Unity3D - General Concepts

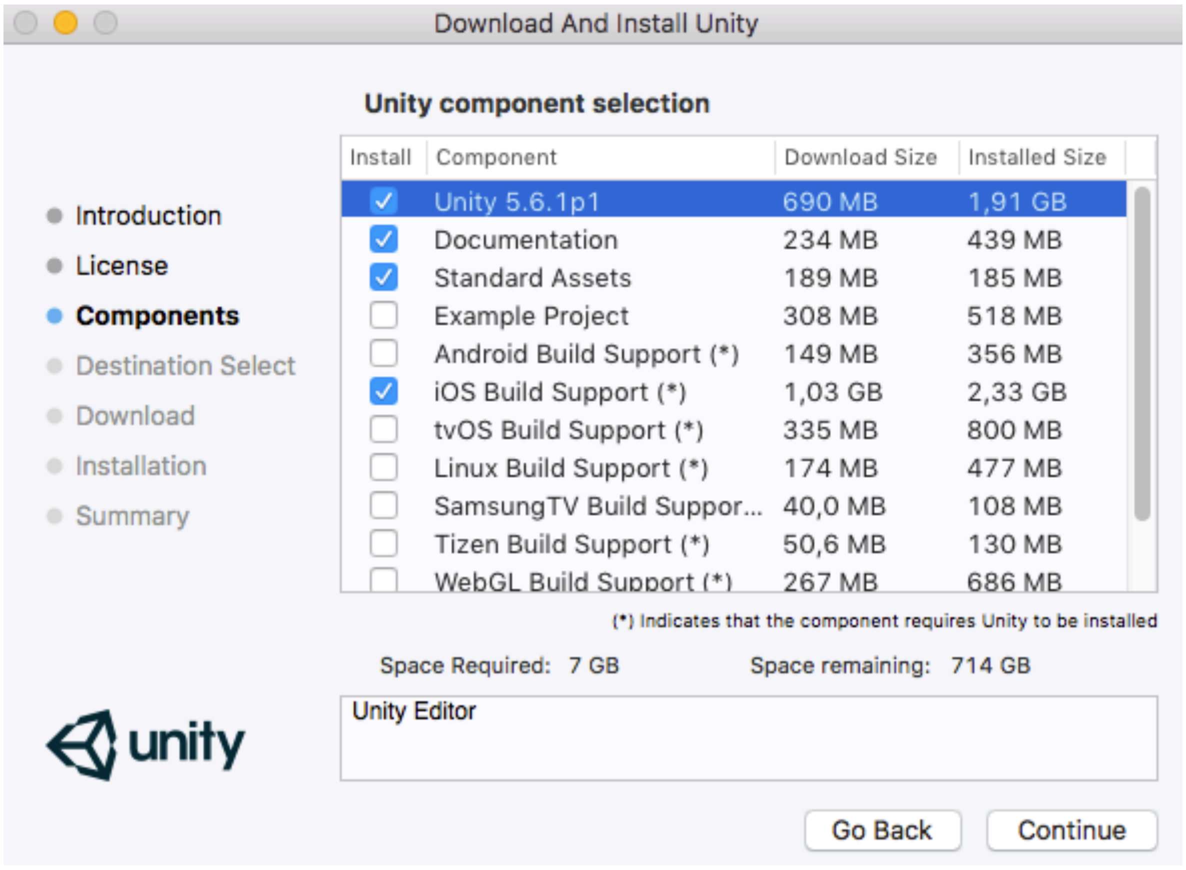
https://unity3d.com/

Unity is a game development platform.
Use Unity to build high-quality 3D and 2D games, deploy them across mobile, desktop, VR/AR, consoles or the Web, and connect with players and customers.
The first version you may probably want to try out is the Personal Edition free of charge. The Professional Edition is provided for a price.
The installation process on my mac was straightforward.
The native language of Unity is C#, but you can also use JavaScript and Boo.

Installation of tools

- Install the newest patch version of Unity 5.6.1p1

http://beta.unity3d.com/download/74c1f4917542/Uni...



Download And Install Unity

- Introduction
- License
- **Components**
- Destination Select
- Download
- Installation
- Summary

Unity component selection

Install	Component	Download Size	Installed Size
✓	Unity 5.6.1p1	690 MB	1,91 GB
✓	Documentation	234 MB	439 MB
✓	Standard Assets	189 MB	185 MB
☐	Example Project	308 MB	518 MB
☐	Android Build Support (*)	149 MB	356 MB
✓	iOS Build Support (*)	1,03 GB	2,33 GB
☐	tvOS Build Support (*)	335 MB	800 MB
☐	Linux Build Support (*)	174 MB	477 MB
☐	SamsungTV Build Suppor...	40,0 MB	108 MB
☐	Tizen Build Support (*)	50,6 MB	130 MB
☐	WebGL Build Support (*)	267 MB	686 MB

