

Getting started with the Unity AFFDEX SDK

CS 684/486 Affective Computing, University of San Francisco

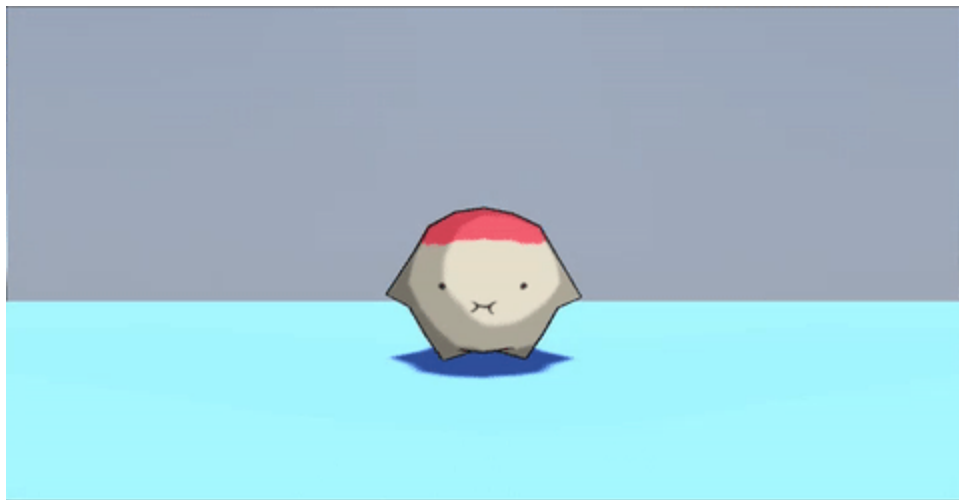
Affectiva's SDKs enable developers to create interactive and exciting emotion-aware apps and digital experiences. Affectiva's SDKs capture and report emotion insights from facial expressions using different devices, analyzing faces obtained from the device camera, a video, or a single image — processed on-device and in real time.

Affectiva has SDKs available for iOS, Android, Web, etc. However this tutorial will go over how to set up a starter project to interact with the Affectiva SDK, using the Affectiva Unity SDK in the Unity software. The tutorial also assumes that you have no previous knowledge of Game Development. There are two main languages used to develop games for the Unity platform, Javascript and C#, this tutorial will be using C#.

This tutorial will go over how to create a game with a character that mimics your emotions. The character will smile when you smile, frown when you frown, and have an idle face the rest of the time. After completing the tutorial you will feel comfortable enough to extend your emotion-aware game to include other emotions and even other characters!

Game Demo

The character is smiling when I smile and is frowning when I frown



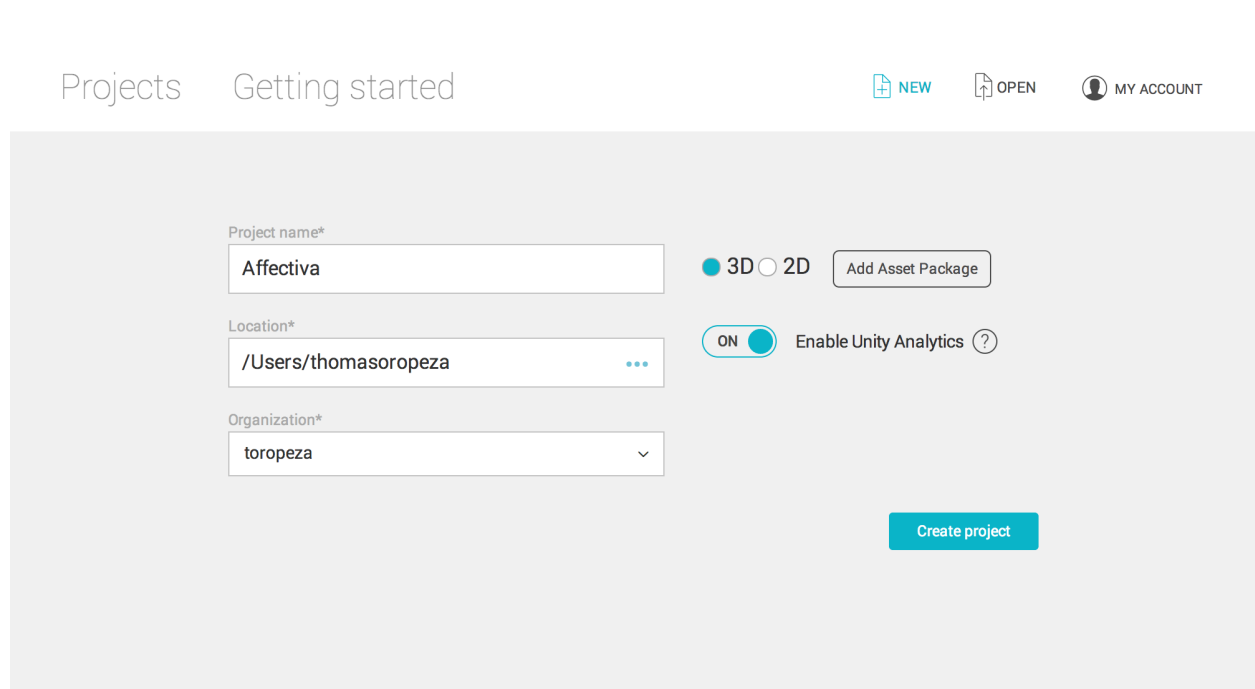
Installing Unity

Unity is free to download and use on Windows and MacOS. Follow the instructions for downloading and installing Unity on your computer. You will also be asked to create a Unity account once Unity is installed.

<https://store.unity.com/download?ref=personal>

Create a new Unity Project

Once logged in you will be brought to the Unity project page. Click on “New” to create a new project with 3D selected.



Projects Getting started

NEW OPEN MY ACCOUNT

Project name*
Affectiva

Location*
/Users/thomasoropeza

Organization*
toropeza

3D 2D Add Asset Package

ON Enable Unity Analytics ?

