UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

3D Augmented Reality

A.Y. 2022/2023

Physics & User Interface elements

LAB experience 2

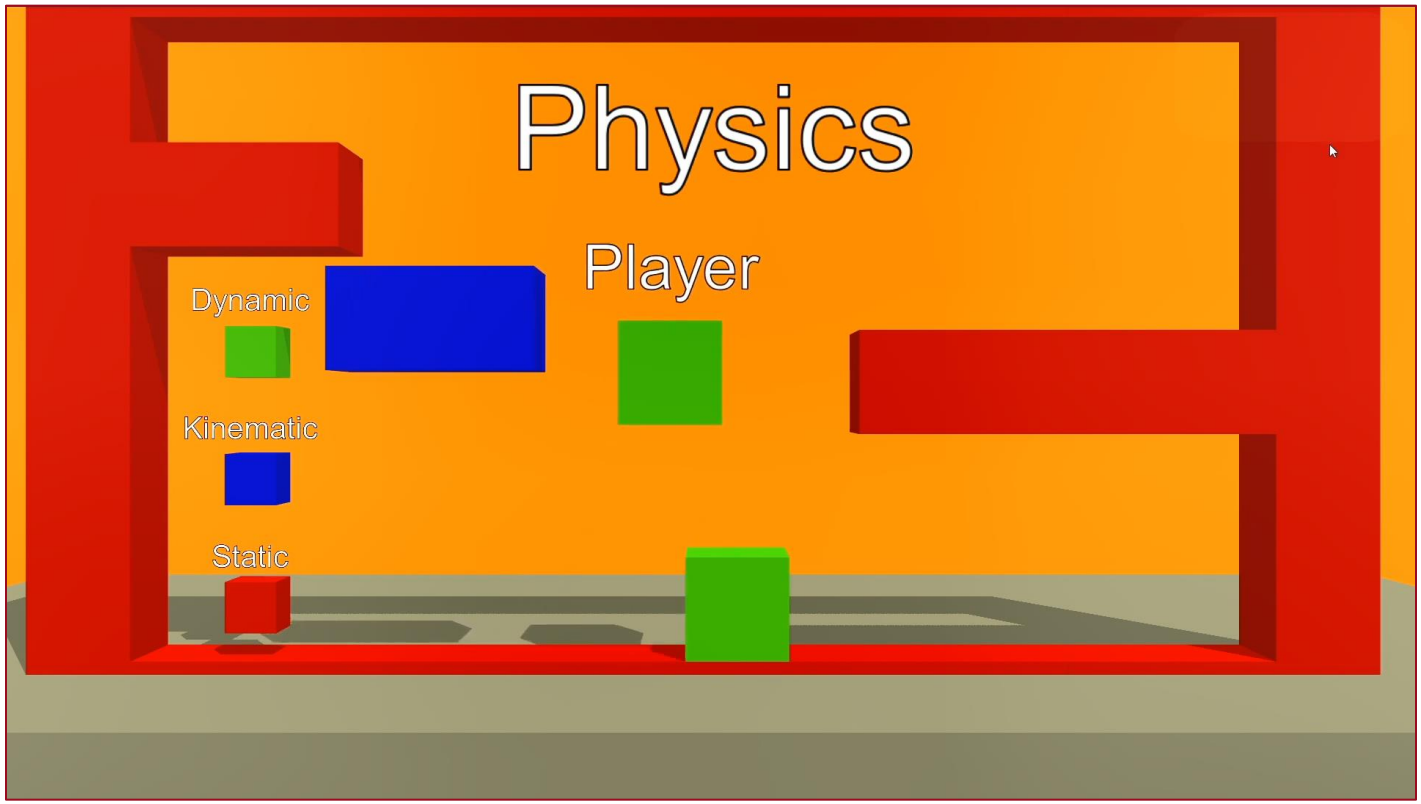
Elena Camuffo

elena.camuffo@phd.unipd.it

Physics in Unity

- Unity offers an **integrated physical engine system**, that helps you simulate physics in your Project ensuring that the objects correctly accelerate in response to collisions, gravity, and other forces.

