**Unity Tips & Tricks**

By Mike Riley

Table of Contents

[Purpose of this Document 4](#_Toc78370690)

[Unity Interface Tips 5](#_Toc78370691)

[Moving between 2D and 3D Project Views/Modes 5](#_Toc78370692)

[Adding Background Image to Fill Canvas 5](#_Toc78370693)

[Adding Scenes to the Project 5](#_Toc78370694)

[Making Canvas Scale with the Screen Size 6](#_Toc78370695)

[Anchoring Screen Elements Relative to the Screen Size 7](#_Toc78370696)

[Useful Web Pages for Use with Unity 8](#_Toc78370697)

[GitHub.COM – Shared Project Database 8](#_Toc78370698)

[DoFont.COM – Fonts for use with Your Projects 8](#_Toc78370699)

[OpenGameArt.ORG – Downloadable 2D & 3D Art, Music, Textures, and Sound Effects 8](#_Toc78370700)

[Freesound.ORG – Website with Free Sound Files for Download 8](#_Toc78370701)

[App.Diagram.Net – Online Flowchart Drawing Program 8](#_Toc78370702)

[Adobe Color Wheel 8](#_Toc78370703)

[Share My Game 8](#_Toc78370704)

[RenderDoc – Graphics Debugger 9](#_Toc78370705)

[Unity Answers Discussion Group 9](#_Toc78370706)

[Las Vegas Unity User Group 9](#_Toc78370707)

[Mirror Networking 9](#_Toc78370708)

[Game Programming Patterns 9](#_Toc78370709)

[Krita – Free Paint Program 9](#_Toc78370710)

[2D Game Art Assets 9](#_Toc78370711)

[Webinars & YouTube Instruction Videos 10](#_Toc78370712)

[Text Mesh Pro – Using Fallback Font Settings 10](#_Toc78370713)

[Useful Unity Manual Pages 11](#_Toc78370714)

[Order of Execution for Event Functions 11](#_Toc78370715)

[Useful Packages from Unity 12](#_Toc78370716)

[Device Simulator 12](#_Toc78370717)

[Useful Assets from the Unity Asset Store 13](#_Toc78370718)

[Text Mesh Pro – Comes with Unity 13](#_Toc78370719)

[Agora Video SDK for Unity – Text & Video Chat Plug-in 13](#_Toc78370720)

[Generally Useful Code Tricks 14](#_Toc78370721)

[Exit Game in Unity Editor 14](#_Toc78370722)

[Laser Defender from Udemy Complete Unity Game Developer 2D Course 15](#_Toc78370723)

[Trigger Sound Effects (Explosions, dropping a Bomb, Firing a Laser, etc.) 15](#_Toc78370724)

[Delay for Loading a Scene 15](#_Toc78370725)

[Music Player with Singleton (Lesson #114) 15](#_Toc78370726)

[Spinning Effect on a Sprite 16](#_Toc78370727)

[Glitch Garden from Udemy Complete Unity Game Developer 2D Course 17](#_Toc78370728)

[Startup Audio for Loading Game 17](#_Toc78370729)

[Setting Canvas to Match World Units Scaling and Scaling Camera to Match 17](#_Toc78370730)

[Placing Sprite Images (Trees) on Canvas Easily 17](#_Toc78370731)

[Slicing a Sprite Sheet for Animation 18](#_Toc78370732)

[Adding Animated Sprite to Game 19](#_Toc78370733)

# Purpose of this Document

This document is to help explain how to best use features of Unity from the Udemy courses on Unity. So it will include references to specific lessons and lectures. Later I will be adding references to other documentation, including web pages and videos that I have also found useful for Unity.

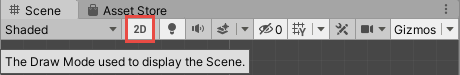
The hope is that this will provide a quick way to look up how I did something for a particular project, so it can be used in future projects.

# Unity Interface Tips

## Moving between 2D and 3D Project Views/Modes

In case you accidentally create a project in the wrong mode you can change it in Unity using the settings for the editor.