(*) Indicates that the component requires Unity to be installed

Space Required: 7 GB Space remaining: 714 GB

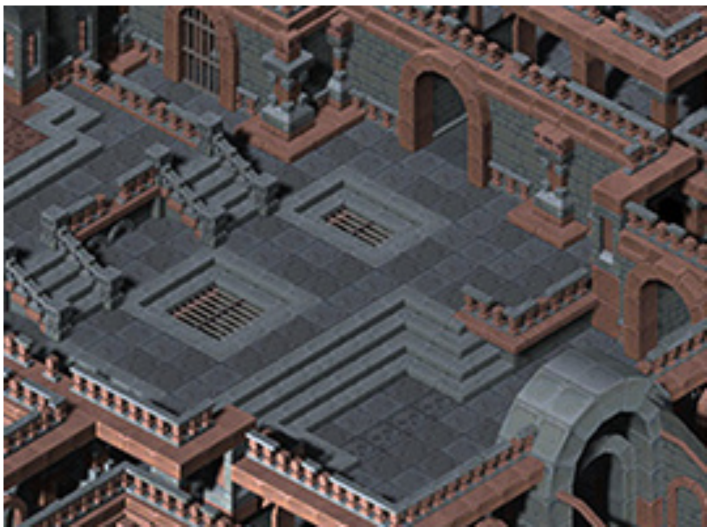
Unity Editor

2D/3D Games

- A game *Scene view* mode can be 2D or 3D
- Sprites are 2D object which stay on a plane. Textures are polygons used to change the appearance of a 3d object. A texture must be attached to a

material, and the material to a game object (e.g a plane). For 2D games, use sprites

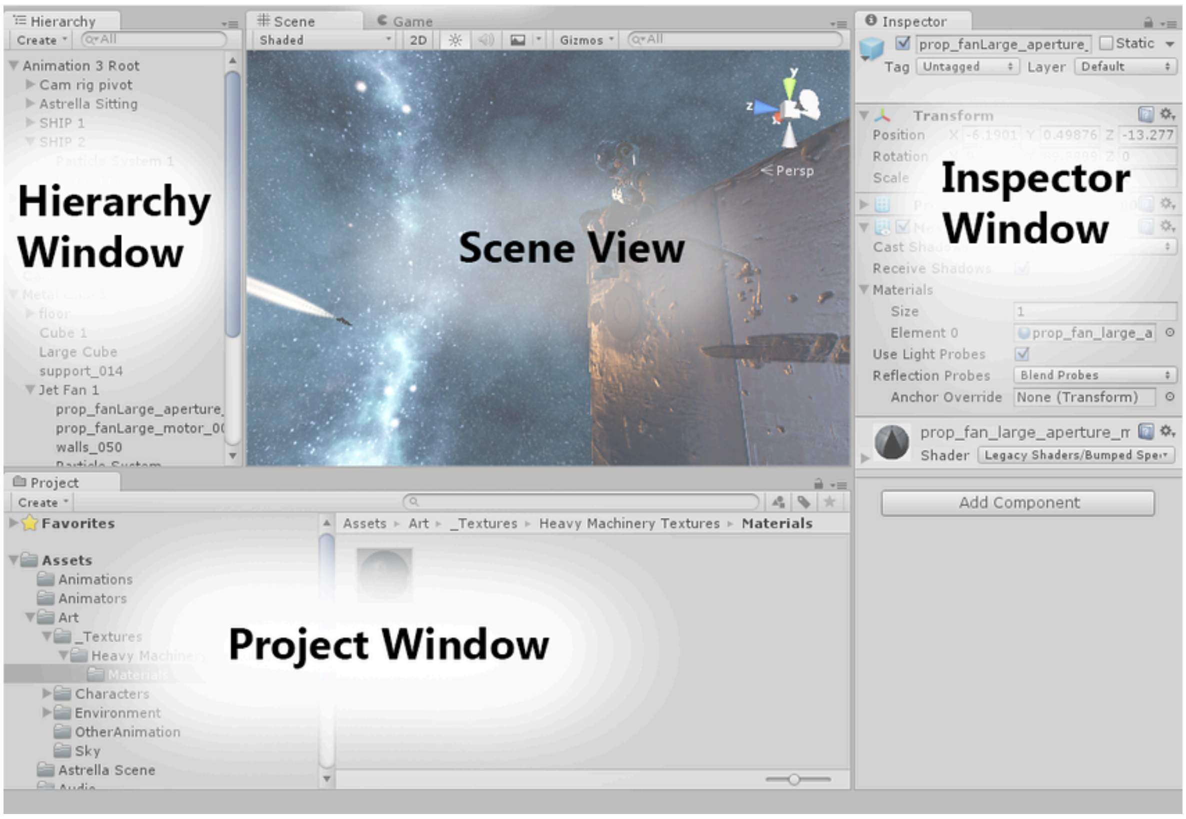
- In 3D games the camera projection mode can be *Perspective* (objects appear larger on screen as they get closer to the camera) or *Orthographic* (also known as 2.5D)



- Some 2D games use 3D geometry for the environment and characters, but restrict the *gameplay* to two dimensions (e.g. Mario)
- There is also a 2D game with perspective camera (cardboard theatre) that gives the parallax scrolling effect

