



Virtual Reality

UE5 Best Practices

CS 415: Game Development

Professor Eric Shaffer



Technical Resources for VR Development in Unreal

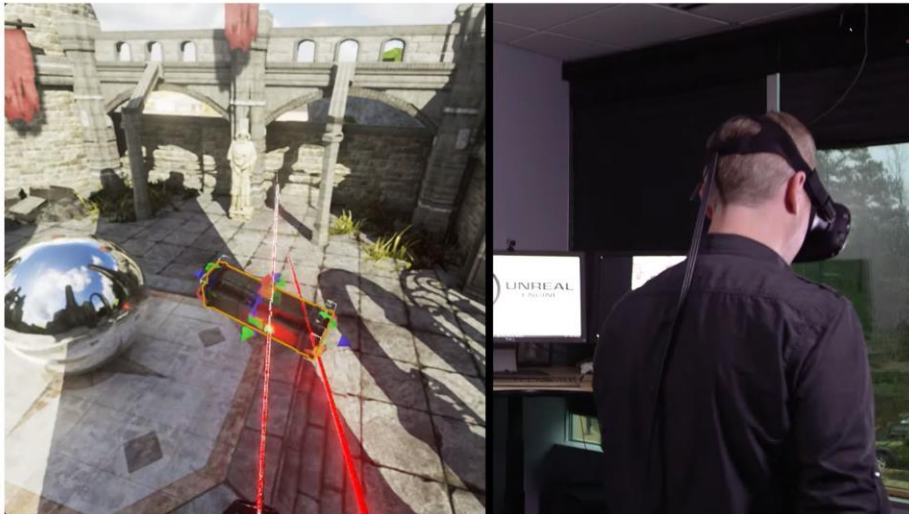
Unreal Engine VR Mode

Build worlds in a virtual reality environment using the full capabilities of the editor toolset, with interactions designed specifically for VR.

Intermediate

Use VR Mode to design and build worlds in a virtual reality environment using the full capabilities of the Unreal Editor toolset combined with interaction models designed specifically for Virtual Reality (VR) world building.

Working directly in VR provides the proper sense of scale necessary to create realistic, believable worlds, while the use of motion controllers means you can build environments with natural motions and interactions.



Unreal Editor can run in VR...

Can speed up design of VR worlds

- <https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/VRMode/>

Technical Resources for VR Development in UE4

Unreal Engine 4.27 Documentation

Sharing and Releasing Projects > XR Development > Virtual Reality Development

Search Documentation...

Filter pages...

+ Setting Up Your Production Pipeline

+ Testing and Optimizing Your Conte...

- Sharing and Releasing Projects

AutoSDK Reference

Building Unreal Engine as a Lib...

+ Containers

+ HTML5 Game Development

+ Linux Game Development

+ Mobile Game Development

+ Patching and DLC

+ Pixel Streaming

- XR Development

+ OpenXR

- Virtual Reality Development

+ VR Platforms

+ VR How-To's

Virtual Reality Best Pract...

+ VR Performance and Pro...

VR Cheat Sheet

+ Augmented Reality Develop...

Setting Device Profiles


Low Latency Frame Syncing

Virtual Reality Development

Virtual reality (VR) refers to an interactive experience where the user's real-world environment is replaced by a virtual environment through a wearable device.


Virtual reality (VR) refers to an interactive experience where the user's real-world environment is replaced by a virtual environment through a wearable device. The Unreal Engine VR framework provides a rich, unified framework for building virtual reality apps using the Unreal Engine.

Platforms




Developing for Oculus VR

Information on using UE4 to develop for Oculus VR.




Developing for SteamVR

Information on using UE4 to develop for SteamVR.




Windows Mixed Reality

Information on using UE4 to develop for Windows Mixed Reality platforms.



Developing for Samsung Gear VR

Information on using UE4 to develop for Samsung Gear VR.



Developing for Google VR

Information on using UE4 to develop for Google VR.

ON THIS PAGE

Platforms

Guides

References

<https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/VR/>

I ILLINOIS

Technical Resources for VR Development in UE5

The screenshot shows the Unreal Engine 5.0 Documentation website. The top navigation bar includes 'Unreal Engine 5.0 Documentation' and a search bar. The left sidebar lists various development topics, with 'XR Development' selected. The main content area is titled 'XR Development' and features a dark background image of a city at night. Below the title, it states 'Develop for augmented reality, virtual reality, and mixed reality devices in Unreal Engine.' A list of experiences is provided: Augmented reality, Virtual reality, and Mixed reality. A paragraph explains that Unreal Engine supports both developing for XR platforms and using XR devices. The section 'Developing for Head-Mounted XR Experiences with OpenXR' is highlighted, with a sub-section 'Developing for Head-Mounted Experiences with OpenXR' and a small image of a person wearing a VR headset. A right sidebar titled 'ON THIS PAGE' lists links to various topics related to XR development.

Unreal Engine 5.0 Documentation > Sharing and Releasing Projects > XR Development

Search Documentation...

Filter pages...

