

Tutorial Master

Quick Start Guide



This is a quick start guide on creating your first tutorial. To learn about Tutorial Master in a greater detail, please visit [support section](#) which is being regularly kept up-to-date.

Table of Contents

Getting Started	3
Terminology	4
Adding a Component.....	5
Module Pooler.....	6
Importing Third-Party Plugins (Optional)	7
Making a Tutorial	8
Creating a tutorial	9
Adding a Stage	10
Stage Trigger	11
Stage Events.....	12
Stage Audio.....	13
Stage Modules.....	14
Running a Tutorial	16
Using Editor.....	17
Using API.....	18
Using Debug Mode.....	20
Useful Links	21

Getting Started

Before we can create our first Tutorial, we need to set up the Tutorial Master first.

Terminology

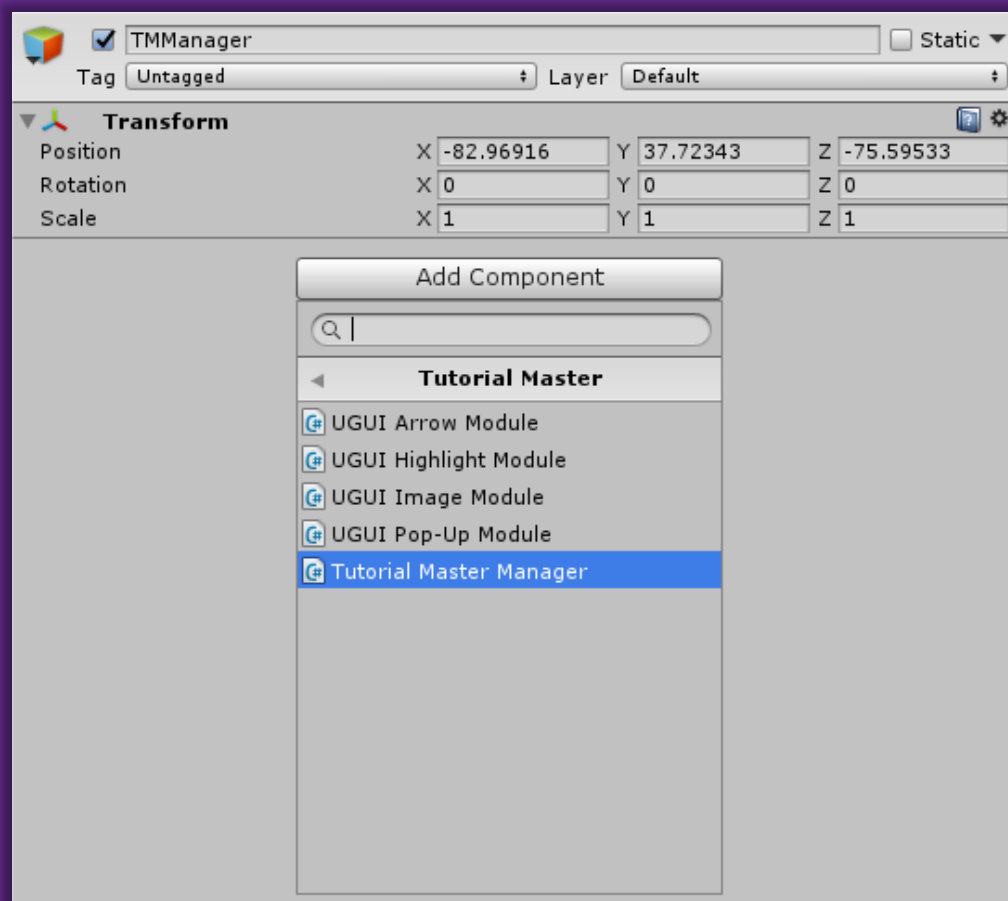
Tutorial Master has its own set of terms, which are best to be familiarized with before starting with the guide.

Tutorial	A set of instructions to guide a player
Stage	Tutorials consist of stages. Each stage carries a single piece of information for a player to understand.
Trigger	A means of proceeding to another part of the tutorial. Examples could be, timers, button clicks or any other event invocation.
Module	Visual guiders that helps player. These could be something as passive as an arrow hovering over a UI element, or as complicated as a Pop-Up box with a message and a clickable button.
Effect	Entrance effects for Modules. These allow player locate Modules much easier and just serve as a nice visual cue in general.
Object Pooling	A software design pattern where an entity is being reused. This saves up performance as you don't have to allocate and deallocate memory on-demand.

