

Sviluppo di Applicazioni AR

Game As a Lab – Anna Vitali

anna.vitali7@unibo.it

Link al Materiale

- Scansionando il **QRCode** potrete visionare il repository contenente il materiale
 - Potete **clonare** il Repository con il comando
`git clone`
 - **Scaricare** direttamente i **files** che vi servono
- Leggere attentamente il `README.md`



Realtà Aumentata?

Augmented Reality

- **Arricchisce** la nostra percezione del mondo fisico riuscendo a **sovraporre** media allineati spazialmente in tempo reale
- L'utente **è in grado** di vedere il mondo esterno
- Il sistema **non effettua**, o non è in grado di realizzare un **mapping spaziale** dell'ambiente, la sua comprensione dello spazio fisico è limitata



Virtual Reality

- Avanzata *human-computer-interface*, in grado di **simulare** un ambiente realistico, **mondo virtuale**, in cui il partecipante all'esperienza viene immerso
- La visione dell'utente viene **oscurata**, non è possibile vedere il mondo esterno



Realtà Aumentata?

Mixed Reality

- **Unisce** mondo virtuale e mondo fisico
- I sistemi di *MR* possiedono una **consapevolezza spaziale**
- L'utente **è in grado** di vedere il mondo esterno
- Gli **ologrammi** sono percepiti come **oggetti reali**, grazie al tipo di **interazione** offerta e all'**occlusione**



Ma che fine ha fatto il Metaverso?

- Premessa: il **metaverso** attualmente non **esiste**, si tratta di una visione a cui si vuole ambire
- Lo scrittore **Neal Stephenson** ha creato questa parola, all'interno del suo romanzo *Snow Crash*
- Questo termine ha acquisito popolarità nel **2022** grazie a **Mark Zuckerberg** quando ha deciso di nominare la sua azienda da **Facebook** a **Meta**



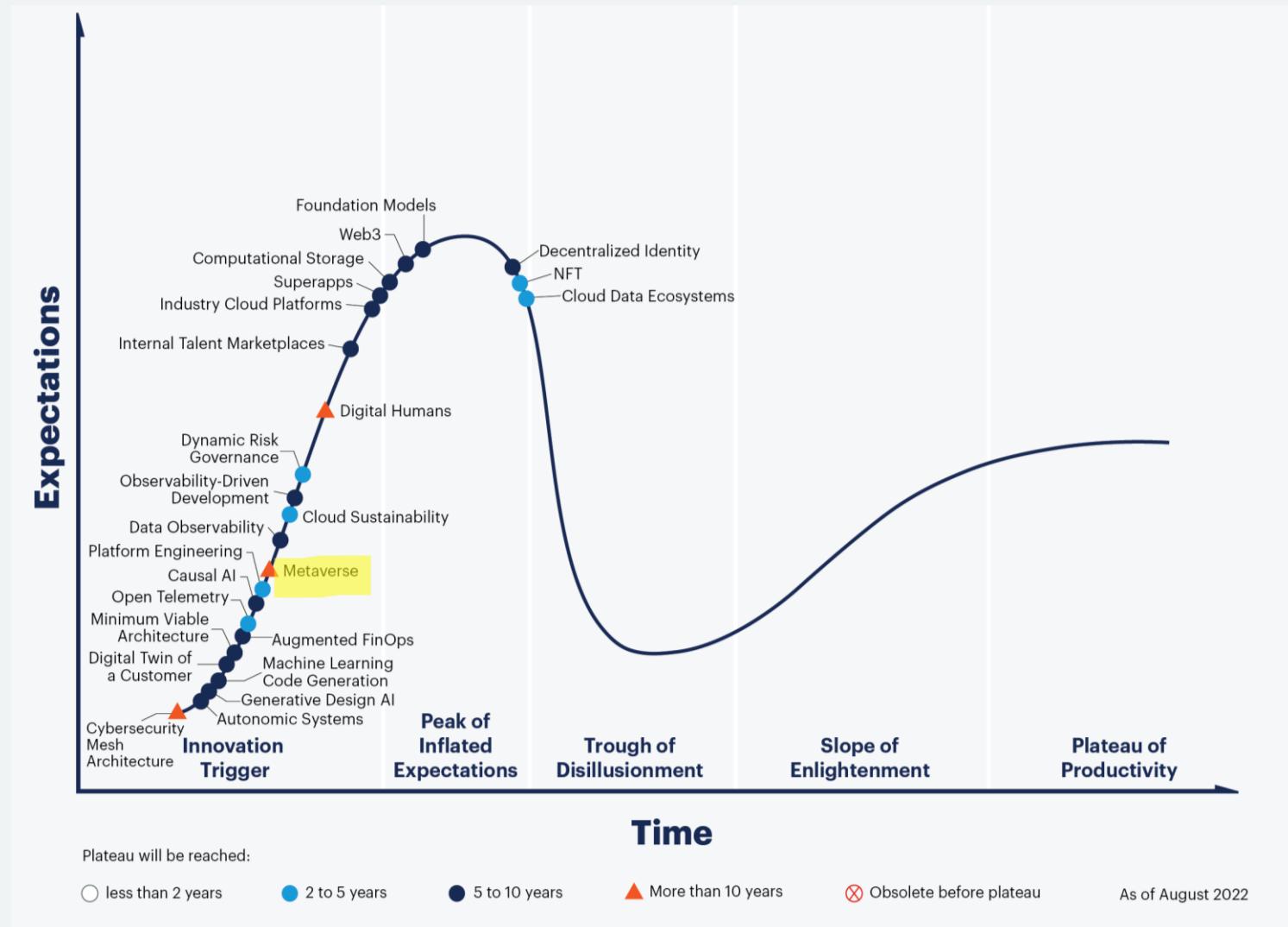
Ma che fine ha fatto il Metaverso?

Poi, proprio come le bolle tech degli anni Novanta, **si è fermato tutto, improvvisamente**. L'azienda di Mark Zuckerberg – la più grande fautrice del progetto – ha virato a 360 gradi e ha spostato uomini e investimenti dal metaverso al nuovo santo graal del mondo tech: l'intelligenza artificiale. Il boom di [ChatGPT](#) e di tutto l'universo collegato all'AI generativa ha fatto crollare le fondamenta del metaverso, lasciando tutti spiazzati. Ma dunque parliamo di una rivoluzione soltanto rimandata oppure di un **grande bluff**?

Fonte: Ok, la bolla è scoppiata e l'attenzione si è spostata tutta sull'intelligenza artificiale. Ma quel gigantesco ambiente virtuale che avevamo immaginato è soltanto un bluff?

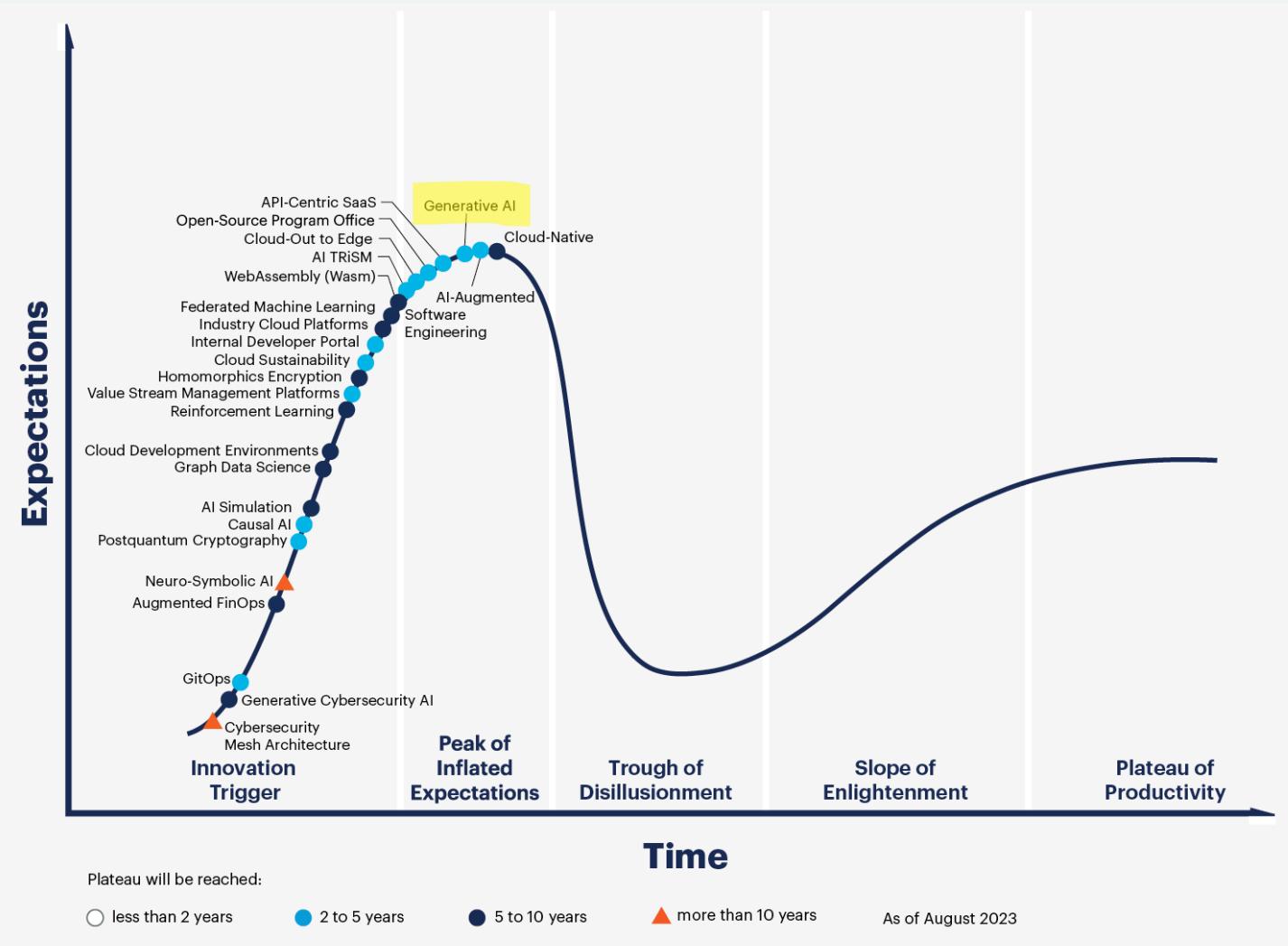
Ma che fine ha fatto il Metaverso?

Gartner Hype Cycle
for Emerging
Technologies
2022



Ma che fine ha fatto il Metaverso?

Gartner Hype Cycle
for Emerging
Technologies
2023



Potenzialità della realtà aumentata

- La Realtà Aumentata rientra in un concetto più ampio chiamato ***Spatial Computing***

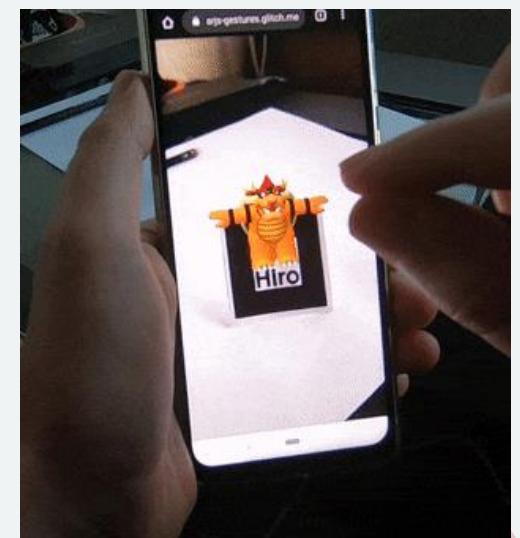
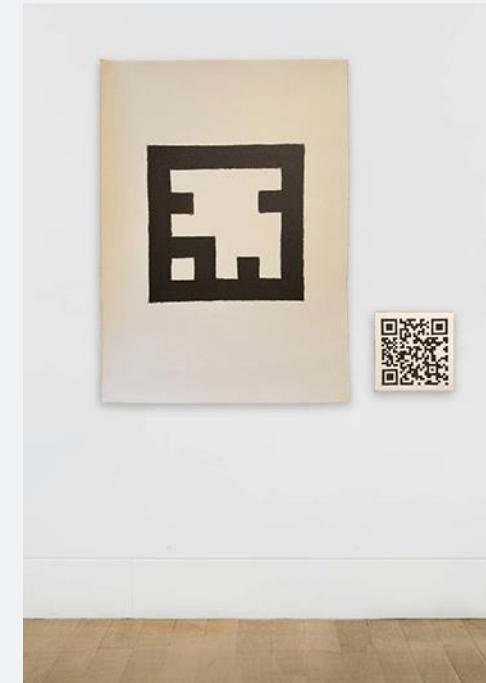
*"I define **Spatial Computing** as human interaction with a **machine in which the machine retains and manipulates referents to real objects and spaces.**[...] Spatial computing proposes hybrid real/virtual computation that **erodes the barriers between the physical and the ideal worlds.**[...] Wherever possible the machine in space and space in the machine should be allowed to **bleed into each other.** Sometimes this means bringing space into the computer, sometimes this means **injecting computation into objects**"*

Simon Greenwold – "Spatial Computing"

Come si interagisce con lo spazio?

Marked Based AR

- Si basa sull'identificazione di **elementi** presenti nell'ambiente, chiamati **marker**
- Il contenuto olografico può essere **posizionato** nell'ambiente sulla base della **posizione del marker**
 - Es sopra al marker o a 2 m di distanza
- Consentono di **mostrare** il contenuto aumentato in punti specifici dello **spazio fisico**
 - Creano punti di riferimento per l'applicazione



Cosa rende buono un target?

- I **target** sono valutati sulla base delle **feature** che possono essere **estratte** dall'immagine
- Una feature è un **dettaglio** nitido e ben **visibile** nell'immagine, come un **angolo** o uno spigolo



Cosa rende buono un target?

- La distribuzione del **colore** è importante, più gli **spigoli** e i **bordi** sono visibili meglio è



★ ★ ★ ★ ★



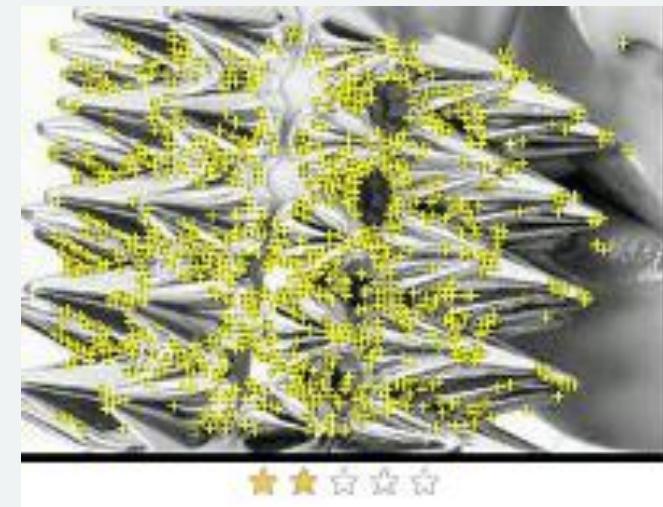
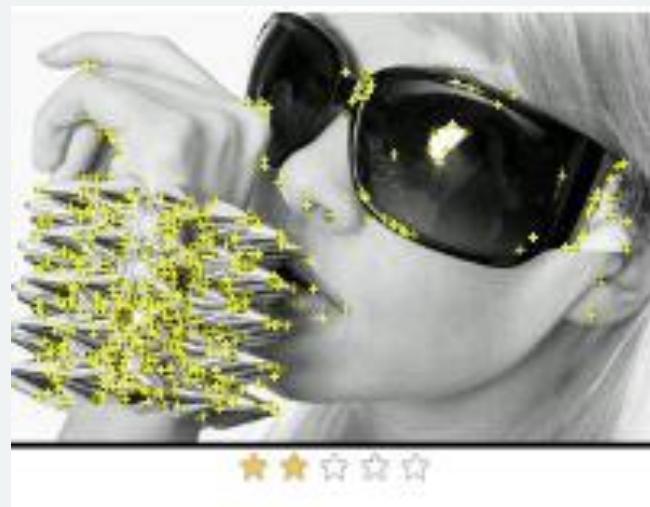
★ ★ ★ ★ ★



★ ★ ★ ★ ★

Cosa rende buono un target?

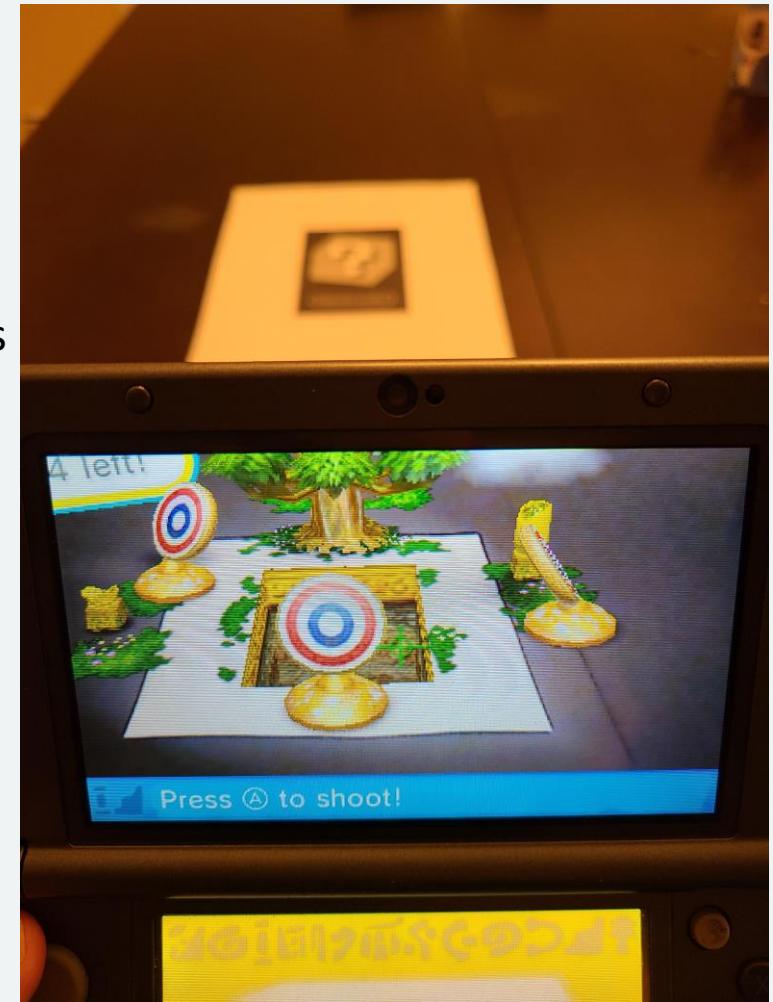
- La **distribuzione** delle feature è importante, più sono distribuite in modo **uniforme** meglio è



AR games che si basano su marker

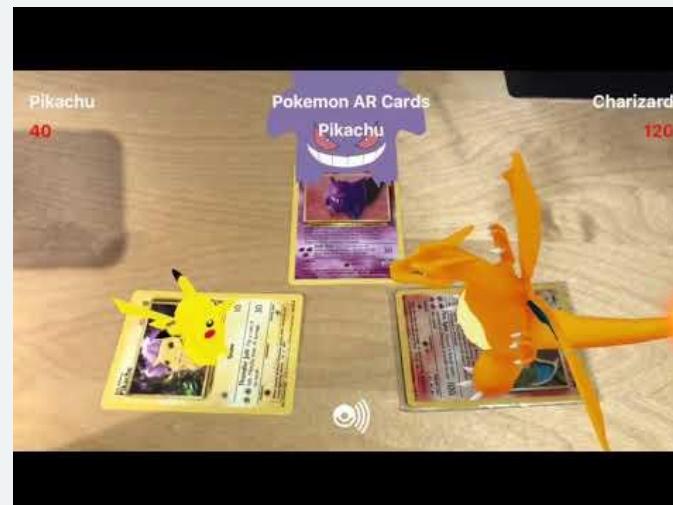


Genesis



Nintendo AR Cards

Pokemon AR Cards



Quali sono gli aspetti da considerare per creare un'applicazione di realtà aumentata?