Create project

This will create a new project and open up Unity's Development Environment.

Importing the Affectiva SDK

Visit the Affectiva Unity page and look for the download link. This will download a Unity package file to your computer.

<https://knowledge.affectiva.com/v3.2/docs/getting-started-with-the-emotion-sdk-for-unity>

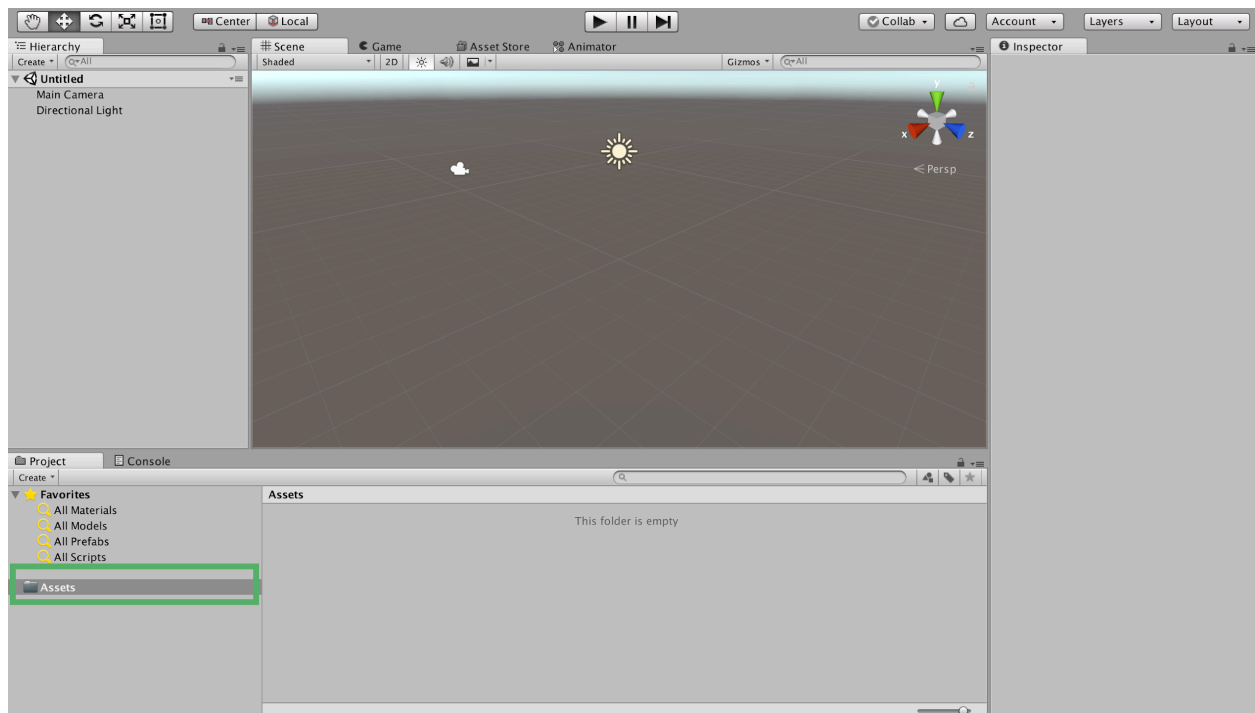
Getting started

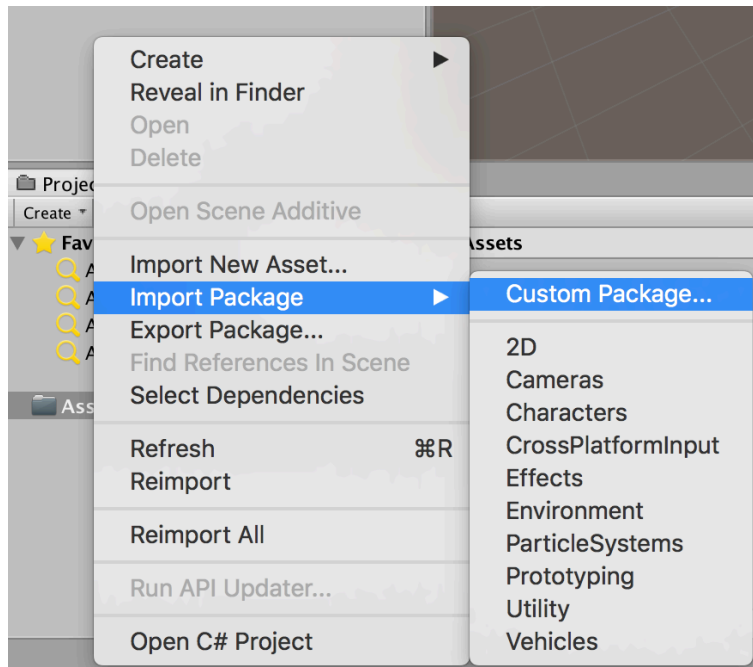
We package our plugin as an asset, like those you can buy on Unity's Asset Store. You can download it from below.

Architecture	Version	Size	
OSX i386/x86_64, Windows x86/x86_64, Android ARM7/x86, iOS AR- M64/ARM7	2.3 (release notes)	207 MB	Download

