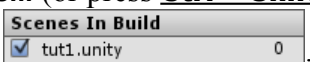
Unity3D Tutorial – Sharing Projects

0. About This Tutorial

1. This is the fourth tutorial in a series of Unity3D tutorials for Windows. This tutorial shows several ways to share a Unity3D project with others, and how to learn from other projects.
2. This tutorial expects that you have your own project in Unity3D, and you want to share it, or learn what else you can add to it from other projects.

1. Build for the Web

1. Select File and **Build Settings...** (or press **Ctrl + Shift + B**) to see the **Build Settings** menu.
2. Check the **Scenes In Build** list 

. This is the list of scenes that your game will include. If the list is empty, press Add Current to put the current scene into the list (you will need to have saved the current scene for this to work).
3. Select Web Player, which is used to make a **Web Player** build, to share your game on the internet.

- Unity3D supports many *build targets* or *platforms*. Changing a game to work on another is usually a complicated and frustrating process called *porting*. Unity3D handles most of that porting effort for you! See the **Platform** list for some platforms Unity3D can build for (some require special licensing).

4. Press Player Settings... to see build settings in the Inspector. **Resolution and Presentation → No Context Menu** is useful if your game uses the right-mouse click for any reason. Each **Platform** has its own **Player Settings** to modify and experiment with.


No Context Menu

5. Press Build And Run. Then, choose or create a folder for your **Web Player** build. Press **Select Folder** to start the build. After the build is done, your web-browser should start the game.
6. If you look in the folder (from in the last step), you will notice two files: a web page (**.html** file), and **Web Player** build file (**.unity3d** file). To put this game on the internet, *upload* those files to a public web-space. Anyone who navigates to the web page can play your game! To learn about making a web page, seek out tutorials on "how to make a webpage" using your favorite search engine!

2. Sharing a Project

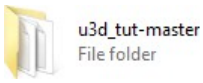
1. Before sharing a project (so that others can add to it, or you can work on it from another computer), it's a good idea to make **Meta Files** visible. Select **Edit → Project Settings → Editor → Version Control → Visible Meta Files**. This will make it easier to work with *source control* systems (like *Git* or *SVN*).
2. Find the **Assets** folder in the file explorer (press the **Right-Mouse Button** on Assets, then Show in Explorer). Find two specific folders: **Assets** and **ProjectSettings**.
3. To share your project, you only need to share the **Assets** and **ProjectSettings** folders. These can be zipped up: **Ctrl + Left-Mouse Button** to select each folder, then **Right-Mouse Button** on a selected folder, **Send to → Compressed (zipped) folder**). The folders could also be *committed* to source control.
4. *DO NOT include the other folders or files*. All other files and folders are temporary, and *regenerated* by Unity3D whenever you modify your project. Again, only share the **Assets** and **ProjectSettings** folders.
5. To open a shared project, select File → **Open Project... → Open Other...**, then select the un-zipped folder that contains **Assets** and **ProjectSettings**.

3. Export/Import Custom Package

1. Unity3D can share parts of a project (without sharing everything) in a **Package**. Select a **Scene** or **Prefab** in the Project, then select Assets → **Export Package...** . Notice that only resources used by the selected **Scene** or **Prefab** are listed. Press Export..., navigate the file chooser to the **Desktop** (or another memorable place), name the package, and press **Save**.
2. To load a package in another project, even on another computer, select Assets → **Import Package** → **Custom Pakage...**, and select the custom package.

- The Export/Import package feature works best when using the same version of Unity3D on the Exporting and Importing computer.

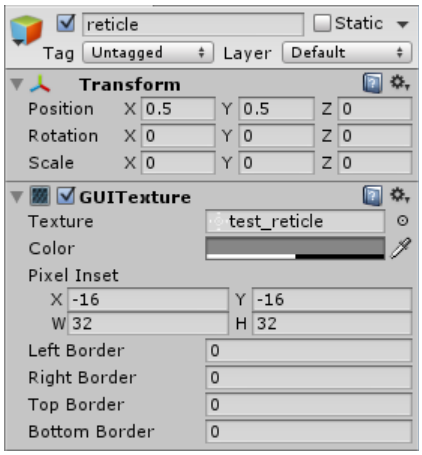
4. Load This Tutorial Project

1. Download the Unity3D project at https://github.com/mvaganov/u3d_tut/archive/master.zip . Unzip this zip archive someplace memorable. More advanced Git users may instead want to *clone* or *fork* from https://github.com/mvaganov/u3d_tut.git for the same effect.
2. In Unity3D, select **File** → **Open Project** → Open Other..., then navigate to 

(the folder where the project's **Assets** and **ProjectSettings** folders are), and Select Folder.

