2021

Alchera Inc.

Alchera Engine Developer Guide

Use Alchera Unity SDK for developing Face & Hand Detection AR application on Android, iOS.
This is a guideline for using AlcheraSDK Engine on Unity3D.

Import AlcheraUnityPlugin_2020.1.0.unitypackage from Unity3D,
(Assets – Import Package – Customer Package…)
and check out Demo scene from Assets/Alchera/Example/01.Scenes/.

alchera

Alchera Engine Developer Guide Contents

alchera

Alchera API Reference

0. Getting Started

Alchera Unity SDK is mobile-optimized engine, supporting the front/rear camera of mobile devices with Landscape Right, Landscape Left, Portrait direction.
Using Unity3D, you can easily develop Face & Hand Detection AR multi-platform application on iOS, Android, Windows and MacOS.
If this is the first time to use Alchera Unity SDK, take a look on some Demo Scene examples.

1. Scenes

Included Scenes

```
├── CaptureScene
├── Face2DFacemark
├── Hand2DSkeleton
├── ComplexScene
├── Face3DSticker
├── Face3DAnimoji
├── Face3DMask
├── FaceARKitScene
└── UI
    ├── DemoUI
    └── Splash
```

If this is the first time to use Alchera Unity SDK, it is easy to start with Demo Scenes.
Use cases are included in the Demo Scenes on Assets/Alchera/Example/01.Scenes/.

1.1 CaptureScene

This scene is only using Unity C#, excluding AlcheraSDK Native Library.
This function shows the camera input on quad with additional functions including Swap, Image capture.
This scene does not have license error by not calling Alchera SDK Native Library. We suggest implementing this scene on first package import in order to check camera / C# error.
Other examples call Alchera SDK Native function, using ImageData structure acquired from this CaptureScene.
(This is not included in Demo app)

1.2 Face2DFaceMark

This scene calculates facial landmark with ImageData acquired from the camera, and position the result on quad by prefab.
It implements only 2D landmark extracted calculation by Uncheking Need3D from FaceService.cs.

1.3 Hand2DSkeleton

This scene calculates hand information with ImageData acquired from camera, and position the result on quad by prefab.
It implements only 2D landmark extracted calculation by Uncheking Need3D from HandService.cs.
Lanmark, RegionBox, Gesture LeftOrRight result can also be checked.
(This version does not provide Hand 3D function)

1.4 ComplexScene

This scene is used under the scenario of using both hand and face simultaneously. In order to make a simple demo, these two calculations are composed of C# code enabling linear implementation. Face can implement 3D calculation by Checking Need3D.

1.5 Face3DSticker

This Scene calculates HeadPose in 3D and positions Prefab(Sticker)with the result. This Scene is expected to be used under the scenario of using various stickers, so when you touch the screen, it changes to the next Prefab(Sticker). Addtional 3D calculation to Check Need3D is mandatory and it is better to avoid unnecessary calculation other than HeadPose using LevelOf3DProcess.

▶ Occlusion function for the Sticker is implemented on Unity3D, enabling the effect of hiding sticker under face. In Unity Layer, set up the Prefab Layer as "StickerOcclusionMask" for masking function, and "StickerOpague", "StickerTransparent" to be masked. (Using Scriptable Render Pipeline function.)

▶ Sticker Prefab, which is in the Demo App, is not included on Alchera Unity SDK due to model License issue.

1.6 Face3DAnimoji

This Scene calculates HeadPose and BlendShape in 3D and changes the position of Prefab(Panda) or SkinnedMesh with the result. Addtional 3D calculation to Check Need3D is mandatory and it is better to avoid unnecessary calculation except HeadPose, Bleendshape, using LevelOf3DProcess as HeadPose_BlendShape. **(This is not included in the Demo App)**

1.7 Face3DMask

This scene provides HeadPose and FaceMask mesh by vertex in 3D. It changes the position of Prefab(Mask) and Mesh with the calculation result. Addtional 3D calculation to Check Need3D is mandatory, and it only works by setting LevelOf3DProcess as HeadPose_BlendShape_Vertex.

