THE UNITY USER INTERFACE

MODULE 2

2.1 Exploring Unity Interface

Summary

Exploring Unity interface – the program allows us to totally rearrange its layout and save it as a default option. The author recommends rearranging the unity so that Hierarchy tab is on the right of the Scene.

Below are Project and Console tabs. Project keeps all your files, and Console tab contains any error your simulation might have.

On the top we have Scene view – where we can rearrange things as developers. Game view – is a view that a gamer will see. The author recommends putting game view below so both of them are run simultaneously.

Under Hierarchy tab, we can adjust Main Camera settings, including sound effects, with transform component we edit the position of objects, also it can be edited by the tools that are in the top right corner, including scaling and rotation.

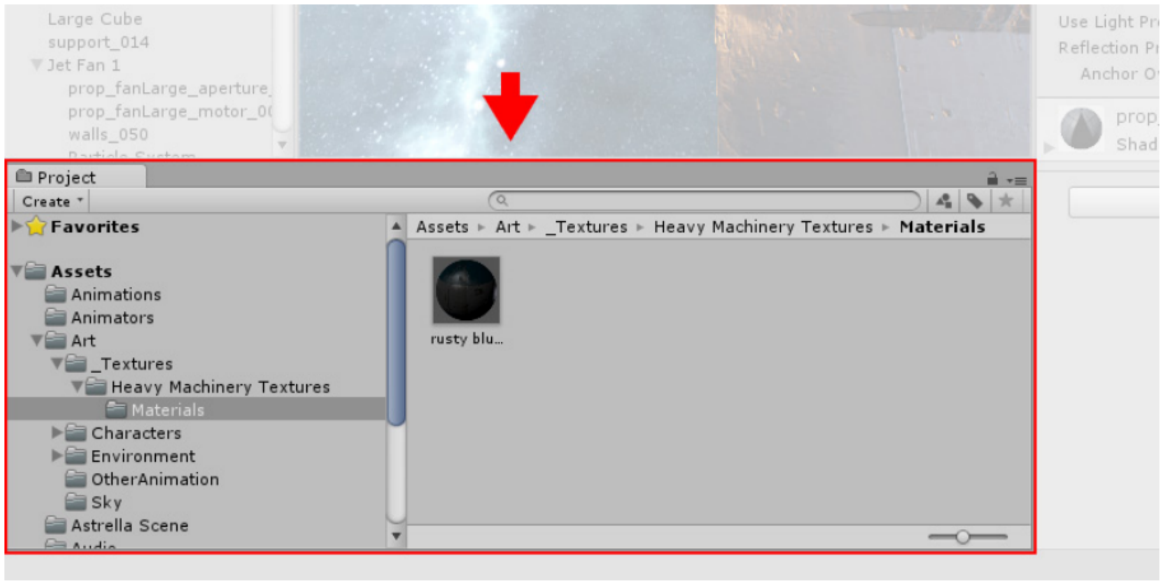
Creating new objects can be done with right-clicking in the hierarchy area and choosing the object.

The scene can be saved by pressing ctrl+s. ctrl+d will duplicate it.

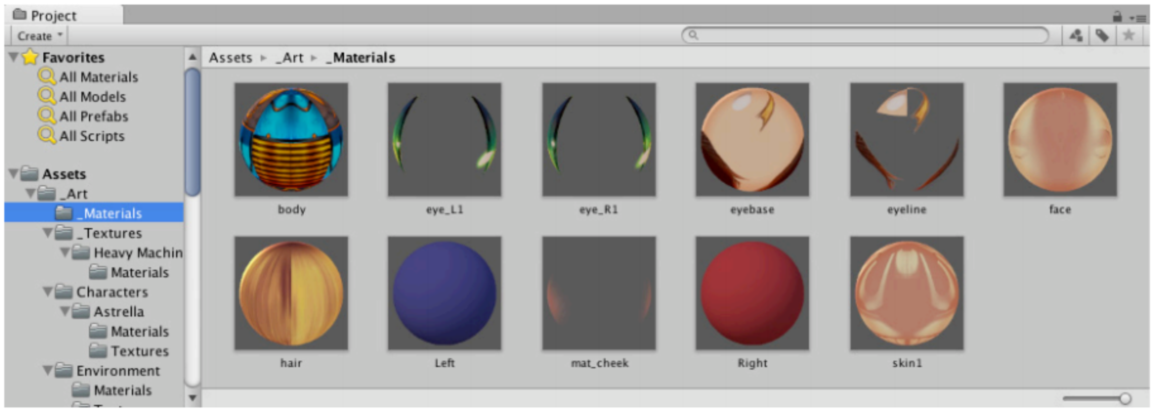
Key Points

- The Unity Interface is fully changeable.
- It is a good idea to have game view and scene view run simultaneously.
- Hierarchy tab controls camera settings, and it should be located on the right of the scene.
- Creating new objects can be done in the hierarchy tab.
- CTRL+S saves and CTRL+D duplicates scenes.

2.1.1 The Project Window



In this view, you can access and manage the assets that belong to your project.

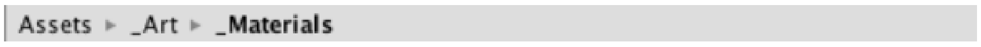


The left panel of the browser shows the folder structure of the project as a hierarchical list. When a folder is selected from the list by clicking, its contents will be shown in the panel to the right. You can click the small triangle to expand or collapse the folder, displaying any nested folders it contains. Hold down Alt while you click to expand or collapse any nested folders recursively.

The individual assets are shown in the right hand panel as icons that indicate their type (script, material, sub-folder, etc). The icons can be resized using the slider at the bottom of the panel; they will be replaced by a hierarchical list view if the slider is moved to the extreme left. The space to the left of the slider shows the currently selected item, including a full path to the item if a search is being performed.

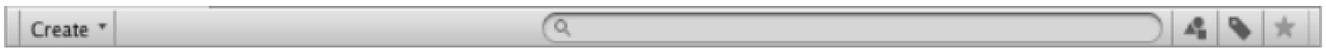
Above the project structure list is a Favorites section where you can keep frequently-used items for easy access. You can drag items from the project structure list to the Favourites and also save search queries there (see Searching below).

Just above the panel is a “breadcrumb trail” that shows the path to the folder currently being viewed. The separate elements of the trail can be clicked for easy navigation around the folder hierarchy. When searching, this bar changes to show the area being searched (the root Assets folder, the selected folder or the Asset Store) along with a count of free and paid assets available in the store, separated by a slash. There is an option in the General section of Unity’s Preferences window to disable the display of Asset Store hit counts if they are not required.



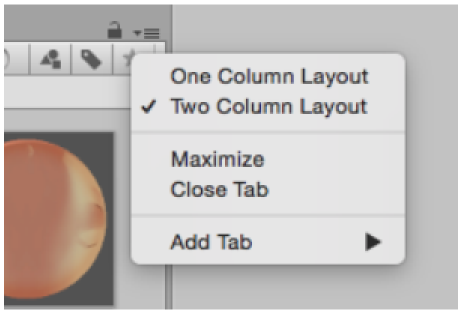
The ‘breadcrumb trail’ shows the path to the folder you are currently viewing

Along the top edge of the window is the browser’s **toolbar**.



Located at the left side of the toolbar, the Create menu lets you add new assets and sub-folders to the current folder.To its right are a set of tools to allow you to search the assets in your project.

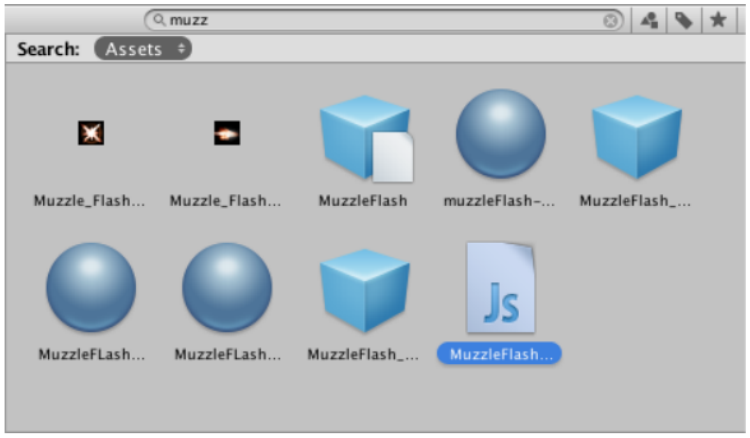
The Window menu provides the option of switching to a one-column version of the project view, essentially just the hierarchical structure list without the icon view. The lock icon next to the menu enables you to “freeze” the current contents of the view (ie, stop them being changed by events elsewhere) in a similar manner to the inspector lock.



