

Dev - Basics 1

Minor Game Design &
Development



HOGESCHOOL ROTTERDAM



Dev Class Intro

- Unity, C#
- Focus on 2D, then 3D
- Github, Discord



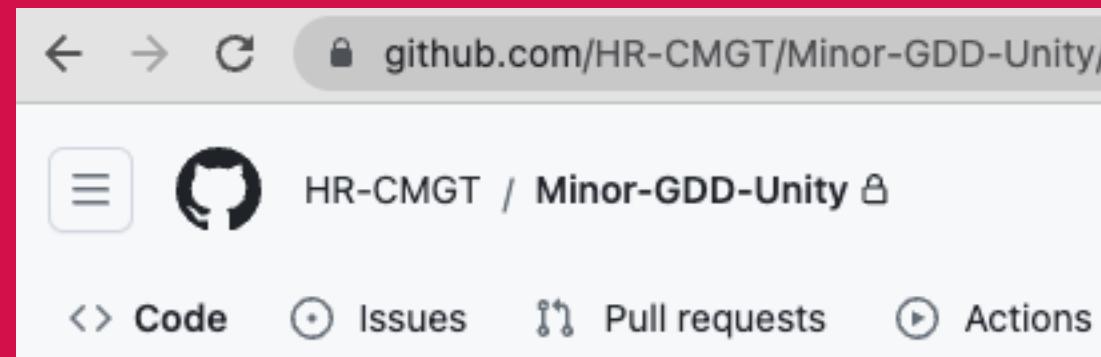
Part 1

- Github MGDD
- Github Documentation
- Unity Startproject
- Unity Package Manager
- Assignment (StartProject)
- Rigidbody2D - Component
- Rigidbody2D - Scripting
- Unity Documentation
- Unity Learn
- Unity's Script Lifecycle



Github MGDD

<https://github.com/HR-CMGT/Minor-GDD-Unity/>



Github Documentation

Resources

Tutorials

Project Files

Unity Tips & Best Practices

Code Snippets

Basics 1

[Presentation](#) - [Project Files](#) - [Resources](#) - [Tutorials](#) - [Assignment](#)

Presentation

This week's [presentation](#) can be found [here](#)

Resources

- Our own [tips, tricks and best practices](#) for working with Unity, with a bunch of gifs
- A list of [external tutorials](#) to help you with specific topics, from learning the basics to creating a certain effect.
- Get graphics, sounds, code and other free stuff from the [resources](#) page