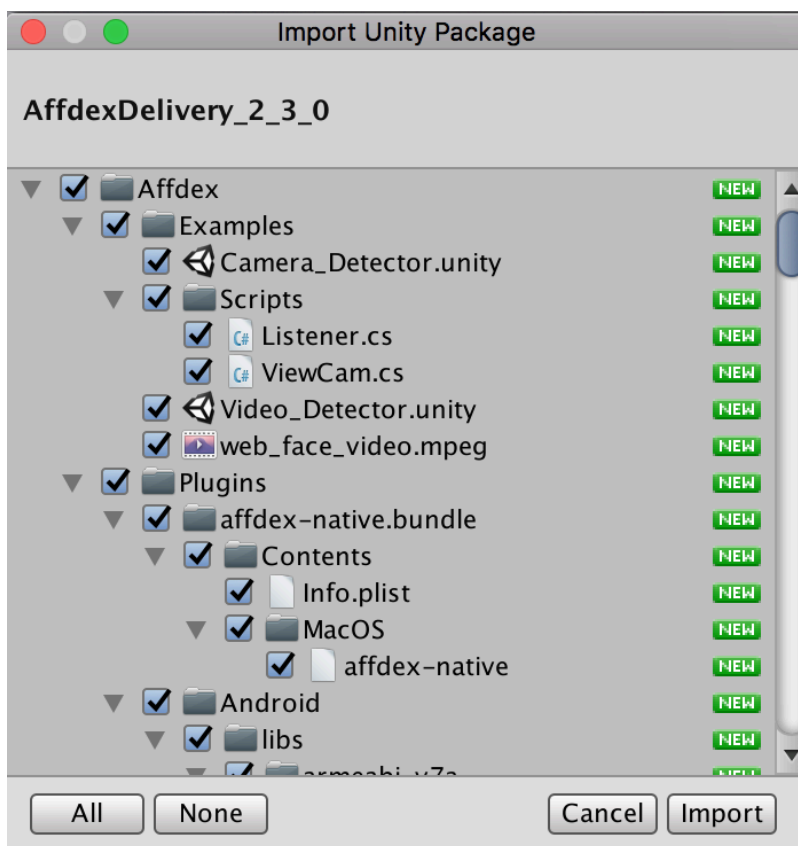
Once you have the Unity package file, navigate to the Assets folder (located at the bottom left) in the Unity development environment and right click it to select Import Package → Custom Package.

*If you cannot find the Assets folder, you can open it by clicking Window → Project in your nav bar.





Then it will ask you which sub-packages you would like to import. By default the necessary ones will already be selected so just click “import”.



Saving Scenes and Projects

You will need to save your scene *and* your project. Ctrl-S will only save your scene. You will *also* need to save your project. Please put your name in your scene name so that we can identify your work easily when grading.

Setting up the Game Scene

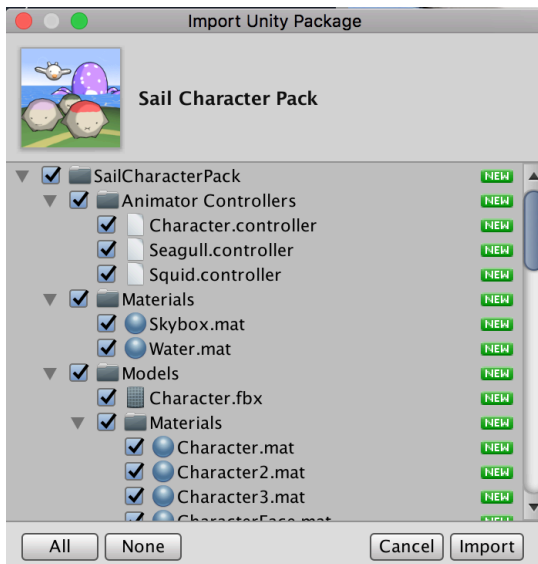
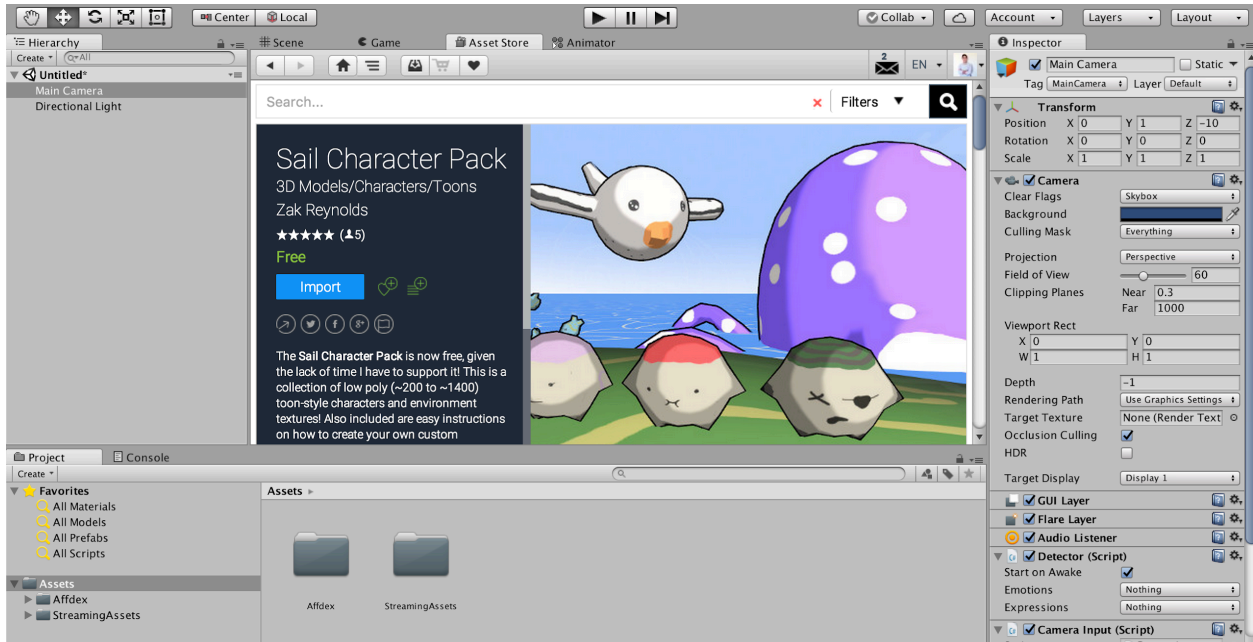
A scene is a 3D overview of the Game Objects in our game. By default when we created a new project, the Main Camera and Directional Light Game Objects are added. We need to add in our character, a plane, and a backdrop.

