Peter Gustafson
writing sample
Epic Games Mar 9, 2021

VR Template in Unreal Engine | Unreal Engine 5.1 Documentation

The **VR Template** is designed to be a starting point for all of your virtual reality (VR) projects in **Unreal Engine**. It includes encapsulated logic for teleport locomotion, an example VR spectator Blueprint and common input actions, such as grabbing and attaching items to your hand.

This page shows you how to get started with the VR Template and how to use it to create your own VR experiences.

Supported Devices

The VR Template is compatible with the following devices:

* Oculus Mobile
    * Quest 1
    * Quest 2
* Oculus PC
    * Rift S
    * Quest with Oculus Link
* Steam VR
    * Valve Index
    * HTC Vive
* Windows Mixed Reality

Before you can use the VR Template, your device will need to be set up for development in Unreal Engine. See the documentation in [XR Development](https://docs.unrealengine.com/5.1/en-US/developing-for-xr-experiences-in-unreal-engine) for your device to ensure it's set up

correctly.

OpenXR

The VR Template uses the [OpenXR](https://docs.unrealengine.com/5.1/en-US/developing-for-head-mounted-experiences-with-openxr-in-unreal-engine) framework, the multi-company standard for VR and AR development. With the OpenXR plugin in Unreal Engine, the template's logic works on multiple platforms and devices without any platform-specific checks or calls.

The OpenXR plugin in Unreal Engine supports extension plugins. You can find extension plugins in the Marketplace or create your own to add functionality to OpenXR that isn't currently in the engine by default.

Pawn, Game Mode, and Player Start

The following objects determine the rules of the VR Template experience and how it's set up.

* UE Object : Pawn
  * Name of Object in VR Template : VRPawn
  * Location : Content > VRTemplate > Blueprints
  * Description : In Unreal Engine, a Pawn is the physical representation of the user and defines how the user interacts with the virtual world. In the VR Template, the Pawn contains the logic for input events from the motion controllers.
* UE Object : Game Mode
  * Name of Object in VR Template : VRGameMode
  * Location : Content > VRTemplate > Blueprints
  * Description : The Game Mode object defines the rules of the experience, such as which Pawn object to use. The Game Mode is set in Project Settings, in the Maps & Modes section.
* UE Object : Player Start
  * Name of Object in VR Template

: Player Start
* Location :
World Outliner
* Description :
The Player Start Actor defines where the Pawn is spawned in the virtual world. For VR experiences, the Player Start's origin is the tracking origin in the virtual world. Since the VR Template is designed for standing experiences in VR, the Player Start is located at floor level.

Input

Input in the VRTemplate is based on the [Action and Axis Mappings Input System](https://docs.unrealengine.com/5.1/en-US/input-in-unreal-engine) in Unreal Engine. See [Input with OpenXR](https://docs.unrealengine.com/5.1/en-US/making-interactive-xr-experiences-in-unreal-engine) for more information on how to set up input with OpenXR.

Movement

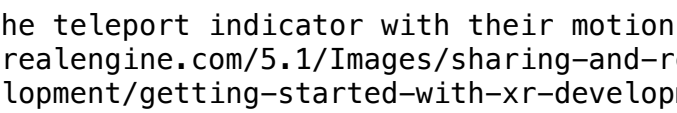
Movement in VR experiences is referred to as _locomotion_. In the VR Template, there are two types of locomotion: **Teleport** and **Snap Turn**. In the Blueprint Editor, open the **VRPawn** to see how both are implemented.

Teleport

To teleport to different locations in the Level:

1. Move your right Motion Controller's thumbstick or trackpad in the direction you want to move. The Teleport visualizer appears in the level to indicate where you will move to.

2. Release the thumbstick or trackpad to teleport to the selected location.



Snap Turn

To rotate your virtual character without moving your head, move your left Motion Controller's thumbstick or trackpad in the direction you want to turn.

 ![User performing snap turns to rotate their view without moving](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_5.jpg)

Setting Allowed Areas for Movement

The Level uses a [Navigation Mesh](https://docs.unrealengine.com/5.1/en-US/content-examples-sample-project-for-unreal-engine) to mark where the user is allowed to move. See the **NavModifierVolume** asset **NavModifier_NoTeleport** as an example of how this can be implemented. The asset's **Area Class** parameter is set to **NavArea_Null**, which prevents the user from moving to that volume.

 ![The teleport indicator disappears when in the No Teleport Zone](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_6.jpg)

The teleport visualizer disappears when it's in the **NavModifier_NoTeleport** volume.

Press the P key on your keyboard to toggle the visualization of navigable areas.

 ![Visualization of the navigable areas in the level](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_7.jpg)

Grab

The VR Template shows a couple different ways to pick up objects and attach them to your hands.

To grab an object in the level:

*   Reach out to a grabbable object, then press and hold the **Grip** button on your Motion Controller.

*   This creates a [Sphere Trace](https://docs.unrealengine.com/5.1/en-US/traces-in-unreal-engine---overview) around the position of your Motion Controller's GripLocation. If there's an Actor with a

GrabComponent in that sphere, it becomes attached to the hand that pressed the **Grip** button.

*   Release the **Grip** button on the same Motion Controller to detach the object from your hand.

