
New UI Widgets Documentation

Release 1.11.2f1

Ilia Novikov

Feb 28, 2020

1	New in 1.10	1
2	Widgets	3
2.1	Recommended Unity UI documentation	3
2.2	Collections	3
2.3	Containers	4
2.4	Dialogs	4
2.5	Input	4
2.6	Misc	5
3	Widgets Generation	7
3.1	Requirements	8
3.2	Supported types	8
3.3	Limitations	9
3.4	Known Problems	9
3.5	INotifyPropertyChanged Support	9
3.6	Replacing generated code	10
3.6.1	Collections	10
3.6.2	Autocomplete	13
4	Using Widgets	15
4.1	Collections	15
4.1.1	Combobox	15
4.1.2	ListView, TileView and Table	15
4.1.3	Paginator	26
4.1.4	TreeView	28
4.2	Containers	31
4.2.1	Accordion	31
4.2.2	Tabs	33
4.3	Dialogs	34
4.3.1	DatePicker	34
4.3.2	Dialog	35
4.3.3	FileDialog	38
4.3.4	FolderDialog	41
4.3.5	Notifications	42
4.3.6	Pickers	44
4.3.7	Popup	45

4.4	Input	46
4.4.1	Autocomplete	46
4.4.2	Calendar	47
4.4.3	Centered Slider	49
4.4.4	ColorPicker	50
4.4.5	DateScroller, DateTimeScroller, TimeScroller	51
4.4.6	RangeSlider	54
4.4.7	Spinner	56
4.5	Misc	57
4.5.1	ProgressbarDeterminate	57
4.5.2	ProgressbarIndeterminate	59
5	Components	61
5.1	Bring to Front	61
5.1.1	Options	61
5.2	Drag and Drop components	61
5.2.1	Drag-and-Drop Support for the Collections	61
5.2.2	Options	61
5.2.3	Custom Drag Support	62
5.2.4	Custom Drop Support	64
5.2.5	Swapping content between Drag and Drop components	66
5.3	Draggable	67
5.3.1	Options	67
5.4	EasyLayout	68
5.4.1	Options	68
5.4.2	Events	72
5.5	Layout Switcher	72
5.5.1	Options	72
5.5.2	Events	72
5.6	Lightbox	72
5.7	Object Sliding	72
5.7.1	Options	72
5.7.2	Helper components	73
5.8	Resizable	73
5.8.1	Options	73
5.8.2	Events	74
5.8.3	Resize Children With Parent	74
5.9	Scrollbar Min Size	74
5.9.1	Options	74
5.10	ScrollRect Events	74
5.10.1	Options	74
5.10.2	Events	75
5.11	Selectable Helper	75
5.12	Sidebar	75
5.12.1	Options	75
5.12.2	Events	76
5.13	Splitter	76
5.13.1	Options	76
5.13.2	Events	77
5.14	Switch Group	77
5.14.1	Options	77
5.15	Table Header	77
5.15.1	Options	77
5.16	Tooltip	77

5.16.1	Options	78
5.16.2	Events	78
5.17	TreeView data source	78
6	Styles (Skins)	79
6.1	Style support for the custom widgets	80
7	Localization Support	81
7.1	Default ListViewIcons and TreeView	81
7.2	Generated ListView, TreeView, TileView	82
7.3	Tabs	83
8	Data Bind for Unity Support	85
9	TextMesh Pro Support	87
9.1	Known problems and solutions	88
10	Support	91
11	Changelog	93
11.1	Release 1.11.2	93
11.2	Release 1.11.1	94
11.3	Release 1.11.0	94
11.4	Release 1.10.4	94
11.5	Release 1.10.3	95
11.6	Release 1.10.2	95
11.7	Release 1.10.1	96
11.8	Release 1.10.0	97
11.9	Release 1.9.3	98
11.10	Release 1.9.2	98
11.11	Release 1.9.1	99
11.12	Release 1.9.0	99
11.13	Release 1.8.5	100
11.14	Release 1.8.4	100
11.15	Release 1.8.3	101
11.16	Release 1.8.2	101
11.17	Release 1.8.0	101
11.18	Release 1.7.4	102
11.19	Release 1.7.2	102
11.20	Release 1.7.0	102
11.21	Release 1.6.5	103
11.22	Release 1.6.0	103
11.23	Release 1.5.0	104
11.24	Release 1.4.2	104
11.25	Release 1.4.1	104
11.26	Release 1.4	104
11.27	Release 1.3	104
11.28	Release 1.2	104
11.29	Release 1.1	105
11.30	Release 1.0	105

CHAPTER 1

New in 1.10

- *Widgets Generation*
- *Styles (Skins)*

Most of the widgets can be used without knowledge of the Unity UI, but some of them require a basic understanding of the Unity UI.

2.1 Recommended Unity UI documentation

- [Canvas](#)
- [RectTransform](#)
- [Events and Event Triggers](#)
- [Mask](#)
- [Transitions](#)
- [Layout Groups](#)

2.2 Collections

Collections for your custom types can be created with *Widgets Generation*.

TileView, Table, TreeGraph does not have default implementation like ListView because of no standard for those widgets, so they should be created with *Widgets Generation*.

- **Combobox** Data type `string`.
- **ComboboxIcons**
- **ComboboxIconsMultiselect** ComboboxIcons with multiple selection support.
- **DirectoryTreeView** *
- **FileListView** *
- **ListView** Data type `string`.
- **ListViewColors** Data type `Color`.

- **ListViewInt** Data type `int`.
- **ListViewIcons**
- **ListViewHeight** Data type `string`.
- **ListViewPaginator** Paginator for `ListView`, `TileView`, and `Table`.
- **TreeView**

2.3 Containers

- **Accordion**
- **Tabs** Tabs buttons displayed on the top side.
- **TabsLeft** Tabs buttons displayed on the left side.
- **TabsIcons** Tabs buttons with an icon and buttons displayed on the top side.
- **TabsIconsLeft** Tabs buttons with an icon and displayed on the left side.

2.4 Dialogs

- **DatePicker** Data type `DateTime`.
- **DateTimePicker** Data type `DateTime`.
- **Dialog Template** Template for the custom dialogs.
- **FileDialog** *
- **FolderDialog** *
- **NotifyTemplate** Template for the custom notifications.
- **PickerBool** Data type `bool`.
- **PickerIcons**
- **PickerInt** Data type `int`.
- **PickerString** Data type `string`.
- **Popup** Template for the custom popup.
- **TimePicker** Data type `TimeSpan`.

2.5 Input

- **Autocomplete** Data type `string`.
- **AutocompleteIcons**
- **ButtonBig**
- **ButtonSmall**
- **Calendar**
- **CenteredSlider** Horizontal direction.
- **CenteredSliderVertical** Vertical direction.
- **ColorPicker**

- **ColorPickerRange**
- **ColorPickerRangeHSV**
- **ColorsList** Should be used with **ColorPicker** to save colors.
- **DateTime** Data type `DateTime`.
- **RangeSlider** Data type `int`. Horizontal direction.
- **RangeSliderVertical** Data type `int`. Vertical direction.
- **RangeSliderFloat** Data type `float`. Horizontal direction.
- **RangeSliderFloatVertical** Data type `float`. Vertical direction.
- **Spinner** Data type `int`.
- **SpinnerFloat** Data type `float`.
- **Switch**
- **Time12** Data type `TimeSpan`. 12-hour format with AM / PM switch.
- **Time24** Data type `TimeSpan`. 24-hour format.

2.6 Misc

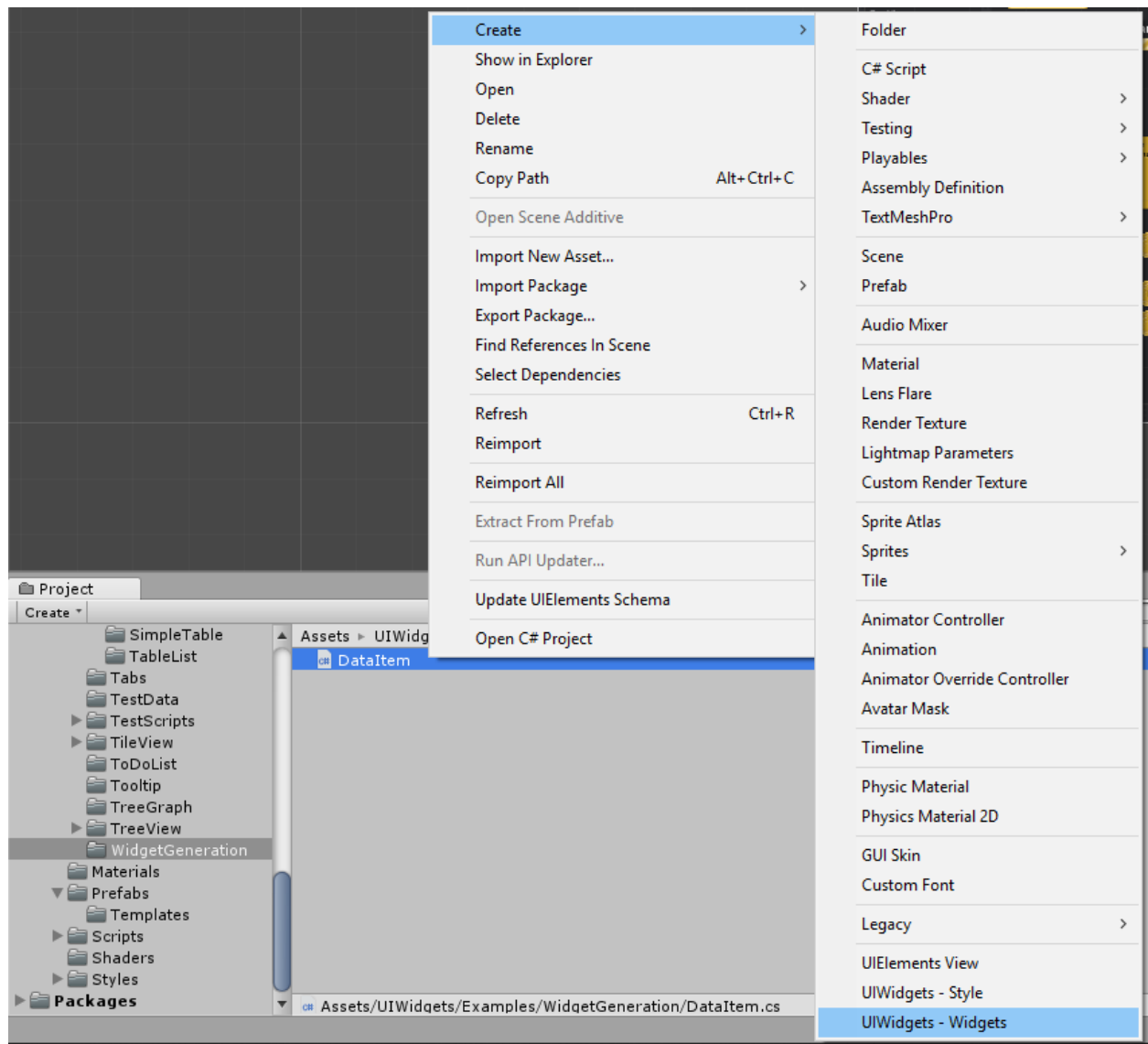
- **AudioPlayer**
- **ProgressbarDeterminate**
- **ProgressbarIndeterminate**
- **ScrollRectPaginator**
- **ScrollRectNumericPaginator**

* not available on platforms with restricted access to file system (like WebGL and UWP).

CHAPTER 3

Widgets Generation

You can generate widgets for your data type with *Context menu / Create / New UI Widgets / Generate Widgets*.



3.1 Requirements

Data type should have at least one public field or public readable property of the supported types.

3.2 Supported types

Text types (string or types convertible to the string):

- `string`
- numeric data types (`int`, `float`, etc)
- any type with overridden `ToString()` method and not derived from `UnityEngine.Object`.

Graphic types:

- `Sprite`
- `Texture2D`

- `Color`
- `Color32`

3.3 Limitations

- **Autocomplete** Requires at least one field or property of the `string` type.
- **Table** Requires at least one field or property of the text type.

3.4 Known Problems

Widget generation does not work with `struct` or `interface` types inside a namespace with some Unity versions due to [bug](#).

Workaround

Change `interface` or `struct` to `class` in the type definition. Then run widgets generation and return type to `interface` or `struct`.

3.5 INotifyPropertyChanged Support

`ObservableList<T>` used by widgets provide support for `INotifyPropertyChanged` interface of the data type, so if property updated and was raised `PropertyChanged` event then widget will be updated.

```
public class ListViewIconsItemDescription : INotifyPropertyChanged
{
    [SerializeField]
    string name;

    public string Name
    {
        get
        {
            return name;
        }

        set
        {
            name = value;
            Changed("Name");
        }
    }

    public event PropertyChangedEventHandler PropertyChanged = (x, y) => { };

    protected void Changed(string propertyName)
    {
        PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
    }

    ...
}
```

This way name of the first item displayed with the widget will be changed:

```
ListView.DataSource[0].Name = "New name";
```

You can disable this behavior with `ObserveItems` property:

```
ListView.DataSource.ObserveItems = false;
// name displayed with the widget will be not changed
ListView.DataSource[0].Name = "New name";
```

3.6 Replacing generated code

Generated code can be freely modified.

Important: Be careful not to overwrite modified scripts if you decide re-run widget generation.

3.6.1 Collections

Widgets to display collections consist of the three classes:

- your custom data type (class, struct or interface)
- Widget class (required because of the generic components not allowed)
- `DefaultItem` class to control tile view

Widget and `DefaultItem` classes created with widget generation for your type and you will need only to modify created `DefaultItem` class if it needs at all.

Functions to modify in the `DefaultItem` class:

- `SetData()` to display passed data. Called when the item displayed or recycled.
- `MovedToCache()` to unload unused resources like *Sprite*. Called when the item is out of sight and not be displayed or recycled (can happen when items list cleared).