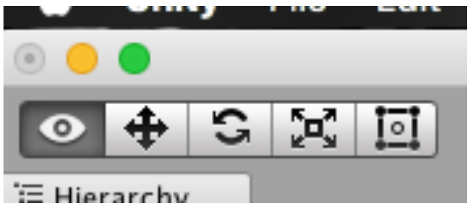
Editor

- The editor has the following main sections



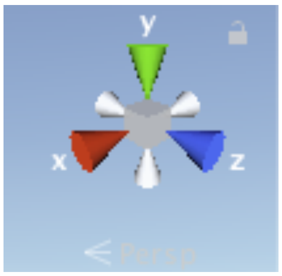
- Project Window: displays your library of assets that are available to use in your project
- Hierarchy Window: is a hierarchical text representation of every object in the scene. Each item in the scene has an entry in the hierarchy, so the two windows are inherently linked. The hierarchy reveals the structure of how objects are attached to one another. Objects are presented in a parent-child-descendants relationship. The Hierarchy Window is a hierarchical text representation of every object in the scene. Each item in the scene has an entry in the hierarchy, so the two windows are inherently linked. The hierarchy reveals the structure of how objects are attached to one another
- Unity can import any 3D model. Also it has primitives like Cube, Sphere, Capsule, Cylinder, Plane and Quad, which can be added to the scene from GameObject > 3D Object.

Scene View



- The five transform buttons shown above can be enable by keys 'Q,'W','E','R','T'
- Q: drag the scene
- W: translate along any axis

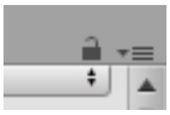
  - E: rotate around any axis
  - R: scale up or down along any axis
  - T: does more transformations of 2D objects
- Click and hold the right mouse button to enter the flight through mode. Use W,A,S,D keys to move the camera
  - W: zoom in
  - S: zoom out
  - A: move left
  - D: move right
  - Q: move down
  - E: move up
- The following object is called Gizmos



Click the different axes to align the camera along it
By clicking on the center of the Gizmos you can change the camera mode

Inspector View

When you need to drag something onto the Inspector window, you may need to temporarily lock the window before to do that. You can accomplish that by clicking the lock symbol

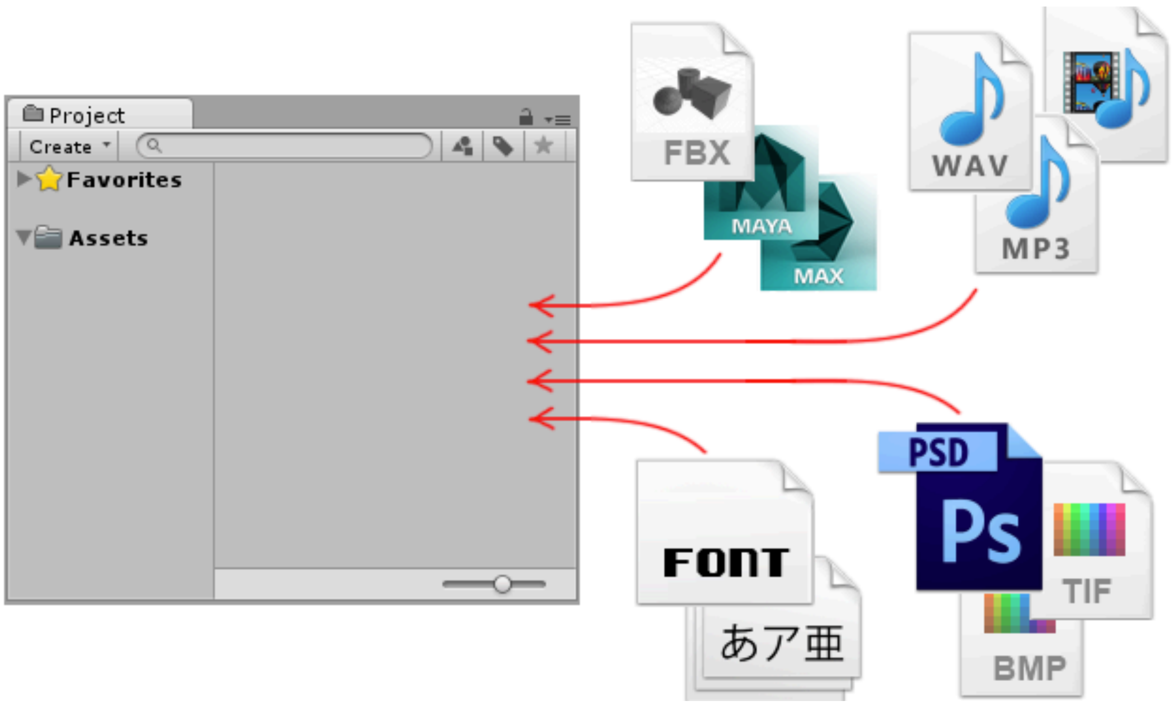


Transform

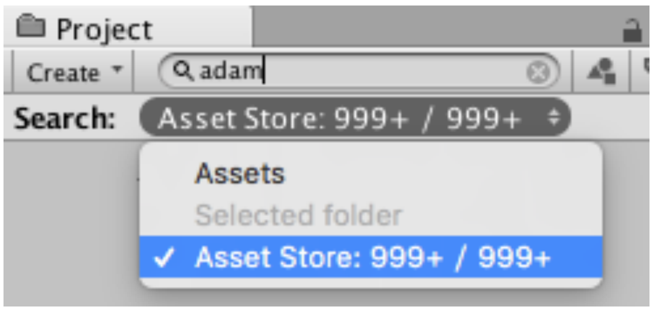
- *Transform* is a component for every GameObject. It specifies Position, Rotation and Scale
- If the GameObject has no parent, its Transform is referenced to the real world, otherwise it is relative to the parent's Transforms

