

Unity3D Tutorial - Beginner Basics

0. About This Tutorial

- 1 Unity3D can be an intimidating *Game Development* tool for Windows and Mac OS. This tutorial will help a beginner feel more comfortable using to make 3D games.
- 2 To run Unity3D well, a computer needs a 3D graphics card, and 2GB or more of RAM.
- 3 To understand this tutorial, you must be comfortable running computer programs in Windows or Mac OS, and using a mouse and keyboard. Knowing programming is not required, but helpful!
- 4 By the end of this tutorial, a beginner should be able to make and play a simple **Scene** in Unity3D, with a **FPSController**, and simple obstacles. More enthusiastic users should be able to create a platform-based 3D maze game, complete with win condition.
- 5 This tutorial should explain enough for a beginner to feel comfortable with Unity3D.


1. Installing Unity3D

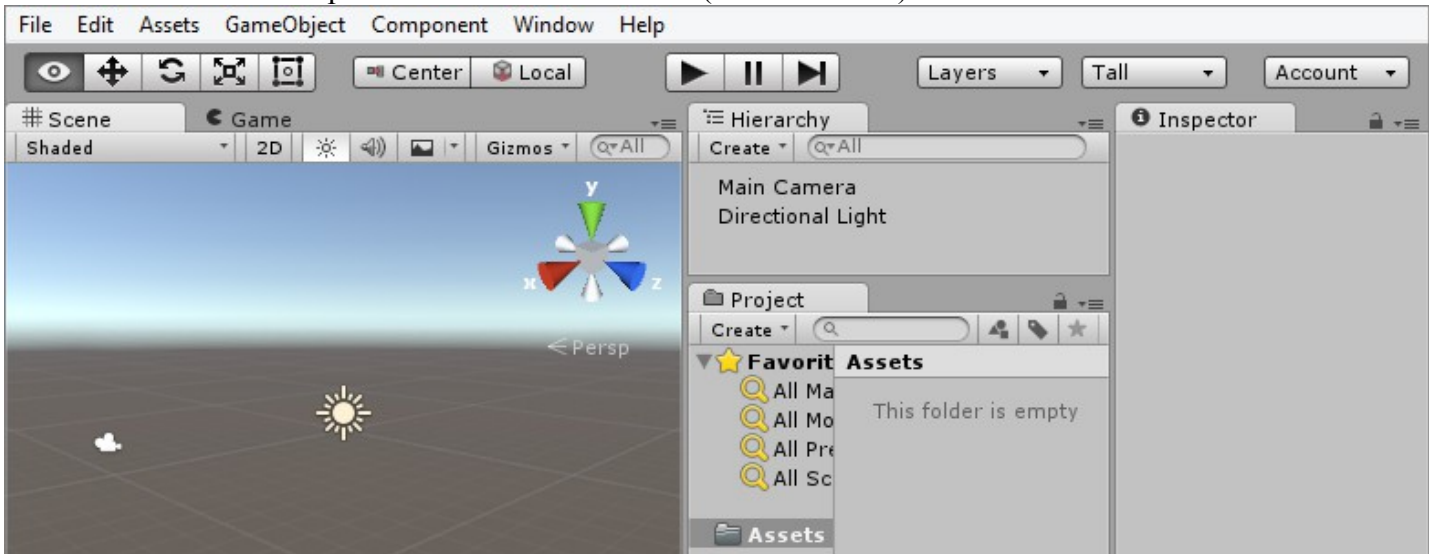
- 1 Head to <http://unity3d.com/unity/download> and download the *free* version. Run the installer. It's a large install so be patient!
-
- 2 While downloading, make sure you have an e-mail account. You need it to finish installation!
 - 3 During install, press **Next >**, and **I Agree** to the license, which says *you are 13 or older, won't do anything illegal with Unity, and respect copyright* (note: *I am not a lawyer, this is not legal advice*).
-
- 4 For slower computers, un-check ☐ **Example Project** to reduce the install, and speed up the first load time.
 - 5 Press **Next >**, **Install**, and wait for the progress bar
 - 6 Finally, after Unity finishes installing press **Finish**.
 - 7 Unity3D should start up on it's own. If it doesn't start automatically, start Unity3D with the shortcut that might be on the desktop, Applications Folder, or by find it in the *Start-menu* or the Windows *dashboard*.
 - 8 If you have a Unity Account, sign in with that, otherwise, **create one**, then sign in.
 - 9 Select **PERSONAL EDITION** for the free version, and **agree** to the license, saying *you will buy the pro version when your games make \$100,000 in revenue* (*I am not a lawyer, this is not legal advice*).
 - 10 Press **OK** at the bottom of the survey, and **Start Using Unity**!

