Final Deliverable

Rose Garden XR

Unity Game (XR Version)

This is the final Unity project that pushes builds for the Meta Quest headsets. A windows machine with a dedicated graphics card is required to develop in VR.

Github: https://github.com/br-zhou/Rose-Garden-VR

Development Environment:

Unity	6000.0.37f1 LTS
Computer	ROG Strix G16 (RTX 4060 Laptop GPU, I9-1400HX, 16GB ram)
OS	Windows 11 Home 24H2 (26100.3775)

Project Setup:

Step 1. Clone the Github repository

Step 2. Open the project in Unity
- Make sure to install 6000.0.37f1 LTS with android build support.
- Ignore the "Enter Safe Mode" warning - we are just missing some dependencies!

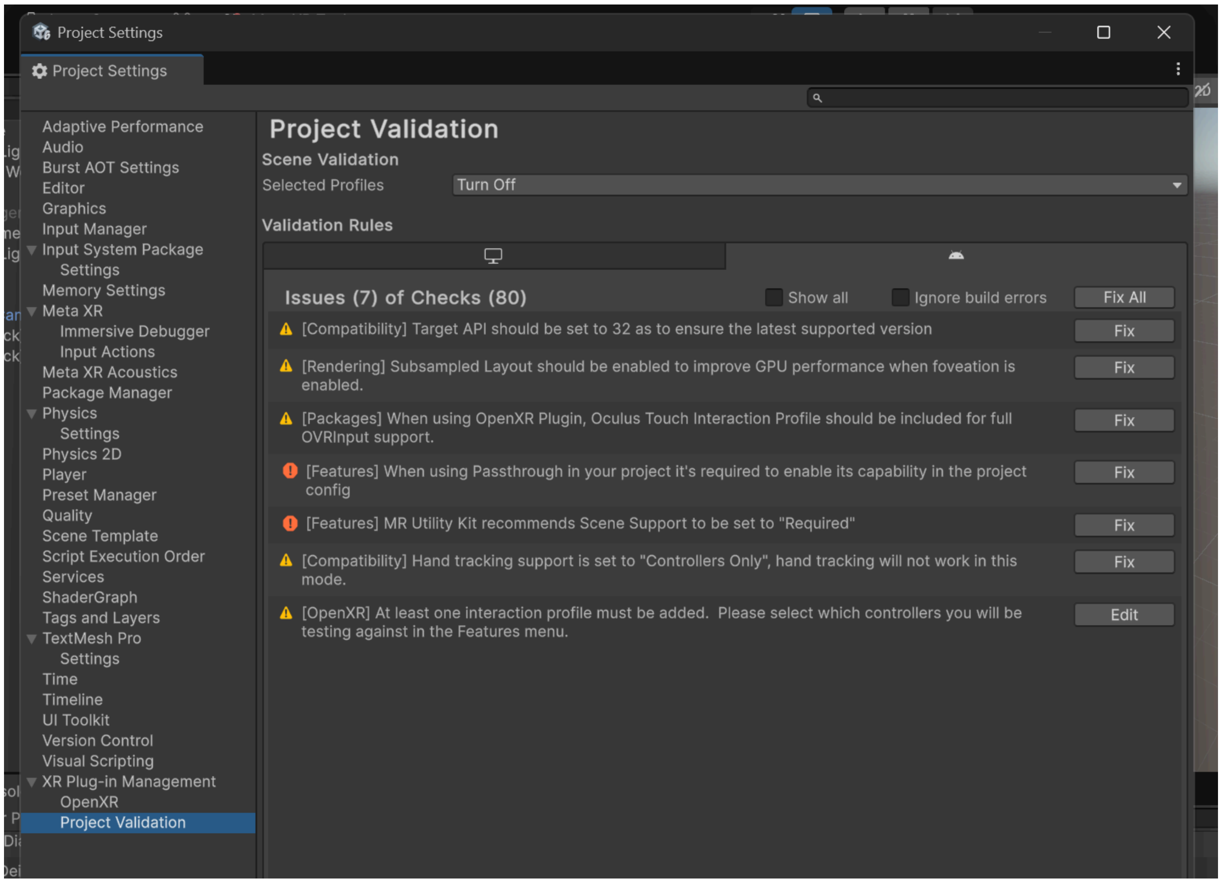
Step 3. Download the Firebase Unity SDK from here:
- https://firebase.google.com/download/unity

Step 4. Import the **FirebaseDatabase.unitypackage** into the project
- You can do this by opening the .unitypackage in the file explorer while the Rose Garden VR project is opened in Unity.

Step 5. Import FireBase API keys into Assets/StreamingAssets (create folder if needed)
- Ignore this step if you don't have a Firebase backend set up yet.
- This tutorial gives a visual for steps 3-5: Firebase Unity Integration Tutorial
- If you want to skip the Firebase Unity Integration, ignore this step.

Step 7. Meta Quest Developer Mode Setup
- Follow this tutorial to configure the required settings to develop for Meta Quest: Meta Quest Unity Setup

Step 8. Project Validation
- Make sure to fix any build issues in the project settings!
- Example Screenshots on Next Page.