In the top right corner, select the dropdown menu to change the view layout, and click on the lock icon to freeze the view.

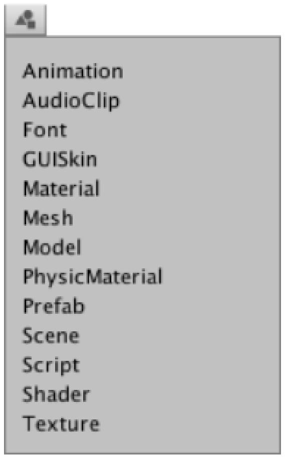
Searching

The browser has a powerful search facility that is especially useful for locating assets in large or unfamiliar projects. The basic search will filter assets according to the text typed in the search box

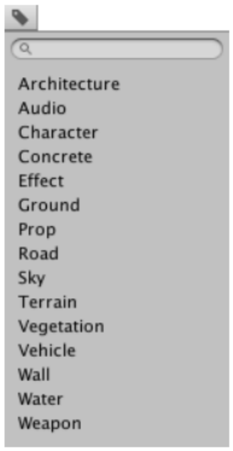


If you type more than one search term then the search is narrowed, so if you type coastal scene it will only find assets with both “coastal” and “scene” in the name (ie, terms are ANDed together).

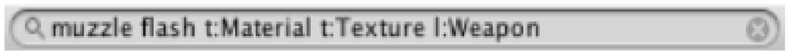
To the right of the search bar are three buttons. The first allows you to further filter the assets found by the search according to their type.



Continuing to the right, the next button filters assets according to their Label (labels can be set for an asset in the Inspector). Since the number of labels can potentially be very large, the label menu has its own mini-search filter box.



Note that the filters work by adding an extra term in the search text. A term beginning with "t:" filters by the specified asset type, while "l:" filters by label. You can type these terms directly into the search box rather than use the menu if you know what you are looking for. You can search for more than one type or label at once. Adding several types will expand the search to include all specified types (ie, types will be ORed together). Adding multiple labels will narrow the search to items that have any of the specified labels (ie, labels will be ORed together).

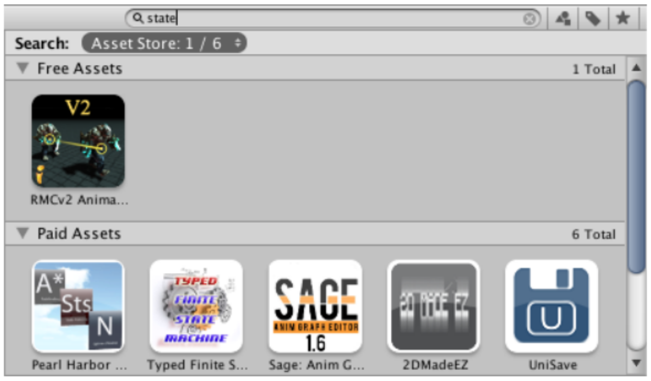


muzzle flash t:Material t:Texture l:Weapon

The rightmost button saves the search by adding an item to the Favourites section of the asset list.

Searching the Asset Store

The Project Browser's search can also be applied to assets available from the Unity Asset Store. If you choose Asset Store from the menu in the breadcrumb bar, all free and paid items from the store that match your query will be displayed. Searching by type and label works the same as for a Unity project. The search query words will be checked against the asset name first and then the package name, package label and package description in that order (so an item whose name contains the search terms will be ranked higher than one with the same terms in its package description).



state

Search: Asset Store: 1 / 6

Free Assets 1 Total

RMCv2 Anima...

Paid Assets 6 Total

Pearl Harbor ... Typed Finite S... Sage: Anim G... 2DMadeEZ UniSave

If you select an item from the list, its details will be displayed in the inspector along with the option to purchase and/or download it. Some asset types have previews available in this section so you can, for example, rotate a 3D model before buying. The inspector also gives the option of viewing the asset in the usual Asset Store window to see additional details.

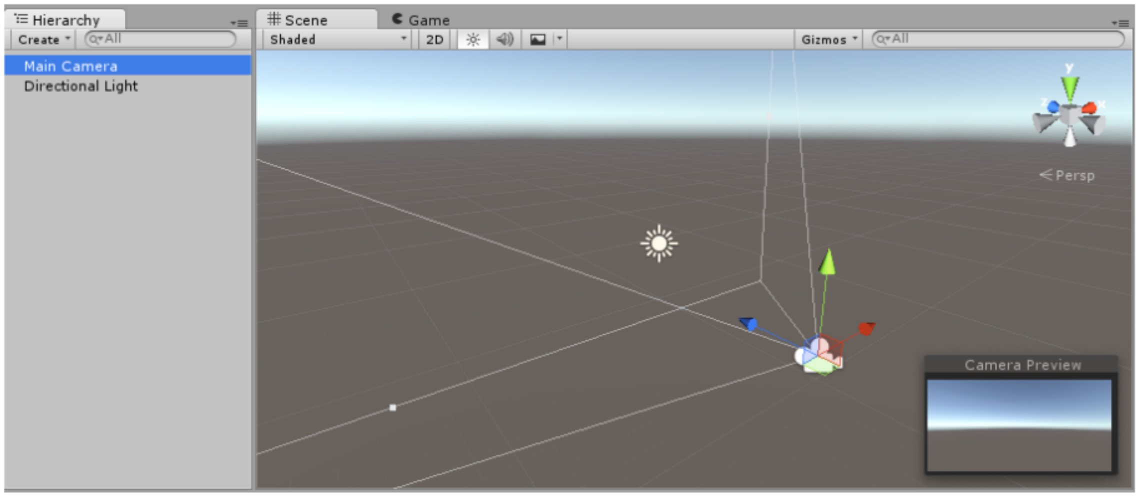
Shortcuts

The following keyboard shortcuts are available when the browser view has focus. Note that some of them only work when the view is using the two-column layout (you can switch between the one- and two-column layouts using the panel menu in the very top right corner).

F	Frame selected (ie, show the selected asset in its containing folder)
Tab	Shift focus between first column and second column (Two columns)
Ctrl/Cmd + F	Focus search field
Ctrl/Cmd + A	Select all visible items in list
Ctrl/Cmd + D	Duplicate selected assets
Delete	Delete with dialog (Win)
Delete + Shift	Delete without dialog (Win)
Delete + Cmd	Delete without dialog (OSX)
Enter	Begin rename selected (OSX)
Cmd + down arrow	Open selected assets (OSX)
Cmd + up arrow	Jump to parent folder (OSX, Two columns)
F2	Begin rename selected (Win)
Enter	Open selected assets (Win)
Backspace	Jump to parent folder (Win, Two columns)
Right arrow	Expand selected item (tree views and search results). If the item is already expanded, this will select its first child item.
Left arrow	Collapse selected item (tree views and search results). If the item is already collapsed, this will select its parent item.
Alt + right arrow	Expand item when showing assets as previews
Alt + left arrow	Collapse item when showing assets as previews

2.1.2 Scenes

Scenes contain the environments and menus of your game. Think of each unique Scene file as a unique level. In each Scene, you place your environments, obstacles, and decorations, essentially designing and building your game in pieces.