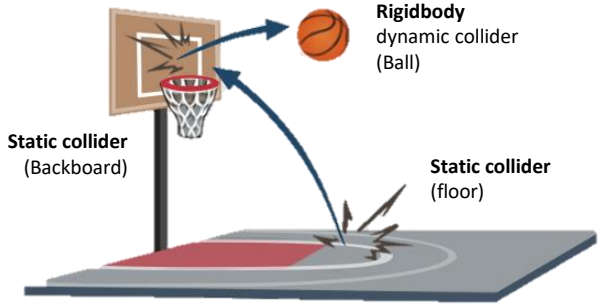
Physics Example

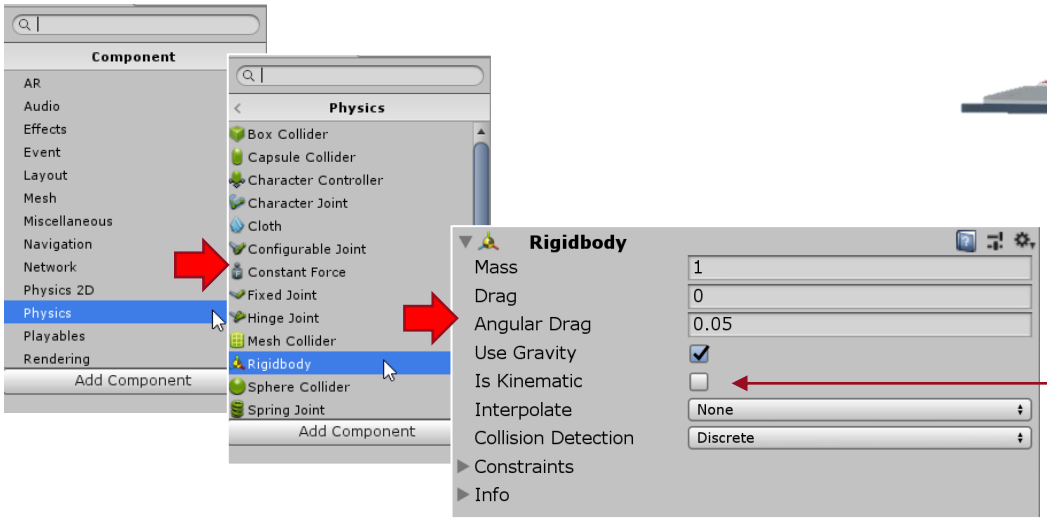


Full video @: https://www.youtube.com/watch?v=v0XmeqzvLF4

Rigidbodies

- **Rigidbody** is the main component that enables physical behaviour for a Game Object.
- Any Game Object must contain a Rigidbody to be influenced by gravity or other forces added via scripting, or to interact with other objects.

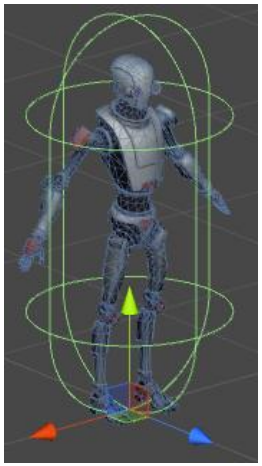


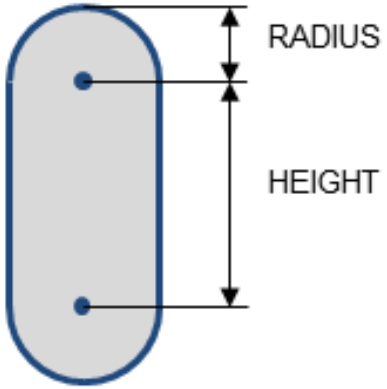


If **isKinematic** is enabled, the rigidbody will be fully controllable by scripting, through the *Transform* properties. Kinematic bodies also affect the motion of other rigidbodies.

Colliders

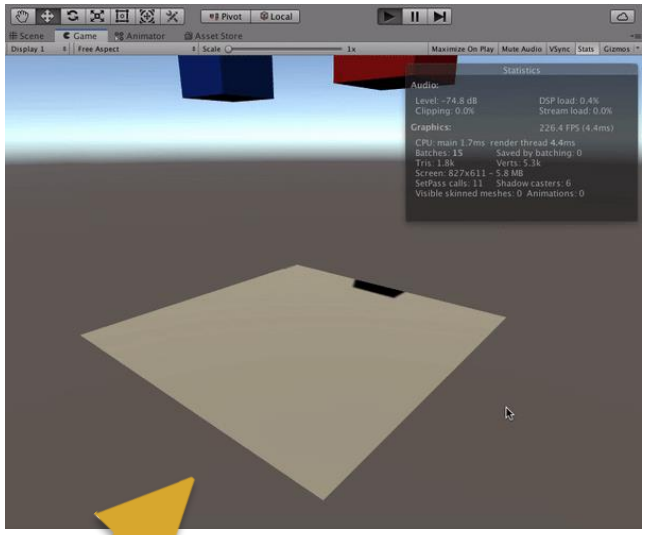
- **Colliders** are components which define the shape of a Game Object for the purposes of physical collisions.
- A collider, which is invisible, does not need to be the exact same shape as the Game Object's mesh. A rough approximation of the mesh is often more efficient and indistinguishable in gameplay.



