Easy Clock Timer Documentation

You are free to modify and adapt this product to your own needs.

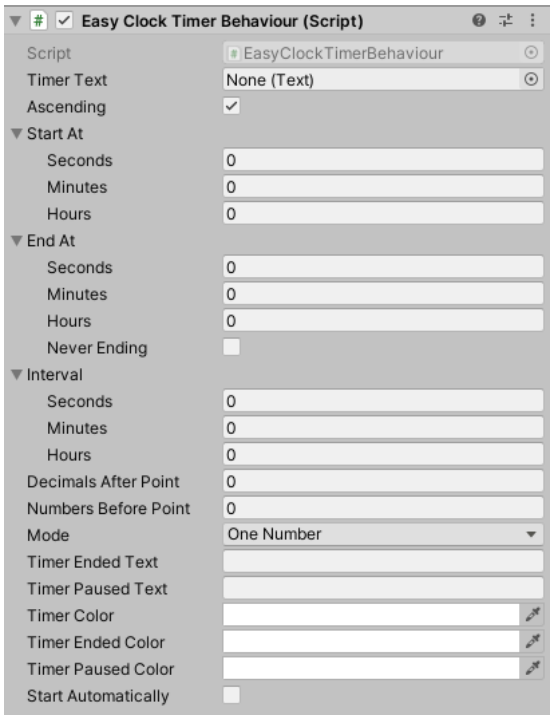
I would recommend opening the EasyClockTimerExample scene to learn the different functionalities.

Thank you for purchasing the Easy Clock Timer component. You can always refer to this document or open the example scene if you need help using the component.

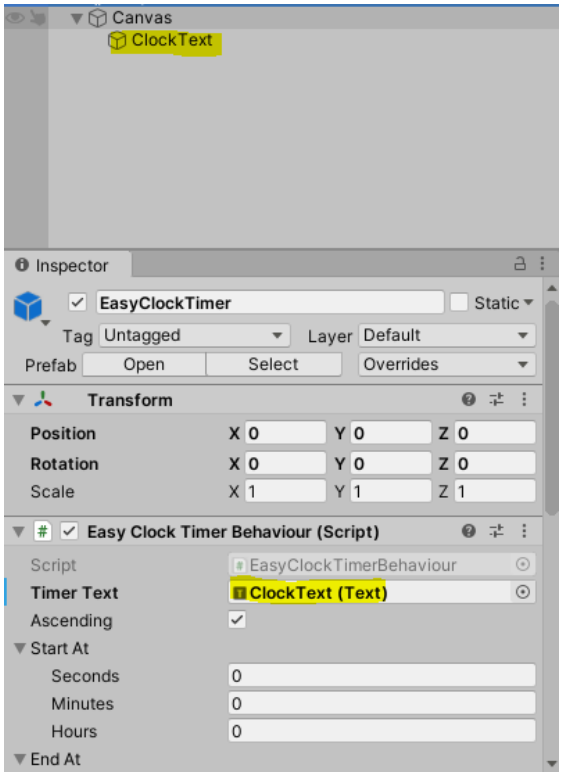
To Use the Easy Clock Timer Component

Create the Timer Text UI object:

- Add the EasyClockTimer prefab to your scene.
- You should have the Easy Clock Timer Behaviour attached to the prefab like following:



- First you need to create and attach a Unity Text UI Object to this script.



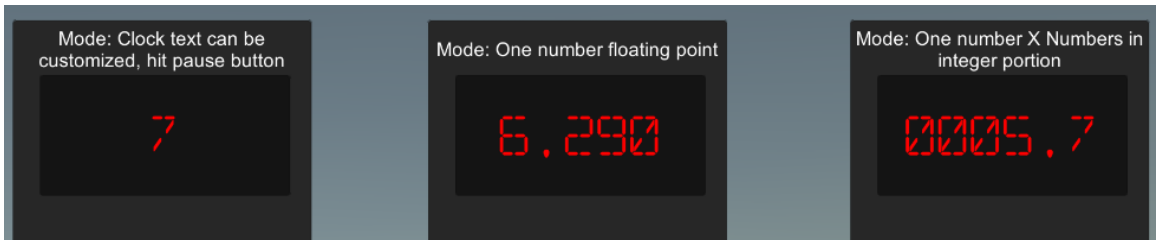
- That will be the clock text displayed. *(Note that this text UI can be added to a 2D or 3D canvas)*

Configure the EasyClockTimerBehaviour to your needs:

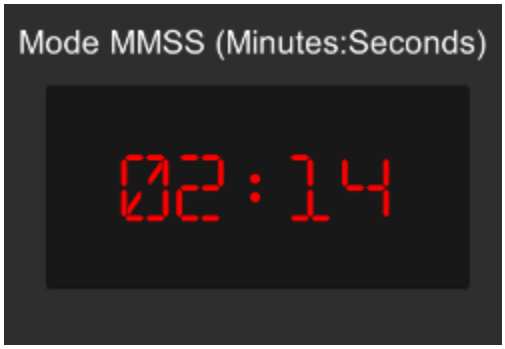
The EasyClockTimerExample scene show different way to use this component so you can open this scene to see what king of timer or clock you want to have.

Mode:

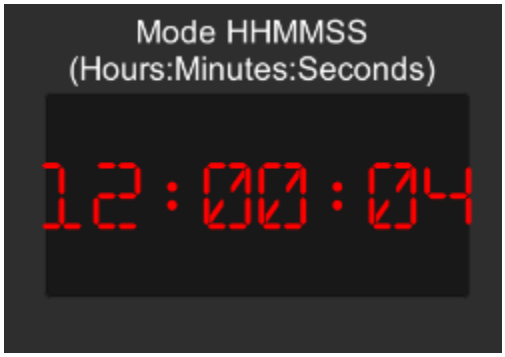
- One Number: This mode will only display the number in seconds. You can also add decimal after the point or before the point.



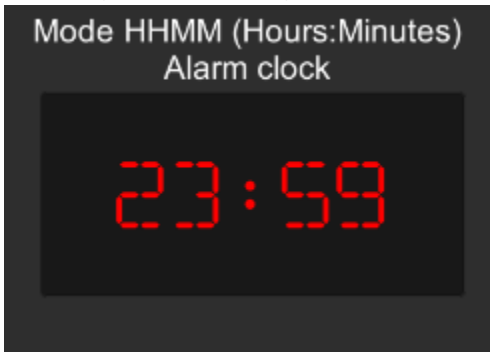
- MMSS (Minutes : Seconds)



- HHMMSS (Hours:Minutes:Seconds)



- HHMM (Hours:Minutes) Just like an alarm clock.



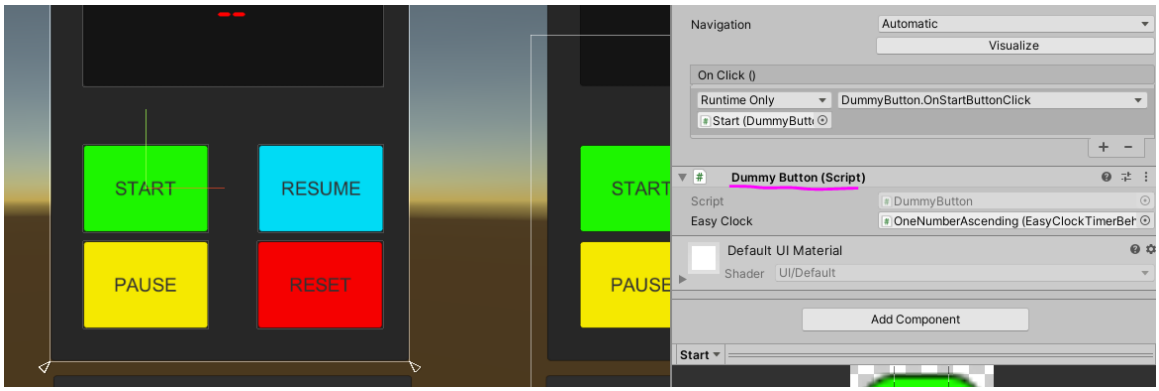
Here I will give you the information on the properties of the EasyClockTimerBehaviour script:

Property	Description
Timer Text	Contains the Text component that we will use to display the clock.
Ascending	Set whether you want the time to go up or down.
Start At	Set the starting time in hours, minutes, and seconds.
End At	Set the ending time in hours, minutes, and seconds.
Never Ending	If you check this property, the End At property will be ignored and the time will never stop.
Interval	Set an interval, if you want to trigger function each X time you can set the interval of time here. I will show how to add trigger function later.
Decimals After Point	Define the number of decimals after the point when using the One Number Mode.
Number Before Point	Same as Decimals After Point but before the point.
Mode	Set the mode (One number, MMSS, HHMMSS, HHMM)
Timer Ended Text (optional)	Define the text to show when the timer ended. If empty the time will be displayed.
Timer Pause Text (optional)	Same as Timer Ended but for pause.
Timer Color	Define the color of the timer text.
Timer Ended Color	Define the color of the timer text when the time end.
Timer Pause Color	Same as Timer ended color but for pause
Start Automatically	Define if you want to start the timer when the scene is loaded or if you want to start it manually through script.

Start, pause, resume, and reset the Timer:

If you do not have the Start Automatically property set, you want to be able to start it from script and here is how.

If you want an easy example on how to do it look at the start, pause, resume and reset buttons in the EasyClockTimerExamples scene.



Lets take the start button for example. I've created a very basic script named Dummy Button that contains a reference to the EasyClockBehaviour script.

```
// This is only an example class on how to use each Timer/Clock Actions. (Start, Pause, Reset)
Script Unity | 0 références
public class DummyButton : MonoBehaviour
{
    public EasyClockTimerBehaviour EasyClock;
    0 références
    public void OnStartButtonClick()
    {
        EasyClock.StartTimer();
    }
    0 références
    public void OnResumedButtonClick()
    {
        EasyClock.ResumeTimer();
    }
    0 références
    public void OnPauseButtonClick()
    {
        EasyClock.PauseTimer();
    }
    0 références
    public void OnResetButtonClick()
    {
        EasyClock.ResetTimer();
    }
}
```

I've added this script to my start button and attach the Button OnClick event to the OnStartButtonClick function.

In the function OnStartButtonClick, I use the EasyClock reference to call the StartTimer function to start the timer.

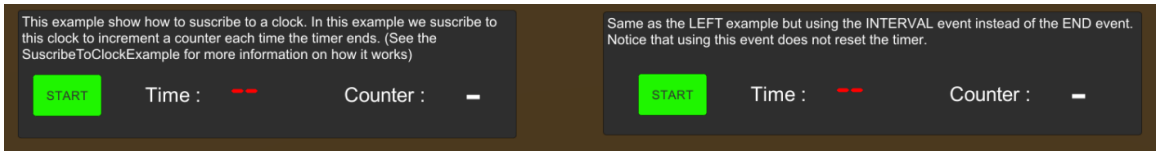
I do the same thing for the pause, resume, and reset button.

So, if you have a reference to the EasyClockTimerBehaviour script, you can start, pause, resume or reset the timer from anywhere you want.

Easy Clock Timer Events:

You can also attach function that you want to execute when the timer starts, pause, resume, end or when you have an interval set up.

The best way to understand how this works is to look at the EasyClockTimerExamples scene:



Those two examples show how to attach a function that increment a counter to a timer event.

For the first example it increments the counter when the timer ends and in the same function it resets the timer to 0.

For the second example it increments the counter when the timer hit the interval set but it does not reset the timer to 0.

See the two SuscribeToClockExample scripts.

```
Script Unity | 0 références
public class SuscribeToClockExample : MonoBehaviour
{
    public Text CounterText;
    // This is the counter that we will increment on each timer end.
    private int _counter = 0;
    // You need a reference to the EasyClockTimerBehaviour to attach a function.
    // As we do here, you can also use the EasyClockReference to play around with the easy clock functions in the action your passing
    public EasyClockTimerBehaviour EasyClockReference;
    Message Unity | 0 références
    private void Start()
    {
        CounterText.text = _counter.ToString();
        EasyClockReference.AttachActionToTimerEvent(TimerAction.END, FunctionThatIWantToExecuteWhenTimerEnd);
    }

    // Note that an action is passed to the timer so it cannot contains arguments.
    // Since we are suscribing to this easy clock ending event, this action will get called each time the time ends.
    1 référence
    void FunctionThatIWantToExecuteWhenTimerEnd()
    {
        // Use the EasyClockReference to reset the timer when it ends.
        EasyClockReference.ResetTimer();
        _counter++;
        CounterText.text = _counter.ToString();
    }


}
```

To attach a function, use the AttachActionToTimerEvent function. This function takes two parameters. The type of event to want to attach to (START, PAUSE, RESUME, RESET, END, INTERVAL) and the function that you want to be executed when the event happens. Here I register the FunctionThatIWantToExecuteWhenTimerEnd() to the END event.

If you want to detach one or multiple functions from an event you can use one of these functions.

```
public void DettachActionFromTimerEvent(TimerAction timerAction, UnityAction
action)

public void DettachAllActionsFromTimerEvent(TimerAction timerAction)

public void DettachAllActions()
```

I really hope that will help you and wish you the best of luck in your projects !!!