Nibiru XR SDK
Unity Developer Guide

V2.5.0

Nanjing Ruiyue (Nibiru) Technology Co., Ltd.

January 2022, Nanjing

Legal Notices

Nanjing Ruiyue (Nibiru) Technology Co., Ltd.
Tel: +86 (025) 89635828
Address: Floor 4, Building 4, Software Valley, No.57, Andemen Street, Yuhuatai District, Nanjing, Jiangsu Province, China.
E-mail: support@inibiru.com
Web: http://dev.inibiru.com
Tech Support QQ Group: **464811686**

Document Updates

Version	Updates
2.5.0	1. Add APIs: a) API to enable/disable Wi-Fi and get its status b) API to enable/disable Bluetooth and get its status c) API to manage controllers d) APIT to install/uninstall APK 2. Fix known bugs
2.4.0	1. Add Xvisio-related APIs
2.3.0	1. Add support to internal reference 2. Integrate SLAM plug-in into XR SDK
2.2.0	Ass SLAM plug-in
2.0.1	1. Support Universal RP. 2. Fix known bugs.
2.0.0	1. Add dynamic-loading functions for controller model. 2. Add signature verification mechanism. 3. Support arm64. 4. Optimize performance.
1.0.4	1. Support Unity version of 5.4.0f3 or higher. 2. Fix known bugs.
1.0.2	1. Optimize key-value interface for easier calling. 2. Add permission request API and examples of object dragging. 3. Optimize document structure. 4. Fix known bugs.
1.0.1	1. Optimize controller-related APIs and script performance. 2. Update version to Unity2018.4.0 LTS. 3. Fix known bugs.
1.0.0	Release version 1.0.0.

Contents

1 Introduction

1.1 Introductions to SDK

Nibiru XR SDK (hereinafter referred to as SDK) is a set of kits for Nibiru VR/AR OS development. It provides Nibiru VR/AR HMDs with VR/AR data APIs to make VR/AR game and application development quick and easy. XR SDK has the following features:

1) Left and right split-screens. It renders left and right eye images respectively on left and right split-screens while keeping 3D depth of field;
2) Head tracking. The head tracker supports low-latency tracking based on the optimization process in Nibiru HMDs;
3) Anti-distortion. The processing of anti-dispersion and anti-distortion depends on the distortion lens parameters, ensuring distortionless display in Nibiru HMDs;
4) Parameters displayed in VR/AR HMDs. SDK dynamically loads and displays the parameters at runtime, and applications can obtain a consistent display without dealing with different parameters from different devices;
5) Auxiliary features. SDK will gradually improve various auxiliary features, including Nibiru Key for head position resetting, object aiming, button-press listener processing, FPS statistics, etc.
6) Unique DTR technology. It allows the full frame image without dropped frames.

If you have any technical problems in the adaptation, please visit Nibiru Developer Community for solutions (http://dev.bbs.inibiru.com/)

Other contact information:

Tech Support E-mail: support@inibiru.com

Tech Support QQ Group: **464811686**

1.2 Introduction to the document

This document introduces how to utilize Nibiru XR SDK to enable 3D rendering, sensor infusion and controller.

It also describes the structure of SDK and the functionalities of APIs.

This document covers some frequently asked questions. Please read the guide carefully before using the SDK so as to increase efficiency.

2 Supported Devices

SDK supports all Nibiru OS AR/VR HMDs.

3 Development Environment Requirements

Unity201845.f1/Unity2019.3 is supported. Unity2018.4.0.f1 is recommended.

Unity2019.4.x LTS is not recommended because memory leak was detected during tests.

We recommend developers to use LTS version for its higher stability. Given the changes in different versions of Unity, please try the version we recommend if you encounter with any problem while using a version which is not recommended.

3.1 Android development environment configurations

Software Name	**Software Version**
JDK	JDK 1.8.0 and above
Android SDK	API Level 19 and above

3.2 URP development environment configurations

Software Name	**Software Version**
Unity	2019.3.6
URP	7.1.8

4 Development Notes

4.1 Stimulation on Android phones

Your application can only run on Android phones in a single screen manner if you don't use Nibiru VR/AR HMDs. For debugging and development, please start the application through Nibiru XR launcher. The setup file for the Nibiru XR launcher is NibiruXRLauncher.apk in the SDK. To make the launcher identify the developed applications properly, you need to check whether the category (<category android:name="*com.nibiru.intent.category.NVR*" />) is declared in AndroidManifest.xml.

If activity is in a VR format, mark a NVR tab: com.nibiru.intent.category.NVR.

If the activity is 2D interface, mark a 2D tab: com.nibiru.intent.category.2D.

If it is in a system rendering interface and in TV mode, mark a TV system rendering tab: com.nibiru.intent.category. SYS_RENDER_TV.

If it is in a system rendering interface and in phone mode, mark a PHONE system rendering tab: com.nibiru.intent.category.SYS_RENDER_PHONE.

We strongly recommend you to develop and debug your applications with Nibiru VR HMDs.

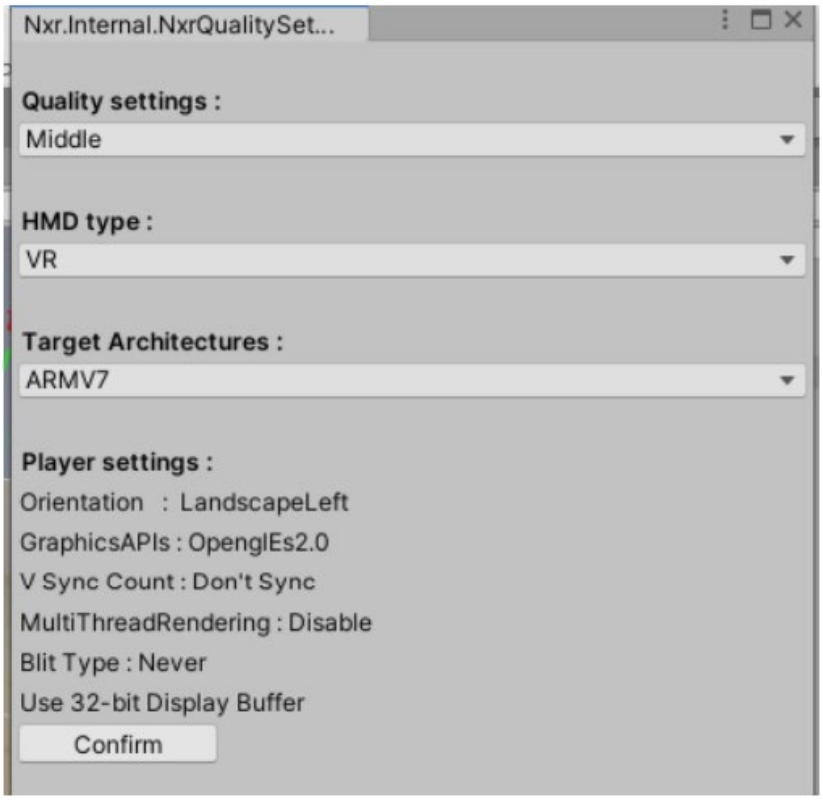
4.2 Unity notes

Please note:

FPS frame rates may vary between Unity versions.

In the running mode of Unity Editor, using Alt+Mouse is the same as rotating the head and moving Ctrl+Mouse is the same as rotating head around Z axis.

SDK provides one-click XR configuration in Nibiru XR - XR Settings of the Editor and its interface is as follows:



Quality Settings includes Low/Middle/High

Low: Shut down Anti-Aliasing

Middle: 2 times Anti-Aliasing

High: 4 times Anti-Aliasing

HMD Type You need to select current target device type

VR: application for VR headsets AR: application for AR headsets

Target Architectures Packaging 32 or 64 bits: the default is 32 bits.

The 64-bit program needs system support to run normally.

Player Settings Modify the default configuration parameters:

Orientation: LandscapeLeft

GraphicsAPIs: OpenglEs2.0

Use 32-bit Display Buffer

Click Confirm to change configuration automatically.

4.3 AndroidManifest document

If AndroidManifest document is not included in the application, please use that one in the SDK.

If it is included in the application, please merge the two.

- Activity needs to be extended from:
  com.nibiru.lib.xr.unity.NibiruXRUnityActivity.

Internet-filter needs to add:

```
<category android:name="com.google.intent.category.CARDBOARD" />
<category android:name="com.nibiru.intent.category.NVR" />
<category android:name="com.nibiru.intent.category.STUDIO" />
```

- Plug-in declarations:

```
<!-- "6DOF", "RECORD", "MARKER" !-->
<meta-data android:value="6DOF " android:name="NIBIRU_PLUGIN_IDS"/>
<meta-data android:value="NxrViewerMain" android:name="NIBIRU_UNITY_VIEWER_NAME"/>
```

- Get target device type:

```
<!--0=vr,1=ar-->
<meta-data android:value="VR" android:name="HMD_TYPE"/>
```

- Get an encrypted version:

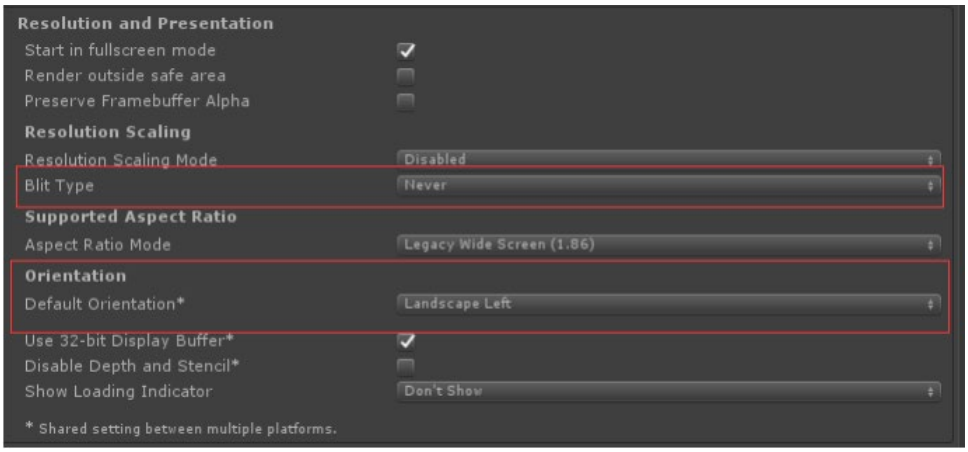
```
<!--The current APK is an encrypted version —⟩
<meta-data android:value="0" android:name="NIBIRU_ENCRYPTION_MODE"/>
```

