Unity

Generated by Doxygen 1.8.16

Chapter 1

TDF02-145 Unity

1.1 Introduction

This is the repository for arduino and hardware related matters for robot for Autism Spectrum Disorder therapy development funded through HEC TDF. The documentation folder contains all the code for Unity division of HEC funded project TDF 02-145. There are multiple files contained in this folder. This code will run on an Arduino Mega.

1.2 JAVA

This section details the procedure of exporting and running this Unity program in the Android Studio IDE and running it with the JAVA code.

1.2.1 Exporting to JAVA

- Go to File $>$ Build Settings... or press Ctrl + Shift + B
- Select Android and click on Switch Platform
- Enable the Export Project Option. The "Build" option will change to "Export".
- Click on Export.
- Navigate to the TDF02-145 Tablet folder. Click on Select Folder.

1.2.2 Opening in Android Studio

- This folder can now be opening in the Android Studio IDE
- Select "Use Android Studio's SDK"
- MainActivity.java is named UnityPlayerActivity.java

1.2.2.1 Note

Exporting for the first time you may see the following comment
"// GENERATED BY UNITY. REMOVE THIS COMMENT TO PREVENT OVERWRITING WHEN EXPORTING AGAIN".
Delete this line. Otherwise the next time you Export it'll overwrite anything you've written in JAVA.

Exporting after already having exported you'll see a UnityPlayerActivity.NEW in addition to the Unity←↩ PlayerActivity.java you already edited.
You may delete this NEW file.

1.2.2.2 Author

Taha Shaheen, Saifullah, Muhammad Wajahat Qureshi

1.2.2.3 Version

chotuX

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

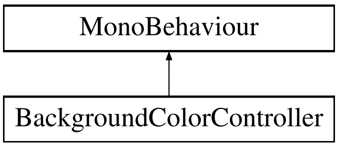
Chapter 5

Class Documentation

5.1 BackgroundColorController Class Reference

Controls the background color.

Inheritance diagram for BackgroundColorController:



Public Member Functions

- void ChangeBackgroundColor (string hexColor)

  Changes value of endColor.

Public Attributes

- Color32 startColor

  Color before transition.
- Color32 endColor

  Color after transition.
- Color32 defaultColor = new Color32(255, 180, 0, 255)

  Contains the default orange color.
- float speed = 1.0F

  Speed with which the color transition occurs.

Private Member Functions

- void Start ()

  Called before the first frame update.
- void Update ()

  Called once per frame.

Private Attributes

- Camera cam

  Camera object.
- float startTime

  Stores time at the start of transition.

5.1.1 Detailed Description

Controls the background color.

Handles facial color and deals with the transition from one to the other

Definition at line 16 of file BackgroundColorController.cs.

5.1.2 Member Function Documentation

5.1.2.1 ChangeBackgroundColor()

```
void BackgroundColorController.ChangeBackgroundColor (
               string hexColor )
```

Changes value of endColor.

- Called from JAVA code
- Receives hexcode of the color as a string
- Breaks it into individual RGB values and converts values into integers
- Updates the value of endColor. Opacity is max (255).

Parameters

hexColor	Color in hexadecimal

Definition at line 92 of file BackgroundColorController.cs.

```
    {

        // fetches current color //
        startColor = Camera.main.backgroundColor;

        // converting from hexadecimal color value to RGB //
        int red, green, blue;
        red = int.Parse(hexColor.Substring(0, 2), System.Globalization.NumberStyles.HexNumber);
        green = int.Parse(hexColor.Substring(2, 2), System.Globalization.NumberStyles.HexNumber);
        blue = int.Parse(hexColor.Substring(4, 2), System.Globalization.NumberStyles.HexNumber);

        endColor = new Color32((byte)red, (byte)green, (byte)blue, 255);

        // reseting start time //
        startTime = Time.time;
    }
```

References endColor, startColor, and startTime.

5.1.2.2 Start()

`void BackgroundColorController.Start ( ) [private]`

Called before the first frame update.

