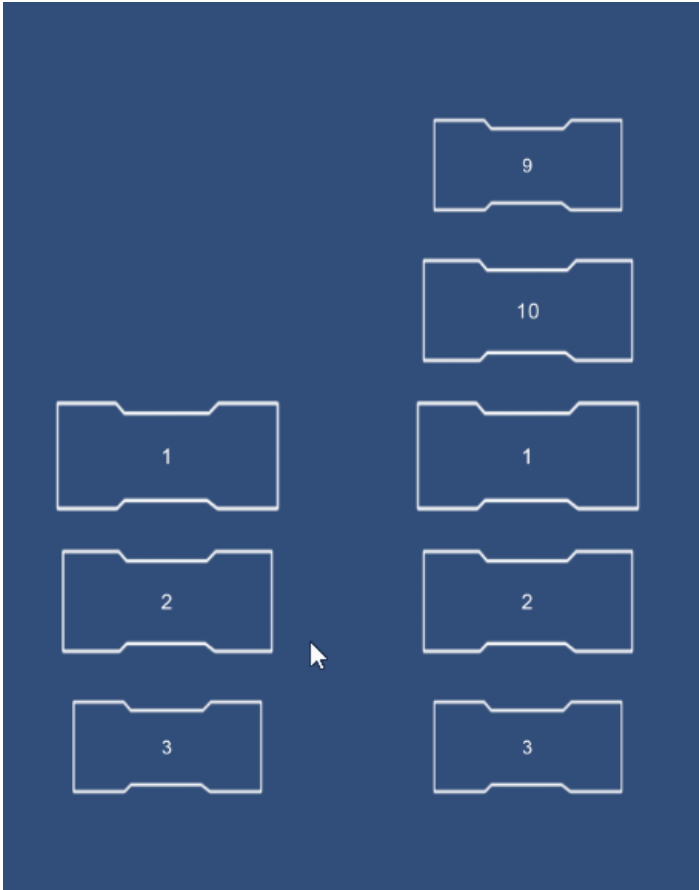


Circular Scrolling List



Features

- Use finite list boxes to display infinite list items
- Use Unity's event system to detect input events
- Use `AnimationCurve` to define the layout and the movement
- Support all three render modes of the canvas plane
- Support Unity 5+ (Tested in Unity 5.6.7f1)

List mode

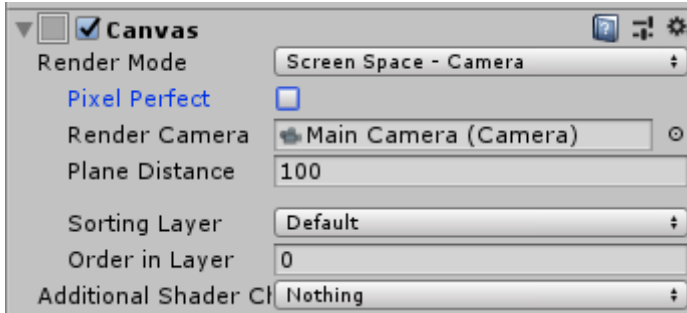
- List type: Circular or Linear mode
- Control mode: Drag, Function, or Mouse wheel
- Direction: Vertical or Horizontal

[Demo video](#)