- Add necessary permissions:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/⟩
<uses-permission android:name="android.permission.BLUETOOTH"/⟩
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" /⟩
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/⟩
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" /⟩
<uses-permission android:name="android.permission.INTERNET" /⟩
<uses-permission android:name="android.permission.GET_TASKS" /⟩
<uses-permission android:name="android.permission.CAMERA"/⟩
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/⟩
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/⟩
<uses-permission android:name="android.permission.WRITE_SETTINGS"/⟩
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/⟩
```

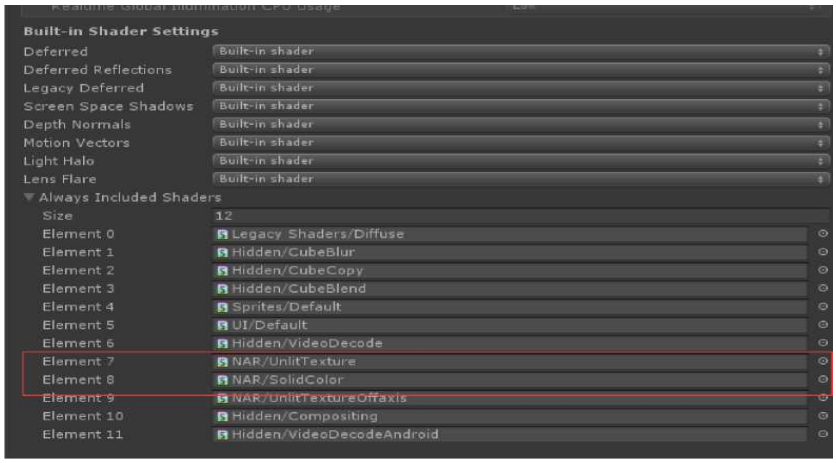
4.4 Recommended configuration for Unity project settings

- Graphics APIs do not support Vulkan for the time being. For OpenGLES2 and OpenGLES3, the developers need to choose according to their needs.
- MultiThread Rendering has not been supported yet. Please do not select.
- In Default Orientation under Unity Player Setting, select Landscape Left (Required). The Blit Type under Resolution Scaling needs to be set as “Never”.



- V Sync Count in Quality should be set as Don’t Sync.

- Edit/Project Settings/Graphics, add NXR/Resource/UnlitTexture.shaderand NXR/Resource/SolidColor.shader into Always Included Shaders.

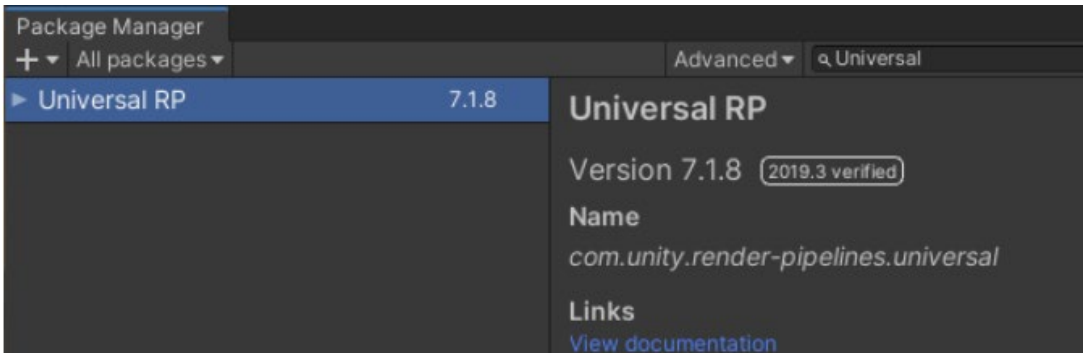


4.5 Universal RP (universal rendering pipeline) configuration instructions

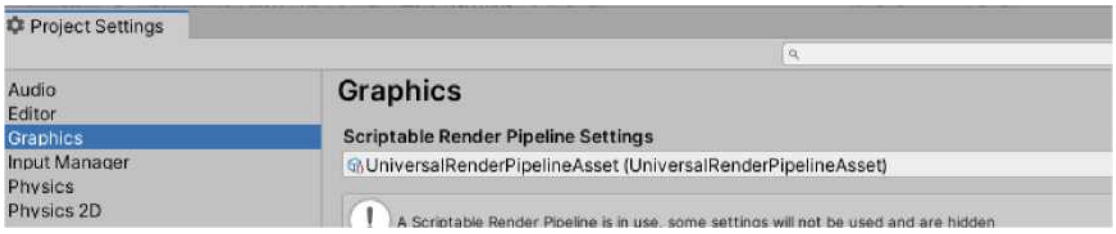
Unity Version: **Unity2019.3.6**

URP Version: **V7.1.8**

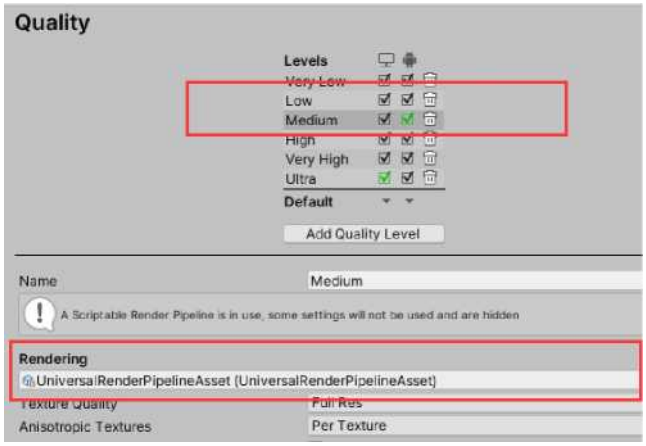
- Open Window/Package Manager, search for Universal RP, and click Install.



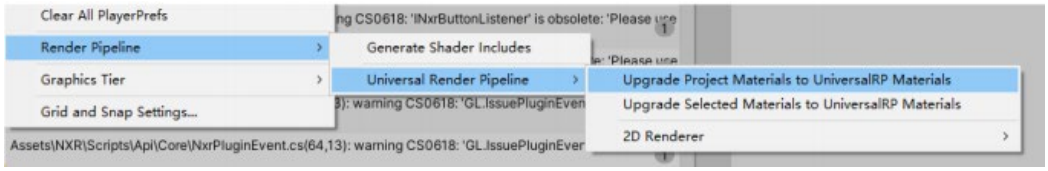
- After installation, create UniversalRenderPipelineAsset, Assets/Create/Remdering/Universal Render Pipeline/Pipeline Asset.



- Open Edit/Project Settings/Graphics/ and select the created PipelineAsset in Scriptable Render Pipeline Settings.



- Upgrade Engineering Materials



Note:

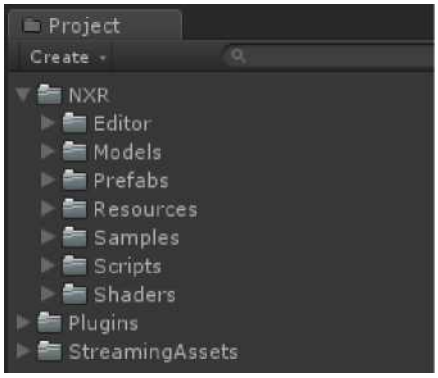
Cancel the SPR Batcher/Dynamic Batching in UniversalRenderPipelineAsset.

Otherwise, memory leaks can occur.

5 Quick Start of SDK Access

5.1 Create a new project and import plug-in

Import NibiruXRSDK_PRO_V1.0.0.unitypackage into the Unity project.



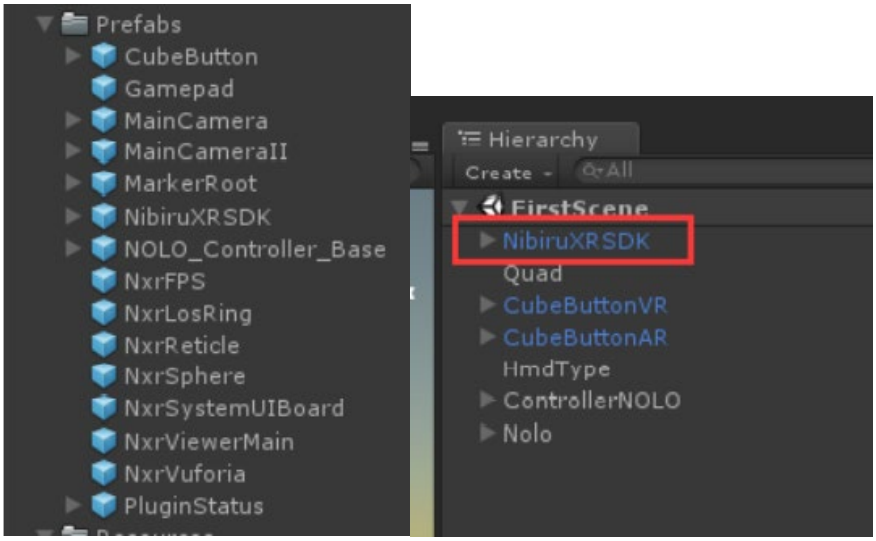
NXR/Samples are demo scenes for your reference.

NXR/Prefabs are prefabs used in the SDK.

NXR/Scripts are script of core functionalities in the SDK.

5.2 The use of prefabs

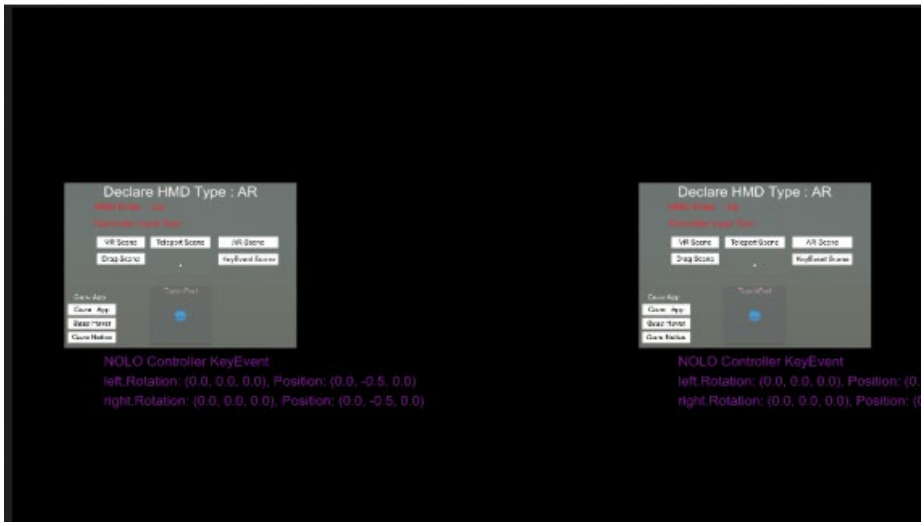
The frequently used prefabs are provided in NXR/Prefabs.



Drag and drop the Nibiru XR SDK prefabs into the scene. Reset the Position and Rotation of Transform components as (0,0,0).

5.3 Simulation run

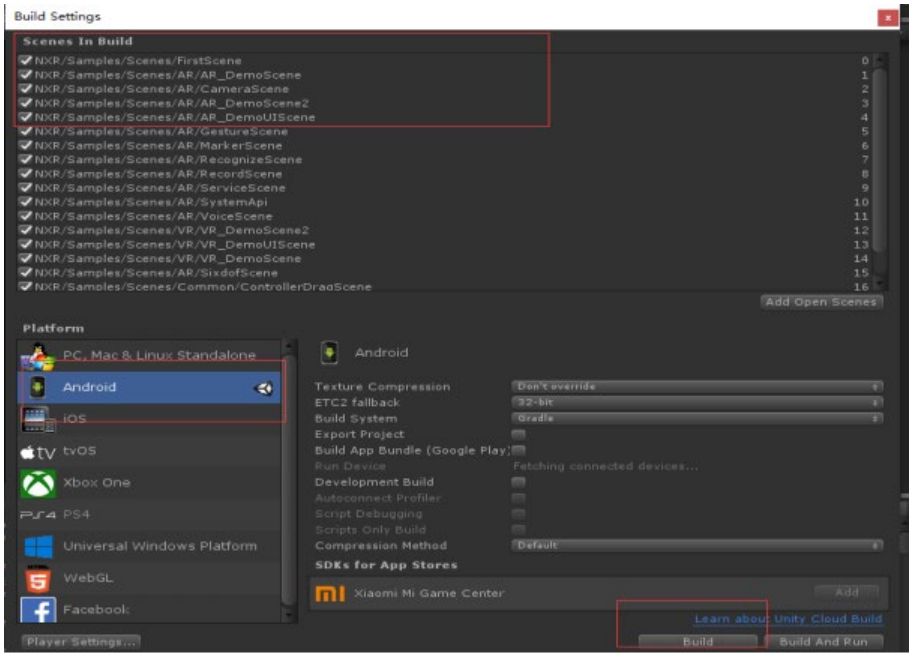
Click the Run button to see the following in the Game window:



5.4 Packaging

Please check the device type and plug-ins currently selected before packaging.

Find NibiruXR in the tool bar at the top of the Editor. Enter XR Settings/Plugin Manager to adjust related configurations as needed. Go to File->Build Settings…, add scenes, switch the platform to Android, and click Build to generate APK and install it to the HMD.



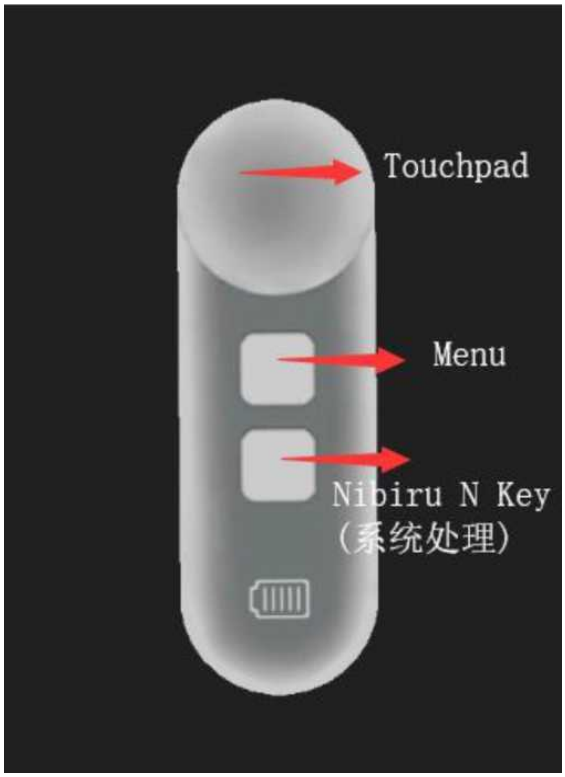
6 HMD and Controller Interaction

6.1 Keys on HMD

Corresponding table between keys on HMD and key values in Unity:

HMD Keys	Android Key Value	Unity Key Value
OK	23/66	10 or KeyCode.JoystickButton0
Up	19	KeyCode.UpArrow
Down	20	KeyCode.DownArrow
Left	21	KeyCode.LeftArrow
Right	22	KeyCode.RightArrow
Back	4	KeyCode.Escape
F1	57	FunctionKeyCode.NF1
F2	130	FunctionKeyCode.NF2

6.2 Keys on controller



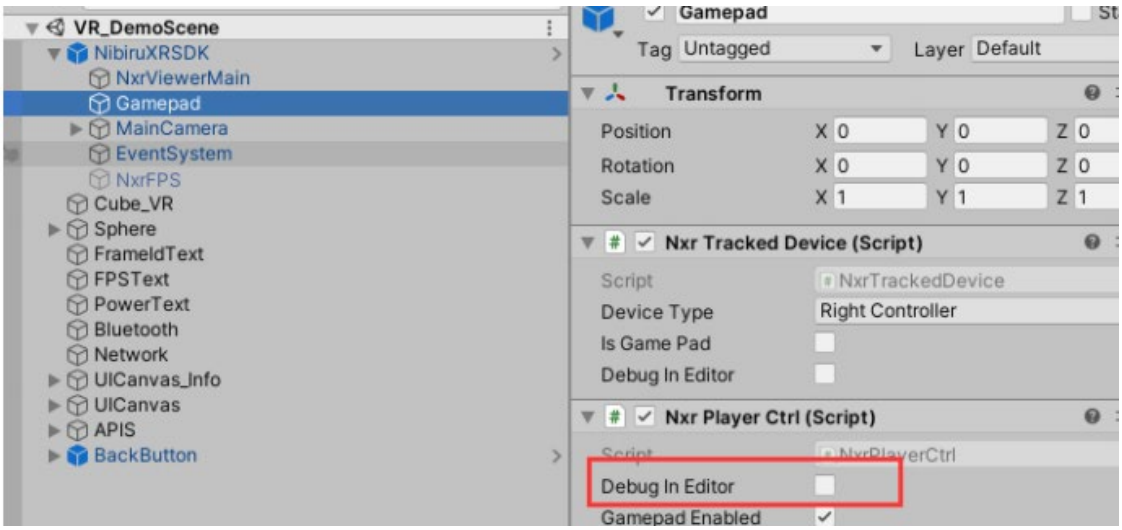
The coordinate range of Touch in Touchpad:

X-direction, left->right (-1~1); Y-direction, up-down (-1~1)

//Controller touchpad and touch coordinates

Vector2 pos = InteractionManager.TouchPadPosition;

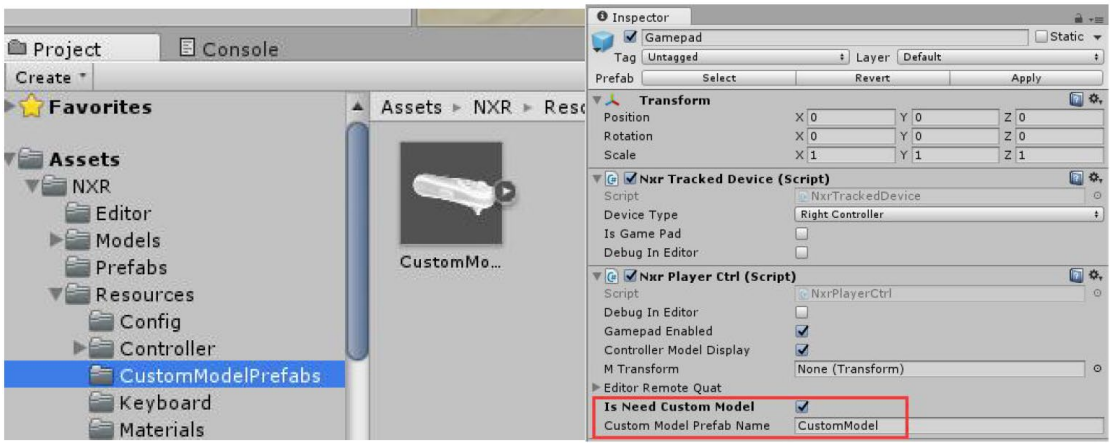
In the Editor, you can select Gamepad at runtime, check DebuglnEditor in NxrPlayerCtrl, and the scene can be loaded with 3DOF models. Developers can perform simple debugging simulations.