- Runs once
- Fetches a reference to Camera instance attached to Unity object "Main Camera" in the Unity environment
- Saves current time
- Puts defaultColor into startColor and endColor so the first transition is not noticeable

Definition at line 60 of file BackgroundColorController.cs.

```
    {
        cam = GetComponent<Camera>();
        cam.clearFlags = CameraClearFlags.SolidColor;
        startTime = Time.time;
        startColor = defaultColor;
        endColor = defaultColor;
    }
```

References cam, defaultColor, endColor, startColor, and startTime.

5.1.2.3 Update()

`void BackgroundColorController.Update ( ) [private]`

Called once per frame.

- Loops
- Compares time and multiplies it by speed constant. The value of the float t only grows.
- Lerps from startColor to endColor ad infinitum

Definition at line 76 of file BackgroundColorController.cs.

```
    {
        // changes from first color to second color through a gradient //
        float t = (Time.time - startTime) * speed;
        GetComponent<Camera>().backgroundColor = Color32.Lerp(startColor, endColor, t);
    }
```

References endColor, speed, startColor, and startTime.

5.1.3 Member Data Documentation

5.1.3.1 cam

```
Camera BackgroundColorController.cam  [private]
```

Camera object.

Instance of the Camera class through which we view the "scene"

Definition at line 23 of file BackgroundColorController.cs.

Referenced by Start().

5.1.3.2 defaultColor

```
Color32 BackgroundColorController.defaultColor = new Color32(255, 180, 0, 255)
```

Contains the default orange color.

Definition at line 40 of file BackgroundColorController.cs.

Referenced by Start().

5.1.3.3 endColor

```
Color32 BackgroundColorController.endColor
```

Color after transition.

Contains the color the background must be after transition

Definition at line 35 of file BackgroundColorController.cs.

Referenced by ChangeBackgroundColor(), Start(), and Update().

5.1.3.4 speed

```
float BackgroundColorController.speed = 1.0F
```

Speed with which the color transition occurs.

Definition at line 45 of file BackgroundColorController.cs.

Referenced by Update().

5.1.3.5 startColor

```
Color32 BackgroundColorController.startColor
```

Color before transition.

Contains the color the background must be before transition

Definition at line 29 of file BackgroundColorController.cs.

Referenced by ChangeBackgroundColor(), Start(), and Update().

5.1.3.6 startTime

```
float BackgroundColorController.startTime [private]
```

Stores time at the start of transition.

Definition at line 50 of file BackgroundColorController.cs.

Referenced by ChangeBackgroundColor(), Start(), and Update().

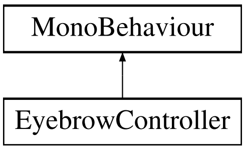
The documentation for this class was generated from the following file:

- BackgroundColorController.cs

5.2 EyebrowController Class Reference

Controls the eyebrows.

Inheritance diagram for EyebrowController:



Public Member Functions

- void SetEmotion (string emotion)

    Sets eyebrow emotion.

Private Member Functions

- void Start ()

    Called before the first frame update.
- void Update ()

    Called once per frame.

Private Attributes

- Animator animator

  Animator object.
- readonly int NO_EYEBROWS = 0

  Universal integer for NO EXPRESSION.
- readonly int ANGRY = 3

  Universal integer for ANGRY.
- readonly int SURPRISED = 2

  Universal integer for SURPRISED.

5.2.1 Detailed Description

Controls the eyebrows.

Handles eyebrows and their animations

Definition at line 15 of file EyebrowController.cs.

5.2.2 Member Function Documentation

5.2.2.1 SetEmotion()

```
void EyebrowController.SetEmotion (
             string emotion )
```

Sets eyebrow emotion.