Unity allows users to create models and put them on the “Asset Store”. Any character found on the Asset Store would work however it is important to note that changing the emotions on the character will be different for each model. We will be using the Sail Character Pack for this tutorial.

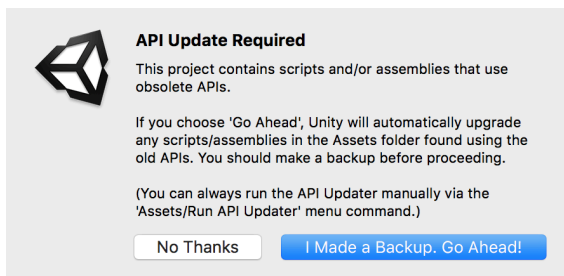
Visit <http://u3d.as/88o>.

Or click on the Asset Store tab and search for ‘Sail Character Pack’.

This will bring you to the Sail Character pack in the Asset Store. There you will be prompted to open the pack in Unity and once opened in Unity you should be able to click import.



Click "Import"



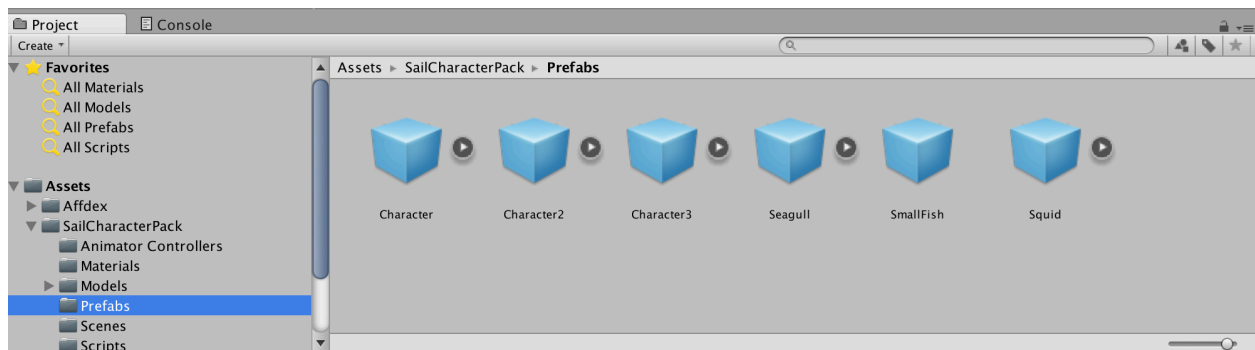
A screen may pop up asking you to update. If so, click "Go Ahead!"

This will add the Sail Character Pack to our project. You will see all that is provided with it in the package in the Assets window.

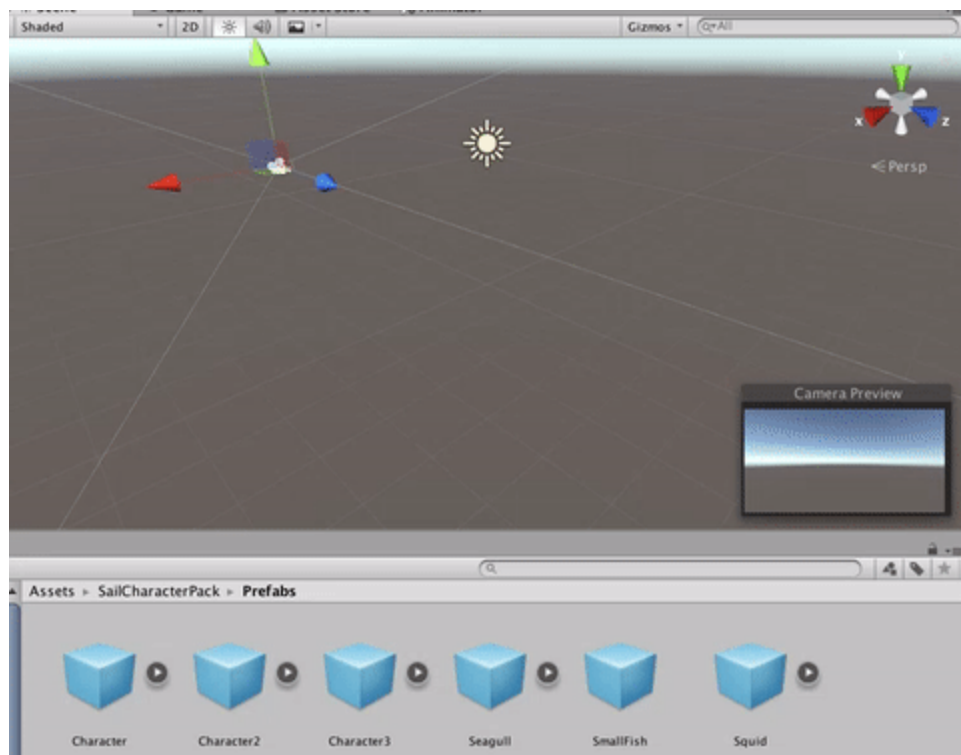
Now let's add a Sailor to our Scene!

Navigate back to the Scene tab, which can be found near the top of the Unity Development Environment in the same navigation bar as the Asset Store.

Once back in the Scene tab, navigate through the Sail Character Pack to the “Prefabs” directory located under Assets → Sail Character Pack → Prefabs.



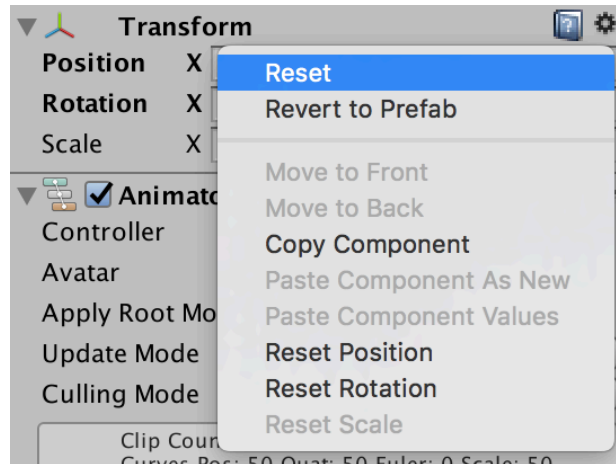
Click and drag the “Character” prefab anywhere onto the Scene like so.



