Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

ArchiVR Project Documentation

Authors: marcus.hudritsch@bfh.ch, sascha.ledermann@bfh.ch

29. Jan. 2026

Version History

20.12.2025 v0.90: Conversion from AHB module documentation to ArchiVR documentation
15.01.2026 v0.91: Added 2 Quick Guide for ArchiVR
29.01.2026 v1.00: Removed all old MD2 references. Corresponds to ArchiVR-Viewer 0.1.5

Table of Contents:

1 Overview

Welcome to the ArchiVR project documentation. The ArchiVR project has two main goals:

- Make visiting 3D spaces in virtual reality (VR) as simple as visiting websites.
- Make deploying architectural 3D scenes to VR as simple and free as possible.

The ArchiVR project consists of

- a web platform (archi-vr.ti.bfh.ch) for managing users and 3D scenes,
- a Unity game engine plugin that enables uploading 3D scenes to ArchiVR, and
- a free viewer application for VR headsets.

The documentation is structured as follows:

1. **Overview**: After this introduction, we will briefly explain Virtual Reality (VR), Mixed Reality (MR), and Augmented Reality (AR). What do they have in common, and what is different between them?
2. **Quick Guide for ArchiVR**: This chapter gives you a quick introduction to the ArchiVR website, the ArchiVR viewer in the VR headset, and how you can create a simple scene for ArchiVR.
3. **Introduction to Unity**: VR applications are almost exclusively used for computer games. Therefore, we must first learn how to use a *Game Engine* to create a VR or AR application for a popular VR headset.
4. **Preparations in CAD Systems**: Popular ***C****omputer-****A****ided* ***D****esign* tools are not yet able to create applications for VR headsets. This chapter outlines some changes you may need to make to your project to export it in specific file formats.
5. **Adapt your Project for ArchiVR**: This chapter covers how you adapt your own architectural project for ArchiVR.
6. **Materials and Lighting in Unity**: Here, we will examine Unity's materials and lights and how to adjust them optimally for our architectural scenes.
7. **Case Study: Circular Tower**: This chapter describes the VR development of a real architectural project. We received the original CAD files from the office Inhelder Osterwalder Architekten in Biel, Switzerland.
8. **Adding Functionality to a VR Scene**: You or your colleagues can teach you how to add links to other locations or scenes. Another topic will be how to animate objects, such as a door opening.
9. **Appendix**: This final chapter provides additional detail that would otherwise overload the weekly tutorial steps.

1.1 What is AR, MR, VR, or XR?

- ***AR*** stands for ***Augmented Reality***, meaning we can augment, add, or enrich a captured or live video with 3D computer graphics on a mobile device such as a phone or tablet. The added 3D object moves with the real, filmed background, giving the impression that an augmented object is placed in the real world.
- ***MR*** stands for ***Mixed Reality*** and refers to an AR variant using see-through glasses, where augmented graphics are blended into the view. Microsoft defined this term for its MR headset, *HoloLens*. In the future, more MR applications will be available for VR headsets equipped with stereo color cameras at the front.

- ***VR*** stands for ***Virtual Reality***, where the adjective virtual means artificial. A VR headset provides a viewer with a perfectly stereoscopic view of a virtual 3D scene, so the user can no longer perceive the real world around them.
- **XR** stands for **Extended Reality** and refers to the collective term for AR, MR, and VR.

Differences and Similarities: All these technologies share the commonality that rendering depends on the mobile device's or headset's position and orientation. Because this is done at a high refresh rate, the viewer feels a strong sense of presence around the augmented object or, in VR, of being in the virtual world. This feeling is called ***immersion***. Immersion is higher when the device is mounted on the viewer's head and delivers correct images to each eye. This viewer dependence is realized by tracking the surrounding space. In other words, the device must continuously detect its position and orientation to determine the correct projection for the augmented graphics. This process is called ***pose estimation***, which stands for **p**osition, **o**rientation, and **s**cale **e**stimation. If the graphics rendering frame rate is too slow or the pose estimation is not precise, our brain is not fooled, and we don't get the immersive feeling.

AR: Where is the iPad relative to the filmed environment?

MR: Where is the Hololens headset in the room?

VR: Where is the VR headset in the room?

ArchiVR is, for the moment, only about VR. In the future, we will add support for mixed-reality scenes in VR headsets.

In chapter 9.1 Introduction to VR, you can find more background information on VR.

1.3 Expected Skills, Hardware & Software

The following hardware and software requirements apply only when creating scenes for ArchiVR. For viewing ArchiVR scenes, you only need a VR headset and a basic computer.

1.3.1 Expected Skills

- General computer user knowledge:
  - You should know what files, folders, and links are on your computer and how to navigate your file system.
  - How to install new software on your computer.
    You will therefore need administrator privileges.
  - No programming skills are necessary.
- Basic knowledge of your architectural design tool.

1.3.2 Expected Hardware & Software

- **VR Headset:** We recommend a Meta mobile VR headset, such as the Meta Quest 3, which costs around 500 CHF. We also recommend the Elite Strap and carry case as an additional accessory. We provide more information in VR headsets in 9.1.2 Meta Quest.

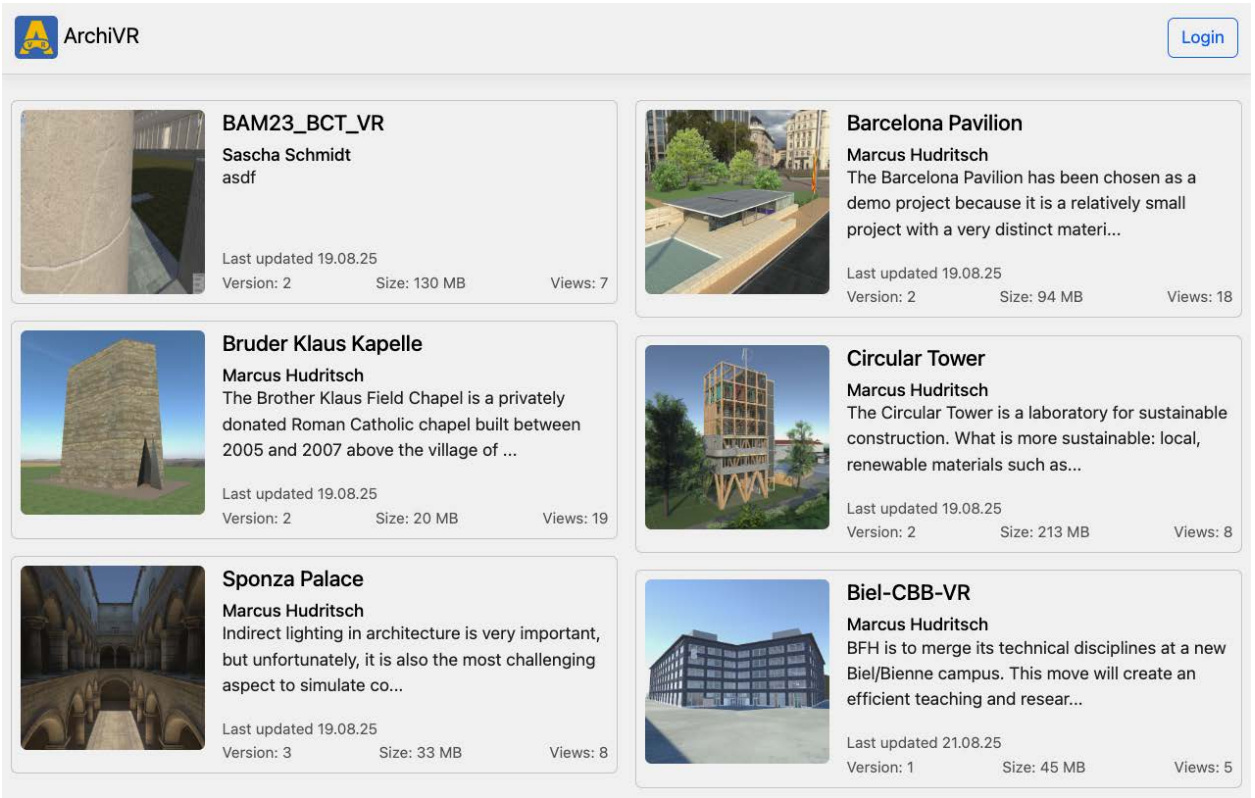
- **Capable Laptop for 3D Computer Graphics:** Architectural design applications are very demanding for every computer. We will use Blender and Unity for this course. Even if these applications run on older Windows or macOS computers, managing them will be exhausting if the machine is more than 5 years old. The newer and more powerful the computer, the better these applications perform.
  - The laptop should also have at least **16 GB of RAM** (internal memory at runtime). You should close any unnecessary applications if you have only 16GB of RAM.
  - You need at least **20 GB of free internal disk space** to install Unity.
  - Each 3D project with Unity can quickly use up to **10-20 GB of disk space**. These projects can also be on an external USB disk.
  - Use the power adapter. Windows laptops on battery power reduce performance.
- **Big Monitor**: This is impossible if you work on a laptop in school. But if you work from home, a larger second monitor is recommended. The bigger, the better.
- **Mouse with 3 Buttons**: Working with 3D design software requires a mouse with three buttons. A laptop with only a trackpad will not work.
- **Install Unity**: To create scenes for ArchiVR, you need to install the Unity engine. This is explained later in 2.3.1 Install Unity.

2 Quick Guide for ArchiVR

This quick guide provides a brief overview of the ArchiVR system, including its website and VR viewer, and explains how to create a simple VR scene quickly. The subsequent main chapters will delve deeper, particularly into the creation of VR scenes for architecture.

2.1 ArchiVR Website

If you go to archi-vr.ti.bfh.ch, you will see at the time of writing (Jan. 2026) the following:



Without logging in, the most popular public scene will be listed at the top.

Currently, you can access the ArchiVR website only from within the BFH network.

2.1.1 Register & Profile

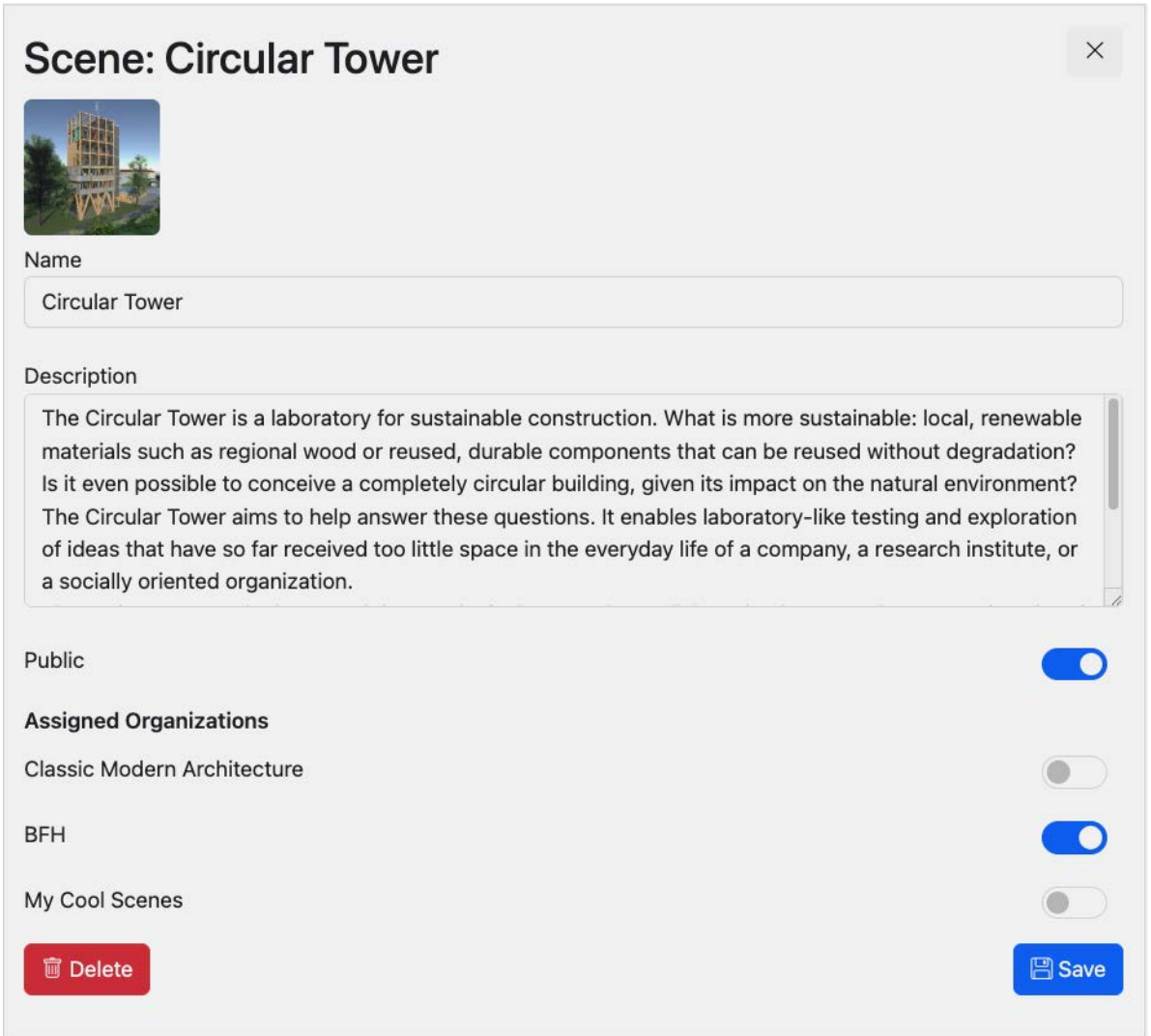
- **To register,** press *Login* in the top-right, then select *Register*. The key is your email address. The new account will be activated once the confirmation email has been answered.
- **To deactivate** your user account, press the *Deactivate Account* button. Your scenes will not be deleted, and you will not be contacted again. You can still log in and reactivate your account later.
- **To delete** your user account, press *Delete Account*. All your scenes and account data will be deleted and cannot be recovered.

2.1.2 Editing Scene Descriptions

- When you created a scene as shown in 2.3 ArchiVR Scene Creation, you can edit its description, image, public visibility, and group assignment by clicking on the *Edit*

button:

- If a scene is neither set to be publicly visible nor assigned to an organization, only the user who created it can view it in the VR viewer.
- When a scene is assigned to an organization, all users of that organization can view it.



2.2 ArchiVR Viewer

The *ArchiVR Viewer* is a software that runs inside a VR headset. For the following explanations, you need a *Meta Quest 2 or 3*. More on this and other headsets can be read in 9.1.2 Meta Quest.

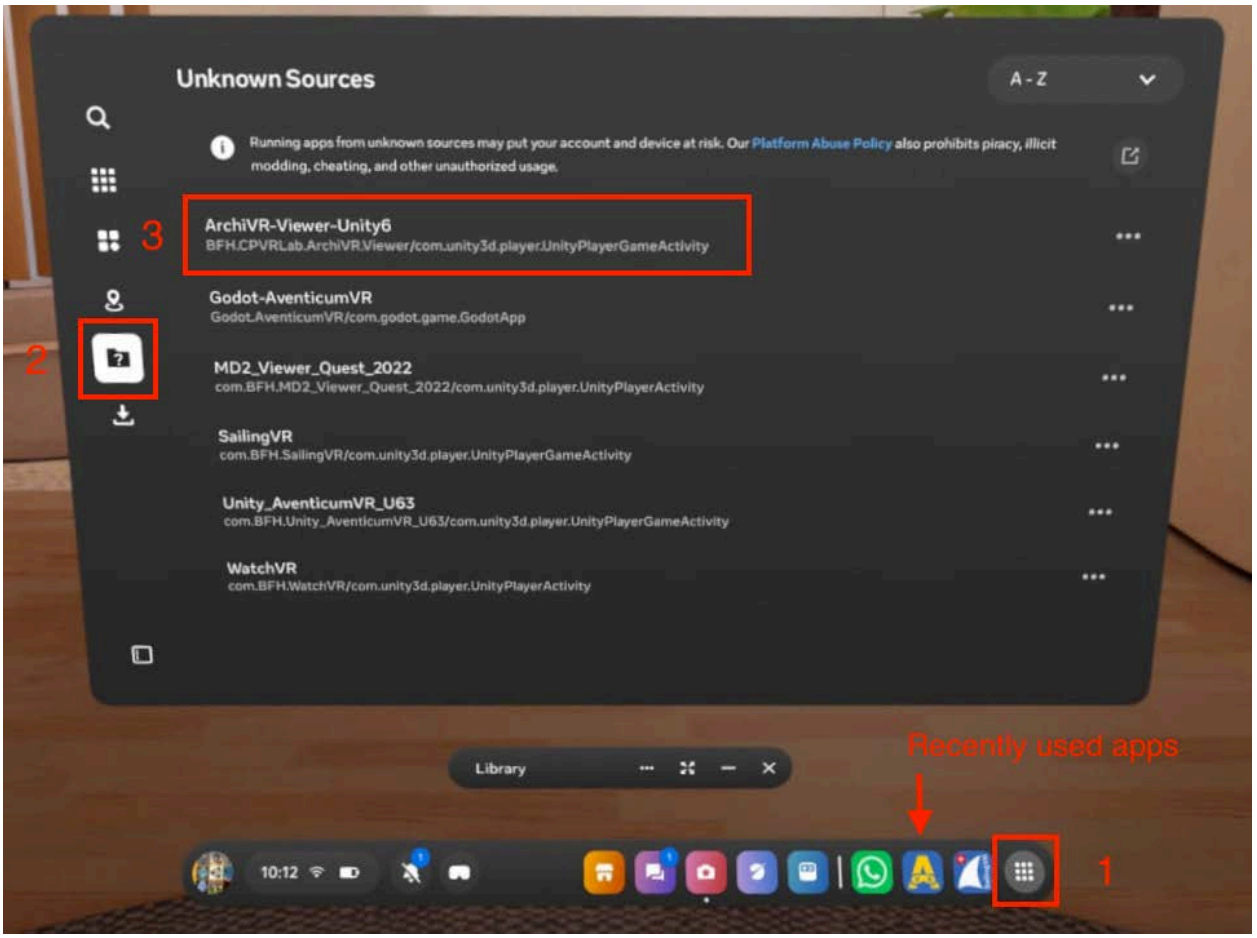
2.2.1 Install the ArchiVR Viewer on a Quest VR Headset

We can install the *ArchiVR Viewer* software either by building it directly on your laptop or by installing an APK (Android Package) with SideQuest. See the viewer repository README for instructions on building the viewer with Unity 6.

To install the APK, download the latest version from our package folder to your computer, then follow the instructions in 9.4 Installing an APK without Unity.

2.2.2 Launching the ArchiVR Viewer

- **Put on the VR headset**: If you have built or installed the *ArchiVR Viewer*, you can unplug the USB-C cable and put on the VR headset. If you have never used the Meta Quest headset, please complete the tutorials in 9.1.2.2 First Steps in Oculus Quest.
- **Find the ArchiVR Viewer**: In the entry room in VR or with the video passthrough, open the main menu with a pinch gesture (index and thumb touching) with your right hand, palm facing you. The main window is a dark gray bar with multiple buttons.
  - 1) Open the app window on the button with the nine dots to the right.
  - 2) In the app window on the left, select the folder with the question mark icon.
  - 3) In this list of manually installed apps, click on the ArchiVR-Viewer-Unity* app.
  - The most recently used apps are just to the left of the nine-dot button.

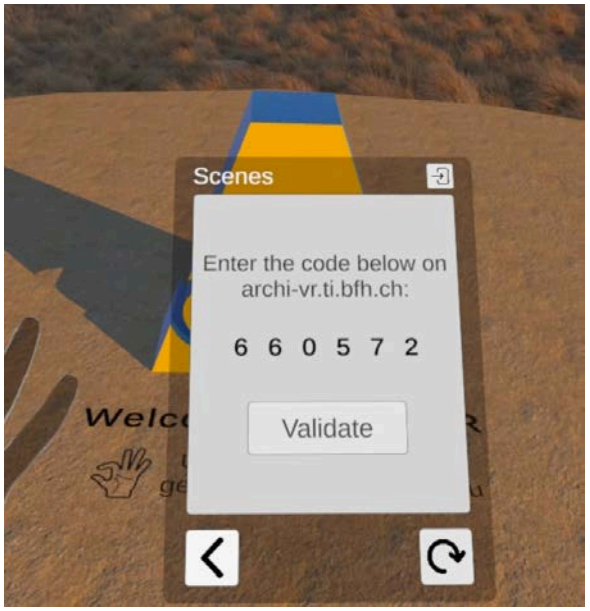


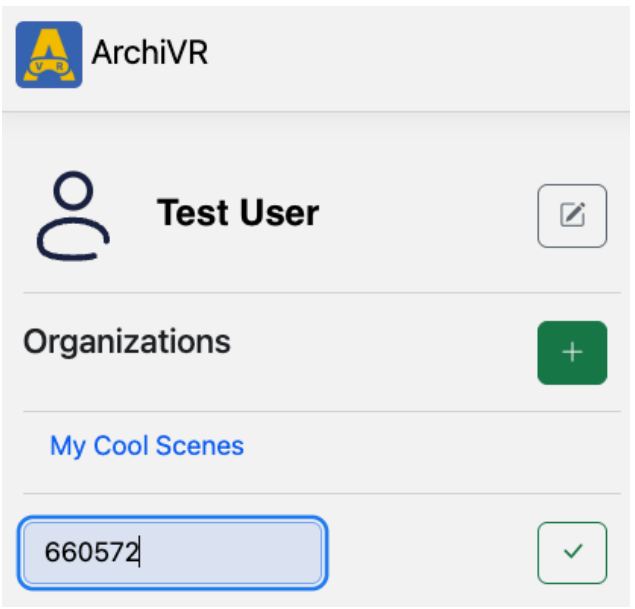
2.2.2 Connect to your ArchiVR Account

After launching the ArchiVR Viewer, you will start in the *Home Scene*, which displays a desert environment image. The only thing you can do is the left-hand-towards-face pinch gesture to open the ArchiVR menu:

Because you are not logged in to the ArchiVR server the bottom-center button with the ArchiVR logo is black. If you click it, you will get a 6-digit number that you need to enter on the website:

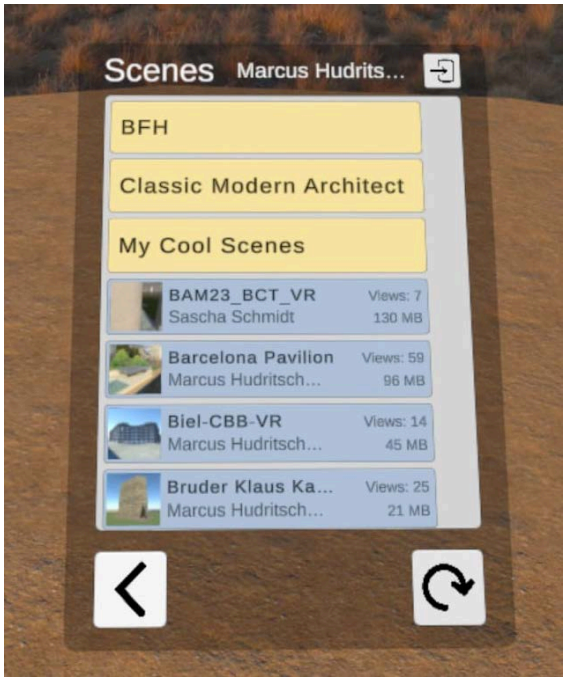
- Remember the 6 digits, then remove the headset.
- On your computer, open a web browser, go to archi-vr.ti.bfh.ch, and log in.
- When logged in, enter the 6 digits at the bottom of the left panel. Like this, you don't have to enter a complicated password in the VR headset.
- Put the VR headset back on and press the *Validate* button. You are now connected to your account and remain so for multiple days.

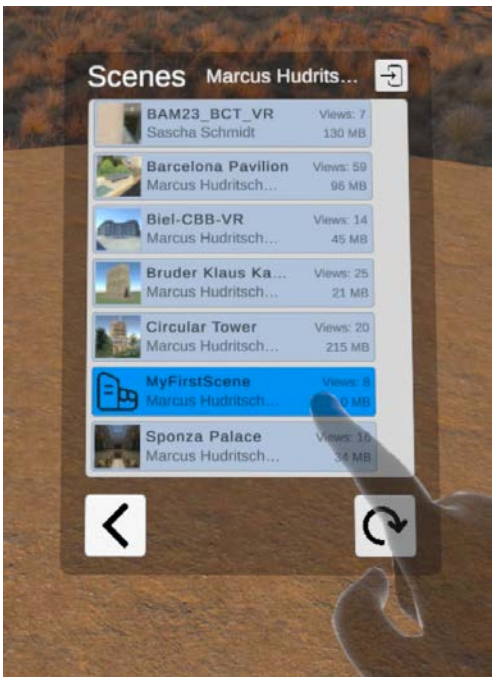




2.2.3 Load a Scene

After pressing again the ArchiVR button, you will see a list of yellow organization buttons that you are assigned to, as well as public and your own private scene. You can scroll the list on the right border. Click on the scene that you want to enter:

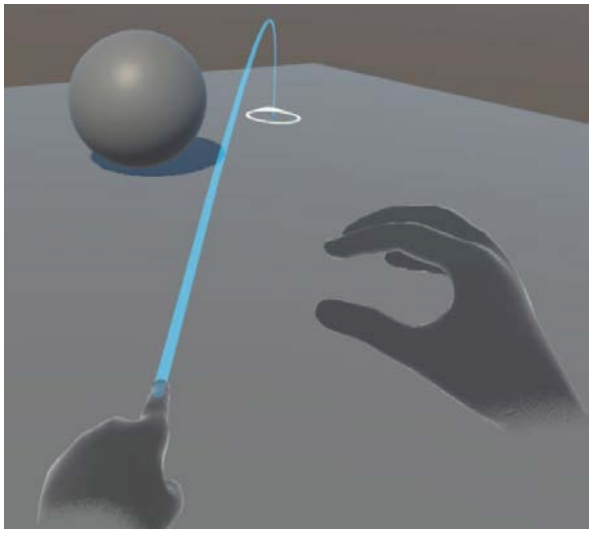




2.2.4 Navigation in a Scene

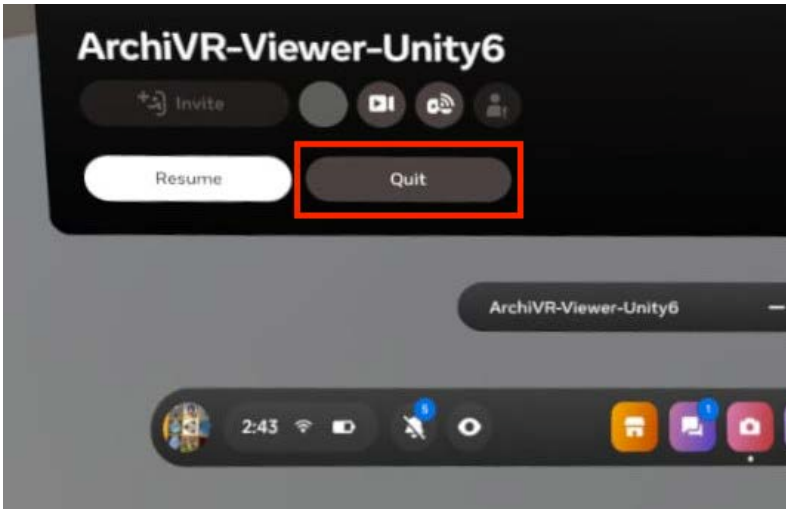
After downloading the scene, you will enter it at the starting point [0,0,0]. You can now walk freely when your physical environment allows it. See the Meta video on setting up a boundary.

If you want to navigate a larger distance, you can do a so-called ***teleportation***: Either with your straight left or straight right index finger, you can aim with a parabolic curve to a target position. If the target position allows teleportation, the curve will be light blue; otherwise, it will be red. With the other hand, you have to execute the teleportation with a pinch gesture (= quick snapping of index and thumb):



2.2.5 Exiting a Scene or the ArchiVR-Viewer

Within the ArchiVR Viewer, you can always open its menu with the left-hand-towards-face pinch gesture. To exit the ArchiVR Viewer, press the power button at the bottom right. Alternatively, you can open the system menu in all VR apps with a right-hand-towards-face pinch gesture and press the Quit button:



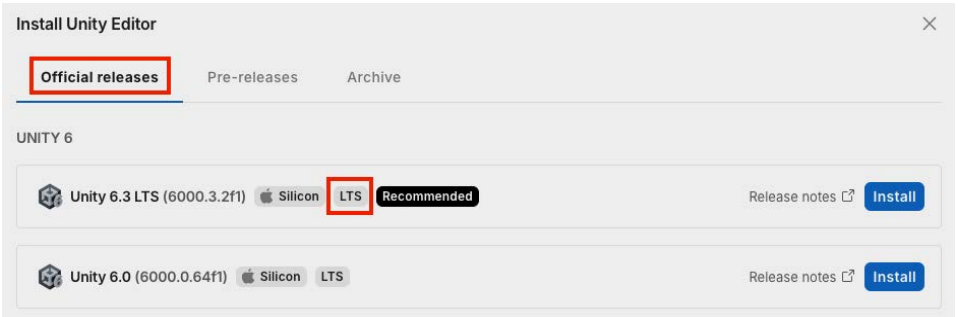
2.3 ArchiVR Scene Creation

The following steps are kept to a minimum for a quick introduction. You will get a deeper introduction to Unity in 3 Quick Introduction to Unity and a deeper tutorial in 5 Adapt your Project for ArchiVR.

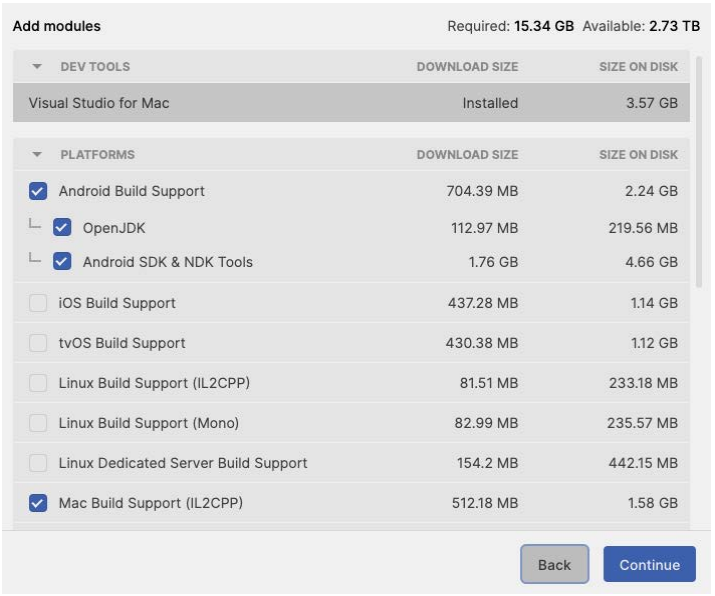
2.3.1 Install Unity

Download and install the latest *Unity Hub* from Unity's website. *Unity Hub* is an installation and project management application because Unity is continually improving and supports multiple versions. You can install the free *Unity Personal* Edition.

- **Create a Unity Account**: Create a Unity account on the following website: https://id.unity.com.
- **Download Unity Hub**: This is the download app that also manages Unity versions and projects.
- **Open Unity Hub and log in** with your Unity ID on the top left.
- **Install the Unity Editor**: In Unity Hub, select *Installs* and click *Install Editor* from the *Official releases*, and install the latest *LTS (Long Term Support)* version of **Unity 6**:



- After the *Install,* select *Android Build Support* with *OpenJDK* and *Android SDK & NDK Tools*. This is needed for the Oculus Quest, which is an Android device:



- Please do NOT uncheck the installation of the *Visual Studio Code* (at the top of the *Dev Tool* section).
- Scroll down and select also the … *Build Support (IL2CPP)* for your platform (*macOS*, *Windows*, or *Linux*)
- Click *Continue* to install.
- If the installation hangs (e.g., on macOS), a background dialog may prompt you to grant file-writing privileges.
- Depending on your internet connection speed, the download and installation may take some time.

2.3.2 Clone & open the archivr-editor-unity6 Project

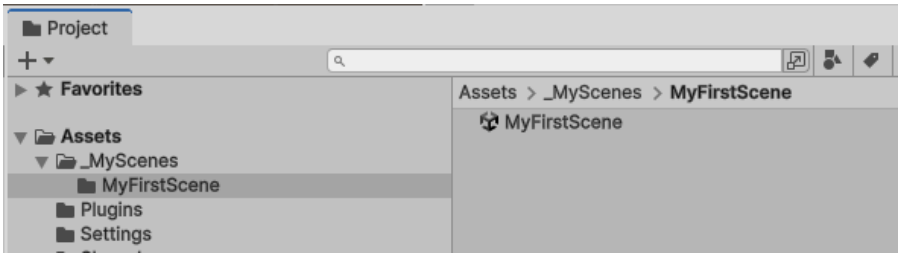
This is a Unity project hosted in a Git repository at https://codeberg.org/ArchiVR/archivr-editor-unity6. Git is a version control system for source code.

- **Clone the GIT Repository**:
  - You need to have GIT installed on your system.
  - Open a console and navigate to a local folder where you want to store it:
    - Don’t store the project in a OneDrive or iCloud synchronized folder.
    - Ensure the full path is no longer than 256 characters (on Windows).
    - Ensure the path doesn’t contain non-ASCII characters (äöü, etc.).
  - Clone the project with:
    ```
    git clone --recurse-submodules https://codeberg.org/ArchiVR/archivr-editor-unity6.git
    ```

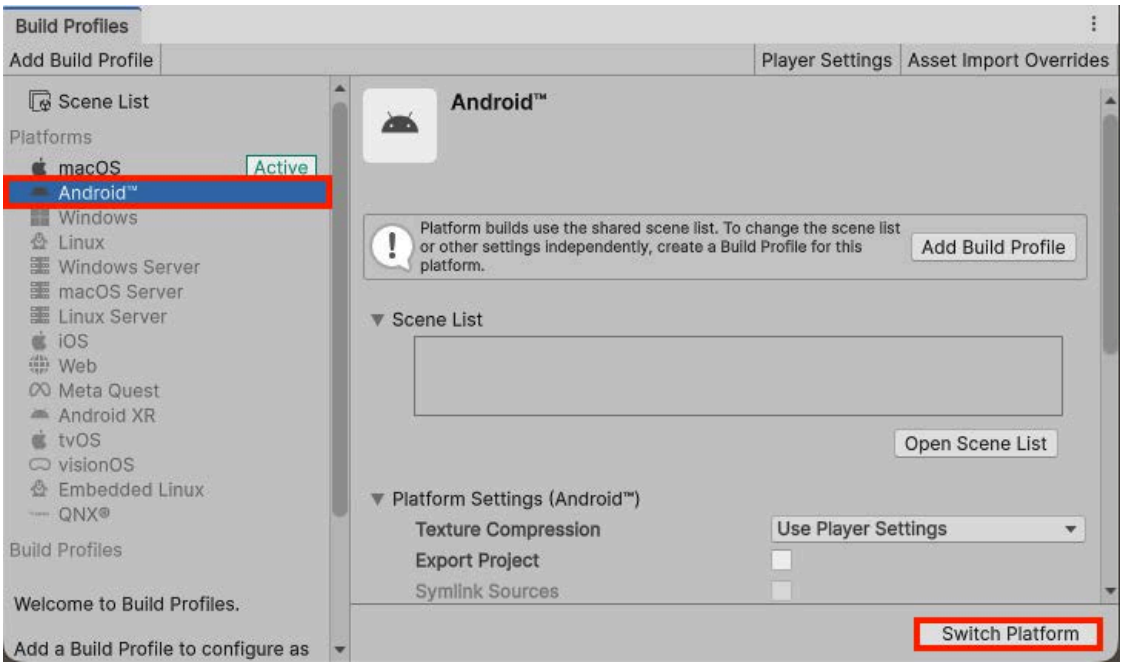
- **Open Unity Hub**, then add the project by clicking the small triangle to the right of the Open button (top right).

- **Open *Unity*** by clicking on the added project *archivr-editor-unity6* in the *Unity Hub*.

- **Save the Scene**: Unity starts with an empty default scene, which we first save. When you select the menu *File > Save*, a scene-saving dialog opens.
  - Create a first subfolder named _MyScenes by clicking New Folder.
  - Create again a subfolder named *MyFirstScene*.
  - Give the scene the name *MyFirstScene* and click *Save*.
  - In the *Project* window, you will see:



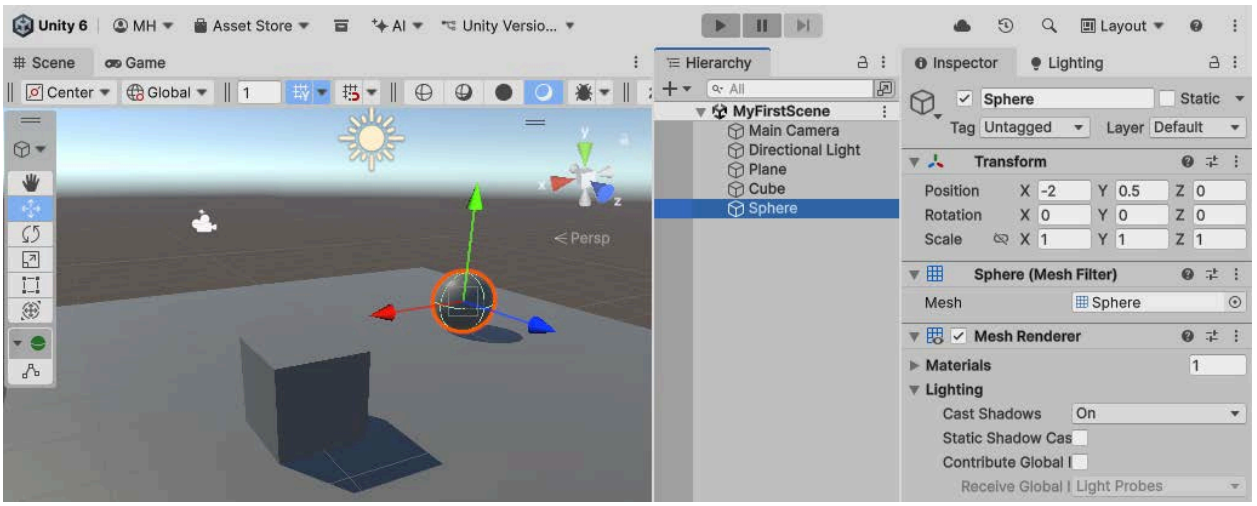
- **Switch to Android**: Go to *File > Build Profiles*:
  - Select *Android* from the platform list to the left.
  - Click on the *Switch Platform* button on the top-right. We do that because the Meta Quest headsets are Android devices:



2.3.3 Create some 3D Objects

We will now add some simple 3D objects so that our first scene is not empty:

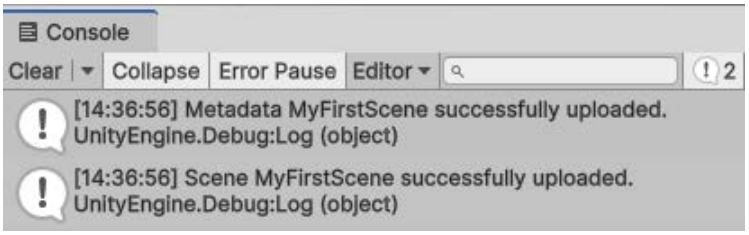
- Add a plane by the menu *GameObject > 3D Object > Plane*.
  - In the *Inspector* window, set the plane's Layer to *Teleport*
- Add a cube by the menu *GameObject > 3D Object > Cube*.
  - In the *Inspector* window, set the cube's position to [2.0, 0.5, 0.0].
- Add a sphere by the menu *GameObject > 3D Object > Sphere*.
  - In the *Inspector* window, adjust the sphere's position to [-2.0, 0.5, 0.0].
- Press CTRL-S (Command-S on macOS) to save the scene.



Our simple scene with the main camera, a directional light, our added cube, and our added and selected sphere.

2.3.4 Build & Upload to ArchiVR

- Open the *ArchiVR* window with the menu *Tools* > *ArchiVR*.
- Enter your *ArchiVR* email address and password, then click *Login*.
- All scenes in your project are listed. Select the one that you want to build and upload.
- Press *Build & Upload*
- Check the *Console* window for the success message: "Scene MyFirstScene successfully uploaded."



If you go now to https://archi-vr.ti.bfh.ch/ and you log in with the same credentials, you will see *MyFristScene* listed among your scenes:

Click on edit on the top-right to add a scene image and a description. All new scenes are by default not public and not assigned to any group. Only you can view them in the ArchiVR Viewer App.

3 Quick Introduction to Unity

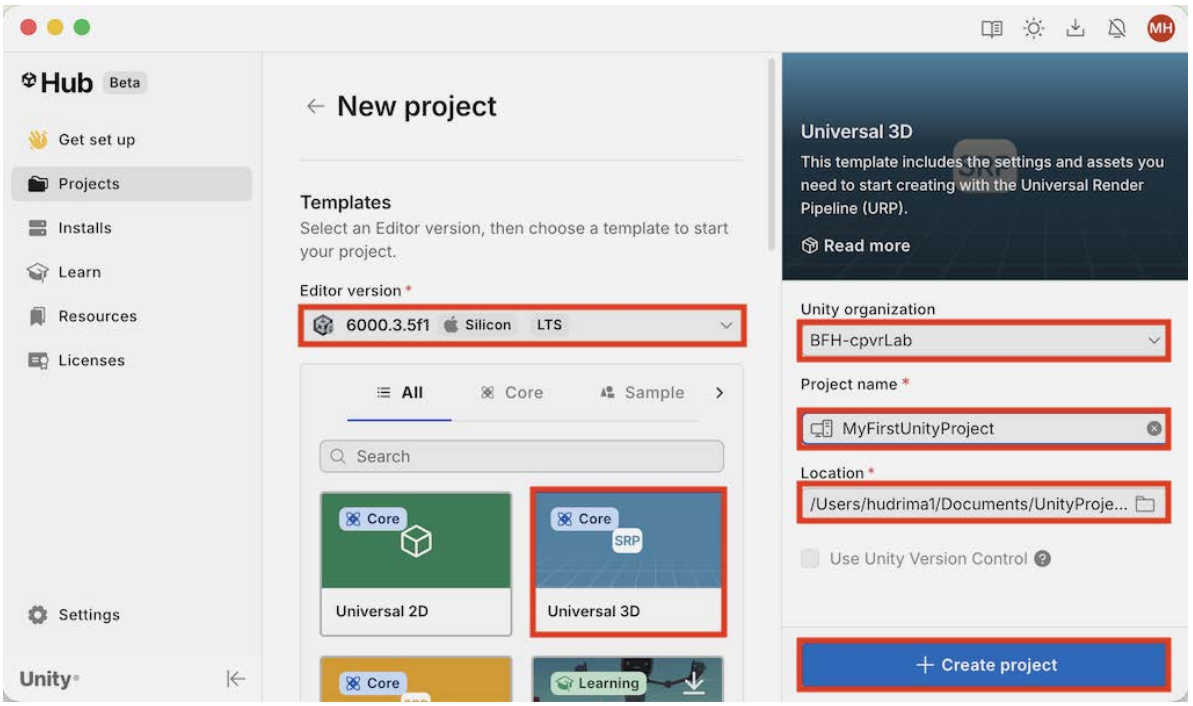
This chapter will give an introduction to the game engine Unity. VR and AR applications are almost exclusively built from game engines such as Unity or Unreal Engine. The primary goal of a 3D engine is to display 3D content in real-time, which means that the scene gets rendered at a minimum of 30 frames per second (FPS), so that you can interact with the scene. VR and MR applications are even more demanding because they must create two images, one for the left and one for the right eye, at an even higher frame rate.

