日期

SLIDE IN SDK USER MANUAL

BASIC CONCEPT

A. Stereo eye rendering

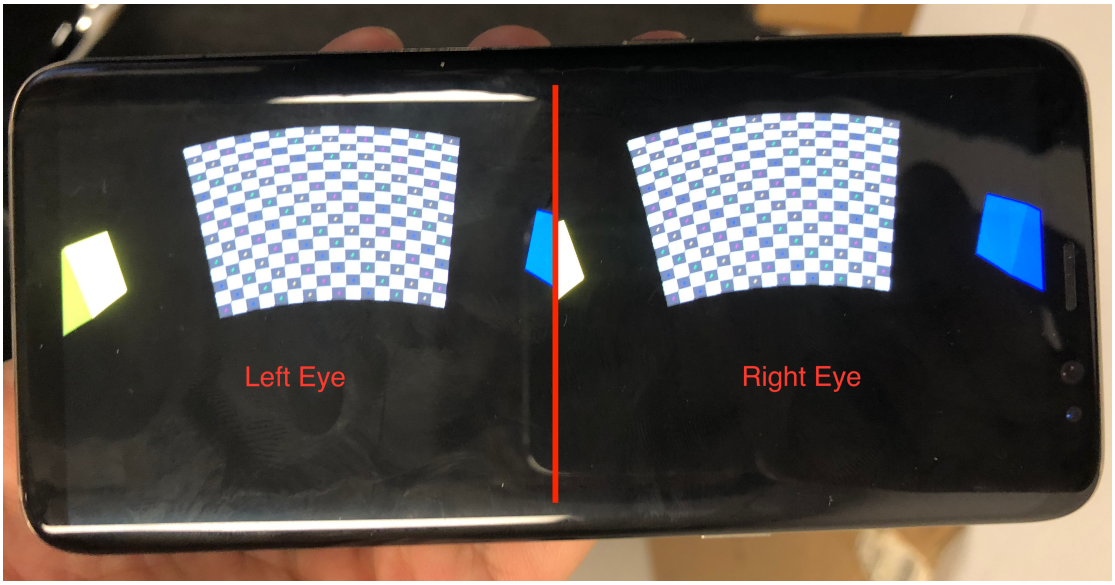
B. VPU Tracking

C. Marker and marker group

D. Calibration profile

The device is composite of two major parts: reflection len for stereo eye, and tag tracking VPU.

VPU track IR marker and sends data to phone via USB cable, while user see virtual content through reflection len.

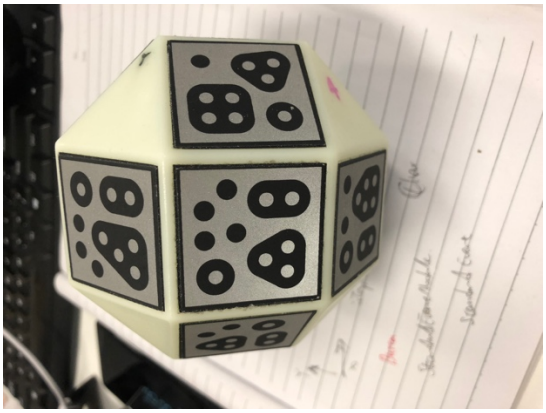


When mobile runs slide in app, screen is divided into two parts for eyes, the image reflects to the len and then seen by eyes. This is the principle of how stereo rendering is achieved.

About marker: we support tracking on single marker:

Or composite marker table:

Or cube marker group:



Developers specify which item to be tracked by passing calibration profile, the calibration profile defines the shape and data that the VPU senses around the environment. For example, if you want to track the cube, you need to pass a calibration profile specifically for cube tracking.

Note: due to marker ID limitation, calibration profiles could be conflict, for example, a standard cube occupies marker ID 1 – 18, so that you can not load profiles fall into the duplication range.

SLIDE IN SDK

SDK available at:

https://github.com/Ximmerse/SlideInSDK

SDK prerequisites:

- Unity 3.0f1 or Unity 4.14f1 (We will update the list after fully test on Unity 2018)
- Android SDK
- Vysor (https://www.vysor.io) : a tool for remote operation on android, not mandatory, but highly recommend for efficient development.
- Samsung S8 or S9.. Up to the date of writing this doc, phone support list is limited to this two. In theory you can use any Android mobile as long as the size fit it, but the left and right screen rect might not be correct size.
- At least one marker.