- Called from JAVA code
- Receives an emotion String
- Runs a switch case that updates the animator's integer named "emotion"
- That triggers an animation

Parameters

emotion	Possible values: HAPPY, SAD, ANGRY, SURPRISED, IDLE, SPEAKING

Definition at line 70 of file EyebrowController.cs.

```
    {
        switch (emotion)
        {
            case "ANGRY":
                animator.SetInteger("emotion", ANGRY);
                break;
            case "SURPRISED":
                animator.SetInteger("emotion", SURPRISED);
                break;
            case "SPEAKING":
                // SPEAKING doesn't change the eyebrows state //
                // This allows for surprised speaking and angry speaking //
                break;
            default:
                // EVerything that doesn't need eyebrows goes here //
                animator.SetInteger("emotion", NO_EYEBROWS);
                break;
        }
    }
```

References ANGRY, animator, NO_EYEBROWS, and SURPRISED.

5.2.2.2 Start()

`void EyebrowController.Start ( )  [private]`

Called before the first frame update.

Runs once.

- Fetches a reference to the Animator instance attached to the Unity object "eyebrows" in the Unity environment

Definition at line 47 of file EyebrowController.cs.

```
    {
        animator = GetComponent<Animator>();
    }
```

References animator.

5.2.2.3 Update()

`void EyebrowController.Update ( )  [private]`

Called once per frame.

Loops.

- Is empty here

Definition at line 57 of file EyebrowController.cs.

```
    {
    }
```

5.2.3 Member Data Documentation

5.2.3.1 ANGRY

```
EyebrowController.ANGRY = 3  [private]
```

Universal integer for ANGRY.

Triggers eyebrows tied to the ANGRY look

Definition at line 40 of file EyebrowController.cs.

Referenced by SetEmotion().

5.2.3.2 animator

```
Animator EyebrowController.animator  [private]
```

Animator object.

Instance of the Animator class. Holds an animator controller which in turn holds animations and handles the switches between them.

Definition at line 22 of file EyebrowController.cs.

Referenced by SetEmotion(), and Start().

5.2.3.3 NO_EYEBROWS

```
readonly int EyebrowController.NO_EYEBROWS = 0  [private]
```

Universal integer for NO EXPRESSION.

Triggers eyebrows to turn off

Definition at line 28 of file EyebrowController.cs.

Referenced by SetEmotion().

5.2.3.4 SURPRISED

```
EyebrowController.SURPRISED = 2  [private]
```

Universal integer for SURPRISED.

Triggers eyebrows tied to the SURPRISED look

Definition at line 40 of file EyebrowController.cs.

Referenced by SetEmotion().

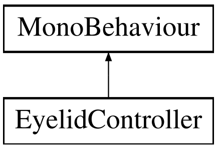
The documentation for this class was generated from the following file:

- EyebrowController.cs

5.3 EyelidController Class Reference

Controls the eyelids.

Inheritance diagram for EyelidController:



Public Member Functions

- void ForcedBlink ()

  Blinks if poked.
- void GoToSleep ()

  Triggers sleep animation.

Private Member Functions

- void Start ()

  Called before the first frame update.
- void Update ()

  Called once per frame.
- void SetPokeFalse ()

  Sets "poke" to FALSE.
- void SetGoToSleepFalse ()

  Sets "poke" to FALSE.

Private Attributes

- Animator animator

  Animator object.

5.3.1 Detailed Description

Controls the eyelids.

Handles eyelids and their animations

Definition at line 15 of file EyelidController.cs.

5.3.2 Member Function Documentation

5.3.2.1 ForcedBlink()

```
void EyelidController.ForcedBlink ( )
```

Blinks if poked.

- The "eyelids" animator controller fed to the Animator instance attached to the Unity object "eyelids" has its intgers poke and goToSleep to FALSE when it starts
- This runs the "blinking" animation on a loop
- When a touch event is registered by the EyePokeHandler class it calls ForcedBlink()
- The "poke" integer of the "eyelids" animator controller gets set to FALSE. This makes sure a "blink" animation can happen. If one is already happening, it gets cancelled.
- The "blink" integer of the "eyelids" animator controller gets set to TRUE trigerring a "poke" animation.
- Sets up a time delayed call (in seconds) to SetPokeFalse()

Definition at line 55 of file EyelidController.cs.

```
{
    animator.SetBool("poke", false);
    animator.SetBool("poke", true);
    //because the forced_poke animation has an exit time we just need to make sure the boolean goes back to false so the animation isn't triggered again //
    Invoke("SetPokeFalse", 1);
}
```

References animator.