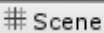

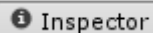
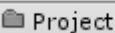
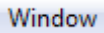
2. Start a Project

- 1 If you un-checked ☐ **Example Project** earlier, just select **New project**. Otherwise, after the **Example Project** finishes loading, press **File**, and select **New Project...**
- 2 Browse **...** to the **Desktop**, and create a **New folder** (**Right-Mouse Button** in the *File Chooser* and select **New** > **Folder**, or press **Ctrl+Shift+N** in Windows, or **⌘+Shift+N** in Mac OS).
- 3 Name your project something like **tutorial1**. Make this a **3D** project.
- 4 Name the folder **unity3d tutorial**, then **Select Folder**. Press **Create project** to create the project.

3. Pick a User Interface Layout

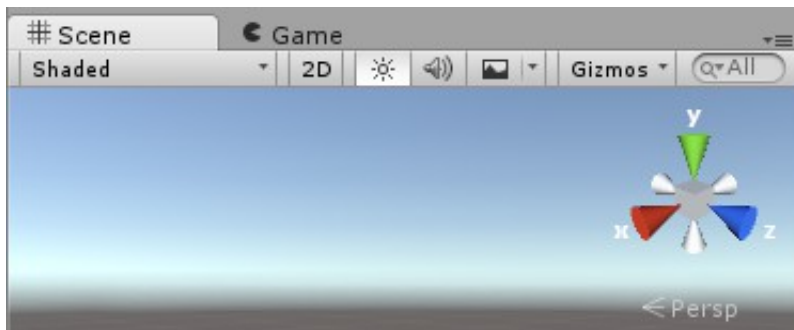
- 1 Choose a User Interface layout using the UI drop-down menu () on the upper-right.
- 2 Below is an example of the **Tall** User Interface (or UI for short):

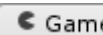
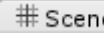



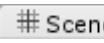
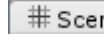
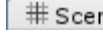
- 3 Make sure you can find , , , and  tabs.
- 4 If you lose any UI, check , or reset the layout with the UI drop-down menu.

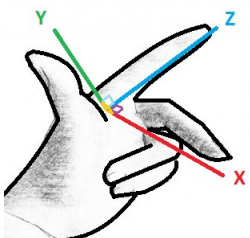
4. Use the Scene User Interface

- 1 Find your  view. It looks something like this:



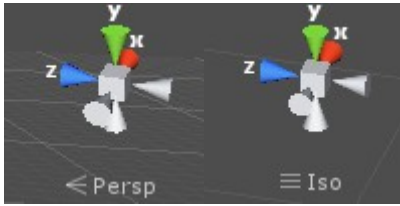
- The  view is used while *play-testing* your game. If you accidentally click that, click  again.
- You can *toggle* (turn on and off) the *sky box* with  button. The *Grid* can be toggled as one of the *Gizmos*, along with many other icons in the *scene* view.

- 2 Turn the  camera by holding the **Right-Mouse Button** and moving the mouse. Turning with the mouse like this is called *Mouselook*.
- 3 While holding the **Right-Mouse Button**, *move* the camera in the  using **W**, **A**, **S**, and **D**. If you have ever played Minecraft, or another First Person Shooter with a mouse-and-keyboard, this should be familiar. If you aren't comfortable with 3D games, this will take some getting used to!
- 4 Notice the 3D **Scene Gizmo** at the Upper Right of the . You'll notice it move when you *Mouselook*. It's diagramming the *XYZ Axis*.