What's in the package:

_DemoAssets : resources for demonstrating usage of the codes.

Controller Management: source code for blue tooth management tool.

Plugins:

Android/x86_64 : libraries for accessing slide in hardware, for Android/Windows (Mac is not supported)

SlideInSDK: HLAPI (high level API) for developing, this is the major folder of slide in sdk.

LunarConsole : a free plugin for browsing logcat logs. This is an optional folder, you can remove it if you don't need it.

PEPlugins : library to provide infrastructure supports for HLAPI, along with helper tools for development.

GETTING STARTED

If you wants to start from scratch, copy these folder to your project :

plugins/Android.

Plugins/x86_64,

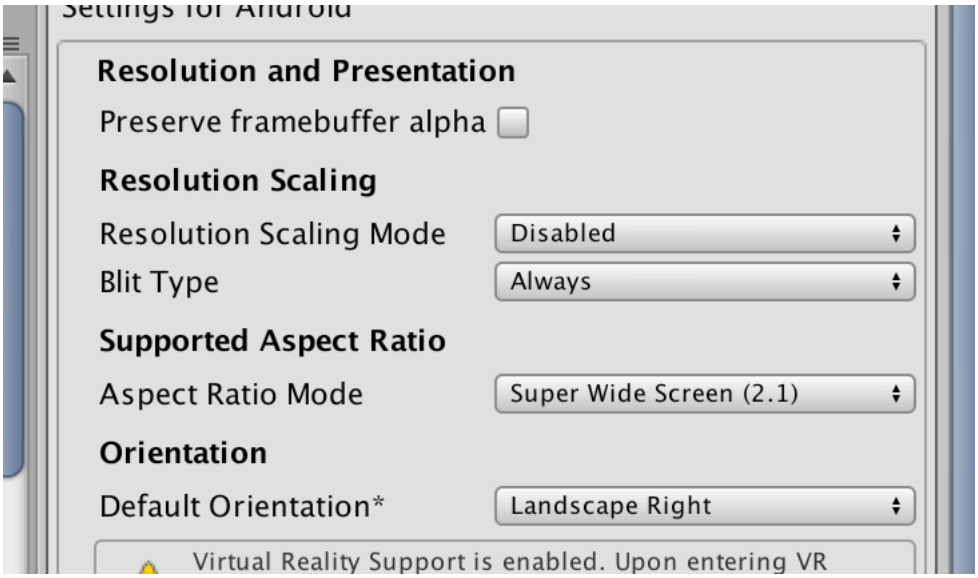
Plugins/PEPlugins

Plugins/SlideInSDK.

Setup your project setting:

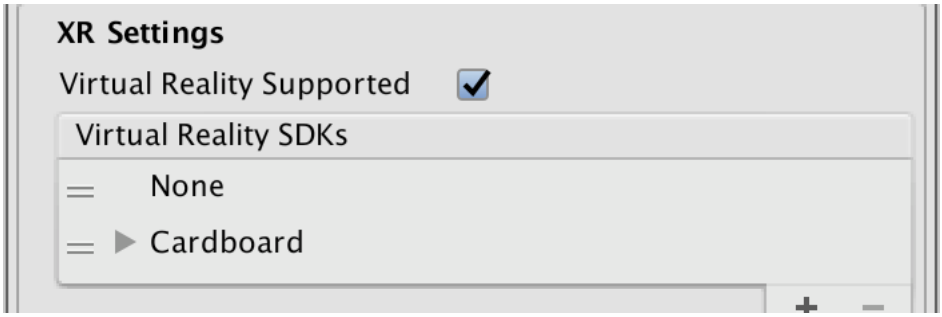
Default orientation set to landscape right.