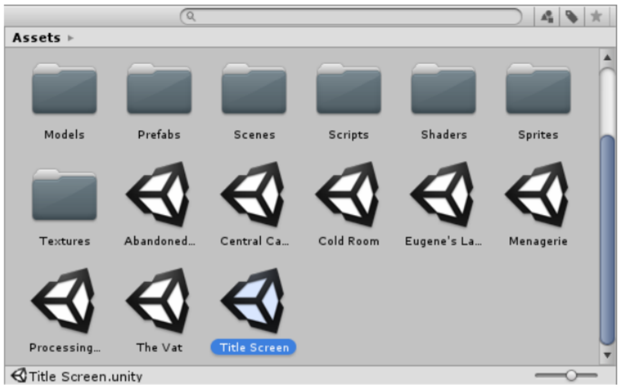
A new empty Scene, with the default 3D objects: a Main Camera and a directional Light.

When you create a new Unity project, your scene view displays a new Scene. This Scene is untitled and unsaved. The Scene is empty except for a Camera (called Main Camera) and a Light (called Directional Light).

Saving Scenes

To save the Scene you're currently working on, choose File > Save Scene from the menu, or press Ctrl + S (Windows) or Cmd + S (masOS).

Unity saves Scenes as Assets in your project's Assets folder. This means they appear in the Project window, with the rest of your Assets.



Saved Scene Assets visible in the Project window

Opening Scenes

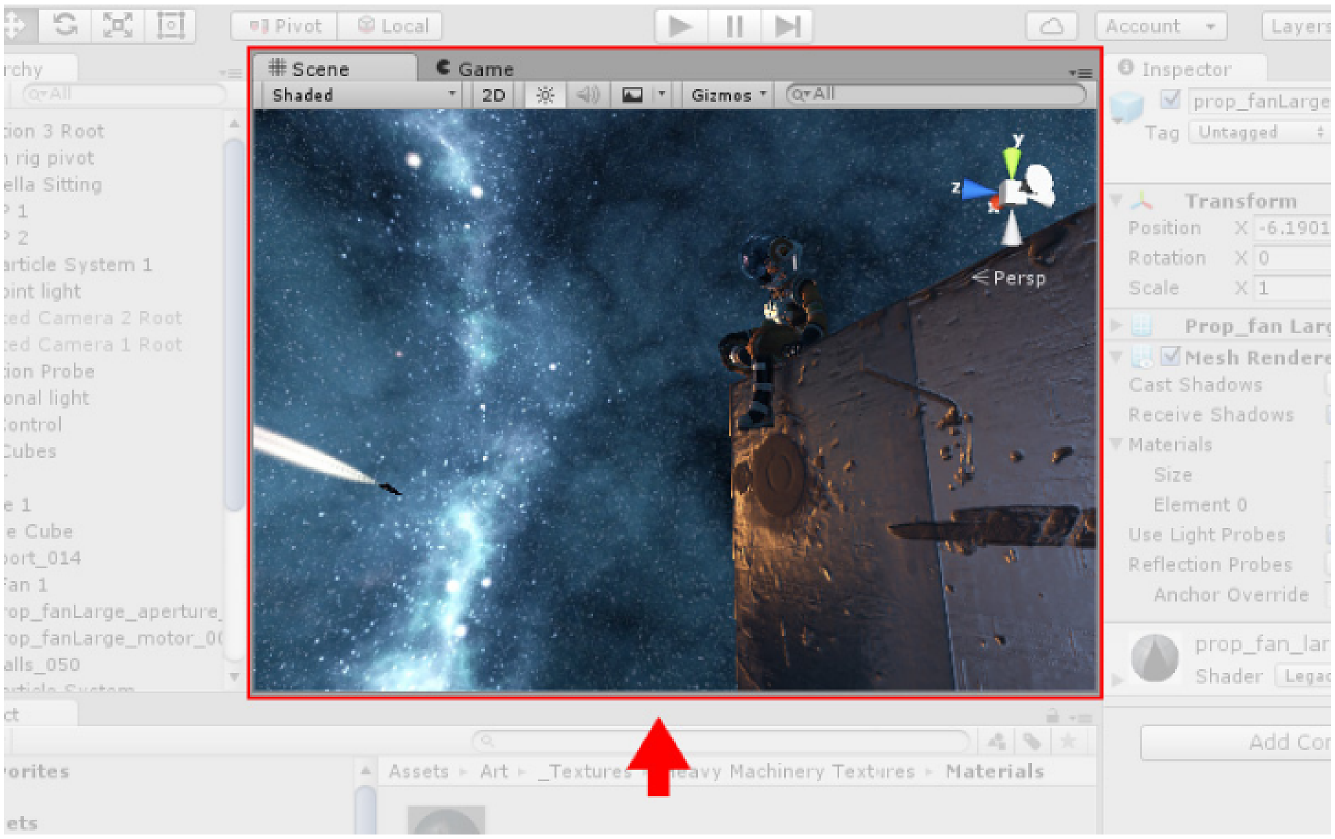
To open a Scene in Unity, double-click the Scene Asset in the Project window. You must open a Scene in Unity to work on it.

If your current Scene contains unsaved changes, Unity asks you whether you want to save or discard the changes.

Multi-Scene editing

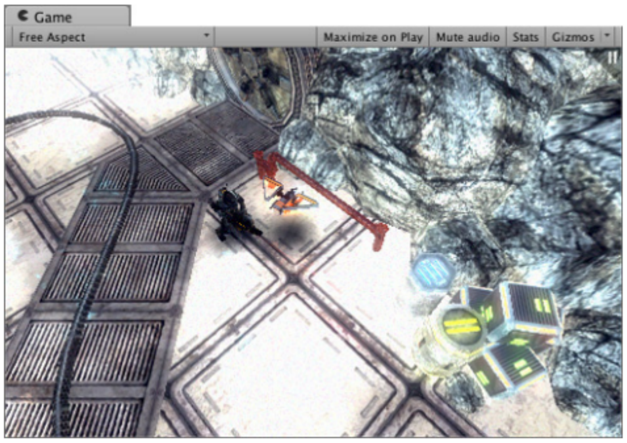
It is possible to have multiple Scenes open for editing at one time. For more information about this, see documentation on Multi-Scene editing.

2.1.3 The Scene View



The Scene view is your interactive view into the world you are creating. You will use the Scene view to select and position scenery, characters, cameras, lights, and all other types of Game Object. Being able to Select, manipulate and modify objects in the Scene view are some of the first skills you must learn to begin working in Unity.

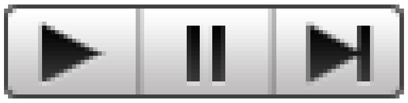
2.1.4 The Game View



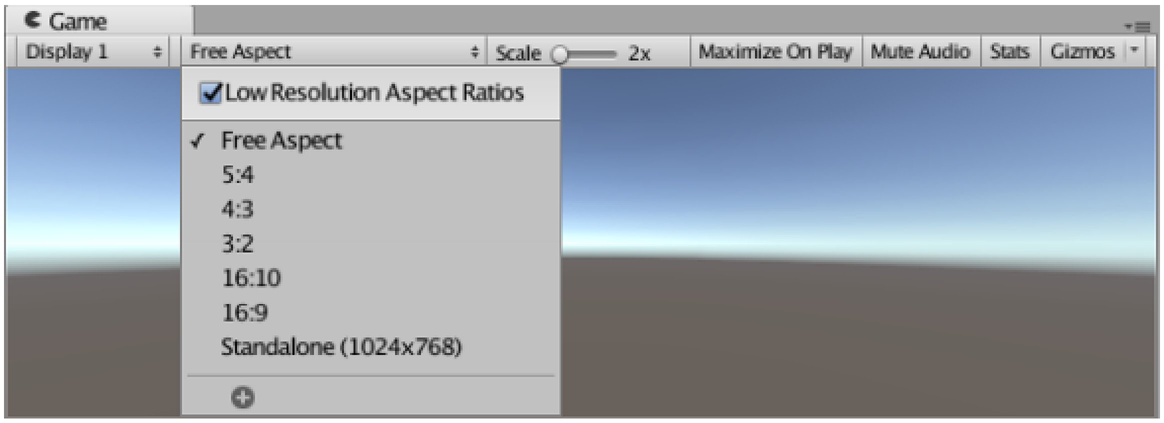
The Game View is rendered from the Camera(s) in your game. It is representative of your final, published game. You will need to use one or more Cameras to control what the player actually sees when they are playing your game. For more information about Cameras, please view the Camera Component page.