Referenced by EyePokeHandler.Update().

5.3.2.2 GoToSleep()

```
void EyelidController.GoToSleep ( )
```

Triggers sleep animation.

- Called from JAVA code.
- Triggers sleep animation by setting "goToSleep" integer of the "eyelids" animator controller to TRUE
- Sets up a time delayed call (in seconds) to SetGoToSleepFalse()

Definition at line 79 of file EyelidController.cs.

```
{
    //This function is called by the Andoird code
    animator.SetBool("goToSleep", true);
    Invoke("SetGoToSleepFalse", 5);
}
```

References animator.

5.3.2.3 SetGoToSleepFalse()

```
void EyelidController.SetGoToSleepFalse ( )  [private]
```

Sets "poke" to FALSE.

Called by GoToSleep(). Sets "goToSleep" integer of "eyelids" animator controller to FALSE

Definition at line 90 of file EyelidController.cs.

```
    {
        animator.SetBool("goToSleep", false);
    }
```

References animator.

5.3.2.4 SetPokeFalse()

```
void EyelidController.SetPokeFalse ( )  [private]
```

Sets "poke" to FALSE.

Called by ForcedBlink(). Sets "poke" integer of "eyelids" animator controller to FALSE

Definition at line 67 of file EyelidController.cs.

```
    {
        animator.SetBool("poke", false);
    }
```

References animator.

5.3.2.5 Start()

```
void EyelidController.Start ( )  [private]
```

Called before the first frame update.

- Runs once
- Fetches a reference to the Animator instance attached to the Unity object "eyelids" in the Unity environment

Definition at line 30 of file EyelidController.cs.

```
    {
        animator = GetComponent<Animator>();
    }
```

References animator.

5.3.2.6 Update()

```
void EyelidController.Update ( )  [private]
```

Called once per frame.

- Loops
- Is empty here

Definition at line 41 of file EyelidController.cs.

```
    {
    }
```

5.3.3 Member Data Documentation

5.3.3.1 animator

```
Animator EyelidController.animator  [private]
```

Animator object.

Instance of the Animator class. Holds an animator controller which in turn holds animations and handles the switches between them.

Definition at line 22 of file EyelidController.cs.

Referenced by ForcedBlink(), GoToSleep(), SetGoToSleepFalse(), SetPokeFalse(), and Start().

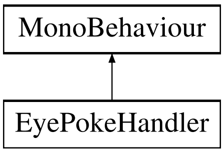
The documentation for this class was generated from the following file:

- EyelidController.cs

5.4 EyePokeHandler Class Reference

Handles poke events.

Inheritance diagram for EyePokeHandler:



Public Attributes

- bool EYE_POKE_ENABLED

  Holds eyepoke enabled state.

Private Member Functions

- void Start ()

  Called before the first frame update.
- void Update ()

  Called once per frame.
- void StopMouthMoving ()

  Sets mouth back to what it was doing.
- void SetEyePokeEnabledState (string state)

  Toggles poking between enabled/or disabled.

Private Attributes

- AudioSource audioSource

  AudioSource object.
- AudioClip audioClip

  AudioClip object.
- MouthController mouthController

  MouthController object.
- EyelidController eyelidController

  EyelidController object.

5.4.1 Detailed Description

Handles poke events.

Handles eyelids, mouth and sound related to when the eyes are poked. Not connected to JAVA code.

Definition at line 15 of file EyePokeHandler.cs.