For example you can replace default widgets used to display item fields with other widgets.

This example show `Item.Number` field displayed with `Spinner` instead of `Text` and field value update with `Spinner` changes.

Original code:

```
namespace UIWidgets.Examples.WidgetGeneration.Widgets
{
    /// <summary>
    /// ListView component for the DataItem.
    /// </summary>
    public class ListViewComponentDataItem : UIWidgets.ListViewItem,
        UIWidgets.IResizableItem,
        UIWidgets.IViewData<UIWidgets.Examples.WidgetGeneration.DataItem>
    {
        ...

        /// <summary>
        /// The Number.
        /// </summary>
        public UnityEngine.UI.Text Number;
```

(continues on next page)

(continued from previous page)

```

...

/// <summary>
/// Gets the current item.
/// </summary>
public UIWidgets.Examples.WidgetGeneration.DataItem Item
{
    get;
    protected set;
}

/// <summary>
/// Sets component data with specified item.
/// </summary>
/// <param name="item">Item.</param>
public virtual void SetData(UIWidgets.Examples.WidgetGeneration.DataItem item)
{
    Item = item;

    if (Number != null)
    {
        Number.text = Item.Number.ToString();
    }

    ...
}

...
}

```

New code:

```

namespace UIWidgets.Examples.WidgetGeneration.Widgets
{
    /// <summary>
    /// ListView component for the DataItem.
    /// </summary>
    public class ListViewComponentDataItem : UIWidgets.ListViewItem,
        UIWidgets.IResizableItem,
        UIWidgets.IViewData<UIWidgets.Examples.WidgetGeneration.DataItem>
    {
        ...

        /// <summary>
        /// The Number.
        /// </summary>
        public UIWidgets.Spinner Number;

        ...

        /// <summary>
        /// Gets the current item.
        /// </summary>
        public UIWidgets.Examples.WidgetGeneration.DataItem Item
        {
            get;

```

(continues on next page)

(continued from previous page)

```

        protected set;
    }

    /// <summary>
    /// Add callbacks.
    /// </summary>
    protected override void Start()
    {
        base.Start();

        if (Number != null)
        {
            Number.onValueChangedInt.AddListener(UpdateNumber);
        }
    }

    /// <summary>
    /// Update Item.Number when spinner value changed.
    /// </summary>
    void UpdateNumber(int value)
    {
        Item.Number = value;
    }

    /// <summary>
    /// Sets component data with specified item.
    /// </summary>
    /// <param name="item">Item.</param>
    public virtual void SetData(UIWidgets.Examples.WidgetGeneration.DataItem item)
    {
        Item = item;

        if (Number != null)
        {
            Number.Value = Item.Number;
        }

        ...
    }

    /// <summary>
    /// Remove callbacks.
    /// </summary>
    protected override void OnDestroy()
    {
        if (Number != null)
        {
            Number.onValueChangedInt.RemoveListener(UpdateNumber);
        }

        base.OnDestroy();
    }

    ...
}

```

If you need to dynamically change the state of the objects like enabling or disabling them and restore state after item recycled then this can be done with *SetData* function:

```
public virtual void SetData(UIWidgets.Examples.WidgetGeneration.DataItem item)
{
    Item = item;

    // set state after item recycled
    ToggableObject.SetActive(item.IsToggableObjectActive);

    ...
}
```

3.6.2 Autocomplete

You can override Startswith, Contains, and GetStringValue functions to use different field or use other match condition.

This example show Text field replaced with SomeOtherText field and match with EndsWith instead of Contains.

Original code:

```
namespace UIWidgets.Examples.WidgetGeneration.Widgets
{
    /// <summary>
    /// Autocomplete for the DataItem.
    /// </summary>
    public class AutocompleteDataItem : UIWidgets.AutocompleteCustom<UIWidgets.
↳ Examples.WidgetGeneration.DataItem,
        ListViewComponentDataItem, ListViewDataItem>
    {
        ...
        /// <summary>
        /// Returns a value indicating whether Input occurs within specified value.
        /// </summary>
        /// <param name="value">Value.</param>
        /// <returns>true if the Input occurs within value parameter; otherwise, false.
↳ </returns>
        public override bool Contains(UIWidgets.Examples.WidgetGeneration.DataItem_
↳ value)
        {
            if (CaseSensitive)
            {
                return value.Text.Contains(Query);
            }

            return value.Text.ToLower().Contains(Query.ToLower());
        }
    }
}
```

New code:

```
namespace UIWidgets.Examples.WidgetGeneration.Widgets
{
    /// <summary>
    /// Autocomplete for the DataItem.
    /// </summary>
```

(continues on next page)

(continued from previous page)

```
public class AutocompleteDataItem : UIWidgets.AutoCompleteCustom<UIWidgets.  
↳Examples.WidgetGeneration.DataItem,  
    ListViewComponentDataItem, ListViewDataItem>  
{  
    ...  
    /// <summary>  
    /// Returns a value indicating whether Input occurs within specified value.  
    /// </summary>  
    /// <param name="value">Value.</param>  
    /// <returns>true if the Input occurs within value parameter; otherwise, false.  
↳</returns>  
    public override bool Contains(UIWidgets.Examples.WidgetGeneration.DataItem_  
↳value)  
    {  
        if (CaseSensitive)  
        {  
            return value.SomeOtherText.EndsWith(Query);  
        }  
  
        return value.SomeOtherText.ToLower().EndsWith(Query.ToLower());  
    }  
}  
}
```

4.1 Collections

4.1.1 Combobox

Combobox is wrapper for `ListView`, so you should mostly use *`ListView` properties and events*.

Also available `AutocompleteCombobox`, this is `Autocomplete` with Combobox-like behavior.

Options

- `ListView TListViewCustom`
 `ListView` with items.
- `ToggleButton Button`
 Button to show and hide `ListView` on click.
- `Current TComponent`
 Template to display selected items.
- `HideAfterItemToggle bool`
 Hide `ListView` right after item selected or deselected.

Events

- `OnShowListView UnityEvent`
- `OnHideListView UnityEvent`

4.1.2 ListView, TileView and Table

- All collections support virtualization: gameobjects will be created only for visible items.
- Add `Selectable` component to use keyboard and gamepad navigation.

- Different ListView, TileView and Table can display the same list.

List View Type



Fig. 1: ListView with Fixed Size.

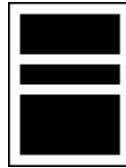


Fig. 2: ListView with Variable Size.

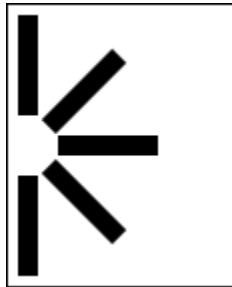


Fig. 3: ListView with Ellipse layout.

ListView.Container settings for the Ellipse type

- **RectTransform.pivot** Defines on which side or corner will be the center point.
- **EasyLayout.Ellipse settings** Width and height usually should be specified, set the same value for the circle.
- **Angle Start** Base rotation for the first item.
- **Angle Step Auto** Should be disabled.
- **Angle Step** Angular distance between items.
- **Fill** Should be `ARC`.
- **Arc Length** Should be **180** if center at the side and **90** if center at the corner.

Settings

- `Interactable bool`
User can interact with ListView.
- `Virtualization bool`
Enable virtualization. If enabled GameObject instantiated only for the visible items; otherwise for the all items.
- `List Type ListViewType`



Fig. 4: TileView with Fixed Size.



Fig. 5: TileView with Variable Size.

Determines how items are displayed.

- `ListViewWithFixedSize`
Works with EasyLayout, Horizontal Layout Group and Vertical Layout Group.
- `ListViewWithVariableSize`
Works with EasyLayout, Horizontal Layout Group and Vertical Layout Group.
- `ListViewEllipse`
Works with EasyLayout.
- `TileViewWithFixedSize`
Works with EasyLayout.
- `TileViewWithVariableSize`
Works with EasyLayout.
- `TileViewStaggered`
Works with EasyLayout.
- `Sort bool`
If enabled items will be sorted with **SortFunc**. Deprecated, replaced with `DataSource.Comparer`.
- `SortFunc Func<IEnumerable<TItem>, IEnumerable<TItem>>`
Not available in the Inspector window Function to sort items. Deprecated, replaced with **DataSource.Comparer**.
- `Data Source ObservableList<TItem>`

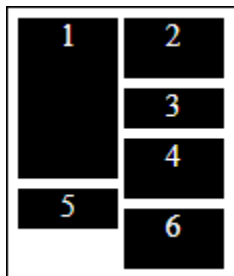


Fig. 6: TileView Staggered.

List of the items.

- **Multiple Select** `bool`

Allow to select multiple items, otherwise only one.

- **SelectedIndex** `int`

Index of the last selected item.

- **SelectedIndices** `List<int>`

Not available in the Inspector window List of the selected items indices.

- **SelectedItem** `TItem`

Not available in the Inspector window Last selected item.

- **SelectedItems** `List<TItem>`

Not available in the Inspector window List of the selected items.

- **Direction** `ListViewDirection`

`ListView` direction.

- `Horizontal`
- `Vertical`

- **DefaultItem** `TComponent`

A prefab used to display item.

- **Container** `Transform`

The container of the instantiated gameobjects used to display items. Should have layout required for the specified `List` Type.

- **ScrollRect** `ScrollRect`

`ScrollRect` used by `ListView`. Required for virtualization support.

- **AllowColoring** `bool`

Change colors of the highlighted and selected items.

- **Colors**

Colors for the text and background elements of the **DefaultItem** instances.

Text and background elements defined with **GraphicsForeground** and **GraphicsBackground** properties of the `TComponent`.

- **Default Color** `Color`
- **Default Background Color** `Color`
- **Highlighted Color** `Color`
- **Highlighted Background Color** `Color`
- **Selected Color** `Color`
- **Selected Background Color** `Color`
- **Disabled Color** `Color`: `multiplicator`, actual color is current color (default, highlighted, selected) * disabled `Color`.

- **Fade Duration** `float`

Time for a smooth color change when the state of an element changes.

- End Scroll Delay `float`

Delay from last scroll event to **OnEndScrolling** event raising.

- Navigation `bool`

Allow to use navigation with keyboard or gamepad.

- Looped List `bool`

Is list looped? First items will be displayed after the last item and scrolling scrolling are infinite. Recommended to disable scrollbar.

- Is Table `bool`

Is ListView will be displayed as a table? Used for correct styles support.

- Set Content Size Fitter `bool`

Changes ContentSizeFitter settings according to the selected direction. Disable if you want to use manual settings.

- Scroll Unscaled Time `bool`

ScrollTo functions will be used unscaled time.

- Scroll Movement `AnimationCurve`

Animation curve for the ScrollTo functions.

- Center The Items `bool`

Display items at the center of the list if items not enough to fill the list.

- Precalculate Item Size `bool`

Precalculate items sizes for List Type with items of variable size.

You can disable this option to increase performance in exchange to less accurate scrolling.

- CanSelect `Func<int, bool>`

The function that determines whether the item with the specified index can be selected.

- CanDeselect `Func<int, bool>`

The function that determines whether the item with the specified index can be deselected.

Events

- OnSelect `UnityEvent<int, ListViewItem>`

Event raised when item selected.

Arguments: index of the selected item and `DefaultItem` instance for the selected item.

- OnDeselect `UnityEvent<int, ListViewItem>`

Event raised when item deselected.

Arguments: index of the deselected item and `DefaultItem` instance for the deselected item.

- OnSelectObject `UnityEvent<int>`

Event raised when item selected.

Arguments: index of the selected item.

- `OnDeselectObject UnityEvent<int>`
Event raised when item deselected.
Arguments: index of the deselected item.
- `OnStartScrolling UnityEvent`
Event raised when scrolling starts.
- `OnEndScrolling UnityEvent`
Event raised when after **End Scroll Delay** from left last scroll event.
- `onSubmit UnityEvent`
Event raised when ListView gameobject has been selected via a “submit” key you specify (default is the return key).
- `onCancel UnityEvent`
Event raised when ListView gameobject has been deselected.
- `onItemSelect UnityEvent`
Event raised when ListView item gameobject has been selected via a “submit” key you specify (default is the return key).
- `onItemCancel UnityEvent`
Event raised when ListView item gameobject has been deselected.
- `OnUpdateView UnityEvent`
Event raised when ListView view was updated.
- `OnFocusIn UnityEvent<BaseEventData>`
Event raised when ListView gameobject received focus.
- `OnFocusOut UnityEvent<BaseEventData>`
Event raised when ListView gameobject lost focus.
- `OnPointerEnterObject UnityEvent<int>`
Event raised when pointer entered ListView item gameobject.
Arguments: index of the item.
- `OnPointerExitObject UnityEvent<int>`
Event raised when pointer exited ListView item gameobject.
Arguments: index of the item.

ListViewComponent Class

Component to display item.

Fields and properties

- `Index int`
Index of the displayed item. Negative if item not displayed or not used by ListView.
- `Owner ListViewBase`
Reference to ListView.

- `GraphicsForeground Graphic[]`
References to the foreground objects like Text.
- `GraphicsBackground Graphic[]`
References to the background objects.

Methods

- `SetData(TItem item)`
Set data.
- `GraphicsColoring(Color foregroundColor, Color backgroundColor, float fadeDuration)`
Called by `ListView` to set colors for the `GraphicsForeground` and `GraphicsBackground`.
- `MovedToCache()`
Called by `ListView` when `GameObject` moved to cache or recycled.
- `StateDefault()`
Called by `ListView` when item in the default state.
- `StateSelected()`
Called by `ListView` when item selected.
- `StateHighlighted()`
Called by `ListView` when item highlighted.