1

Le **differenze** fra i diversi **dispositivi**

2

Se si tratta di un gioco **multi-player** o **singolo**

3

Come l'utente può/deve **interagire** con lo **spazio** e gli **altri utenti**

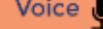
Differenze fra i dispositivi

I dispositivi AR possono essere Tablet, Smartphone e Smartglasses che consentano la **visione** del mondo esterno.

Che possono **distinguersi** per

1. Capacità **Hardware** e di rendering differenti
2. **Sistema operativo** (IoS, Android, Windows)
3. Interfaccia di **interazione** (pulsanti, mani, schermo touch)



Augmented Reality User Interaction Types	
Explicit Interactions	Implicit Interactions
Hand gestures 	Voice 
Hand controllers 	Head/device Movement  Touch screen 

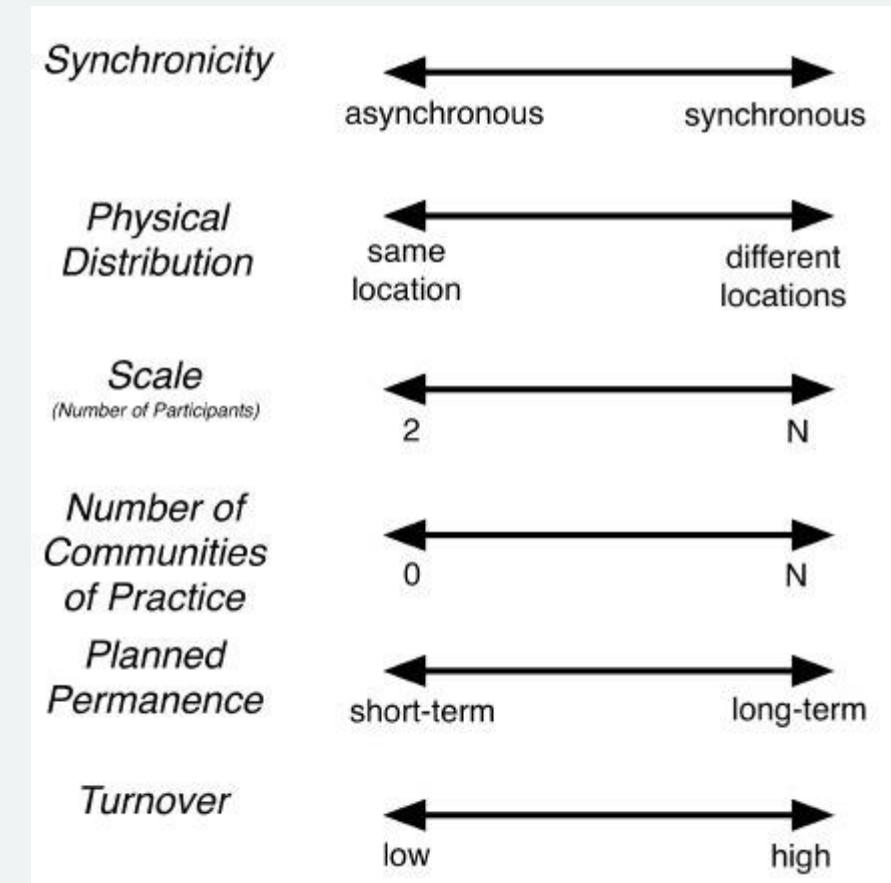
Numero di utenti coinvolti

Quando abbiamo a che fare con più utenti, dobbiamo scontrarci con un **sistema distribuito** e tutti i problemi che ne conseguono.

L'applicazione diventa **collaborativa** e occorre considerare i seguenti aspetti:

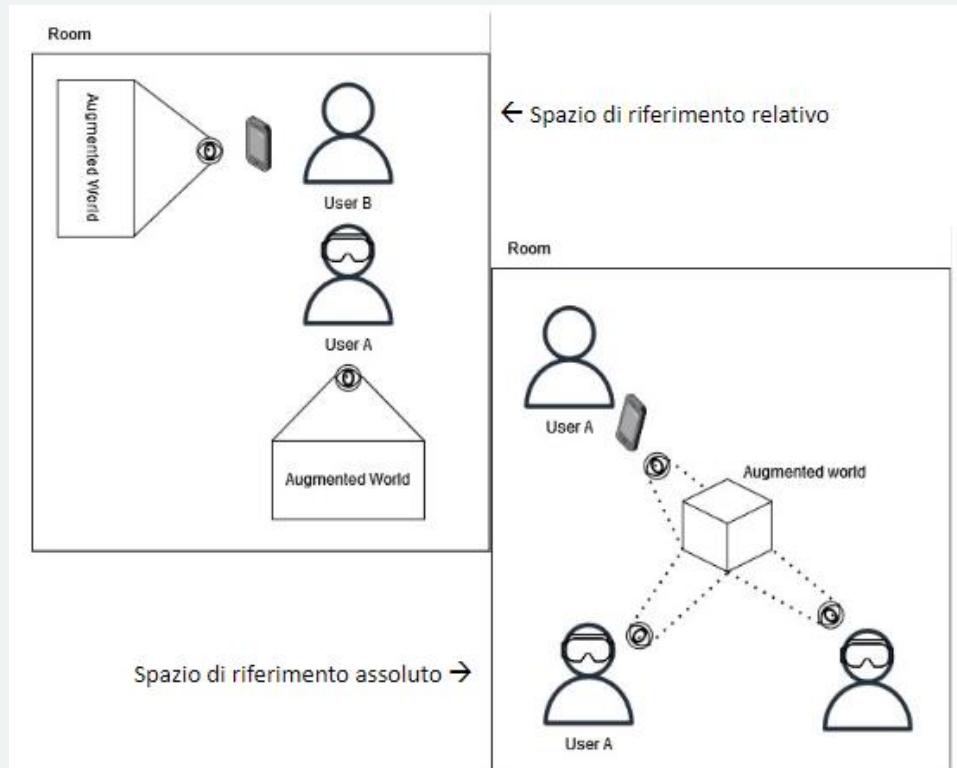
- Sincronia
- Distribuzione fisica
- Numero di partecipanti
- Permanenza pianificata
- Turnover

Infine occorre stabilire l'**infrastruttura di rete**

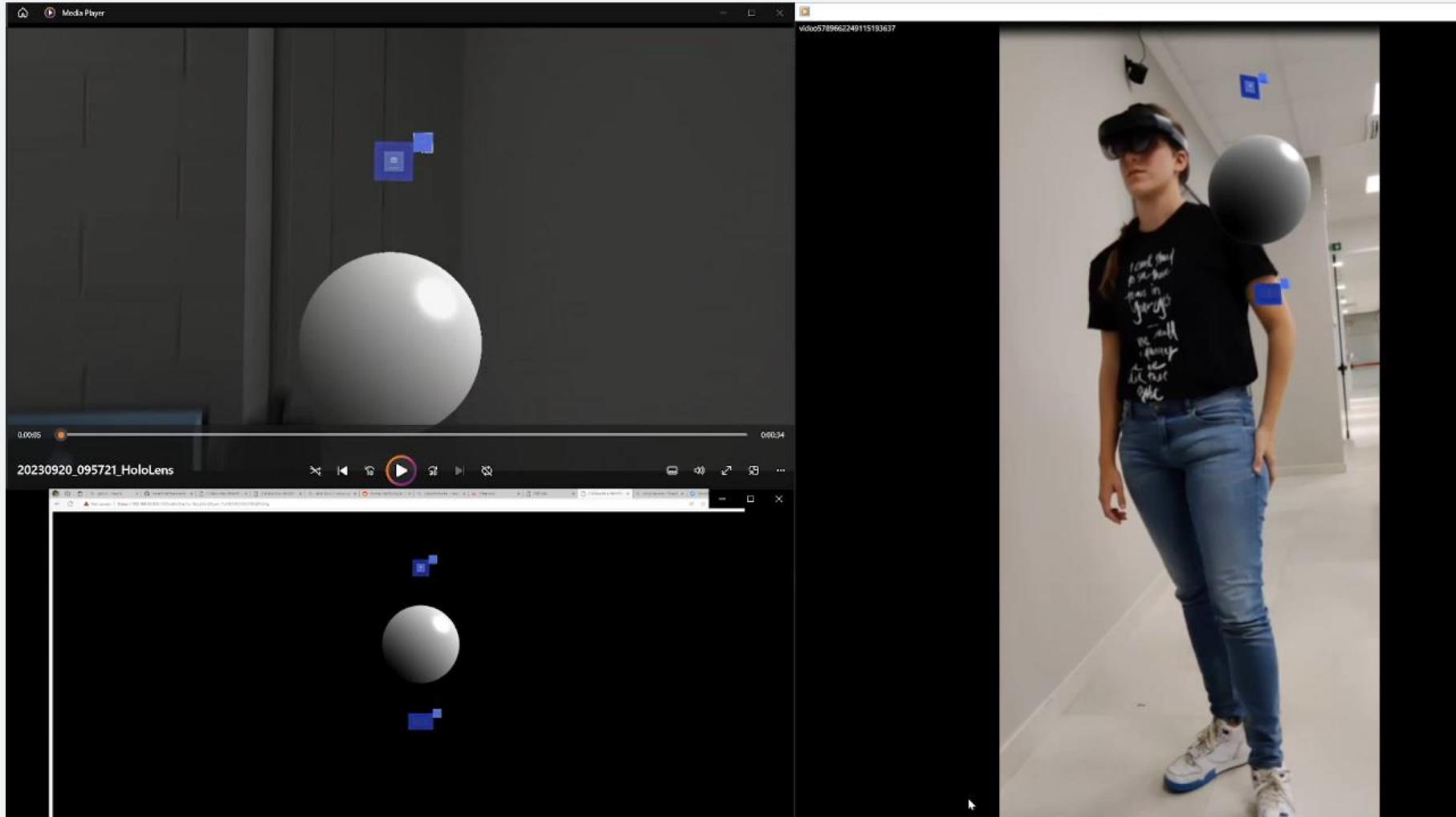


Spazio di riferimento condiviso

- Si definisce *3D Collaborative Virtual Environment*, un ambiente in cui **tutti i partecipanti** osservano lo stesso **spazio 3D** e vedono gli oggetti con la stessa **dimensione, posizione relativa e orientazione**
- Gli utenti che partecipano **all'esperienza** devono possedere uno **stesso sistema di riferimento**, che può essere:
 - Assoluto
 - Relativo



Demo



Tutto bellissimo ma... come si realizza un'applicazione di AR?



Partiamo con la scelta degli strumenti

- Esistono **ambienti di sviluppo e framework** pensati apposta per creare applicazioni di Augmented Reality, i quali nascono in principio per lo sviluppo di videogiochi....
- Negli ultimi anni si è fatto un passo avanti, passando allo sviluppo di applicazioni AR basate sul web il che rende **automaticamente gestite** alcune differenze fra i dispositivi

Web Based Application	Device Based Application
Babylon.js	Unity
Tree.js	Unreal Engine
WebXR	OpenXR

Che cos'è Babylon.js?

Babylon.js è un **framework javascript** basato su WebGL

- + Gestisce in **autonomia** le differenze fra i diversi dispositivi,
- + Offre un ambiente di **Playground** dove è possibile scrivere ed eseguire dei **test**
- + Il programma realizzato può essere **integrato** in un'applicazione Web

- Occorre progettare **l'infrastruttura di rete** per usarlo correttamente
- Occorre conoscere il linguaggio **Javascript**



Che cos'è Unity?

Unity è un **motore grafico** multi-piattaforma

- + Presenta una **Special GUI**, apposta per la realizzazione di applicazioni,
- + **Non** serve necessariamente scrivere degli **scripts** per ottenere i risultati voluti
- Cambia molto velocemente per addattarsi alle esigenze di mercato



Sviluppare applicazioni AR librerie dedicate

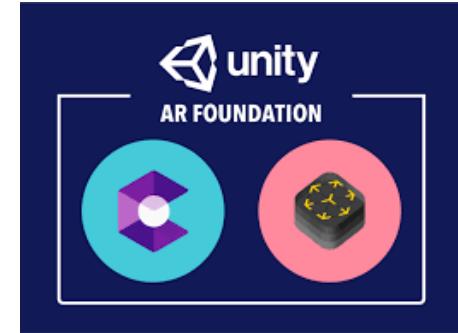
Per sviluppare applicazioni AR le seguenti **librerie** possono aiutare

1. Vuforia

- Software Development Kit (SDK)
- Utilizza la **Computer Vision** per riconoscere e tracciare immagini e oggetti 3D nel mondo, in tempo reale

2. AR Foundation

- Software Development Kit (SDK)
- Consente la creazione di applicazioni AR **multi-piattaforma**
- Adotta funzionalità **di Computer Vision** per Image, Object, Face e Body tracking



Iniziamo!

Realizzazione di un'applicazione Web Based e Device based

Obiettivi

- Capire gli aspetti base degli elementi di una scena in Babylon.js
- Utilizzare un target da mantenere tracciato durante l'esecuzione
- Mostrare un contenuto olografico quando un target viene inquadrato e se questo si sposta
- Creazione di una GUI per muovere l'ogramma

• Installare node.js, npm e babylon.js

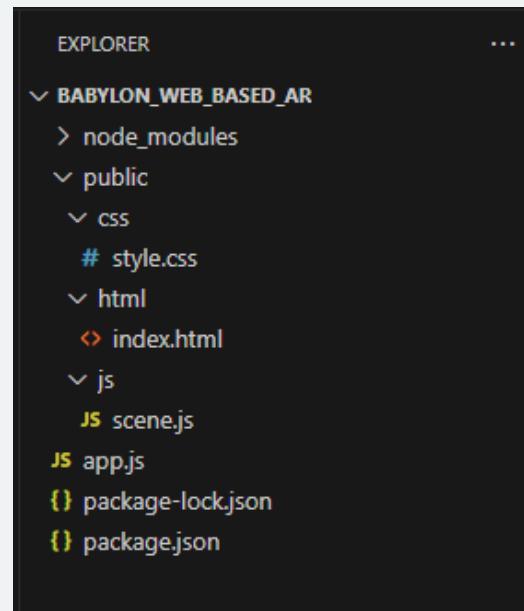
- Installare node.js e npm: <https://nodejs.org/en/download/package-manager>
- Installare babylon.js: <https://doc.babylonjs.com/guidedLearning/createAGame/gettingSetUp>

Impostare il progetto

- Creare la cartella di progetto e lanciare i seguenti comandi

```
npm init  
npm install --save-dev @babylonjs/core  
npm install --save-dev @babylonjs/inspector  
  
npm install express
```

- Creare i seguenti files e directories



Contentuo del file style.css

```
body, html, #renderCanvas {  
    height:100%;  
    width: 100%;  
}
```

Content of the index.html file

```
<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">
    <title>Game As a Lab AR</title>
    <script src="https://cdn.babylonjs.com/babylon.js"></script>
    <script src="https://preview.babylonjs.com/gui/babylon.gui.min.js"></script>
    <script
src="https://preview.babylonjs.com/loaders/babylonjs.loaders.min.js"></script>
        <link rel="stylesheet" type="text/css" href="../css/style.css">
</head>
<body>
    <canvas id="renderCanvas"></canvas>
    <script type="module" src="../js/scene.js"></script>
</body>
</html>
```

Creazione di un certificato

- Dato che WebXR consente l'esecuzione **solo su pagine Web sicure (https) o localhost**, per lanciare l'applicazione on un **IP address specifico**, occorre associare un **certificato** alla pagina, che può essere creato così:

```
openssl genrsa -out private_key.pem  
openssl req -new -key private_key.pem -out csr.pem  
openssl x509 -req -days 9999 -in csr.pem -signkey private_key.pem -out cert.pem
```

Contenuto del file app.js

```
const https = require("https");
const fs = require("fs");
const express = require("express");
var path = require("path");
const app = express();
const port = 3000;
const host = 'localhost' // webXR works only on https connection, insert your IP address here instead of localhost

https.createServer(
{
  key: fs.readFileSync("private_key.pem"),
  cert: fs.readFileSync("cert.pem"),
}
, app)
.listen(port, host, () => {
  console.log('Server started at https://'+ host + ':' + port);
});

app.use(express.static(path.join(__dirname, '/public')));

app.get("/", function (req, res) {
  res.sendFile(path.join(__dirname, 'public/html/index.html'));
})
```

Impostazione della scena

- Per prima cosa occorre **inizializzare** Babylon.js e impostare come elemento in cui si visualizzerà la scena il **canvas** presente nella pagina

```
const canvas = document.getElementById('renderCanvas');
const engine = new BABYLON.Engine(canvas, true);
```

- Occorre impostare la **funzione** che gestirà il **contenuto** della scena, nonché il suo **rendering**

```
const createScene = function(){
    //TODO: Create a scene
}

createScene().then(sceneToRender => {
    engine.runRenderLoop(function(){
        sceneToRender.render();
    });
});
```

