Virtual reality experiments for cognitive science

Jonathan MIRAULT and Stephane DUFAU

Laboratoire de psychologie cognitive
CNRS & Aix-Marseille University, France

Guide written in SEPTEMBER 2020

Forewords

This guide was developed in the Cognitive Psychology Lab (CNRS & Aix-Marseille University) using the following Virtual Reality setup:

- one **DELL Precision desktop** (Intel Core i7 8700K processor, 6 cores at 3.7 Ghz; 16 GB of RAM) with a NVIDIA GTX 1060 graphics card. The monitor was standard.
- one **HTC Vive Pro** (cost: ~1000/1500 $). Beware of the location of the stores as product may vary from one place to the other.

The scientific programs created in this guide were successfully tested on both the HTC and the FOVE headsets. Chosen software is free-of-use.

You can refer to the related scientific article published at *(t.b.a)*

1. Software installation

1.1. Unity

Unity (https://unity.com) is a cross-platform game engine that supports external C# scripts. Download the version that suits your professional situation.

1.2. SteamVR

SteamVR (https://store.steampowered.com/app/250820/SteamVR/) is handy middleware that supports many headsets including HTC and FOVE: one experiment programmed in Unity can be played on many headsets without extensive manipulation.

First, download Steam (https://store.steampowered.com/about/), install it, plug your headset and follow the procedure to install SteamVR.

Alternatively, SteamVR installation can be achieved within Unity (download the "SteamVR" plug-in). To do this, go to the "Asset store" in Unity, type "SteamVR Plugin", then download and import it. Import and/or accept all the windows that will pop up onscreen.

SteamVR supports most of the virtual reality headsets but in rare case, you will need additional drivers like, for example, with the Fove or Razer Hydra headsets.

Prior the first utilization, define the kind of setup between "room-scale" and "standing only" in the SteamVR software, clicking on "Room setup" (Figure 1).

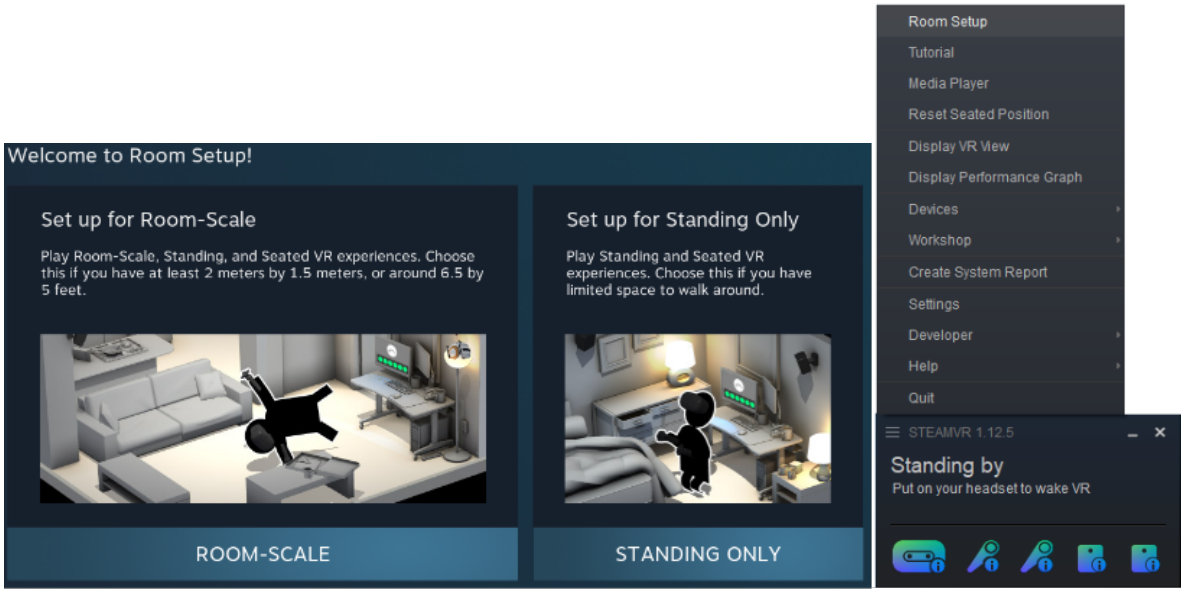


Figure 1. Room setup selection window (left) and SteamVR options' list (right).