Get items

```
var items = listView.DataSource;
```

Set items

```
var items = new ObservableList<ListViewIconsItemDescription>();
listView.DataSource = items;

var items2 = new List<ListViewIconsItemDescription>();
listView.DataSource = items2.ToObservableList();
```

Display same list with ListView, TileView or Table

```
var items = new ObservableList<ListViewIconsItemDescription>();
listView.DataSource = items;
tileView.DataSource = items;
table.DataSource = items;
```

Get last selected index

```
Debug.Log(listView.SelectedIndex);
```

Get selected indices

```
var indices = listView.SelectedIndices;
Debug.Log(string.Join(", ", indices.ConvertAll(x => x.ToString()).ToArray()));
```

Last selected item

```
Debug.Log(listView.SelectedItem.Name);
```

Get selected items

```
var selected_items = listView.SelectedItems;
Debug.Log(string.Join(", ", selected_items.ConvertAll(x => x.Name).ToArray()));
```

Delete specified item

```
listView.DataSource.Remove(items[0]);
```

Delete item by index

```
listView.DataSource.RemoveAt(0);
```

Clear list

```
listView.DataSource.Clear();
```

Add item

```
var new_item = new ListViewIconsItemDescription()
{
    Icon = sampleIcon,
    Name = "test item",
};
listView.DataSource.Add(new_item);
```

Add items

```
var new_items = new List<ListViewIconsItemDescription>()
{
    new_item,
    new_item,
    new_item,
};
listView.DataSource.AddRange(new_items);
```

Optimization

```
// Use BeginUpdate() and EndUpdate() to keep widget from updating on each change.
// All changes after BeginUpdate() call will be displayed with EndUpdate() call.
var items = listView.DataSource;
items.BeginUpdate();

items.Clear();
items.Add(new_item);
items.Add(new_item);
items.Add(new_item);
items.AddRange(new_items);
```

(continues on next page)

(continued from previous page)

```
items.RemoveAt(0);

// widget will be updated after EndUpdate() call
items.EndUpdate();
```

Replace item

```
listView.DataSource[0] = new ListViewIconsItemDescription()
{
    Name = "new item"
};
```

Sort

```
// Sort by LocalizedName or Name in ascending order
Comparison<ListViewIconsItemDescription> ItemsComparisonAsc = (x, y) => x.Name.
↳CompareTo(y.Name);

// sort by LocalizedName or Name in descending order
Comparison<ListViewIconsItemDescription> ItemsComparisonDesc = (x, y) => -(x.Name).
↳CompareTo(y.Name);

// sort items only once
items.Sort(ItemsComparisonAsc);
```

Enable permanent sort

```
items.Comparison = ItemsComparisonDesc;
```

Important: Items will be always sorted, but if you use `.BeginUpdate()` then items will be re-sorted only after `.EndUpdate()` call.

Disable permanent sort

```
items.Comparison = null;
```

Set selected index

```
listView.SelectedIndex = 1;
```

Or:

```
listView.Select(1);
```

Behavior is different if you enable `MultipleSelect`:

- `listView.SelectedIndex = 1` last selected item will be deselected and specified item will be selected.
- `listView.Select(1)` new item will be added to selected items.

Deselect

```
listView.SelectedIndex = -1;
```

Or:

```
listView.Deselect(1);
```

Scroll to item

```
listView.ScrollToAnimated(index);
```

Example of ListView with Filter

```
namespace UIWidgets.Examples
{
    using System.Collections.Generic;
    using UIWidgets;
    using UnityEngine;
    using UnityEngine.Serialization;

    /// <summary>
    /// Sample ListViewIcons with filter.
    /// </summary>
    public class ListViewIconsWithFilter : ListViewIcons
    {
        [SerializeField]
        List<ListViewIconsItemDescription> listItems = new List
        <ListViewIconsItemDescription>();

        ObservableList<ListViewIconsItemDescription> originalItems;

        /// <summary>
        /// Get or sets items.
        /// </summary>
        public ObservableList<ListViewIconsItemDescription> OriginalItems
        {
            get
            {
                if (originalItems == null)
                {
                    originalItems = new ObservableList<ListViewIconsItemDescription>
                    <ListViewIconsItemDescription>();
                    originalItems.OnChange += Filter;
                }

                return originalItems;
            }

            set
            {
                if (originalItems != null)
                {
                    originalItems.OnChange -= Filter;
                }
            }
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

        originalItems = value;

        if (originalItems != null)
        {
            originalItems.OnChange += Filter;
        }
    }

    /// <summary>
    /// Search string.
    /// </summary>
    protected string Search = string.Empty;

    /// <summary>
    /// Filter data using specified search string.
    /// </summary>
    /// <param name="search">Search string.</param>
    public void Filter(string search)
    {
        Search = search;
        Filter();
    }

    /// <summary>
    /// Copy items from OriginalItems to DataSource if it's match specified string.
    /// </summary>
    protected void Filter()
    {
        DataSource.BeginUpdate();
        DataSource.Clear();

        if (string.IsNullOrEmpty(Search))
        {
            // if search string not specified add all items
            DataSource.AddRange(OriginalItems);
        }
        else
        {
            // else add items with name starts with the specified string
            var finded = OriginalItems.FindAll(x => x.Name.StartsWith(Search));
            DataSource.AddRange(finded);
        }

        DataSource.EndUpdate();
    }

    /// <summary>
    /// Init this instance.
    /// </summary>
    public override void Init()
    {
        base.Init();

        // call Filter() to set initial DataSource
        Filter();
    }

```

(continues on next page)

(continued from previous page)

```
}  
}
```

4.1.3 Paginator

Important: ScrollRect.Content anchors should be setted to top left corner.

How to select paginator

- If you need paginator with fixed items quantity per page use ListViewPaginator.
- If you need paginator where the page size is equal ScrollRect size use ScrollRectPaginator. Add TileViewScrollRectFitter if you also need the whole number of items on one page.
- Use ScrollRectPaginator for any ScrollRect outside ListView, TileView etc.

Settings

- ScrollRect ScrollRect
ScrollRect to work with.
- Default Page RectTransform (optional)
Template GameObject to display inactive pages.
- Active Page RectTransform (optional)
Template GameObject to display active page.
- Prev Page RectTransform (optional)
GameObject, go to the previous page.
- Next Page RectTransform (optional)
GameObject, go to the next page.
- Direction PaginatorDirection
Scroll direction.
 - Auto detect direction by ScrollRect settings and ScrollRect.content size.
 - Horizontal scroll in the horizontal direction
 - Vertical scroll in the vertical direction
- Fast Drag Distance float
Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.
- Fast Drag Time float
Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.
- Forced Position PaginatorPagePosition
Automatically scroll to the nearest page after drag ended if not meet *Fast Drag* condition.
 - None automatical scroll disabled

- `OnStart` automatical scroll enabled; page aligned by the left side of the `ScrollRect` (or the top side if scroll in the vertical direction)
 - `OnCenter` automatical scroll enabled; page aligned by the center side of the `ScrollRect`
 - `OnEnd` automatical scroll enabled; page aligned by the right side of the `ScrollRect` (or the bottom side if scroll in the vertical direction)
- `Animation bool`
Enable animation.
- `Current Page int`
Default page.

Events

- `OnPageSelect UnityEvent<int>`

ScrollRectPaginator Options

- `Page Size Type PageSizeType`
If *Page Size Type* = *Auto* page size is equal to scroll rect size, if *Page Size Type* = *Fixed* will be used *Page Size* value.
 - `Auto`
 - `Fixed`
- `Page Size float`
Size of the page.
- `Page Spacing float`
Space between pages.
- `Movement AnimationCurve`
Animation curve.
- `Unscaled Time bool`
Run animation with unscaled time.

ListViewPaginator Options

- `PerPage int`
Items count on one page, for `TileView` this is rows or columns count per page.

`ListViewPaginator` works with `ListLiew`, `TileView` (in this case `PerPage` is rows or columns count) and `TreeView`. `ListView` animation settings used if animation enabled.

Tile View ScrollRect Fitter

Resize `ScrollRect` to fit the whole number of columns and rows. Add it to gameobject with `TileView` script.

4.1.4 TreeView

- All collections support virtualization: gameobjects will be created only for visible items.
- Add `Selectable` component to use keyboard and gamepad navigation.

Attention: Different `TreeView`'s cannot display same nodes, unlike `ListView`, `TileView`, and `Table`.

Get nodes

```
public TreeView Tree;

ObservableList<TreeNode<TreeViewItem>> nodes;

void Start()
{
    nodes = Tree.Nodes;
}
```

Get selected nodes

```
Tree.SelectedNodes.ForEach(x =>
{
    // do something with selected node
    Debug.Log(x.Item.Name);

    var component = Tree.GetItemComponent(x.Index);

    // not displayed component will be null
    if (component != null)
    {
        component.DoSomething();
    }
});
```

Add listeners

```
void AddListeners()
{
    Tree.NodeSelected.AddListener(ProcessSelectedNode);

    Tree.NodeDeselected.AddListener(ProcessDeselectedNode);
}

void ProcessSelectedNode(TreeNode<TreeViewItem> node)
{
    Debug.Log("selected: " + node.Item.Name);
}

void void ProcessDeselectedNode(TreeNode<TreeViewItem> node)
{
    Debug.Log("deselected: " + node.Item.Name);
}
```

Select node

```
Tree.SelectNode(nodes[1].Nodes[0]);
```

Select node with subnodes

```
Tree.SelectNodeWithSubnodes(nodes[1].Nodes[1]);
```

Deselect node

```
Tree.DeselectNode(nodes[1].Nodes[0]);
```

Deselect node with subnodes

```
Tree.DeselectNodeWithSubnodes(nodes[1].Nodes[1]);
```

Scroll to node

```
Tree.ScrollToAnimated(node);
```

Add node

```
var test_item = new TreeViewItem("added");
var test_node = new TreeNode<TreeViewItem>(test_item);
nodes.Add(test_node);
```

Hide nodes

```
nodes[1].IsVisible = false;
nodes[2].Nodes[1].IsVisible = false;
```

Collapse node

```
nodes[0].Nodes[0].IsExpanded = false;
```

Expand node

```
nodes[0].Nodes[0].IsExpanded = true;
```

Change node name

```
nodes[0].Item.Name = "Node renamed from code";
nodes[0].Nodes[1].Item.Name = "Another node renamed from code";
```

Sort

```
// Compare nodes by Name in ascending order
Comparison<TreeNode<TreeViewItem>> comparisonAsc = (x, y) => x.Item.Name.CompareTo(y.
↪ Item.Name);
```

(continues on next page)

(continued from previous page)

```

// Compare nodes by Name in descending order
Comparison<TreeNode<TreeViewItem>> comparisonDesc = (x, y) => -x.Item.Name.
↪CompareTo(y.Item.Name);

public void SortAsc()
{
    nodes.BeginUpdate();
    ApplyNodesSort(nodes, comparisonAsc);
    nodes.EndUpdate();
}

public void SortDesc()
{
    nodes.BeginUpdate();
    ApplyNodesSort(nodes, comparisonDesc);
    nodes.EndUpdate();
}

void ApplyNodesSort<T>(ObservableList<TreeNode<T>> nodes, Comparison<TreeNode<T>> ↪
↪comparison)
{
    // apply sort for current nodes
    nodes.Sort(comparison);
    // apply sort for child nodes
    nodes.ForEach(node =>
    {
        if (node.Nodes != null)
        {
            ApplyNodesSort(node.Nodes as ObservableList<TreeNode<T>>, comparison);
        }
    });
}

```

Filter nodes

```

public void Filter(string nameContains)
{
    // Maintains performance while items are added/removed/changed
    // by preventing the widgets from drawing
    // until the EndUpdate() method is called.
    nodes.BeginUpdate();

    SampleFilter(nodes, x => x.Name.Contains(nameContains));

    // Apply changes.
    nodes.EndUpdate();
}

bool SampleFilter(IObservableList<TreeNode<TreeViewItem>> nodes, Func<TreeViewItem, ↪
↪bool> filterFunc)
{
    return nodes.Count(x =>
    {
        var have_visible_children = (x.Nodes==null) ? false : SampleFilter(x.Nodes, ↪
↪filterFunc);
        x.IsVisible = have_visible_children || filterFunc(x.Item);
    });
}

```

(continues on next page)

(continued from previous page)

```

        return x.IsVisible;
    }) > 0;
}

```

Reset filter

```

public void ResetFilter()
{
    nodes.BeginUpdate();
    nodes.ForEach(SetVisible);
    nodes.EndUpdate();
}

void SetVisible(TreeNode<TreeViewItem> node)
{
    if (node.Nodes != null)
    {
        node.Nodes.ForEach(SetVisible);
    }

    node.IsVisible = true;
}

```

Clear nodes

```

public void Clear()
{
    nodes.Clear();
}

```

4.2 Containers

4.2.1 Accordion

Options

- Items (DataSource) `ObservableList<AccordionItem>`
 Items.
 AccordionItem fields:
 - `ToggleObject GameObject` Click on this object open or close *ContentObject*.
 - `ContentObject GameObject`
 - `Open bool` Default state of the *ContentObject*.
- Only One Open `bool`
 Only one item can be open at the same time.
- All Items Can Be Closed `bool`
 Allow to close all items; otherwise at least one item always will be opened.
- Animate `bool`
 Animate open and close.

- Animation Duration `float`
Animation Duration.
- Unscaled Time `bool`
Run animation with unscaled time.
- Direction `AccordionDirection`
 - Horizontal
 - Vertical
- Resize Method `ResizeMethods`
 - Size - change width or height of the `ContentObject`.
 - Flexible - change `LayoutElement` `flexibleWidth` or `flexibleHeight` of the `ContentObject`.
- Disable Closed `bool`
Disable closed `ContentObjects`.