Impostazione della scena

- Aggiungiamo gli elementi di base, una fonte di **luce** e una **camera**

```
const createScene = function(){
    const scene = new BABYLON.Scene(engine);
    scene.clearColor = new BABYLON.Color3.Black;

    // start point for light (1, 1, 0)
    const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(1, 1, 0), scene)
    light.intensity = 1

    const alpha = 3 * Math.PI/2;
    const beta = Math.PI/50;
    const radius = 220;
    const target = new BABYLON.Vector3(0, 0, 0); // (0, 0 ,0) is the user position

    const camera = new BABYLON.ArcRotateCamera("Camera", alpha, beta, radius, target, scene);
    camera.attachControl(canvas, true);

    return scene
}
```

Creazione di una sessione XR

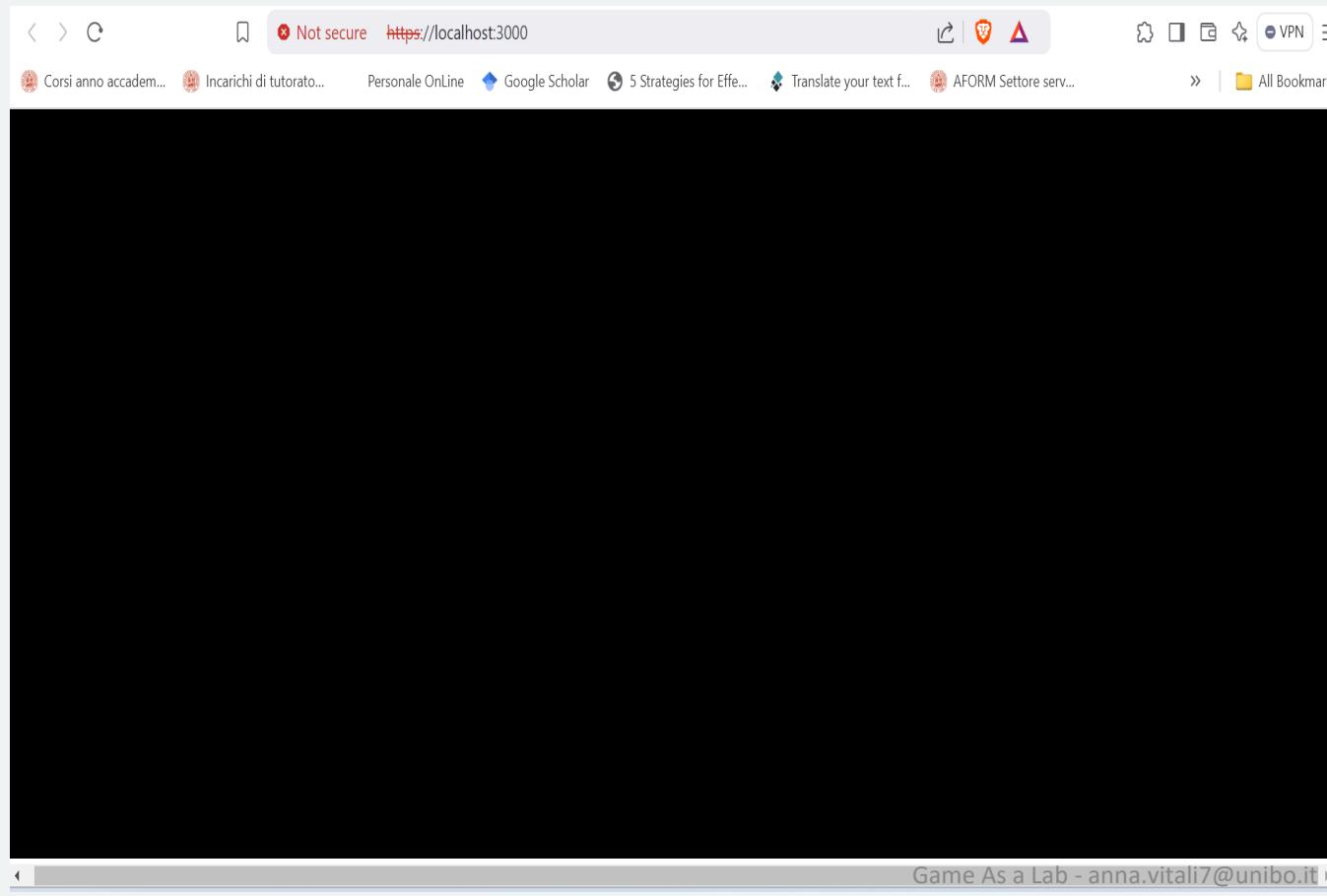
- Aggiungiamo gli elementi di base, una fonte di **luce** e una **camera**

```
const createScene = function(){
    //codice scritto prima...
    const supported = await BABYLON.WebXRSessionManager.
        IsSessionSupportedAsync('immersive-ar')
    let xrHelper

    if (supported) {
        console.log("IMMERSIVE AR SUPPORTED");
        xrHelper = await scene.createDefaultXRExperienceAsync({
            uiOptions: {
                sessionMode: 'immersive-ar',
                referenceSpaceType: "local-floor"
            }
        });
    }
    return scene
}
```

Creazione sessione XR

- Eseguiamo l'applicazione tramite il comando: `node ./app.js` eseguito nella cartella di progetto



Creazione del featuresManager

- Il tracciamento di un'immagine è riconosciuto come una feature da Babylon.js quindi occorre impostare il **FeaturesManager**

```
const createScene = async function(){
    //codice precedente...

    if(xrHelper !== undefined){
        const featuresManager = xrHelper.baseExperience.featuresManager;
        enableFeatures(featuresManager);
    }

    return scene;
}

const enableFeatures = function (featuresManager){
    //TODO
}
```

Image Tracking Feature

- Occorre impostare **l'immagine** che vogliamo tracciare
- Quando specifichiamo il target occorre inserire anche le **dimensioni** nel nostro caso: **0.1** che equivale a **10 cm**



Image Tracking Feature

- Nella nostra cartella di progetto inseriamo l'immagine in un'apposita **directory** e la referenziamo nel codice

```
const enableFeatureImageTracking = function (featuresManager, hologram){  
    try {  
        const imageTracking = featuresManager.enableFeature(BABYLON.WebXRFeatureName.IMAGE_TRACKING, "latest",  
    {  
        images: [  
            {  
                src: "../img/image_target.png",  
                estimatedRealWorldWidth: 0.2  
            },  
        ]  
    });  
  
    //TODO  
  
    }catch(error){  
        console.log("Image tracking not supported in this browser or device.");  
        console.log(error)  
    }  
}
```

Creiamo il contenuto da mostrare

- Associamo al target il contenuto da mostrare quando viene **inquadrato**
- Babylon mette a disposizione alcune forme di base di oggetti, prendiamo il **cubo**

```
const createBoxHologram = function (scene, position, size){  
    var cube = BABYLON.MeshBuilder.CreateBox("cube", {size: size}, scene);  
    cube.position = position;  
  
    const material = new BABYLON.StandardMaterial("material", scene);  
    material.diffuseColor = BABYLON.Color3.White();  
  
    cube.material = material;  
  
    return cube  
}
```

Mostriamo l'ologramma

- Quando abbiamo a che fare con il **target** dobbiamo definire **due funzioni**
 1. La prima ci dice se l'immagine **può** essere tracciata dal sistema
 2. La seconda ci consente di mostrare gli ologrammi **quando** il target viene **inquadrato**

```
imageTarget.onUntrackableImageFoundObservable.add((image) => { //TODO});  
imageTarget.onTrackedImageUpdatedObservable.add((image) => { //TODO});
```

Mostriamo l'ologramma

- ```
const enableFeatureImageTracking = function (featuresManager, hologram){
 try {

 //codice scritto prima...

 imageTracking.onUntrackableImageFoundObservable.add((image) => {
 console.log("image not found :(")
 });

 imageTracking.onTrackedImageUpdatedObservable.add((image) => {
 image.transformationMatrix.decompose(hologram.scaling,
hologram.rotationQuaternion, hologram.position);
 });

 }catch(error){
 console.log("Image tracking not supported in this browser or device.");
 console.log(error)
 }
}
```

# Mostriamo l'ologramma

- Per far sì che l'ologramma **segua l'immagine** occorre associare a cube gli stessi valori di posizione, scala e rotazione che possiede il target

```
imageTarget.onTrackedImageUpdatedObservable.add((image) =>{
 image.transformationMatrix.decompose(hologram.scaling,
 hologram.rotationQuaternion,hologram.position);
});
```

# Creazione GUI

- Creiamo una funzione per creare i nostri pulsanti di movimento

```
const createButton = function(name, text,
width, height, color, background, cornerRadius,
top, left, onClick) {
 var button =
BABYLON.GUI.Button.CreateSimpleButton(name,
text);
 button.width = width;
 button.height = height;
 button.color = color;
 button.cornerRadius = cornerRadius;
 button.background = background;
 button.top = top;
 button.left = left;
 button.onPointerUpObservable.add(onClick);
 return button;
}
```

# Creazione GUI

- Creiamo **quattro pulsanti** che ci consentono di spostare il cubo

```
const createPositioningGUI = function (hologram) {
 var advancedTexture =
BABYLON.GUI.AdvancedDynamicTexture.CreateFullscreenUI("PositioningGUI");

 const width = "200px";
 const height = "200px";
 const color = "white";
 const background = "gray";
 const cornerRadius = 20;

 var up = createButton("up", "UP", width, height, color, background, cornerRadius, 700,
250, function() {
 hologram.position.y += 0.1;

 });
 advancedTexture.addControl(up); //continua nella slide successiva...
```

# Creazione GUI

```
var down = createButton("down", "DOWN", width, height, color, background, cornerRadius,
950, 250, function() {
 hologram.position.y -= 0.1;
});
advancedTexture.addControl(down);

var left = createButton("left", "LEFT", width, height, color, background, cornerRadius,
850, 50, function() {
 hologram.position.x -= 0.1;
});
advancedTexture.addControl(left);

var right = createButton("right", "RIGHT", width, height, color, background,
cornerRadius, 850, 450, function() {
 hologram.position.x += 0.1;
});
advancedTexture.addControl(right);
}
```

# Risultato



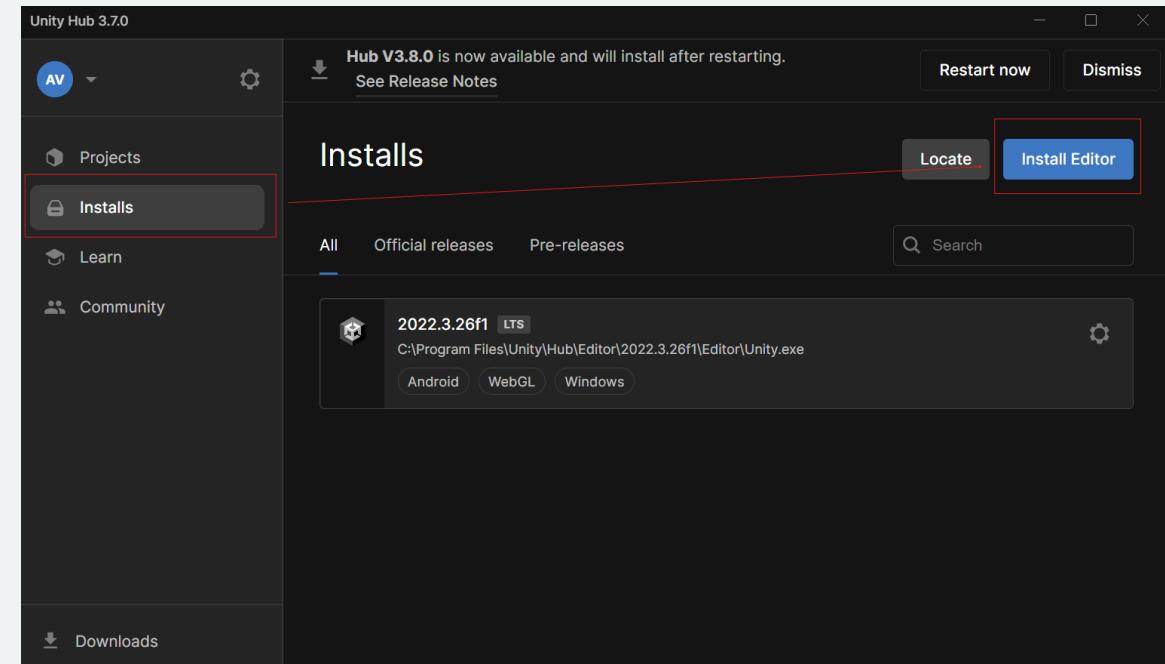
# Obiettivi

- Capire gli aspetti base degli elementi in Unity
- Utilizzare Vuforia e il portale che mette a disposizione per gli Sviluppatori
- Creare un proprio target
- Mostrare un contenuto olografico quando un target viene inquadrato

# • Installiamo Unity e relativi package

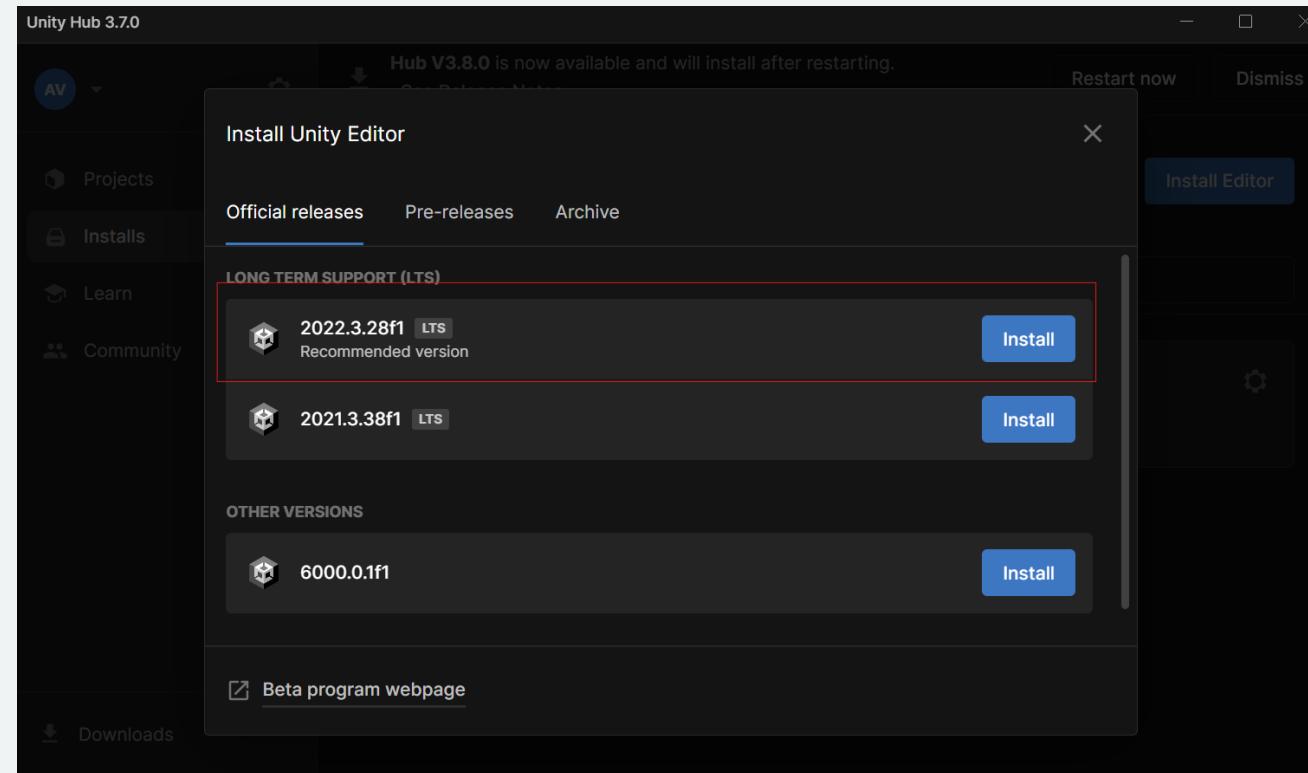
- Collegarvi alla pagina ufficiale:  
<https://unity.com/download>
- Cliccare sul pulsante Download
- Una volta installato l'Hub, apritelo tramite l'apposita icona  

- Cercate la voce installs nel menu rapido e cliccate su il pulsante Install Editor



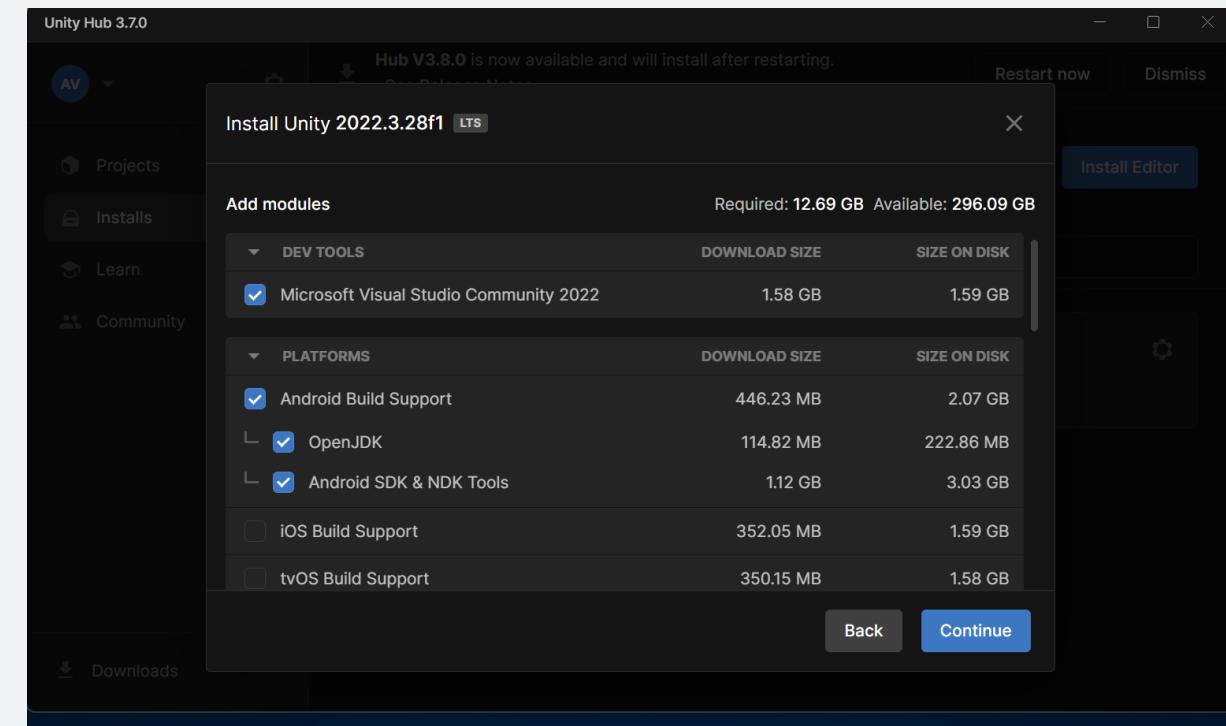
# • Installiamo Unity e relativi package

- Scegliete la versione Long Term Support (LTS) più recente, anche evidenziata come “Recommended version”



# • Installiamo Unity e relativi package

- Nel menu che si apre cliccando sul pulsante **Install Editor**, oltre a **Visual Studio Community**, se non lo avete già installato, selezionate anche **Android Build Support**, in quanto intendiamo utilizzare Unity per sviluppare Applicazioni Android (**NOTA**: i pacchetti possono essere aggiunti anche in un secondo momento sempre tramite l'Hub scegliendo la versione per cui si vogliono aggiungere tramite la schermata installs)



# Creiamo un progetto

1. Apriamo Unity Hub
2. Andiamo in Projects
3. Creiamo un nuovo progetto 3D (Built – In Render Pipeline )
4. Gli assegniamo il nome «Game\_as\_a\_lab»
5. (ci vorrà un pochino)

# Creiamo un progetto

New project  
Editor Version: 2022.3.26f1 LTS

All templates

- Core
- Sample
- Learning

Search all templates

- 2D (Built-In Render Pipeline)  
Core
- 3D (Built-In Render Pipeline)  
Core
- Universal 2D  
SRP Core
- AR Mobile  
Core
- Universal 3D  
SRP Core



3D (Built-In Render Pipeline)  
This is an empty 3D project that uses Unity's built-in renderer.