Adding a Component

Create a new GameObject in your scene, you can name it whatever you want. It's a good practice to dedicate a single GameObject to Tutorial Master to keep things clean and clutter-free.

Go to *Add Component > Tutorial Master* and select *Tutorial Master Manager*.



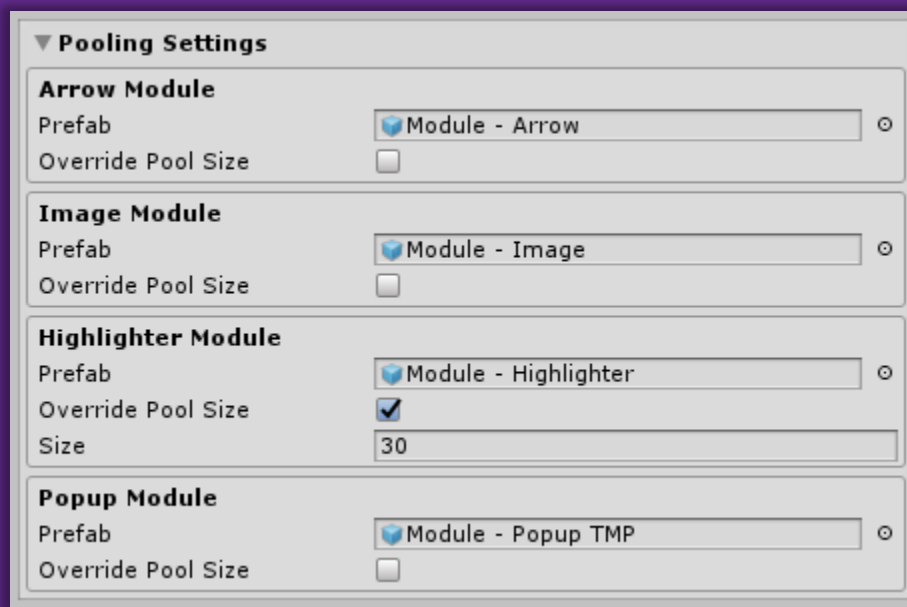
Module Pooler

Module Pooler takes care of reusing module prefabs, thus improving runtime performance.

Tutorial Master comes with ready-made Modules that you can use as placeholders. Let's use them!

1. Locate Module Prefabs in *Tutorial Master > Prefabs*

2. In the "Pooling Settings" of Tutorial Master Manager, drag module prefabs from the "Prefabs" folder into prefab slots.