5.4.2 Member Function Documentation

5.4.2.1 SetEyePokeEnabledState()

```
void EyePokeHandler.SetEyePokeEnabledState (
             string state )  [private]
```

Toggles poking between enabled/or disabled.

- Called from JAVA
- Changes the value of EYE_POKE_ENABLED enabling or disabling a poke event response.

Parameters

state	TRUE or FALSE

Definition at line 113 of file EyePokeHandler.cs.

```
113                                               {
114         switch (state.ToUpper()) {
115             case "TRUE":
116                 EYE_POKE_ENABLED = true;
117                 break;
118             case "FALSE":
119                 EYE_POKE_ENABLED = false;
120                 break;
121         }
122     }
```

References EYE_POKE_ENABLED.

5.4.2.2 Start()

`void EyePokeHandler.Start ( )  [private]`

Called before the first frame update.

- Runs once
- Fetches a reference to AudioSource instance attached to Unity object "ouch_zone" in the Unity environment
- Locates a reference to the MouthController instance already present in the Unity environment
- Locates a reference to the EyelidController instance already present in the Unity environment
- Extracts the audioClip fed to the AudioSource instance of the Unity object "ouch_zone" in the Unity environment
- sets pokes to be enabled

Definition at line 57 of file EyePokeHandler.cs.

```
57                  {
58         audioSource = GetComponent<AudioSource>();
59         mouthController = FindObjectOfType<MouthController>();
60         eyelidController = FindObjectOfType<EyelidController>();
61         audioClip = audioSource.clip;
62 
63         EYE_POKE_ENABLED = true;
64     }
```

References audioClip, audioSource, EYE_POKE_ENABLED, eyelidController, and mouthController.

5.4.2.3 StopMouthMoving()

`void EyePokeHandler.StopMouthMoving ( )  [private]`

Sets mouth back to what it was doing.

Called by Update(). Sets "emotion" integer of the "mouth" animator controller STOP.

Definition at line 102 of file EyePokeHandler.cs.

```
102                            {
103     mouthController.SetSpeaking("STOP");
104     }
```

References mouthController, and MouthController.SetSpeaking().

5.4.2.4 Update()

`void EyePokeHandler.Update ( )  [private]`

Called once per frame.

- Loops
- Detects a touch and checks if poking is enabled.
- If the touch is within the specified poke zone
- an audio plays
- the mouth moves
- the eyes blink
- Sets up a time delayed call (in seconds equal to the length of the audioClip) to StopMouthMoving()

Definition at line 77 of file EyePokeHandler.cs.

```
77                {
78          if ((EYE_POKE_ENABLED) && (Input.touchCount > 0) && (Input.GetTouch(0).phase ==
       TouchPhase.Began)) {
79              Debug.Log("Touching");
80
81              Vector2 worldPoint = Camera.main.ScreenToWorldPoint(Input.GetTouch(0).position);
82              RaycastHit2D hit = Physics2D.Raycast(worldPoint, Vector2.zero);
83
84              if (hit.collider != null) {
85                  Debug.Log(hit.collider.name);
86                  // audio plays //
87                  audioSource.Play();
88                  // mouth moves and stops moving after audio ends //
89                  mouthController.SetSpeaking("START");
90                  Invoke("StopMouthMoving", audioClip.length);
91                  // eyes blink //
92                  eyelidController.ForcedBlink();
93              }
94          }
95      }
```

References audioClip, audioSource, EYE_POKE_ENABLED, eyelidController, EyelidController.ForcedBlink(), mouthController, and MouthController.SetSpeaking().

5.4.3 Member Data Documentation

5.4.3.1 audioClip

`AudioClip EyePokeHandler.audioClip  [private]`

AudioClip object.

Instance of the AudioClip class. Holds an audio clip.

Definition at line 27 of file EyePokeHandler.cs.

Referenced by Start(), and Update().

5.4.3.2 audioSource

```
AudioSource EyePokeHandler.audioSource  [private]
```

AudioSource object.

Instance of the AudioSource class. Holds an AudioClip object and settings on how to run it etc.

