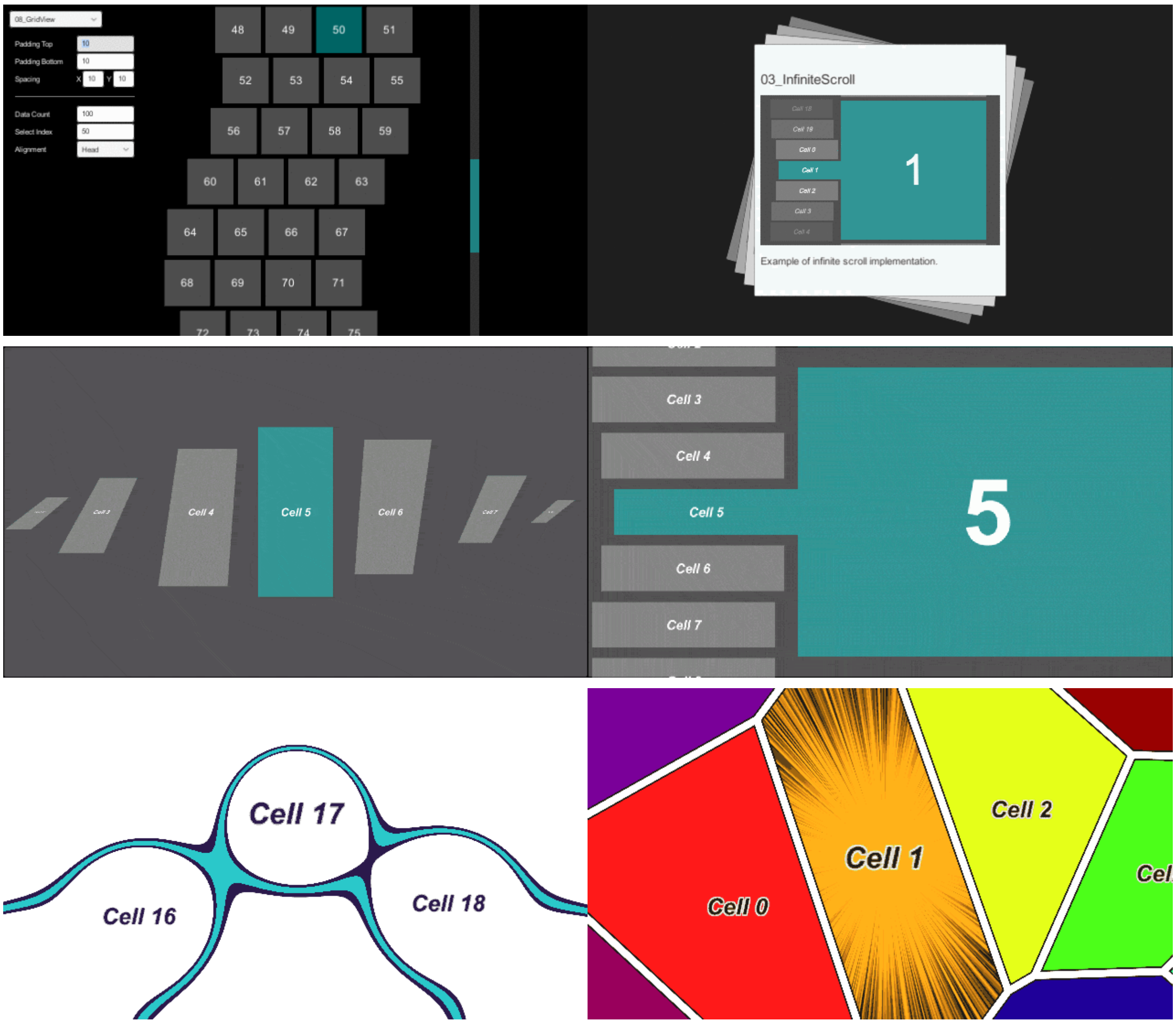
FancyScrollView

license MIT | demo WebGL | API Documentation | openupm v1.8.7

A general-purpose ScrollView component that can implement highly flexible animations. Infinite scrolling is also supported.



Requirements

unity 2019.4+ | .NET 4.x

Installation

Unity Asset Store

Consider purchasing for further development by purchasing from the Unity Asset Store .