Prefabs

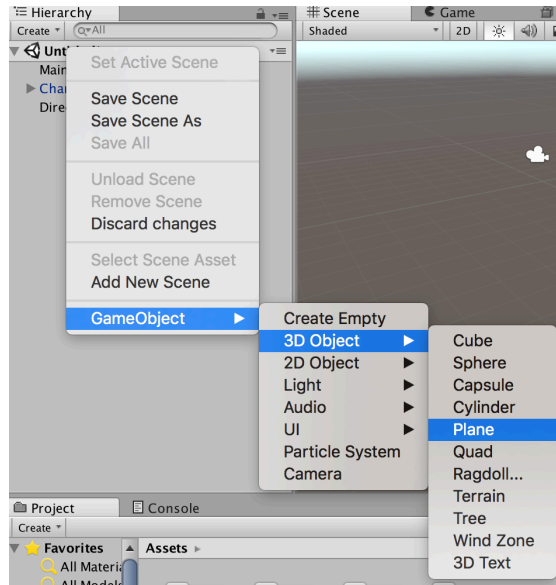
Prefabs are Gameobjects that wrap different properties. In this case we are dragging over a Character prefab from the SailCharacterPack. This Character will have materials, animation, body components, and facial features pre-built into it that we can control. Later we will be referencing the Character's mouth to make the Sailor frown or smile!

Now the Character will be added to the Game Object hierarchy and our Scene. To center our Character, select the character in the Hierarchy panel and find the "Transform" section in the Inspector. Click on the cogwheel settings icon and select "Reset"

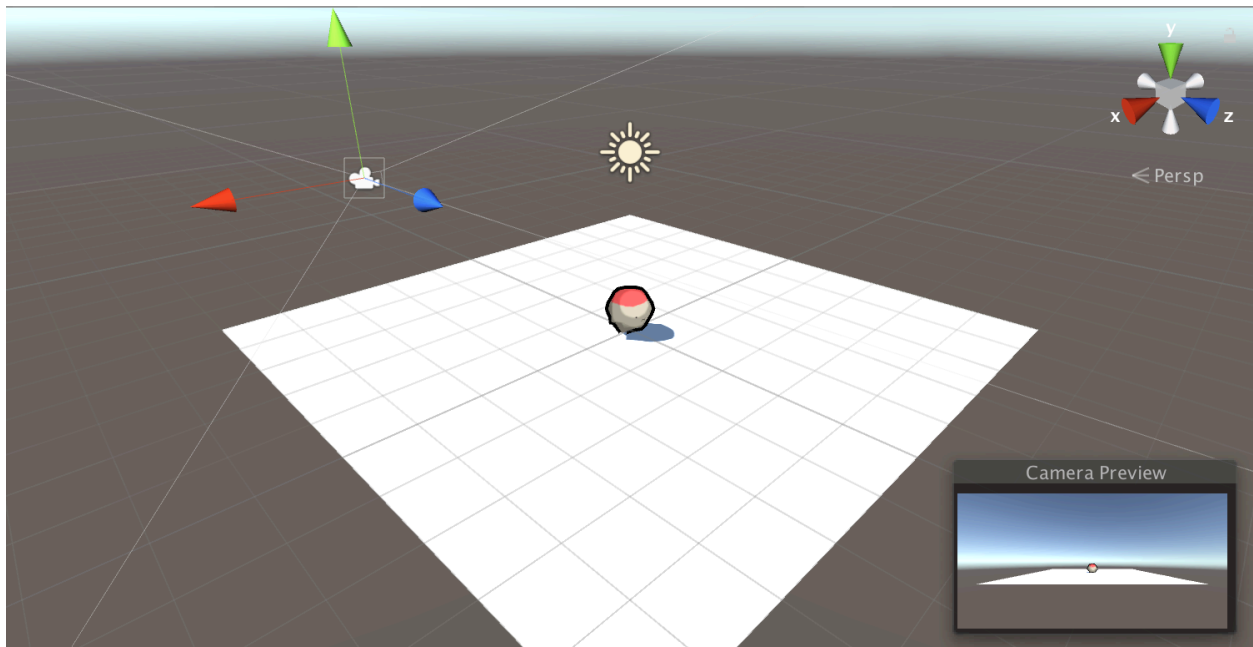


The character will now be centered in the scene.

At this point the character is floating in space. To add a ground you can add a plane Gameobject. To do this right click on the scene name in the hierarchy window (by default it is called untitled until you save the scene) and click on Gameobject → 3D Object → Plane. This will add a plane at the origin on the scene. If the plane is not at the origin you can click on "Reset" in the Transform settings section of the Inspector in the same way we centered the Character.



Your scene should look similar to the following at this point.

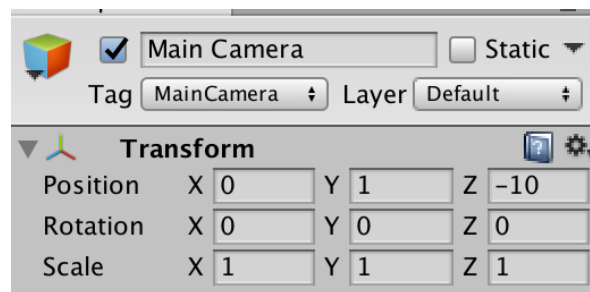


If you look closely you might notice that the camera is facing the back of the Character. We need it to point at the front of the character to see its facial features. The camera will be the user's perspective when the game is played.

