USAGE INSTRUCTIONS

ScrollrectItemsAdapter8, **BaseParams** and **BaseItemViewsHolder** are the 3 core classes in our small library dedicated to both optimize a Scroll View and programmatically manage its contents.

You can use it both for a horizontal and vertical ScrollView.

ScrollrectItemsAdapter8 it's an abstract class, because you need to provide the height or width (depending if it's a vertical or horizontal **ScrollRect**) of each item (via your implementation of **ScrollrectItemsAdapter8.GetItem[Height/Width](int index)** callback) and to populate the views with data for each item (via your implementation of **ScrollrectItemsAdapter8.InitOrUpdateItemViewHolder(BaseItemViewsHolder viewsHolder)** callback).

It's recommended to manually go through example code provided in **ScrollRectItemsAdapterExample.cs** and **SimpleTutorial.cs** in order to fully understand the mechanism. You'll find detailed comments in core areas. You may even use this script directly without implementing your own, in some simple scenarios.

(Some may find it more easy to consult the example code+scene directly ,without reading this tutorial)

IMPLEMENTATION

*(Follow these steps while constantly looking at how it's done in the example code in **SimpleTutorial.cs*** and optionally in ***ScrollRectItemsAdapterExample.cs**)*

Here's the normal flow you'll follow after you've created a Scroll View using **GameObject->UI->Scroll View:**

1. create your own implementation of **BaseItemViewsHolder**, let's name it **MyItemViewsHolder**

2. create your own implementation of **BaseParams** (if needed), let's name it **MyParams**

3. create your own implementation of **ScrollRectItemsAdapter8<MyParams, MyItemViewsHolder>**, let's name it **MyScrollRectItemsAdapter**

4. instantiate **MyScrollRectItemsAdapter**

5. call **MyScrollRectItemsAdapter.ChangeItemCountTo(int count)** once (and any time your dataset is changed) and two things will happen:

5.1. **if** the **ScrollRect** has vertical scrolling, **MyScrollRectItemsAdapter.GetItemHeight(int index)** will be called <count> times (with index going from 0 to <count-1>)
else if the **ScrollRect** has horizontal scrolling, **MyScrollRectItemsAdapter.GetItemWidth(int index)** will ... [idem above] ...

5.2 **CreateViewsHolder(int)** will be called for each view that needs created. Once a view is created, it'll be re-used when it goes off-viewport

- **newOrRecycledViewsHolder.root** will be null, so you need to instantiate your prefab (or whatever), assign it and call **newOrRecycledViewsHolder.CollectViews().** Alternatively, you can call its

`AbstractViewHolder.Init(..)` method, which can do a lot of things for you, mainly instantiate the prefab and (if you want) call **`CollectViews()`**

- after creation, only **`MyScrollRectItemsAdapter.UpdateItemViewHolder()`** will be called for it when its represented item changes and becomes visible

- this method is also called when the viewport's size grows, thus needing more items to be visible at once

5.3. **`MyScrollRectItemsAdapter.UpdateViewsHolder(MyItemViewsHolder)`** will be called when an item is to be displayed or simply needs updating:

- use **`newOrRecycledViewsHolder.itemIndex`** to get the item index, so you can retrieve its associated model from your data set (most common practice is to store the data list in your **`Params`** implementation)

- **`newOrRecycledViewsHolder.root`** is not null here (given the view holder was properly created in **`CreateViewsHolder(..)`**). It's assigned a valid object whose UI elements only need their values changed (common practice is to implement helper methods in the view holder that take the model and update the views themselves)

5. call **`MyScrollRectItemsAdapter.Dispose()`** when you're done using it (usually, in the **`ScrollRect's OnDestroy()`**)

EXAMPLE SCENES & UTILITIES

All the example scenes & the utlity scripts are provided on an "as-is" base. Their main purpose is to demonstrate the feature-set and show you the basic code-flow when implementing the adapter, following the recommended best-practices & conventions.