UNIVERSITÀ DEGLI STUDI
DI MILANO

Unity 101 – Recap of Basics

A Quicksilver Guide to the Less Obvious Stuff

Dario Maggiorini (dario@di.unimi.it)

PDM9
Playlab fOr inNovation in Games

What Unity is NOT

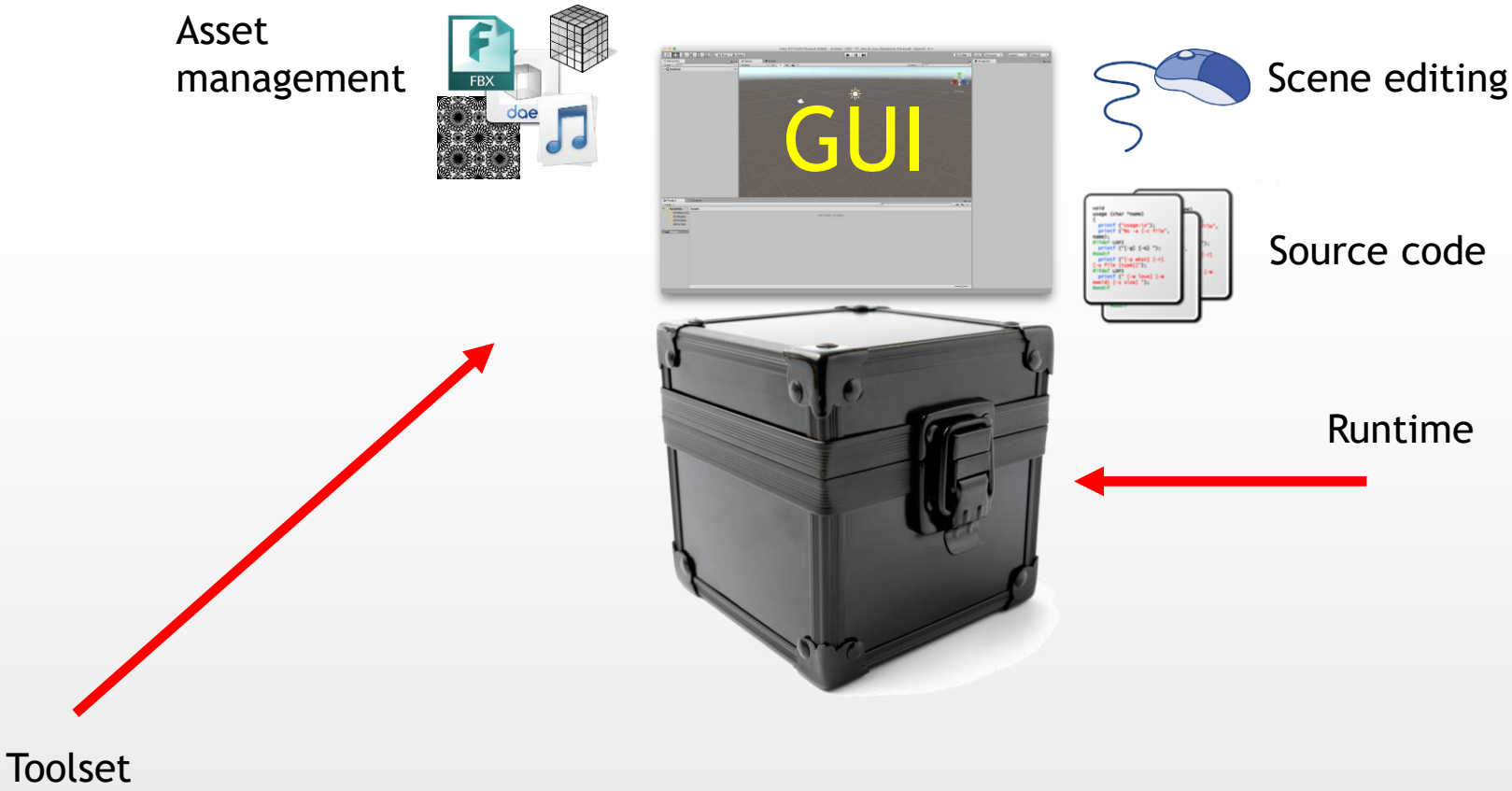
- A 3D editor

- A revision control system

- A development environment

- … and a game engine