Project Validation

Scene Validation

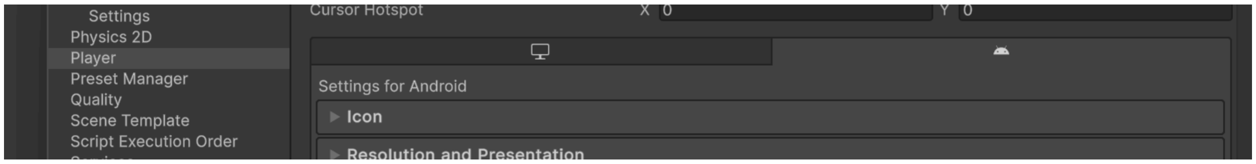
Selected Profiles: Turn Off

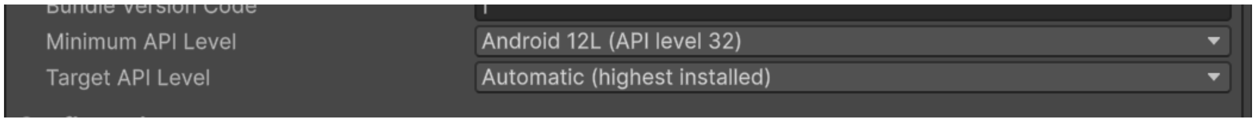
Validation Rules

Issues (7) of Checks (80) — Show all — Ignore build errors — Fix All

- [Compatibility] Target API should be set to 32 as to ensure the latest supported version — Fix
- [Rendering] Subsampled Layout should be enabled to improve GPU performance when foveation is enabled. — Fix
- [Packages] When using OpenXR Plugin, Oculus Touch Interaction Profile should be included for full OVRInput support. — Fix
- [Features] When using Passthrough in your project it's required to enable its capability in the project config — Fix
- [Features] MR Utility Kit recommends Scene Support to be set to "Required" — Fix
- [Compatibility] Hand tracking support is set to "Controllers Only", hand tracking will not work in this mode. — Fix
- [OpenXR] At least one interaction profile must be added. Please select which controllers you will be testing against in the Features menu. — Edit

Tip: Make sure to resolve any red errors.

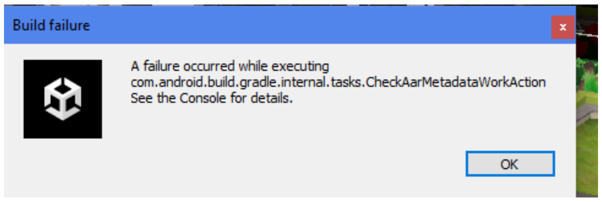




Minimum API Level: Android 12L (API level 32)

Target API Level: Automatic (highest installed)

Build Issues: When building for the meta Quest, you need to set your Project Settings > Player > Android API level to the ones above. Builds will fail otherwise.

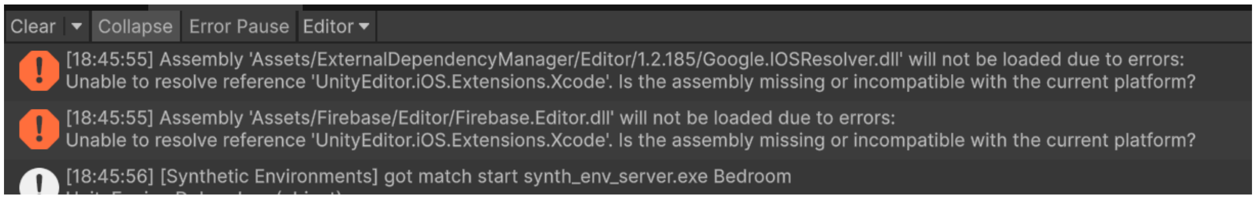


Build failure

A failure occurred while executing com.android.build.gradle.internal.tasks.CheckAarMetadataWorkAction
See the Console for details.

OK

You will see this error when building for the headset if you don't set the correct API level. You can safely ignore any warnings that pop up during the build stage.



```
[18:45:55] Assembly 'Assets/ExternalDependencyManager/Editor/1.2.185/Google.IOSResolver.dll' will not be loaded due to errors:
Unable to resolve reference 'UnityEditor.iOS.Extensions.Xcode'. Is the assembly missing or incompatible with the current platform?
[18:45:55] Assembly 'Assets/Firebase/Editor/Firebase.Editor.dll' will not be loaded due to errors:
Unable to resolve reference 'UnityEditor.iOS.Extensions.Xcode'. Is the assembly missing or incompatible with the current platform?
[18:45:56] [Synthetic Environments] got match start synth_env_server.exe Bedroom
```

Expected Errors: You will see two errors when running the dev build. Don't worry! They're related to the IOS builds for Firebase. Since the Quest 3 is an android platform, we can safely ignore them.

Note: A full copy of the XR project files has also been cloned onto one of the EML desktops's external drives.

Additional Information:
Please reference the repository README for additional information on the key scripts used in the project.