1.8 FaceARKitScene

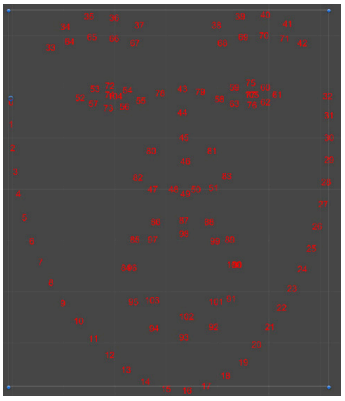
This Scene is based on the same Interface as Unity Interface provided on Apple ARKit. Function can be called based on Event, and if you are familiar with ARKit Unity Interface, start with this Scene. Code for optimizing the Interface to existing Demo is exposed in C#. **(This is not included in the Demo App)**

1.9 UI - DemoUI, Splash

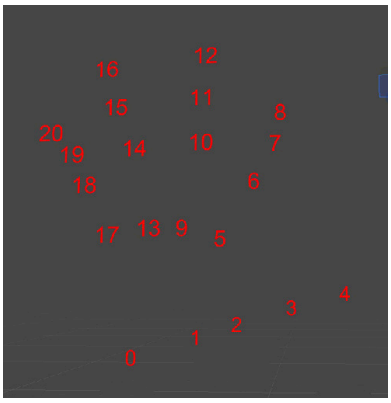
This Scene manages the UI on the Demo App. The code is written in Unity C#.

1.10 Common

▶ Order of face/hand landmark(figure1, figure2) has a guide in Assets/Alchera/Resources/.



<Figure1> Face Landmark



<Figure2> Hand Landmark

▶ Number of Face/Hand to be detected at once can be set at [Face/Hand]Service.cs/MaxCount. There is no limit to the number, but set the number considering the app spec.

▶ This is implemented for Pooling Prefab in the early stage considering the spec, and intended to show Prefab based on the number of detected body. This code can be found in Draw~.cs code.

▶ Code for Example/02.scripts/ is written as an example for C#, it also works in different language.

▶ Below is the Gesture Information that can be identified.

<Figure3> Gesture (Gun)

<Figure4> Gesture (Mini_heart)

<Figure5> Gesture (Okay)

<Figure6> Gesture (One)

<Figure7> Gesture (Paper)

<Figure8> Gesture (Rock)

<Figure9> Gesture (Thumbs_up)

<Figure10> Gesture (V)

2. Key Concepts

Alchera Unity SDK works in 5 steps.

1. Initializing 2. Camera/Quad 3. Detecting 4. Rendering 5. Releasing

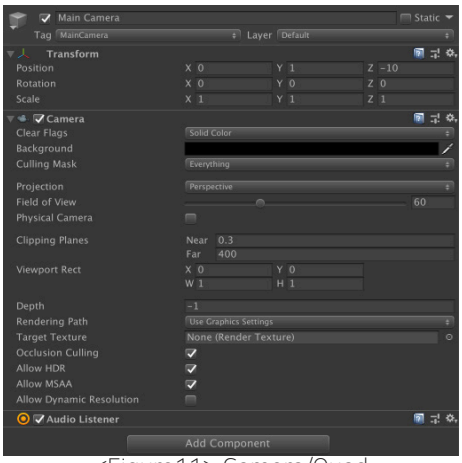
2.1 Initializing

In order to implement Detection, it is mandatory to call Init first. This basically implement 2D Initializing and when Need3D, on Face/HandService.cs, is set true, 3D Library will be also initialized.

2.2 Camera/Quad

Alchera Unity SDK use Unity default camera, the Main Camera.
By using default camera, it is compatible with other applications,
and supports Landscape Right, Landscape Left, Portrait direction for iOS/Android front and rear camera. Since it is written only in Unity C#, this can be checked on CaptureScene without Native Function Call License.