You can click on the 2D button as shown to switch between 2D and 3D views:



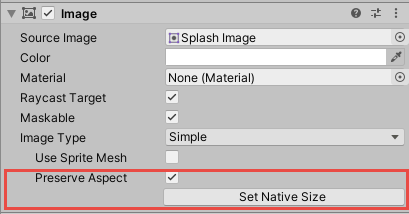
Next you must choose the Edit 🡪Project Settings 🡪 Editor from the menus. As of Unity 2019, selecting Project Settings will open a dialog box from which you can choose Editor. You then need to change Default Behavior Mode to be the 2D or 3D setting you selected with the Scene button.

## Adding Background Image to Fill Canvas

With the image already in a folder under Assets, right-click on Canvas and select UI 🡪 Image to put an image on it. Be sure to set the correct aspect ratio or resolution setting you need to use. You can one to the list by clicking the plus (+) sign as I did here to add 1080P with 1920x1080 resolution:



Click on the *Image* item in the hierarchy to show it in the Inspector. In order, select *Preserve Aspect* and then click on the *Set Native Size* button. This will make the image the same size as the canvas, assuming it is the exact same resolution.

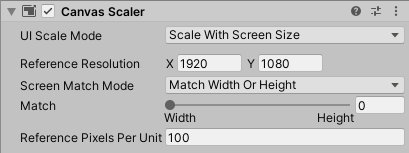


## Adding Scenes to the Project

Click on the *File 🡪 Build Settings* menu selection in the Unity menus. Clicking on the *Add Open Scenes* button will add your current open scene. You can drag any other scenes, or all of them, into the *Scenes in Build* text box. Each scene has a check box next to the name that must be selected for the scene to be enabled. You can drag them into whatever order you wish to have them kept, which will change the scene numbers.

## Making Canvas Scale with the Screen Size

Select the Canvas in the Hierarchy and in the Inspector open the **Canvas Scaler (Script)** section for viewing. Set **UI Scale Mode** to **Scale with Screen Size**. For **Reference Resolution** set it to the native resolution you want as a default to scale from. This should be a size that will hold your largest image, which is usually your background image, say 1920x1080.



Now when you look in the Scene mode you should see the blue dots for the corners matching your background image.



You can also set **Screen Match Mode** to **Match Width or Height**, **Expand**, or **Shrink**. Shrink will cut of content near the edges, which is okay if that scene has nothing important there.

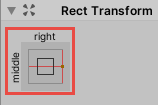
These modes can leave your canvas in a letterbox with areas on the size or above and below that are outside your canvas area. By default, these appear blue, but the convention is to make them black. You can set this color by selecting the **Main Camera** and setting the **Background Color** to black (RGB 0,0,0).

## Anchoring Screen Elements Relative to the Screen Size

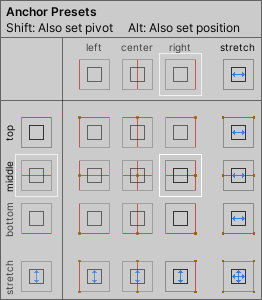
A good example of this is when you want text or buttons to be relative to the edge of the screen. Since the screen size may change on you, depending on the display device being used, this keeps important screen elements visible regardless of the aspect ratio. See **Making Canvas Scale with the Screen Size** for an example of how to set your canvas so it is independent of the screen size and aspect ratio. This is typically true if you have set the **Screen Match Mode** to **Shrink** for the Canvas.

As an example, once you have placed some text relative to your default screen size, you should test it with different aspect ratios to see how it looks in 16:9, 4:3, etc. If elements would fall outside the visible, you should consider anchoring them to be relative to some place on the screen. The places are the center, the corners, or the sides of the screen.

You select the anchor point for an item by opening the **Rect Transform** properties in the Inspector and clicking on this element in it:



Clicking on this button (for lack of a better word) will bring up a sub-window element that lets you choose to what place on the canvas you wish to anchor this object. The currently selected choice is highlighted with a white box around it. In the example below the element will be relative to the middle of the right side of the canvas:



Once you have chosen anchors for the elements you wish to have them, test with the different aspect ratios and resolutions that are defined. You can add your own to the list that Unity comes with, if you have specific ones you know you will need to support that do not fit the standard list.