Definition at line 21 of file EyePokeHandler.cs.

Referenced by Start(), and Update().

5.4.3.3 EYE_POKE_ENABLED

```
bool EyePokeHandler.EYE_POKE_ENABLED
```

Holds eyepoke enabled state.

Boolean which holds the status on whether or not poking is enabled.

Definition at line 45 of file EyePokeHandler.cs.

Referenced by SetEyePokeEnabledState(), Start(), and Update().

5.4.3.4 eyelidController

```
EyelidController EyePokeHandler.eyelidController  [private]
```

EyelidController object.

Instance of the custom EyelidController class. Handles eyelid animations and transitions.

Definition at line 39 of file EyePokeHandler.cs.

Referenced by Start(), and Update().

5.4.3.5 mouthController

```
MouthController EyePokeHandler.mouthController  [private]
```

MouthController object.

Instance of the custom MouthController class. Handles mouth animations and transitions.

Definition at line 33 of file EyePokeHandler.cs.

Referenced by Start(), StopMouthMoving(), and Update().

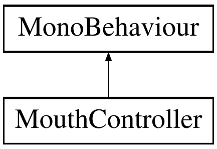
The documentation for this class was generated from the following file:

- EyePokeHandler.cs

5.5 MouthController Class Reference

Controls the mouth.

Inheritance diagram for MouthController:



Public Member Functions

- void SetEmotion (string emotion)

  Sets mouth emotion.
- void SetSpeaking (string speaking)

  Handles mouth movement in SPEAKING emotion.

Private Member Functions

- void Start ()

  Called before the first frame update.
- void Update ()

  Called once per frame.

Private Attributes

- Animator animator

  Animator object.
- readonly int SAD = 1

  Universal integer for SAD.
- readonly int ANGRY = 1

  Universal integer for ANGRY.
- readonly int HAPPY = 0

  Universal integer for HAPPY.
- readonly int SURPRISED = 2

  Universal integer for SURPRISED.
- readonly int SPEAKING = 5

  Universal integer for SPEAKING.
- readonly int IDLE = 4

  Universal integer for IDLE.
- int currentEmotion

  Holds present emotion.

5.5.1 Detailed Description

Controls the mouth.

Handles the mouth and its animations

Definition at line 15 of file MouthController.cs.

5.5.2 Member Function Documentation

5.5.2.1 SetEmotion()

```
void MouthController.SetEmotion (
             string emotion )
```

Sets mouth emotion.

- Called from JAVA code
- Receives an emotion String
- Runs a switch case that updates the animator's integer named "emotion"
- That triggers an animation

Parameters

emotion	Possible values: HAPPY, SAD, ANGRY, SURPRISED, IDLE, SPEAKING

Definition at line 86 of file MouthController.cs.

```
                                            {
        switch (emotion) {
            case "ANGRY":
                animator.SetInteger("emotion", ANGRY);
                break;
            case "SAD":
                animator.SetInteger("emotion", SAD);
                break;
            case "HAPPY":
                animator.SetInteger("emotion", HAPPY);
                break;
            case "SURPRISED":
                animator.SetInteger("emotion", SURPRISED);
                break;
            case "IDLE":
                animator.SetInteger("emotion", IDLE);
                break;
        }
    }
```

References ANGRY, animator, HAPPY, IDLE, SAD, and SURPRISED.

5.5.2.2 SetSpeaking()

```
void MouthController.SetSpeaking (
              string speaking )
```

Handles mouth movement in SPEAKING emotion.

- Called from JAVA code
- Receives a String pertaining to whether the mouth should speaking or not speaking
- Saves the current emotion value unless it is speaking. This prevents a speaking loop.
- Updates the animator's integer named "emotion" to SPEAKING. That triggers an animation.
- Once it receives a STOP instruction from JAVA, it sets the "emotion" value back to what it was before it was set to SPEAKING.

