FACE AND GAZE TRACKING

Corso Realtà Virtuale 2020/2021

eleonora.chitti@unimi.it

WHAT IS EYE TRACKING

The gaze tracking indicates the detection of the eye movements, observing the position of the eyes' pupils in relation to the face position and rotation.

In computer science the gaze tracking can be used to interact with a digital system in real time.

- Gaze Tracking applications:
  - Commercial and entertainment (e.g., marketing studies)
  - Rehabilitation support or assistive technologies (e.g., AAC applications, gaze pointers interaction applications)
  - Learning and education (e.g. [1])

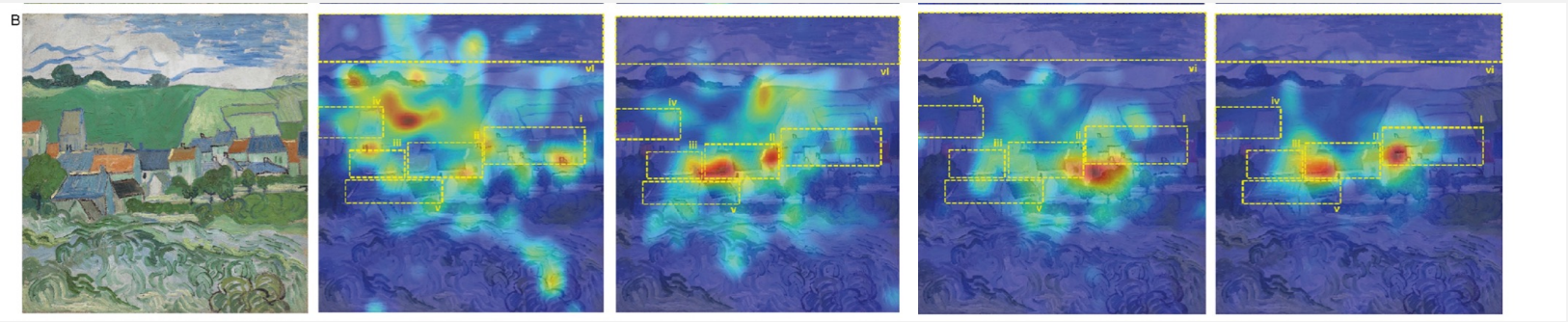


Image from [1]

Heat maps of 2 sessions per group of observing a painting (View of Auvers), first group of Children vs second group of Adults

HOW EYE TRACKING WORKS

[2] describes the eye tracking as a technique used to measure where user's eyes are focusing when the user is looking at something.

From eye tracking techniques you can extract:

- Fixation point: the direction of the gaze, i.e. where the user is looking at
- Fixation time: the duration of looking at a point
- Scan-path: Sequence of fixation points, i.e., in what order and where the eyes are moving.

There are two types of gaze tracking techniques:

- Screen fixation gaze tracking
- Wearable glasses gaze tracking

HOW EYE TRACKING WORKS

The eyes position and movements can be tracked with the PCCR - Pupil Center Corneal Reflection method, [2] [3] [4] [5] that works by tracking the eyes with near-infrared light / light source that is reflected in the cornea, tracking the light and its reflection to record how eyes are moving.