For unity 3.0 f1, turn to “Debug” inspector to access this property , it was disable in normal inspector view:

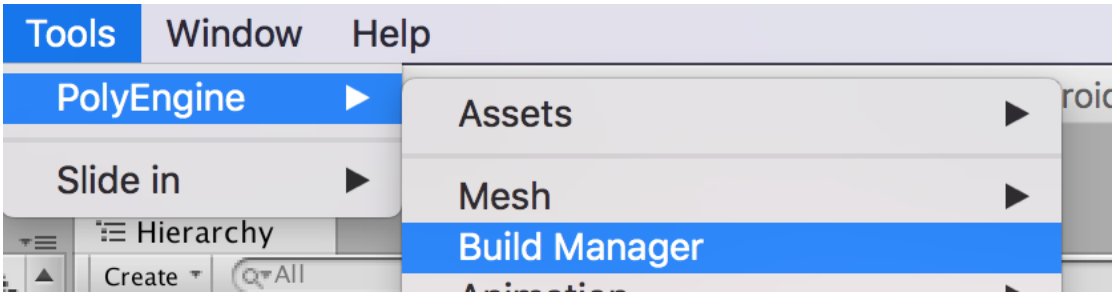


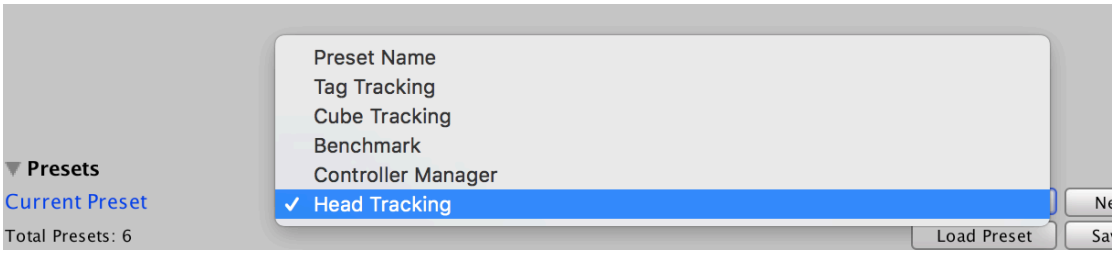
XR Setting following this pattern:

We need google cardboard for head tracking:



SDK offers a quick setup tool “Build Manager”



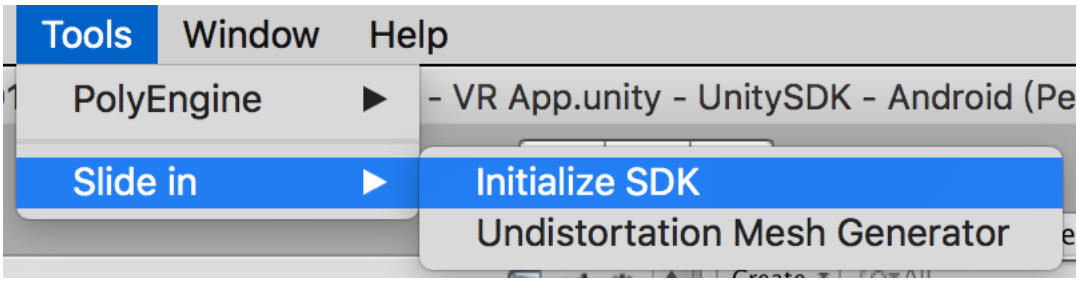


Simply choose the predefined build preset and click “Load Preset” , the build package ID, app name and google VR dependencies will be setup.

You can create your own preset by “New preset”.

Finally, setup SDK via : Tools/Slide in/Initialize SDK

This action will setup the necessary built-in shaders:



Build and deploy the app to check out how slide in works !

HELLO WORLD:

Now let’s start a hello world scene to track a marker, assumes that you have a small marker ID = 1.