Play Mode

Use the buttons in the Toolbar to control the Editor Play Mode and see how your published game plays. While in Play Mode, any changes you make are temporary, and will be reset when you exit Play Mode. The Editor UI darkens to remind you of this.



Game view Control Bar



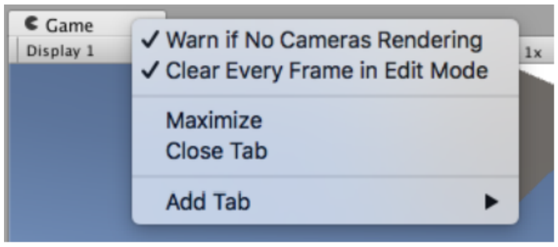
Button	Function
Display	Click this to choose from a list of Cameras if you have multiple Cameras in the Scene. This is set to Display 1 by default. (You can assign Displays to cameras in the Camera module, under the Target Display drop-down menu.)
Aspect	Select different values to test how your game will look on monitors with different aspect ratios. This is set to Free Aspect by default.
Low Resolution Aspect Ratops	Check this box if you want to emulate the pixel density of older displays: This reduces the resolution of the Game view when an aspect ratio is selected. It is always enabled when the Game view is on a non-Retina display.
Scale slider	Scroll right to zoom in and examine areas of the Game screen in more detail. It also allows you to zoom out to see the entire screen where the device resolution is higher than the Game view window size. You can also use the scroll wheel and middle mouse button to do this while the game is stopped or paused.
Maximize on Play	Click to enable: Use this to make the Game view maximize (100% of your Editor Window) for a full-screen preview when you enter Play Mode.
Mute Audio	Click to enable: Use this to mute any in-game audio when you enter Play Mode.
Stats	Click this to toggle the Statistics overlay, which contains Rendering Statistics about your game's audio and graphics. This is very useful for monitoring the performance of your game while in Play Mode.
Gizmos	Click this to toggle the visibility of Gizmos. To only see certain types of Gizmo during Play Mode, click the dropdown arrow next to the word Gizmos and only check the boxes of the Gizmo types you want to see. (See Gizmos Options below.)

Gizmos Menu

The Gizmos Menu contains lots of options for how objects, icons, and gizmos are displayed. This menu is available in both the Scene View and the Game view. See documentation on the Gizmos Menu for more information.

Advanced options

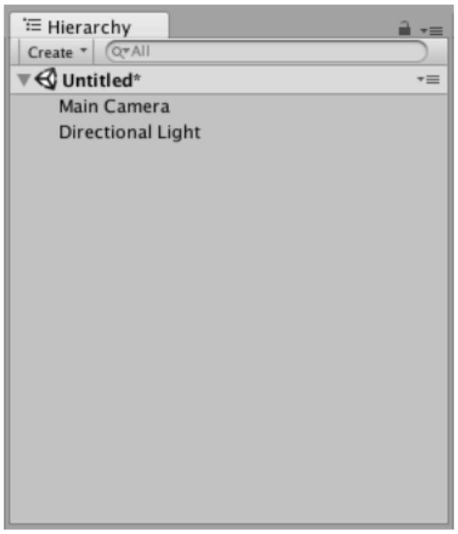
Right-click the Game tab to display advanced Game View options.


Game
Display 1
1x
✓ Warn if No Cameras Rendering
✓ Clear Every Frame in Edit Mode
Maximize
Close Tab
Add Tab ▶

Warn if No Cameras Rendering: This option is enabled by default: It causes a warning to be displayed if no Cameras are to the screen. This is useful for diagnosing problems such as accidentally deleting or disabling a Camera. Leave this enabled unless you are intentionally not using Cameras to render your game.

Clear Every Frame in Edit Mode: This option is enabled by default: It causes the Game view to be cleared every frame when your game is not playing. This prevents smearing effects while you are configuring your game. Leave this enabled unless you are depending on the previous frame's contents when not in Play Mode.

2.1.5 The Hierarchy Window



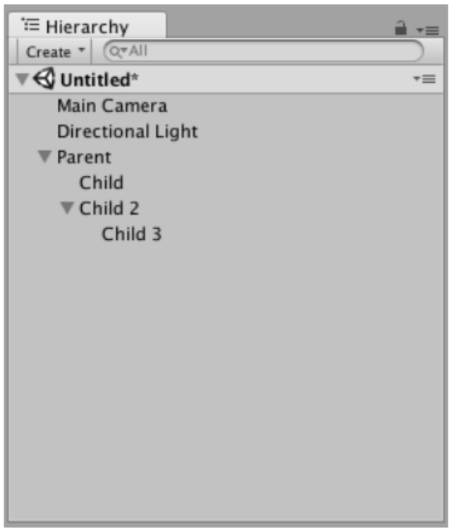
The default Hierarchy window view when you open a new Unity project

The Hierarchy window contains a list of every GameObject (referred to in this guide as an "object") in the current Scene. Some of these are direct instances of Asset files (like 3D models), and others are instances of Prefabs, which are custom objects that make up most of your game. As objects are added and removed in the Scene, they will appear and disappear from the Hierarchy as well.

By default, objects are listed in the Hierarchy window in the order they are made. You can re-order the objects by dragging them up or down, or by making them "child" or "parent" objects (see below).

Parenting

Unity uses a concept called Parenting. When you create a group of objects, the topmost object or Scene is called the "parent object", and all objects grouped underneath it are called "child objects" or "children". You can also created nested parent-child objects (called "descendants" of the top-level parent object).



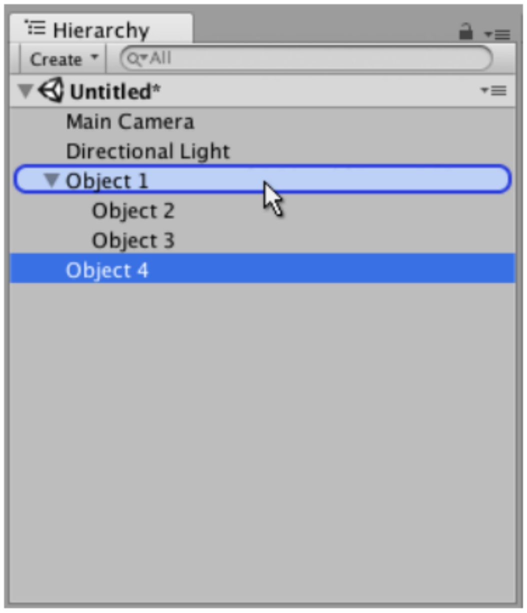
In this image, Child and Child 2 are the child objects of Parent. Child 3 is a child object of Child 2, and a descendant object of Parent.

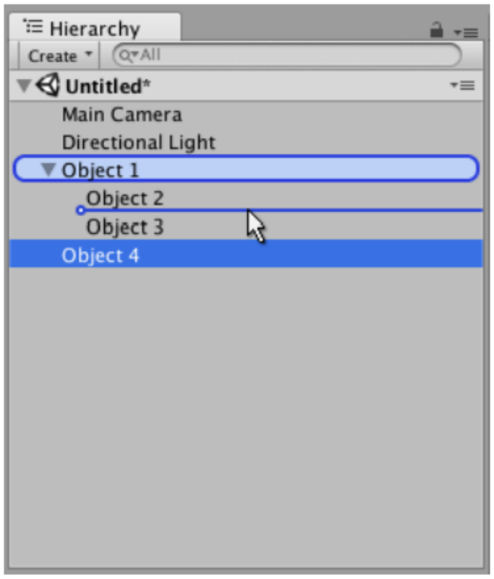
Click a parent object's drop-down arrow (on the left-hand side of its name) to show or hide its children. Hold down the Alt key while clicking the drop-down arrow to toggle visibility of all descendant objects of the parent, in addition to the immediate child object.

Making a child object

To make any object the "child" of another, drag and drop the intended child object onto the intended parent object in the Hierarchy.

In this image, Object 4 (selected) is being dragged onto the intended parent object, Object 1 (highlighted in a blue capsule).