# Useful Web Pages for Use with Unity

## GitHub.COM – Shared Project Database

The website is: <http://github.com>

This is a great place to store your projects, especially is using them with a development team. You can have different branches for experimenting with features in your project. It also gives you a way to go back to known good version in case you get lost going down a rabbit hole later.

It also allows you to work on the same project using multiple computers, by simply downloading the latest one onto the system you are working on at the time.

The GitHub Desktop utility gives you an easy GUI interface to use to create and maintain your projects. You can download it from the GitHub website.

## DoFont.COM – Fonts for use with Your Projects

The website is: <http://www.dafont.com>

Many fonts are free to use here. They work great with Text Mesh Pro text fields.

Once you download a font (pay attention to license requirements!) you can extract it under Assets. Using a Fonts folder with a subfolder of the font name is best. On Windows, double click on the TTF file and click on the Install button of the window that opens up.

To add a font, first select the TTF file from your assets (icon is: “Aa”), then use Assets 🡪 Create 🡪Text Mesh Pro 🡪 Font Asset to create the Text Mesh Pro asset in your project from the TTF file. The icon for the is a capital “F”. Now it will appear in the list of fonts for your Text Mesh Pro object.

## OpenGameArt.ORG – Downloadable 2D & 3D Art, Music, Textures, and Sound Effects

The website is: <http://opengameart.org/user>

This is used in the Laser Defender project in the Unity 2D class by Udemy in lesson #114: Music Player With Singleton. While you may use a free download, please put an attribution about anything you may use in a game. The license terms will vary based on what the artist that posted it wants applied to it. You can use what is there, even for commercial projects. Be sure to read the FAQs about the different license types.

## Freesound.ORG – Website with Free Sound Files for Download

This is mentioned in the Laser Defender lessons #111 – Trigger Sound Effects.

## App.Diagram.Net – Online Flowchart Drawing Program

This is a useful website for drawing State Diagram flowcharts. It is completely free to use. The extension is called .drawio for the files it creates.

## Adobe Color Wheel

The website is: <http://color.adobe.com/create/color-wheel>

This is a free website from Adobe that gives you a selection of colors that go together for use in your game design. Colors are shown with the hex values, which makes them easy to use in Unity.

## Share My Game

The website is: <http://sharemygame.com>

This is a location where you can share your game for others to see and try out for free. It is only there for a limited time, but you can post updates to it at any time.

## RenderDoc – Graphics Debugger

RenderDoc is a free MIT licensed stand-alone graphics debugger that allows quick and easy single-frame capture and detailed introspection of any application using Vulkan, D3D11, OpenGL & OpenGL ES or D3D12 across Windows 7 - 10, Linux, Android, Stadia, or Nintendo Switch™.

## Unity Answers Discussion Group

The website is: <http://answers.unity.com/index.html>

This is a general discussion group where you can ask questions about Unity and search the list for possible answers. The search capability isn’t that great though. You can sign up to have questions emailed to you so you can see them and even answer them. I like to file useful ones away for future reference.

There are other links there for Forums, Unity User Groups

## Las Vegas Unity User Group

The website is: <https://www.meetup.com/Las-Vegas-Unity3D-Meetup/>

Their Facebook group is: <https://www.facebook.com/groups/lasvegasunitymeetup/>

They also have this website: <https://bitbucket.org/LasVegasUnityMeetup/meetupprojects>

## Mirror Networking

The website is: <https://mirror-networking.com/>

They provide a high level networking API for Unity that provides several different types of low level transports. Supposedly you can make it so both the server and client run from the same code base.

## Game Programming Patterns

The website is: <https://gameprogrammingpatterns.com/>

This is someone that wrote a book on patterns used to program games. The book is available for free via the website, or you can buy one.