2. One tool for several experimental configurations

2.1. Display the Windows Desktop in the headset

Displaying the Windows desktop (computer-like headset use) is great for running web browsers and other software already installed on the computer: an immersive experience for the participant and a control of external visual distractor for the experimenter. Any experiment tool such as OpenSesame, E-Prime, Matlab or DMDX can easily be converted into a 2D-projection in a virtual environment. In the 3D virtual world, the screen is displayed at a fixed location, mimicking the reality, meaning that when the user moves their head, the display does not follow their head. This sort of reality replication is the key to avoid virtual sickness (nausea).

To display the Windows desktop of the computer, run SteamVR and press the menu button of a controller, then click on the "Desktops" icon on the bottom left hand corner (Figure 2, top panel). You can modify the size of the screen from 75 to 150% (Figure 2, bottom panel).

Figures 2. Access to the Windows desktop from the menu of SteamVR (top) and choosing the size of the display (bottom).

You can alternatively install a Steam plugin such as Virtual desktop (https://www.vrdesktop.net; 12 $) that has many more features (multi-monitor, custom environment, etc).

2.2. World creation in Unity

The world created in this section is generic and will serve in all the proposed experiments.

World creation. When Unity is launched, an empty scene is created. It is composed of an infinite world divided into 2 zones: a dark one for the ground and a blueish one for the sky. The two zones are separated by a curved horizontal line. In the scene, there is by default one light and one camera. In all the VR experiments proposed here, we change the *external* setting of the camera to setting corresponding to the *participant's point of view* (his/her gaze). The camera setting in this context corresponds to a Point Of View shot (subjective camera). We will call the participant the "Player". For most of the experiments, we also need a plane (a physical

surface) on which the Player and some virtual objects can stand on. These objects (a square, a ball, etc.) are the "gameObjects". So that's all for our initial 3D world in Unity: a Player in a scene with gameObjects the Player can interact with.

Object creation. Surfaces and 3D objects are easy to set: into the "Hierarchy" panel (the location of this panel could change according to your software configuration), right-click in the empty zone. Under the "3D Object" menu, select the objects to put in the world. In the "Inspector" panel, you can precisely set the location and the size of the objects. Keep in mind that 1 unit stands for 1 meter in this software.

Component creation. It is important to dissociate the virtual environment from virtual reality. Within virtual reality, you can create a virtual environment for a 2D use or one for a 3D use. We focused in this guide on 3D worlds. We therefore want to set the compatibility of the created environment to virtual reality: for this, go to **Unity > Edit > Project Settings > Player > XR Settings**. Make sure that (i) the "Virtual Reality Supported" option box is checked and (ii) "OpenVR" is the Virtual Reality SDK selected (see Figure 3).