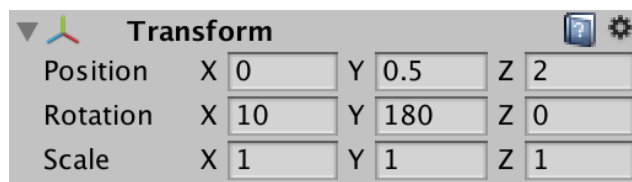
Select the Main Camera Gameobject in the hierarchy and you should see a small window at the bottom right of your scene view that gives a preview of what the camera sees. Let's fix the camera to face the front of the character and move it a little closer so we can see the facial expressions a little more clearly.

*If you cannot find the Hierarchy view, click on Window → Hierarchy to open it

With the Main Camera selected, take a look at the transform section in the inspector. The transform of a Gameobject describes where the object is located in the scene and how it is sized.

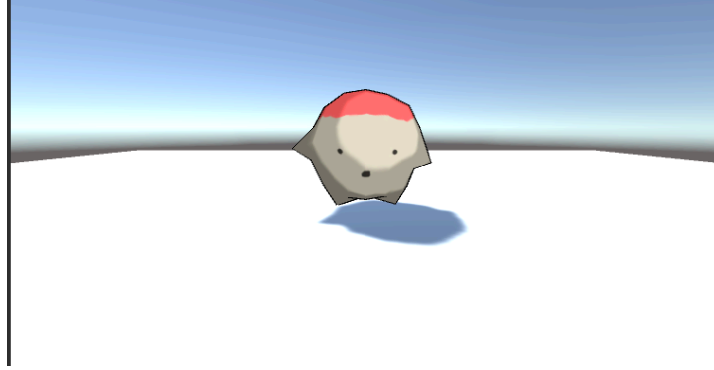


Set the transform of the Main Camera to the following properties. This should give you a good perspective of the character.



You can play around with the transform of the Main Camera and the Directional Light Gameobjects in the scene to get the look that you would like.

Now press the play button at the top of Unity and this will run the game and you should see the character's face from the perspective of the Main Camera.



Stop the play before moving onto the next section.

Configuring the Affectiva SDK

Now that the Affectiva SDK is integrated into our Unity project, we need to configure it so that it uses the computer's camera and send facial data to us.

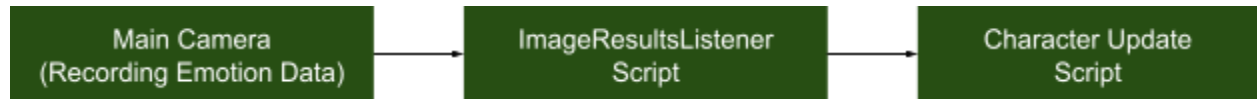
Configuring the Affectiva SDK for our game takes three steps.

1. Adding the Affectiva Scripts to the "Main Camera" Gameobject
2. Write our own script to parse the facial data
3. Write a script to update the character's face

Scripts

A Unity game is composed of many different Gameobjects. These can be characters, cameras, and props. They are typically by default motionless and inanimate. Scripts are what control their behavior and how they should interact with each other.

The control flow of our game is the following

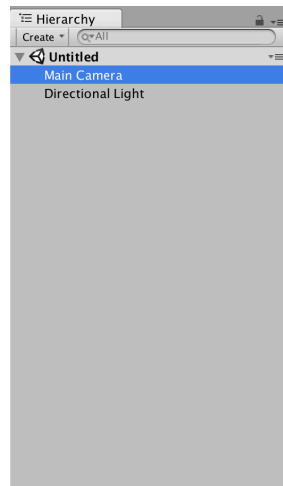


The Main Camera of our game will be connected to the Affectiva SDK, which parses images of the player's face. This parsed data will be sent to the ImageResultsListener script which we will implement. This script will be where we will save the different levels of the user's emotions. Then we will write a Script for updating the Characters face depending on the most dominant emotion on the user. For example, if the user is mostly happy, then the character's face should smile, and if the user makes a sad face the character will too.

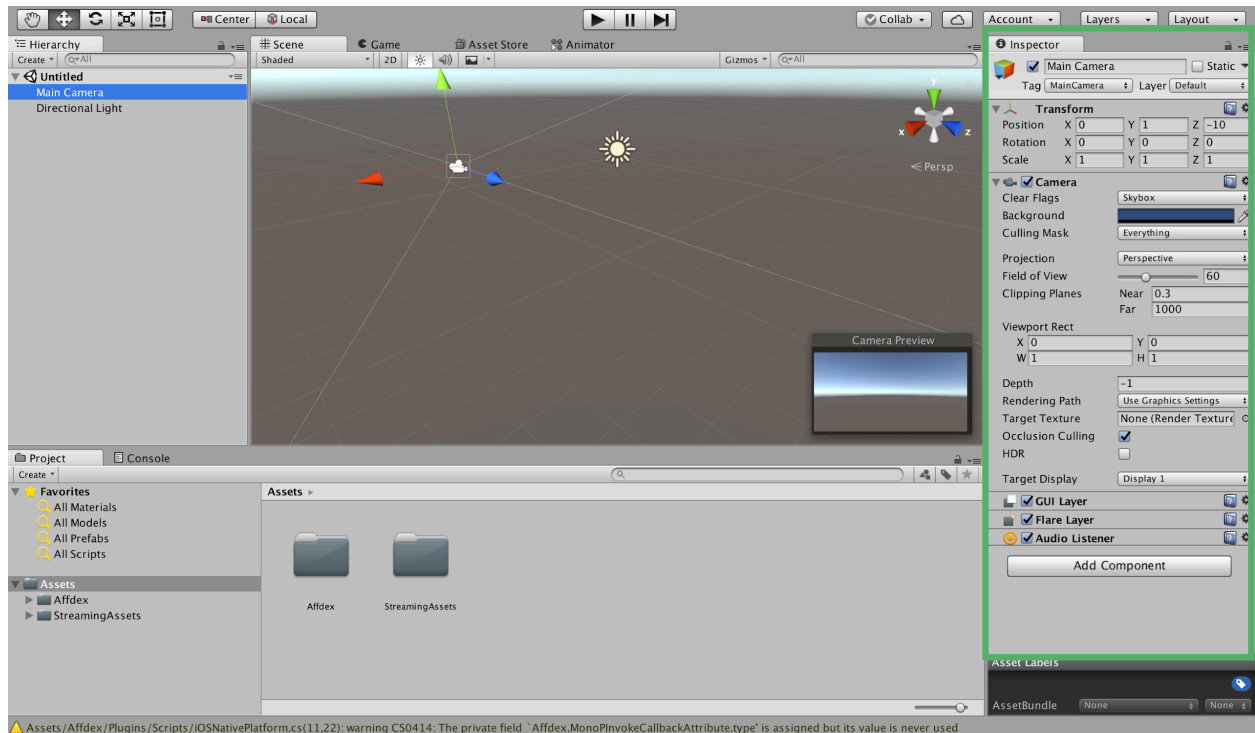
Adding the Affectiva Scripts to the "Main Camera" Gameobject

