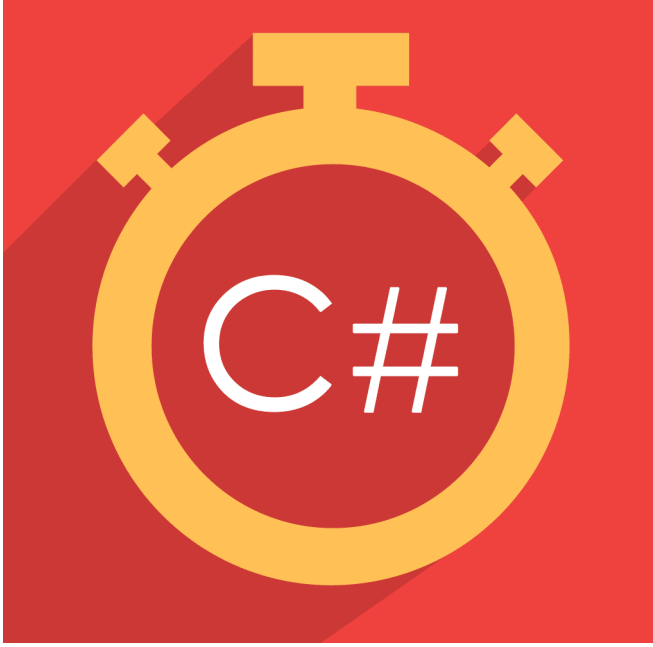
Classes

Interfaces



Timer.cs

```
Get interval
public float Interval()

Get total loops count (INFINITE (which is uint.MaxValue) if is constantly looping)
public uint LoopsCount()

Get how many loops were completed
public uint CurrentLoopsCount()

Get how many loops remained to completion
public uint RemainingLoopsCount()

Get the delegate to execute
public UnityAction Delegate()

Get total remaining time
public float RemainingTime()

Get total elapsed time
public float ElapsedTime()

Get elapsed time in current loop
public float CurrentCycleElapsedTime()

Get remaining time in current loop
public float CurrentCycleRemainingTime()

Checks whether this timer is ok to be removed
public bool ShouldClear()

Checks if the timer is paused
public bool IsPaused()
```

```
Pause / Inpause timer
public void SetPaused(bool bPause)

Get total duration, (INFINITE if it's constantly looping)
public float Duration()


Compare frequency (calls per second)
public static bool operator >(Timer A, Timer B)

Compare frequency (calls per second)
public static bool operator <(Timer A, Timer B)

Compare frequency (calls per second)
public static bool operator >=(Timer A, Timer B)

Compare frequency (calls per second)
public static bool operator <=(Timer A, Timer B)
```

TimersManager.cs

```
Set timer
Owner - The object that contains the timer. Required in order to remove the timer if the
object is destroyed
Timer - Timer to add
public static void SetTimer(object Owner, Timer timer)
```

```
Set a timer that loops LoopCount times
Owner - The object that contains the timer. Required in order to remove the timer if the
object is destroyed.
interval - Interval(in seconds) between loops
LoopsCount - How many times to loop
unityAction - Delegate
public static void SetTimer(object Owner, float interval, uint LoopsCount, UnityAction
unityAction)
```

```
Set a timer that activates only once.
Owner - The object that contains the timer. Required in order to remove the timer if the
object is destroyed.
interval - Interval(in seconds) between loops
unityAction - Delegate
public static void SetTimer(object Owner, float interval, UnityAction unityAction)
```

```
Set an infinitely loopable timer
Owner - The object that contains the timer. Required in order to remove the timer if the
object is destroyed.
interval - Interval(in seconds)
unityAction - Delegate
public static void SetLoopableTimer(object Owner, float interval, UnityAction
unityAction)
```

```
Add a list of timers. Works great with List<Timer> in inspector. See 'TimersList.cs' for
an example.
Owner - Owner of timers. This should be the object that have these timers. Required in
order to remove the timers if the object is destroyed.
Timers - Timers list
public static void AddTimers(object Owner, List<Timer> Timers)
```

```
Remove a certain timer
unityAction - Delegate name
public static void ClearTimer(UnityAction unityAction)
```

```
Get timer by name (which is the delegate's name)
unityAction - Delegate name
public static Timer GetTimerByName(UnityAction unityAction)
```

```
Get timer interval. Returns 0 if not found.
unityAction - Delegate name
public static float Interval(UnityAction unityAction)
```

```
Get total loops count (INFINITE (which is uint.MaxValue) if is constantly looping)
unityAction - Delegate name
public static uint LoopsCount(UnityAction unityAction)
```

```
Get how many loops were completed
unityAction - Delegate name
public static uint CurrentLoopsCount(UnityAction unityAction)
```

```
Get how many loops remained to completion
unityAction - Delegate name
public static uint RemainingLoopsCount(UnityAction unityAction)
```

```
Get total remaining time
unityAction - Delegate name
public static float RemainingTime(UnityAction unityAction)
```

```
Get total elapsed time
unityAction - Delegate name
public static float ElapsedTime(UnityAction unityAction)
```

```
Get elapsed time in current loop
unityAction - Delegate name
public static float CurrentCycleElapsedTime(UnityAction unityAction)
```

```
Get remaining time in current loop
unityAction - Delegate name
public static float CurrentCycleRemainingTime(UnityAction unityAction)
```

```
Verifies whether the timer exits
unityAction - Delegate name
public static bool IsTimerActive(UnityAction unityAction)
```

```
Checks if the timer is paused
unityAction - Delegate name
public static bool IsTimerPaused(UnityAction unityAction)
```

```
Pause / Unpause timer
unityAction - Delegate name
bPause - true - pause, false - unpause
public static void SetPaused(UnityAction unityAction, bool bPause)
```

```
Get total duration, (INFINITE if it's constantly looping)
unityAction - Delegate name
public static float Duration(UnityAction unityAction)
```