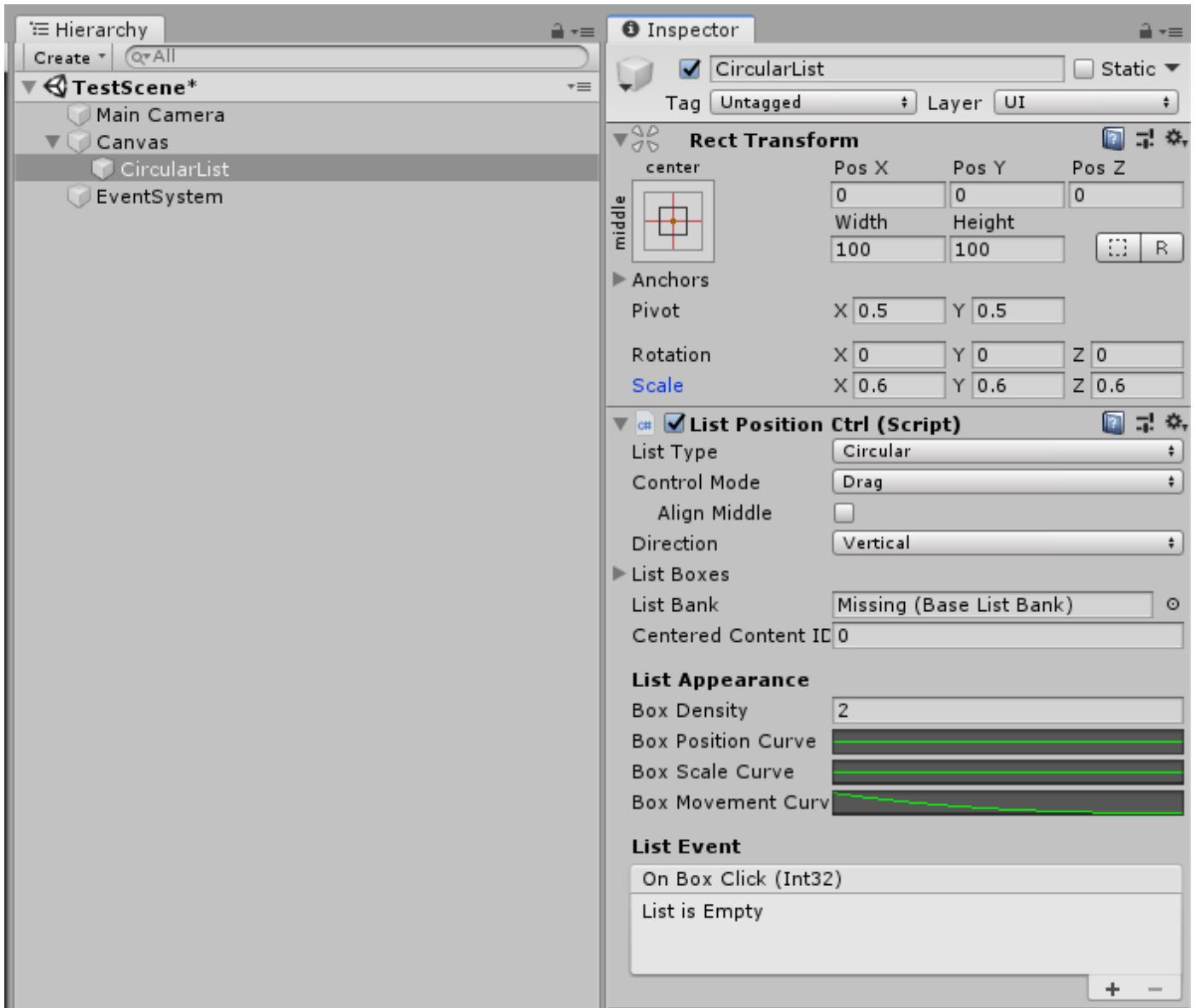
How to Use

Set up the List

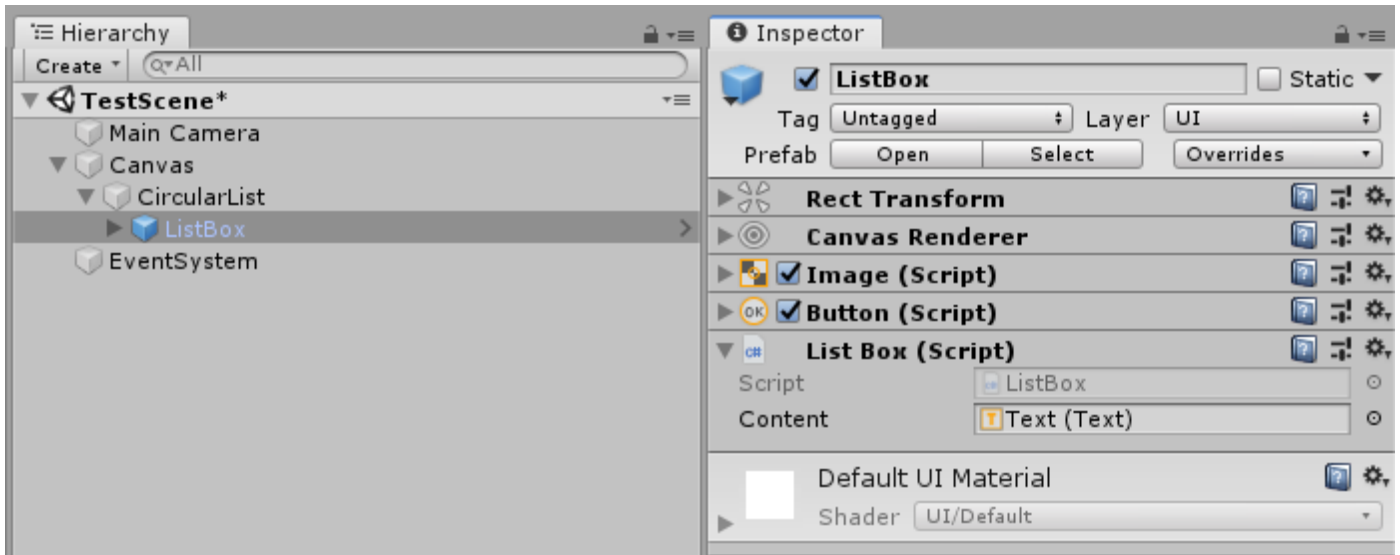
1. Add a Canvas plane to the scene. Set the render mode to "Screen Space - Camera" (for example), and assign the "Main Camera" to the "Render Camera".



