



# UNITY FOR ROBOTICS

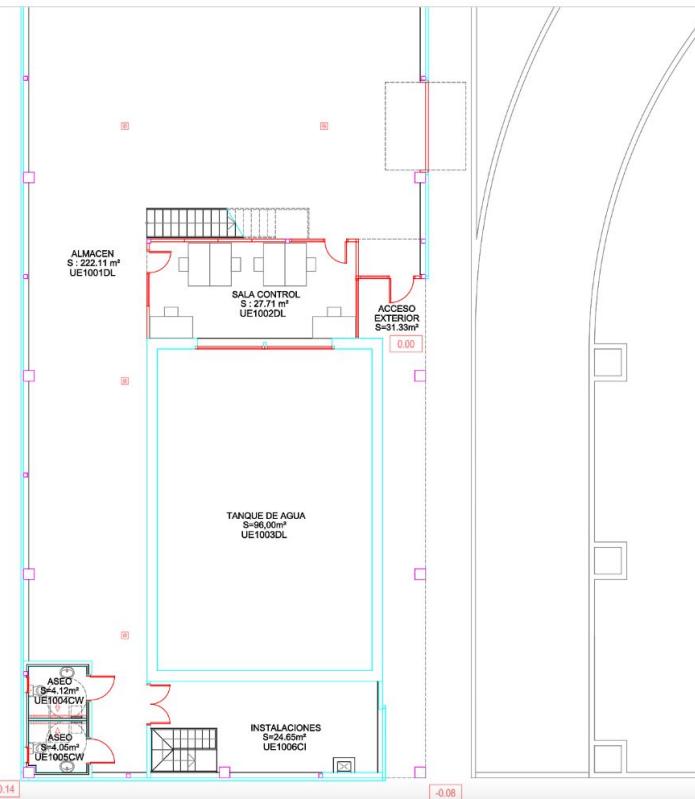
CHAPTER 6: Exercise Scenario CIRTESU

Raúl Marín Prades

Open CIRTESU building plan  
(e.g. Autodesk Viewer)

- 2D View
- ▼ Sheets
  - ALZADOS
  - P.BAJA
  - P.CUBIERTA
  - P.PRIMERA
  - P.SÓTANO
  - SECCIONES

Edades Configuración



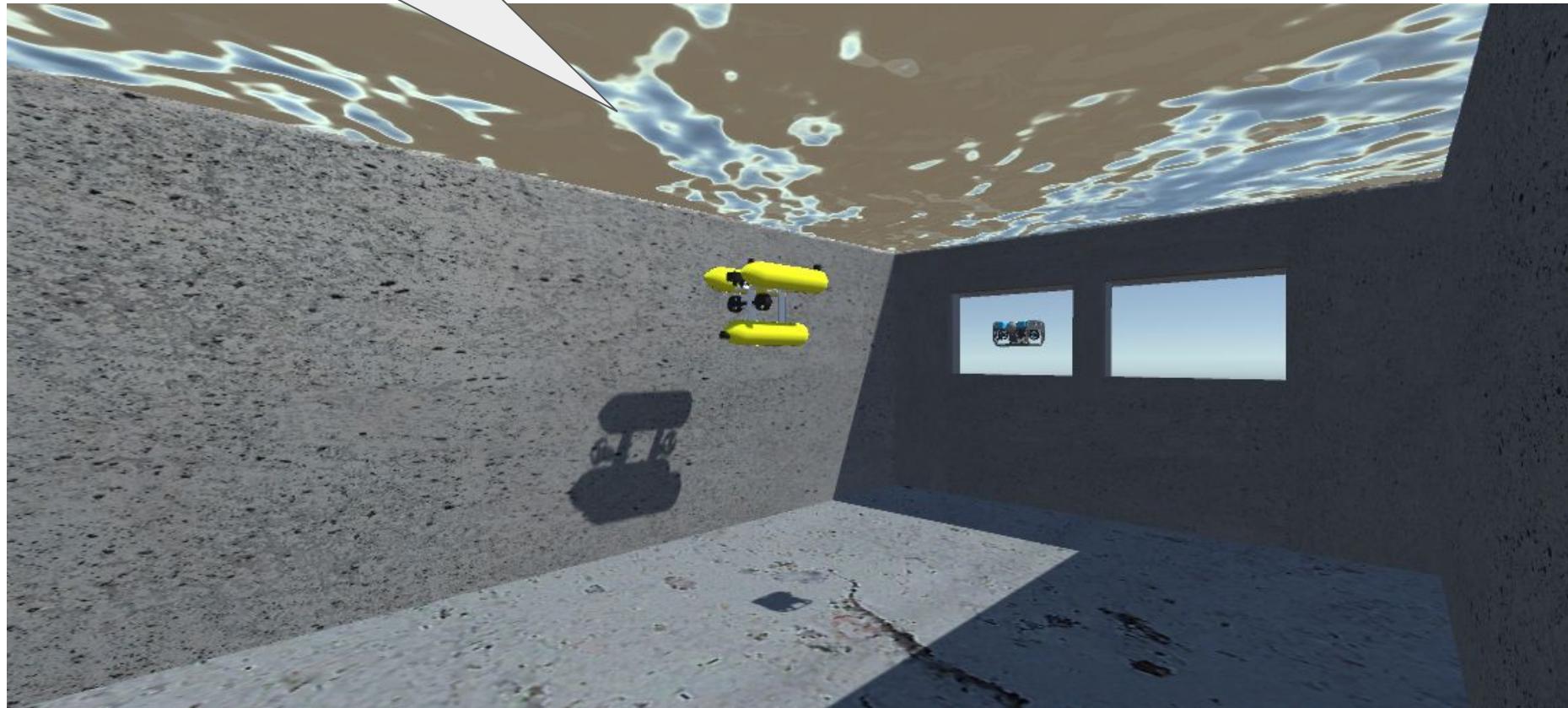
PROYECTO BÁSICO Y DE EJECUCIÓN  
"CENTRO INVESTIGACIÓN EN ROBÓTICA Y TECNOLOGÍAS SUBACUÁTICAS EN PARCELA -E- DEL PARQUE CIENTÍFICO Y TECNOLÓGICO DE LA UNIVERSITAT JAUME I DE CASTELLÓ"

**JU** UNIVERSITAT JAUME I CASTELLÓ  
INGENIEROS TÉCNICOS  
OFICINA TÉCNICA D'OBRES I PROJECTES  
JOSE ROGER DOLS  
LUIS VELLÓN BELLIDO

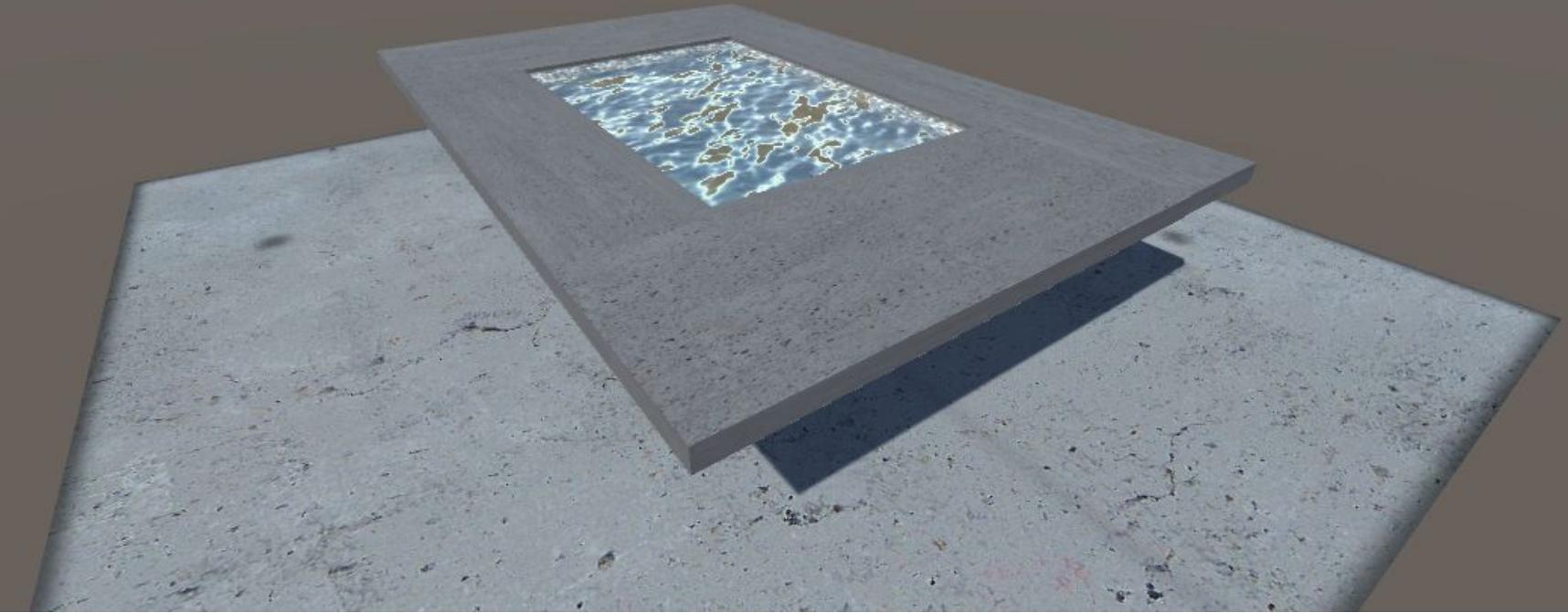
NIVEL 0 : DISTRIBUCIÓN,  
MOBILIARIO Y SUPERFICIES

03

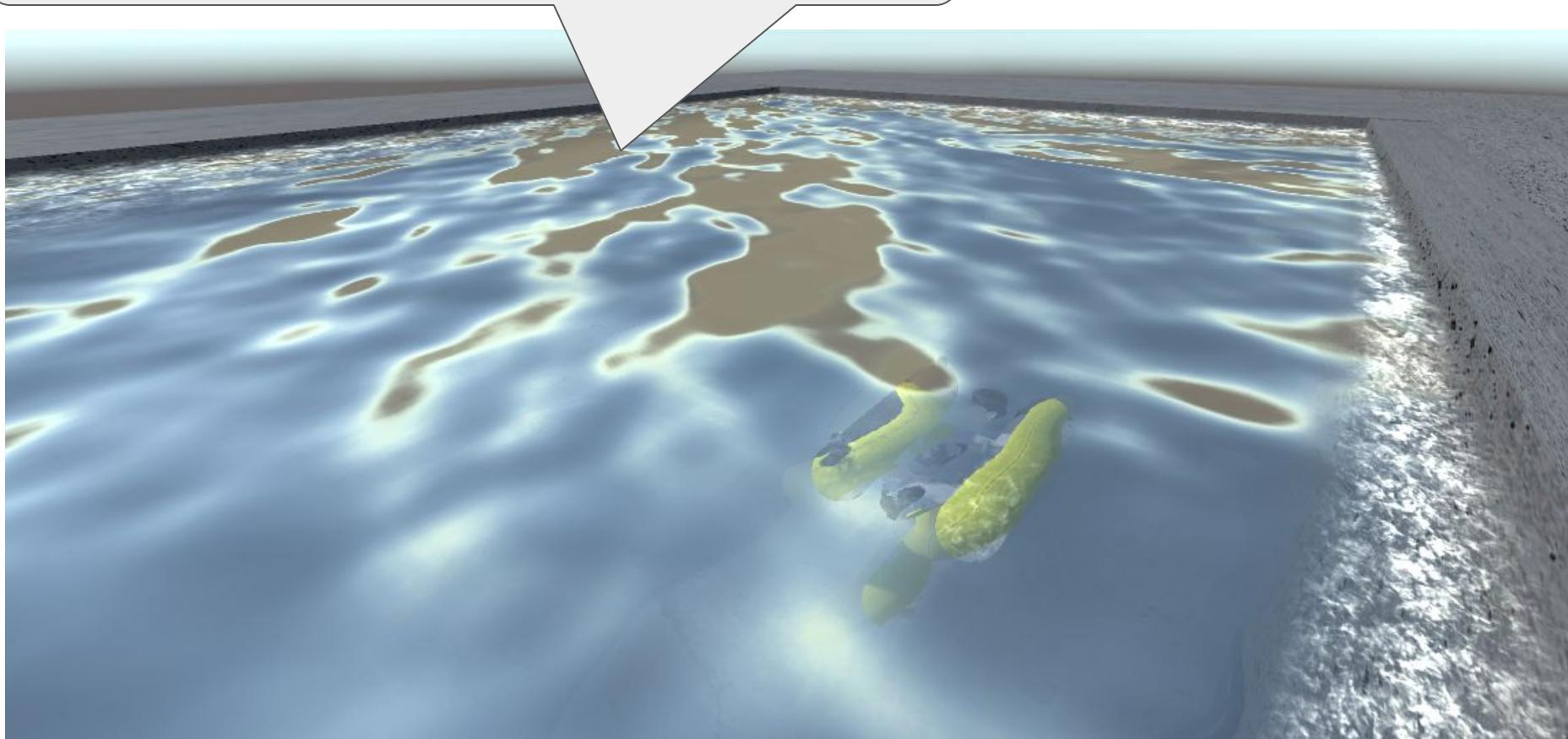
Create Unity Scene using Unity  
Blocks, OnShape, etc. Use Asset  
Store for concrete materials