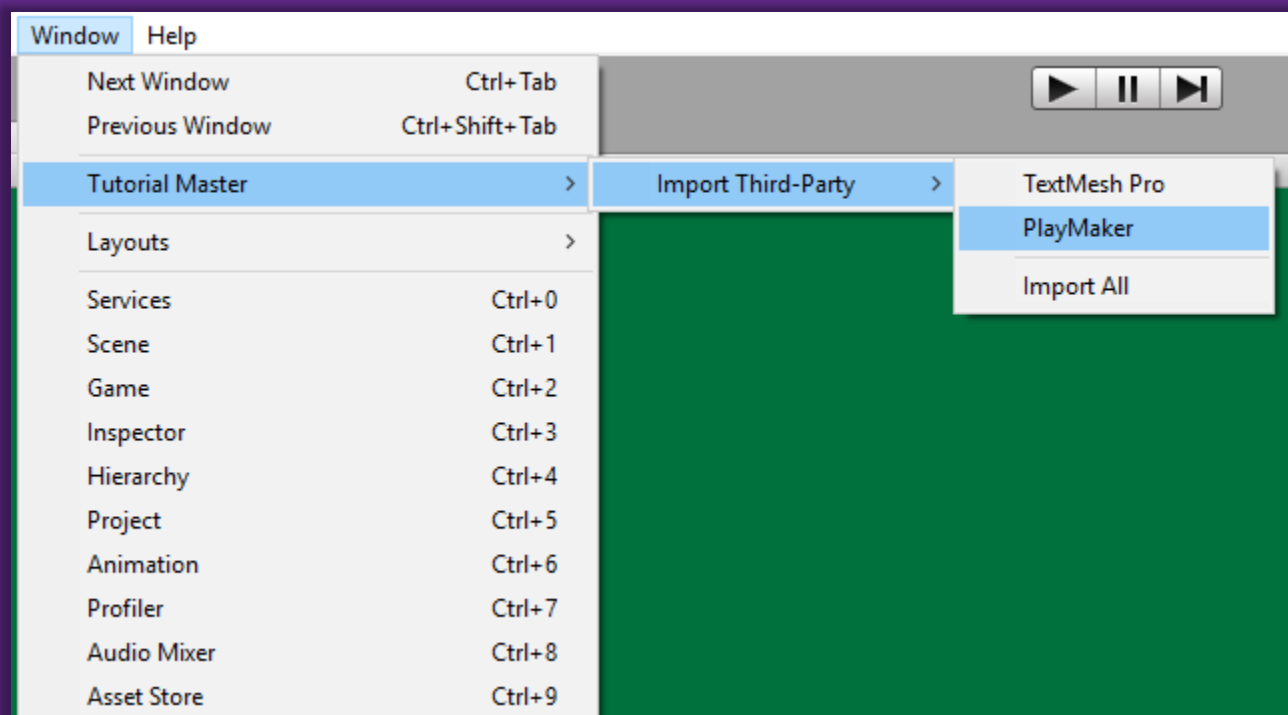
And that's that! When you have your own Tutorial with Stages ready, Tutorial Master will instantiate the maximum number of prefabs that could be used simultaneously for each module. For example, if, out of 5 stages, you have one stage where you use 5 Popup Modules at the same time and other Stages use 1 Popup Module, a total of 5 Popup Modules will be instantiated.

You can also set a custom size, but in doing so you may run into runtime errors if you set the number too low.

Importing Third-Party Plugins (Optional)

Tutorial Master comes with third-party plugin importer which makes it easier to quickly import a plugin that you may need. Be sure that you have that plugin imported first or the project won't be able to compile.

The Third-Party import menu can be accessed through *Windows > Tutorial Master > Import Third-Party*



Exported plugin scripts can be found inside of *Tutorial Master/Third-Party/Active/* directory, organized into folders.

If you're not planning on using any third-party plugins, feel free to delete the whole of "Third-Party" folder to save space. However, you would need to reimport Tutorial Master to recover deleted files to be able to access the menu Third-Party menu.

Making a Tutorial

This section covers the creation of a Tutorial and its stages. We'll also go through each stage sub-setting in brief.

Creating a tutorial

Navigate to the “Tutorials List” section and press Create Tutorial. A new tutorial will be created, and new sections will popup: “Tutorial Settings” and “Stages List”.