Assets

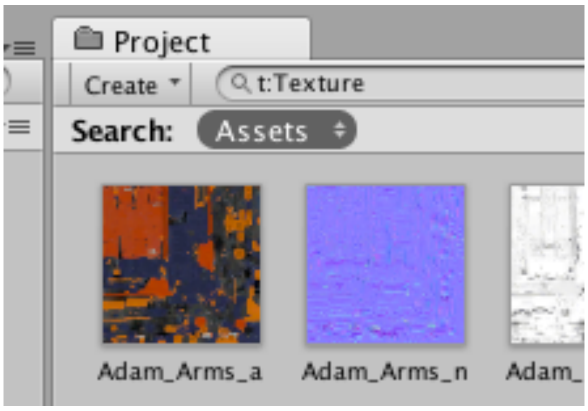
- An *asset* is representation of any item that can be used in your game or project and has a corresponding persistent file in the filesystem



- *Asset packages* are pre-made bundle of contents such as images, styles, lighting effects, in-game character controls, etc.
- The simplest way to safely move or rename your assets is to always do it from within Unity's project folder. This way, Unity will automatically move or rename the corresponding meta file.
- If you want to bring collections of assets into your project, you can use Asset Packages
- You can save your 3D files from most common 3D software packages in their native format (eg, .max, .blend, .mb, .ma) into your Assets folder, they will be imported by calling back to your 3D package's FBX export plugin. This requires the 3D application to be installed on the same computer as Unity. Alternatively you can export as FBX from your 3D app into your Unity Projects folder.
- Unity natively imports Blender files. Save your .blend file in your project's Assets folder. When you switch back into Unity, the file is imported automatically. If you modify your .blend file, Unity will automatically update whenever you save.
- The Unity Asset Store is home to a growing library of free and commercial assets. The location of the downloaded asset files is ~/Library/Unity/Asset Store. To open the Asset Store from Editor go to *Window > Asset Store*.
- Check available assets from the Store from within Project window



- Search function can be filtered by type



- More assets can be organised, imported and exported as Assets Packages

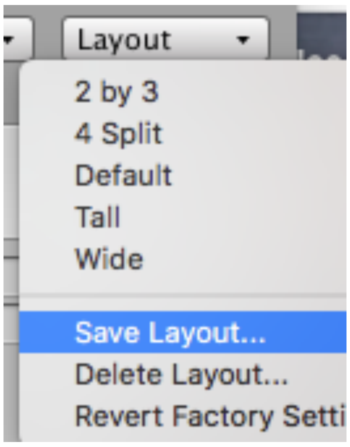
Prefabs

- *Prefabs* are preconfigured templates for game objects that you create in the scene and store in the project. They can then be instantiated or cloned during the game. They are used for everything from rockets to enemies, to procedural levels.
- To make a prefab simply create your game object in it's desired configuration in your scene, from whatever components you need, and then drag it into the Prefabs folder. Ten it can then be deleted from the scene.
- To edit a prefab you can either select it in the project window, and adjust properties in the inspector. Or drag an instance of it back in to the scene, edit it in the inspector, and then press the *Apply* button at the top. This saves changes back in to the prefab and you can then remove the instance from the scene yet again.
- If you have many instances of a prefab, and you make edits to one of them, and decide you would like others to be the same you can hit *Apply*. The other instances will inherit this update from the prefab's settings. Likewise if you make a change to one of your instances and you decide you don't like it anymore you can revert to the settings of the prefab by clicking the *Revert* button at the top.

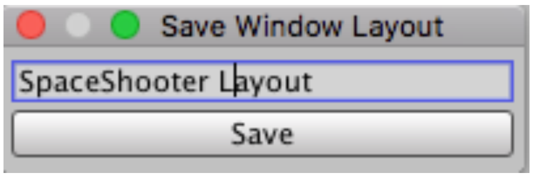
Layout

There are several built-in options for the editor layout.
However, depending on your need, you may prefer a different one.
For example, in editing the Space Shooter game project, the aspect ratio of the screen is portrait. Then the “2 by 3” layout could be improved by dragging the Game tab side by side withe the Scene tab