[Read more](#)

**PROJECT SETTINGS**

Project name

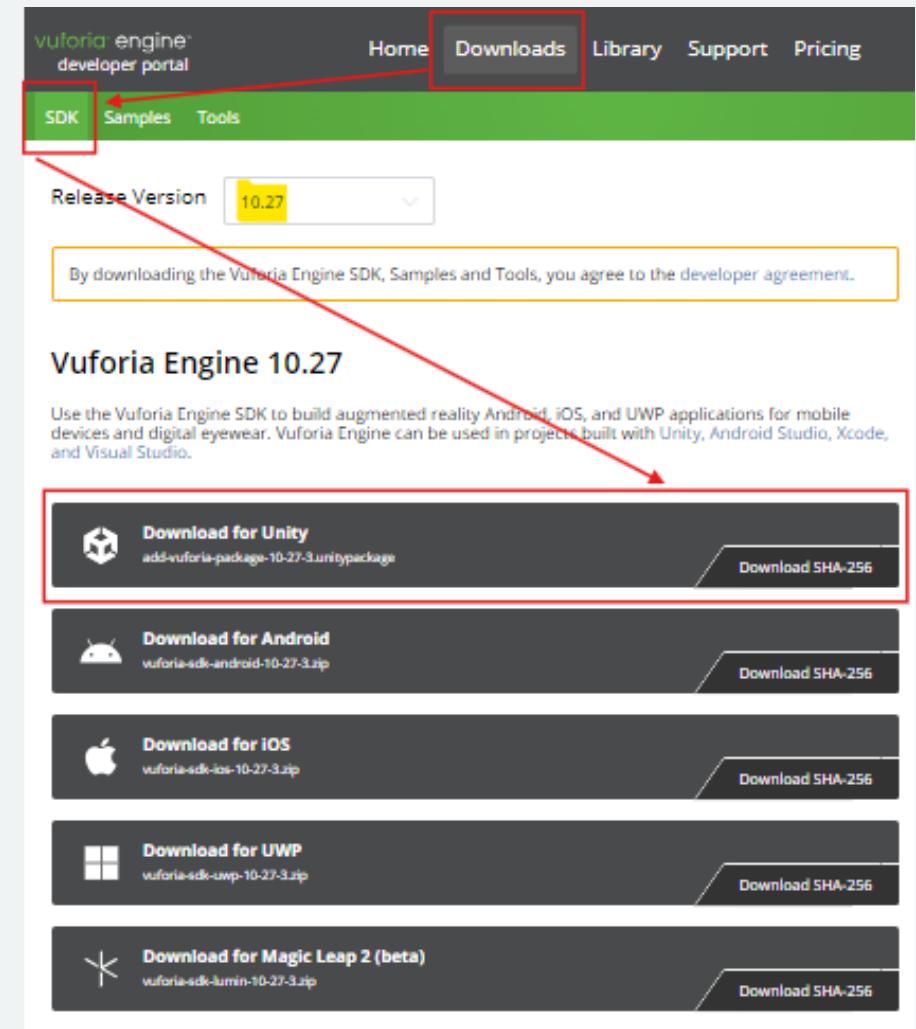
Location

[Cancel](#) [Create project](#)

Game As a Lab - anna.vitali7@unibo.it

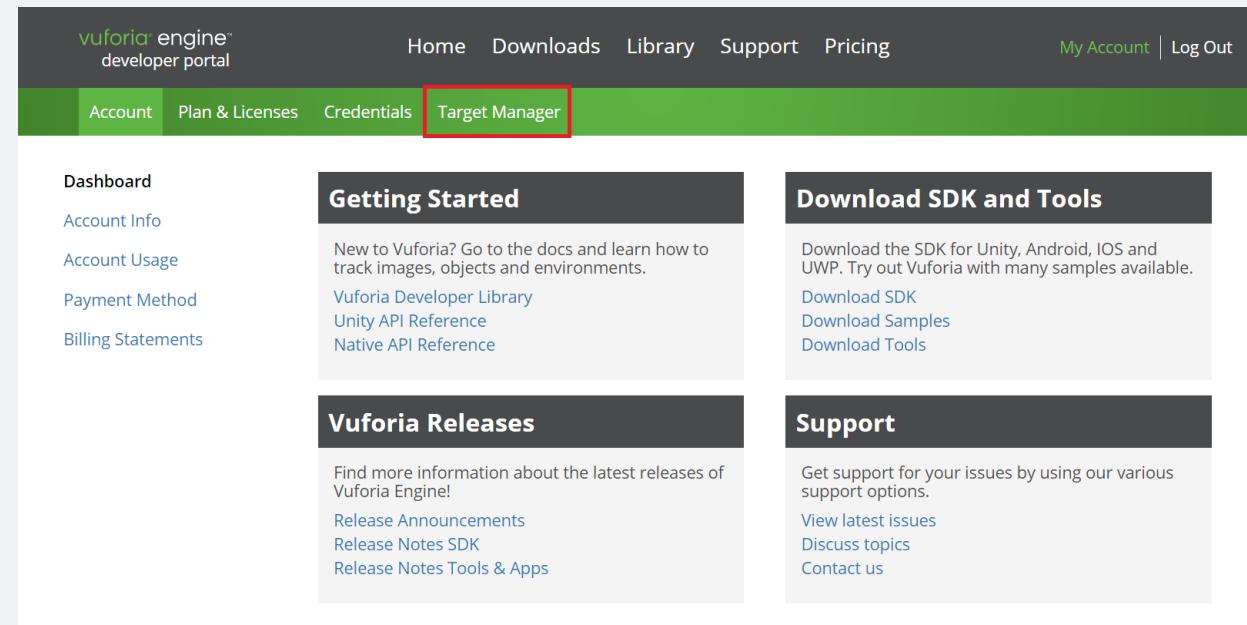
# Installiamo il Vuforia Unity Package

1. Ci registriamo al portale di Vuforia con il nostro account UNIBO
2. Ci collegiamo al Developer Portal di Vuforia:  
<https://developer.vuforia.com/downloads/sdk>
3. Andiamo su Downloads
4. Scarichiamo il pacchetto per Unity



# Crezione di un image target per vuforia

1. Sempre dal portale ci **registriamo** per avere un Account Sviluppatore:  
<https://developer.vuforia.com/auth/register>
2. Accediamo al nostro Account
3. Ci spostiamo nella sezione Target Manager e creiamo il nostro target



# Creazione di un target

1. Andiamo su Target Manager e creiamo un nuovo database

The screenshot shows the 'Target Manager' section of a software interface. At the top, there is a navigation bar with tabs: 'Account', 'Plan & Licenses', 'Credentials', and 'Target Manager'. The 'Target Manager' tab is highlighted with a red box and has a red arrow pointing from it to a green button labeled 'Generate Database', which is also highlighted with a red box. Below the navigation bar, the title 'Target Manager' is displayed in bold black text. A descriptive text below the title reads: 'Use the Target Manager to create and manage databases and targets.' To the left of this text is a search bar with the placeholder 'Search'. At the bottom of the screenshot, there is a table header with four columns: 'Database', 'Type', 'Targets', and 'Date Modified'.

| Database | Type | Targets | Date Modified |
|----------|------|---------|---------------|
|----------|------|---------|---------------|

# Creazione di un target

1. Indichiamo un nome e specifichiamo come **Type Device**

Generate Database

Database Name \*

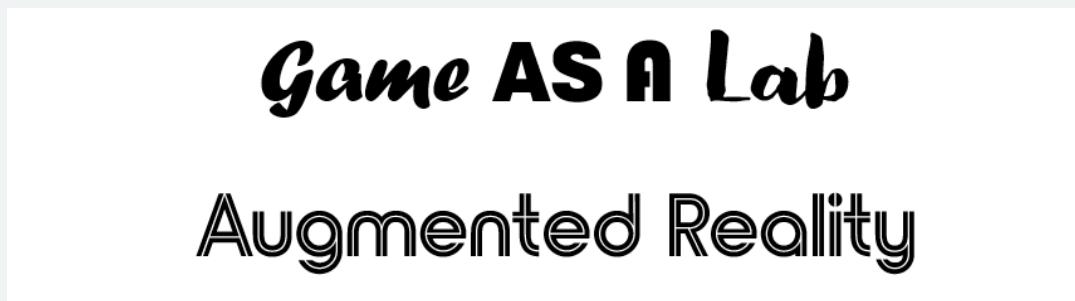
Type:

Device  
 Cloud  
 VuMark

Cancel Generate

# Creazione di un target

1. Apriamo il database e creiamo un nuovo target a partire da un'immagine
2. La nostra immagine di riferimento sarà questa:

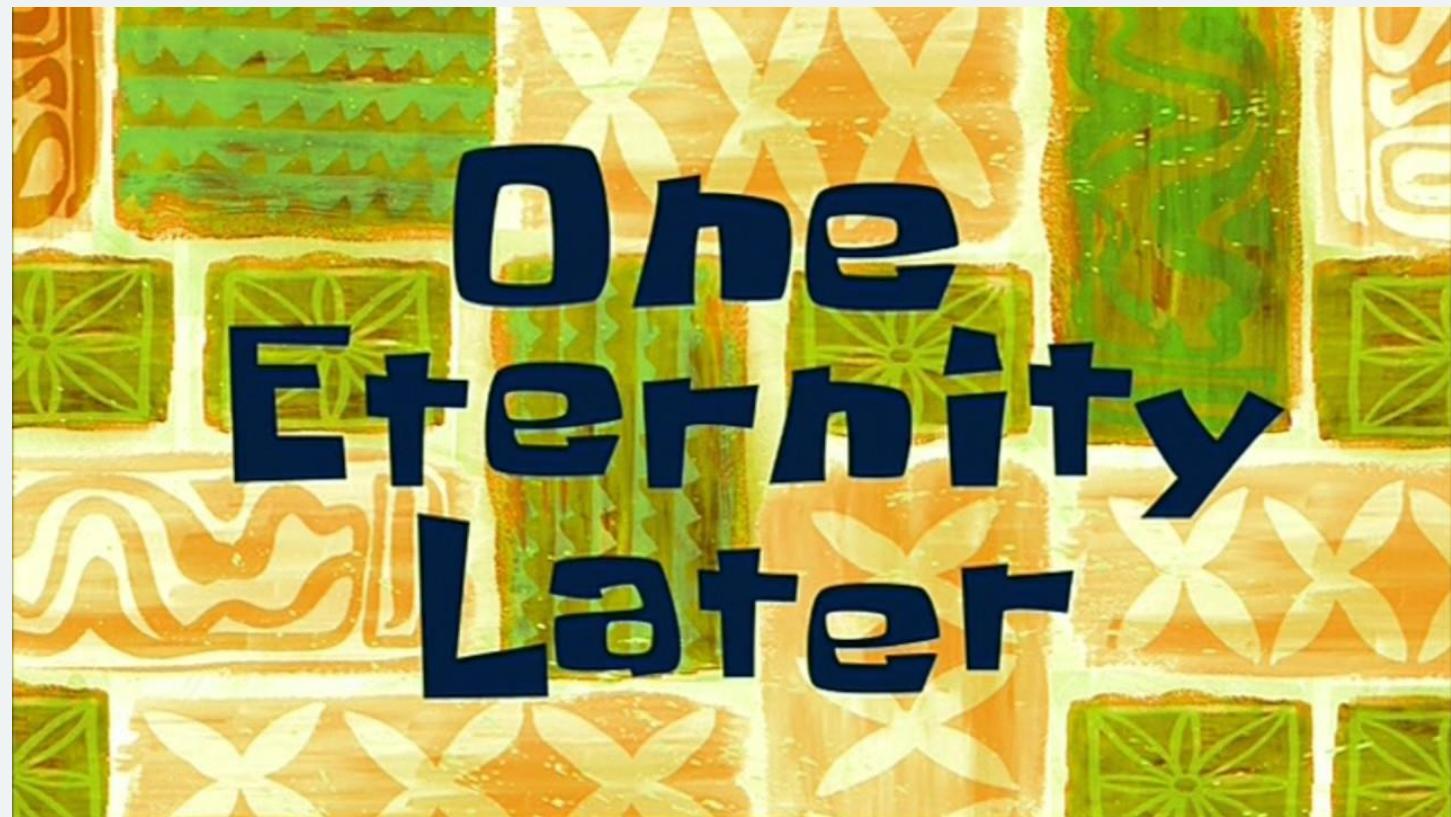


- Quando creiamo un target dobbiamo indicare la sua larghezza
- Nel nostro caso è un'immagine 21x7 (cm)
- In Unity l'unità di riferimento è il metro quindi indichiamo 0.21

**NOTA:** Quando creiamo un target Vuforia ci indica anche la qualità di questo

# Torniamo su Unity

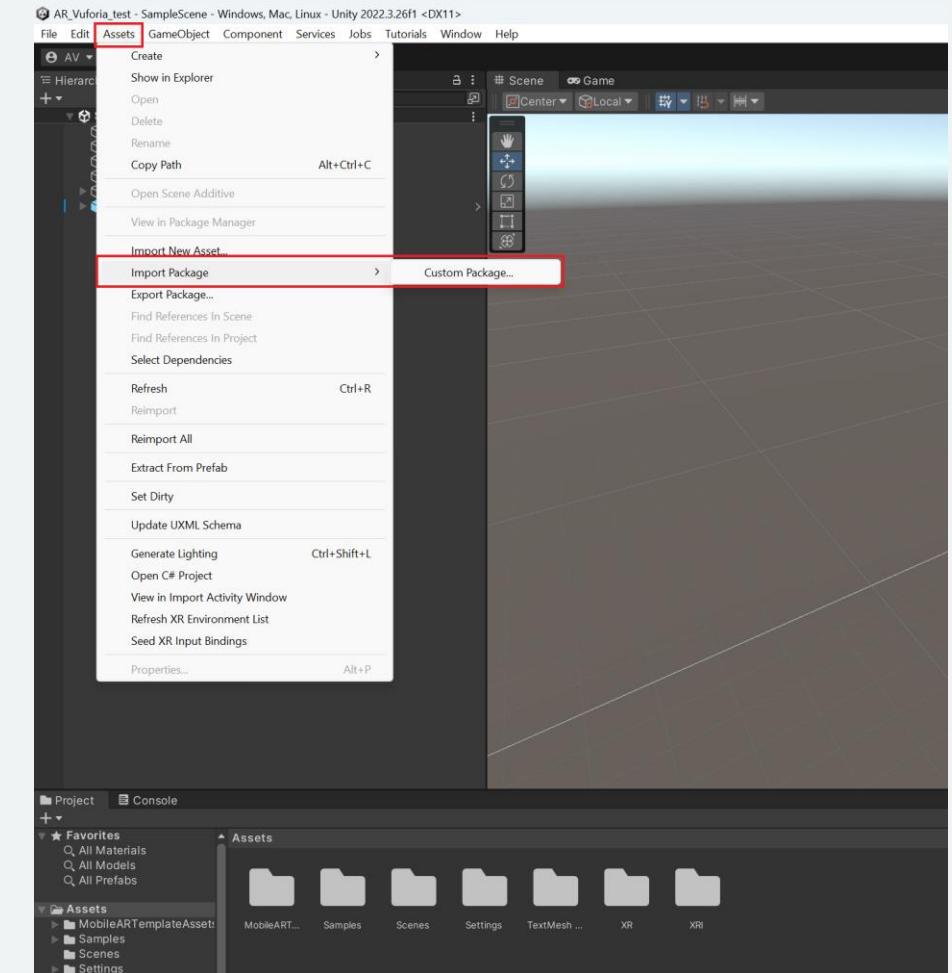
- Dopo che il nostro progetto è stato aperto e creato...



