//If the the false value is streamed from Kinect for more than half seconds also it does not get dumped cause there is no male gesture being recorded then it's considered a "real" false value, when that happens reset timer2

if (dumptimer2 > 0.5)

{

timer2 = 0;

}

//If the female gesture is held for more than 1 second, then proceed to next step

if (timer2 > 1)

{

//Checking if the model is already in "female state", if yes, do nothing. If no, access DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs to change the KeyCheck value to 1 to initiate transformation from male state to female state

if ((DoubleSideShaderMorph\_lower.blendshape < 100) & (DoubleSideShaderMorph\_lower.blendshape > 0) & (DoubleSideShaderMorph\_upper.blendshape < 100) & (DoubleSideShaderMorph\_upper.blendshape > 0))

{

return;

}

{

DoubleSideShaderMorph\_lower.KeyCheck = 2;

DoubleSideShaderMorph\_upper.KeyCheck = 2;

DoubleSideShaderMorph1\_lower.KeyCheck = 2;

DoubleSideShaderMorph1\_upper.KeyCheck = 2;

}

}

}

//Phase 2 events, these are for testing purpose only now, no pratical functions yet.

if (Phase == 2)

{

DoubleSideShaderMorph1\_lower.KeyCheck = 1;

DoubleSideShaderMorph1\_upper.KeyCheck = 1;

}

}

}

**DoubleSideShaderMorph.cs**

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

//The method of DoubleSideShaderMorph is used along with DoubleSideShaderMorph1 to achieve the function of a female model transforming into a male model

public class DoubleSideShaderMorph : MonoBehaviour {

//KeyCheck is used in DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs to control model transformation between male version and female version, 0 is default/initiate value, 1 is female to male process, 2 is male to female process

public int KeyCheck;

//speedcoefficient determines how fast the metamorphosis happens, speedcoefficient>=0

public float speedcoefficient;

//blendshape is the blendshape value accessed from the model, 0-(0-1)<=blendshape<=100+(0~1)

public float blendshape;

//Access PhhaseChange.cs

public PhaseChange PhaseChange;

//texturemorph is the transparency of the shader, 0<=texturemorph<=1

private float texturemorph;

//Access the shader of the object

private Renderer rend;

//blendshapespeed is the final value of how fast the metamorphosis happens after speedcoefficient is multiplied by 100 times so that the change speed is fast

private float blendshapespeed;

//Access the model's blendshape value

private SkinnedMeshRenderer skinnedMeshRenderer;

//textureprogress is used to calculate the value of texturemorph

private float textureprogress;

//Phase is the value that determine the what event to happen under different values, its value is extracted from PhaseChange.cs

private int Phase;

//Set default values to variables and assign variables to accessed scripts

void Start () {

//Access PhaseChange.cs to extract the value of its Phase to local

Phase = PhaseChange.Phase;

//Assign default value to Keycheck

KeyCheck = 0;

//Access the value of user input changespeed and multiply it by 100 times to make it work to the script

blendshapespeed = 100 \* speedcoefficient;

//Assign rend to access the shader of the object

rend = GetComponent<Renderer>();

//This was used to make sure that the transparent value of the shader was assigned, but is no longer needed

//rend.material.shader = Shader.Find("Ciconia Studio/Double Sided/Transparent/Diffuse Bump");

//Assign skinnedMeshRenderer to access the model's blendshape value

skinnedMeshRenderer = GetComponent<SkinnedMeshRenderer>();

}

// Update is called once per frame

// Check the value of KeyCheck every frame to confirm the transformation (blendshapes and texture)

void FixedUpdate() {

//For debug purpose, allows keyboard input to change the value of KeyCheck

if (Input.GetKeyDown("1"))

{

KeyCheck = 1;

}

if (Input.GetKeyDown("2"))

{

KeyCheck = 2;

}

//The defalut Phase, right now there's only onw phase

if (Phase == 1)

{

//KeyCheck is 1 which means the process is to transform to male model and texture from female ones

if (KeyCheck == 1)

{

//If the blendshape value of the model is over 100, which means the model is already in male state, then reset the KeyCheck and textureprogress

//It should be blendshape >= 0

if (blendshape > 100)

{

KeyCheck = 0;

textureprogress = 0;

}

//If it's not in male state, then this happens

else

{

//Change the blendshape value based on deltaTime from 0 to 100

blendshape = blendshape + blendshapespeed \* Time.deltaTime;

//Assign the value to the model

skinnedMeshRenderer.SetBlendShapeWeight(0, blendshape);