- + Programming and Scripting
- + Making Interactive Experiences
- + Animating Characters and Objects
- + Working with Audio
- + Working with Media
- + Setting Up Your Production Pipeline
- + Testing and Optimizing Your Conte...
- Sharing and Releasing Projects
 - + Packaging and Cooking Games
 - + General Platform Support
 - + General Mobile Development
 - + iOS, iPadOS, and tvOS
 - + Android
 - XR Development
 - + Developing for Head-Mount...
 - + Developing for Handheld Au...
 - + Getting Started with XR Dev...
 - + Making Interactive XR Exper...
 - + Creating UI for XR Experien...
 - + Sharing XR Experiences
 - + XR Performance and Profiling
 - + Supported XR Devices

XR Development

Develop for augmented reality, virtual reality, and mixed reality devices in Unreal Engine.

XR refers to the collection of the following experiences:


- **Augmented reality:** Where sensory information overlays the user's view of the real world through a handheld or wearable device.
- **Virtual reality:** Where a virtual environment replaces the user's view through a wearable device.
- **Mixed reality:** Where an experience is a blend between augmented reality and virtual reality.

Unreal Engine supports both developing for XR platforms and using XR devices in your content creation pipeline. The sections below contain links to documentation on how you can work with XR devices in your projects.

Developing for Head-Mounted XR Experiences with OpenXR

OpenXR is a royalty-free, open standard that provides high-performance access to XR platforms and devices. With OpenXR, you can create an immersive experience in Unreal Engine that can run on any system that supports the OpenXR APIs. Currently, OpenXR in Unreal Engine only supports head-mounted devices.

This section describes how OpenXR works in Unreal Engine.



Developing for Head-Mounted Experiences with OpenXR

Develop on head-mounted AR and VR devices with OpenXR in Unreal Engine.

ON THIS PAGE

- Developing for Head-Mounted XR Experiences with OpenXR
- Developing for Handheld Augmented Reality Experiences
- Getting Started with XR
- Making Interactive XR Experiences
- Creating UI for XR Experiences
- Shared Experiences in XR
- Supported XR Platforms in Unreal Engine
- Performance and Profiling with XR
- Using XR for Content Creation

<https://docs.unrealengine.com/5.0/en-US/developing-for-xr-experiences-in-unreal-engine/>

VR Development Best Practices

Substantially different platform from mobile or console/PC

- Different performance demands
- Different user interface principles needed

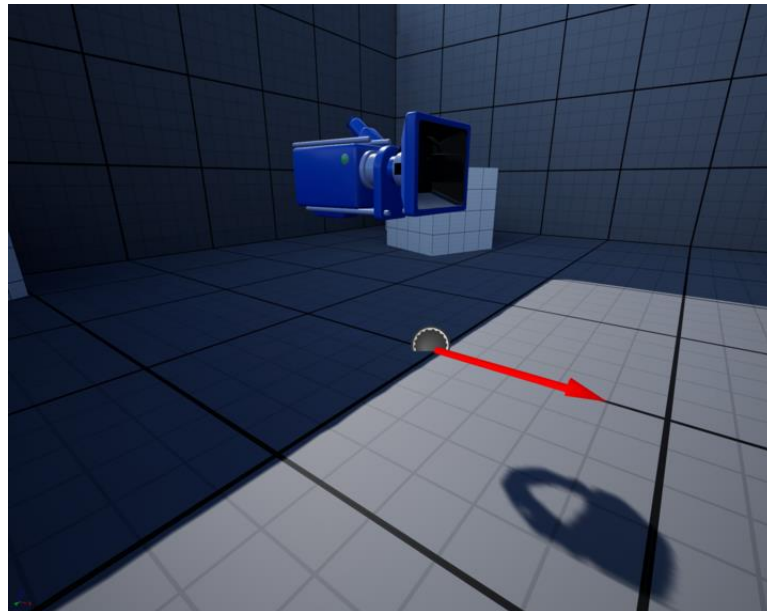
The screenshot shows the Unreal Engine 4.27 documentation website. The top navigation bar includes links for PRODUCTS, SOLUTIONS, NEWS & EVENTS, LEARN, COMMUNITY, SUPPORT, and MARKETPLACE. The main content area is titled "Virtual Reality Best Practices" and includes a "Beginner" tag. A sidebar on the left lists various development topics, with "Virtual Reality Best Practices" highlighted. The main text explains that virtual reality (VR) is an immersive new medium and provides a "VR Project Settings" section with a "NOTE" stating that VR projects can be Blueprint or C++. Below this, a "Project Settings" panel is shown with options for "Choose whether to create a Blueprint or C++ project" and "Choose the closest expanded target platform". A right-hand sidebar titled "ON THIS PAGE" lists various settings and considerations, including VR Project Settings, VR .ini Settings, VR Frame Rate Optimization, VR World Scale, VR and Simulation Sickness, VR Camera Setup, VR Character Settings, Character Height and Width, Movement Speed, Camera Location, VR Content Considerations, Known Limitations, and Normal Mapping Issues.

