stARkit_device Unity SDK

1 Introduction

1.1 Purpose

This document provides information about how to install and use Lenovo AR framework SDK.

1.2 Background

Lenovo AR framework is a high performance; low latency AR on Android requires access to AR full function features and software optimizations on daystAR glass device.
The AR framework SDK provides access to these AR features which would otherwise be unavailable to developers.
In addition, the AR framework SDK implements the core low latency rendering work on daystar AR glass, which will help developers to create and show high quality AR content what they want. Which feature named “Display engine”.
The core AR framework SDK features include:

- Display Engine
- 3Dof head tracking
- 6Dof position tracking
- Gesture
- Camera image service

2.2 SDK package contents

2.2.1 LAR Display engine

All in “LARsuite” SDK

/LARSuite	Common utility code used by the SDK samples
/Plugins/Android/libs	Core libraries
/Sample	Unity plug-in needed to create Unity applications utilizing the SDK
Texture	Image resource used in animation in sdk

2.2.2 SLAM

liblarclient.so	Library in “LARsuite” package, for developer to call functions from slam service You can get point cloud data information by calling so file
liblarplugin.so	Binary in “LAR glass” package. Is the library called internally by sdk

2.2.3 Gesture

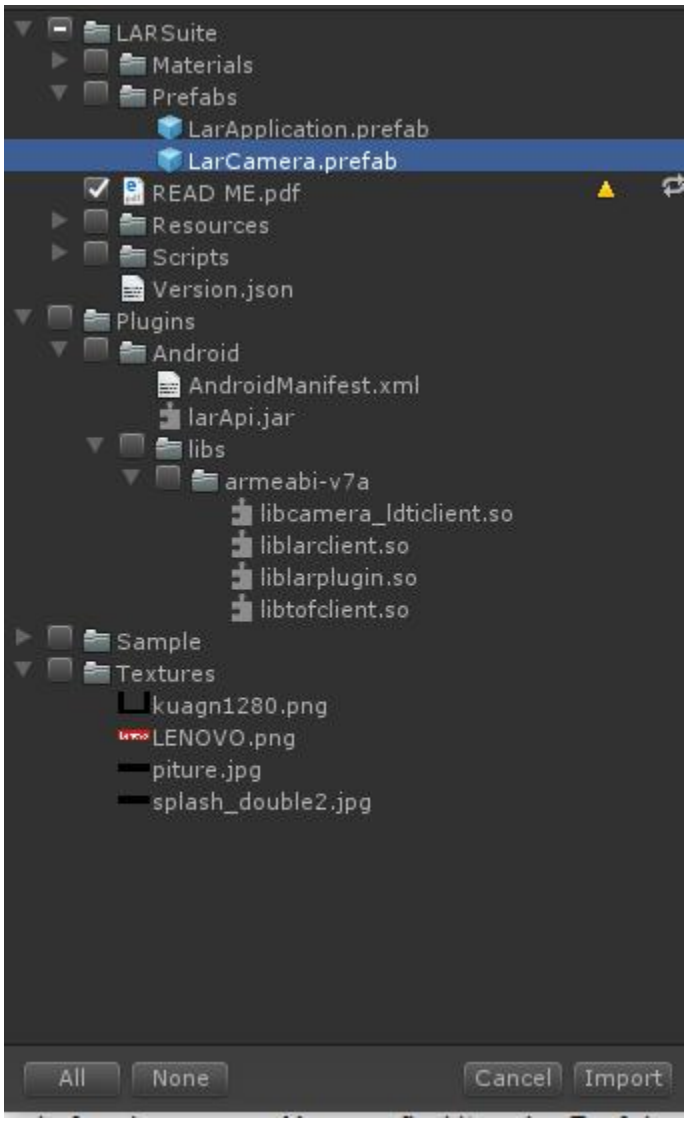
libtofclient.so	Library in “LARsuite” package, for developer to call functions from gesture service

