Created by Sage Miller
sage00miller@gmail.com
sagemiller.net

Custom Timer Documentation

Hello and welcome to this PDF! In this document you will learn how to use the Custom Timer asset to its full potential. A walkthrough video of this asset can be found at: https://youtu.be/qRzU5VDnAU0. I will start a thread on the Unity Forums after this asset gets published and link it in the Youtube video's description.

This document will cover the following, in this order:

1. Getting Started

2. Dissecting the Timer - How the CustomTimer gameobject is set up in the hierarchy and a detailed description of every field in the CustomTimer script.

3. Timer End Event - How to call any public function when the timer is complete.

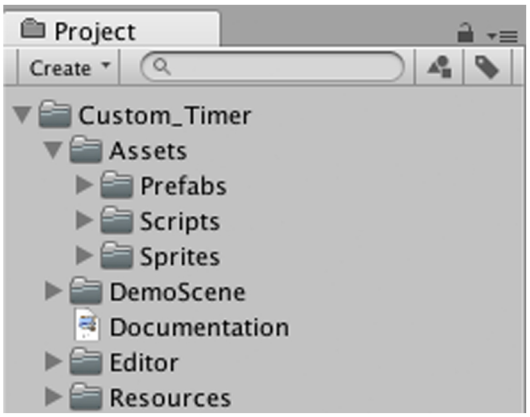
4. Public Functions - A list of all the public functions in the CustomTimer script and what they do.

5. Advanced Customization Options - Changing image sprites, text font/style, and adding extra gameobjects to the timer's hierarchy.

6. Future Features

Getting Started

After importing the package you should see a new folder in your project view titled “Custom_Timer”.



Project view of Custom_Timer Folder

The package contains:
•20 prefabs of different timer styles which are ready to use or customize
•Three C# scripts (one is in the Editor folder)
•Four sprite images
•One demo scene
•One documentation pdf

You can open the Demo Scene found in the DemoScene folder to view 20 pre-made timers.

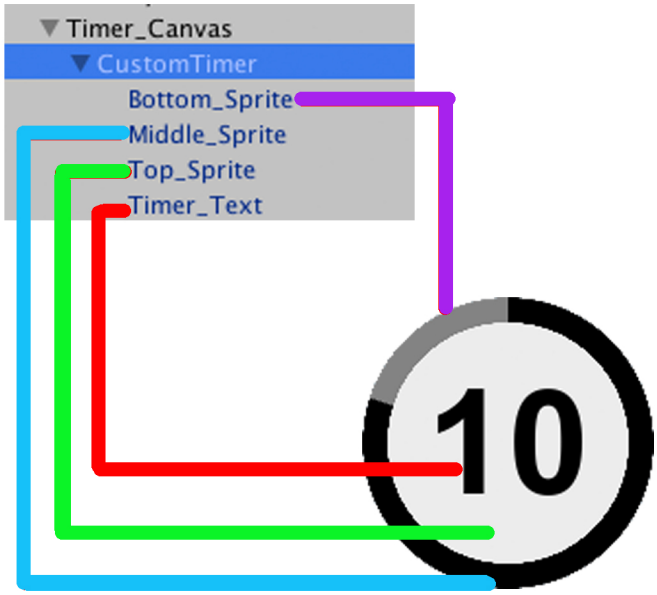
These 20 timers are all in the same canvas. The CustomTimer must be childed to a **UI Canvas** to be seen. You can place multiple timers in several different canvases.

View of the Demo Scene

Dissecting the Timer

The Custom Timer is made up of three sprites and one text object childed under a CustomTimer gameobject.

The image to the right shows a timer comprised of black text and three circle sprites. Top_Sprite is white, Middle_Sprite is black, and Bottom_Sprite is gray. The middle and bottom are the same size while the top circle is smaller.



Objects of the Custom Timer and what they represent in the scene.

The CustomTimer parent object contains the script that controls nearly everything about how our timer looks and works. Lets talk about what each field does!

Duration: How long should the timer run for? Must be entered in seconds.

Count Up or Down: Should the text start from 0 and count up to the duration, or start at the duration and count down to 0. You can ignore this if you do not want to show any text.

Fill Up or Down: Using the above Timer image as an example, the black bar can either start empty and fill up over time or start full and empty out over time.

CustomTimer script found on the CustomTimer parent object

Dissecting the Timer (cont) + Sprite Settings

Rotation: Rotate the sprites to change where they start filling from. By default, at 0 rotation, they start filling from the top, 90 starts from the left, 180 from the bottom, 270 from the right, and anything inbetween.

Flip Fill Direction: By default images fill up clockwise and fill down counterclockwise. Toggle this to reverse this.

>Sprite Settings: The dropdowns for each sprite contain the same fields so we will cover all three Sprite Settings at once.

