## 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.

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 directly without reading this tutorial. But it helps to read it)***

## 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:

1. **if** the `ScrollRect` has vertical scrolling (only top-to-bottom is currently supported. It shouldn't be hard to mimic a bottom-to-top, anyways), `MyScrollRectItemsAdapter.GetItemHeight(int index)` will be called <count> times (with index going from 0 to <count-1>)
**else** if the `ScrollRect` has horizontal scrolling (only left-to-right is currently supported, idem), `MyScrollRectItemsAdapter.GetItemWidth(int index)` will ... [idem above] ...

2. `MyScrollRectItemsAdapter.InitOrUpdateItemViewHolder(MyItemViewsHolder newOrRecycledViewsHolder)` will be called ONLY for the items currently visible and each time a new one will become visible:

- use `newOrRecycledViewsHolder.itemIndex` to get the item index, so you can retrieve its associated data model from your data set

- `newOrRecycledViewsHolder.root` will be null if the item is not recycled. So you need to instantiate your prefab (or whatever), assign it and call `newOrRecycledViewsHolder.CollectViews()`

- `newOrRecycledViewsHolder.root` won't be null if the item is recycled. This means that it's assigned a valid object whose UI elements only need their values changed

- update `newOrRecycledViewsHolder`'s views from its associated data model

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