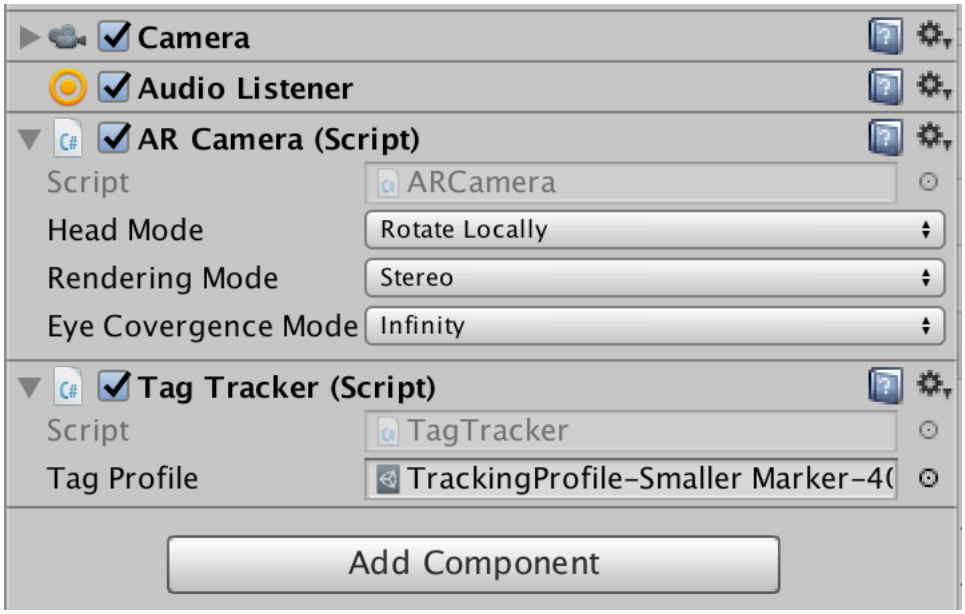
In a new empty scene:

1. Add “AR Camera” to the main camera gameObject

2. Add “Tag Tracker” to the main camera gameObject.

3. Drag “TrackingProfile-Smaller Marker-40mm” to tag tracker “Tag Profile” property.

Your MainCamera object should looks like this:



4. Creates a dynamic marker via menu: GameObject/Create Other/Ximmerse/Slide In SDK/Dynmaic marker target, set marker ID on MarkerIdentity component.

5. Create a quad (GameObject/3D Object/Quad), scale to [0.04, 0.04, 0.04], then attaches to the dynamic marker target object you created at step 4.

Build the scene and deploy to phone, make sure the app have write SDCard permission, run it.

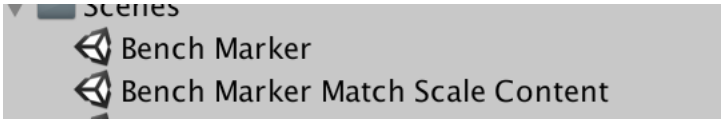
BENCH MARKER:

In the previous example you create a dynamic marker. Dynamic marker translates its position and rotation, while the camera remains unchanged.

Bench marker behaves opposites to dynamic marker, bench marker’s transform would not change during tracking, instead it changes ARCamera’s transform according to its relative position / rotation to the VPU.

Bench marker is useful to act as “Ground”.

There’re build in demo scenes for bench marker usage:



By scaling a bench marker's scale, you can map the camera's position/rotation by content scale. See the second demos above, it maps the bench marker to a 10 x 10 house.

Bench marker show case: camera's position to match the content scale.

DEVELOP IN WINDOWS:

Tag tracker module supports developing in windows, please install the hardware driver and then you can access the tag tracker data directly in windows unity editor or windows unity player.

Note: the driver is not very stable for windows, developing in unity windows editor might lead to random crash.

DEVELOPER UTILITY:

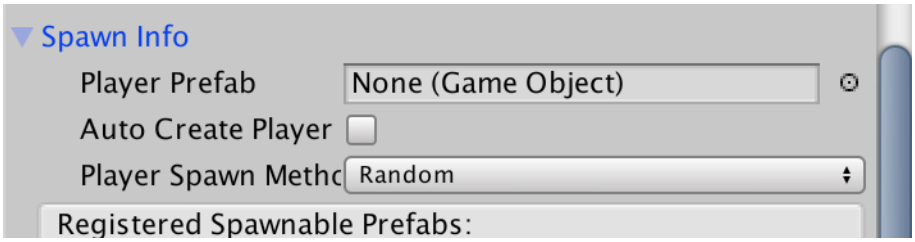
As well as SDK, we deliver tool for helping developer getting start easily. These tools is embedded in PolyenginePlugins library.

Remote sync transform:

Tools to sync client 's object's transform information to unity editor server. Developer have an intuitive knowledge on virtual game object's position and rotation in unity editor.