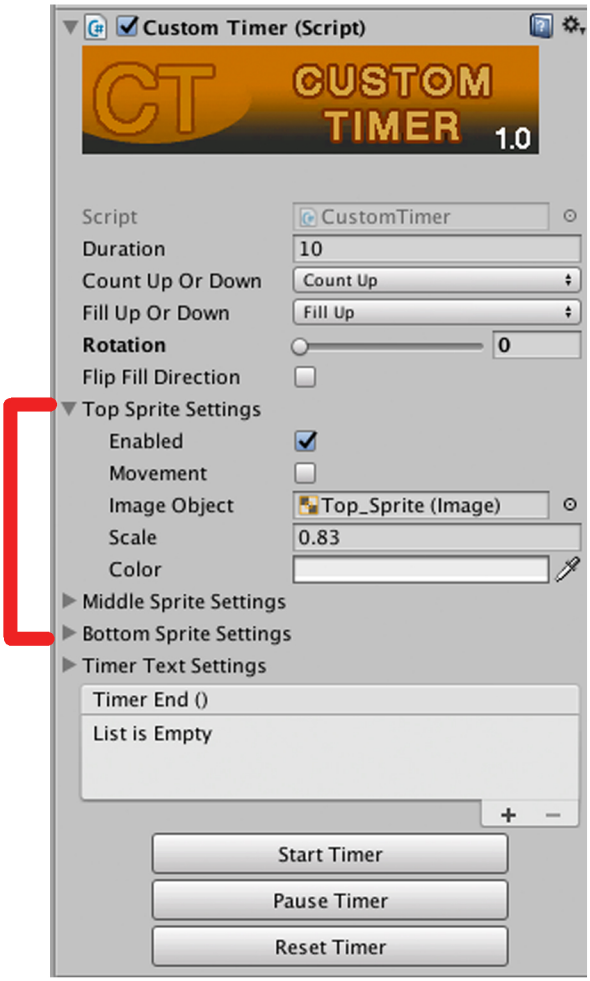
Enabled: Show or hide the sprite.

Movement: Should this sprite fill up/down over time?

Image Object: Drag in the proper sprite gameobject (from the **scene**, childed under the gameobject containing this script).

Scale: The size of the sprite.

Color: The color of the sprite. Can also change opacity here.



View of the Top Sprite Settings. The same options are available for all 3 sprites.

Text Settings & End Event

>Timer Text Settings: This dropdown is similar to the sprite ones, but controls the text.

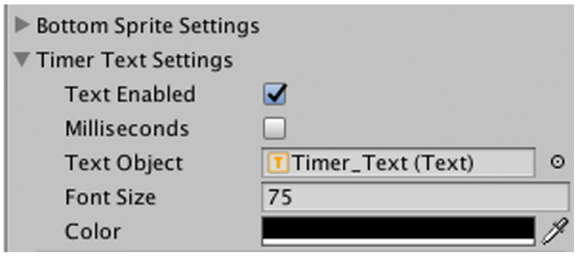
Text Enabled: Show/Hide the text.

Milliseconds: Show/hide milliseconds.

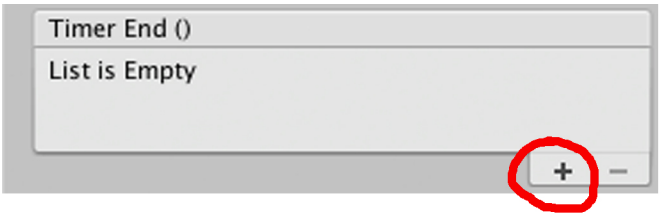
Text Object: Drag in the proper text gameobject (from the **scene**, childed under the gameobject containing this script).

Font Size: Size of the text.

Color: Color of the text.



View of the Timer Text Settings.



Click the plus (+) to add an Event.

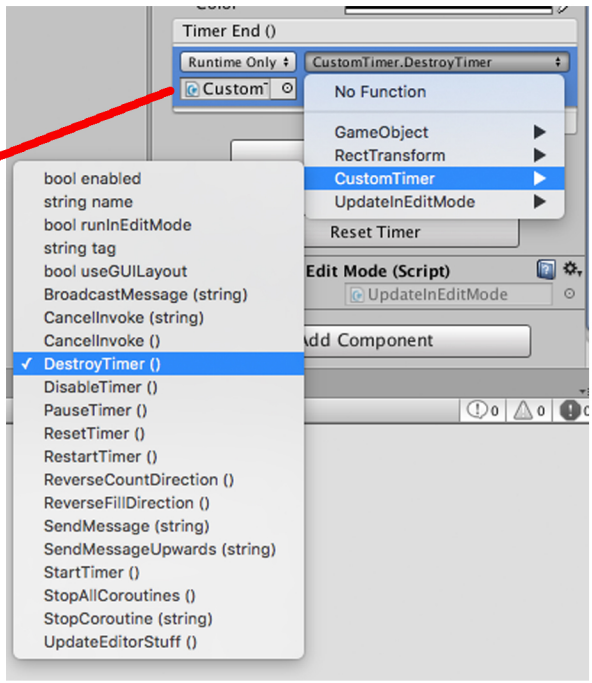
Timer End Event

When the timer has finished the TimerEnd() Event is called (you can leave it empty). You can choose to call one of the 6 public functions contained in the CustomTimer script, or call a function from one of your own scripts!