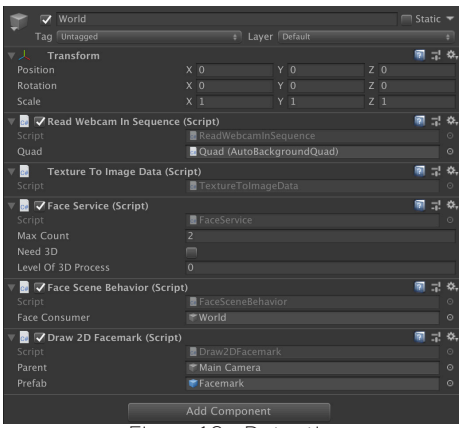
Demo examples implement camera screens by applying Camera Input Texture on Quad, adjusting Transform and this method can be checked on AutoBackgroundQuad.cs. ReadWebcamInSequence.cs reads camera input, TextureToImageData.cs creates ImageData to be used in Alchera SDK.



<Figure11> Camera/Quad

2.3 Detecting

In this stage, [Face/Hand]Service.cs call Detection with ImageData as an input and get each results as FaceData, HandData structure.
~SceneBehavior.cs is the script that implements the main flow of Demo App.



<Figure12> Detecting

2.4 Rendering

Manage the Prefab to show after the Detection on Draw~.cs. Use Prefab in advance by pooling in early stage.
Implements the calculation and rendering using results(FaceData, HandData) from Detection on ~Prefab.cs inside the Prefab.

2.5 Releasing

This is the process to unlock the memory when the app is shut down or not in use.

3. Unity Example Scripts

3.1 Assets/Alchera/SDK/

[Face/Hand]Service.cs uses Interfaces and Alchera functions.
It allocates the memory including [Face/hand]Data and parameter using Translator struct in advance, post-process on Detected results using Fetch if needed, and only returns the memory containing valuable data.

3.2 Assets/Alchera/Example/02.Scripts

This script is for Unity Demo example, using AlcheraNative Interface.

SceneBehaviorWorks/

This script controls the basic app flow. It creates ImageData from WebCamTexture, implements Detection if there is any Detector, and Consume those data. It is included in Start() for statement instead of Update() function in order to individually manage Detection Cycle considering the discord between each camera device and Camera texture.

TextureWorks/

ReadWebcam.cs : This script manage overall information about the camera including WebCamTexture, device orientation, front/rear and platform.
ReadWebcamInSequence.cs : This script controls and updates Task<Texture>, adjusting to camera updates.
TextureToImageData : This script receives WebCamTexture and returns ImageData needed for Alchera SDK.
SaveLastTexture.cs : This script saves texture as an image format.

DrawWorks/

This script Consume Fetched [Face/Hand]Data. It creates Prefab in advance for pooling, and call Prefab function only for detected body. This folder is suggested to be duplicated for each usage.

PrefabWorks/

This script enables Prefab managed by DrawWork to implement function or rendering using [Face/Hand]Data.
Create EmptyObject on the top while creating Prefab, and add this script as a Component. This folder is suggested to be duplicated for each usage.

AutomatedWorks/

This script enables Quad, that renders camera texture, to automatically change size and position with camera FOV, Screen width/height, Device rotation, Platorm and front/rear camera. It is added as Component in Quad object.

UIWorks/

This script manages the UI on the Demo App.

DemoScript/

This script adds simple effects to the examples.

ARKitInterfaceWorks/

This script example use similar Interface with Apple ARKit Unity Demo by wrapping Call SDK once more.
If you are familiar with ARKit Unity Interface, work with this script. If not, it is suggested to use other example Scene.

4. Unity Settings

4.1 Common

▶ Create Demo Scene in Unity 2019.2. In lower version, AlcheraSDK function may work but Unity error can occur on Sharder or Material since Universal Render Pipeline (LWRP) was applied on the Demo.