- You can imagine the XYZ Axis in real life using a hand gesture, like this:
 - The Z axis (Depth) is forward, along your pointer finger
 - The Y axis (Height) is up, along your thumb
 - The X axis (Width) is to the side, along your middle finger
- If you've studied Algebra or Geometry, you may have seen an *XY Grid*, also called a *Cartesian Plane* or *Euclidean space*. The *XYZ Axis* is like that, but with a Z dimension too! Not only Width and Height, but also Depth!



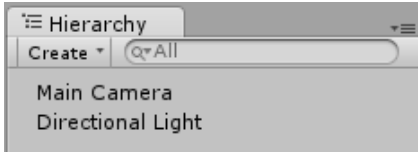


- If you Click on the cube in the center of the 3D **Scene Gizmo**, you will toggle how the camera draws 3D, between *Perspective* and *Isometric*.
- **Perspective** is how we see in real life: far-away-things are smaller than close-by things.
- **Isometric**, also called *orthographic*, means distance does not change size, so it is useful for lining up objects that are far apart.

5 The **# Scene** will help you organize your game, in 3D space.

5. Use the Hierarchy User Interface

1 The **Hierarchy** will help you organize your scene as a list of **Game Objects**.



- Everything currently in the game **Scene** will be listed here!
- You can add a new element to the scene with **Create**.
- Quickly Double-Click on a listed element to adjust the view, showing it.
- Slowly Double-Click on a listed element to rename it!

2 Press **Create** and a 3D Object → Cube. It should appear in the **Hierarchy**, and the **# Scene**.

3 If you cannot see the new Cube, Quickly Double-Click on its name in the **Hierarchy**.

4 Other simple **Game Objects** you should experiment with creating: Sphere, Capsule, Cylinder, Plane. Don't be afraid to try other things too! You can **Undo** (**Ctrl+Z** or **⌘+Z**) almost anything, including adding a new object. To **Delete** an extra object, select it and press **Delete**, or press the **Right-Mouse Button** over the object in the **Hierarchy** and select **Delete** from the menu.

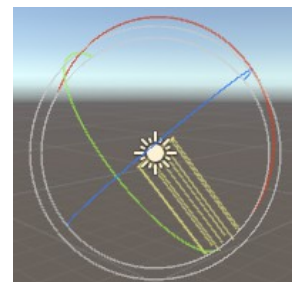
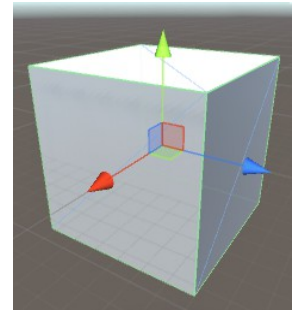
5 You can also find the same create menu through **GameObject**.

6 Move around using **WASD** and **Mouselook** (hold down the **Right-Mouse Button**), to see where the **Cube** and **Directional Light** are.

7 If you don't see the **Translate Gizmo** (as pictured to the right), select the Cube, and press **+**, in the upper-left of the Unity3D UI (User Interface).

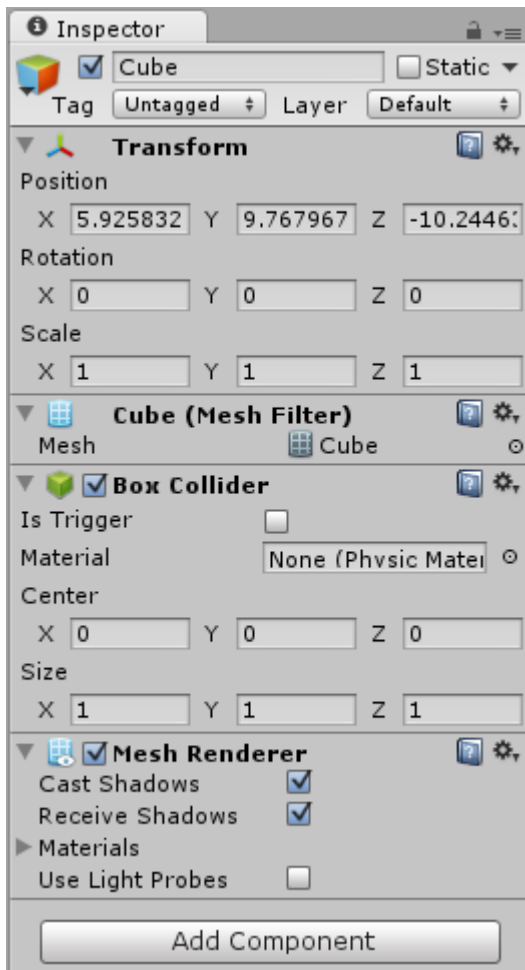
8 Try to move the **Cube** with the **Translate Gizmo** (colored *Arrows* and *Planes* in the **# Scene**). **Drag** the *Arrows* with your mouse, or **Drag** the *Planes* between the lines of the arrows.