3.1.1 Overview

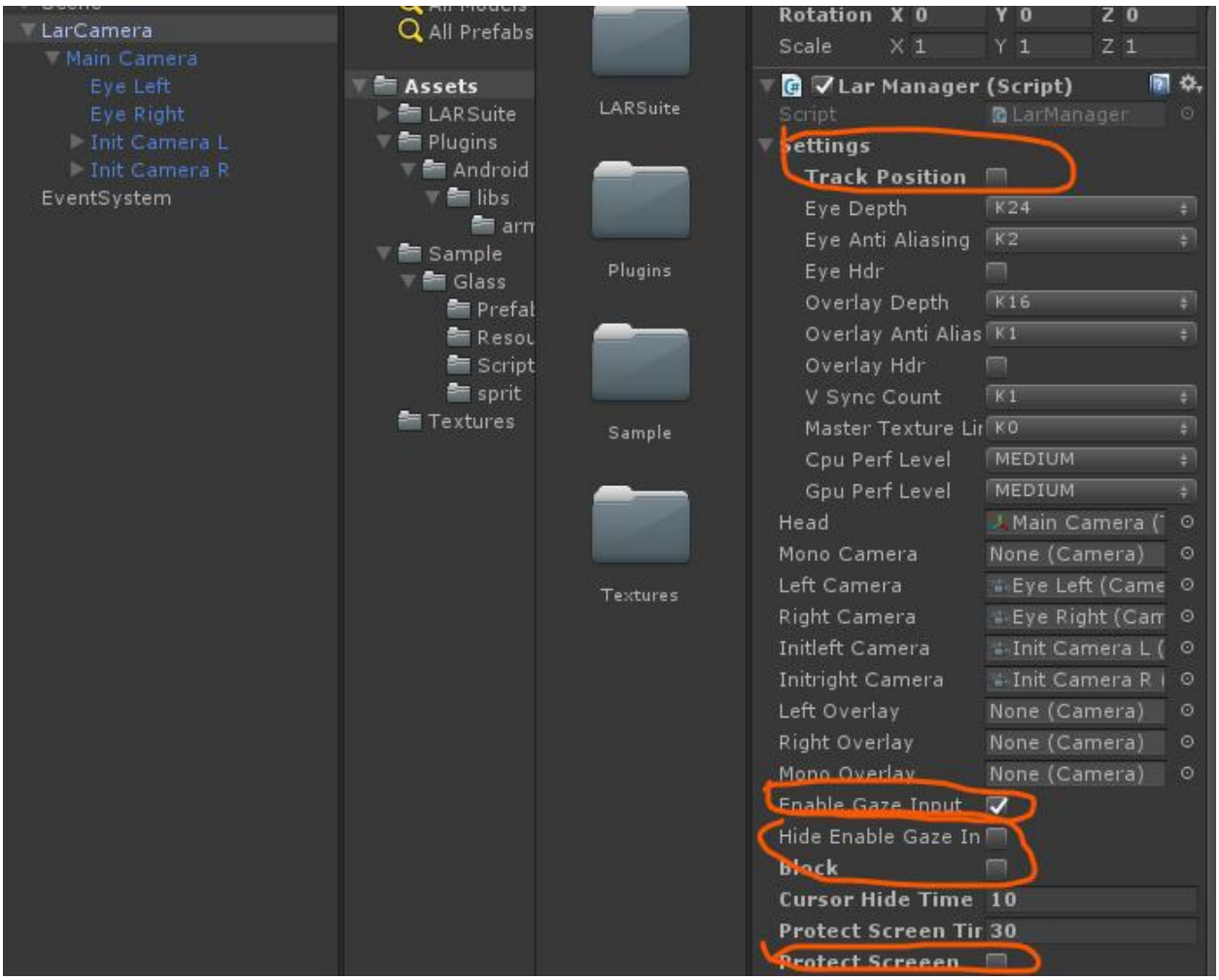
This section describes how to use Lenovo AR Display Engine with our release package. Lenovo Display Engine release package is made of two parts: Display Engine Lib and LarUnityPackage. Display Engine Lib is built-in in device's system image. LarUnityPackage can be imported to Uninty3D and used for your AR application developing.

3.2.2 How to use Lar Unity package

1. Import the Lar unity package into your Unity3D project.



2. Use LarCamera instead of main camera. You can find it under Prefabs folder named LarCamera.prefab.
3. Choose 6dof/3dof from LarCamera -> Settings -> Track Position
4. Choose whether to make the cursor appear： Enable GazeInput



There are mainly three modules in Scripts Folder: head tracking, inputs and stereo render.

LarManger Script on LarCamera.prefab uses to update the viewers position and orientation from the data read from the glass gyro sensors. Glass driver has export the gyro data as standard android gyro data, so we could read the gyro data directly, we can also read the gyro data and update the viewers position and orientation.