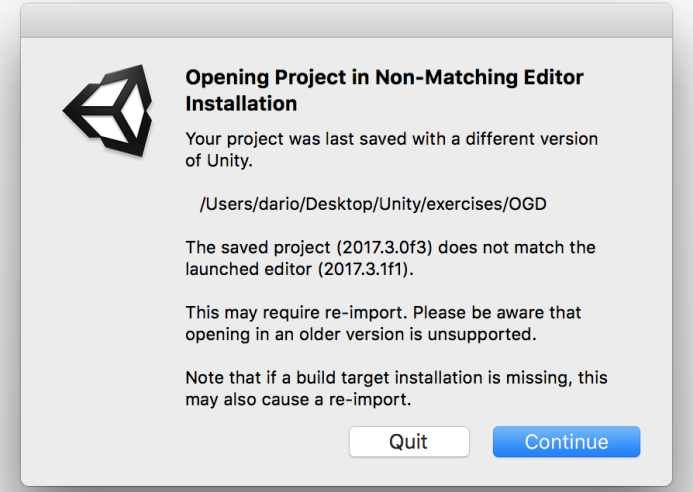
What IS Unity

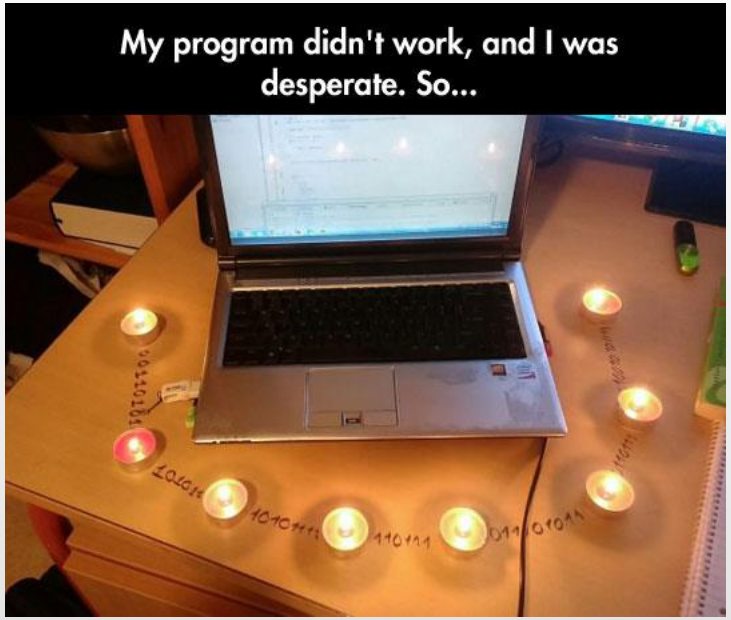


What IS Unity

A Warning About Versions

- Never, EVER, change your unity version during development
- Even minor versions can insert (subtle) incompatibilities and break all your work
  - **Mind about keeping a backup!**