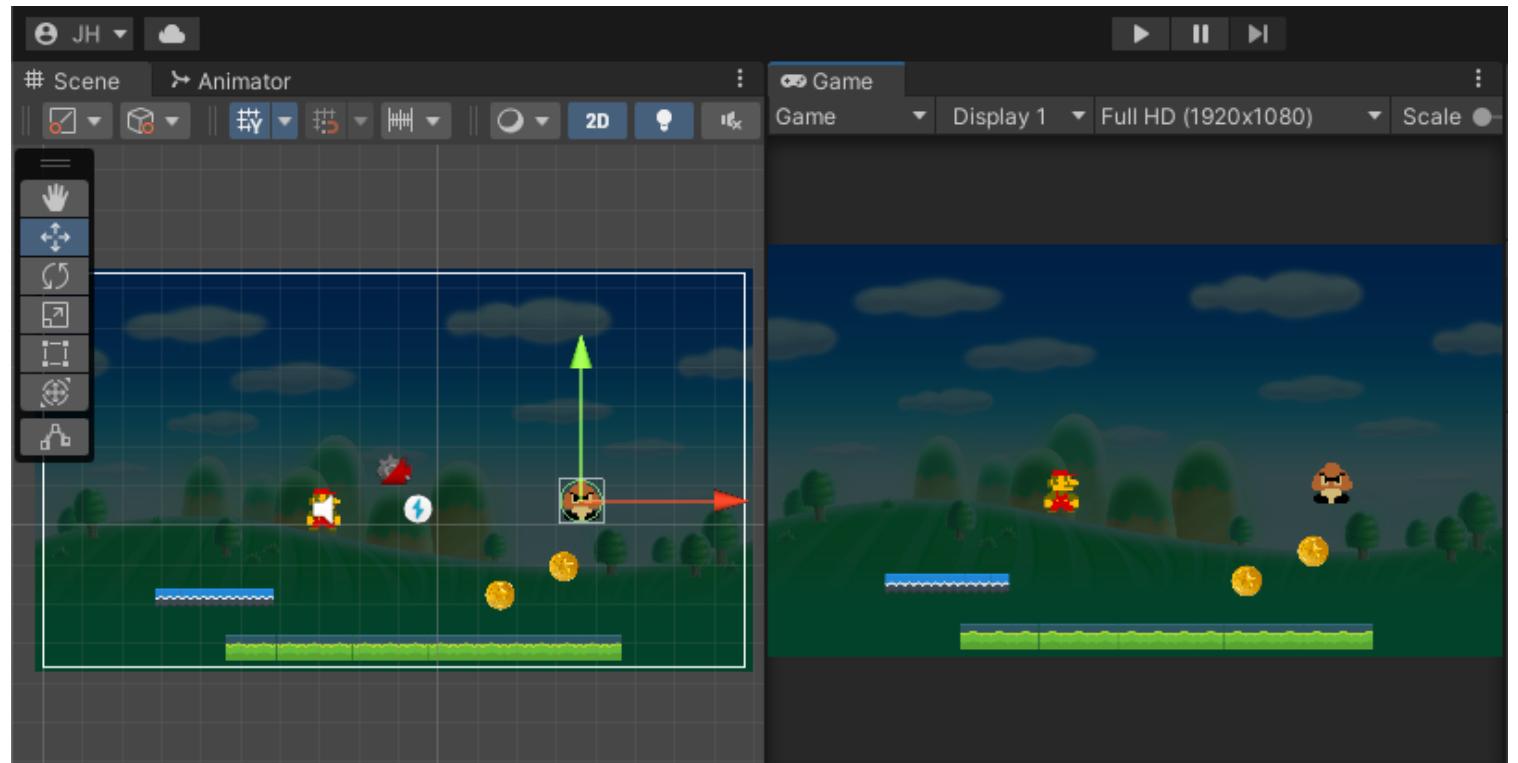
Assignment

1. Download and open Unity
2. Create a new 2D project



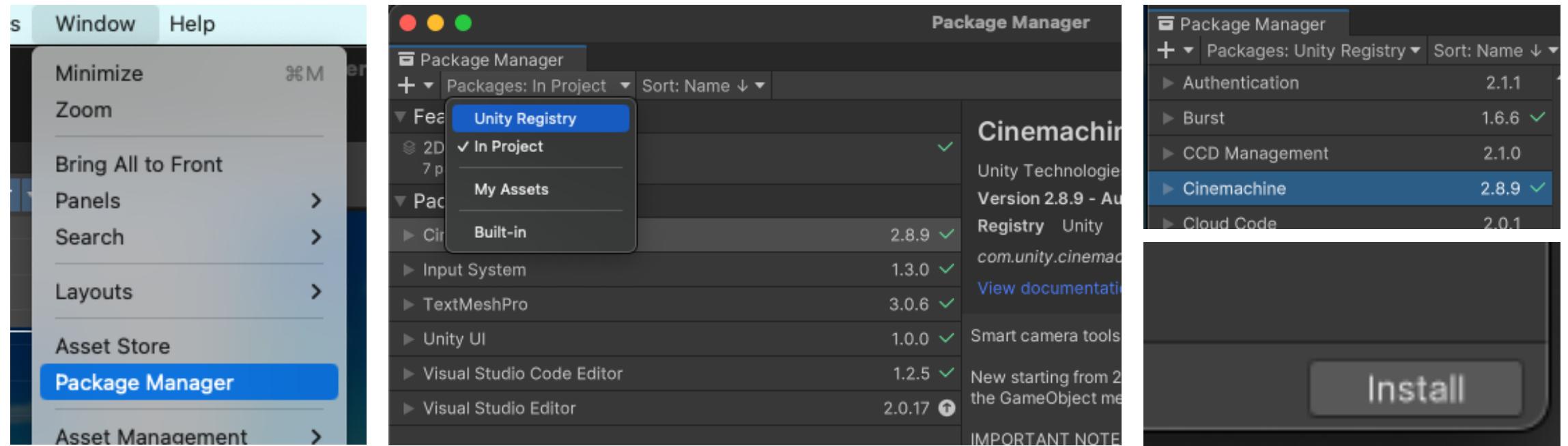
Unity Startproject

UnityPackage
[[Github link](#)]



Unity Package Manager

Open Package Manager, select Unity Registry, install Cinemachine & Input System



Assignment (StartProject)

Camera Follow

Configure the Player

Programming the Player

Jump

Move

ResetPlayer

OnCollisionEnter2D

CollectCoin

Jump Sprite

Add Coins

Play Around



Rigidbody2D - Component

Note: Requires a Collider2D to work!