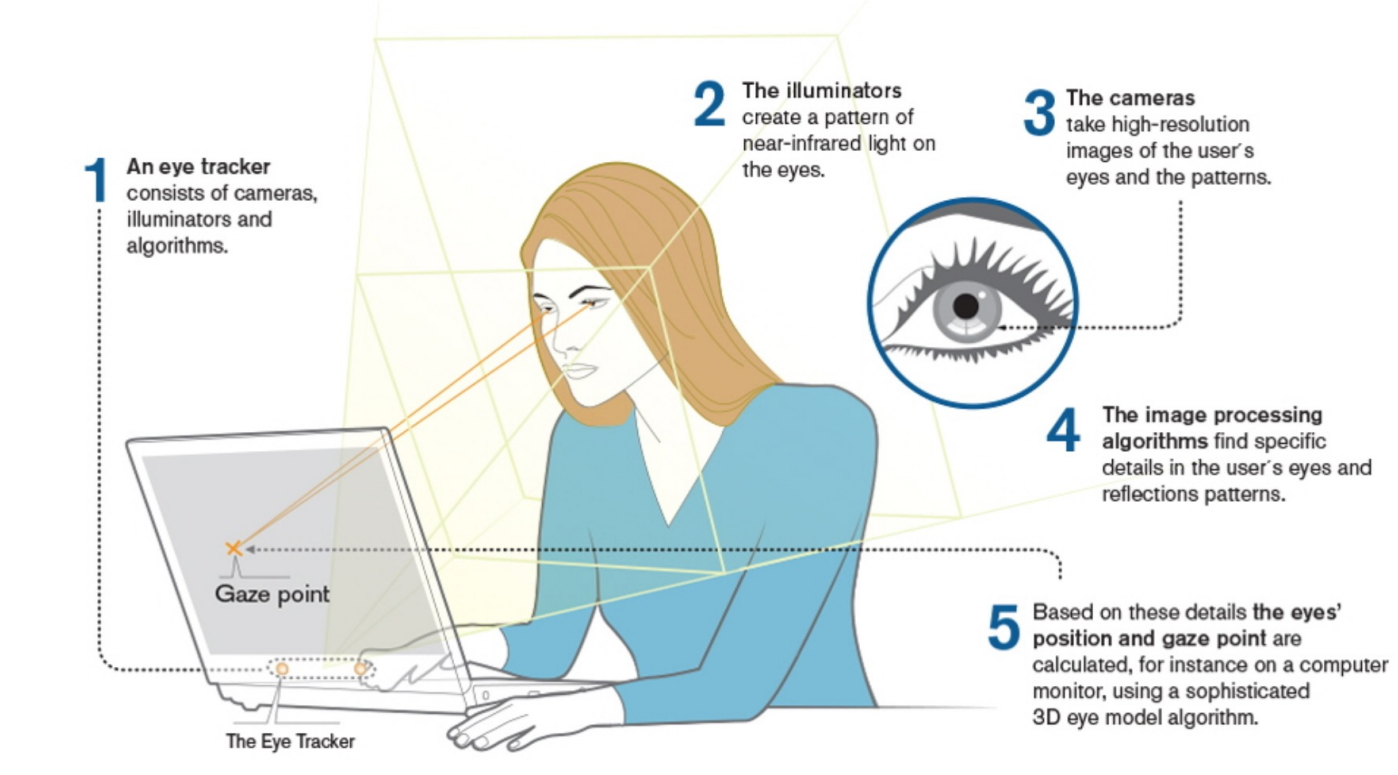


Image from https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/

GAZE TRACKING TECHNOLOGIES

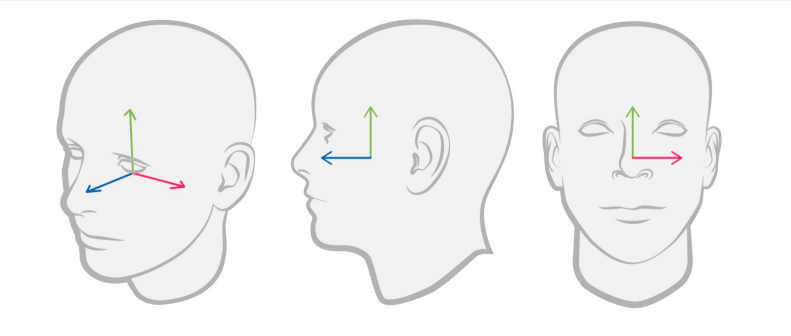
- Software + Hardware (with a Depth Camera) that support Gaze Tracking are:
  - ARKit - Apple devices with TrueDepth front-facing camera (iPhone 10 or more / iPad Pro) https://developer.apple.com/documentation/arkit/content_anchors/tracking_and_visualizing_faces
  - Mixed Reality Toolkit - Hololens 2 https://docs.microsoft.com/en-us/windows/mixed-reality/design/eye-tracking
  - RealSense D400 series https://www.intelrealsense.com/types-of-tracking-overview/
  - EyeTribe (no more available) https://en.wikipedia.org/wiki/The_Eye_Tribe
  - Tobii https://vr.tobii.com
- a list of other gaze tracking software libraries is available here: https://imotions.com/blog/free-eye-tracking-software/

ARKIT FACE TRACKING

https://developer.apple.com/documentation/arkit/content_anchors/tracking_and_visualizing_faces

FACE TRACKING: ARFaceAnchor

- Face position and orientation:
  the property *transform* describes the face position and rotation relative to the world coordinates of the current AR session.