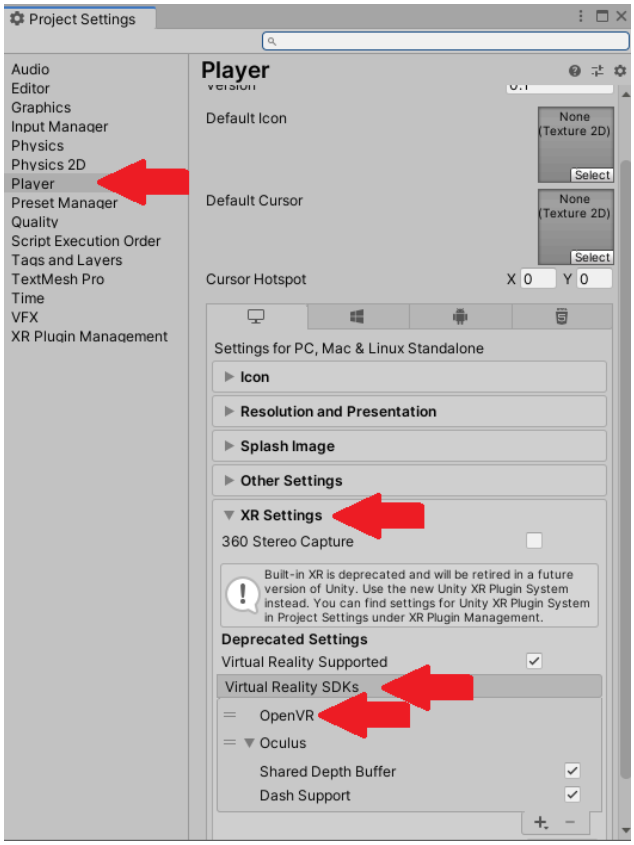


Figure 3. Setting "OpenVR" as the main SDK in Project Settings.

Next, import the SteamVR plugin (installed in 1.2) into your project. This plug-in has many components. One of them, the Player prefab, is to be put into the scene. For this, after the import step, type "Player" in the search bar of the Project panel. Drag and drop the Player prefab in the scene onto your plane. For a human-size display, set the Y position value to zero in the "Inspector" panel.

2.3. Record eye positions (eye tracking feature)

Getting the eye-tracking data follows 2 steps: (i) some modifications on the Unity side and (2) the creation of a C# script. Using the world created in 2.2, add some new piece of software. This process is described in section 3 (External sources) of this document. At step 6 of the additional setup, the VIVE SDK is used (which needs SRanipal's SDK), but it could be replaced by the Tobii item which does not need any additional SDK.

After this additional setup, we will modify the script to continuously record the gaze position (90 Hz sampling rate) and record an event (trigger) when the gaze hits a target object. The script, called "SaveEyeTrackingData.cs", is available in the actual repository. It originates from a script proposed in the examples of the Tobii SDK. The script HighlightAtGaze from Tobii is copied to the actual repository.

To record eye movements, create a new C# script and attach it to the TobiiXR Initializer. Copy the code from "SaveEyeTrackingData.cs" to your script. What the script does is the following: each time the eye position is updated (90 Hz sampling rate), the position and orientation of the eye is recorded. When the program quits, data is saved

2.4. Head and mouse tracking

Following a method identical to 2.3 (record eye position), change the settings of the "Eye Tracking Providers", from the Tobii option (if "TobiiXR Initializer" was installed) or the "VIVE" option to the new setting "Nose Direction" (see Figure 4 below). Identically, select "Mouse" to track the computer's mouse.

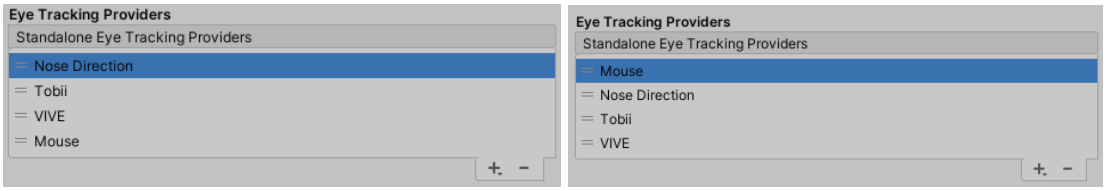


Figure 4. Selection of the "Nose Direction" (left panel) or the "Mouse" mode (right panel) in the "Eye Tracking Providers" settings.

2.5. Controller tracking

Tracking and recording controllers requires more work. First, make some modification in Unity. In the Hierarchy panel, develop both the Player item and the SteamVRObjects item (child of Player). You will see two gameObjects named "LeftHand" and "RightHand" (see Figure 5). They correspond to the two controllers.