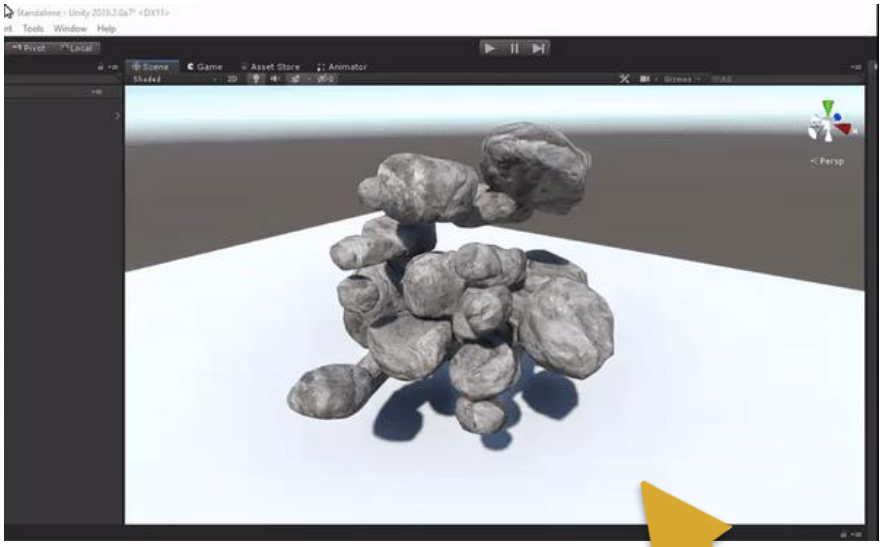


- **Dynamic colliders** are colliders attached to Game Objects that have a Rigidbody component attached.
- **Static colliders** are colliders attached to Game Objects that have no Rigidbody component attached (used for motionless elements of the scene, e.g., walls).
- Static colliders **can interact** with dynamic colliders, but they don't move in response to collisions.

Physics Example



Boxes: dynamic colliders (rigidbodies)
Plane: static collider



Stones: dynamic colliders (rigidbodies)
Plane: static collider

Script Example

The **Physic Material** describes how to handle colliding objects (e.g., the ball's bounce on the surface).

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

public class Ball : MonoBehaviour
{
    // Method to manage the collision with another object
    private void OnCollisionEnter(Collision collision)
    {
        // select a color with the function and change the material property
        Color randomColor = SelectRandomColor();
        GetComponent<Renderer>().material.color = randomColor;
    }

    // Update is called once per frame
    private Color SelectRandomColor()
    {
        // create a new color picking random values of red, green and blue
        Color newColor = new Color(
            UnityEngine.Random.Range(0f, 1f),
            UnityEngine.Random.Range(0f, 1f),
            UnityEngine.Random.Range(0f, 1f)
            );

        // return the created color
        return newColor;
    }
}
```

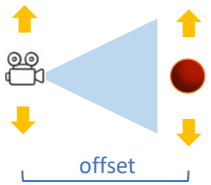
OnCollisionEnter()

Override the method for detecting a collision with another Game Object. When a collision is detected, the ball changes its color.

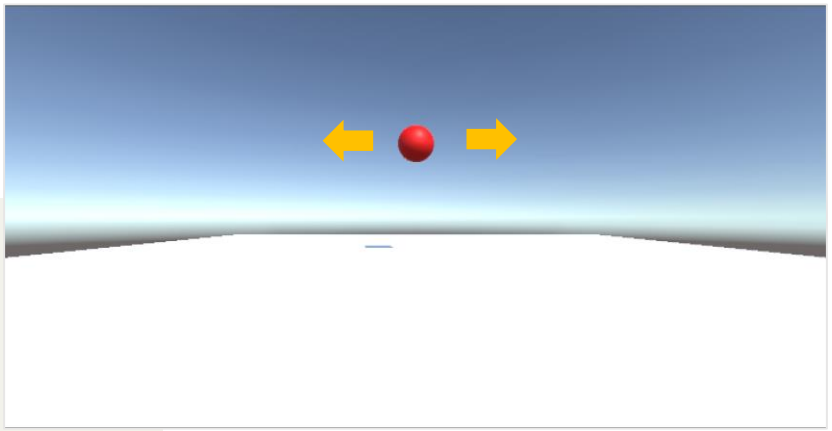
SelectRandomColor()

User defined function to pick a random RGB color.

Player Control



Make the camera follow the Player with an **offset**, copying its position over time.