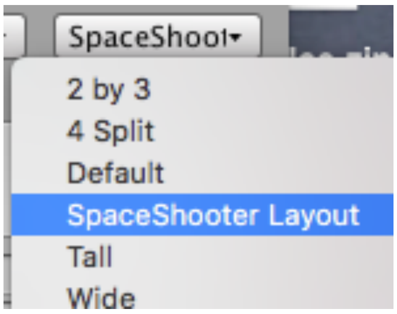
Once you have done it, click *Save Layout*



then assign a name to your custom layout



This will make your layout appearing in the list



Layers

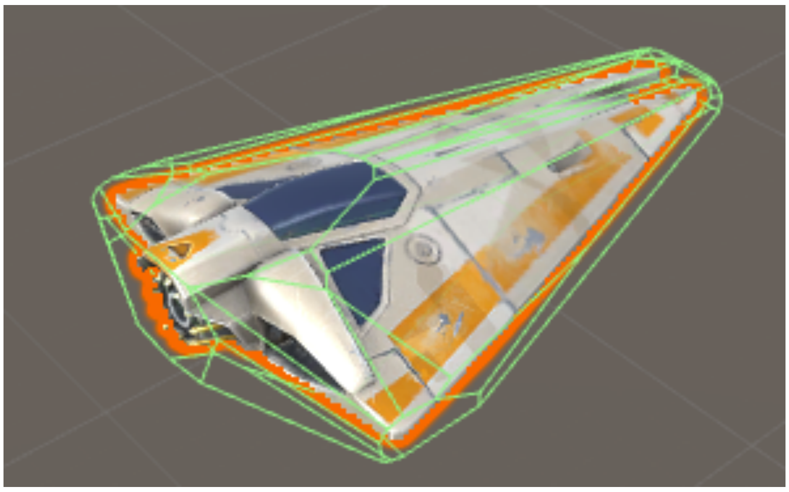
Layers are most commonly used by Cameras to render only a part of the scene, and by Lights to illuminate only parts of the scene. But they can also be used to selectively ignore colliders or to create collisions.

Tags

Tags are marker values that that you can use to identify objects in your project

Physics

- A Collider is kind of like a cage that wrap up the Rigidbody and allows to detect collisions
- There are several Collider shapes you can adapt to the RigidBody, Sphere Collider, Box Collider, Capsule Collider, and the more advanced (and computationally heavy) Mesh Collider



Scripts

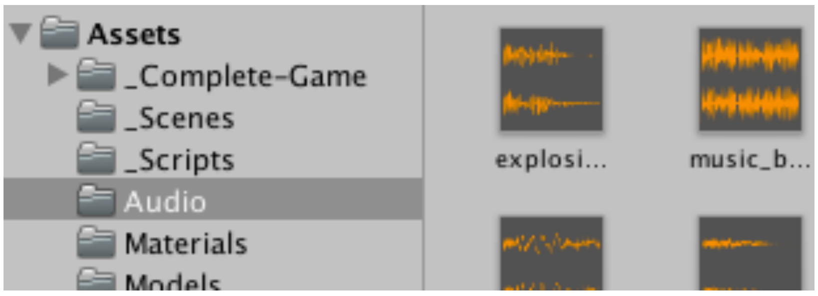
- *MonoDevelop-Unity* is the editor bundled with Unity 3D, but you can use different editors as well
- *MonoBehaviour* is the base class from which every Unity script derives
- *MonoBehaviour.Awake()* is called when the script instance is being loaded, before the *Start()*
- *MonoBehaviour.Start()* is called once at the very first frame of the game
- *MonoBehaviour.FixedUpdate()* is called every fixed framerate frame, when the physics equations are updated. FixedUpdate should be used instead of Update when dealing with Rigidbody components. For example when applying a force to a rigidbody

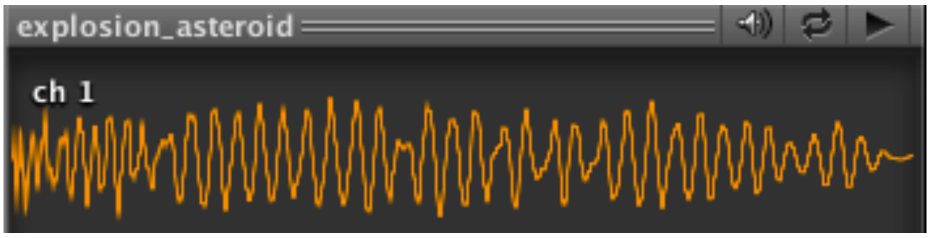
Audio

In Unity there are three main audio components:

- Audio clip: audio files
- Audio source: object's component within the scene which play audio clips
- Audio listener:

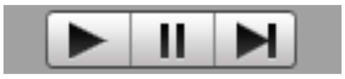
Find audio clips in the *Assets/Audio* folder. Select a audio clip and from the Inspector and Preview window change audio parameters, see the waveform, play the track.





Play in the editor

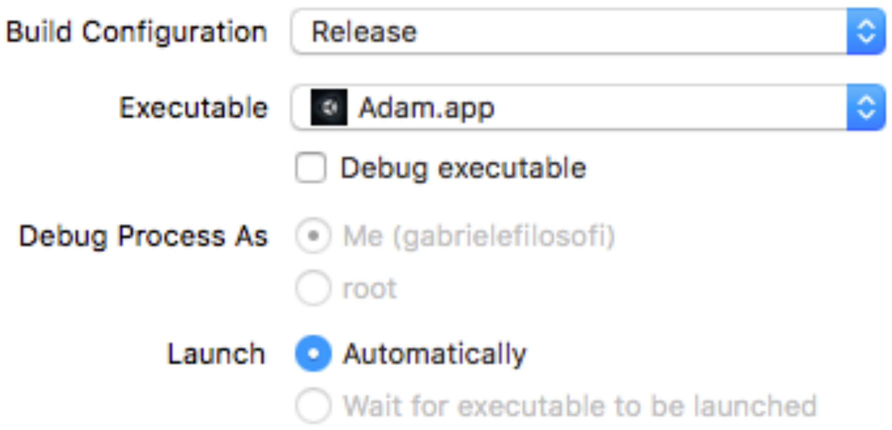
to run the game in the Unity Editor use the buttons in the toolbar.