Events

- `OnToggleItem` `UnityEvent<AccordionItem>`

Open item

```
Accordion.Open(Accordion.DataSource[0]);
```

Close item

```
Accordion.Close(Accordion.DataSource[0]);
```

Toggle item

```
Accordion.ToggleItem(Accordion.DataSource[0]);
```

Set items

```
Accordion.DataSource = new ObservableList<AccordionItem>()
{
    new AccordionItem()
    {
        ToggleObject = Header1,
        ContentObject = Content1,
        Open = true,
    },
    new AccordionItem()
    {
        ToggleObject = Header2,
        ContentObject = Content2,
        Open = false,
    },
    new AccordionItem()
    {
        ToggleObject = Header3,
```

(continues on next page)

(continued from previous page)

```

    ContentObject = Content3,
    Open = false,
  },
};

```

4.2.2 Tabs

Options

- `Container Transform`
Container for the tabs buttons.
- `DefaultTabButton Button`
Button template for the inactive tabs.
- `ActiveTabButton Button`
Button for the active tab.
- `TabObjects Tab[]`
Tabs array, contains names and references to the tabs gameobjects.
Tab fields:
 - `Name string`
 - `TabObject GameObject`
- `DefaultTabName string`
Name of the tab opened by default.
- `KeepTabsActive bool`
If true does not deactivate hidden tabs.

Events

- `OnTabSelect UnityEvent<int>`
Receive index of the selected tab.

Select tab

```
Tabs.SelectTab(Tabs.TabObjects[0]);
```

Enable tab

```
Tabs.EnableTab(Tabs.TabObjects[0]);
```

Disable tab

```
Tabs.DisableTab(Tabs.TabObjects[0]);
```

4.3 Dialogs

4.3.1 DatePicker

Options

- Calendar DataBase

Reference to the Date widget.

- Date Change Only bool

If true select date only when date changes; otherwise select date on click.

```
namespace UIWidgets.Examples
{
    using System;
    using UIWidgets;
    using UnityEngine;
    using UnityEngine.UI;

    /// <summary>
    /// Test DatePicker.
    /// </summary>
    public class TestDatePicker : MonoBehaviour
    {
        [SerializeField]
        DatePicker PickerTemplate;

        [SerializeField]
        Text Result;

        DateTime currentValue = DateTime.Today;

        /// <summary>
        /// Open picker and log selected value.
        /// </summary>
        public void Test()
        {
            // create picker by template
            var picker = PickerTemplate.Clone();

            // show picker
            picker.Show(currentValue, ValueSelected, Canceled);
        }

        void ValueSelected(DateTime value)
        {
            currentValue = value;
            Debug.Log("value: " + value);
        }

        void Canceled()
        {
            Debug.Log("canceled");
        }

        /// <summary>
        /// Open picker and display selected value.
    }
}
```

(continues on next page)

(continued from previous page)

```

/// </summary>
public void TestShow()
{
    // create picker by template
    var picker = PickerTemplate.Clone();

    // show picker
    picker.Show(currentValue, ShowValueSelected, ShowCanceled);
}

void ShowValueSelected(DateTime value)
{
    currentValue = value;
    Result.text = "Value: " + value;
}

void ShowCanceled()
{
    Result.text = "Canceled";
}
}

```

4.3.2 Dialog

Options

- Buttons Templates `ReadOnlyCollection<Button>`
Templates for the buttons.
- Content Root `RectTransform`
Root gameobject for the content.
- Title Text `Text` (obsolete)
GameObject to display title. Replaced with the *DialogInfo*.
- Content Text `Text` (obsolete)
GameObject to display text. Replaced with the *DialogInfo*.
- Icon `Image` (obsolete)
GameObject to display icon. Replaced with the *DialogInfo*.
- Dialog Info `DialogInfoBase`
Component to display the dialog info.

Show() Method Parameters

All parameters are optional.

- title `string`
Dialog title.
- message `string`
Dialog message.

- `buttons IList<DialogButton>`
Dialog buttons.
DialogButton fields:
 - `Label string`
Button label.
 - `Action Func<int, bool>`
Function to run on button click. Receive button index and return `true` to close dialog; otherwise `false`.
 - `Template Index int`
Index of the button template.
- `focusButton string`
Button with focus by default.
- `position Vector3?`
Dialog position.
- `icon Sprite`
Dialog icon.
- `modal bool`
Modal dialog.
- `modalSprite Sprite`
Background image for the modal dialog.
- `modalColor Color?`
Background color for the modal dialog.
- `canvas Canvas`
Canvas to display dialog. Required if dialog template is prefab.
- `content RectTransform`
Dialog content. Can be used instead of the *message* and *icon*.
- `onClose Action`
Action to run when dialog closed.
- `onCancel Func<int, bool>`
Function to run when dialog canceled. Receive -1 and return `true` if dialog should be closed.

Minimal code

```
// create dialog from template
var dialog = dialogTemplate.Clone();
// show dialog
dialog.Show();
// specify root canvas if dialog cloned from prefab
dialog.Show(canvas: canvas);
```

Advanced

```
// create dialog from template
var dialog = dialogPrefab.Clone();
// show dialog with following parameters
dialog.Show(
    title: "Modal Dialog",
    message: "Simple Modal Dialog.",
    buttons: new DialogButton[]
    {
        new DialogButton(
            "Close", // label
            Dialog.AlwaysClose, // Func<int, bool>, receive button index and return true_
            ↪to close dialog, otherwise false
            0 // button index in ButtonsTemplates
        ),
    },
    focusButton: "Close",
    modal: true,
    modalColor: new Color(0, 0, 0, 0.8f)
);
```

Adding new behaviour

1. Create helper component

```
using UnityEngine;
using UnityEngine.UI;

public class DialogInputHelper : MonoBehaviour {
    [SerializeField]
    public InputField Username;

    [SerializeField]
    public InputField Password;

    // Reset values
    public void Refresh()
    {
        Username.text = "";
        Password.text = "";
    }

    public bool Validate()
    {
        var valid_username = Username.text.Trim().Length > 0;
        var valid_password = Password.text.Length > 0;

        if (!valid_username)
        {
            Username.Select();
        }
        else if (!valid_password)
        {
            Password.Select();
        }

        return valid_username && valid_password;
    }
}
```

(continues on next page)

(continued from previous page)

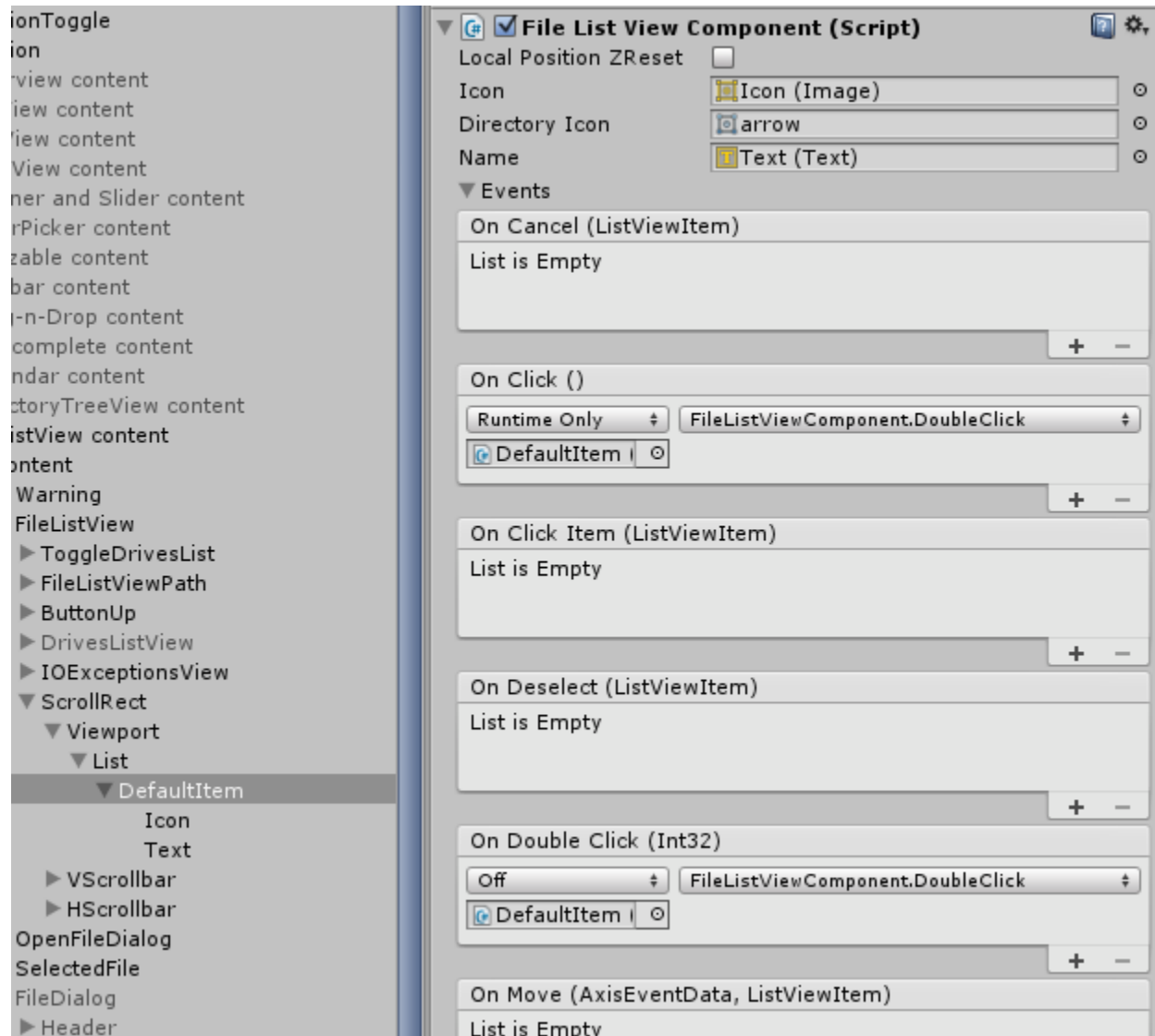
```
}  
}
```

2. Show dialog.

```
public void ShowDialogSignIn()  
{  
    var dialog = dialogSignIn.Clone();  
    var helper = dialog.GetComponent<DialogInputHelper>();  
    helper.Refresh();  
  
    dialog.Show(  
        title: "Sign into your Account",  
        buttons: new DialogButton[]  
        {  
            // on click call SignInNotify  
            new DialogButton("Sign in", (index) => SignInNotify(helper)),  
  
            // on click close dialog  
            new DialogButton("Cancel", Dialog.AlwaysClose),  
        },  
        focusButton: "Sign in",  
        modal: true,  
        modalColor: new Color(0, 0, 0, 0.8f)  
    );  
}  
  
bool SignInNotify(DialogInputHelper helper)  
{  
    //if username or password empty than don't close dialog  
    if (!helper.Validate())  
    {  
        return false;  
    }  
  
    //show notification  
    var message = "Sign in.\nUsername: " + helper.Username.text + "\nPassword:  
→<hidden>";  
    notifySample.Clone().Show(message, customHideDelay: 3f);  
  
    return true;  
}
```

4.3.3 FileDialog

If you want to open directories and select files with a single click instead of the double-click just move `FileListView.DefaultItemDoubleClick` callback to `OnClick` event.



Options

- **File List View FileListView**
FileListView.
- **Confirm Dialog PickerBool**
Dialog to get confirmation if *Request Confirmation If File Exists* enabled.
- **FilenameInput InputField**
Input for the filename.
- **OkButton Button**
Button to close dialog.
- **FileShouldExists bool**
Selected file should exists.
- **Request Confirmation If File Exists bool**

Show *Confirm Dialog* if file exists.

Using *FileDialog*.

```
namespace UIWidgets.Examples
{
    using UIWidgets;
    using UnityEngine;
    using UnityEngine.UI;

    /// <summary>
    /// Test FileDialog.
    /// </summary>
    public class TestFileDialog : MonoBehaviour
    {
        [SerializeField]
        FileDialog PickerTemplate;

        [SerializeField]
        Text Result;

        string currentValue = string.Empty;

        /// <summary>
        /// Show picker and log selected value.
        /// </summary>
        public void Test()
        {
            // create picker by template
            var picker = PickerTemplate.Clone();

            // show picker
            picker.Show(currentValue, ValueSelected, Canceled);
        }

        void ValueSelected(string value)
        {
            currentValue = value;
            Debug.Log("value: " + value);
        }

        void Canceled()
        {
            Debug.Log("canceled");
        }

        /// <summary>
        /// Show picker and display selected value.
        /// </summary>
        public void TestShow()
        {
            // create picker by template
            var picker = PickerTemplate.Clone();

            // show picker
            picker.Show(currentValue, ShowValueSelected, ShowCanceled);
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

void ShowValueSelected(string value)
{
    currentValue = value;
    Result.text = "Value: " + value;
}

void ShowCanceled()
{
    Result.text = "Canceled";
}
}

```

4.3.4 FolderDialog

Options

- Directory Tree View `DirectoryTreeView`
`DirectoryTreeView`.
- Ok Button `Button`
 Button to close dialog.

```

namespace UIWidgets.Examples
{
using UIWidgets;
using UnityEngine;
using UnityEngine.UI;

    /// <summary>
    /// Test FolderDialog.
    /// </summary>
    public class TestFolderDialog : MonoBehaviour
    {
        [SerializeField]
        FolderDialog PickerTemplate;

        [SerializeField]
        Text Result;

        string currentValue = string.Empty;

        /// <summary>
        /// Show picker and log selected value.
        /// </summary>
        public void Test()
        {
            // create picker by template
            var picker = PickerTemplate.Clone();

            // show picker
            picker.Show(currentValue, ValueSelected, Canceled);
        }

        void ValueSelected(string value)

```

(continues on next page)

(continued from previous page)

```
        {
            currentValue = value;
            Debug.Log("value: " + value);
        }

        void Canceled()
        {
            Debug.Log("canceled");
        }

        /// <summary>
        /// Show picker and display selected value.
        /// </summary>
        public void TestShow()
        {
            // create picker by template
            var picker = PickerTemplate.Clone();

            // show picker
            picker.Show(currentValue, ShowValueSelected, ShowCanceled);
        }

        void ShowValueSelected(string value)
        {
            currentValue = value;
            Result.text = "Value: " + value;
        }

        void ShowCanceled()
        {
            Result.text = "Canceled";
        }
    }
}
```

4.3.5 Notifications

Important: If you want to display more than one notification at the same time, then *notification container* should have *layout group* component like `EasyLayout`. Start positions of notifications are determined with `Group Position`.