9 Select the **Directional Light**, then rotate it with the **Rotation Gizmo**. You should notice the lighting on the cube change. If not, press **☀**, at the top of the **# Scene**, to toggle lights. If you create more cubes in the scene, you may notice shadows moving as you rotate the light!



6. Use the Inspector User Interface

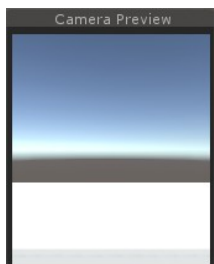
1 Select an object in the **Hierarchy** or **Scene** and you'll see its details in the **Inspector**.



- You can rename an object with the text field at the top.
- Hide this object if you un-check ☒ next to the name.
- All **Game Objects** have a **Transform**, which can be changed with the input fields. The **Transform** consists of:
 - **Position** in XYZ space
 - **Rotation**, which is how it is turned, on the XYZ axis
 - **Scale**, which is how it is stretched across the XYZ axis
- You can also use other **Gizmos** and the mouse to change the **Position**, **Rotation**, and **Scale**. The buttons that enable those **Gizmos** are at the top-left of the Unity User Interface, in a panel that looks like

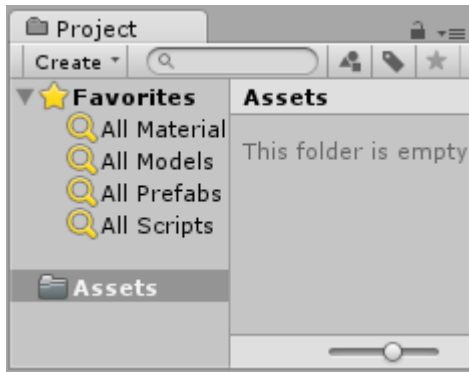
- You might see these **Components** somewhere in your scene:
 - **Cube (Mesh Filter)**: makes the cube *model*, or **Mesh**
 - **Box Collider**: makes the cube collide-able like a box
 - **Mesh Renderer**: allows **Game Object** to render (show)
 - **Light**: set a light, with color, brightness, and more
 - **Camera**: will **render** a *view* to the screen
 - **Flare Layer**: allows *lens-flares* to be rendered
 - **GUI Layer**: allows custom **GUIs** to be rendered
 - **Audio Listener**: allows sounds to be heard in-game
- Much of Unity development is creating and editing your own **Components**, which are added to **Game Objects**.

- 2 Select your **Cube**. You can click on it in the **Hierarchy**, or in the **Scene**.
- 3 Set the Cube's **Position** to X 0 Y -2 Z 0. You can change each number by selecting the text-box with the number in it, and typing in a new number.
- 4 Set the Cube's **Scale** to X 20 Y 1 Z 10.
- 5 Rename your Cube to ground (and press **Enter** after renaming it!)
- 6 **Double-click** on ground in the **Hierarchy** to auto-adjust the **Scene** view.
- 7 A **Point Light**, will create light even if the directional light is facing away. Create a **Point Light**. Set your **Point Light's Position** to X 0 Y 3 Z 0.







