Unreal

Version 4.12

Copyrights and Trademarks

Contents

Unreal Introduction

Unreal provides built-in support for Oculus Rift and Gear VR development on Windows.

Unreal apps run on the Oculus platform automatically apply stereoscopic rendering to the main camera as well as positional and orientation tracking for the Rift, or orientation tracking for Gear VR.

For Epic's documentation about Oculus development, go to *https://docs.unrealengine.com/latest/INT/Platforms/VR/*.

Oculus SDK

Most developers use the Epic Games public version of UE4 and our SDK, available to EpicGames/UnrealEngine subscribers at *https://github.com/Oculus-VR/UnrealEngine*.

We also maintain our own version of the SDK, which is usually one version ahead of the public version of UE4. If you need the latest VR features that have not yet shipped in the public Unreal engine, you may download the integration package from our SDK Downloads UE4 sub-forum. If you don't have access to it and you hold a UE4 license, please refer to the UE4-Oculus-Install.txt file inside the package for installation instructions.

Getting Started

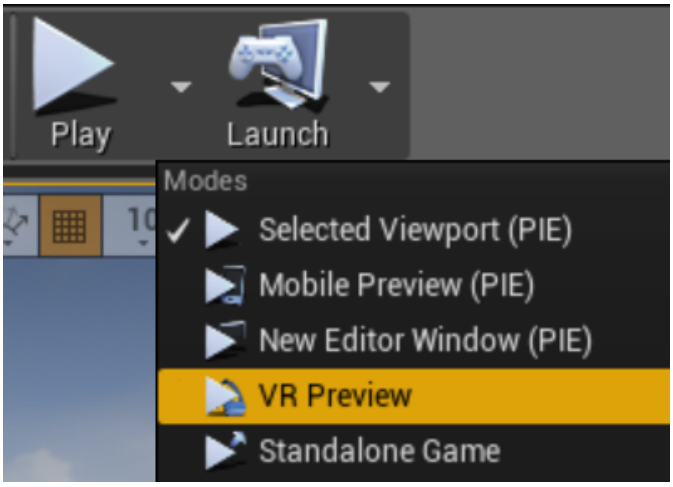
Overview

The easiest way to get started with Oculus development in Unreal is to experiment with the HMD support provided by default as a Player Start. If Unreal detects an Oculus Rift runtime (e.g, the Oculus app is installed), it will automatically provide orientation and positional tracking and stereoscopic rendering for any game.

Standalone PC executables will automatically display in Oculus Rift when run in full-screen mode if a Rift is detected on your system. You may build a project normally and preview it in the Engine by selecting the VR Preview mode in the Play button pulldown options.

Note: In order to play an in-development application, you will need to enable running applications from unknown sources in the Oculus app settings, available through the gear icon in the upper-right. Select *Settings* > *General* and toggle *Unknown Sources* on to allow.

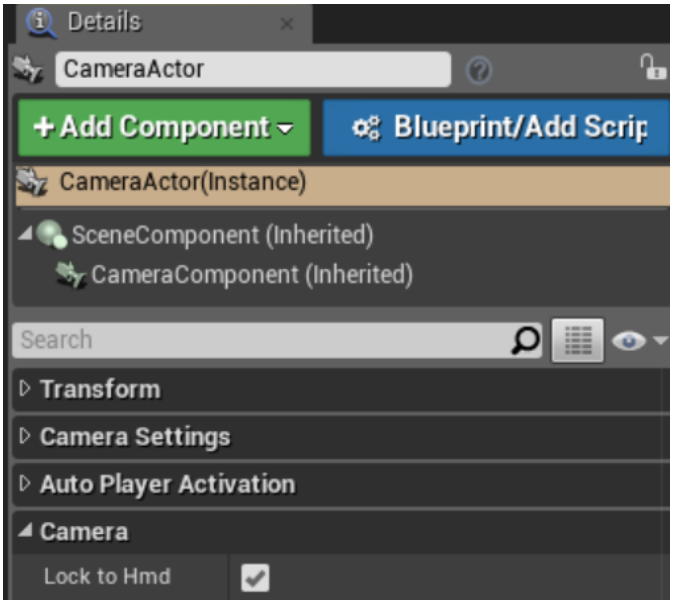
To preview a level in the Rift while editing with the engine, select the dropdown arrow near the Play button and select VR Preview.



To play a scene in the Rift with a standalone PC game, simply maximize the game window and it will be mirrored to the Rift display.

Camera VR

This is a good way to get a quick sense of VR development with minimal overhead, but for actual development, we generally recommend adding a Camera actor to your map and enabling Oculus support by selecting the camera in Viewport and checking the *Lock to Hmd* checkbox in the *Details* tab (available in UE4 11).