# Importiamo Vuforia in Unity

1. Ci spostiamo nell'editor
2. Clicchiamo su **Assets** poi su **Import Package** e **Custom Package**
3. Scegliamo l'**SDK** di Vuforia che abbiamo scaricato

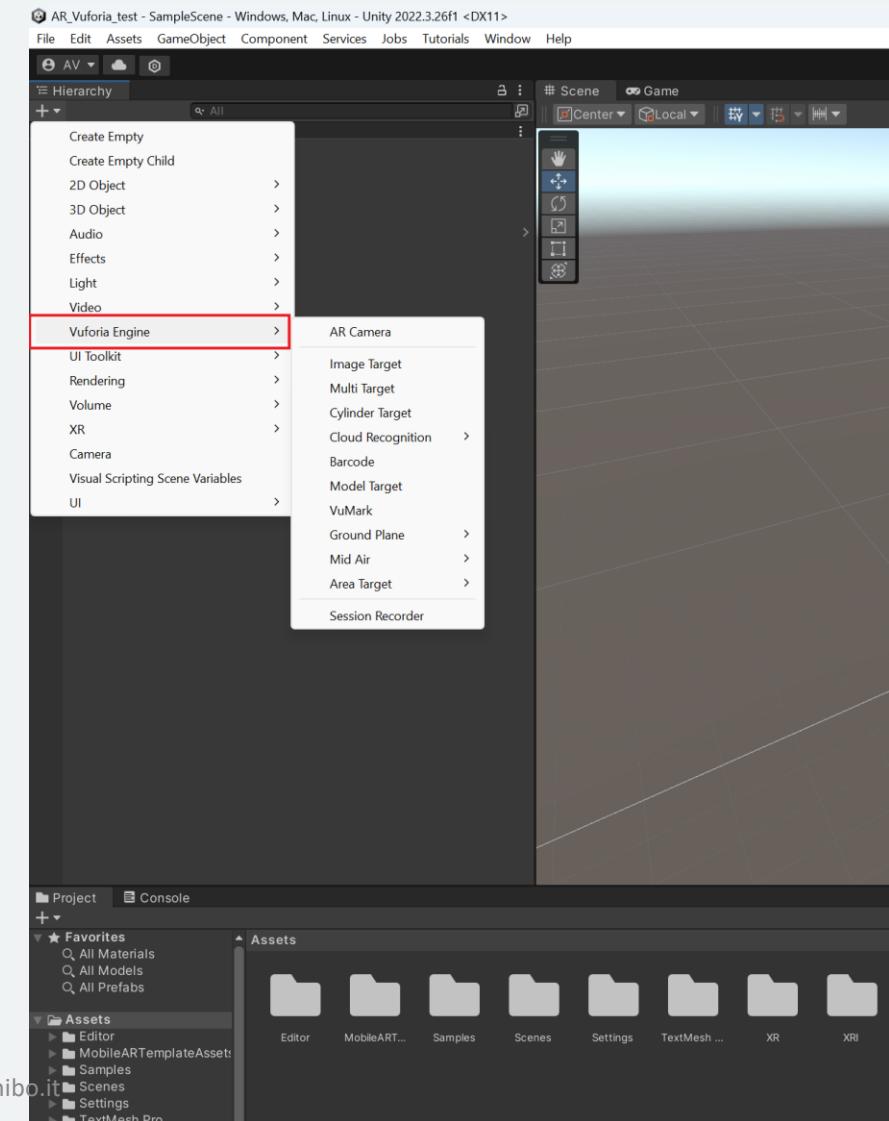
In alternativa si può fare **drag and drop** del package scaricato in Unity



# Importiamo Vuforia in Unity

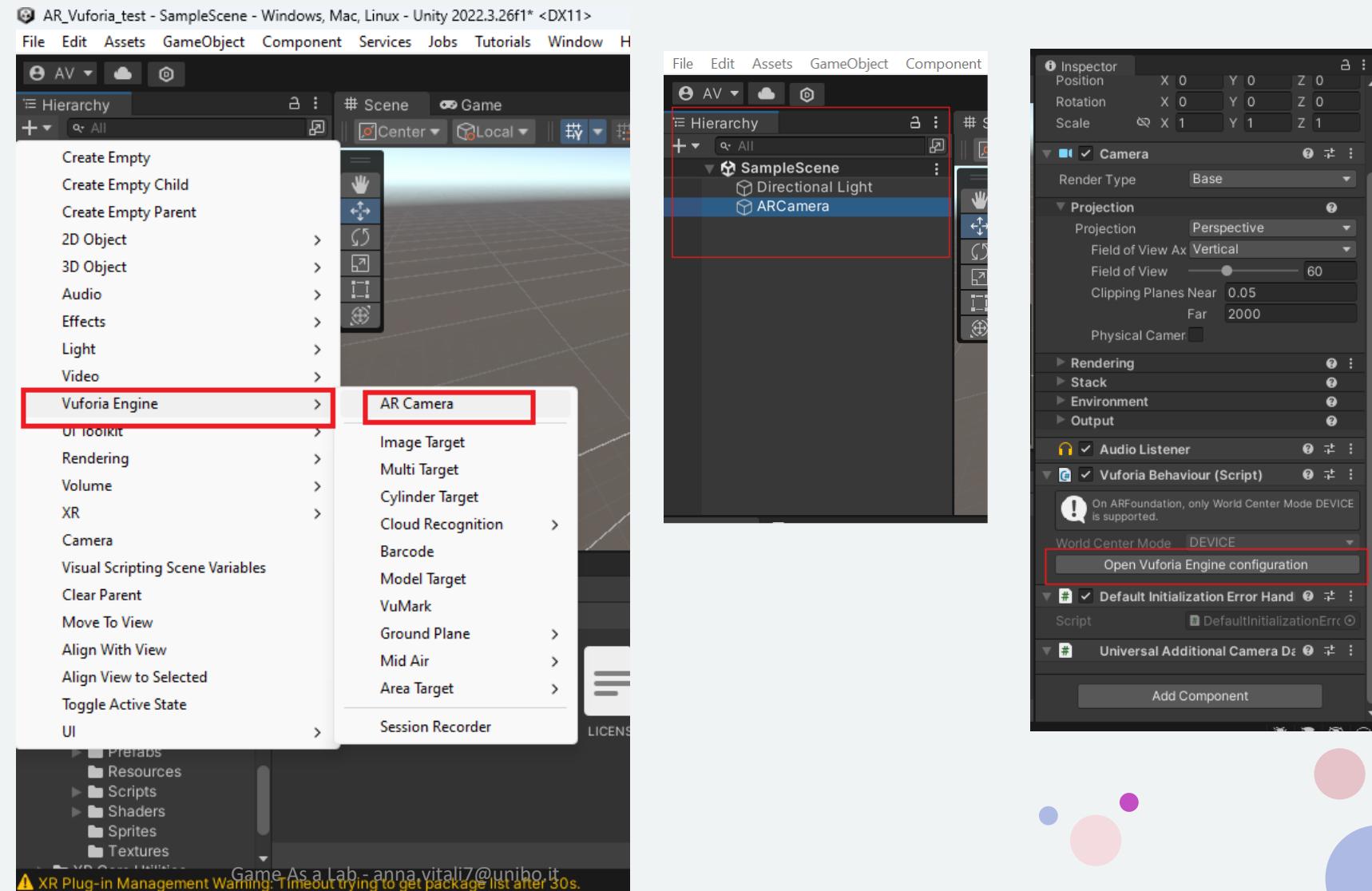
Se abbiamo fatto le cose nel modo giusto

- cliccando sul  della scena, dovremo riuscire a vedere la voce **Vuforia Engine**



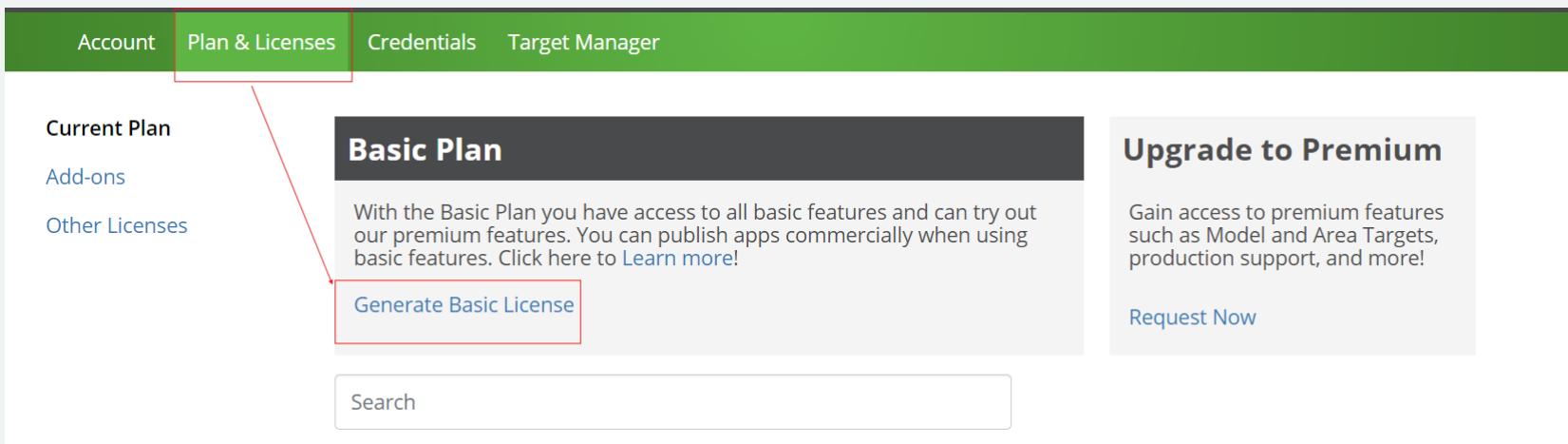
# Impostiamo la nostra API Key su Unity

1. cliccando sul **+** della scena, dovremo riuscire a vedere la voce **Vuforia Engine**
2. clicchiamo su Vuforia Engine e sulla voce **AR Camera**
3. Cancelliamo la **Main Camera**
4. Nell'Inspector clicchiamo su **Open Vuforia Engine Configurator**

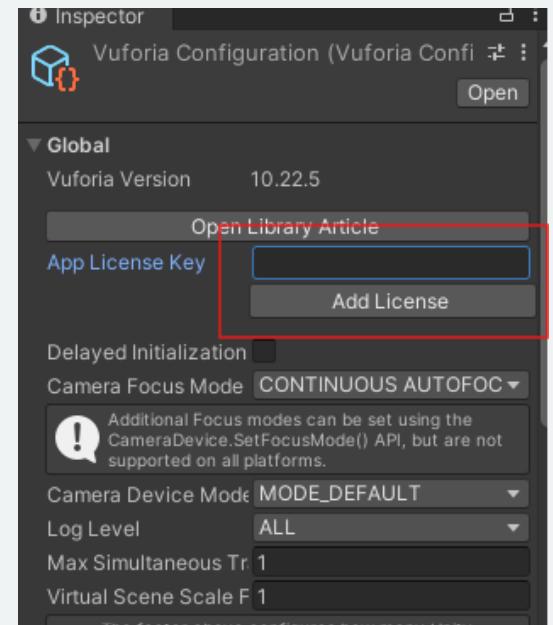


# Impostiamo la nostra API Key su Unity

4. Andiamo sul **developer portal** di Vuforia aggiungiamo la nostra licenza **basic** dall'apposita pagina
5. Inseriamo la licenza nella voce apposita dell'Inspector

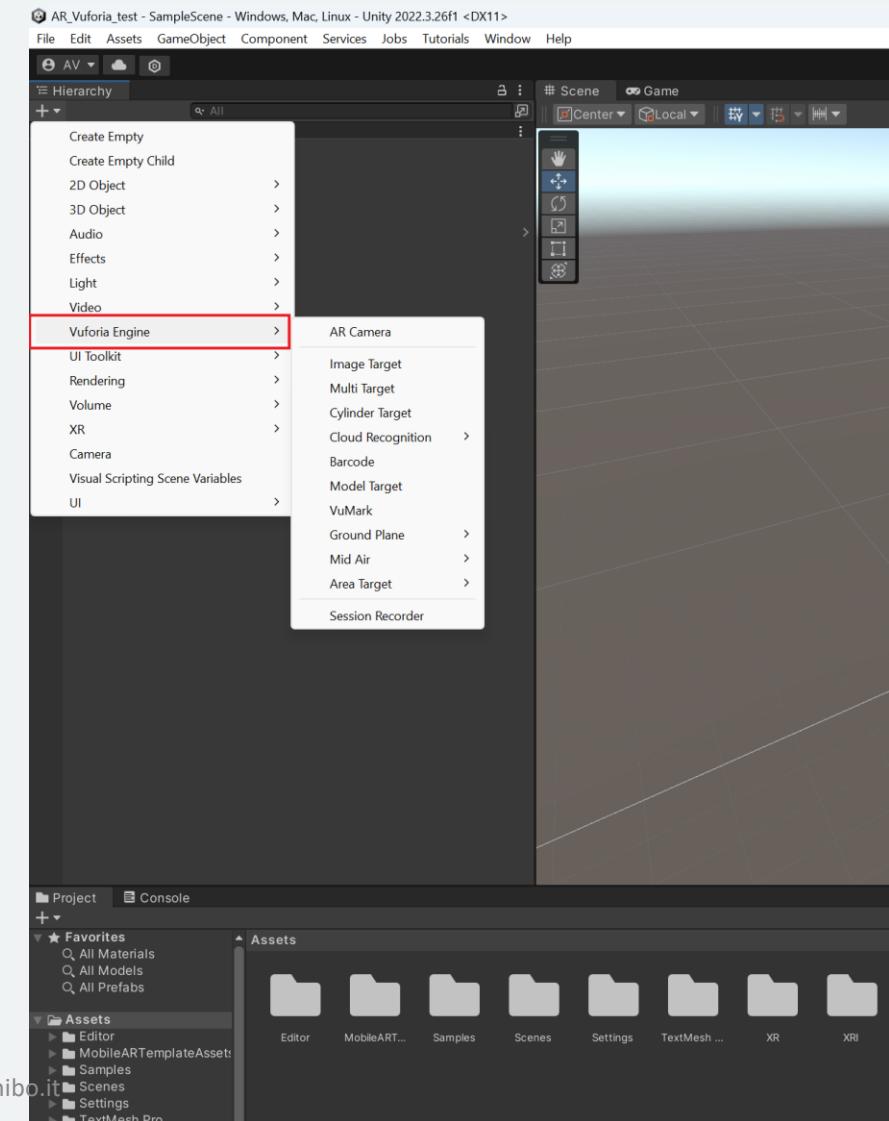


The screenshot shows the Vuforia developer portal interface. At the top, there are tabs: Account, Plan & Licenses (which is highlighted with a red box), Credentials, and Target Manager. Below the tabs, there are sections for Current Plan, Add-ons, and Other Licenses. The Current Plan section displays the "Basic Plan". It includes a description: "With the Basic Plan you have access to all basic features and can try out our premium features. You can publish apps commercially when using basic features. Click here to [Learn more!](#)". Below this description is a red-bordered button labeled "Generate Basic License". To the right of the Basic Plan section is another section titled "Upgrade to Premium" with a description: "Gain access to premium features such as Model and Area Targets, production support, and more!" and a "Request Now" button.



Se abbiamo fatto le cose nel modo giusto

- cliccando sul  della scena, dovremo riuscire a vedere la voce **Vuforia Engine**



# Usiamo il nostro image target

- Dal nostro **Target Manager** scarichiamo il database che abbiamo creato

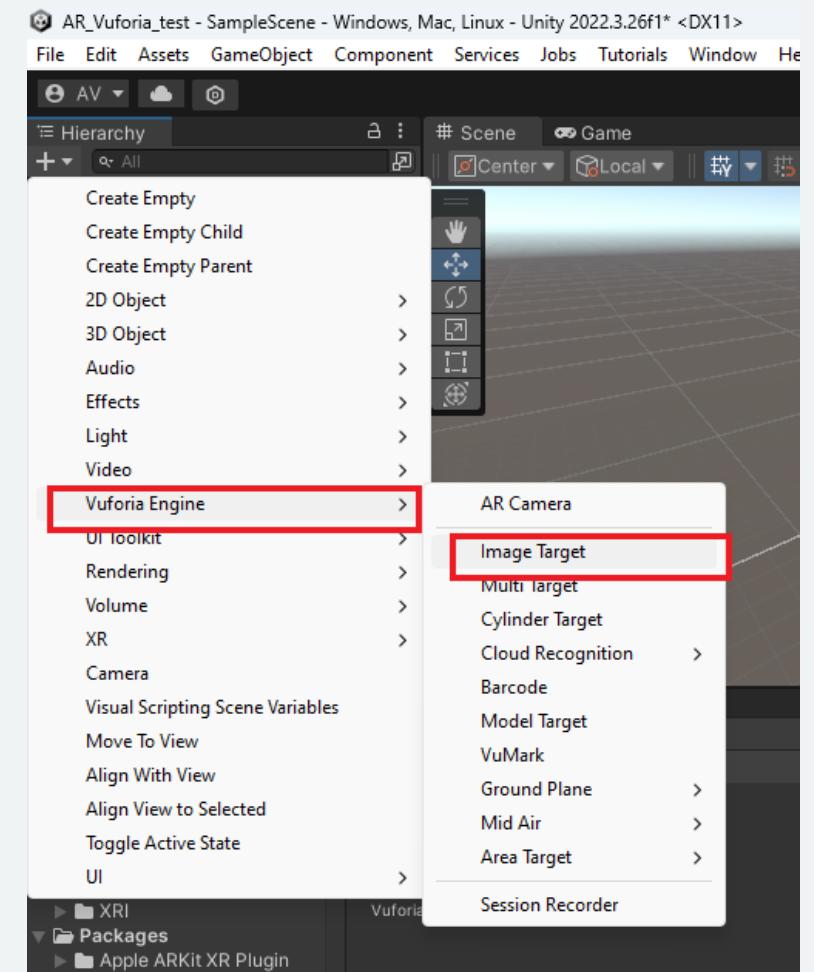
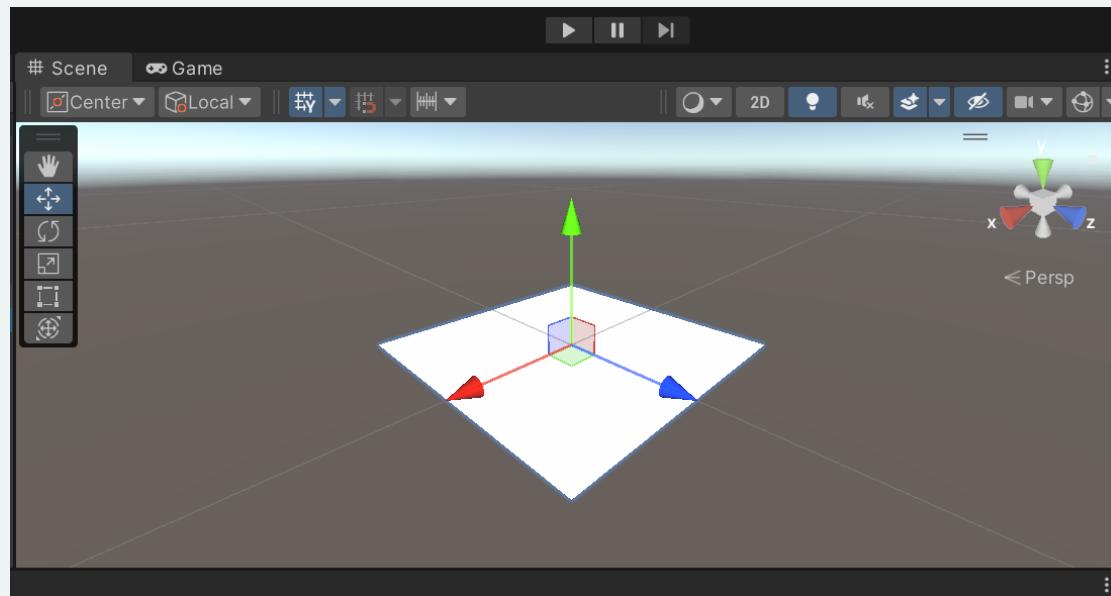
The screenshot shows the 'Target Manager' section of a software interface. At the top, there are tabs: Account, Plan & Licenses, Credentials, and Target Manager (which is highlighted with a red box). Below the tabs, the path 'Target Manager > game\_as\_a\_lab' is shown. The main area displays a target named 'game\_as\_a\_lab' with an edit link. It is categorized as 'Type: Device'. A 'Targets (1)' button is visible. On the right, a large green 'Download Database (All)' button is highlighted with a red box. Below this, a table lists the target details:

| Image | Target Name              | Type  | Rating ⓘ | Status | Date Modified |
|-------|--------------------------|-------|----------|--------|---------------|
|       | game_as_a_lab_marker_rgb | Image | ★★★★★    | Active | Nov 04, 2024  |

A 'Download Database' dialog box is shown. It states '1 of 1 active targets will be downloaded' and specifies the target name as 'game\_as\_a\_lab'. It asks to 'Select a development platform:' with two options: 'Android Studio, Xcode or Visual Studio' (unchecked) and 'Unity Editor' (checked). At the bottom are 'Cancel' and 'Download' buttons, with 'Download' highlighted with a red box.