Options

- `Hide Button` `Button`
Button to close notification.
- `Text` `Text` (obsolete)
`GameObject` to display the notification text. Replaced with `NotifyInfo`.
- `Hide Delay` `float`
Delay before notification automatically hidden.
- `Unscaled Time` `bool`
Delay with unscaled time.

- **SlideUpOnHide** `bool`
Start slide up animations after hide current notification. Turn it off if its managed with `HideAnimation`.
- **NotifyInfo** `NotifyInfoBase`
Component to display the notification text.

Minimal code

```
// get notification instance by template name (name of existing GameObject with Notify component).  
var notification = notifySample.Clone();  
// show notification  
notification.Show();
```

Advanced

```
var notification = notifySample.Clone();  
// show notification  
notification.Show(  
    // Show notification with following text  
    message: "Simple Notification.",  
    // Hide it after 4.5 seconds  
    customHideDelay = 4.5f,  
    // Run specified animation on hide  
    hideAnimation = Notify.AnimationCollapseVertical,  
    // without SlideUpOnHide  
    slideUpOnHide = false  
);
```

Default Hide Animations

Note: Hide Animation is coroutine that accepts `Notify` instance and play hide animation for this instance. You can specify any custom coroutine.

- **AnimationRotateHorizontal** Rotate notification on X axis.
- **AnimationRotateVertical** Rotate notification on Y axis.
- **AnimationCollapseHorizontal** Resize width of the notification.
- **AnimationCollapseVertical** Resize height of the notification.
- **AnimationSlideRight** Slide notification on right.
- **AnimationSlideLeft** Slide notification on left.
- **AnimationSlideUp** Slide notification on up.
- **AnimationSlideDown** Slide notification on down.

Configurable Hide Animations

- **AnimationRotateBase**
Arguments:
 - **Notify** `notify` Notification instance.

- **bool isHorizontal** Rotate in horizontal or vertical direction.
- **float timeLength** Length of animations in seconds.

- **AnimationCollapseBase**

Arguments:

- **Notify notify** Notification instance.
- **bool isHorizontal** Resize in horizontal or vertical direction.
- **float speed** Resize speed in points per second.

- **AnimationSlideBase**

Arguments:

- **Notify notify** Notification instance.
- **bool isHorizontal** Slide in horizontal or vertical direction.
- **float direction** Slide direction, -1f for left/down, +1f for right/up.
- **float speed** Slide speed in points per second.
- **bool animateReplacement** Animate other notifications.

```
NotifyTemplate.Clone().Show(  
    "Notification message.",  
    customHideDelay: 3f,  
    hideAnimation: x => Notify.AnimationSlideBase(x, true, -1f, 200f, true)  
);
```

4.3.6 Pickers

Base class for the custom pickers.

```
namespace UIWidgets.Examples  
{  
    using System.Linq;  
    using UIWidgets;  
    using UnityEngine;  
  
    public class PickerIntTest : MonoBehaviour  
    {  
        [SerializeField]  
        PickerInt PickerTemplate;  
  
        int currentValue = 0;  
  
        public void Test()  
        {  
            // create picker by template  
            var picker = PickerTemplate.Clone();  
  
            // set values from template  
            picker.ListView.DataSource = PickerTemplate.ListView.DataSource.  
↪ ToObservableList();  
            // or set new values  
            // picker.ListView.DataSource = Enumerable.Range(1, 100).  
↪ ToObservableList();  
        }  
    }  
}
```

(continues on next page)

(continued from previous page)

```

        // show picker with callbacks
        picker.Show(currentValue, ValueSelected, Canceled);
    }

    // will be called if value selected
    void ValueSelected(int value)
    {
        currentValue = value;
        Debug.Log("value: " + value);
    }

    // will be called if cancel button pressed
    void Canceled()
    {
        Debug.Log("canceled");
    }
}

```

4.3.7 Popup

Options

- Title Text Text (obsolete)
GameObject to display title. Replaced with the *DialogInfo*.
- Content Text Text (obsolete)
GameObject to display text. Replaced with the *DialogInfo*.
- Icon Image (obsolete)
GameObject to display icon. Replaced with the *DialogInfo*.
- Info DialogInfoBase
Component to display the popup info.

Minimal code

```

// create popup from template
var popup = popupTemplate.Clone();
// show popup
popup.Show();
// specify root canvas if popup cloned from prefab
dialog.Show(canvas: canvas);

```

Advanced

```

// create popup from template
var popup = popupTemplate.Clone();
// show popup with following parameters
popup.Show(
    title: "Modal popup",
    message: "Simple Modal popup.",

```

(continues on next page)

(continued from previous page)

```
modal: true,  
modalColor: new Color(0, 0, 0, 0.8f)  
);
```

4.4 Input

4.4.1 Autocomplete

Options

- Input Field `InputField`
Input field.
- Target List View `TListView`
ListView to display available values.
- Display List View `TListView`
Selected value will be added to this ListView.
- Allow Duplicate `bool`
TargetListView can have duplicated items.
- Data Source `List<TValue>`
List of the all values.
- Filter `AutocompleteFilter`
Filter settings.
 - `Startswith`
Value should starts with the specified input.
 - `Contains`
Value should contains with the specified input.
- Case Sensitive `bool`
Is filter case sensitive?
- Delimiter Chars `char[]`
Delimiter chars to split input to the words.
- Input Type `AutocompleteInput`
Filter with the current word or the whole input.
 - `Word`
 - `AllInput`
- Result `AutocompleteResult`
What to do with input after value selected.
 - `Append`
 - `Replace`

- **Min Length** `int`
Minimal length of the input to start search.
- **Search Delay** `float`
The delay in seconds between when a keystroke occurs and when a search is performed.
- **Unscaled Time** `bool`
Delay with unscaled time.

Events

- **OnOptionSelected** `UnityEvent`
- **OnOptionSelectedItem** `UnityEvent<TValue>`

```
namespace UIWidgets.Examples
{
    using UIWidgets;
    using UnityEngine;

    public class AutocompleteIconsText : MonoBehaviour
    {
        [SerializeField]
        public AutocompleteIcons Autocomplete;

        [SerializeField]
        ListViewIconsItemDescription item;

        void Start()
        {
            Autocomplete.OnOptionSelectedItem.AddListener(SetItem);
        }

        void OnDestroy()
        {
            Autocomplete.OnOptionSelectedItem.RemoveListener(SetItem);
        }

        void SetItem(ListViewIconsItemDescription newItem)
        {
            item = newItem;
        }
    }
}
```

4.4.2 Calendar

Note: `DateTime.TimeOfDay` is not setted or changed by Calendar.

Options

- **Interactable** `bool`
Is interactable?
- **Date** `DateTime`

Current date.

- `Date Min DateTime`

Minimal date.

- `Date Max DateTime`

Maximum date.

- `First Day Of Week DayOfWeek`

First day of the week.

- `Container RectTransform`

Container for the dates.

- `Calendar Date Template CalendarDateBase`

Template for the date.

- `HeaderContainer RectTransform`

Container for the day of weeks.

- `Calendar Day Of Week Template CalendarDayOfWeekBase`

Template for the day of week.

- `Date Text Text`

Text to display the current date.

- `Month Text Text`

Text to display the current month.

Events

- `OnDateChanged UnityEvent<DateTime>`
- `OnClick UnityEvent<DateTime>`

```
namespace UIWidgets.Examples
{
    using UnityEngine;

    /// <summary>
    /// Test Calendar.
    /// </summary>
    public class TestCalendar : MonoBehaviour
    {
        /// <summary>
        /// Calendar.
        /// </summary>
        [SerializeField]
        protected UIWidgets.Calendar Calendar;

        /// <summary>
        /// Start this instance.
        /// </summary>
        protected virtual void Start()
        {
            Calendar.OnDateChanged.AddListener(ProcessDate);
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    // change first day of the week
    Calendar.FirstDayOfWeek = System.DayOfWeek.Sunday;

    // change culture
    Calendar.Culture = new System.Globalization.CultureInfo("en-US");

    // change calendar
    SetCalendar(new System.Globalization.JapaneseCalendar());
}

void ProcessDate(System.DateTime dt)
{
    Debug.Log(dt);
}

void SetCalendar(System.Globalization.Calendar calendar)
{
    Calendar.Culture.DateTimeFormat.Calendar = calendar;
    Calendar.UpdateCalendar();
}
}

```

4.4.3 Centered Slider

The differences from a default slider:

- zero at center
- positive and negative parts have different scales.

Options

- Value `int`
Current value.
- Use Value Limits `bool`
Value cannot exceed the specified limits.
- Limit Min `int`
Minimal limit of the value.
- Limit Max `int`
Maximum limit of the value.
- Value Min `int`
Minimal value.
- ValueMax `int`
Maximal value.
- Step `int`
Value step.
- Whole Number Of Steps `bool`

Whole number of steps for the value.

- `Handle RangeSliderHandle`

Handle to drag.

- `UsableRangeRect RectTransform`

Usable range.

- `FillRect RectTransform`

GameObject to display fill (line from center to the current value).

Events

- `OnValuesChange UnityEvent<int>`
- `OnChange UnityEvent`

Set value

```
slider.Value = 150;
```

Set display limits

```
slider.LimitMin = -500;  
slider.LimitMax = 250;
```

Set value limits

```
slider.UseValueLimits = true;  
slider.ValueMin = -100;  
slider.ValueMax = 200;
```

4.4.4 ColorPicker

Set color

```
ColorPicker.Color = Color.cyan;
```

Get color

```
Debug.Log (ColorPicker.Color);
```

Add listener

```
void Start()  
{  
    ColorPicker.OnChange.AddListener (ColorChanged);  
}  
  
void ColorChanged (Color32 color)  
{  
    Debug.Log ("selected color: " + Color);  
}
```


4.4.5 DateScroller, DateTimeScroller, TimeScroller

Note: `DateTime.TimeOfDay` is not setted or changed by `DateScroller`, but changed by `DateTimeScroller`.

Note: [DateTime Formats Strings](#)

DateScroller Settings

- `Interactable bool`
User can interact with `ListView`.
- `Current Date As Default bool`
 - `Default Date DateTime (string in Inspector window)`
- `Default Date Min DateTime (string in Inspector window)`
Minimal selectable date.
- `Default Date Max DateTime (string in Inspector window)`
Maximum selectable date.
- `Format string`
Format to parse **Default Date**, **Default Date Min**, and **Default Date Max**.
- `Independent scroll bool`
If enabled any time period changes will not change other time periods.
- `Years bool`
Display years scroller.
 - `Years Scroller Scroller`
 - `Years Step int`
 - `Years Format string`
- `Months bool`
Display months scroller.
 - `Months Scroller Scroller`
 - `Months Step int`
 - `Months Format string`
- `Days bool`
Display days scroller.
 - `Days Scroller Scroller`
 - `Days Step int`
 - `Days Format string`
- `Events`
 - `OnDateChanged UnityEvent<DateTime>`

Event raised when date changed.

Arguments: selected datetime.

- `OnClick` `UnityEvent<DateTime>`

Event raised when date setted or changed.

Arguments: selected datetime.

DateTimeScroller Settings

Same settings as `DateScroller` with addition:

- `Hours` `bool`

Display hours scroller.

- `Hours Scroller` `Scroller`
- `Hours Step` `int`
- `Hours Format` `string`
Used if **AMPM** disabled.
- `Hours AMPM Format` `string`
Used if **AMPM** enabled.

- `Minutes` `bool`

Display minutes scroller.

- `Minutes Scroller` `Scroller`
- `Minutes Step` `int`
- `Minutes Format` `string`

- `Seconds` `bool`

Display seconds scroller.

- `Seconds Scroller` `Scroller`
- `Seconds Step` `int`
- `Seconds Format` `string`

- `AMPM` `bool`

Display AMPM scroller.

- `AMPM Scroller` `Scroller`
- `AMPM Format` `string`

TimeScroller Settings

- `Interactable` `bool`

User can interact with `ListView`.

- `Current Time As Default` `bool`

- `Time Text` `TimeSpan` (string in Inspector window)

- `Default Time Min` `TimeSpan` (string in Inspector window)

Minimal selectable time.

- Default Time Max TimeSpan (string in Inspector window)

Maximum selectable time.

- Format string

Format to parse **Time Text**, **Default Time Min**, and **Default Time Max**.

- Independent scroll bool

If enabled any time period changes will not change other time periods.

- Hours bool

Display hours scroller.

- Hours Scroller Scroller
- Hours Step int

- Minutes bool

Display minutes scroller.

- Minutes Scroller Scroller
- Minutes Step int

- Seconds bool

Display seconds scroller.

- Seconds Scroller Scroller
- Seconds Step int

- AMPM bool

Display AMPM scroller.

- AMPM Scroller Scroller

- Events

- OnTimeChanged UnityEvent<TimeSpan>

Event raised when time changed.