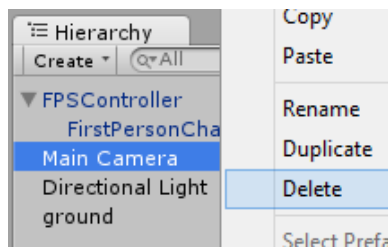
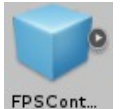
- Select the **Main Camera** to see **Camera Preview** in the **Scene**.
- Move around in the **Scene** with **WASD** and *Mouselook* (while holding the **Right-Mouse Button**), then with the **Main Camera** still selected, from the **GameObject** menu, select **Align With View**, or just press **Ctrl+Shift+F** or **⌘+Shift+F**.
- You should notice the **Camera Preview**, and the **Inspector** changed.
- This **Align With View** can change any selected **Transform**.

7. Use the Project User Interface



- The **Project** window is a list of **Assets** (Game Objects, pictures, sound, scripts, ...) that *could be* in your **Scene**. It's different from the **Hierarchy** window, which is a list of **Game Objects** that *are* in your **Scene**.
- You can **Import** new assets with the **Assets** menu item.
- The **Assets** folder in **Project** mirrors the actual file system! To see it in the file system, press the **Right-Mouse Button** on **Assets**, then **Show in Explorer** or **Reveal in Finder**.
- Use the slider at the bottom to change icon size.

- 1 To import a **Character Controller** from the **Standard Assets** Library, select the **Assets** menu, then **Import Package** and Characters.
- 2 You can press **Import**, but to make the project smaller and faster, click the arrow  to be  for **FirstPersonController**, **RollerBall**, and **ThirdPersonController**. Then, un-check the  for **RollerBall**, and **ThirdPersonController**. DO Keep **FirstPersonController** checked.
- 3 The **Assets** folder in **Project** now has an arrow . **Click** the arrow to toggle folder listing.
- 4 In the **Project**, navigate Assets → Standard Assets → Characters → FirstPersonController → Prefabs.
- 5 Select the **FPSController**, which looks like a cube in the **Project**, but wireframe **Capsule** in the **Scene**.
- 6 Drag-and-drop the **First Person Controller** into the **Hierarchy**.








- There are now 2 **Main Cameras** in your Game! The **Scene** started with one, and **FPSController** came with one as well (named FirstPersonCharacter)!
- **Delete** the **Main Camera** that was in the **Scene** *first*, the one *not* inside the **First Person Controller**. Select it and press **Delete**, or press the **Right-Mouse Button** over the **Main Camera** and select **Delete**.
- This is not required, but may help with debugging scripts later!

- 7 Set the **Position** of the **First Person Controller** to X 0 Y 0 Z 0.
- 8 Rename the **FPSController** to something like “player”, to help keep things simple for later.
- 9 **Save** your **Scene** by selecting **File**, and **Save Scene** (**Ctrl+S** or **⌘+S**). Name it something like “tutorial”, and be sure to save it in the **Assets** folder, or in a folder inside of it.
- 10 **Save** often, especially before you try something new that you have never done before!



