UMTL

dfki ai Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

SIC Saarland Informatics Campus

UNIVERSITÄT DES SAARLANDES

Seminar
Sweat and Survive - the VR Edition

Introduction to the Unity 3D-Engine

April 29, 2025

Dr. Felix Kosmalla & Dr. André Zenner
Saarland University & DFKI

Important Notice #1!

Please **submit documents & presentations**
on the day of the deadline **via e-mail to both of us!**

Thanks! ☺

Important Notice #2!

Each of you will receive 2 things today, **please return after the semester**:

(1) 5 Vive Tracker Staps
(2) Chest Strap for Heart Rate Tracker

Both for personal use! ☺

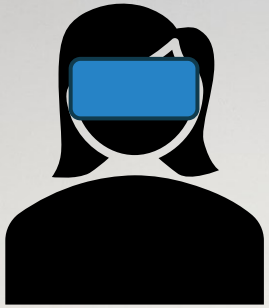
Important Notice #3!

Each group should **send an e-mail to Felix to schedule a meeting in the lab.**

Thanks! ☺

Who has developed with Unity before?

Who has developed a VR/AR project with Unity before?



Today you will learn about …

… basics of working with Unity.

… basics of VR in Unity.

… some tips and tricks that might help you in your project.

Disclaimer!

This is not a comprehensive Unity course!

This session is meant to be:

- a *reminder* for those already familiar with Unity
- a *kick-start on "how to think the Unity way"* for those new to Unity

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

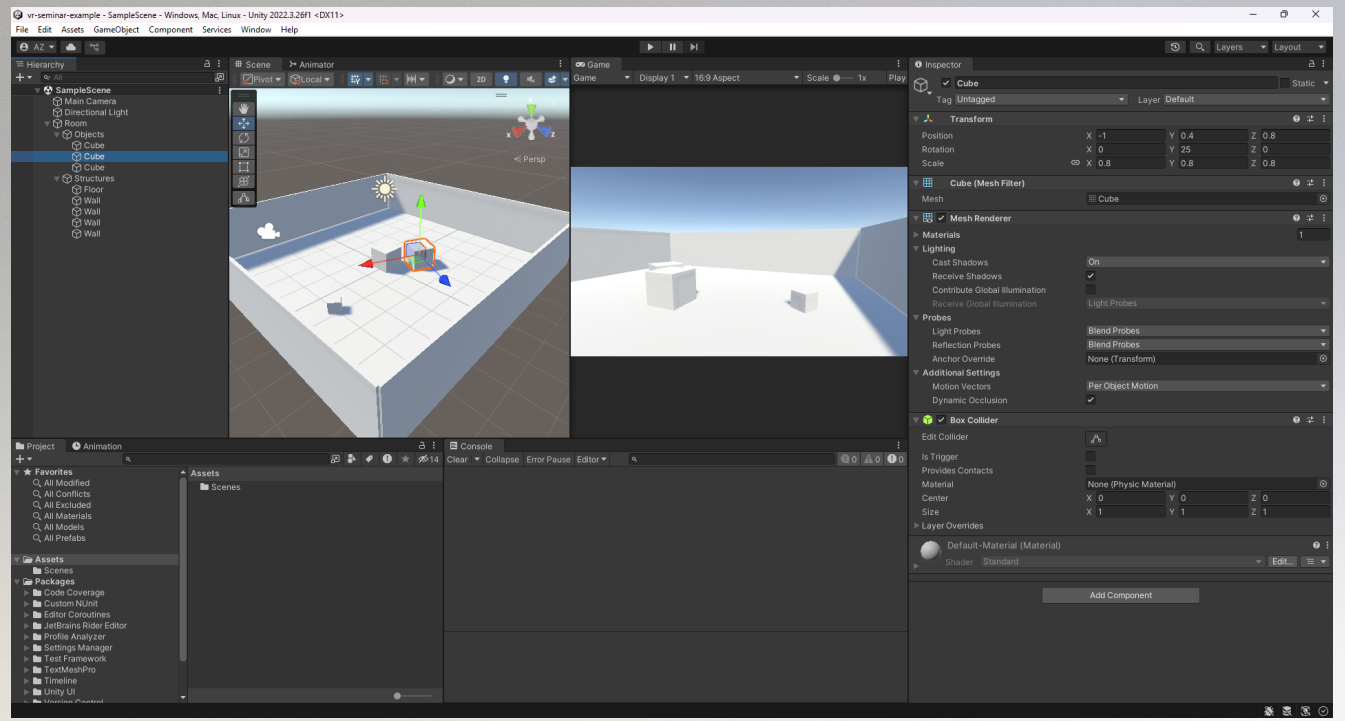
Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Unity

a powerful cross-platform game engine



Unity

a powerful cross-platform game engine

Our impact by the numbers

Over 1.2 million

Over 1.2 million monthly active users using the Unity Editor[1]

3B downloads per month

Made with Unity games on mobile averaged 3B downloads per month[2]

70% of the top 1000 mobile games

More than 70% of the top 1000 mobile games are made with Unity[3]

28% of the top 1,000 PC games

At least 28% of the top 1,000 PC games on Steam are made with Unity[4]

70% of top-selling VR games

More than 70% of top-selling VR games on Meta Store are made with Unity[5]

68 billion impressions

Unity Ads and ironSource ads serve over 68 billion impressions each month[6]

Source: Unity Website (https://unity.com/our-company)

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

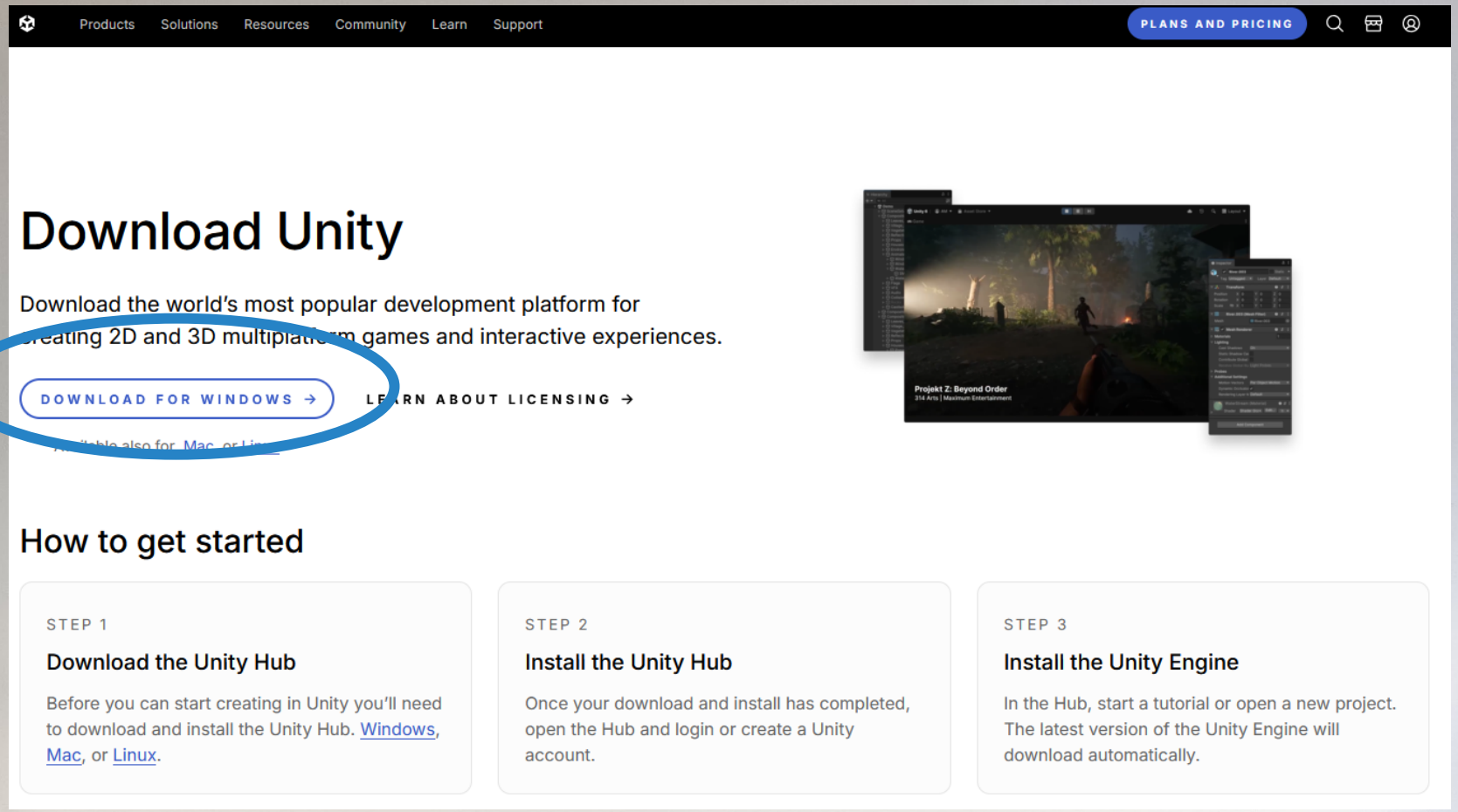
Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Unity Setup

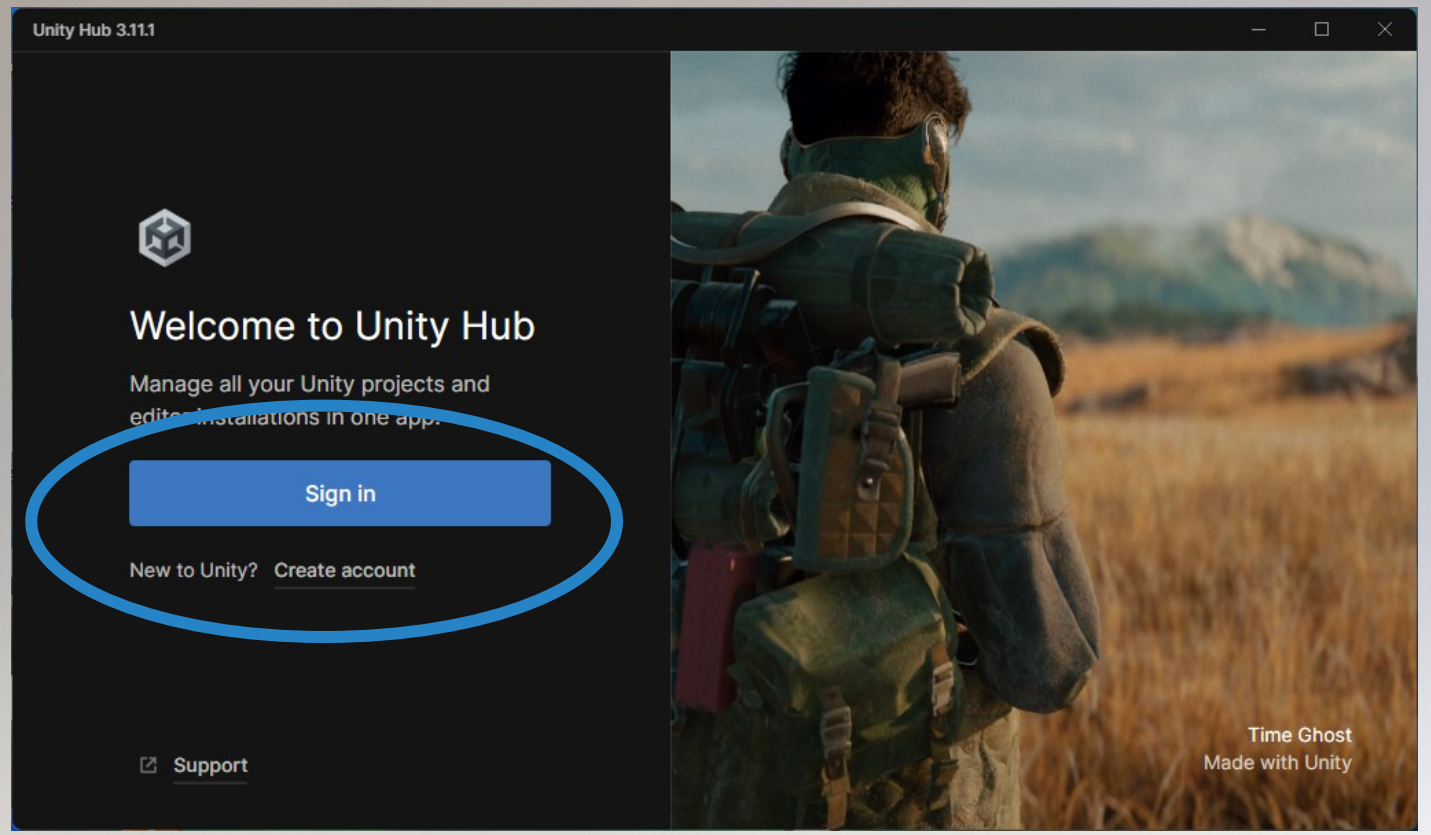
Step 1 – Download & Install the Unity Hub



https://unity.com/download

Unity Setup

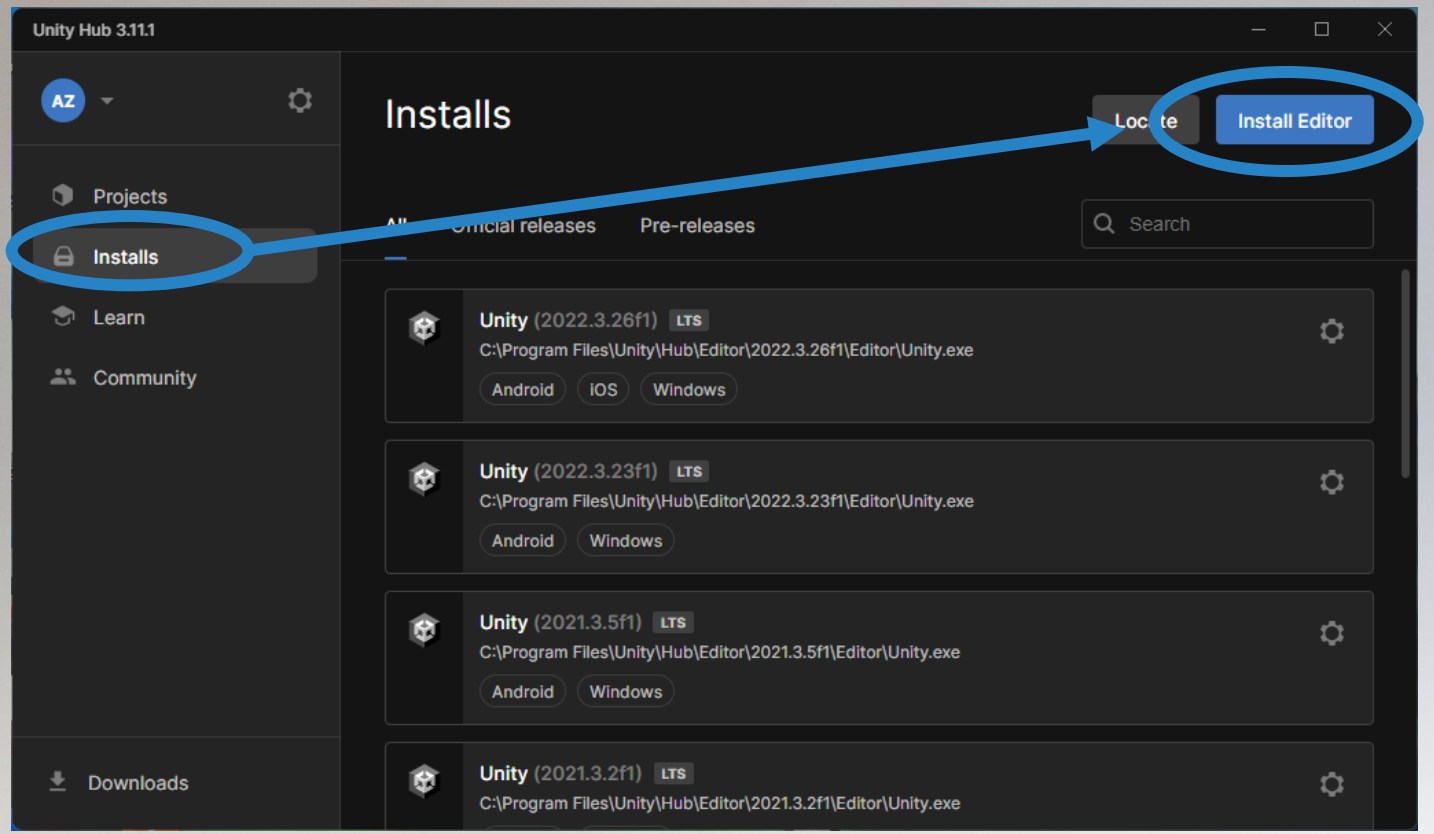
Step 2 – Start the Unity Hub & Log In



If you don't have an account, create one with your student mail address.
(… and Steam, and Microsoft potentially)

Unity Setup

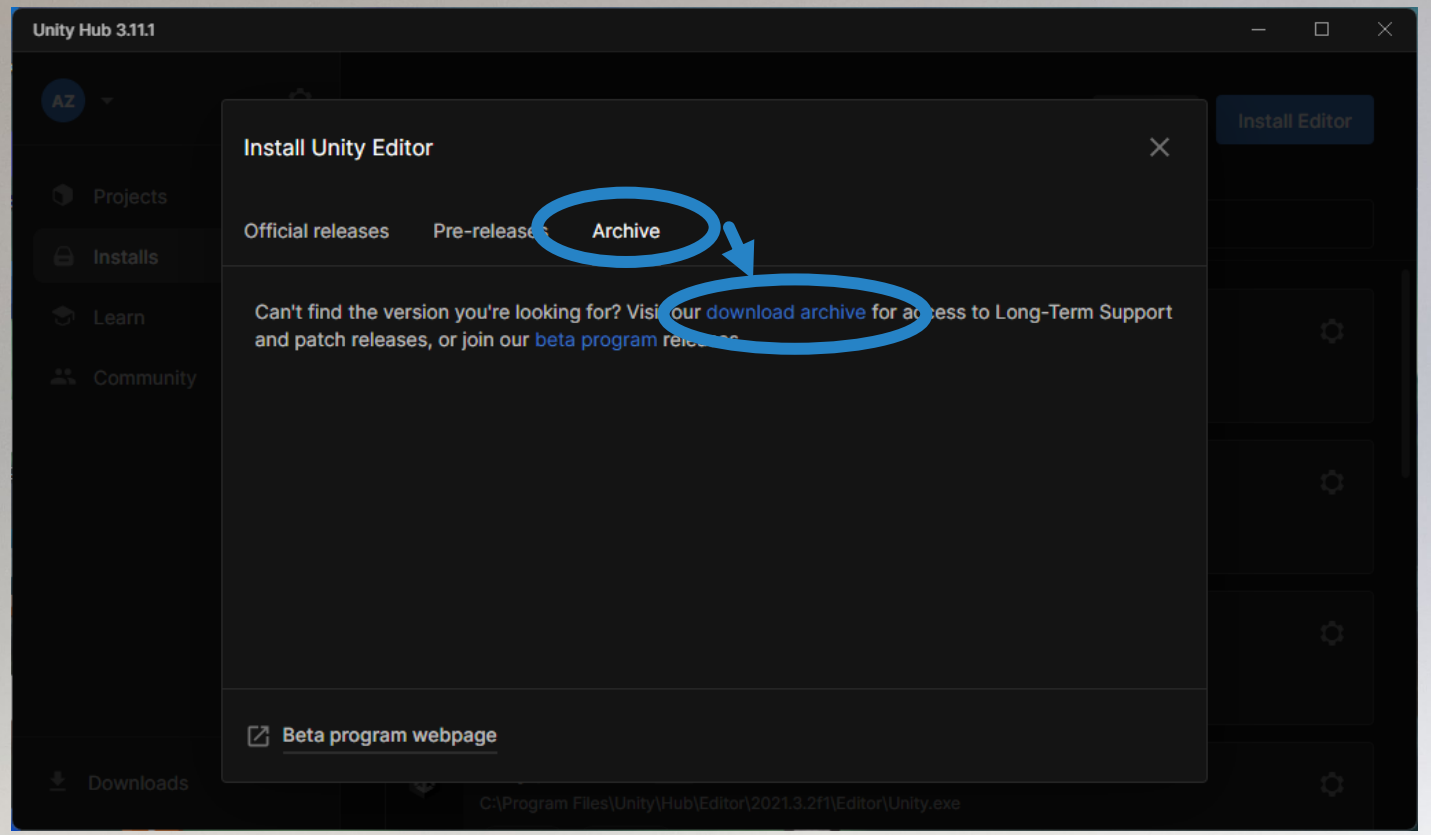
Step 3 – Install Unity version **2021.3.38f1**