Adding the Affectiva scripts to the Main Camera Gameobject is very simple. Find the Main Camera Gameobject in the Unity Hierarchy view.

*If you cannot find the Hierarchy view, click on Window → Hierarchy to open it



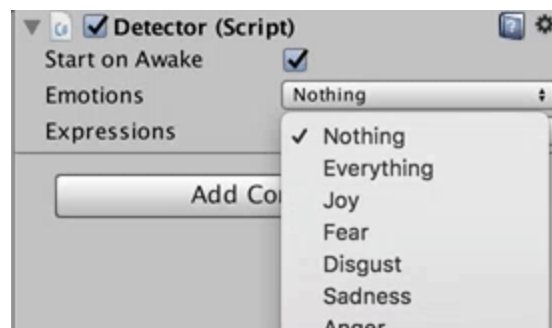
Selecting the Main Camera will also display its many properties in the Inspector panel.



With the Main Camera still selected, select the “Add Component” button at the bottom of the “Inspector” panel.

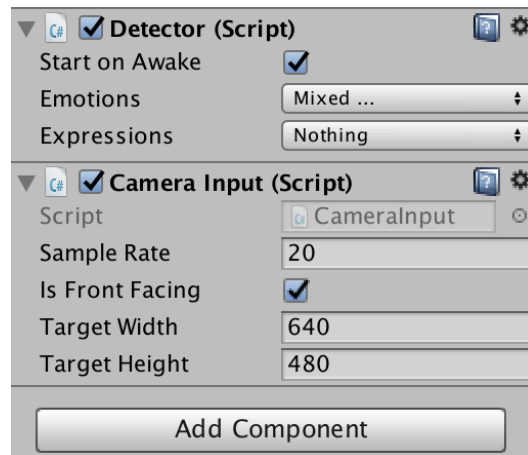
Then click on Scripts → Affdex → Detector. This will add the Detector script to your Main Camera.

This Detector can be seen at the bottom of the inspector while the Main Camera is selected. There is a dropdown named Emotions. Be sure to check Joy and Sadness.



Now add the Camera Input Script in the same way. It is located under Scripts → Affdex → Camera Input.

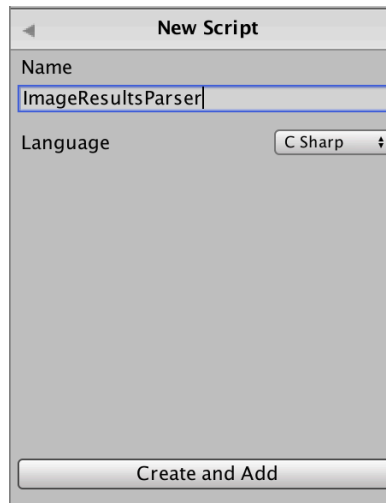
Your Main Camera should now have the two scripts located at the bottom of the inspector panel.



A reminder to save your scene *and* your project at this stage and as you continue.

Writing our own script to parse the facial data

With the character selected in the hierarchy window, click “Add Component” in the inspector. Then click on “New Script” with the language set to C# and name it ImageResultsParser



This will create a new script associated with the Character GameObject. To edit the script click on its cogwheel settings icon and click edit script. This will open it in Unity’s default editor MonoDevelop.

Copy and paste the following code into the script.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Affdex;

//Parsed image data will be sent to the ImageResultsListener methods
public class ImageResultsParser : ImageResultsListener {

    //emotion levels
    public float joyLevel;
    public float sadnessLevel;

    public override void onFaceFound (float timestamp, int faceId)
    {

    }

    //called when the Affectiva SDK loses a facial detection
    public override void onFaceLost (float timestamp, int faceId)

    {
        //set emotion levels to 0 (will cause character to be Idle)
        joyLevel = 0;
        sadnessLevel = 0;
    }

    //called every second whether there is a face detected or not
    public override void onImageResults (Dictionary<int, Face> faces)
    {
        if (faces.Count > 0){
            //set emotion levels
            faces [0].Emotions.TryGetValue (Emotions.Joy, out joyLevel);
            faces [0].Emotions.TryGetValue (Emotions.Sadness, out sadnessLevel);
        }
    }
}
```

This script extends the Interface ImageResultsListener. Doing so provides three methods: onFaceFound(), onFaceLost(), and onImageResults().

The TryGetValue() methods will try to get the specified emotion on the first face found and will output it to the variable provided with the out keyword.

Writing a script to update the character's face

Now that we have variables with the user's constant emotion levels, we can refer to them in order to set the Character's emotions properly.

Switch back to the Scene view in Unity and select the Character Gameobject again. You will notice that there are two scripts associated with this character, one of them being the one that we created and another called "Facial Expressions". This script belongs to the Character prefab. It is already setup to control the character's emotions, but we need to edit it to set the emotions based on the user's current emotions.

Open the Facial Expressions script by clicking on the cogwheel settings icon then "Edit Script". Examine the code that manipulates the character's facial expressions. You will want to use similar code for your own control over the facial expressions later.

Copy and paste the following code into the script file.