▼ Tutorials List

Create Tutorial

Untitled Tutorial

▼ Tutorial Settings

Stages: 0
Status: Not Playing

Name

Untitled Tutorial

▼ Events

On Tutorial Start ()
List is Empty

On Tutorial End ()
List is Empty

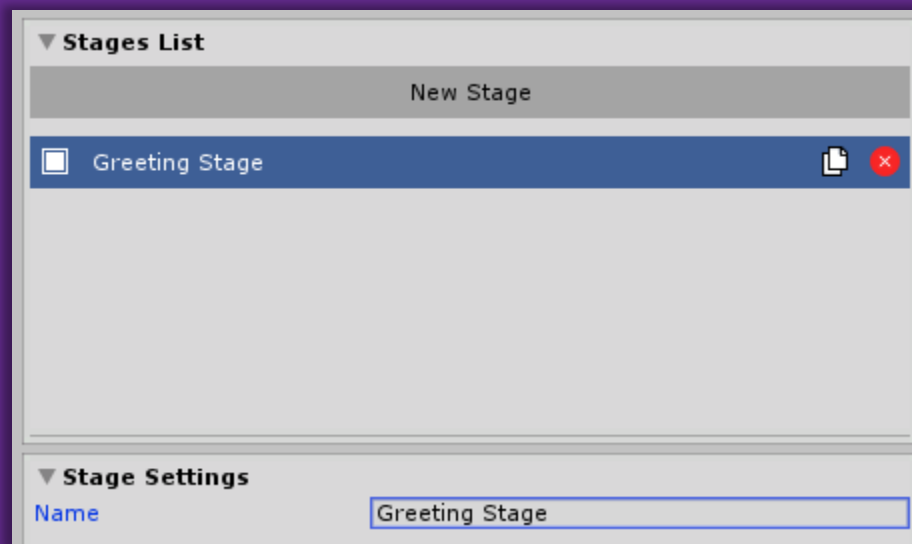
▼ Stages List

New Stage

Adding a Stage

Each tutorial consists of stages. Ideally, you would want to present a single piece of helpful advice to player per stage (e.g. show a button where he can access an inventory menu).

Click on *Add Stage* in “Stages List” (which appears after creating a tutorial) and you should see your first Stage created.



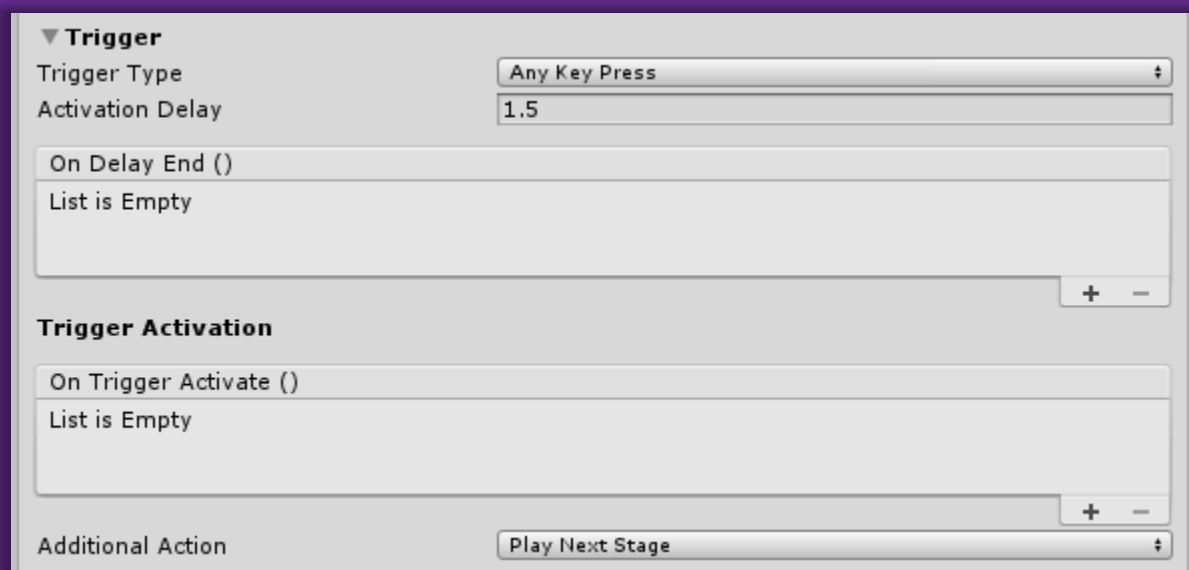
You can set the name for a Stage. This becomes useful when as you start adding more stages in the future to quickly identify their functionality and what they do.

Stage Trigger

If you enter a Stage, there must be a way to leave it. This is where triggers come in. A Stage must have a purpose: you show a piece of information to user, then you might expect feedback from him, such as clicking on a button or waiting 5 seconds.

For now, let's let player proceed to another Stage whenever any key is pressed. To do that, select "Any Key Press" in the *Trigger Type*.

You also have an option for "Activation Delay", which delay trigger activation. This is quite handy if you don't want your players spam through the whole tutorial.



The screenshot shows a configuration panel for a Stage Trigger. It has a title bar with a dropdown arrow and the text "▼ Trigger". Below the title bar, there are three main sections:

- Trigger Type:** A dropdown menu currently set to "Any Key Press".
- Activation Delay:** A text input field containing the value "1.5".
- On Delay End ():** A list box containing the text "List is Empty". To the right of the list box are "+" and "-" buttons.