The face transform coordinate system is right-handed.
Image from
https://developer.apple.com/documentation/arkit/arfaceanchor

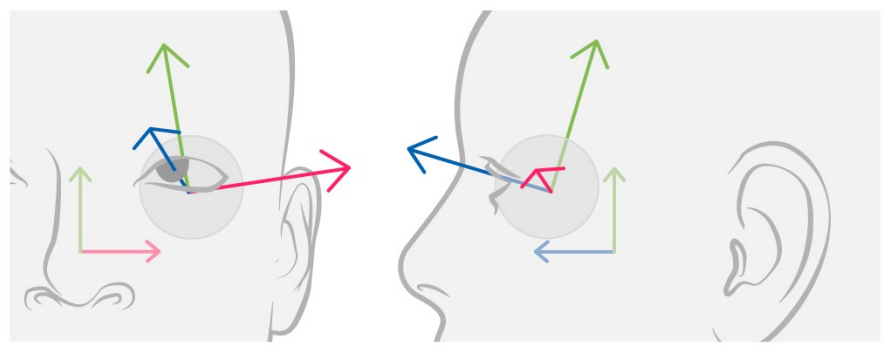
- Face Topology:
  the *geometry* property provides a model (*ARFaceGeometry*) to represent the face topology, as face shape and current facial expression as a 3D mesh.
- Facial expression:
  The *blendShapes* provides a model of the current facial expression, it is defines as a dictionary containing as a key locations of specific parts of the face or facial feature (*BlendShapeLocation*) and as values the coefficients that represent the movement of the corresponding facial feature.

https://developer.apple.com/documentation/arkit/arfaceanchor

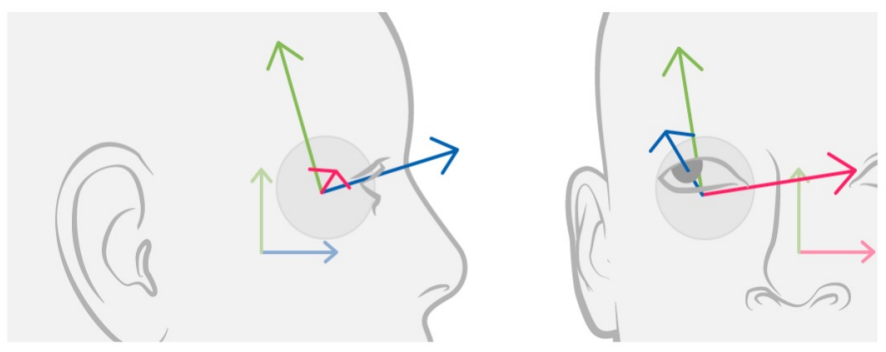
EYE TRACKING:
leftEyeTransform and rightEyeTransform

Indicates the

- Position of the eye pupil or eyeball relative to the position of face.
- Rotation of the eye pupil or eyeball indicates the orientation of the eyeball:
  a rotation on the x axis indicates that the user is looking up or down
  a rotation on the y axis indicates that the user is looking to the left or to the right



Left Eye
https://developer.apple.com/documentation/arkit/arfaceanchor/2968191-lefteyetransform



Right Eye
https://developer.apple.com/documentation/arkit/arfaceanchor/2968193-righteyetransform

FIXATION POINT: lookAtPoint

In addition, through tracking the eyes' pupils' position it can be estimated the fixation point, i.e. the point the user is directing the gaze.

