Building to a VR headset

This series of articles will describe the process of building to VR headsets and go in-depth for building to some of the most common platforms. However, you are encouraged to conduct individual research since development can have different results on different devices. Please ask a technician for assistance if you find any issues.

- **How to build a project onto a VR headset**
- **How to build to a Meta Quest 2**
- **How to build to a Vive Pro or Vive Pro Eye**

How to build a project onto a VR headset

1. Install Unreal Engine and Required VR SDKs:

Download and install the latest version of Unreal Engine from the Epic Games website. Identify the VR platform and headset you want to target (e.g., Oculus Rift, HTC Vive, Oculus Quest, etc.). Install the corresponding VR SDK and plugin for Unreal Engine. For example, for Oculus headsets, you can use the Oculus VR Plugin.

2. Create a New VR Project:

Launch Unreal Engine and create a new project. Choose the appropriate VR template. If you're new to VR development, the "Virtual Reality" or "VR Template" is a good starting point.

3. Configure Project Settings:

Go to Edit > Project Settings. Under Platforms > Virtual Reality, enable the VR platforms you want to support (e.g., Oculus, SteamVR). Configure VR-specific settings like tracking method (e.g., room-scale, seated) and controller bindings.

4. Design VR Environment and Assets:

Create or import 3D assets for your VR environment using external tools or Unreal Engine's built-in modeling tools. Set up the VR-specific player pawn (character) and camera. Make sure the camera is set to match the user's head movement.

5. Implement VR Interactions:

Set up VR-specific input bindings for the headset and controllers. Unreal Engine provides a VR Motion Controller component for handling controller input. Implement VR interactions, such as grabbing objects, interacting with buttons, and teleportation. For advanced interactions, you may need to use Blueprints or C++ to handle custom logic.

6. Optimize for Performance:

VR applications require high performance for a smooth experience. Optimize your assets and level design to maintain a stable frame rate (e.g., optimize textures, use LODs, reduce draw calls). Use level streaming techniques to manage resources efficiently, especially for large and complex environments.

7. Test in VR:

Connect your VR headset to your development machine. Launch the VR app from the Unreal Engine editor to test it in VR. Test all interactions and movements to ensure everything works as intended.

8. Package the App:

Once your VR app is ready for distribution, package it for the target VR platform. Go to File > Package Project > (Choose the target platform). Follow the on-screen instructions to generate the appropriate package for your VR headset.

9. Distribute the App:

For PC VR platforms (e.g., Oculus Rift, HTC Vive), you can distribute the app through online stores like Oculus Store or SteamVR. Follow the submission guidelines for each platform. For standalone VR headsets (e.g., Oculus Quest), you may need to follow specific distribution channels, like Oculus Store for Quest apps.

10. Update and Maintain:

Regularly update and maintain your VR app to improve performance, add new features, and stay compatible with the latest VR hardware and software updates. Remember that building VR apps can be a complex process, and it's crucial to test and optimize your app thoroughly to ensure a comfortable and immersive experience for users. Refer to the official documentation and community resources provided by Unreal Engine and the specific VR platform you're targeting for more in-depth guidance and troubleshooting.

How to build to a Meta Quest 2

Building a VR app for the Meta Quest 2 from Unreal Engine involves a series of steps, including setting up the development environment, configuring project settings, and packaging the app. Below are detailed instructions to guide you through the process:

1. **Install Unreal Engine and Required SDKs:**
   - Download and install the latest version of Unreal Engine from the Epic Games website (https://www.unrealengine.com/).
   - Ensure you have the Android SDK and NDK installed on your development machine. You can download these tools through Android Studio or separately from the Android developer website (https://developer.android.com/studio).

2. **Set Up Oculus Development Hub:**
   - Download and install Oculus Development Hub from the Oculus developer website (https://developer.oculus.com/).
   - Log in with your Oculus developer account or create a new one if you don't have one already.

3. **Enable Developer Mode on Quest 2:**
   - On your Meta Quest 2, navigate to Settings > Oculus Quest 2 > More Settings > Developer Mode.
   - Enable Developer Mode and follow the instructions to set it up.

4. **Create a New VR Project:**
   - Launch Unreal Engine and create a new project.
   - Choose the "Virtual Reality" or "VR Template" as a starting point for your Meta Quest 2 project.

5. **Configure Project Settings:**
   - Go to Edit > Project Settings.
   - Under Platforms > Android, configure Android-specific settings such as package name, minimum SDK version (recommended to use Android 7.0 or higher), and any required permissions.

- Under Platforms > Android > APKPackaging, enable "Package For Oculus Mobile Devices."
- Enter your Oculus app ID obtained from the Oculus developer dashboard.

6. **Set Up Oculus Quest 2 for Development:**
   - Connect your Meta Quest 2 to your development machine using a USB cable.
   - In the Oculus Development Hub, you should see your connected device under "My Devices."
   - Follow the prompts to authorize the device for development.