Below these sections is a section titled "Trigger Activation". It contains:

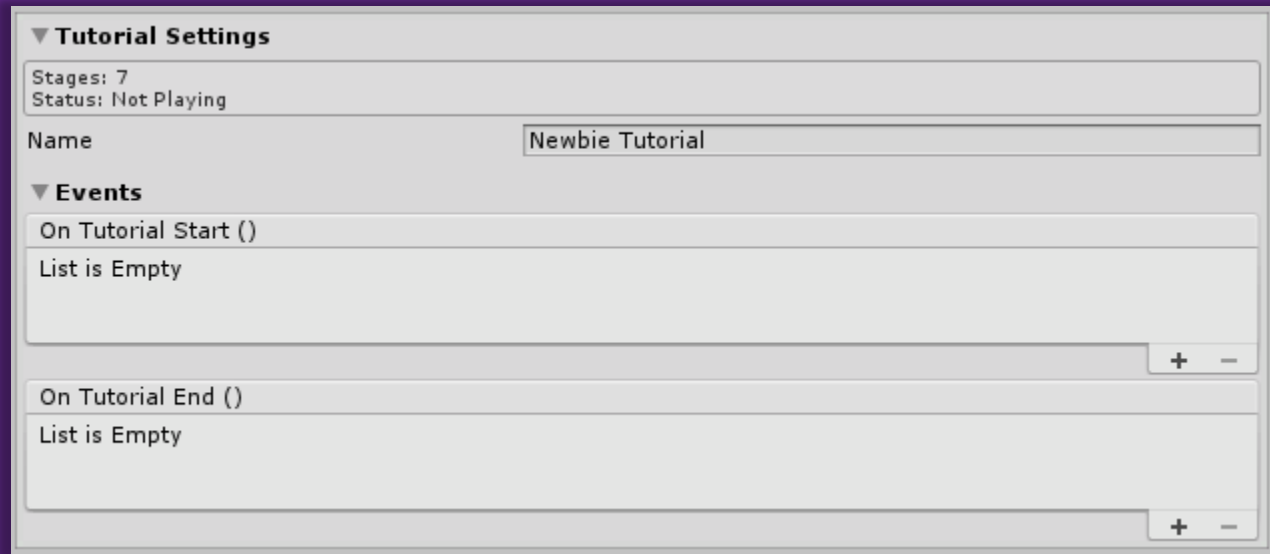
- On Trigger Activate ():** A list box containing the text "List is Empty". To the right of the list box are "+" and "-" buttons.

At the bottom of the panel is a section titled "Additional Action" with a dropdown menu set to "Play Next Stage".

You can also have additional logic to trigger activation through UnityEvents.

Stage Events

The “Events” section, contains UnityEvents that will be invoked whenever this tutorial starts and stops playing. Very useful if you want to add custom logic on top of Tutorial Master.

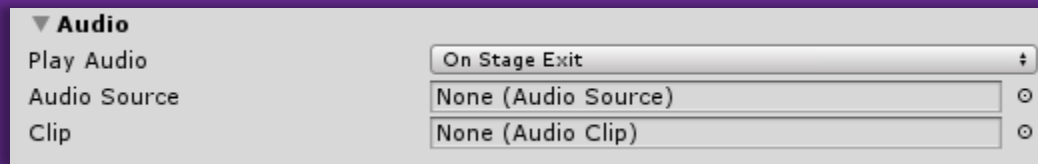


The screenshot shows the 'Tutorial Master' settings window. It has a 'Tutorial Settings' section with fields for 'Stages: 7', 'Status: Not Playing', and 'Name: Newbie Tutorial'. Below this is an 'Events' section with two event lists. The first list is for 'On Tutorial Start ()' and the second is for 'On Tutorial End ()'. Both lists are currently empty and have expand/collapse controls (+/-) on the right.

▼ Tutorial Settings	
Stages: 7	Status: Not Playing
Name	Newbie Tutorial
▼ Events	
On Tutorial Start ()	
List is Empty	
	+ -
On Tutorial End ()	
List is Empty	
	+ -

Stage Audio

This is where you can choose whether you want to play an audio clip. An audio clip is played only once.