Arguments: selected time.

```
namespace UIWidgets.Examples
{
    using UnityEngine;

    /// <summary>
    /// Test DateScroller.
    /// </summary>
    public class TestDateScroller : MonoBehaviour
    {
        /// <summary>
        /// DateScroller.
        /// </summary>
        [SerializeField]
        protected UIWidgets.DateBase DateScroller;
    }
}
```

(continues on next page)

(continued from previous page)

```

    /// <summary>
    /// Start this instance.
    /// </summary>
    protected virtual void Start()
    {
        DateScroller.OnDateChanged.AddListener(ProcessDate);

        // change culture
        DateScroller.Culture = new System.Globalization.CultureInfo("en-US");

        // change calendar
        DateScroller.Culture = new System.Globalization.CultureInfo("ja-JP");
        DateScroller.Culture.DateTimeFormat.Calendar = new System.Globalization.
↪ JapaneseCalendar();
    }

    void ProcessDate(System.DateTime dt)
    {
        Debug.Log(dt);
    }
}

```

4.4.6 RangeSlider

Slider with two handles for minimum and maximum. Has versions for the `int` and `float` types.

Options

- Type `RangeSliderType`
Type of the slider.
 - `AllowHandleOverlay`
Handles can intersects. Value scale is constant.
 - `DisableHandleOverlay`
Handles can not intersects. Value scale is variable.
- Value Min `int/float`
Minimal value.
- Value Max `int/float`
Maximal value.
- Step `int/float`
Step of the value.
- Limit Min `int/float`
Value cannot be less that this.
- Limit Max `int/float`
Value cannot be more that this.
- Handle Min `RangeSliderHandle`

Handle to change the minimal value.

- `Handle Max RangeSliderHandle`

Handle to change the maximal value.

- `UsableRangeRect RectTransform`

Usable range.

- `FillRect RectTransform`

GameObject to display fill (line from minimal value to the maximal value).

- `Whole Number Of Steps bool`

Whole number of steps for the value.

Events

- `OnValuesChange UnityEvent<int, int>/UnityEvent<float, float>`
- `OnChange UnityEvent`

Set values

```
slider.ValueMin = 10;
slider.ValueMax = 80;
```

Set step

```
slider.Step = 2;
```

Set limits

```
slider.LimitMin = 0;
slider.LimitMax = 100;
```

Add listener

```
void Start()
{
    slider.OnValuesChange.AddListener(SliderChanged);
}

void SliderChanged(int min, int max)
{
    if (slider.WholeNumberOfSteps)
    {
        Debug.Log(string.Format("Range: {0:000} - {1:000}; Step: {2}", min, max, slider.
↪Step));
    }
    else
    {
        Debug.Log(string.Format("Range: {0:000} - {1:000}", min, max));
    }
}
```

4.4.7 Spinner

Has versions for the `int` and `float` types.

Options

- `Value Min int/float`
Minimal value.
- `Value Max int/float`
Maximal value.
- `Step int/float`
Step of the value.
- `SpinnerValue int/float`
Current value.
- `Validation SpinnerValidation`
Validate value on specified event.
 - `OnKeyDown`
 - `OnEndInput`
- `AllowHold bool`
Change value on button hold.
- `HoldStartDelay float`
Delay of hold in seconds to start change value.
- `HoldChangeDelay float`
Delay of hold in seconds between each change value.
- `Plus Button ButtonAdvanced`
Button to increase value.
- `Minus Button ButtonAdvanced`
Button to decrease value.

Events

- `onPlusClick UnityEvent`
- `onMinusClick UnityEvent`

Spinner Events

- `onValueChangeInt UnityEvent<int>`
- `onEndEditInt UnityEvent<int>`

SpinnerFloat Options

- Format `string`
Value format.
- Decimal Separators `char[]`
Decimal separators.
- Number Style `NumberStyles`
Style of the number.

SpinnerFloat Events

- `onValueChangedFloat` `UnityEvent<float>`
- `onEndEditFloat` `UnityEvent<float>`

Set maximum

```
spinner.Max = 100;
```

Set minimun

```
spinner.Min = 0;
```

Set value

```
spinner.Value = 10;
```

Set step

```
spinner.Step = 1;
```

Get value

```
Debug.Log(spinner.Value);
```

4.5 Misc

4.5.1 ProgressbarDeterminate

Progress animation is based on *FillMethod* of the *Full Bar Mask* and *Full Bar Border*.

Options

- Max `int`
Maximum value of the progress.
- Value `int`
Current value of the progress.
- Full Bar Mask `Image`

Image to display progress. Image type should be `Filled`.

- `Full Bar Border Image`

Border image to display progress. Image type should be `Filled`.

- `Text Type ProgressbarTextTypes`

How to progress should be displayed as text.

- `None`

Does not display text.

- `Percent`

Show progress as percent like *15%*

- `Range`

Show progress as text like *15 / 100*

- `Speed float`

Animation speed in the seconds.

- `Speed Type ProgressbarSpeedType`

Specifies how speed should be interpreted.

- `TimeToValueChangedOnOne`

Speed is time to change progress on 1.

- `ConstantSpeed`

Speed is time to change progress from 0 to Max. If value changed from 0 to Max/2 than animation takes speed/2 seconds.

- `ConstantTime`

Speed is time to change progress from current value to new value.

- `Unscaled Time bool`

Run animation with unscaled time.

- `Text Func Func<ProgressbarDeterminateBase, string>`

Custom function to convert progress value to the text. Overwrites `Text Type` settings.

- `Background Image`

Background image.

- `Empty Bar Image`

Empty bar image.

- `Full Bar Image Image`

Full bar Image.

- `Empty Bar Text Text`

Text to display progress.

- `Full Bar Text Text`

Text to display progress.

Set value

```
Progressbar.Animate(value);
```

Stop animation

```
Progressbar.Stop();
```

4.5.2 ProgressbarIndeterminate

Options

- **Direction** `ProgressbarDirection`
Animation direction.
 - Horizontal
 - Vertical
- **Bar** `RawImage`
Image to animate. Use texture type ``texture` and set *Wrap Mode* to repeat.
- **Border** `Image`
Border image.
- **Mask** `Image`
Mask.
- **Speed** `float`
Animation speed.
- **Unscaled Time** `bool`
Run animation with unscaled time.

Start animation

```
Progressbar.Animate();
```

Stop animation

```
Progressbar.Stop();
```


5.1 Bring to Front

Use it to bring to front selected `GameObject`. Commonly used with `Dialog` or `Draggable` objects.

5.1.1 Options

- `With Parents bool`

Bring to front `GameObject` with parents `GameObjects`.

5.2 Drag and Drop components

5.2.1 Drag-and-Drop Support for the Collections

Different drag-and-drop components used with different widgets. Default widgets already have drag-and-drop components. For the generated widgets drag-and-drop components create automatically. Default Drag components usually attached to `DefaultItem`. Default Drop components usually attached to widgets and `TreeView.DefaultItem`.

Drag will be cancelled with `OnCancel` event from `EventSystem` (for example by pressing *Esc*).

You can remove drag-and-drop components from the widgets to disable drag-and-drop functionality.

5.2.2 Options

- `ListView TListView` (optional)

`ListView`.

- `DragInfo TComponent` (optional)

Component to display the dragged data.

- `DragInfo Offset Vector3`

Offset from the cursor position for the `DragInfo`.

- Allow Drop Cursor Texture2D (optional)
Cursor when drop allowed.
- Allow Drop Cursor Hot Spot Vector2
Cursor hot spot.
- Denied Drop Cursor Texture2D (optional)
Cursor when drop denied.
- Denied Drop Cursor Hot Spot Vector2
Cursor hot spot.
- Default Cursor Texture Texture2D (optional)
Default cursor.
- Default Drop Cursor Hot Spot Vector2
Cursor hot spot.
- Handle DragSupportHandle (optional)
Handle.
- Delete After Drop bool
Delete item from ListView after drop.

5.2.3 Custom Drag Support

You can add own drag support with component inherited from `DragSupport<TItem>` implementation.

Methods

- `InitDrag(PointerEventData eventData)` (required): set Data value to drag
- `Dropped(bool success)` (optional): what to do after the drop happened or canceled

Here is basic example of the drag support for the InputField:

```
namespace UIWidgets.Examples
{
    using UnityEngine;
    using UnityEngine.EventSystems;
    using UnityEngine.UI;

    /// <summary>
    /// Drag support for the InputField.
    /// </summary>
    [RequireComponent(typeof(InputField))]
    public class InputFieldDragSupportBase : DragSupport<string>
    {
        /// <summary>
        /// Set Data, which will be passed to the Drop component.
        /// </summary>
        /// <param name="eventData">Current event data.</param>
        protected override void InitDrag(PointerEventData eventData)
        {
            Data = GetComponent<InputField>().text;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

    }
}

```

This example show how to display draggable data:

```

namespace UIWidgets
{
    using UnityEngine;
    using UnityEngine.EventSystems;
    using UnityEngine.Serialization;
    using UnityEngine.UI;

    /// <summary>
    /// Drag support for the InputField.
    /// </summary>
    [RequireComponent(typeof(InputField))]
    public class InputFieldDragSupport : DragSupport<string>
    {
        /// <summary>
        /// Set Data, which will be passed to Drop component.
        /// </summary>
        /// <param name="eventData">Current event data.</param>
        protected override void InitDrag(PointerEventData eventData)
        {
            Data = GetComponent<InputField>().text;

            ShowDragInfo();
        }

        /// <summary>
        /// Called after the drop completed.
        /// </summary>
        /// <param name="success">true if Drop component received data; otherwise,
        ↪ false.</param>
        public override void Dropped(bool success)
        {
            HideDragInfo();

            base.Dropped(success);
        }

        /// <summary>
        /// Component to display draggable info.
        /// </summary>
        [SerializeField]
        public GameObject DragInfo;

        /// <summary>
        /// DragInfo offset.
        /// </summary>
        [SerializeField]
        public Vector3 DragInfoOffset = new Vector3(-5, 5, 0);

        /// <summary>
        /// Start this instance.
        /// </summary>
        protected virtual void Start()

```

(continues on next page)

(continued from previous page)

```

{
    if (DragInfo != null)
    {
        DragInfo.SetActive(false);
    }
}

/// <summary>
/// Shows the drag info.
/// </summary>
protected virtual void ShowDragInfo()
{
    if (DragInfo == null)
    {
        return;
    }

    DragInfo.transform.SetParent(DragPoint, false);
    DragInfo.transform.localPosition = DragInfoOffset;

    DragInfo.SetActive(true);

    DragInfo.GetComponentInChildren<Text>().text = Data;
}

/// <summary>
/// Hides the drag info.
/// </summary>
protected virtual void HideDragInfo()
{
    if (DragInfo == null)
    {
        return;
    }

    DragInfo.SetActive(false);
}
}
}

```

5.2.4 Custom Drop Support

You can add own the drop support with `IDropSupport<TItem>>` implementation.

Methods

- `CanReceiveDrop(TItem data, PointerEventData eventData)`: determine if the drop can be accepted or not, can used to display the drop preview.
- `Drop(TItem data, PointerEventData eventData)`: process the dropped data.
- `DropCanceled(TItem data, PointerEventData eventData)`: process the cancelled drop, can used to hide the drop preview or the drop indicator.

Here is example code shows how to add `TreeNode<TreeViewItem>` and `string` drop support to the *InputField*, after drop *InputField* value would be set to the dropped node name or the dropped string.

`CanReceiveDrop` function allow to accept only nodes with names ends with */*.

```

namespace UIWidgets.Examples
{
    using UnityEngine;
    using UnityEngine.UI;
    using UnityEngine.EventSystems;

    /// <summary>
    /// TreeNode drop support for the InputField.
    /// </summary>
    [RequireComponent(typeof(InputField))]
    public class InputFieldDropSupport : MonoBehaviour, IDropSupport<TreeNode
    <TreeViewItem>>, IDropSupport<string>
    {
        /// <summary>
        /// InputField.text value before drop.
        /// Can be used to swap content with drag source.
        /// </summary>
        public string OriginalData;

        #region IDropSupport<string>

        /// <summary>
        /// Handle dropped data.
        /// </summary>
        /// <param name="data">Data.</param>
        /// <param name="eventData">Event data.</param>
        public void Drop(string data, PointerEventData eventData)
        {
            var input = GetComponent<InputField>();
            OriginalData = input.text;
            input.text = data;
        }

        /// <summary>
        /// Determines whether this instance can receive drop with the specified data_
    <and eventData.
    <and eventData.
        /// </summary>
        /// <returns>true if this instance can receive drop with the specified data and_
    <eventData; otherwise, false.</returns>
        /// <param name="data">Data.</param>
        /// <param name="eventData">Event data.</param>
        public bool CanReceiveDrop(string data, PointerEventData eventData)
        {
            return true;
        }

        /// <summary>
        /// Handle canceled drop.
        /// </summary>
        /// <param name="data">Data.</param>
        /// <param name="eventData">Event data.</param>
        public void DropCanceled(string data, PointerEventData eventData)
        {
        }

        #endregion
    }
}

```

(continues on next page)

(continued from previous page)

```

#region IDropSupport<TreeNode<TreeViewItem>>

/// <summary>
/// Handle dropped data.
/// </summary>
/// <param name="data">Data.</param>
/// <param name="eventData">Event data.</param>
public void Drop(TreeNode<TreeViewItem> data, PointerEventData eventData)
{
    var input = GetComponent<InputField>();
    OriginalData = input.text;
    input.text = data.Item.Name;
}

/// <summary>
/// Determines whether this instance can receive drop with the specified data_
↪and eventData.
/// </summary>
/// <returns>true if this instance can receive drop with the specified data and_
↪eventData; otherwise, false.</returns>
/// <param name="data">Data.</param>
/// <param name="eventData">Event data.</param>
public bool CanReceiveDrop(TreeNode<TreeViewItem> data, PointerEventData_
↪eventData)
{
    return data.Item.Name.EndsWith("1");
}

/// <summary>
/// Handle canceled drop.
/// </summary>
/// <param name="data">Data.</param>
/// <param name="eventData">Event data.</param>
public void DropCanceled(TreeNode<TreeViewItem> data, PointerEventData_
↪eventData)
{
}

#endregion
}

```

5.2.5 Swapping content between Drag and Drop components

Original content of the drop component saved to `IDropSupport<T>.OriginalData` field. And content should be swapped in the `DragSupport<T>.OnEndDrag()` function

```

namespace UIWidgets.Examples
{
    using UnityEngine;
    using UnityEngine.EventSystems;
    using UnityEngine.UI;

    /// <summary>
    /// Drag support with content swap for the InputField.
    /// </summary>

```

(continues on next page)

(continued from previous page)

```

[RequireComponent(typeof(InputField))]
public class InputFieldDragSwapSupport : InputFieldDragSupport
{
    /// <summary>
    /// Called by a BaseInputModule when a drag is ended.
    /// </summary>
    /// <param name="eventData">Current event data.</param>
    public override void OnEndDrag(PointerEventData eventData)
    {
        if (!IsDragged)
        {
            return;
        }

        var target = FindTarget(eventData);
        if (target != null)
        {
            target.Drop(Data, eventData);
            Dropped(true);

            // replace dragged text with drop target text
            GetComponent<InputField>().text = (target as InputFieldDropSupport).
↪OriginalData;
        }
        else
        {
            Dropped(false);
        }

        IsDragged = false;
        Cursor.SetCursor(DefaultCursorTexture, DefaultCursorHotSpot, Compatibility.
↪GetCursorMode());
    }
}

```

5.3 Draggable

Dragging gameobject.