<https://docs.unrealengine.com/4.27/en-US/SharingAndReleasing/XRDevelopment/VR/VRBestPractices/>

VR Camera Setup

Seated experience:

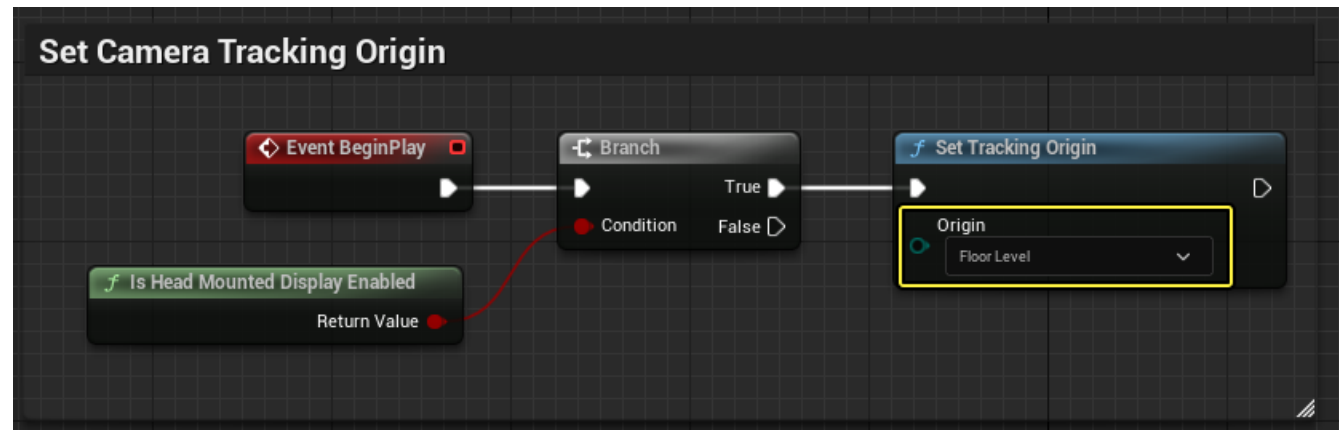
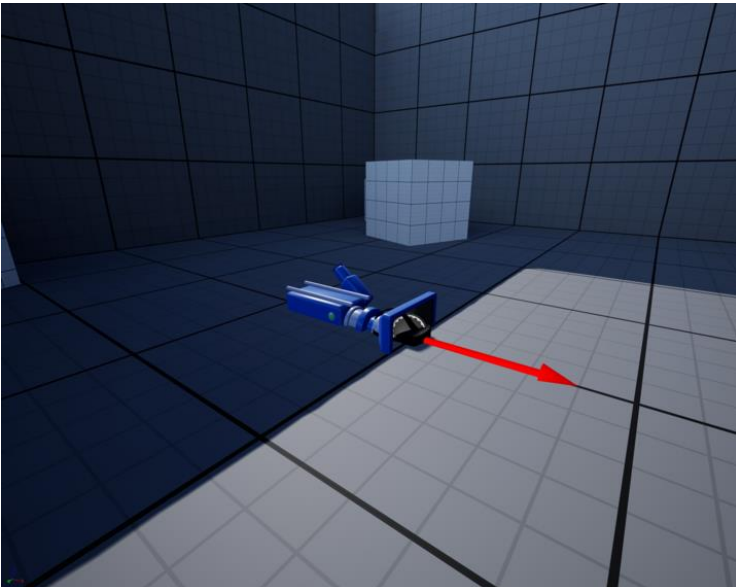
- You need to raise the camera origin to the desired player height for the project.
- And call the Set Tracking Origin function, with Origin set to "Eye Level".



VR Camera Setup

Standing experience:

- Set the camera origin to **0**, relative to the pawn's root which is typically on the ground
- Attach a camera component to a scene component at the base of the pawn, at the ground level
- And call Set Tracking Origin with the Origin parameter set to Floor Level.



Avoiding Simulation Sickness

Keep users in control of the camera:

Cinematic cameras, or anything that takes control of camera movements away from the player, contribute to user discomfort in immersive experiences.

Camera effects, such as head bobbing and camera shaking, should be avoided because they can lead to user discomfort if the user is not controlling them.

Avoiding Simulation Sickness

Field of View (FOV) must match the device:

The FOV value is set through the device's SDK and internal configuration, and matches the physical geometry of the headset and lenses.

Because of this, the FOV should not be modified in UE, and should not be modifiable by the user.

If the FOV value is changed, the world can appear to warp when you turn your head, leading to discomfort.

Avoiding Simulation Sickness

Use lights and colors that are more dim, and avoid smearing:

You might need to use dimmer lights and colors than you normally would.

Strong and vibrant lighting in VR can cause simulation sickness to occur more quickly.

Using cooler shades and dimmer lights can help prevent user discomfort.

This also helps with preventing smearing between bright and dark areas on the display.