//Change the textureprogress value based on deltaTime

textureprogress = textureprogress + speedcoefficient \* Time.deltaTime;

//Transparency value goes from 1 to 0, transparency value of Female\_Torso\_Texture\_1's shader should be on 1 when the model is in female states, it should be on 0 when the model is in male states

texturemorph = Mathf.Lerp(1f, 0f, textureprogress);

//Assign the value of shader transparency value to the shader

rend.material.SetFloat("\_Transparency", texturemorph);

}

}

//KeyCheck is 2 which means the process is to transform to female model and texture from male ones

if (KeyCheck == 2)

{

//If the blendshape value of the model is below 100, which means the model is already in female state, then reset the KeyCheck and textureprogress

//It should be blendshape <= 0

if (blendshape < 0)

{

KeyCheck = 0;

textureprogress = 0;

}

//If it's not in male state, then this happens

else

{

//Change the blendshape value based on deltaTime from 100 to 0

blendshape = blendshape - blendshapespeed \* Time.deltaTime;

//Assign the value to the model

skinnedMeshRenderer.SetBlendShapeWeight(0, blendshape);

//Change the textureprogress value based on deltaTime

textureprogress = textureprogress + speedcoefficient \* Time.deltaTime;

//Transparency value goes from 0 to 1, transparency value of Female\_Torso\_Texture\_1's shader should be on 1 when the model is in female states, it should be on 0 when the model is in male states

texturemorph = Mathf.Lerp(0f, 1f, textureprogress);

//Assign the value of shader transparency value to the shader

rend.material.SetFloat("\_Transparency", texturemorph);

}

}

}

// not filled in yet, waiting for determined activity to be put into production

if (Phase == 2)

{

}

}

}

**DoubleSideShaderMorph1.cs**

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

//The method of DoubleSideShaderMorph1 is modified from DoubleSideShaderMorph to achieve the function of a male model transforming into a female model

public class DoubleSideShaderMorph1 : MonoBehaviour {

//KeyCheck is used in DoubleSideShaderMorph.cs and DoubleSideShaderMorph1.cs to control model transformation between male version and female version, 0 is default/initiate value, 1 is female to male process, 2 is male to female process

public int KeyCheck;

//speedcoefficient determines how fast the metamorphosis happens, speedcoefficient>=0

public float speedcoefficient;

//blendshape is the blendshape value accessed from the model, 0-(0-1)<=blendshape<=100+(0~1)

public float blendshape;

//Access PhhaseChange.cs

public PhaseChange PhaseChange;

//texturemorph is the transparency of the shader, 0<=texturemorph<=1

private float texturemorph;

//Access the shader of the object

private Renderer rend;

//blendshapespeed is the final value of how fast the metamorphosis happens after speedcoefficient is multiplied by 100 times so that the change speed is fast

private float blendshapespeed;

//Access the model's blendshape value

private SkinnedMeshRenderer skinnedMeshRenderer;

//textureprogress is used to calculate the value of texturemorph

private float textureprogress;

//Phase is the value that determine the what event to happen under different values, its value is extracted from PhaseChange.cs

private int Phase;

//Set default values to variables and assign variables to accessed scripts

void Start () {

//Access PhaseChange.cs to extract the value of its Phase to local

Phase = PhaseChange.Phase;

//Assign default value to Keycheck

KeyCheck = 0;

//Access the value of user input changespeed and multiply it by 100 times to make it work to the script

blendshapespeed = 100 \* speedcoefficient;

//Assign rend to access the shader of the object

rend = GetComponent<Renderer>();

//This was used to make sure that the transparent value of the shader was assigned, but is no longer needed

//rend.material.shader = Shader.Find("Ciconia Studio/Double Sided/Transparent/Diffuse Bump");

//Assign skinnedMeshRenderer to access the model's blendshape value

skinnedMeshRenderer = GetComponent<SkinnedMeshRenderer>();

}

// Update is called once per frame

// Check the value of KeyCheck every frame to confirm the transformation (blendshapes and texture)

void FixedUpdate() {

//For debug purpose, allows keyboard input to change the value of KeyCheck

if (Input.GetKeyDown("1"))

{

KeyCheck = 1;

}

if (Input.GetKeyDown("2"))

{

KeyCheck = 2;

}

//The defalut Phase, right now there's only onw phase

if (Phase == 1)

{

//KeyCheck is 1 which means the process is to transform to male model and texture from female ones

if (KeyCheck == 1)

{

//If the blendshape value of the model is over 100, which means the model is already in male state, then reset the KeyCheck and textureprogress