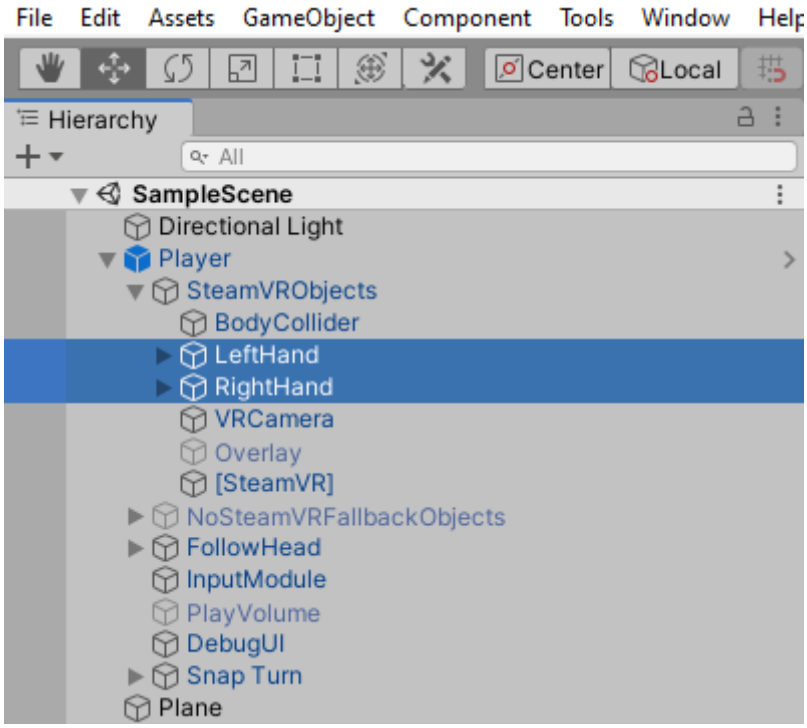


Figure 5. Hierarchy panel and child items of Player.

For each GameObject, follow this two-step procedure: (i) create the new tag and (ii) associate the tag to the controller (hand). In details, for each controller (i.e. “Lefthand” and “RightHand” item), go to the Inspector panel and add a new tag. Here, we create two tags named “LeftHand” and “RightHand” that are respectively associated to each controller.

Next, create a new C# script that is added to the Player item. Open the script editor and copy-paste the content of the “TrackingControllers.cs” script.

What the script does is the following: each time the controller information is updated, the position (3 axes), rotation (3 axes) and button states (3 buttons) are recorded. When the program quits, data is saved. There are two types of buttons: a trigger (named “Squeeze”, actioned by the index finger) that is dynamic and is mapped from 0 to 1, and two other buttons (named “Teleport” and “GrabGrip”, actioned by the thumb) that are binary coded (0: not pressed, 1: pressed).

3. EXTERNAL SOURCES

3.1. Setting the HTC Vive eye-tracking

Eye Tracking setup for HTC VIVE (from https://vr.tobii.com/sdk/develop/unity/getting-started/vive-pro-eye/.)

HTC VIVE Pro Eye Development Guide

This page will give you a step-by-step walkthrough on how to set up your HTC VIVE Pro Eye and how to create a simple scene in Unity using the Tobii XR SDK.

In the finished scene, you will be able to highlight game objects by looking at them.

With VIVE Pro Eye, developers are now able to create more immersive experiences using precision eye tracking and foveated rendering. The headset features 120Hz tracking and 0.5°–1.1° accuracy for amazing eye tracking performance and is the preferred VR headset for NVIDIA Variable Rate Shading (VRS).

HTC VIVE Sense SDK gives access to the eye tracking capabilities of VIVE Pro Eye through Unity and Unreal plugins as well as from native C. The Tobii XR SDK supports the VIVE Sense SDK. All use cases and source code samples are compatible with the SRanipal APIs as well as Tobii APIs. Find out more about VIVE Sense SDK here.