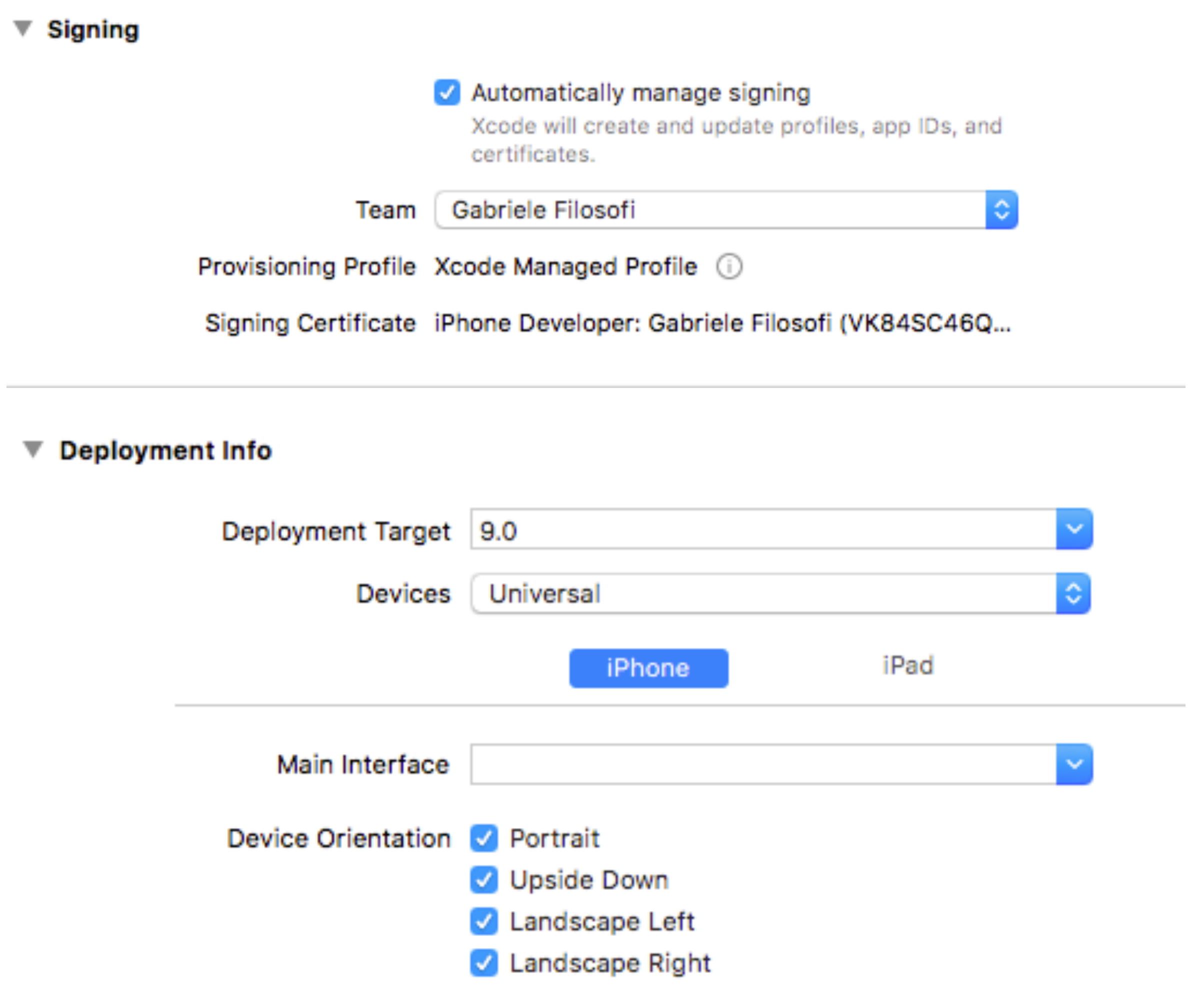
Or use the keyboard:
- Cmd+P: play/stop
- Cmd+Shift+P: pause/restart
- Cmd+Opt+P: step

Build

- Under File > Build Settings, select the desired target platform
- Click Switch Platform button
- Click *Player Settings* button
- Set the Bundle Identifier, and any other relevant parameters
- Press the Build button
- If not already done, click the down arrow to create a folder Builds
- Set the name of the build product
- In the case of a iOS build open the *Unity-iPhone.xcodeproj* in Xcode
- Connect the iOS device to the Mac and select it inside Xcode
- Set the Team identifier, edit the Scheme to set the configuration, check the warnings window, etc.
- Clean, Build, Run the app



Build Configuration: Release
Executable: Adam.app
☐ Debug executable
Debug Process As: [✓ Me (gabrielefilosofi)] root
Launch: [✓ Automatically] Wait for executable to be launched



Signing

✓ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

Team: Gabriele Filosofi
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iPhone Developer: Gabriele Filosofi (VK84SC46Q...