Body Types:

Dynamic / Kinematic / Static

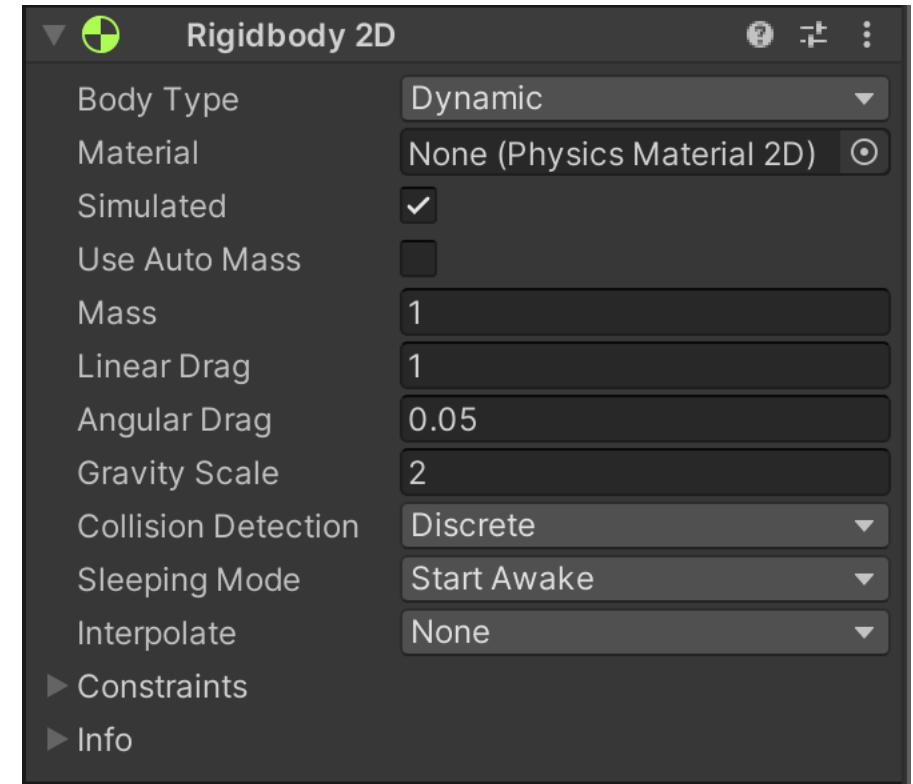
Constraints:

Freeze Position X/Y, Freeze Rotation Z

Material:

PhysicsMaterial2D, friction and bounciness

Mass, Gravity Scale and Linear Drag ('air-friction') settings for "weight"



Rigidbody2D - Scripting

Frequently used properties and functions of Rigidbody2D

Tip: use AddForce [[script ref](#)] for moving a dynamic Rigidbody2D, and use a big

```
rb = GetComponent<Rigidbody2D>();

rb.velocity = Vector2.right; // set the velocity directly, ignore mass
rb.AddForce(Vector2.right); // push the object, include mass
rb.isKinematic = true; // "freeze" the object
rb.MovePosition(new Vector2(0, 0)); // set the world position directly
```

[Unity Script Reference - Rigidbody2D](#)



Unity Documentation

1. Unity Manual

“Tutorials” on how Unity’s systems work

The screenshot shows the Unity Manual interface. At the top, there's a navigation bar with the Unity logo, 'Manual' (which is highlighted in blue), 'Scripting API', a search bar, and a language selector set to 'English'. Below the bar, the version is listed as 'Version: 2022.3'. The main content area has a sidebar on the left titled 'Unity Manual' with various categories like 'Unity User Manual 2022.3 (LTS)', 'New in Unity 2022 LTS', etc. The main content area displays the 'Introduction to Rigidbody 2D' page, which includes a breadcrumb trail ('Unity User Manual 2022.3 (LTS) / 2D game development / Physics 2D Reference / Rigidbody 2D / Introduction to Rigidbody 2D'), back and forward navigation arrows, and a 'SWITCH TO SCRIPTING' button. The page content discusses what a Rigidbody 2D component does and how it differs from its 3D counterpart.