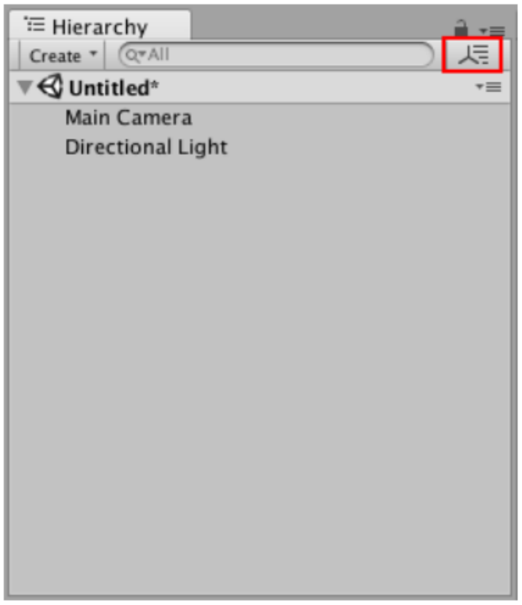


You can also drag-and-drop an object alongside other objects to make them "siblings" - that is, child objects under the same parent object. Drag the object above or below an existing object until a horizontal blue line appears, and drop it there to place it alongside the existing object.

In this image, Object 4 (selected) is being dragged between Object 2 and Object 3 (indicated by the blue horizontal line), to be placed here as a sibling of these two objects under the parent object Object 1 (highlighted in a blue capsule).

Child objects inherit the movement and rotation of the parent object. To learn more about this, see documentation on the Transform component.

Alphanumeric sorting



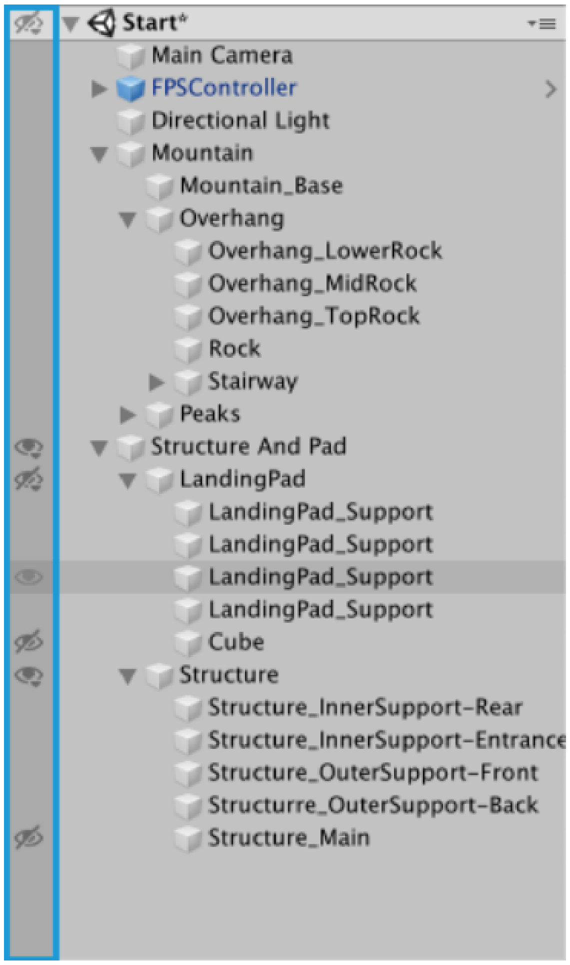
The order of objects in the Hierarchy window can be changed to alphanumeric order. In the menu bar, select Edit > Preferences in Windows or Unity > Preferences in OS X to launch the Preferenceswindow. Check Enable Alpha Numeric Sorting.

When you check this, an icon appears in the top-right of the Hierarchy window, allowing you to toggle between Transform sorting (the default value) or Alphabetic sorting.

Toggling Scene Visibility

The Scene visibility controls in the Hierarchy window allow you to quickly hide and show GameObjects in the Scene view without changing their in-game visibility. This is useful for working with large or complex Scenes where it can be difficult to view and select specific GameObjects.

For more information, see the documentation on Scene Visibility.



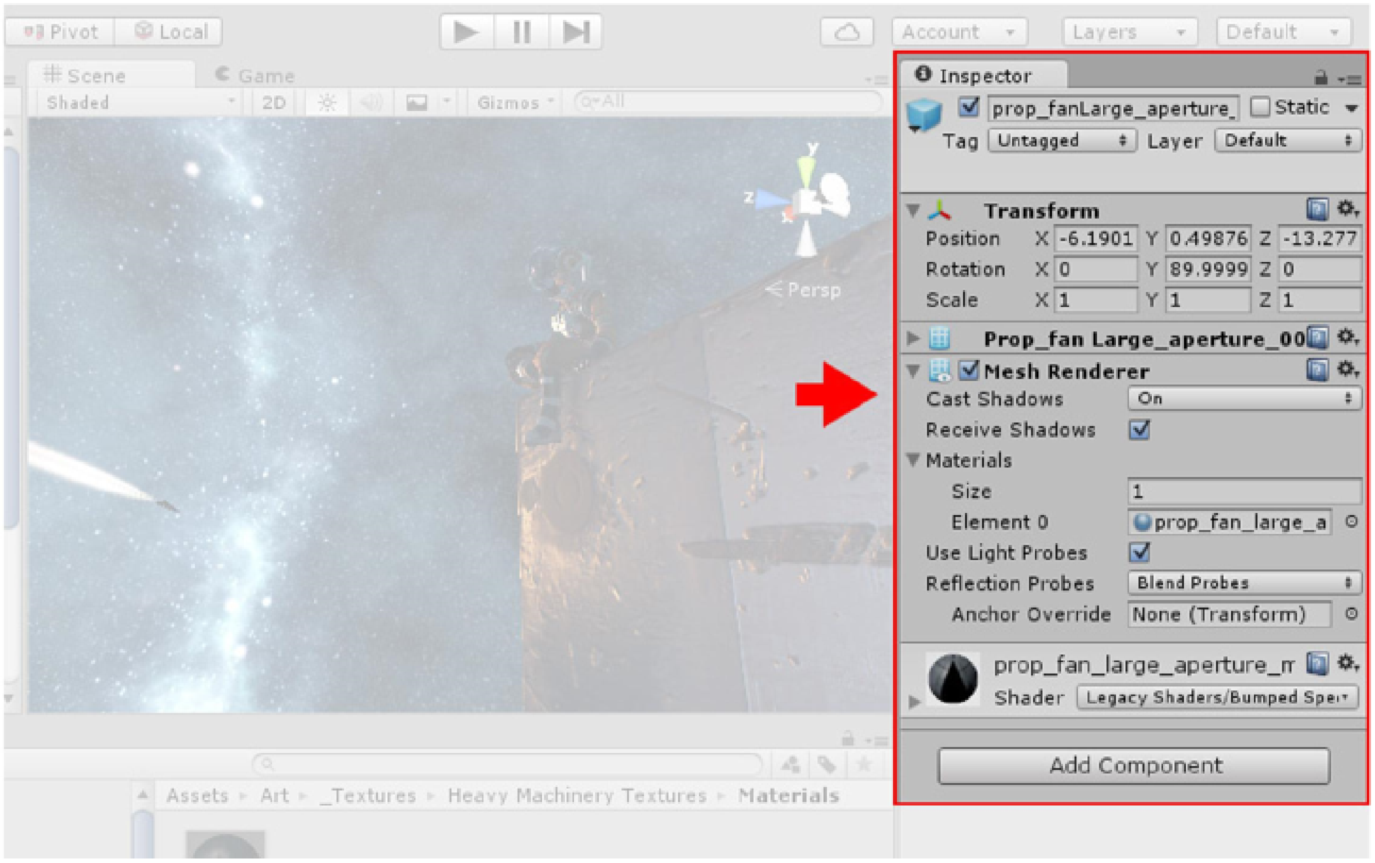
Scene visibility icons/toggles

Multi-Scene editing

It is possible to have more than one Scene open in the Hierarchy window at the same time. To find out more about this, see the Multi-Scene Editing page.