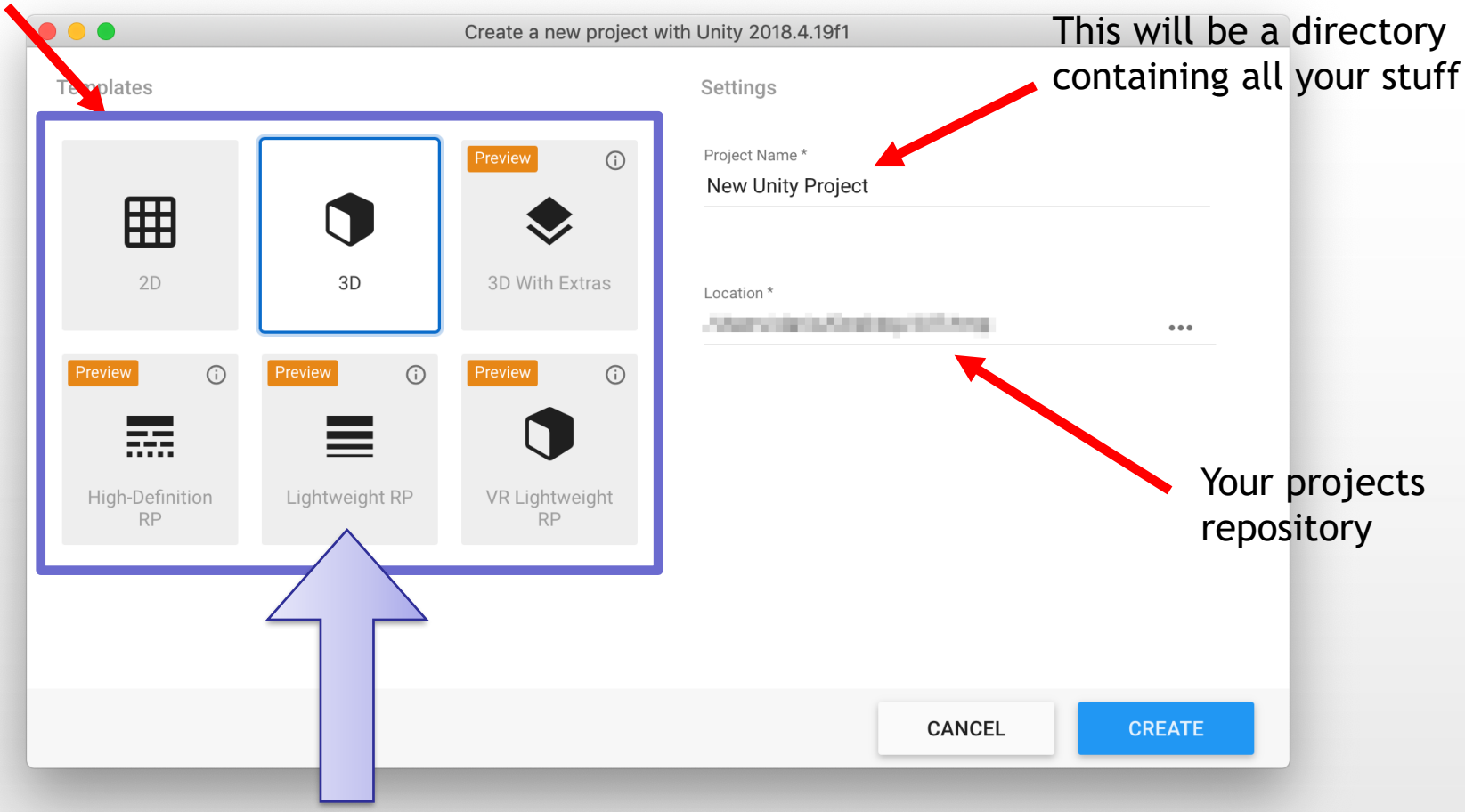




Starting Up

(Your hub might look different depending on your version)