In the left field, drag in the object which contains the script you want to call a function from. In the example to the right I dragged in the Timer object this script is attached to.

Click on the dropdown to the right to pick a function. Choose the component you want, in our case CustomTimer, then choose the function.

You can call multiple functions in one event. They will be called in the order they are listed.

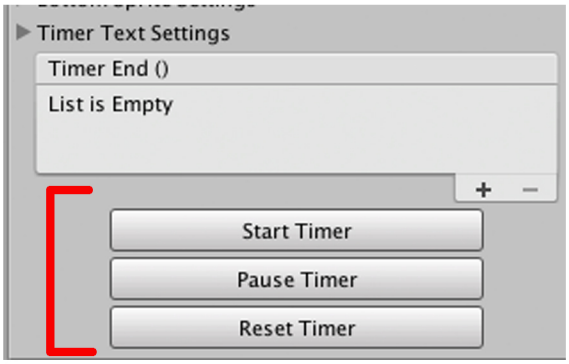


Choosing to call the DestroyTimer() function when the timer ends

Buttons & Public Functions

Three buttons can be found at the bottom of the CustomTimer Script.

Start Timer: Calls the StartTimer() function.
Pause Timer: Calls the PauseTimer() function.
Reset Timer: Calls the ResetTimer() function.



View of the convenient buttons

Public Functions

StartTimer(): Start the timer. Resumes if paused. Also resets progress to 0 if the timer is completed.

PauseTimer(): Pause/stop the timer.

ResetTimer(): Set the progress to 0 and **pause** the timer.

RestartTimer(): Set the progress to 0 and **start** the timer.

DisableTimer(): Disable the gameobject.

DestroyTimer(): Destroy the gameobject.

ReverseFillDirection(): Changes the “Fill Up Or Down” field to the unselected option.

ReverseCountDirection(): Changes the “Count Up Or Down” field to the unselected option.