▶ Tested External Camera : Logitech C920 webcam (1280x720)

▶ Edit - Project Settings - Graphics

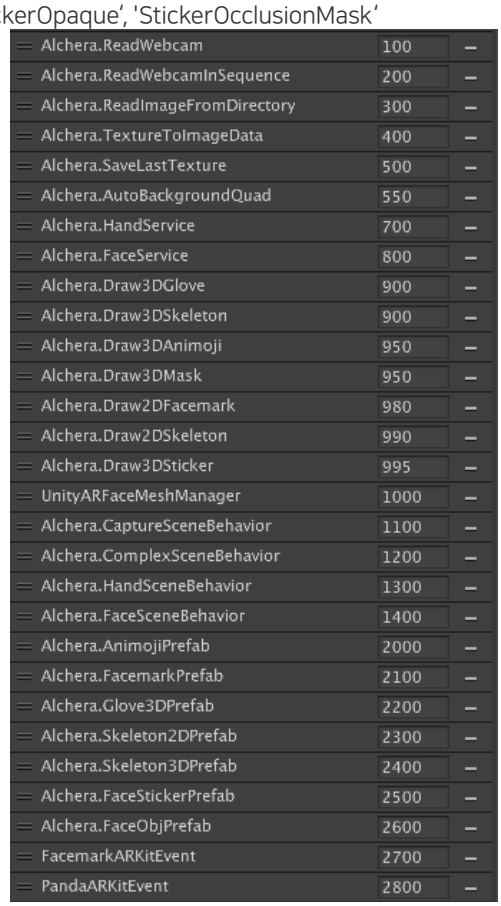
- Scriptable Render Pipeline Settings : LightweightAsset
- Tags and Layers : Tags : 'WebCamQuad' , Layers: 'StickerTransparent', 'StickerOpaque', 'StickerOcclusionMask'
- Script Execution Order (Refer to <Figure13>)

▶ Edit - Project Settings - Player - Resolution and Presentation

- Allowed Orientations for Auto Rotation - Portrait Upside Down : UnCheck

▶ Edit - Project Settings - Player - Other Settings

- Allow 'unsafe' Code : Check



<Figure13> Script Execution

4.2 Android

Supporting Platform : armeabi-v7a, arm64-v8a

▶ Edit - Project Settings - Player - Other Settings

- Graphic APIs : OpenGLES2

  (Example does not work on 2019.2 version Vulkan, OpenGLES3 due to Unity bug internally. It is being adjusted on Unity3D document.)[1]
- Write Permission : External (SDCard)
- Minimun API Level : Android 8.0 'Oreo' (API level 26) is suggested due to Google policy.
- Scripting Backend : IL2CPP
- Target Architectures : ARMv7a, ARM64

4.3 iOS

Supporting Platform : armv7, armv7s, arm64

▶ Edit - Project Settings - Player - Other Settings

- Camera Usage Description : "add camera permission description"
- Target Minimun iOS Version : 10.3

5. License

Please contact Alchera Inc.

1 Vulkan-UnityVideoplayer relevant issue (https://docs.unity3d.com/ScriptReference/Video.VideoPlayer.html?_ga=2.161231871.331366313.1574922129-10526039.1572238404)

6. Alchera Classes

6.1 FaceLib

Finds 2D face based on input image and find facial landmark(feature information including eye, nose, mouth).
Detect function is implemented with Detection or Tracking. Detection identifies face from the full screen, Tracking tracks face position based on facial information from previous frame. Since the performance time of Detection is longer then Tracking, generally implement Tracking in advance and implement Detection occasionally in between.

Init(ref FaceLib.Context context)

Initializes FaceLib.

context.detectableSize : designate minumum pixel size for face identification. smaller number detects smaller face but if the number is too low, speed slows down.

(Default size is 128)

context.logStateMode : set debug variable as 0 to check Library action.

context.maxCount : define maxCount number of face to detect/track. If existing face is over maxCount, detects the bigger face first.

context.InitPath : designate deep learning network location for face recognition.

Detect(ref FaceLib.Context context, ref ImageData image, FaceData* facePtr)

Detects face from ImageData image input and save the FaceData result from detected face.