Avoiding Simulation Sickness

Movement speed should not change:

Users should start at full speed instead of gradually accelerating to full speed

Avoid post process effects that greatly affect what the user sees:

Avoid effects like Depth of Field and Motion Blur to prevent user discomfort

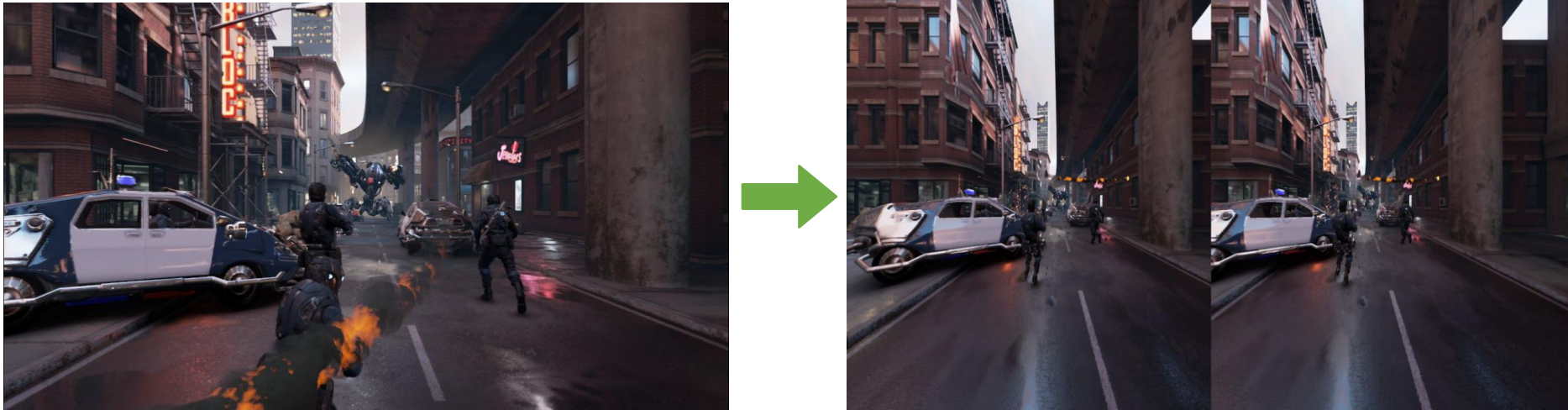
Consider things like:

Character height, width, speed, and camera location need to be modified for VR characters.

Frame Rate is Hugely Important

Need low latency, high framerate to respond to head movement...so people don't get sick

The frame is rendered twice, once per eye



Maintain a very simplistic approach to making your content.

- o Minimize complex shaders as best possible.
- o Add detail to the mesh within reason in lieu of relying of complex shaders for surface details.

Lighting

You should always use Static lighting and lightmaps when making a VR project

- The cheapest option to render

If you need to use dynamic lighting

- Make sure to limit the number of dynamic lights to as few as possible
- Make sure that they never touch one another

Target Framerrates

HMD Device	Target Frame Rate
Oculus Rift S	90
Oculus Quest 1	72
Oculus Quest 2	Situational: 72/80/90/120
HTC Vive	90
HTC Vive Pro	90
Valve Index	90 minimum, up to 144
HP Reverb	90
Windows Mixed Reality VR	90
PSVR	Situational: 60/120, 90/90, and 120/120
HoloLens 2	60

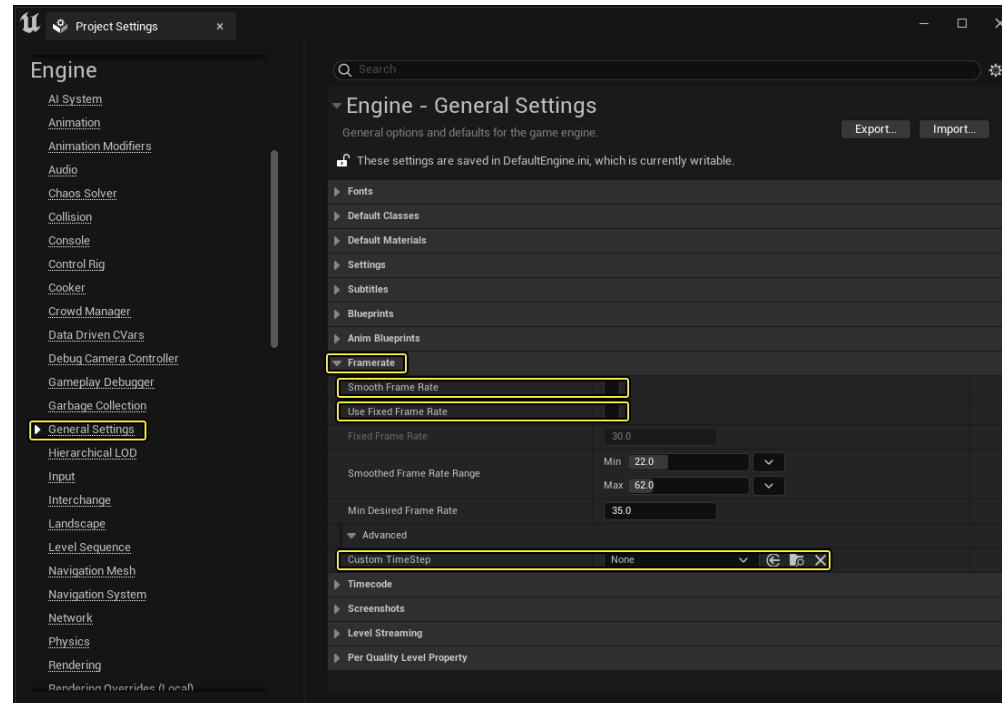
Framerate Optimization

Most VR applications implement their own procedures to control the VR frame rate.