//It should be blendshape >= 0

if (blendshape > 100)

{

KeyCheck = 0;

textureprogress = 0;

}

//If it's not in male state, then this happens

else

{

//Change the blendshape value based on deltaTime from 0 to 100

blendshape = blendshape + blendshapespeed \* Time.deltaTime;

//Assign the value to the model

skinnedMeshRenderer.SetBlendShapeWeight(0, blendshape);

//Change the textureprogress value based on deltaTime

textureprogress = textureprogress + speedcoefficient \* Time.deltaTime;

//Transparency value goes from 0 to 1, transparency value of Female\_Torso\_Texture\_2's shader should be on 0 when the model is in female states, it should be on 1 when the model is in male states

//This is the part that is different from DoubleSideShaderMorph.cs

texturemorph = Mathf.Lerp(0f, 1f, textureprogress);

//Assign the value of shader transparency value to the shader

rend.material.SetFloat("\_Transparency", texturemorph);

}

}

//KeyCheck is 2 which means the process is to transform to female model and texture from male ones

if (KeyCheck == 2)

{

//If the blendshape value of the model is below 100, which means the model is already in female state, then reset the KeyCheck and textureprogress

//It should be blendshape <= 0

if (blendshape < 0)

{

KeyCheck = 0;

textureprogress = 0;

}

//If it's not in male state, then this happens

else

{

//Change the blendshape value based on deltaTime from 100 to 0

blendshape = blendshape - blendshapespeed \* Time.deltaTime;

//Assign the value to the model

skinnedMeshRenderer.SetBlendShapeWeight(0, blendshape);

//Change the textureprogress value based on deltaTime

textureprogress = textureprogress + speedcoefficient \* Time.deltaTime;

//Transparency value goes from 1 to 0, transparency value of Female\_Torso\_Texture\_2's shader should be on 0 when the model is in female states, it should be on 1 when the model is in male states

//This is the part that is different from DoubleSideShaderMorph.cs

texturemorph = Mathf.Lerp(1f, 0f, textureprogress);

//Assign the value of shader transparency value to the shader

rend.material.SetFloat("\_Transparency", texturemorph);

}

}

}

// not filled in yet, waiting for determined activity to be put into production

if (Phase == 2)

{

}

}

}

**DetectPresence.cs**

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

using Windows.Kinect;

using System;

//The function of the method DetectPresence is to determine whether if there is people standing infront of the Kinect radar camera using the feedback data provided by Kinect

public class DetectPresence : MonoBehaviour

{

//Variables for accessing Kinect

public GameObject BodySrcManager;

private BodySourceManager bodymanager;

private Body[] bodies;

//Variables to determine the distance between the recognized sketlon and Kinect

public float Xdistance;

public float Zdistance;

public double Xposition;

public double Zposition;

public double oldXposition;

public double oldZposition;

//Variables used as timers to determine whether if a skelton is recognized long enough to be register as a presence, this is to prevent false positive

public int Stoptrigger;

public float stoptimer;

public float stoptimer2;

public float stoptimer3;

//The two players were to test the functionality of Kinect tracking two different skeltons, they are not being used right now

public GameObject player1;

public GameObject player2;

//These two variables are to calculate the angle between the person standing infront of Kinect and Kinect, they are accessed by ZoneReaction.cs

public float FloatangleRadians;

public double angleRadians;

// Old method for square zone presence detection, bascially the skelton position needs to be inside of a square area inorder to be register as a presence

// if used, the min and max distances will need to be input based on set-up of kinect in the space

//public float Zone1\_timer;

//public float Zone1\_ZminDistance;

//public float Zone1\_ZmaxDistance;

//public float Zone1\_XminDistance;

//public float Zone1\_XmaxDistance;

//public int Zone1\_trigger;

//public int Zone1\_presence;

//public float Zone2\_timer;

//public float Zone2\_ZminDistance;

//public float Zone2\_ZmaxDistance;

//public float Zone2\_XminDistance;

//public float Zone2\_XmaxDistance;

//public int Zone2\_trigger;

//public int Zone2\_presence;

// Use this for initialization

void Start()

{

//if Kinect does not feed any data, debug log

if (BodySrcManager == null)

{

Debug.Log("Assign Gameobject with body source manager");

}

//if Kinect is live then initiat BodySourceManager script

else

{

bodymanager = BodySrcManager.GetComponent<BodySourceManager>();

}

//!!!I forgot why i put this here...