OpenUPM

Add the package to the Unity Project from the OpenUPM registry

Add the package to the Unity Project from the OpenUPM registry.

```
openupm add jp.setchi.fancyscrollview
```

Unity Package Manager

`Packages/manifest.json`Add a reference to the repository to a file in your project directory .

```
{
  "dependencies": {
    "jp.setchi.fancyscrollview":
"https://github.com/setchi/FancyScrollView.git#upm"
  }
}
```

Features

You can implement scrolling animation freely

When FancyScrollView updates the scroll position, it passes the normalized position of the viewport range to each cell. On the cell side, the position and appearance of scrolling are controlled by the cell itself based on the value of `0.0`~ `1.0`. The sample uses Animator and mathematical expressions to implement movement during scrolling.

Operates lightly even if the number of data items is large

Only the number of cells needed for display will be generated and the cells will be reused. You can check the operation while actually increasing the number of data in Demo . FancyScrollRect and FancyGridView in, cell in the scroll margins until it is re-use can also be specified.

You can freely exchange messages between cells and scroll views.

`Context`Via, you can simply implement the process of detecting the click of a cell in the scroll view and issuing instructions to the cell from the scroll view. An implementation example (Example/02_FocusOn) is included, so please refer to it.

You can scroll or jump to a specific cell

You can also specify the number of seconds to move and Easing. For more information API Documentation of Class Scroller Please refer to.

You can set the scrolling behavior in detail

You can set the behavior related to scrolling, such as the presence or absence of inertia and the deceleration rate. For more information API Documentation of Class Scroller Please refer to.

Supports snap

When snap is enabled, it will move to the nearest cell just before scrolling stops. You can specify the speed threshold at which snapping starts, the number of seconds to move, and Easing. FancyScrollRect and FancyGridView do not support snapping.

Supports infinite scroll

You can implement infinite scrolling by setting the following in Inspector.

1. When `Loop` is turned on, cells are cycled so that the last cell is before the first cell and the first cell is after the last cell.
2. Have been used in the sample `Scroller`when using the, `Movement Type`wo `Unrestricted`by setting in, scroll range is unlimited. Infinite scrolling can be realized by combining with 1.

An implementation example (Examples/03_InfiniteScroll) is included, so please refer to it as well. FancyScrollRect and FancyGridView do not support infinite scrolling.

Examples

demo WebGL

See FancyScrollView/Examples .

Name	Description
01_Basic	This is an implementation example of the simplest configuration.
02_FocusOn	It is an implementation example that focuses on the left and right cells with a button.
03_InfiniteScroll	It is an implementation example of infinite scroll.
04_Metaball	This is an implementation example of metaball using a shader.
05_Voronoi	Voronoi implementation example using a shader.
06_LoopTabBar	It is an implementation example that switches screens with tabs.
07_ScrollRect	`ScrollRect`This is an implementation example of the style with scroll bar .
08_GridView	It is an implementation example of grid layout.
09_LoadTexture	An example of loading and displaying textures.

Usage

In the simplest configuration,

- Object for passing data to cell
- cell
- Scroll view

Implementation is required.

Implementation

Defines an object for passing data to cells.

```
class ItemData
{
    public string Message { get; }

    public ItemData(string message)
    {
        Message = message;
    }
}
```

`FancyCell<TItemData>` And implement your own cell.

```
using UnityEngine;
using UnityEngine.UI;
using FancyScrollView;

class MyCell : FancyCell<ItemData>
{
    [SerializeField] Text message = default;

    public override void UpdateContent(ItemData itemData)
    {
        message.text = itemData.Message;
    }

    public override void UpdatePosition(float position)
    {
        // position は 0.0 ~ 1.0 の値です
        // position に基づいてスクロールの外観を自由に制御できます
    }
}
```

`FancyScrollView<TItemData>` And implement your own scroll view.

```
using UnityEngine;
using System.Linq;
using FancyScrollView;

class MyScrollView : FancyScrollView<ItemData>
{
    [SerializeField] Scroller scroller = default;
    [SerializeField] GameObject cellPrefab = default;

    protected override GameObject CellPrefab => cellPrefab;

    void Start()
    {
        scroller.OnValueChanged(base.UpdatePosition);
        scroller.OnValueChanged(base.UpdatePosition);
    }

    public void UpdateData(IList<ItemData> items)
    {
        base.UpdateContents(items);
        scroller.SetTotalCount(items.Count);
    }
}
```

Fills the scroll view with data.

```
using UnityEngine;
using System.Linq;

class EntryPoint : MonoBehaviour
{
    [SerializeField] MyScrollView myScrollView = default;

    void Start()
    {
        var items = Enumerable.Range(0, 20)
            .Select(i => new ItemData($"Cell {i}"))
            .ToArray();

        myScrollView.UpdateData(items);
    }
}
```

See Examples and API Documentation for more details.

Author

setchi

License

MIT