7. **Install Oculus Quest 2 ADB Drivers (if needed):**
   - If your computer doesn't recognize the Quest 2, you may need to install the Oculus ADB Drivers. The drivers can be installed through the Oculus Development Hub or separately from the Oculus website.

8. **Build and Run the VR App:**
   - In Unreal Engine, select the "Launch" or "Package" option for Android.
   - Choose "Quest 2" as the target device and build the APK.
   - Once the APK is built, it will be automatically installed on your connected Meta Quest 2 device.

9. **Testing and Debugging:**
   - Disconnect the Meta Quest 2 from your development machine after the app is installed.
   - Put on the headset and navigate to the "Unknown Sources" section in the Library to find and launch your app.
   - Test the app thoroughly on the Meta Quest 2, ensuring all VR interactions and functionalities work as intended.

10. **Optimize Performance (if needed):**

- Use Unreal Engine's profiling tools to identify performance bottlenecks and optimize your VR app for smooth performance on the Meta Quest 2.
- Optimize textures, reduce draw calls, and use level streaming to manage resources efficiently.

11. **Submit to Oculus Store (Optional):**

   - If you plan to distribute your app through the Oculus Store, you need to follow the Oculus submission guidelines and perform necessary testing for approval.

Remember, Meta Quest 2 development with Unreal Engine requires attention to performance optimization and adherence to Oculus store guidelines (if applicable). Regularly test your app on the device to ensure a comfortable and immersive experience for users. Also, keep up-to-date with the latest Unreal Engine updates and Oculus SDK changes to maintain compatibility with future releases.

How to build to a Vive Pro or Vive Pro Eye

Building a VR app for the Vive Pro Eye with Unreal Engine involves several steps, including setting up the development environment, configuring project settings, and packaging the app. Below are detailed instructions to guide you through the process:

1. **Install Unreal Engine and Required SDKs:**
   - Download and install the latest version of Unreal Engine from the Epic Games website (https://www.unrealengine.com/).
   - Ensure you have SteamVR installed on your development machine. You can download SteamVR through the Steam client (https://store.steampowered.com/steamvr).

2. **Set Up SteamVR for Vive Pro Eye:**
   - Connect your Vive Pro Eye headset to your development machine.
   - Launch SteamVR and follow the on-screen instructions to set up the headset and controllers.

3. **Create a New VR Project:**
   - Launch Unreal Engine and create a new project.
   - Choose the "Virtual Reality" or "VR Template" as a starting point for your Vive Pro Eye project.

4. **Configure Project Settings:**
   - Go to Edit > Project Settings.
   - Under Platforms > Windows, configure Windows-specific settings such as the package name and minimum supported Windows version.
   - Under Platforms > Windows > VR, enable "SteamVR" as the target VR platform.
   - Configure VR-specific settings like tracking method (e.g., room-scale, seated) and controller bindings.

5. **Design VR Environment and Assets:**
   - Create or import 3D assets for your VR environment using external tools or Unreal Engine's built-in modeling tools.
   - Set up the VR-specific player pawn (character) and camera. Make sure the camera is set to match the user's head movement.

6. **Implement VR Interactions:**
   - Set up VR-specific input bindings for the headset and controllers. Unreal Engine provides a Motion Controller component for handling controller input.
   - Implement VR interactions, such as grabbing objects, interacting with buttons, and teleportation.
   - For advanced interactions, you may need to use Blueprints or C++ to handle custom logic.

7. **Optimize for Performance:**
   - VR applications require high performance for a smooth experience. Optimize your assets and level design to maintain a stable frame rate (e.g., optimize textures, use LODs, reduce draw calls).
   - Use level streaming techniques to manage resources efficiently, especially for large and complex environments.

8. **Build and Run the VR App:**
   - In Unreal Engine, select the "Launch" or "Package" option for Windows.
   - Choose "VR Preview" as the target device and build the executable.
   - Once the executable is built, it will automatically launch SteamVR, and you can put on your Vive Pro Eye headset to test the app.

9. **Testing and Debugging:**
   - Test the app thoroughly on the Vive Pro Eye headset, ensuring all VR interactions and functionalities work as intended.
   - Use SteamVR's built-in debugging tools to monitor performance and resolve any issues that arise during testing.

10. **Optimize Eye Tracking Interactions (if needed):**

   - If your VR app utilizes eye tracking features of the Vive Pro Eye, ensure that the interactions and visual feedback are optimized for a seamless and comfortable user experience.
   - Use Unreal Engine's Eye Tracking API to access and utilize eye tracking data in your app.

11. **Package the App for Distribution:**

- Once your VR app is ready for distribution, package it for Windows.
- Follow the guidelines for content distribution on the Viveport store or other distribution platforms if you plan to distribute your app to a broader audience.

Remember, Vive Pro Eye development with Unreal Engine requires attention to performance optimization and compatibility with SteamVR. Regularly test your app on the Vive Pro Eye headset to ensure a comfortable and immersive experience for users, and consider integrating eye tracking features for a more innovative user interaction.