There are little external inputs in glass except the touch panel, the Gaze is the most important input available in glass. You could use Gaze like a mouse in PC, just move the gaze cursor in a different way:

moving with the head. We supply a Gaze Inputs Module to generate Gaze Events and pump into unity engine. We provide four Gaze Events now: Gaze Enter/Move/Click/Exist, more Gaze Events coming in Future, like Gaze Drag. We also supply Gaze Event Trigger to handle the generated Gaze Events, developer could attach a gaze event trigger to a UI control element and response to gaze event. Gaze input module works more like the common pointer input module in common unity applications, So, existing pointer event handlers could be ported to glass gaze event with little effort. In the unity editor, developer could enable or disable the gaze input module through glass controller.

Accompany with the gaze input module, there is a gaze cursor rendered in front of the glass eyes. Gaze cursor has three states: Gaze Enter/MoveOver/Exist. There are visual feedback when the gaze cursor enter or exist the gaze target. We have developed a default Gaze Cursor render, developers also could implement their own gaze cursor render derived from `GazeCursorBase`.

Note：

We need set unity settings as following:

1： Anisotropic Textures -> per texture

（Edit ->project settings -> quality ->rendering）

2：AA -> disabled

（Edit ->projectsettings -> quality ->rendering）

3：V Sync Count = Don'tSync （Edit ->project settings -> quality ->other）

4：Player settings －> Default orientation : landscape left

5：Player settings －> Rendering -> Multithreaded Rendering : disable

6：Player settings －> Configuration -> Write Permission : External(SDCard)

Follow the instructions below to modify the parameter values in PlaySetting

1. Create a new script derived from `GazeEventTrigger` class and attach to the Cube, name `CubeEventHandler`, to make the Cube object receive the Gaze input event.

```
public class CubeEventHandler : GazeEventTrigger {

    private Color _color;
    private MeshRenderer _render;

    void Start() {
        _render = gameObject.GetComponent<MeshRenderer>();
        _color = _render.material.color;
    }

    public override void OnGazeClick(PointerEventData data) {
        _render.material.color = Color.cyan;
    }

    public override void OnGazeEnter(PointerEventData data) {
        _render.material.color = Color.red;
    }

    public override void OnGazeExist(PointerEventData data) {
        _render.material.color = _color;
    }

    public override void OnGazeMove(PointerEventData data) {

    }
}
```

You could receive several **Gaze Input Events**, including **GazeEnter**, **GazeClick**, **GazeMove** and **GazeExist**. The tutorial just change the cube color when the Gaze Cursor move over and a Click event occur.

You could also use the standard unity event trigger to let gameobject interactive with the Gaze Input Events. We mapped the Gaze Event to Unity Pointer Event: Pointer Enter to Gaze Enter, Pointer Click to Gaze Click, Pointer Move to Gaze Move and Pointer Exist to Gaze Exist. We create `CylinderEventHandler` to the Cylinder to response the event trigger.

```
public class CylinderEventHandler : MonoBehaviour {

    private Color _color;
    private MeshRenderer _render;

    void Start() {
        _render = gameObject.GetComponent<MeshRenderer>();
        _color = _render.material.color;
    }

    public void OnPointerClick() {
        _render.material.color = Color.cyan;
    }

    public void OnPointerEnter() {
        _render.material.color = Color.red;
    }

    public void OnPointerExist() {
        _render.material.color = _color;
    }
}
```

We recommend to use `GazeEventTrigger` directly, it's simple and friendly to later SDK update.

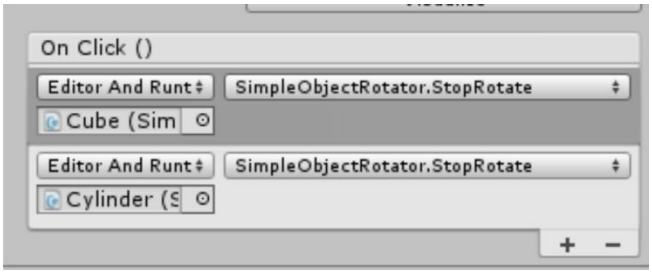
2. Switch to **Android** in the **Build Settings**. Input your **Company Name** and **Product Name**, Change the **Bundle Identifier**.
3. Connect **Lenovo Glass** and Build & Run.
   Put Lenovo glass on, you could see the simple scene we crated and a Gaze Cursor. Move your head left and right, the Gaze Cursor moves, when it hover over the Cube or Cylinder, the Gaze Event trigger and the color of the beneath game object change to Red. When Gaze Click event happens, it change to Cyan.