InvokeRepeating("Presence", 0.5f, 0.5f);

}

// Update is called once per frame

void FixedUpdate()

{

//calculate the angle between the person standing infront of Kinect and Kinect

angleRadians = Math.Atan2(Xdistance, Zdistance) \* 100;

FloatangleRadians = Convert.ToSingle(angleRadians) \* -1;

//if Kinect fails to provide any data, return and check again until it provides data

if (bodymanager == null)

{

return;

}

//bodies is the number of how many skelton is being tracked by Kinect

bodies = bodymanager.GetData();

//if no skelton is being tracked, check again until one is tracked

if (bodies == null)

{

// Old method for square zone presence detection

//Zone1\_trigger = 0;

return;

}

//cycling through all the skeltons tracked by Kinect

foreach (var body in bodies)

{

//if there's no skelton being tracked this happen, right now nothing happens

if (bodies == null)

{

continue;

}

//if there are skeltons being tracked

if (body.IsTracked)

{

//get and calculate position from Kinect data

Zdistance = body.Joints[JointType.Neck].Position.Z;

Xdistance = body.Joints[JointType.Neck].Position.X;

Zposition = Math.Round(Zdistance, 1);

Xposition = Math.Round(Xdistance, 1);

//if person is still more than 2 seconds, stoptimer and stoptimer 3 starts

if ((Zposition == oldZposition) & (Xposition == oldXposition))

{

stoptimer = stoptimer + Time.deltaTime;

stoptimer3 = stoptimer3 + Time.deltaTime;

}

//if person is moving, stoptimer2 starts

if ((Zposition != oldZposition) || (Xposition != oldXposition))

{

stoptimer2 = stoptimer2 + Time.deltaTime;

}

// the following is a temporary fix for that Kinect constantly stream two different sets of data when having 2 players recognized, with this fix the two data sets should not interfere with each other's stop timer reads

if (stoptimer >2)

{

Stoptrigger = 1;

stoptimer = 0;

}

if (stoptimer3 > 0.5)

{

stoptimer2 = 0;

stoptimer3 = 0;

}

if (stoptimer2 > 1)

{

stoptimer = 0;

stoptimer2 = 0;

stoptimer3 = 0;

}

// Old method for square zone presence detection

//if ((Zdistance > Zone1\_ZminDistance) & (Zdistance < Zone1\_ZmaxDistance) & (Xdistance > Zone1\_XminDistance) & (Xdistance < Zone1\_XmaxDistance))

//{

// Zone1\_trigger = 1;

//}

//if ((Zdistance > Zone2\_ZminDistance) & (Zdistance < Zone2\_ZmaxDistance) & (Xdistance > Zone2\_XminDistance) & (Xdistance < Zone2\_XmaxDistance))

//{

// Zone2\_trigger = 2;

//}

//if ((Zdistance < Zone1\_ZminDistance) || (Zdistance > Zone1\_ZmaxDistance))

//{

// Zone1\_trigger = 0;

//}

//if ((Xdistance < Zone1\_XminDistance) || (Xdistance > Zone1\_XmaxDistance))

//{

// Zone1\_trigger = 0;

//}

//if ((Zdistance < Zone2\_ZminDistance) || (Zdistance > Zone2\_ZmaxDistance))

//{

// Zone2\_trigger = 0;

//}

//if ((Xdistance < Zone2\_XminDistance) || (Xdistance > Zone2\_XmaxDistance))

//{

// Zone2\_trigger = 0;

//}

//if (Zone1\_trigger == 1)

//{

// Zone1\_timer = Zone1\_timer + Time.deltaTime;

//}

//if (Zone1\_trigger == 0)

//{

// Zone1\_timer = 0;

//}

//if (Zone1\_timer > 1)

//{

// Zone1\_presence = 1;

//}

//if (Zone1\_timer == 0)

//{

// Zone1\_presence = 0;

//}

//if (Zone1\_timer > 10)

//{

// Zone1\_trigger = 0;

//}

//if (Zone2\_trigger == 2)

//{

// Zone2\_timer = Zone2\_timer + Time.deltaTime;

//}

//if (Zone2\_trigger == 0)

//{

// Zone2\_timer = 0;

//}

//if (Zone2\_timer > 1)

//{

// Zone2\_presence = 1;

//}

//if (Zone2\_timer == 0)

//{

// Zone2\_presence = 0;

//}

//if (Zone2\_timer > 10)

//{

// Zone2\_trigger = 0;

//}

}

}

}

//I forgot what is this for

void Presence()

{

oldXposition = Xposition;

oldZposition = Zposition;

}

}

**ZoneReaction.cs**

using UnityEngine;

using System.Collections;

using System.Collections.Generic;