Depends on your game

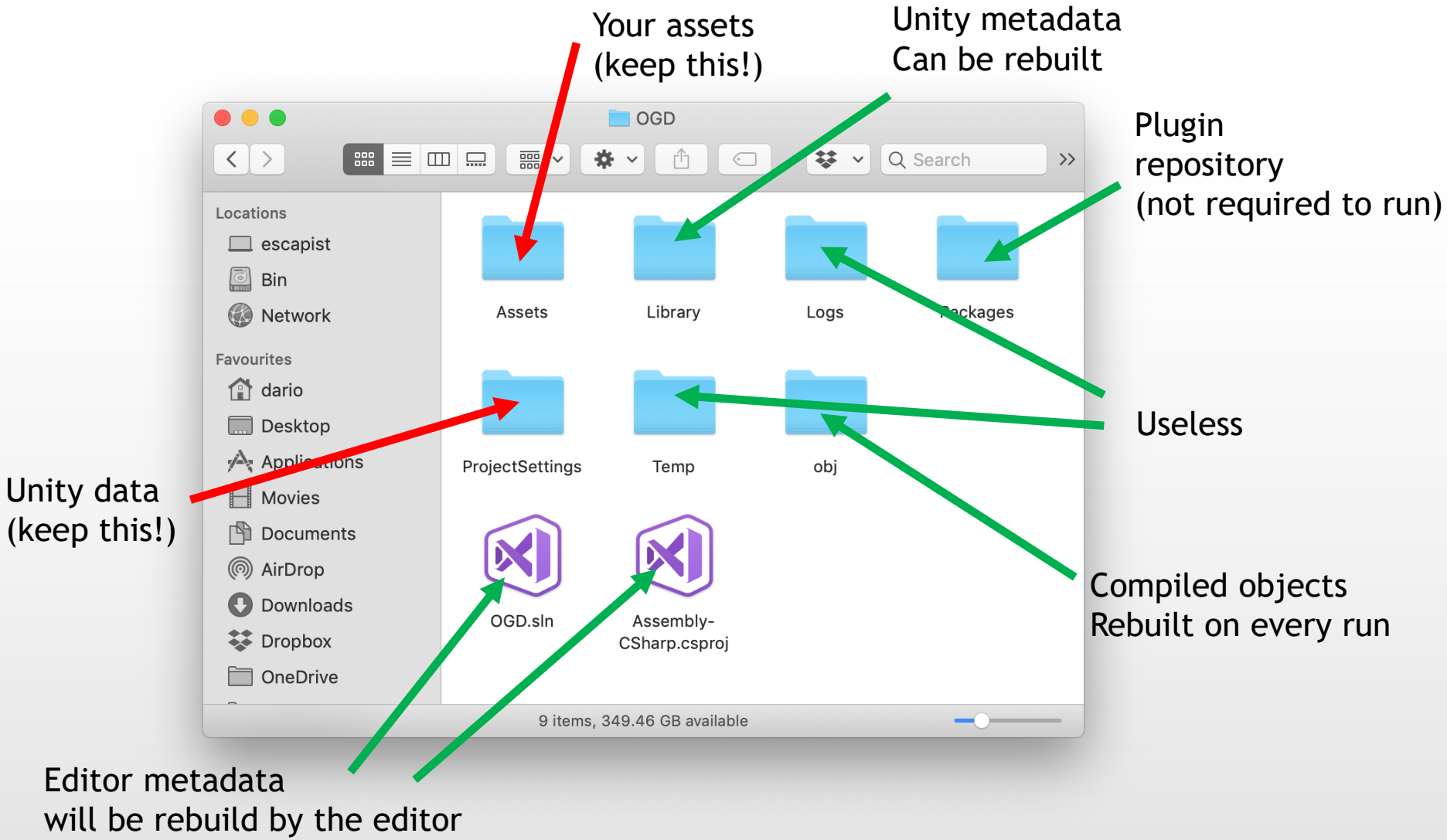


This will be a directory containing all your stuff

Your projects repository

These are just presets.
You can manually rebuild them from the editor

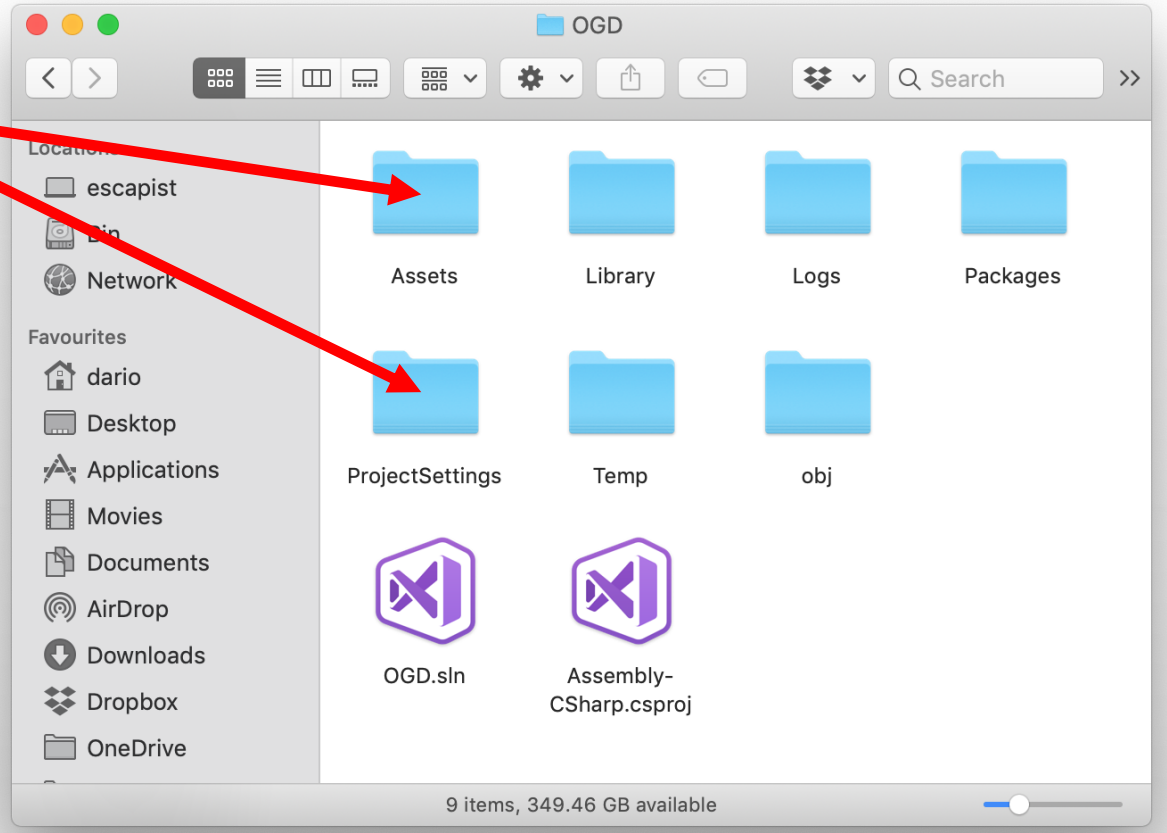