Practical Warnings: Make sure to update your drivers when developing! The newer 40 series graphics cards have graphical issues and crashes often. These issues are not present in older 20 series cards. It is better to develop on a more stable system, but it is preferable to not develop on Unity at all. Since Meta recently released their passthrough API, there should be a touch designer implementation soon.

Unity Game (FPS Version)

This is a FPS (first-person shooter) version of the viewfinder project. We have two versions of our game we keep in synchronization because developers on Mac can not directly run VR. The only difference between the two projects is that XR uses a VR headset to run and FPS is controlled using mouse and keyboard. Usually, changes are first made in FPS then copied to XR.

Code & Assets

Github: https://github.com/Amon3141/rose-garden-viewfinder-fps

Below are the locations of important assets:

/Assets/Objects	3D objects
/Assets/Material	Materials for the 3D objects
/Assets/Scripts	C# scripts
/Animation	Animation Controllers
/Firebase	Database-related

More detailed descriptions of each script are written in the README.md file.

Development Environment

Unity	6000.0.36f1 LTS
Computer	MacBook Pro (2020), Apple M1 chip, 16GB
OS	macOS Sequoia 15.4

Project Setup

Step 1. Clone the Github repository
Step 2. Open the project in Unity
- Make sure to install 6000.0.36f1 LTS with android build support

Step 3. Download the Firebase Unity SDK from here:
- https://firebase.google.com/download/unity

Step 4. Import the **FirebaseDatabase.unitypackage** into the project
Step 5. Import FireBase API keys into Assets\StreamingAssets
- Ignore this step if you don't have a Firebase backend set up yet.
- This tutorial gives a visual for steps 3-5: Firebase Unity Integration Tutorial
- If you want to skip the Firebase Unity Integration, ignore this step.

Additional Information

Please reference the repository README for additional information on the key scripts used in the project.

Mobile Web Application

The mobile web application is a website used to send custom messages into the XR experience. Both the frontend and backend can be hosted for free on Google's Firebase backend service.

Framework & Libraries: React 19.0.0, Vite, TypeScript
Testing Environment

- Browser: Chrome
- Screen Size: Responsive layout tested on desktop and mobile viewports
- OS: Deployed using Google Firebase

Github: https://github.com/zhanginc/message-app-mdia-470

- Link to the github for the frontend application
- Assets for the web application are within the public folder in the github repository

Hosting the Application:
This part is focused on how to attach a backend (Firebase) to the frontend application. You will need a backend to store the messages that user inputs. It is broken down in two parts:

(1) Deploying the initial web application on your Firebase account

(2) Creating and connecting the real-time database

Setting up the Web Application:
First, clone the repository **Github Repository**. After cloning the repository, you need to add a .env file at the root level (frontend folder). The .env file is keeps the app's ***secrets*** from ever being committed to version control (being seen by other people):
Copy and paste this into your .env file:

```
VITE_FIREBASE_API_KEY=
VITE_FIREBASE_AUTH_DOMAIN=
VITE_FIREBASE_PROJECT_ID=
VITE_FIREBASE_STORAGE_BUCKET=
VITE_FIREBASE_MESSAGING_SENDER_ID=
VITE_FIREBASE_APP_ID=
VITE_FIREBASE_MEASUREMENT_ID=
```

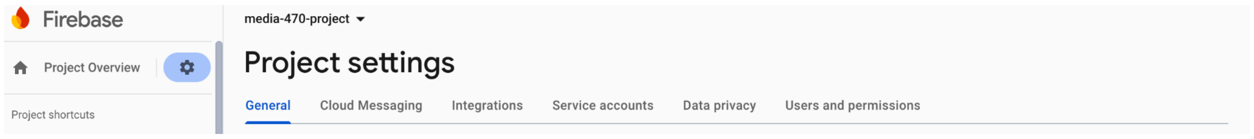
More info on securing API keys: Securing Firebase Keys in Vite

Setting up Firebase:

1. You will need a Firebase account

2. Create a firebase project following these instructions

3. Once the project is created, then go to Project Settings click on the option to add Firebase to your web app.



It will generate a firebaseConfig, which is what we will use to input in the .env file. For example, if my firebaseConfig has the variable **apiKey** containing a value of **1234567.** This value should be copy and pasted into **VITE_FIREBASE_API_KEY=**

VITE_FIREBASE_API_KEY=1234567

Ensure to copy + paste the remaining parts of the firebaseConfig into the .env file.

Running the Application:

Ensure that you are in the frontend folder before beginning. This next command runs the web app on local host, so you can test out changes if things are working

```
npm run dev
```

Deploying to Firebase:

When we want to push our changes into production, we need to run these two commands.

npm run build (dist folder, which are the public files that firebase serves)

firebase deploy --only hosting (pushes build to Firebase)

Creating the Realtime Database:

In order for the messaging to work, we will need a backend to hold the data.

To moderate the message inputs, add a Realtime Database following these instructions

Once the Realtime Database is created, it will contain a simple user interface that the moderator can view messages and delete any that are offensive/profane etc.

Now the frontend application should be hosted on Firebase, alongside with the Real Time Database backend.