Placing a camera in the scene allows you to control the orientation of the camera view when the game loads, so that you can control the exact perspective that will be visible to the user. This is not possible with the Play Start described above.

An additional benefit to using the Camera actor is that you can attach meshes and they will update their position following the HMD view with very little latency. This is generally the best way to add head-locked elements such as cockpit details. Note that we generally discourage head-locked UI elements, but it can be an attractive feature when used carefully.

Platform Features

Unreal applications may use our Platform SDK to add features related to security, community (e.g., matchmaking), revenue (e.g., in-app purchases), and engagement (e.g., leaderboards). For more information, see our *Platform SDK documentation*.

Entitlement checking may be used to protect apps from unauthorized distribution. For more information and instructions, see *Entitlements: App Authorizations* in our Platform guide. Applications should always have entitlement checks enabled for submissions to the Oculus store.

Unreal Mobile Development

Mobile Development Setup

Begin preparing for Gear VR development by following the instructions in our Mobile *Device Setup* guide, and install the Java Development Kit (JDK), Native Development Kit, and Android SDK before beginning development. Most Unreal developers do not need to install Android Studio.

Then do any remaining necessary setup for Android development for Unreal, following Epic's *Android Quick Start*.

Be sure to review all of the relevant performance and design documentation for Oculus mobile development. Mobile apps are subject to computational limitations which should be taken into consideration from the ground up.

Application Signing

Mobile applications require two different signatures at different stages of development. Be sure to read the *Application Signing* section of the Mobile SDK documentation for more information.

Note that your Gear VR application will not run without an Oculus Signature File (osig).

To add your osig for Unreal development:

1. Follow the instructions in *Application Signing* to download your osig.
2. Navigate to the following directory: \Engine\Build\Android\Java\.
3. Create the new directory *assets* inside the directory. Be sure the name is uncapitalized.
4. Copy your osig to this directory.

Workflow

We recommend setting your phone to *Developer Mode*, so you can run VR apps on your phone without inserting it into the Gear VR headset. This allows you to quickly review your progress on the device without much overhead.

To run your app during development, we generally recommend connecting your phone via USB, selecting the *Launch* pulldown menu, and then selecting your phone under the listed devices. Particularly if you are modifying maps and shaders but not changing the code, this is often much faster than building an APK.

It is possible to preview a Gear VR application in the Oculus Rift during development. However, it is not generally useful to do so, because Rift applications are subject to substantially different performance requirements (see "Performance Targets" in *Debugging and Performance Analysis in Unreal*). You may find it easiest to use the 2D preview in Unreal, and then either build an APK or use *Launch on Device* when you need to view the app in VR.

Note: Mobile Content Scale Factor is not currently supported for Oculus development.

Unreal Console Commands

This document reviews useful console commands available for Unreal development.

Table 1: Configuration Commands

Command	Description
stereo onloff	Enables/Disables stereo mode.
stereo e=0.064	Eye distance (m).
stereo w2m=100	Overrides default worldunits-to-meters scale.
stereo ncp=10 fcp=10000	Overrides near clipping and/or far clipping planes for stereo rendering (in cm).
stereo cs=1 ps=1	Overrides camera and position scale.
stereo show	Shows current ipd and head model offset.
stereo reset	Resets stereo settings.
hmd enableldisable	Enables/Disables HMD.
hmd pd [0..3.0]	Sets pixel density in the center (default is 1.0).
hmd sp [30..300]	Overrides screenpercentage for stereo mode. (Deprecated, use 'hmd pd xxx' instead).
hmd hqdistortion onloff	Enables/Disables high-quality distortion.
hmd mirror onloff	Enables/Disables mirroring to the desktop window.
hmd mirror mode [0..4]	Sets mirror mode: 0=Distorted, 1=Undistorted, 2=SingleEye, 3=SingleEye Letterboxed, 4=SingleEye Cropped
hmdpos onloffltoggle	Enables/Disables/Toggles positional tracking.
hmdpos enforce onloff	Enables/Disables head tracking even if not in stereo (for testing purposes).
hmdpos reset {yaw}	Resets position and rotation, applies yaw (in degrees) if provided.
hmdpos resetrot {yaw}	Resets rotation only, applies yaw (in degrees) if provided.
hmdpos resetpos	Resets position only.
hmdpos show	Outputs status of positional tracking to log.

Command	Description
hmdpos floorleye	Selects tracking origin.