To enable grabbing on an actor, add the GrabComponent blueprint to the actor and select the Grab Type in the Details window. On BeginPlay, the collision profile of the component's parent is set to PhysicsActor, which is the trace channel used for the Sphere Trace in VRPawn.

You can open the GrabComponent Blueprint class to see how the grab system is implemented in the VR template.

GrabType

You can set the type of grab in an object's **GrabComponent** to define how the object attaches to your hand.

The following grab types are defined in the **GrabType Enum** asset:

*   **Free**: The Actor stays in the position and orientation relative to where the Motion Controller grabbed it. See the small cubes as an example.

     ![User grabbing and rotating cubes with their motion controller](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_8.jpg)

*   **Snap**: The Actor has a specific position and orientation relative to the Motion Controller. See the pistols as an example.

     ![User picking up the pistols which snap to the motion controllers](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_9.jpg)

*   **Custom:** With this option, you can add your own logic for the grab action, using the **OnGrabbed** and **OnDropped** events. You can also utilize other exposed variables, such as the **bIsHeld** boolean variable, which is a flag that specifies whether an object is currently held by the user. Examples of other types of custom grab actions that can be created include two-handed grabs, dials, levers, and other complex behaviors, such as Actors that don't overlap geometry when held.

You can define a haptic effect that occurs when grabbing an object by setting the OnGrabHaptic Effect variable in the GrabComponent for the object. An example of a haptic effect includes Motion Controller vibrations.

Important functions used in GrabComponent are abstracted into a common interface using the Blueprint Interface asset VRInteraction BPI. With this Blueprint Interface asset, the VRPawn can simplify its logic and avoid using multiple checks for each kind of object. For more on Blueprint Interface and their benefits, see [Blueprint Interface](https://docs.unrealengine.com/5.1/en-US/blueprint-interface-in-unreal-engine).

Pressing the menu button on your motion controller opens the VR Template's menu. The menu is built with Unreal Motion Graphics (UMG). See

\[UMG UI Designer\](programming-and-scripting/programming-language-implementation/cpp-in-unreal-engine/unreal-engine-cpp-tutorials/UMG)

for more information on how to use this tool.

 ![User interacting with the VR menu with two options: restart and real life](https://docs.unrealengine.com/5.1/Images/sharing-and-releasing-projects/xr-development/getting-started-with-xr-development/vr-template/image_10.jpg)

In the **Content Browser**, **Content > VRTemplate > Blueprints > WidgetMenu** is the UMG asset for designing and creating logic for the menu, and the **Menu** Blueprint defines how the controller interacts with the Widget.

VR Spectator

With [Spectator Screen Mode](https://docs.unrealengine.com/5.1/en-US/virtual-reality-spectator-screen-in-unreal-engine), others can view the VR experience while someone uses the head-mounted display (HMD). The VR Template implements an example of Spectator Screen Mode using the **VR Spectator** Blueprint.

There are two ways to enable Spectator Mode in the VR Template:

*   Press **Tab** to toggle Spectator Mode during the session.

*   Set **VRSpectator bSpectatorEnabled** to **true**.

VR Spectator is not compatible with Mobile VR devices, such as Oculus Quest..

You can control the VR Spectator to view the user with the HMD in the virtual world, with your mouse and keyboard or a gamepad using the following inputs:

* Action : Toggle VRSpectator
  * Mouse & Keyboard Input : Tab
  * Gamepad Input : Bottom Face button
* Action : Move Forward
  * Mouse & Keyboard Input : W
  * Gamepad Input : Left Thumbstick up
* Action : Move Backward
  * Mouse & Keyboard Input : S
  * Gamepad Input : Left Thumbstick down
* Action : Move Left
  * Mouse & Keyboard Input : A
  * Gamepad Input : Left Thumbstick left
* Action : Move Right
  * Mouse & Keyboard Input : D
  * Gamepad Input : Left Thumbstick right
* Action : Move Up
  * Mouse & Keyboard Input : E or Spacebar
  * Gamepad Input : Right shoulder button
* Action : Move Down
  * Mouse & Keyboard Input : Q
  * Gamepad Input :

```
                Left shoulder button
*                       Action                  :
        Pitch
  *                     Mouse & Keyboard Input                  :
                        Move mouse forward or backward

  *                     Gamepad Input                   :
                Right thumbstick Y axis
*                       Action                  :
        Yaw
  *                     Mouse & Keyboard Input                  :
                        Move mouse left or right
  *                     Gamepad Input                   :
                Right thumbstick X axis
*                       Action                  :
        Change Field of View (FOV)
  *                     Mouse & Keyboard Input                  :
                        Move mouse wheel
  *                     Gamepad Input                   :
                Right trigger / left trigger
*                       Action                  :
        Reset FOV
  *                     Mouse & Keyboard Input                  :
                        Click middle mouse button
  *                     Gamepad Input                   :
                N/A
*                       Action                  :
        Toggle Fade In and Out
  *                     Mouse & Keyboard Input                  :
                        F
  *                     Gamepad Input                   :
                Top Face button
*                       Action                  :
        Reset Rotation
  *                     Mouse & Keyboard Input                  :
                        R
  *                     Gamepad Input                   :
                N/A
```

You can use the VR Spectator as a starting point for implementing multiplayer experiences on the same PC.