```
using UnityEngine;
using System.Collections;

public class FacialExpressions : MonoBehaviour {

    //Script containing levels of emotions
    ImageResultsParser userEmotions;

    //player Transform used to obtain reference to UserEmotions script
    Transform player;

    //Used to change the face from smiling to frowning
    public Renderer faceRenderer;

    private Material faceMaterial;
    private Vector2 uvOffset;

    // Initialization and Setting references
    void Start () {
        player = GameObject.FindGameobjectWithTag ("Sailor").transform;
        userEmotions = player.GetComponent<ImageResultsParser> ();
        uvOffset = Vector2.zero;
        faceMaterial = faceRenderer.materials[1];
    }
}
```



```

// Setting the user's dominant emotion every frame
void Update () {
    // TODO for Question 2
}

//sets the Character's emotion to Idle (Emotionless)
void setIdle(){
    // TODO for Question 2
}

//sets the Character's emotion to Joyful (Smiling)
void setJoyful() {
    // TODO for Question 2
}

//sets the Character's emotion to Sad (Frowning)
void setSad() {
    // TODO for Question 2
}
}

```

The Start() method

```

void Start () {
    player = GameObject.FindGameObjectWithTag ("Sailor").transform;
    userEmotions = player.GetComponent<ImageResultsParser> ();
    uvOffset = Vector2.zero;
    faceMaterial = faceRenderer.materials[1];
}

```

We obtain a reference to the player transform via GameObject Tag (which we will add later). This reference is how we will obtain the ImageResultsParser script with the variables containing the user's emotion levels.

It is important to note that there is a public variable in this class.

```

public Renderer faceRenderer;

```

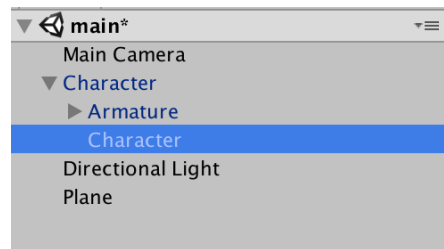
This faceRenderer variable is how we obtain a reference to the mouth of the character. However we are not setting the reference for it in the Start() method.

This reference needs to be set in Unity. Open the Unity Environment and click on the Character Gameobject in the hierarchy window.

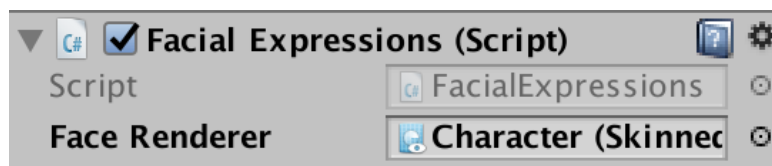
You will notice that our script has a new field FaceRenderer : None (Renderer)



We need to connect the Sailor Character's Face Renderer to our script. This can be done by simply selecting the renderer in the hierarchy window and dragging it directly into this field. The Face Renderer also named "Character" and it is nested inside of the Character Gameobject.

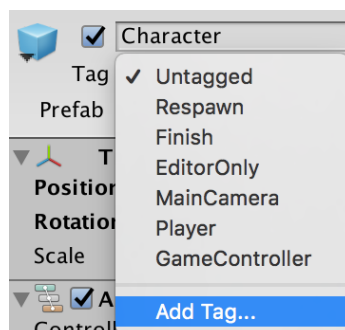


Select it and drag it straight into the Face Renderer field of the script



Lastly we need to add the Tag which will help us reference the Gameobject from the script.

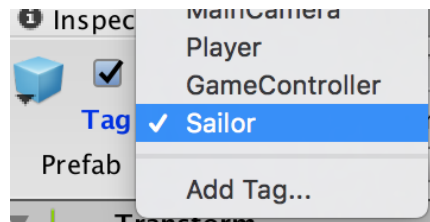
Select the Character Gameobject, and click on the Tag dropdown, then "Add Tag".



Then create a new tag and be sure to name it “Sailor”. This needs to be spelled correctly or it will not be found in the Script.



Next click on the Character GameObject in the Hierarchy window again and set the tag to “Sailor”.



Question 2 - Finishing the Code

a. Write the Update() method.

```
void Update () {  
    // TODO for Question 2  
}
```

The Update() method finds the dominant emotion and calls the appropriate method to set the emotion on the character. Therefore, you must get the emotions of joy and sadness from the public variables in the ImageResultsParser class.

If the dominant emotion is less than some threshold, such as 40 (it is on a scale from 0-100), then it is not strong enough so just set the character to be idle.

b. The Update() method will need to call setIdle(), setJoyful(), and setSad() at the appropriate times.

```
//sets the Character's emotion to Idle (Emotionless)  
void setIdle(){  
    // TODO for Question 2
```

```
}

//sets the Character's emotion to Joyful (Smiling)
void setJoyful() {
    // TODO for Question 2
}

//sets the Character's emotion to Sad (Frowning)
void setSad() {
    // TODO for Question 2
}
```

You will need to manipulate the `uvOffset` which is the offset of the mouth to make it smile or frown. The `Start()` method is called when the game begins so we just set the `uvOffset` to 0 which is the same as making the character idle. The `uvOffset` is an instance of the `Vector2` class ([see the Unity docs for Vector2](#)). In physics, a vector is a representation of both quantity and direction. The `Vector2` class represents 2D positions and vectors so that the material of the character's skin can be moved in the direction and quantity you want to make it smile or frown or whatever you want!

Testing the Game

Once you finish question 2, try clicking on the Play button at the top of Unity. The game will start and the camera will be connected. Make sure your face is positioned properly in front of the camera then try smiling and frowning. The character should mimic you!

Further Reading

For more information about Unity or Affectiva's Unity SDK refer to the following links:

[Unity Tutorials](#)

[Unity Affdex Documentation](#)

[Unity's Asset Store](#)

[Unity Scripting API](#)

Tutorial Written by **Thomas Oropeza** and **Beste Filiz Yuksel**