Because of this, you should disable several general UE4 project settings that can interfere with VR applications.

Follow these steps to disable general framerate settings in UE:

1. From the editor's main menu, choose **Edit > Project Settings** to open the Project Settings window.
2. In the Project Settings window, choose **General Settings** in the Engine section.
3. In the Framerate section:
 1. Disable Smooth Frame Rate.
 2. Disable Use Fixed Frame Rate.
 3. Set Custom TimeStep to None.



Scale Your World Carefully

Getting the scale of your world correct is important things for a good user experience possible on VR platforms.

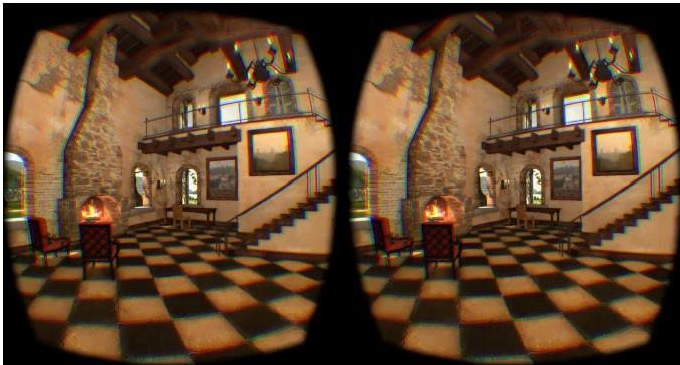
Having the wrong scale can lead to all kind of sensory issues for users and could even result in Simulation Sickness.

- Make sure camera is at the correct height for an average person
- Make sure distance between pupils matches real-world distance

Objects are most easily viewed in VR when they are in a range of 0.75 to 3.5 meters from the player's camera.

Inside of UE4, 1 Unreal Unit (UU) is equal to 1 Centimeter (CM).

Objects inside Unreal are best viewed when they are 75 UU to 350 UU away when using VR.



In the Tuscany demo from Oculus VR, there are not enough familiar objects to precisely resolve depth and size. Have you ever been to a villa like this? Are the floor tiles a familiar size? Is the desk too low?

VR and Transparency

In 3D graphics, rendering transparency is extremely costly

- Transparency will generally have to be re-evaluated per frame
- Possibly too costly for VR which needs to be very performant

You can use the **DitherTemporalAA** Material Function.



This Material Function will allow a Material to look like it is using transparency (sort of)

Fake Everything You Can

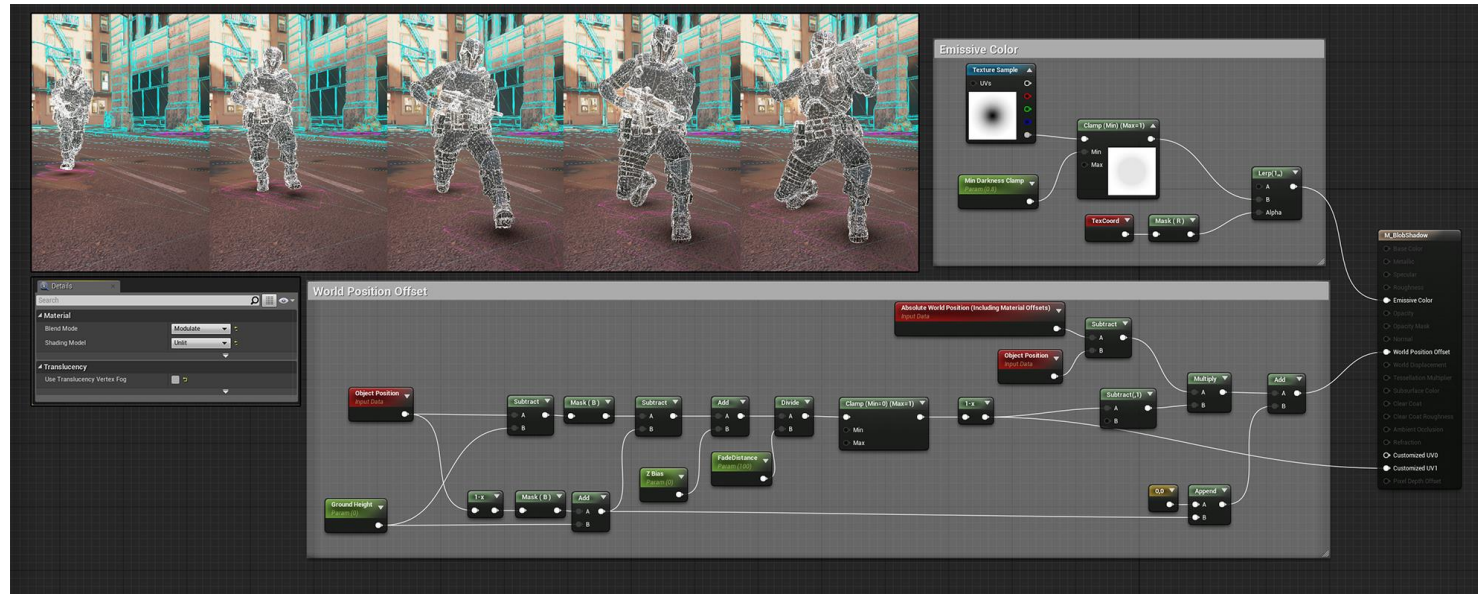
Find clever ways to recreate expensive rendering options, like dynamic shadows or lighting.