The *lookAtPoint* vector abstracts from the *leftEyeTransform* and *rightEyeTransform* matrices to estimate what point, relative to the face, the user's eyes are focused upon.

https://developer.apple.com/documentation/arkit/arfaceanchor/2968192-lookatpoint

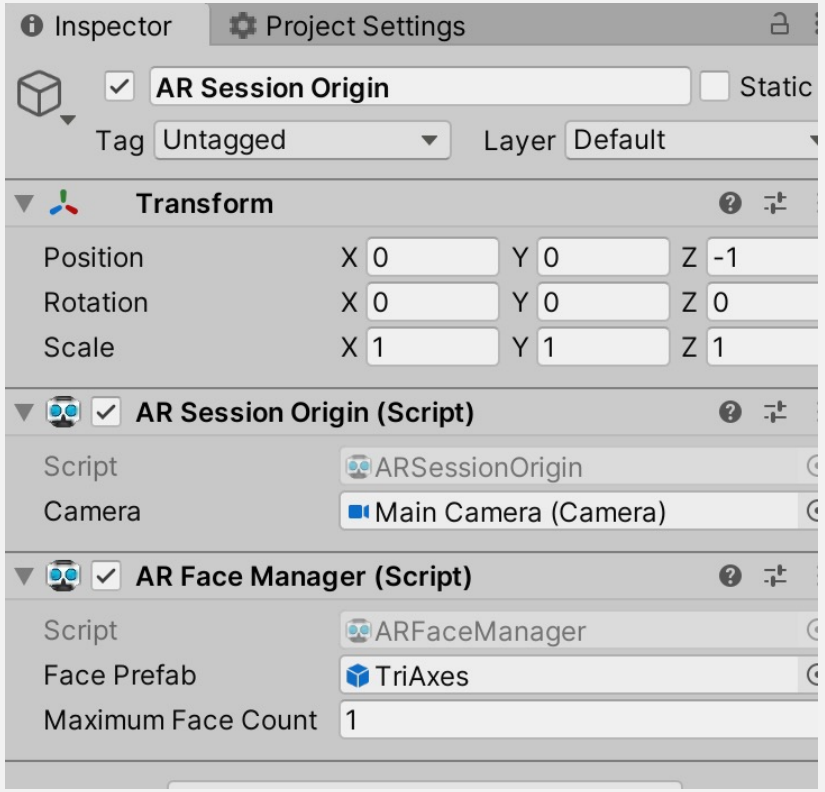
UNITY3D
ARKIT WITH ARFOUNTATION

https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/api/UnityEngine.XR.ARFoundation.html

Class ARFaceManager

This script should be added to the AR Session Origin Prefab.

This script manages (creates / update / deletes) the Face Prefab with attached the ARFace script component.



Some ARFaceManager interesting properties:

- *requestedMaximumFaceCount* : get or set the maximum possible faces tracked simultaneously
- *supportedFaceCount* : get the maximum supported possible faces tracked simultaneously

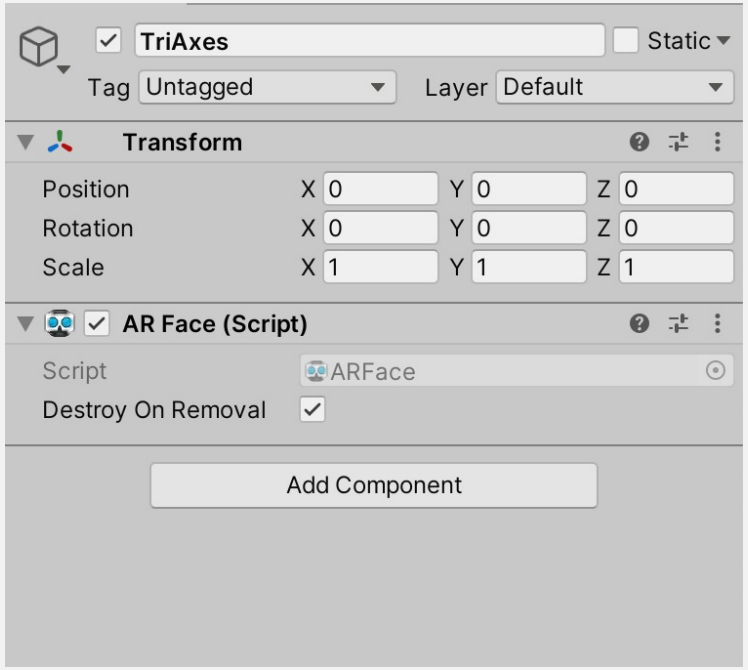
https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/api/UnityEngine.XR.ARFoundation.ARFaceManager.html

Class ARFace

The class ARFace contains all the information about the detected face.