GetTimerValue(): Returns a **float** between 0 and 1. The value is 0 when timer hasn’t started, 0.25 when its a quarter complete, 0.50 when its halfway done, 1 when it is complete, and everything inbetween. The value is the same regardless of filling or counting up/down.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class TestScript : MonoBehaviour {
6
7      public CustomTimer ct;
8
9      // Update is called once per frame
10     void Update ()
11     {
12         if (Input.GetKeyDown(KeyCode.S))
13         {
14             ct.StartTimer();
15         }
16
17         if (Input.GetKeyDown(KeyCode.P))
18         {
19             ct.PauseTimer();
20         }
21
22         if (Input.GetKeyDown(KeyCode.R))
23         {
24             ct.ResetTimer();
25         }
26
27         if (Input.GetKeyDown(KeyCode.V))
28         {
29             print(ct.GetTimerValue());
30         }
31     }
32 }
```

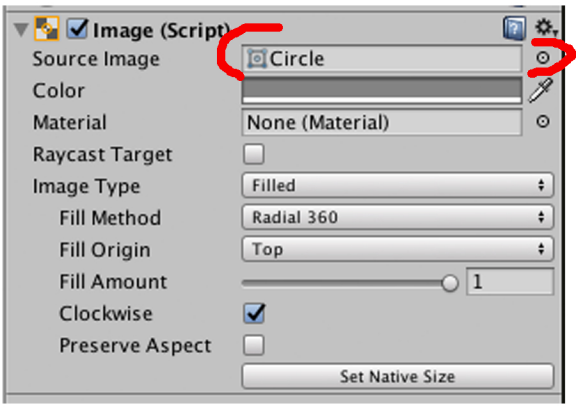
Example of a script referencing a CustomTimer instance, then calling different functions when different keys are pressed.

Changing Image Sprites & Text Font

There are a few customization options that are not available through the main script, but can be manually changed. These include **changing an image's sprite and changing the font text/style/position**. To do this you must select the child object in the hierarchy that you want to change.

Most of the fields in these objects are handled by the main CustomTimer script and must be edited there. I will add these exceptions to the main script if it is frequently requested.

Image



With the Bottom_Sprite object selected, we can change the Source Image to be any sprite we want. The other fields are handled by the main script.

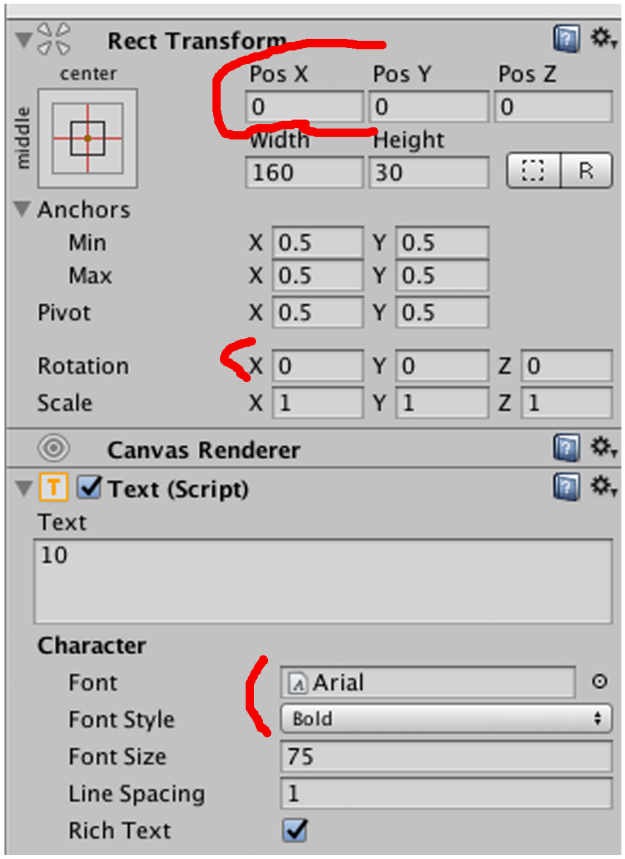
Misc. Folders/Scripts

The Resources folder contains the orange image thats on top of the CustomTimer script. If this file is deleted there will be empty space at the top of the script instead of the image.

The Editor folder contains a script that gives our inspector a custom look and enables the 3 buttons.

The UpdateInEditMode script allows changes to your timer to happen outside of Play mode. This script should be on all timers, otherwise changes in your timer (like color) won't be reflected in the scene view until you hit Play.

Text



With the Timer_Text object selected, we can change the font, fontstyle, position, and rotation. The other fields are handled by the main script.

Adding Extra Gameobjects in the Hierarchy

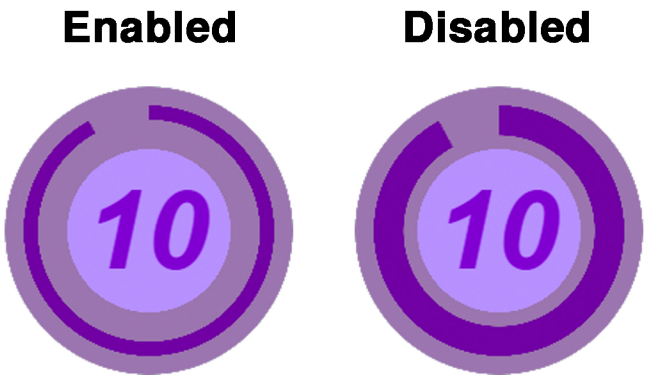
It is possible to add extra gameobjects childed to the CustomTimer. You will **not** be able to control these through the main script.

Hierachy of the CustomTimer_EXTRA prefab.

In the prefab titled “CustomTimer_EXTRA” we can see an example of adding an extra circle sprite to a timer.

We place a circle under the top sprite and above the middle sprite, then give it the same color as the bottom sprite. This **gives the illusion of a thinner ring** sprite, because our extra sprite is covering up some of the middle sprite.

Get creative! You can do something like this with any sprite of your creation.



Comparison of CustomTimer_EXTRA with and without the extra sprite. The extra sprite is a circle the same gray-purple color as the bottom sprite, layered under the top sprite.

Thank You!

Thank you so much for using the Custom Timer asset! I hope it serves you well. If you have any questions or issues please email me at: sage00miller@gmail.com. Also, if you haven't already, be sure to check out the Demo Video at: https://youtu.be/qRzU5VDnAU0.

I want to continue working on this, adding features and overall improving it. **Please let me know any suggestions you have!** Anything from renaming a variable to a brand new feature.

Some features I am considering...

•Adding the manual changes (changing sprite or font, seen above) to the main script.

•Adding toggles to show hours and minutes, maybe ability to enter duration in hours, minutes.

•Adding a color-over-time option for each sprite and the text. Choose your own colors.

•Creating more ready-to-use sprites (like the blue one to the right).

Let me know if you're interested in any of those and I'll make it priority!