Gaze to Canvas

Scene location: /LARSuite /Sample /Glass/ GazeToCanvas.unity

1. Create new Unity project "GazeToCanvas"
2. Import LARSuite SDK as Custom Package
3. Create and Add simple 3D Objects, like a Cube and Cylinder under an empty gameobject "Scene"
4. Create a **UI Canvas** and a **Panel** under it, create a **Button** inside the Panel
5. Find the prefab "**Lotus Application.prefab**" under /**Prefabs** folder. Drag and add it to Unity Scene Hierarchy panel. Put it under the top level scene hierarchy.
6. Find the prefab "**LarCamera**.prefab" under /LARSuite /Prefabs folder. Drag and add it to Unity Scene Hierarchy panel. Put it under the top level scene hierarchy.

7. Enable the **Enable Gaze Input** Checkbox
8. In the Button **Inspector**, add Cube and Cylinder's as Button's **On Click** Responder to call the stop the simple object rotation.



9. Switch to **Android** in the **Build Settings**. Input your **Company Name** and **Product Name**, Change the **Bundle Identifier**.
10. Connect **Lenovo Glass** and Build & Run.
    Put Lenovo glass on, move the **Gaze Cursor** to the button and trigger Gaze Click event, the Cube and Cylinder stop rotate.

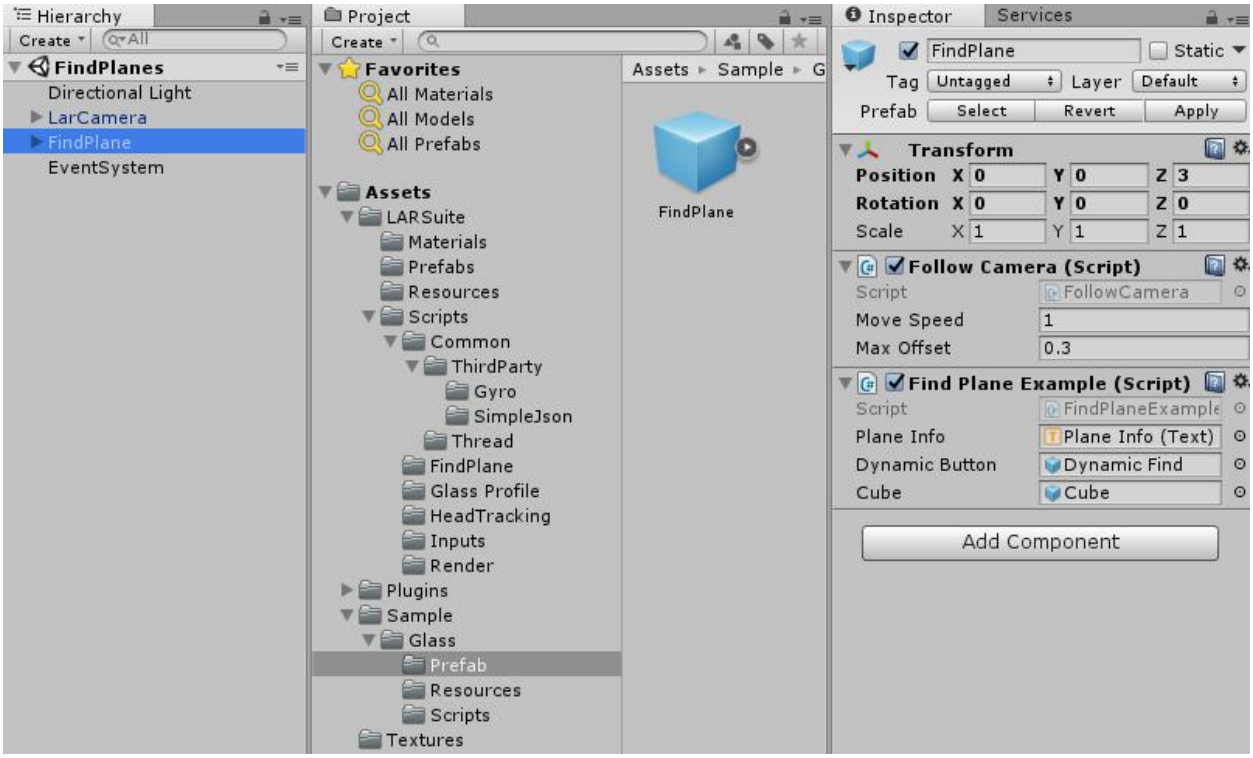
Find Planes

1. Create new Unity project "new project"

2. Find the prefab "**LarCamera**.prefab" under LARSuite /**Prefabs** folder. Drag and add it to Unity Scene Hierarchy panel. Use LarCamera instead of main camera.
3. Choose 6dof from LarCamera -> Settings -> Track Position
4. Choose GazeInput interaction function。
5. Find the prefab "**FindPlane**.prefab" under Sample/**Prefab** folder. Drag and add it to Unity Scene Hierarchy panel.