Function to convert Unity Texture to ImgaData is defined on TextureToImageData.cs in the example.

image.WebcamWidth : webcam width.

image.WebcamHeight : webcam height.

image.DetectionWidth : detection width.

image.DetectionHeight : detection height.

image.Degree : action to Image rotation and Camera rotation.

image.Data : pointer for Pixel data memory

image.OffsetX, image.OffsetY : set up when cropping input image, not using the whole image.

facePtr : array including contect.maxCount number of FaceData.

facePtr[i].ID : Face ID of FaceData[i] (output).

facePtr[i].Landmark : 2D facial Landmark of FaceData[i](output).

This function returns number of detected face(output).

Other variables from FaceData is related to 3D fitting, not usable in FaceLib.

Release(ref FaceLib.Context context)

Releases the resource on FaceLib.

6.2 Face3DLib

Based on 2D face and Landmark detected by FaceLib, detects 3D face and emotion for AR Emoji.

Init(ref Face3DLib.Context context3D)

Initializes Face3DLib.

context3D : struct space for 3D fitting

context3D.levelOf3DProcess : Adjust the range for 3D calculation in enum type.

levelOf3DProcess = 0,oOnly calculates target.Headpose. (For simple 3D stickers including glasses.)

levelOf3DProcess = 1, calculates target.Headpose, target/GetAnimation. (For Animojis and etc).

levelOf3DProcess = 2, calculates target.Headpose, target/GetAnimation and target.Vertices. (For complicated effects on the face).

Set(ref Face3DLib.Context context3D, int width, int height, float fieldOfView)

Receives data for 3D fitting, set up in param.

context3D : struct space for 3D fitting

width : screen width.

height : screen height.

FieldOfView : camera's field of view used for screen rendering.

Process(ref Face3DLib.Context context3D, ref FaceData face)

Returns 3D facial position and Blendshape emotion with 2D face and lanmark input that FaceLib detected.

context3D : struct space for 3D fitting.

face is the pointer for 2D face detected by FaceLib.

face.Landmark : 2D facial landmark.

face.HeadPose : 3D face position and rotation according to 2D landmark (output).

face.GetAnimation : 3D face emotion according to 2D landmark (output).

face.Vertices : vertex of matched 3D face mesh (output).

6.3 HandLib

Finds ROI area for hand and 2D Skeleton based on input image.
By implementing Detect function, hand detection is implemented on entire area of input image and then tracking is implemented on the detected hand. Also estimates 2D Skeleton position for each hand on each image and the result is returned through handPtr.

Init(HandLib.Context context)

Initialize FaceLib.

context.maxCount : define the maxCount of hand detection/tracktion. If existing hands are over maxCount, detects the bigger hand first.

context.minHandSize : minimum size to identify hand. Default is 60.

context.InitPath : designate the location of deep learning network for face recognition.

Detect(HandLib.Context context, ImageData image, HandData* handPtr)

Detects hand from 2D ImageData image and save HandData result.

Function to convert Unity Texture to ImgaData is defined on TextureToImageData.cs in the example.

image.WebcamWidth : webcam width.

image.WebcamnHeight : webcam height.

image.DetectionWidth : detection width.

image.DetectionHeight : detection height.

image.Degree : action to image rotation, camera rotation

image.Data : pointer for Pixel data memory

image.OffsetX, image.OffsetY : set up when cropping input image, not using the entire image.

handPtr : array including contect.maxCount number of handData.

handPtr[i].NumPoints : number of Skeleton Points on hand (output).

handPtr[i].ID : hand ID of HandData[i] (output).

handPtr[i].Posture : hand gesture type of HandData[i] (output).

- supports [v, gun, rock, paper, mini_heart, one, thumbs_up,okay, unknown]

handPtr[i].LeftOrRight : detecting right/left hand of HandData[i] (output).

handPtr[i].Box : hand BoxRigion of HandData[i] (output).

handPtr[i].Center : Screen coordinate of the center of hand.

handPtr[i].Points : hand Skeleton Points of HandData[i] (output) .

This function returns number of detected hand(output).

Other variables from HandLib is related to 3D fitting, not usable in HandLib.

Release(HandLib.Context context)

Release the resource allocated on HandLib.

Still have Questions?

If you have any questions about this document, please contact to contact@alcherainc.com.

alchera

Make your AI dreams a reality

www.alcherainc.com