

# Super ScrollView for UGUI 2.2

## Overview

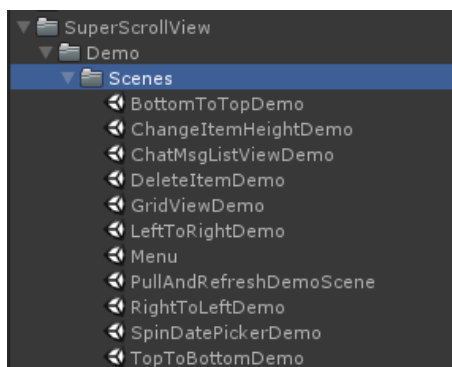
LoopListView2 is a component attaching to the same gameobject of UGUI ScrollRect. It helps the UGUI ScrollRect to support any count items with high performance and memory-saving.

For a ScrollRect with 10,000 items, LoopListView2 does not really create 10,000 items, but create a few items based on the size of the viewport.

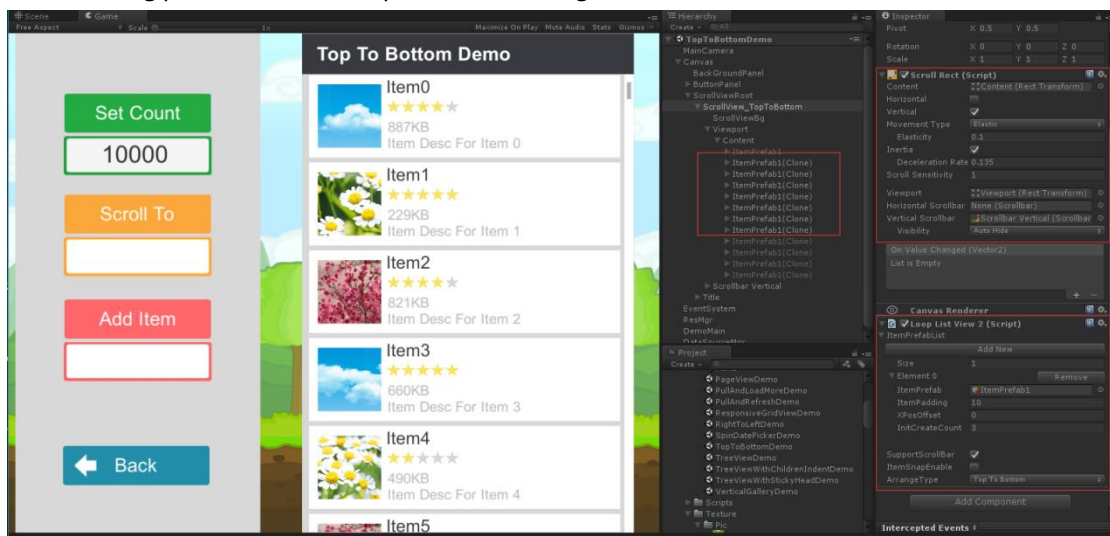
When the ScrollRect moving up, for example, the LoopListView2 component would check the topmost item's position, and once the topmost item is out of the viewport, then the LoopListView2 component would recycle the topmost item, and at the same time check the downmost item's position, and once the downmost item is near the bottom of the viewport, the LoopListView2 component would call the **onGetItemByIndex** handler to create a new item and then position the new created item under the downmost item, so the new created item becomes the new downmost item.

Every item can use a different prefab and can have different height/width.

There are several examples to help you learn the LoopListView2 component, in the folder with path : Assets -> SuperScrollView -> Demo -> Scenes. And **Menu scene is the entry**.



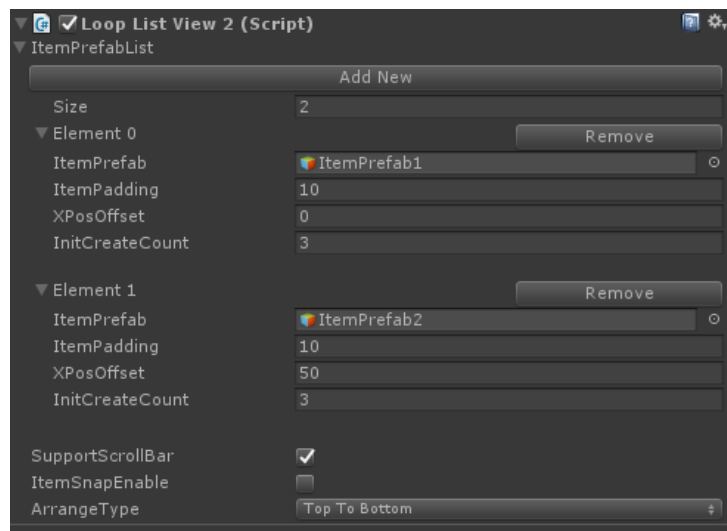
The following picture is what a TopToBottom arranged scrollrect looks like:



In the above picture, the scrollrect have 10000 items, but in fact, only 7 items really created.