5.3.1 Options

- Handle GameObject (optional)
 GameObject used to drag current GameObject.
- Horizontal bool
 Allow horizontal drag movement.
- Vertical bool
 Allow vertical drag movement.
- Restriction DraggableRestriction:
 - None: no restriction.
 - Strict: does not allow drag outside parent.

- `After Drag`: does not allow drag outside parent, applied when drag ended.
- `Curve AnimationCurve`
Animation curve used to animate applied **After Drag** restriction.
- `UnscaledTime bool`
Run animation with unscaled time.

5.4 EasyLayout

EasyLayout provides different layouts that not available with default layout groups.

5.4.1 Options

- `Main Axis Axis`
Determine how elements will be placed (at horizontal or vertical direction first).
- `Layout Type LayoutTypes`
 - `Compact`: Compactly places the elements.
 - `Grid`: Places elements in the grid. Cell size is not fixed and depend on elements sizes in the same row and column.
 - `Flex`: Places elements like CSS flexbox layout.
 - `Staggered`: Places elements one-by-one to the shortest column or row depending on the main axis.
 - `Ellipse`: Places elements one-by-one on the border of the ellipse or the circle starting from `Angle Start` and `Angle Step` distance between items.
- `Group Position Anchors`
Only for the `Compact` and `Grid` layouts.
Combination of horizontal (`Left`, `Center`, `Right`) and vertical (`Upper`, `Middle`, `Lower`) positions.
Elements combine to the group, this option specifies group position relative to the parent.
- `Row Align HorizontalAligns`
Only for the `Compact` layout.
Element position in the row (`Left`, `Center`, `Right`).
- `Inner Align InnerAligns`
Only for the `Compact` layout.
Column position relative to the group (`Top`, `Middle`, `Bottom`).
- `Compact Constraint CompactConstraints`
Only for the `Compact` layout.
 - `Flexible`: Rows and columns count depends on the parent size.
 - `Max Column Count`
 - `Max Row Count`
- `Compact Constraint Count int`

Only for the `Compact` layout.

Max count of the rows or columns for the `Compact Constraint` option.

- `Cell Align Anchors`

Only for the `Grid` layout.

Elements position relative to the cell size. Same as `Group Position`.

- `Grid Constraint GridConstraints`

Only for the `Grid` layout.

- `Flexible`: Rows and columns count depends on the parent size.
- `Fixed Column Count`
- `Fixed Row Count`

- `Grid Constraint Count int`

Only for the `Grid` layout.

Count of the rows or columns for the `Grid Constraint` option.

- `Flex Setting EasyLayoutFlexSettings`

Only for the `Flex` layout.

- `Wrap bool`

If disabled elements will all placed onto one line (row or column).
- `Justify Content EasyLayoutFlexSettings.Content`

Alignment along the main axis. Also distribute extra free space on the main axis.

 - * `Start`: elements placed at the start of the line.
 - * `Center`: elements placed at the center of the line.
 - * `End`: elements placed at the end of the line.
 - * `Space Between`: first element at the start of the line, last element at the end of the line, other elements placed between them with evenly spacing.
 - * `Space Around`: first and last elements are placed with $1n$ space from the edges, other elements placed with $2n$ space between them.
 - * `Space Evenly`: elements are placed so that the spacing between any two element and the space to the edges is equal.
- `Align Content EasyLayoutFlexSettings.Content`

Alignment of the lines (columns or rows) along the cross axis. Also distribute extra free space on the cross axis.

 - * `Start`: lines placed to the start of the parent.
 - * `Center`: lines placed to the center of the parent.
 - * `End`: lines placed to the end of the parent.
 - * `Space Between`: first line to the start of the parent, last line to the end of the parent, other lines placed between them with evenly spacing.
 - * `Space Around`: first and last lines are placed with $1n$ space from the edges, other lines placed with $2n$ space between them.

- * `Space Evenly`: line are placed so that the spacing between any two lines and the space to the edges is equal.
- `Align Items EasyLayoutFlexSettings.Items`
 - Define how elements are placed out along the cross axis on the line (column or row).
 - * `Start`
 - * `Center`
 - * `End`
- `Staggered Settings EasyLayoutStaggeredSettings`
 - Only for the `Staggered` layout.
 - `Fixed Block Count bool`
 - Count of the rows or columns.
 - `Blocks Count int`
- `Ellipse Settings EasyLayoutEllipseSettings`
 - Only for the `Ellipse` layout.
 - Set equal width and height for the circle layout.
 - `RectTransform` pivot is used as the center of the ellipse.
 - `Width Auto bool`
 - `RectTransform` width is used as the width of the ellipse.
 - `Width float`
 - Ellipse width if `Width Auto` disabled.
 - `Height Auto bool`
 - `RectTransform` height is used as the height of the ellipse.
 - `Height float`
 - Ellipse height if `Height Auto` disabled.
 - `Angle Start float`
 - Position of the first element in the degrees.
 - `Angle Step Auto bool`
 - Are elements placed with equal angular distance or specified `Angle Step`?
 - `Angle Step float`
 - Elements placed with specified angular distance between neighbour elements.
 - `Fill EllipseFill`
 - Determines how to calculate the distance between elements if `Angle Step Auto` enabled.
 - * `Closed`: angular distance is 360 degrees divided into the elements count; distance is the same between the first and last elements.
 - * `Arc`: angular distance is arc length divided into the elements count minus one
 - `Arc Length float`

Distance between first and last elements if `Angle Step Auto` enabled and `Fill` is `Arc`.

Can be more than 360 degrees.

- `Align EllipseAlign`

Determines how elements are placed on the ellipse border.

- * `Outer`: right borders of the elements are placed on the ellipse border.

- * `Center`: center of the elements are placed on the ellipse border.

- * `Inner`: left borders of the elements are placed on the ellipse border.

- `ElementsRotate bool`

Rotate elements according to position or not.

- `ElementsRotationStart float`

Initial rotation of the elements.

- `Spacing Vector2`

Empty space between elements.

Can be more than specified value for `Flex` layout.

- `Symmetric bool`

Use symmetric margin.

- `Margin Vector2`

Empty space from parent edges.

- `Skip Inactive bool`

Do not reserve space for disabled elements.

- `Right To Left bool`

The order of placement of elements.

- `Top To Bottom bool`

The order of placement of elements.

- `ResetRotation bool`

Reset rotation of the elements to 0.

- `Children Width ChildrenSize`

- `Do nothing`: do not resize elements.

- `Set Preferred`: set element width to `Preferred Width`.

- `Set Max From Preferred`: set maximum of the `Preferred Width` from the all elements.

- `Fit Container`: increase elements width proportionally `Flexible Width` to fit parent width.

- `Shrink On Overflow`: decrease elements width if summary width more than parent width with margin.

- `Children Height ChildrenSize`

Similar to `Children Width`

5.4.2 Events

- Settings Changed `UnityEvent`
Event, raised after any setting was changed.

5.5 Layout Switcher

Allow to create different layouts with same `GameObjects` for different screen sizes and aspect ratios. Used when anchors, pivots and layout group not enough to create layout with different aspect ratios support.

Save values of the position, size, anchors, pivot, rotation, scale, active/disable state for each layout.

5.5.1 Options

- Objects `List<RectTransform>`
List of the controlled objects.
- Default Display Size (inches) `float`
Display size to use when actual display size cannot be detected.
- Layouts `List<UILayout>`
List of the layouts.
 - Name `string`
Layout name.
 - Aspect Ratio `Vector2`
Aspect ratio for this layout.
 - Max Display Size (inches) `float`
Maximum size of the display for this layout (layout will not be used if display size more than specified).

5.5.2 Events

- LayoutChanged `UnityEvent<UILayout>`

5.6 Lightbox

Lightbox is a component used to display overlay image.

5.7 Object Sliding

Component to drag `GameObject` horizontally or vertically between specified positions.

5.7.1 Options

- Positions `List<float>`
Allowed positions for this object.
- Direction `ObjectSlidingDirection`
Slide direction.

- Horizontal
 - Vertical
- Movement `AnimationCurve`
Animation curve.
- Unscaled Time `bool`
Animate with unscaled time.

5.7.2 Helper components

This components used to automatically set *Positions* instead of the manual input.

- Object Sliding Horizontal Helper
 - Object on Left `List<RectTransform>`
List of the objects on the left side of the current object.
 - Object on Right `List<RectTransform>`
List of the objects on the right side of the current object.
- Object Sliding Vertical Helper
 - Object on Top `List<RectTransform>`
List of the objects on the top side of the current object.
 - Object on Bottom `List<RectTransform>`
List of the objects on the bottom side of the current object.

5.8 Resizable

Allow to resize gameobject.

5.8.1 Options

- AllowResize `bool`
Allow resize.
- Update RectTransform `bool`
Change RectTransform size.
- Update LayoutElement `bool`
Change LayoutElement size.
- Active Region `float`
Distance from border where resize allowed.
- Min Size `Vector2`
Minimal size.
- Max Size `Vector2`
Maximum size, not applied if size is zero.
- Keep Aspect Ratio `bool`

Aspect ratio applied after `MinSize` and `MaxSize`, so if default aspect ratio not equal `MinSize` and `MaxSize` aspect ratio then real size may be outside limit with one of the axis.

- `Resize Directions Resizable.Directions`

Allowed resize directions.

5.8.2 Events

- `OnStartResize UnityEvent<Resizable>`
- `OnResize UnityEvent<Resizable>`
- `OnEndResize UnityEvent<Resizable>`

5.8.3 Resize Children With Parent

There are a few ways to resize children with parent:

- Use `RectTransform anchors` to set children size relative to parent with padding from borders.

Probably setting anchors to horizontal stretch (for the labels or buttons) or horizontal and vertical stretch (for the long text or `ListView`) will be enough.

[Video](#) about anchors.

- Add `Layout Group` (`Horizontal Layout Group`, `Vertical Layout Group`, `Grid Layout Group`, [EasyLayout](#)) to parent with enabled `Control Child Size options`.

It is a more complex way, and it will be harder to achieve the desired result.

If you want to add/remove/enable/disable children from a script and automatically reposition them after this, then `Layout Group` is the right way to do this.

5.9 Scrollbar Min Size

Allow to set minimal scrollbars sizes of the `ScrollRect`.

5.9.1 Options

- `Horizontal Min Size float`
Minimal size of the horizontal scrollbar.
- `Vertical Min Size float`
Minimal size of the vertical scrollbar.

5.10 ScrollRect Events

Provide pull events for the `ScrollRect`.

5.10.1 Options

- `Required Movement float`
Required pull distance to raise events.

5.10.2 Events

- `OnPull` `UnityEvent<PullDirection>`
- `OnPullAllowed` `UnityEvent<PullDirection>`
- `OnPullCancel` `UnityEvent<PullDirection>`
- `OnPullUp` `UnityEvent`
- `OnPullDown` `UnityEvent`
- `OnPullLeft` `UnityEvent`
- `OnPullRight` `UnityEvent`

5.11 Selectable Helper

Selectable works only with one Graphic component, Selectable Helper allows to control more Graphic components.

5.12 Sidebar

Component to drag sidebar from behind the screen.

5.12.1 Options

- `Interactable` `bool`
Enable or disable the ability to drag the sidebar.
- `Curve` `AnimationCurve`
Animation curve for the open and close animations.
- `Direction` `SidebarAxis`
Drag direction to open sidebar.
- `Animation Type` `SidebarAnimation`
 - `Overlay`
 - `Push`
 - `Scale Down`
 - `Uncover`
 - `Slide Along`
 - `Slide Out`
 - `Resize`
 - `Scale Down and Push`
- `Scale Down Limit` `float`
Content scale cannot be lower this value for the `ScaleDown` animation.
- `Is Open` `bool`
Is sidebar opened?
- `Modal` `bool`

Is sidebar should be closed with the click outside of the sidebar?

- `ScrollRect Support` `bool`
Allow to handle children `ScrollRect`'s drag events.
- `Content RectTransform`
Content `GameObject`. Required by some animations.
- `Animate With Layout` `bool`
Change Content `LayoutElement` size during animation.
- `Optional Handle` `GameObject` (optional)
Handle to open and close sidebar.
- `Unscaled Time` `bool`
Run animations with unscaled time.

5.12.2 Events

- `OnOpen` `UnityEvent`
- `OnClose` `UnityEvent`
- `OnOpeningStarted` `UnityEvent`
- `OnClosingStarted` `UnityEvent`

5.13 Splitter

Resize neighboring or specified gameobjects on drag. Should be used with layout group.