//This method is to determine what animation to play when Kinect recognized certain gesture

public class ZoneReaction : MonoBehaviour {

//These are all the variables to be accessed from other scripts

public DetectPresence detectpresence;

public Transformation transformation;

public bool malegesture;

public bool femalegesture;

private int Phase;

public int timerTrigger;

private int Stoptrigger;

public PhaseChange PhaseChange;

public GameObject player1;

public GameObject player2;

public GameObject player1translate;

public GameObject player2translate;

//These two are used in old method of presence detectation

private int Zone1\_Trigger;

private int Zone2\_Trigger;

//These are the animation control variables

private Animator anim1;

private Animator anim2;

//Timers used to prevent false positive

private float timer1;

private float timer2;

public float timer;

public float timerb;

// Use this for initialization

void Start () {

//define the animation controller

anim1 = player1.GetComponent<Animator>();

anim2 = player2.GetComponent<Animator>();

}

// Update is called once per frame

void FixedUpdate() {

//this tells the gameobject to face towards the person that is standing still in the kinect sensor range

player1translate.transform.eulerAngles = new Vector3(0, detectpresence.FloatangleRadians, 0);

player2translate.transform.eulerAngles = new Vector3(0, detectpresence.FloatangleRadians, 0);

//Old method for detecting presence

//Zone1\_Trigger = detectpresence.Zone1\_presence;

//Zone2\_Trigger = detectpresence.Zone2\_presence;

//Get data from DetectPresence.cs

Stoptrigger = detectpresence.Stoptrigger;

//Get data from Transformation.cs

malegesture = transformation.MaleGesture;

femalegesture = transformation.FemaleGesture;

//Get data from PhaseChange.cs

Phase = PhaseChange.Phase;

// if viewer holds "male gesture" correctly and over 1 second, then the model plays the "maleReaction"

if (Phase == 1)

{

if (malegesture)

{

timer1 = timer1 + Time.deltaTime;

}

if (malegesture != true)

{

timer1 = 0;

}

if (timer1 > 1)

{

if (timer == 0)

{

anim1.Play("MaleReaction");

anim2.Play("MaleReaction");

timerTrigger = 1;

}

}

if (femalegesture)

{

timer2 = timer2 + Time.deltaTime;

}

if (femalegesture != true)

{

timer2 = 0;

}

if (timer2 > 1)

{

if (timer == 0)

{

anim1.Play("FemaleReaction");

anim2.Play("FemaleReaction");

timerTrigger = 1;

}

}

// player will activate and play "scare" if viewer is standing still, DetectPresence.cs timer is defined

if (Stoptrigger == 1)

{

if (timerb == 0)

{

anim1.Play("Scare\_L");

anim2.Play("Scare\_L");

timerTrigger = 2;

detectpresence.Stoptrigger = 0;

}

}

if (timerTrigger == 1)

{

if (timerb != 0)

{

timer = timer + Time.deltaTime;

timerb = timerb + Time.deltaTime;

}

if (timerb == 0)

{

timer = timer + Time.deltaTime;

}

}

if (timerTrigger == 2)

{

if (timer != 0)

{

timer = timer + Time.deltaTime;

timerb = timerb + Time.deltaTime;

}

if (timer == 0)

{

timerb = timerb + Time.deltaTime;

}

}

if (timerb > 2)

{

timerTrigger = 0;

timerb = 0;

}

// timer causes a pause before the nexct reaction will activate

if (timer > 5)

{

if (timerb != 0)

{

timerTrigger = 2;

timer = 0;

}

if (timerb == 0)

{

timerTrigger = 0;

timer = 0;

}

}

// if once the leap forward animation has played, it is now returned to the 0 0 0

// state once its completed. Otherwise, it would loop, building the rotational radians turn

if (anim1.GetCurrentAnimatorStateInfo(0).IsName("Idle"))

{

//reset model position

player1.transform.position = new Vector3(0, 0, 0);

player2.transform.position = new Vector3(0, 0, 0);

player1.transform.rotation = new Quaternion(0, 0, 0, 0);

player2.transform.rotation = new Quaternion(0, 0, 0, 0);

}

}

if (Phase == 2)

{

if (malegesture)

{

timer1 = timer1 + Time.deltaTime;

}

if (malegesture != true)

{

timer1 = 0;

}

if (timer1 > 1)

{

if (timer == 0)

{

anim1.Play("MaleReaction");

anim2.Play("MaleReaction");

timerTrigger = 1;

}

}

}

}

}