2. Unity Script Reference

Overview of Scripting API

The screenshot shows the Unity Script Reference interface. At the top, there's a navigation bar with the Unity logo, 'Manual' (which is highlighted in blue), 'Scripting API', a search bar, and a language selector set to 'C#'. Below the bar, the version is listed as 'Version: 2022.3'. The main content area displays the 'Rigidbody2D' class page. It includes a sidebar with a list of related classes like ResourceRequest, Resources, ResourcesAPI, Rigidbody, and others. The main content area has a 'Description' section explaining that the Rigidbody2D class provides 2D physics functionality for 2D sprites. It also lists 'See Also' links for other related classes like Rigidbody, SpriteRenderer, Collider2D, and Joint2D. A 'Properties' table is shown at the bottom, with one entry for 'angularDrag'.



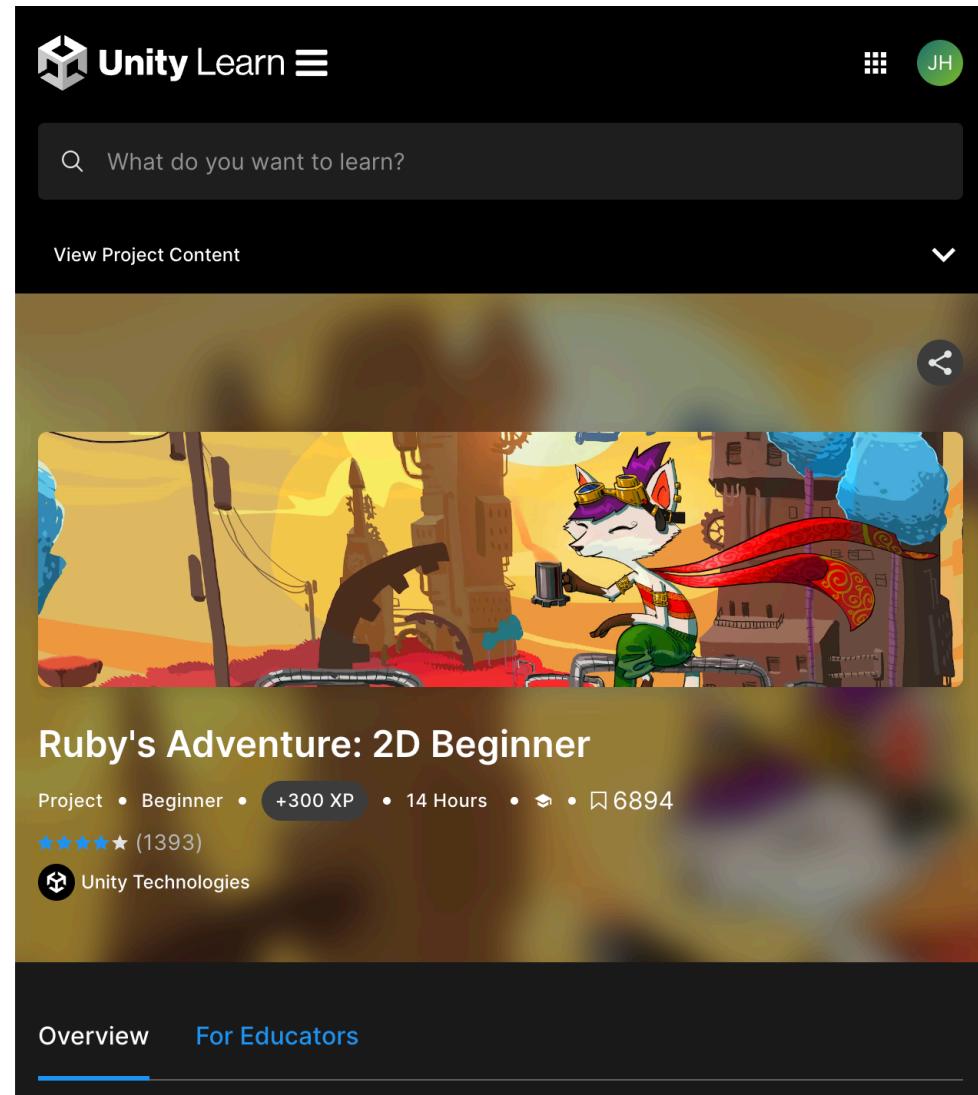
Unity Learn

Unity Learn:

- Unity's official tutorials
- All "difficulty" levels
- (New) systems explained
- Up-to-date

Ruby's Adventure: 2D Beginner

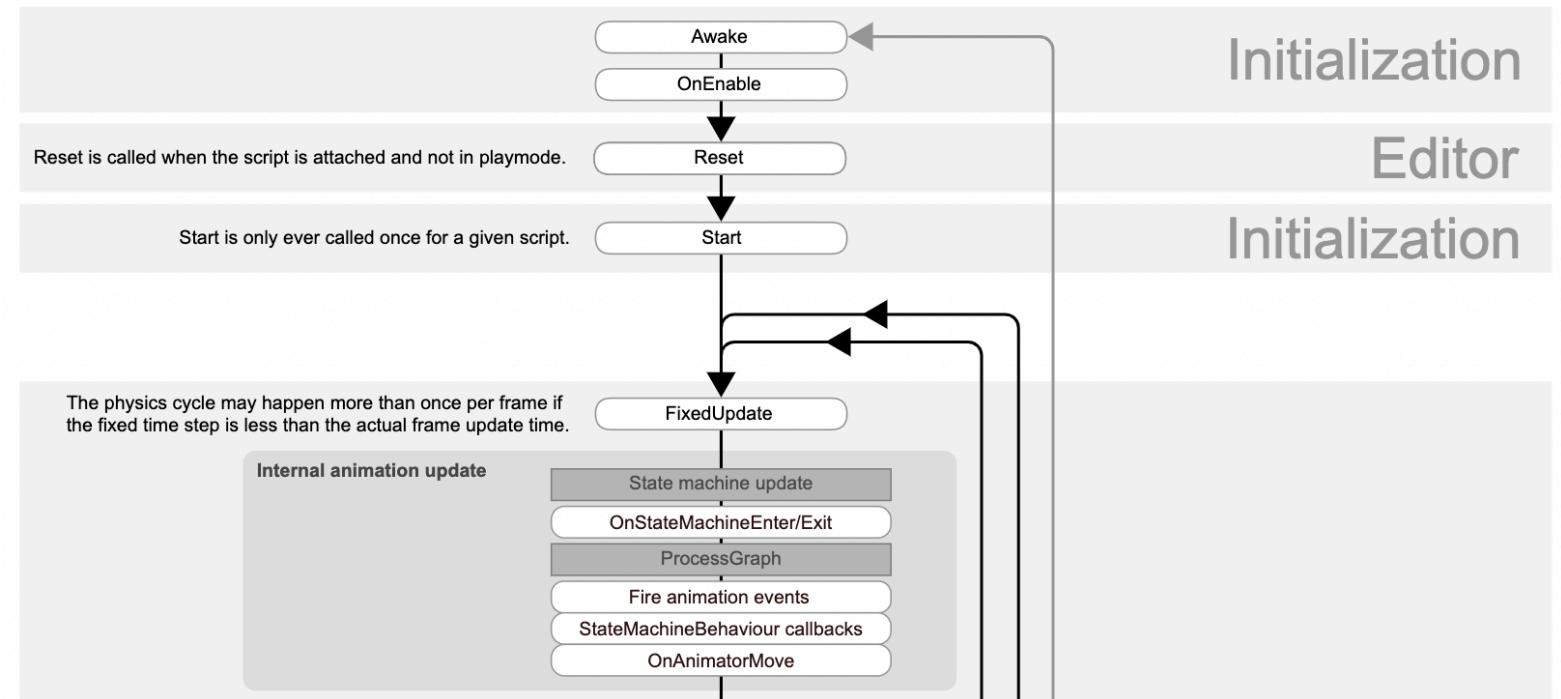
Recommended tutorial for 2D game dev [\[link\]](#) ->



Unity's Script Lifecycle

Most used, in order of execution:

- Awake
- OnEnable
- Start
- FixedUpdate
- OnCollision
(+Enter/Exit/Stay)
- Update
- OnDisable



From: [Unity Manual - Order of execution for event functions](#) [link]



Part 2

- Unity's Component-based workflow
- Creating and Destroying GameObjects
- Using Prefabs [TODO]
- Referencing (Game)Objects