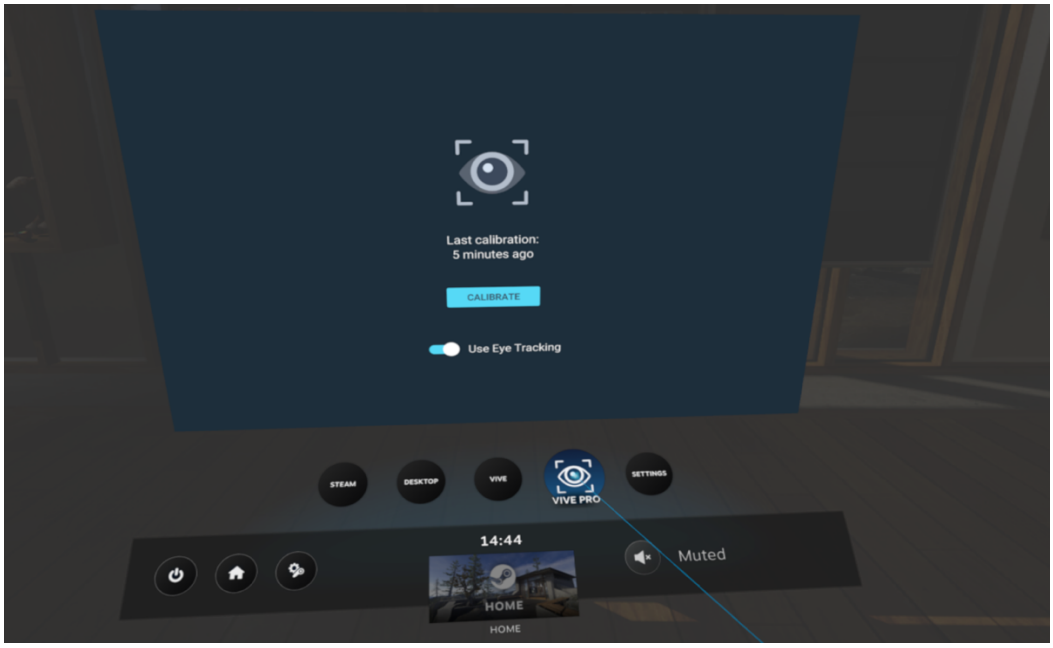
Step 1: Set up your headset

Follow the VIVE Pro Eye Setup guide.

Step 2: Calibrate for your eyes

The VIVE Pro Eye Setup guide will automatically start the calibration at the end of the setup flow.

If you need to re-calibrate or if you switch users, you can always open the calibration tool from the SteamVR dashboard.

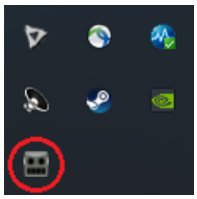


Step 3: Download and Import the VIVE SRanipal SDK

Download the VIVE SRanipal SDK and import the .unitypackage into your Unity project.

Make sure the VIVE Sranipal SDK works before going to the next step.

The tray icon eyes turn green when eye tracking is active; this should happen when your Unity application is running.



Step 4: Import the Tobii XR SDK

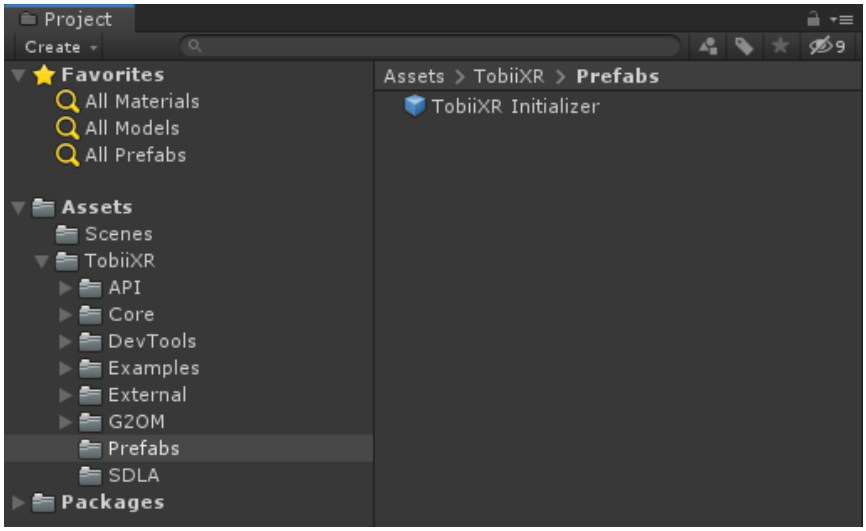
Download the Tobii XR SDK for Unity and import it into your Unity project.

Remember to enable VR support in your Unity project. We support Unity 2018.2.5f1 or later.

If you have an older version of the Tobii XR SDK, remember to remove it before importing the new version to avoid conflicts.

Step 5: Add the TobiiXR Initializer prefab to your scene

The prefab can be found in the Prefabs folder.



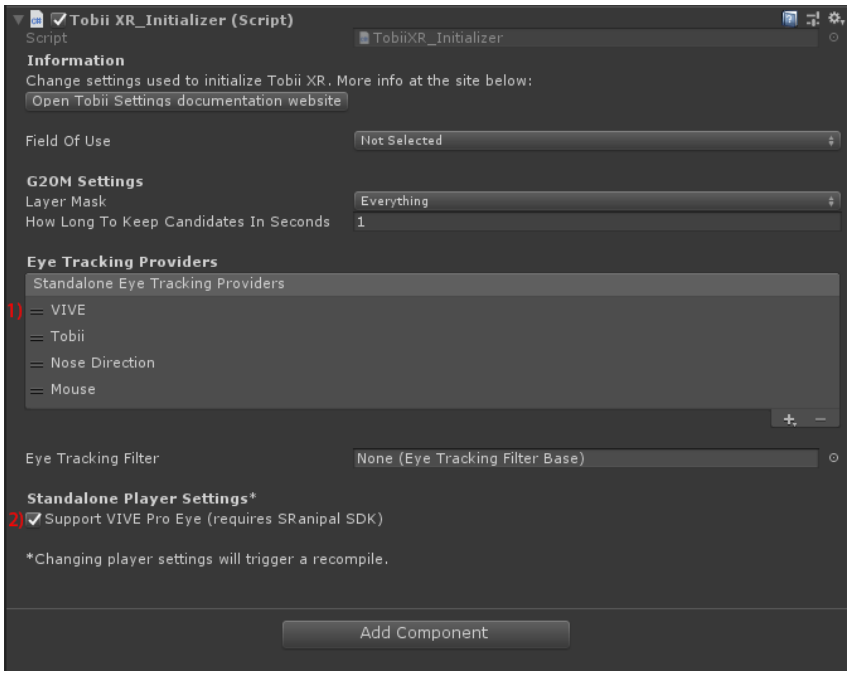
The TobiiXR_Initializer script attached to the prefab calls TobiiXR.Start() to initialize the the Tobii XR SDK.

Step 6: Configure the Tobii XR SDK

To configure the Tobii XR SDK, edit the fields of the TobiiXR Initializer prefab in your scene.

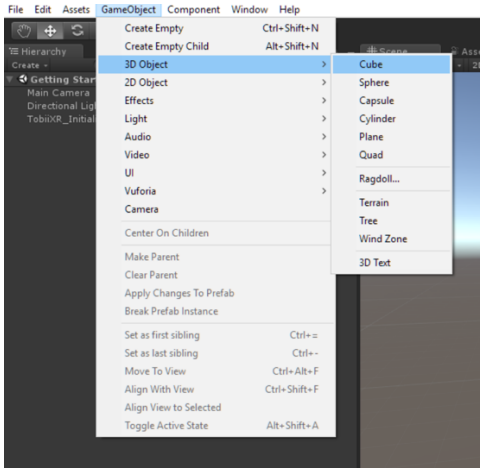
Make sure the **VIVE** provider is at the top of the **Eye Tracking Providers** list.

Enable the **Support VIVE Pro Eye** option.