In [Showdown](#), having the characters cast dynamic shadows proved to be too expensive per-frame

However, this made the characters look like they were floating while moving

To fix this, fake blob shadows were introduced

They dynamically adjust their position and intensity based on how close the character was to an object



More Tips...

LODs and aggressive culling are a must

Use real geometry instead of:

- Normal maps (e.g. surface detail)
- Sprites (e.g. particle effects)
- Billboards (e.g. foliage)
- Flipbook movies (e.g. explosions)

Anti-aliasing:

- Use None, FXAA or hardware MSAA

Minimize number of drawcalls

- VR doubles the drawcall count

Minimize post-process effects

- High frame rate is critical

Known Limitations: Screen Space Effects

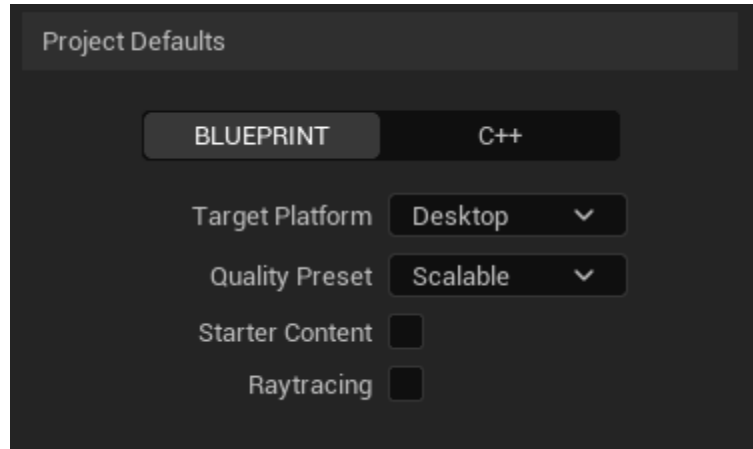
Screen Space Reflections(SSR): While SSR will still work in VR, the reflections they produce could have issues matching up to what it is reflecting in the world. Instead of using SSR, use Reflection Probes as they are much cheaper and suffer less from reflection-alignment issues.

Screen Space Global Illumination: Screen space techniques can generate differences between the displays for the two eyes in the HMD. These differences can cause user discomfort. See [Lighting Types](#) for recommended lighting types in VR for replacements.

Known Limitations: Ray Tracing

Ray Tracing:

VR apps using ray tracing currently aren't able to maintain the resolution and frame rate needed for a comfortable VR experience.