Why did we choose Unity?

- **Free:** The Personal Edition of Unity for students and freelancers with less than 200k USD turnover.
- **It is the most popular game engine for beginners**, with extensive resources and a large community.
- **The most used game engine is for mobile platform deployments,** such as Apple iOS, Google Android, and Microsoft HoloLens. The Oculus Quest VR headset is also a mobile device running Android.
- **Works on Windows, macOS, and Linux**.
- **Usable for the development of laptop computers**. 3D Game Engines are among the most demanding applications you will use on a computer.

3.1 Create a new Unity Project

- **Install Unity Hub and the latest LTS version of Unity.**
  See 2.3.1 Install Unity for instructions on installation.
- **Open Unity Hub and create a Unity Project**: Select *Projects* on the left side and click *New Project* in the top-right corner. Choose the ***Universal 3D Core*** template and set the *Project name* to *MyFirstUnityProject*. Choose the folder where your project will be created on your file system. Untick the *Connect to Unity Cloud* Option. You are not recommended to store your project in a cloud-synced folder, such as Microsoft OneDrive or Apple iCloud. **Achtung: Since Windows 11, the default Documents folder has been mapped to a OneDrive folder. Don't save Unity projects there, because OneDrive doesn't handle them well.**
  Store instead the project on e.g. C:/Users/{youUserName}/UnityProjects.
  On Windows, make also sure that your file path is at most 256 characters.

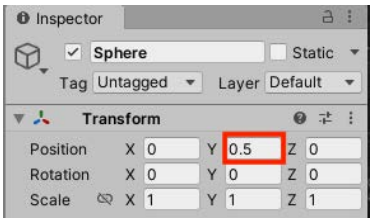


Your project will be created and opened in Unity.

3.2 Unity's Main User Interface

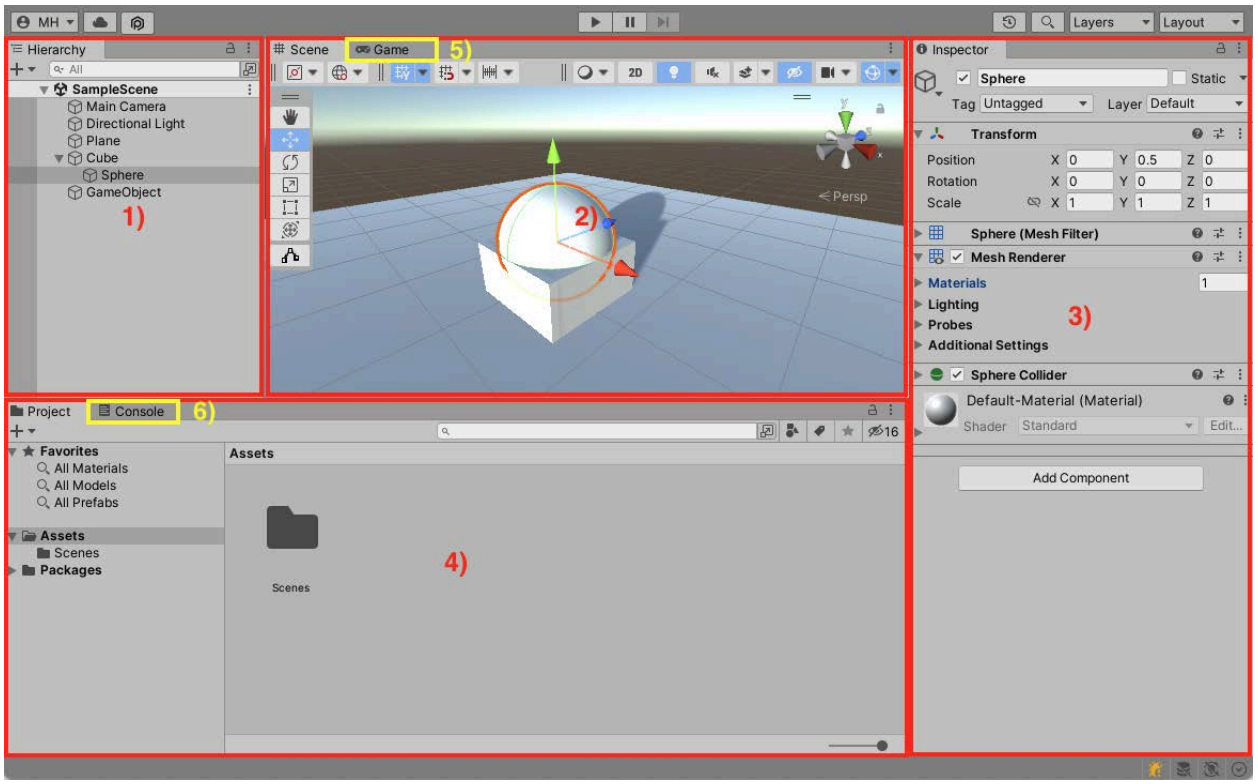
Add Game Objects: The Unity scene will contain only the *Main Camera* and the *Directional Light*. The *game object* (GO, for short) is the central entity type with a few default components and supports adding more components. The camera and the light are invisible GOs in 3D and are represented by a symbol (aka "gizmo"). To see some 3D objects, you can:

- **Add a plane** by the *menu* > *GameObject* > *3D Object* > *Plane*.
- **Add a cube** by the *menu* > *GameObject* > *3D Object* > *Cube*.
- **Add a sphere** by clicking on the cube in the *Hierarchy* window and pressing the right mouse button (RMB) and selecting *3D Object* > *Sphere*.
    - The sphere is invisible because it is inside the cube.
    - To move it up, type in the *Inspector* window 0.5 in the Y field of the *Transform's Position*:



- **Press Ctrl-S** (Cmd-S on Mac) to save the scene.

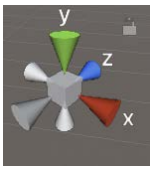
The standard Unity User Interface (UI) includes multiple sub-windows that can be freely resized, docked, undocked, stacked on top of one another (i.e., tabbed), or closed. You can save or restore a layout using the *Window* > *Layouts* menu. The default layout contains the following windows:



The default window layout of Unity with the sections that are explained in the following. As with other 3D applications, you will profit from a bigger monitor because many more sub-windows can be opened. Sometimes it is also advantageous to have the Scene and the Game window side by side. All windows can be opened from the Window menu.

1) ***Hierarchy***: Shows a hierarchical list of all GOs in the scene with their names.
   - You can add GOs by clicking on the right mouse button (RMB) > *3D Object*.
   - You can add a GO to a selected GO in the same way.
   - You can change the hierarchy by just selecting, dragging & dropping GOs.
   - You can also add an empty GO with RMB > *Create Empty* that acts as a parent for one or more child GOs. In our scene, the sphere is a child of the cube GO. If you transform a parent GO, all children will transform accordingly.

2) ***Scene***: Shows the scene in 3D and acts as the primary 3D editor.
   You can change the view of the scene in many ways:
   - Move the view with the middle mouse button (MMB).
   - Rotate the view with the Alt-left mouse button (Alt-LMB).
   - Zoom the view with the mouse wheel or shift-right mouse button (Shift-RMB).
   - Set Perspective or Isometric Projection by clicking on the center cube of the *axis gizmo* in the top-right corner of the *Scene* window.
   - View along an axis by clicking on one of the axis cones of the *axis gizmo*.
   - Center the view on the selected GO by pressing the key F.
   - Navigate with the WASD(QE) keys: By pressing the RMB, you can fly through the scene like in the game. The mouse gives the directions and the WASD keys move you forward, left, backward, or to the right.



   You can transform one or multiple selected GOs as follows:
   - Select one GO with the LMB. The selected GO is surrounded by orange.
   - Select multiple GO with Shift-LMB.
   - Move a GO with the *Move* tool (key Q) and the LMB along an axis or plane.
   - Rotate a GO with the *Rotate* tool (key W) and the LMB on an axis circle.
   - Scale a GO with the *Scale* tool (key E) and the LMB on an axis cube.

- 2D Move & Scale a GO with the Rect tool (key R).
- The Transform Tool shows the *Move*, *Rotate,* and *Scale* gizmos all at once.
- Snap-on grid transform (move, rotate & scale) by pressing Ctrl (Cmd on Mac).
- Undo any action by pressing Ctrl-Z (Cmd-Z on Mac).

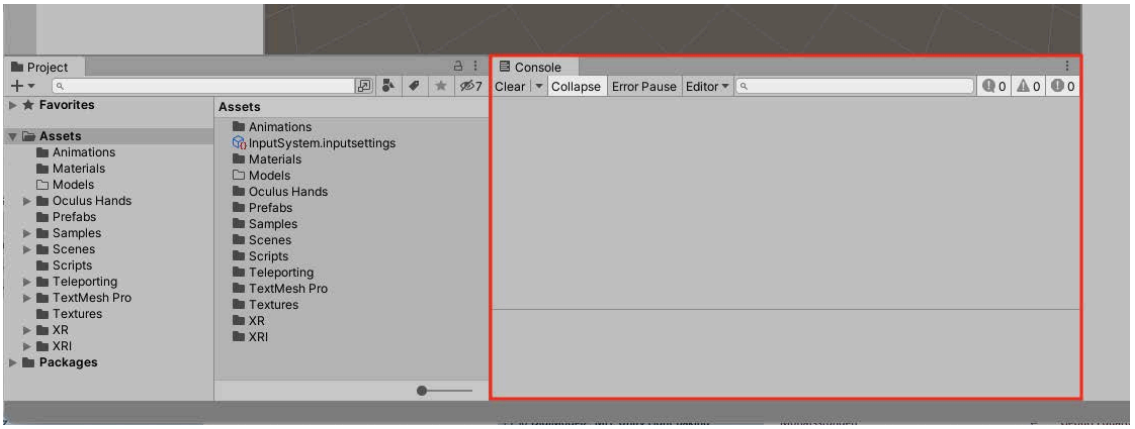
Most of the above interactions can also be activated with the buttons in the scene editor window. When hovering over a button, a tooltip will appear.

3) ***Inspector***: Shows all the components of a selected GO. All components can be expanded to show all their properties, or collapsed to fit on a single line with a small triangle at the top-left. All GOs will have at least the *Transform* component, which displays the numerical parameters for their *Position*, *Rotation*, and *Scale*. All visible 3D GOs will also include a *Mesh Filter*, a *Mesh Renderer,* and a *Material* component. We can add many more components to GOs. Game development is, in essence, adding programming script components to GOs!

4) ***Project***: Displays all project files in the project's Assets folder.
   - You can create subfolders to keep your files in an organized order.
   - Unity is monitoring the project folder constantly. Adding, moving, or deleting files on the filesystem or within Unity will be recognized and appropriate actions will be processed.
   - With the slider on the bottom-right, you change the appearance of the files.

5) ***Game***: By pressing the *Play* button in the top-center of the main window, you can start a game. This will cause the *Game* window tab (e) to pop to the front. Without any further programming, nothing more will happen than what the *Main Camera* sees from its position. You can stop playing by pressing the *Play* button again.

6) ***Console***: Shows all errors, warnings, or log messages. In the status bar at the bottom of the window, you will only see the last message. It is therefore recommended to move the Console window to a location where it is always visible.



No Mesh Editing out of the box: You may wonder how to edit a 3D object. You can't do that in Unity out of the box. You can only transform a 3D object and the underlying 3D triangle mesh. Unity supports only a few very basic 3D shapes, such as planes, cubes, spheres, cylinders, capsules, and quads. Everything else has to be imported in specific 3D file formats.

Unity is a Game Engine for Game Development

We are using Unity to create a VR app with predefined functionality. It has been developed by the cpvrLab with some scripts in a programming language. It is not the goal of this course to teach you programming, but it's worthwhile to have a rough understanding of what the basic principles are behind it. In 9.7 Introduction to Unity Scripting, we show you how you can create a simple game.

4 Preparations in CAD Systems

No architects design their projects in game engines. They use dedicated *Computer-Aided Architectural Design* (CAAD) software tools, such as ArchiCAD, Vectorworks, Allplan, or Rhino. These software tools are not yet able to create applications for VR headsets and will likely never be able to do so. The reason is pretty simple: A VR application is itself a very sophisticated piece of software, highly optimized for visualization performance. We will provide additional insights into this field of VR rendering in the VR chapter of the appendix.

But most CAD systems can export specific file formats that allow exchanging a 3D project with another tool, and that's what we are going to do in this chapter.

Major Condition: 3D!

We will experience our design in the VR app, our design in 3D on a scale of 1:1. The major condition for your design is that it is built in your CAD system in 3D. Some of the above-mentioned CAD systems also allow the design in 2D, which is obviously not enough if you want to experience our designs in space.

4.1 Demo Projects

We provide you with various demo projects that can be downloaded from the ArchiVR website.

4.1.1 Bruder Klaus Kapelle

The Brother Klaus Field Chapel is a privately donated Roman Catholic chapel, built between 2005 and 2007, located above the village of Mechernich-Wachendorf in the Eifel region of Germany.

Aerial view from the Bruder Klaus chapel. The chapel was built according to plans by the Swiss architect Peter Zumthor. For the construction, a tent-shaped construction made of 112 spruce trunks was first erected. The chapel's body was made of tamped concrete around this inner structure, which was laid in 50 cm layers to a height of 12 meters in 24 days by a volunteer tamping team, together with specialist craftsmen, according to the region's traditional craftsmanship. A mott fire was maintained inside for three weeks in the fall of 2006, which charred the tree trunks and detached them from the concrete, making them easy to remove [Source: Wikipedia].

4.1.2 Barcelona Pavilion

The *Barcelona Pavilion* has been chosen as a demo project because it is relatively small and has a very distinct materialization. The Barcelona Pavilion, designed by Ludwig Mies van der Rohe and Lilly Reich, was the German Pavilion for the 1929 World Exposition in Barcelona. The original pavilion was demolished after the exhibition, but in the 1980s a group of Catalan architects reconstructed it from the original plans at its original location. You can see various good documentaries on YouTube about the Barcelona Pavilion.

The reconstructed Barcelona Pavilion today (Image: Wikipedia). Even if the ceilings and walls appear to be solid, that impression is deceptive. A steel framework supports all walls, ceilings, and even the floor base. All covered stone slabs result in a precise joint pattern from which the structure can be read.

4.1.3 Sponza Atrium

Indirect lighting in architecture is very important, but unfortunately, it is also the most difficult aspect to simulate correctly and fast in computer graphics. In Chapter 5.3.2 Indirect Lighting in Unity, we will use the Sponza Atrium test scene. It is a well-known benchmark for indirect lighting (aka global illumination) in computer graphics because the atrium gets only light from the top and rarely direct sunlight in the summertime.

The Sponza atrium is within the Sponza Palace, a 16th-century palace in Dubrovnik, Croatia. Frank Meinl initially designed this model for the German game engine company Crytek. It is reasonably low-poly and runs in real time on laptops.

4.2 Import / Export Frustration

Exchanging 3D models between different modeling and rendering software tools has always been problematic. No open 3D file format is fully supported by all tools. Most 3D applications support only a few exchange formats (e.g., FBX, OBJ, DAE, GLTF). For the import/export between the tools we use (*Rhino*, *Blender,* and *Unity*), **we recommend the FBX file format**.

One-Way Workflow: These tools are highly complex, so it is impossible to exchange projects without losing information. Most importantly, all object-type information from CAD systems gets lost because rendering applications don't know what a wall or a window is. They treat every object as a triangular mesh. But material and lighting settings are also very complex and are rarely transferred in full.

Consequently, you will continue to design in your favorite CAD tool and use a dedicated, either offline or real-time, rendering system solely for visualization. If you modify your project in a visualization tool, e.g., materials or lighting, you cannot bring those changes back into your CAD tool.

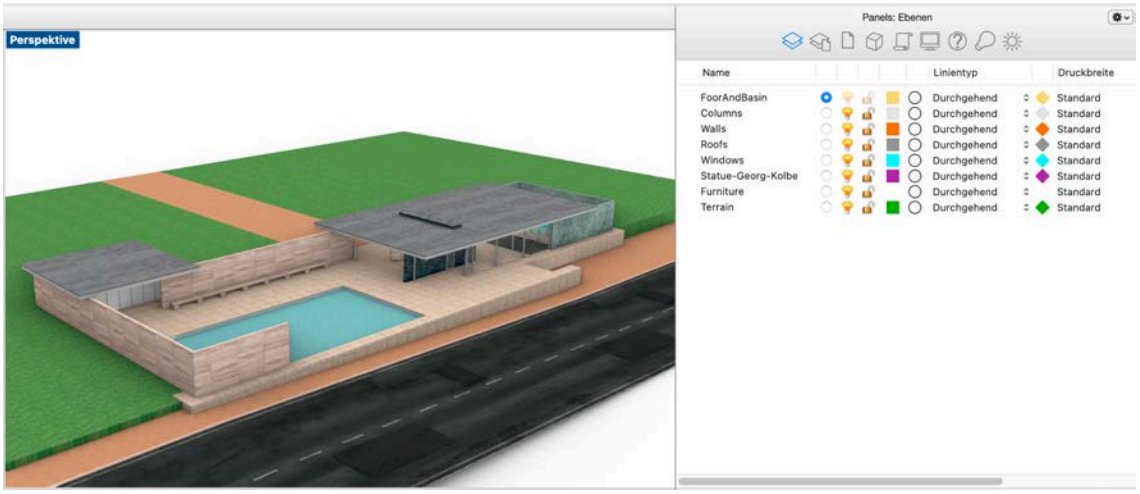
For this chapter, we will use the 3D model of the Barcelona Pavilion. The original 3D model (*BarcelonaPavilion.blend*) was built in Blender using plans, images, and videos sourced online. The Rhino version (*BarcelonaPavilion.3dm*) was created with an FBX export from Blender and imported into Rhino.

4.2.1 Preparations and Export in Rhino

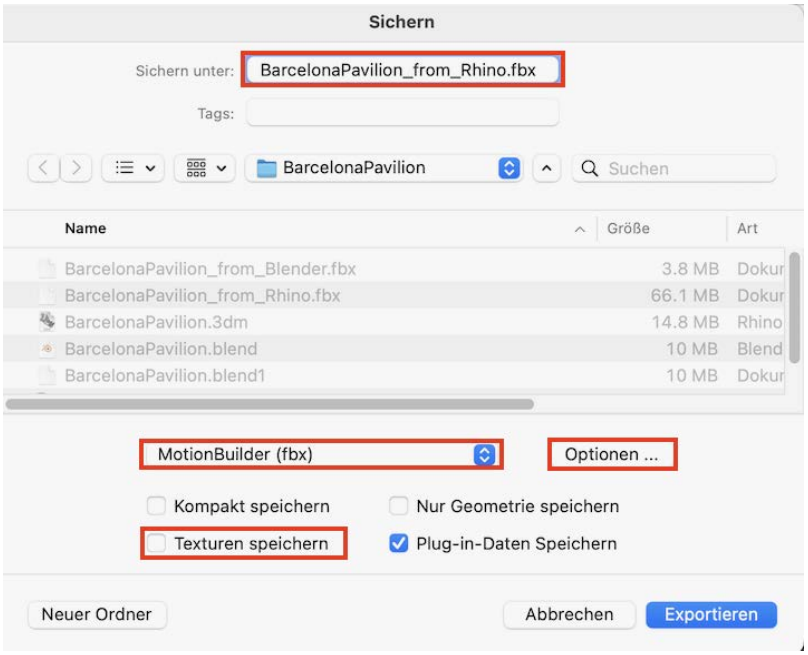
Rhino is commercial CAD software developed by Robert McNeel & Associates, an American, privately held, employee-owned company founded in 1969. Rhino geometries are based on the NURBS mathematical model, which produces mathematically precise representations of curves and freeform surfaces in computer graphics (as opposed to polygon-mesh-based applications).

Rhino supports various export formats that Unity would understand (FBX, DirectX, and OBJ). The following explanations focus on the FBX format:

- **Open the Rhino version of the project in Rhino** (*BarcelonaPavilion.3dm*):
  - Objects are grouped in layers that correspond to the primary structural elements.
  - All objects are open or closed polygon networks because the project was imported with FBX from Blender.
  - The scene is only lit by the standard Rhino skylight.

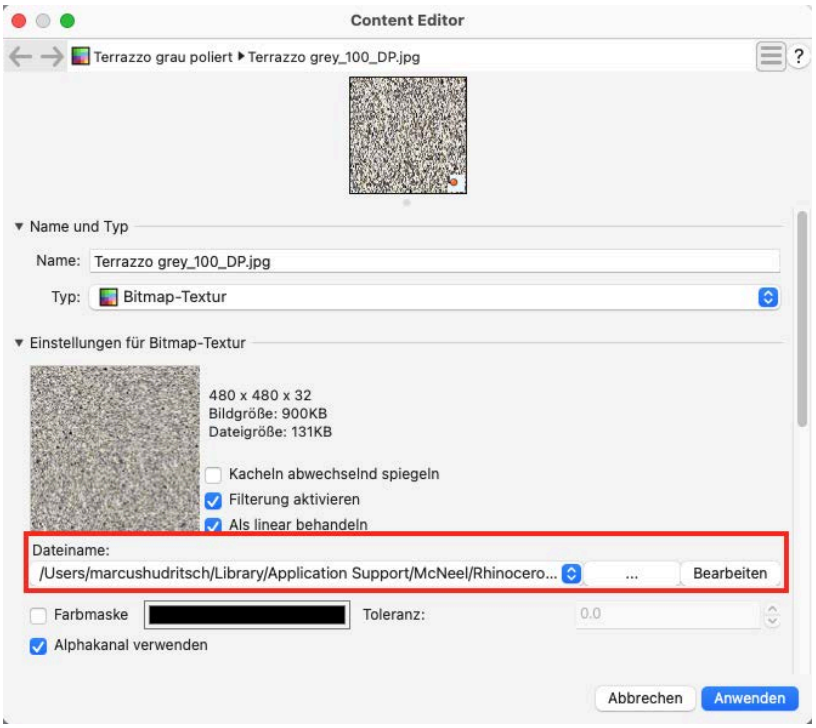


- **Export the Rhino project as an FBX file**: Choose *File* > *Export …*:
  - Set the export file type to *MotionBuilder (fbx)*.
  - Set the new file name as *BarcelonaPavilion_from_Rhino.fbx*
  - Select the *Save Textures* option.



Unfortunately, this option is unreliable. If you have chosen materials from Rhino's built-in material library, the textures involved are not exported.
To find the built-in textures, you have to click on the name of the texture to open the texture window:

In the *Content Editor,* you can see where on your system *Rhino* has installed all textures of all materials:



- Click on the *Options …* button and leave all options as follows:

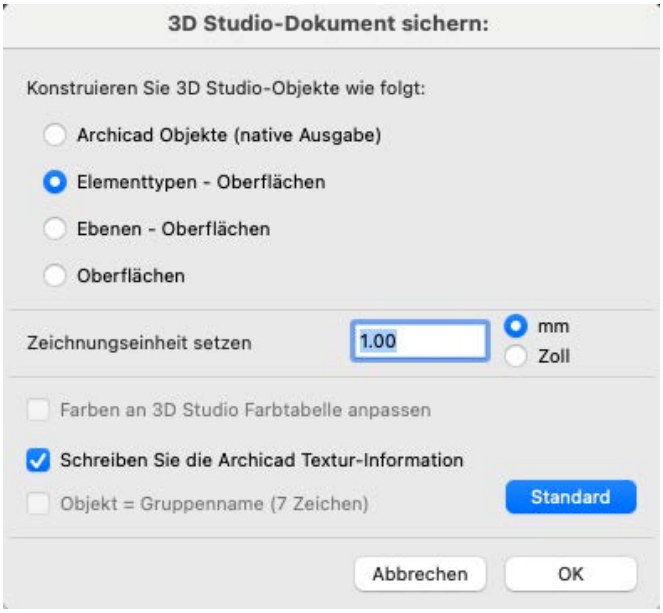
- **Click on *Export***: Finally, a dialog will appears that allows you to control the resolution of the mathematical surfaces. Try the default settings. If you're still seeing very high-resolution objects, you can move the *Adaptive SubD Grad* slider further to the left:

4.2.2 Preparations and Export in ArchiCAD

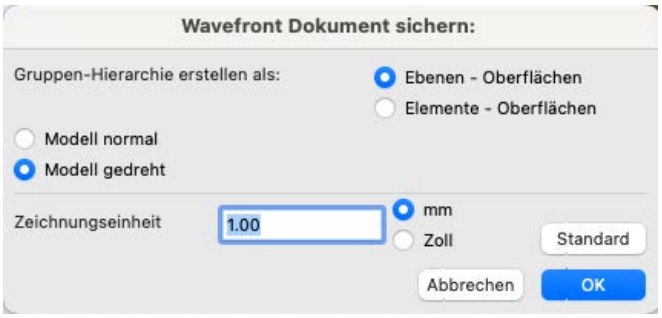
ArchiCAD is a CAD Software founded in 1982 and developed in Budapest. From the beginning, it was a fully 3D planning tool and, therefore, one of the first tools to support the BIM methodology. Since 2022, it has belonged to the Nemetschek Group.

In ArchiCAD, you can export the currently visible objects over the menu *Save as …* with the following formats that can be imported by Unity:

- **Collada (extension DAE)**: The Collada file export has, unfortunately, no structuring options. All Archicad objects appear in increasing numbers.
- **3DStudio (extension 3DS)**: This file export can structure the Archicad objects by various options, as shown in the following screenshot. Unfortunately, the exported objects mostly have increasing object numbers as names.



- **Wavefront (extension OBJ)**: This veteran file format can export Archicad objects structured by *layers* and *materials* or by *elements* and *materials*. Its big advantage is that the chosen export choice concatenates the object names and are not just incrementing numbers.



4.2.4 Preparations and Export in Vectorworks

Vectorworks was founded in 1985 and has been part of the Nemetschek Group since 2000. It was the first architecture CAD system to receive the IFC4 certification and has since supported the Building Information Modelling (BIM) strategy.

Chapter 7 Case Study: Circular Tower details how we handled a bigger Vectorworks project.

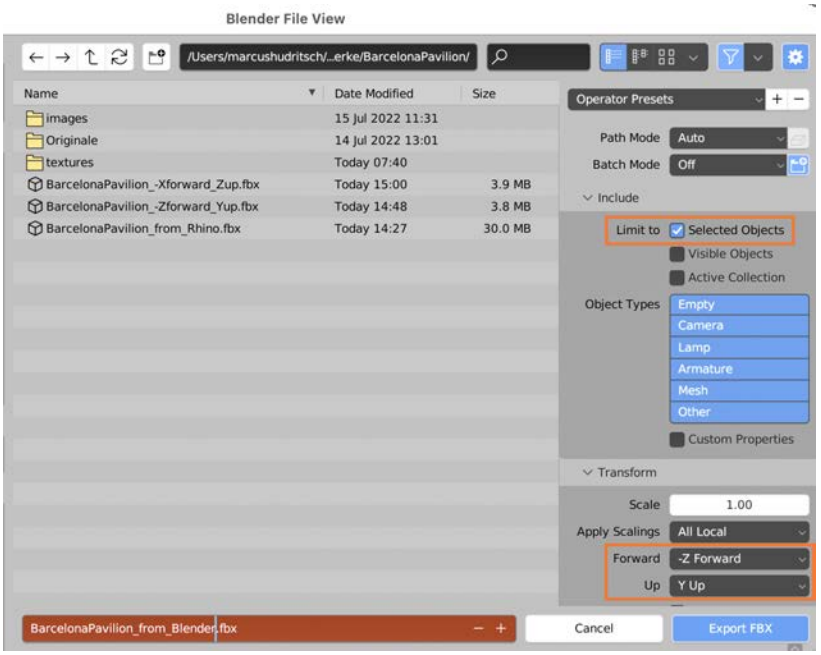
4.2.5 Preparations and Export in Blender

Blender is a free and open-source 3D computer graphics software tool for creating animated films, visual effects, art, 3D-printed models, motion graphics, and interactive 3D applications [Source: Wikipedia].

Blender is not especially suited for the architectural planning process. It is used mainly for visualization with its built-in rendering engine, Cycles. However, many guidelines and add-ons can enhance its capability for architectural modeling.

- **Open the Blender version of the Project in Blender** (*BarcelonaPavilion.blend*):
  - Objects are grouped in empty nodes corresponding to the major structural elements.
  - The scene has a directional light source for the sun.
- **Hide unwanted objects**: Hide the following objects so that they won't be exported: *_Camera*, _Plan, _Section,* and *_Sun* by clicking on the eye icon in the scene hierarchy:

- **Choose File > Export > FBX (*.fbx)**:
  - Set the export file name as *BarcelonPavilion_from_Blender.fbx*.
  - Choose the *Include* option for *Limit to Selected Objects*.
  - Choose the *Transform* option for *Forward* as *-Z Forward* and *Up* as *Y Up*. Unity is a *Y-Up system,* whereas Blender and Rhino are *Z-Up systems*.



- **Click on *Export FBX***.

- **Alternative**: On a system where Blender and Unity are installed, Unity can import a blend file directly by automatically using the Blender FBX export function in the background. Like this, you have no control over the export options.

5 Adapt your Project for ArchiVR

This chapter aims to make it as easy as possible to build your first VR app, so you can start exploring your designs in VR. To achieve this goal, we will provide you with a predefined project with all VR capabilities for the Oculus Quest headset that are already built in.

In Chapter 6, Materials and Lighting in Unity, we will improve key visual aspects for architects.

5.1 Install Unity & Clone ArchiVR-Editor Project

Do as already explained in

2.3.1 Install Unity and

2.3.2 Clone & open the archivr-editor-unity6 Project

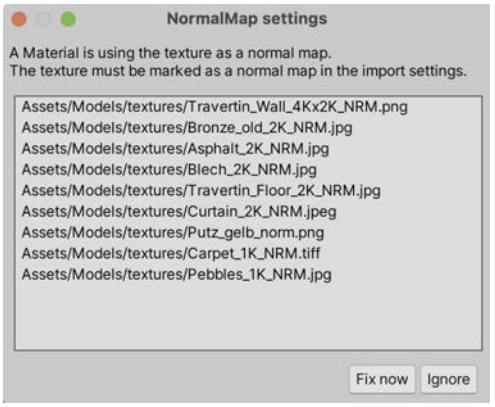
5.2 Prepare your Scene for ArchiVR

Please also do this part in a clean new 3D project from Unity Hub. It would miss our ArchiVR upload functionality. Please use the archivr-editor-unity6 project.

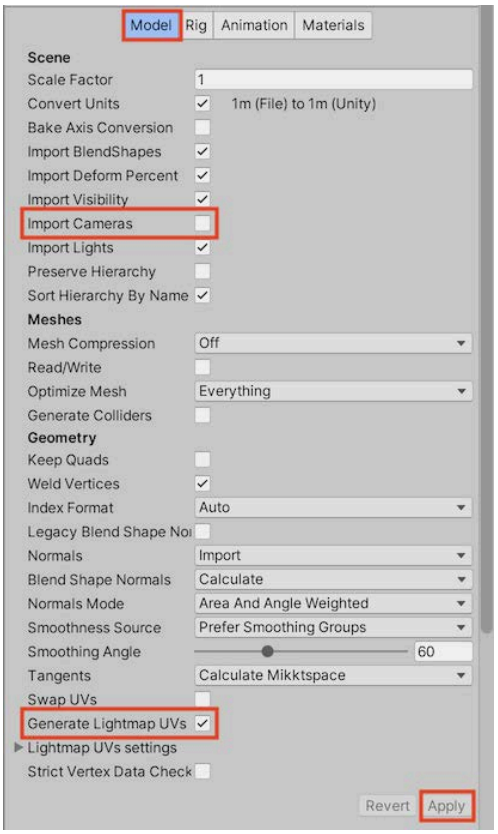
5.2.1 Import your 3D Model

In this part, you will create your scene and import and place your model from whatever CAD system you have used. We will not build it for the VR headset; we will upload it to a remote server. This is much faster and easier, and you can share it with your colleagues.

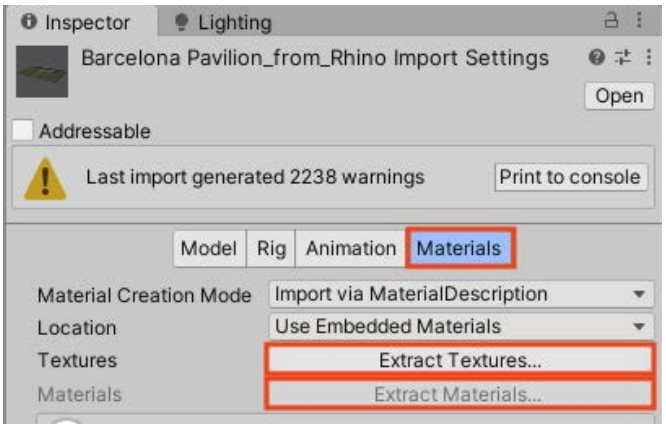
- **Create multiple subfolders** in Unity's *Project* window with *right-click* > *Create* > *Folder*:
  Assets/_MyScenes/
  {yourName}_{yourBFHShortCut}/
  {mySceneName}/
  Models
- **Copy the model files**: In the file browser of your operating system, copy your model file (e.g., the FBX) and the *texture* folder into the *Models* folder. I copied the *BarcelonePavilion_from_Rhino.fbx* and the *texture* folder from the Moodle page. Unity automatically starts importing the files. If the import recognizes normal map textures, click on *Fix now*:



- **Adjust the imported model**: Select the imported model file in the project window and apply the following settings in the *Inspector* window on the *Model* tab:



- **Adjust the imported model materials**: We have to extract the textures and materials on the *Material* tab so that we can modify them in Unity:

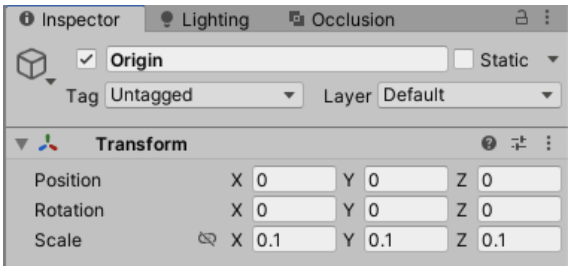


- **Press *Apply*** after the modification at the bottom right of the inspector window.

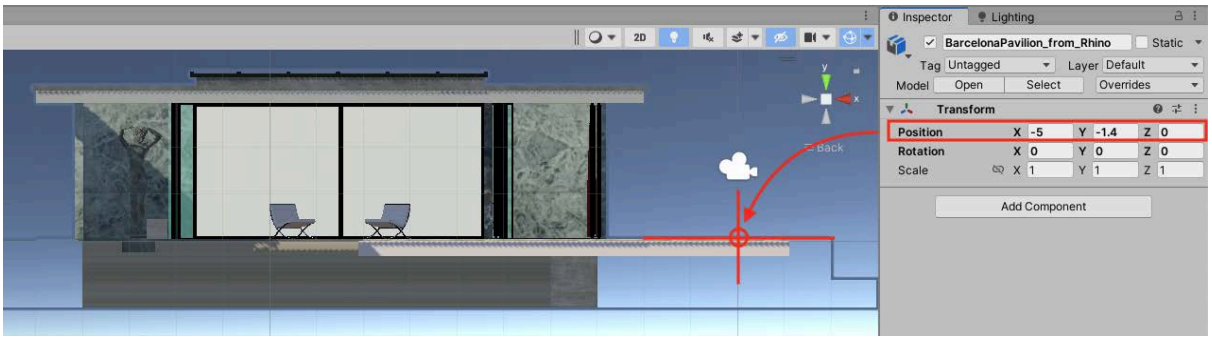
5.2.2 Create a New Scene

- **Create a new Scene**: Click on your *{mySceneName}* folder and create a new scene by *right-clicking> Create > Scene* and name it whatever you want.
- **Open the new Scene** by double-clicking it in the *Project* window.
- **Save your Scene** in your newly created folder with your name.
- **Place a sphere as origin**: Create a sphere with *GameObject > 3D Object > Sphere* and name it *Origin*. Make sure its position is [0,0,0]. Scale it to [0.1,0.1,0.1]. It will be a

visual reference to where the VR experience starts.



- **Place your model** by dragging your imported file from the *Project* window into the *Hierarchy* window.
- **Move your model**: Select it in the *hierarchy* window and change its *position* so that the world-[0,0,0]-point is useful for starting the VR experience. In my case of the Barcelona Pavilion, I have to move the model center to [-5,-1.46,0]:

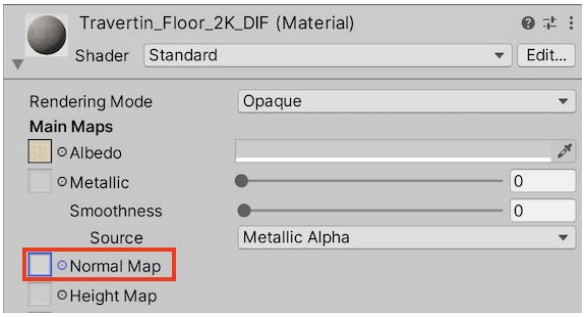


- **Save the scene**: Save the scene by pressing CTRL-S (Command-S on Mac).

5.2.3 Adjust the Materials

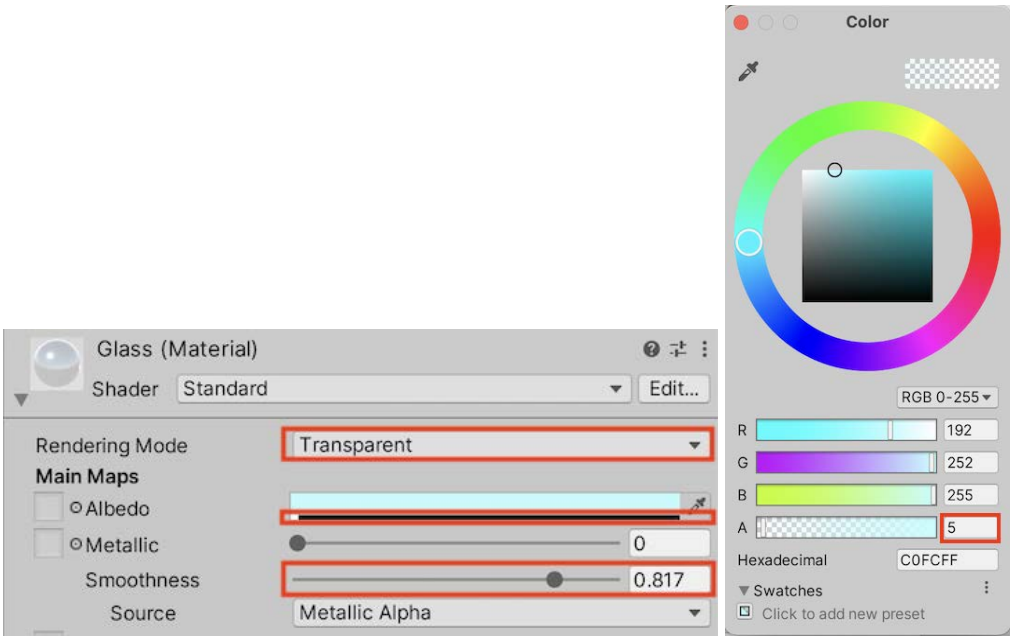
The last step in the material import settings allows us to adapt the material settings in Unity. Most CAD systems differ in how they define materials, and we often lose information during FBX export and import.

- **Missing Texture Maps:** Even if you have extracted textures and materials from the FBX files, Rhino often forgets to export all textures. See 4.2.1 Preparations and Export in Rhino for more information.
- **Missing Normal Maps**: Many materials lack normal maps, resulting in a poorer structural appearance. If no normal map is assigned, click Normal Map and select the texture file. It mostly has the same name as the color (Albedo) texture, but with the ending NRM.