2.1.6 The Inspector Window

Projects in the Unity Editor are made up of multiple GameObjects that contain scripts, sounds, Meshes, and other graphical elements such as Lights. The Inspector window (sometimes referred to as "the Inspector") displays detailed information about the currently selected GameObject, including all attached components and their properties, and allows you to modify the functionality of GameObjects in your Scene.



```
prop_fanLarge_aperture_          Static
Tag  Untagged        Layer  Default

Transform
Position   X -6.1901   Y 0.49876   Z -13.277
Rotation   X 0         Y 89.9999   Z 0
Scale      X 1         Y 1         Z 1

Prop_fan Large_aperture_00
Mesh Renderer
Cast Shadows          On
Receive Shadows       ✓
Materials
  Size                1
  Element 0           prop_fan_large_a
Use Light Probes      ✓
Reflection Probes     Blend Probes
  Anchor Override     None (Transform)

prop_fan_large_aperture_m
Shader  Legacy Shaders/Bumped Spe
```

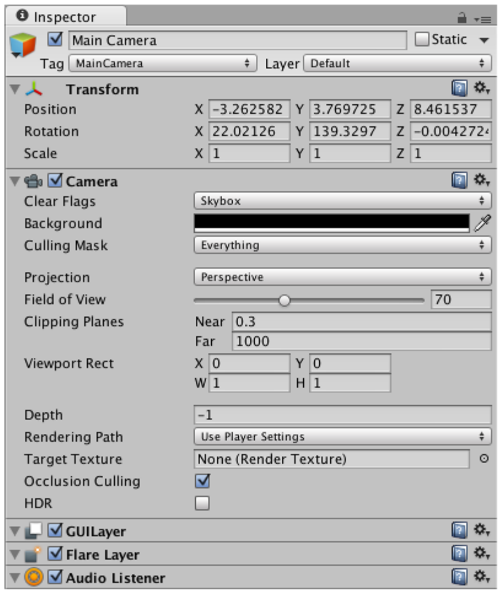
The Inspector in its default position in Unity

Inspecting GameObjects

Use the Inspector to view and edit the properties and settings of almost everything in the Unity Editor, including physical game items such as GameObjects, Assets, and Materials, as well as in-Editor settings and preferences.

When you select a GameObject in either the Hierarchy or Scene view, the Inspector shows the properties of all components and Materials of that GameObject. Use the Inspector to edit the settings of these components and Materials.

The image above shows the Inspector with the Main Camera GameObject selected. In addition to the GameObject's Position, Rotation, and Scale values, all the properties of the Main Camera are available to edit.

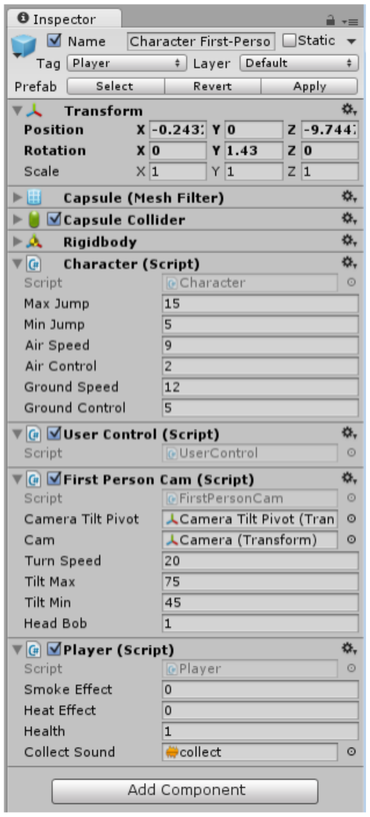


```
Inspector
✓ Main Camera                         Static
Tag MainCamera        Layer Default
Transform
Position      X -3.262582  Y 3.769725  Z 8.461537
Rotation      X 22.02126   Y 139.3297  Z -0.0042724
Scale         X 1          Y 1         Z 1
✓ Camera
Clear Flags        Skybox
Background
Culling Mask       Everything
Projection         Perspective
Field of View      70
Clipping Planes    Near 0.3
                   Far  1000
Viewport Rect      X 0   Y 0
                   W 1   H 1
Depth              -1
Rendering Path     Use Player Settings
Target Texture     None (Render Texture)
Occlusion Culling  ✓
HDR                ☐
✓ GUILayer
✓ Flare Layer
✓ Audio Listener
```

The Inspector window displaying settings for a typical GameObject and its components

Inspecting script variables

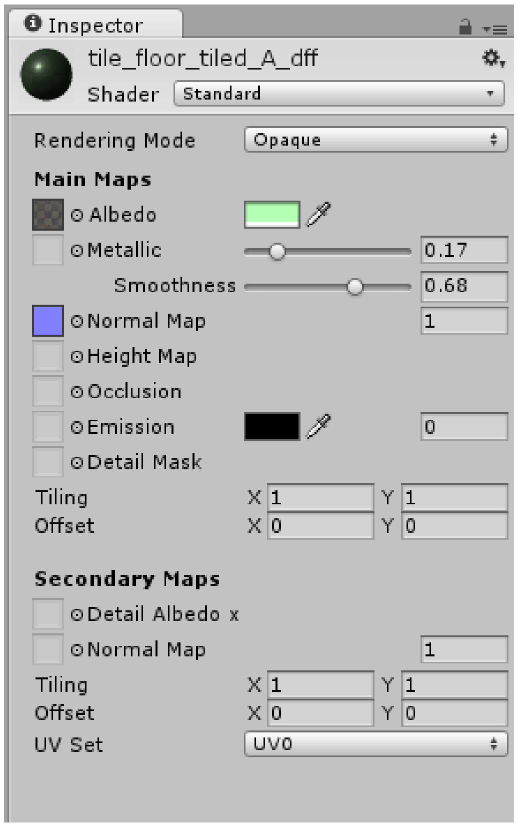
When GameObjects have custom script components attached, the Inspector displays the public variables of that script. You can edit these variables as settings in the same way you can edit the settings of the Editor's built-in components. This means that you can set parameters and default values in your scripts easily without modifying the code.



```
Inspector
✓ Name  Character First-Perso         Static
Tag Player            Layer Default
Prefab   Select   Revert   Apply
Transform
Position      X -0.243:  Y 0     Z -9.744:
Rotation      X 0        Y 1.43  Z 0
Scale         X 1        Y 1     Z 1
Capsule (Mesh Filter)
✓ Capsule Collider
Rigidbody
Character (Script)
Script              Character
Max Jump            15
Min Jump            5
Air Speed           9
Air Control         2
Ground Speed        12
Ground Control      5
✓ User Control (Script)
Script              UserControl
✓ First Person Cam (Script)
Script              FirstPersonCam
Camera Tilt Pivot   Camera Tilt Pivot (Tran
Cam                 Camera (Transform)
Turn Speed          20
Tilt Max            75
Tilt Min            45
Head Bob            1
✓ Player (Script)
Script              Player
Smoke Effect        0
Heat Effect         0
Health              1
Collect Sound       collect
Add Component
```

The Inspector window displaying settings for a GameObject with several custom scripts attached. The scripts' public properties are available to edit

Inspecting Assets



Inspector

tile_floor_tiled_A_dff

Shader	Standard
Rendering Mode	Opaque

Main Maps

Albedo	🟩	
Metallic		0.17
Smoothness		0.68
Normal Map		1
Height Map		
Occlusion		
Emission	■	0
Detail Mask		
Tiling	X 1	Y 1
Offset	X 0	Y 0

Secondary Maps

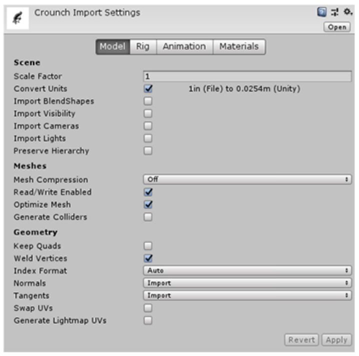
Detail Albedo x		
Normal Map		1
Tiling	X 1	Y 1
Offset	X 0	Y 0
UV Set	UV0	

The Inspector window displaying the settings for a Material Asset

When an Asset is selected in your Project window, the Inspector shows you the settings related to how that Asset is imported and used at run time (when your game is running either in the Editor or your published build).