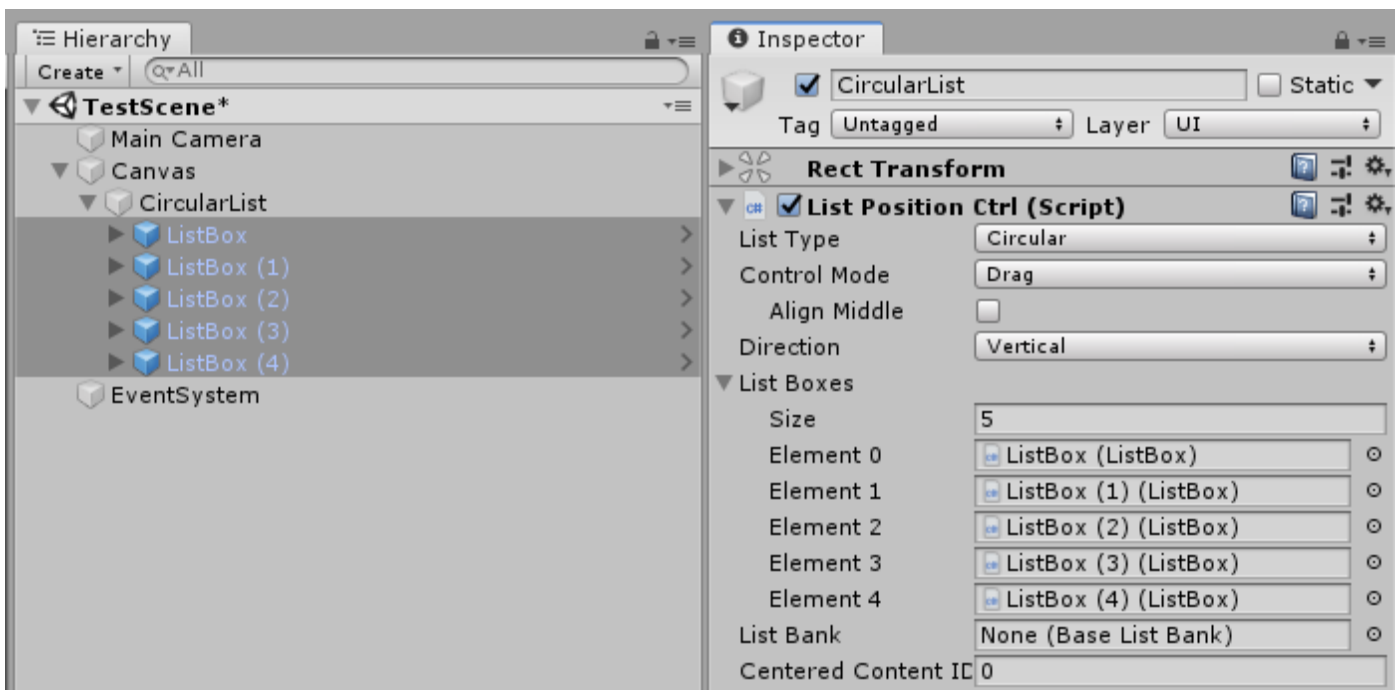
2. Create an empty gameobject as the child of the canvas plane, rename it to `CircularList` (or another name you like), set the scale to 0.6, and attach the script `ListPositionCtrl.cs` to it.