# Usiamo il nostro image target

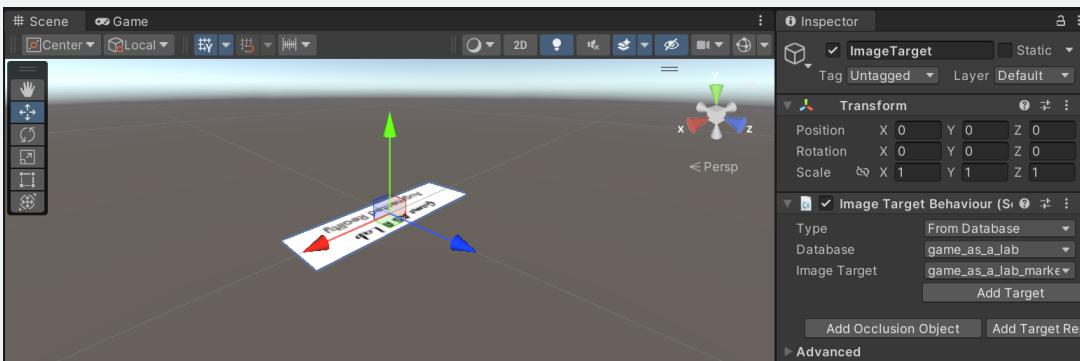
1. Su Unity clicchiamo su e dopo aver selezionato la voce Vuforia Engine, clicchiamo su **Image Target**
2. Apparirà un nuovo **elemento nella scena**, che rappresenta il nostro **target** (per ora bianco)



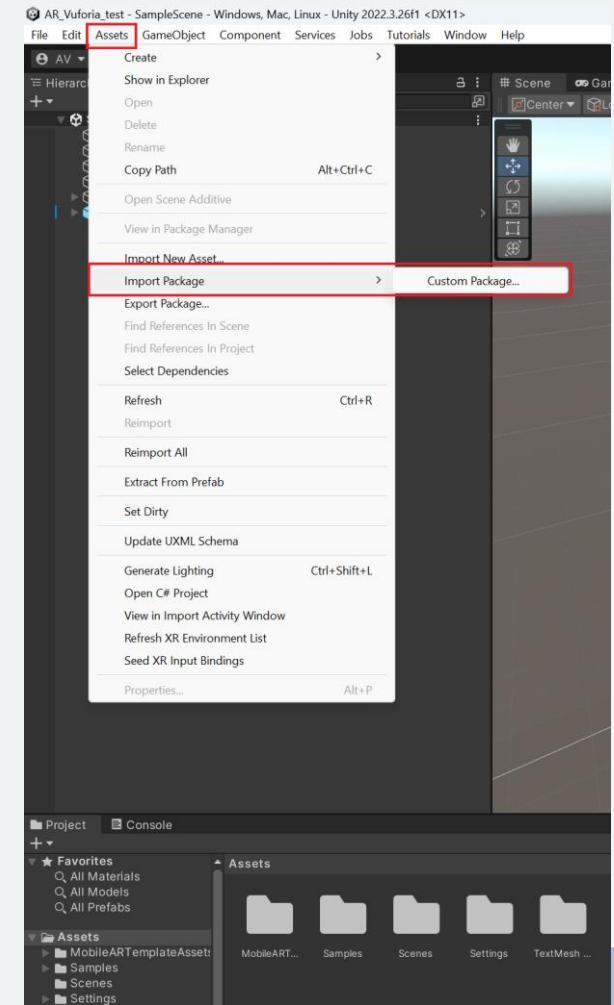
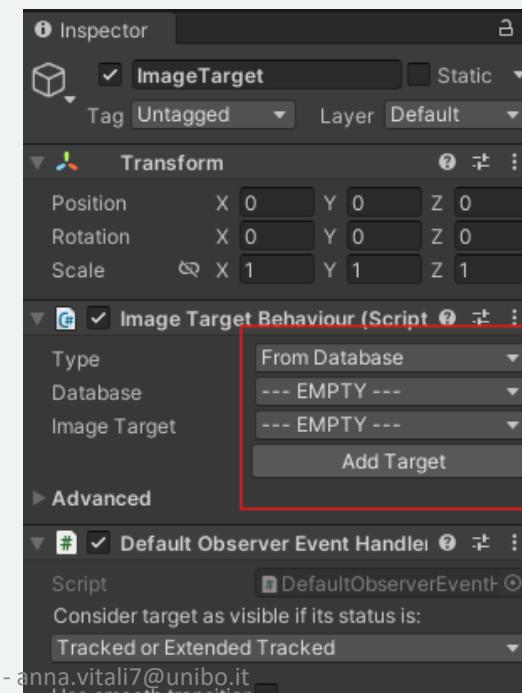
# Usiamo il nostro image target

1. Clicchiamo su **Assets** poi su **Import Package** e **Custom Package**
2. Scegliendo il database che abbiamo scaricato
3. Clicchiamo sull'Image target e nell'inspector ora dovremmo vedere le opzioni del database
4. Impostiamo l'utilizzo del nostro database
5. Selezioniamo il nostro target

Se abbiamo fatto tutto correttamente il target dovrebbe apparire nella scena

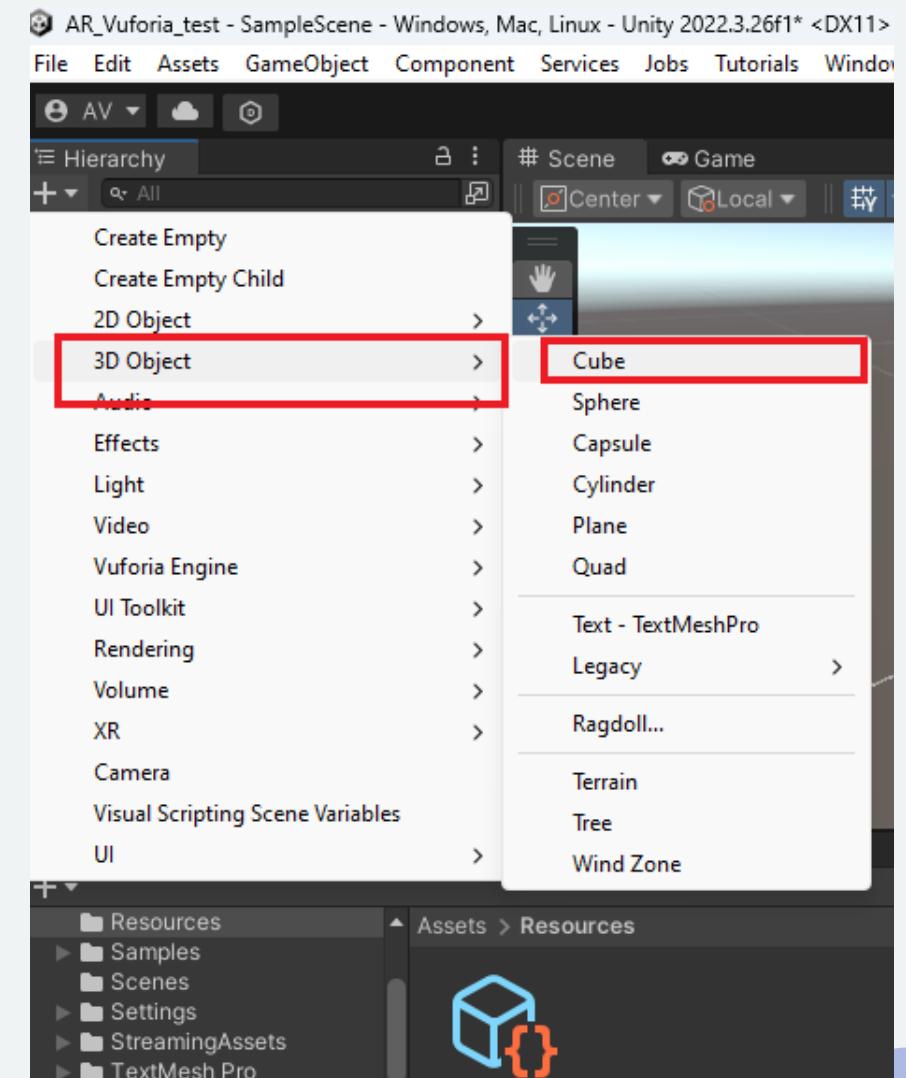
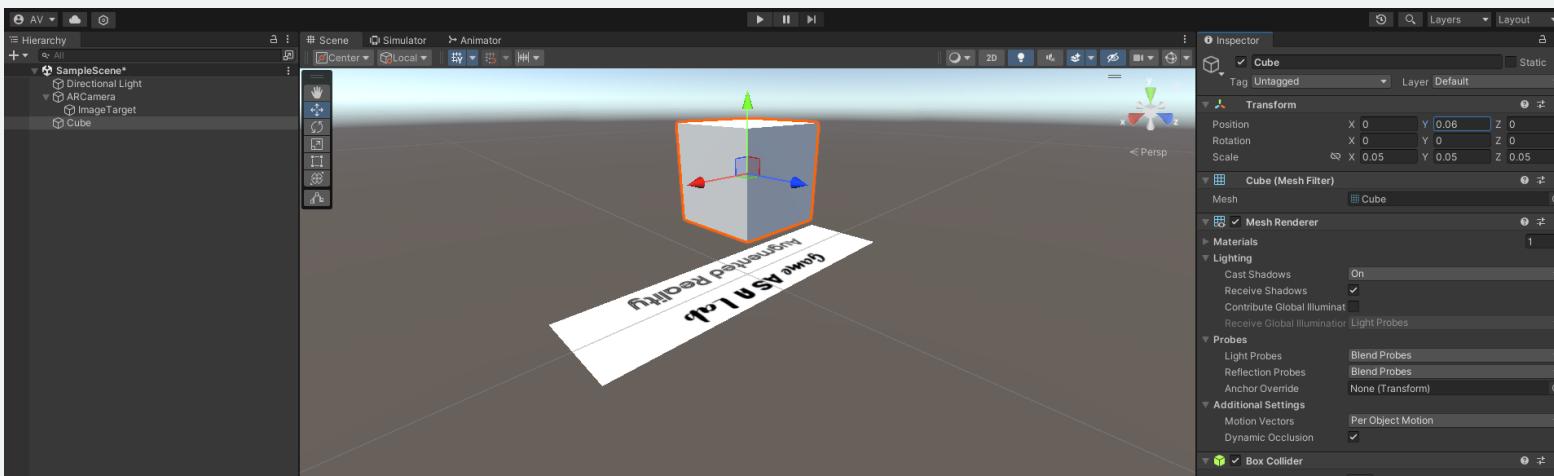


Game As a Lab - anna.vitali7@unibo.it



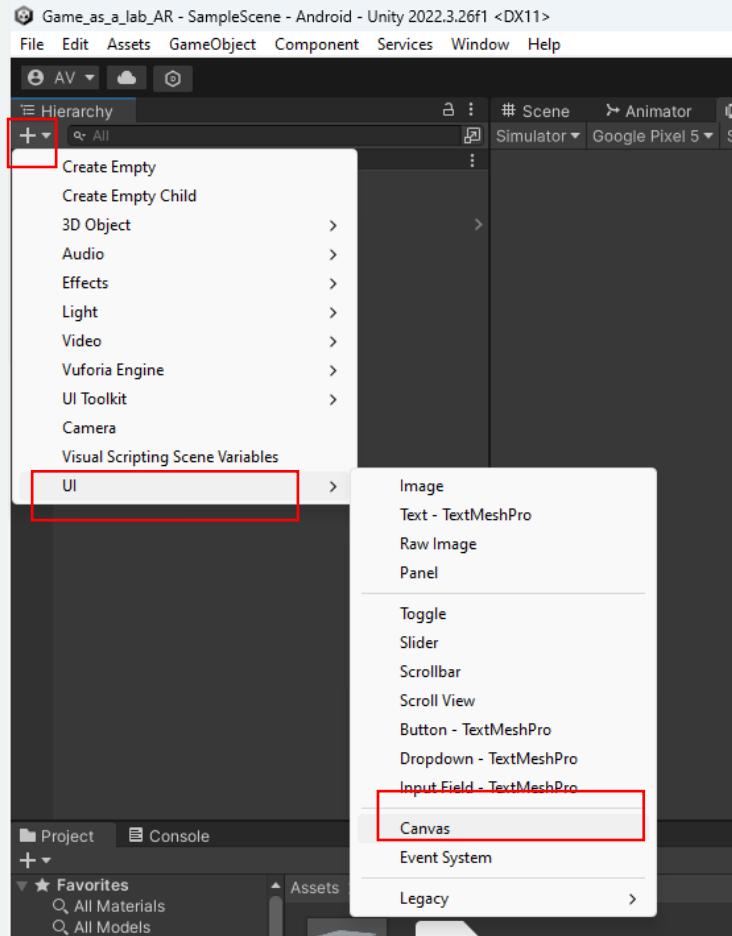
# Visualizziamo un ologramma

1. Clicchiamo su poi su **3D Object** e infine su **Cube**
2. Spostiamo il Cubo creato in modo che sia figlio dell'Image target, lo **ridimensioniamo** e lo **posizioniamo** a piacimento



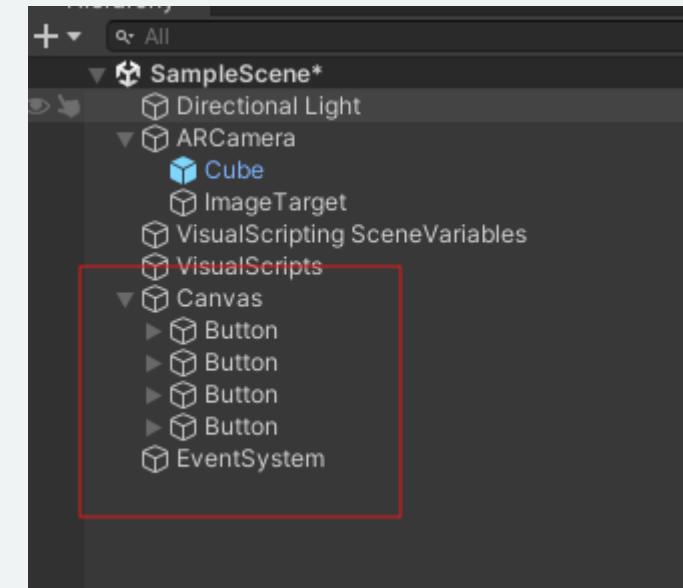
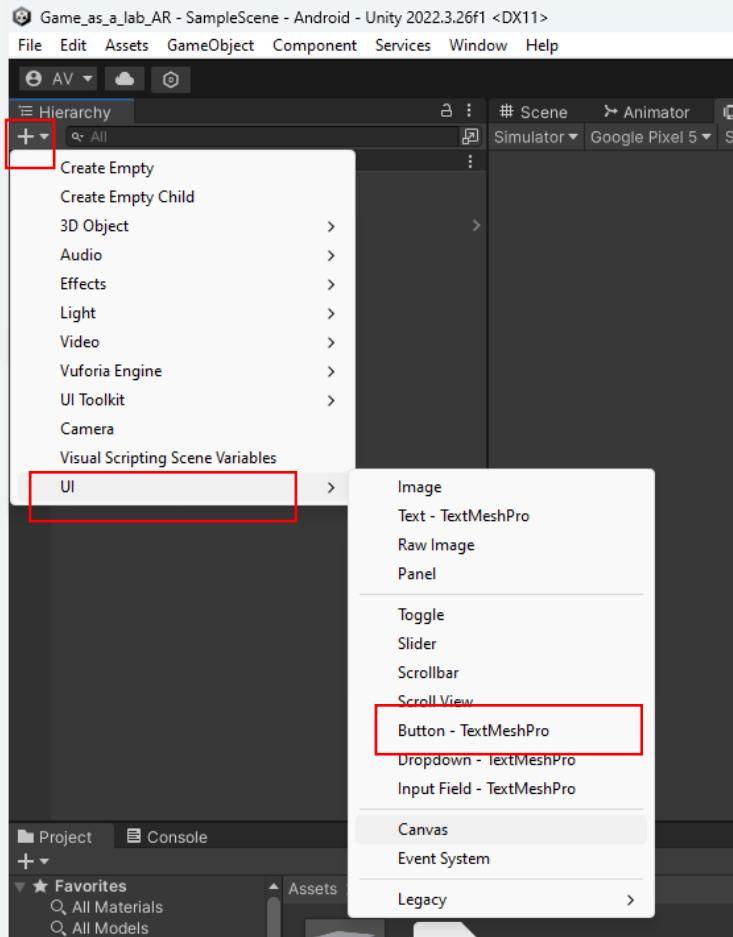
# Creazione di una GUI