In the new version, the SDK will automatically load the matching controller model according to the system configuration. If you want to modify the controller model (Support both 3DOF and 6DOF), you need to make your controller model a prefab (Refer to how “CustomModel.prefab” is created) and put it under “Assets/Resources/CustomModelPrefabs”, select Gamepad, check “IsNeesCustomModel” of “NxrPlayerCtrl” to enable the function of customizing controller model, and set your CustomModelPrefabName.

[Sample scene: NXR/Samples/Scenes/MainScenes/CustomCtrlScene]

Please note: If there are too many scenes to modify, you can change “IsNeesCustomMode” and “CustomModelPrefabName” these two parameters in “NxrPalyerCtrl.cs” script.



Developers can obtain the corresponding key status through the following interface:

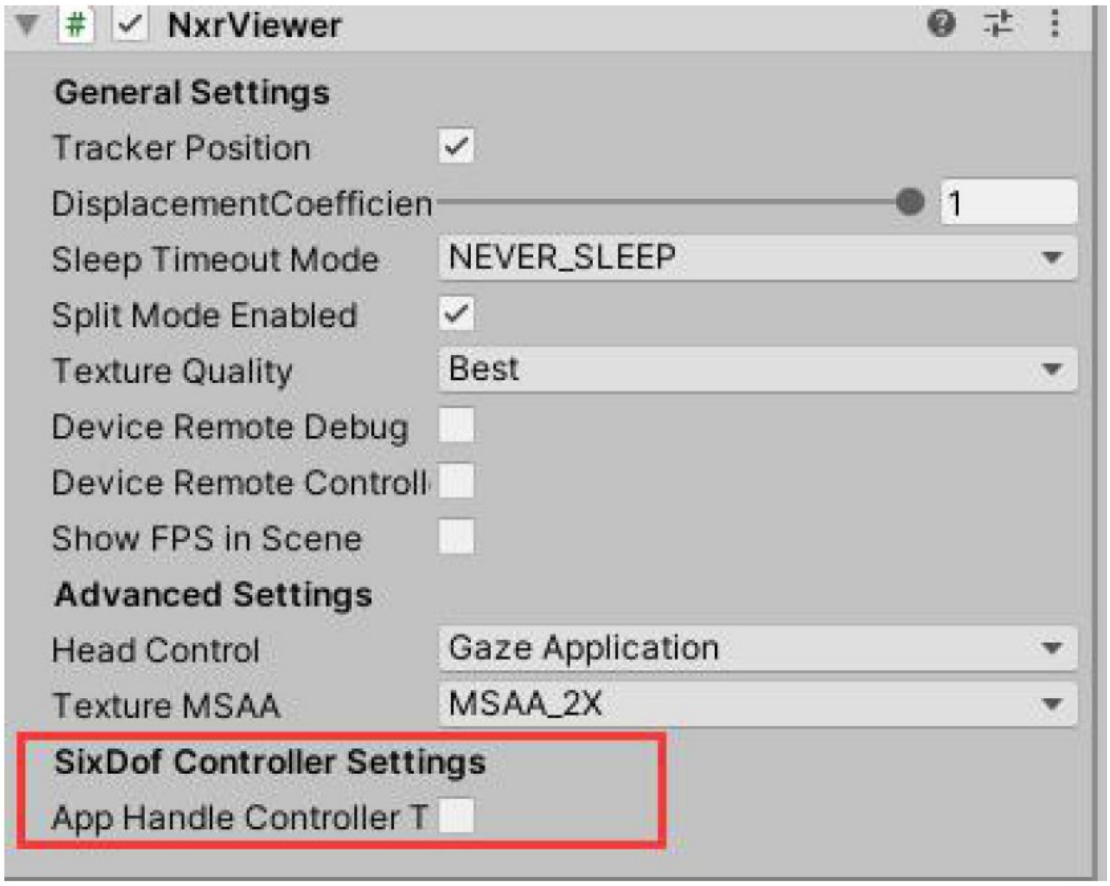
HMD keys pressed	NxrInput. GetKeyDown(int keycode)
HMD keys long pressed	NxrInput.GetKeyPressed(int keycode)
HMD keys released	Nxrlnput.GetKeyUp(int keycode)
3DOF controller keys pressed	NxrInput.GetControllerKeyDown(int keycode)
3DOF controller keys long pressed	Nxrlnput.GetControllerKeyPressed(int
3DOF controller keys released	Nxrlnput.GetControllerKeyUp(int keycode)
Nolo controller keys pressed	Nxrlnput.GetControllerKeyDown(int keycode, int NACTlON_HAND_TYPE type)
Nolo controller keys long pressed	Nxrlnput.GetControllerKeyPressed(int keycode, int NACTlON_HAND_TYPE type)
Nolo controller keys released	Nxrlnput.GetControllerKeyUp(int keycode, int NACTlON_HAND_TYPE type)

Key value or keycode:

CKeyEvent.KEYCODE_CONTROLLER_TOUCHPAD_TOUCH	Controller Touchpad Event Key
CKeyEvent. KEYCODE_CONTROLLER_TRIGGER	6DOF
CKeyEvent.KEYCODE_CONTROLLER_MENU	Controller
CKeyEvent.KEYCODE_CONTROLLER_TOUCHPAD	Keys on Controller Touchpad

CKeyEvent.KEYCODE_CONTROLLER_VOLUMN_DOWN	Controller volume down
CKeyEvent.KEYCODE_CONTROLLER_VOLUMN_UP	Controller volume up
CKeyEvent.KEYC0DE_3D0F_C0NTR0LLER_TRIGGER	3DOF

Note: The Trigger key will be used as Confirm key by default in the SDK. Current key value: CkeyEvent.KEYCODE_CONTROLLER_TRIGGER_INTERNAL. If the current application needs to use the Trigger key, you can check “App Handle Controller Trigger Event” in “NxrViewer” configuration to receive Trigger recall.



7 API Interface Function List

7.1 Adjust camera FOV parameter

Functionality	Function name and calling method
Update the FOV for both eyes, range: [5,80]	NxrViewer.Instance.UpdateCameraFov(40);
Restore default FOV	NxrViewer.lnstance.ResetCameraFov();
Update the distance of the far clip plane (called in Start())	NxrViewer.Instance.UpdateCameraFov(2000f)

7.2 HMD performance-level parameter

Functionality	Function name and calling method
Nibiru HMD performance levels (three levels, including LOW, NORMAL and HIGH, are divided by NxrGlobal.PERFORMANCE)	NxrGlobal.platPerformanceLevel

7.3 Startup image configuration reference

Functionality	Function name and calling method
Startup split screen reference	NXR/Resource/ directory provides a startup image reference: Nibirulaunch_Black_1K.png/Nibirulaunch_Black_2K.png.

7.4 Soft keyboard input controls

Support uppercase and lowercase letters, numbers and punctuation marks.

Functionality	Function name and calling method
Set where the strings are required to be displayed in the Text component	NibiruKeyBoard.Instance.SetText(text);
Get keyboard Transform	NibiruKeyBoard.Instance.GetKeyBoardTransform();
Get strings entered in the keyboard	NibiruKeyBoard.Instance.GetKeyBoardString();
Display keyboard: pageIndex=0 for letter, pageIndex=1 for number	NibiruKeyBoard.Instance.Show(0, new Vector3(0, - 0.3f, 1), new Vector3(30, 0, 0));
Display the keyboard in the default position	NibiruKeyBoard.Instance.Show();

7.5 Get data of head posture

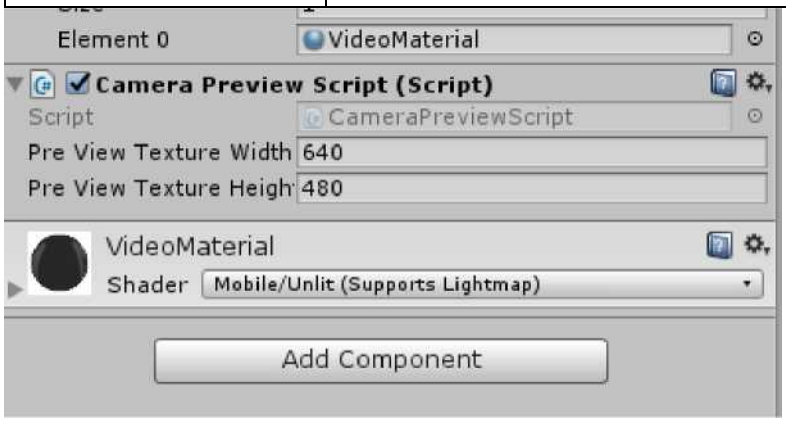
Functionality	Function name and calling method
Get data of current head posture	NxrViewer.Instance.GetHead().transform

7.6 Camera preview image

Demo scene: NXR/Samples/Scenes/AR/CameraScene

Functionality	Function name and calling method
Registration status callback	NibiruService.OnCameraIdle += CameraIdle; NibiruService.OnCameraBusy += CameraBusy;

Query status	int cameraId = 1; (It may be different for cameras of different devices: 0 usually for rear camera, 1 usually for front camera) NxrViewer.Instance.GetNibiruService().GetCamera Status(cameraId);
Call preview image	//1=using front camera (HMD), 0=using rear camera (handheld host) NxrViewer.Instance.GetNibiruService().StartCamera PreView(cameraId);}
Pause preview image	nibiruService.StopCamereaPreView();



In the demo scene, the preview image is displayed on Panel, and CameraPreviewScript.cs should be mounted to GameObject. You can modify the width and height of preview texture lest it impacts performance (aspect ratio should be kept as 16:9).

Note: If the preview image is black, please modify cameraId.

The hardware should support camera to use this API. If there's any problem, please check the running LOG or whether there's camera permission under Assets\Plugins\Android\ AndroidManifest.xml:

<uses-permission android:name="android.permission.CAMERA"/>

The permission is added in the SDK by default.

7.7 Gaze point control and selection

Demo scene: NXR/Samples/Scenes/FirstScene

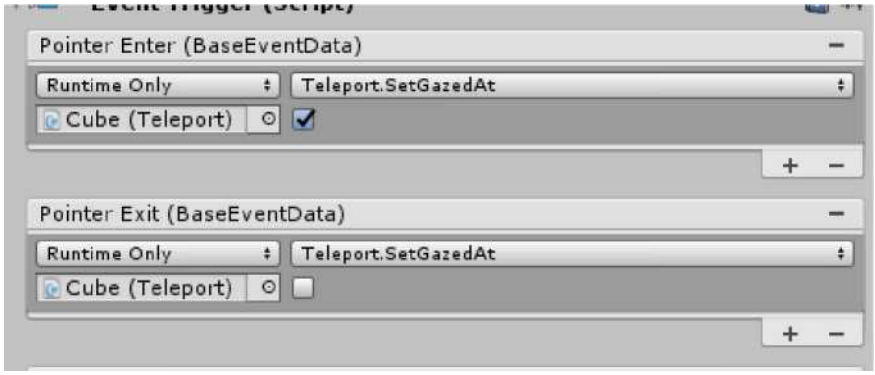
Functionality	Function name and calling method
System gaze point	NxrViewer.Instance.HeadControl = HeadControl.GazeSystem;
APP gaze point	NxrViewer.Instance.HeadControl = HeadControl.GazeApplication;
APP gaze point Hover	NxrViewer.Instance.HeadControl = HeadControl.GazeHover;

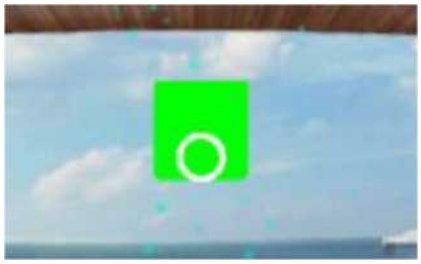
Functionality	Function name and calling method
Change size of system gaze point	NxrViewer.Instance.GazeApi(GazeTag.Set_Size, ((int) GazeSize.Larger).ToString());
Change color of system gaze point	NxrViewer.Instance.GazeApi(GazeTag.Set_Color, "0.043f_0.435f_0.043f"); RGB Range (0~1)
Change visibility of system gaze point	NxrViewer.Instance.GazeApi(GazeTag.Hide);

Due to the effect of interpolation frame, APP gaze point will jitter, but the system gaze point will not.

Now the object, such as the Cube in the scene, can be aimed at, how to monitor this aiming event? SDK provides an API INxrGazeResponder.cs, through which you can monitor the Entry, Exit, and Confirm events during the aiming

In Demo, we provide a Teleport.cs demo, mount it on the Cube object, and add Event Trigger component for Cube, and the event types of Pointer Enter, Pointer Exit and Pointer Click under the Event Trigger.





If the Cube above is selected, its color will change. If Cube is selected, pressing OK will change the position randomly.

For click event from Button component of Canvas, you can directly add processing logic to the On Click event.

7.8 Key function interface

Demo scene: NXR/Samples/Scenes/Common/InputKeyScene

Functionality	Function name and calling method
HMD Key	NxrInput.GetKeyDown(int keycode) NxrInput.GetKeyPressed(int keycode) NxrInput.GetKeyUp(int keycode)
3DOF Controller Key	NxrInput.GetControllerKeyDown(int keycode) NxrInput.GetControllerKeyPressed(int keycode) NxrInput.GetControllerKeyUp(int keycode)
Nolo Controller Key	NxrInput.GetControllerKeyDown(int keycode, int NACTION_HAND_TYPE type) NxrInput.GetControllerKeyPressed(int keycode, int NACTION_HAND_TYPE type) NxrInput.GetControllerKeyUp(int keycode, int NACTION_HAND_TYPE type)

7.9 View lock and interface resetting

Functionality	Function name and calling method
View lock	NxrViewer.Instance.LockHeadTracker=true;
View resetting	NxrViewer.Instance.Recenter();

7.10Texture quality

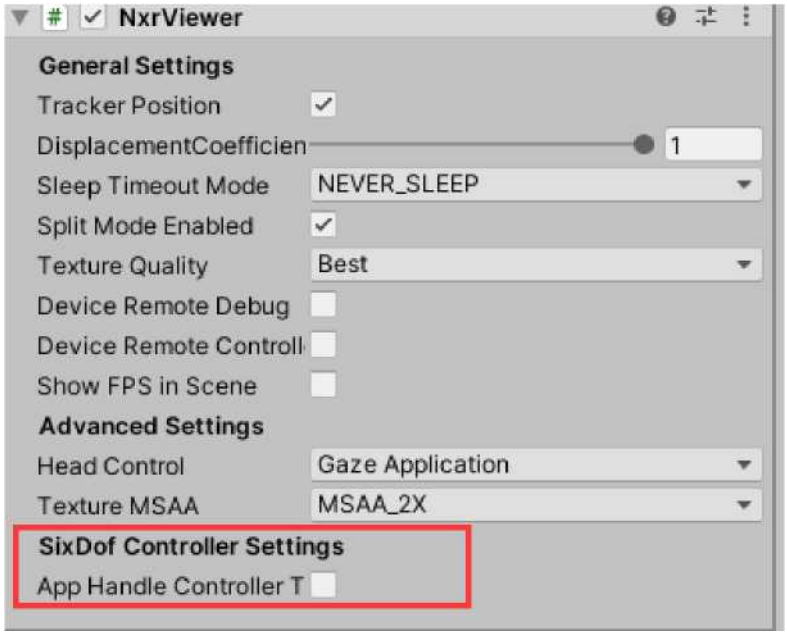
In NxrViewer, you can see the Texture Quality option, the default is “Good” (normal image quality without fuzziness). Other options are Simple (the lowest image quality with fuzziness) and Best (the clearest image with the highest quality).

Notes: It’s better to make adjustments in NXR/Prefabs/NxrViewerMain. It is appropriate to modify a separate scenario in the Editor.

7.116DOF controller

If the 6DOF dual controllers are connected, the SDK will automatically display the model for interaction.

By default, the controller Trigger button is used as an confirm button in the SDK. If the current application requires a Trigger button, the App Handle Controller Trigger Event can be checked in the NxrViewer configuration to receive a Trigger button response.



7.12System-related APIs

Functionality	Function name and calling method
Windowing System: suitable when the whole scene is 2D UI instead of SDK.	NxrViewer.Instance.SetSystemSplitMode(flag); (flag=1: system divides the current scene into left and right screens. flag= 0 The split-screen effects will be the same as mounting our camera script)
Get Android SD card path, e.g. /storage/emulated/0	NxrViewer.Instance.GetStoragePath();
Virtual mouse registration services, only applicable for 2D apps in common split-screen mode	NxrViewer.Instance.GetNibiruService().RegisterVirtualMouseService
Virtual mouse logout service	NxrViewer.Instance.GetNibiruService().UnRegisterVirtualMouseService
Virtual mouse control	NxrViewer.Instance.GetNibiruService().SetEnableVirtualMouse(bool enabled)
Get Bluetooth status while network status: 0 = no network, 1 = network available	NibiruTaskApi.GetNetworkStatus()
Get network status while Bluetooth status: 0 = off, 1 = on	NibiruTaskApi.GetBluetoothStatus()
Set IPD (called in Start or use keys to trigger)	NxrViewer.Instance.SetIpd(0.060f);
Get IPD	NxrViewer.Instance.GetIpd();
Get MAC address	NibiruTaskApi.GetMacAddress();
Get device ID	NibiruTaskApi.GetDeviceId();
Start an app according to the package name	NibiruTaskApi.LaunchAppByPkgName("com.nibiru.vr.lib2.test");

Open/Close Wi-Fi	NibiruTaskApi.SetWifiEnable(bool enable);
Open/Close Bluetooth	NibiruTaskApi.SetBluetoothEnable(bool enable);
Manage controllers	NibiruTaskApi.GoToControllerDriver();
Install APK	NibiruTaskApi.InstallApk(string apkPath);
Get callback of APK installation (Succeed/Fail)	NibiruTaskApi.SetInstallSuccessCallback(InstallSuccess Callback callback); NibiruTaskApi.SetInstallFailedCallback(InstallFailedCall back callback);
Install and auto launch APK	NibiruTaskApi.InstallAndStartApk(string apkPath);
Uninstall APK	NibiruTaskApi.UninstallApk(string packageName);
Get callback of APK uninstallation (Succeed/Fail)	NibiruTaskApi.SetUninstallSuccessCallback(UninstallS uccessCallback callback); NibiruTaskApi.SetUninstallFailedCallback(UninstallFail edCallback callback);

NibiruService nibiruService = NxrViewer.Instance.GetNibiruService();

Functionality	**Function name and calling method**
Get software version of driver board	nibiruService.GetVendorSWVersion()
Get value of light sensor: (please try not to call frequently)	nibiruService.GetLightValue()
Get value of distance sensor: (please try not to call frequently)	nibiruService.GetProximityValue()
Get screen brightness value	nibiruService.GetBrightnessValue()
Adjust screen brightness	nibiruService.SetBrightnessValue(nibiruService.GetBrightnessValue() - 1);
2D/3D display mode	nibiruService.GetDisplayMode()
Switch 2D/3D display mode	nibiruService.SetDisplayMode(DISPLAY_MOD E.MODE 3D);
Hide/display touchpad cursor	nibiruService.SetEnableTouchCursor(false);
Process Sensor data of the callback	NibiruService.OnSensorDataChangedHandler += onSensorDataChanged; Register and accept acceleration, gyro, geomagnetic data callbacks nibiruService.RegisterSensorListener(SENSOR _TYPE.ACCELEROMETER,SENSOR_LOCATION. CONTROLLER); SENSOR_LOCATION.CONTROLLER: Get Sensor data from the host computer SENSOR_LOCATION.HMD: Get Sensor data from the HMD Cancel and disable API
Get device ID	NibiruTaskApi.GetDeviceId();
Start an app according to the package name	NibiruTaskApi.LaunchAppByPkgName("com.

7.13Advanced services in Pro version

Note: Plug-ins mentioned here have no relations to Unity plug-ins. They represent function APIs like gesture/voice recognition and 6DOF displacement. Pro SDK is paid version. You must purchase the corresponding license to use the related functions.

Minimum requirements for system version

Settings-System-About Device-Version | Version Code

Version Type: MR0001, system version must be Nibiru 1.10.XXX or above

7.13.1 Introduction to authorization

Generate verification file through the authorization tool

(https://dev.inibiru.com/#/Droduct/tools), and import project

After installation, log in, authorize, and enter the following screen:

Step1: Open the authorization tool and start generating Secret Key.

Step2: Fill in the application package name, ensuring it matches the Bundle Identifier in the Unity player settings.

Step3: Specify generating path, click the Generate button, and then paste the Secret Key into Unity after successful generation. Meanwhile, copy NibiruSDKKey.bin from the corresponding path to Assets/Plugins/Android/assets.

Step 4: Click the NibiruXR tab on top of the Unity Editor, and select SDK Verify to enter the Secret Key configuration window. Follow the corresponding steps, place the NibiruSDKKey.bin file, paste the Secret Key string, and finally click OK to complete the configuration.

Step5:Once the Secret Key configuration is complete, you can proceed with the packaging operation.

7.13.2 Verification result for plug-ins

Get the current verification status by NxrGlobal.verifyStatus. If it fails, the current screen will be frozen and keep outputting: “Verify Failed: ... ” LOG. If it’s successful, each plug-in can be used properly.

7.13.3 Plug-in support status

Plug-ins might not be supported in some devices due to the hardware differences, e.g. no Camera or microphone in the device. Please get support status of plug-ins to see if they’re supported.

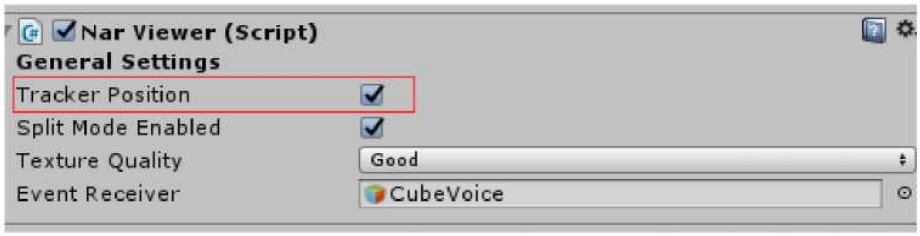
Functionality	**Function name and calling method**
Get the current service connecting status (after the service is connected, you can check the plug-in support status)	NxrViewer.Instance.serviceReadyUpdatedDelegate += new NxrViewer.ServiceReadyUpdatedDelegate(OnServiceReady);
Judge if the plug-in is declared	NibiruTaskApi.IsPluginDeclared(Nxr.Internal.PLUGIN_ID.SIX_DOF)
Judge whether the plug-in is supported	NibiruTaskApi.IsPluginSupported(Nxr.Internal.PLUGIN_ID.SIX_DOF)

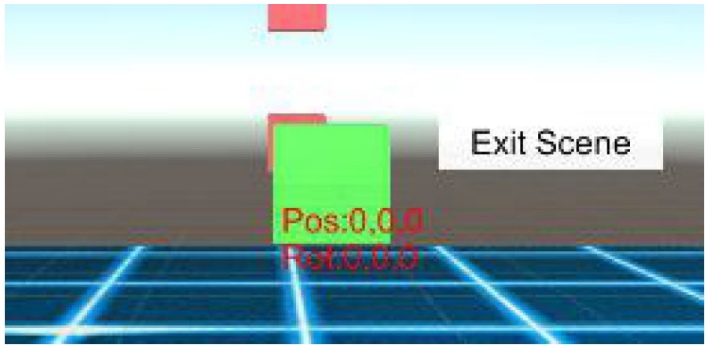
Note: Please first check the plug-in support status before development to ensure the hardware support.

7.13.4 6DOF displacement

Demo scene: NXR/Samples/AR/AR_DemoScene

NXR/Samples/Scripts/Pro/Nibiru6DofTest.cs



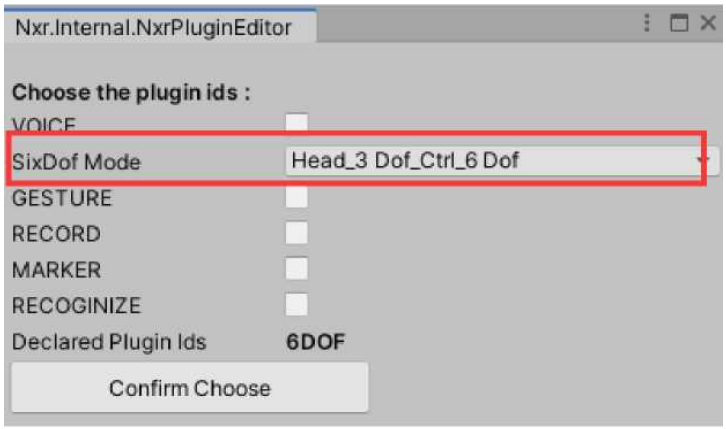


Position: 0,0,0 represents the current displacement info (x,y,z)

Tracker Position option under General Settings of NxrViewerMain means whether to enable displacement feature.

If it is required to be disabled, call NxrViewer.Instance.TrackerPosition=false or modify NXR/Prefabs/NxrViewerMain prefab to disable.

Functionality	**Function name and calling method**
Monitor the change of displacement info	NxrViewer.onSixDofPosition += OnHeadPosition; void OnHeadPosition(float x, float y, float z) NxrViewer.onSixDofPosition -= OnHeadPosition;



For 6D0F, there are three modes available in the plug-in selection interface:

3DOF headset with 3DOF controller, 3DOF headset with 6DOF controller (default),

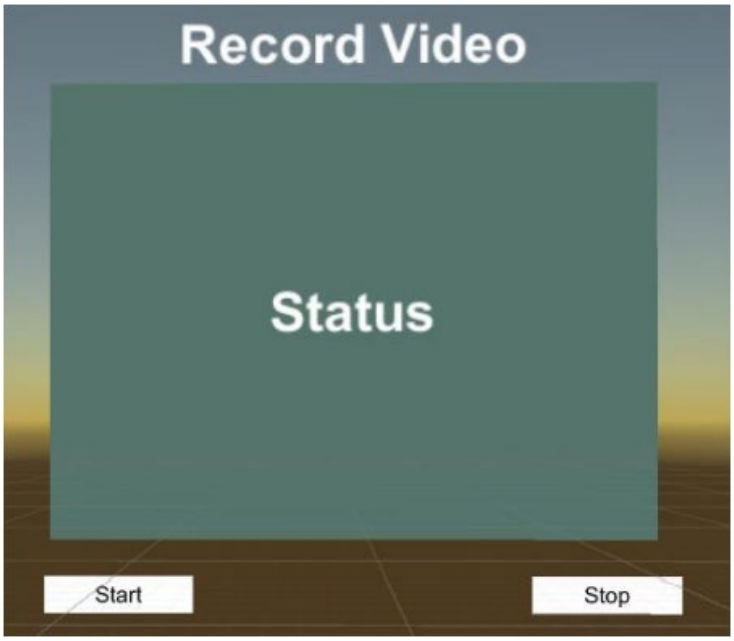
and 6DOF headset with 6DOF controller.

Note: All 6DOF displacements are absolute displacements.

7.13.5 Video recording

Demo scene: NXR/Samples/AR/RecordScene

NXR/Sample/Scripts/Pro/RecordTestScript.cs



Video recording will overlay the current scene and camera images, and output them to the mp4 file.

- Support 480p, 720p and 1080p
- Set front or rear camera
- Set video rate

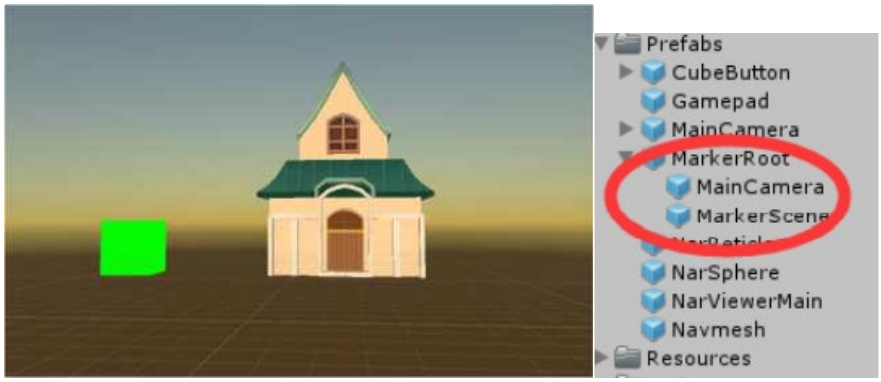
Functionality	**Function name and calling method**
Set camera ID	NibiruService.SetCaptureCamereaId(CAMERA_lD.FRONT);
Set resolution	NibiruService.SetC aptureVideoSize(VIDEO_SIZE.V720P);
Set mp4 file path	string filePath = NxrViewer.Instance.GetStoragePath() + "/unityrecord.mp4";
Start recording	NxrViewer.Instance.GetNibiruService().StartCapture(filePath);
Pause recording	NxrViewer.Instance.GetNibiruService().StopCapture();

(Refer to NXR/Sample/Scripts/Pro/RecordTestScript.cs)

Note:

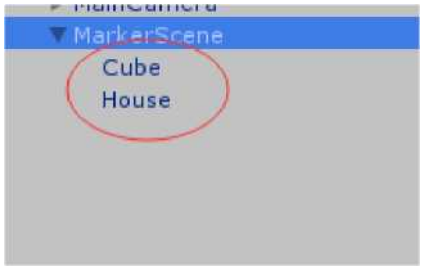
1. **The recorded video content is the overlaying result of rendering image in the scene and the camera preview.**
2. **Camera is needed to enable screen recording. Therefore, gesture/marker recognition and other functions which need camera cannot be used at the same time.**

7.13.6 Marker recognition



Demo scene: NXR/Samples/MarkerScene

Drag MarkerRoot from Prefabs to the scene. You can see a Cube and a house under the directory of MarkerScene. The default is to not display them, but after successful marker recognition, they will show up automatically.



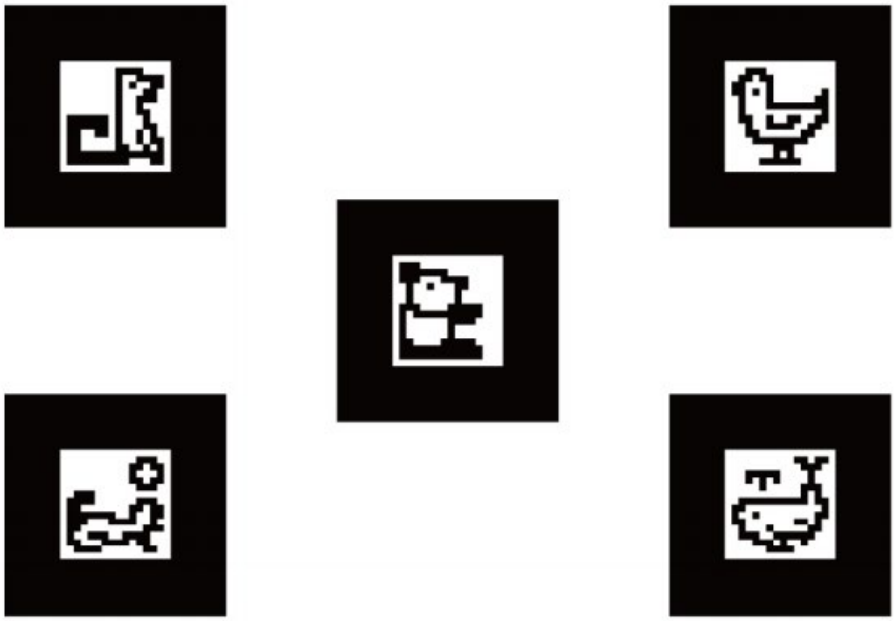
Instructions to Marker:

Marker data is stored in Assets\Plugins\Android\assets\MarkerData. We provide a set of default marker configuration files (please do not modify them). Corresponding main marker images are in Assets\NXR\Resources\Nibiru_Marker.png

Functionality	**Function name and calling method**
Start recognition	NxrViewer.Instance.GetNibiruService().StartM akerRecognize();
Stop recognition	NxrViewer.Instance.GetNibiruService().StopM akerRecognize();
Monitor Marker status	NibiruMarker.OnMarkerFoundHandler = MarkerFound; NibiruMarker.OnMarkerLostHandler = MarkerLost;

Print images in the sample directory of Marker Guide/Marker. Then arrange them

according to pictures below (the size and distance of the prints should strictly follow the illustration).



Note:

1. **New marker configuration files added by developers should be put under the directory of Assets\Plugins\Android\assets\MarkerData.**
2. **Camera is needed to enable screen recording. Therefore, gesture/marker recognition and other functions which need camera cannot be used at the same time.**

7.14 Objects dragging

Demo scene：NXR/Samples/Common/ControllerDragScene

Add NxrDragableItem.cs to objects that need to be dragged, and create new script DragTest.cs to process selected event.

Functionality	Function name and calling method
If the object is being dragged	NxrDragableItem.IsDraging
Start dragging and the object will follow	NxrDragableItem.OnBeginDrag(Transform parent);
Stop dragging and the object will be updated	NxrDragableItem.OnEndDrag(Transform parent);

In DragTest.cs, you can select an object and click the Confirm button to drag it to some place by turning your head or using the controller. Re-press the Confirm button to finish dragging.

7.15Permission request

Functionality	Function name and calling method
Permission request	NxrViewer.Instance.GetNibiruService().RequsetPermission(string[])

Permission name list:

Camera: "android.commission.CAMERA";

Write external storage: "android.permission.WRITE_EXTERNAL_STORAGE";

Read external storage: "android.permission.READ_EXTERNAL_STORAGE";

Location: "android.commission.ACCESS_COARSE_LOCATION";

Network state: "android.permission.ACCESS_NETWORK_STATE";

Settings: "android.permission.WRITE_SETTINGS";

Bluetooth: "android.permission.BLUETOOTH";

Bluetooth management: "android.permission.BLUETOOTH_ADMIN";

Internet: "android.commission.Internet";

Tasks: "android.permission.GET_TASKS";

Audio Record: "android.permission.RECORD_AUDIO";

Phone state: "android.permission.READ_PHONE_STATE";

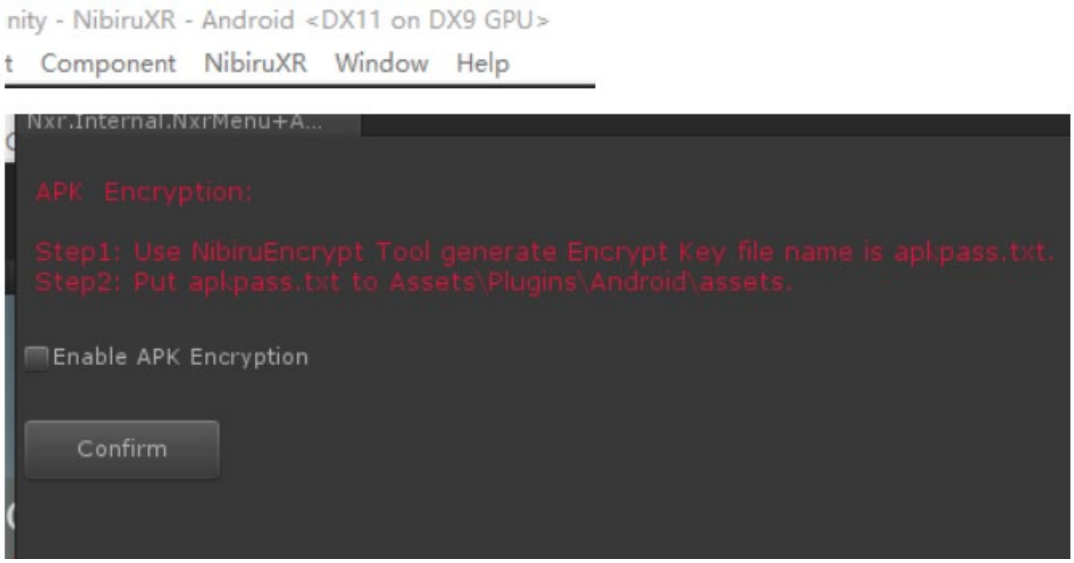
If the permission request fails, please modify Player Settings/Other Settings/Target API Level, and select API level 22 to skip it.

If the current system does not support it, the SDK will output prompt LOG:

*******[RequestPermission Warnning]******* *******

System Currently Not Support, Please Take Care !!!******* ……

7.16APK encryption



Find Nibiru XR in the top tool bar of the Editor, enter APK Encryption, check Enable APK Encryption, and click Confirm.

Please download the encryption tool on

https://dev.inibiru.eom/#/product/tools

The encrypted APK can only run on designated HMDs.

1. Generate encryption file with the encryption tool, and put the file in Assets/Plugins/Android/assets.

2. Pack the APK.

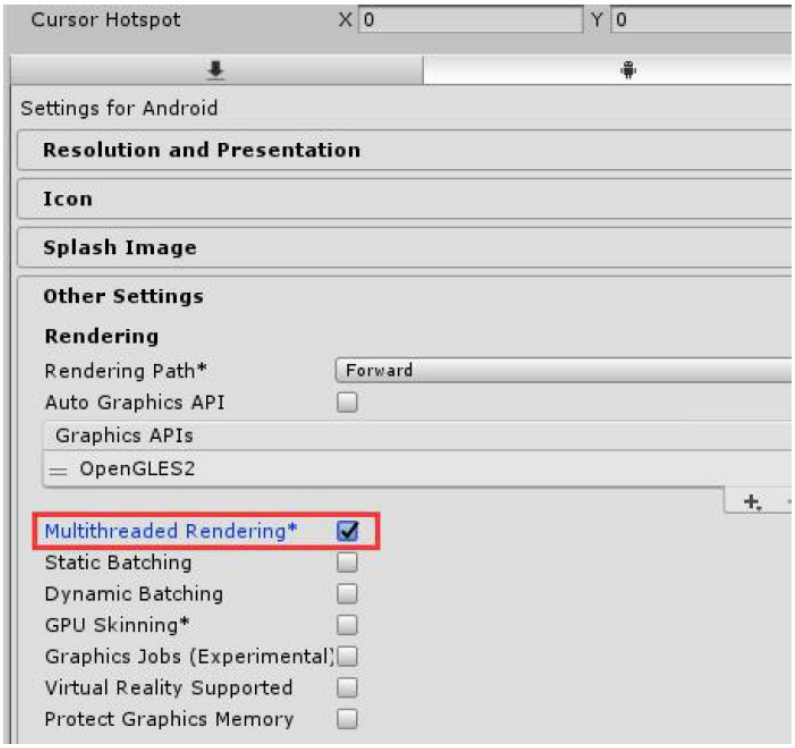
7.17Multithreaded Rendering

Starting from Nibiru XR OS SDK 2.1.0, we support multithread rendering of your applications in our Nibiru XR OS. You can check “Multithread Rendering” in “PlayerSettings”—>“Other Settings”, then pack the APK and run it.

Check the log to confirm Multithread Rendering is enabled:

graphicsMultiThreaded= (ture: enabled or false: not enabled);

If you get logs as shown below, it means the system does not support Multithread Rendering and you may get black screen and other malfunctionalities.

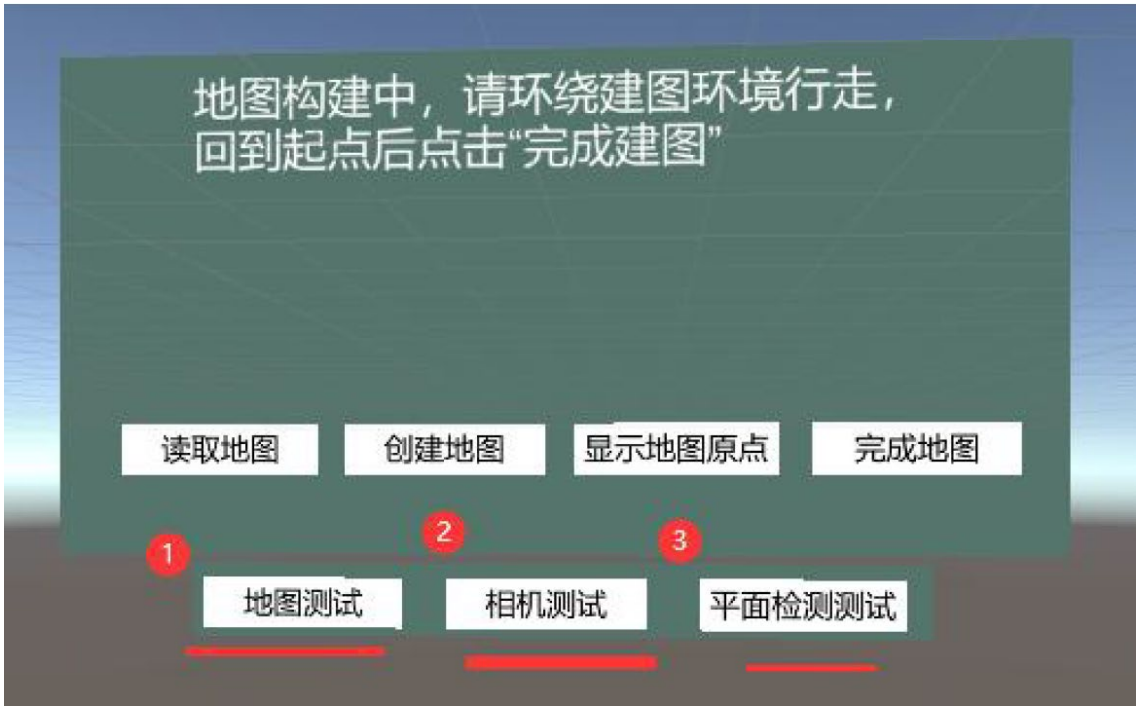


8 SLAM

8.1 SLAM Sample Scene

[Sample Scene: NXRSlam/Demo/SlamScene]

The test scene includes ①Map, ②Camera, ③Plane. You can change to different scene to test by clicking the buttons shown in the picture below.



8.2 SLAM API Function List

NxrSlam.Instance function name	**Parameters**	**Return Value**	**Description**
Init	None	bool: true – Successful initialization flase – No successful	Initialization

		initialization	
EnableNativeLog	bool enable		Whether to enable bottom log printer
MapStatusAndQuality Event	int quality: quality of map		Listen changes in map status and quality
MapPercentEvent	float percent: Percentage of coincidence		Listen coincidence of maps
StreamDataEvent	Support data callback of RGB, TOF, FISHEYE cameras		Listen camera data callback
PlaneDetectionEvent	Support data callback of planar points and normal vectors		Listen data callback of planar points
UnInit	None		Destroy
StartPlaneDetection	None	bool: true – success false –	Start detecting plane

		failure	
StopPlaneDetection	None	bool: true – success false – failure	Stop detecting plane
StartBuildMap	None	bool: true – success false – failure	Start building map
StopBuildMap	string path: Path of generated map	bool: true – success false – failure	Stop building map
StopSlam	None	None	Stop SLAM
StartSteaming	NxrSlam.RawDataType type : Camera type	None	Start camera streaming
StopStreaming	NxrSlam.RawDataType type : Camera type	None	Stop camera streaming

StartSlamWithMap	string path : Path of generated map	bool: true – success false – failure	Load map under “path”

Sample script of Demo:

NXRSlam/Demo/SlamApi.cs

Include all API call camples.

How to call the API:

Initialization —> Register callback —> Enable camera, map, plane detection —>

Handle callback data —> Stop camera, map, plane detection —> Exit and destroy.

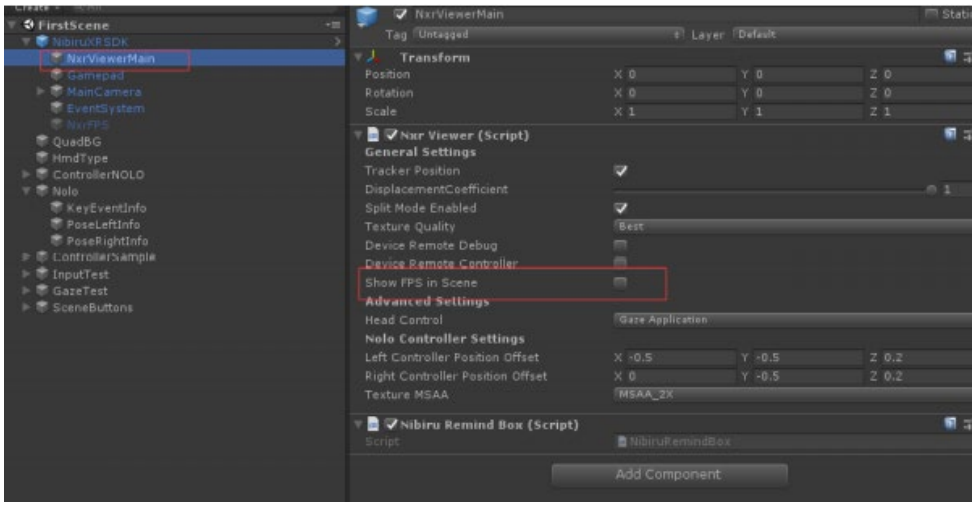
8.3 Meaning of part structure parameters

NXRStreamData	rawDataType (types of data flow of camera): RGB, TOF, FISHEYE, THERMAL
	data: camera data in RGB, TOF, THERMAL
	left: Left eye data of FISHEYE camera
	right: Right eye data of FISHEYE camera
	leftSize: Size of left eye data of FISHEYE camera

	rightSize: Size of right eye data of FISHEYE camera
	type: Data type is only available in TOF and RGB
	width: Width resolution
	height: Height resolution
	rgbCodecFormat: Only available for RGB. Right now YUV420p is used.
	timestamp: Time stamp
NXRPlaneData	points: Points contained on the plane
	normal: Normal vector of the plane
	size: Number of points
	id: Plane ID. There may be multiple planes that need to be distinguished by ID.

9 Auxiliary Tools

Display frame rate: enter NibiruXRSDK/NxrViewerMain, and check Show FPS in Scene.



10 Frequently Asked Questions

1 If 2D UI with the orthographic camera is included in the scene, how to split the screen?

We recommend putting 2D UI into 3D scene. You can see more details in the fifth entry above. It is best to keep only one main camera in the whole scene.

2 Why the split-screen is displayed in the Editor, but not in the HMD at runtime?

1) Check whether NAR tab is added to AndroidManifest.

2) Check whether the camera mounted with the script is MainCamera.

3 Why it fails to package APK?

1) Check whether the JAR package referred by Android conflicts.

2) View the Console error log in the Editor to analyze the reasons specifically.

4 Why it crashes or goes black at runtime when the packaging succeeds?

1) Open the LogCat of Eclipse, and filter out the related Log by “Tag=Unity” to check if there’s unexpected error.

2) l/Unity(4350): NullReferenceException at UnityEngine.MateriaL.ctor (UnityEngine.Shader shader) [0x00000]

When you encounter with this error, open Edit->Project Settings->Graphics, and drag UnlitTexture.shader and SolidColor.shader from the NAR/Resources directory to the Always Included Shaders list on the right.

3) Check if there is duplication of JAR under Plugins/Android.

4) Check whether Multithreaded Rendering is selected in PlayerSetting. If it’s enabled, please disable it first before packaging.

5 Frame rate is unstable and sometimes there are frame drops. Here are some suggestions for optimization:

1) RaycastTarget in Image components of UGUI will cost some efficiency. Thus,

you can choose not to select it to save efficiency. Text components have the same problem. Generally, just key pressing needs to receive response events in UI. So, most of images and texts do not need to enable RaycastTarget.

2) Use Update, LateUpdate, and FixedUpdate less to improve performance and save battery power. Use more events (not SendMessage, please use your own events or event delegations in C#)

3) If not necessary, do not use real-time shadow but use global lighting to improve runtime efficiency.

6 Are there any requirements for frame rate and scene to meet fluency?

Scene frame rate should not be less than 30 FPS. The recommended scene frame rate is above 45 FPS.

7 Suggestions for scene optimization:

1) Triangular meshes of the model should be within 50,000 for a single eye. The number of vertices should be within 50,000.

2) Minimize or eliminate the use of Unity's lights, but use Lightingmap instead.

3) Minimize or eliminate the use of particle systems.

8 Frame rate is displayed normally but screen stutters

Check related configurations in Unity Player Settings: select Never instead of Multithreaded Rendering in Blit Type.