The ARFace script is contained in the XR folder, and it should be added as a component in the Prefab.

That Prefab is the one that you will give to the ARFaceManager component in the scene (see previous slide).



Some ARFace interesting properties:

- leftEye : contains data (position, rotation, …) about the left eye in relation to the face
- rightEye: contains data about the right eye in relation to the face
- fixationPoint: vector3 of the eye fixation point in relation to the face

ARFace interesting event:

- updated

https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/api/UnityEngine.XR.ARFoundation.ARFace.html

blendShapes and UNITY3D

blendShapes coefficients are available in unity arkit – arfoundation.

BlendShapes can be retrieved from the method *GetBlendShapeCoefficients(TrackableId, Allocator)* in the class *ARKitFaceSubsystem.*

The method *GetBlendShapeCoefficients* returns a *NativeArray<ARKitBlendShapeCoefficient>*

You can retrieve blendShapes in the following way:

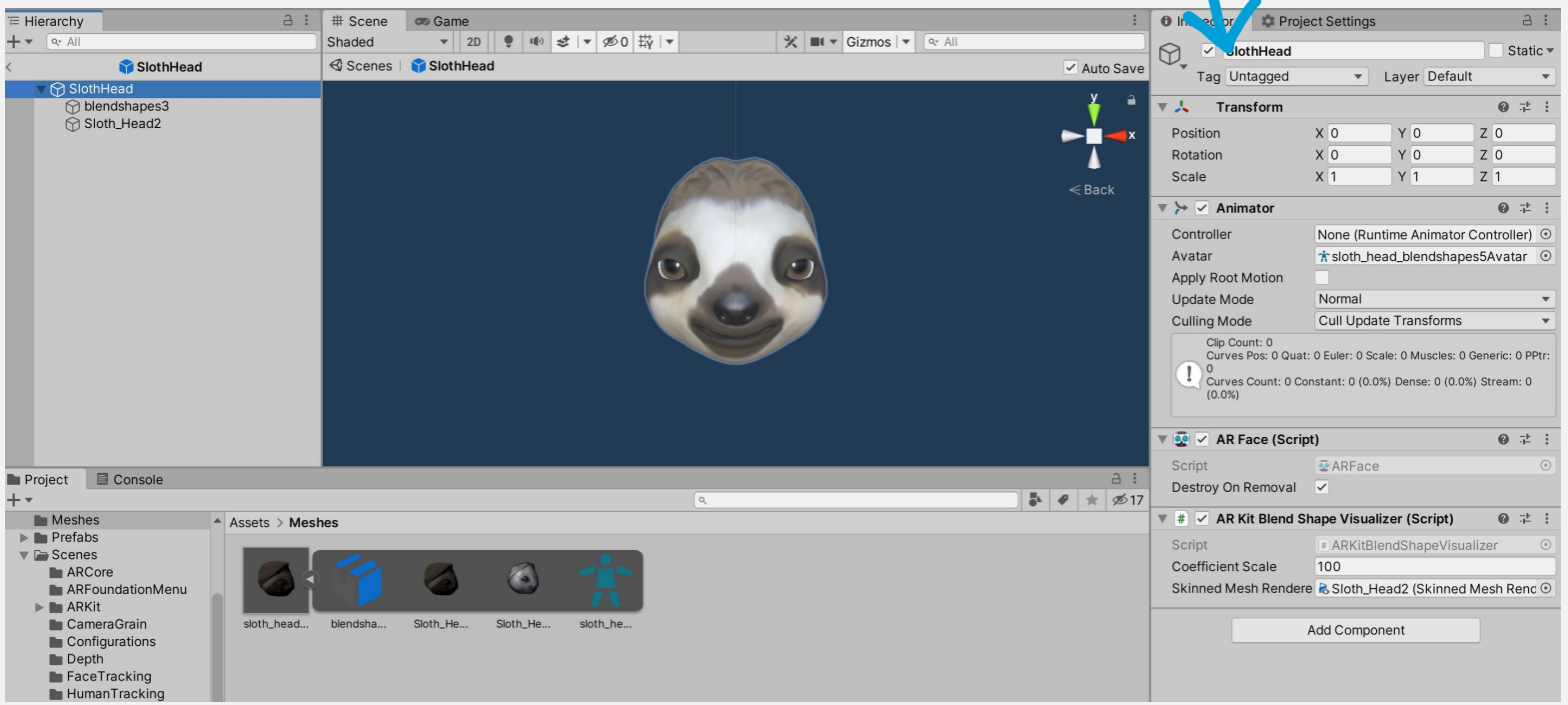
```
var faceManager = FindObjectOfType <ARFaceManager> ();
if (faceManager != null)
{
    m_ARKitFaceSubsystem = (ARKitFaceSubsystem) faceManager.subsystem;
}

var blendShapes = m_ARKitFaceSubsystem.GetBlendShapeCoefficients(m_Face.trackableId,Allocator.Temp)
```