```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class AddPlayerControlledVelocity : MonoBehaviour
6   {
7       // Speed parameter
8       [SerializeField]
9       float speed = 5f;
10
11      // Variables to select the keys to get when the force is applied
12      [SerializeField]
13      KeyCode keyPositive;
14      [SerializeField]
15      KeyCode keyNegative;
16
17      // FixedUpdate is executed together with the physics engine
18      void FixedUpdate()
19      {
20          // Build a new force vector according to 'speed'
21          Vector3 v3Force = new Vector3((speed / 100), 0f, 0f);
22
23          // Get keyboad positive and negative keys to move the Game Object
24          if (Input.GetKey(keyPositive))
25              GetComponent<Rigidbody>().velocity += v3Force;
26
27          if (Input.GetKey(keyNegative))
28              GetComponent<Rigidbody>().velocity -= v3Force;
29      }
30  }
```

Variables

The keys for positive and negative directions can be left and right arrows.

FixedUpdate()

Get key values and add a force vector in the chosen direction.
Note: *Update()* runs once per frame. *FixedUpdate()* runs according to a fixed rate and in conjunction with the physics engine.

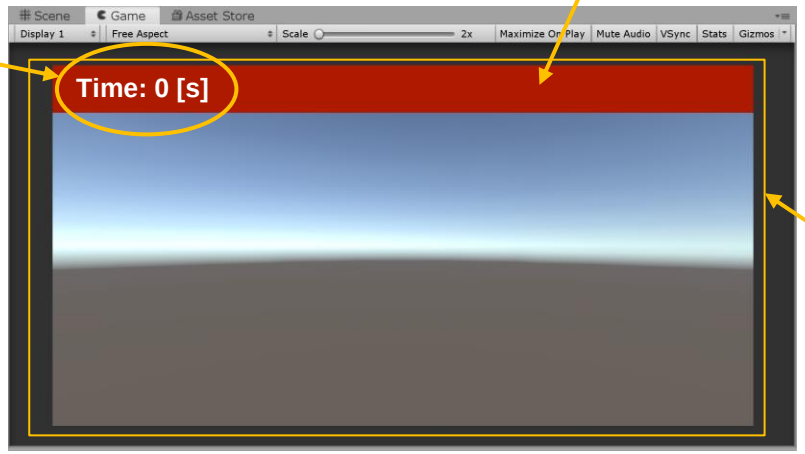
User Interface (UI)

- A **User Interface** is what is displayed to the user when the application is running, and is made of set of objects responsible for the interaction between the application and the user.
- Unity provides a simple **UI toolkit** for developing user interfaces for games and applications.

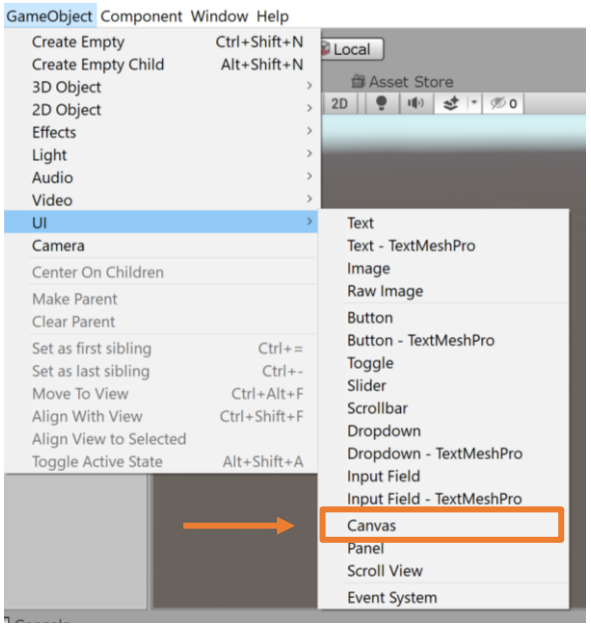
UI elements

The **Image** component is fixed on the top of the viewport, even though the content in the scene changes.

The **Text** changes dynamically according to a Script attached.

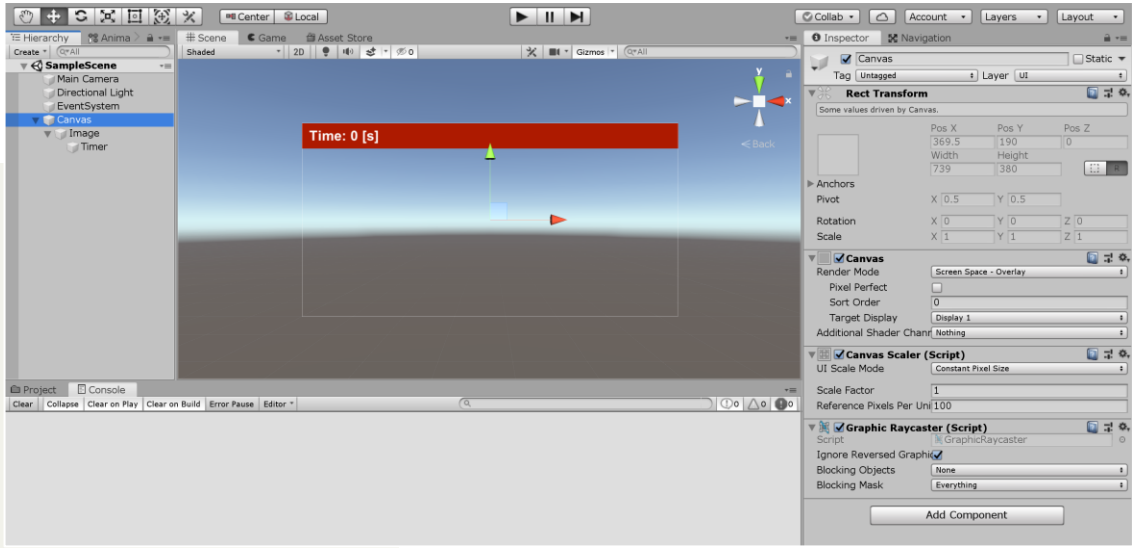


The **Canvas** covers the whole viewport.



- The **Canvas** is the Game Object where to draw UI elements in Unity. Put a canvas in your scene to start building your UI.
- The other elements (e.g. Text, Images, Buttons etc.) need to be set as its **children** to be visualized.

Script Example