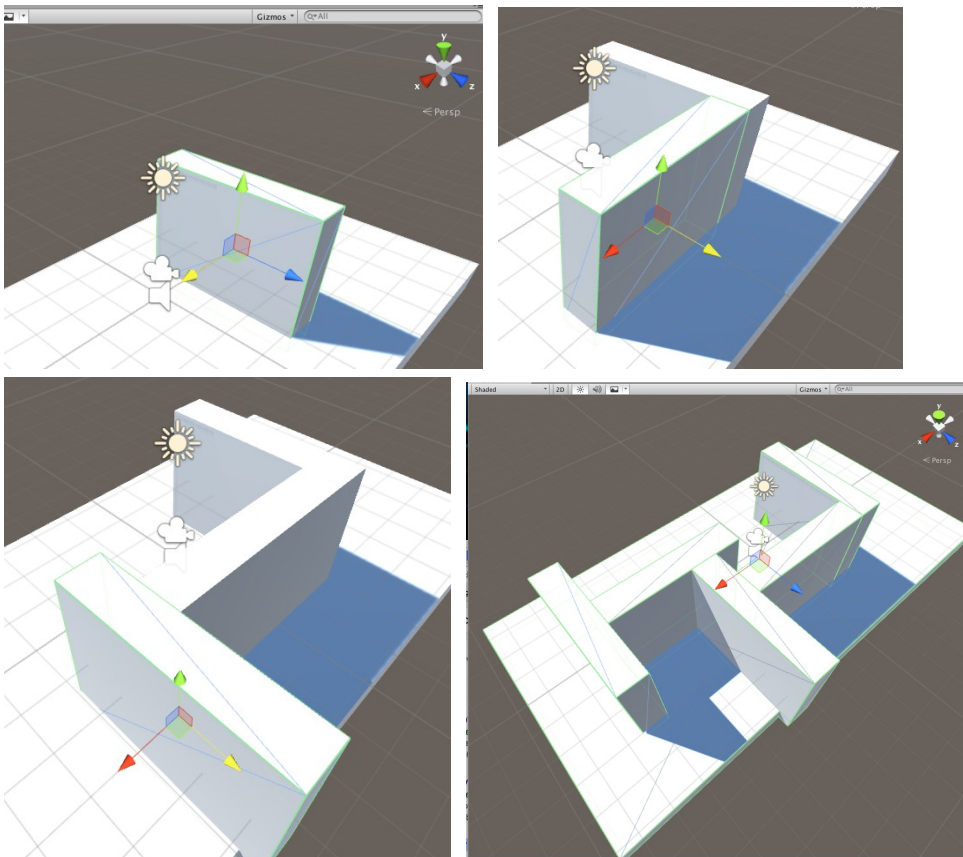
8. Play-Test

- 1 Press the Play  button (in the middle, near the top, with one of these buttons   ).
- 2 The **Game** view is now active, and the rest of the UI changed color *very slightly*.
- 3 To move the **FPSController** (that was renamed to **player**), move the mouse (*Mouselook*), and use the **WASD** keys for movement, just like in the **Scene**. You can use **Space Bar** to jump!
- 4 If you fall, you can stop and restart the game by pressing .
- 5 If your game seems broken, set the **player's Position** to X 0 Y 0 Z 0. Also, make sure you have a **ground cube** at **Position** X 0 Y -2 Z 0 and the **ground cube's Scale** of X 20 Y 1 Z 10. The player might be partially *in* the ground cube.

- 6 While the game is playing, select the **player** in the **Hierarchy**, then **Click** back into the **Game** to give mouse and keyboard *focus* back to the game. Notice that when you move the player, the **Inspector** shows the new state of the player's **Transform**.
- 7 You can change the details of any **Game Object** while the game is running with the **Inspector**. BUT! **Any changes you make while in Play Mode can't be saved!**
- 8 You should adjust your Unity settings to *more* clearly show when the game is running, so you don't accidentally make changes that can't be saved. Go to **Edit**, **Preferences...**, **Colors**, and change the **General** Playmode tint to something obvious. Like, hot pink.
- 9 **Click** on the colored rectangle, and select a color using the color chooser.

9. "Iterate" On Your Game

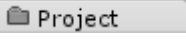
- *Iterate means to do something over-and-over again.* Game Developers use that word to describe working on a game. Most of real Game Development is the same few things over and over:
 - 1 Come up with an idea to try
 - 2 Use tools (like Unity3D) to try it out
 - 3 Test it out, to see if it works like expected
 - 4 If it doesn't work, or there's more to do (there almost always is), go back to the first step!
- This is also called the *Iterative Process*, and it is also similar to the *Scientific Method*.
- Try to add more **Game Objects** and move them around to create a more interesting **Scene**:
 - 1 Try to use several cubes and lights to build a maze. You can **Scale** the cubes into walls, and even additional ground *platforms*!
 - 2 Add lighting. Don't be afraid to edit objects in the **Inspector**. Unity can **Undo** (**Ctrl+Z** or **⌘+Z**) and **Redo** (**Ctrl+Y** or **⌘+Y**) any of your changes. Also, don't forget to save often (**Ctrl+S** or **⌘+S**)!
 - 3 *Play-test* your game after making some changes, to see if it is more fun!
- You could try to make a simple maze, like this (6 **Cubes**, 1 **Directional Light**, and a **FPSController**):