blendShapes and UNITY3D

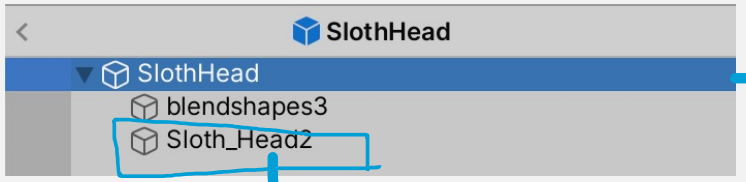
You can use *blenShapes* coefficients in many ways: for example

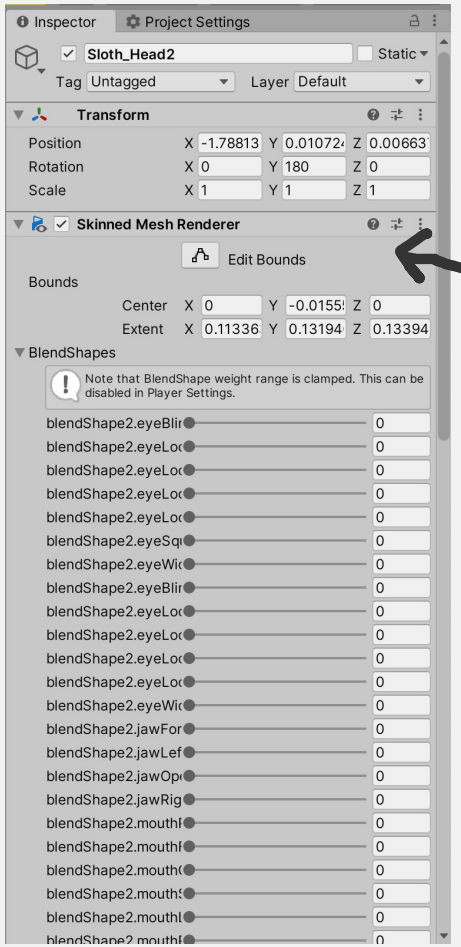
- to check in the user is blinking, i.e. if the right eye or/and the left eye is closed;
- or to map the coefficients into a rigged mesh of a face;