Parameters

speaking	Possible values: START and STOP

Definition at line 116 of file MouthController.cs.

```
116                                              {
117
118         switch (speaking) {
119             case "START":
120                 // save previous mouth state //
121                 if(animator.GetInteger("emotion") != SPEAKING)
122                     currentEmotion = animator.GetInteger("emotion");
123                 // set state to speaking //
124                 animator.SetInteger("emotion", SPEAKING);
125                 break;
126             case "STOP":
127                 // reset mouth state after done speaking //
128                 animator.SetInteger("emotion", currentEmotion);
129                 break;
130         }
131
132     }
```

References animator, currentEmotion, and SPEAKING.

Referenced by EyePokeHandler.StopMouthMoving(), and EyePokeHandler.Update().

5.5.2.3 Start()

```
void MouthController.Start ( )  [private]
```

Called before the first frame update.

- Runs once
- Fetches a reference to the Animator instance attached to the Unity object "mouth" in the Unity environment
- sets default starting emotion

Definition at line 63 of file MouthController.cs.

```
63                 {
64         animator = GetComponent<Animator>();
65         currentEmotion = HAPPY; //just in case scenario
66     }
```

References animator, currentEmotion, and HAPPY.

5.5.2.4 Update()

```
void MouthController.Update ( )  [private]
```

Called once per frame.

- Loops
- Is empty here

Definition at line 74 of file MouthController.cs.

```
74                {
75     }
```

5.5.3 Member Data Documentation

5.5.3.1 ANGRY

```
MouthController.ANGRY = 1  [private]
```

Universal integer for ANGRY.

Triggers mouth tied to the ANGRY look

Definition at line 48 of file MouthController.cs.

Referenced by SetEmotion().

5.5.3.2 animator

```
Animator MouthController.animator  [private]
```

Animator object.

Instance of the Animator class. Holds an animator controller which in turn holds animations and handles the switches between them.

Definition at line 21 of file MouthController.cs.

Referenced by SetEmotion(), SetSpeaking(), and Start().

5.5.3.3 currentEmotion

```
int MouthController.currentEmotion  [private]
```

Holds present emotion.

Contains the present emotional state.

Definition at line 54 of file MouthController.cs.

Referenced by SetSpeaking(), and Start().

5.5.3.4 HAPPY

```
MouthController.HAPPY = 0  [private]
```

Universal integer for HAPPY.

Triggers mouth tied to the HAPPY look

Definition at line 48 of file MouthController.cs.

Referenced by SetEmotion(), and Start().

5.5.3.5 IDLE

```
MouthController.IDLE = 4  [private]
```

Universal integer for IDLE.

Triggers mouth tied to the IDLE look

Definition at line 48 of file MouthController.cs.

Referenced by SetEmotion().

5.5.3.6 SAD

```
MouthController.SAD = 1  [private]
```

Universal integer for SAD.

Triggers mouth tied to the SAD look

Definition at line 48 of file MouthController.cs.

Referenced by SetEmotion().

5.5.3.7 SPEAKING

```
MouthController.SPEAKING = 5  [private]
```

Universal integer for SPEAKING.

Triggers mouth tied to the SPEAKING look

Definition at line 48 of file MouthController.cs.

Referenced by SetSpeaking().

5.5.3.8 SURPRISED

```
MouthController.SURPRISED = 2  [private]
```

Universal integer for SURPRISED.

Triggers mouth tied to the SURPRISED look

Definition at line 48 of file MouthController.cs.

Referenced by SetEmotion().

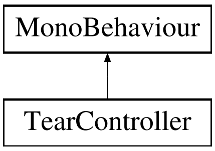
The documentation for this class was generated from the following file:

- MouthController.cs

5.6 TearController Class Reference

Controls the tear.

Inheritance diagram for TearController:



Public Member Functions

- void SetEmotion (string emotion)

  Sets tear state.

Private Member Functions

- void Start ()

  Called before the first frame update.
- void Update ()

  Called once per frame.