Table 2: Misc Commands

Command	Description
hmd stats	Shows HMD-related stats.
hmd grid	Toggles lens-centered grid.
hmd updateongt onloff	Enables/Disables updating HMD pose on game thread. On by default.
hmd updateonrt onloff	Enables/Disables updating HMD pose on render thread, for lower latency. On by default.
hmd cubemap [gearvr] [xoff=N] [yoff=N] [zoff=N] [yaw=N]	Generates a cube map image of your application. May be used for VR app previews in the Oculus Store. Cube map PNGs will be saved in the directory GameDir/Saved/Cubemaps. gearvr: If specified, cube map size will be 6x1024x1024, otherwise it will be 6*2048x2048. xoff, yoff, zoff: Offset from the current player's location. yaw: override yaw rotation (degrees).
hmd setint PerfHUDMode [0..4]	Selects performance HUD mode, set to 0 to disable.
hmd setint DebugHudStereoMode [0..3]	Selects debug HUD stereo mode, set to 0 to disable.
hmdversion	Prints Oculus SDK version used and Oculus Plugin info.

For more information, see "Useful VR Console Commands" in Unreal's *VR Cheat Sheet*.

Debugging and Performance Analysis in Unreal

VR application debugging is a matter of getting insight into how the application is structured and executed, gathering data to evaluate actual performance, evaluating it against expectation, then methodically isolating and eliminating problems.

When analyzing or debugging, it is crucial to proceed in a controlled way so that you know specifically what change results in a different outcome. Focus on bottlenecks first. Only compare apples to apples, and change one thing at a time (e.g., resolution, hardware, quality, configuration).

Always be sure to profile, as systems are full of surprises. We recommend starting with simple code, and optimizing as you go - don't try to optimize too early.

Performance Targets

Before debugging performance problems, establish clear targets to use as a baseline for calibrating your performance.

These targets can give you a sense of where to aim, and what to look at if you're not making frame rate or are having performance problems.

Below you will find some general guidelines for establishing your baselines, given as approximate ranges unless otherwise noted.

Mobile

- 60 FPS (required by Oculus Store)
- 50-100 draw calls per frame
- 50,000-100,000 triangles or vertices per frame

PC

- 90 FPS (required by Oculus Store)
- 500-1,000 draw calls per frame
- 1-2 million triangles or vertices per frame

For more information, see:

- *PC SDK Developer Guide*
- *Mobile VR Application Development*

Rift Performance HUD

The Oculus Performance Heads-Up Display (HUD) is an important, easy-to-use tool for viewing timings for render, latency, and performance headroom in real-time as you run an application in the Oculus Rift. The HUD is easily accessible through the Oculus Debug Tool provided with the PC SDK. You may activate it in the Viewport by pressing the ~ key.

For more details, see the *Performance Heads-Up Display* and *Oculus Debug Tool* sections of the Oculus Rift Developers Guide.

Mobile Debug Console

If your phone is set to Developer Mode, you may bring up a debug console for VR apps by pressing the screen with four fingers simultaneously while the application is running.

Enter `stat unit` in the console for information about your application framerate and GPU performance.

Additional Third-Party Tools

ETW + GPUView

Event Tracing for Windows (ETW) is a trace utility provided by Windows for performance analysis. *GPUView* view provides a window into both GPU and CPU performance with DirectX applications. It is precise, has low overhead, and covers the whole Windows system. Custom event manifests.

ETW profiles the whole system, not just the GPU. For a sample debug workflow using ETW to investigate queueing and system-level contention, see Example Workflow: PC below.

Windows 10 replaces ETW with *Tracelogging*.

Systrace

Reports complete Android system utilization. Available here: *http://developer.android.com/tools/help/systrace.html*

NVIDIA NSight

NSight is a CPU/GPU debug tool for NVIDIA users, available in a *Visual Studio version* and an *Eclipse version*.

Mac OpenGL Monitor

An OpenGL debugging and optimizing tool for OS X. Available here: *https://developer.apple.com/library/mac/technotes/tn2178/_index.html#//apple_ref/doc/uid/DTS40007990*

APITrace

https://apitrace.github.io/

Other Resources

For detailed information about Oculus development, go to:

- Unreal: Virtual Reality Development: *https://docs.unrealengine.com/latest/INT/Platforms/VRZ/*
- Unreal: Oculus Rift wiki: *https://wiki.unrealengine.com/Oculus_Rift*
- Oculus Forums/Unreal: *https://forums.oculus.com/developer/categories/unreal*
- Unreal Forums/VR Development: *https://forums.unrealengine.com/forumdisplay.php?27-VR-Development*

Contact

Visit our developer support forums at *https://developer.oculus.com*.

Our Support Center can be accessed at *https://support.oculus.com*.