Unity's Component-based workflow



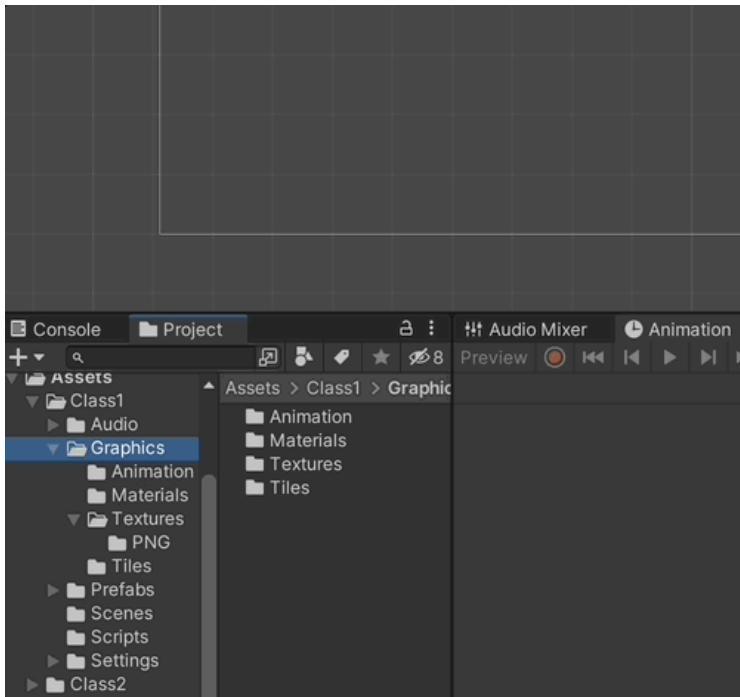
- Inspector shows all components per GameObject
(← e.g. Transform, Player.cs script, CapsuleCollider2D, Rigidbody2D).
- Every component can be referenced by script
- Every GameObject can be referenced by a component on it



Creating and Destroying GameObjects

Method 1: Drag into Scene view

Tip: dragging a Sprite into a scene will create a new game object



Method 2: By script

tip: keep lists of things you've spawned

```
public Enemy enemyPrefab;
private List<Enemy> spawnedEnemiesList;

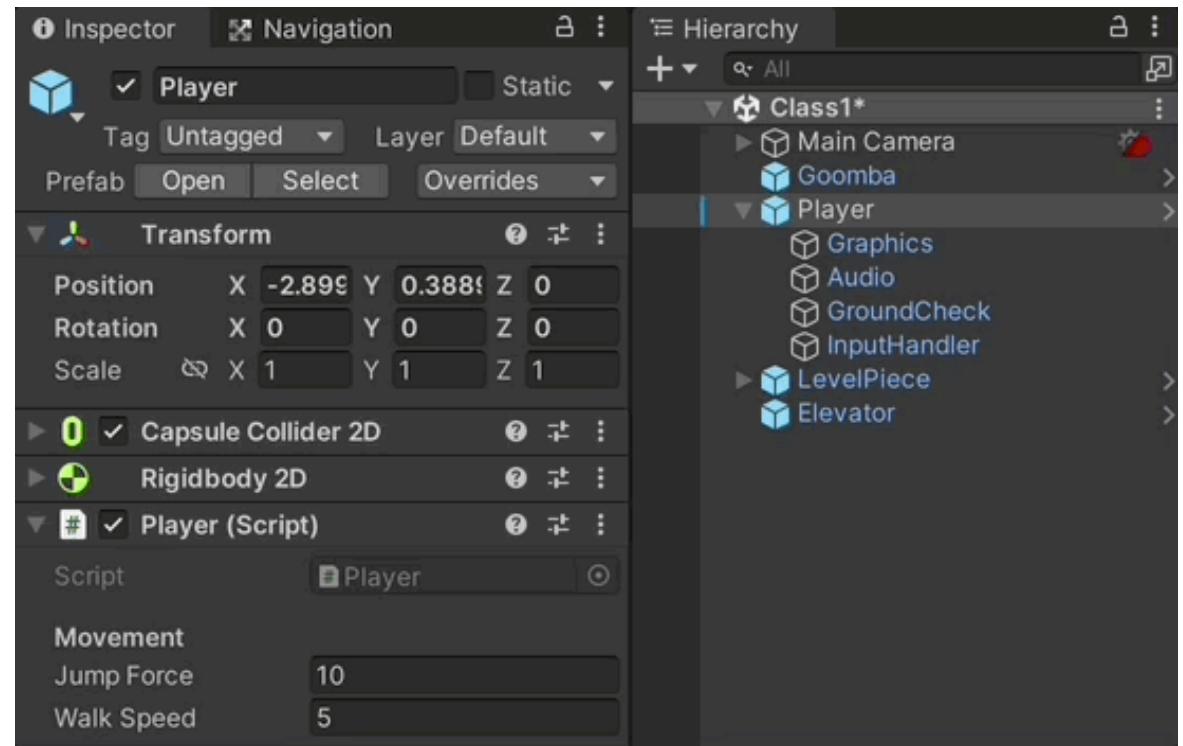
public void SpawnEnemy()
{
    Enemy newEnemy = Instantiate(enemyPrefab);
    spawnedEnemiesList.Add(newEnemy);
}

public void DestroySpecificEnemy(Enemy specificEnemy)
{
    if (spawnedEnemiesList.Contains(specificEnemy)){
        spawnedEnemiesList.Remove(specificEnemy);
        Destroy(specificEnemy);
    }
}
```