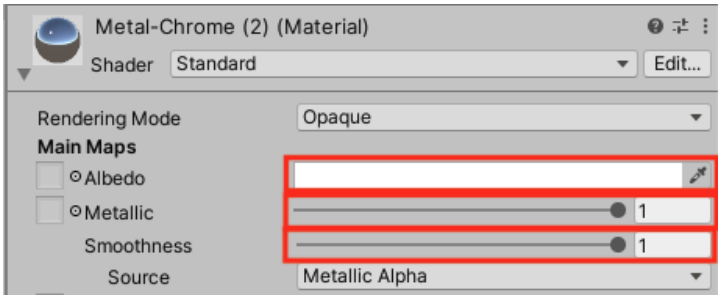


- **Wrong Transparency**: Window glass or water surfaces should be transparent and glossy. To achieve this, set the *Rendering Mode* to *Transparent*, the *Smoothness* to a high

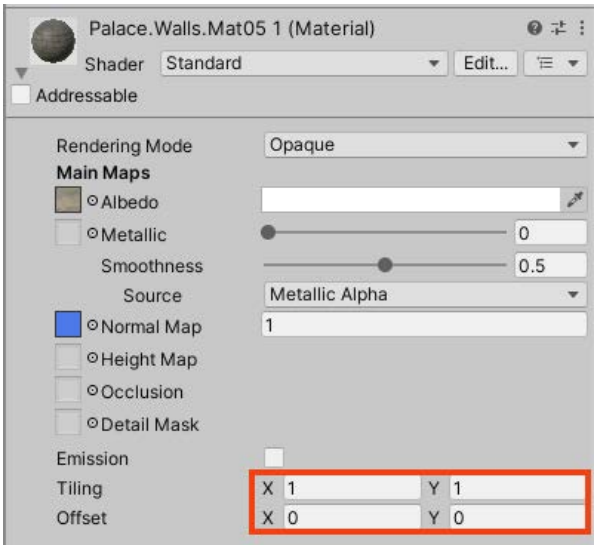
value, and the alpha value (A) of the *Albedo* (diffuse color) to a low value:



- **Wrong Metallic Reflections**: Metals reflect light differently than non-metals. The chrome steel of the *Barcelona Pavilion* is highly reflective (*Albedo = White, Metallic = 1, Smoothness = 1*):



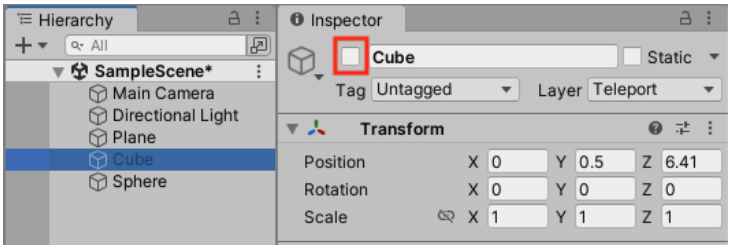
- **Adjust the Texture resolution**: The texture scale is often not exported properly. To change the scale or offset of texture maps, you can adjust the *Tiling* and/or *Offset* value in the material properties in the *Inspector* window of the selected object.



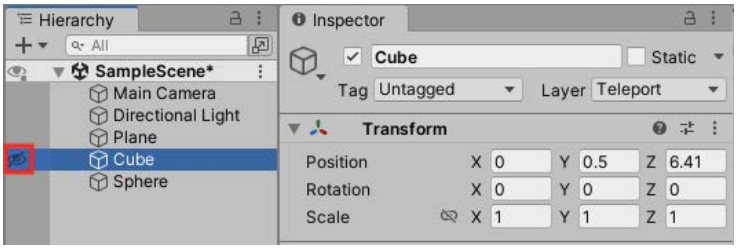
5.2.4 Show and Hide Objects

You will have some objects in your *Rhino* project that you don't want to see in VR, such as the window cutout volumes. To hide an object or an entire group of objects in Unity but not delete them, you have two options:

- **Disable objects** in the editor and in the built VR app:

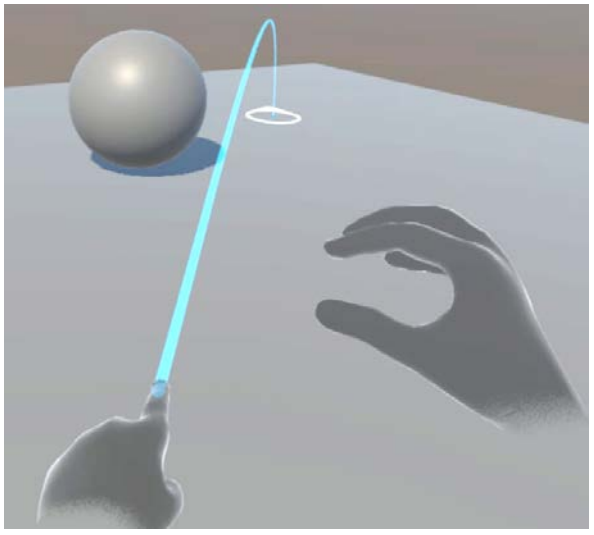


- **Hide objects** only in the editor. These objects will still appear in the VR app:



5.2.5 Add Teleportation

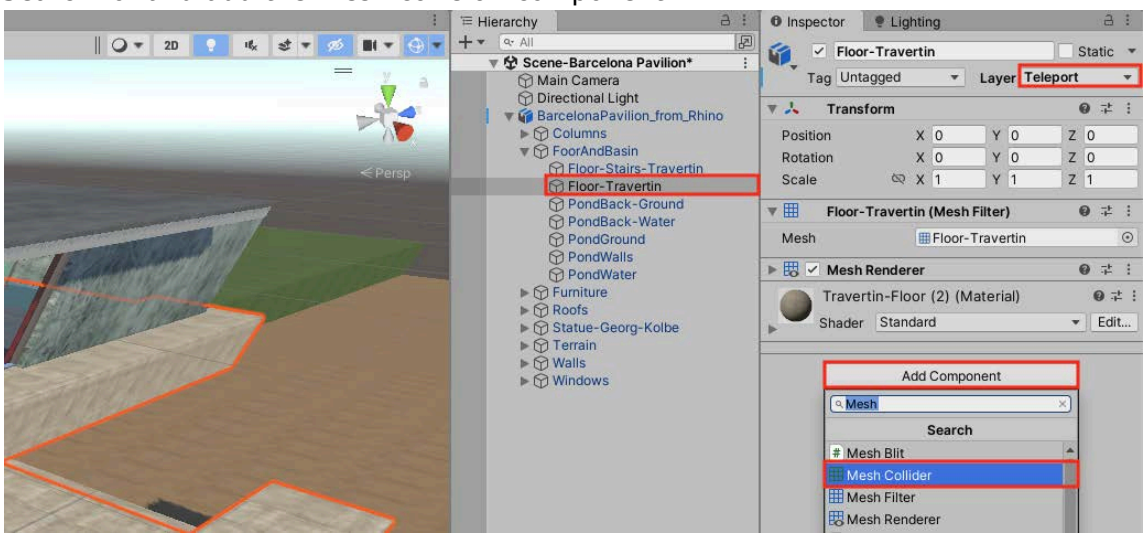
In our real VR space (aka *Boundary,* see 4.1.2.2 First Steps in Oculus Quest), we have limited space to walk around. If you want to navigate a larger distance, you can do a so-called ***teleportation***: Either with your straight left or straight right index finger, you can aim with a parabolic curve to a target position. If the target position allows teleportation, the curve will be light blue; otherwise, it will be red. With the other hand, you have to execute the teleportation with a ***pinch gesture*** (= quick snapping of index and thumb):



Teleportation lets you navigate to places where you are allowed to. Be aware that this pinch gesture works best if the headset's cameras can see the thumb and index finger. If your eyes can not see them, the cameras probably can't either.

- **Add Mesh Collider**: We must add a mesh collision component to the objects we want to teleport to. Select such an object and click on *Add Component* in the *Inspector* window.

Search for *and add the mesh collision* component.



- **Set the Layer to Teleport**: In addition, all objects where you want to be able to teleport must have their *Layer* set to *Teleport* on the (top-right of the *Inspector* window). Wherever you are allowed to teleport, you will see a bright blue parabola.
- **Set the Layer to Non-Teleport**: If you explicitly don't want to teleport on an object, you can set its *Layer* to *Non-Teleport*. This will also make it red when an object to teleport is underneath or behind it.

5.2.6 Build the Scene

The following two sub-chapters describe two ways to build and install the scene:

- **Build a Local Scene installed on the Headset**: This is the standard procedure we followed at the beginning of the chapter, and it is also the standard procedure Unity and other VR creation tools follow.
  - *Advantages*:
    - Full flexibility
    - Full privacy. Only people using your headset can see the scene.
    - No dependency on our server system.
  - *Disadvantages*:
    - The headset needs to be set up correctly for development.
    - The headset needs to be connected during the build process.
    - The headset needs to be disconnected for testing (recommended).
    - Your scene can only be seen on your headset.
    - Your scene can not be shared with others.

- **Build a Remote Scene stored on the ArchiVR Server**: Unity allows the building of the scene so that it can then be stored on a server and downloaded at run-time to the headset.
  - *Advantages*:
    - Faster build process.
    - No cable connection or disconnection is required.
    - Your scene can be shared with others. This is a massive advantage because it allows you to share your scene with other students, friends, or even clients of an architect. A modified scene is immediately available to people with access to the server and the viewer application installed on their Quest.
    - Your scene can be seen on other VR headsets that support OpenXR.
  - *Disadvantages*:
    - The Viewer app must be built and/or installed once on the headset.

5.2.6.1 Build a Local Scene installed on the Headset

- **Add your Scene**: At the top of the *Build Settings* dialog, you can see a list of scenes that will be included in the build. So far, you can see only a scene named *Base Scene,* which includes all the functionality we provided you for VR navigation in this project. The *Start* and *Example Scene* are the scenes from the Viewer Project.
  To add your scene to the Barcelona pavilion, press *Add OpenScenes*. You can add multiple scenes here that you can switch to with a VR menu.

- Press ***Build and Run*** to start the build process. The first time can take several minutes, but subsequent builds are much faster.
- **Test your scene** on your headset.

5.2.6.2 Build a Remote Scene stored on the ArchiVR Server

This is all the same as already explained in 2.3.4 Build & Upload to ArchiVR.

5.4 Checklist for Unity VR Creation

1.3.2 Expected Hardware & Software

- ☐ Do you have enough memory (RAM) to run your CAD software and Unity?
- ☐ Do you use the latest Unity LTS version?
- ☐ Did you install the Android Module with Unity?
- ☐ Is your project not a OneDrive folder?

4.3 Create your own Scene for your VR app:

- ☐ Has your model moved so that the start position [0,0,0] is at a useful location?
- ☐ Are your materials corrected with all your textures?
- ☐ Are your objects that you don't want to see correctly disabled?
- ☐ Do all objects that teleport have a *Mesh Collider* and a Layer set to *Teleport*?

4.3.1 Import the 3D Model:

- ☐ Is the Platform switched to *Android*?
- ☐ Is your scene and model within *Assets/_MyScenes/{myName}/{mySceneName}*?
- ☐ Are the model materials extracted so that they can be modified?
- ☐ Are all the needed textures copied into a subfolder within your model's folder?

5.3.2.3 Lightmapping: Baking the Light:

- ☐ Have you set the *Generate Lightmap UV* option in the model's import settings?
- ☐ Are your light sources set to *Baked* mode?
- ☐ Is your model set to *static* or at least to *Static (Contribute GI)*?
- ☐ Did you generate the light maps by pressing *Generate Lighting*?

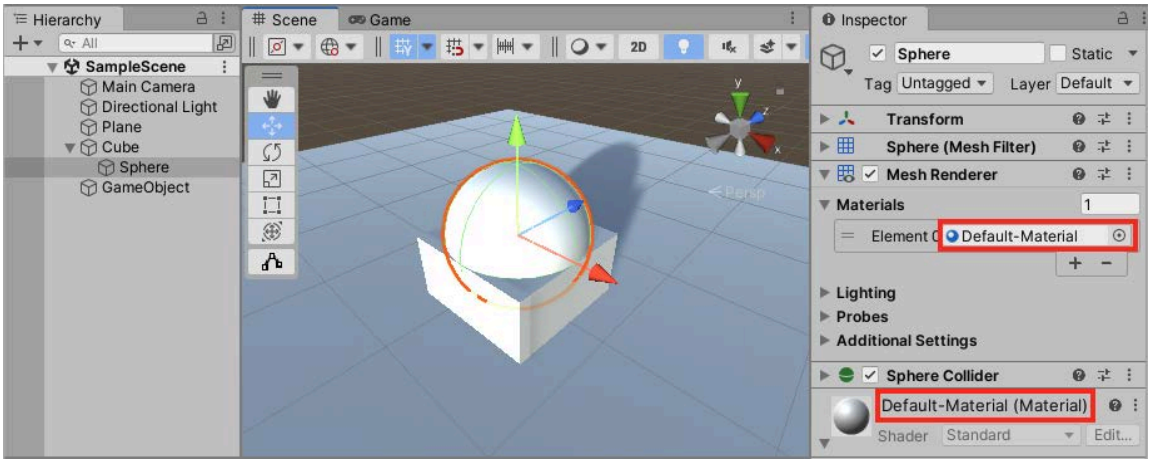
4.3.5.2 Build a Remote Scene stored on the ArchiVR Server:

- ☐ Did you create an account on https://archi-vr.ti.bfh.ch/?
- ☐ In Unity, are you logged in under *Tools* > *ArchiVR*?
- ☐ Did you 2.3.4 Build & Upload to ArchiVR and check the log output for success?
- ☐ Do you see your project on https://archi-vr.ti.bfh.ch/?
- ☐ Did you install the 2.2.1 Install the ArchiVR Viewer on a Quest VR Headset?

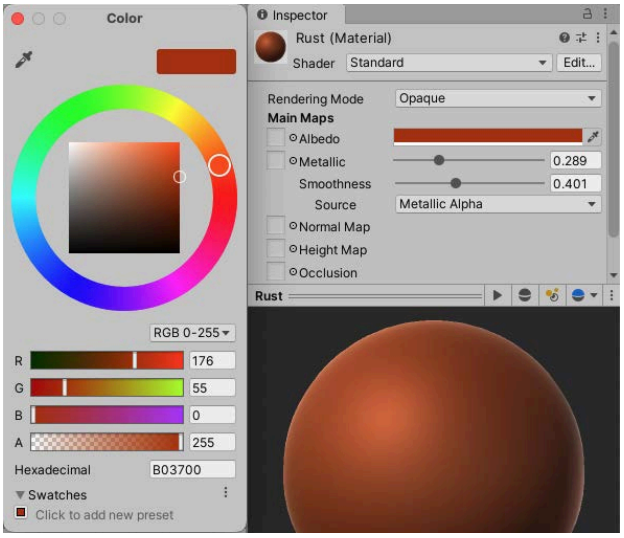
6 Materials and Lighting in Unity

6.1 Materials in Unity

- **One or more materials**: Every game object (GO) with a mesh has a *Mesh Renderer* with one or more materials assigned to it. The material itself is an additional component of the GO.
- **Default Material**: Every GO that we add to the menu, *GameObject > 3D Object* has the default material assigned. We can not change this default material.



- **Create new material**:
  - Over the *Project* window, press RMB and select *Create > Material*.
  - Give it the name *"Red"*.
  - **Albedo**: You can see all the material parameters in the Inspector window. Click on the white color of the *Albedo* field and choose a red color tone:



Albedo is a synonym for the *diffuse color*, or the material's color.

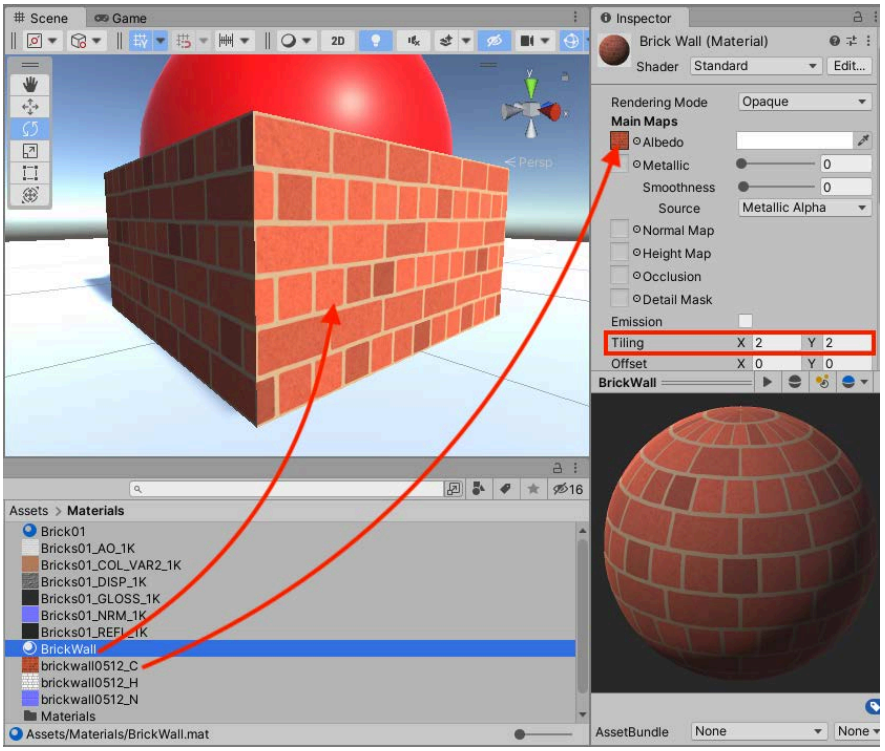
  - **Metallic**: Use this slider to adjust a material's metallicness. In reality, there are only fully metallic or non-metallic materials. Metals reflect less of their Albedo color.
  - **Smoothness**: The more you increase the *Smoothness* slider, the sharper and glossier the reflections. These reflections (the light and the skybox) are not real global reflections. These simplified reflections can be calculated in

real-time, whereas accurate reflections are much more expensive. The *Smoothness* property is usually called *Roughness,* which is the inverse of *Smoothness*.

- **Assign the Red Material** to the sphere by dragging it from the *Project* window onto the sphere in the *Scene* editor. You can assign a material to multiple GOs, but if you change its parameter, their appearance will change.

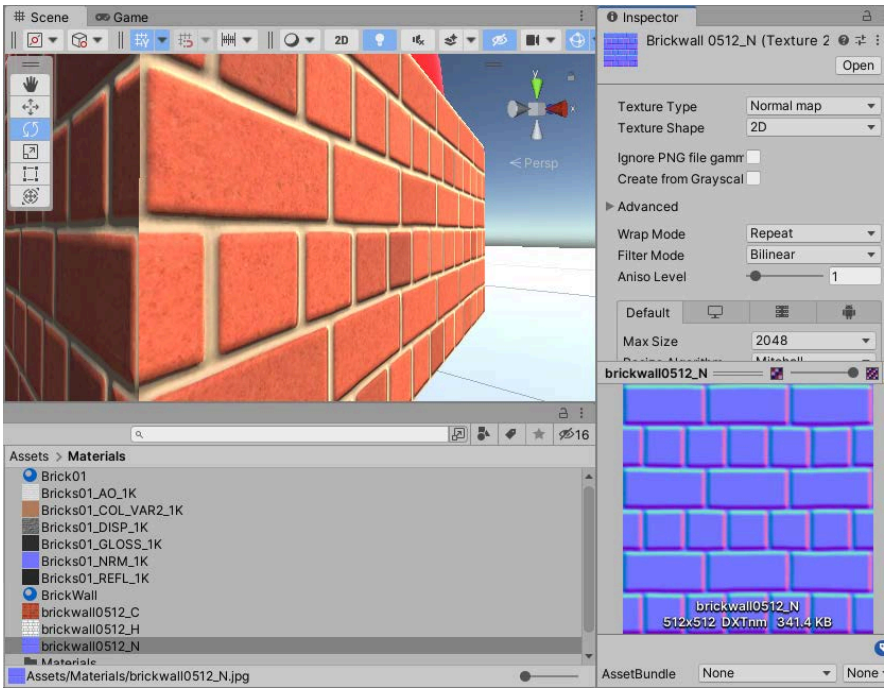
- **Materials with Textures**: To the left of the material property names, you see a little empty square. All properties with this square also accept an image (aka *texture*) as input. The so-called texture or UV coordinates determine which part of a *texture is mapped* to the object. These can only be modified in modeling tools.

  - **Copy Textures**: Copy all files from the *Textures* folder from our repository to the *Material* folder in your Unity project.
  - **Create Brickwall Material**: In the *Material* folder, create a new material with *RMB > Create > Material* and name it *Brickwall*.
  - **Add Albedo Texture**: Drag the *brickwall0512_C.jpg* file onto the little square of the *Albedo* field.
  - **Assign the Material**: Drag the *Brickwall* material onto the cube:



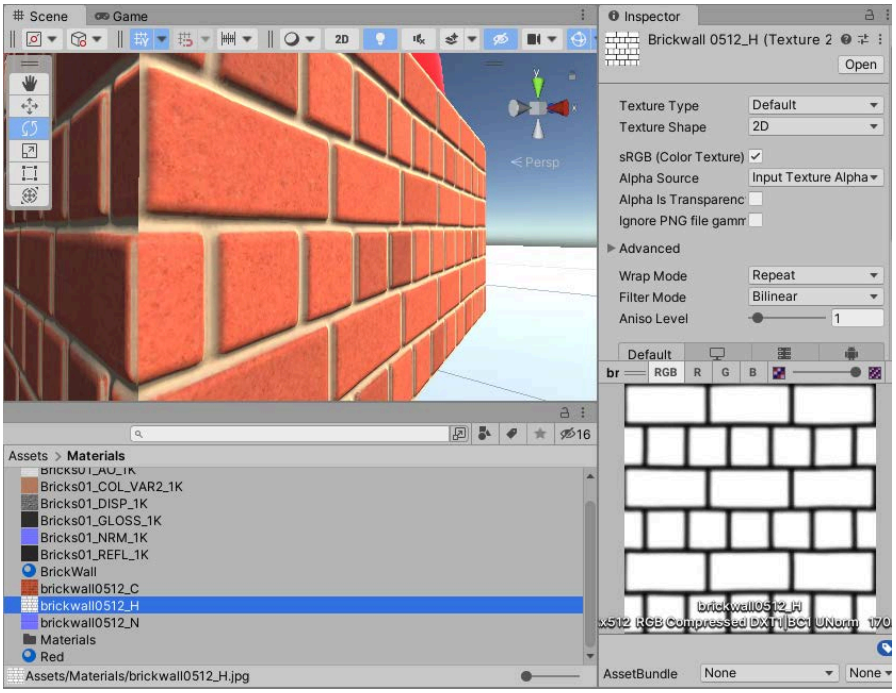
The cube now appears much more detailed, but its underlying mesh has not changed.

  - **Increase Tiling**: You can increase the mapping's resolution by setting factors higher than 1 in the *Tiling X & Y* fields.
  - **Add Normal Map**: Drag the *brickwall0512_N.jpg* file onto the *Normal Map* field. If the message "*This texture is not marked as a normal map*" appears, press *Fix Now*.



With the normal map applied, even more details appear on the surface. Normal maps store a normal vector for each pixel in the image's RGB channels. A perpendicular normal has a light blueish-violet color. All other colors stand for a distorted normal vector. In the materials shader program that runs on the GPU, the normal vector is used to mimic more or less lighted areas, such as shadows in the brick joints.

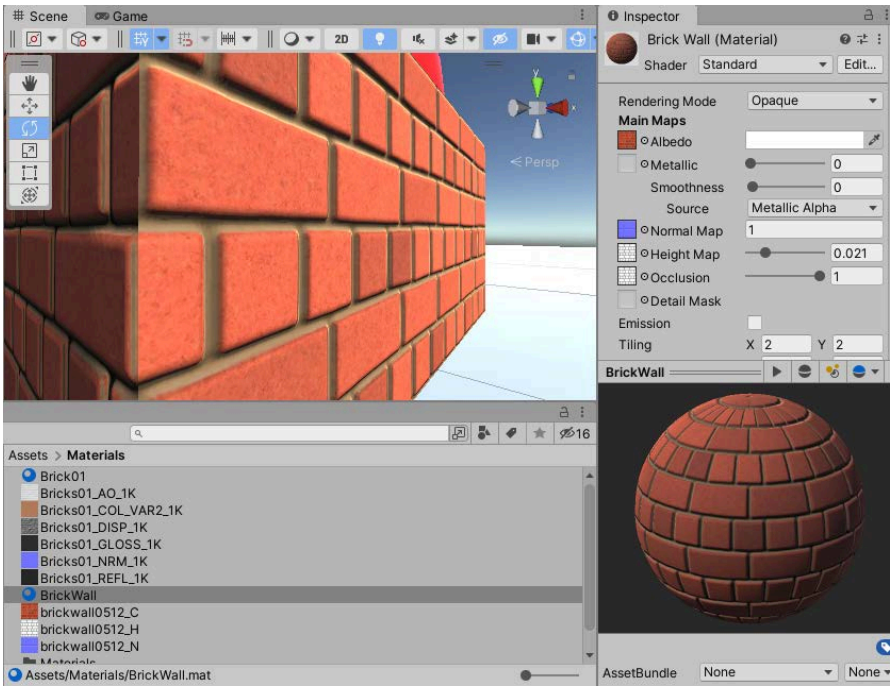
- **Add Height Map**: By adding the *brickwall0512_H.jpg* file to the *Height Map* field, you can give the material a more depth-like appearance. A height map is a grayscale image where white represents high values, and black represents low values.



With the additional height map, the single bricks seem to stand out of the plane of the walls. Again, this is just a mimicking effect. Each face of the cube still only has two triangles.

- **Add Occlusion Map**: By adding the same *brickwall0512_H.jpg* file to the *Occlusion* field, we can tell the material where the occluded areas should

appear darker because less light arrives there:



Exercise with Materials:

The materials still look pretty artificial. *T*he second set of texture images begins with *Brick01* and is in the Textures folder.

6.2 Lighting in Unity

6.2.1 Direct Lighting in Unity

Direct lighting is the reflection of surfaces with specific materials. It is calculated by summing all light directly from active light sources, ignoring light reflected from other (passive) surfaces.

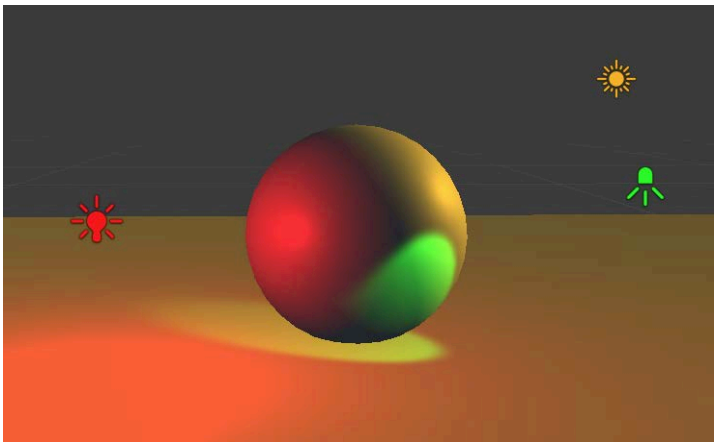
6.2.1.1 Light Sources in Unity

All rendering engines offer about the same types of light sources:

- **Point Lights** are located at a point in the scene and emit light equally in all directions.
- **Spot Lights** are point lights that emit light in a cone-shaped beam.
- **Directional Lights** are infinitely far away and emit light in a single direction.

All these light sources are theoretical sources and themself invisible. If you want to create a visible-light source, such as a light bulb, you must model its geometry and assign it an emissive material. For more details on light sources, please refer to the documentation of your rendering tool:

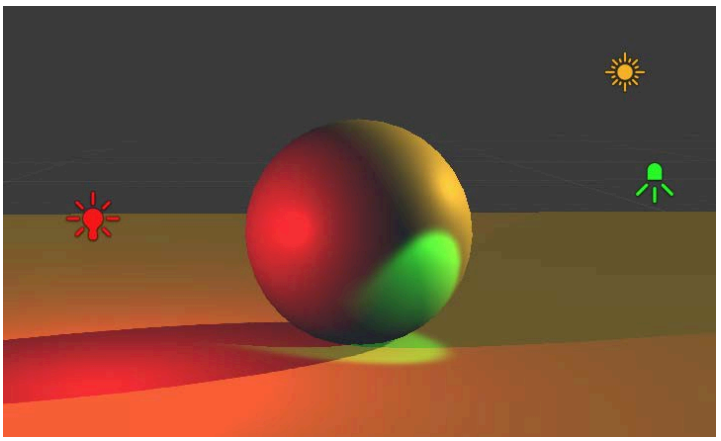
- Types of lights in Unity
- Light objects in Blender
- Create Lights in Rhino



A white sphere on a white plane is enlightened with a red pointlight, a green spotlight, and an orange directional light. Note that the red and green lights have a range and smoothly fade into it. The directional light symbol (aka gizmo) is only the location for editing its direction. Its light source is infinitely far away and has no range.

Direct Lights with Shadows

All the above-mentioned light sources can cast shadows. This procedure is quite expensive because it needs an additional rendering pass for each light source with shadows.

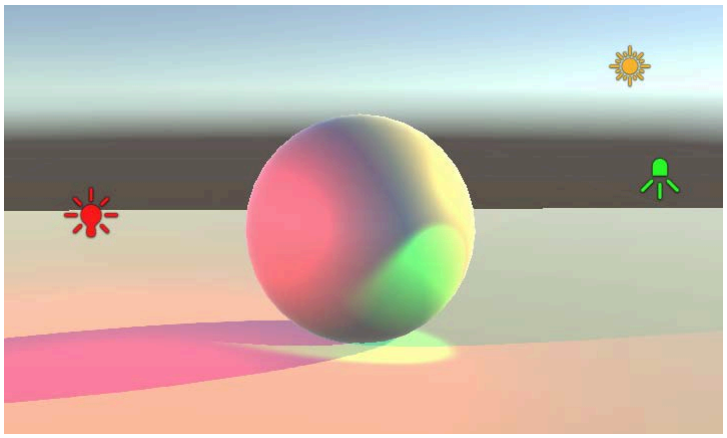


Number of Lights in a Scene

Light and shadow calculations are the most expensive part of computer graphics. No matter if you want to render in real-time or bake the lighting, as we will do in 5.3.2.3 Lightmapping: Baking the Light. As a simple rule of thumb, **the performance decreases linearly** with the number of lights. In the Forward rendering mode used on mobile devices, there is even a **restriction of max. light lights** that can affect a single object. These restrictions are one reason for light baking. Light baking will also take longer with more lights, but we can experience the scene in real time once it's done.

6.2.1.2 Environmental Light Source

Scenes illuminated only by direct light look mostly dark because all ambient light is missing. In all render engines, you can define an environmental light with a constant (ambient) color or with a skybox. In Unity, by default, you get a generated skybox that gives white surfaces a blueish tint.



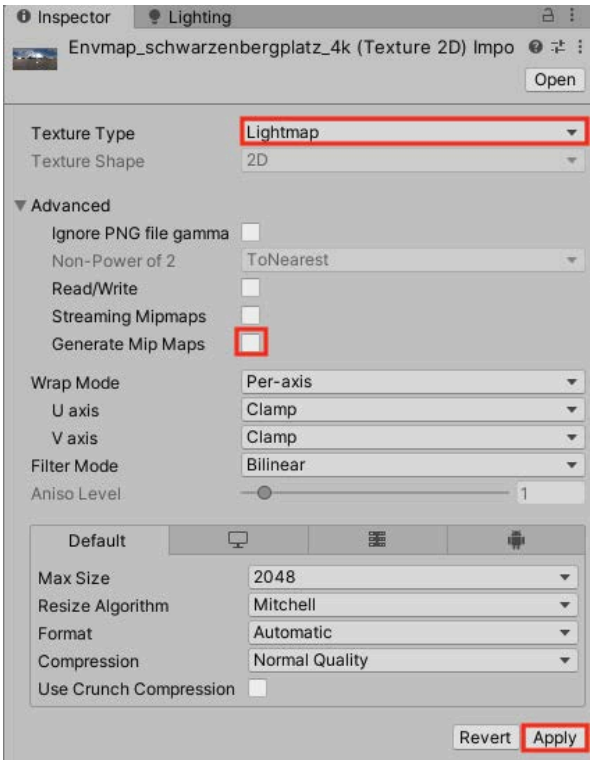
High Dynamic Range Images

A skybox used as an environmental light can also be defined using a 360° equirectangular image. These images are not standard but *high dynamic range (HDR) images, where multiple photos at different apertures are stored in a single* file. A good source for such HDRIs is the website polyhaven.com. If you want to create your own HDRI at the location of your project, you will need a 360° camera. See Chapter 9.6 Create your own 360° HDR Image for more details.

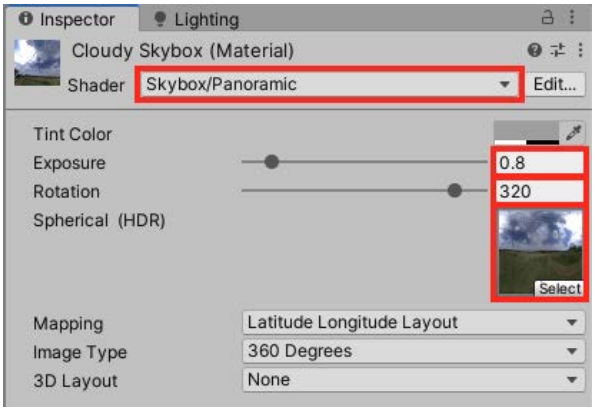
To create and assign such a skybox in Unity, do the following steps:

- **Copy the HDRI-Image** (*Demo-Projekte/Sponza/Textures/envmap_cloud_layers_4k.hdr*) from the demo projects into your project's *Assets* folder.

- **Adjust the import settings in Unity**: Select the imported HDRI image in your *Project* window and adjust its import settings as follows:

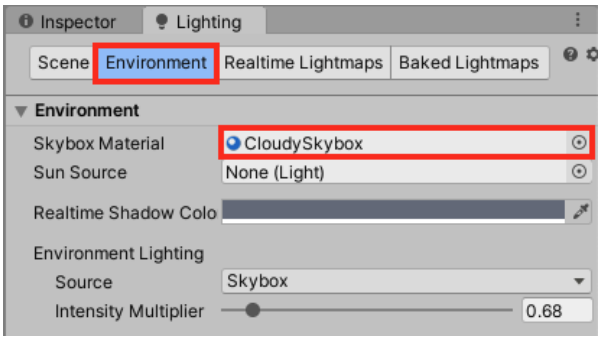


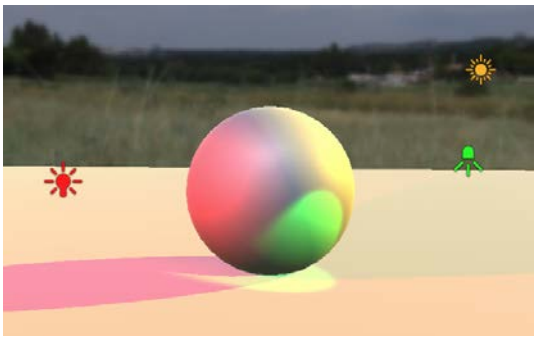
- **Create a new material** by right-clicking > Create > Material, then name it *CloudySkybox*.
- **Set materials shader** to *Skybox/Panoramic*.
- **Select the HDR image** *envmap_cloud_layers_4k.hdr* as the *Spherical (HDR)* image.
- **Set the Exposure and Rotation** to the following values:



- **Open the Lighting window** by clicking on the menu *Window* > *Rendering* > *Lighting*.

- **Select the new Skybox** in the Environment tab:





6.3.2 Indirect Lighting in Unity

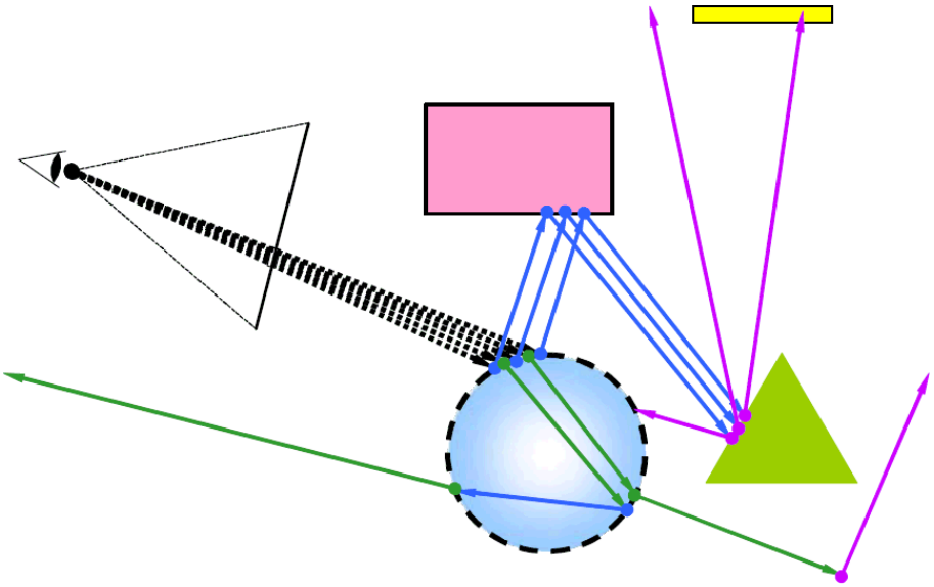
Indirect lighting in architecture is, of course, essential, but unfortunately, it is also the most challenging aspect to simulate accurately and quickly in computer graphics. For this chapter, we have chosen the Sponza Atrium test scene. It is a famous benchmark for indirect lighting (global illumination) in computer graphics because the atrium gets only light from the top and rarely direct sunlight in the summertime. The atrium is within the Sponza Palace, a 16th-century palace in Dubrovnik, Croatia. Frank Meinl originally designed this model for the German game engine company Crytek. It is reasonably low-poly and runs in real time on laptops. Intel also has a much more demanding, high-poly version with all 4K textures.

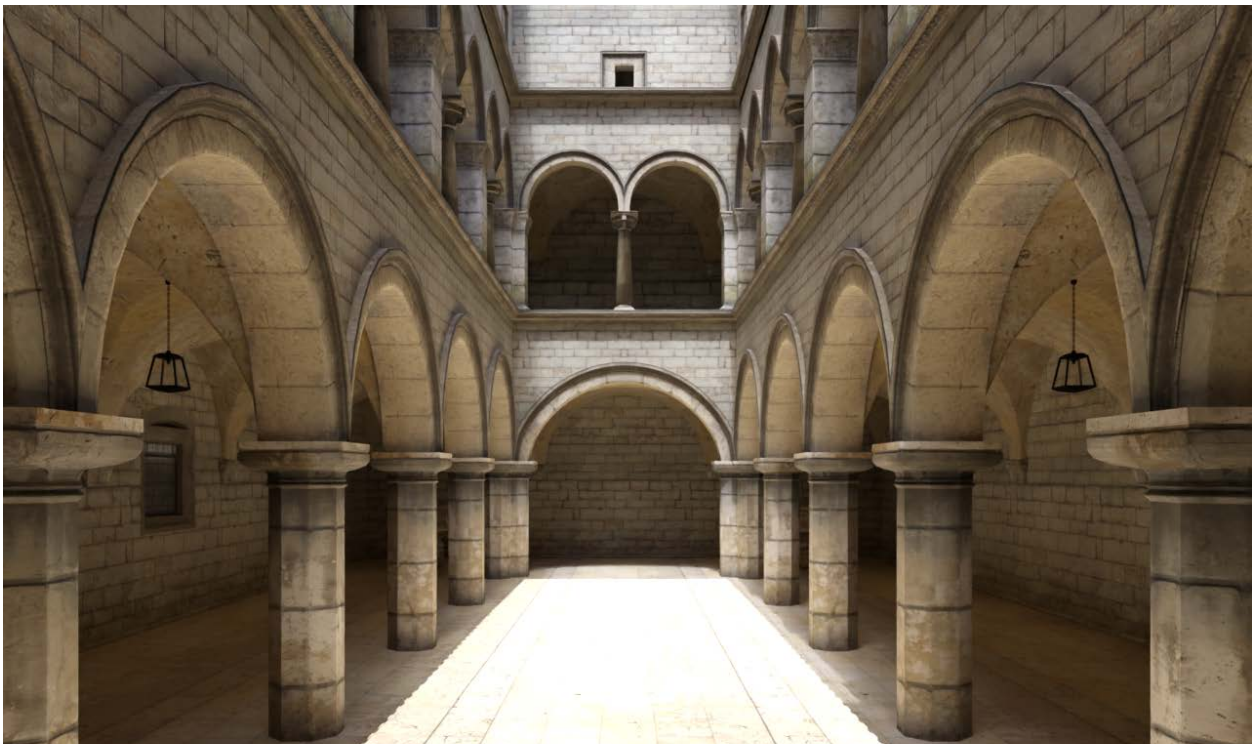
6.3.2.1 Comparison of Rendering with Photographs

- **Render in Rhino or Blender**: You can find the Sponza model in the Rhino (3dm) and Blender (blend) file formats. The high-end rendering in both tools is made with the Cycles renderer from Blender.
  To start the rendering process, do the following steps:
  - **In Rhino**: Double-click the *Sponza.3dm* file to open *Rhino* and choose the menu *Render* > *Render*.
  - **In Blender**: Double-click the *Sponza.blend* file to open *Blender* and choose the menu *Render* > *Render Image*.

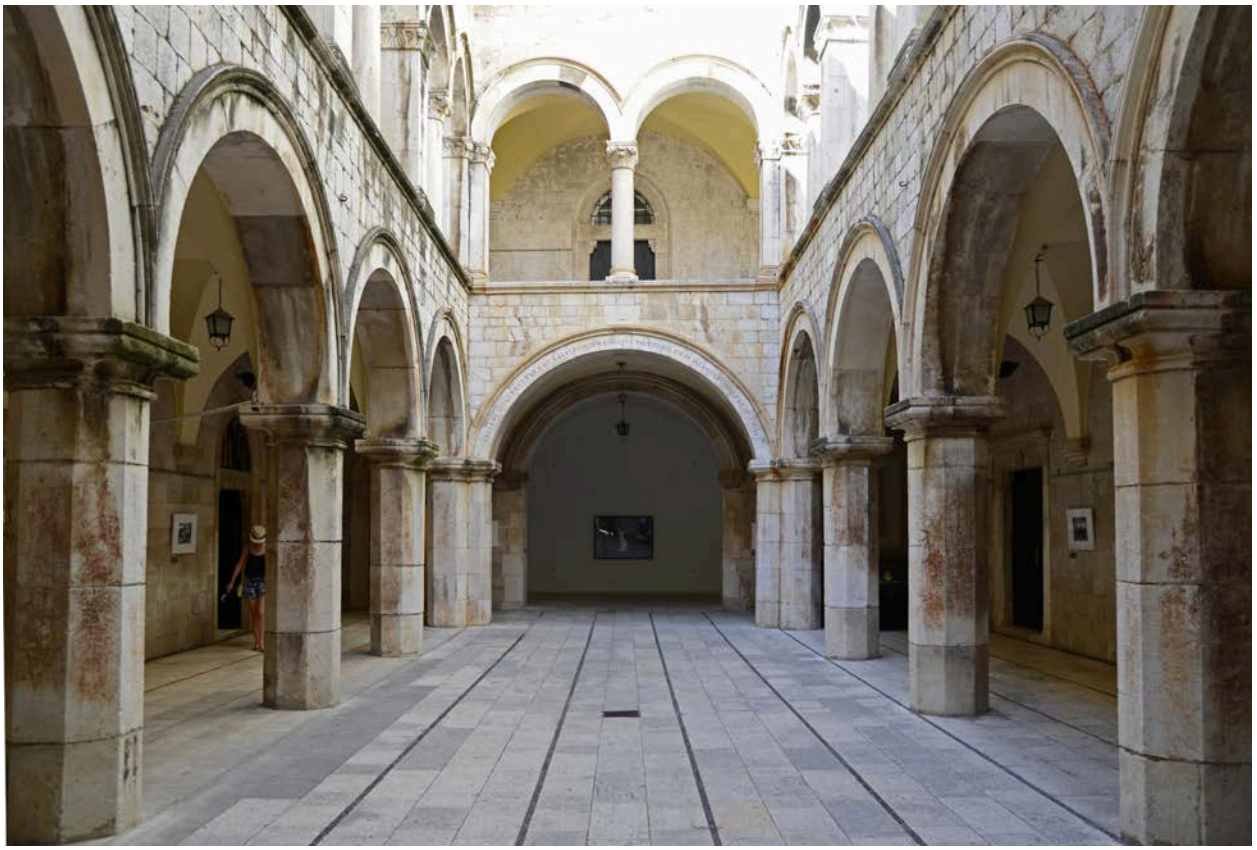
- **Path Tracing**: Depending on your computer, such a rendering takes 2-10 minutes. The *Cycles* renderer uses *Path Tracing,* where many rays (100 in the Blender rendering) get shot through each pixel. At every intersection, they can be reflected

or absorbed. In the end, the pixel color gets averaged over all paths.