Deployment Info

Deployment Target: 9.0
Devices: Universal

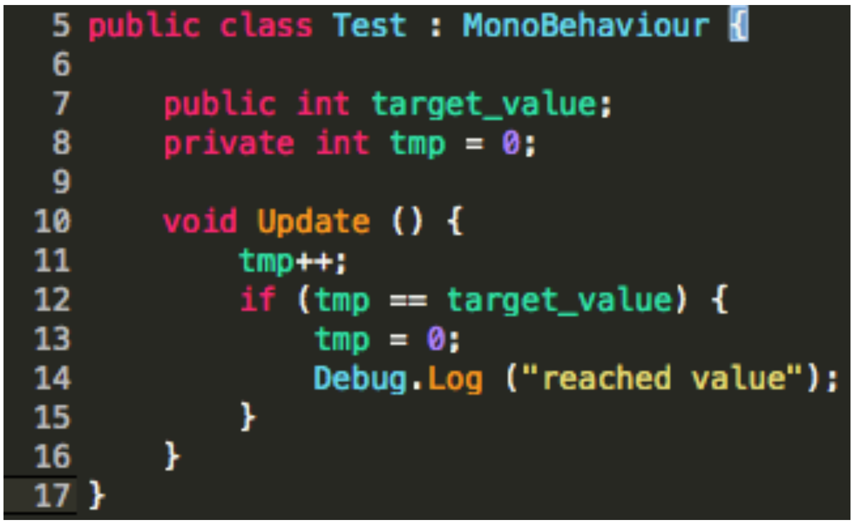
iPhone | iPad

Main Interface:
Device Orientation: ✓ Portrait
✓ Upside Down
✓ Landscape Left
✓ Landscape Right

Profiling

To assess the performance of the game we run the profiler in the Unity Editor.

- Create a test project with a scene, a GameObject and a script attached to the GameObject like this



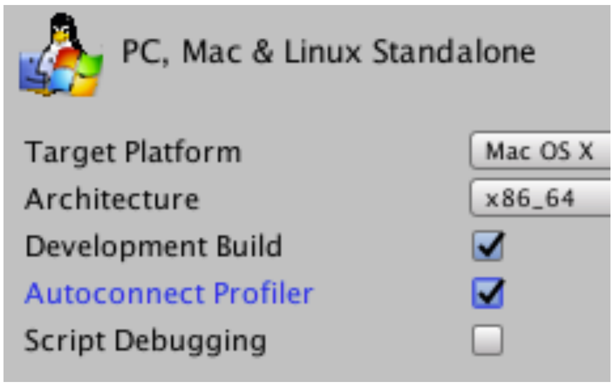
```
public class Test : MonoBehaviour {

    public int target_value;
    private int tmp = 0;

    void Update () {
        tmp++;
        if (tmp == target_value) {
            tmp = 0;
            Debug.Log ("reached value");
        }
    }
}
```

- Click *Window > Profiler*
- Delete all profiler except those which we care about, CPU Usage, Renderer

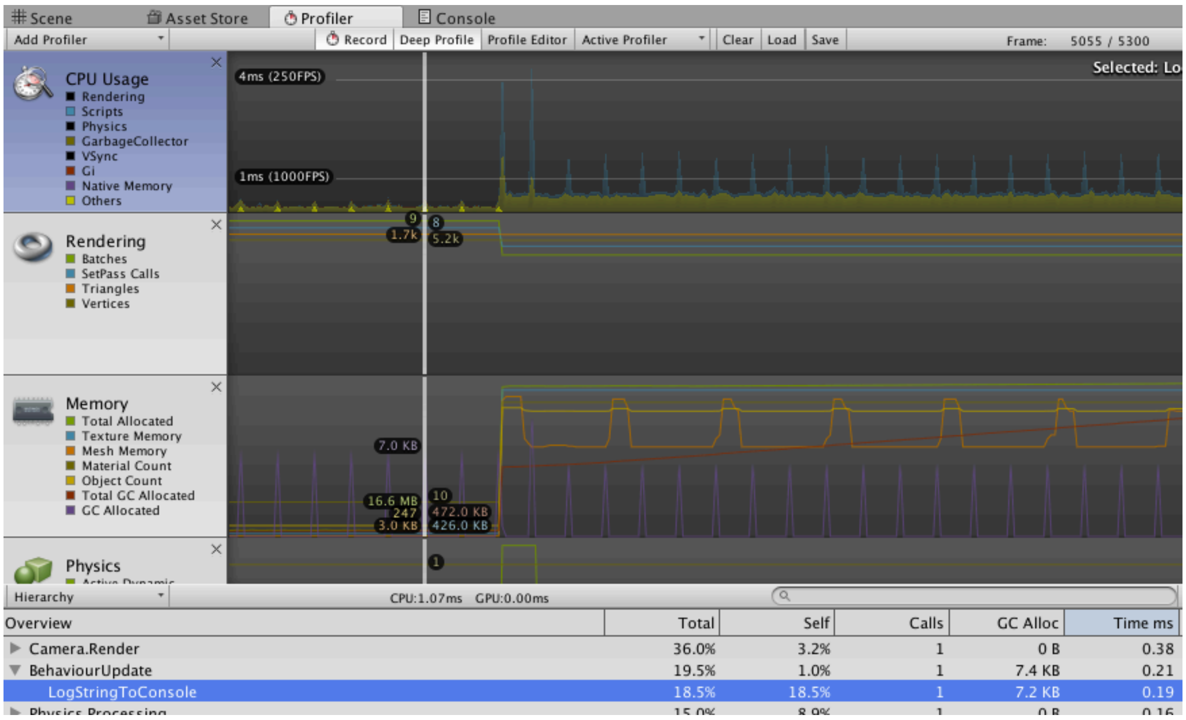
and Memory
- In *File > Build Settings*, switch to MacOS X, check “Development Build” and “Autoconnect Profiler”