Here is the online version: <https://gameprogrammingpatterns.com/contents.html>

## Krita – Free Paint Program

The website is: <https://krita.org/en/>

## 2D Game Art Assets

The website is: <http://www.2dgameartguru.com>

Tutorials on creating game 2D art using lots of software for creating the art. He uses Inkscape primarily, which is free. By using vector drawing tools it is easier for non-artists to create images, he says.

# Webinars & YouTube Instruction Videos

## Text Mesh Pro – Using Fallback Font Settings

<https://create.unity3d.com/thank-you-text-mesh-pro-webinar>

# Useful Unity Manual Pages

## Order of Execution for Event Functions

This is described on <http://docs.unity3d.com/Manual/ExecutionOrder.html> and has a state diagram chart that is very useful. It has references to many other methods that can be useful in the text after the chart.

# Useful Packages from Unity

## Device Simulator

There is a simulator that will simulate all sorts of phone and device screens. In the case of some Apple phones, you have a notch, so it simulates that.

Go to Windows 🡪 Package Manager and when it is up select Advanced and enable Preview, since it is currently a preview. You enable it by going to Window 🡪 General 🡪 Device Simulator.

Note: After you install it, you must restart Unity for the Device Simulator option to be shown.

<https://docs.unity3d.com/Packages/com.unity.device-simulator@2.2/manual/index.html>

# Useful Assets from the Unity Asset Store

## Text Mesh Pro – Comes with Unity

There is one in the standard API included with Unity now, so you don’t need to download it from the Asset Store. It is better to use this for any text items than the older Text class.

There are two types for this type of text object. The TextMeshPro type can be used in 3D space. The TextMesshProUGUI type is what you use on a canvas, which is the UGUI environment that it requires. You might use the TextMeshPro type for showing some health info or other text that needs to follow something within a scene, for example.

## Agora Video SDK for Unity – Text & Video Chat Plug-in

Their website is : <https://www.agora.io/en/unity/>

I set up a login here: <https://sso.agora.io/en/login/>

This was recommended to me. I still don’t understand how the licensing would work for it, but it looks very capable to allow for Teams-type video conferencing for games.

You can find it under Home -> Tools -> Video -> Agora Video SDK for Unity

# Generally Useful Code Tricks

## Exit Game in Unity Editor

To exit a game while in the Unity editor you can’t use the same method as you would for a standalone game. Note that the logic must be disabled, so it doesn’t even compile, when you build it for a standalone game. If you don’t, you will get an error that prevents it from running as a standalone program.

Here is the code I use:

public void ExitGame()

{

if (Application.isPlaying & !Application.isEditor)

Application.Quit(); // We may return from this, but the program will terminate at the end of the frame

#if true

else

UnityEditor.EditorApplication.isPlaying = false; // Handle being in the editor, but set #if to true to use it

#endif

} // ExitGame()

You simply change #if true to be #if false when you build for a standalone game and it will run just fine. This is useful for testing that your exit game logic works while in the Unity Editor.

# Laser Defender from Udemy Complete Unity Game Developer 2D Course

## Trigger Sound Effects (Explosions, dropping a Bomb, Firing a Laser, etc.)

While most audio formats are supported, in Laser Defender he used OGG files for these effects. Here is example code used to create a bomb and play the sound. Note that the position in 3D space for the sound to play is that of the camera. Changing that position would change the volume and possibly where the sound seems to come from (assuming Surround Sound is being used).