```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5
6   public class Timer : MonoBehaviour
7   {
8
9       // Start is called before the first
10      // frame update
11      void Start()
12      {
13          // Obtain the current time.
14          string currTime = "Time: 0 [s]";
15
16          // Display the current time.
17          gameObject.GetComponent<Text>().text = currTime;
18      }
19
20      // Update is called once per frame
21      void Update()
22      {
23          // Obtain the current time.
24          string currTime = "Time: " + Mathf.Ceil(Time.time).ToString() + " [s]";
25
26          // Display the current time.
27          gameObject.GetComponent<Text>().text = currTime;
28      }
29  }
```

Update()

The new time is computed relying on the apposite function and then stored in the Text component of the UI Game Object.

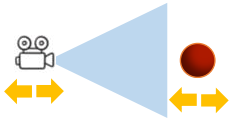
EXERCISE: Ball Game

1. **Load the Scene**

   Import in your Unity the Base project for the lab (folder *BallGame_Exercise*) and build other platforms in order to create a Game environment.

2. **Add player control**

   Write a script to make the ball moving on the platforms according to physical forces and make the camera follow it with an offset.



3. **Add User Interface**

   Create User Interface elements to visualize a “Game Over message” when the ball hits lava, and a “Victory message” when the ball hits a specific “victory element”, at the end of the path.

EXERCISE: Ball Game

4. (ADVANCED) Restart game

Add a button element to the “Game Over message” and the “Victory message” to make the scene restart as it is hit.

Suggestion: You can manage the change of Scene using the *SceneManager.LoadScene* method (Check the documentation for further details).

5. (OPTIONAL) Add Levels

You can create more than one scene, corresponding to different difficulty levels and link through the “victory element” of the previous scene.

Suggestion: Start going to *File > BuildSettings* and drag and drop the second scene in *Scenes in Build.*