Files and Folders



Files and Folders

This is all you need to save/backup/share of your project (GIT included!)

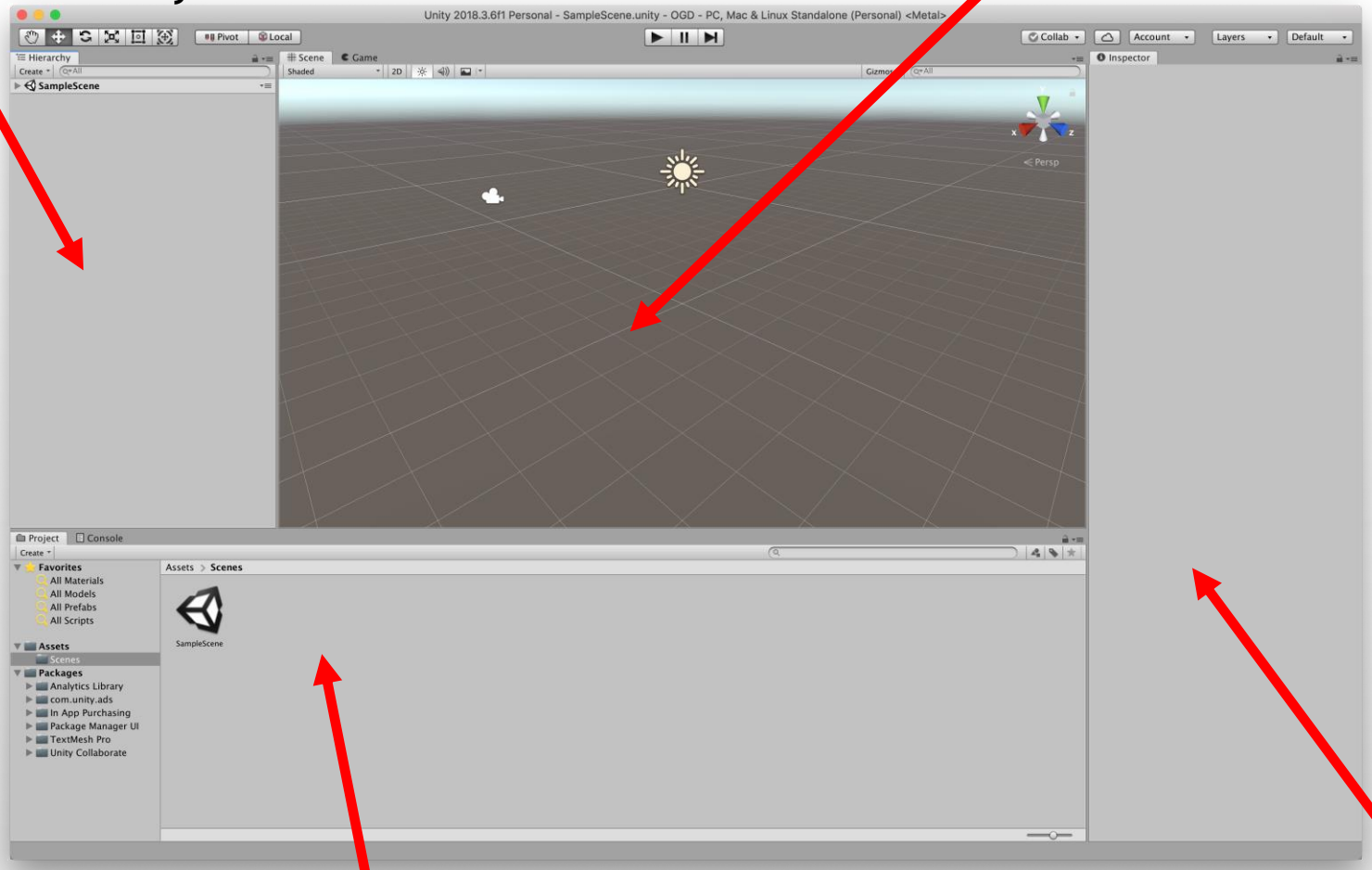
Moreover, they will be only around 10% of the total storage space