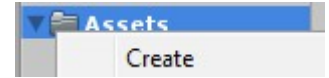


- To **Duplicate** a **Game Object**, press the **Right-Mouse Button** over it in the **Scene** and select **Duplicate**, or select it and press **Ctrl+D** or **⌘+D**.
- For more *digital* (or *snap-to-grid*) control while adjusting **Game Objects** with the **Position**, **Rotation**, and **Scale** **Gizmos**, hold **Ctrl** or **⌘**, while **Dragging** the mouse.
- The(p)osition and (s)cale of the 6 cubes in the example ← on the left:
 - p(0,-2,0) s(20,1,10)
 - p(-3,0,0) s(1,4,5)
 - p(0,0,2) s(5,4,1)
 - p(3,0,3) s(1,4,5)
 - p(4,0,0) s(5,4,1)
 - p(7,0,0) s(1,4,5)

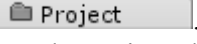
10. Add a Script Component: Restart After Falling

- 1 To bring the *player* back to the beginning if the *player* falls off of a platform, we will need a *script*.
- 2 There's a LOT to learn about *scripting*, which is a way of saying *simple programming*. During this part of the tutorial it may be helpful to have someone who knows about programming around to help you!

- 3 Press the **Right-Mouse Button** over the **Assets** folder, in the . Then select **Create**, and **C# Script**. **C#** is pronounced “See Sharp”.



- 4 Name the script **Fall**, with a capital 'F'. You can also rename it with the **Right-Mouse Button** menu.

- 5 **Double-click**, or press **Enter** on the **Fall** script in the . Then, wait a minute or two! The script editor, **MonoDevelop**, takes some time to load.



- 6 Add to the **Fall** script in **MonoDevelop**, so that it looks like this:


```
Fall.cs x
selection
1 using UnityEngine;
2 using System.Collections;
3
4 public class Fall : MonoBehaviour {
5     Vector3 startLocation;
6     public float lowestAllowed = -20;
7     // Use this for initialization
8     void Start () {
9         startLocation = transform.position;
10    }
11
12    // Update is called once per frame
13    void Update () {
14        if(transform.position.y <= lowestAllowed) {
15            transform.position = startLocation;
16        }
17    }
18 }
19
```

- If **MonoDevelop** opens multiple times, just close the extra *instances*.
- **Unity3D** and **MonoDevelop** support several scripting languages, including:
 - **C#**: the *strictest* option. *Strict* means it requires more code, but that helps MonoDevelop offer better *auto-complete*. C# Supports entire .NET 2.0 library. This is the most common choice.
 - **JavaScript**: or *UnityScript*, which has fairly simple *syntax*, very close to *web JavaScript*.
 - **Boo**: closely related to *Python*. Least common and least supported by Unity communities.


- 7 The *lines* with the **green** next to the *line number* are the ones you will need to type.

- 8 Copy this script **exactly**. Every *capital letter* is *capital*, and every *lowercase letter* is *lowercase*.

- 9 The *curly braces* **{ }** are on the keyboard near the Enter key, above and to the right of your right pinky.


- 10 Save the file in **MonoDevelop** (with **Ctrl+S**, **⌘+S**, or select the **File** menu, then  **Save**).

- 11 You can check to see if your script is correct in Unity by checking the bottom bar of the User Interface:


this example error is a simple case of misspelling: “*transfrom*” instead of “*transform*”

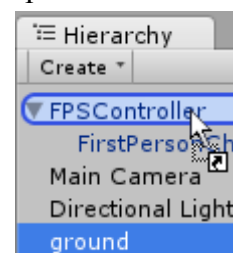
- 12 If there is an error, read the error message and use it to help you discover what is wrong, to solve the problem. This is where having someone who knows about programming will be very helpful!