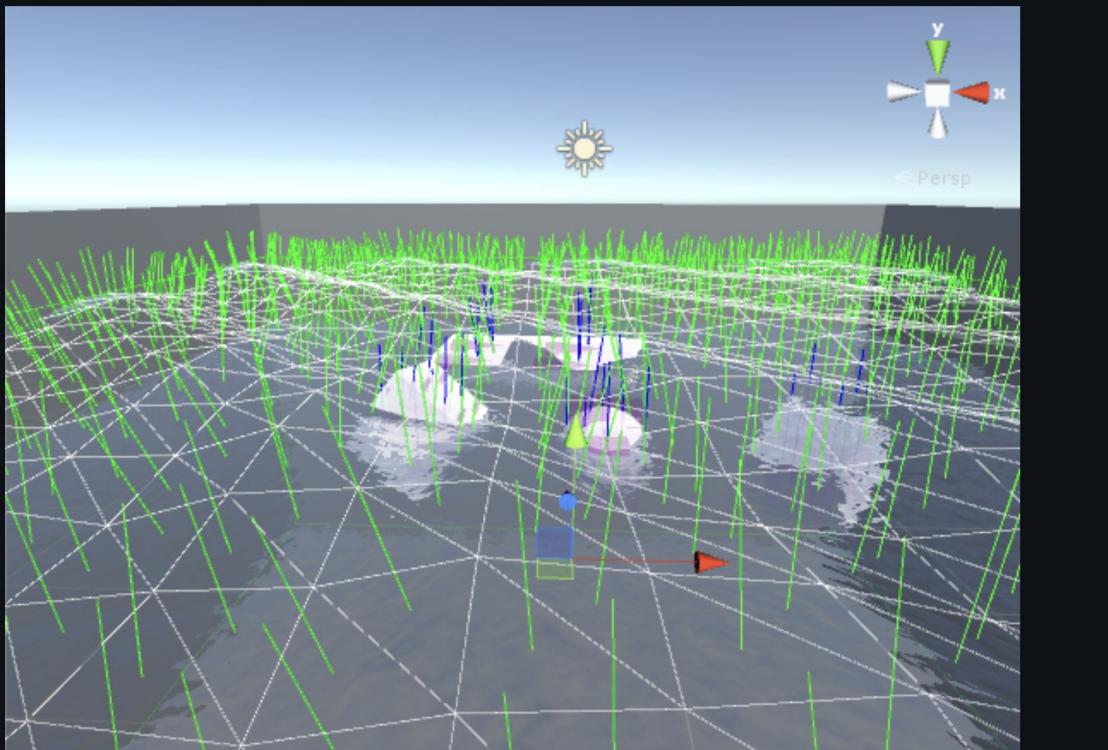
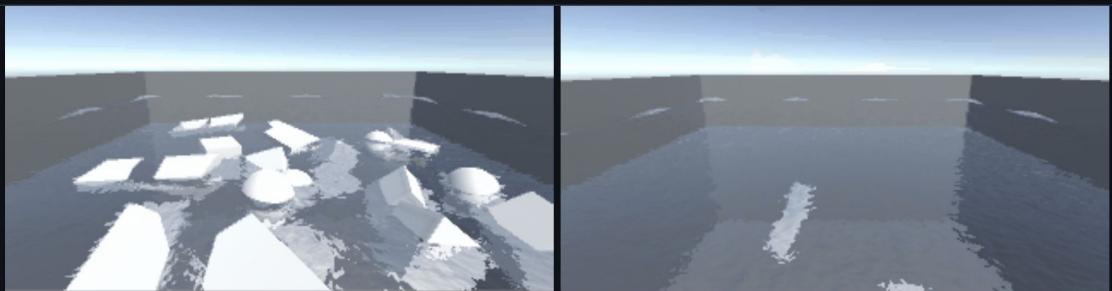


Study Water Plane Options (e.g. Aquas, Gaia, HDRP 2022)



<https://assetstore.unity.com/packages/2d/textures-materials/water/simple-water-shader-upr-191449>

The screenshot shows a Mozilla Firefox browser window with multiple tabs open. The main tab displays the Unity Asset Store page for the "Simple Water Shader URP" asset. The page features a purple header banner with the text "Save up to 96% on Mega Bundles of time-saving tools to help power your projects". Below the banner, the Unity Asset Store logo is visible, along with a search bar and navigation links for 3D, 2D, Add-Ons, Audio, Essentials, Templates, Tools, VFX, and Sale. The asset itself is categorized under 2D > Textures & Materials > Water. The main content area shows a large image of a 3D scene with water reflections and a checkered floor. To the right, the asset details are listed: "Simple Water Shader URP" by IgniteCoders, rated 4.5 stars (64 reviews), and supported by 100,000+ forum members. It is labeled as "FREE". Below this, a review from user MartinH8921 is shown, stating "Works perfectly. Very pleased." and providing a detailed description of how it helped them in their game development. At the bottom, standard asset store information is provided: License agreement (Standard Unity Asset Store EULA), License type (Extension Asset), and File size (2.8 MB). The browser's address bar and various system icons are also visible.

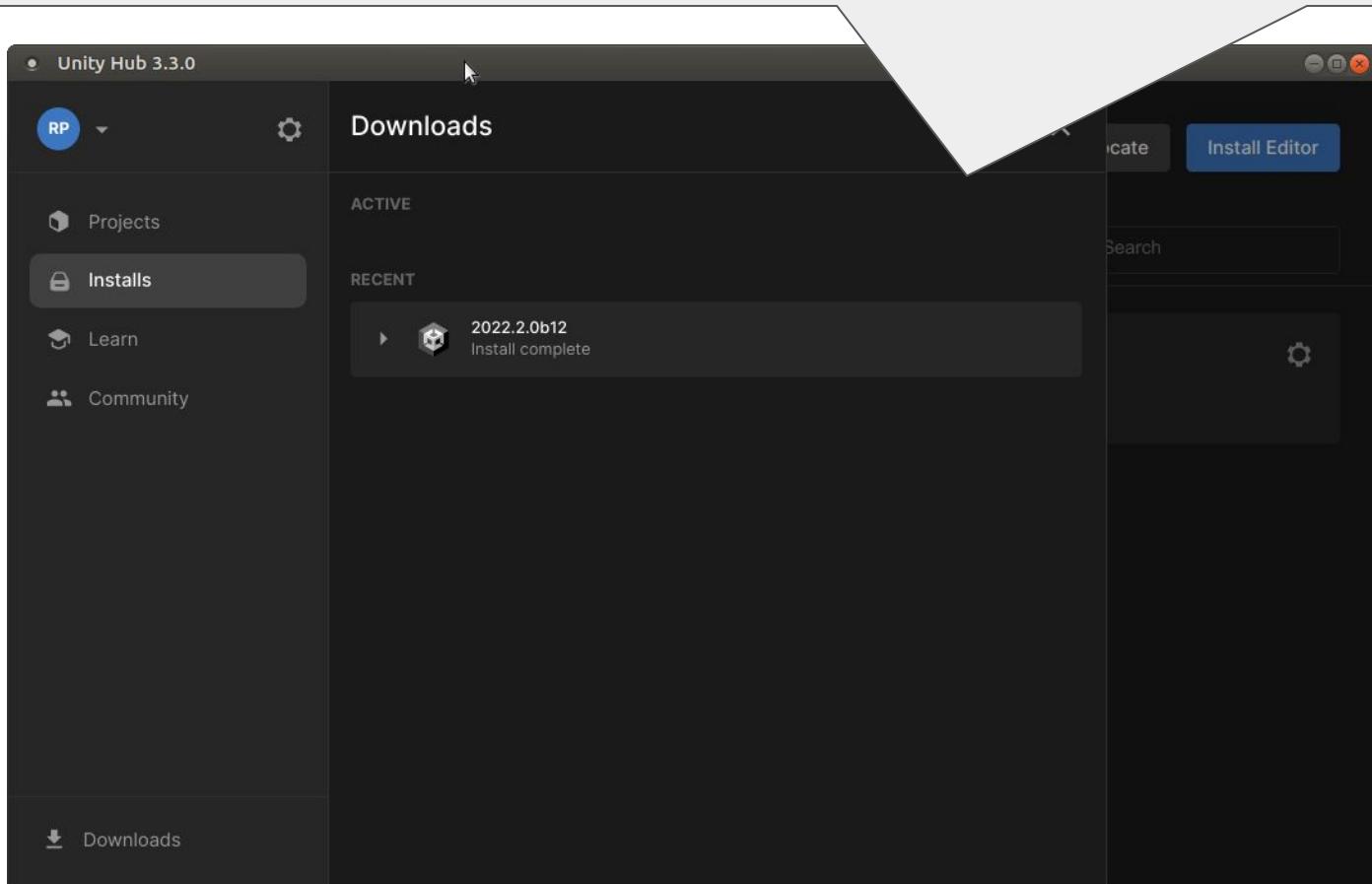


<https://github.com/dbrizov/NaughtyWaterBuoyancy>

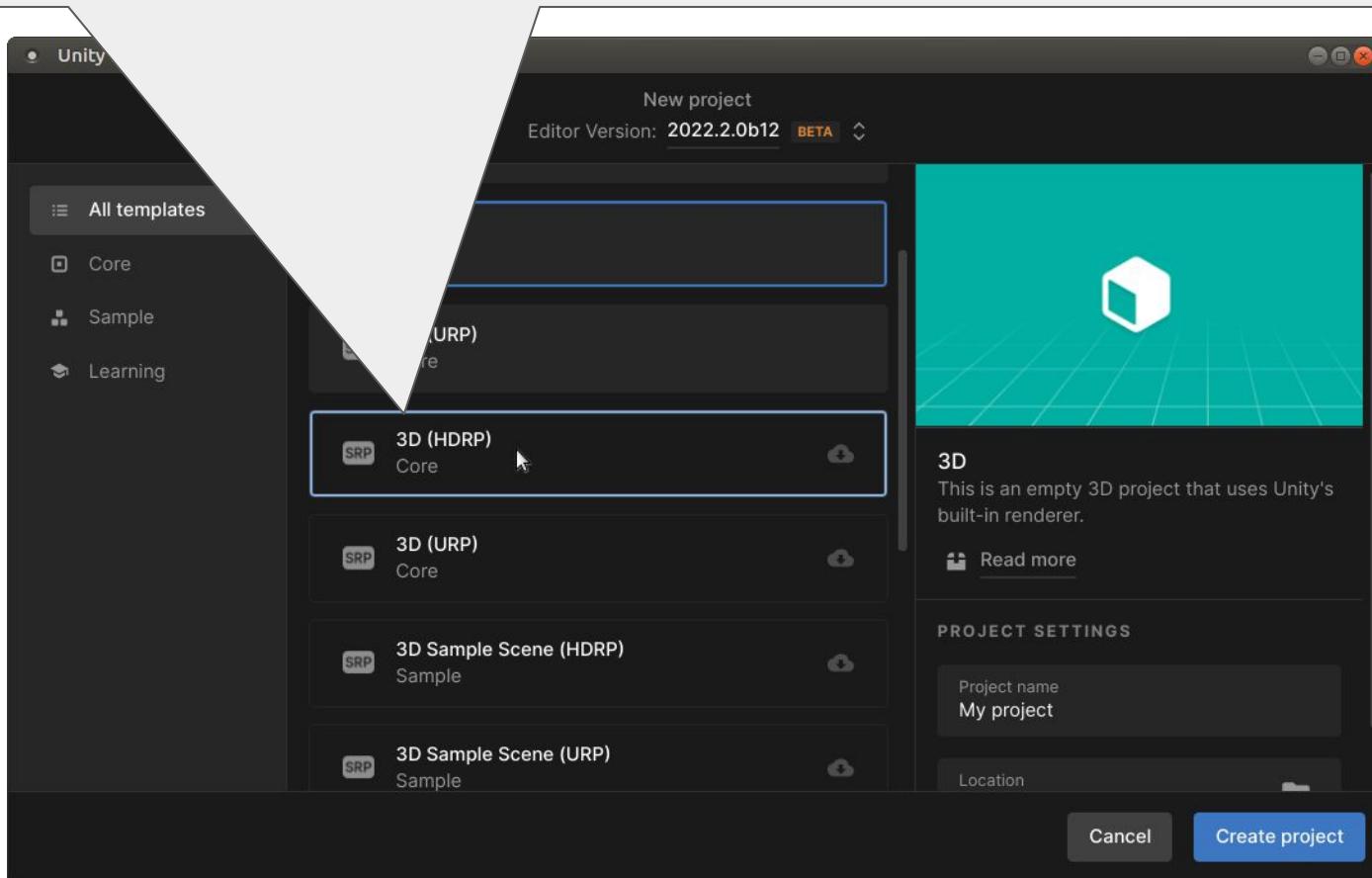
Tutorial:

<https://www.youtube.com/watch?v=KKXZ3860o7w>

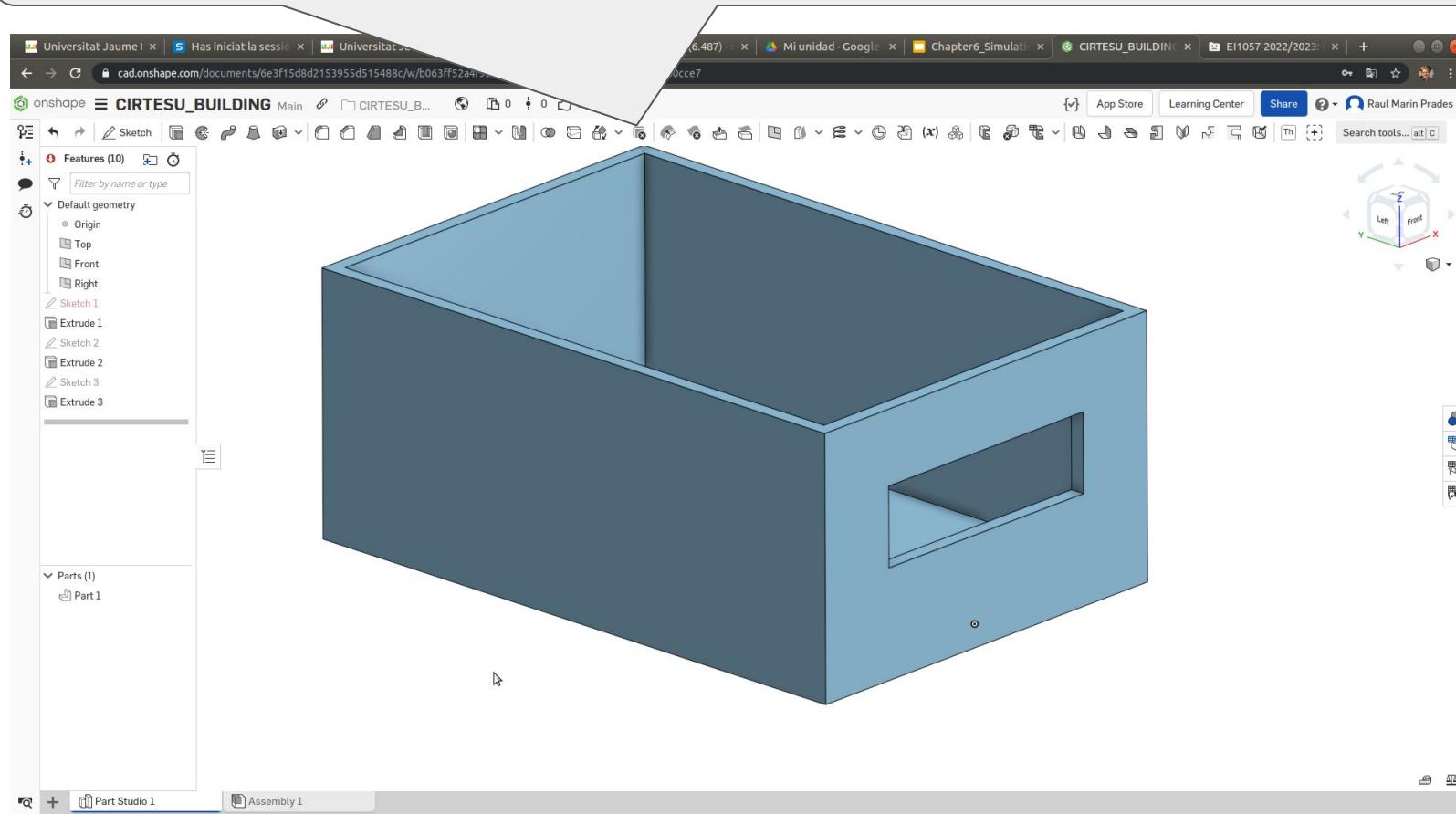
Run Unity Hub and Install Unity editor more recent than 2022.2.0b8



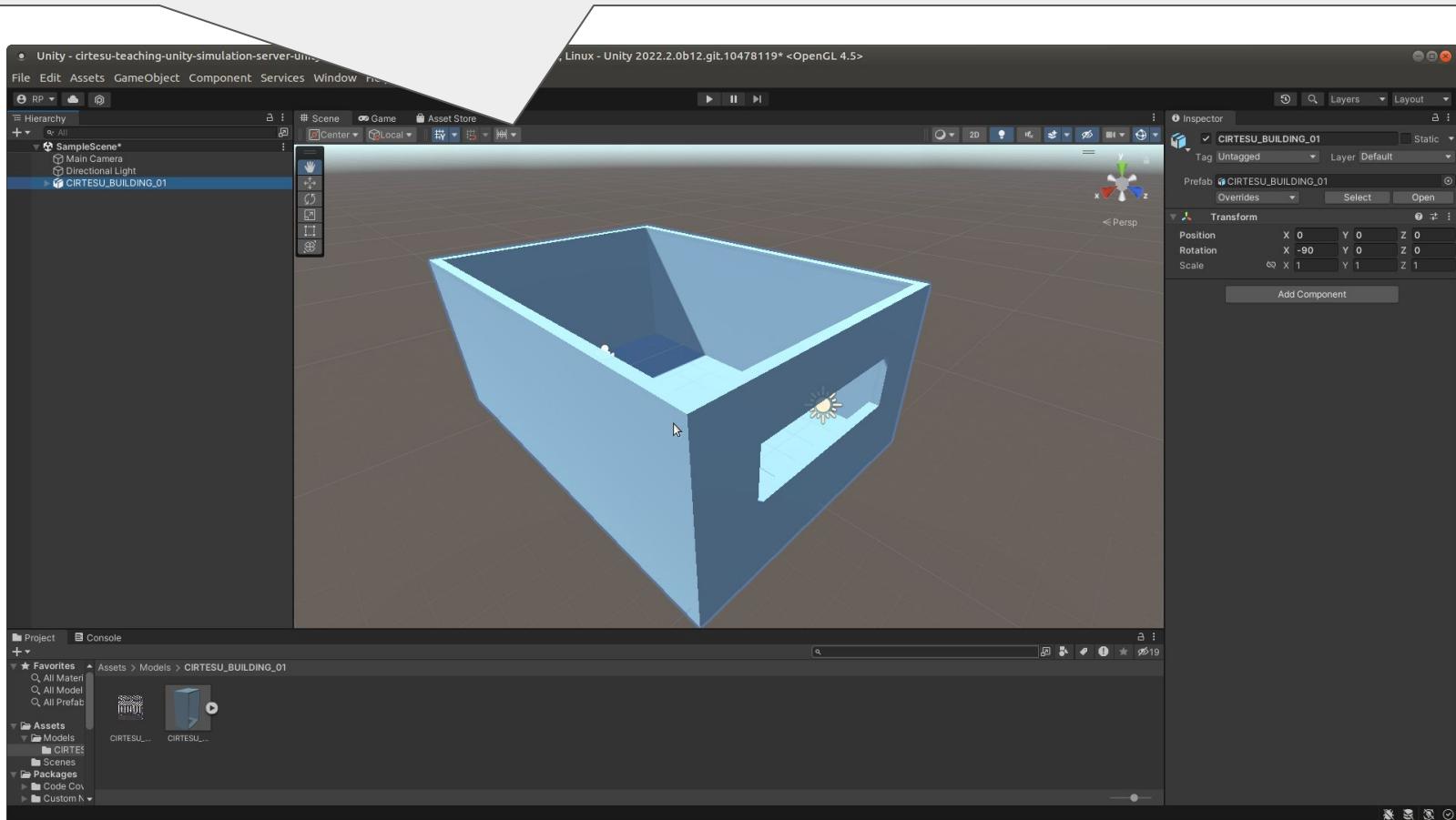
## Create new 3D HDRP Project (contains Water Plane)



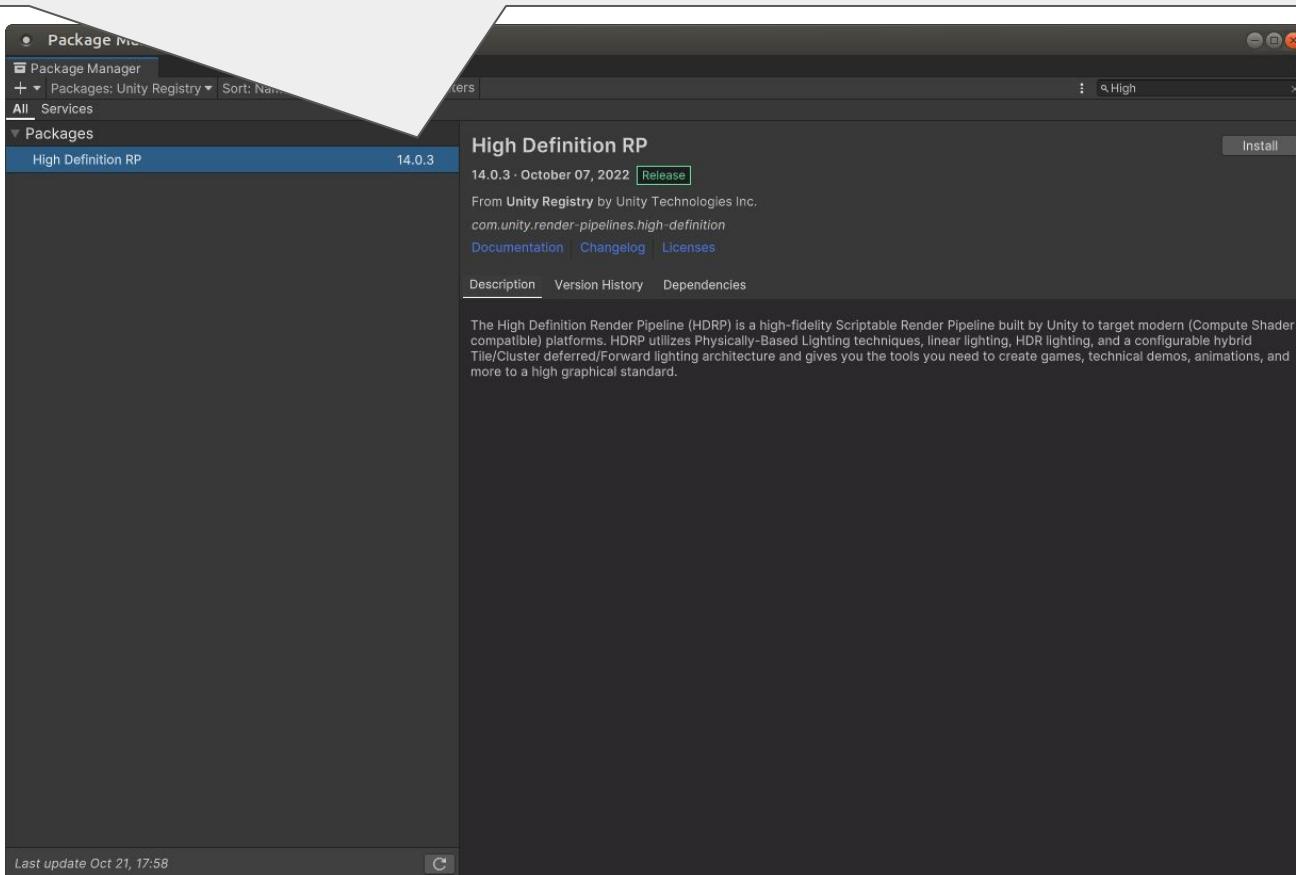
Desing with Unity or a 3D Designer tool (e.g. Onshape) the pool scene, and import it to the Unity Scene



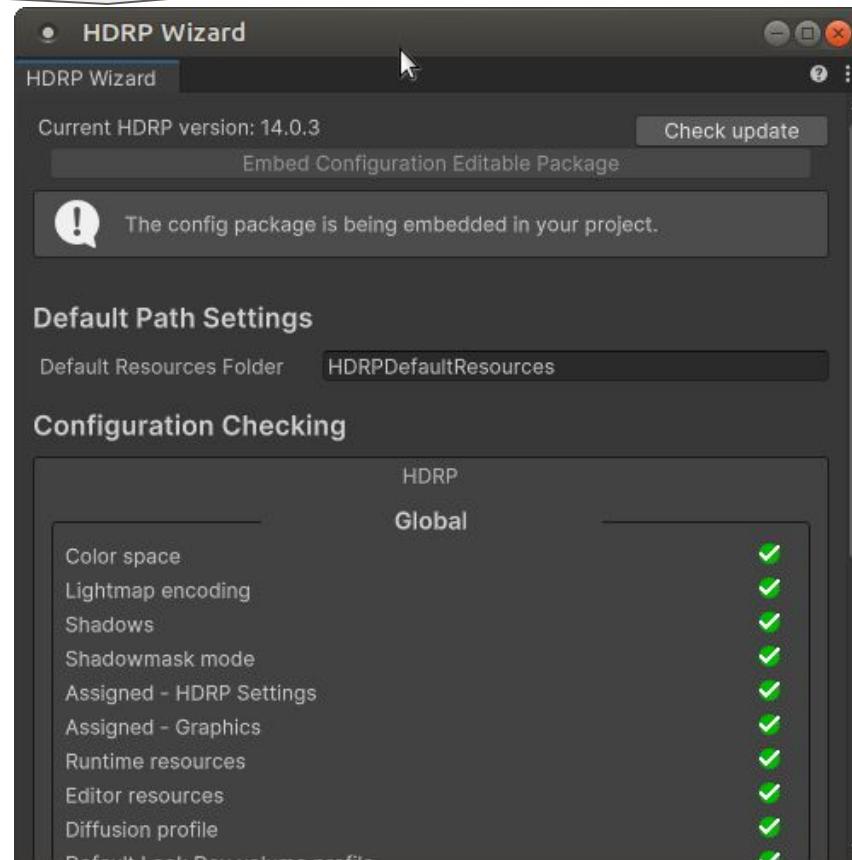
Import model (e.g. obj) to Unity by copying the file in the Assets folder and drag it to the hierarchy window