3. Create a Button gameobject as the child of the `CircularList` , rename it to `ListBox` , change the sprite and the font size if needed.
4. Attach the script `ListBox.cs` to it, assign the gameobject "Text" of the Button to the "Content" of the `ListBox.cs` , and then create a prefab of it .



5. Duplicate the gameobject `ListBox` or create gameobjects from the prefab as many times as you want (4 times here, for exmaple), and assign them to the "List Boxes" of the script `ListPositionCtrl.cs` .



Create ListBank

1. Create a new script named `MyListBox.cs` and launch the editor.
2. Inherit the abstract class `BaseListBank` (The class `BaseListBank` inherits the `MonoBehaviour` , therefore, you could initialize list contents in `Start()` and attach the script to a gameobject).
3. There are two functions which must be implemented:

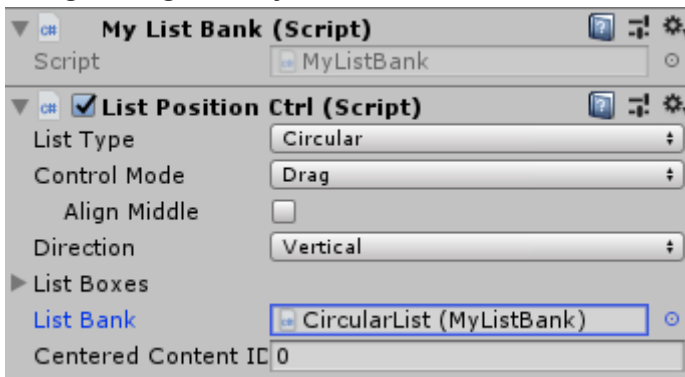
- `public string GetListContent(int index)` : Get the string representation of the specified content.
- `public int GetListLength()` : Get the number of the list contents.

```
// The example of the simplest ListBank
public class MyListBank: BaseListBank
{
    private int[] _contents = {
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10
    };

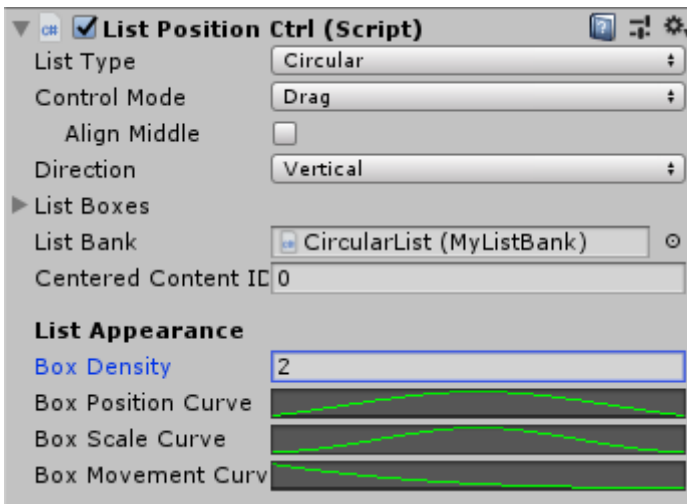
    public override string GetListContent(int index)
    {
        return _contents[index].ToString();
    }

    public override int GetListLength()
    {
        return _contents.Length;
    }
}
```

4. Attach the script `MyListBank.cs` to the gameobject `CircularList` (or another gameobject), and assign the gameobject to the "List Bank" of the script `ListPositionCtrl.cs`.