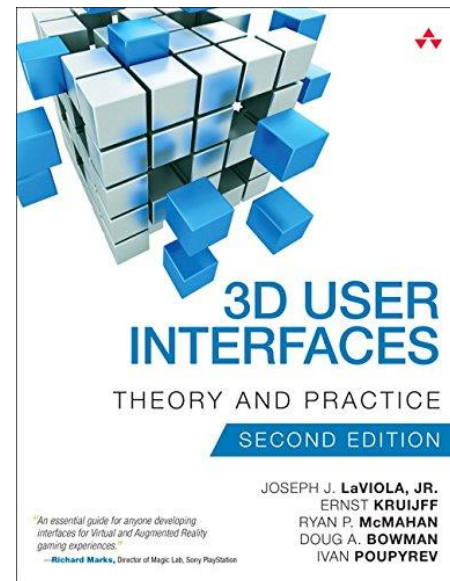


Known Limitations: 2D User Interfaces

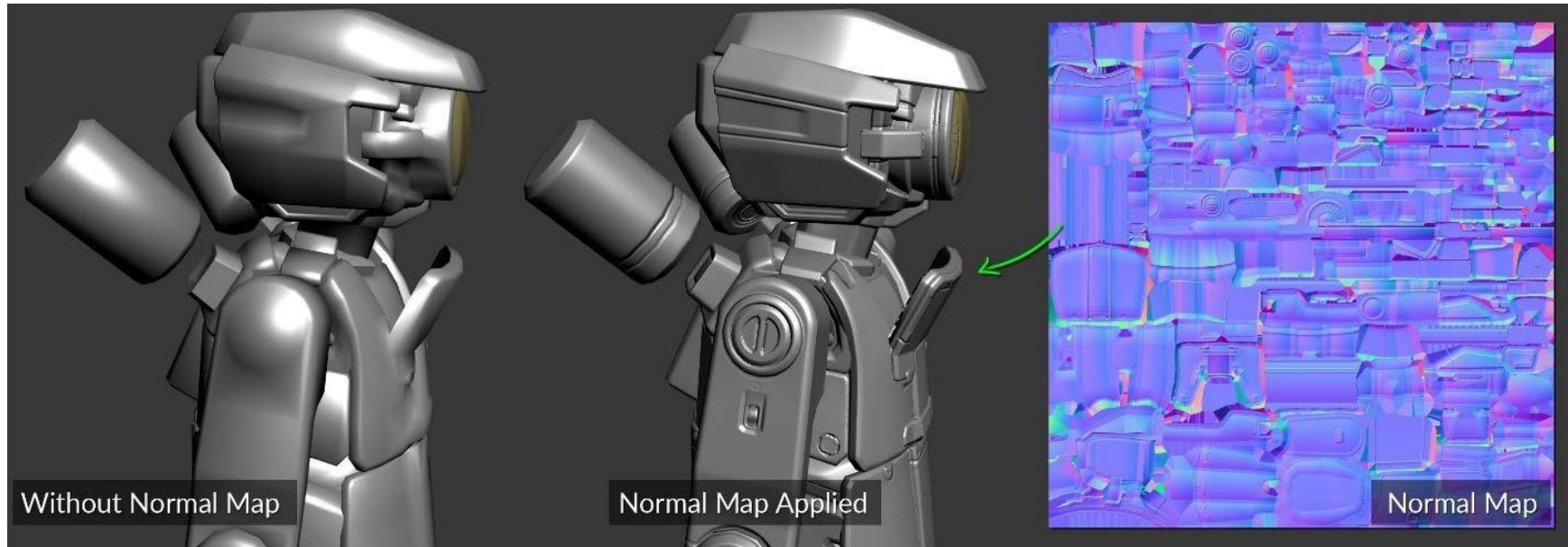
- **User Interface:**

2D UI is not supported in stereo rendering since it does not work well when viewed in stereo. Instead of 2D UI, use a widget component placed in the 3D world. See [Creating 3D Widget Interaction](#) for more on how to create these widgets.

There's a lot more to know about user interface design for VR...use teleportation...use audio instead of text...etc.



Normal Map Issues



When viewing Normal maps on objects in VR, they do not have the same visual impact as on desktop/console.

This is because normal mapping does not account for a binocular display or motion parallax.

Normal maps will come out looking flat when viewed with a VR device.

Use Parallax Mapping



Parallax mapping takes Normal mapping to the next level by accounting for depth cues

Normal mapping does not.

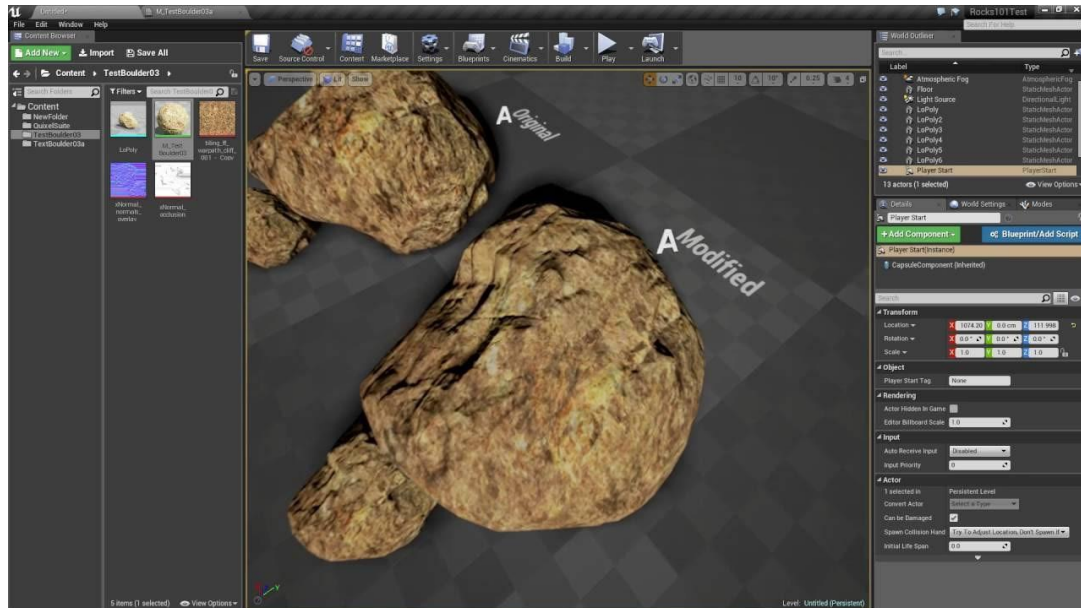
A Parallax mapping shader can better display depth information,
making objects appear to have more detail than they do.

This is because no matter what angle you look at,
a Parallax map will always correct itself to show you the correct depth information from that view point.

A good use of a Parallax map would be for cobblestone pathways and fine detail on surfaces.

Use Tessellation Shader Displacement

- Tessellation Shader Displacement will displace 3D Geometry in real time
- Adding details that are not modeled into the object.
- Tessellation shaders create more vertices and displace them in 3D Space.



More Performance Considerations

For the VR experience to feel smooth, your game needs to run at 75 hz to 90 hz

To see the current framerate type in “stat fps” or “stat unit” in your console when running the game.