Rendering made with Blender and 100 samples (=paths) per pixel.

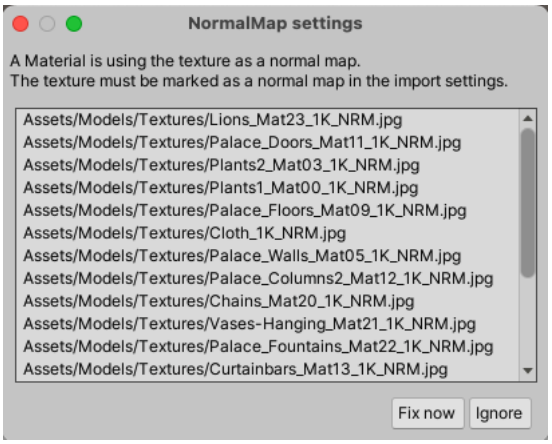


We can see quite a few differences: the viewpoint is different, and sunlight shines straight downward (perpendicular to the floor) in the rendering. The photographed scene is illuminated without direct sunlight. You can also compare the other photographs in the folder Fotos.

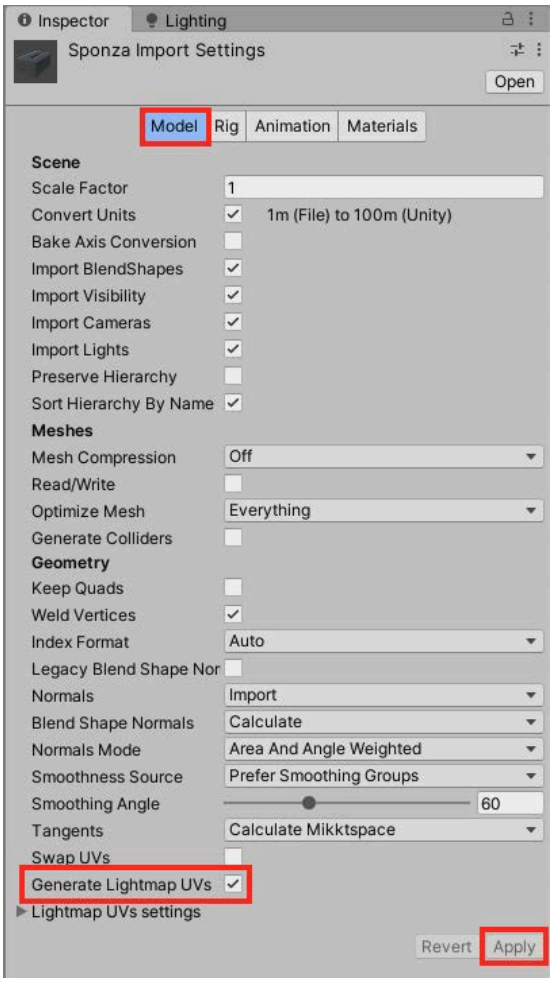
Even if the Cycles renderer is known to be physically accurate, we must recognize that we cannot see the physical reality. Our eyes act as cameras and can adapt to the scene's brightness by opening or closing the iris. Similarly, you can adjust the exposure and the gamma in the Blender and Rhino render settings.

6.3.2.2 Import and adjust the Sponza Model in Unity

- **Create a new 3D project in Unity Hub** and name it *SponzaLighting*.
- **Create a new folder named *Models*** in the *Asset* folder.
- **Copy the model files**: Copy the *Demo-Projekte/Sponza* folder into the *Models* folder of the Unity project. Unity automatically starts importing the files. If the import recognizes normal map textures, some may have to be corrected. Click on *Fix now*:

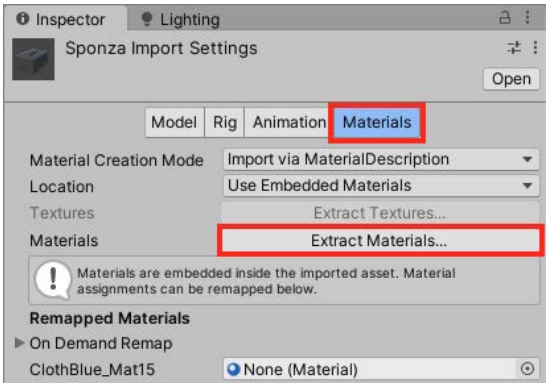


- **Adjust the imported model**: Select the imported model file in the project window and apply the following settings in the *Inspector* window:



Generate Lightmap UVs: Needed to bake direct and/or indirect lighting. Baking indirect lighting is essential for any architectural scene. Baking direct lighting with shadows can massively improve rendering performance because no real-time lighting has to be calculated during rendering. The disadvantage, on the other hand, is that you won't get any real-time lighting effects.

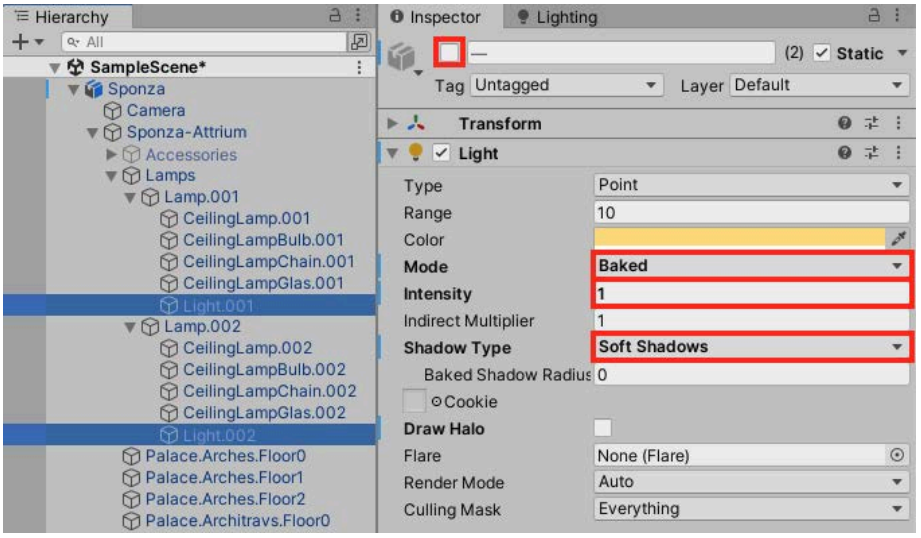
- **Export the materials**: To adjust the imported materials, we have to export them. On the *Material* tab of the *Sponza Import Settings,* click on *Extract Materials …* Create a new folder named *Materials* and choose it as the export folder.



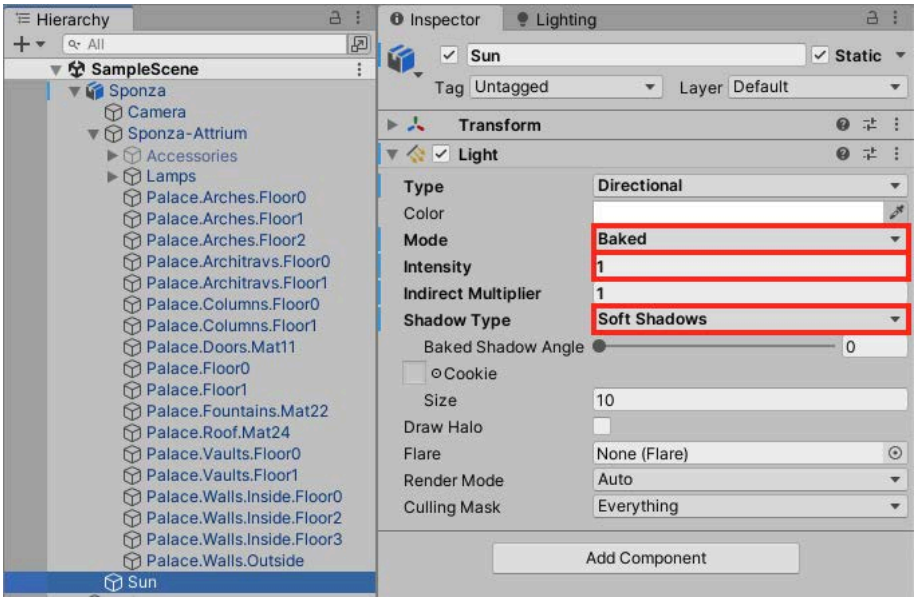
- **Press Apply** after the modification.

- **Delete the Main Camera and Directional Light** in the *Hierarchy* window. We don't need them because we will use the camera and lights in the model.

- **Add the Sponza model to the scene** by dragging it from the *Project* window into the *Hierarchy* window. Ensure in the *Inspector* window that its *Transform Position* is at [0,0,0].
- **Set the Sponza GO to static**: Select the *Sponza* GO in the *Hierarchy* window and check the *Static* option in the *Inspector* window on the top right.
- **Deactivate all Accessories**: We want to concentrate on the lighting and don't need all accessories in the Sponza atrium. You can deactivate the *Accessories* group game object in the *Inspector* window.
- **Adjust and turn off the lamps**: Two point lights are in the side corridors (Light.001 & Light.002). Adjust the following parameters and turn them off by unchecking the top checkbox:

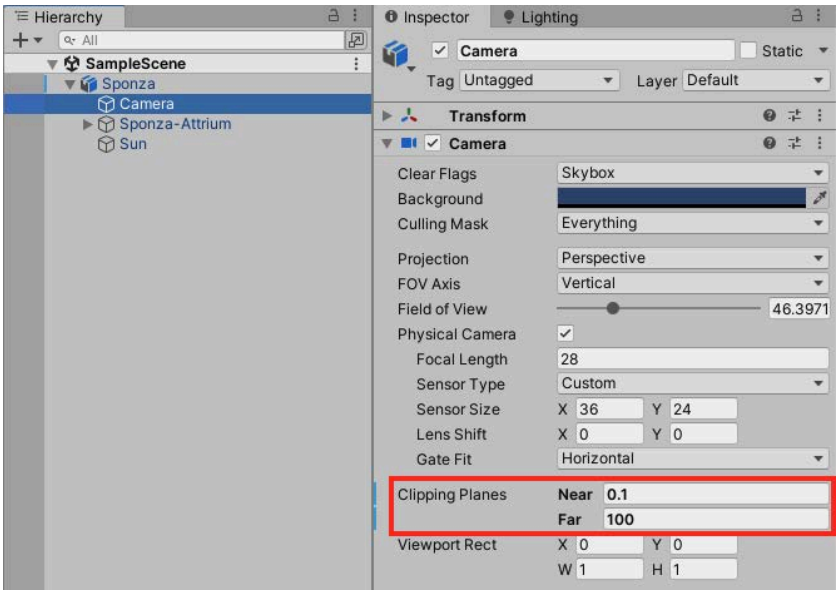


- **Adjust the sunlight** by the following parameters:

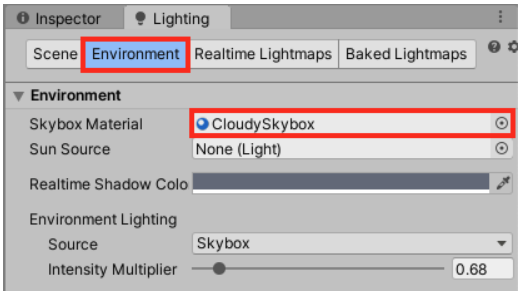


***Light Modes**: There are three light modes: Realtime, Mixed, and Baked:*
***Realtime**: In this mode, all aspects of the light calculation, direct light, and shadows get calculated at runtime in every frame, and no precalculated indirect lighting is included. This is good, e.g., if you want to change the light source at runtime. It is therefore the most expensive light mode and quickly becomes unsuitable for VR because two frames must be calculated.*
***Mixed**: In this mode, we can still have real-time shadows from the direct lighting, but all indirect lighting can be precalculated in a lightmap.*
***Baked**: All lighting effects get precalculated into light maps in this mode. This is the most suitable mode for apps on battery-powered VR headsets because it delivers the highest framerates. The disadvantage, of course, is that no more real-time shadows can be calculated.*

- **Adjust the camera**: Not all parameters were correctly imported. Adjust the near and far clipping planes to 0.1 and 100:



- **Assign the CloudySkybox** in the *Environment* tab of the *Lighting* window:



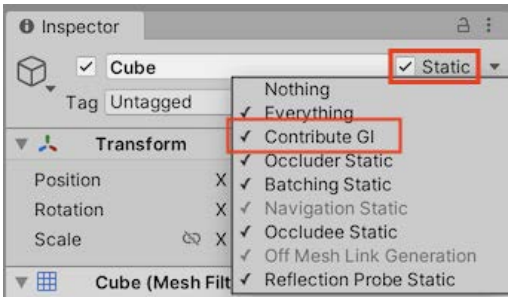
The Sponza atrium with direct lighting:

Without light mapping, the scene is very bright even in the regions where no direct light arrives. This is due to the direct specular reflection from the blueish skybox environment. Direct light calculations are fast, but they don't account for blocking geometries (hence the term "direct"). Only the bright rectangle on the floor gets sunlight.

6.3.2.3 Lightmapping: Baking the Light

Calculating indirect illumination is far too expensive to do in real time, as we have seen during the rendering with the Cycles renderer (in Blender or Rhino). The only way to achieve this critical illumination portion is to precalculate it and store it in additional texture images. When this illumination is baked into images, we, of course, can no longer dynamically change, e.g., the direction of the sunlight. This *light mapping* process is also known as *light baking*.

Only Static Objects can be Lightmapped:
Because light and shadows are stored in additional texture image files (= lightmaps), they will no longer change at runtime. This works only for game objects that are set to *Static* or at least *Static (Contribute GI)*:

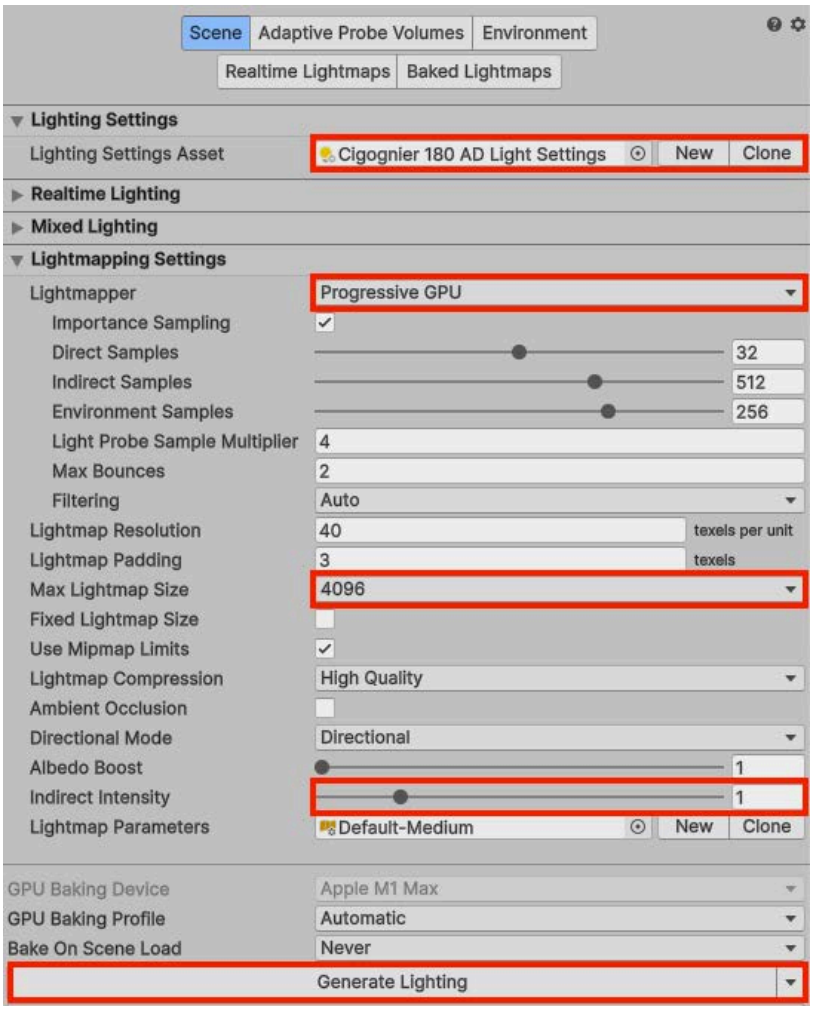


The lighting in Unity (and the other tools) can be tweaked with many parameters, which can be pretty overwhelming for beginners.

- In the *Lighting* window
- In the *Inspector* window of the *Light*
- In the *Inspector* window of the *Skybox*

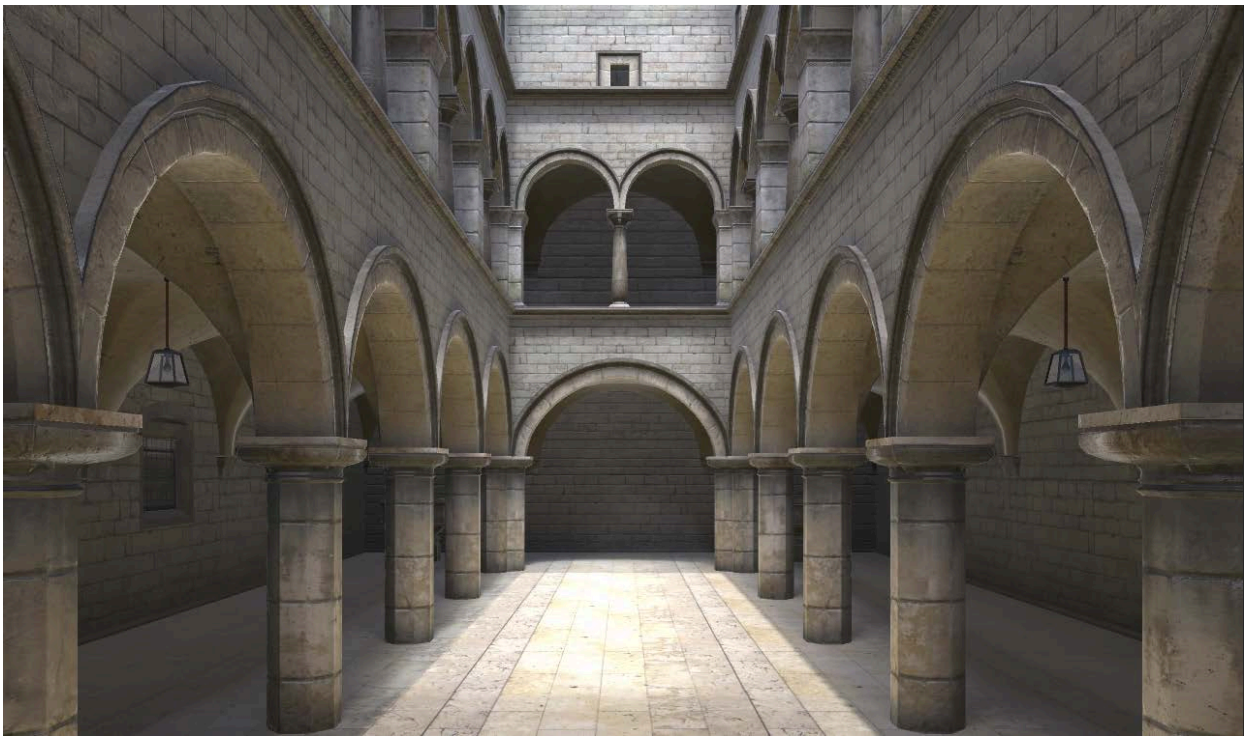
In the Lighting window:

- **Lighting Settings**: In the *Lighting* window, we first have to create a new *Lighting Settings* asset by clicking on *New Lighting Setting* and giving it a name like *SponzaLighting*. These settings just store all parameters of the *Lighting* window.
- **Realtime Lighting**:
  - Leave *Realtime Global Illumination* deactivated. It is a deprecated feature.
- **Mixed Lighting**:
  - Leave the *Baked Global Illumination* activated.
  - The *lighting mode* determines how real-time shadows are generated for lights in *mixed* mode. For such lights, leave it on *Shadowmask*. For *Baked* lights, everything gets baked: direct and indirect light and shadows.
- **Lightmapping Setting**: We explain only the parameters that we modify. For all others, you can hover your mouse over the parameter name or refer to the Unity documentation on Lightmap Settings.
  - *Lightmapper*: Choose either *Progressive GPU* or *Progressive CPU*:
    - *Progressive GPU (Preview)*: Light mapping is calculated on the *GPU* (*Graphics Processing Unit*), which is much faster than on the *CPU* (*Central Processing Unit*), but less stable for large models.
    - *Progressive CPU*: Slower but more stable for large models.
  - *Indirect Intensity*: Spaces with only indirect light tend to be dark. You can boost it with a value higher than 1.
  - *Max Lightmap Size*: For larger project increase this value to reduce the no. of generated lightmaps. The Quest headsets can also handle large light maps.

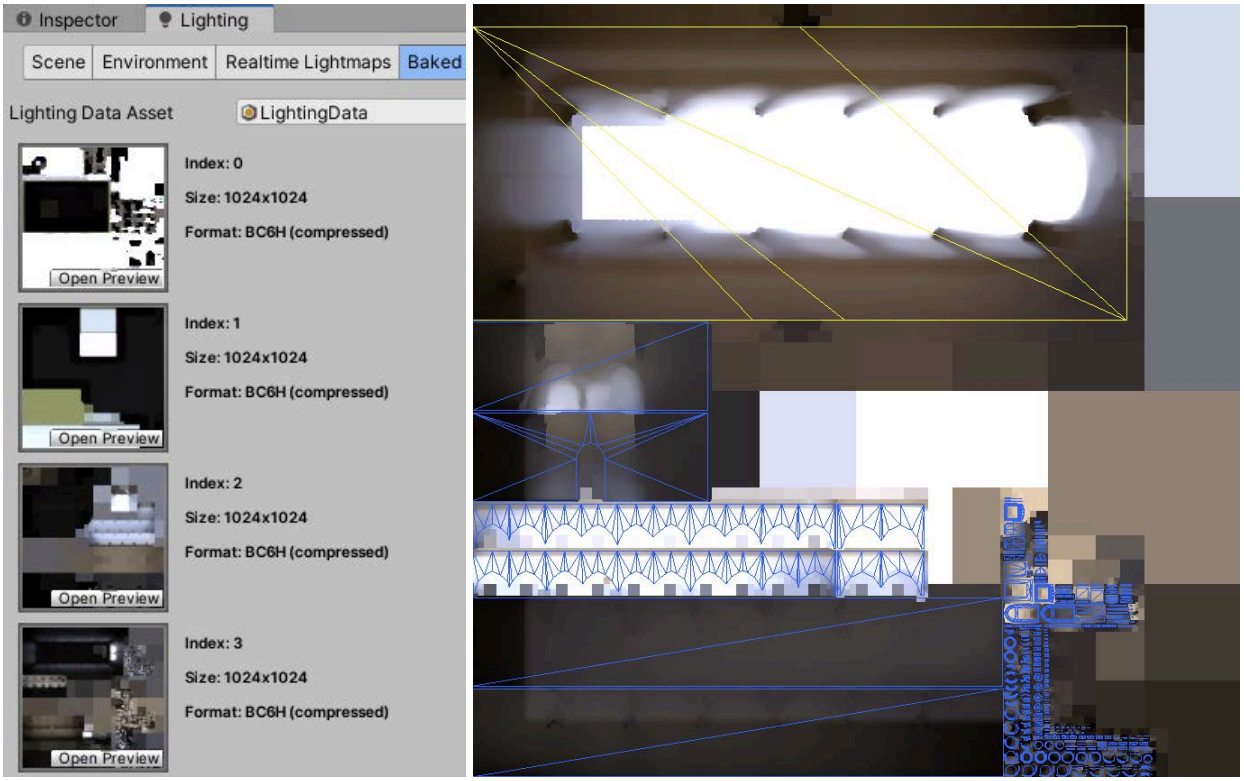


- **Press *Generate Lighting***: Save with Ctrl-S (Cmd-S on Mac) the scene and press the button *Generate Lighting.* Depending on the computer, this can last between 30 sec. to several minutes. To speed up the lightmapping process, you can:
    - Lower the *Lightmap Resolution*.
    - Lower the number of *Indirect Samples*.
    - Lower the number of *Environment Samples*.

After the lightmapping process, the scene looks much better:



You will find the generated lightmaps in the *Baked Lightmaps* tab of the *Lighting* window.

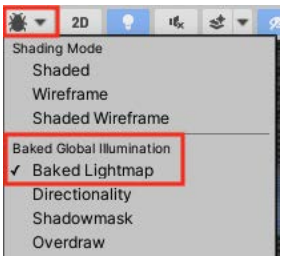


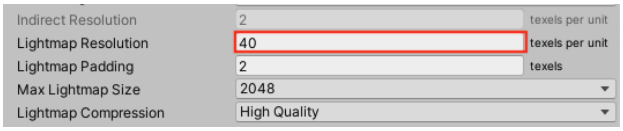
The selected floor mesh in the Scene window and open the preview window of the yellow-surrounded light map.

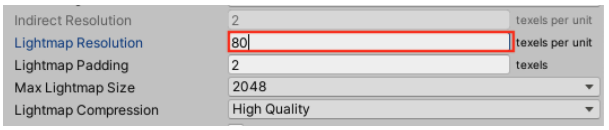
Adjusting the Density of the Lightmap

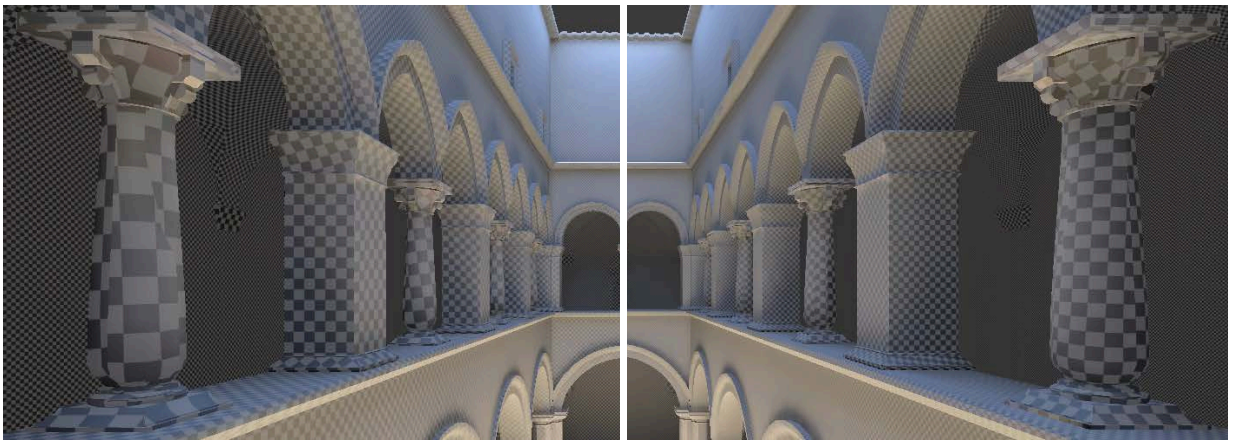
The density of the light maps is primarily controlled by the Lightmap Resolution parameter in the Lighting window's Lightmap Settings. The higher the value, the longer the lightmap generation will take.

You can visualize the baked lightmap resolution with the drawing mode for *Baked Lightmap*:

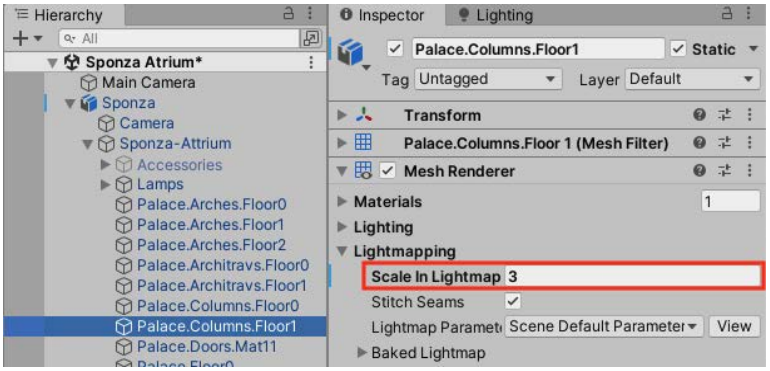








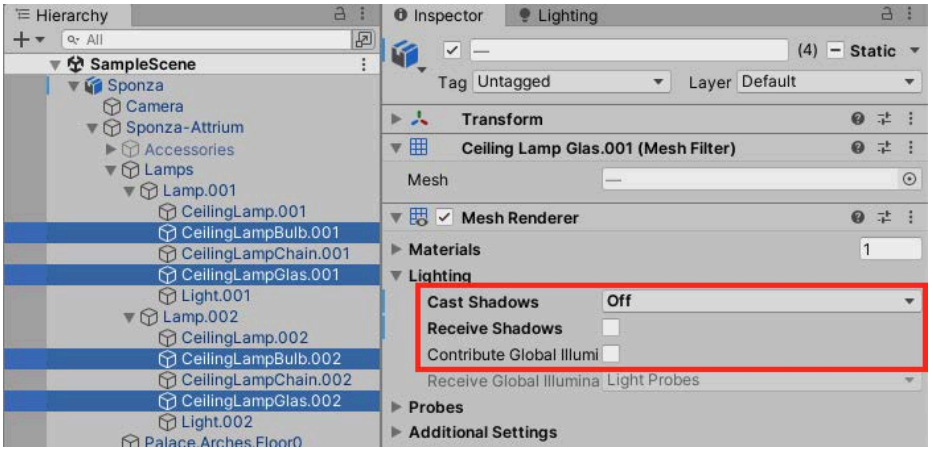
The resolution is not the same on all scene objects, as seen in the images above. To increase the resolution in a specific game object, such as the round columns, you have to increase its *Scale in Lightmap* within the *Lightmapping* settings of its *Mesh Renderer* component:



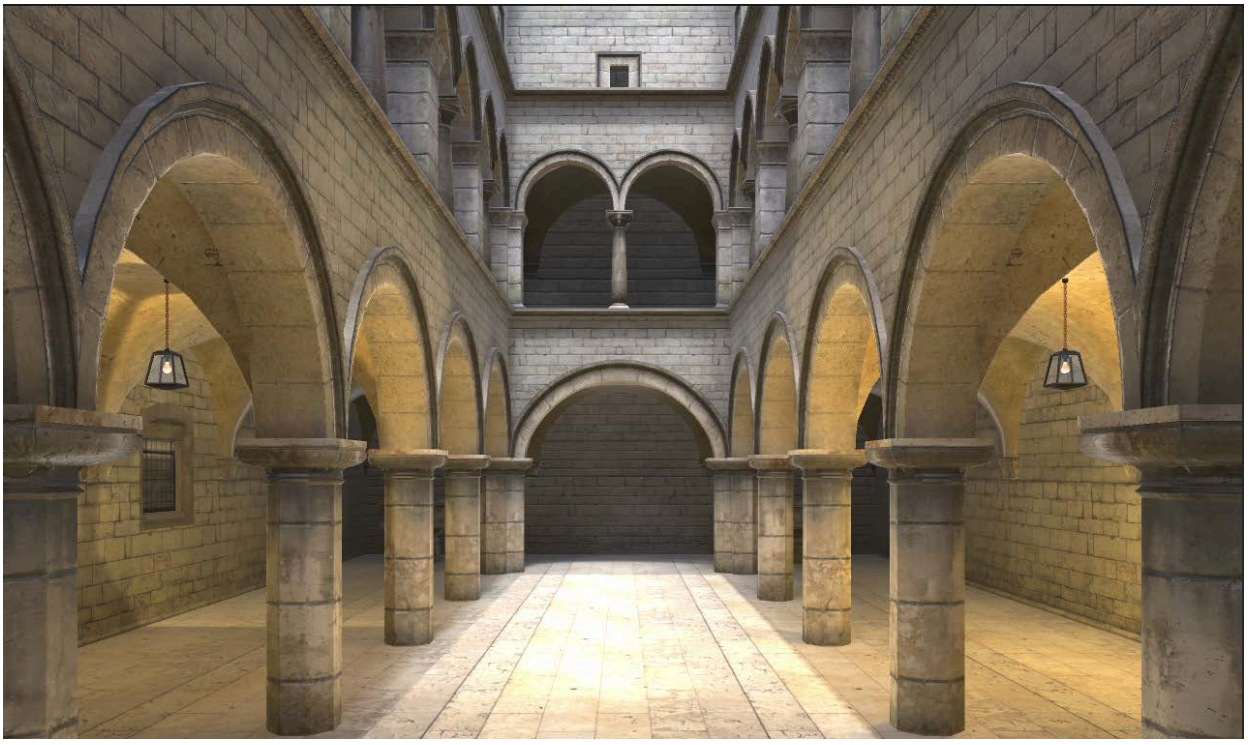
Lightmapping with Pointlights in Lamps

You can also turn on the ceiling lamps in the left and right hallways and bake their lights:

- **Clear Baked Data**: To delete the baked lightmaps, click the little black triangle to the right of the *Generate Lighting* button and select *Clear Baked Data*.
- **Turn on the Light objects**: Turn on the *Light.001* and *Light.002* game objects by checking their enabled checkbox. Make sure their *Light* component is enabled as well.
- **Disable shadow casting**: Even if the point lights' direct illumination illuminates the hallways, baking will not work because the *CeilingLampBulb* and *CeilingLampGlas* objects enclose the light sources. You have to turn off their shadow casting:

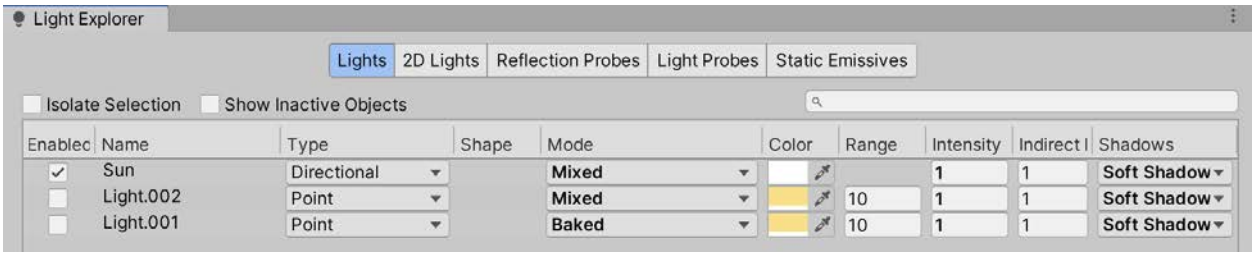


- **Press Generate Lighting** to rebake the light maps.



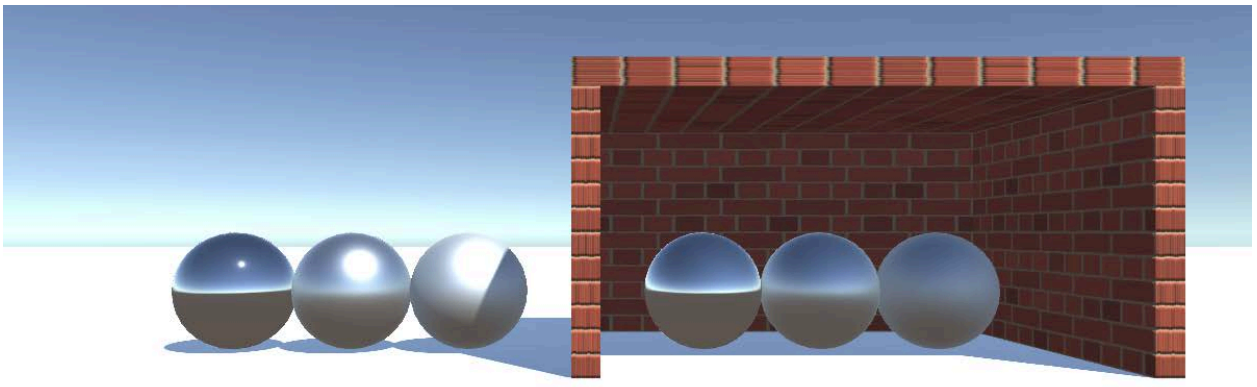
The Sponza atrium with sunlight, environment light, and two lamps.

You can easily lose the overview when there are many lights in your scene. Unity offers the *Light Explorer* that you can open with *Window* > *Rendering* > *Light Explorer*, where you can see and modify the most important parameters:

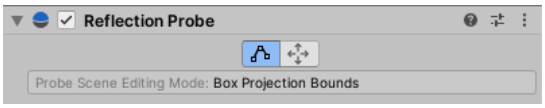


6.3.2.4 Reflection Probes

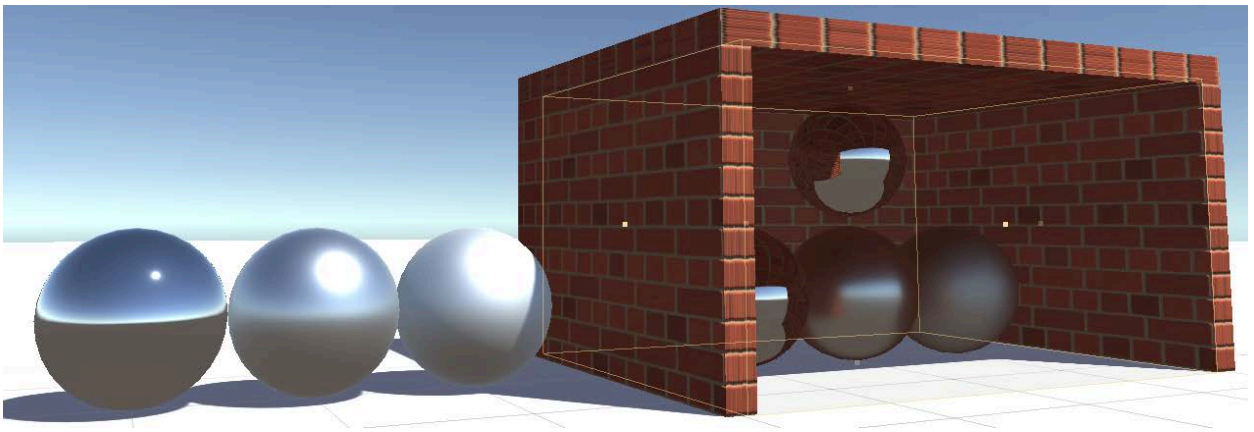
If the smoothness is> 0, the material becomes glossy, as shown in the images below. The shinier they are, the more they reflect the environment, which, in our case, is the default blueish skybox. Everything that is slightly smooth gets this blueish reflection, no matter if the object can see the sky or not, such as the spheres in the house.



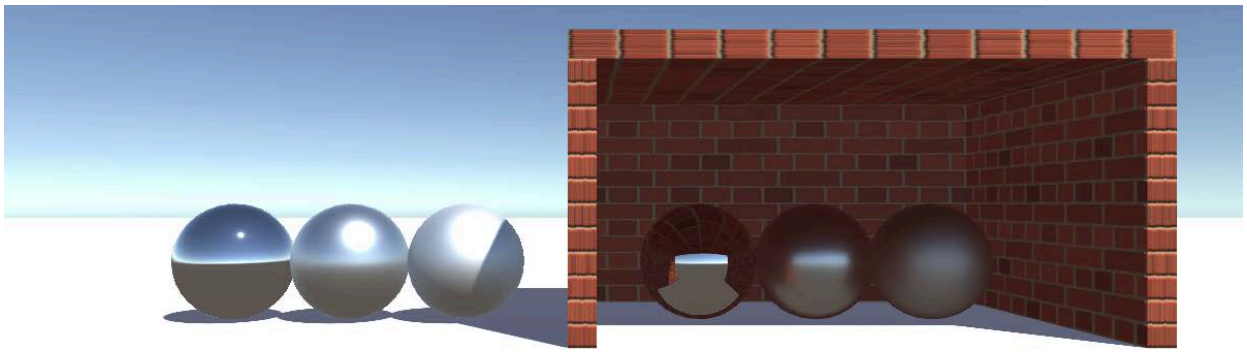
To avoid that, you can define a *Reflection Probe* by adding the *Reflection Probe* component to an empty GO. A reflection probe has a center and an extent that you can edit. You can enter edit mode by pressing the left button at the top-left of the *Reflection Probe* component to change the extent and the right button to change the center.



After pressing *Bake,* you will get a sphere at its center that visualizes the captured environment.