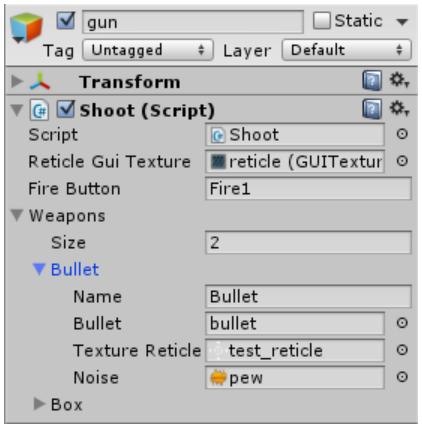
5. Learning About A Unity3D Project

1. The first thing you should do after loading a new project is Play the project! Find a scene in the Project to play. You can always manually search through the Project, but you can also quickly and easily list every scene by using the **Project Search** bar . Next to the Search icon, type "t:Scene", or press the **Search by Type** icon and select **Scene**. For now, select **tut4**, which is the most interesting and complex scene in the project.
2. Play the Scene to verify that it works correctly. There are many new things in this tutorial to learn about but lets start with the static image attached to the middle of the screen, acting like a targeting reticle. We can find it using Unity3D's search feature.
3. In the Project Search bar, type "t:Texture", or press the **Search by Type** icon and select **Texture**. This will find any image in this project. Find the **test_reticle** texture in Project.
4. To discover what is causing the **test_reticle** to appear in the game, **Right-Mouse Button** click on **test_reticle**, and select **Find References In Scene**. Notice the Hierarchy has changed, showing only 2 items: **gun** and **reticle**.
5. Select the **reticle** object first. This is a GameObject with only a **GUITexture** component. This **GameObject** that shows the image.



- **test_reticle** is referenced by **GUITexture**'s **Texture** field. A different texture can be set by a *drag-and-drop*, or by pressing .
- The **Position** values in the **Transform** are the *percentage location* on the screen. 0.5 means 50%, or half-way, which is why the reticle is *anchored* half-way vertically and half-way horizontally on the screen.
- The **Scale** values determine how big to make the image. A scale of 0 tells the texture to draw normally, using no stretching based on screen size.
- The Pixel Inset values move from the anchor and also scale the image.
- **Color** is used to set the color of the image, including it's transparency.

6 Select the **gun** object, then clear the Search in the Hierarchy to see that it is attached to the **player**'s **Main Camera** (explained in the previous tutorial). This object has a the **Shoot (Script)** component, which sets the texture of the **GUITexture** in the **GameObject** reticle.



- This Shoot script is more complex than it's equivalent from the previous tutorial. Notice that this script supports a **Weapons** *array*. Load the script to discover more about how it works, including how to switch weapons.
- Each weapon in the **Weapons** array is a `Serializable` class *data-structure* of type `Weapon`. Unity3D allows easy editing of `Serializable` classes, and lists (like the standard array here, or other lists like `System.ArrayList` or `System.Generic.List`).

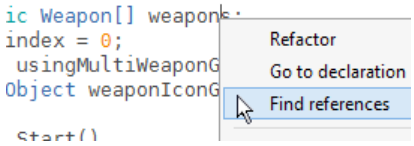
7 Notice that most objects in Hierarchy are *hierarchically* organized, inside of empty **GameObjects**, named by category (enemies, world, lights, platforms). When creating your own Unity3D project, consider doing the same to help organize the scene.

- Organizing objects in *hierarchies* makes it easy to hide information, which helps reduce confusion (also called *Managing Complexity*). Confusion from complexity, is one of the biggest challenges with computers systems of all kinds. The better the complexity is managed, the more complex (and interesting) you can make the system!

6. Learning About Programming

1 Most modern Integrated-Development-Environments (IDEs), like **MonoDevelop**, or *Visual Studio* (the free version is called *Visual Studio Express*), can search too. A *find* option should be available in the **Edit** menu, or by pressing the **Ctrl + F**. This is common even in Web browsers! IDEs will also usually have a *find-in-all-files* option, which should also be in the **Edit** menu, or found with **Ctrl + Shift + F**.

- Some IDEs have *find-references* searches too, which find actual code references instead of just matching text. In **MonoDevelop**, you can find-references with the **Right-Mouse Button** context menu for a variable, function, or class.



2 The best way to learn about programming is to find examples of code that does what you want, and practice implementing it yourself! This project has interesting scripts that you may want to practice implementing in your own game:

- `RestartAfterFall` – Teleport a GameObject to its start location if it is lower than a certain height.
- `ShowMessage` – Print a message to just-below-the-middle of the screen for a specified amount of time.
- `TimedDestroy` – Destroy an object after the specified amount of time.
- `LevelSwitch` – Change which scene the game is playing when a key is pressed.
- `ParticleCulling` – Remove a particle from the game after it is finished emitting.
- `PlaySound` – Play a sound when a button is pressed (**Edit** → **Project Settings** → **Input** for button info).
- `Wander` – Randomly rotate this **GameObject**, and cause it's **CharacterMotor** to move forward.
- `KeyPressChangeBool` – Change a field in another script when a key is pressed (using C# reflection).
- `HitByProjectile` – Trigger code when this **GameObject** collides with a `Projectile` **GameObject**.
- `Projectile` – Cause a **GameObject** to launch forward, and trigger a `HitByProjectile` (even if moving too fast, like the bullet-through-paper problem).
- `Shoot` – Shoot from an array of `Weapon` objects, switching between them with a key press.
- `TriggerArea` – If a specified **GameObject** reaches this **Collider**'s (trigger) area, create other objects.
- `CameraLerp` – Cause the camera to do a **L**inear int**erp**olation (smooth movement) when this is created.

- In your favorite search engine, type "Unity3D" followed by some script you want to be able to do, and you may find free examples of it! You can also consider asking more difficult questions at the official Unity3D forums: http://forum.unity3d.com/.