- 13 If there are no errors, *drag-and-drop* the **Fall** script from the  onto *player* (which is the renamed **FPSController**).

- 14 You should notice a  component in *player*. Scroll down in the **Inspector** to find it.

If you accidentally added the script more than once, remove extras by pressing the

Right-Mouse Button over the extra script, and select **Remove Component**.



- 15 You can change how the Fall script works by changing **Lowest Allowed** in the  **Inspector**.
- 16 Test out the script by jumping from your game area! If it does not work, think about what might fix it, and try it out, or ask a friend to help you *Debug* your problem. *Debugging*, or *Troubleshooting*, which means *fixing-computer-problems*, is a very important and powerful skill for a *Game Developer* to develop for themselves. Think of these problems as challenges to improve your skills with!

11. More Scripting Examples

- Here are some more scripts you can try to figure out and play with (each script will need its own file):

ClickColor.cs

```
using UnityEngine;
using System.Collections;

public class ClickColor : MonoBehaviour {
    Color originalColor;
    public Color color = Color.red;
    void Start () {
        originalColor = renderer.material.color;
    }
    void OnMouseDown () {
        if(renderer.material.color == originalColor) {
            renderer.material.color = color;
        } else {
            renderer.material.color = originalColor;
        }
    }
}
```

TouchSensitive.cs

```
using UnityEngine;
using System.Collections;
// put this on the player
public class TouchSensitive : MonoBehaviour {
    void OnControllerColliderHit(
        ControllerColliderHit hit) {
        Damage dmg = hit.gameObject.
            GetComponent<Damage>();
        if(dmg != null) {
            dmg.DoDamage(this.gameObject);
        }
    }
}
```

Damage.cs

```
using UnityEngine;
using System.Collections;
// put this on something that damages the player
public class Damage : MonoBehaviour {
    public void DoDamage(GameObject go) {
        Fall f = go.GetComponent<Fall>();
        if(f != null) {
            go.transform.position =
                new Vector3(0, f.lowestAllowed, 0);
        }
    }
}
```

FoundIt.cs

```
using UnityEngine;
using System.Collections;

public class FoundIt : MonoBehaviour {
    public GameObject player;
    public float distance = 2;
    public string message = "You win!";
    public static GameObject FindGobj(System.Type t) {
        object[] f;
        f = UnityEngine.Object.FindObjectsOfType(t);
        if(f != null && f.Length > 0) {
            return ((CharacterController)f[0])
                .gameObject;
        }
        return null;
    }
    void Start() {
        if(player == null) {
            player = FindGobj(
                typeof(CharacterController));
        }
    }
    void OnGUI() {
        if(player == null) {
            print("No player looking for "+name+"!");
            Destroy(this);
            return;
        }
        float calculatedDistance = Vector3.Distance(
            transform.position,
            player.transform.position);
        if(calculatedDistance < distance) {
            GUILayout.Label(message);
        }
    }
}
```

- These scripts can create color-changing objects, goal objects, and penalty objects that restart the player!
- Someone who feels comfortable with Programming may want to check out the **Monobehaviour** documentation at <http://docs.unity3d.com/Documentation/ScriptReference/MonoBehaviour.html>.

12. What Else Can Unity3D Do?

- There is still SO MUCH that Unity3D can do to create games! You can use scripting to implement any kind of gameplay you can think of, but scripting is only one part of it! You should also know about:
 - Materials**, to change the color and texture of objects
 - Particle Effects**, to make explosions water effects, and things that look like magic!
 - Rigidbody Physics**, to allow objects to interact with each other in interesting ways
 - 3D Models and Animations**, which you can import from tools like *Blender*, *3DS Max*, or *Maya*
 - Music and Sound Effects**, which can add a lot to your game
 - And so much more! **Terrain** editors, *Web* and *Mobile* development, **Shaders**, **Lighting**, *Novel Game Controllers*, *Networking*, Graphical User Interface (**GUI**), the Unity **Asset Store**, ...
- 1 Find the Unity3D User Guide is at: <http://docs.unity3d.com/Documentation/Manual/UserGuide.html>