Configure the List Mode and Appearance



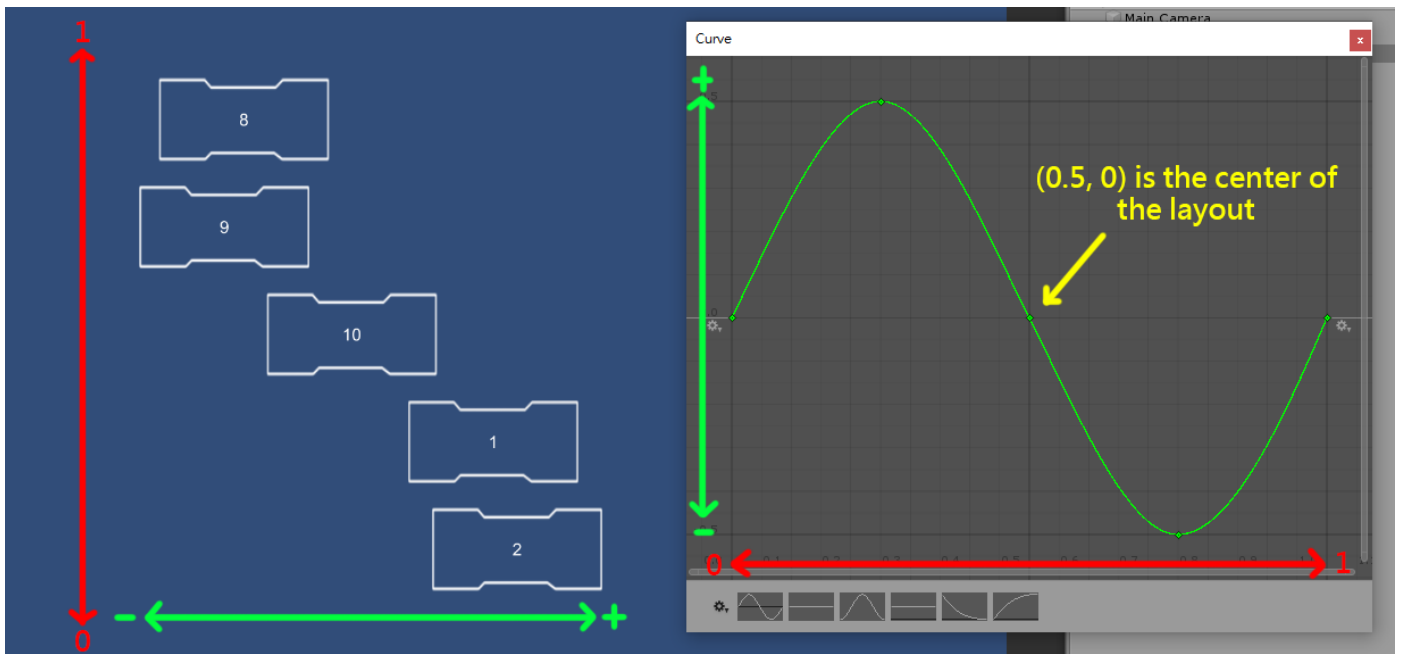
Basic Configuration

- List Type: Circular or Linear
- Control Mode: Drag, Function, or Mouse Wheel
 - Align Middle: Whether to align a box at the center of the list when the list stop moving. Only available in Drag mode.
 - If the Function mode is selected, move the list by invoking `ListPositionCtrl.MoveOneUnitUp()` and `ListPositionCtrl.MoveOneUnitDown()`. For example, you could assign these two functions to buttons to control the list.
- Direction: Vertical or Horizontal
- Centered Content ID: The initial content ID for the centered (or focused) box