Here is an article on what audio formats are supported by Unity: <http://support.unity.com/hc/en-us/articles/206484803-What-are-the-supported-Audio-formats-in-Unity->

private void Fire()

{

GameObject bomb = Instantiate(projectile, transform.position,

Quaternion.identity) as GameObject;

bomb.GetComponent<Rigidbody2D>().velocity = new Vector2(0, bombSpeed);

AudioSource.PlayClipAtPoint(bombDropSFX, Camera.main.transform.position,

bombSoundVolume);

} // Fire()

## Delay for Loading a Scene

You sometimes need to delay a bit before continuing, like loading the next scene. Calling a method like this will delay for the number of seconds you specify in the variable delayInSeconds:

void SomeMethod()

{

yield return StartCoroutine(WaitToLoad());

}

IEnumerator WaitToLoad()

{

yield return new WaitForSeconds(delayInSeconds);

SceneManager.LoadScene("GameOver");

} // WaitToLoad()

You can just have the WaitForSeconds() call in here to have a more general delay method and pass the number of seconds to delay as a parameter to the method and it will return to the caller once the delay period is over. This example is one that was designed to load the Game Over scene using a value you can set in the Unity Editor. Since WaitToLoad() is running in a separate thread, whatever is to be delayed **must** be in the method.

## Music Player with Singleton (Lesson #114)

* Download a music track and put it in an *Asset* folder called *Music*.
* Create an Empty Game Object (reset the Transform info, as usual) and add an Audio Source component.
  + Be sure to click on the Loop checkbox to play it in a continuous loop.
  + Add a new script called MusicPlayer to the object.
* Drag it into your Prefab folder to make it a prefab and put it in your other scenes.
* Change the Start() method in the MusicPlayer scripts to be Awake() and use code like the following to make it a Singleton object:

void Awake()

{

if (FindObjectsOfType(GetType()).Length > 1)

Destroy(gameObject); // Destroy any that come afterwards

else

DontDestroyOnLoad(gameObject); // Make the first one immortal

} // Awake()

## Spinning Effect on a Sprite

This is lesson video #118 and is implemented in the Spinner.cs script. I changed the name to specify how many times it would spin per second. You simply add this line in the Update() method to spin on the Z axis:

gameObject.transform.Rotate(0, 0, degreesOfSpinPerSecond\* Time.deltaTime);

# Glitch Garden from Udemy Complete Unity Game Developer 2D Course

## Startup Audio for Loading Game

In the Splash Screen scene do a *Create Empty* to create a new *Game Object* and add an Audio Source component to it. Recommend doing a Reset of the Transform position of the Game Object, because sound output had a position. Then simply add the audio file to the Audio Source component. Select Play On Awake checkbox to enable playing when the scene starts.

## Setting Canvas to Match World Units Scaling and Scaling Camera to Match

* We want 1 Grass Square = 1 World Unit.
* Canvas width is 1920 pixels.
* Our Squares are 160x160 pixels.
* Number of Squares = 1920/160 = 12.
* We want 12 World Units as our width.
* The canvas needs to be scaled by 12 / 1920 = 0.00625.
* Scale the canvas so 1 World Unit = 1 Square.
* Resize and align your camera.
* Repositioned canvas to (5,3), but not sure why those numbers were chosen yet.
  + Position the camera to the same location to center it on the canvas.
  + Set the camera Size to be (CanvasHeight / SquareHeight) / 2 = (1080 / 160) /2 = 3.375

Create a new Sprite in the hierarchy (knob is a default one for Unity and works well). Positioning the sprit at (1,3) you should see it in the center of the leftmost column, right in the middle of the middle square.

## Placing Sprite Images (Trees) on Canvas Easily

* Add tree images to Canvas (they will appear near 0,0).
* Change *Order in Layer* to be 10 (Canvas is 0 and Defender and Attacker images will be 5, so this will cover them).
  + You normally only have a Sorting layer in the *Sprite Renderer* section for the image.
* Resize if needed after positioning the image, but be sure not to completely cover any playing area square (partial is okay).
  + If necessary, you can leave part of the image off the edge of the camera view.

Note that instead of numbers you can name your layers, which will be easier to keep track of if you have a lot of layers. You do this by clicking on Sorting Layer, which is Default when using just numbers, and selecting *Add Sorting Layer*. This opens a different panel in the Inspector area with the list of layers. *Default* is the first layer (at the top) normally. You can add new layers by clicking on the plus sign (+), which creates a new place to put the name of your layer. Layers at the top of the list are “higher” than ones lower in the list. You can move a layer name by clicking on the word layer and dragging it to a new position in the list. To delete a named layer drag it to the end of the list and click the minus sign (-). You can get out of the layer list by clicking on anything in the hierarchy.