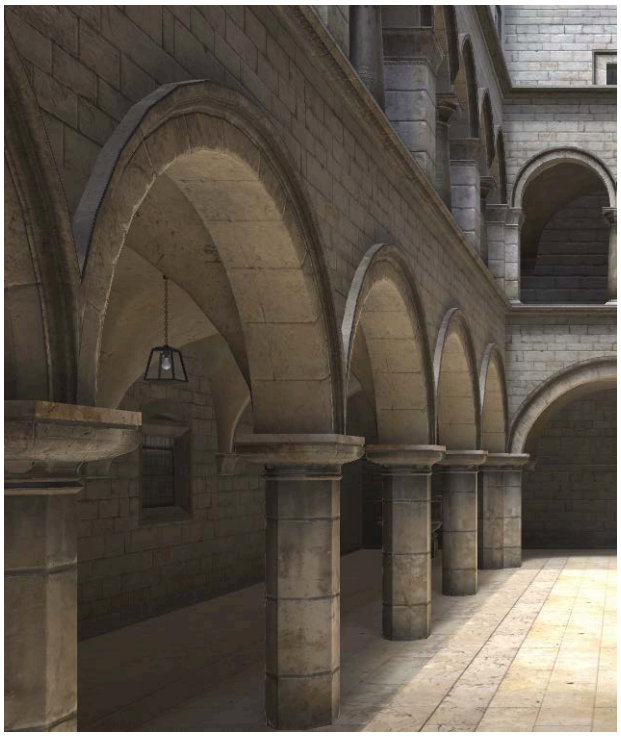


All objects that intersect the bounds of the reflection probe now get their environment reflection from the reflection prob and not anymore from the sky:

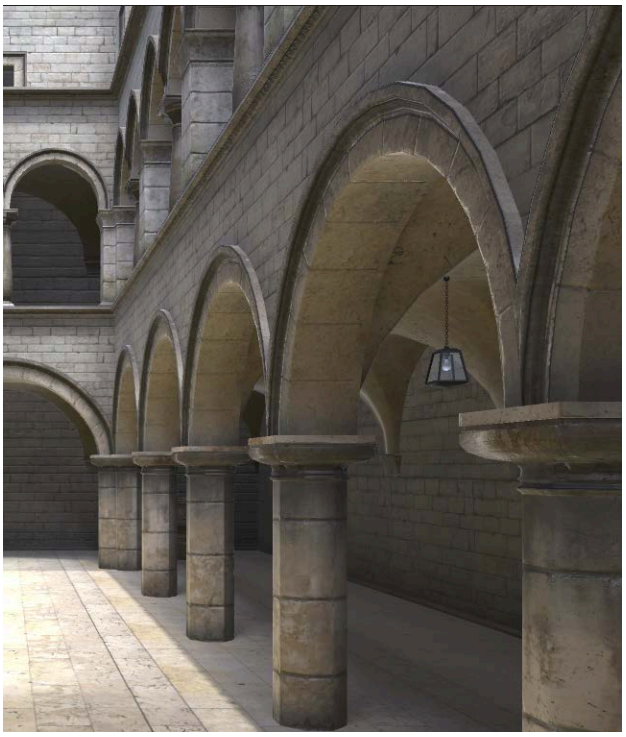


The glossy spheres in the house reflect the environment from the reflection probe. Because the white ground plane also pierces the reflection probe volume, its shadows now have a slightly brownish tint. This is okay in the house, but not okay outside. For very reflective materials, you may increase the resolution in Cubemap Capture Settings.

If you add now a reflection probe to the ground floor and one to the first floor, you can avoid this bluish tint from the skybox:



Sponza atrium with reflection probes



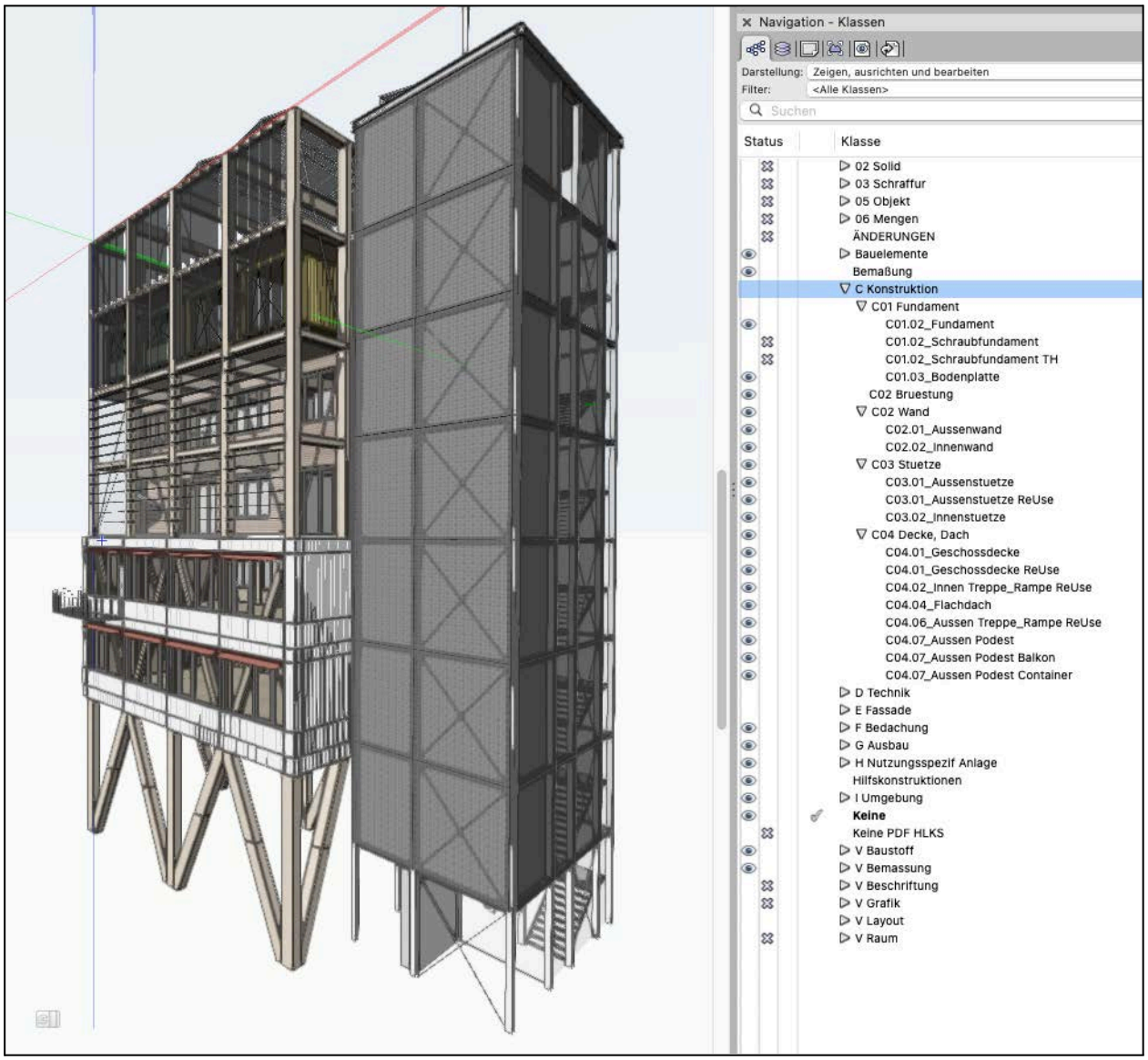
Sponza atrium without reflection probes

7 Case Study: Circular Tower

This chapter describes the VR development from an actual architectural project, from which we kindly received the original CAD files. It is the project SETZKASTEN from the office Inhelder Osterwalder Architekten in Biel, Switzerland. The project won the *Circular Tower* competition organized by TecLab. You can try out the project with the scene *Demos/Ledermann/Circular Tower*.

7.1 CAD Export

The Circular Tower project is developed in Vectorworks Design Suite 2023:



As with other CAD tools, Vectorworks offers multiple possibilities to structure a project. The two main ones are classes and layers. In addition, you can, of course, group objects hierarchically. **There is no single best way to structure a building project**. Often and naturally, a building is structured by:

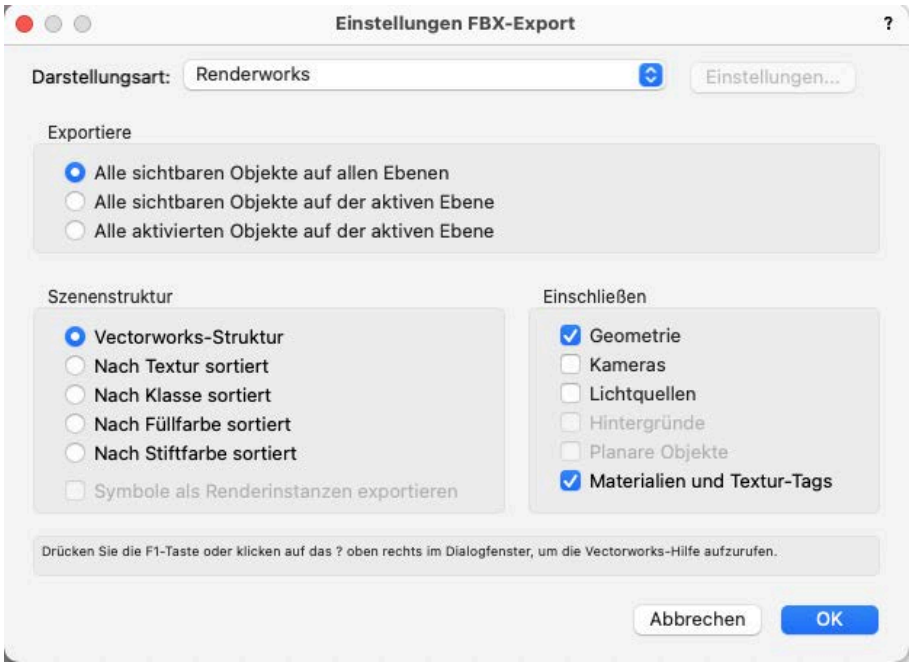
- Floors (from the lowest to the highest)
  - Chronological work types
    - Foundation, supporting structures, non-supporting, interior, installation, etc.

But you can also invert this structure:

- Chronological work types

- Floors (from the lowest to the highest)

Vectorworks offers many export file formats with different restructuring options. The export dialog for the FBX file format looks as follows:



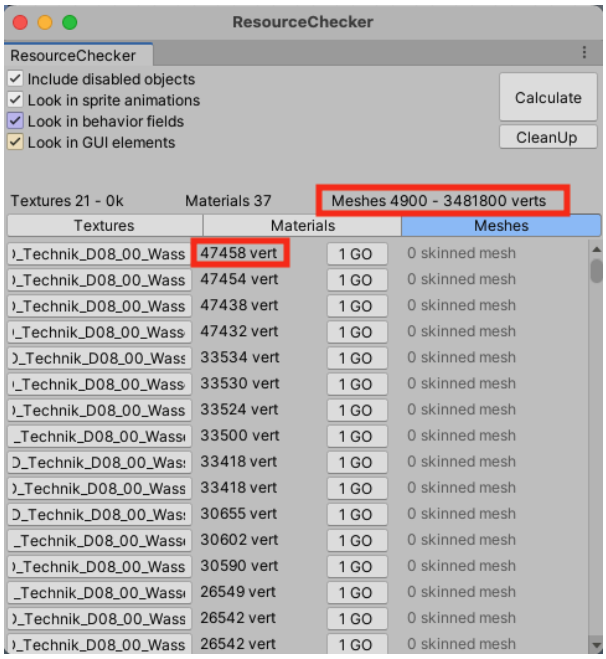
The most important aspects of such an export are:

- **Identifiable Objects**: All the objects must be identifiable by their name in the next tool (e.g., Unity or Blender).
- **Preserved Structure**: The more hierarchical the structure is preserved, the better. The worst export would be a flat list of objects with names like obj00001, obj00002, etc.
- **Full Material Export**: All material definitions, including all texture and color information, must be exported.

7.2 Quick Analysis in Unity

When importing the FBX file directly into Unity and then creating a VR project as described in 4 Creating our first VR App we achieved a framerate of around 10 FPS in the Meta Quest 2 headset. For a comfortable VR experience, this is far too low; we should aim for a framerate above 50 fps to avoid cybersickness. But for a quick project view in VR 10 fps is ok, and you could continue in Unity by adding the teleportation.

By opening the *Resource Checker* tool in Unity (see 8.5.1.1 Draw less), we can see that the project has around two dozen textures and materials, but way too many meshes (4’900) with many vertices (3’481’800). The largest object has 47458 vertices. The number of objects and their resolution do not change if we export the project sorted by texture or Vectorworks class.



What do you want to achieve with the VR visualization?

In the following chapters, you will learn how to optimize this and similar projects so that they will run again with 60 and more fps. But we have to warn you that this optimization will be quite expensive. The steps described in Chapter 9.3 Optimization in Blender and 9.4 Optimization in Unity took us about 3 days of work and are ordered with decreasing importance. So you really have to decide if this optimization is worth it and what you want to achieve with your VR visualization:

- **Design Visualization**: If you only want a quick design check during the project's evolution in your office, 10 FPS is still okay. You can also improve the framerate by using a desktop PC with a cable-bound VR headset in your office.
- **Marketing Visualization**: If you want to gain new customers or visualization is highly important for winning a project competition or public acceptance, then it might be worth going the extra mile in optimization.

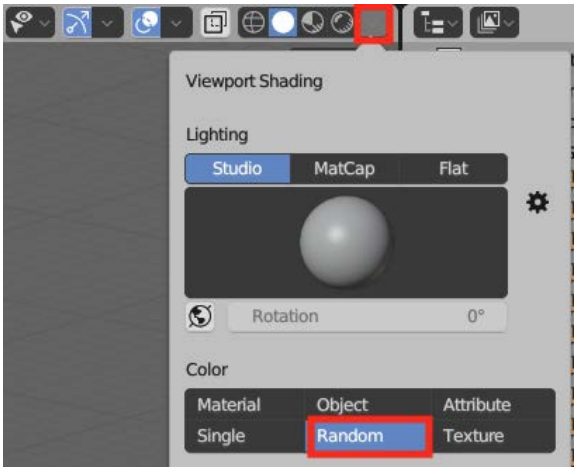
7.3 Optimization in Blender

Ideally, we would optimize our project in the source application, in our case, Vectorworks, so that we would have only two involved software packages (Vectorworks and Unity). Unfortunately, this is often not possible in CAD Tools because real-time rendering at 60 FPS, or even VR, is not their purpose. You can't often influence the resolution of library objects such as sanitary furniture. Objects with round shapes often have a resolution that is far too high. The object at the top of the list above is a toilet.

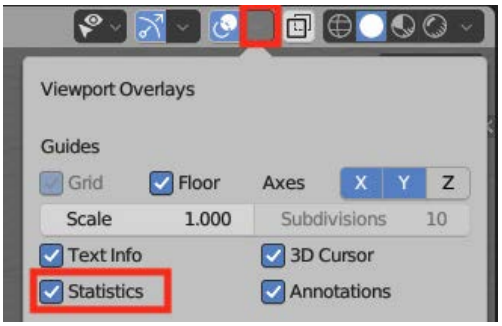
Blender is an ideal software package for these optimizations because it offers compelling, general-purpose modeling features and is entirely free and open source. Blender is the 3D tool with the largest online community, with thousands of YouTube tutorials. Only the official Blender YouTube channel has more than 1 million subscribers. A good starting point here is the Blender 2.80 Fundamentals playlist.

7.3.1 Reduce the Number of Objects

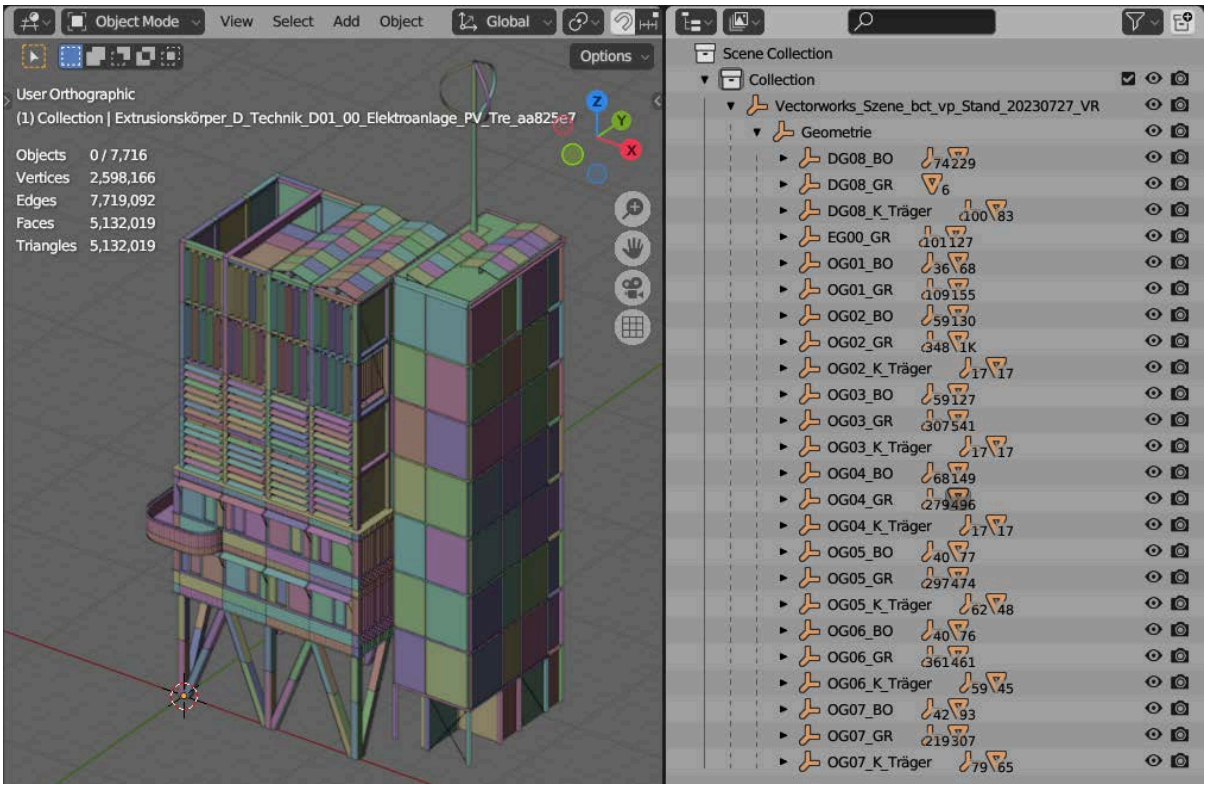
- **Delete the default objects**: After starting Blender, you can delete the default objects by pressing the key A (for select all) and then X (for delete). Blender can be used fastest with keyboard shortcuts.
- **Import the FBX file**: Use the menu *File* > *Import* > *FBX (*.fbx)* to import the FBX file exported from Vectorworks.
- **View in Solid Viewport Shading**: Press the key Z, 6 to change the viewport shading to solid. To see individual objects best, choose random colors for the solids:



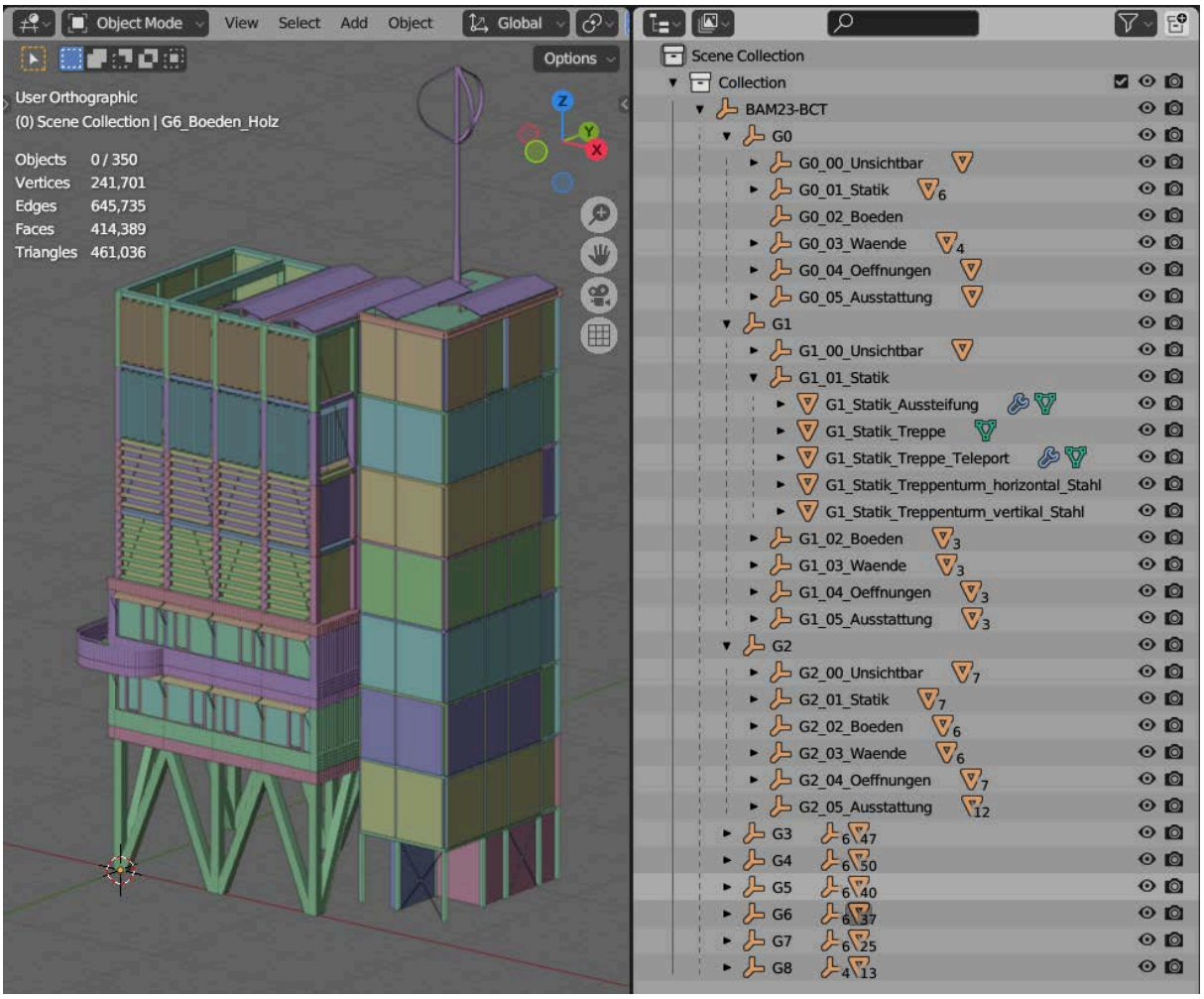
- **Show the statistics**: To see some statistics on the model, choose:



After the import, the statistics in the top-left corner show that the model has 7716 objects (meshes) with more than 5 Mio. triangles. On the right side, you can see the hierarchical scene tree with its groups and subgroups. This is the exported hierarchy from Vectorworks, roughly organized by floors.
Regarding the different colors of the PV solar panels, you can see that they are all separate objects. This causes Unity to do a draw command for each object on the graphics card. This is obviously not very efficient.



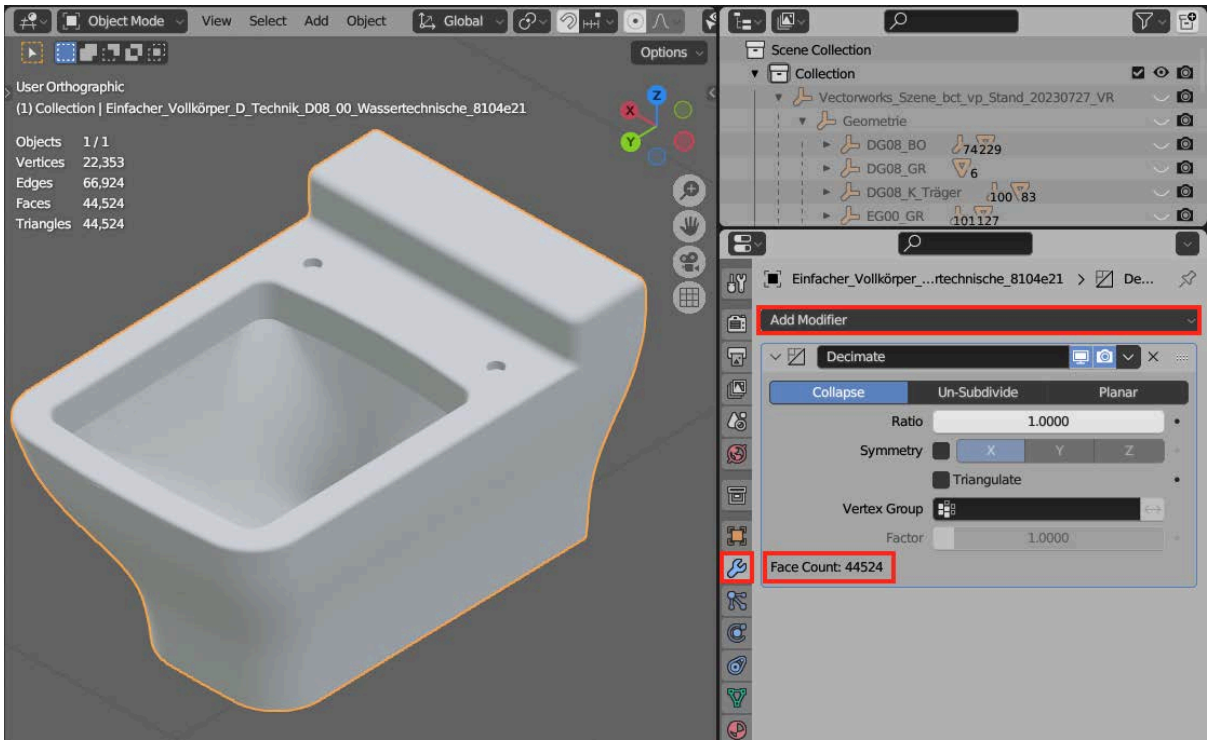
- **Join and rename objects with the same material per floor**: To reduce the number of objects, we can now join multiple objects with the same material into one. You can do that by shift-left-clicking on the objects and then pressing Ctrl-J to join all to the last selected. You should also give the joined object a new name. To still have the floor structure, you can do this grouping floor by floor.
  As you can see in the scenegraph to the right in the next image, I have grouped all objects into 8 floors (*G0-G8*) and sub-grouped them into classic construction types. The invisible subgroup (*G*_00_unsichtbar*) holds all invisible objects that can be later disabled in Unity. With this joining and grouping, only 350 objects remain out of 7’716.



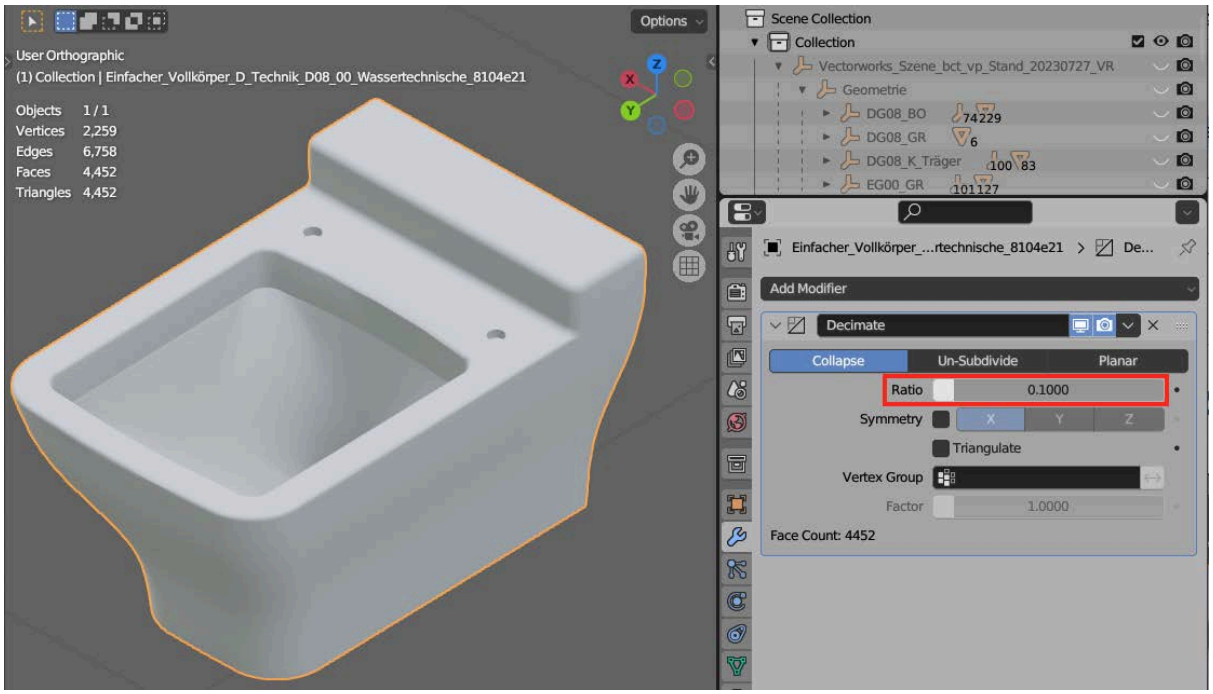
7.3.2 Reduce the Number of Triangles

Only joining objects will not reduce the number of triangles in their meshes. In general, roundish geometries are composed of many triangles used to build these soft shapes. Often, these are library furniture items. However, steel profiles such as the H-columns or just cylindrical objects often have a far too high resolution than is really needed. To reduce the number of triangles, do the following steps:

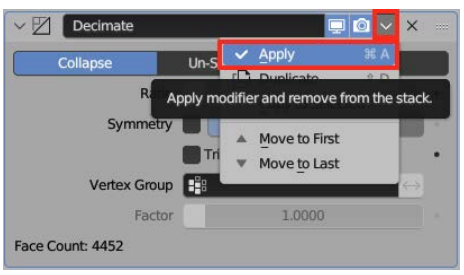
- **Select the suspicious object** and press shift-H to hide all other objects.
- **View the object in the material preview** by pressing Z-2.
- **Add a Decimate Modifier** in the Modifier section of the property window. A modifier is a component that modifies an object visually but doesn't apply the change to the object if you store the project as a blend file. As you can see in the modifier at the bottom left, the toilet has unmodified 44'524 faces (= triangles).



You can now enter 0.1 in the Ratio field to reduce the number of faces to 10%. You can hardly see a difference. Often, you can set ratios between 0.05 and 0.3 without major visual quality loss:

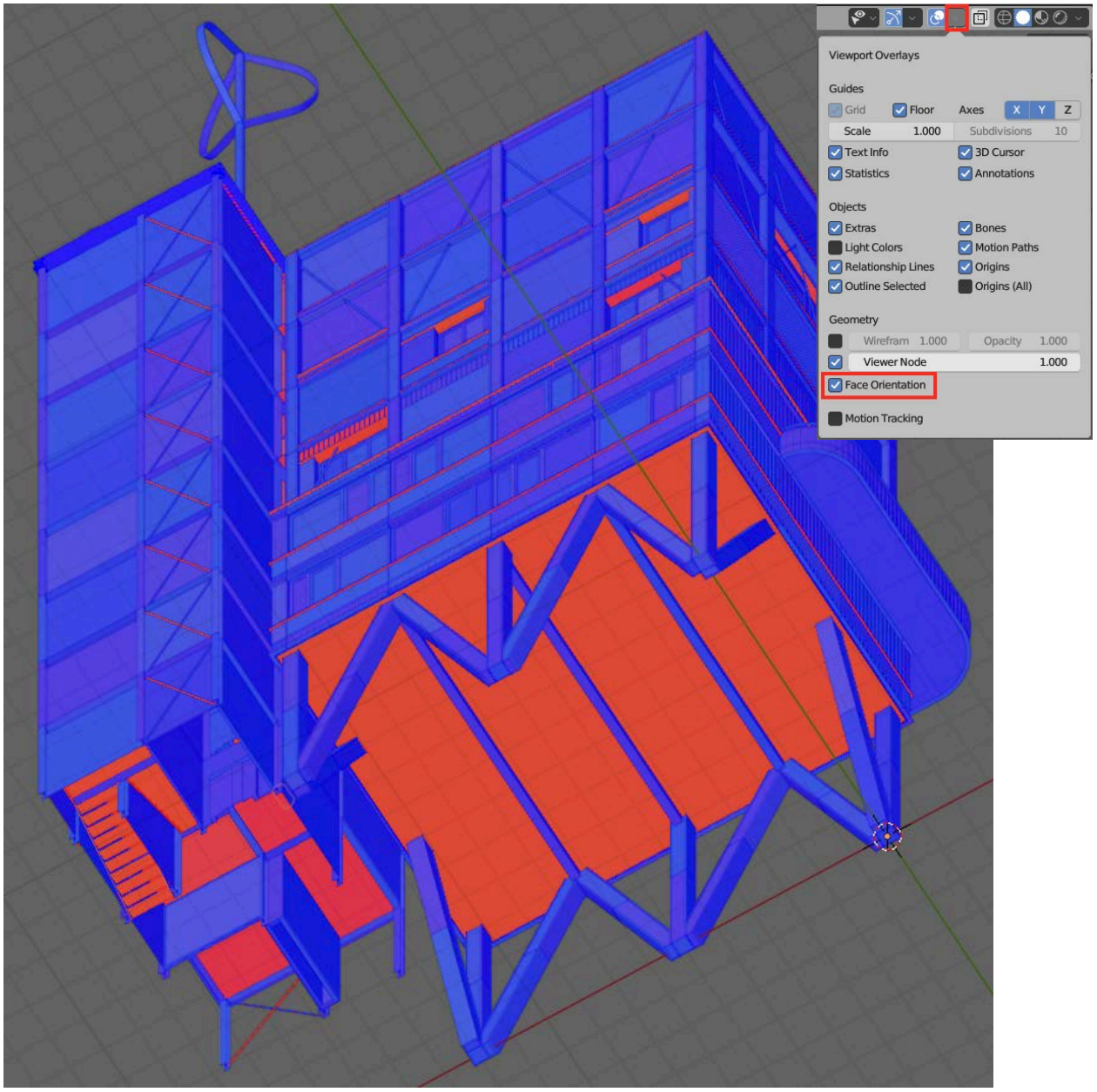


You can also slide with the mouse pointer in the ratio field. To see the toilet's wireframe, switch to wireframe view mode with Z-4. If you want to apply a modification, you can choose *Apply* in the little submenu. You don't have to do this because Unity will automatically do an FBX export from the blend file and apply all modifiers automatically.
With these decimations, the number of triangles was reduced from 5 Mio to below 0.46 Mio triangles.



7.3.3 Fix Backface Errors

Real-time rendering engines optimize performance by drawing only the front faces. Backfaces will not be rendered and will therefore be invisible. For solid objects like a cube or a sphere, you will only see front faces when modeled correctly. Unfortunately, neither Vectorworks nor the FBX export cares about this optimization. To see these problematic back faces in red, you can turn on the *Face Orientation* overlay.



Without fixing these wrong backfaces, we would see, in the above image, through the floor into the first-floor rooms. To fix the objects with wrong backfaces, do the following:

- Select the object and go into *Edit Mode* with the tab key.
- Select all faces of the object with the key A.
- Choose the menu *Mesh* > *Normals* > *Recalculate Outside.*

Be aware that a single triangle has no inside or outside. It always has a front and a backside. If you want to make a plane without volume, you have to create triangles for each side. Volumes with all back faces pointing outward are often created by mirroring an object.

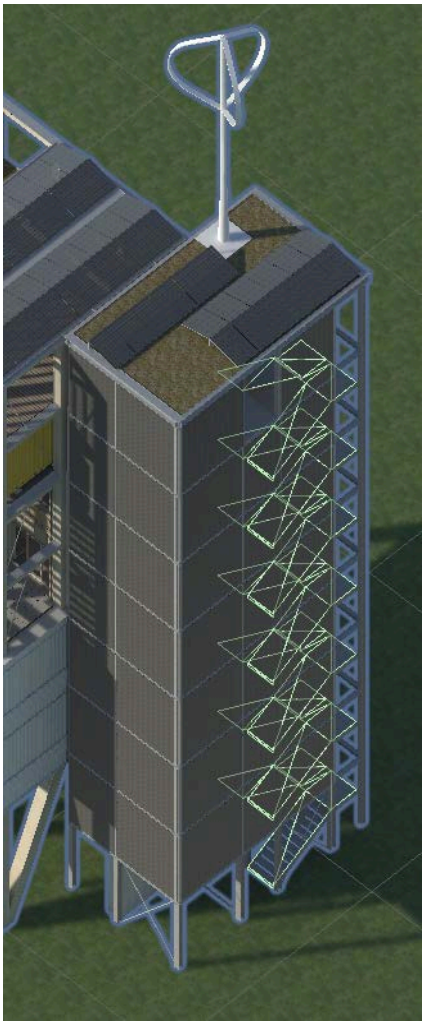
7.4 Optimization in Unity

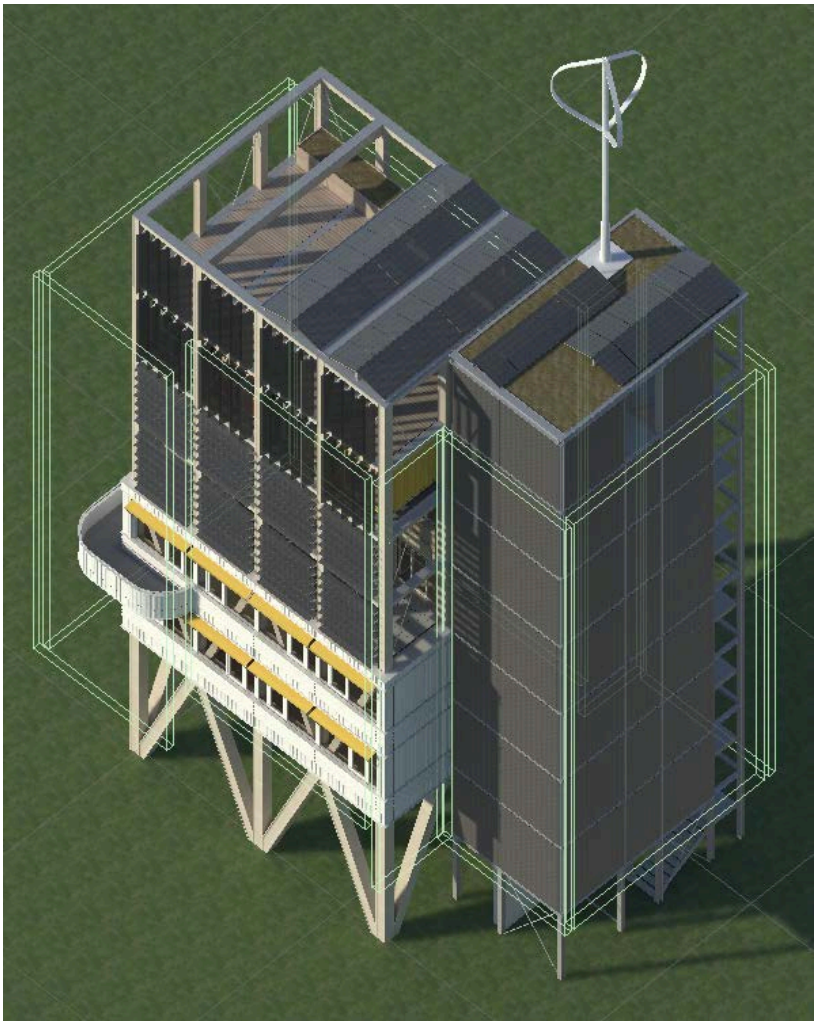
After the optimization, we were able to create a VR app in Unity, as described in 4, Creating our first VR App. You can also check the 4.4 Checklist for Unity VR-App Creation.

7.4.1 Improving Teleportation

For easier teleportation, we added the following:

- **Staircase Teleportation Mesh**: For teleportation on the staircase, we had to add an invisible mesh over it. We did this in Blender, but you could also do this as described in 8.2.2.6 Adding Dedicated Teleportation Mesh.
- **Non-Teleporting Colliders**: Because we can teleport through walls, the user often teleports himself back down to the ground. To avoid this, we surrounded the building with box colliders and set them to the layer Non-Teleport. When the teleportation ray collides with them, it gets red, and the teleportation is refused.





7.4.2 Improving Quality

7.4.2.2 Fix Shiny Materials

The roughness or smoothness (= 1-roughness) is often not imported correctly and is then set to a default value of 0.5. These objects reflect 50% of the light from the bright default skybox, which looks false if, e.g., in a room, the sky is not at all visible. To fix this, you can add a reflection probe as explained in 5.3.2.4 Reflection Probes, or you can just turn down the smoothness to 0.

7.4.2.3 Fix Transparency and Backfacing Errors

Transparent objects are rendered in real time. For correct transparency, engines render all opaque (= non-transparent) objects, and in a second pass, all transparent objects sorted from back to front in the viewing direction. Be aware that objects with transparent textures (= textures with an alpha channel) also belong to the transparent objects. In the image below, the fence wire should appear in front of the solar panel, but it doesn't. The fence wire and the solar panels have transparent parts in their textures. The error in the left image is caused by combining all solar panels on the same floor into a single object and all fence wires into a single object. To solve the problem, you have to split up these objects so that the sorting in the view direction works correctly again.

Back-to-front sorting is only performed at the object level. If you have multiple transparent polygons in one object, such as the different leaves of the plant in the following image, you can not solve this problem anymore. The leaf marked with a red circle should appear behind the leaf below it. It doesn't because its polygons were rendered after the polygons of the lower leaves.

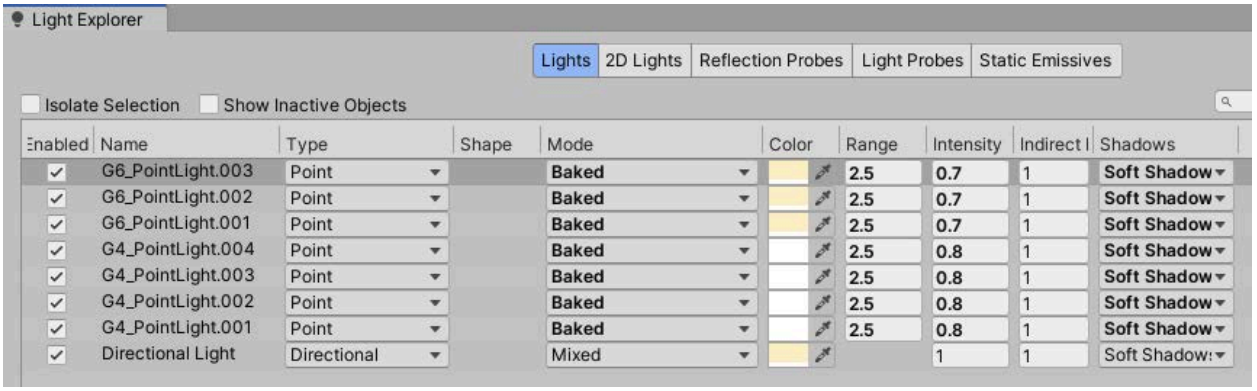
As explained in 9.2.3 Fix Backface Errors, objects should always be modeled as volumes with all front-facing polygons. If objects have only single faces (triangles), such as the

leaves in the upper image, we will not see the backside because they aren't rendered. Leaves seen from the back (blue circle) are just invisible.