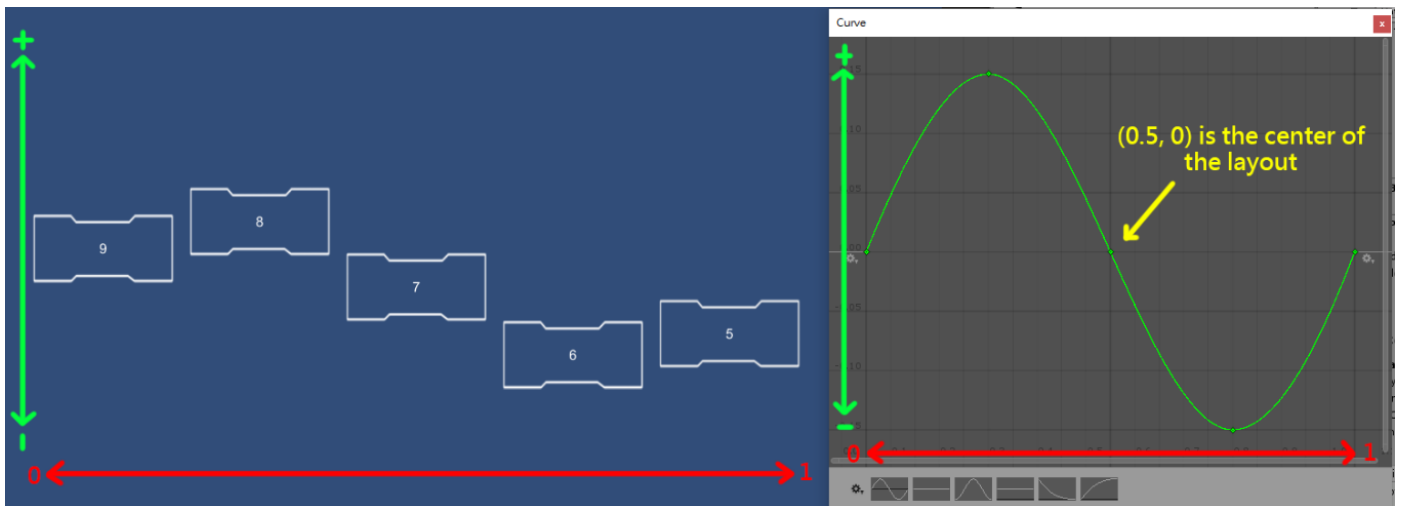
List Appearance

- Box Density: The gap between boxes. The larger, the closer.
- Box Position Curve: The curve specifying the box position. The x axis is the major position of the box, which is mapped to $[0, 1]$ (from the smallest value to the largest value). The y axis is the factor of the passive position.

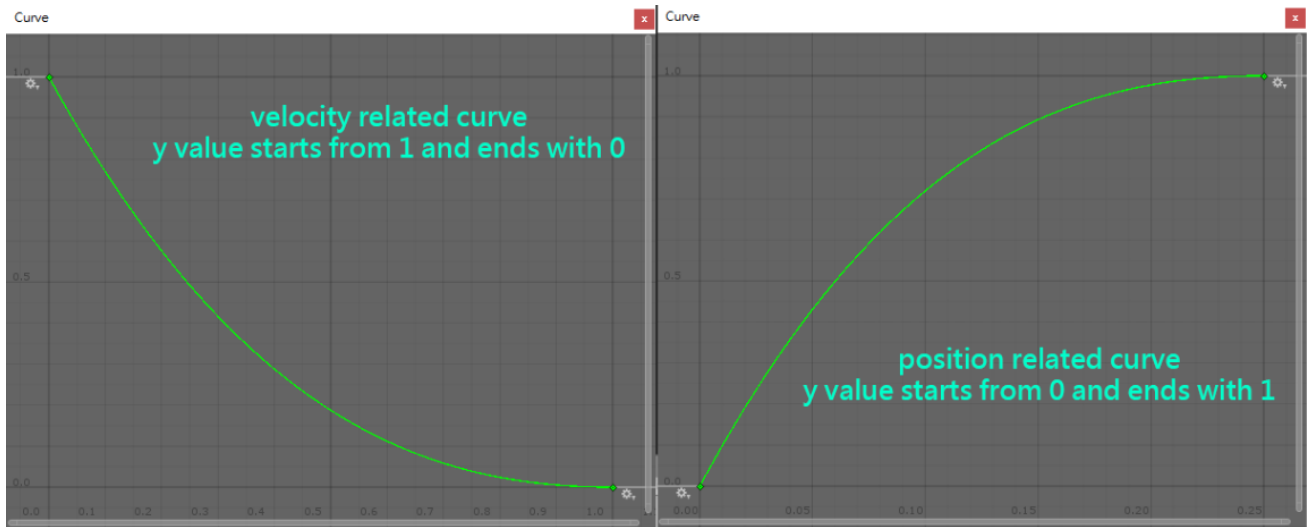
For example, in the vertical mode, the major position is the y position and the passive position is the x position:



It is intuitive in the horizontal mode:

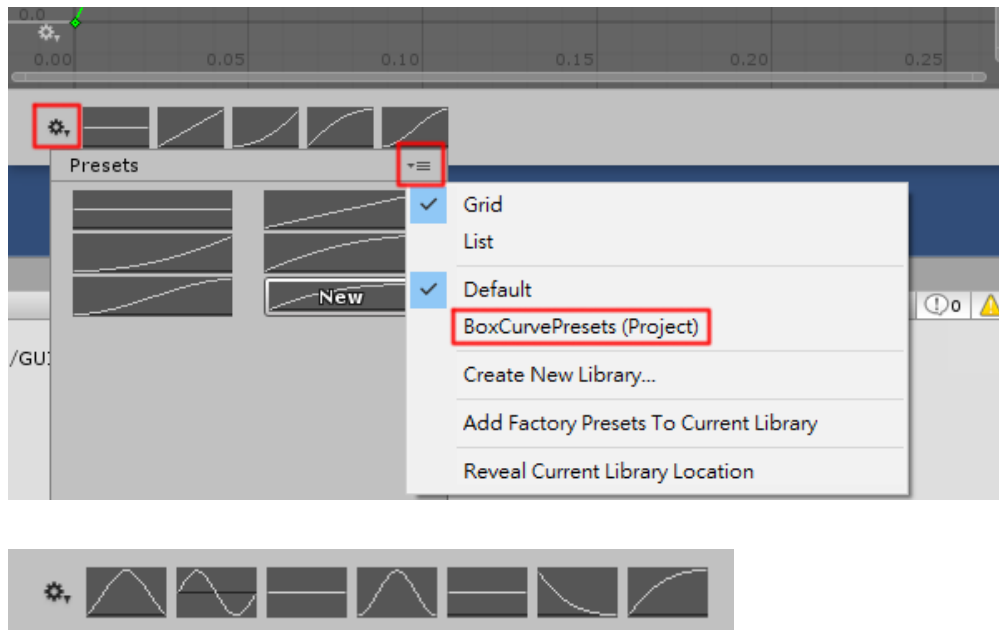


- Box Scale Curve: Similar to the Box Position Curve, but the y axis defines the scale value of the box at that major position.
- Box Movement Curve: The curve specifying movement of the box. The x axis is the movement duration in seconds, which starts from 0. The value of y axis is depended on the mode:
 - In the Drag mode, it is the factor relative to the releasing velocity;
 - In the Function or Mouse Wheel mode, it is the factor relative to the target position.



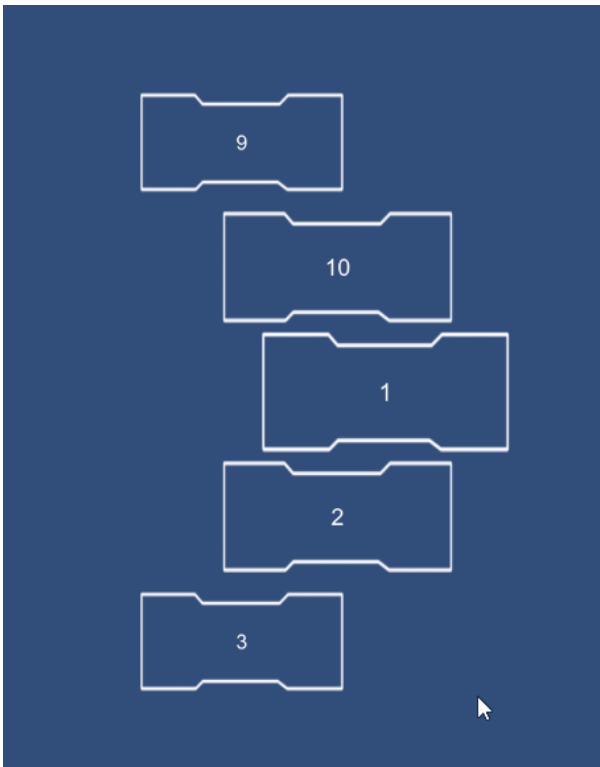
Curve Presets

The project provides curve presets. Open the curve editing panel and select the `BoxCurvePresets` to use them.



The first three curves are position curves, the 4th and 5th one are scale curves, the 6th one is a velocity related curve, and the last one is a position related curve.

After configuration, the set up of the list is done! Click Play button of the scene to check the list. You could adjust the position and the size of the list by setting the position and the scale of the gameobject `CircularList`.



Get the ID of the Selected Content

There are two ways to get ID of the selected content.