## Make sure the High Definition Rendering Pipeline Asset is installed



## Review the HDRP Wizard Status and Fix problems



Project Settings

Project Settings

Adaptive Performance  
Audio  
Burst AOT Settings  
Editor  
Graphics  
  HDRP Global Settings  
Input Manager  
Memory Settings  
Package Manager  
Physics  
Physics 2D  
Player  
Preset Manager  
Quality  
  HDRP  
Scene Template  
Script Execution Order  
Services  
  Version Control  
ShaderGraph  
Tags and Layers  
Textmesh Pro  
Time  
Timeline  
Toolchain Management  
UI Builder  
Version Control  
VFX  
Visual Scripting  
XR Plugin Management

## HDRP

HDRenderPipelineAsset

**Rendering**

Color Buffer Format	R11G11B10
Lit Shader Mode	Deferred Only
Multisample Anti-aliasing Quality	None
Motion Vectors	<input checked="" type="checkbox"/>
Runtime AOV API	<input type="checkbox"/>
Dithering Cross-fade	<input checked="" type="checkbox"/>
Terrain Hole	<input type="checkbox"/>
Transparent Backface	<input checked="" type="checkbox"/>
Transparent Depth Prepass	<input checked="" type="checkbox"/>
Transparent Depth Postpass	<input checked="" type="checkbox"/>
Custom Pass	<input checked="" type="checkbox"/>
Custom Buffer Format	R8G8B8A8
Realtime Raytracing (Preview)	<input type="checkbox"/>
Supported Ray Tracing Mode (Preview)	Both

**Ray tracing is not supported on your device. Please refer to the documentation.**

**LOD Bias**

Low	1
Medium	1
High	1

**Maximum LOD Level**

Low	0
Medium	0
High	0

**Decals**

Enable

Draw Distance	1000
Atlas Width	4096
Atlas Height	4096

Metal and Ambient Occlusion Properties

Maximum Clustered Decals on Screen	512
------------------------------------	-----

Layers

Additive Normal Blending

High Precision Normal Buffer

**Dynamic resolution**

▶ Low res Transparency

▶ Water

  Enable

    Simulation Resolution

    Script Interactions

▶ Lighting

▶ Lighting Quality Settings

▶ Material

▶ Post-processing

Project Settings/Quality/HDRP/Enable Water

RP

Hierarchy

- + SampleScene
  - >Main Camera
  - Directional Light
- CIRTESU\_BUILDING\_01

:

Scene Game Asset Store

Center Local

Project

Assets

Editor

Components

Services

Build Settings

Player

Presets

Quality

HDRP

Scene Template

Script Execution Order

Services

Version Control

ShaderGraph

Tags and Layers

TextMeshPro

Time

Timeline

Toolchain Management

UI Builder

Version Control

VFX

Visual Scripting

XR Plugin Management

## Project Settings

## Project Settings

Adaptive Performance

Audio

Burst AOT Settings

Editor

Graphics

HDRP Global Settings

Input Manager

Memory Settings

Package Manager

Physics

Physics 2D

Player

Preset Manager

Quality

HDRP

Scene Template

Script Execution Order

Services

Version Control

ShaderGraph

Tags and Layers

TextMeshPro

Time

Timeline

Toolchain Management

UI Builder

Version Control

VFX

Visual Scripting

XR Plugin Management

## Player

Company Name

Product Name

Version

Default Icon

Default Cursor

Cursor Hotspot

If necessary Deactivate Auto Graphics API for Linux and set Vulkan on top of OpenGL Core

Settings for Windows, Mac, Linux

Icon

Resolution and Presentation

Splash Image

Other Settings

Rendering

Color Space\*

Auto Graphics API for Windows

Auto Graphics API for Mac

Auto Graphics API for Linux

Reordering the list will switch editor to the first available platform

Graphics APIs for Linux

= Vulkan

OpenGLCore

Color Gamut For Mac\*

= sRGB

Static Batching

Sprite Batching Threshold

GPU Compute Skinning\*

Graphics Jobs

Lightmap Encoding

HDR Cubemap Encoding

Lightmap Streaming

Streaming Priority

Frame Timing Stats

OpenGL Profiler GPU Recorders

Virtual Texturing (Experimental)\*

## Console

Clear Collapse Error Pause Editor

[18:32:27] Platform StandaloneLinux64 with graphics API OpenGLCore is not supported with HDRP.  
Change the platform/device to a compatible one or remove incompatible graphics APIs.

[18:33:27] Platform StandaloneLinux64 with graphics API OpenGLCore is not supported with HDRP.  
Change the platform/device to a compatible one or remove incompatible graphics APIs.

[18:34:15] Platform StandaloneLinux64 with graphics API OpenGLCore is not supported with HDRP.  
Change the platform/device to a compatible one or remove incompatible graphics APIs.

[18:34:49] Platform StandaloneLinux64 with graphics API OpenGLCore is not supported with HDRP.  
Change the platform/device to a compatible one or remove incompatible graphics APIs.

Platform StandaloneLinux64 with graphics API OpenGLCore is not supported with HDRP.

RP



Hierarchy

Scene Game Asset Store



SampleScene\*

Main Camera

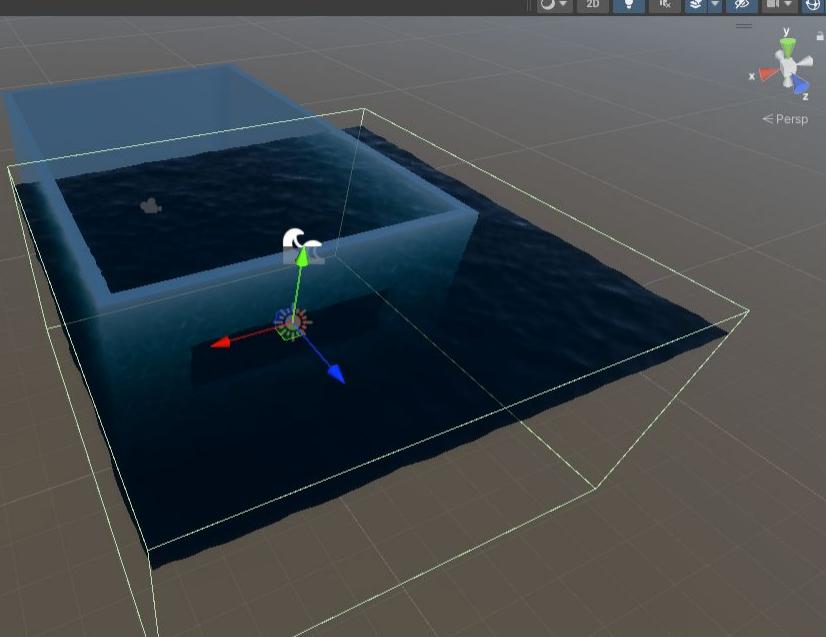
Directional Light

CIRTESU\_BUILDING\_01

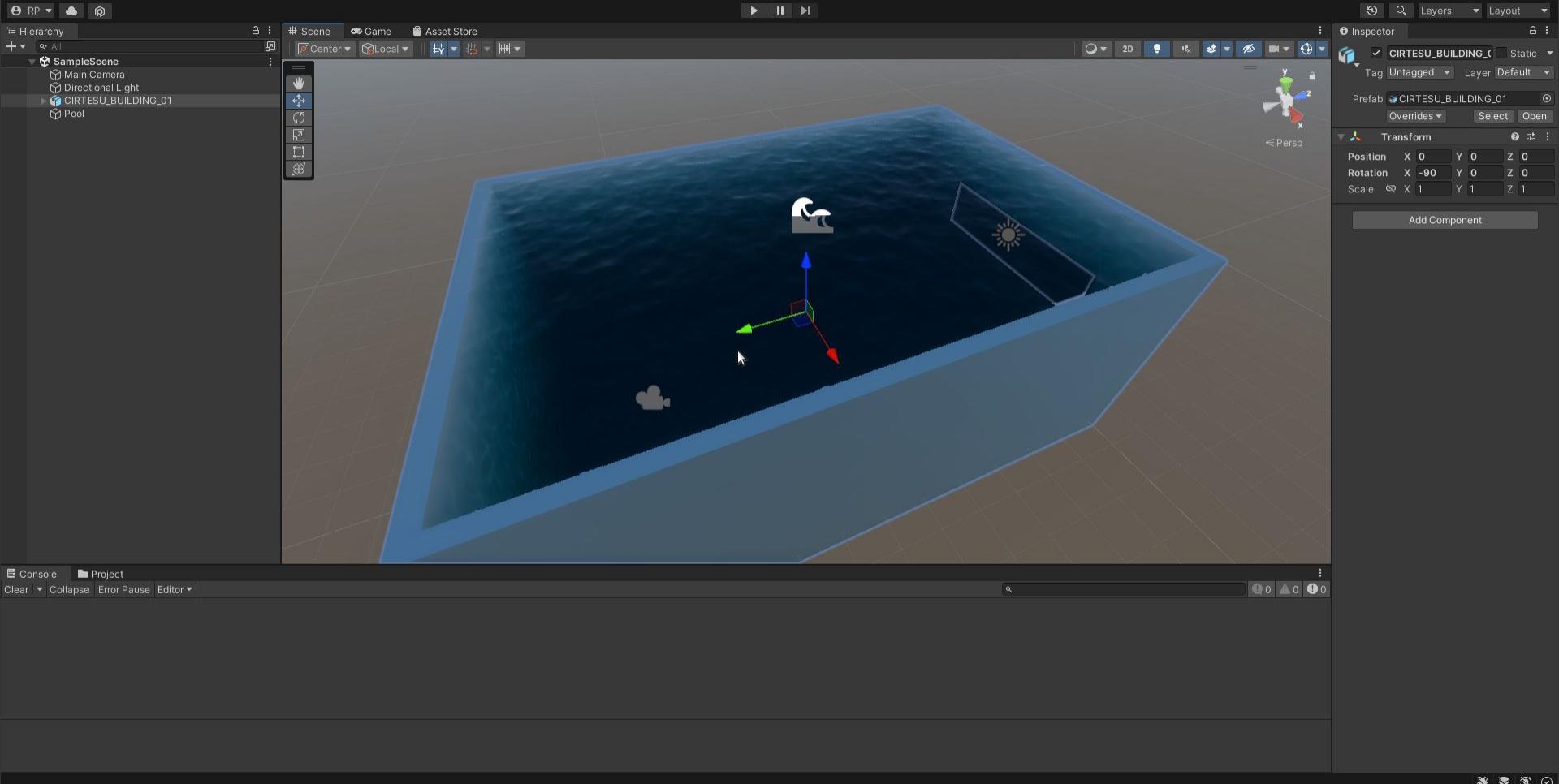
Pool

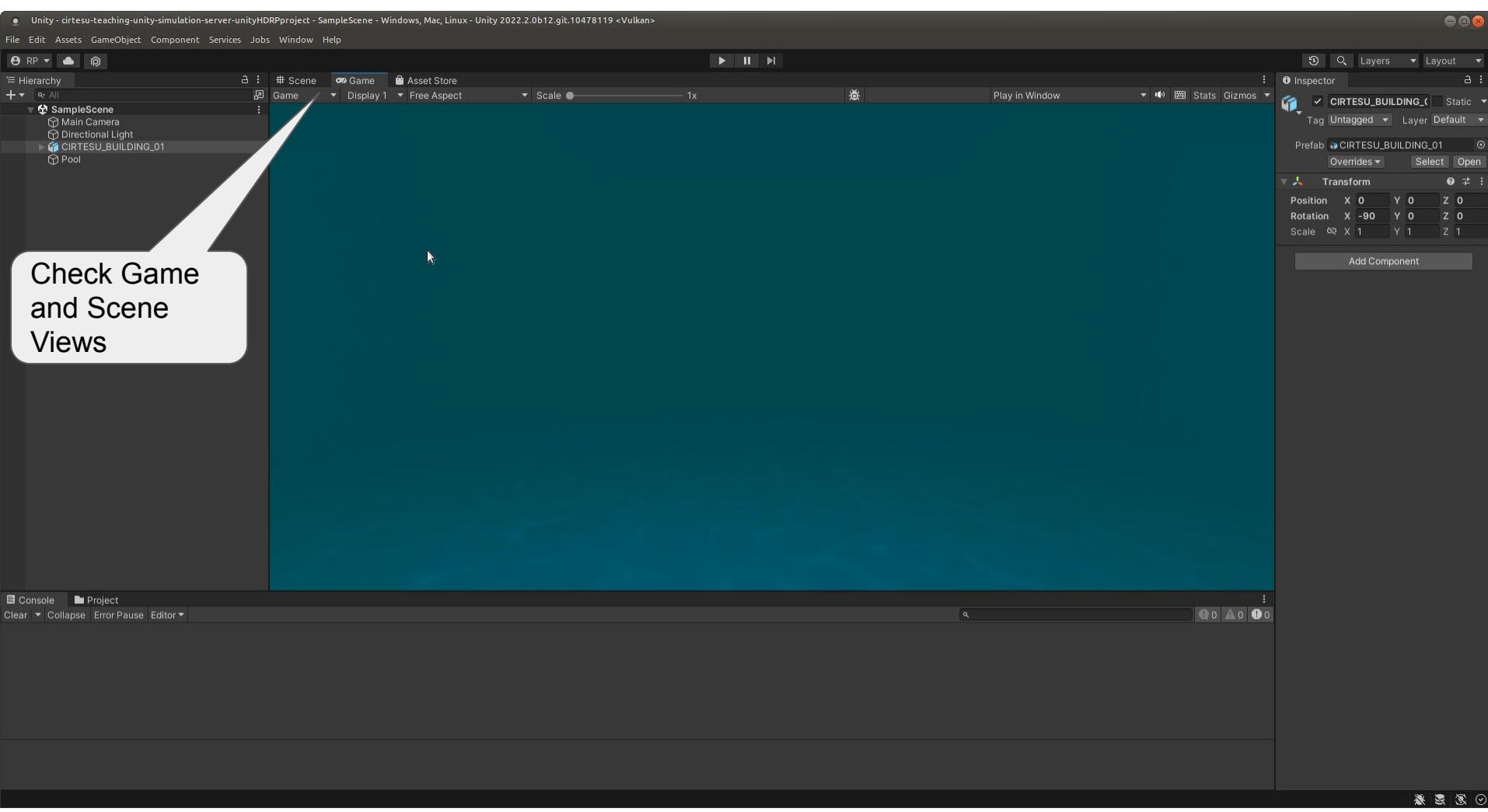


Add Water Pool Plane by  
Assets>Create>Water Plane and  
adjust properties



RP





[https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/quick\\_setup.md](https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/quick_setup.md)