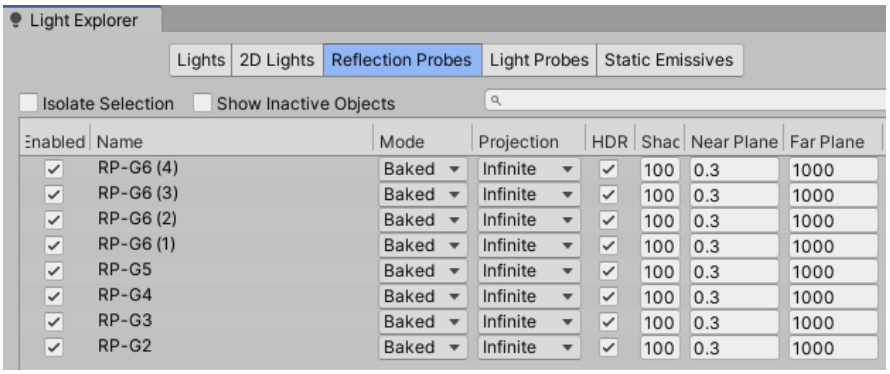
7.4.2.4 Adding additional Light Source and Reflection Probes

Light sources in exported file formats rarely get imported correctly. A viable option is to import only meshes without cameras or light sources, and to start with the default directional light already present in the Unity scene. Because we don't use the expensive feature of *High Dynamic Range (HDR)* on mobile devices, our indoor spaces often look dark. HDR is a post-processing operation that increases the exposure of an image that is too dark and mimics the iris adaptation of our eyes.

The only workaround in the built-in, universal render pipeline is to addlights in dark rooms. We did this in our project in the office rooms on the 4th floor and in the containers on the 6th floor. Unity offers a nice overview window for all light sources in a project called *Light Explorer*:



Reflection Probes control reflections on glossy materials, as explained in 5.3.2.4. Because many materials have a default shininess of 0.5, we also get blueish sky reflections indoors, where the sky is not visible at all. To avoid this, we either lower the shininess or place a reflection probe in the specific room. The *Light Explorer* also gives you an overview of them:



7.4.3 Improving Performance in Unity

So far, we have optimized the model in Blender for the number of objects and triangles, achieving an almost optimal performance of 60-70 fps. But we can, of course, add more geometry to the project with more furniture or equipment.

The overall goal of performance optimization in Unity is to show only what is visible. Unity automatically renders only what is visible to the user (aka view frustum culling). But if you look from the ground floor up to the entire building, as in the following image, Unity will render all the building's objects, even if you can't see them all. It will render all internal objects that are not visible from the ground.

7.4.3.1 Unity Occlusion Culling

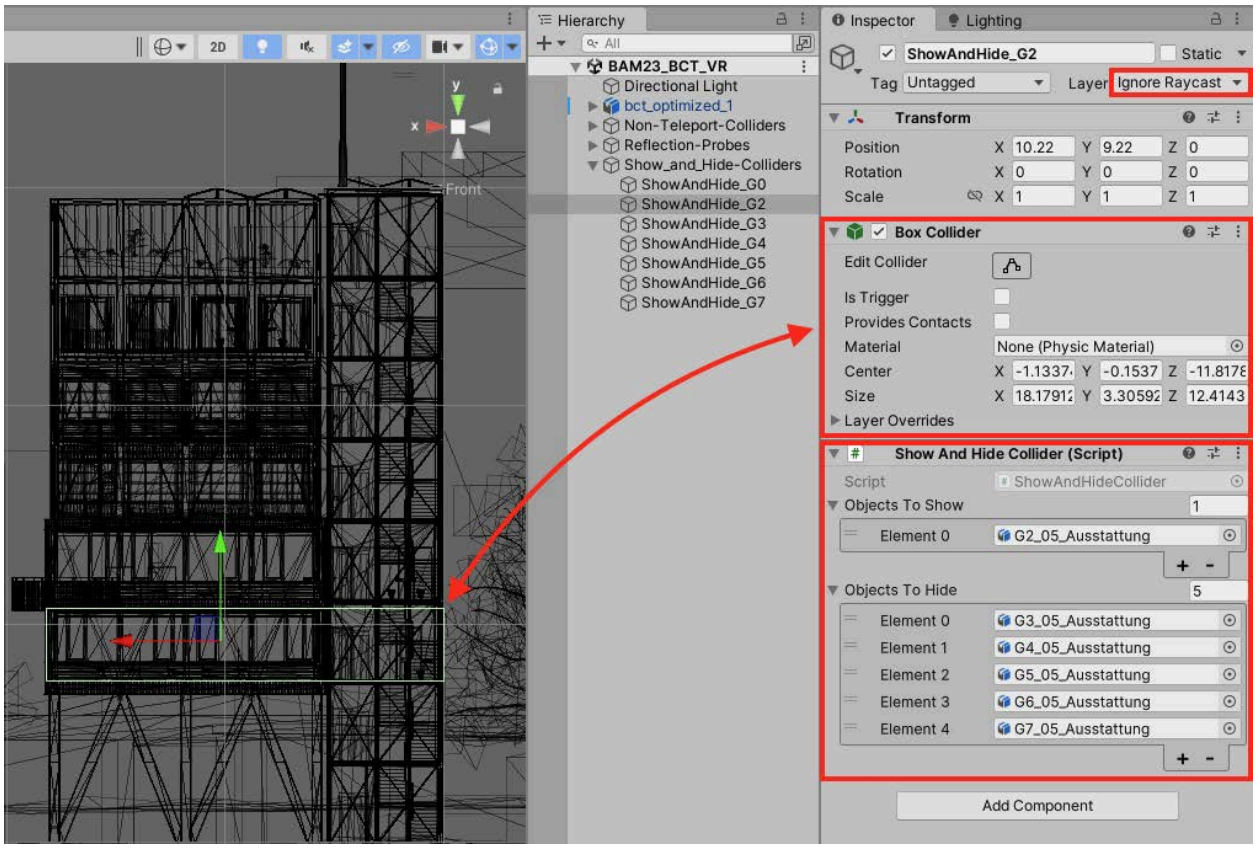
This optimization is built into Unity and allows you to generate an occlusion culling acceleration structure. It is well documented, but its parameters are not so easy to adjust optimally. See the online documentation for more information.

7.4.3.2 Show and Hide Collider

The following method is a simple script that allows you to show and hide objects when the user enters a box collider.

- Surround the space with the box collider where you want to show and hide something. Set its layer to *Ignore Raycast*.
- Add the script *ShowAndHideCollider.cs* from the scripts folder.
- Click the +-sign to add a new element to the lists *Objects to Show* and *Objects to Hide*.
- Drag the objects or group objects to these lists.

In the following image, you can see that we want to show the equipment on the second floor (*G2_05_Ausstattung*) when we enter it, and hide all equipment from the other floors that we can not see. When we exit the surrounding box collider, the mentioned objects get enabled or disabled oppositely.

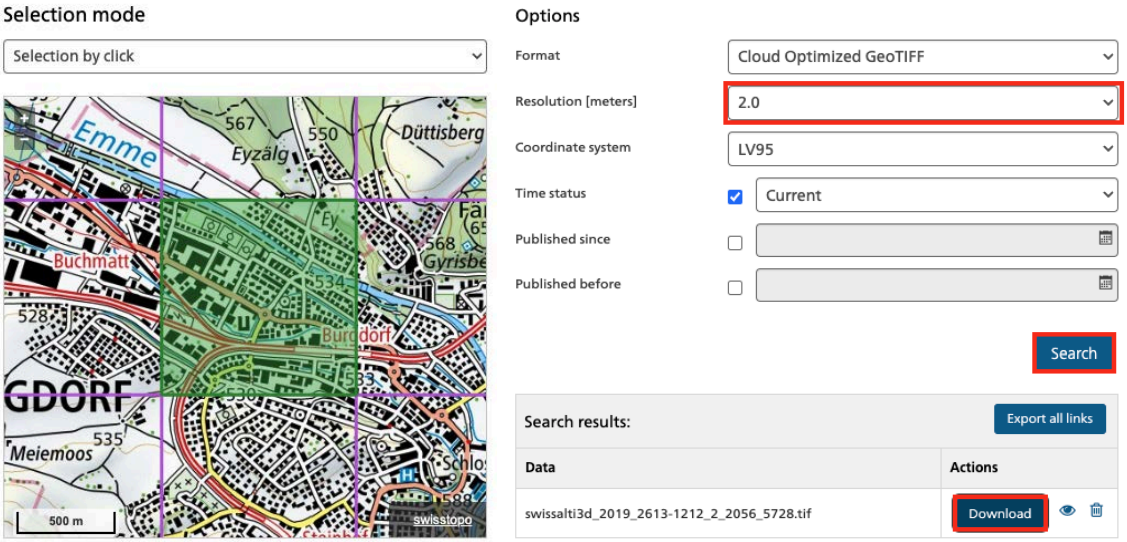


Such an improvement is especially recommendable when you want to add far more furniture and details to the floor's equipment (*G*_05_Ausstattung* group node). As we have seen in chapter 9.3.2, Reduce the Number of Triangles, these objects often have high resolution and can significantly degrade performance.

7.5 Adding Environment

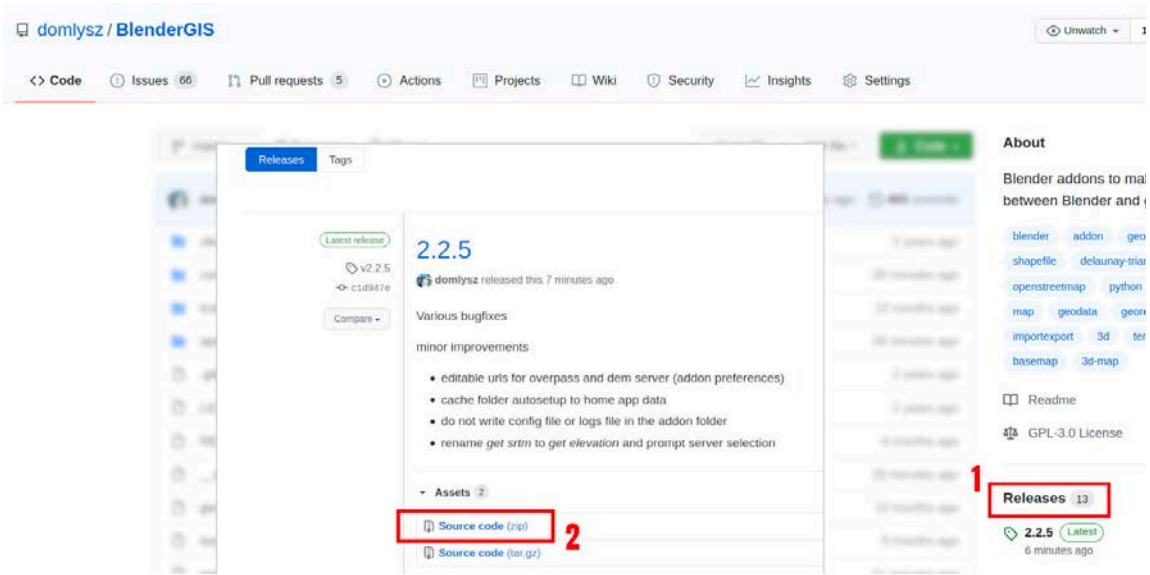
7.5.1 Adding Terrain from Swisstopo

- **Download the terrain elevation file (SwissAlti3D) from swisstopo**:
  - Go to https://www.swisstopo.admin.ch/en/geodata/height/alti3d.html
  - Scroll down and zoom in on the location of your project.
    Select the tile of 1 x 1 km terrain, choose the 2m resolution, and press *Search*:

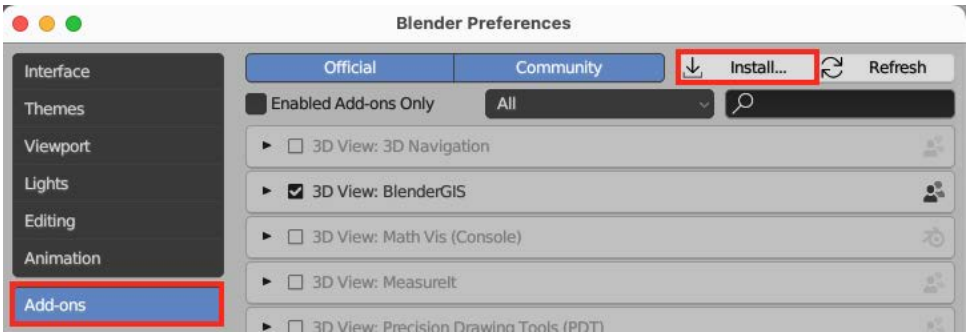


Press *Download* to load the GeoTIFF file (extension *tif* or *tiff*) to your computer.

- **Install BlenderGIS addon**: BlenderGIS is a very powerful *Geographic Information System (GIS)* addon for Blender.
  - Download the source code zip file from https://github.com/domlysz/BlenderGIS:

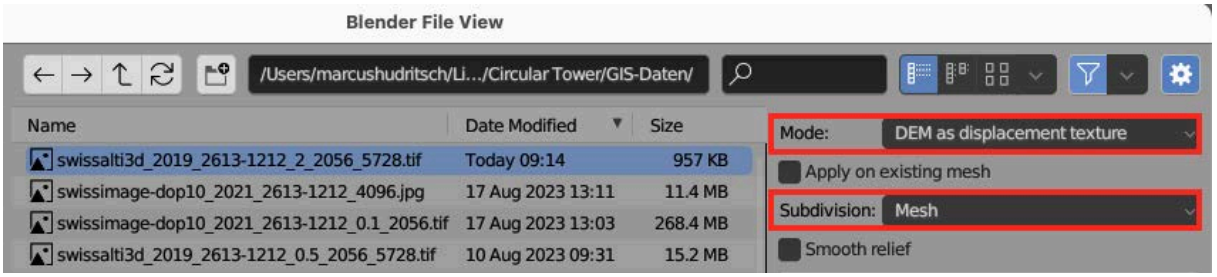


  .
  - Open Blender and install the addon by choosing *Edit > Preferences: Addons > Install* and select the downloaded zip file:



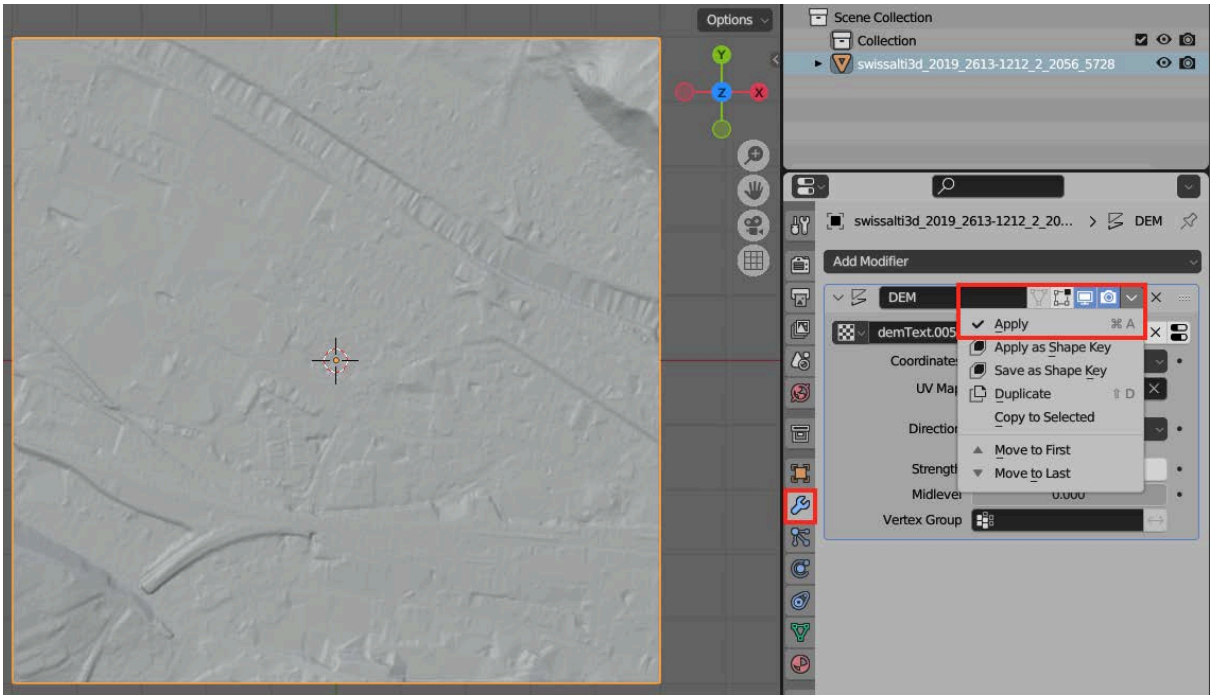
After the installation, you will have a new menu named *GIS.*

- **Import the terrain file into Blender**
  by choosing *GIS > Import > Georeferenced Raster (tif, jpeg, jp2, png)* and selecting the mode *DEM as displacement texture* option and the *Mesh* subdivision method.



Press *Import Georaster* to import the data. The import takes a while because it reads 500x500 heights from the 1km patch. With 0.5m resolution, it would be 2000 x 2000 height values.

- **Apply DEM Modifier**: To apply the imported height values to a mesh, select the object and press *Apply* in the Modifier section:

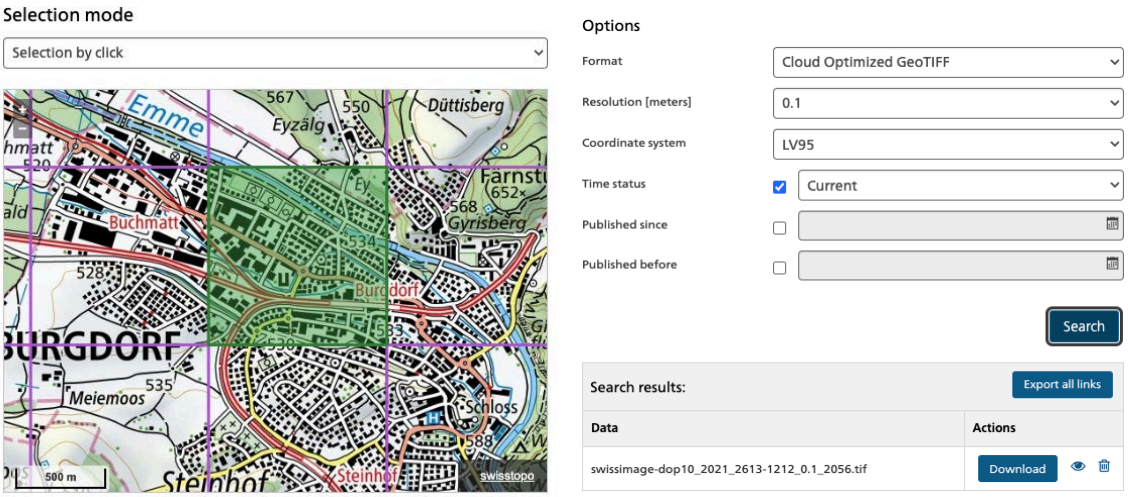


The terrain, at 2m resolution, is still far too heavy for Unity. Don't decimate it now. We will do that after we apply the orthophoto as a texture.

7.5.2 Adding Orthophoto from Swisstopo

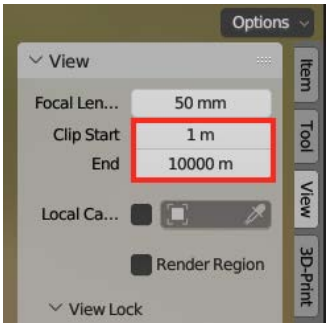
- **Download the orthophoto from Swisstopo**:

- Go to the Swisstopo page for swissImage10.
- Scroll down and zoom in on the location of your project.
  Select the tile of orthophoto, choose the 0.1m resolution, and press *Search*

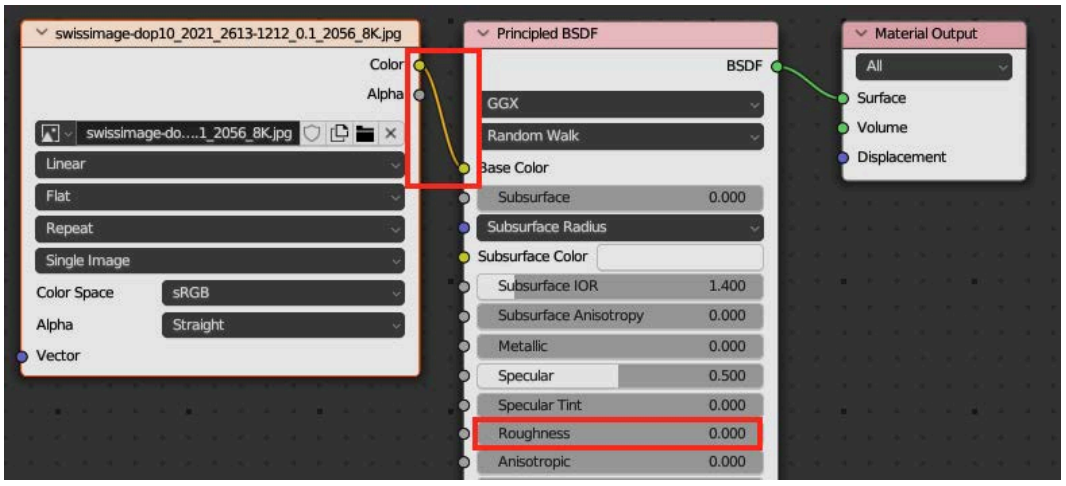


- Press *Download* to download the 10’000 x 10’000 pixel large GeoTiff file.

- **Scale down GeoTiff File**: The largest texture size supported by GPUs is 8K x 8K. We therefore have to downscale the orthophoto from 10’000 x 10’000 to 8’192 x 8’192 pixels. Save the image file as JPG with a compression quality of 95.
- **Create Material for Terrain with Orthophoto texture**:
  - **New Material**: With the terrain object selected, click on *New* in the *Material* tab of the properties to create a new material for the terrain and name it *Orthophoto*.
  - **Switch to the *Shading* workspace**. You may not be able to see the terrain anymore because the camera view settings are set to narrow by default. Open the right-side panel with the key N and adjust the *View* settings as follows:



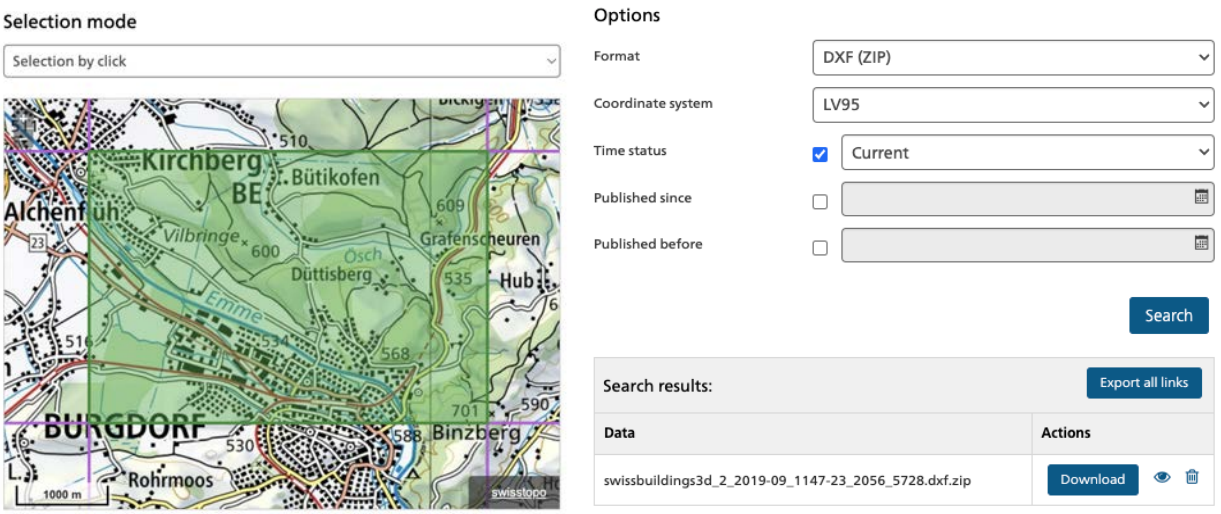
  - **Add the Orthofoto**:
    - Drag the 8K orthophoto into the *Shader Editor* and connect its *Color* output pin with the *Base Color* input pin of the *Principled BSDF* block.
    - Slide down the *Roughness* property to 0.



The 3D terrain in Blender was created with the 2m SwissAlti3D elevation dataset and the matching orthophoto with 8k resolution. We will crop the terrain and decimate its resolution in a later step.

7.5.3 Adding 3D Houses from Swisstopo

- Go to the Swisstopo page swissBUILDINGS 2.0.
- Scroll down, select your patch on the map, press *Search,* and click *Download* to download the zipped DXF file:

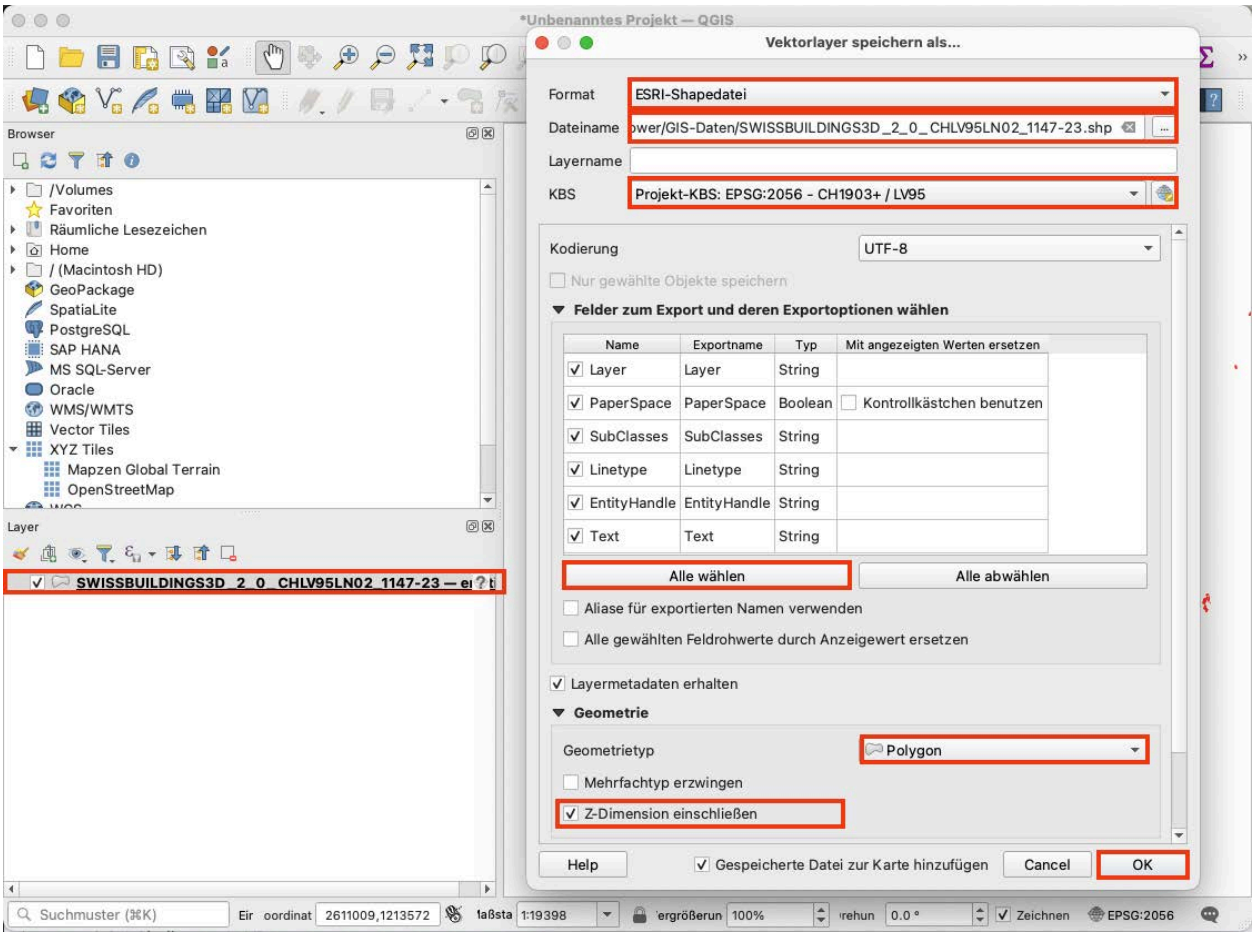


The import into Blender is now a real pain:

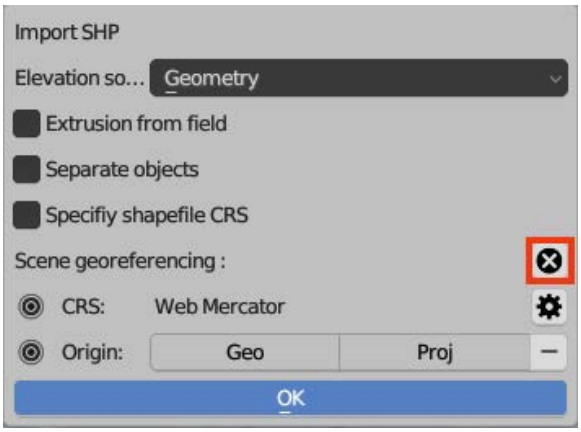
- The Blender plugin for the Autodesk DXF format does not work. This is most probably due to the hate relationship between the most commercial CAD company, *Autodesk,* and the most open and free software, *Blender*. You can ask swisstopo for other formats, but this service is not free!
- FreeCAD, another free CAD software, also crashes when importing the downloaded swissBUILDINGS DXF file.
- QGIS, a free, open-source, and widely used GIS (Geographic Information System) software, can import the file and export it as an ESRI shapefile, which Blender can import with the BlenderGIS addon.
- Vectorworks can import the swissBUILDINGS DXF file and export it as fbx.

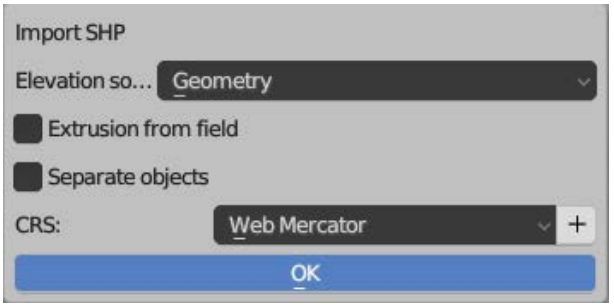
- **Download and install QGIS** from qgis.org/de/site/forusers/download.html
- Open QGIS and drag the unzipped swissBUILDINGS dxf file into it.
  An Import-Dialog wird angezeigt, und du kannst auf Layer hinzufügen (Add Layer) klicken.
- **Export Layer as ESRI-Shape file**:

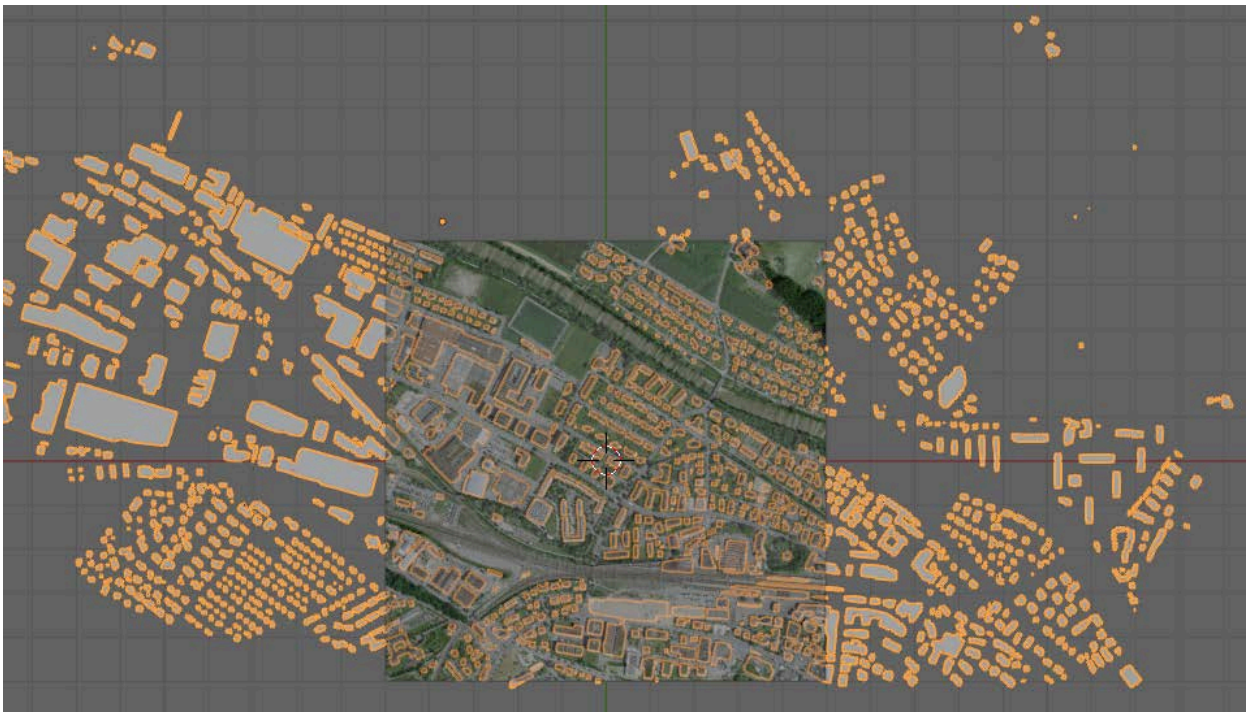
- Right-click on the right side of the layer name and choose *Export > Objekte speichern als …*
- In the export dialogue, choose *ESRI-Shapefile* as *Format*.
- Set location and filename. I chose the same name and location as the dxf file but with the extension *shp*.
- Choose the Swiss coordinate system (KBS: Koordinatenbezugssystem) *EPSG:2056 -CH1903+ /LV95*
- Choose "Alle wählen" to select all fields to export
- Under *Geometry,* choose *Polygon* as type and include the *Z-Dimension*.
- Click *OK* to export.



- **Import the ESRI-Shape file into Blender** by choosing the menu GIS > Import > Shapefile (.shp): Click first the x-button to deselect *Scene georeferencing* and then click OK with *Web Mercator as CRS*:







- **Adjust the height of the Terrain**: The terrain's height is set to its actual elevation in meters above sea level. The imported houses, on the other hand …

7.5.4 Adding 3D Bushes and Trees

Adding vegetation to a 3D scene can quickly degrade the performance of real-time applications. Plants have an incredible degree of detail when viewed up close, but are also often seen in the far background.

There are several possibilities to render plants in 3D:

- **3D models without transparency**:
  - **Abstract**: This simplified method, aka low-poly style, is often used in games where the rendering power is prioritized for smooth animations rather than for a higher degree of static details.
  - **Detailed**: This type of plant mode, in which every leaf is modeled with triangles, is only used in offline renderings because real-time rendering is not possible due to the high memory and rendering costs.

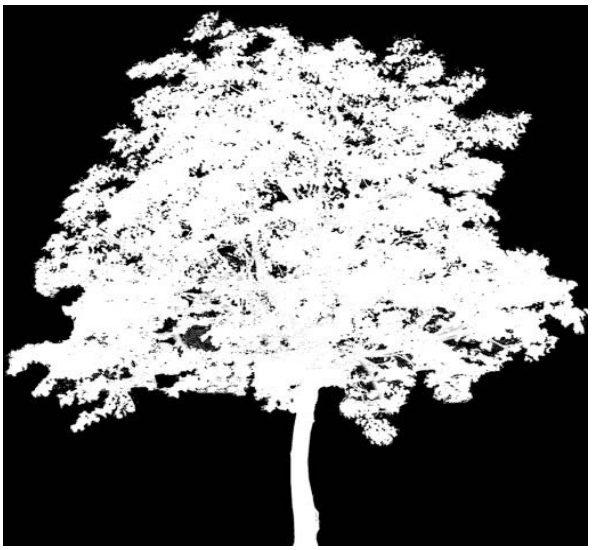
Abstract low-poly tree model

Detailed tree model with ten thousand triangles

- **3D models with transparency**: Texture images with an alpha channel can add a lot of visual details with only a few polygons. The only disadvantage of transparent objects is

that their rendering must occur in a separate pass after opaque (non-transparent) objects, and their rendering order must be sorted from back to front for correct blending. This sorting is done on the object level only for performance reasons, not at the triangle level.

A tree out of 4 objects with transparent textures.



The texture's alpha channel stores the transparency.

Multiple trees rendered with sorting.

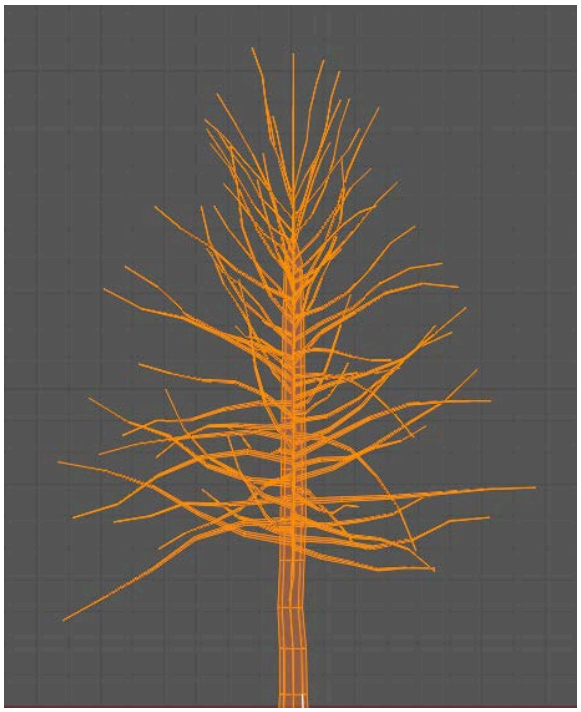
Multiple trees rendered, sorted from back to front.

Front and back face: For performance reasons, real-time engines render only the front faces of objects. The front side of a triangle is defined by the winding order of its three vertices. Most of the time, this order is counter-clockwise for the front face.

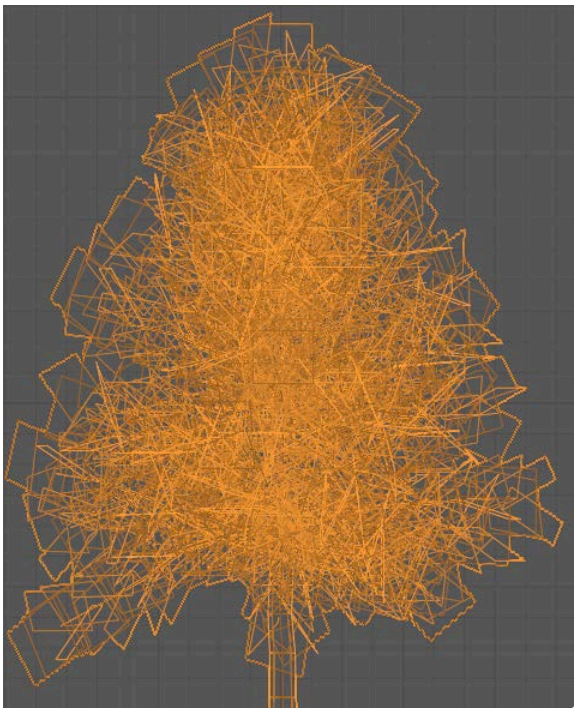
If you draw a transparent texture on a single quad (two triangles) now, you will only see it from the front. To see it also from the back, you must create two quads at the same place. One to the front and one to the back.

- **Static and detailed**: A detailed tree consists of the tree stem and branches with an opaque bark material and many quads with transparent leaf textures. Like this, you can create 3D trees with a fairly decent amount of geometry, where you can walk around and get a good impression of their dimensions. But a single tree with many leaf quads is not negligible and will cost its render time.
- **Static billboard**: If you only want to show a tree or bush in the background and don't want to walk around the plants, you can display it with only view quads. This type is also known as *billboarding* and is almost free in terms of rendering costs.
- **Dynamic Levels of Detail (LOD)**: For desktop 3D applications, there are dynamic LOD tree systems, such as SpeedTree, that show a tree in detail when you are close up and as a billboard when you are far away. This is impressive on a single monitor

but strange in a VR headset because the billboards get automatically rotated toward the viewer.



The tree stem and branches with opaque material.

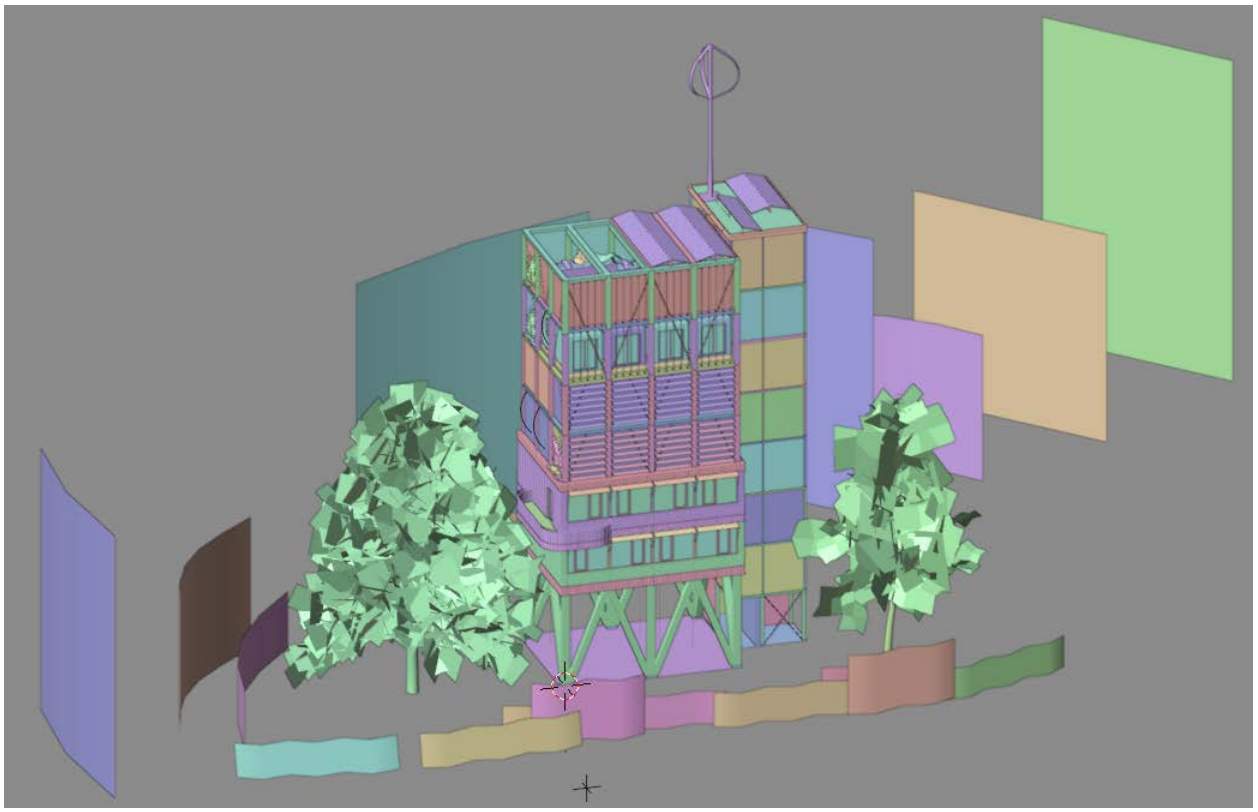


The single-sided leaf quads have a transparent texture.