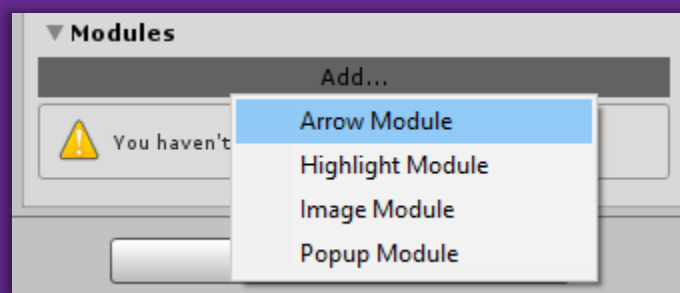


You have an option to play an audio clip when you enter a Stage or when you exit it.

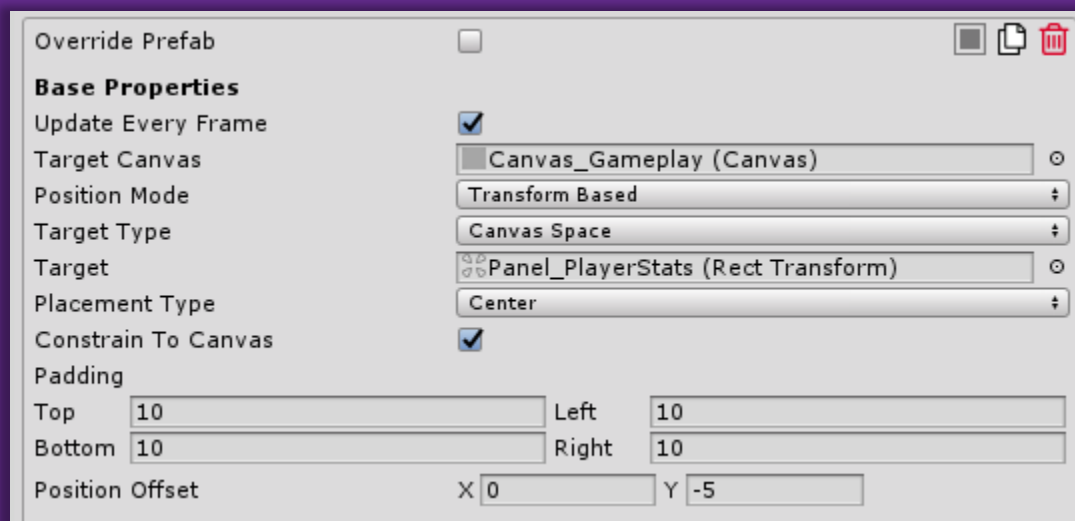
Stage Modules

This is where you can add modules. You can add as many modules imaginable. However, remember that using lots of Modules could lead to performance problems on low-end platforms (that also depends on how modules are configured).

To add a module, simply click on “Add Module” and select a Module from a dropdown. In our case, let’s add an Arrow Module



While different modules have unique functionalities and settings, all modules share base properties, which mainly determine how a module is positioned within a Canvas.



To keep things simple, we want to have an arrow pointing at a button for the first Stage:

1. Set *Target Canvas* to match the Canvas where your button is going reside in.
2. Set *Position Mode* to be “Transform Based”.
3. *Target Type* should be “Canvas Space” since we want to point at a UI element
4. Set *Target* property to a button we want to point at. This will ensure that the Arrow Module will always be positioned based on position of a button.
5. Set *Placement Type* to the “Right”. This will place Arrow Module so that it would be positioned on the right side of the target UI element, taking its size into an account.

Moving on, we can see “Module Properties” section for an Arrow Module. If you’ve set the *Placement Type* to “Right”, then *Point Direction* should be set to the “Left” so that Arrow will point at the UI element.

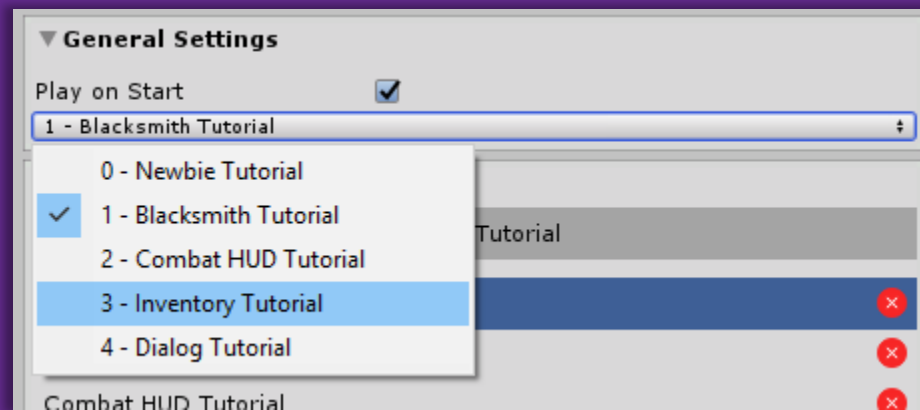
Running a Tutorial

Once we have created enough stages for our tutorial, it's time to take it for a run! There are various ways you can run a tutorial, both through editor and through API.