Each type of Asset has a different selection of settings. The images below demonstrate some examples of the Inspector displaying the import settings for other Asset types:

The Model tab of the Model Import Settings window:



Crounch Import Settings

Scene

Scale Factor	1
Convert Units	✓ 1in (File) to 0.0254m (Unity)
Import BlendShapes	☐
Import Visibility	☐
Import Cameras	☐
Import Lights	☐
Preserve Hierarchy	☐

Meshes

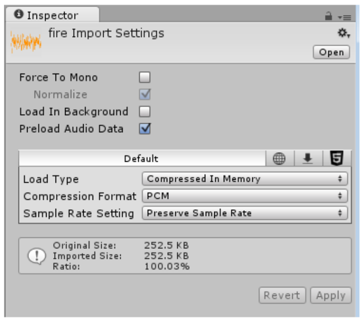
Mesh Compression	Off
Read/Write Enabled	✓
Optimize Mesh	✓
Generate Colliders	☐

Geometry

Keep Quads	☐
Weld Vertices	✓
Index Format	Auto
Normals	Import
Tangents	Import
Swap UVs	☐
Generate Lightmap UVs	☐

The Inspector window displaying the import settings for an .fbx file containing 3D Models

The Audio Clip Import Settings window:



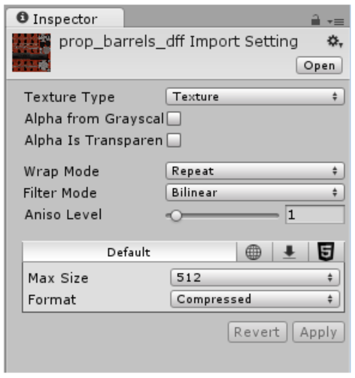
fire Import Settings

Force To Mono	☐
Normalize	✓
Load In Background	☐
Preload Audio Data	✓
Load Type	Compressed In Memory
Compression Format	PCM
Sample Rate Setting	Preserve Sample Rate

Original Size:	252.5 KB
Imported Size:	252.5 KB
Ratio:	100.03%

The Inspector window displaying the import settings for an Audio file

The Texture Import Setting window:



prop_barrels_dff Import Setting

Texture Type	Texture
Alpha from Grayscal	☐
Alpha Is Transparen	☐
Wrap Mode	Repeat
Filter Mode	Bilinear
Aniso Level	1
Max Size	512
Format	Compressed

The Inspector window displaying the Import Settings for a Texture

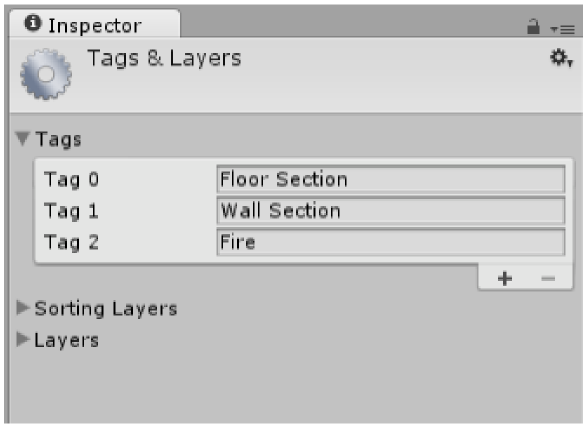
Prefabs

If you have a Prefab selected, some additional options are available in the Inspector window.

For more information, see documentation on Prefabs.

Project settings

When you select any of the Project Settings categories (menu: Editor > Project Settings), these settings are displayed in the Inspector window. For more information, see documentation on Project Settings.



Tag 0	Floor Section
Tag 1	Wall Section
Tag 2	Fire

The Inspector window displaying the Tags & Layers Project Settings panel

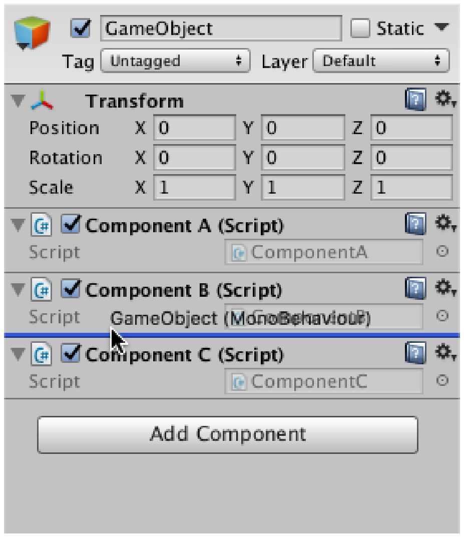
Icons and labels

You can assign custom icons to GameObjects and scripts. These display in the Scene view along with built-in icons for GameObjects such as Lights and Cameras.

For more about icons and labels, see Unity documentation on assigning icons.

Re-ordering components

To reorder components in the Inspector window, drag-and-drop their headers from one position to another. When you drag a component header, a blue insertion marker appears. This shows you where the component should go when you drop the header:



Dragging and dropping components on a GameObject in the Inspector window

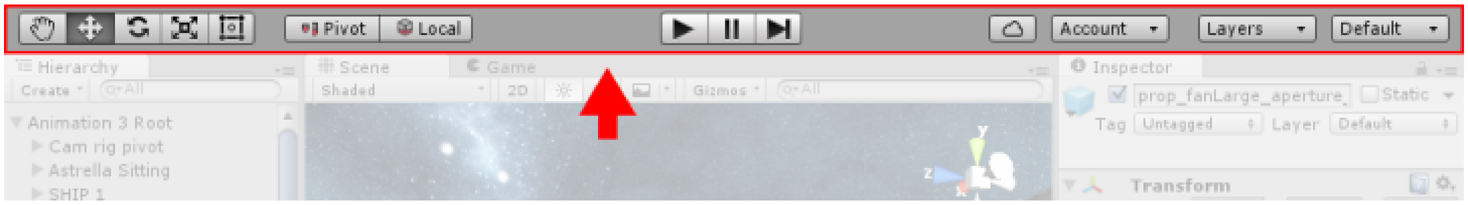
You can only reorder components on a GameObject. You can't move components between different GameObjects.

You can also drag and drop script Assets directly into the position you want them to appear.

When you select multiple GameObjects, the Inspector displays all of the components that the selected GameObjects have in common. To reorder all of these common components at once, multi-select the GameObjects, then drag-anddrop the components into a new position in the Inspector.

The order you give to components in the Inspector window is the order you need to use when querying components in your user scripts. If you query the components programmatically, you'll get the order you see in the Inspector.

2.1.7 The Toolbar



The Toolbar consists of seven basic controls. Each relate to different parts of the Editor.

	Transform Tools – used with the Scene View
Pivot Local	Transform Gizmo Toggles – affect the Scene View display
	Play/Pause/Step Buttons – used with the Game View
	Cloud Button - opens the Unity Services Window.
Account	Account Drop-down - used to access your Unity Account
Layers	Layers Drop-down – controls which objects are displayed in Scene View
Layout	Layout Drop-down – controls arrangement of all Views

unity

THE UNITY USER INTERFACE

MODULE 2

2.2

Summary

In the section a script creation process is explained.

At first adding script folder to Project assets must happen - Right click in the folder and add a new C# Script. We can create script in assets and add it to the scene. When we double-click this script it will open our IDE, for example visual studio where we can write and edit scripts.

Breaking down the script:

After the name of the script there is a Monobehavior class – a script means you can drag it along with the project.

After the Start, we can make the object move, we can type transform.localScale for example, and it will act accordingly. We can save the script and go back to unity, we can see that when we play the scene our object moves.

We can use Update() function, if we type transform.localscale * =1.01f giving it a float value.

The Start() function will only happen once after we click play, Update() punction will happen continuously.

We can add conditional fuctions IF(), so other fuctions run if the conditions are met.

Key Points

- Script folder must be added to project assets (C# script)
- Double clicking C# script opens visual studio.
- Monobehavior class (default class) means a script will be dragged along.
- Update() function happens continuously after playing the scene
- Start() functions play once
- IF() functions can be used to add conditions.