The entire tree has around 6200 triangles.

All leaf quads use the same transparent texture.

Because the Circular Tower is surrounded by many trees, we decided on a compromise: two detailed trees with transparent leaves close to the tower, and simple bush and tree billboards farther away.



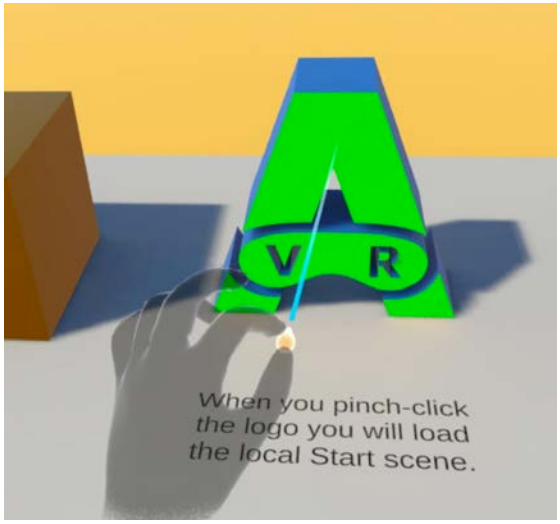
8 Adding Functionality

So far, we can only teleport in our scenes. Most of the following add-ons are demonstrated within the *Example Scene* within the archivr-viewer-unity6 project. The scene file is in the *Assets/Scenes* folder.

Doing a Pinch-Click

Many of the following functionalities are executed when you gaze at the objects where you added a certain functionality. These are normal 3D objects that get a new highlight color (e.g., green) when you gaze at them with your central viewing direction. To execute the action, you do a so-called pinch-click:

- You first close the thumb and index finger until you get a straight ray.
- To click, you then close the thumb and index finger.



8.1 Adding Links

A webpage can have links to words or images, and if you click on them, your browser will load the linked pages or images. That is the essence of HTTP (Hyper Text Transfer Protocol), which allows you to link different web pages.

Similarly, you can place links in your 3D scene, and if you click on them with the pinch gesture, the viewer will load the new scene. There are three types of links:

- Links to another remote scene from ArchiVR.
- Links to another local scene that are built within the viewer app.
- Links to another position within the same scene.

How do we see what a link is? In HTML, links are usually underlined words. In our scene, links get colored when you gaze at them. You must enable the *Use Gaze Interaction* option in the viewer settings.

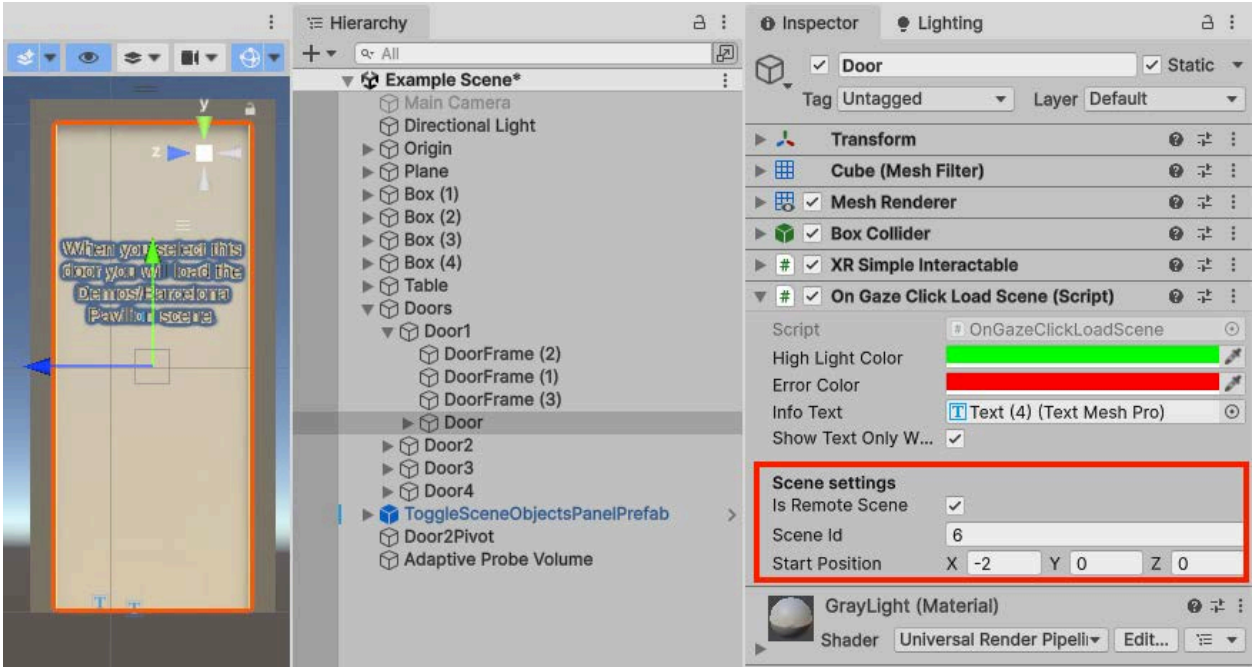
8.1.1 Link for Loading a Local or Remote Scene

To create a link on a selected object that loads another scene, add a collider (box, sphere, or mesh collider) and the scripts *Shared/Scripts/OnGazeClickLoadScene*. You can define the following parameter:

- **High Light Color**: The color that is overlaid when you are gazing at the object, and the link is available.

- **Error Color**: The color that is overlaid when you are gazing at the object, and the link is not available.
- **Info Text**: If you add a Text object, this text will be shown when gazed at. You can add a text object by using the menu *GameObject > 3D Object > Text - TextMeshPro*. This text object will be in the scene and can be transformed like any other object.
- **Scene Settings:**
  - **Is Remote Scene**: Check this option if the scene is from the ArchiVR platform.
  - **Scene Id**: If the scene is from ArchiVR, you can find the ID on the Archi-VR Website, in the scene's detail view, as the last part of the URL. For example, 6 for the scene at https://archi-vr.ti.bfh.ch/Scene/Index/**6**
    If the scene is local (it comes with the viewer), the Scene Id is the number from the Scene List:

  - **Start Position**: If you want to start at a specific position in the linked scene, enter it here. Otherwise, the default starting position is [0, 0, 0].



8.1.1 Link for Jumping to Another Position

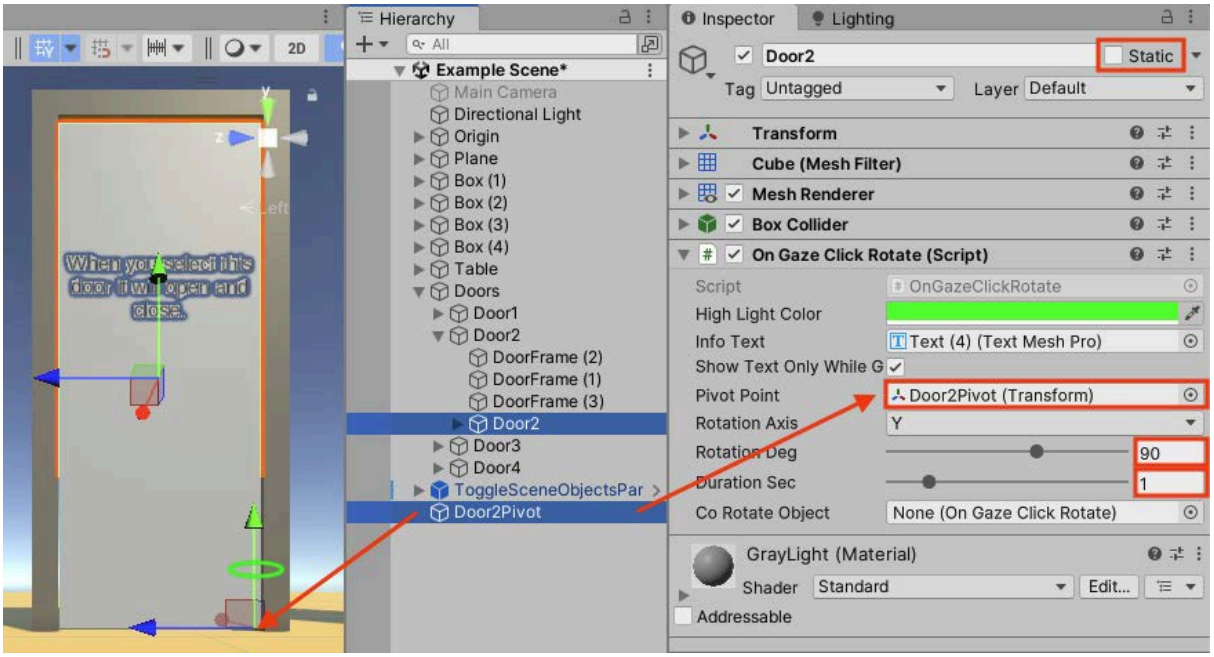
To create a link that makes you jump to another place in the same scene, add a collider (box, sphere, or mesh collider) and the scripts *Shared/Scripts/OnGazeClickJumpTo*. You can define the following parameter:

- **High Light Color**: The color that is overlaid when you are gazing at the object, and the link is available.
- **Error Color**: The color that is overlaid when you are gazing at the object, and the link is not available.
- **Info Text**: If you add a Text GameObject, this text will be shown.
- **Jump to Target**: Drag a game object into this field. You will jump to its position.

8.2 Adding Object Transforms

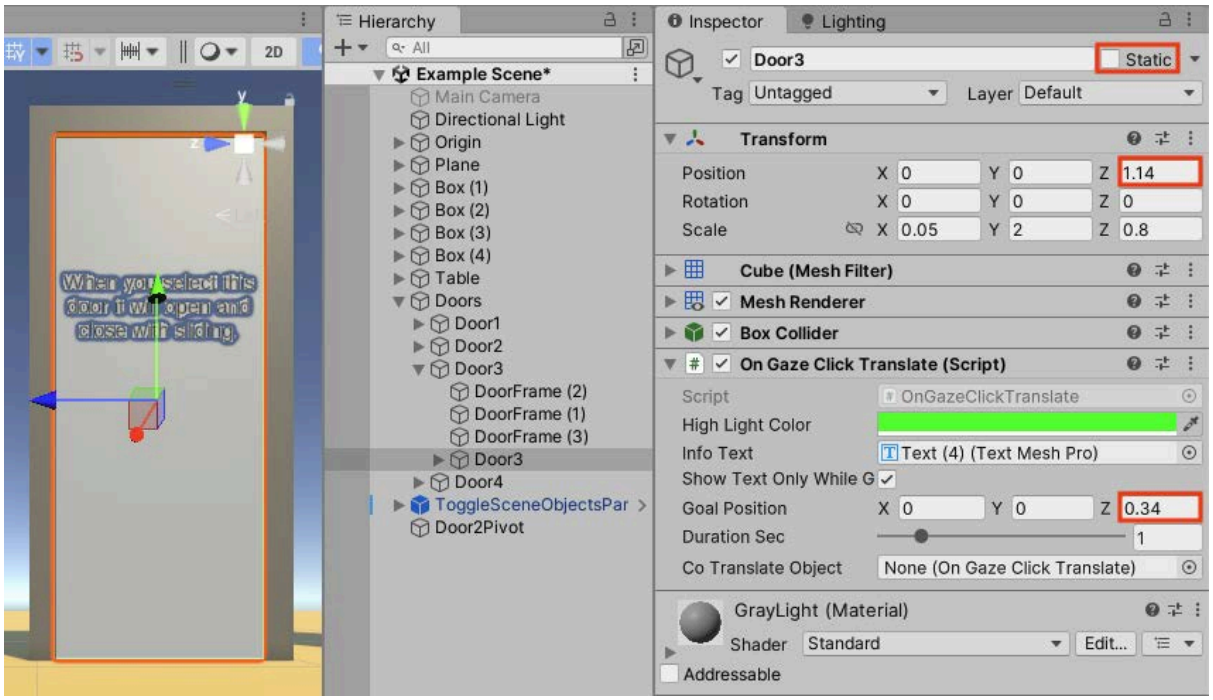
By adding one of the scripts *OnGazeClickRotate*, *OnGazeClickTranslate,* or *OnGazeClickScale,* you can rotate and translate (move) objects when you gaze and click at them.

- **To rotate** an object, select it and add the script *OnGazeClickRotate*:
  - Uncheck the *Static* checkbox; otherwise, the object can not rotate.
  - *Pivot Point*: If no pivot point is set, the object will rotate around its origin, which is at the center of the door in the image below. If you can't influence the object's origin, you can add an empty game object with *GameObject > Create Empty* as the rotation point (*Door2Pivot* in the image below).
  - *RotationAxis*: Rotation axis of the object or the additional pivot point.
  - *Rotation Deg*: Rotation amount in degrees.
  - *Duration Sec*: Rotation duration in seconds.
  - *Co-Rotation Object*: To open both doors at once, drag the other door part into this field.



You can find an example rotation on the second door from the left in the *Example Scene*.

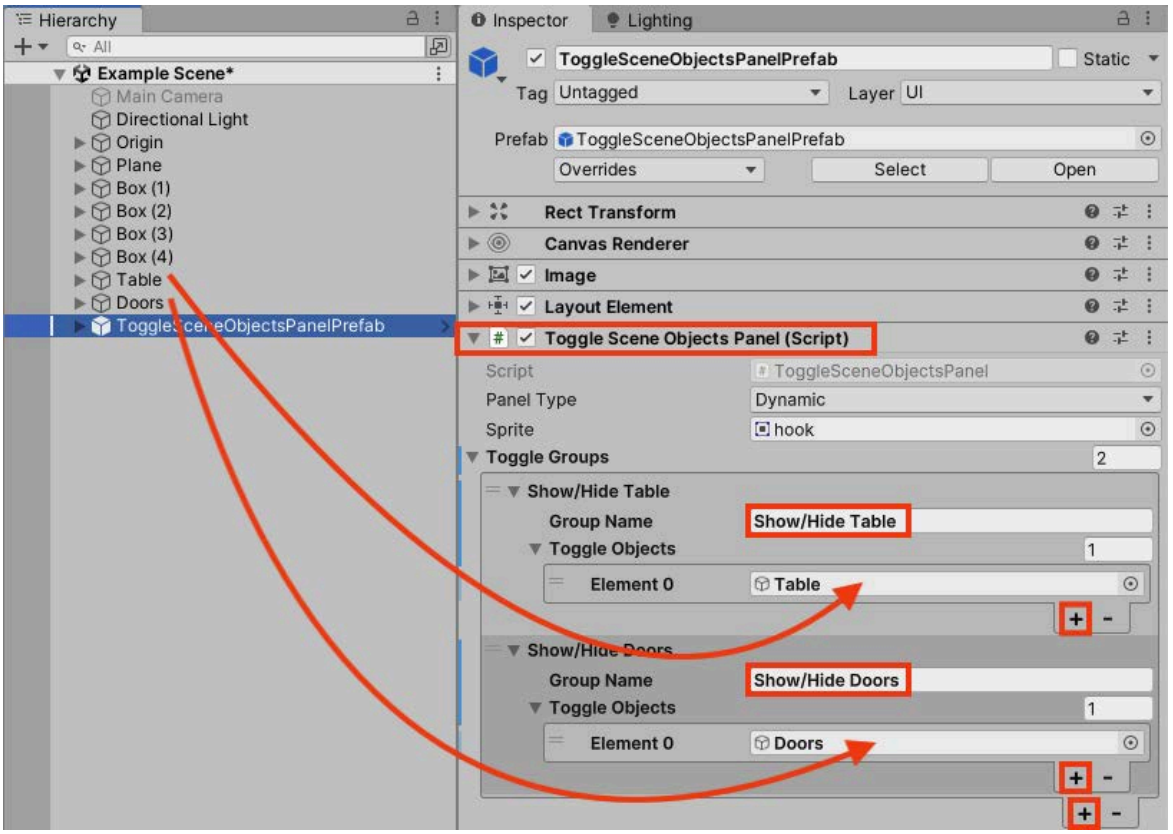
- **To translate** an object, select it and add the script *OnGazeClickTranslate*.
  - Uncheck the *Static* checkbox; otherwise, the object cannot move.
  - *Goal Position*: The goal position is in meters. The example door is 0.8 wide in its z-dimension and 1.14 wide in the z-direction. If we want to move it 0.8 to the right, the goal position must be 1.14 - 0.80 = 0.34.
  - *Duration Sec*: Movement duration in seconds.
  - *Co-Translate Object*: To open both doors at once, drag the other door part into this field.



You can find an example translation on the third door from the left in the *Example Scene*.

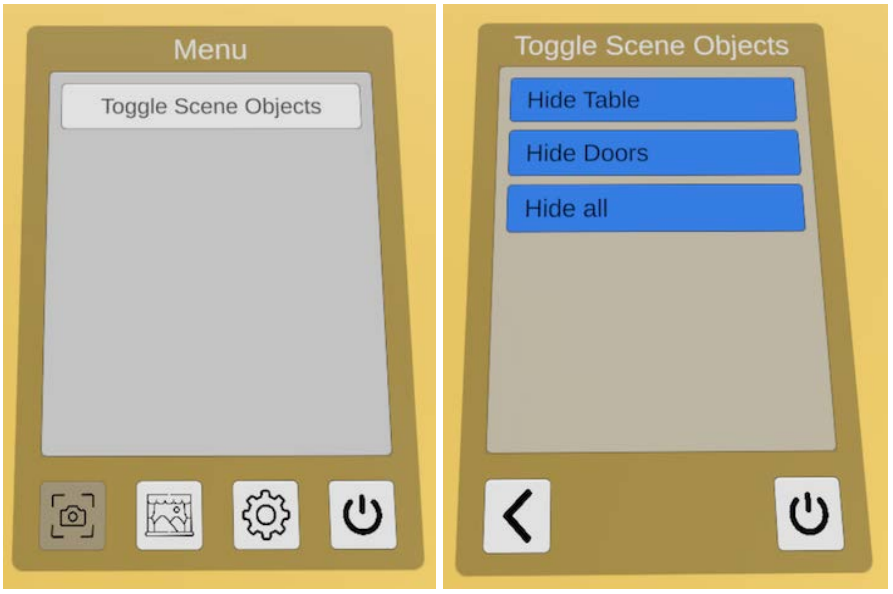
8.3 Show and Hide Objects

- **Add the Menu prefab to the scene**: *Assets* > *_Viewer* > *Prefabs* > *ToggleSceneObjectsPanelPrefab.*
- **Create toggle groups**: Select the *ToggleSceneObjectsPanelPrefab* in the hierarchy:
  - Look for the script *Toggle Scene Objects Panel* in the inspector
  - Expand the section called *Toggle Groups*.
  - Add as many *Toggle Groups* as you need with the [+] button. Give each group a name and add all required GameObjects to the *Toggle Objects* list by adding new entries with the [+] button of the *Toggle Objects* list.



- **Save and upload your scene**.

The new menu will be added to the first panel as a button when entering your scene. Examples of this script are in the *Example Scene*, *Barcelona Pavilion,* and *Circular Tower* scene.

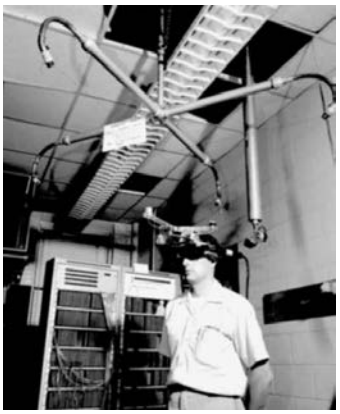


9 Appendix

9.1 Introduction to VR

9.1.1 History of VR

1965: The idea was born: As early as 1965, CG pioneer Ivan Sutherland developed the Head Mounted Display (HMD) idea. Two stereoscopic 3D images are projected directly into each eye. The dynamically generated images also depended on the head position and orientation. The dream of virtual space was born and failed due to inadequate display technology.



Ivan Sutherland invented the first Head-Mounted Display (HMD) in 1965. If you watch the video on YouTube, you will recognize that this first HDM was a mixed-reality display that augments a cube into a see-through display.

1985: The term VR was born: Jaron *Lanier* coined the term VR in 1985. A real hype arose, and several new tools, such as the data glove for interaction in virtual space, were developed. It was still the era of big, heavy cathode-ray monitors, and the first small displays built into headsets had such low resolution (max. 320 x 240) that one could imagine VR only with considerable goodwill.

Jaron Lanier coined Virtual Reality in 1985 and invented the data glove.

2015: VR arrives in the Gamer Market: After another 30 years of computer and screen technology development, the implementation of virtual reality was within reach. In 2016, Oculus and Valve launched their VR headsets, Oculus Rift and HTC Vive, for Windows. Although the prices of about 1000 USD for the headsets and about 2000-3000 USD for the driving PC were not low, they enabled broader use beyond universities and companies for the first time.

The most important improvements to the VR headsets from the 80s were:

- High resolution of 1080 x 1200 pixels per eye
- High 110-120° viewing angle
- Head-Tracking: Accurate head position and rotation are required for good immersion.
- Low delay: A real-time simulation requires at least 60 images per second. Therefore, the computer must render an eye image within 8 ms.

2020: VR arrives in the broad Consumer Market: In 2015, Facebook acquired the VR pioneer company Oculus, intending to spread VR widely in the social media consumer market. For this purpose, a new standalone, battery-powered VR headset, Oculus Quest, was released in 2019 at a very low price. Facebook's seriousness about VR is evidenced by its 2022 name change to Meta, which promoted its ambitious project, the Metaverse.

9.1.2 Mobile VR: Meta Quest

The first *Meta Quest* headset is a fully standalone device, features two six-degree-of-freedom (6DOF) controllers, and runs on a built-in computer with Android as the operating system.

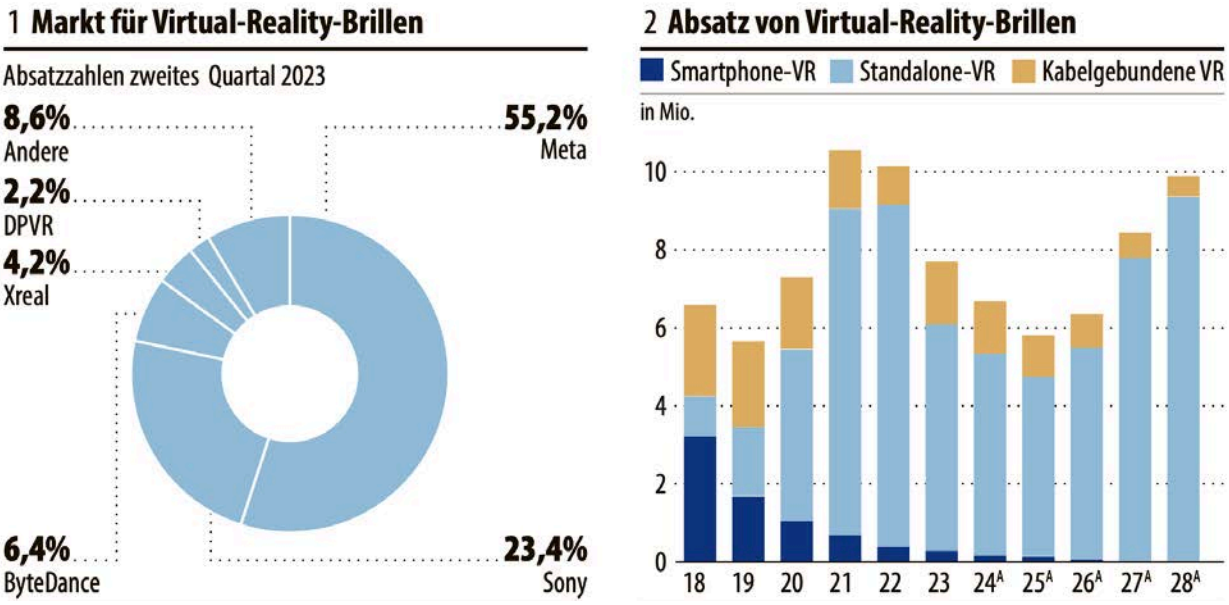
The Quest uses an inside-out 6DOF tracking system named Oculus Insight. The system relies on four wide-angle cameras mounted at each corner of the headset to track the headset and its controllers in 3D space. These cameras can also detect your hands so you can use the headset without controllers.

Meta produces the Quest in the following models:

- 2019: Oculus Quest
- 2021: Quest 2
- 2022: Quest Pro
- 2023: Quest 3

To be fair, we have to note here that Meta is not the only producer of mobile VR headsets:

- 2021: PICO Neo3
- 2022: PICO 4
- 2023: Vive Focus 3
- 2023: Vive XR Elite
- 2024: Apple Vision Pro



Meta and their mobile VR headsets will dominate the VR market from 2019 until today. The advantages of cableless mobile headsets over their cable-bound competitors are so clear, even with their lower graphic performance that their market share is estimated to rise in the future (Source: Finanz und Wirtschaft, February 2024)

We try to support all mobile VR headsets using the OpenXR VR standard. For more information, see 9.2.2 More Development Techniques for VR.

The Oculus Quest that we lend you, costs about 500 Swiss francs.
Please treat it with absolute care. Our Oculus Quest package contains:

- The Oculus Quest headset with the label Quest-??
- Two controllers with the labels Quest-?? and a pair of rechargeable type AA batteries
- A charger with the label Quest-??
- A USB-C to USB-3 cable
- A USB-C to USB-C cable
- A case for everything
- A silicon cover that can be disinfected.

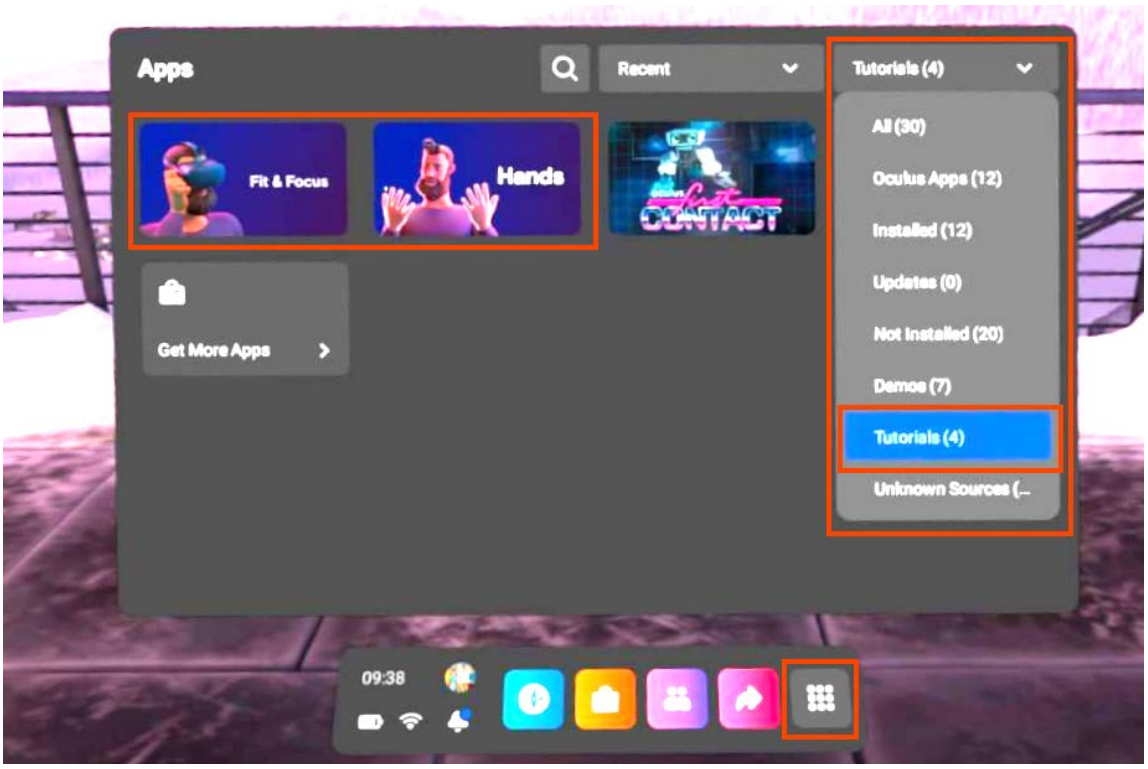
9.1.2.1 Only for the first-time Setup

For new Oculus devices, one has a few steps once. We have described them in Appendix 9.3 Meta Quest Developer Setup. The headsets that you get provided are already set up. Please do NOT change this setup.

9.1.2.2 First Steps in Oculus Quest

If you have never used befor the Meta Quest headset, we recommend the following short video tutorials:

1. Watch the Oculus Quest 2: Comfortable Fit
2. Watch the Oculus Quest 2: Maintenance and Care
3. Watch the Oculus Quest 2: Wearing Glasses if you are wearing glasses.
4. Watch the Oculus Quest 2: Setting the Guardian
5. Put on the Quest headset and watch the tutorials *Fit & Focus* and *Hands* using the controllers:

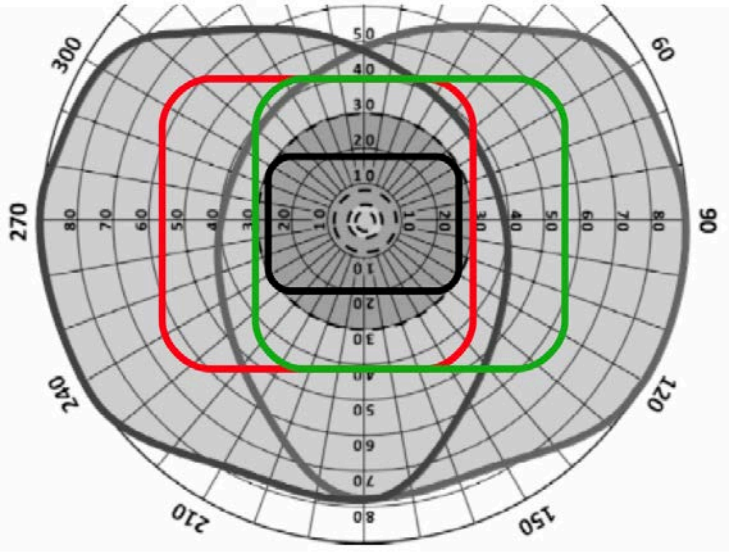


6. Turn on Hand Tracking: If the hand-tracking feature is not enabled, you can do so under *Movement Tracking (Bewegungsverfolgung)* settings.

9.1.3 Challenges in VR

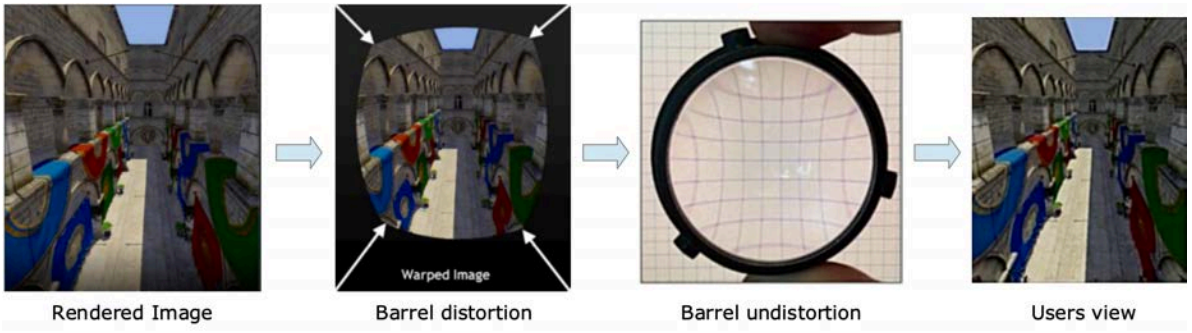
VR rendering is probably the most demanding rendering task that a GPU can do:

- **Render twice**: Due to the stereoscopic view for the left and the right eye, the entire scene has to be rendered twice. It's just the double work a GPU has to do.
- **Render extra fast**: The *flicker fusion threshold* is the frequency where the *persistence of vision (POV)* starts. It depends strongly on the illuminance and the retinal angle. Our color cones have a high dynamic range but are slow whereas. Our grayscale rods have on the other hand a low dynamic but are fast. This means that we see slowly in the center where the cones are in our retina and fast at the outer angles where we mostly have rods. Evolution is thought to have developed this feature so that we can react very quickly to objects that enter our field of vision horizontally.

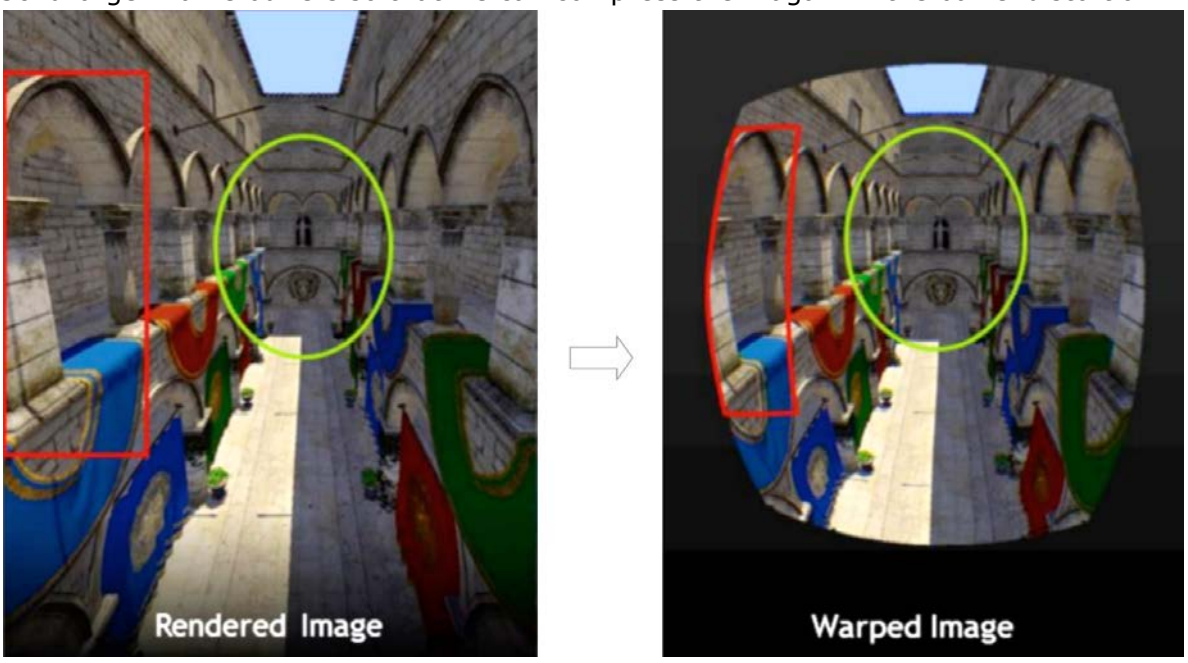


Within our normal monitor, tv, or cinema view angle (black frame w. 50°) our POV starts at around 25-30 FPS. Our displays in VR headsets cover about 90° each (red and green frame). To achieve POV in these outer angles we need 60-90 FPS.

- **Render extra-large**: The screens must be placed very close to the eyes to achieve such a large viewing angle. However, our eyes cannot focus on this distance. Therefore we have to look through two lenses that cause a relatively strong pincushion distortion. To compensate for this distortion, the scene must be rendered with barrel distortion.

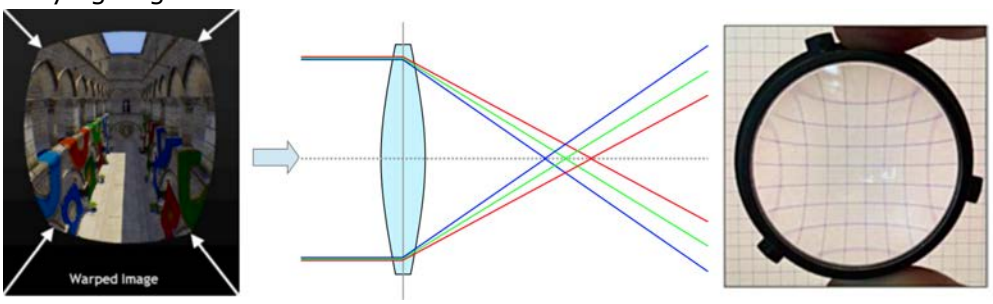


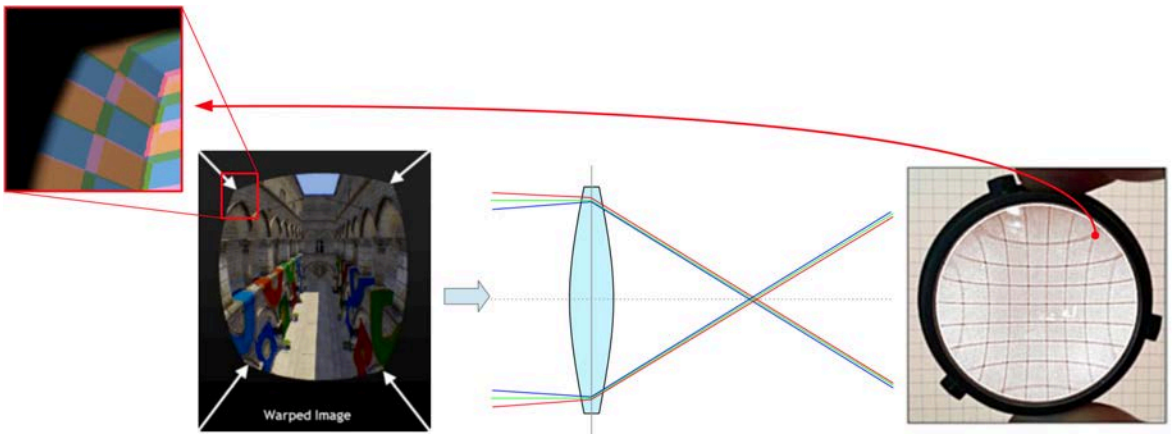
Because of this pre-distortion, the pixel density per region is not everywhere equal. While the density in the center remains about the same, we have to render much more area for the outer viewing angles. This means that we even have to render into about 30% larger frame buffers so that we can compress them again in the barrel distortion.



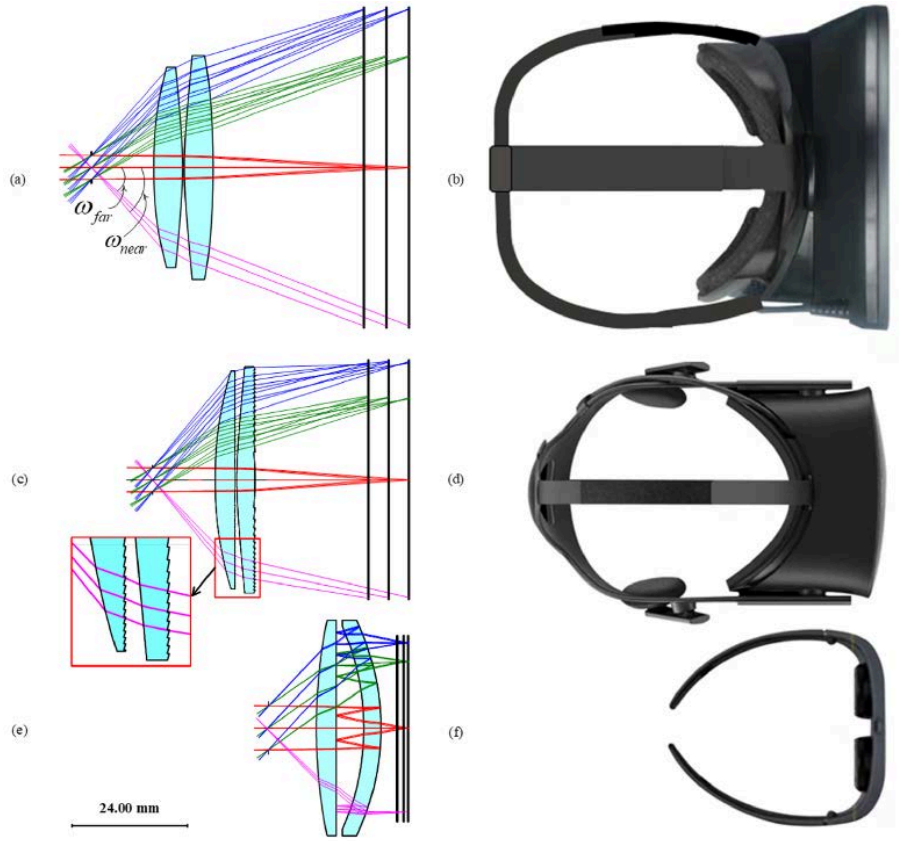
Frame buffer with corresponding pixel areas before and after barrel distortion. (Images: Nvidia)

- **Correct color aberrations**: Also because of this pre-distortion, a chromatic aberration occurs in the outer angles. To compensate for this, the RGB color channels must be rendered with varying degrees of distortion.