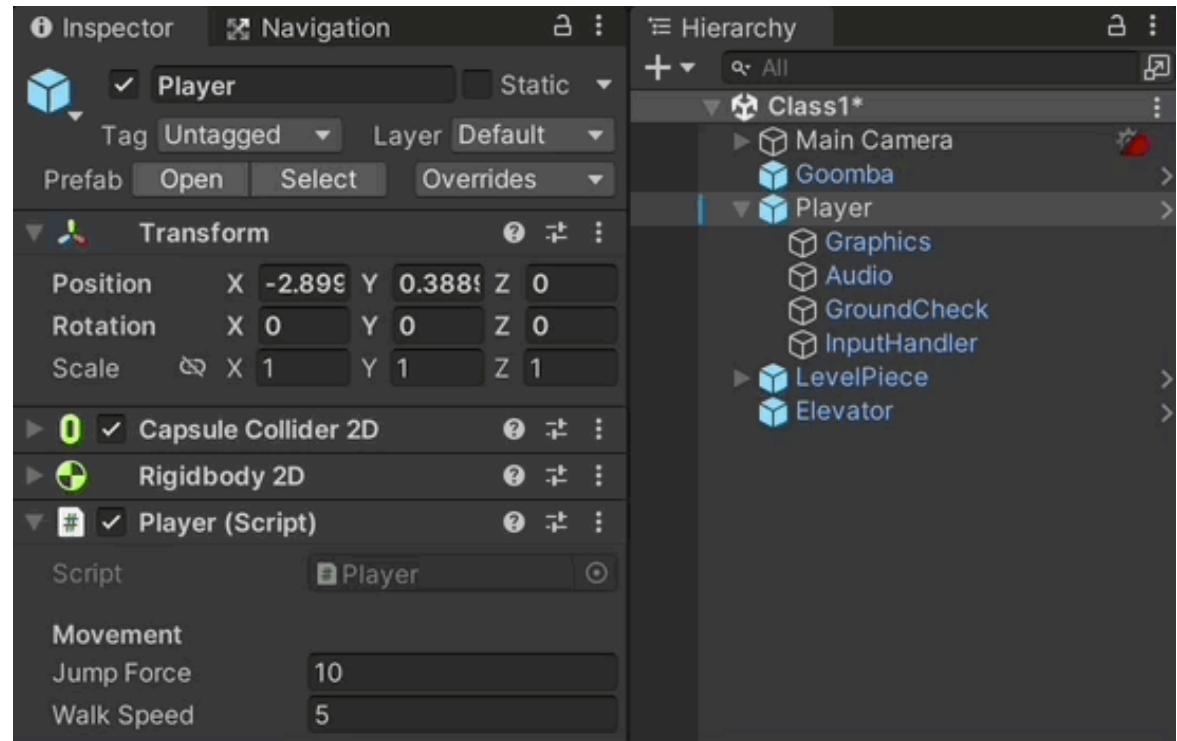
Using Prefabs

- Prefab editing
- Dragging prefabs into a scene



Referencing (Game)Objects – Public variables

- If you make a variable public, it will show up in the Inspector view
- Use this for referencing other objects
- **Warning:** Don't drag scripts from Project view into the Inspector view!
Always use scripts that are already in the scene.



Referencing (Game)Objects - by Component

- Player ***is*** a GameObject
- Player ***has*** a Player.cs script on it (which is a component)

You can reference a GameObject by the GameObject class, or by one of the components that the GameObject has.