2.2.1 Scripting

```
    vignette.blur = (1-health) * 2 + smokeEffect [illegible]
    vignette.blurDistance = (1-health) * 2 + [illegible]
    vignette.chromaticAberration = heatEffect * [illegible]
}


void OnTriggerStay(Collider c)
{
    var fire = c.GetComponent<Fire>();
    if (fire && fire.alive)
    {
        float dist = 1-((transform.position - fire.[illegible]
        NearHeat(dist);
    }


    var smoke = c.GetComponent<[illegible]
    if (smoke && smoke.GetComponent<[illegible]
    {
        float dist = 1-((transform.position - smoke.[illegible]
        NearSmoke(dist);
    }
}

void OnCollisionEnter(Collision c)
{
    var healthBox = c.gameObject.GetComponent<[illegible]
    if (healthBox)
    {
```

Scripting is an essential ingredient in all games. Even the simplest game needs scripts, to respond to input from the player and arrange for events in the gameplay to happen when they should. Beyond that, scripts can be used to create graphical effects, control the physical behaviour of objects or even implement a custom AI system for characters in the game.

Scripting is a skill that takes some time and effort to learn. The intention of this section is not to teach you how to write script code from scratch, but rather to explain the main concepts that apply to scripting in Unity.

See the Knowledge Base Editor section for troubleshooting, tips and tricks.

2.2.2 Creating and Using Scripts

The behavior of GameObjects is controlled by the Components that are attached to them. Although Unity's built-in Components can be very versatile, you will soon find you need to go beyond what they can provide to implement your own gameplay features. Unity allows you to create your own Components using scripts. These allow you to trigger game events, modify Component properties over time and respond to user input in any way you like.

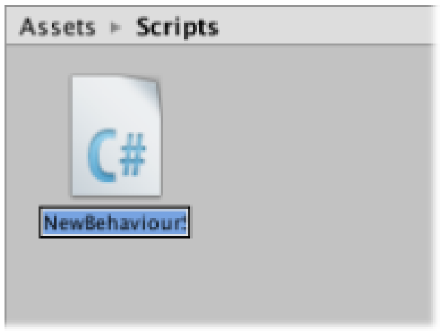
Unity supports the C# programming language natively. C# (pronounced C-sharp) is an industry-standard language similar to Java or C++.

In addition to this, many other .NET languages can be used with Unity if they can compile a compatible DLL - see here for further details.

Learning the art of programming and the use of these particular languages is beyond the scope of this introduction. However, there are many books, tutorials and other resources for learning how to program with Unity. See the Learning section of our website for further details.

Creating Scripts

Unlike most other assets, scripts are usually created within Unity directly. You can create a new script from the Create menu at the top left of the Project panel or by selecting Assets > Create > C# Script from the main menu.



The new script will be created in whichever folder you have selected in the Project panel. The new script file's name will be selected, prompting you to enter a new name.

It is a good idea to enter the name of the new script at this point rather than editing it later. The name that you enter will be used to create the initial text inside the file, as described below

Anatomy of a Script file

When you double-click a script Asset in Unity, it will be opened in a text editor. By default, Unity will use Visual Studio, but you can select any editor you like from the External Tools panel in Unity's preferences (go to Unity > Preferences). The initial contents of the file will look something like this:

```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour {

    // Use this for initialization
    void Start () {

    }

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

    }

}
```

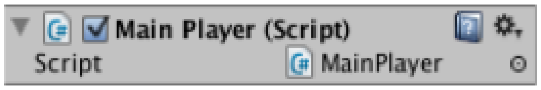
A script makes its connection with the internal workings of Unity by implementing a class which derives from the built-in class called MonoBehaviour. You can think of a class as a kind of blueprint for creating a new Component type that can be attached to GameObjects. Each time you attach a script component to a GameObject, it creates a new instance of the object defined by the blueprint. The name of the class is taken from the name you supplied when the file was created. The class name and file name must be the same to enable the script component to be attached to a GameObject.

The main things to note, however, are the two functions defined inside the class. The Update function is the place to put code that will handle the frame update for the GameObject. This might include movement, triggering actions and responding to user input, basically anything that needs to be handled over time during gameplay. To enable the Update function to do its work, it is often useful to be able to set up variables, read preferences and make connections with other GameObjects before any game action takes place. The Start function will be called by Unity before gameplay begins (ie, before the Update function is called for the first time) and is an ideal place to do any initialization.

Note to experienced programmers: you may be surprised that initialization of an object is not done using a constructor function. This is because the construction of objects is handled by the editor and does not take place at the start of gameplay as you might expect. If you attempt to define a constructor for a script component, it will interfere with the normal operation of Unity and can cause major problems with the project.

Controlling a GameObject

As noted above, a script only defines a blueprint for a Component and so none of its code will be activated until an instance of the script is attached to a GameObject. You can attach a script by dragging the script asset to a GameObject in the hierarchy panel or to the inspector of the GameObject that is currently selected. There is also a Scripts submenu on the Component menu which will contain all the scripts available in the project, including those you have created yourself.The script instance looks much like any other Component in the Inspector:



Once attached, the script will start working when you press Play and run the game. You can check this by adding the following code in the Start function:-

```
// Use this for initialization
void Start ()
{
    Debug.Log("I am alive!");
}
```

Debug.Log is a simple command that just prints a message to Unity's console output. If you press Play now, you should see the message at the bottom of the main Unity editor window and in the Console window (menu: Window > General > Console).

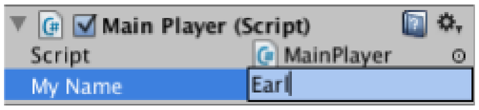
2.2.3 Variables and the Inspector

When creating a script, you are essentially creating your own new type of component that can be attached to Game Objects just like any other component.

Just like other Components often have properties that are editable in the inspector, you can allow values in your script to be edited from the Inspector too.

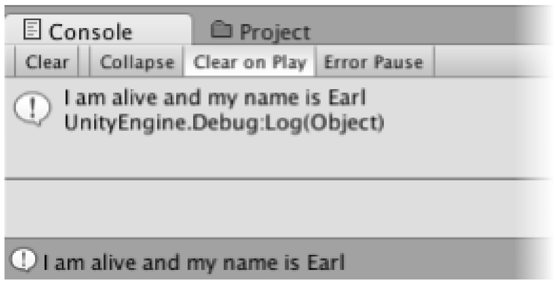
```
using UnityEngine;
using System.Collections;

public class MainPlayer : MonoBehaviour
{
    public string myName;

    // Use this for initialization
    void Start ()
    {
        Debug.Log("I am alive and my name is " + myName);
    }
}
```

This code creates an editable field in the Inspector labelled "My Name".



Main Player (Script)
Script MainPlayer
My Name Earl

Unity creates the Inspector label by introducing a space wherever a capital letter occurs in the variable name. However, this is purely for display purposes and you should always use the variable name within your code. If you edit the name and then press Play, you will see that the message includes the text you entered.



```
I am alive and my name is Earl
UnityEngine.Debug:Log(Object)

I am alive and my name is Earl
```

In C#, you must declare a variable as public to see it in the Inspector.

Unity will actually let you change the value of a script's variables while the game is running. This is very useful for seeing the effects of changes directly without having to stop and restart. When gameplay ends, the values of the variables will be reset to whatever they were before you pressed Play. This ensures that you are free to tweak your object's settings without fear of doing any permanent damage.

2.2.4 C# Job System Overview

How the C# Job System works

The Unity C# Job System allows users to write multithreaded codethat interacts well with the rest of Unity and makes it easier to write correct code.

Writing multithreaded code can provide high-performance benefits. These include significant gains in frame rate. Using the Burst compiler with C# jobs gives you improved code generationquality, which also results in substantial reduction of battery consumption on mobile devices.

An essential aspect of the C# Job System is that it integrates with what Unity uses internally (Unity's native job system). User-written code and Unity share worker threads. This cooperation avoids creating more threads than CPU cores, which would cause contention for CPU resources.

For more information, watch the talk Unity at GDC - Job System & Entity Component System.