In the Unity Hub:
Installs → Install Editor

Unity Setup

Step 3 – Install Unity version **2021.3.38f1**

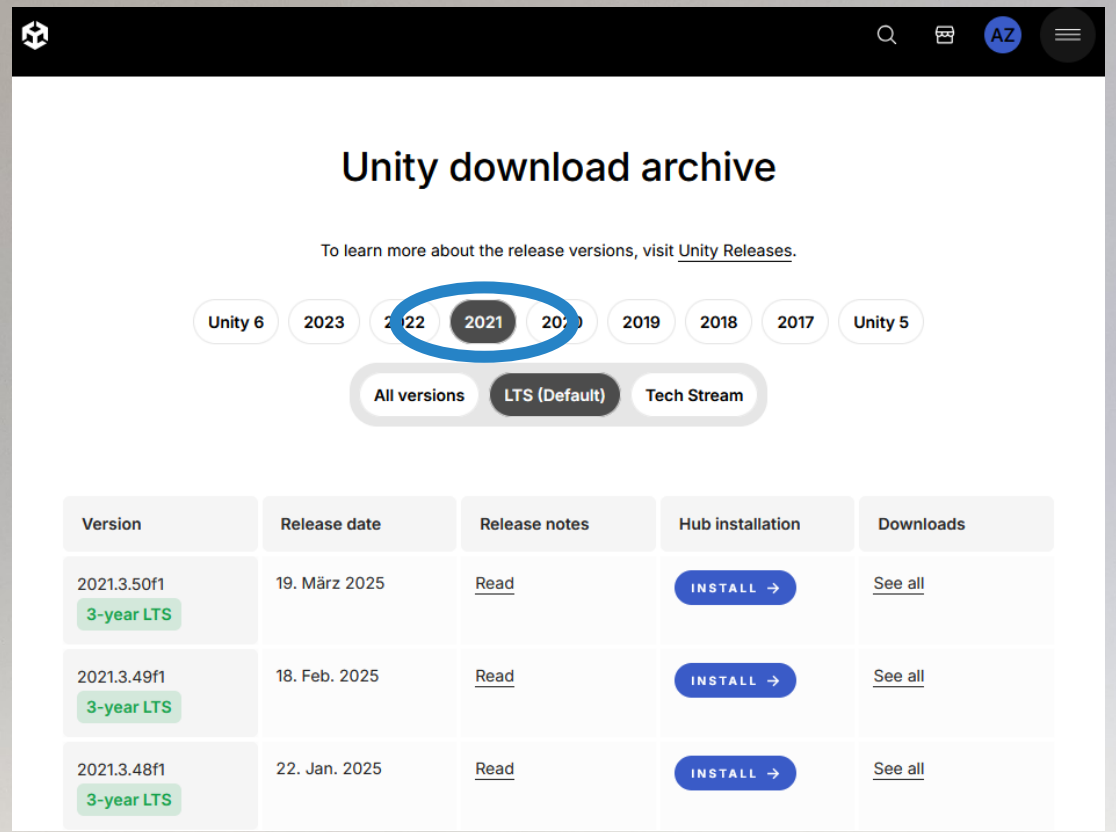


In the Unity Hub:

Installs → Install Editor → Archive → Download Archive

Unity Setup

Step 3 – Install Unity version **2021.3.38f1**

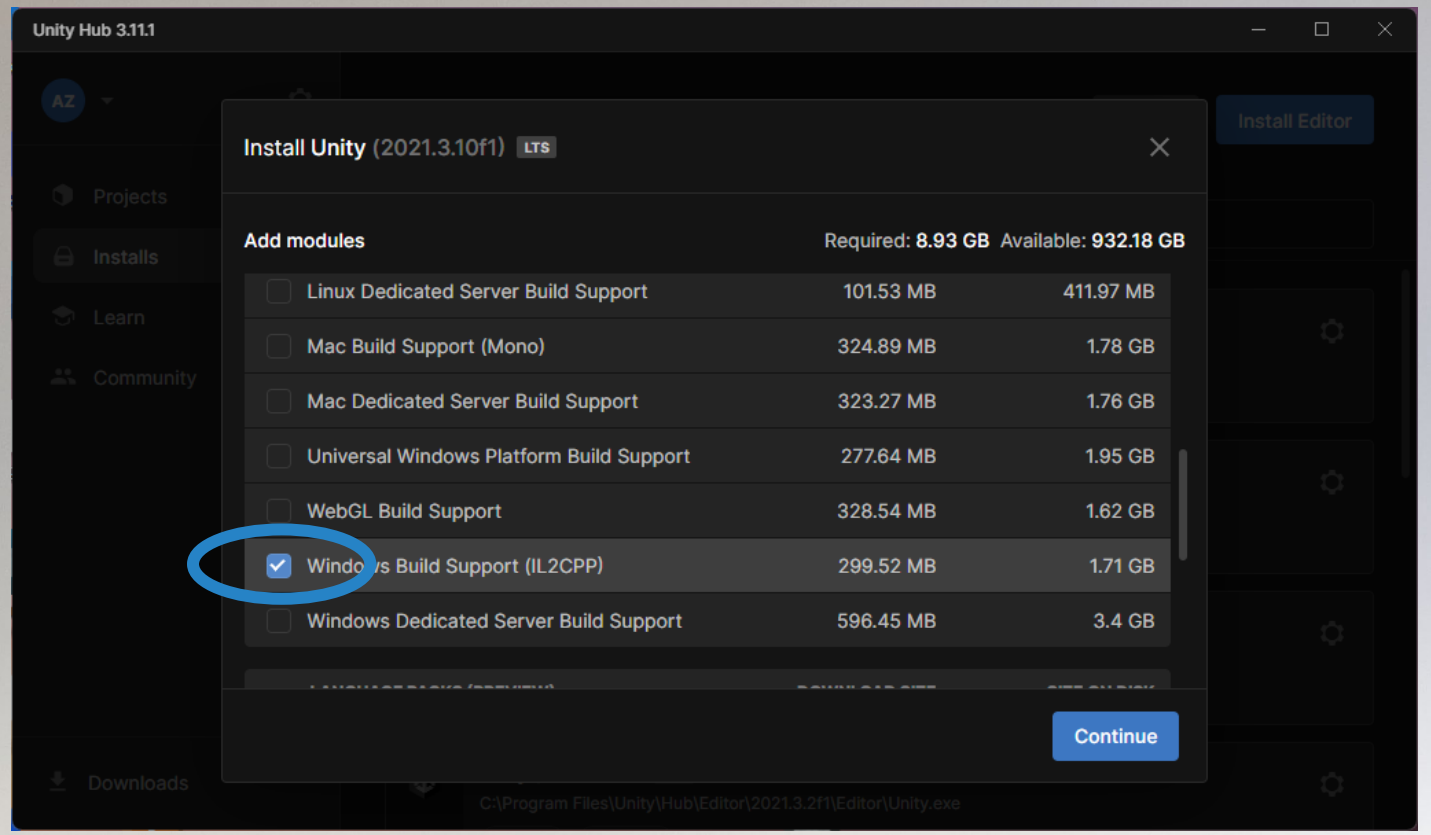


In the Unity Hub:

Installs → Install Editor → Archive → Download Archive → 2021 → scroll down and install **2021.3.38f1**

Unity Setup

Step 3 – Install Unity version **2021.3.38f1**



In the Unity Hub:

Installs → Install Editor → Archive → Download Archive → 2021 → scroll down and install **2021.3.38f1**

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

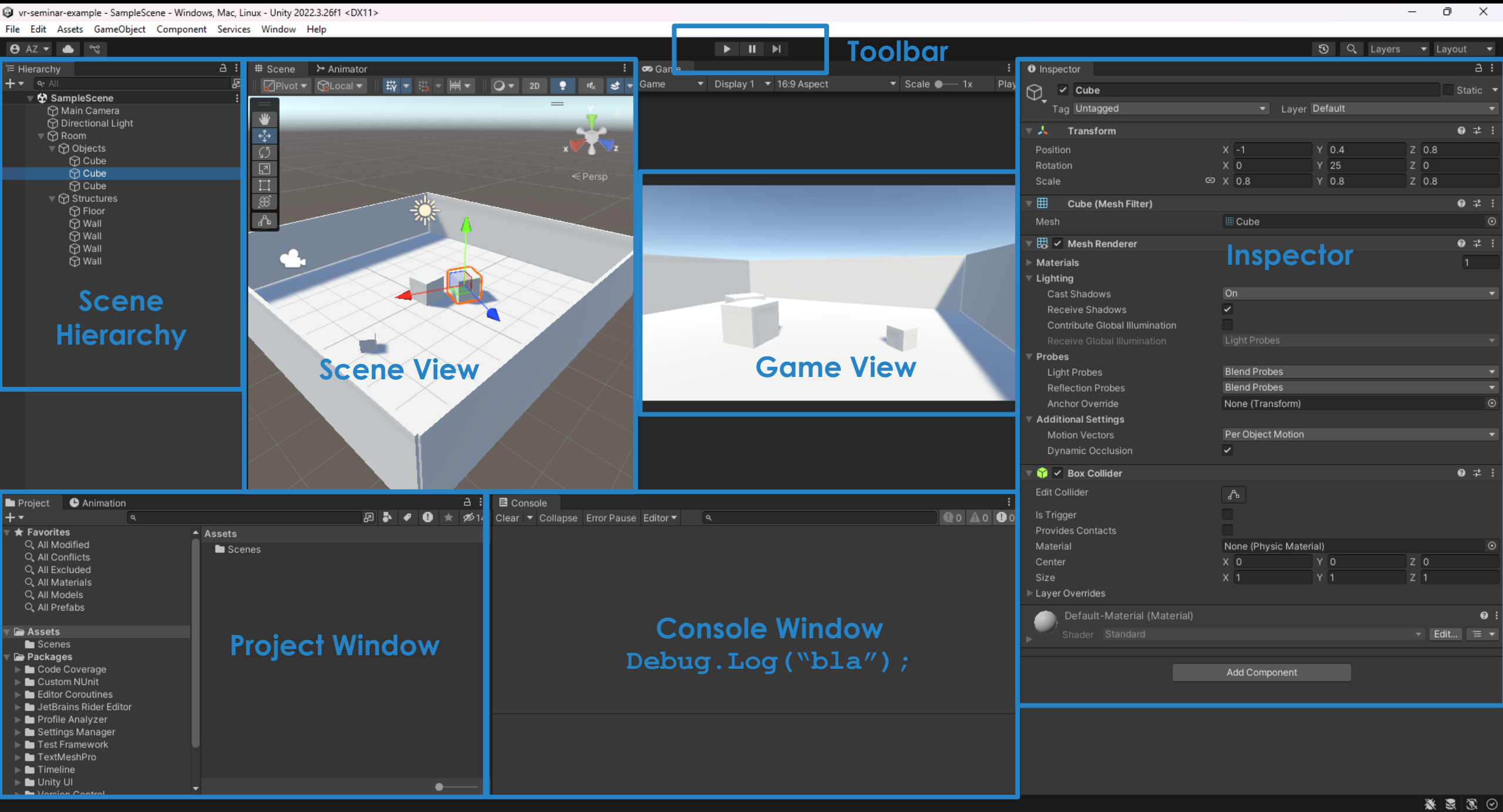
General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

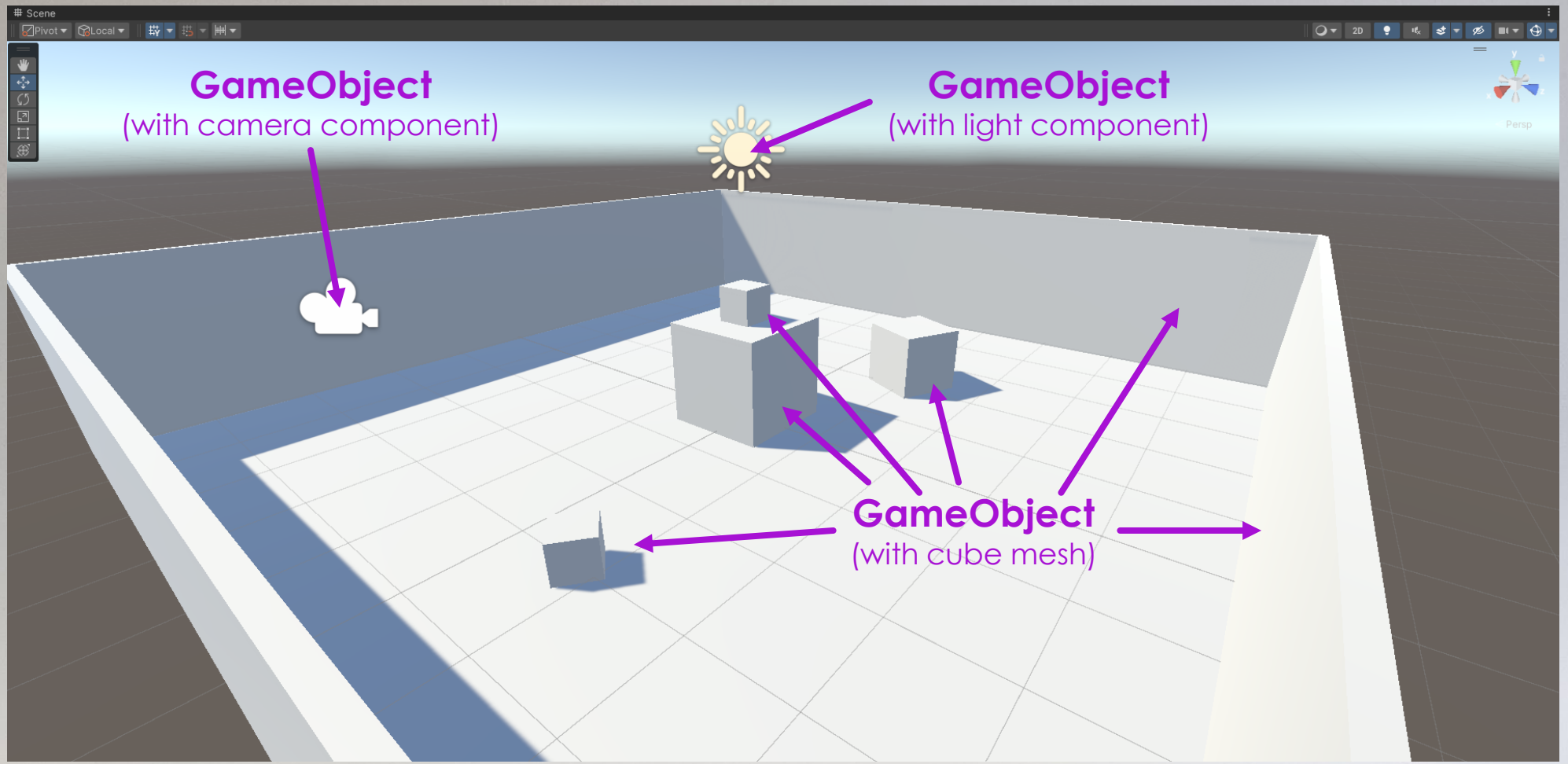
- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers


vr-seminar-example - SampleScene - Windows, Mac, Linux - Unity 2022.3.26f1 <DX11>
File Edit Assets GameObject Component Services Window Help
Toolbar
Hierarchy
SampleScene
Main Camera
Directional Light
Room
Objects
Cube
Cube
Cube
Structures
Floor
Wall
Wall
Wall
Wall
Scene Hierarchy
Scene
Animator
Persp
Scene View
Game
Display 1
16:9 Aspect
Scale 1x
Game View
Inspector
Cube
Static
Tag Untagged
Layer Default
Transform
Position X -1 Y 0.4 Z 0.8
Rotation X 0 Y 25 Z 0
Scale X 0.8 Y 0.8 Z 0.8
Cube (Mesh Filter)
Mesh Cube
Mesh Renderer
Materials 1
Lighting
Cast Shadows On
Receive Shadows
Contribute Global Illumination
Receive Global Illumination Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Per Object Motion
Dynamic Occlusion
Box Collider
Edit Collider
Is Trigger
Provides Contacts
Material None (Physic Material)
Center X 0 Y 0 Z 0
Size X 1 Y 1 Z 1
Layer Overrides
Default-Material (Material)
Shader Standard
Edit...
Add Component
Inspector
Project
Animation
Favorites
All Modified
All Conflicts
All Excluded
All Materials
All Models
All Prefabs
Assets
Scenes
Packages
Code Coverage
Custom NUnit
Editor Coroutines
JetBrains Rider Editor
Profile Analyzer
Settings Manager
Test Framework
TextMeshPro
Timeline
Unity UI
Assets
Scenes
Project Window
Console
Clear Collapse Error Pause Editor
Console Window
Debug.Log("bla");

The Scene

defines a virtual environment as a collection of GameObjects



State of the scene in **frame 0**

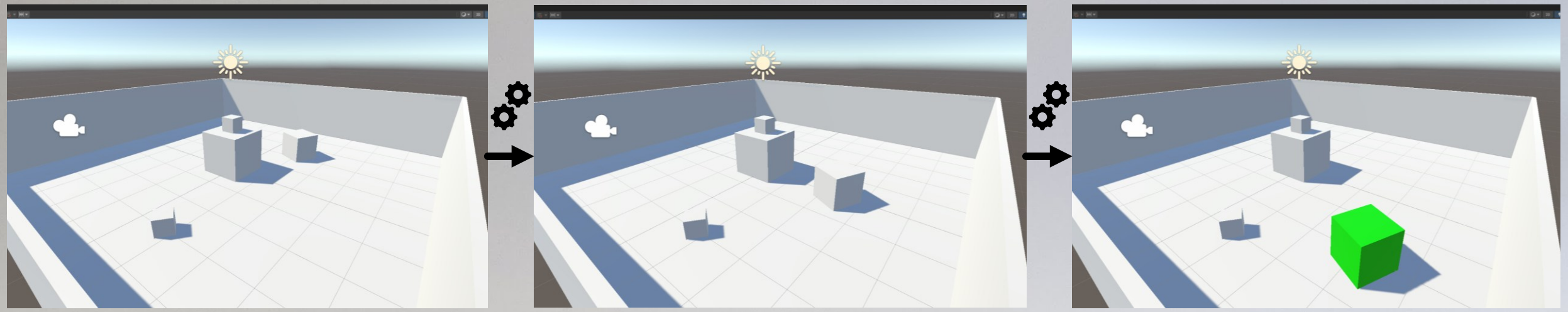
The Scene

"lives" by changing from state to state (here called: frames)

all GameObjects are in their start configuration

cube position values changed

cube position values changed
+
cube material changed



Frame 0

Frame 1

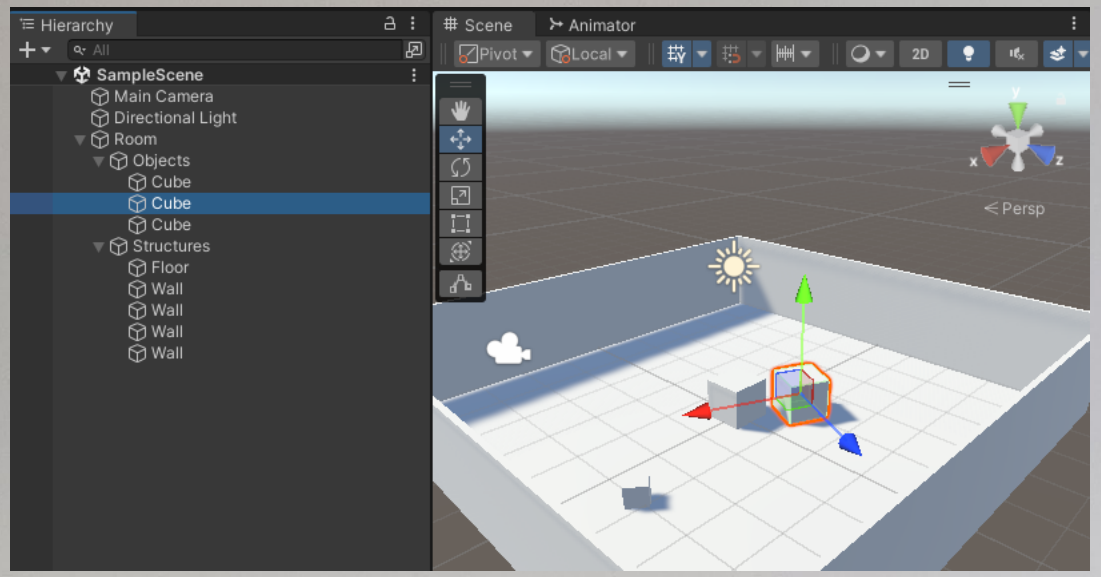
Frame 2

For VR, the goal is at least 90 frames per second (FPS)

= 11 milliseconds max. computation per frame

The Scene Hierarchy

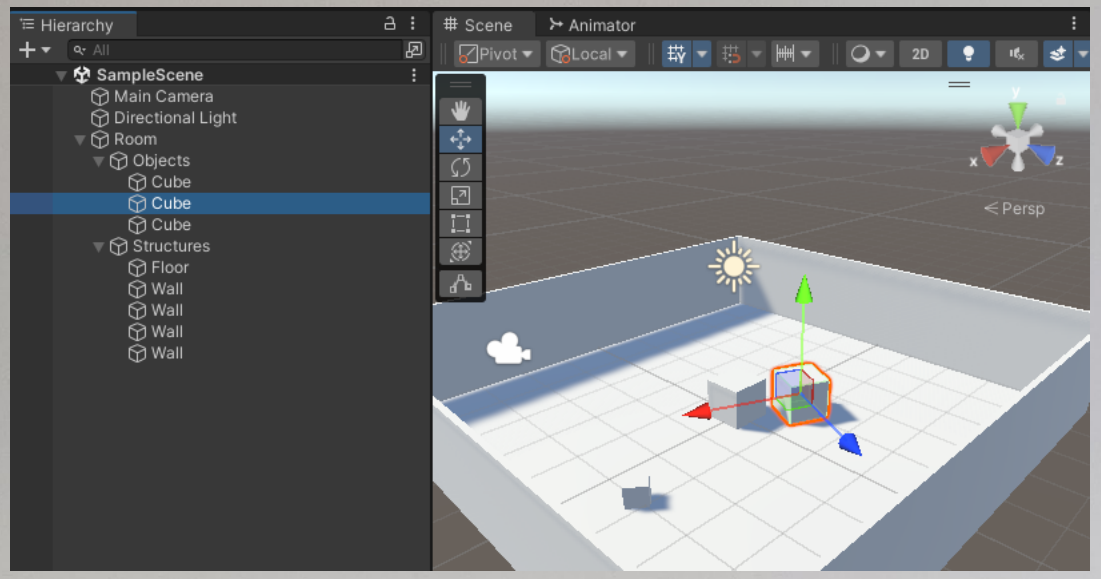
organizes all GameObjects in a tree structure

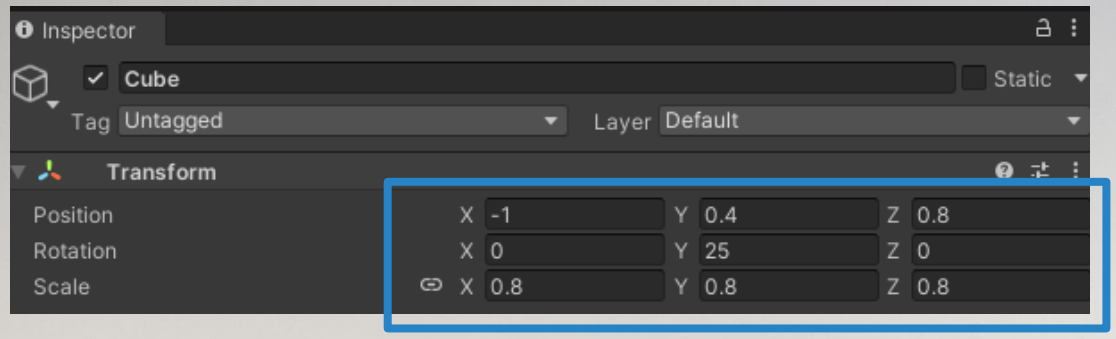


- **Hierarchical Structure**
  All GameObjects in the scene are organized in a parent-child hierarchy. Helps keep the scene organized by grouping related objects together.
- **Transformation Propagation**
  Moving, rotating, or scaling a parent affects all its children, enabling grouped transformations.
- **Activation Control**
  Activating or deactivating a parent GameObject also affects all its children, simplifying visibility and interaction management.

Transformation

defines the position and orientation of a GameObject relative to its parent



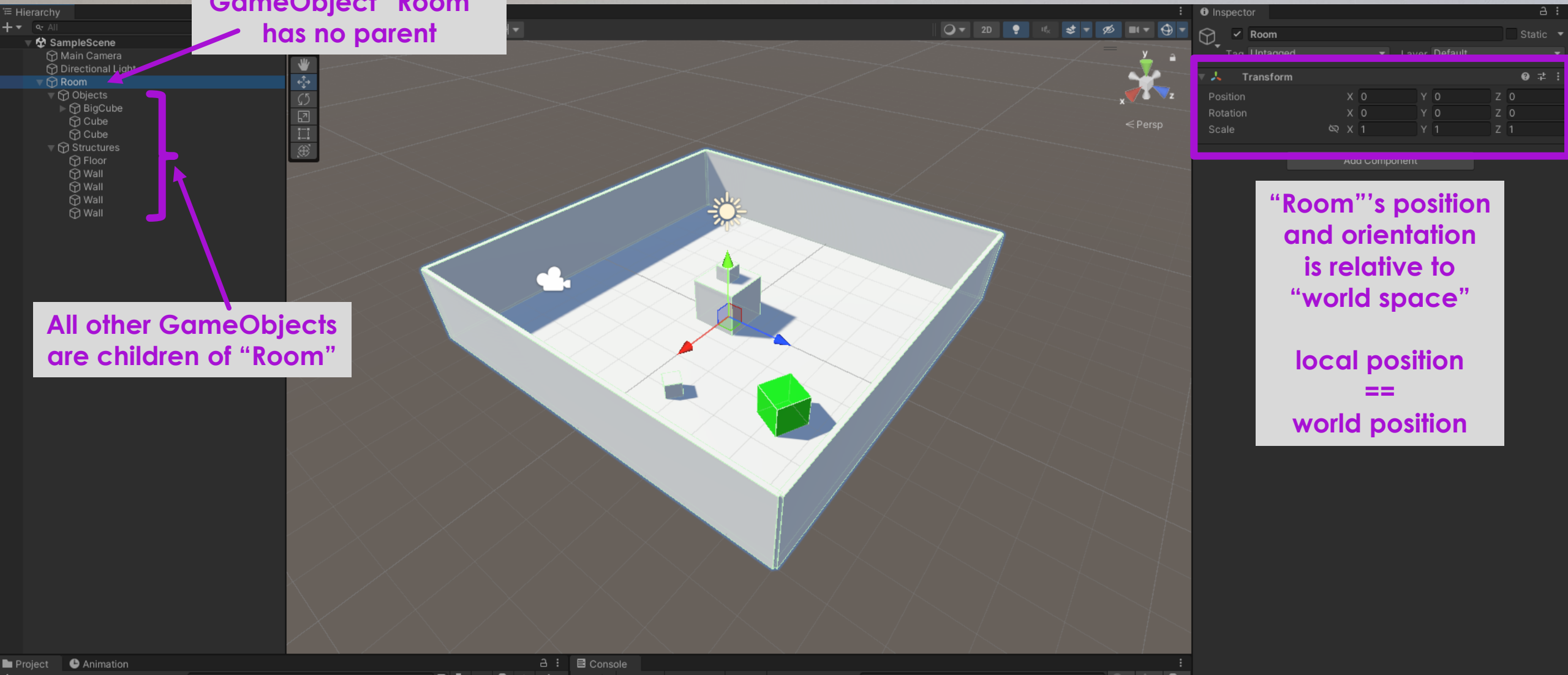


- **Hierarchical Structure**
  All GameObjects in the scene are organized in a parent-child hierarchy. Helps keep the scene organized by grouping related objects together.
- **Transformation Propagation**
  Moving, rotating, or scaling a parent affects all its children, enabling grouped transformations.
- **Activation Control**
  Activating or deactivating a parent GameObject also affects all its children, simplifying visibility and interaction management.

always relative to the parent object's position and orientation!

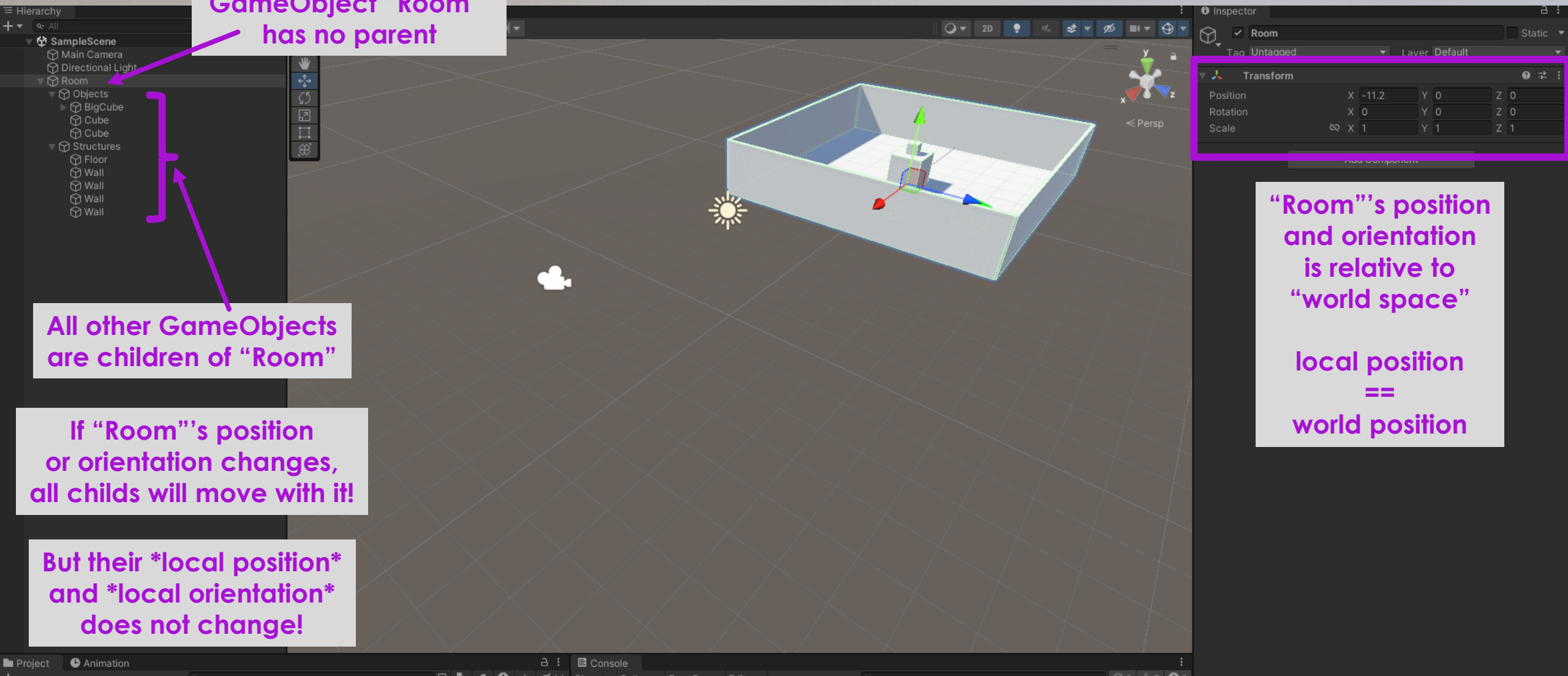
Hierarchical Structure

GameObjects are organized in a parent-child hierarchy.
Allows grouping related objects together.



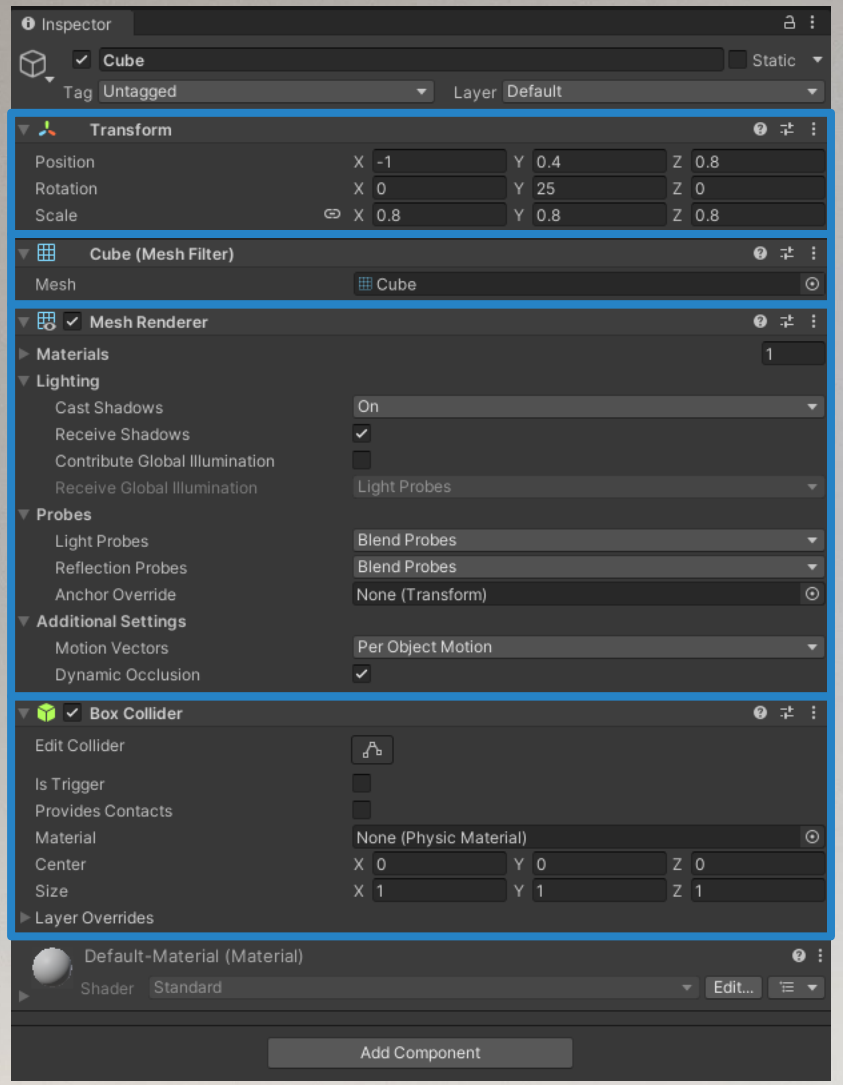
Transformation Propagation

Moving, rotating, or scaling a parent affects all its children



Components

define the behaviors and properties of a GameObject



position & orientation

mesh

rendering of the mesh

collider

- **Components are attached to GameObjects**
  Define behaviors or properties of a GameObject. Modular.
- **Reusability**
  Can be reused across different GameObjects.
- **Customization**
  By combining different components, you can customize the behavior and appearance of GameObjects.
- **Built-in Components**
  Unity provides a variety of built-in components, such as Rigidbody for physics, Collider for collision detection, and Renderer for visual representation.
- **Script Components**
  You can create custom components using C# scripts to add specific functionality to GameObjects.
- **Inspector Integration**
  Components are managed and configured through the Inspector window, allowing for easy adjustments and fine-tuning.

Quick Live Demo

Hierarchy, Transformations, Components

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

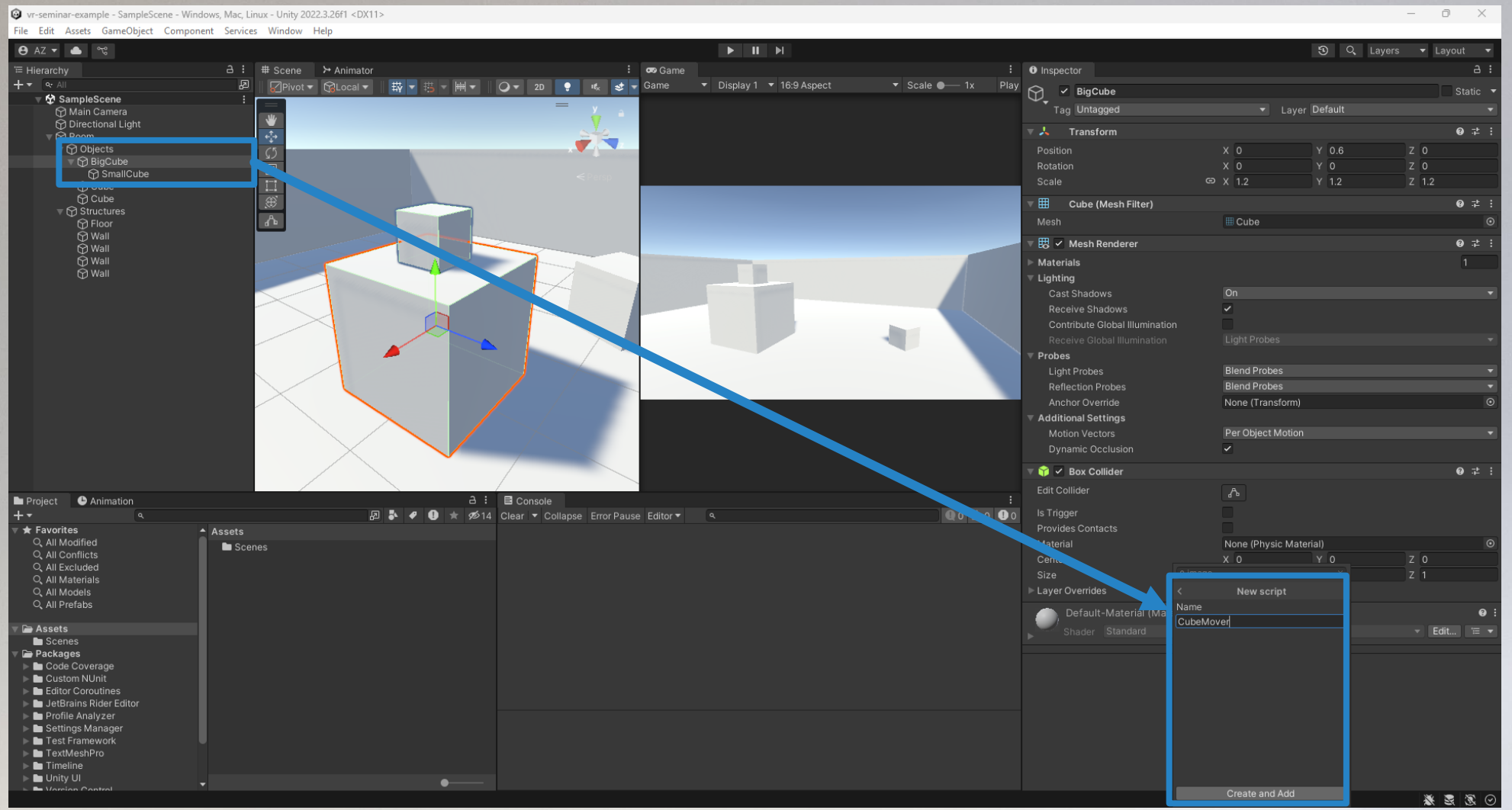
Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

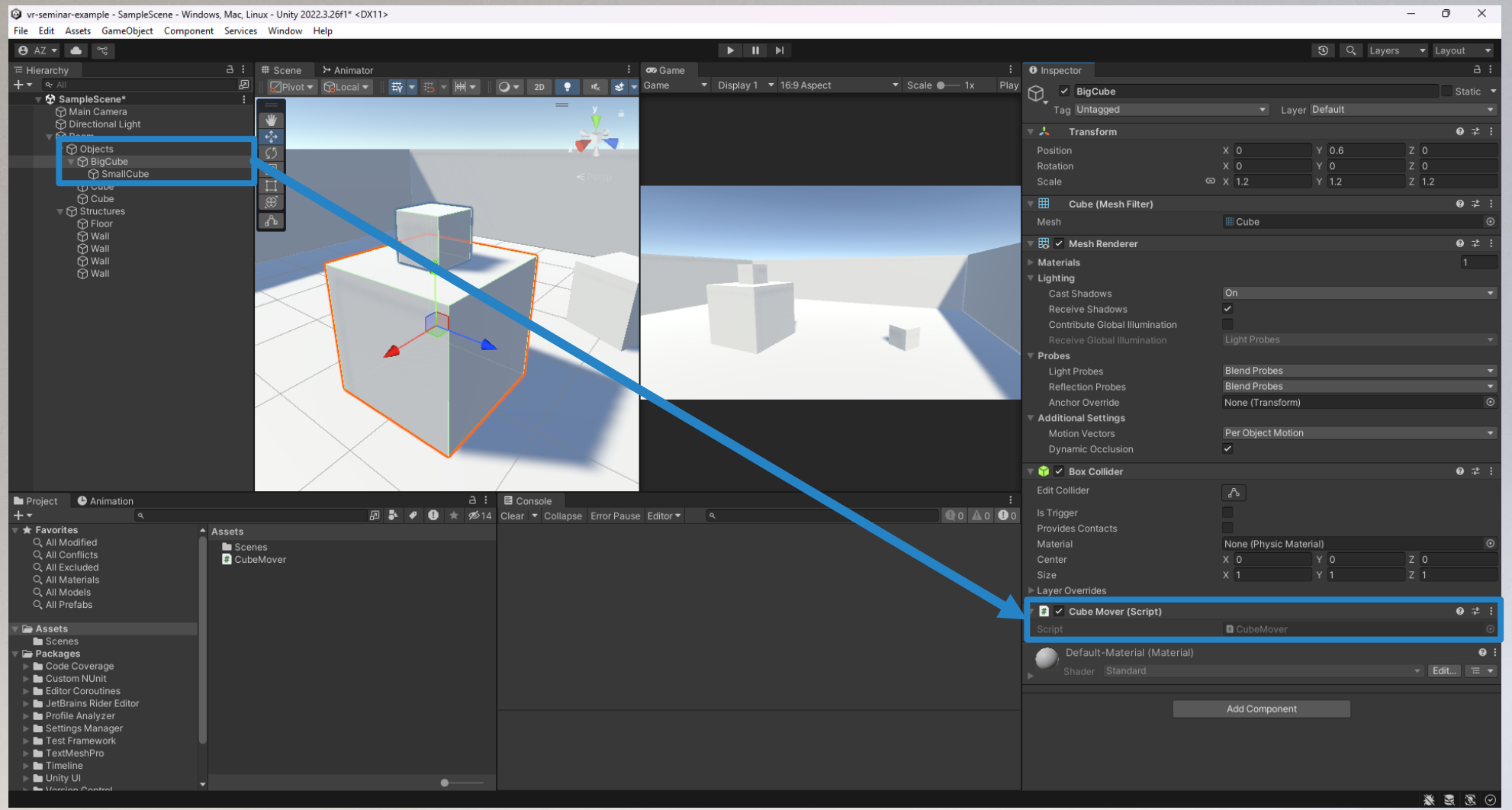
Script Components

allow you to program the behavior of GameObjects



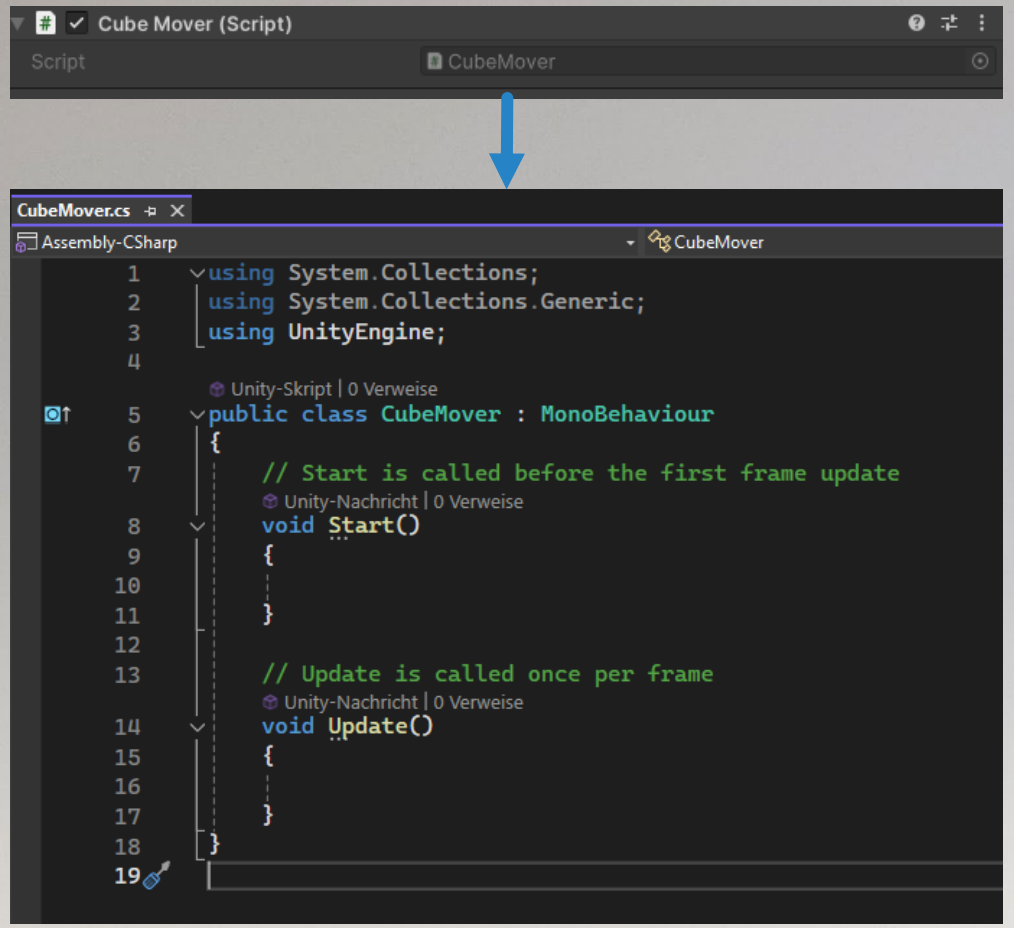
Script Components

allow you to program the behavior of GameObjects

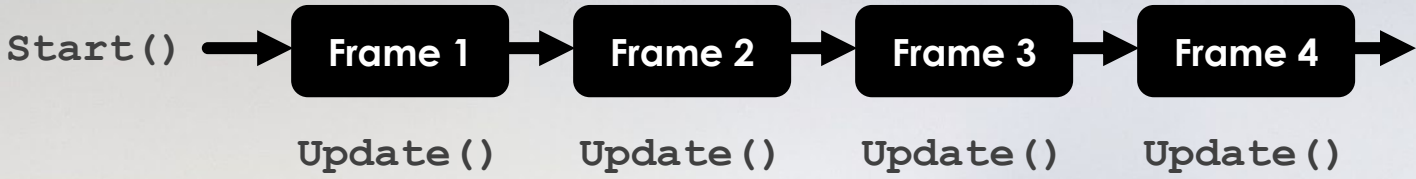


Script Components

allow you to program the behavior of GameObjects

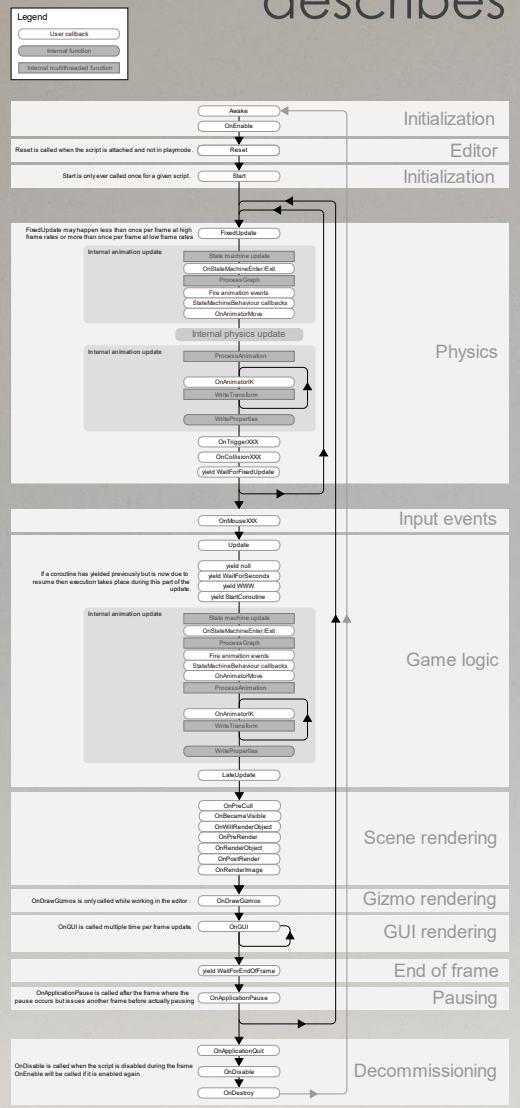


- **C# Scripting**
  Program the behavior of GameObjects with C#.
  Easy to program and easy to debug.
- **Comprehensive Framework**
  Unity offers many very useful pre-implemented functions and data structures (covering, e.g., Math, Physics, Input, Output, Audio, Rendering, etc.).
  Scripts can access GameObjects and their components.
- **Basic Concept: Frames**
  As a 3D engine, Unity is basically running a real-time "simulation" of your scene, computing the state of the scene one frame after the other.



Script Lifecycle

describes which function is called when during the life of a Unity scene



common functions you can add logic to

Your script needs to inherit from the **`MonoBehaviour`** class

`Awake()` called once before anything else

↓

`OnEnable()` called every time the object becomes active

↓

`Start()` called only once before the first frame update

↓

`Update()` called once per frame

↓

`OnDisable()` called when the script is disabled during the frame

↓

`OnDestroy()` called when the GameObject is destroyed

Full list of functions online:
https://docs.unity3d.com/6000.0/Documentation/Manual/execution-order.html

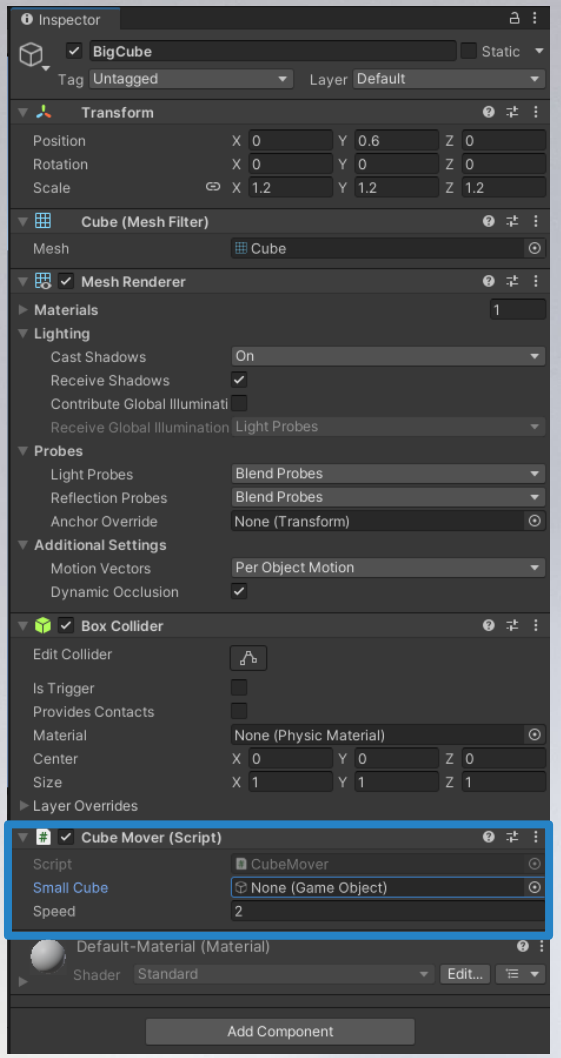
Script Example

scripting the small cube to move up and down

CubeMover.cs

Assembly-CSharp | CubeMover | Update()

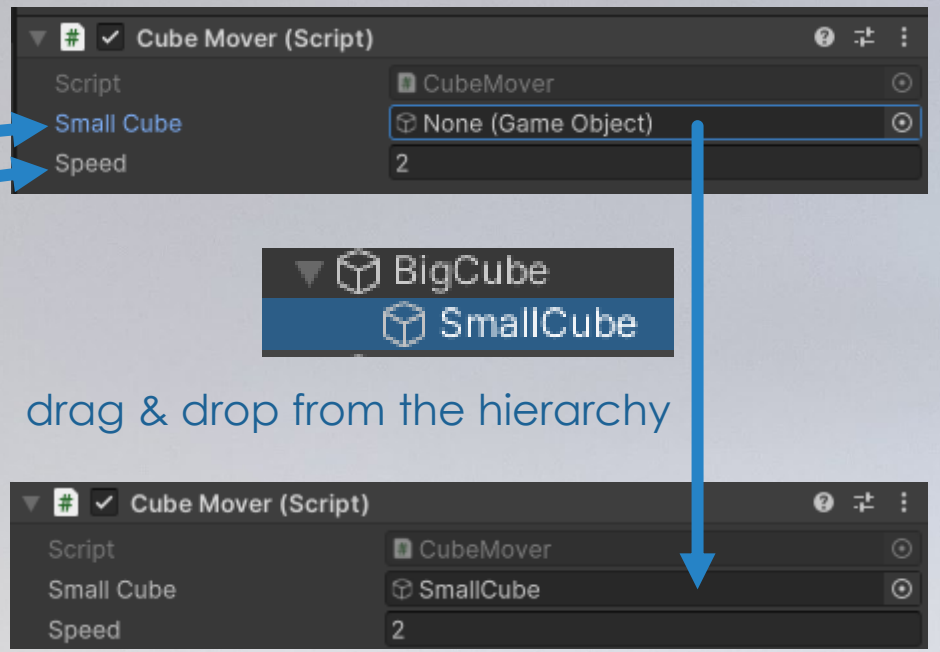
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

/// <summary>
/// A script moving the small cube on top of the big cube up and down, and spinning it around.
/// </summary>
Unity-Skript (1 Objektverweis) | 0 Verweise
public class CubeMover : MonoBehaviour
{
    // Reference to the small cube
    public GameObject smallCube;

    // Speed of movement
    public float speed = 2.0f;

    // Initial position of the small cube
    protected Vector3 initialPosition;

    Unity-Nachricht | 0 Verweise
    void Start()
    {
        // Store the initial position of the small cube
        initialPosition = smallCube.transform.localPosition;
    }

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        // Move the small cube up and down
        float newY = initialPosition.y + Mathf.Sin(Time.time * speed);
        smallCube.transform.localPosition = new Vector3(initialPosition.x, newY, initialPosition.z);
    }

}
```



Script Example

scripting the small cube to move up and down

```
CubeMover.cs
Assembly-CSharp                CubeMover                Update()
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   /// <summary>
6   /// A script moving the small cube on top of the big cube up and down, and spinning it around.
7   /// </summary>
        Unity-Skript (1 Objektverweis) | 0 Verweise
8   public class CubeMover : MonoBehaviour
9   {
10      // Reference to the small cube
11      public GameObject smallCube;
12
13      // Speed of movement
14      public float speed = 2.0f;
15
16      // Initial position of the small cube
17      protected Vector3 initialPosition;
18
            Unity-Nachricht | 0 Verweise
19      void Start()
20      {
21          // Store the initial position of the small cube
22          initialPosition = smallCube.transform.localPosition;
23      }
24
            Unity-Nachricht | 0 Verweise
25      void Update()
26      {
27          // Move the small cube up and down
28          float newY = initialPosition.y + Mathf.Sin(Time.time * speed);
29          smallCube.transform.localPosition = new Vector3(initialPosition.x, newY, initialPosition.z);
30      }
31
32  }
33
```



Quick Live Demo

Moving Cube

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

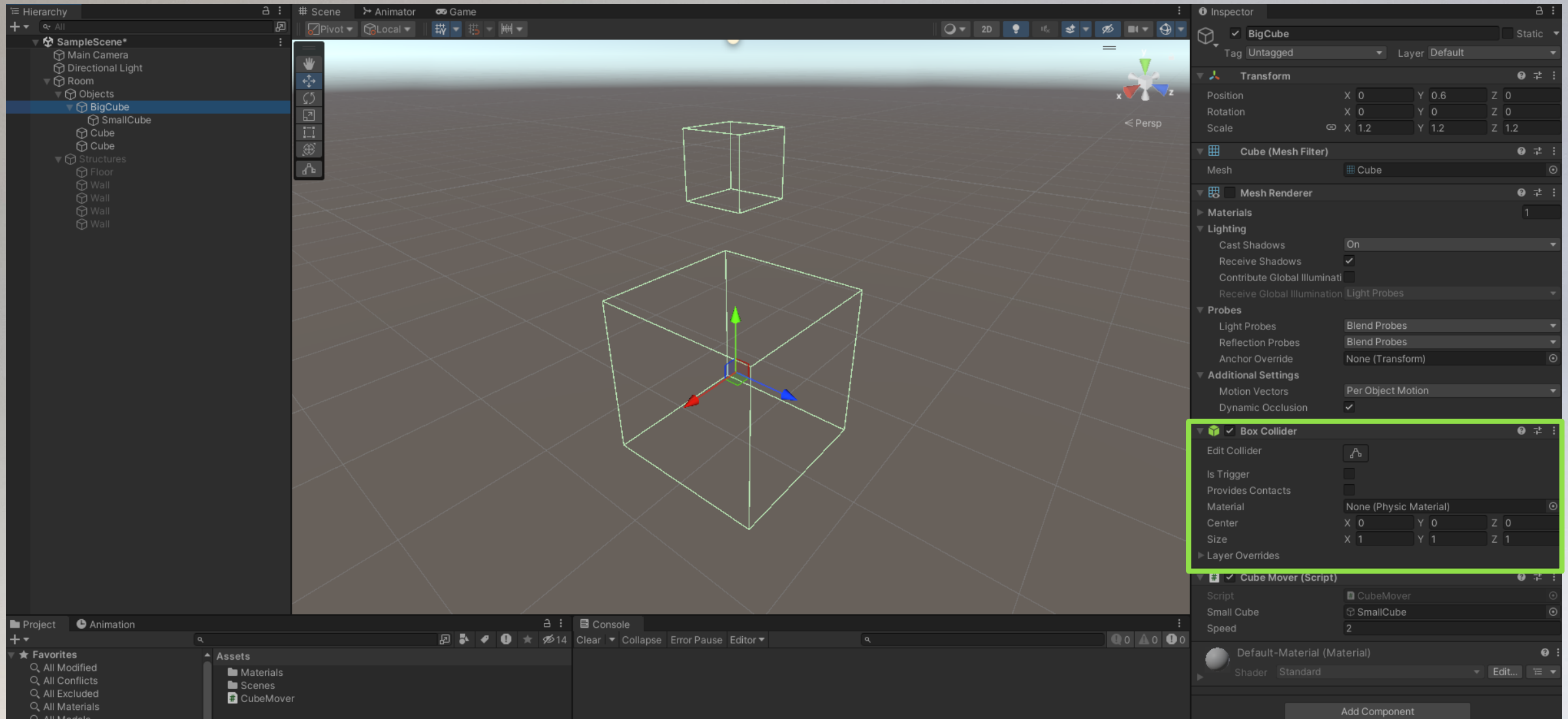
Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

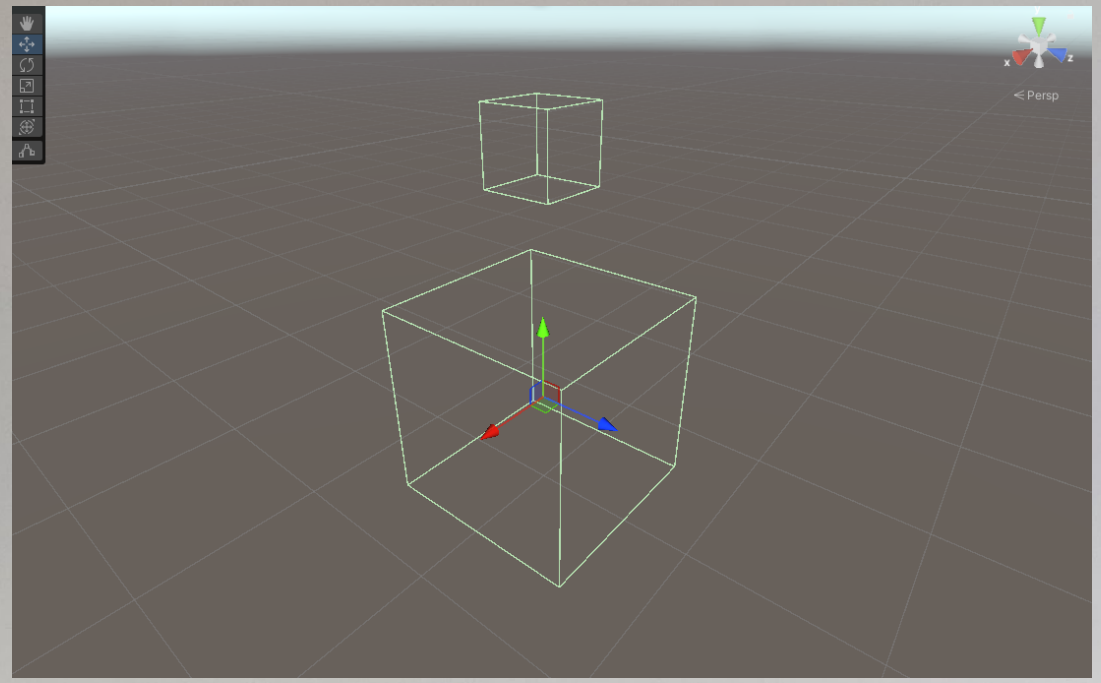
Colliders

allow you to detect and react to intersections of GameObjects



Colliders

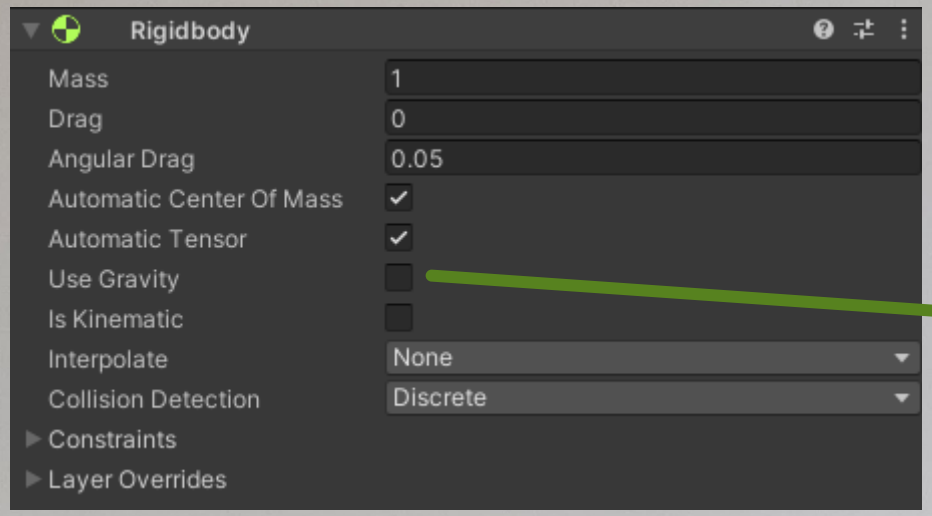
allow you to detect and react to intersections of GameObjects

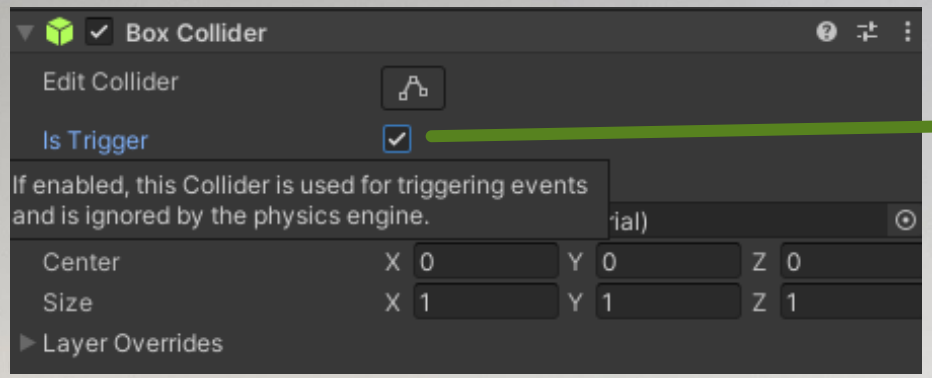


- **Collision Detection**
  Colliders define the shape of GameObjects for physical interactions. **Need an additional Rigidbody component!**
- **Trigger Events**
  Enable detection of overlaps and trigger custom behaviors.
- **Types**
  Various types like BoxCollider, SphereCollider, and MeshCollider for different shapes.
- **Physics Simulation**
  Work with Rigidbody components to simulate realistic physics.

Triggers

allow you to detect and react to intersections of GameObjects





- **Setting up a Trigger Collider:**
  To detect collisions between two objects while ignoring the objects in physics computations:
  - add a Rigidbody component to both objects
  - set `UseGravity` to `false` on both
  - make sure both objects have a Collider
  - set `IsTrigger` to `true` on both
  - add a script to one of the objects, which implements `OnTriggerEnter()` and `OnTriggerExit()`

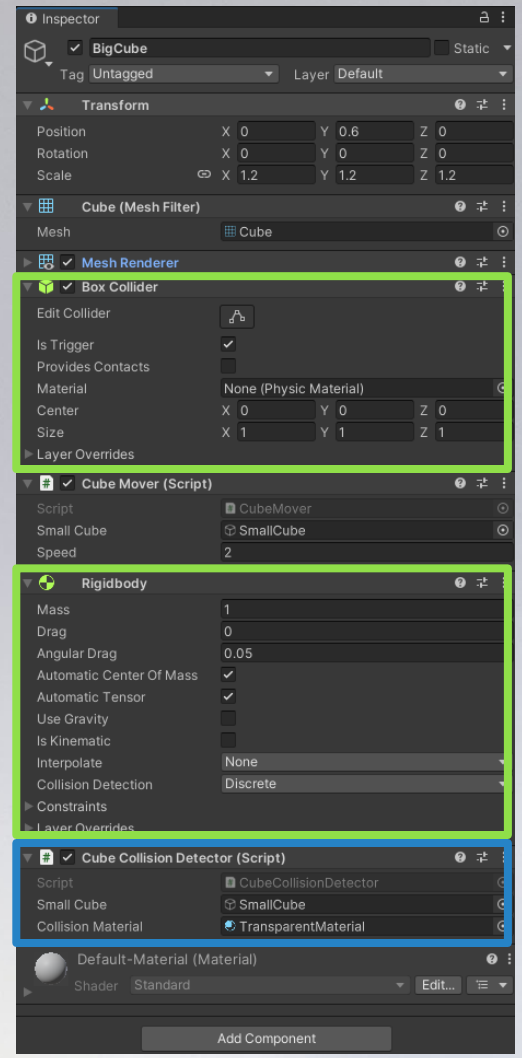
Triggers

allow you to detect and react to intersections of GameObjects

CubeCollisionDetector.cs | CubeMover.cs

Assembly-CSharp | CubeCollisionDetector | Update()

```
using System.Collections;
using System.Collections.Generic;
using System.Runtime.CompilerServices;
using UnityEngine;

Unity-Skript (1 Objektverweis) | 0 Verweise
public class CubeCollisionDetector : MonoBehaviour
{
    public GameObject SmallCube;            //reference to the small cube
    public Material CollisionMaterial;      //material to be assigned during collisions

    protected Material NoCollisionMaterial; //material to be assigned when there is no collision

    // Start is called before the first frame update
    Unity-Nachricht | 0 Verweise
    void Start()
    {
        //save the original material of the big cube
        NoCollisionMaterial = GetComponent<MeshRenderer>().material;
    }

    // Update is called once per frame
    Unity-Nachricht | 0 Verweise
    void Update()
    {
        //nothing to do here
    }

    Unity-Nachricht | 0 Verweise
    void OnTriggerEnter(Collider other)
    {
        if (other.gameObject == SmallCube)
        {
            Debug.Log("Collision with the small cube detected!");
            GetComponent<MeshRenderer>().material = CollisionMaterial;
        }
    }

    Unity-Nachricht | 0 Verweise
    void OnTriggerExit(Collider other)
    {
        if (other.gameObject == SmallCube)
        {
            Debug.Log("Collision with the small cube ended!");
            GetComponent<MeshRenderer>().material = NoCollisionMaterial;
        }
    }
}
```



Quick Live Demo

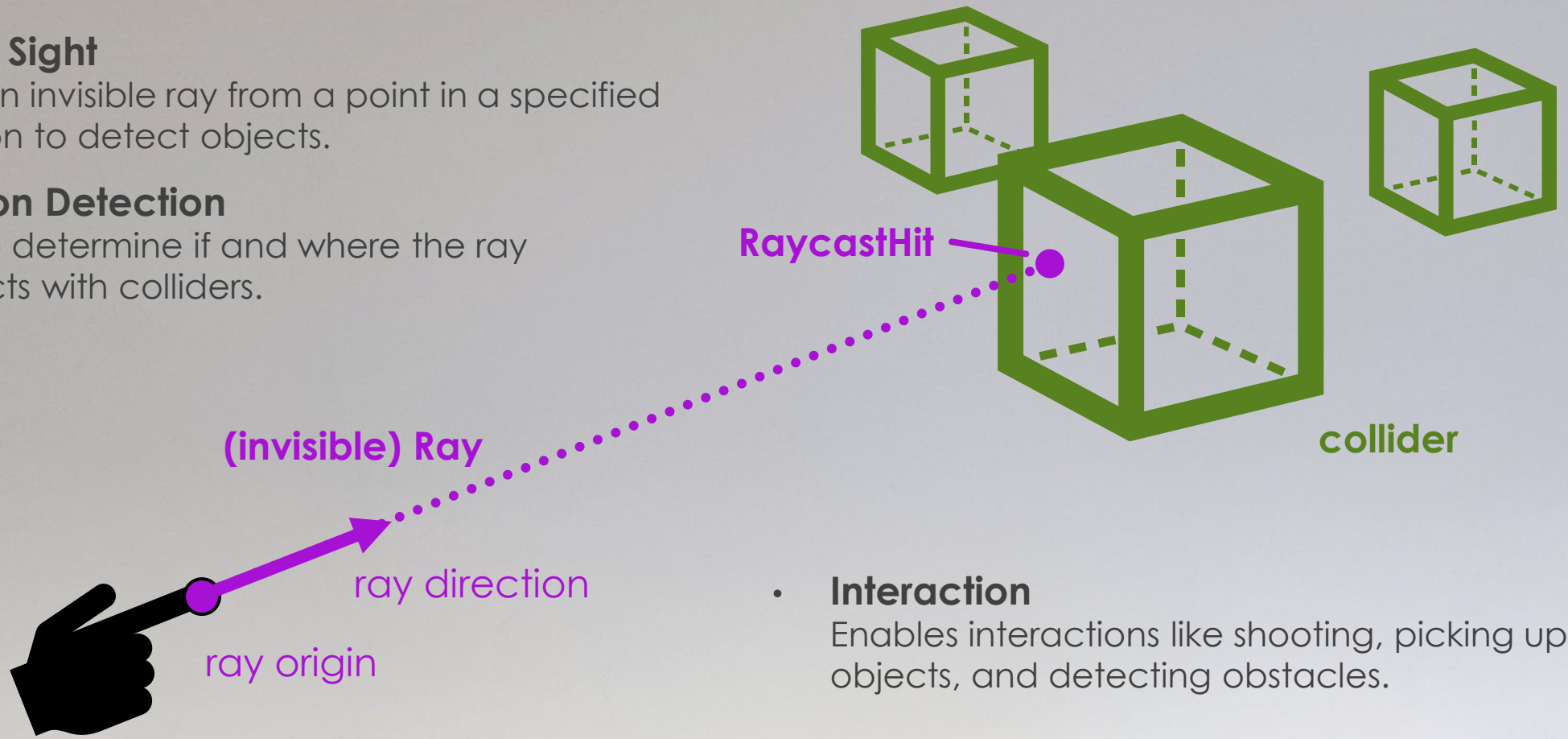
Collision Detection

(switch on BoxCollider)

Raycasting

selecting objects at a distance

- **Line of Sight**
  Casts an invisible ray from a point in a specified direction to detect objects.
- **Collision Detection**
  Used to determine if and where the ray intersects with colliders.



- **Interaction**
  Enables interactions like shooting, picking up objects, and detecting obstacles.

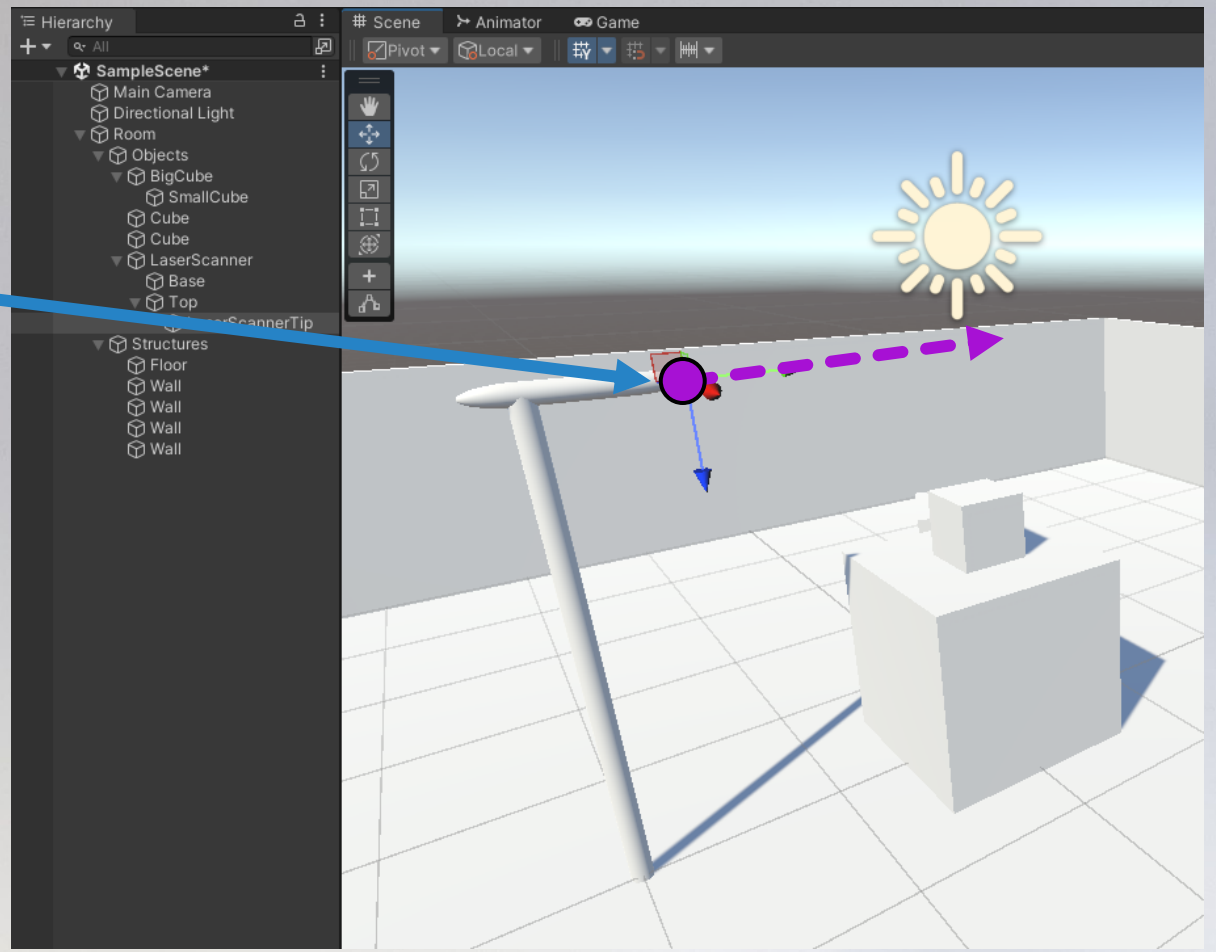
Triggers

allow you to detect and react to intersections of GameObjects

RaycastController.cs | CubeCollisionDetector.cs | CubeMover.cs

Assembly-CSharp | RaycastController | Update()

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity-Skript | 0 Verweise
public class RaycastController : MonoBehaviour
{
    public GameObject LaserTip;
    public Material NoHitMaterial;
    public Material HitMaterial;

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        // Cast a ray from the tip of the laser along its longitudinal axis
        Ray ray = new Ray(LaserTip.transform.position, LaserTip.transform.up);
        RaycastHit hit;

        // Check if the ray hit a collider in the scene
        if (Physics.Raycast(ray, out hit) && hit.collider.gameObject.name == "SmallCube")
        {
            Debug.Log($"Raycast hit {hit.collider.gameObject}!");
            ChangeLaserScannerMaterial(HitMaterial);
        }
        else
        {
            ChangeLaserScannerMaterial(NoHitMaterial);
        }
    }

    2 Verweise
    public void ChangeLaserScannerMaterial (Material material)
    {
        foreach (MeshRenderer renderer in GetComponentsInChildren<MeshRenderer>())
        {
            renderer.material = material;
        }
    }
}
```



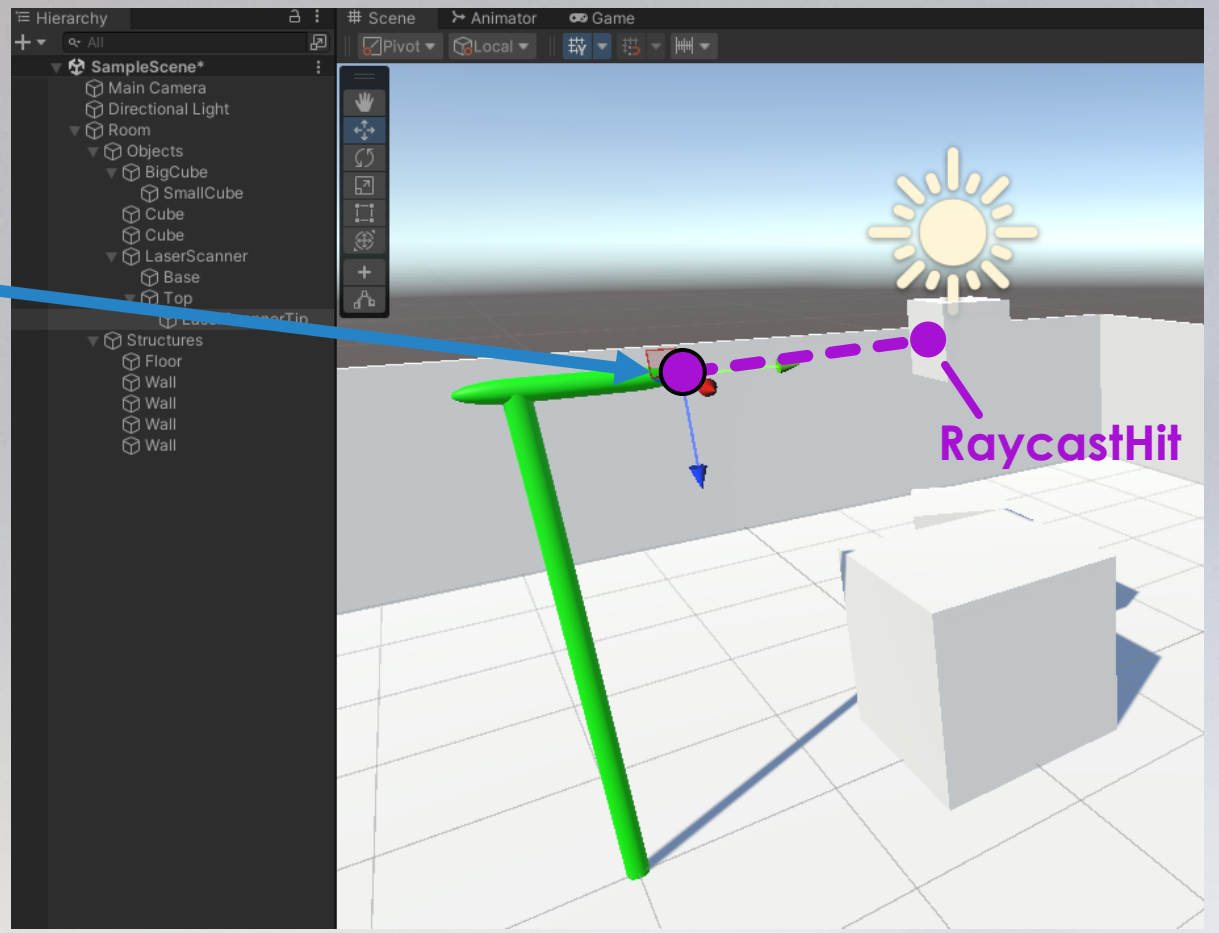
Triggers

allow you to detect and react to intersections of GameObjects

RaycastController.cs | CubeCollisionDetector.cs | CubeMover.cs

Assembly-CSharp | RaycastController | Update()

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity-Skript | 0 Verweise
public class RaycastController : MonoBehaviour
{
    public GameObject LaserTip;
    public Material NoHitMaterial;
    public Material HitMaterial;

    Unity-Nachricht | 0 Verweise
    void Update()
    {
        // Cast a ray from the tip of the laser along its longitudinal axis
        Ray ray = new Ray(LaserTip.transform.position, LaserTip.transform.up);
        RaycastHit hit;

        // Check if the ray hit a collider in the scene
        if (Physics.Raycast(ray, out hit) && hit.collider.gameObject.name == "SmallCube")
        {
            Debug.Log($"Raycast hit {hit.collider.gameObject}!");
            ChangeLaserScannerMaterial(HitMaterial);
        }
        else
        {
            ChangeLaserScannerMaterial(NoHitMaterial);
        }
    }

    2 Verweise
    public void ChangeLaserScannerMaterial (Material material)
    {
        foreach (MeshRenderer renderer in GetComponentsInChildren<MeshRenderer>())
        {
            renderer.material = material;
        }
    }
}
```



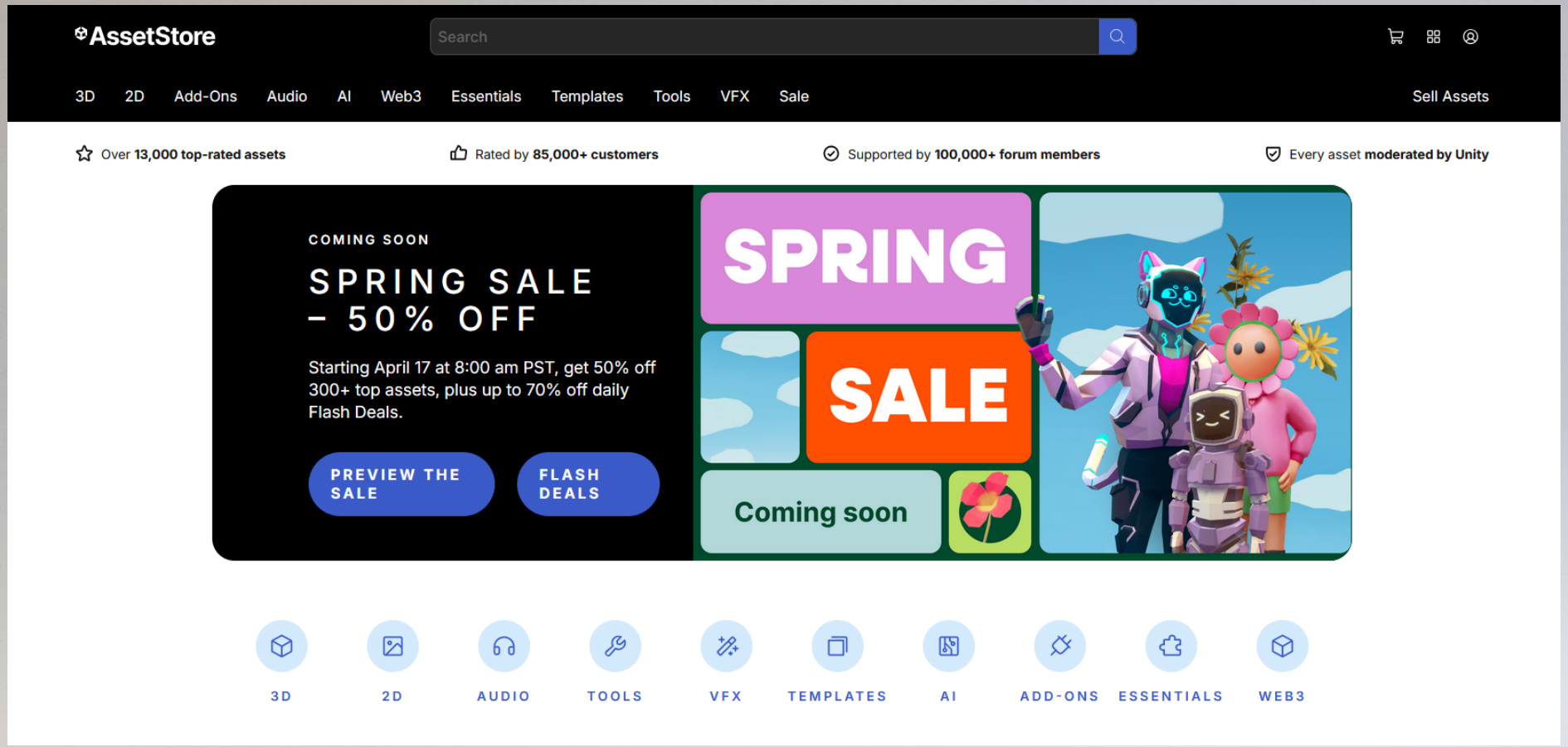
Quick Live Demo

Ray Casting

(switch on LaserScanner)

Asset Store

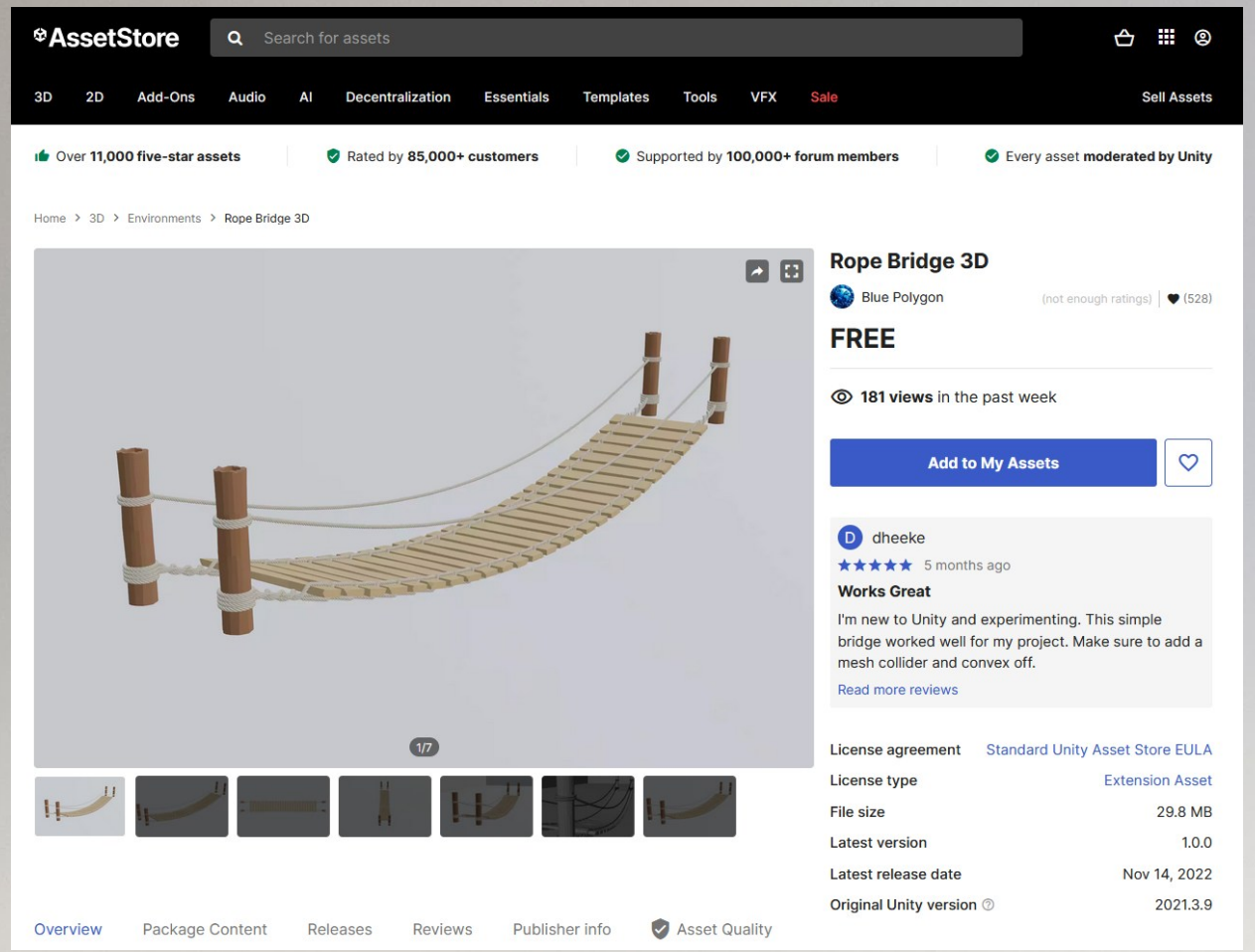
find and import free 3D models, textures, sounds, animations, etc.
to build an immersive virtual environment



https://assetstore.unity.com/

Asset Store

find and import free 3D models, textures, sounds, animations, etc.
to build an immersive virtual environment



Important!

Acknowledge all the assets and materials that you use in your project!

https://assetstore.unity.com/packages/3d/environments/rope-bridge-3d-222563

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices
- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

VR Hardware

used in this project

4x
Base Stations

9x
Vive Tracker

...

HTC Vive Pro
(with wireless Add-On)

2x
HTC Vive Pro Controller

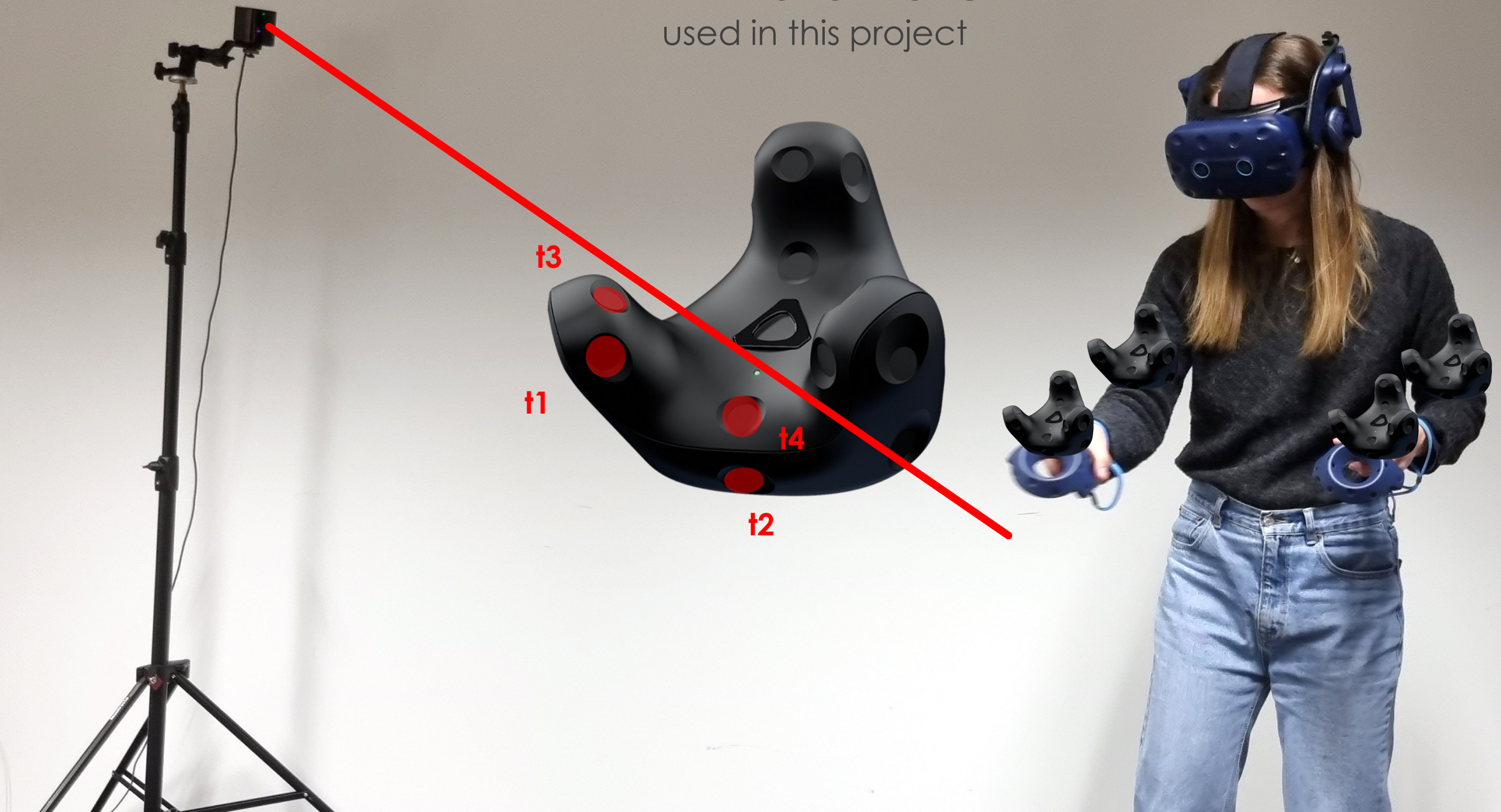
VR Hardware

tracks the position and orientation
of the VR headset and the trackers

infrared light

VR Hardware

used in this project



VR Hardware

used in this project

infrared light

computed from timing differences:
3D position & **3D orientation**

(SteamVR does this for you! ☺)

VR Integration

making your scene VR-compatible

There are multiple ways to do this.

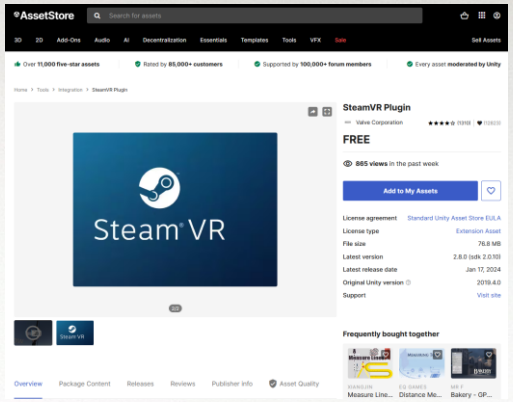
For this project, we recommend using:

SteamVR

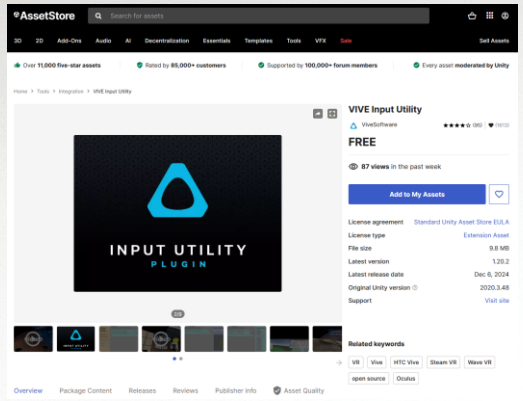
+

VIVE Input Utility

both are in the Asset Store
both are already in the project template



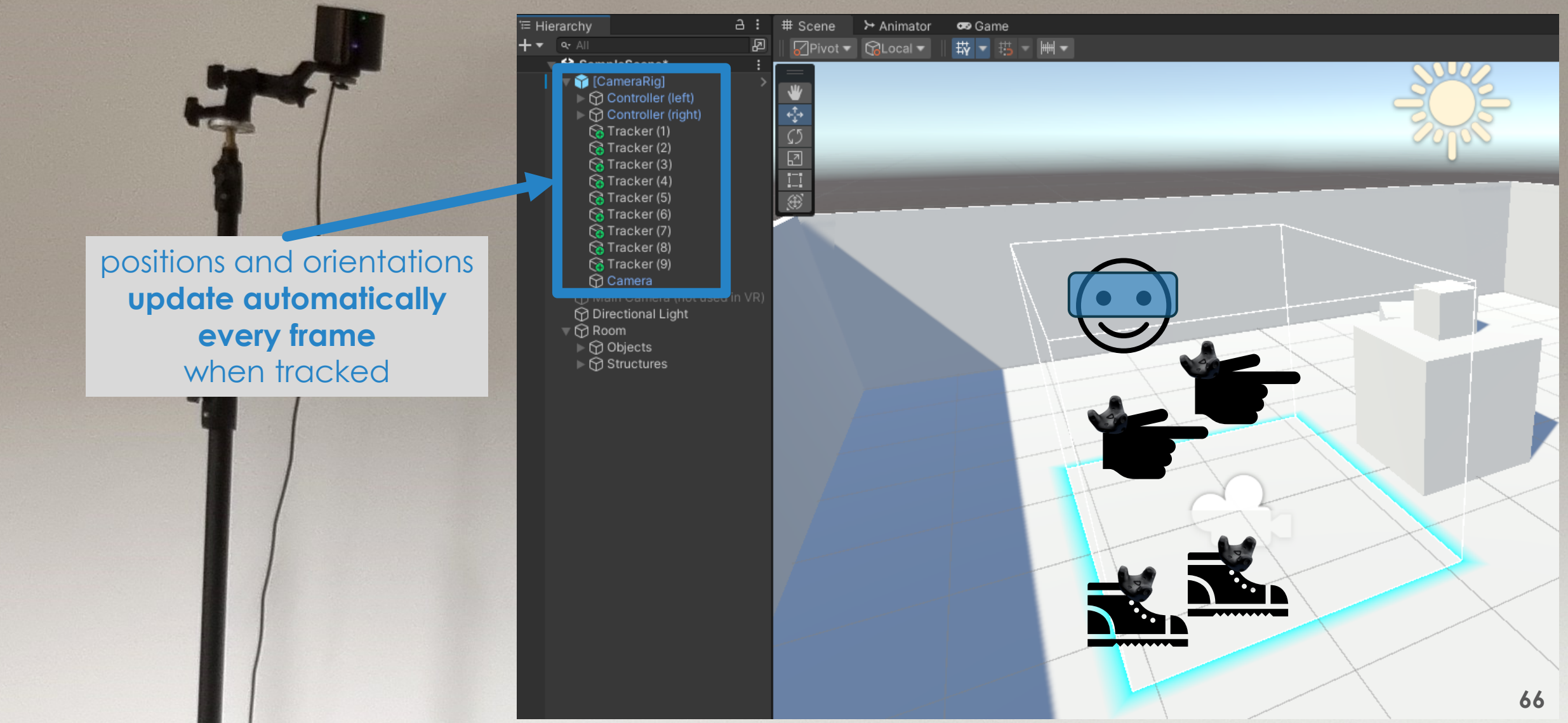
https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647



https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219

VR Integration

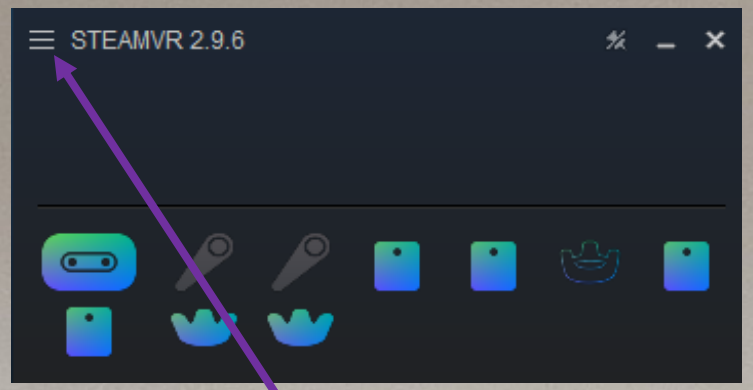
place the [CameraRig] prefab in your scene



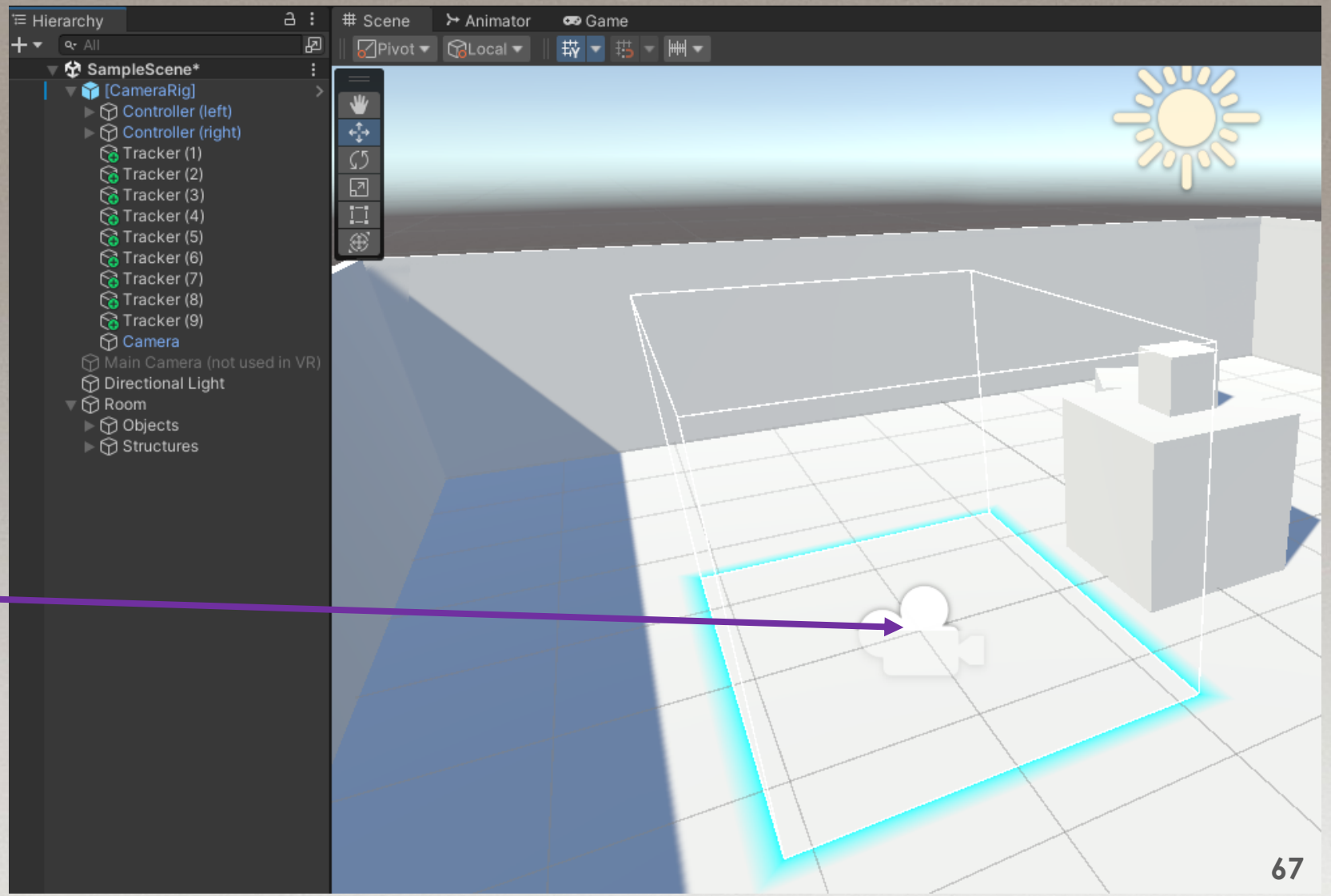
positions and orientations **update automatically every frame** when tracked

VR Setup

before working with VR, complete the SteamVR room setup

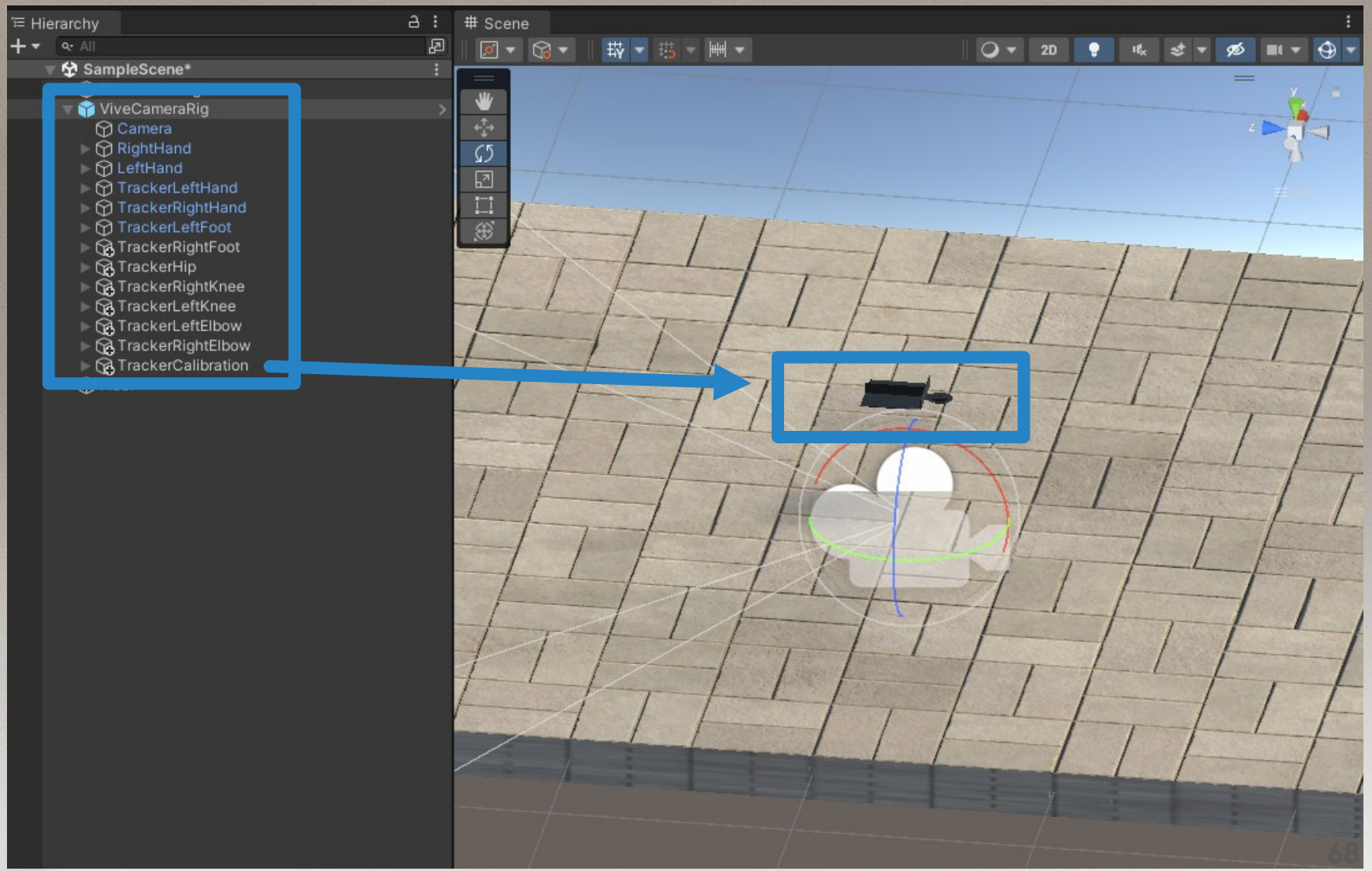


Perform "Room Setup"
to define the
tracking space origin
("Standing Setup" is usually sufficient)



VR Integration

you can use the prepared ViveCameraRig in our example scene



Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices
- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Tracking a Body Part

a simple calibration procedure

To track a body part in VR:

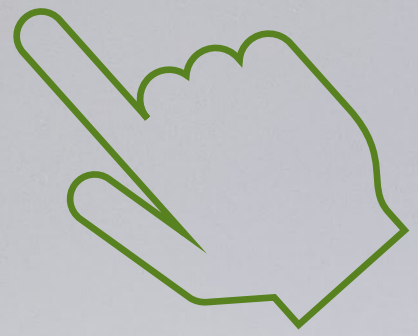
1. Find or create a **3D model of the body part**
   (Asset Store or self-made; can be abstract)

Tracking a Body Part

a simple calibration procedure

To track a body part in VR:

1. Find or create a **3D model of the body part**
   (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly



Hierarchy:

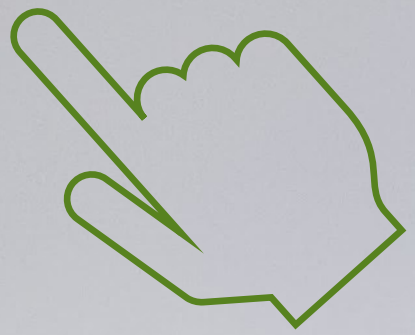
- **Virtual Hand**

Tracking a Body Part

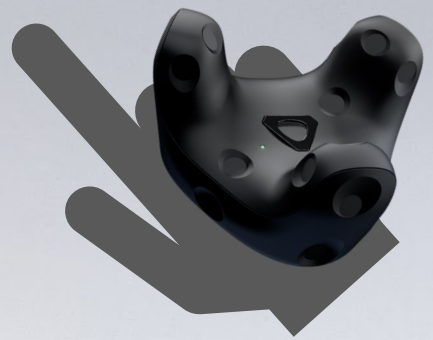
a simple calibration procedure

To track a body part in VR:

1. Find or create a **3D model of the body part** (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly
3. Attach a **Vive tracker to your real body part**



Hierarchy:
- **Virtual Hand**
- **Tracker**

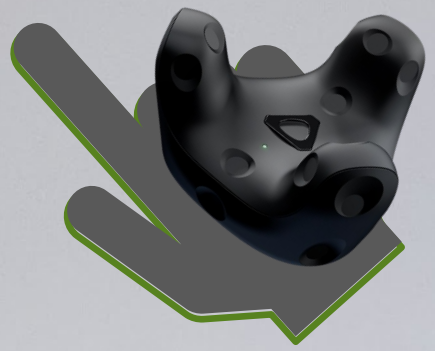


Tracking a Body Part

a simple calibration procedure

To track a body part in VR:

1. Find or create a **3D model of the body part** (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly
3. Attach a **Vive tracker to your real body part**
4. After starting the scene, move your **real body part** to align it with the **virtual body part** in the scene



Hierarchy:

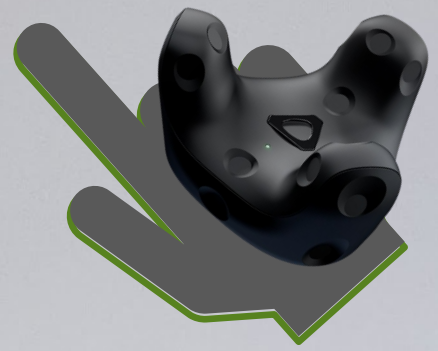
- **Virtual Hand**
- **Tracker**

Tracking a Body Part

a simple calibration procedure

To track a body part in VR:

1. Find or create a **3D model of the body part** (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly
3. Attach a **Vive tracker to your real body part**
4. After starting the scene, move your **real body part** to align it with the **virtual body part** in the scene
5. Make the **virtual body part** a child of the **tracker**
6. The virtual body part now follows the tracker ☺



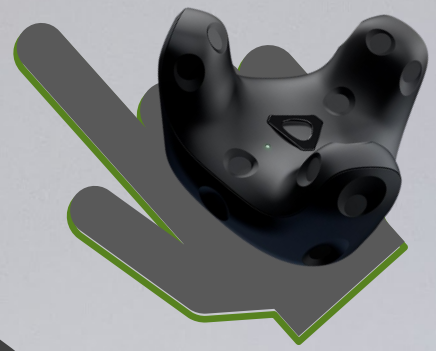
Hierarchy:
- **Tracker**
  - **Virtual Hand**

Tracking a Body Part

a simple calibration procedure

To track a body part in VR:

1. Find or create a **3D model of the body part** (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly
3. Attach a **Vive tracker to your real body part**
4. After starting the scene, move your **real body part** to align it with the **virtual body part** in the scene
5. Make the **virtual body part** a child of the **tracker**
6. The virtual body part now follows the tracker ☺



Hierarchy:
- **Tracker**
  - **Virtual Hand**

How to Align?

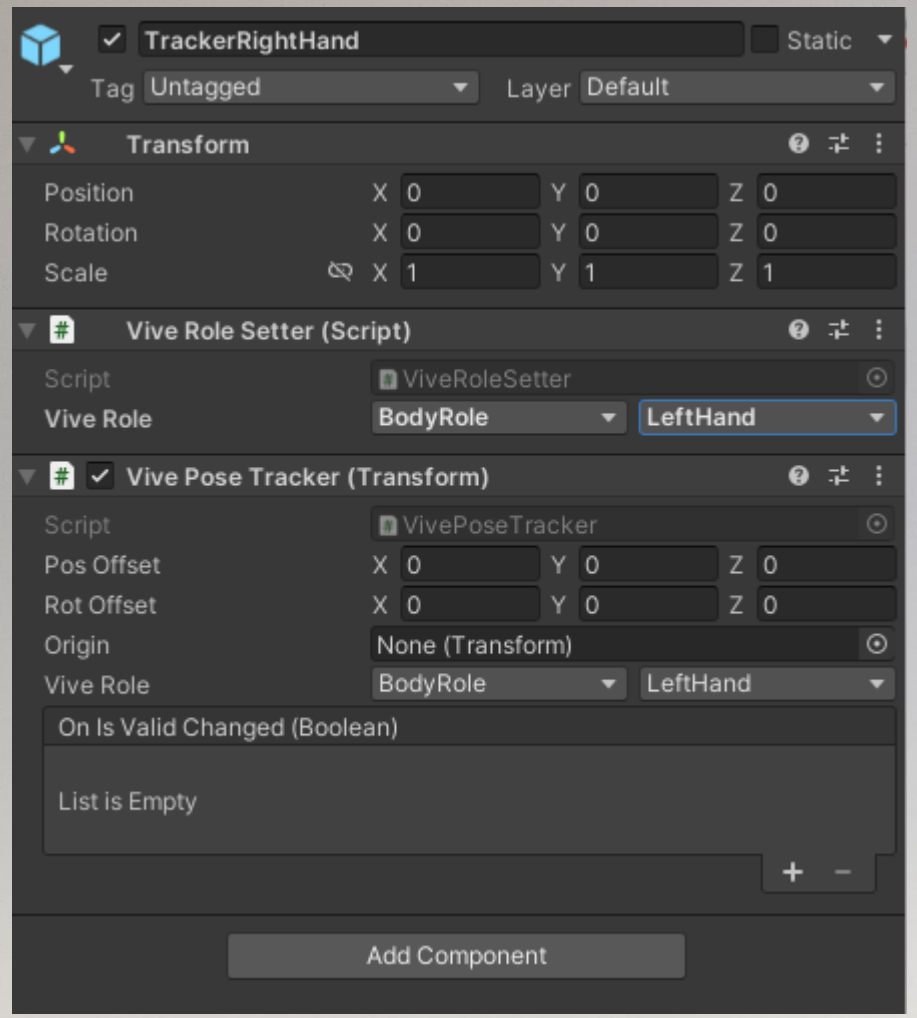
It is good practice to place both (real & virtual) body parts, e.g.:
- inside a tracked calibration template
- in a strategic location aligned with some physical feature (e.g., corner of a table)

Use the tracked VR controllers or unused trackers to determine the virtual position of the real feature/template.


climbtrack

Using Multiple Vive Trackers

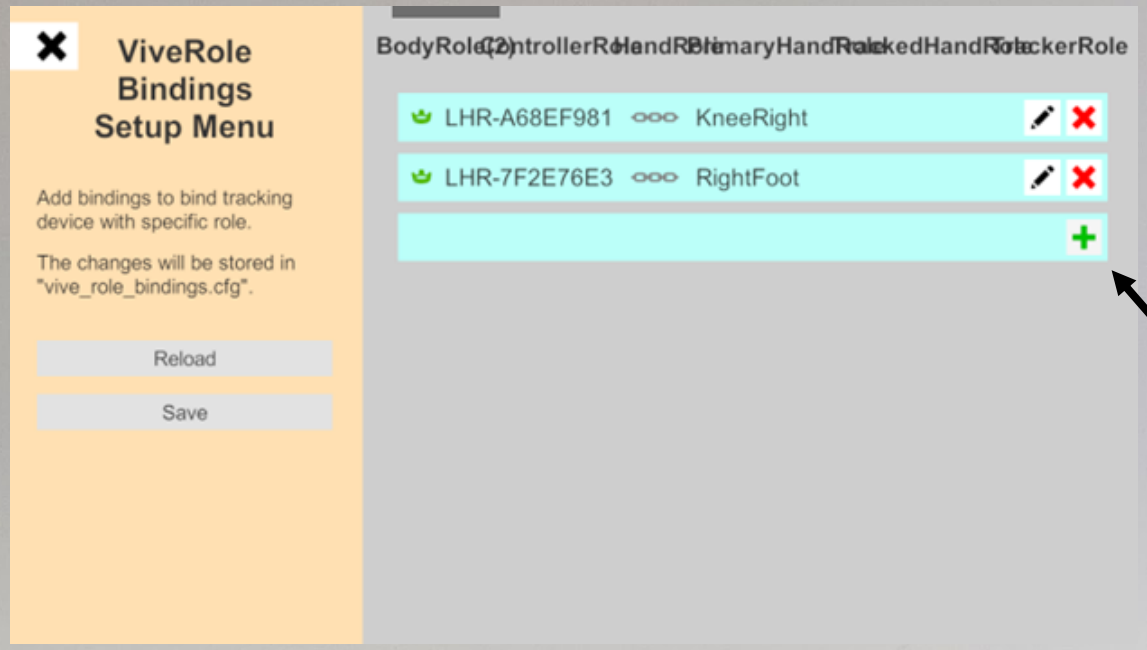
is possible by assigning each tracker a role



- **Problem:**
  Organizing multiple trackers in one scene.
- **Solution:**
  1. for each tracker object in the scene:
     assign it a role on the attached `ViveRoleSetter` script
     (e.g., "Left Hand", "Right Foot", etc.)

Using Multiple Vive Trackers

is possible by assigning each tracker a role

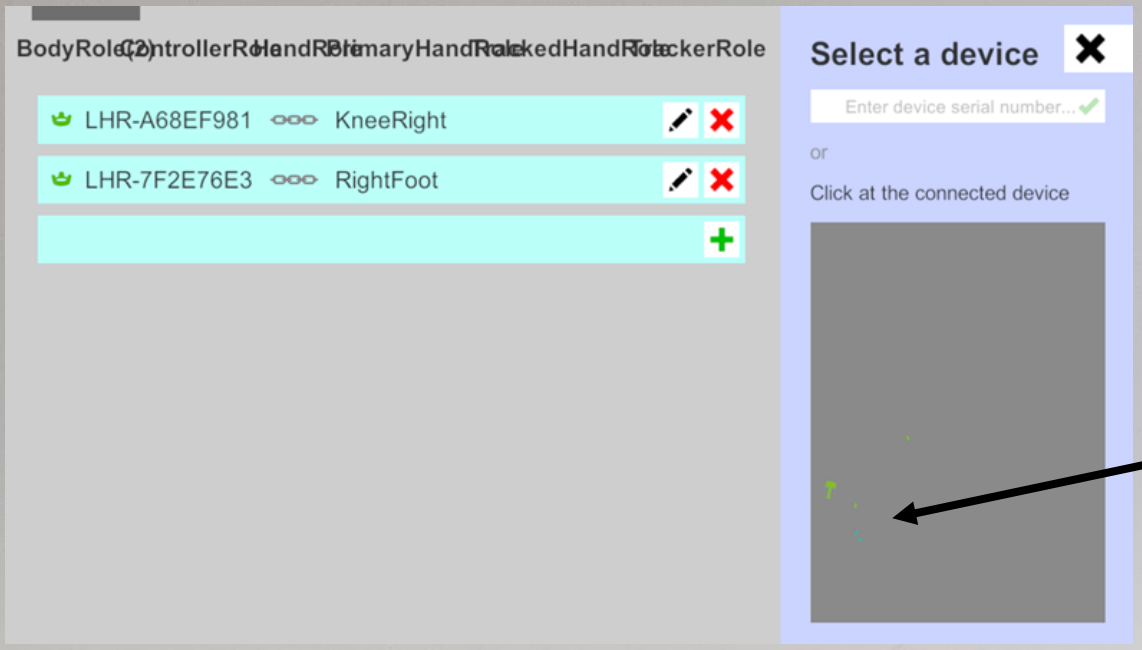


create a new binding

- **Problem:**
  Organizing multiple trackers in one scene.
- **Solution:**
  1. for each tracker object in the scene:
     assign it a role on the attached `ViveRoleSetter` script
     (e.g., "Left Hand", "Right Foot", etc.)
  2. map the physical trackers to the roles using the Vive Input Utility
     (hit `Ctrl+B` in the GameView while the scene is running to open the "Binding UI")

Using Multiple Vive Trackers

is possible by assigning each tracker a role

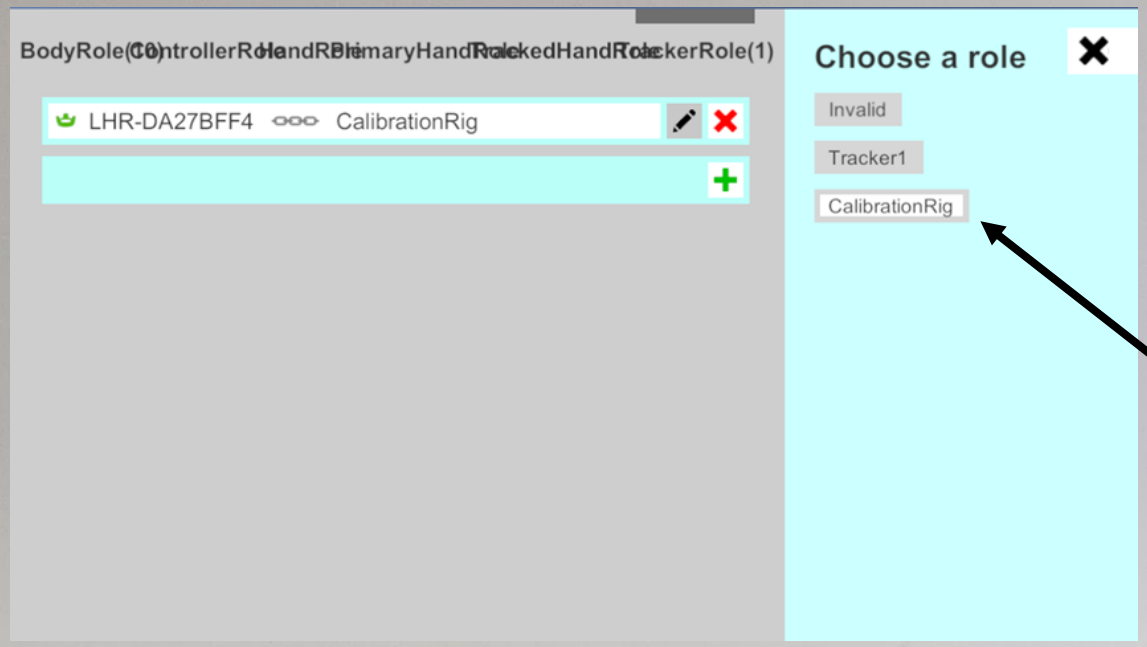


select a tracker

- **Problem:**
  Organizing multiple trackers in one scene.
- **Solution:**
  1. for each tracker object in the scene:
     assign it a role on the attached `ViveRoleSetter` script
     (e.g., "Left Hand", "Right Foot", etc.)
  2. map the physical trackers to the roles using the Vive Input Utility
     (hit `Ctrl+B` in the GameView while the scene is running to open the "Binding UI")

Using Multiple Vive Trackers

is possible by assigning each tracker a role

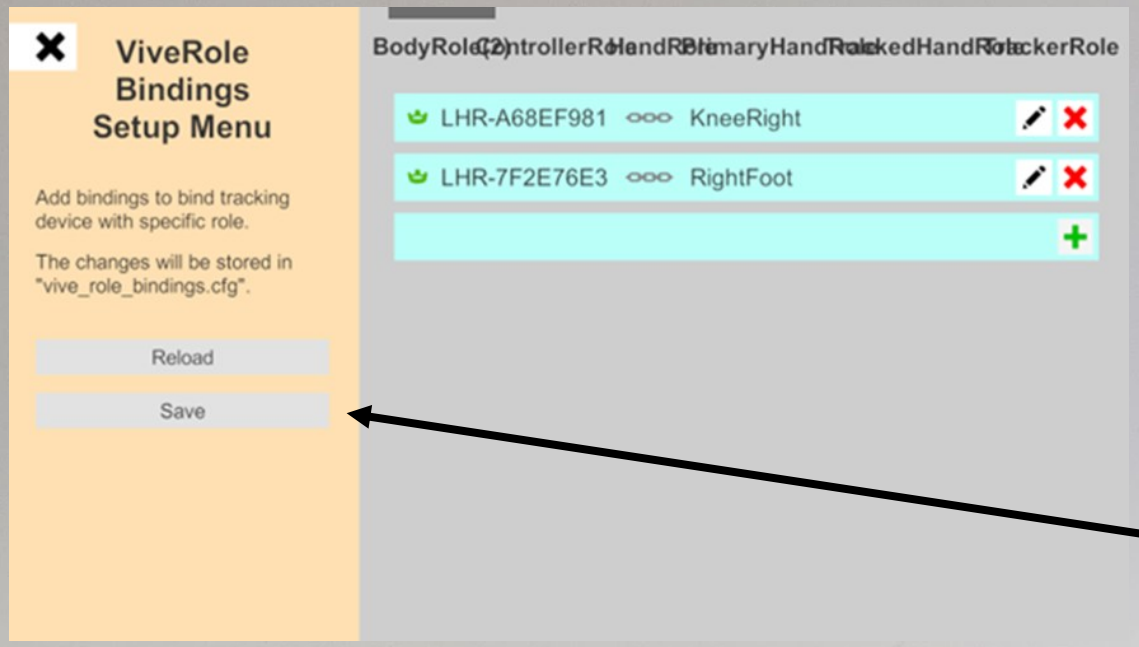


choose a role

- **Problem:**
  Organizing multiple trackers in one scene.
- **Solution:**
  1. for each tracker object in the scene:
     assign it a role on the attached `ViveRoleSetter` script
     (e.g., "Left Hand", "Right Foot", etc.)
  2. map the physical trackers to the roles using the Vive Input Utility
     (hit `Ctrl+B` in the GameView while the scene is running to open the "Binding UI")

Using Multiple Vive Trackers

is possible by assigning each tracker a role



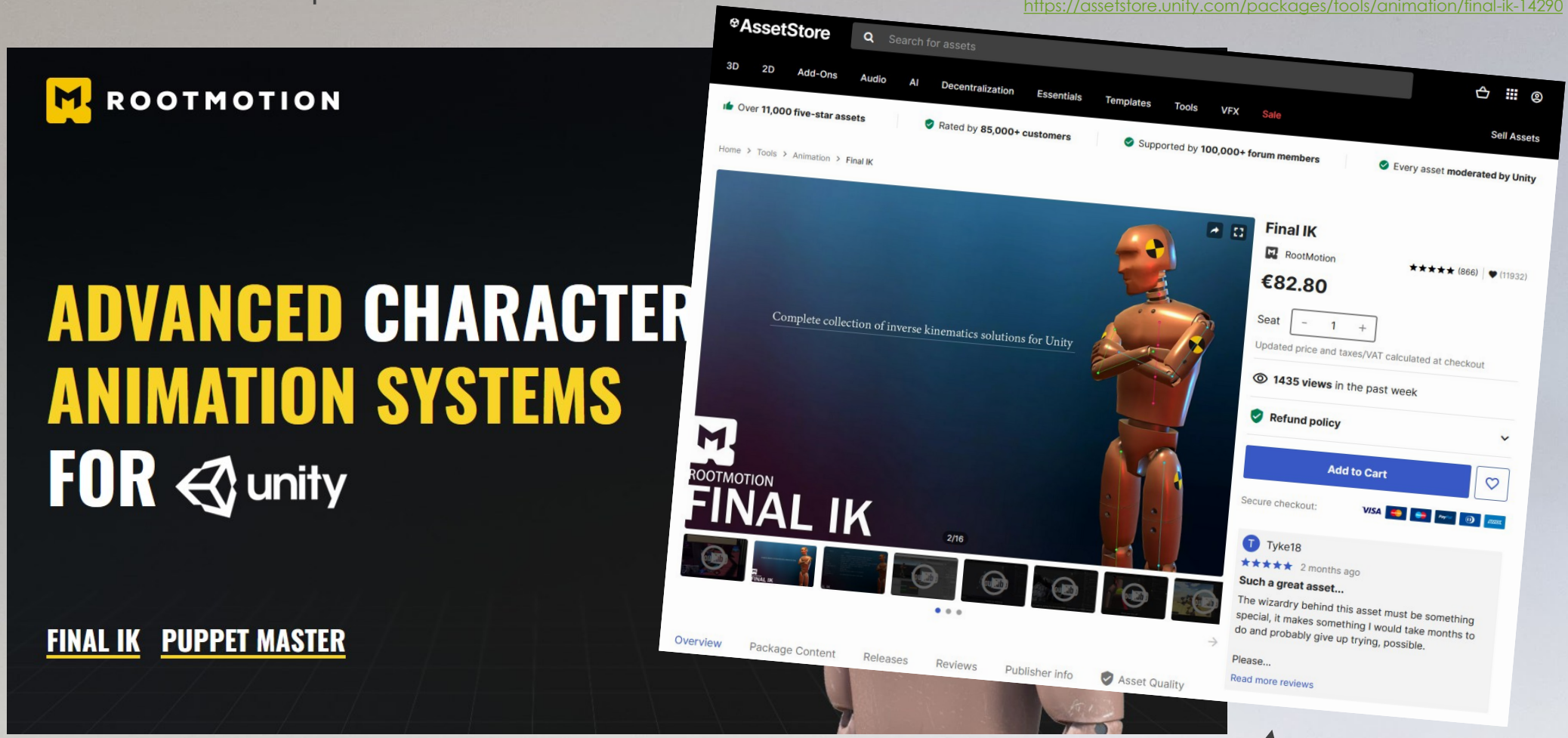
save the binding (and reuse it next time)

- **Problem:**
  Organizing multiple trackers in one scene.
- **Solution:**
  1. for each tracker object in the scene:
     assign it a role on the attached `ViveRoleSetter` script
     (e.g., "Left Hand", "Right Foot", etc.)
  2. map the physical trackers to the roles using the Vive Input Utility
     (hit `Ctrl+B` in the GameView while the scene is running to open the "Binding UI")
  3. save the mapping to reuse it

Tracking a Full Body

possible with inverse kinematics

https://assetstore.unity.com/packages/tools/animation/final-ik-14290

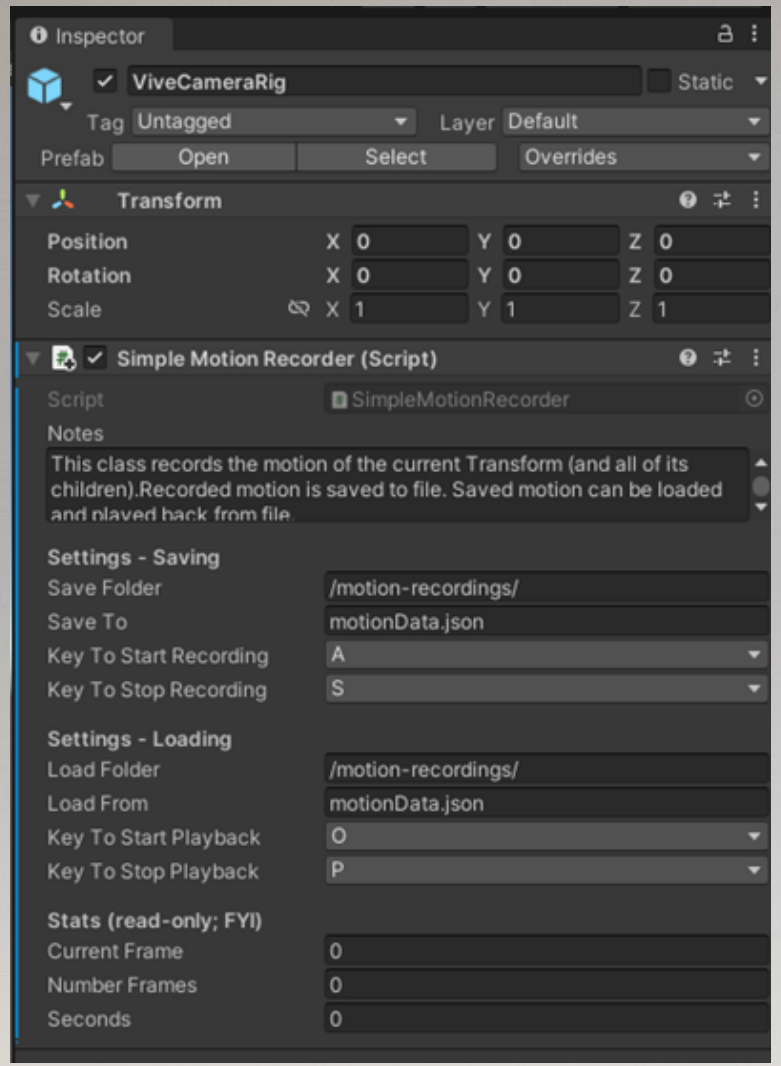


http://root-motion.com/

The **FinalIK** inverse kinematics package is included in the template project.
You can use it if you like – but you don't have to.

Motion Recording & Playback

is useful when developing and testing from home



- **Problem:**
  At home, you might not have access to a VR system with 9 trackers.

- **Solution:**
  - Record example motions in the lab using our `SimpleMotionRecorder` script.
  - Recorded motions are saved to file
  - Play back recorded motions at a later point (e.g., at home)

- **Example:**
  "MotionRecorderExample" scene in the template project (in the SimpleMotionRecorder → Scenes folder)

- **Important:**
  The name of each child needs to be unique!

Quick Live Demo

Motion Recording & Playback

(record and play back movement of Camera in [CameraRig])

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

- Questions & Answers

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

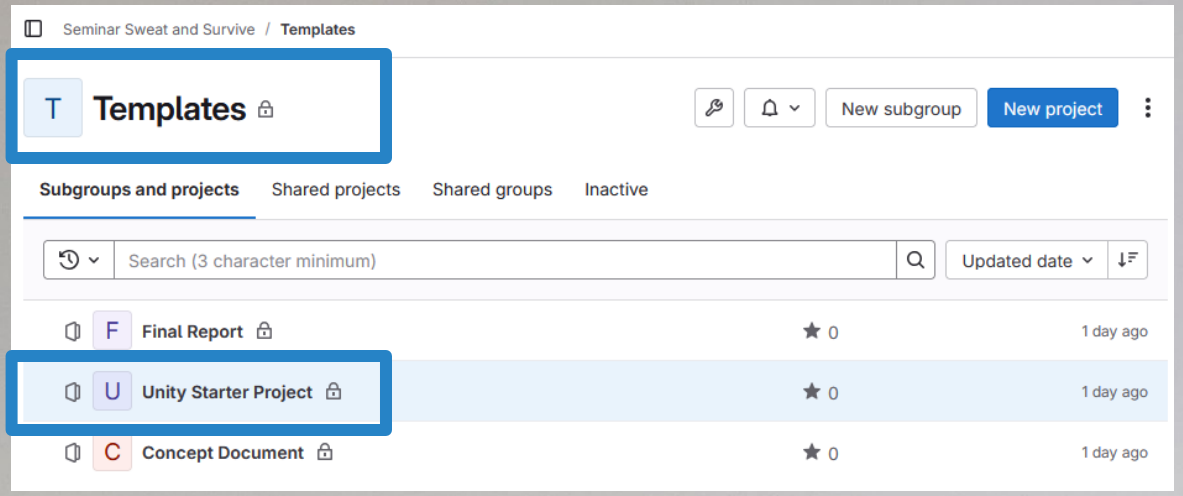
Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- **Project Template & Best Practices**

Questions & Answers

Project Template

available for forking on GitLab



https://umtl-git.dfki.de/seminar-sweat-and-survive/templates/unity-starter-project

The template project:

- is made for Unity version 2021.3.38f1
- features a suitable .gitignore for Unity projects (https://github.com/github/gitignore/blob/main/Unity.gitignore)

and already includes:

- SteamVR (https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647)
- Vive Input Utility (https://assetstore.unity.com/packages/tools/integration/vive-input-utility-64219)
- FinalIK (https://assetstore.unity.com/packages/tools/animation/final-ik-14290)
- the simple Motion Recorder & Player (ask us if you have questions)

Build upon the included template scene ☺

Best Practices

can be found online

General Best Practices:

- write easy-to-understand code
- avoid almighty "God-classes"
- use intuitive function and variable names
- if you use code from somewhere else (e.g., ChatGPT, tutorials, …): acknowledge the original source in the comments (e.g., with a link)

Unity is well documented in the official Unity Documentation
(https://docs.unity.com/)

Learning:

- You can find good tutorials online (e.g., on YouTube)
- LLMs can help you get started too! (e.g., ChatGPT)

Best Practices

can be found online

ChatGPT

How can I make an object in my Unity scene blink when the user looks at it?

Learning:

- You can find good tutorials online (e.g., on YouTube)
- **LLMs can help you get started too! (e.g., ChatGPT)**

Best Practices

can be found online

ChatGPT

How can I make an object in my Unity scene blink when the user looks at it?

a good starting point!

Learning:

- You can find good tutorials online (e.g., on YouTube)
- **LLMs can help you get started too! (e.g., ChatGPT)**

```
public class LookAtBlink : MonoBehaviour
{
    public Camera playerCamera; // assign your main camera in the Inspector
    public float maxDistance = 10f;
    public float blinkInterval = 0.5f;
    private Renderer objRenderer;
    private bool isLooking = false;
    private float blinkTimer = 0f;

    void Start()
    {
        objRenderer = GetComponent<Renderer>();
    }

    void Update()
    {
        Ray ray = new Ray(playerCamera.transform.position, playerCamera.transform.forward);
        RaycastHit hit;

        // Check if the object is being looked at
        if (Physics.Raycast(ray, out hit, maxDistance))
        {
            if (hit.transform == transform)
            {
                isLooking = true;
            }
            else
            {
                isLooking = false;
                objRenderer.enabled = true; // make sure it's visible again
            }
        }
        else
        {
            isLooking = false;
            objRenderer.enabled = true;
        }

        // Handle blinking
        if (isLooking)
        {
            blinkTimer += Time.deltaTime;
            if (blinkTimer >= blinkInterval)
            {
                objRenderer.enabled = !objRenderer.enabled;
                blinkTimer = 0f;
            }
        }
    }
}
```

Best Practices
can be found online

ChatGPT

BUT!

You need to understand all your code!

This means:
You need to be able to **answer questions about it** and **explain it**,
and **comment which parts were generated** by ChatGPT etc.

☺

Learning:

- You can find good tutorials online (e.g., on YouTube)
- **LLMs can help you get started too! (e.g., ChatGPT)**

```
public class LookAtBlink : MonoBehaviour
{
    public Camera playerCamera; // assign your main camera in the Inspector
    public float maxDistance = 10f;
    public float blinkInterval = 0.5f;
    private Renderer objRenderer;
    private bool isLooking = false;
    private float blinkTimer = 0f;

    void Start()
    {
        if (isLooking)
        {
            blinkTimer += Time.deltaTime;
            if (blinkTimer >= blinkInterval)
            {
                objRenderer.enabled = !objRenderer.enabled;
                blinkTimer = 0f;
            }
        }
    }
}
```

Agenda:

General Basics

- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

Questions & Answers

Agenda:

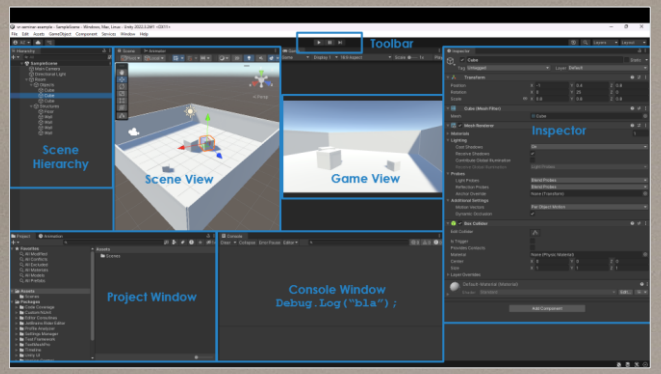
General Basics

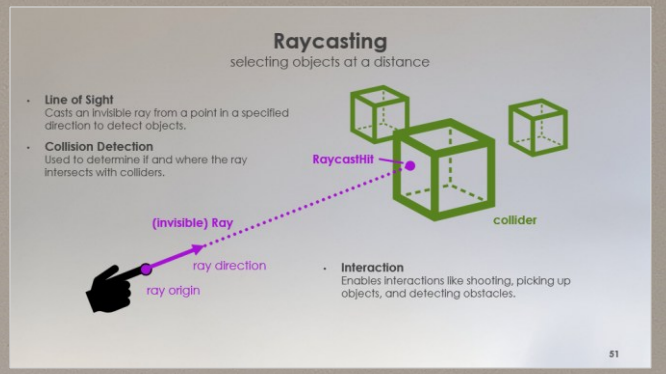
- Introduction
- Unity Setup
- Unity User Interface & Scene Graph
- Unity Programming

Project-Related Basics

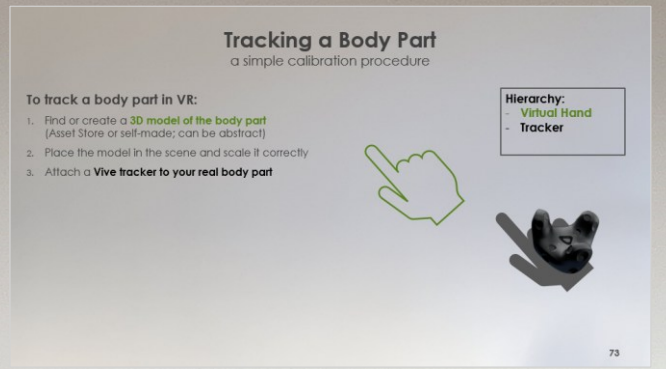
- Making Scenes Interactive
- Virtual Reality Integration
- Body Tracking
- Project Template & Best Practices

Questions & Answers


Toolbar
Scene Hierarchy
Scene View
Game View
Inspector
Project Window
Console Window
Debug.Log("bla");


Raycasting
selecting objects at a distance
Line of Sight
Casts an invisible ray from a point in a specified direction to detect objects.
Collision Detection
Used to determine if and where the ray intersects with colliders.
RaycastHit
collider
(invisible) Ray
ray direction
ray origin
Interaction
Enables interactions like shooting, picking up objects, and detecting obstacles.
51


VR Hardware
used in this project
4x
Base Stations
HTC Vive Pro
(with wireless Add-On)
9x
Vive Tracker
2x
HTC Vive Pro
Controller
61


Tracking a Body Part
a simple calibration procedure
To track a body part in VR:
1. Find or create a 3D model of the body part (Asset Store or self-made; can be abstract)
2. Place the model in the scene and scale it correctly
3. Attach a Vive tracker to your real body part
Hierarchy:
- Virtual Hand
- Tracker
73

Questions?