1. Create callback function
2. Get the centered content ID

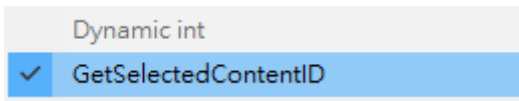
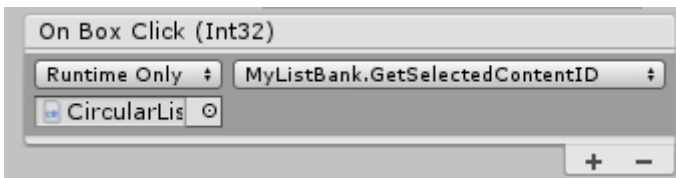
Create Callback Function

When a box is clicked, the `ListPositionCtrl1` will launch the event `onBoxClick` (actually launch from the `Button.onClick` event). The callback function (or the listener) for the event must have 1 parameter for receiving the ID of the selected content.

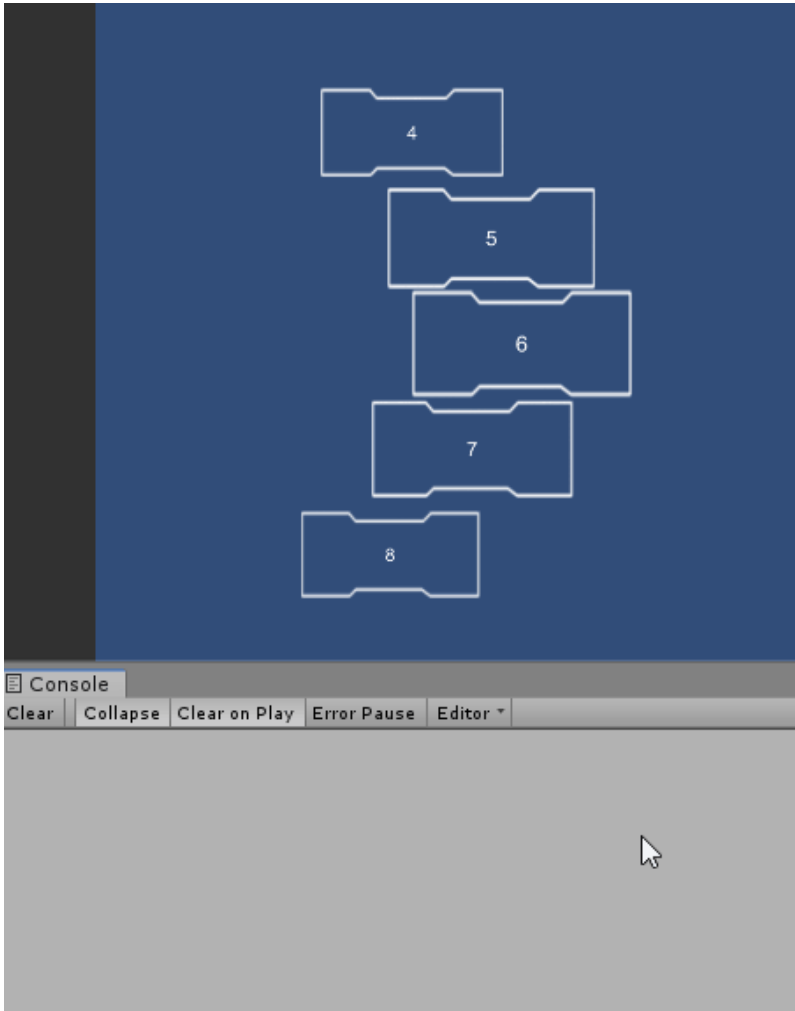
For example, add a function `GetSelectedContentID` as the callback function to the class `MyListBank`.

```
public void GetSelectedContentID(int contentID)
{
    Debug.Log("Selected content ID: " + contentID.ToString() +
        ", Content: " + GetListContent(contentID));
}
```

Then, add it to the "On Box Click (Int 32)" of the script `ListPositionCtrl1.cs` in the inspector. (Note that select the function in the "dynamic int" section)



It will be like:



Get the Centered Content ID

The other way is to invoke the function `ListPositionCtrl1.GetCenteredContentID()` which will find the list box closest to the center and return the content ID of it.

For example, create a function which will update the content of the centered box to the Text, and use a Button to invoke it.

```

public class MyApplication: MonoBehaviour
{
    public ListPositionCtrl list;
    public Text displayText;

    public void DisplayCenteredContent()
    {
        int contentID = list.GetCenteredContentID();
        string centeredContent = list.listBank.GetListContent(contentID);
        displayText.text = "Centered content: " + centeredContent;
    }
}

```

It will be like:

