



Qualcomm Technologies, Inc.

Qualcomm® Snapdragon™ eXtended Reality Unity Integration

User Guide

February 6, 2019

Qualcomm Snapdragon is a product of Qualcomm Technologies, Inc. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc. or its other subsidiaries.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer ("export") laws. Diversion contrary to U.S. and international law is strictly prohibited.

Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision history

Revision	Date	Description
A	May 2016	Initial release
B	Sept 2016	VR SDK 1.1 Update
C	March 2017	VR SDK 2.0 Update
D	July 2017	VR SDK 2.1 Update
E	January 2019	XR SDK 3.0 Update

Contents

- 1 Introduction 4**
 - 1.1 Purpose4
 - 1.2 Background.....4
- 2 Installation 5**
 - 2.1 System requirements5
 - 2.2 SDK contents5
- 3 Using the SDK 6**
 - 3.1 Unity integration6
 - 3.1.1 Scene setup.....6
 - 3.1.2 Player settings6
 - 3.1.3 Quality settings6
 - 3.1.4 Sample BoxWorld7
 - 3.1.5 Sample ControllerTest.....7

Tables

- Table 2-1 SDK contents5

1 Introduction

1.1 Purpose

This document provides information about installing the Qualcomm® Snapdragon™ eXtended Reality (XR) SDK Unity Integration.

1.2 Background

High performance, low latency virtual reality on Android requires access to numerous new features and software optimizations on the target platform. The Snapdragon XR SDK provides access to these features which would otherwise be unavailable to developers.

In addition, the Snapdragon XR SDK implements many of the core low latency XR rendering functionality developers require to create high quality XR content.

The core Snapdragon XR SDK features include:

- Asynchronous timewarp
 - Barrel distortion
 - Chromatic aberration correction
 - Display stabilization/reprojection
 - Single buffered rendering
 - Layering (masks/overlays)
- 3DOF/6DOF sensor fusion (DSP, 1000 Hz)
- Stereo eye tracking (DSP, 120 Hz)
- CPU/GPU power management

2 Installation

To install the Snapdragon XR Unity Integration, unzip the distribution package to any preferred location.

2.1 System requirements

A Snapdragon 835-based device running Android 7.1 loaded with the LA 2.0 (1463+) CRM, or later is required to utilize the Snapdragon XR SDK libraries.

2.2 SDK contents

The Snapdragon XR Unity Integration contains the items listed in [Table 2-1](#).

Usually only /packages and /sample/BoxWorld will be used.

Table 2-1 SDK contents

/doc	SDK documentation
/packages	Unity Integration packages for Unity 2017 and Unity 2018 onward
/samples	Samples projects that demonstrate usage
/plugin/build	Gradle and Android-NDK for building unity integration archives
/plugin/project	Unity project for creating SDK integration packages
/plugin/source	Native C/C++ headers and code for interfacing with the SDK APIs

3 Using the SDK

3.1 Unity integration

3.1.1 Scene setup

1. From a new or existing project, import the Snapdragon VR SDK package from **Assets > Import Package > Custom Package**, selecting the `/packages/svrUnityIntegration[32|64]bit.unpackage` file from SDK release. Use 32bit for Unity 2017; 64bit for Unity 2018 and later.
2. In the **Unity** dialog, leave all boxes checked and select **Import**.
3. Suggest **Close** and then **Reopen** Unity to sync references in prefabs.
4. Create an instance of a SvrCamera rig by dragging the prefab from `/Assets/SVR/Prefabs` into the scene.
5. Please try to avoid adding game objects to SvrCamera hierarchy to make SDK updates easier. Instead, use Transform SvrManager.Instance head and gaze to sync objects to head and eyes, respectively, instead.

Note that execution in the editor will display the message “No cameras rendering”. It is a known Unity issue that can be ignored.

3.1.2 Player settings

1. Select **File > Build Settings > Android** and choose **Player Settings**.
2. In the *Resolution and Presentation* section, set the *Default Orientation*.
 - a. Landscape Right for Snapdragon 835
 - b. Landscape Left for Snapdragon 845
 - c. Portrait for Snapdragon 855 plus VR Viewer
3. In the *XR Settings* section, **Multithreaded Rendering** is not recommended as it adds one frame of display latency.
4. In the *Other Settings* section, **Virtual Reality Supported** should not be set.

3.1.3 Quality settings

5. Select **Edit > Project Settings > Quality** to select the following recommended options:
 - ☐ *Anisotropic Textures* = **Per Texture**
 - ☐ *Anti Aliasing* = **Disabled**

Anti Aliasing should be enabled in the properties of the SvrCamera node instantiated in the scene. Note that HDR must be disabled on the SvrCamera for anti-aliased eye buffers to be properly created.

- *V Sync Count* = **Don't Sync**

3.1.4 Sample BoxWorld

The sample BoxWorld contains four unity scenes: boxWorld, overlayWorld, planeWorld and gridWorld. The Android BACK button or Windows Escape key will advance to the next scene and exit the application. In the unity editor, right mouse button will control the orientation, and center mouse button the position, of the head. SvrManager.Settings.TrackEyes is enabled for all but the first scene.

6. boxWorld.unity demonstrates the simple use of stereo left/right eye layers. In the SvrCamera hierarchy, only the 'Eye Left' and 'Eye Right' game objects are active - the others are disabled. Upon SvrCamera initialization, SvrEye components are allocated for the left and right eyes. Unity render textures are created and assigned to each unity camera. The eye target textures are passed to the sxr sdk for display processing and timewarp reprojection.
7. overlayWorld.unity demonstrates the use of stereo overlay layers in addition to the stereo eye layers in the previous example. In the SvrCamera hierarchy, 'Overlay Left' and 'Overlay Right' are now active too. Upon SvrCamera initialization, SvrOverlay components are allocated for the left and right overlays. The overlay layers are displayed after the eye layers. Overlay layers are not timewarp reprojected. In this sample, overlays are used to demonstrate the display of objects that are position relative to the svr head. The Reticle in the center of the screen, Hat near the top and Pointer near the bottom are all displayed smoothly without timewarp reprojection induced judder. The Reticle and Pointer sync to the eyes direction.
8. planeWorld.unity demonstrates the use of SvrEye and SvrOverlay layer components used to display textures of imageType StandardTexture in addition to RenderTexture. In this sample, the SnapdragonLogo texture is displayed as a camera-space Reticle in the center of the screen and as a world-space object in the level. The SvrEye and SvrOverlay components are Added to the SvrCamera in the scene and the ImageTexture, ImageTransform and ImageCamera objects are assigned. This sample also demonstrates the use of multiple svr objects in the eye layers. The composite order of the objects within the eye or overlay layer is controlled by LayerDepth with larger values draw after smaller values and the overlay layer(s) are drawn after the eye layer(s). The Reticle is synced to the eyes direction.
9. gridWorld.unity demonstrates the use of foveated rendering with eye render targets greater than 1024 x 1024. The SvrEye layer components are added with ResolutionScaleFactor greater than one assigned. Foveated rendering is enabled with SvrManager.Settings Foveated Gain values greater than zero. Foveated Gain values greater than 1 will cause a reduction in bin resolution toward the extends while Area preserves more of the focal region. In this example, Foveated Gain (4, 4) and Area (2) is specified creating a subtle foveation that saves approximately 20% of GPU draw. The foveated area Focal Point is synced to the eyes position.

3.1.5 Sample ControllerTest

Setup

- Install the ControllerTest apk
- Install the XimmerseControllerService apk
- Install the BTConfig[1.0.0-Flip] apk from https://github.com/Ximmerse/SDK_Flip/tree/master/Tools

Pairing

1. Launch the XimmerseControllerService app
2. Press and hold the Back button then press and hold the Start button on the controller
3. Select Bind in the XimmerseControllerService app
4. Release both buttons on the controller

ControllerTest - Demonstrates the use and integration of the xImmerse 3DoF controller.

XimmerseControllerService - Service that talks to Ximmerse Controller Daemon and provides the values to the SXR SDK in a format that is required by the SXR SDK.

BTConfig[1.0.0-Flip] – Ximmerse daemon that talks to the bluetooth controller.