Unity Interface Anatomy

Scene hierarchy

Scene window



Asset database

Objects inspector

Defining Your Own ~~Components~~ Evolution

UK Rulez!

- A component is a class extending *MonoBehaviour*
- This class, once compiled, becomes an asset that can be added to any gameobject
  - A panel will be shown in the inspector through introspection
  - Public fields will become the component UI elements
  - Methods will be invoked when needed
- No. There is no “main” function here!
- You can do any operation and access any data, component, and gameobject as you do with the unity GUI
  - ... and much more!

An Empty Script

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

public class Rotation : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
}
```

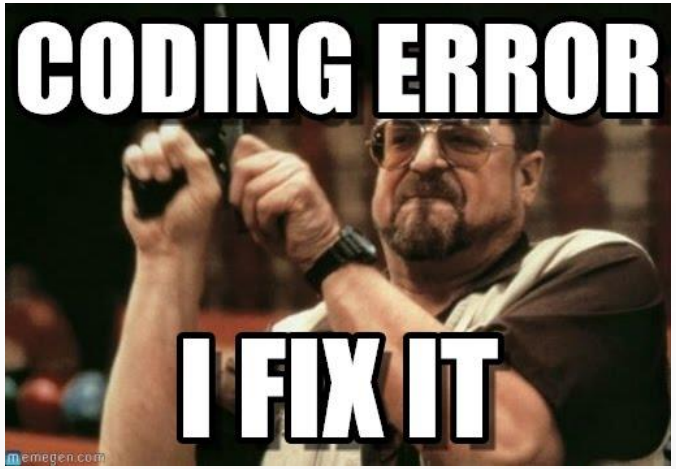
- Start() will be called when object is first placed in the scene
  (even if you cannot see it!)
- Update() will be called at every frame

Not the same “frames” as on the screen!

A frame, in this case is the time required by Unity to update all gameobects in the scene. That is to say, calling all the update() functions and restart the cycle

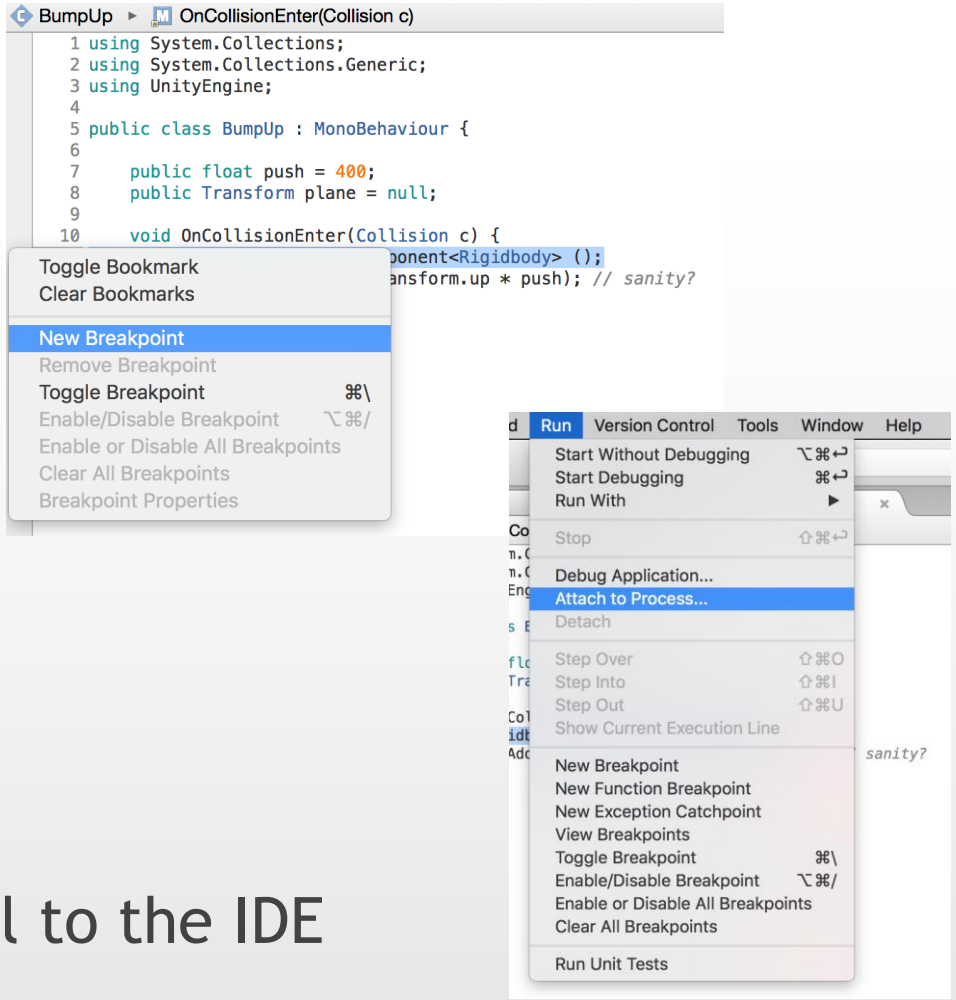
No Software is Bug Free

- But, unfortunately, the console output of Unity is also very crappy
  - Delayed flush
  - Lot of clutter
  - Difficult to read
    - There is the stacktrace in every message
  - **Uses a lot of CPU**



But We Can Debug!

- Add a breakpoint with your IDE
- Attach debugging to the unity process
- Start the game in Unity and proceed as usual
- Unity will pause and give control to the IDE upon reaching the breakpoint



The Single Thread Menace

- An important contract holds:
  Start() and Update() are not interruptible!

- This is to ensure consistency of shared resources
  - Did you care about that in the operating system class?