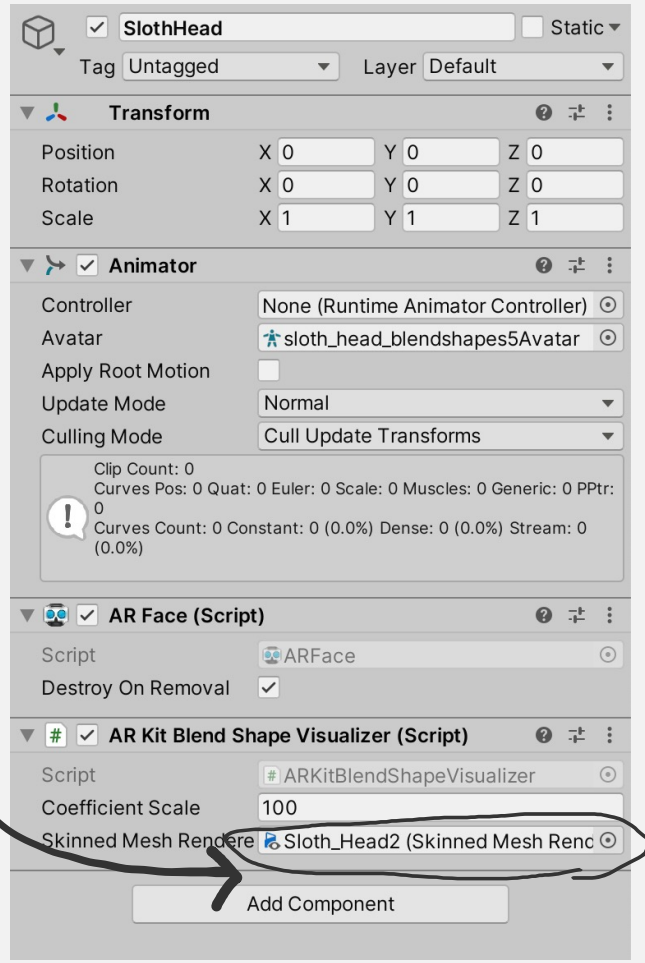


This scene is part of the GitHub Unity Samples here https://github.com/Unity-Technologies/arfoundation-samples/blob/main/Assets/Scenes/FaceTracking/ARKitFaceBlendShapes.unity
In the next slide you can find a description of SlothHead Prefab

Map *blendShapes* into a rigged mesh







The SlothHead GO has the rigged animation avatar in the animator component

The Sloth_Head2 GO has the Unity3D Skinned Mesh Renderer https://docs.unity3d.com/2020.3/Documentation/Manual/class-SkinnedMeshRenderer.html

The AR Kit Blensd Shape Visualizer is a sample script from the Unity GitHub Samples repo, it maps the *blendShapes* in the Skinned Mesh Renderer of Sloth_Head2

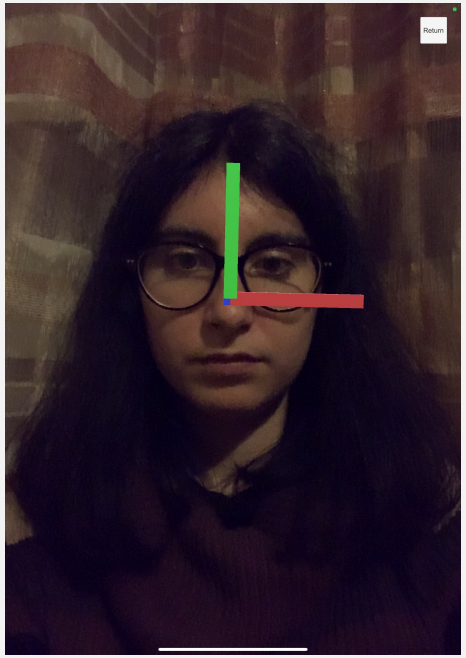
OTHER ARKIT + ARFOUNDATION SAMPLES FROM

https://github.com/Unity-Technologies/arfoundation-samples/tree/main/Assets/Scenes/FaceTracking