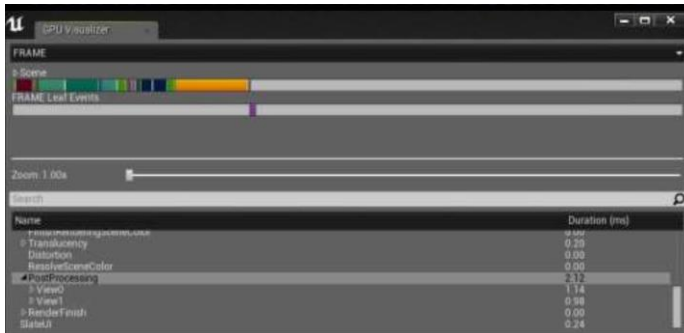
For GPU Profiling:

To capture a single frame with GPU timings press Ctrl+Shift+, or type in “profilegpu” in the console.

This command dumps accurate timings of the GPU.

You will find that certain processes are a heavy burden on the framerate when using VR

Ambient Occlusion is one common example



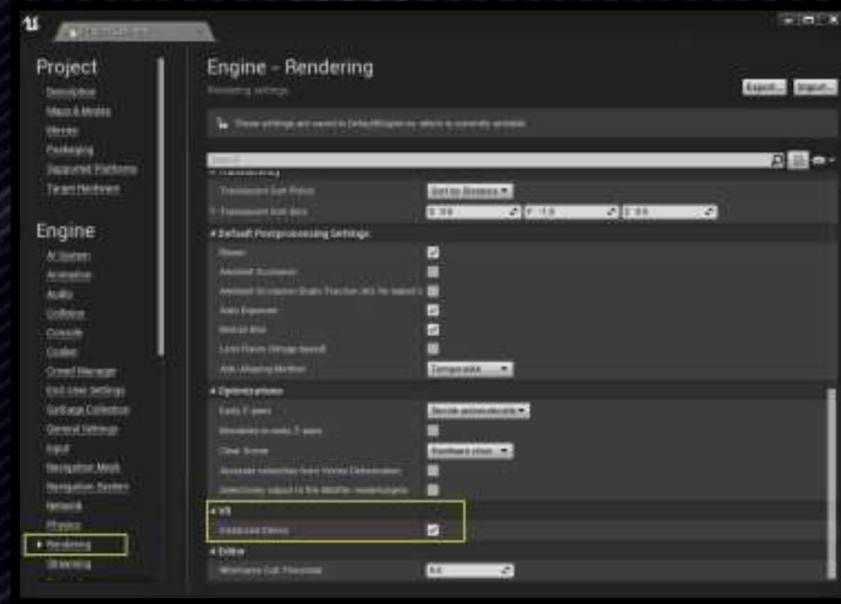
<https://docs.unrealengine.com/4.27/en-US/TestingAndOptimization/PerformanceAndProfiling/GPU/>

Consider Instanced Stereo

The latest 4.11 release introduces Instanced Stereo Rendering, check the video below for a comparison video of how that works.

"Basically, we're utilizing hardware instancing to draw both eyes simultaneously with a single draw call and pass through the render loop. This cuts down render thread CPU time significantly and also improves GPU performance. Bullet Train was seeing ~15 – 20% CPU improvement on the render thread and ~7 – 10% improvement on the GPU." – Ryan Vance.

To enable this feature in 4.11 and above, go to your Project Settings and look for "Instanced Stereo" under the Rendering category.



Disable Heavy Post-Processors

- Add a Post Process(PP) volume to your level if there is not already one there.
- Select the PP volume and in the Post Process Volume section enable the **Unbound** option so that the settings in the PP volume will be applied to the entire level.



- Expand the Settings of the Post Process Volume and then go through each section and **disable any active PP** settings by enabling that property by clicking on it and then set the value from the default, usually 1.0, to 0 to disable the feature.



- When doing this you will not need to hit every section and **set all the properties to 0**. Instead first disable the really heavy hitting features like Lens Flares, Screen Space reflections, Temporal AA, SSAO, and anything else that might have an impact on performance.
- While a lot of the features are disabled by setting things in your .INI this ensures that nothing will happen to performance if someone deletes the .INI by mistake.

VR in UE 5...Is Now Possible



Hiroyan
@H1ROY4N

The Epic gods have done it again... Lumen and Nanite in [#UnrealEngine](#) [#VirtualReality](#) using [#UE5-main](#) (5.2). This is by far the most detail and realism I have ever seen in [#VR](#).



Nanite and Lumen initially didn't work in VR. Developers of VR apps had to use the legacy geometry and lighting systems, negating many of the advantages of the new engine. But with [Unreal Engine 5.1](#) Nanite & Lumen now work with VR, [as confirmed by](#) Heavenuue CEO Alex Coulombe. Coulombe told us his test build maintained 90 frames per second on laptop with a mobile RTX 3080, streaming to a Quest headset using Air Link. He [posted a public download link](#) of his test app for anyone interest in testing Nanite & Lumen in VR.

The public Unreal Engine 5 GitHub build works for PC VR, but Meta has [its own UE5 branch it recommends](#) for native Quest 2 development. Nanite & Lumen don't work on Android though, and aren't designed to work with mobile chips in general, so native Quest 2 games will be limited to the legacy tech.

<https://uploadvr.com/unreal-engine-5-lumen-nanite-vr/>

<https://twitter.com/i/status/1579060446937550848>