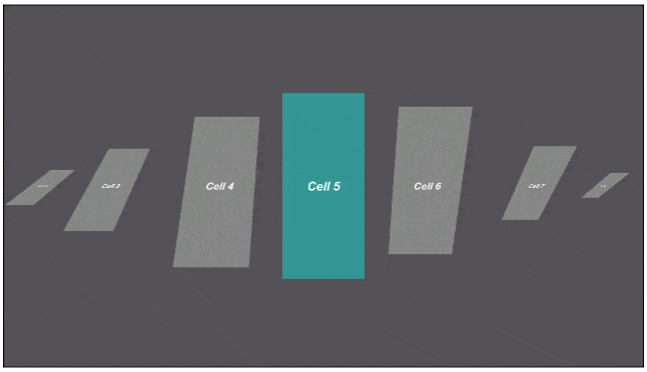
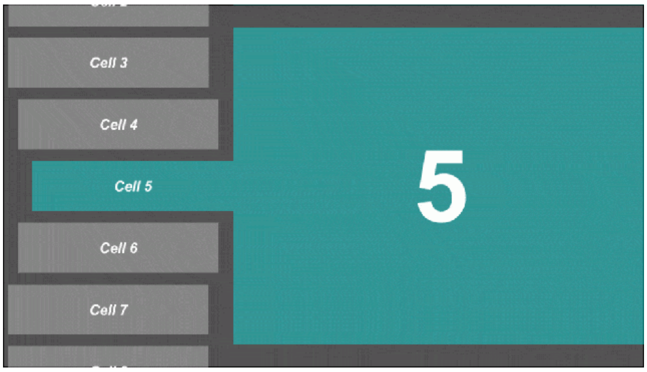
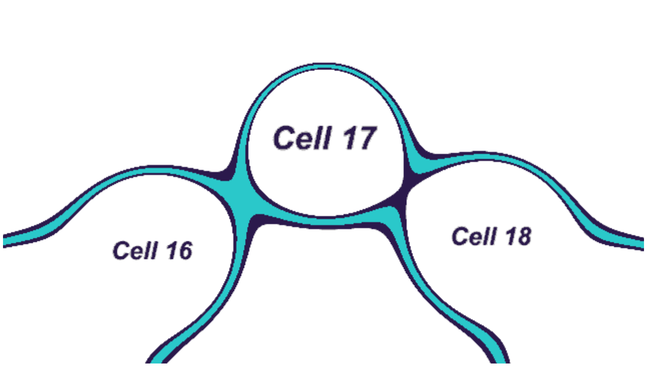
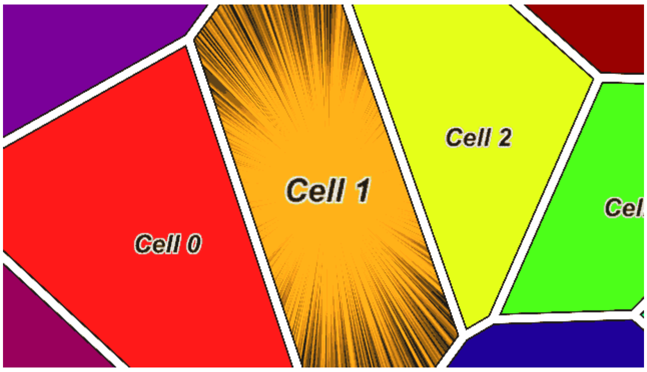
FancyScrollView license MIT

A generic ScrollView component that can implement highly flexible animations. Infinite scrolling is also supported.









Demo

https://setchi.jp/FancyScrollView/

Requirements

- Unity 2019.2 or later.
- .NET 4.x Scripting Runtime

Installation

Unity Asset Store

Install the package from the Asset Store into the project.

Unity Package Manager *(Example scenes not included)*

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

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

Manual

Clone or Download this repository.

Features

You can freely implement scroll animation

When FancyScrollView updates the cell position, it passes the normalized value of the visible region to each cell. On the cell side, you can freely control the appearance of the scroll based on a value between 0.0 and 1.0. The sample uses Animator to control cell movement.

It works lightly even if the number of data is large

Only the number of cells required for display is generated and the cells are reused.

Messages can be exchanged freely between cells and scroll view

`Context`It is possible to simply implement the process of detecting that the cell is clicked in the scroll view or issuing an instruction to the cell from the scroll view. An implementation example (Examples / 02_FocusOn) is included, so please refer to it.

Scroll and jump to a specific cell

You can also specify the number of seconds to move and Easing. For details , see Scroller-Methods in the API Reference .

You can fine-tune the scrolling behavior

You can set scroll behavior such as inertia and deceleration rate. For details , see Scroller-Inspector in API Reference .

Supports snapping

When snapping is enabled, it moves to the nearest cell just before scrolling stops. You can specify the speed threshold at which snapping begins, the number of seconds to move, and Easing.

Supports infinite scroll

Infinite scrolling can be implemented by setting the following in Inspector.

1. `FancyScrollView`When `Loop`is turned on, the cells circulate so that the last cell is placed before the first cell, and the first cell is arranged 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 achieved by combining with 1.