- Per creare una GUI occorre **aggiungere** un **Canvas** alla scena



# Creazione di una GUI

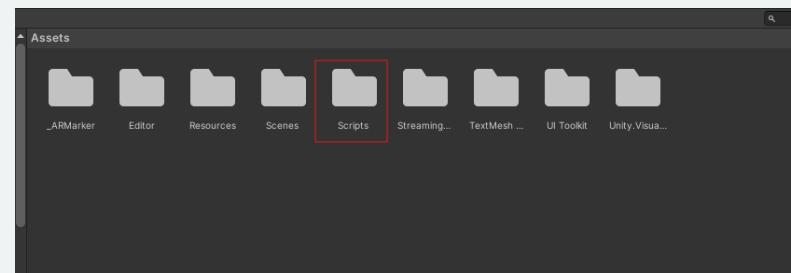
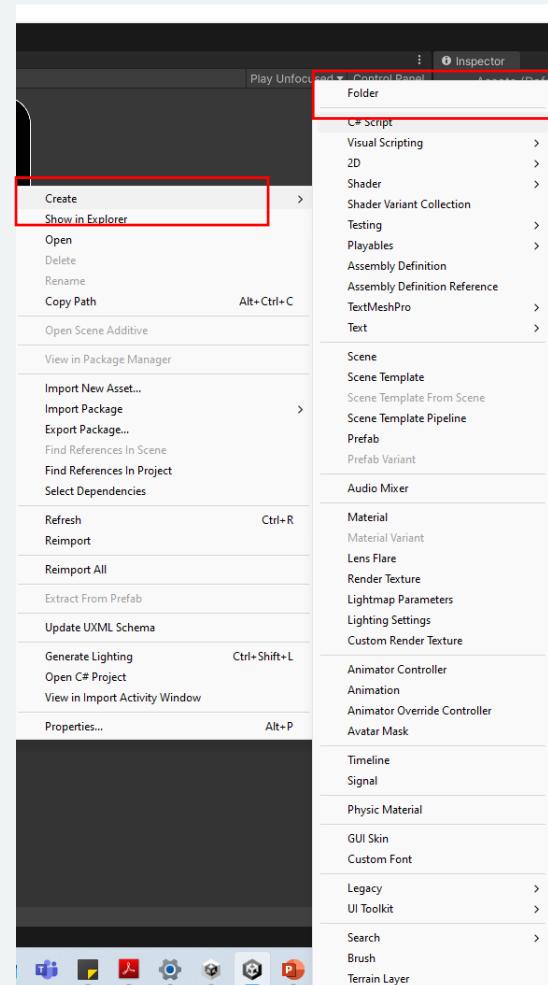
- A questo punto al canvas aggiungiamo i 4 pulsanti



# Creazione di uno Script

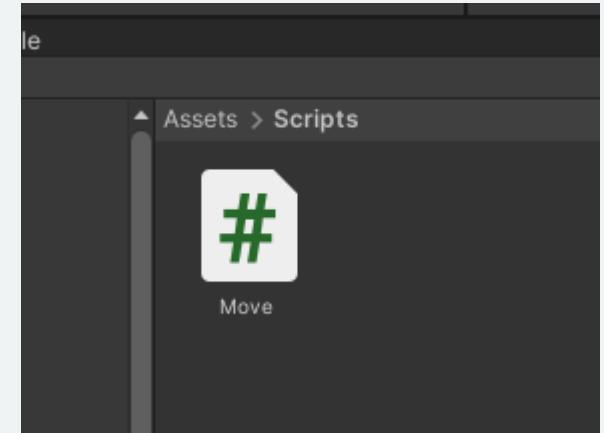
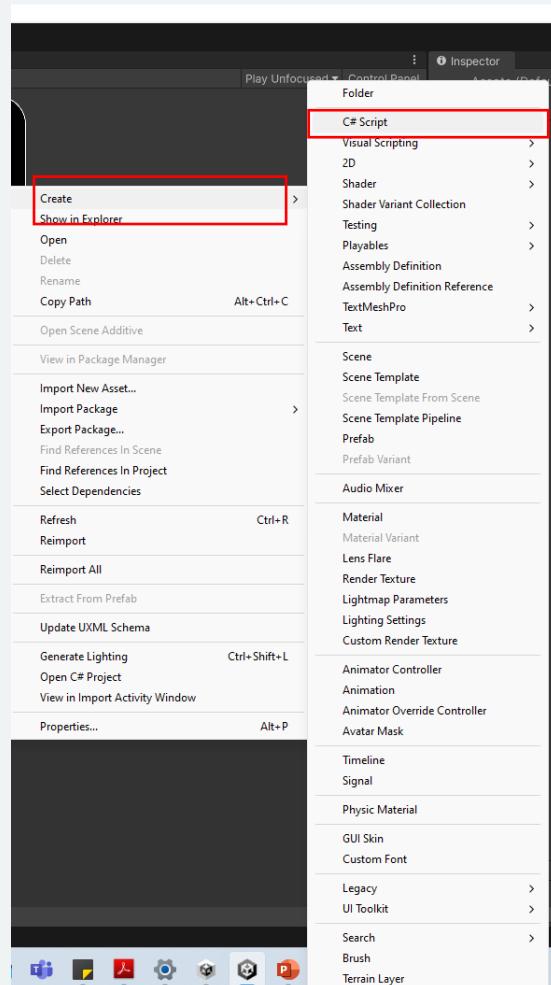
- Creiamo lo script che ci consenta di gestire il movimento del nostro cubo

1. Dalla cartella **Assets** del nostro progetto, cliccando il **pulsante destro del mouse** creiamo una **nuova cartella**
2. Gli assegniamo il nome **Scripts**



# Creazione di uno Script

3. Ci spostiamo nella cartella **Scripts**
4. Clicchiamo il **tasto destro** del mouse e selezioniamo **Create -> C# Script**
5. Assegniamo allo script il nome **Move**
6. Lo apriamo nel nostro editor C#



# Script Move.cs

1. Eliminiamo i metodi **Start()** e **Update()**
2. Stabiliamo una **costante** di movimento **moveAmount**
3. Creiamo i **quattro** metodi di movimento

```
public class Move : MonoBehaviour
{
 public float moveAmount = 0.1f;

 public void MoveUp()
 {
 // TODO: Implement logic to move the GameObject up by moveAmount
 }

 public void MoveDown()
 {
 // TODO: Implement logic to move the GameObject down by moveAmount
 }

 public void MoveLeft()
 {
 // TODO: Implement logic to move the GameObject left by moveAmount
 }

 public void MoveRight()
 {
 // TODO: Implement logic to move the GameObject right by moveAmount
 }
}
```

# Script Move.cs

- Occorre aggiornare la posizione dell'oggetto a cui lo Script è associato
  - possiamo aggiungere un nuovo vettore a quello della posizione attuale, che applica il cambiamento

```
public class Move : MonoBehaviour
{
 public float moveAmount = 0.1f;

 public void MoveUp()
 {
 transform.position += new Vector3(0, moveAmount, 0);
 }

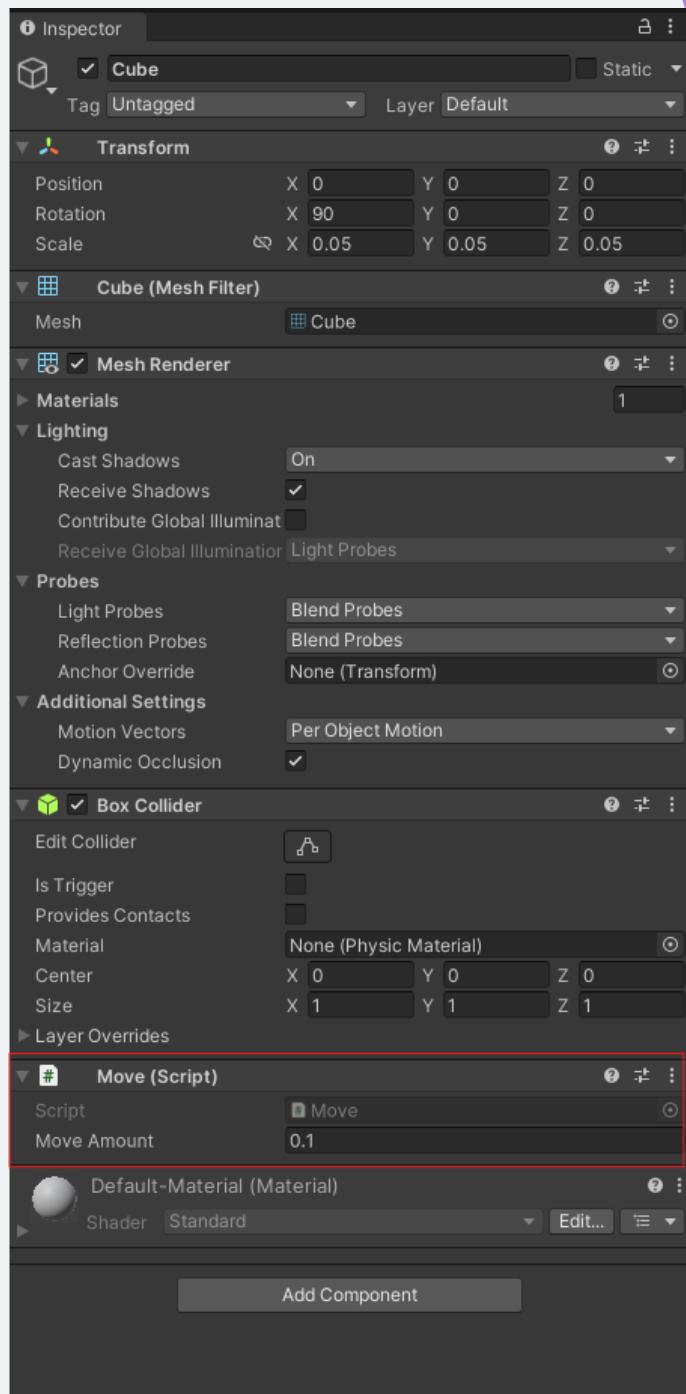
 public void MoveDown()
 {
 transform.position += new Vector3(0, -moveAmount, 0);
 }

 public void MoveLeft()
 {
 transform.position += new Vector3(-moveAmount, 0, 0);
 }

 public void MoveRight()
 {
 transform.position += new Vector3(moveAmount, 0, 0);
 }
}
```

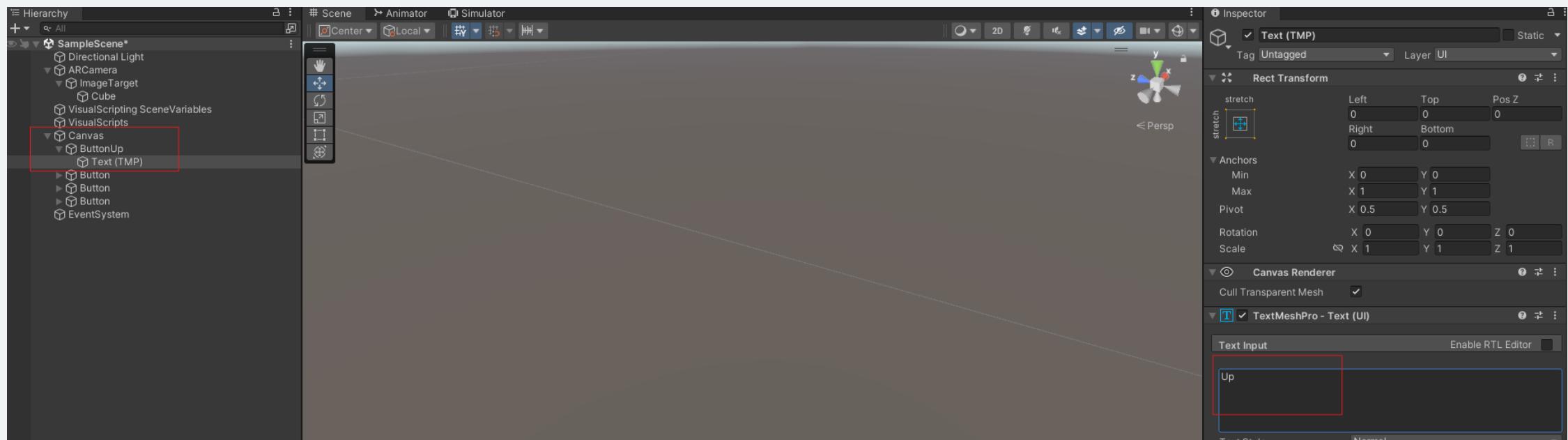
# Script Move.cs

- **Associamo lo Script al nostro cubo**
  - **Lo trasciniamo e lo lasciamo sul cubo della scena**



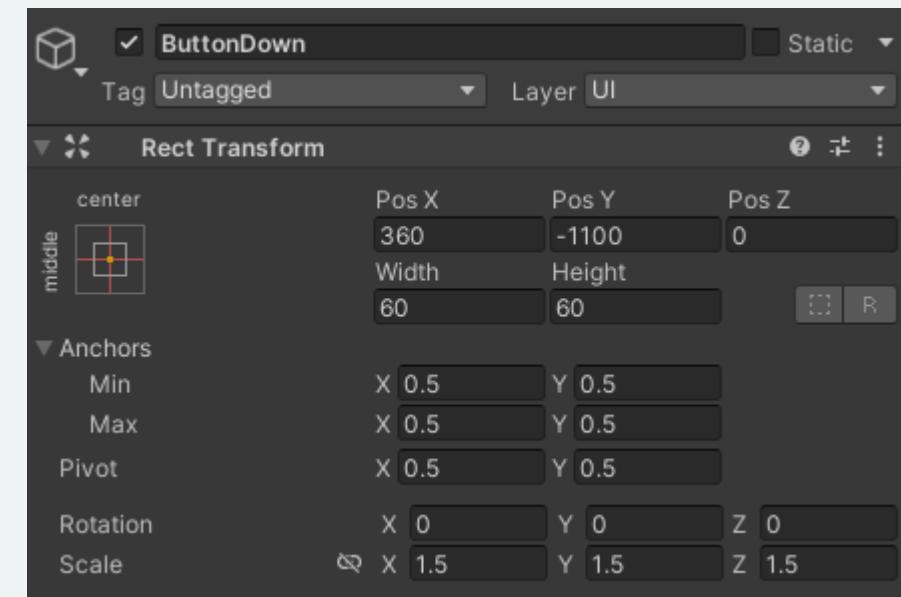
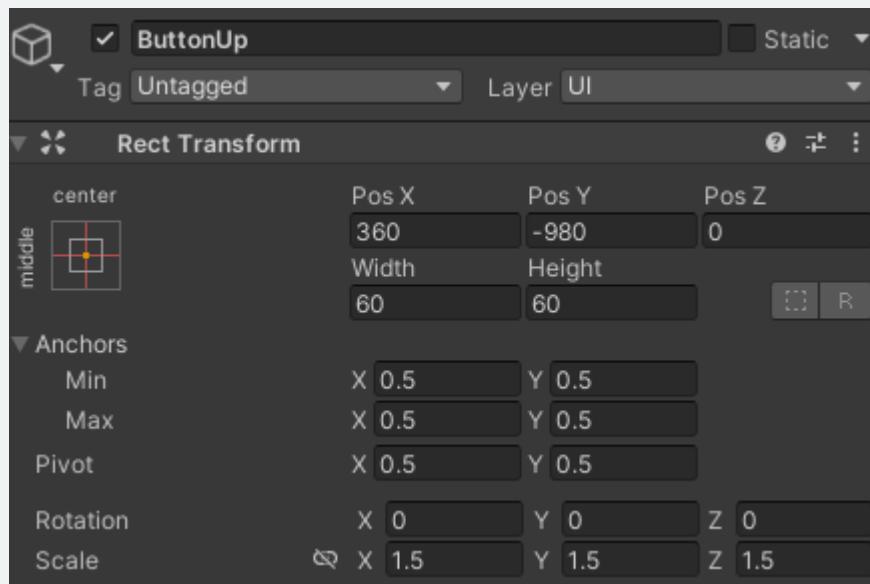
# Creazione di una GUI

- Sistemiamo i nostri pulsanti
  - associandoli un testo: Up, Down, Left, Right
  - Una posizione



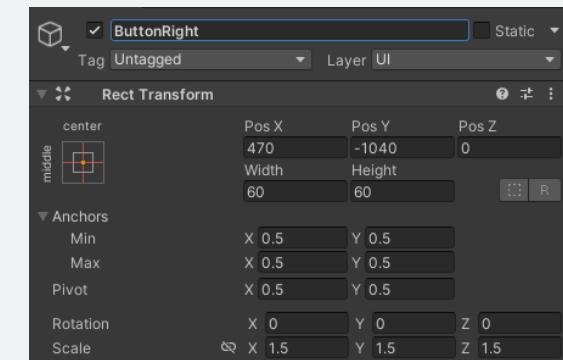
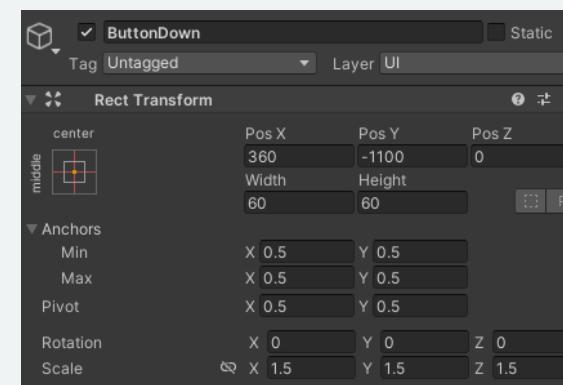
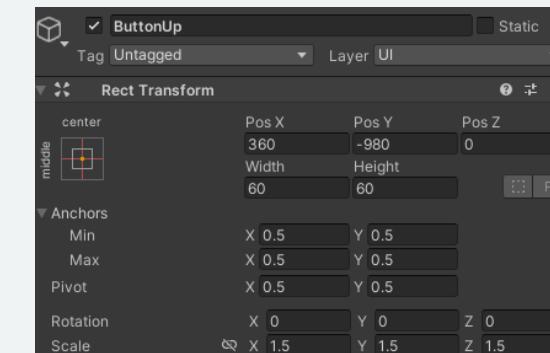
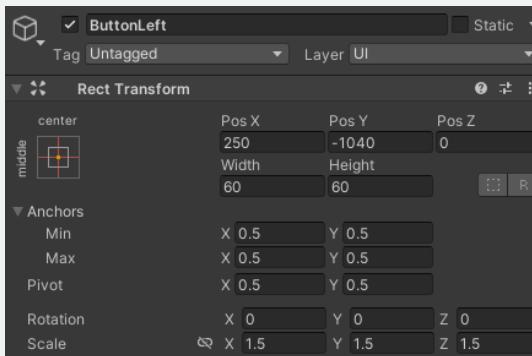
# Creazione di una GUI