## Slicing a Sprite Sheet for Animation

A sprite sheet is a PNG with multiple images placed on a transparent background (PNG images support transparency, but JPEG does not). To animate the sprites, you need to “slice” the images up into individual images.

* Click on the icon for the sprite sheet image you need to slice.
* In the Inspector, change *Sprite Mode* to be *Multiple* (default is *Single*).
* Click on the *Sprit Editor* button to bring up the Sprite Editor.

|  |  |
| --- | --- |
|  |  |

The Sprite Editor is a pop-up dialog box that you close once you have completed your editing. The image above shows the menu when you click on the *Slice* button. You may need to widen the dialog slightly to see all the buttons as shown.

* Click on the *Slice* button to open that sub-dialog option.
* Leaving it set for *Automatic* and click on the *Slice* button at the bottom of the sub-dialog box.
* This will put a white outline around each image in the sheet.
* Click the *Apply* button (which was greyed out before doing the slice operation) to apply the changes.
  + Clicking on the *Revert* button will revert the slicing changes.
* Click on any image in the sheet to see the name and property of any sub-image in the sheet.
  + You will see info about the sprite image and can change the *Pivot* and *Pivot Unit Mode* for that image.
  + The Sprite Sheet will display in the Assets folder with an arrow that can be clicked to show all the sprites in it. Clicking on an individual sprint image brings up the properties in the Inspector.

Table

Description automatically generated with medium confidence

## Adding Animated Sprite to Game

To add an animated sprite into the game, do the following:

* Right click in the *Hierarchy* and select *Create Empty*, changing the name to what you want to call the sprite.
  + Don’t forget to do a *Reset* for the *Transform* section in the Inspector.
* Click on *Add Component* button and add a *Sprite Renderer*.
  + For *Sprite* choose the frame you wish the sprite to start animating with (*Lizard\_Walk\_7* in the lesson example).
* Be sure the Animator and Animation windows are somewhere in your Unity UI
  + You can enable/disable these using the *Window* 🡪 *Animation* menu selection submenu.
  + In the Animation tab you can open the menu using the three dots on the far right and enable feature to be displayed, like *Frame Rate*.

Terminology for animations:

* **Animator Component** – Assigns animations to *GameObjects* through an *Animator Controller*.
* **Animator Controller** – Arrangement of animations and transitions (state machine).
* **Animation** – Specific pieces of motion images.
* **Sprite Renderer** – Displays the 2D sprite on the screen

Here are the steps to creating the animation sequence:

* In your *Assets* folder create a new folder called *Animations*. You may want to have subfolders for multiple animations.
* In your folder right-click and create an *Animation Controller* (name it for what makes sense to you).
  + Selecting the Animation Controller and clicking on the tab for the *Animator* window will show the states.
  + Name this to a name that makes sense, like *Lizard*.
* Create an *Animation* by:
  + Go to your sprite sheet and clicking on the arrow to show all the frames.
  + Select all the frames you want in the animation.
  + Right-click on one of the selected frames and select Create 🡪 Animation from the pop-up menu.
  + Name it appropriately, like *Lizard Walking*.
  + In the Inspector click on the *Loop Time* checkbox to make sure the animation loops when played.
  + Move this *Animation* object into your Animation folder with the Animation Controller created earlier.
* On the Game Object you added the Sprite Renderer to click on *Add Component* and add an *Animator*.
  + For *Controller* select the name of the controller you created earlier.
* Click on the Animator tab and drag the Animation you created earlier from the Animations folder into the Animator Window, which should have *Entry*, *Any State*, and *Exit* states in it after it was created, and it gets linked with a line to *Entry* automatically.
  + The line is called a *Transition*.
  + When you click on the Play button you should see your animation working now.
    - You may need to adjust the sample rate to make it look good. Do this by selecting the Animation object in your Animations folder and adjusting it in the Animation tab window.