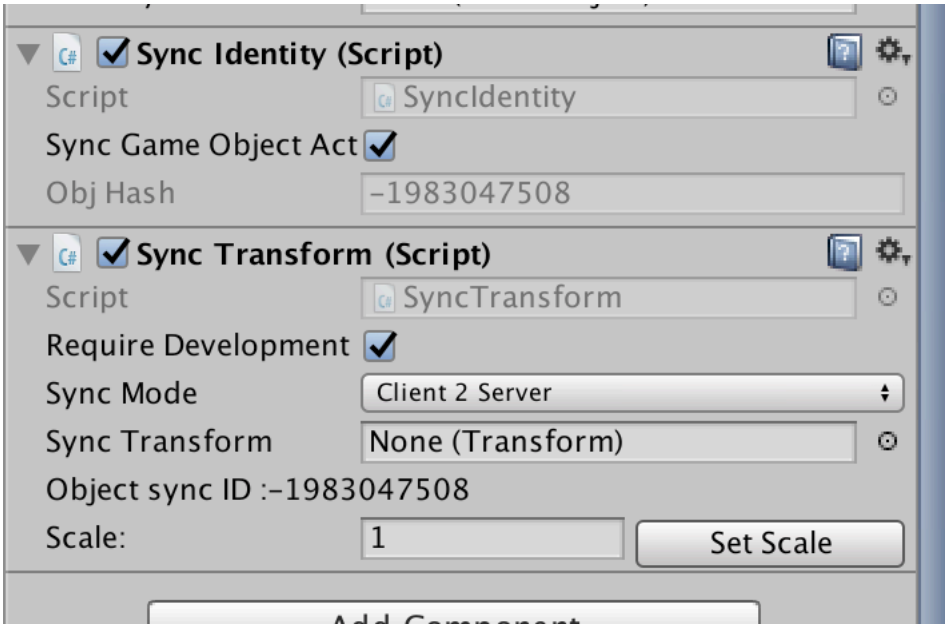
1. Create a gameobject , add "Network Sync" component, it will automatically create a NetworkManager component.

You may want to uncheck the "Auto create player" from Network Manager as it will throw out warning message if you don't assign a player prefab



1. Create a gameobject , add "Network Sync" component, it will automatically create a NetworkManager component.

2. Add component "SyncTransform" to the gameobject where you wants to browse in unity editor, a "SyncIdentity" component will be added too.



3. Build and starts app in slide in, make sure phone and unity editor runs .in same WiFi network, you'll see the transform with "Sync transform" is synchronized in unity editor.

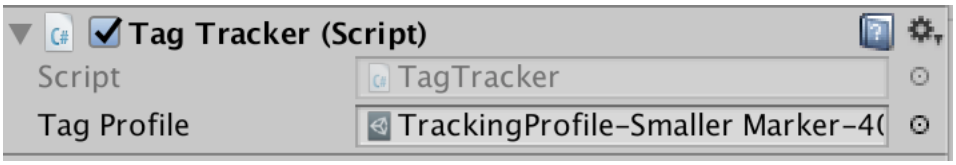
Lunar is very helpful tool to browse logout output directly at phone, you just need to drag "Lunar" prefab to your scene and that's it.

For more about Lunar : https://assetstore.unity.com/packages/tools/gui/lunar-mobile-console-free-82881

MARKER CALIBRATION FILE

At folder Assets/Plugins/Android/assets, there're .json and .data files , the .dat files are binary calibration data for VPU to recognize the marker, the json file maps marker's ID and size.

Developers are not supposed to modify the json and dat files manually. SDK offers class MarkerTrackingProfile to wrap calibration files to pass to low level system. Simply drag MarkerTrackingProfile object to TagTracker :



SDK deliver with these marker profiles:

TrackingProfile-Benchmarker-100mm-34	For tracking table maker group (ID 34)
TrackingProfile-Benchmarker-100mm-35	For tracking table marker group (ID 35)
TrackingProfile-Benchmarker-100mm-36	For tracking table marker group (ID 36)
TrackingProfile-Controller-05-B	For tracking controller marker group (ID = 40)
TrackingProfile-Controller-06-B	For tracking controller marker group (ID = 40) Note: 05B is conflicted to 06B
TrackingProfile-Cube	For tracking cube marker group (ID = 41)

TrackingProfile-Smaller Marker-40mm	For tracking single markers (ID = 0 – 17)