- Sistemiamo i nostri pulsanti
  - associandoli un testo: Up, Down, Left, Right
  - Una posizione



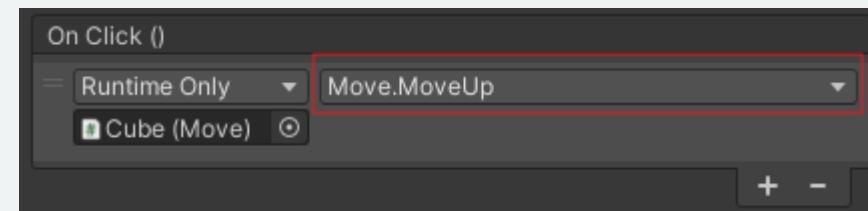
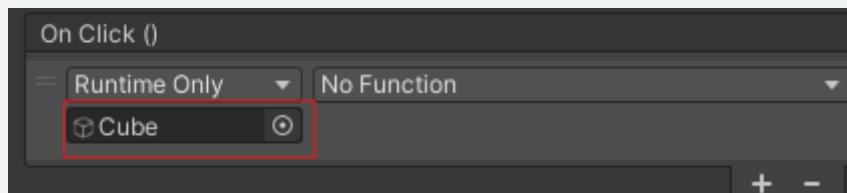
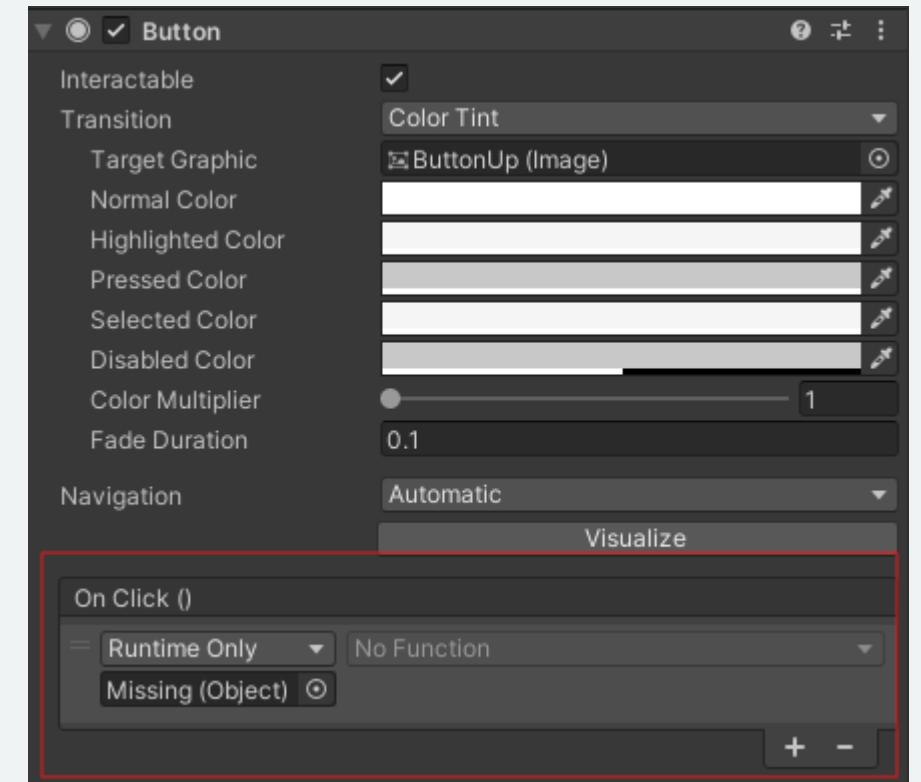
# Creazione di una GUI

- Sistemiamo i nostri pulsanti
  - associandoli un testo: Up, Down, Left, Right
  - Una posizione



# Gestione di un Evento

- Ogni bottone di **default** ha associato l'evento **On Click()**
- Cliccando sul pulsante **+** è possibile aggiungere una funzione da richiamare
  1. Associamo a **Missing Object** il nostro **Cubo**
  2. Clicchiamo su **No Function** e cerchiamo la voce **Move** (il nostro script)
  3. Selezioniamo il **metodo** da associare al bottone
  4. **Ripetiamo** per tutti i pulsanti



# Eseguire l'applicazione

- Per poter vedere il risultato, avendo a disposizione il target utilizzato

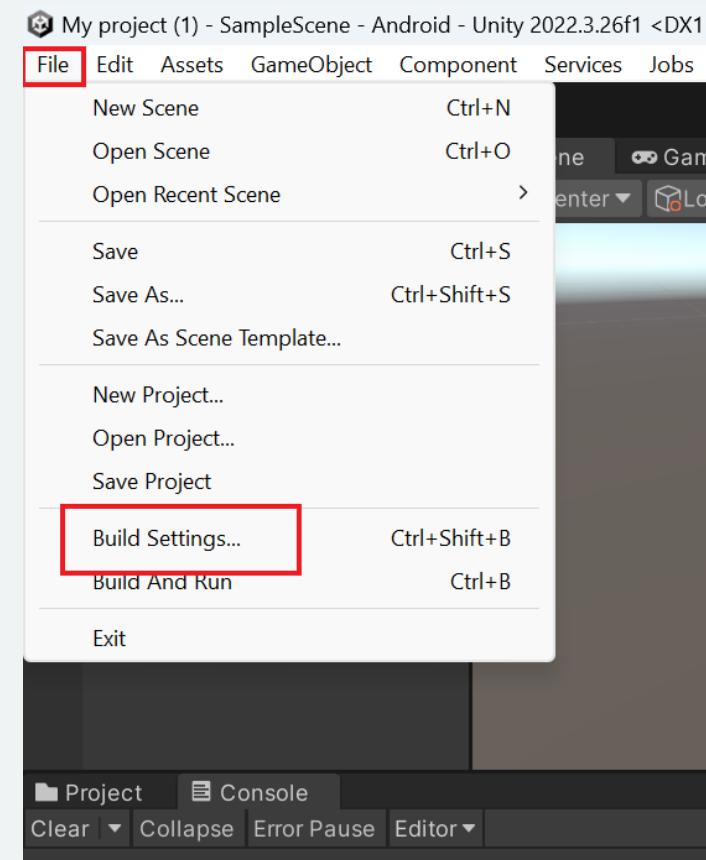
## Esecuzione da PC

- Possiamo premere il pulsante  e eseguire l'applicazione sul pc, attivando la telecamera

# Eseguire l'applicazione

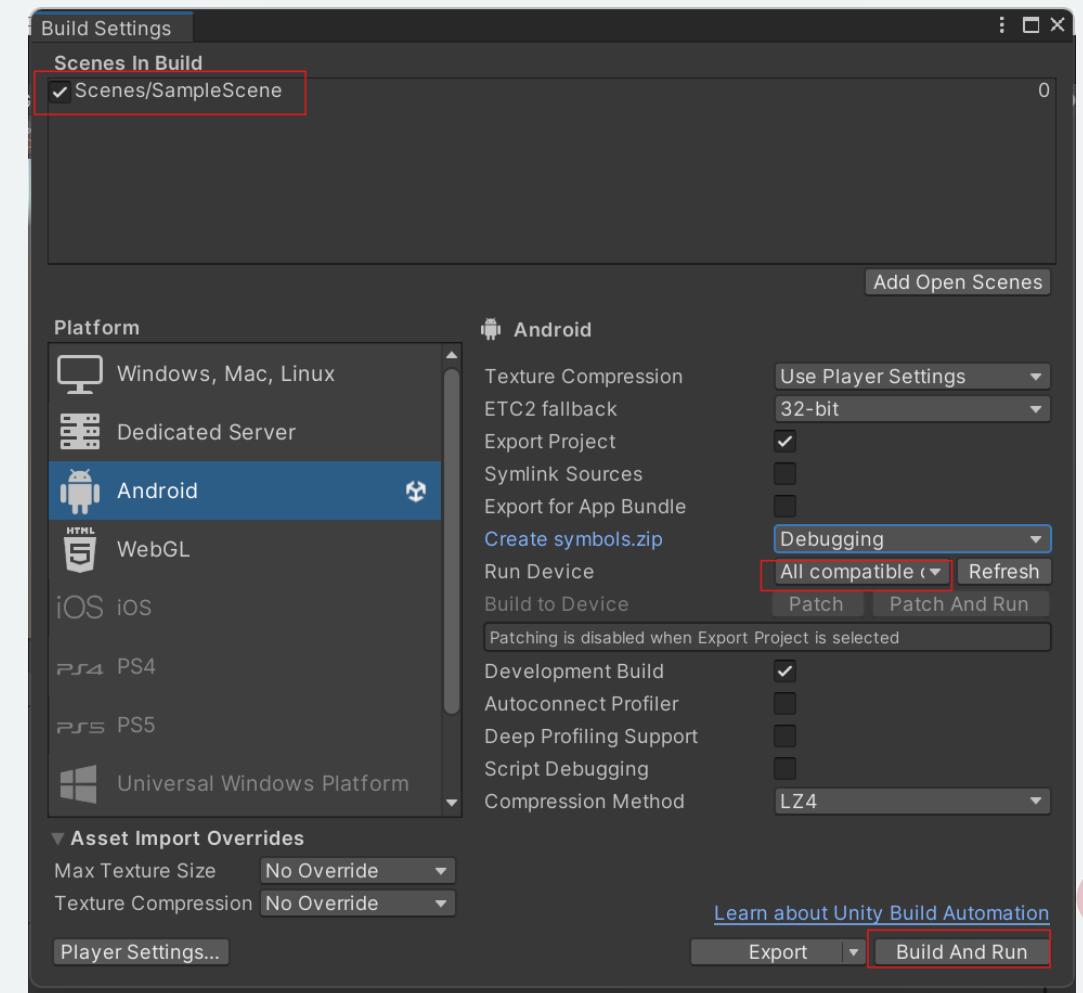
## Esecuzione da dispositivo Mobile

1. Attiviamo la **modalità sviluppatore** sul nostro dispositivo e il **DEBUG USB**
2. Colleghiamo il dispositivo tramite cavo USB-C al computer
3. Andiamo su **File -> Build Setting** e selezioniamo **Android**



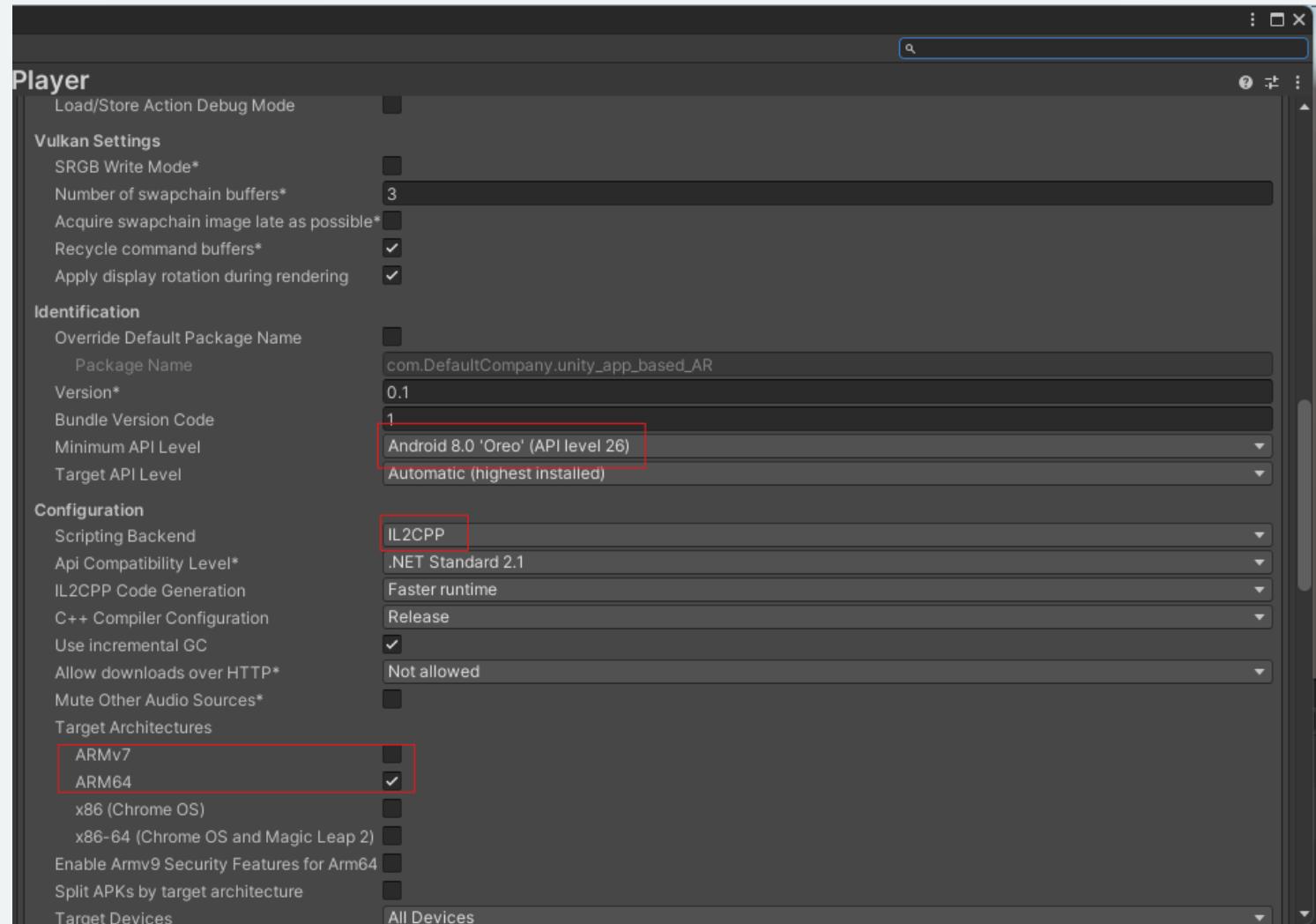
# Eseguire l'applicazione

4. Spuntiamo la **scena corrente**, per effettuare la build
5. Selezioniamo **Android** come piattaforma



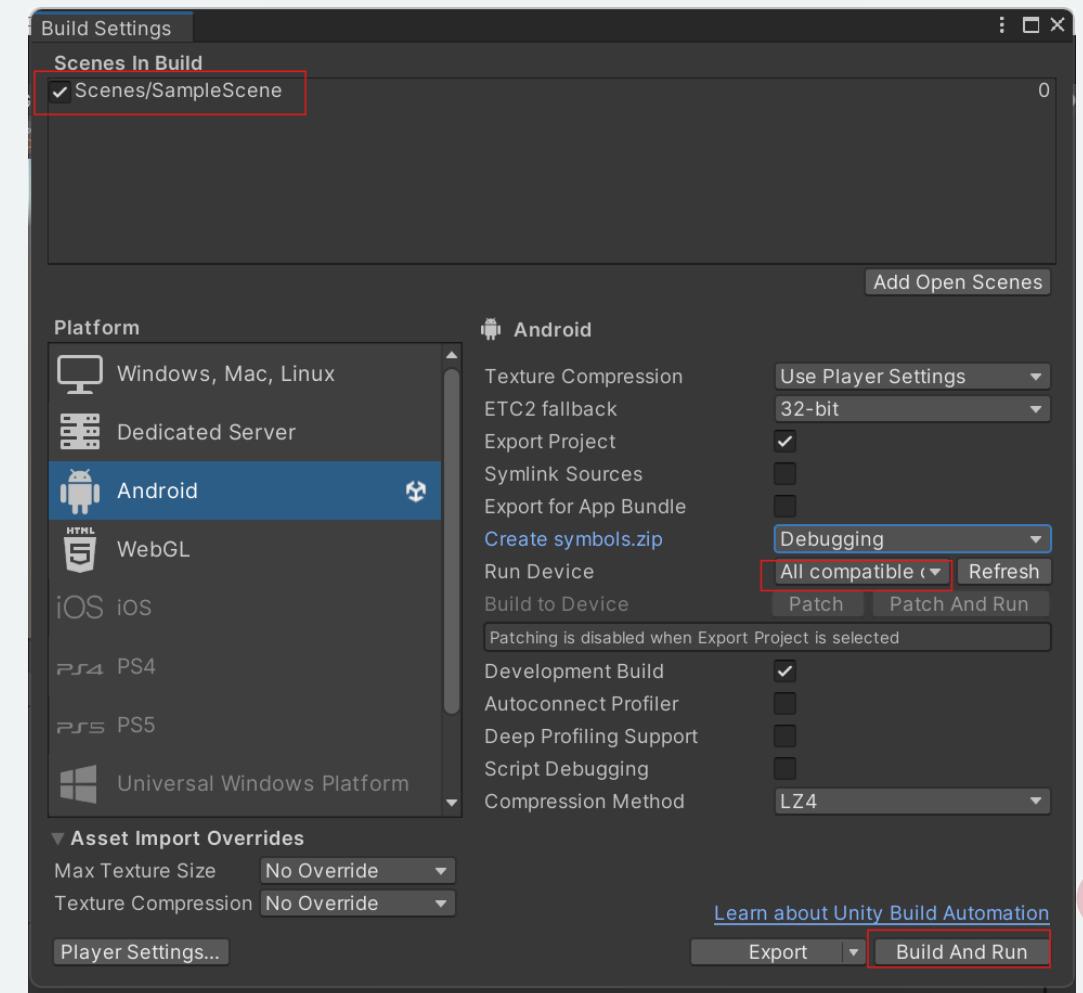
# Eseguire l'applicazione

4. Clicchiamo su Player Settings
5. Impostiamo come Minimum API Level «**Oreo**» **26**
6. Come Scripting Backend **IL2CPP** e Target Architecture **ARM64**
7. Chiudiamo il Player



# Eseguire l'applicazione

8. Se abbiamo fatto tutto correttamente dovremo vedere apparire il nome del nostro device su cui effettuare la build
9. Clicchiamo sul pulsante **Build and Run**
10. Associamo il **nome che vogliamo** all'applicazione e salviamo



# Risultato



# Risorse utili

## WEB Based AR

- [Babylon.js per principianti](#)
- [Animazione in Babylon.js](#)
- [MRTK support in Babylon.js](#)
- [Tutorial Microsoft Babylon.js](#)

## APP Based AR

- [Unity per principianti](#)
- [Tutorial AR application in Unity](#)