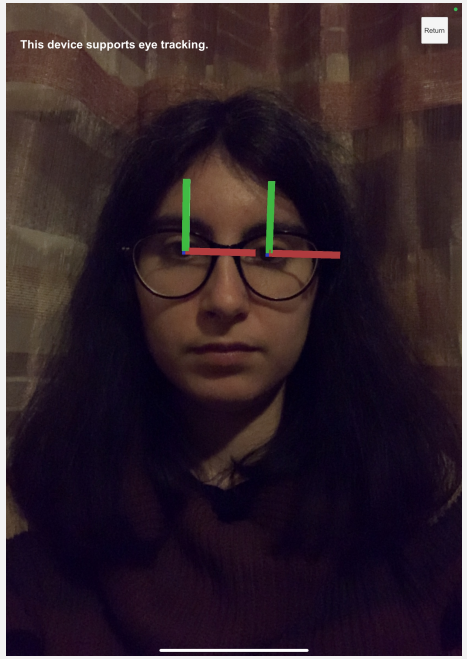
useful links:

- https://github.com/Unity-Technologies/arfoundation-samples
- https://github.com/Unity-Technologies/arfoundation-demos

4 ARFOUNDATION + ARKIT SAMPLES



Face Pose



Eye Pose

BlendShapes

Fixation Point

RECALL
ARFOUNDATION UNITY SETUP

GET STARTED WITH ARFOUNDATION 4.1

AR foundation allows to work with different AR platforms inside Unity3D

You can install ARFoundation in Unity's Package Manager

In addition to that Package, you should install also the Plugin you need, as ARKit for IOS apps or ARCore for Android apps development.

Supported Platform Packages

The following platform packages and later implement the AR Foundation features indicated above:

Package Name	Version
ARCore XR Plugin	4.1
ARKit XR Plugin	4.1
ARKit Face Tracking	4.1
Magic Leap XR Plugin	6.0
Windows XR Plugin	5.0

ARFOUNDATION SETUP

1. Go to Window > Package Manager
2. In the Package Manager select “Packages: Unity Registry” then search the term AR
3. Install ARFoundation
4. Install ARKit or ARCore, depending on your target device, today we will install ARKit face
5. Now Create a new Scene
6. Add a GameObject > XR > AR Session
   AR Session supports enabling and disabling XR on the target device
7. Add a GameObject > XR > AR Origin
   Because AR devices provide their data in "session space", which is an unscaled space relative to the beginning of the AR session, the ARSessionOrigin performs the appropriate transformation into Unity space

https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html#using-ar-foundation

ARFOUNDATION SETUP

8. Go to File > Build Settings > Switch Platform to the target device iOS
9. Now in the Build Settings click on Player Settings to open that panel
10. In the Player Settings in iOS under the voice "Other Settings" → "Configuration" check the box corresponding to "Requires ARKit support"
11. Now Edit > Project Settings > XR Plug-in Management enable the Provider Plugin setup, i.e. check the box ARKit https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html#provider-plugin-setup
12. Before building the project in the Player Settings under the voice "Other Settings" → "Identification" you can change the Bundle Identifier, that should be a unique identifier of your project.
13. Now you can build the Project

https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.1/manual/index.html#using-ar-foundation

ARFOUNDATION SETUP

From Unity Docs:

https://github.com/Unity-Technologies/arfoundation-samples#why-is-arkit-face-tracking-a-separate-package

Why is ARKit Face Tracking a separate package?

For privacy reasons, use of ARKit's face tracking feature requires additional validation in order to publish your app on the App Store. If your application binary contains certain face tracking related symbols, your app may fail validation. For this reason, we provide this feature as a separate package which must be explicitly included.

REFERENCES

[1] Walker F, Bucker B, Anderson NC, Schreij D, Theeuwes J. Looking at paintings in the Vincent Van Gogh Museum: Eye movement patterns of children and adults. PLoS One. 2017 Jun 21;12(6):e0178912. doi: 10.1371/journal.pone.0178912. PMID: 28636664; PMCID: PMC5479528.

[2] Eye tracking: what do we look at when we are looking? Web article, 18 mar 2018, Alexia Revueltas Rouz, Centre for Research in Digital Education https://www.de.ed.ac.uk/news/eye-tracking-what-do-we-look-when-we-are-looking (date accessed 10-05-2021)

[3] https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/ (date accessed 10-05-2021)

[4] https://imotions.com/blog/eye-tracking-work/ (date accessed 10-05-2021)

[5] https://www.tobii.com/group/about/this-is-eye-tracking/ (date accessed 10-05-2021)