# Inspector Settings

In the Inspector, to make sure the LoopListView2 component works well, there are several parameters need to set:



**ItemPrefabList:** this is the existing items to clone.

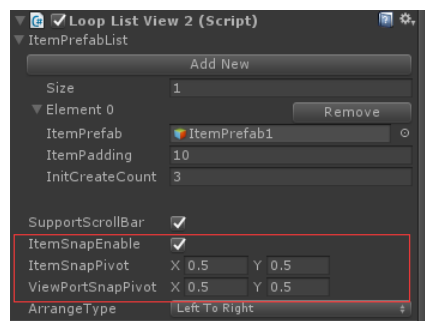
Every item can use a different prefab and every prefab can have different **default padding**( the amount of spacing between each item in the scrollrect). And also, every prefab can have different default x local pos(for **Vertical** scrollrect) or default y local pos(for **Horizontal** scrollrect), and here, is called (X/Y)PosOffset. Every prefab has a pool for getting and recycling action, and the **InitCreateCount** is the count created in pool at start.

In fact, in runtime, every item can have different padding and (X/Y)PosOffset, you can change an item's padding and (X/Y)PosOffset in your `onGetItemByIndex` callback. You can get/set them by `LoopListViewItem2`. `Padding` and `LoopListViewItem2.StartPosOffset`.

**Important note: All the itemPrefab's localScale need to be (1,1,1).**

**SupportScrollbar:** if checked, the LoopListView2 component would support scrollbar.

**ItemSnapEnable:**



if checked, the LoopListView2 component would try to snap item to the configured location in viewport.

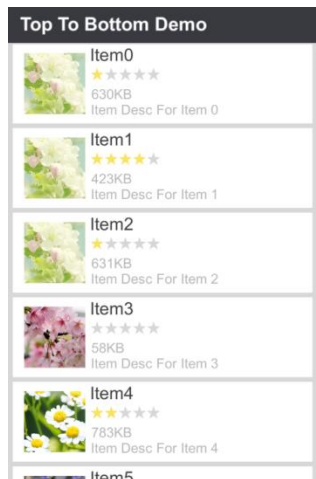
**ItemSnapPivot** is the location of the snap pivot point of the item, defined as a fraction of the size of the rectangle itself. 0,0 corresponds to the lower left corner while 1,1 corresponds to the

upper right corner.

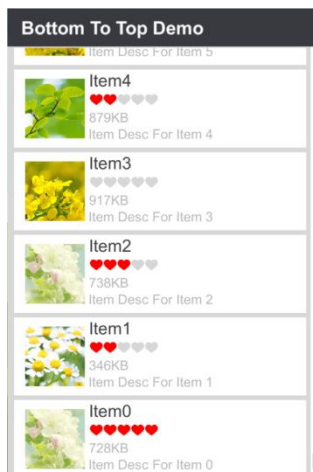
**ViewPortSnapPivot** is the location of the snap pivot point of the ScrollRect ViewPort, defined as a fraction of the size of the rectangle itself. 0,0 corresponds to the lower left corner while 1,1 corresponds to the upper right corner.

**ArrangeType**: the scroll direction, there are four types:

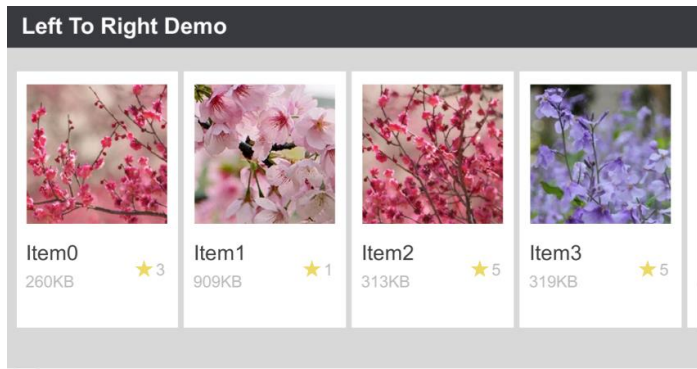
- (1) **TopToBottom**: this type is used for a vertical scrollrect, and the item0,item1,...itemN are positioned one by one **from top to bottom** in the scrollrect viewport, just like the following:



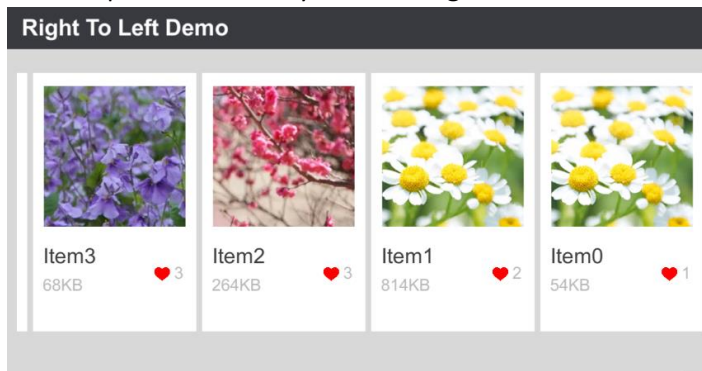
- (2) **BottomToTop**: this type is used for a vertical scrollrect, and the item0,item1,...itemN are positioned one by one **from bottom to top** in the scrollrect viewport, just like the following:



- (3) **LeftToRight**: this type is used for a horizontal scrollrect, and the item0,item1,...itemN are positioned one by one **from left to right** in the scrollrect viewport.



- (4) **RightToLeft:** this type is used for a horizontal scrollrect, and the item0,item1,...itemN are positioned one by one **from right to left** in the scrollrect viewport.



# Important Public Method

```
public void InitListView(int itemTotalCount,
    System.Func<LoopListView2, int, LoopListViewItem2> onGetItemByIndex,
    LoopListViewInitParam initParam = null)
```

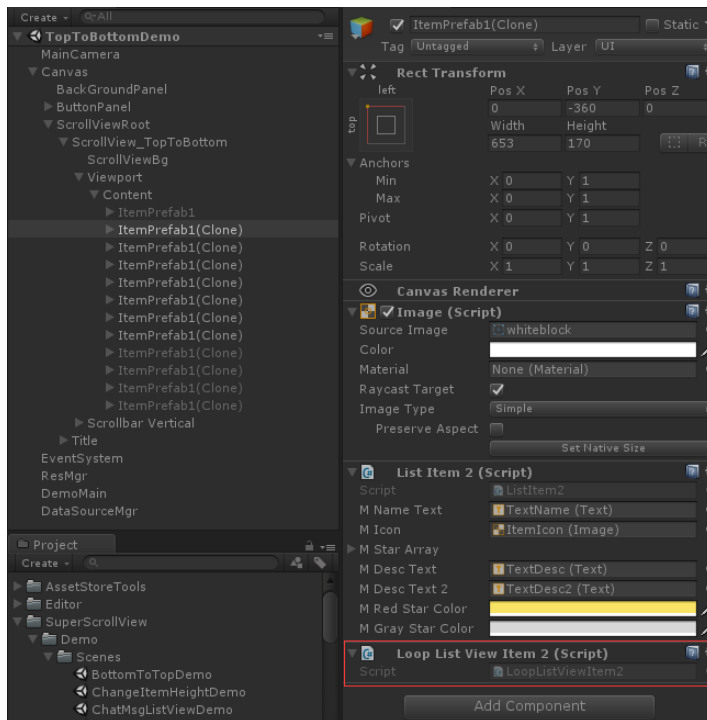
**InitListView** method is to initiate the LoopListView2 component. There are 3 parameters:

**itemTotalCount**: the total item count in the scrollbar. If this parameter is set -1, then means there are infinite items, and scrollbar would not be supported, and the ItemIndex can be from **-MaxInt** to **+MaxInt**. If this parameter is set sssa value  $\geq 0$ , then the ItemIndex can only be from 0 to **itemTotalCount -1**.

**onGetItemByIndex**: when an item is getting in the scrollrect viewport, this Action will be called with the item' index as a parameter, to let you create the item and update its content.

