

# LigthDev Unity Framework

5 parts:

- 1) Core
- 2) Events
- 3) Pool
- 4) Scriptable
- 5) UI

## 1. **Core** (LightDev.Core namespace)

- 1) **Base**. Base is a class in LightDev framework that contains some useful methods and inherited from MonoBehaviour.

### 2) **IAutoLoadable** and **AutoLoader**.

Class that implements IAutoLoadable has to have ***static constructor***. This constructor will be called by AutoLoader using ***reflection***. It will be ***called before first scene loaded*** in the game.

Note: static constructor will be called only once.

Usage example: you have player in the game. In order to not to call each time GameObject.FindWithTag (or something like that) in every script, create PlayerManager that implements IAutoLoadable. Subscribe in static constructor on event that will be raised when Player created, and get access to Player. Now you can access to Player using this PlayerManager.

## 2) Events (LigthDev namespace)

You can easily add event like this. You do not need to create Event object using operator NEW. It will be created automatically in static constructor using reflection. To raise event, use Call() method.

```
namespace LigthDev
{
    public partial class Events
    {
        public static Event ApplicationPaused;
        public static Event ApplicationResumed;
        public static Event SceneLoaded;
    }
}
```

### 3) Pool (LightDev.Pool namespace)

**PoolsManager** can manipulate prefabs that put in Resources folder and has component script that inherited from **PoolableElement**.

**What do you need?** Create *C# script* and *inherit* it *from PoolableElement*. Create GameObject, attach this script to it. *Put GameObject in Resources folder*.

#### ***PoolsManager:***

- 1) PoolsManager.RetrieveElement<T>() returns element of type T from pool.
- 2) PoolsManager.ReturnElement(PoolableElement) returns object to pool.
- 3) PoolsManager.ReturnElements<T>() returns all active elements of type T to pool.
- 4) PoolsManager.DestroyElements<T>() destroy all elements of type T.

#### ***PoolableElement:***

When PoolableElement is taken from pool Subscribe() and OnRetrieved() methods are called.

When PoolableElement is returned to pool Unsubscribe() and OnReturned() methods are called.

#### ***Remember:***

- 1) Awake, Start methods are called only once, as object is instantiated only once.
- 2) Do not forget to annul reference to PoolableElement, if you have returned it to pool. As this object is in pool, and if you decided to manipulate it, it can be used by something else in the game.

#### 4) ScriptableObjects (LightDev.Scriptable namespace)

If you have ***only one instance*** of scriptable object in game. For example, it could be settings.

Create ***C# script*** that ***inherited from AutoLoadableScriptable***. Create asset and put it to ***Resources folder***.

Access this scriptable object using  
ScriptableManager.GetScriptableObject<T>().

## 5) UI (LightDev.UI namespace)

To create flexible UI use **CanvasManager** and **CanvasElement**.

**CanvasManager**: manipulates CanvasElements. Attach CanvasManager component to canvas. It will find all children that has CanvasElement component. Then in Awake method, it will activate CanvasElement gameObjects, call Subscribe methods, and the deactivate gameObjects.

### **CanvasElement:**

- 1) In Subscribe() method, subscribe on events and decide when to show or hide UI using Show()/InstantShow(), Hide()/InstantHide(). Do not forget to unsubscribe in Unsubscribe().
- 2) To activate gameObject, use Show() and InstantShow().
- 3) To hide gameObject, use Hide() and InstantHide().
- 4) Show(): gameObject is activated, then called OnStartShowing(), after *showTime* delay OnFinishShowing() called.
- 5) InstantShow(): gameObject is activated, then called OnStartShowing() and OnFinishShowing().
- 6) Hide(): OnStartHiding() called, after *hideTime* delay OnFinishHiding() called, and then gameObject is deactivated.
- 7) InstantHide(): OnStartHiding() called, then OnFinishHiding() and then gameObject is deactivated.