Private Attributes

- Animator animator

  Animator object.
- readonly int SAD = 1

  Universal integer for SAD.
- readonly int NOT_SAD = 0

  Universal integer for NO EXPRESSION.

5.6.1 Detailed Description

Controls the tear.

Handles the tear and its animation

Definition at line 15 of file TearController.cs.

5.6.2 Member Function Documentation

5.6.2.1 SetEmotion()

```
void TearController.SetEmotion (
             string emotion )
```

Sets tear state.

- Called from JAVA code
- Receives an emotion String
- Runs a switch case that updates the animator's integer named "emotion"
- That triggers an animation

Parameters

emotion	Possible values: HAPPY, SAD, ANGRY, SURPRISED, IDLE, SPEAKING

Definition at line 63 of file TearController.cs.

```
                                    {
        switch (emotion){

            case "SAD":
                animator.SetInteger("emotion", SAD);
                break;
            case "SPEAKING":
                // SPEAKING doesn't change the tear state //
                // This allows for teary speaking //
                break;
            default:
                animator.SetInteger("emotion", NOT_SAD);
                break;
        }
    }
```

References animator, NOT_SAD, and SAD.

5.6.2.2 Start()

```
void TearController.Start ( )  [private]
```

Called before the first frame update.

- Runs once
- Fetches a reference to the Animator instance attached to the Unity object "tear" in the Unity environment

Definition at line 40 of file TearController.cs.

```
40                {
41          animator = GetComponent<Animator>();
42      }
```

References animator.

5.6.2.3 Update()

```
void TearController.Update ( )  [private]
```

Called once per frame.

- Loops
- Is empty here

Definition at line 50 of file TearController.cs.

```
50                 {
51
52      }
```

5.6.3 Member Data Documentation

5.6.3.1 animator

```
Animator TearController.animator  [private]
```

Animator object.

Instance of the Animator class. Holds an animator controller which in turn holds animations and handles the switches between them.

Definition at line 21 of file TearController.cs.

Referenced by SetEmotion(), and Start().

5.6.3.2 NOT_SAD

```
TearController.NOT_SAD = 0  [private]
```

Universal integer for NO EXPRESSION.

Triggers the tear to turn off

Definition at line 32 of file TearController.cs.

Referenced by SetEmotion().

5.6.3.3 SAD

```
TearController.SAD = 1  [private]
```

Universal integer for SAD.

Triggers objects and animations tied to the SAD look

Definition at line 32 of file TearController.cs.

Referenced by SetEmotion().

The documentation for this class was generated from the following file:

- TearController.cs

Chapter 6

File Documentation

6.1 BackgroundColorController.cs File Reference

Handles background color.

Classes

- class BackgroundColorController

  Controls the background color.

6.1.1 Detailed Description

Handles background color.

Contains code pertaining to facial color and how it changes.

6.2 EyebrowController.cs File Reference

Handles eyebrows.

Classes

- class EyebrowController

  Controls the eyebrows.

6.2.1 Detailed Description

Handles eyebrows.

Contains code pertaining to eyebrows, their animations and their transitions.

6.3 EyelidController.cs File Reference

Handles eyelids.

Classes

- class EyelidController

  Controls the eyelids.

6.3.1 Detailed Description

Handles eyelids.

Contains code pertaining to eyelids, their animations and their transitions.

6.4 EyePokeHandler.cs File Reference

Handles poke events.

Classes

- class EyePokeHandler

  Handles poke events.

6.4.1 Detailed Description

Handles poke events.

Handles everything that has to be done when the eyes are poked.

6.5 MouthController.cs File Reference

Handles the mouth.

Classes

- class MouthController

  Controls the mouth.

6.5.1 Detailed Description

Handles the mouth.

Contains code pertaining to the mouth, its animations and its transitions.

6.6 README.md File Reference

6.7 TearController.cs File Reference

Handles the tear.

Classes

- class TearController
  Controls the tear.

6.7.1 Detailed Description

Handles the tear.

Contains code pertaining to the tear.

Index