PC, Mac & Linux Standalone

Target Platform	Mac OS X
Architecture	x86_64
Development Build	✓
Autoconnect Profiler	✓
Script Debugging	

- From the Unity Editor click *File > Run and Build*
- The application get started and the profiler starts grabbing data
- Quit the application
- Move the cursor to a memory peak
- Select the CPU Usage row and look at the process LogStringToConsole which takes much of the CPU time

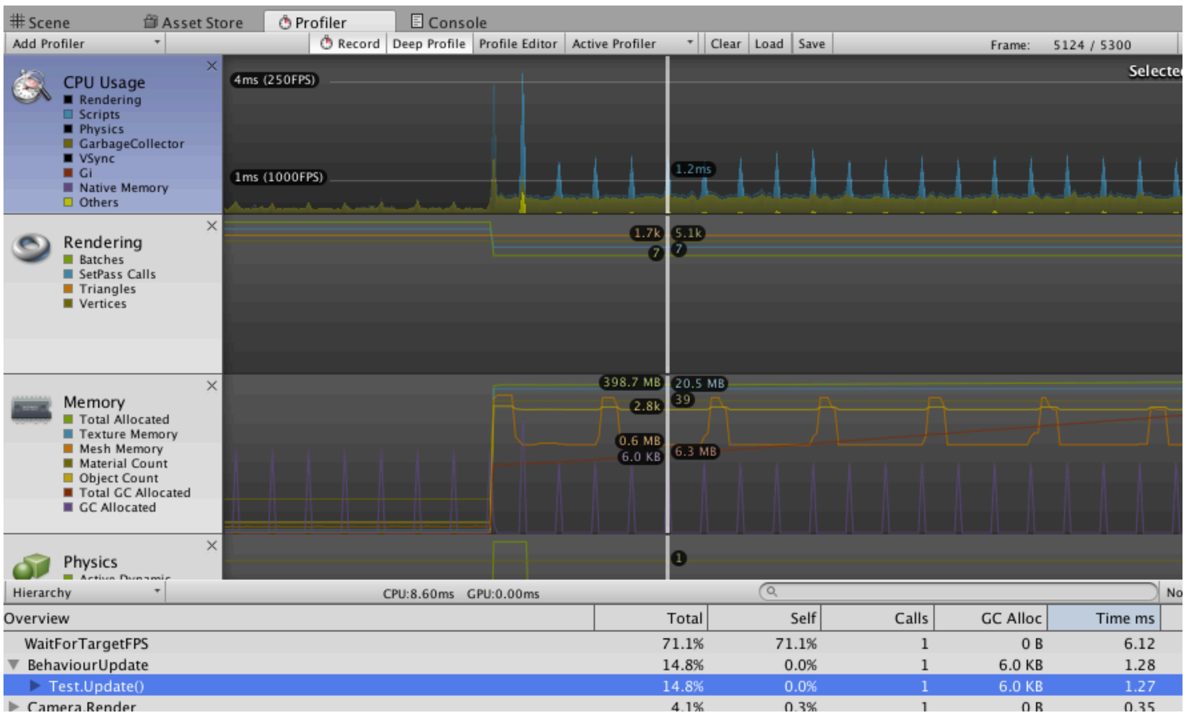


CPU:1.07ms GPU:0.00ms

Overview	Total	Self	Calls	GC Alloc	Time ms
Camera.Render	36.0%	3.2%	1	0 B	0.38
BehaviourUpdate	19.5%	1.0%	1	7.4 KB	0.21
LogStringToConsole	18.5%	18.5%	1	7.2 KB	0.19
Physics.Processing	15.0%	8.9%	1	0 B	0.16

- We could do the same profiling running the application in the Unity Editor. Then click Play button in the toolbar
- The application get started and the profiler starts grabbing data
- Click again the Play butto to stop the application
- Again, move the cursor to a memory peak
- Select the CPU Usage row and look at the process LogStringToConsole
- You will notice this process takes a little bit time more, because of the Editor overhead. Running the application on the target without the profiler enabled will speed further the processing

- When profiling for mobile applications, you should specify the target IP address



Overview	Total	Self	Calls	GC Alloc	Time ms
WaitForTargetFPS	71.1%	71.1%	1	0 B	6.12
▼ BehaviourUpdate	14.8%	0.0%	1	6.0 KB	1.28
▶ Test.Update()	14.8%	0.0%	1	6.0 KB	1.27
▶ Camera.Render	4.1%	0.3%	1	0 B	0.35