Using Editor

Tutorial Master allows you to start a Tutorial when the scene starts. This is quite convenient if you want to quickly run a tutorial without a need of writing code yourself.

You can set which tutorial to start when a scene starts inside of “General” settings. Make sure that *Play on Start* checkbox is checked and you should see a dropdown of all tutorials.



And this is pretty much it! Next time you run your scene with a Tutorial Master Manager component enabled, a tutorial of your choice will start.

Using API

It's not always ideal to start tutorial when the game starts and therefore an API can be used to grant us greater control over that.

TutorialMasterManager includes a set of functions and methods that make it easy to do that. Therefore, you would need a reference to a *TutorialMasterManager* to access those functionalities.

Let's try replicating the editor script. Create a separate script (let's call it *TutorialController*). Include *HardCodeLab.TutorialMaster* namespace above and add the following:

```
using UnityEngine;
using HardCodeLab.TutorialMaster;

public class TutorialController : MonoBehaviour
{
    public TutorialMasterManager tmManager;

    void Start()
    {
        tmManager = GetComponent<TutorialMasterManager>();
        tmManager.StartTutorial();
    }
}
```

Attach the *TutorialController* to the same GameObject where *TutorialMasterManager* is residing in.

Start the game and the first tutorial should start!

This is great, but what if we want to go to navigate around tutorial with arrow keys?

Add an *Update()* method to our existing *TutorialController* script and populate it with the following:

```
using UnityEngine;
using HardCodeLab.TutorialMaster;

public class TutorialController : MonoBehaviour
{
    public TutorialMasterManager tmManager;

    void Start()
    {
        tmManager = GetComponent<TutorialMasterManager>();
        tmManager.StartTutorial();
    }

    void Update()
    {
        if (Input.GetKeyDown(KeyCode.LeftArrow))
        {
            tmManager.PrevStage();
        }

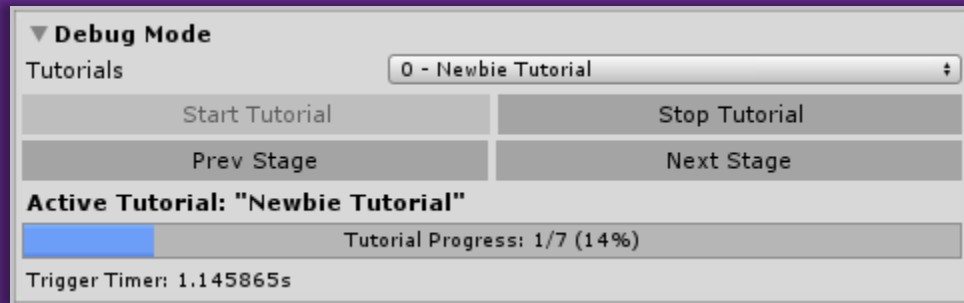
        if (Input.GetKeyDown(KeyCode.RightArrow))
        {
            tmManager.NextStage();
        }
    }
}
```

Play the game again and now we can navigate around the tutorial with arrow keys!

Note that when you're at the last Stage of the Tutorial and call *NextStage()*, the Tutorial will stop.

Using Debug Mode

Debug Mode is a set of controls that allow you to start, stop and traverse around tutorials. This section shows up at the top of the Inspector during the “Play Mode”.



Debug Mode also shows the current tutorial being played as well as the progress of that tutorial. If the Stage has timer enabled, it shows how much time is left before it's elapsed.

Useful Links

Official Website	https://hardcodelab.com
Support Email	support@hardcodelab.com
Support Site	https://support.hardcodelab.com
Twitter	https://twitter.com/@HardCodeLab
YouTube	https://youtube.com/c/HardCodeLab