**LoopListViewItem2** is the return value of **onGetItemByIndex**

Every created item has a **LoopListViewItem2** component auto attached:



LoopListViewItem2 component is very sample:

```
public class LoopListViewItem2 : MonoBehaviour
{
    int mItemIndex = -1;
    int mItemId = -1;
    LoopListView2 mParentListView = null;
    bool mIsInitHandlerCalled = false;
    string mItemPrefabName;
    RectTransform mCachedRectTransform;
    float mPadding;
```

The **mItemIndex** property indicates the item's index in the list, as mentioned above, if **itemTotalCount** is set -1, then the **mItemIndex** can be from **-MaxInt** to **+MaxInt**. If **itemTotalCount** is set a value  $\geq 0$ , then the **mItemIndex** can only be from 0 to **itemTotalCount -1**.

The **ItemId** property indicates the item's id. This property is set when the item is created or fetched from pool, and will no longer change until the item is recycled back to pool.

The following codes is an example of **onGetItemByIndex**:

```
LoopListViewItem2 OnGetItemByIndex(LoopListView2 listView, int index)
{
    if (index < 0 || index >= DataSourceMgr.Get.TotalItemCount)
    {
        return null;
    }

    //get the data to showing
    ItemData itemData = DataSourceMgr.Get.GetItemDataByIndex(index);
    if(itemData == null)
    {
        return null;
    }

    /*get a new item. Every item can use a different prefab, the parameter of the
NewListViewItem is
the prefab' name.
And all the prefabs should be listed in ItemPrefabList in LoopListView2 Inspector
Setting */
    LoopListViewItem2 item = listView.NewListViewItem("ItemPrefab1");
    ListItem2 itemScript = item.GetComponent<ListItem2>();
    // IsInitHandlerCalled is false means this item is new created but not fetched from
pool.
    if (item.IsInitHandlerCalled == false)
    {
        item.IsInitHandlerCalled = true;
        itemScript.Init(); // here to init the item, such as add button click event listener.
    }

    //update the item' s content for showing, such as image, text.
    itemScript.SetItemData(itemData, index);
    return item;
}
```

```
public LoopListViewItem2 NewListViewItem(string itemPrefabName)
```

This method is used to get a new item, and the new item is a clone from the prefab named itemPrefabName. This method is usually used in **onGetItemByIndex**.

```
public void SetListItemCount(int itemCount, bool resetPos = true)
```

This method may use to set the item total count of the scrollview at runtime. If this parameter is set -1, then means there are infinite items, and scrollbar would not be supported, and the ItemIndex can be from **-MaxInt** to **+MaxInt**. If this parameter is set a value  $\geq 0$ , then the

ItemIndex can only be from 0 to **itemTotalCount -1**.

If resetPos is set false, then the scrollrect's content position will not changed after this method finished.

```
public LoopListViewItem2 GetShownItemByItemIndex(int itemIndex)
```

To get the visible item by itemIndex. If the item is not visible, then this method return null.

```
public LoopListViewItem2 GetShownItemByIndex(int index)
```

All visible items is stored in a List<LoopListViewItem2> , which is named `mItemList` ; this method is to get the visible item by the index in visible items list. The parameter index is from 0 to `mItemList.Count`.

```
public int ShownItemCount
{
    get
    {
        return mItemList.Count;
    }
}
```

To get the total count of all visible items.

```
public void RefreshItemByItemIndex(int itemIndex)
```

To update a item by itemIndex. if the itemIndex-th item is not visible, then this method will do nothing. Otherwise this method will first call `onGetItemByIndex(itemIndex)` to get an updated item and then reposition all visible items' position.

```
public void RefreshAllShownItem()
```

This method will update all visible items.

```
public void MovePanelToItemIndex(int itemIndex, float offset)
```

This method will move the scrollrect content's position to ( the positon of itemIndex-th **item** + offset ), and **in current version the itemIndex is from 0 to MaxInt, offset is from 0 to scrollrect viewport size**.

```
public void OnItemSizeChanged(int itemIndex)
```

For a vertical scrollrect, when a visible item's height changed at runtime, then this method should be called to let the LoopListView2 component reposition all visible items' position.

For a horizontal scrollrect, when a visible item's width changed at runtime, then this method should be called to let the LoopListView2 component reposition all visible items' position.

```
public void FinishSnapImmediately()
```

Snap move will finish at once.

```
public int CurSnapNearestItemIndex
```

Get the nearest item index with the viewport snap point.

```
public void SetSnapTargetItemIndex(int itemIndex)
```

Set the snap target item index. This method is used in PageViewDemo.

```
public void ClearSnapData()
```

Clear current snap target and then the LoopScrollView2 will auto snap to the CurSnapNearestItemIndex.