An implementation example (Examples / 03_InfiniteScroll) is included.

Examples

See FancyScrollView / Examples .

Name	Description
01_Basic	This is an example of the simplest configuration.
02_FocusOn	This is an example of focusing on the left and right cells with a button.
03_InfiniteScroll	An implementation example of infinite scroll.

Usage

In the simplest configuration,

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

Implementation is required.

Implementation

Defines an object for passing data to cells.

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

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

`FancyScrollViewCell<TItemData>` Inherit and implement your own cell.

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

public class MyScrollViewCell : FancyScrollViewCell<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> Inherit and implement your own scroll view.

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

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

    protected override GameObject CellPrefab => cellPrefab;

    void Start()
    {
        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;

public 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);
    }
}
```

API Reference

`FancyScrollView<TItemData, TContext>`

Abstract base class for scroll views that control cells.

```
public abstract class FancyScrollView<TItemData, TContext>
    : MonoBehaviour where TContext : class, new()
```

`Context` Use here when is not required.

```
public abstract class FancyScrollView<TItemData>
    : FancyScrollView<TItemData, FancyScrollViewNullContext>
```

Inspector

Type	Name	Summary
`float`	`Cell Interval`	Specify the cell spacing between float.Epsilon and 1.0.
`float`	`Scroll Offset`	Specifies the scroll offset; for example, if you specify 0.5 and the scroll position is 0, the position of the first cell is 0.5.
`bool`	`Loop`	Turn on to cycle the cells so that the last cell before the first cell, the first cell after the last cell, and turn on if you want infinite scrolling.
`Transform`	`Cell Container`	Specifies the transform that is the parent element of the cell.

Properties

Type	Name	Summary
`GameObject`	`CellPrefab`	Cell prefab.
`IList<TItemData>`	`ItemsSource`	Items source.
`TContext`	`Context`	Context.

Methods

Type	Name	Summary
`void`	`UpdateContents(IList<TItemData> itemsSource)`	Updates the contents.
`void`	`Refresh()`	Refreshes the cells.
`void`	`UpdatePosition(float position)`	Updates the scroll position.

`FancyScrollViewCell<TItemData, TContext>`

An abstract base class for cells.

```
public abstract class FancyScrollViewCell<TItemData, TContext>
    : MonoBehaviour where TContext : class, new()
```

`Context` Use here when is not required.

```
public abstract class FancyScrollViewCell<TItemData>
    : FancyScrollViewCell<TItemData, FancyScrollViewNullContext>
```

Properties

Type	Name	Summary
`int`	`Index`	Gets or sets the index of the data.
`bool`	`IsVisible`	Gets a value indicating whether this cell is visible.
`TContext`	`Context`	Context.

Methods

Type	Name	Summary
`void`	`SetupContext(TContext context)`	Setup the context.
`void`	`SetVisible(bool visible)`	Sets the visible.
`void`	`UpdateContent(TItemData itemData)`	Updates the content.
`void`	`UpdatePosition(float position)`	Updates the position.

`Scroller`

A component that controls the scroll position.

```
public class Scroller
    : UIBehaviour, IBeginDragHandler, IEndDragHandler, IDragHandler
```

Inspector

Type	Name	Summary
`RectTransform`	`Viewport`	Specifies the RectTransform that will be the viewport, and performs gesture detection within the range of the RectTransform specified here.
`ScrollDirection`	`Direction Of Recognize`	Specify the direction to recognize the gesture with Vertical or Horizontal.
`MovementType`	`Movement Type`	Specifies the behavior to use when content moves beyond the scroll range.

Type	Name	Summary
`float`	`Elasticity`	Specifies the amount of elasticity to use when the content moves beyond the scroll range.
`float`	`Scroll Sensitivity`	Specify the scroll sensitivity.
`bool`	`Inertia`	Specify inertia on / off.
`float`	`Deceleration Rate`	Effective only when Inertia is on, specify the deceleration rate.
`bool`	`Snap.Enable`	Check to enable Snap.
`float`	`Snap.Velocity Threshold`	Specify the threshold speed at which Snap starts.
`float`	`Snap.Duration`	Specify the travel time during Snap in seconds.
`Ease`	`Snap.Easing`	Specify Easing for Snap.

Methods

Type	Name	Summary
`void`	`OnValueChanged(Action<float> callback)`	Set the callback when the scroll position changes.
`void`	`OnSelectionChanged(Action<int> callback)`	Set a callback when the selected cell changes.
`void`	`JumpTo(int index)`	Jump to the specified cell.
`void`	`ScrollTo(int index, float duration, Action onComplete = null)`	Scroll to the specified cell.
`void`	`ScrollTo(int index, float duration, Ease easing, Action onComplete = null)`	Scroll to the specified cell.
`void`	`ScrollTo(int index, float duration, Func<float, float> easingFunction, Action onComplete = null)`	Scroll to the specified cell.
`void`	`SetTotalCount(int totalCount)`	Set the total number of items. (Index: 0 ~ totalCount-1)

Author

setchi

License

MIT