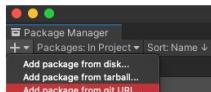
The screenshot shows a GitHub browser interface. The URL in the address bar is [https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/quick\\_setup.md](https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/quick_setup.md). The page title is "Unity-Robotics-Hub / tutorials / quick\_setup.md". The commit history shows a merge from "peifeng-unity" to "Merge v0.5.0 (#260)" at the top, with a note about 8 contributors. Below the commit details, there's a summary of the file: "20 lines (12 sloc) | 1.45 KB". On the right side of the file content area, there are various GitHub UI elements like "Go to file", "Raw", "Blame", and "Edit".

## Installing the Unity Robotics packages

This page provides brief instructions on installing the Unity Robotics packages. Head over to the [Pick-and-Place Tutorial](#) for more detailed instructions and steps for building a sample project.

1. Create or open a Unity project.

Note: If you are adding the URDF-Importer, ensure you are using a [2020.2.0+](#) version of Unity Editor.
2. Open [Window -> Package Manager](#).
3. In the Package Manager window, find and click the [+](#) button in the upper lefthand corner of the window. Select [Add package from git URL...](#).

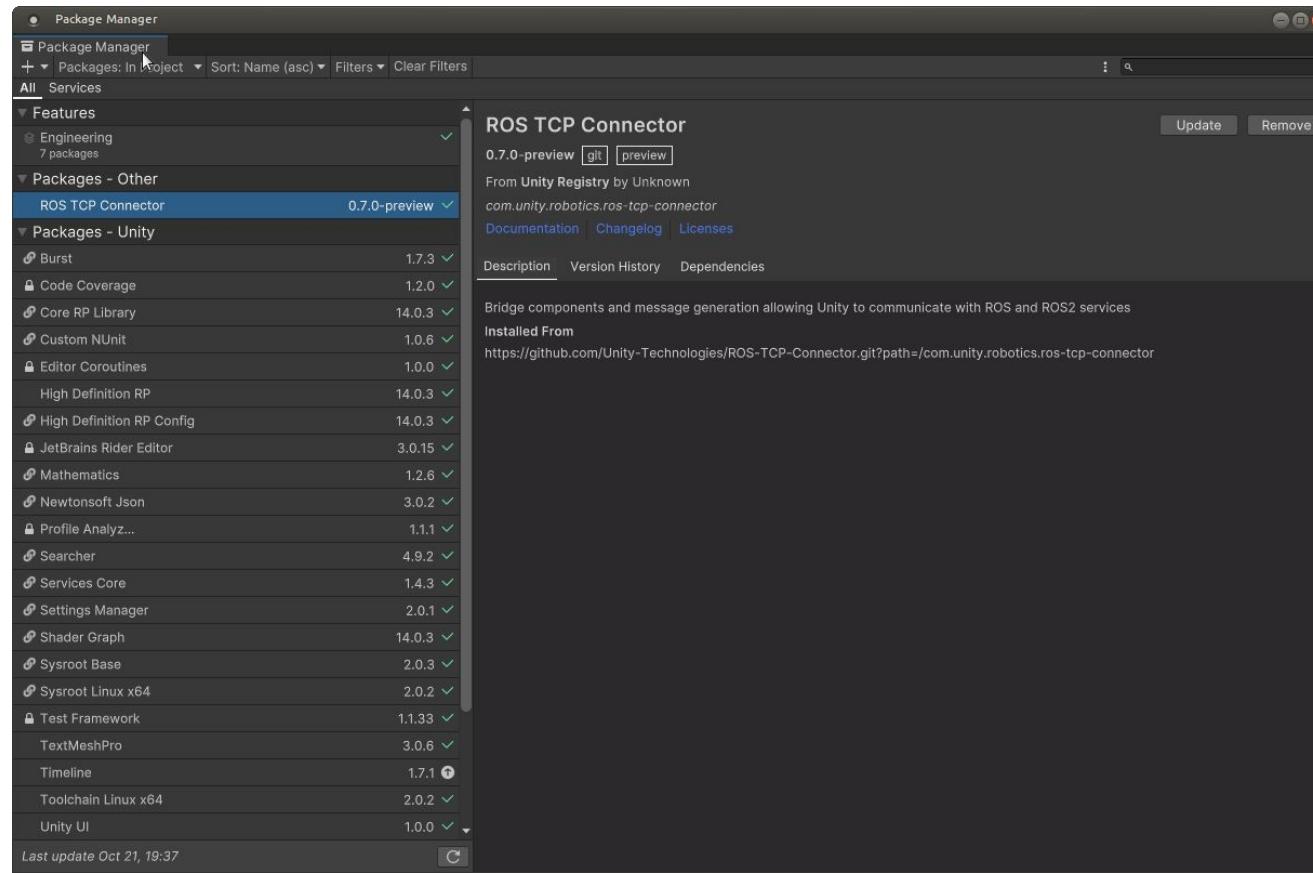


4. Enter the git URL for the desired package. Note: you can append a version tag to the end of the git url, like `#v0.4.0` or `#v0.5.0`, to declare a specific package version, or exclude the tag to get the latest from the package's `main` branch.
  - i. For the [ROS-TCP-Connector](#), enter `https://github.com/Unity-Technologies/ROS-TCP-Connector.git?path=/com.unity.robotics.ros-tcp-connector`.
  - ii. For the [URDF-Importer](#), enter `https://github.com/Unity-Technologies/URDF-Importer.git?path=/com.unity.robotics.urdf-importer`.
5. Click [Add](#).

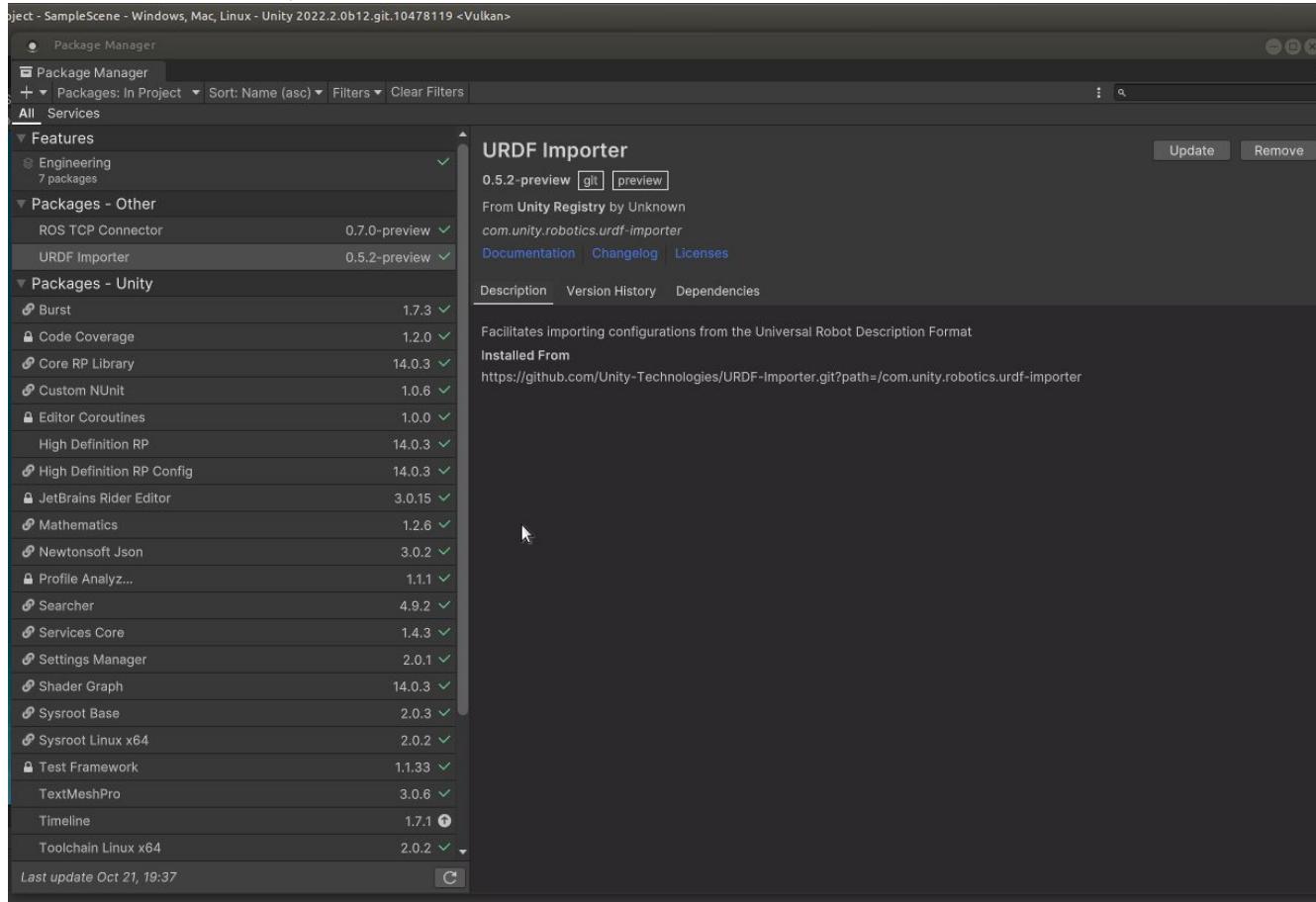
To install from a local clone of the repository, see [installing a local package](#) in the Unity manual.

## Install ROS TCP Connector

<https://github.com/Unity-Technologies/ROS-TCP-Connector.git?path=/com.unity.robotics.ros-tcp-connector>

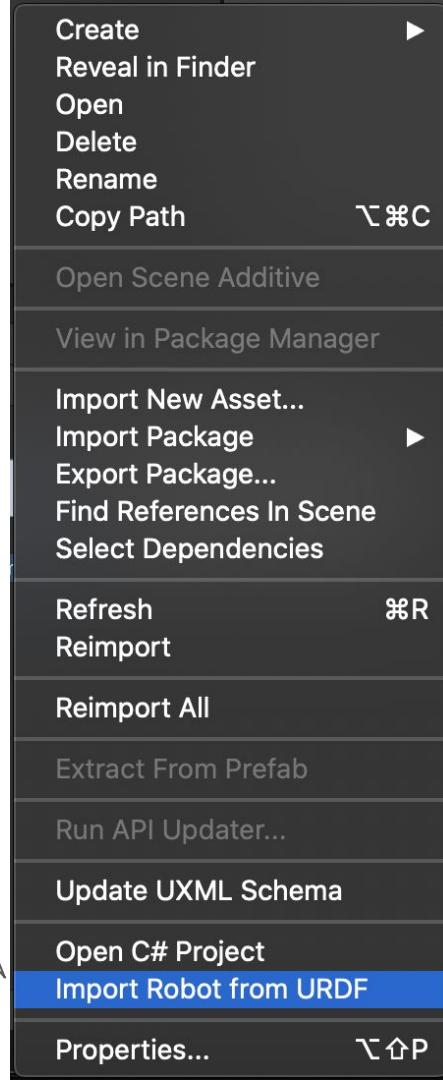


## Install ROS URDF Importer



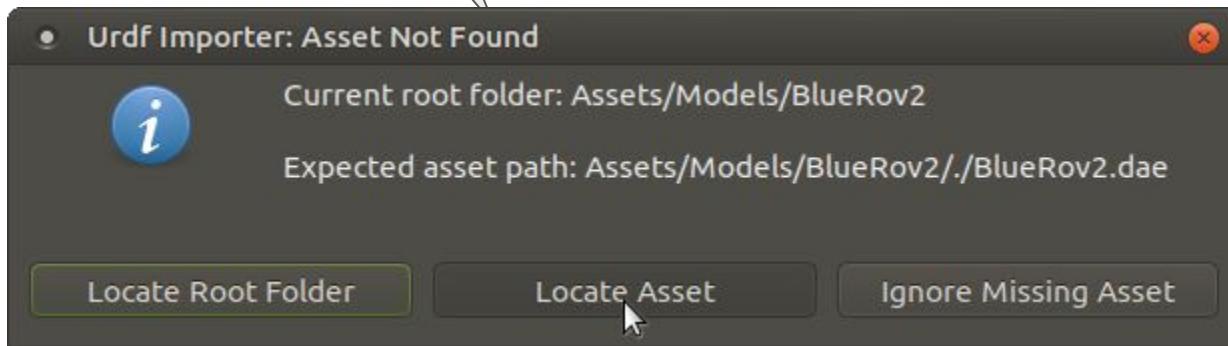
Download Bluerov URDF Xacro File from bluerov\_ros\_playground repository and use xacro tool to get the URDF file. Copy the .dae and urdf files in Assets/Models directory.

[https://github.com/patrickelectric/bluerov\\_ros\\_playground/tree/master/model](https://github.com/patrickelectric/bluerov_ros_playground/tree/master/model)

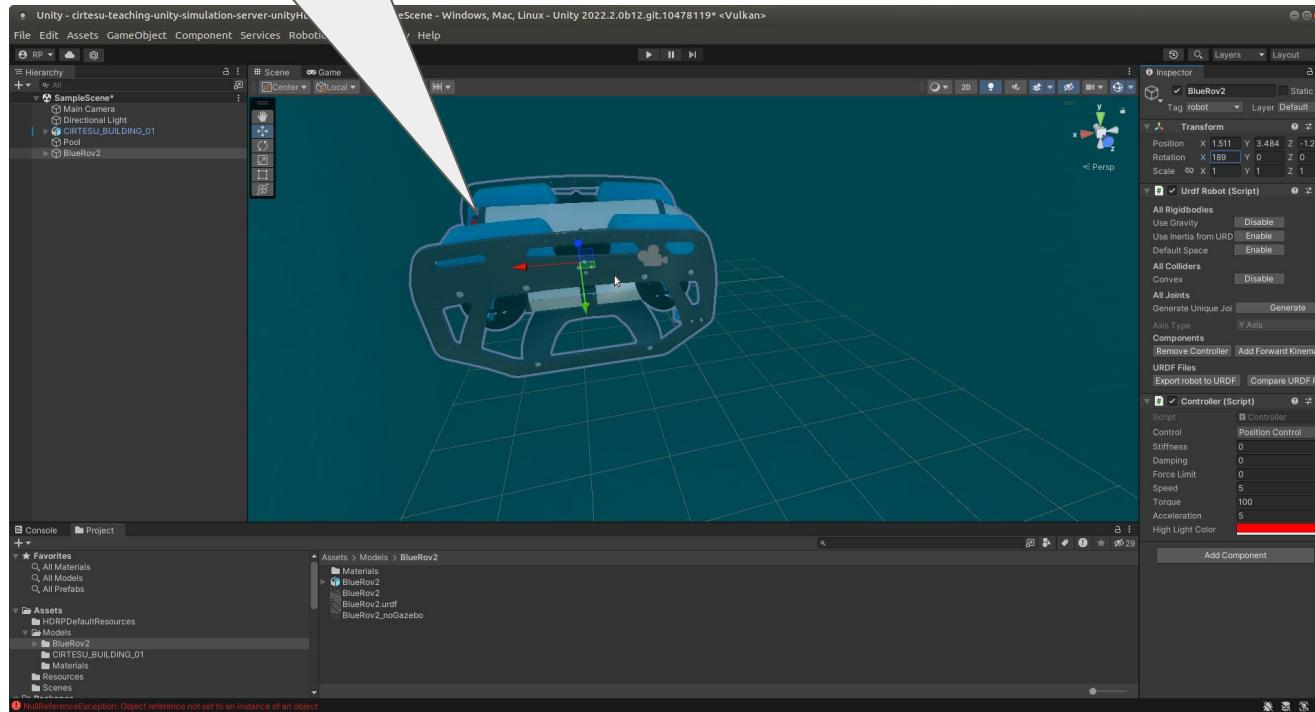


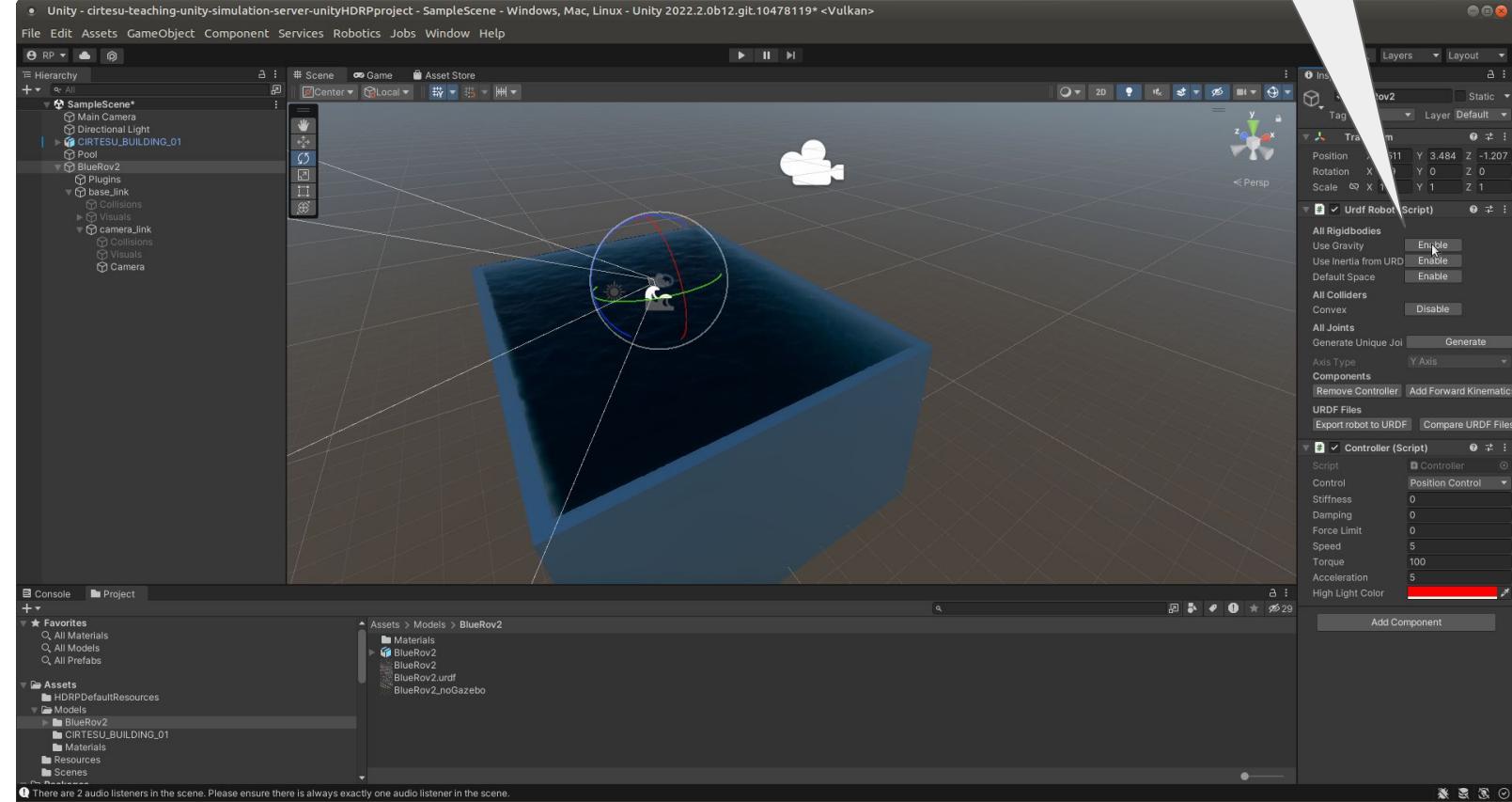
In Unity, Project Window, Right-click the URDF File and select Option... "Import Robot from URDF"

Indicate the BlueRov2.dae file location

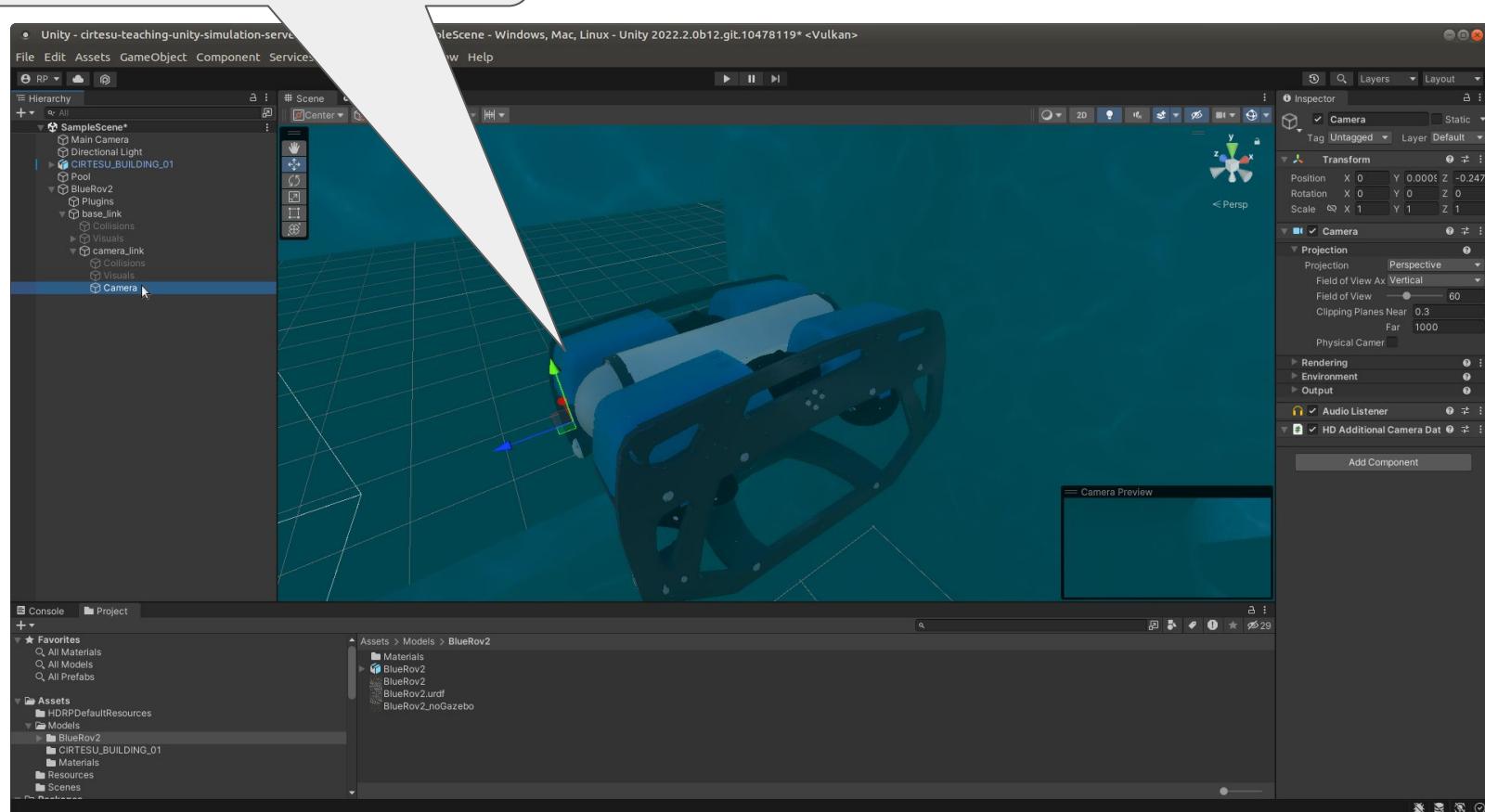


Adjust GameObject Properties to make it show in the right direction

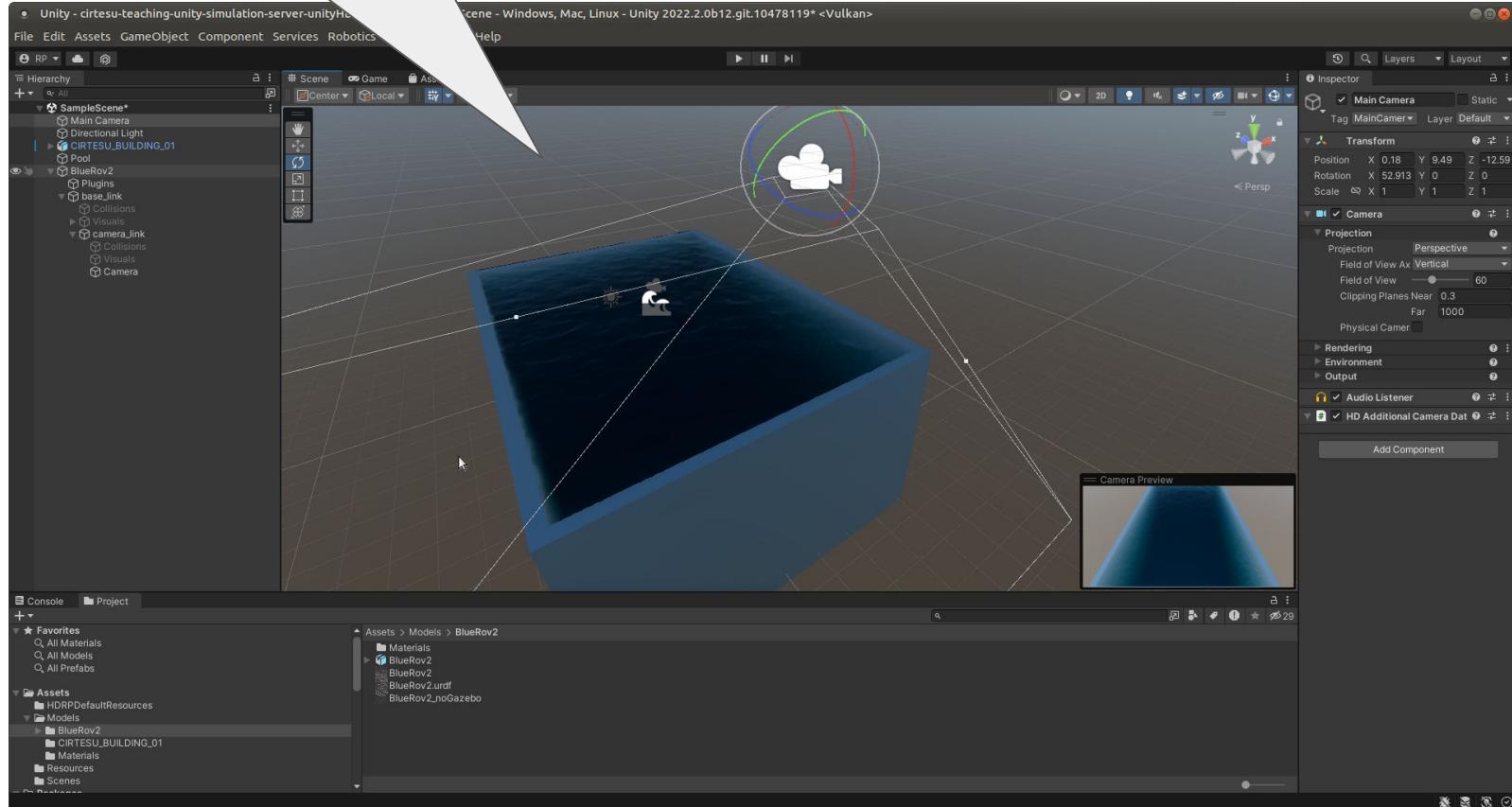


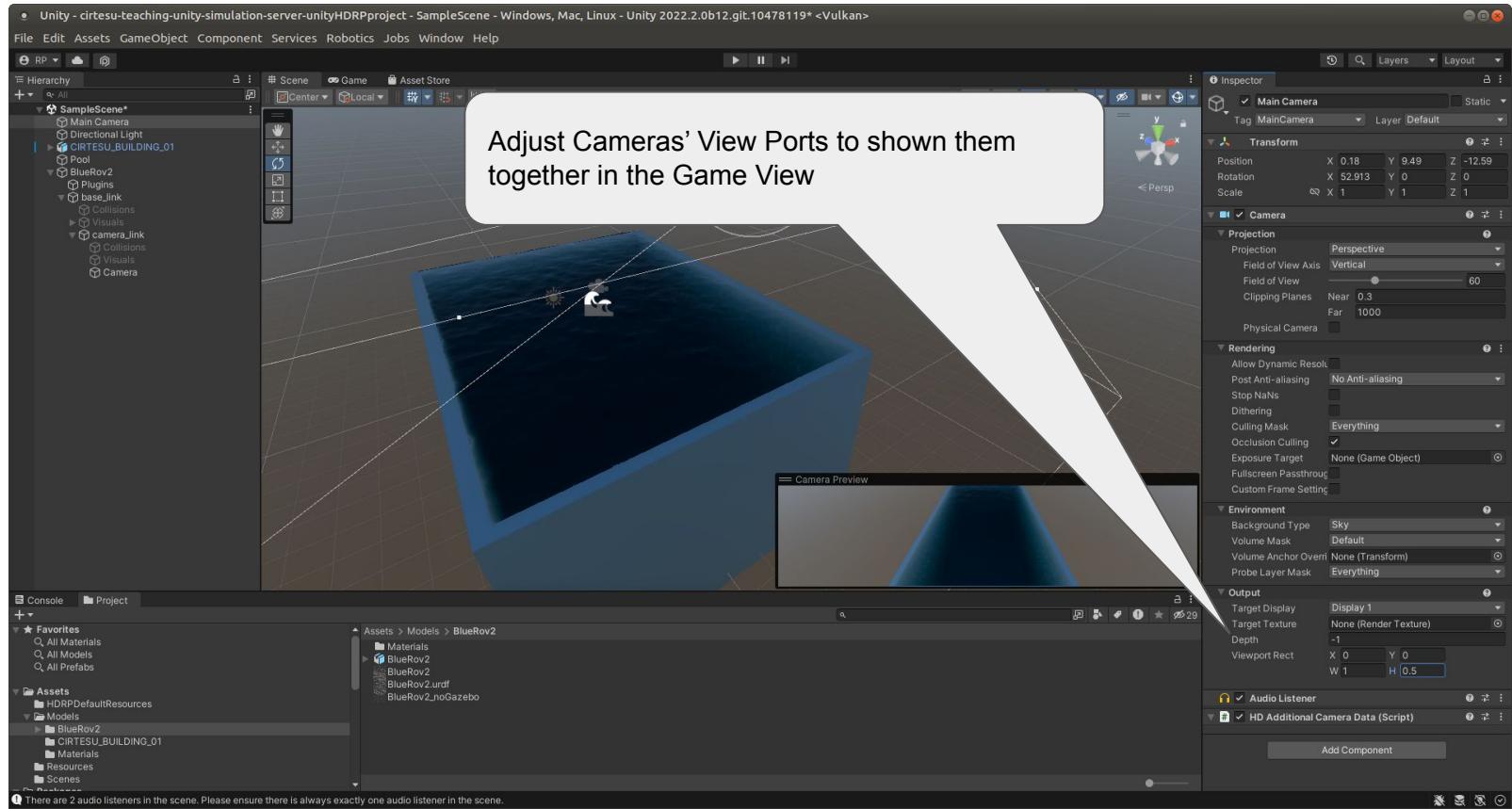


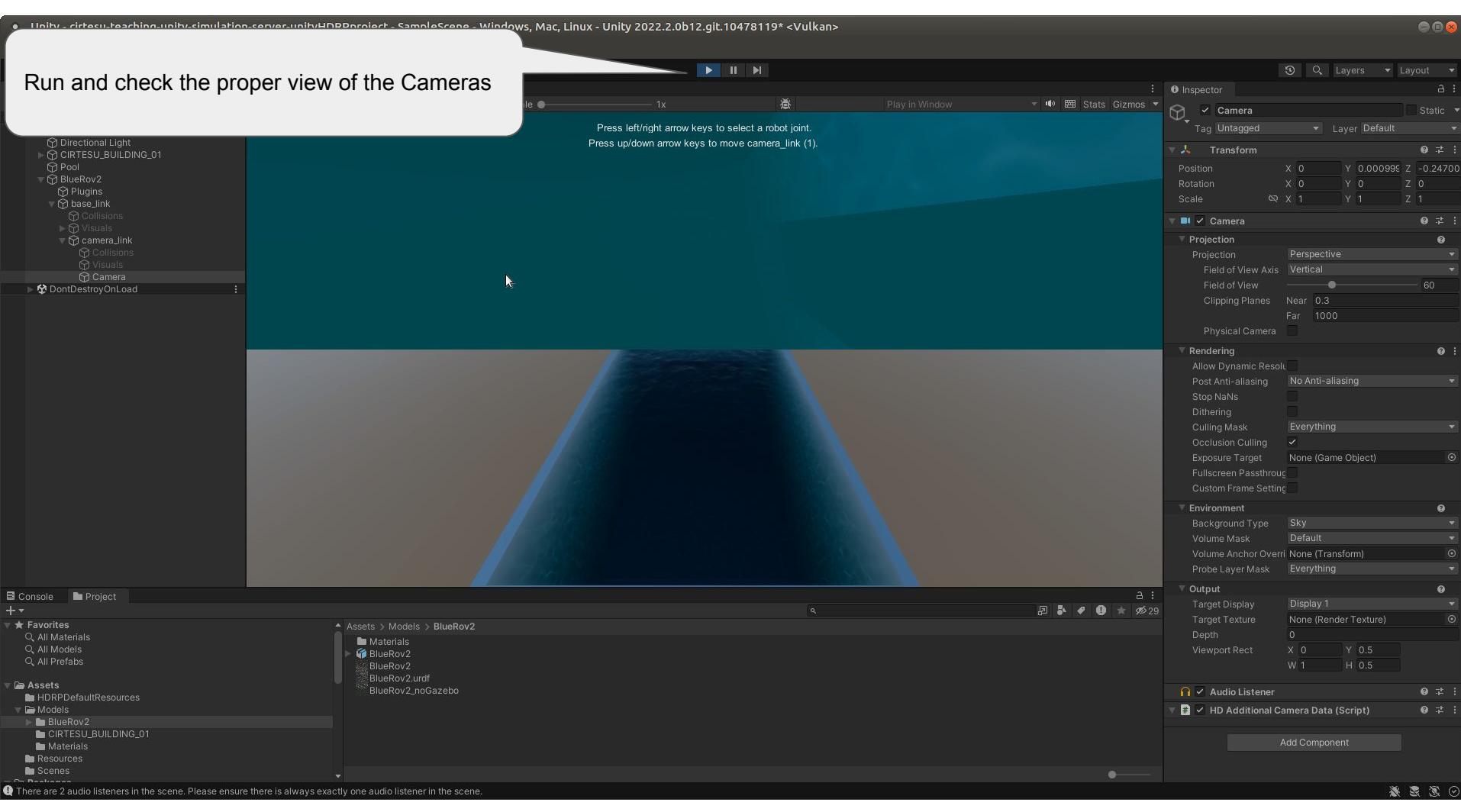
Add GameObject Camera to the Bluerov under camera\_link



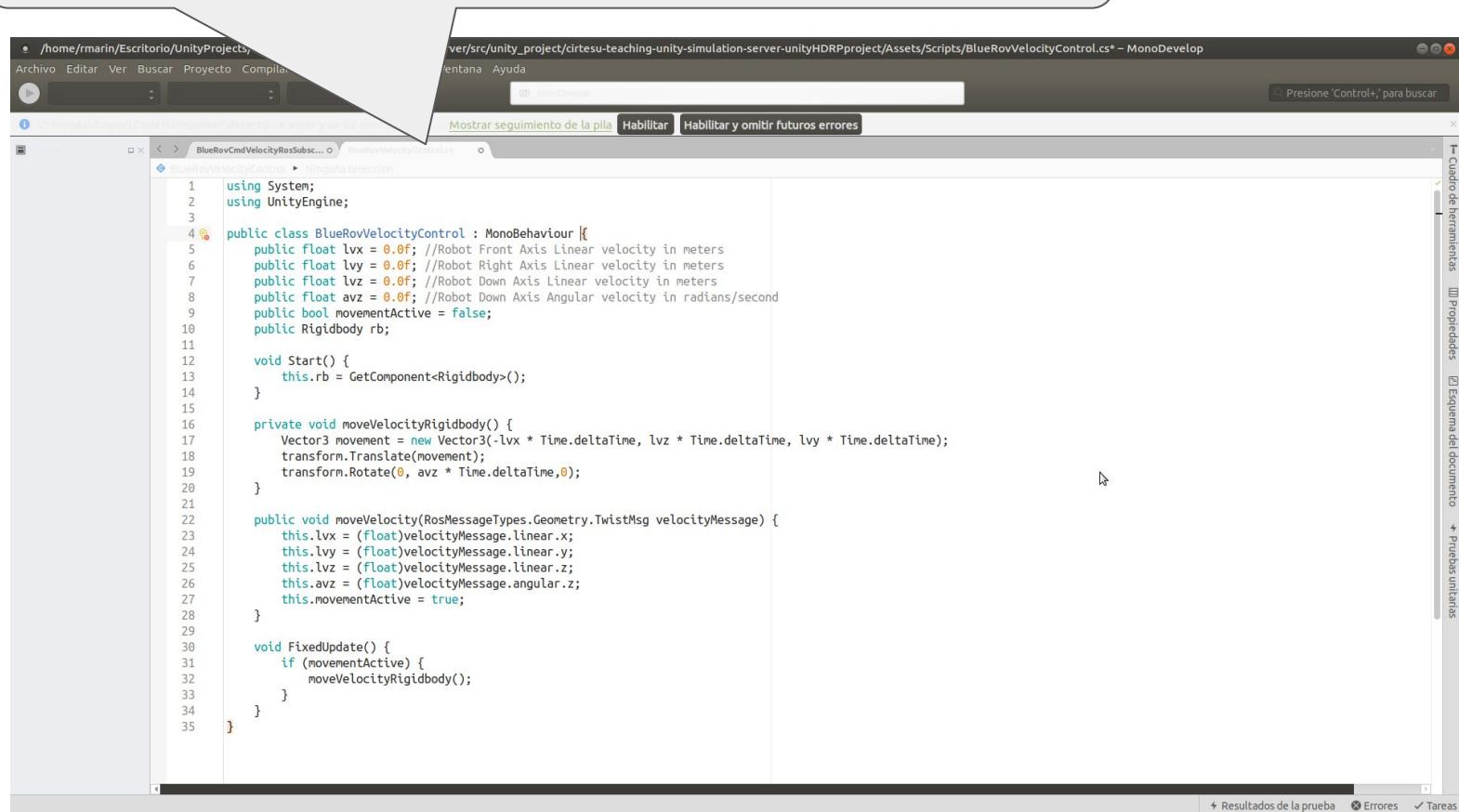
Adjust MainCamera Properties to look at the top of the Pool







Script to move the Bluerov simulated robot. Test changing the properties while running and adjust it as required to your robot



The screenshot shows the MonoDevelop IDE interface with the file `BlueRovVelocityControl.cs` open. The code defines a `BlueRovVelocityControl` class that inherits from `MonoBehaviour`. It contains variables for linear velocities (lvx, lvy, lvz) and angular velocity (avz), and a boolean for movement activity. The `Start()` method initializes the rigidbody component. The `moveVelocityRigidbody()` method updates the robot's position and orientation based on the current time delta. The `moveVelocity` method handles incoming ROS messages. The `FixedUpdate()` method checks if movement is active and calls the movement update if true.

```
using System;
using UnityEngine;

public class BlueRovVelocityControl : MonoBehaviour {
    public float lvx = 0.0f; //Robot Front Axis Linear velocity in meters
    public float lvy = 0.0f; //Robot Right Axis Linear velocity in meters
    public float lvz = 0.0f; //Robot Down Axis Linear velocity in meters
    public float avz = 0.0f; //Robot Down Axis Angular velocity in radians/second
    public bool movementActive = false;
    public Rigidbody rb;

    void Start() {
        this.rb = GetComponent<Rigidbody>();
    }

    private void moveVelocityRigidbody() {
        Vector3 movement = new Vector3(-lvx * Time.deltaTime, lvz * Time.deltaTime, lvy * Time.deltaTime);
        transform.Translate(movement);
        transform.Rotate(0, avz * Time.deltaTime, 0);
    }

    public void moveVelocity(RosMessageTypes.Geometry.TwistMsg velocityMessage) {
        this.lvx = (float)velocityMessage.linear.x;
        this.lvy = (float)velocityMessage.linear.y;
        this.lvz = (float)velocityMessage.linear.z;
        this.avz = (float)velocityMessage.angular.z;
        this.movementActive = true;
    }

    void FixedUpdate() {
        if (movementActive) {
            moveVelocityRigidbody();
        }
    }
}
```

Script to subscribe to ROS Velocities. Add it to the Bluerov imported gameObject

The screenshot shows the Unity Editor interface with a script editor window open. The title bar indicates the path: "/home/rmarin/Escritorio/cirtesu-teaching-unity-simulation-server/src/unity\_project/cirtesu-teaching-unity-simulation-server-unityHDRPproject/Assets/Scripts/BluerovVelocityControl.cs". The menu bar includes "Archivo", "Editar", "Ver", "Buscar", "Proyectos", "Ejecutar", "Herramientas", "Ventana", and "Ayuda". A search bar at the top right says "Presione 'Control+', para buscar". Below the menu is a toolbar with icons for play, stop, and other functions. The main window displays the C# code for "BlueRovVelocityControl.cs". The code defines a class "BlueRovCmdVelocityRosSubscriber" that implements the "MonoBehaviour" interface. It uses the "UnityEngine" and "Unity.Robotics.ROSTCPConnector" namespaces. The class has a public string variable "topic" set to "bluerov1/cmd\_vel" and a public reference to "BlueRovVelocityControl" named "blueRovVelocityControl". The "Start" method subscribes to the "topic" using the "Subscribe<RosMessageTypes.Geometry.TwistMsg>" method of the "ROSConnection" component. The "VelocityChange" method logs the received message and calls the "moveVelocity" method of the "blueRovVelocityControl" component. The code editor has syntax highlighting and a status bar at the bottom.

```
using UnityEngine;
using Unity.Robotics.ROSTCPConnector;

public class BlueRovCmdVelocityRosSubscriber : MonoBehaviour
{
    //public GameObject robot;
    public string topic = "bluerov1/cmd_vel";
    public BlueRovVelocityControl blueRovVelocityControl;

    void Start()
    {
        this.blueRovVelocityControl = GetComponent<BlueRovVelocityControl>();
        ROSConnection.GetOrGetInstance().Subscribe<RosMessageTypes.Geometry.TwistMsg>(topic, VelocityChange);
    }

    void VelocityChange(RosMessageTypes.Geometry.TwistMsg velocityMessage)
    {
        Debug.Log(" " + velocityMessage);
        this.blueRovVelocityControl.moveVelocity(velocityMessage);
    }
}
```

Install ROS TCP End point in your linux ROS workspace:

<https://github.com/Unity-Technologies/ROS-TCP-Endpoint>

Run it with \$roslaunch ros\_tcp\_endpoint endpoint.launch

```
• /home/rmarin/catkin_ws_2022/src/ROS-TCP-Endpoint/launch/endpoint.launch http://localhost:11311
Archivo Editar Ver Buscar Terminal Solapas Ayuda
rmarin@MSI-PE70-6QE: ~ × /home/rmarin/catkin_ws_2022/src/... × rmarin@MSI-PE70-6QE: ~/catkin_w... × rmarin@MSI-PE70-6QE: ~/catkin_w...
rmarin@MSI-PE70-6QE:~/catkin_ws_2022/src$ rosrun ros_tcp_endpoint endpoint.launch
... logging to /home/rmarin/.ros/log/30a108aa-54b0-11ed-981c-e094679c73e5/roslaunch-MSI-PE70-6QE-23280.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://MSI-PE70-6QE:45695/

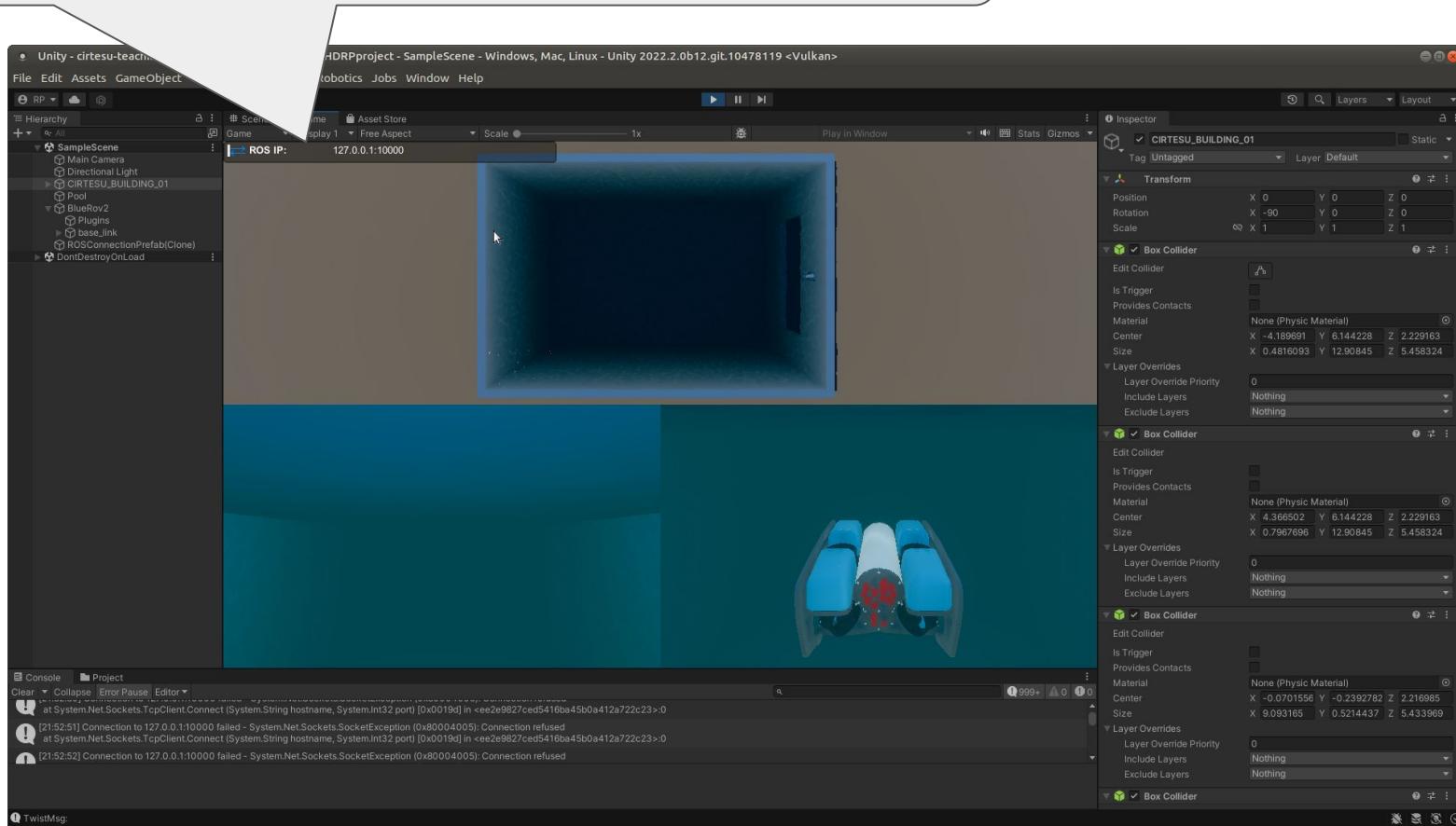
SUMMARY
=====
PARAMETERS
  * /rosdistro: melodic
  * /rosversion: 1.14.3
  * /unity_endpoint/tcp_ip: 0.0.0.0
  * /unity_endpoint/tcp_port: 10000

NODES
  /
    unity_endpoint (ros_tcp_endpoint/default_server_endpoint.py)

auto-starting new master
process[master]: started with pid [23290]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 30a108aa-54b0-11ed-981c-e094679c73e5
process[rosout-1]: started with pid [23302]
started core service [/rosout]
process[unity_endpoint-2]: started with pid [23305]
[INFO] [1666735123.931678]: Starting server on 0.0.0.0:10000
[INFO] [1666735124.164807]: Connection from 127.0.0.1
[INFO] [1666735124.197443]: RegisterSubscriber(bluerov1/cmd_velocity, <class 'geometry_msgs.msg._Twist.Twist'>) OK
[INFO] [1666735124.223081]: RegisterSubscriber(/tf, <class 'tf2_msgs.msg._TFMessage.TFMessage'>) OK
```

Connected to ros\_tcp\_endpoint



## ROS1 Example of Teleop using keyboard

Install and run:

[https://github.com/ros-teleop/teleop\\_twist\\_keyboard.git](https://github.com/ros-teleop/teleop_twist_keyboard.git)

ROS2:

[https://github.com/ros2/teleop\\_twist\\_keyboard.git](https://github.com/ros2/teleop_twist_keyboard.git)

```
● rmarin@MSI-PE70-6QE: ~/catkin_ws_2022/src
Archivo Editar Ver Buscar Terminal Solapas Ayuda
rmarin@MSI-P... ✘ /home/rmarin/... ✘ rmarin@MSI-PE... ✘ rmarin@MSI-PE...
For Holonomic mode (strafing), hold down the shift key:
-----
U      I      0
J      K      L
M      <      >

t : up (+z)
b : down (-z)

anything else : stop

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%

CTRL-C to quit

currently:      speed 0.5          turn 1.0
```

## Script to Capture the JPG from a Camera render

```
using System;
using UnityEngine;

public class CameraCapturer : MonoBehaviour
{
    public bool IsCaptureEnable = false;
    Camera _camera;
    public int resWidth;
    public int resHeight;
    public byte[] jpg;

    void Start() {
        this._camera = GetComponent<Camera>();
    }

    private void FixedUpdate() {
        if (this.IsCaptureEnable) {
            this.jpg = getJPGFromCurrentCamera();
            this.IsCaptureEnable = false;
        }
    }

    public byte[] getCapturedJpegImage() {
        this.IsCaptureEnable = true;
        while (this.IsCaptureEnable) { }
        if (this.jpg != null) {
            return this.jpg;
        }
        return null;
    }

    private byte[] getJPGFromCurrentCamera() {
        try {
            RenderTexture rt = new RenderTexture(resWidth, resHeight, 24);
            _camera.targetTexture = rt;
            Texture2D screenShot = new Texture2D(resWidth, resHeight, TextureFormat.RGB24, false);
            _camera.Render();
            RenderTexture.active = rt;
            screenShot.ReadPixels(new Rect(0, 0, resWidth, resHeight), 0, 0);
            _camera.targetTexture = null;
            RenderTexture.active = null;
            Destroy(rt);
            byte[] bytes = screenShot.EncodeToJPG();
            return bytes;
        }
        catch (Exception e) {
            return null;
        }
    }
}
```

# Exercise

Add an script to the Bluerov Camera in order to Publish in a ROS Camera topic

Visualize the camera topic via RVIZ or ROS package

Teleoperate the simulated BlueRov from ROS, visualizing the camera and sending cmd\_vel commands via keyboard

# Useful links

Onshape tutorials

<https://learn.onshape.com/catalog>

Onshape-to-robot documentation

<https://onshape-to-robot.readthedocs.io/en/latest/>

Onshape-to-robot examples

<https://github.com/rhoban/onshape-to-robot-examples/>