6.Switch to **Android** in the **Build Settings**. Input your **Company Name** and **Product Name**, Change the **Bundle Identifier**.

7.Connect **Lenovo Glass** and Build & Run.



There are two script components on the FindPlane prefab. The Follow Script can make UI interface follow the head movement . And the FindPlaneExample script can realize search and display of space plane .

In the FindPlaneExample script.

1. Call “FindPlane.SetMovEnable(true)” method in the Start function to open TOF Camera.
2. Call “FindPlane.SetCallBack(s_CallBack)” method before calling “FindPlane.StartFind()” method. “FindPlane.SetCallBack(s_CallBack) ” can set up some plane related information。
3. You can start searching planes through “FindPlane.StartFind()” method

4. At last .When you want to end the search for the plane.You can call the “ FindPlane.StopFind();”.

Glass Profile

Save glass properties, lenses separation, fov, projection fix, etc. There is a default glass profile in SDK.

1.Create new Unity project “GlassprofileTest”

2.Import LARSuite SDK as Custom Package
3.Create and Add simple 3D Objects, like a Cube and Cylinder under an empty gameobject “Scene”, you can attach a simple animation script to the Cube and Cylinder.
4.Find the prefab “**LarCamera**.prefab” under LARSuite / **Prefabs** folder. Drag and add it to Unity Scene Hierarchy panel. Put it under the top level scene hierarchy.
5.Create a new script class and attach to the LarCamera, name readGlassProfile

You can get the custom profile by this method: “GlassProfileManager.Instance.CustomProfile()”

You can get the Default Profile by this method:
“GlassProfileManager.Instance.DefaultProfile()”

Create a class to receive it:

```
GlassProfile glassProfile=GlassProfileManager.Instance.CustomProfile();
```

Then you can get the parameters in the configuration file

```
GlassProfile glassProfile= GlassProfileManager.Instance.CustomProfile ();
Debug.Log("LenseSeperationLeft" + glassProfile.LenseSeperationLeft);
Debug.Log("LenseSeperationLeft" + glassProfile.LeftProperty.PosX
    + " " + glassProfile.LeftProperty.PosY + " " + glassProfile.LeftProperty.PosZ);
Debug.Log(glassProfile.LeftProperty.RotX + " " + glassProfile.LeftProperty.RotY
    + " " + glassProfile.LeftProperty.RotZ);

Debug.Log(glassProfile.LenseSeperationRight);
Debug.Log("LenseSeperationRight" + glassProfile.RightProperty.PosX
    + " " + glassProfile.RightProperty.PosY + " " + glassProfile.RightProperty.PosZ);
Debug.Log("LenseSeperationRight" + glassProfile.RightProperty.RotX
    + " " + glassProfile.RightProperty.RotY + " " + glassProfile.RightProperty.RotZ);
Debug.Log(glassProfile.LenseFrustumData.Fov);
```

You can get the width and height of the screen by this:

```
Vector2 ScreenSize = glassProfile.ScreenSize;
```

You can get the FOV of the camera by this:

```
Int fov = glassProfile.LenseFrustumData.Fov
```

At last,You can set the camera parameters with the obtained data。

PointCloud

You can get point cloud data information by calling so file

```
[DllImport("larclient")]
public static extern bool start_slam();
[DllImport("larclient")]
public static extern bool stop_slam();
[DllImport("larclient")]
public static extern int slam_init();

[DllImport("larclient")]
public static extern bool mode_switch(bool mode);
[DllImport("larclient")]
public static extern void get_point_cloud(out int len, IntPtr data);

GCHandle dataObject;
IntPtr dataInptr;
```

Gesture

You can get information about current device gestures

```
[DllImport("tofclient")]
public static extern void movEnableDisable(bool enable);
[DllImport("tofclient")]
public static extern int getMovStatus();
[DllImport("tofclient")]
public static extern void  tofStatusCallback(TofStatusCallback status);


public delegate void TofStatusCallback(int tLen);
public static TofStatusCallback statusCallback;
```