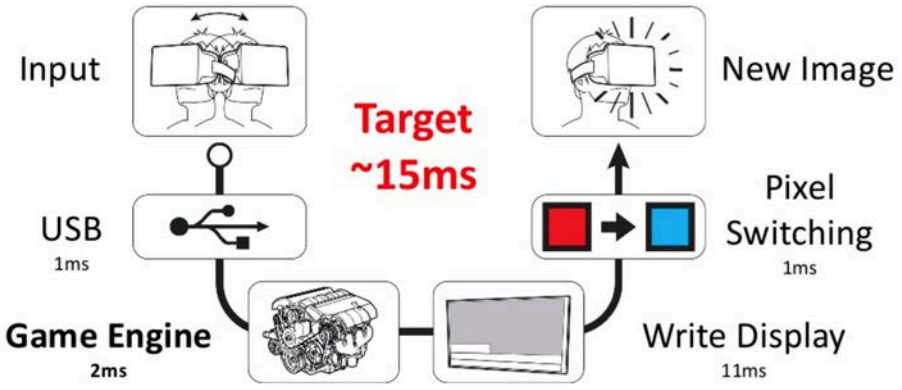


Chromatic aberration before and after the per-channel distortion (Images: Nvidia and Wikipedia). Another possibility would be to build VR headsets with more expensive achromatic lenses.



The VR lenses are evolving quickly and allow closer and smaller displays.
a) Oculus Rift DevKit1 in 2014, b) Oculus Rift in 2018, and Pancake lenses in 2022.

- **Fast-tracking**: For a good immersion we need about 60-90 FPS. That means that we only have about 11-17ms of time to generate both images. This also includes the time it takes to calculate the position and orientation (= pose estimation) of the headset. The tracking must therefore also be extremely fast.



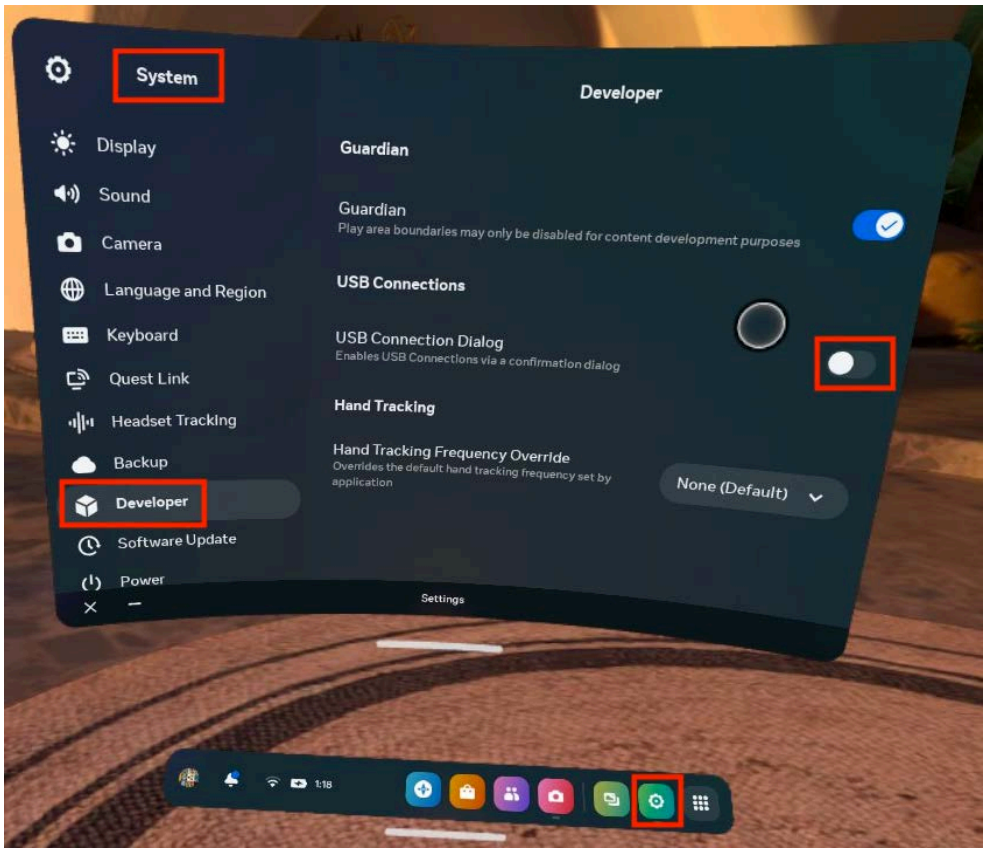
Max. time between a head movement and the display of the images (Images: Oculus)

9.2 One-Time Development Setup for Meta Quest

Warning: For new Meta Quest devices, one has to do the following preparation.

Please don't do this on the headsets we provide you. We have already done the following steps. Please don't use other accounts on our devices or install other apps from the Meta app store.

- **Create a Meta Account** by going to https://auth.meta.com/
- Create a Meta Developer Account by going to: https://developer.oculus.com
  - Register with the Meta account
  - Set the mobile phone number for 2-factor authentication.
  - There is no need for a credit card.
- **Download & install the Oculus app** from the app store of your phone OS.
- **WLAN Connection**: At BFH, use the Wifi name *Eduroam,* not *BFH*.
- **Enable Bluetooth** on your mobile phone.
- **Log in** with the following predefined account:
  - E-Mail: ???
  - Password: ???
- **Connect Quest-Headset to WIFI**
  - **WLAN:** eduroam
  - **CA-Certificate:** No Validation
  - **user:** BFH-Shortname@bfh.ch
  - **password:** BFH-Password
- **Turn on the Quest** headset by pressing the button on the right side for a few seconds.
  - At the first time a 5-digit code is shown; remember it.
- **In the Meta App** of the mobile phone
  - Go into the menu (3 dashes at the bottom)
  - Choose *Devices (Geräte)*
  - Choose the menu at the top (3 dots) > Add new Device
  - Choose your device type (e.g. Quest 2). It is searched now over Bluetooth.
  - Couple Quest Headset by entering the 5-digit code that you read at startup.
- **Set Developer Mode** in the mobile phone's Meta App: This is needed so that we can install our app on the Quest from our computers.
  - Go into the menu (3 dashes at the bottom)
  - Choose *Devices (Geräte)*
  - Choose the menu at the top (3 dots) > Change Device
  - Choose Headset Setting (*Geräte Einstellungen*)
  - Choose *Developer Mode* (*Entwicklermodus*)
  - Activate *Developer Mode*
- **Disable the *USB Connection Dialog***: Each time you connect the Quest headsets, you will be asked to allow access from the connected device (your laptop). This is annoying when developing. To disable this confirmation inside the Quest headset, do the following:
  - Go to the Settings
  - Go to the *System* settings
  - Go to the *Developer* settings
  - Disable the *USB Connection Dialog*



9.3 More Development Techniques for VR

9.3.1 Debugging a Quest VR App

A good overview of several techniques for debugging a Meta Quest is presented in a YouTube video by ShipBit. The easiest way is to use the Android Debug Bridge (adb) with the logcat command and the restriction to the Unity log messages. You can find adb in the platform-tools folder of the installed Android SDK:

```
platform-tools % ./adb logcat -s Unity
```

For more help on how to use the adb tool, see:
https://developer.oculus.com/documentation/native/android/ts-adb/

9.3.2 Screen Casting from the Quest to a PC or Mac

9.3.2.1 Screen Casting with Oculus Casting to a PC or Mac Browser

This is the built-in casting functionality to cast the left eye screen to a web browser:

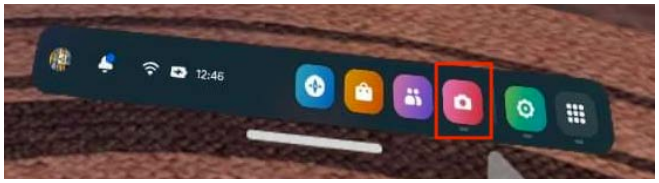
1. Head to oculus.com/casting on any up-to-date Chrome or Edge web browser. You must log in with your Meta account using an annoying three-factor authentication.
2. Slip on your Quest/Quest 2 headset and select the "Share" option in the Oculus universal menu. The headset and the web browser must be on the same Wi-Fi network.
3. Click on "Cast" and select the desired PC on your list of available devices.

9.3.2.2 Screen Casting with scrcpy

There is a simple Android tool that lets you copy the screen to a PC or Mac via TCP/IP. It is called *scrcpy* and can be downloaded from GitHub. This is also the only way to see the camera's see-through feature from Oculus. You can also record the capture. The latency is acceptable even if both eyes are copied. Read the readme for installation instructions. The advantage of this method is that you don't need a Wi-Fi network, but your headset needs to be connected to the computer with a cable.

9.3.3 Screenshots Inside VR

In all VR systems, you will find one or more possibilities to take screenshots or even video recordings inside VR. In the Meta Quest headsets, you must initiate this with the *pink button and the camera icon* on the main menu.



After starting the screenshot, a timer gives you about 4 seconds to switch to the app where you want to take it. The screenshot is then stored inside the headset's file system, and you have to connect it to your computer to access it.

A more comfortable way is to use our screenshot button in the menu and let the image be sent to your email address.

9.4 Installing an APK without Unity

The Meta Quests (1, 2, 3 & Pro) and the PICO headsets are Android devices, meaning they run the same operating system as Google's Android mobile phones. To install an APK on the Quest, you have the following options:

- Using the **SideQuest App**:
  - **Download SideQuest Advanced Installer** from https://sidequestvr.com/setup-howto for your OS.
    Do NOT download the Easy Installer.
  - **Install SideQuest**: View the YouTube video with an installation tutorial:
    - Install the SideQuest application on your computer.

        - Install the Oculus drivers from https://developer.oculus.com/downloads/package/oculus-adb-drivers/ if you have no Unity installation
        - Set the Quest headset into developer mode as described in 9.3 Meta Quest Developer Setup
        - Connect the Quest headset and accept the computer connection.
    - **Install the APK with SideQuest**: If your headset is connected, you will see a green light at the top-left:

      Press the *Install APK file from folder on computer* button and choose the APK file to install. Wait until the installation is finished.

- Using the **Android SDK** from within Android Studio:
    - **Install Android Studio**: Android Studio from JetBrains is the official development environment for any Android device. Download and install it.
    - **Use the ADB Tool** (*Android Debug Bridge*): Follow the instructions from https://developer.oculus.com/documentation/native/android/ts-adb/

- **Start the App**: Hand-installed apps are a little hidden:
    - Click on the button to the left with the *9 dots* in the main menu to open the *App Library.*
    - Click into the *Search apps* text field at the top.
    - Choose *Unknown Sources* in the second field at the top.
    - Choose *Recent* in the last field at the top.

You should now see your hand-installed app at the top of the *Unknown Sources* list. Click on it to start the app.

9.5 Improvements for VR Rendering

The *Meta Quest* VR headsets are mobile devices that run under battery power as your mobile phone does. You can not expect the same performance and quality as from a cable-bound VR headset that receives its images from a PC with a dedicated graphics card that consumes 500 watts alone.

As described in Chapter 8.2.1 Challenges in VR it is important to achieve a framerate higher than 60 fps (frames per second). If it is lower you will immediately notice a lagging when you look around and you won't feel present in the scene. In such a case you will ask for measures to improve the performance first before you want to improve the image quality.

The build settings in Unity are therefore set in a way to achieve first a good framerate by reducing some render quality settings.

- You can monitor your framerate by opening the built-in menu and clicking on the bug icon. Here you can activate the framerate by checking the option ***Show FPS***:

- Alternatively, you can monitor your frame rate in more detail using the OVR Metrics Tool.

You must work on some 8.5.1 Performance Improvements if you get a low framerate.

If you are happy with the framerate you can work on some 8.5.2 Quality Improvements.

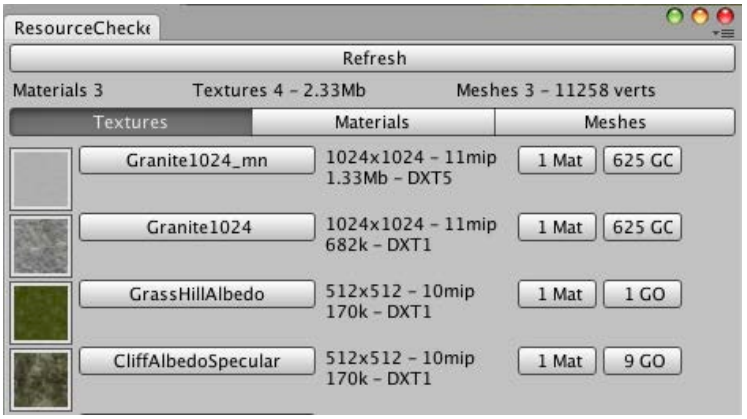
9.5.1 Performance Improvements

The following recommendations are a short list of measures that you can take to improve the performance of mobile headsets. In chapter 8 Case Study: Circular Tower we described these steps in more detail with a larger architectural project.

9.5.1.1 Draw less

- **Reduce the number of objects**: Many exports from CAD tools are suboptimal with way too many individual objects. Try to reduce that number by merging objects with the same material. Every object with a specific material is executed with a so-called *Draw Call*. You can see the no. of draw calls in the statistics shown in the *Settings* dialogue.

- **Reduce the model complexity**: Exported objects from CAD tools can easily have thousands of triangles. A good tool for checking the mesh sizes is the free *Resource Checker* tool from the *Asset Store:*
    - The *Resource Checker* tool is designed to help bring visibility to resources used in your scenes (e.g. what assets are using up memory, which meshes are a bit too detailed, where are my materials and textures being used):



    - The no. of triangles that are currently rendered can be seen in the statistics.
- **Reduce texture size**: Don't use textures bigger than 2k in size (2048 x 2048 pixels). Mobile headsets don't have as much memory as desktop graphics cards.

9.5.1.1 Draw faster

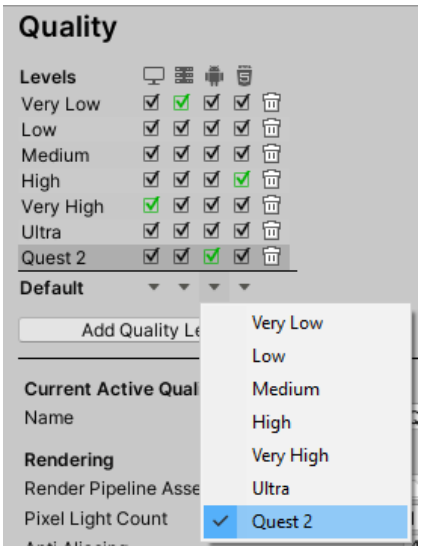
- **Don't do dynamic lighting**.
  Bake it all as described in 5.3.2.3 Lightmapping the Sponza Scene.
- **Batching**: By setting objects to *Static GI* you set them also to *Batching static* which allows Unity to merge many little objects with the same material into one object (drawcall). The no. of batched objects can also be seen in the statistics.
- **Decrease Texture Render Quality**: Todo.

9.5.2 Quality Improvements

9.5.2.1 Define Quality Settings for Quest 2

In this chapter, we will look into tweaks and settings to get the most out of our VR applications visually, without straining too much on performance. Since the Meta Quest Headsets work as standalone devices, they can not rely on a powerful PC to do the demanding calculations. Our project should already have these settings:

- **Create new Quality Settings**: *Edit > Project Settings > Quality*. Click *Add Quality Level* and name it Quest 2. Click on the little Down Arrow in the table in the Android column and select your new Quality Level to set it as the Default Level.

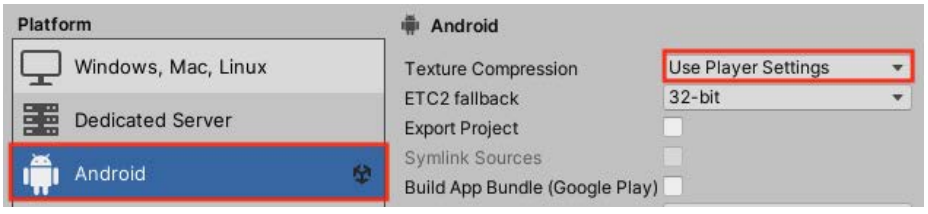


- **Configure the new Quality Level**: With the new Quality Level selected, adapt the configuration to the following:
  - *Pixel Light Count*: 1
  - *Anti-Aliasing*: 4x Multi-Sampling
  - *Enable Realtime* Reflection Probes
  - *VSync Count*: Don't Sync
  - *Texture Quality*: Full Res
  - *Anisotropic Textures*: Per Texture
  - *Soft Particles*: Disable
  - *Particle Raycast Budget*: 128
  - *Billboards Face Camera Position*: Enabled
  - *Shadowmask Mode*: Shadowmask
  - *Shadows*: Hard Shadows Only
  - *Shadow Resolution*: Medium Resolution or High Resolution
  - *Shadow Distance*: 5
  - *Shadow Cascades*: Two Cascades

  - If your application experiences lag, you can play around with some of the settings. Changing the Shadow Resolution and the Shadow Distance most affects the Quality.

9.5.2.1 Improve Texture Rendering

- **Use ASTC Texture Compression for the Android platform**:
  - Check the *Build Settings* that *Texture Compression* is set to *Use Player Settings*:

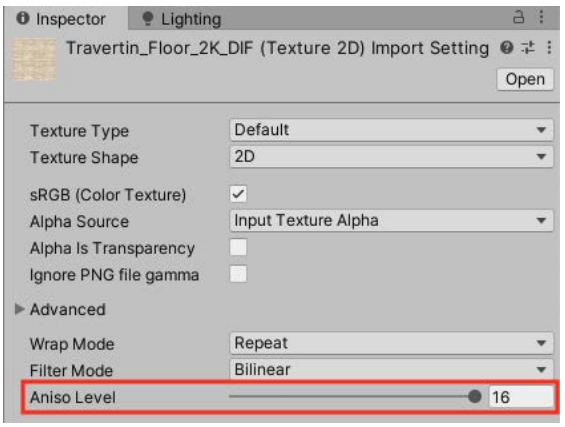


  - Check the *Project Settings* > *Player* > *Other Settings* > *Rendering,* you set the *Texture Compression Format* to ***ASTC***.

- **Use a higher-quality Texture Filtering**: Texture Filtering is the process of minifying the texture image. It is often used in architectural scenes, where we often look at wall or floor textures into the depth of space. In the following image, we look horizontally over the floor and see the travertine texture up close in its original

resolution. This texture has to be applied at further distances in more minified versions.

You can improve this filtering with a higher *Aniso Level* in the *Import Settings* of the texture image:

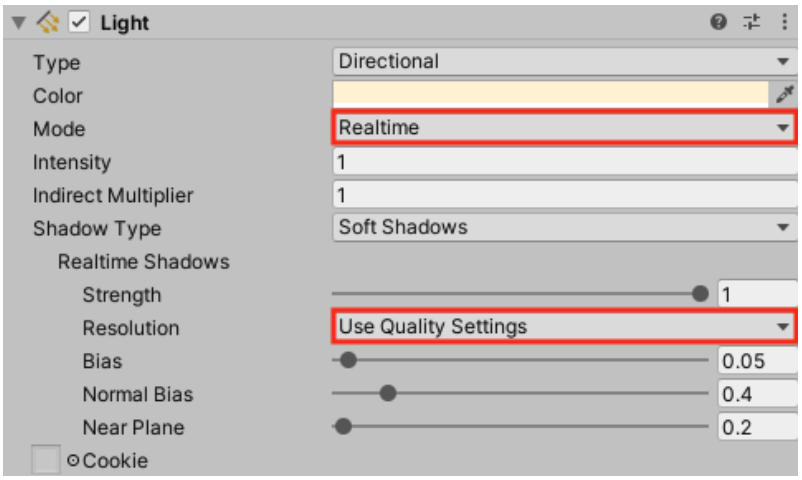


From left to right: 1, 4, 8 & 16 Aniso(tropic) Level. The number means how many texels (= texture image pixels) are involved in creating the final pixel displayed. The more are, the more expensive it is in terms of performance.

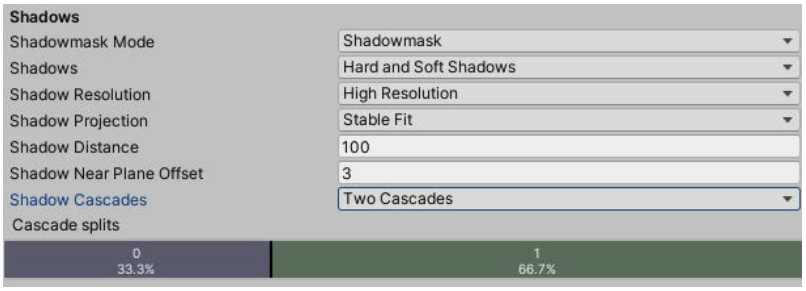
9.5.2.2 Improve Shadow Quality with Realtime Lighting

As described in 5.2 Lighting in Unity, dynamic lighting with shadows is mostly too expensive for mobile VR. If your scene is straightforward, you can still do it, but the default shadow quality is inferior.

To improve it you must set your light source mode to *Realtime* or *Mixed* and the *Resolution* to *Use Quality Settings*:

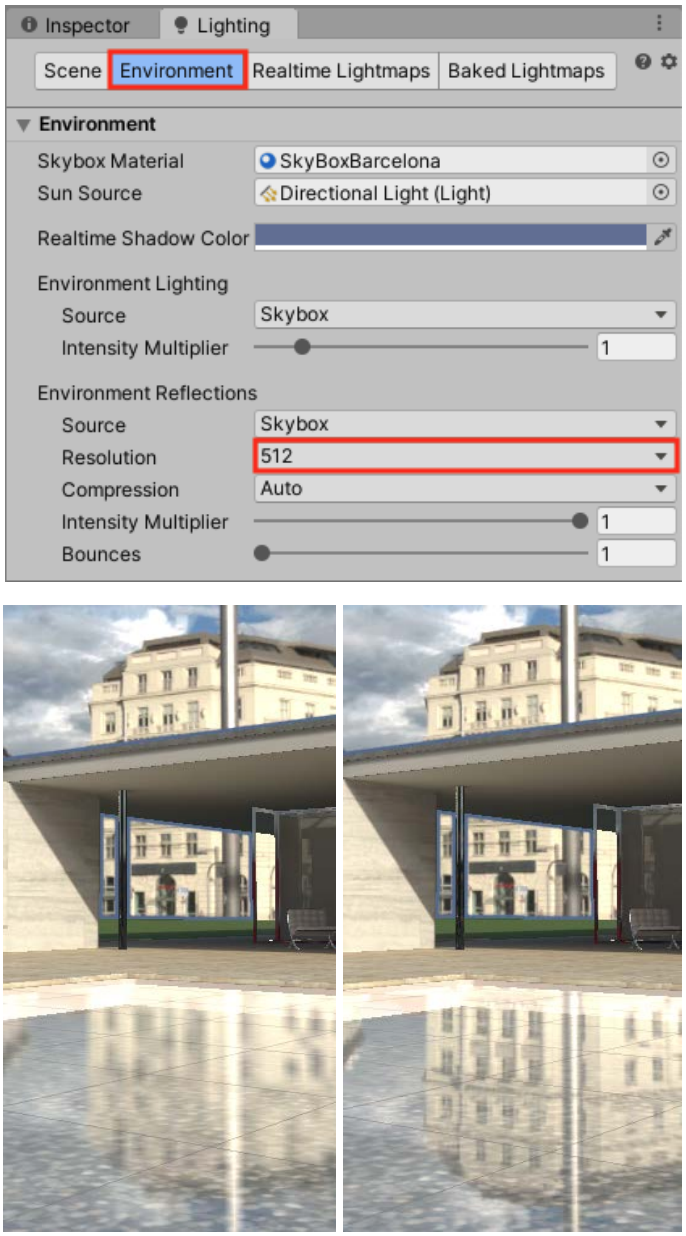


In the *Project Settings* under *Quality* set the following settings:



9.5.2.3 Improve the Skybox Reflection Quality

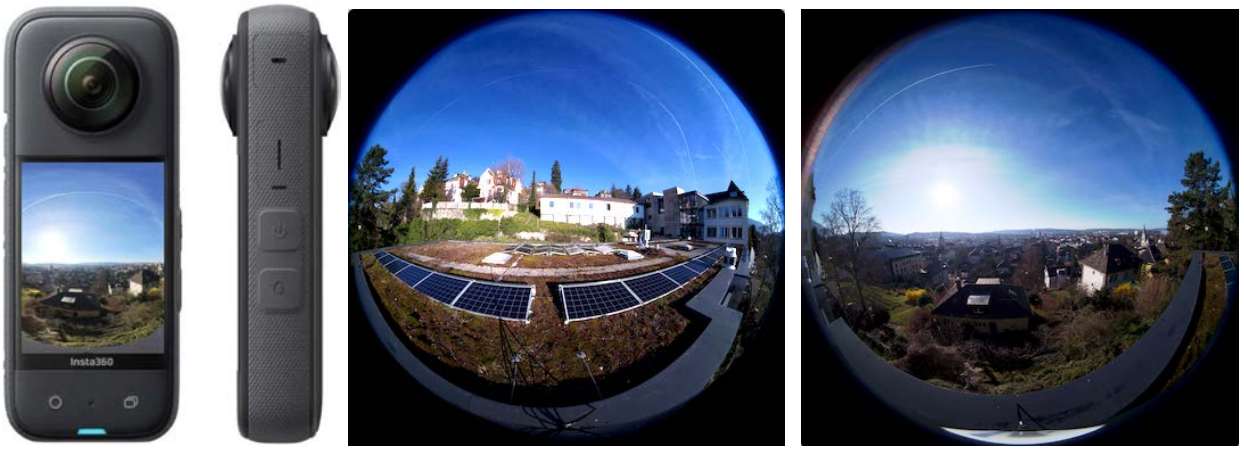
Environment reflections on skybox images are created by default with a lower resolution than the background skybox image. You can increase this resolution in the *Lighting* window in the *Environment* tab with the *Environment Reflection* > *Resolution*:



Left: The standard resolution of 128 is used for the environment reflection. Right: The increased resolution of 512 means that a cube map of 6 images with 512x512 pixels is used for the reflection.

9.6 Create your own 360° HDR Image

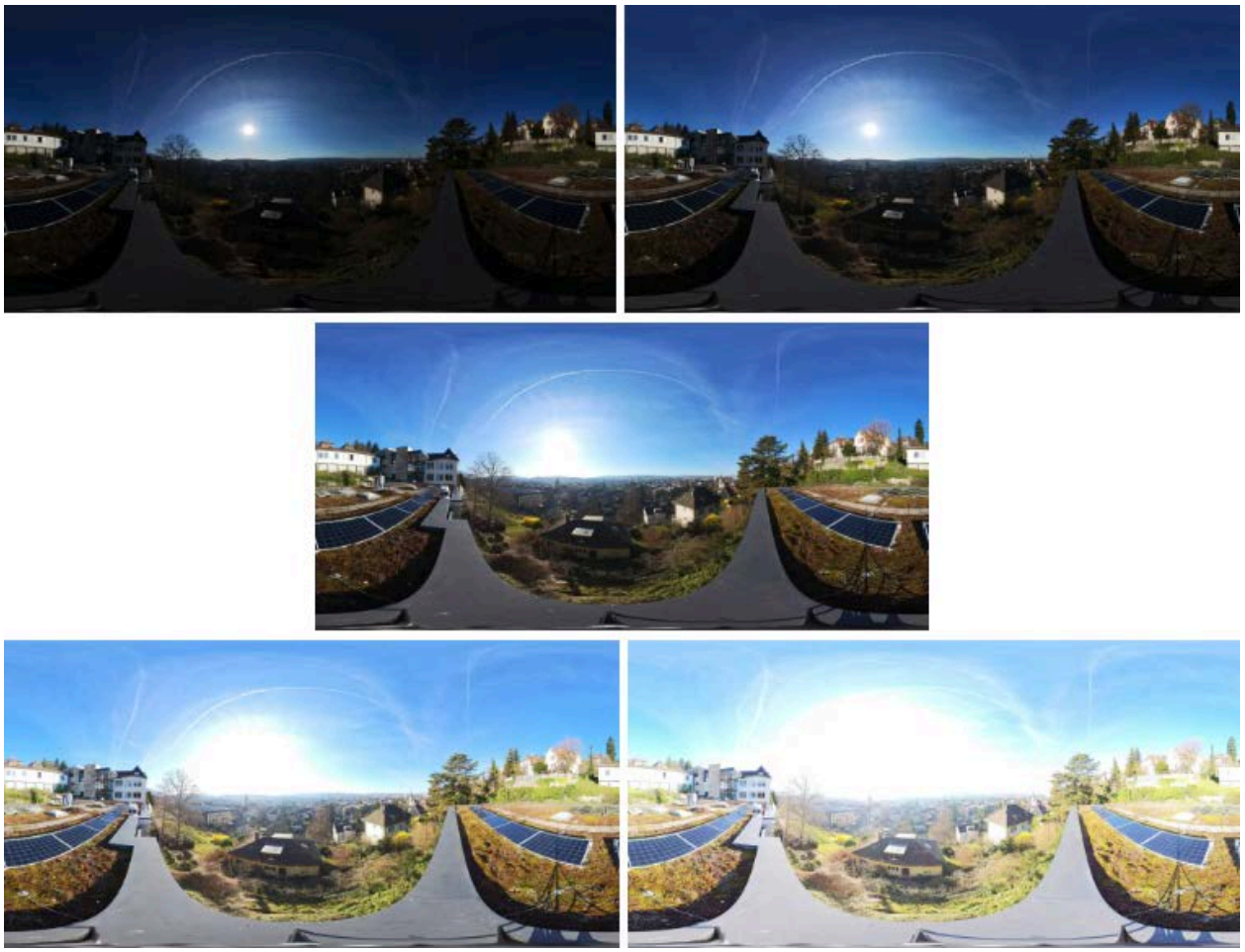
A 360° camera has two fisheye lenses with a 180° field of view each. Stitched together, you can capture 360° of an environment.



*The **Insta360 One X3** camera with two raw fisheye images that get later stitched, and reprojected to an equirectangular and cylindrical projection within the Insta360-Studio software (see next images).*

To create your own 360° HDR image, do the following:

- **Place the 360° camera** on a tripod in an empty place with nothing up close to the camera. The smaller the tripod feet are the better so that they will be almost invisible in the final images.
- **Control the camera from the mobile app**. Don't take photos while you are standing at the camera because you will be also in those photos ;-). If you don't have a mobile app, you have to use a timer to get away from the camera.
- **Switch the camera to HDR Photo mode** and set the automatic capture of 5-7 images with ±1.0-1.4 apertures. With 5 images set with ±1.0, the camera will capture at -2, -1, 0, +1, and +2 aperture offsets.
- **Capture 5-7 photos with different exposures** at an absolutely stable position.
- **Upload the images from the camera to your computer** via the mobile app or with the Mini-SDCard from the camera.
- **Import the images into the Insta360-Studio desktop software**.
- **Export the images as equirectangular projections in full resolution**.
- **Combine the images into one and export it as an HDR image**:
  - **With Photoshop**: If you have Photoshop you can follow this YouTube video to combine all exposures into one HDR image.
  - **With Picturenaut**: This free windows tool allows you to combine multiple exposures into one HDRI and export it as an HDR file (Radiance, *.hdr file type).



Five equirectangular 360° images at five different exposures.

9.7 Introduction to Unity Scripting

We are using Unity to create a VR app with predefined functionality. The cpvrLab developed this functionality using scripts and a programming language. Unity is originally a game engine for the development of real-time games.

In this chapter, we show you how to create a simple game to give you an overview of the basic principles.

1. **Scale the Plane:** Select the plane in the *Scene* or *Hierarchy* window. In the Inspector window, change the *scale factors* to [10,1,10].
2. **Move the Sphere:** Select the sphere and change its position [0,2,0].
3. **Add Rigidbody component**: Select the sphere and choose the menu *Add Component* > *Physics* > *Rigidbody*. By adding components to GOs, we can add functionality to them. The *Rigidbody* (Deutsch: *Starrkörper*) enables the GO to act under the control of physics. It can receive forces and torque to make your objects move realistically.
4. **Press Play** and see how the sphere falls and gets stopped by the cube. The integrated physics engine calculates a new position by applying the Earth's gravitation to the sphere's mass. It does this every 50ms, so 20 times per second. In addition, it also checks every time the sphere collides with another object. If so, it gets stopped. The

sphere collides with the cube because the sphere has an additional *Sphere Collider* and the cube has a *Box Collider* component.

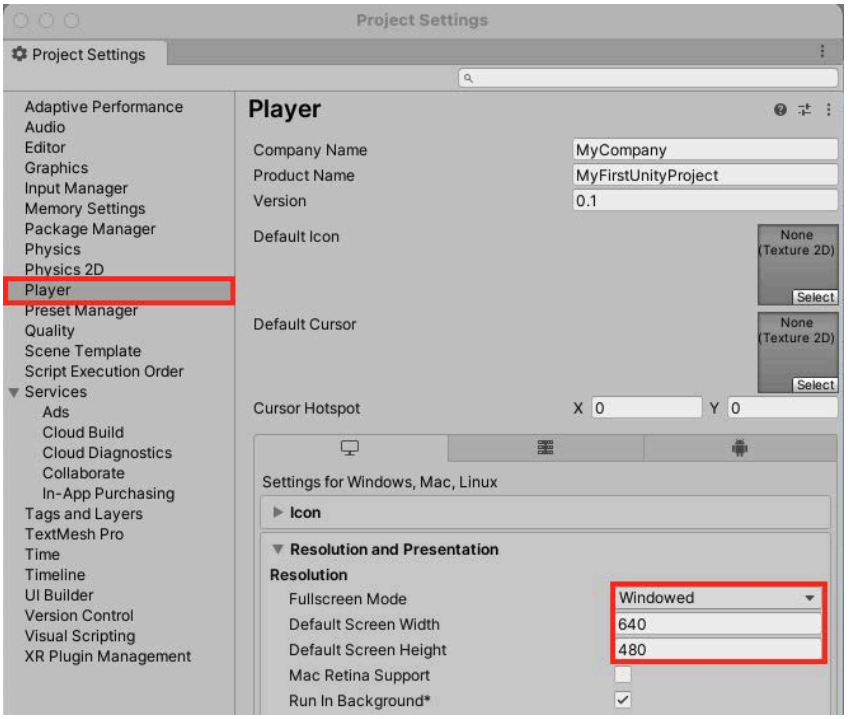
5. **Press Play again to stop the game**: Don't forget to press the *Play* button again to stop the game. This is one of the major beginner mistakes. You quickly forget that the game still runs, if nothing happens on the screen. If you continue to edit the game while it is running, all changes will be lost after stopping the game.

6. **Add Script**: Select the sphere and click the *Add Component* button at the bottom of the *Inspector* window. Choose *New Script* and name it *SphereBehaviour*. Double-click on the *SphereBehaviour* script name to open the programming environment. This can last the first time a few seconds. This works only if the development environment was also installed during the installation. You can edit programming code in any text editor. But a dedicated coding editor or programming environment is recommended for good syntax highlighting.
   When you see the template code, select it all and delete it. Copy and paste the following code. The code is written in the programming language called C#. What it does should be more or less clear from the code comments behind the //:

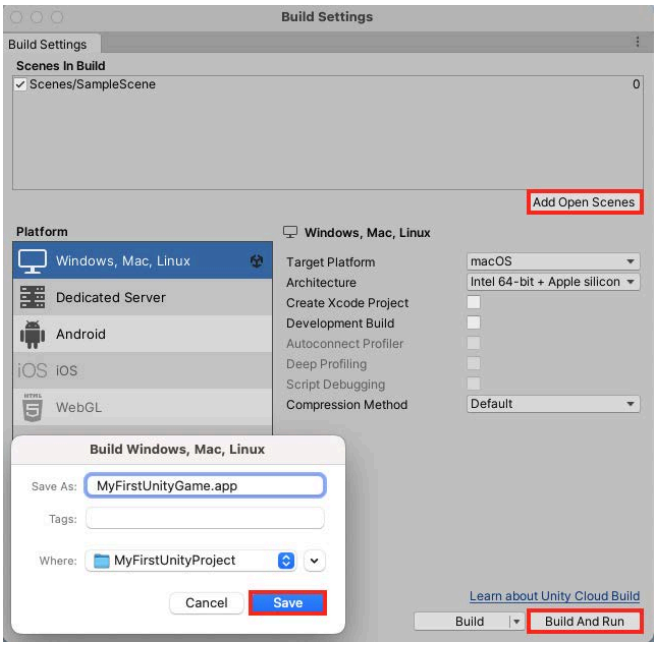
```
using UnityEngine;

// C# class that inherits from MonoBehaviour and implements additional behavior
public class SphereBehaviour : MonoBehaviour
{
   // Private variables only visible for this class
   private const float _thrustAcceleration = 1000f;
   private const float _jumpForce = 500f;
   private Rigidbody _rigidbody;

   // Gets called once at startup
   private void Start()
   {
       // Get the component from the same game object
       _rigidbody = GetComponent<Rigidbody>();
   }

   // Gets called every 50ms by the Unity physics subsystem
   private void Update()
   {
       // Get the horizontal input keys (A/D or cursor left/right)
       var h = Input.GetAxis("Horizontal");

       // Get the horizontal input keys (W/S or cursor up/down)
       var v = Input.GetAxis("Vertical");

       // Calculate acceleration from the horizontal and vertical cursor keys
       var acceleration = new Vector3(h, 0f, v) * _thrustAcceleration * Time.deltaTime;

       // Tell the physics engine to add a force by this acceleration
       _rigidbody.AddForce(acceleration, ForceMode.Acceleration);

       // If the space bar is pressed, add a jump force
       if (Input.GetKeyDown(KeyCode.Space))
           _rigidbody.AddForce(Vector3.up * _jumpForce, ForceMode.Force);
   }
}
```

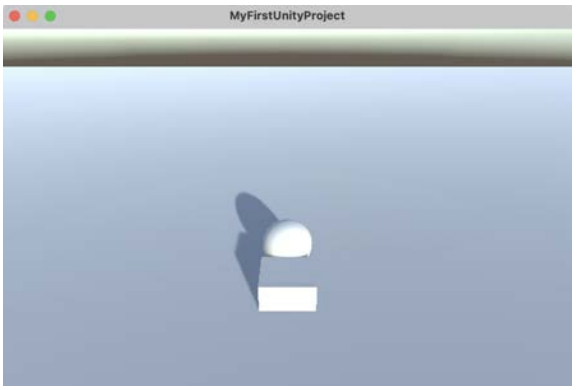
7. **Save the script** in the development environment with (CTRL-S or Command-S on Mac). Unity will compile the script in the background to check if there are syntax errors in it. Programming languages have strict syntax. Any missing brackets {} that define code blocks or missing semicolons (;) will generate a syntax error and the game can not be started. These errors have to be fixed in advance.

8. **Press Play and play the game**: Switch back to Unity. Unity automatically recognizes the changed script and recompiles it. Press *Play* and try your first game by pressing the cursor keys.
9. **Building the game**: You can only play your game from within the Unity editor. But we can also deploy our game into a standalone executable.
   a. Choose the menu command *File > Build Settings …*
   b. Add the open scene by clicking *Add Open Scenes.*
   c. Click *Player Settings* in the lower-left corner. Choose *Windowed* for the *Fullscreen Mode* and the desired *Default Screen Width & Height*:



   d. Back in the *Build Setting* click *Build and Run …*
   e. Name your game *MyFirstUnityGame* in the *Save As* field and click *Save*. An executable application will be built and started depending on your operating system.





Summary:

- **A game in one hour**: This example overwhelms you if you have never used a programming language. But you also recognize how easy it is to create your first game. It needed not more than an hour of introduction (chapters 2.1-2.3)!
- **Adding components**: In essence, game development in Unity is adding functionality to game objects with components. The most flexible one is a script component with your own commands in it. These commands then get executed at specific points at runtime, e.g., at startup (the routine *Start*), at every frame of the game (the routine *Update*), or when physics is involved at *FixedUpdate*.
- **It can be a deep water**: If you study the code closely, you will also recognize that a good understanding of linear algebra and a minimal understanding of physics is advisable.
- **But don't worry**, you will not program in this course. We will provide you with all the necessary scripts.

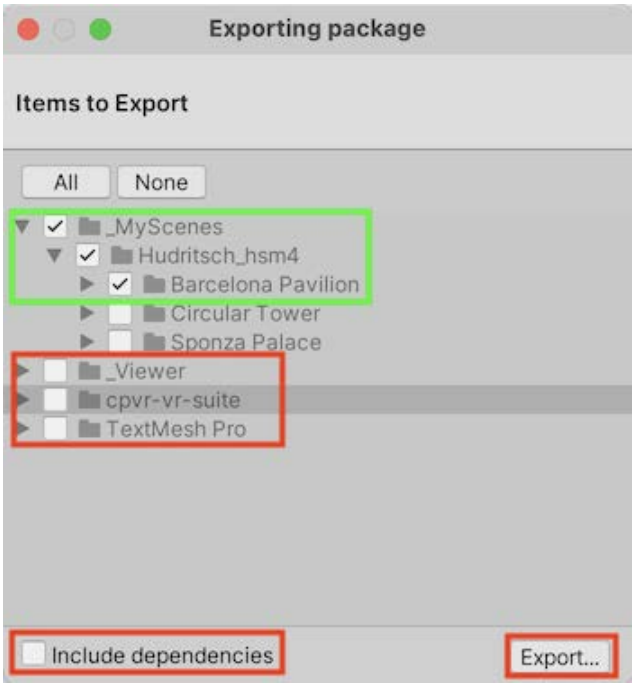
Game enhancement: With this logic, you could quickly improve this game to a billiard-style game with multiple balls bouncing off from surrounding rails. To make the balls bounce, you must assign so-called physical materials. Look in Chapter 5.1 Materials in Unity if you want to add some color to your game objects.

9.8 Export & Import a Scene as a Unity Package

9.8.1 Export your Scene as a Unity Package

You can export a scene with all configurations for archiving purposes as a *.unitypackage file. This is advisable if you want to keep your work with a small footprint or give it to someone else.

- **Open your scene** so that it is open in the editor window.
- **Export as unitypackage:** In the project window. Right-click on your scene file:
  - *Assets/_MyScenes/{YourName}/YourSceneName.unity > Export*
  - You should see only the files that your scene is using.
  - Only select the files from your scene (in the green frame).
  - Deactivate all that is not directly related to your scene.
  - Press the *Export…* button to start the export.
  - Give a name *yourSceneName.unitypackage*.



9.8.2 Import a Scene from a Unity Package

Do the following steps to import an exported scene from your colleagues:

- **Import Unity Package**: Choose the menu *Assets > Import Package > Custom Package...* and choose the Unity package to import.
  - Leave all files checked and click *Import*.
  - You can now modify the imported scene and decide to build as described in 4.3.5 Build the Scene