- If you get stuck in an Update(), any Update(), Unity will freeze
  - You will be required to kill and restart the whole environment !

Performing Side Tasks

- You can spawn a secondary thread (it is C# after all)
  … but the secondary thread will NOT be able to access any scene data (to guarantee consistency again)
  - Yes, this sucks!

- You can write a stateful *Update()*
  - But then you must spawn a gameobject for every task

- … or you can delegate the task to a coroutine

Coroutine

- It is just like a method which can “get suspended” and restart later without bothering any *Update()*

- Typical usage are tasks to be performed only once, have a delayed effect, or need a specific execution rate
  - Steering
  - Jumping
  - Counting
  - Fading

Fade on Click

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

public class BumpNFade : MonoBehaviour {

    public float push = 400f;
    public float fadeTime = 3f;

    void OnCollisionEnter(Collision c) {
        Rigidbody rb = GetComponent<Rigidbody> ();
        rb.AddForce (c.gameObject.transform.up * push);
    }

    void OnMouseDown() {
        StartCoroutine (Fade ());
    }

    // Remember to set te rendering mode of the shader to "fade"

    IEnumerator Fade() {
        Material m = GetComponent<Renderer> ().sharedMaterial;
        Color c;
        for (float f = 1f; f >= 0; f -= 0.1f) {
            c = m.color; c.a = f; m.color = c;
            yield return new WaitForSeconds(fadeTime / 10f);
        }
        c = m.color; c.a = 1f; m.color = c; // danger here!
        Destroy (gameObject);
    }
}
```

Will be called by the runtime when we click on the object

Hidden traps here!

Set transparency to appropriate value

Suspend and get rescheduled in 0.3 seconds (3/10)

When the last statement is reached, the coroutine is over and all resources are freed

Happy Hacking

Dario Maggiorini (dario@di.unimi.it)