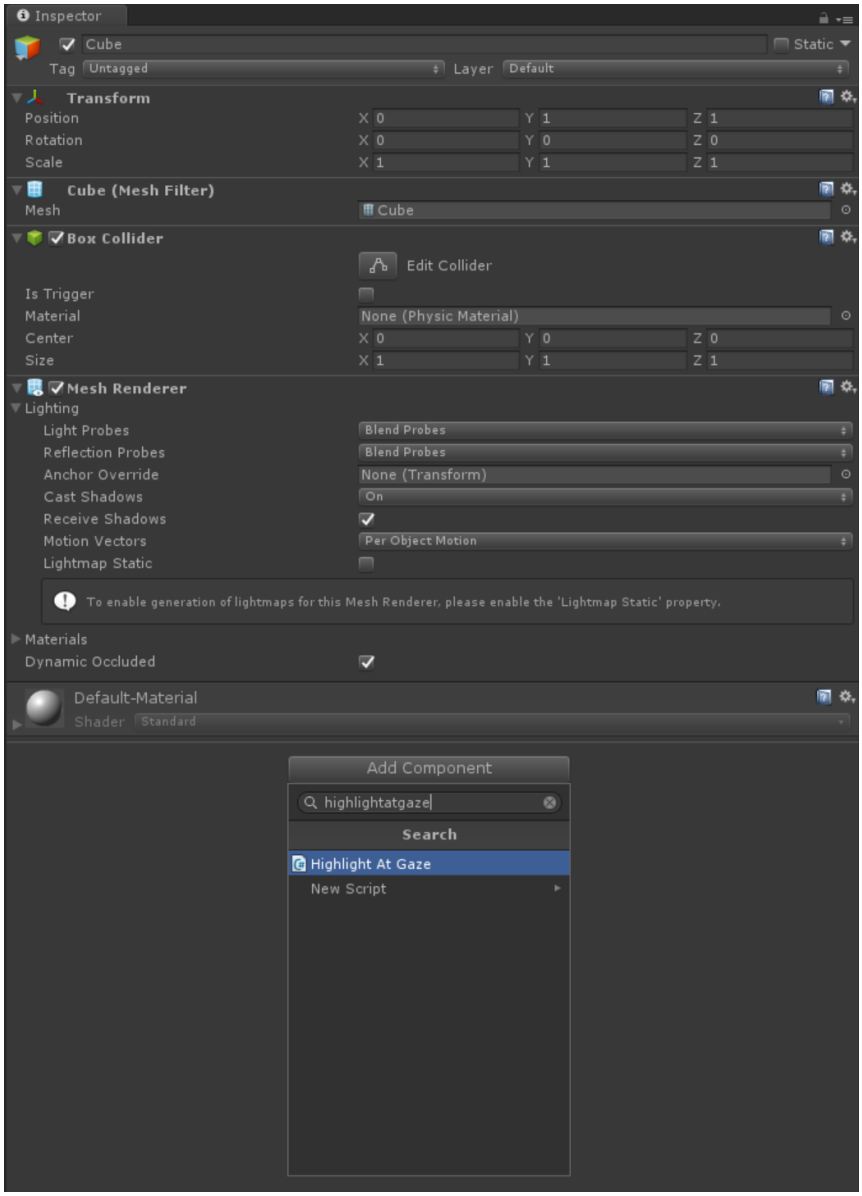


Read more about the providers and other settings in the Tobii XR Settings page.

Step 7: Create a cube and place it somewhere in the scene



Step 8: Add the HighlightAtGaze script to the cube



The `HighlightAtGaze` component implements the `IGazeFocusable` interface, which will be called whenever the object receives or loses focus.

Step 9: Run the scene

By pressing play, you can now highlight the cube by looking at it.

If you want, you can add more objects that react to gaze to the scene in order to test and play around.

If you want to build your solution, make sure to build it for 64-bits.

Next steps

Congratulations! You're now up and running with the Tobii XR SDK for Unity and can start developing with eye tracking in VR.

Check out our Showcases section for demos and prototypes.

Check out the Documentation page, or try out our Unity Examples.

Read our Design section if you want to know more about how to design eye tracking interactions in VR.