5.13.1 Options

- `Type` `SplitterType`
 - Horizontal: change heights of the gameobjects.
 - Vertical: change widths of the gameobjects.
- `Update RectTransform` `bool`
Change `RectTransform` size of the left and right gameobjects.
- `Update LayoutElement` `bool`
Change `LayoutElement` size of the left and right gameobjects.
- `Mode` `SplitterMode`
 - Auto: use previous and next siblings in hierarchy.
 - Manual: use specified targets to resize.
- `Previous Object` `RectTransform`
Left (or top) object to resize.
- `Next Object` `RectTransform`
Right (or bottom) object to resize.

5.13.2 Events

- `OnStartResize` `UnityEvent<Splitter>`
- `OnResize` `UnityEvent<Splitter>`
- `OnEndResize` `UnityEvent<Splitter>`

5.14 Switch Group

Same as Toggle Group, but for the Switch widget.

5.14.1 Options

- `Allow Switch Off` `bool`

Is it allowed that no switch is on? If this option is enabled, pressing the switch that is currently on will change it to off, so that no switch is on. If this setting is disabled, pressing the switch that is currently on will not change its state.

5.15 Table Header

Used with `ListView` on table mode. Allow columns resizing and reordering.

Important: `TableHeader` and the `ListView.DefaultItem` should have same amount of the children `GameObjects` (cells count should match with header cells count).

5.15.1 Options

- `List` `ListViewBase`

Controlled `ListView`.

- `Allow Resize` `bool`

Allow to change columns width.

- `Allow Reorder` `bool`

Allow to change columns order.

- `On Drag Update` `bool`

Update column width during drag, if disabled column width will be changed after the drag ended.

- `Active Region` `float`

Distance from border where resize allowed.

- `Drop Indicator` `LayoutDropIndicator`

Indicator to display new column position during column reordering.

5.16 Tooltip

Displaying the tooltip on object focus.

5.16.1 Options

- `Tooltip Object GameObject`
GameObject used as tooltip.
- `Bring To Front bool`
Bring tooltip object to front.
- `Show Delay float`
Delay in seconds before tooltip displayed.
- `Unscaled Time bool`
Delay with unscaled time.

5.16.2 Events

- `OnShow UnityEvent`
- `OnHide UnityEvent`

5.17 TreeView data source

Used in editor mode, allow to edit TreeView nodes.

Important: Work only with default TreeView. Custom TreeView's are not supported.

CHAPTER 6

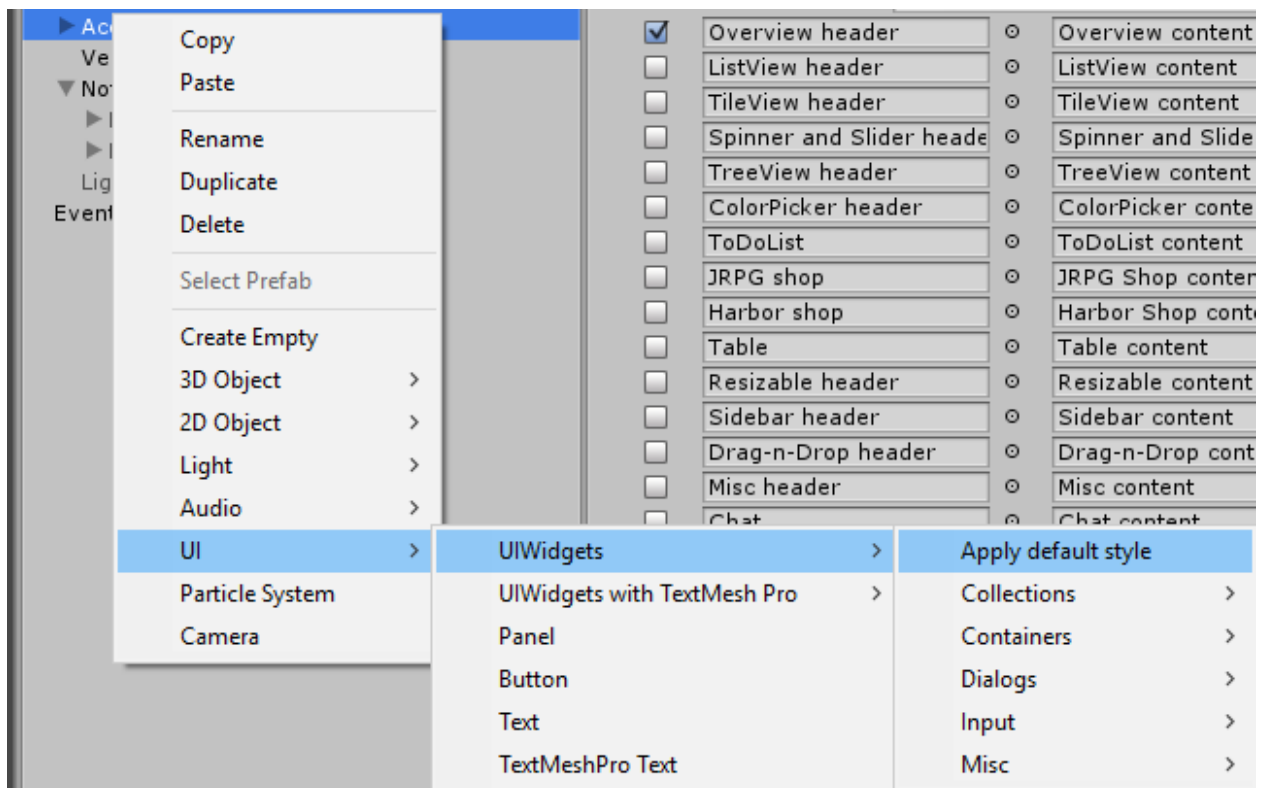
Styles (Skins)

Styles are like skins. They are used to change the **Color**, **Text**, and **Images** options of the widgets.

New UI Widgets contains two predefined styles: *Default* and *Blue*.

New style can be created with menu *Assets / Create / New UI Widgets / Style*.

You can set any style to use as default. Default style will be applied for the created widgets. Also, you can apply style for objects on the scene with *UI / New UI Widgets / Apply default style*.



Styles has two modes: *fast* and *detailed* settings:

- *Fast* allow to quickly set settings for all widgets with *Apply Fast Settings* button.
- *Detailed* allow to tune settings for each widget type separately.

Note: Recommended **Sprites** settings to avoid graphical artifacts:

- *Filter Mode* = *Point* for sprites with straight lines, *Trilinear* in other cases
 - *Generate Mip Maps* = disabled
 - *Compression* = *None* (you can try other settings, but be careful because sometimes compression artifacts can be seen only in *Play mode*)
-

6.1 Style support for the custom widgets

You can add style support for your widgets with `IStylable` implementation.

```
using UIWidgets.Styles;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Image))]
public class CustomPanel : MonoBehaviour, IStylable
{
    public virtual bool SetStyle(Style style)
    {
        style.Collections.MainBackground.ApplyTo(GetComponent<Image>());

        return true; // true if style applied to children gameobjects; otherwise false.
    }
}
```

Note: Widgets created with *Widgets Generation* already have style support.

7.1 Default ListViewIcons and TreeView

ListViewIcons and TreeView items have LocalizedName field, and you can use it for localization support.

```
using UnityEngine;
using System.Collections;
using UIWidgets;

[RequireComponent (typeof (ListViewIcons))]
public class ListViewIconsLocalization : MonoBehaviour
{
    ListViewIcons targetListView;

    public ListViewIcons TargetListView;
    {
        get
        {
            if (targetListView==null)
            {
                targetListView = GetComponent<ListViewIcons>();
            }

            return targetListView;
        }
    }

    void Start ()
    {
        LocalizationSystem.OnLanguageChanged += Localization;
        Localization();
    }

    void OnDestroy ()
    {

```

(continues on next page)

(continued from previous page)

```
LocalizationSystem.OnLanguageChanged -= Localization;
}

public void Localization()
{
    TargetListView.DataSource.BeginUpdate();
    TargetListView.DataSource.ForEach(x => x.LocalizedName = LocalizationSystem.
↪getLocalizedString(x.Name));
    TargetListView.DataSource.EndUpdate();
}
}
```

7.2 Generated ListView, TreeView, TileView

You can change component class to add localization support.

```
public class SomeItemComponent : ListViewItem, IViewData<SomeItem>
{
    SomeItem item;

    void Start()
    {
        LocalizationSystem.OnLanguageChanged += Localization;
    }

    void OnDestroy()
    {
        LocalizationSystem.OnLanguageChanged -= Localization;
    }

    public virtual void SetData(SomeItem newItem)
    {
        item = newItem;
        Localization();
    }

    public virtual void Localization()
    {
        Text.text = item.LocalizedField ?? item.OriginalField;

        Description.text = LocalizationSystem.getLocalizedString(item.Description);
    }
}
```

If you need sorting for some fields, you can add special fields for localization. For other fields, you can apply localization in `Component.SetData()` function.

```
Comparison<SomeItem> ItemsComparison = (x, y) => (x.LocalizedField ?? x.
↪OriginalField).CompareTo(y.LocalizedField ?? y.OriginalField);
ListView.DataSource.Comparison = ItemsComparison;
```


7.3 Tabs

Use derived class from `TabButtonComponent` for Tabs with overridden `SetButtonData` method or `TabIconActiveButton` and `TabIconDefaultButton` for `TabIcons` with overridden `SetData` method.

```
public class TabButtonComponentLocalized : TabButtonComponent
{
    public override void SetButtonData(Tab tab)
    {
        Name.text = LocalizationSystem.getLocalizedString(tab.Name);
    }
}

public class TabIconActiveButtonLocalized : TabIconActiveButton
{
    public override void SetData(TabIcons tab)
    {
        base.SetData(tab);
        Name.text = LocalizationSystem.getLocalizedString(tab.Name);
    }
}
```

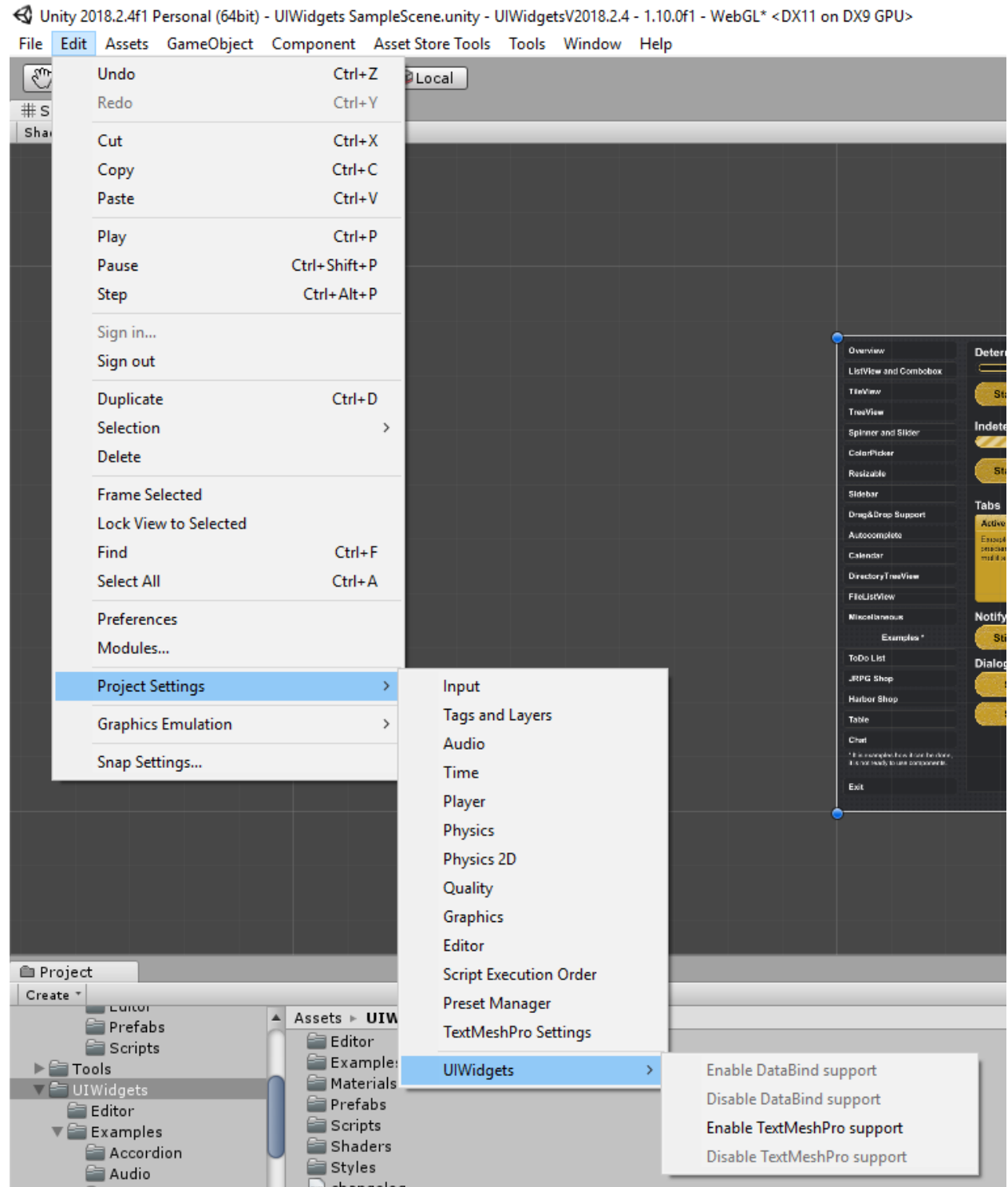
Data Bind for Unity Support

You can enable **Data Bind for Unity** support with *Project Settings / New UI Widgets / Enable DataBind support*. If **Data Bind for Unity** not installed option will not be available.

After enabling support:

- will be available **Data Bind** support for default widgets
- for generated widgets support can be added with context menu *Assets / New UI Widgets / Add Data Bind support*

Disable support with *Project Settings / New UI Widgets / Disable DataBind Pro support*.



TextMesh Pro Support

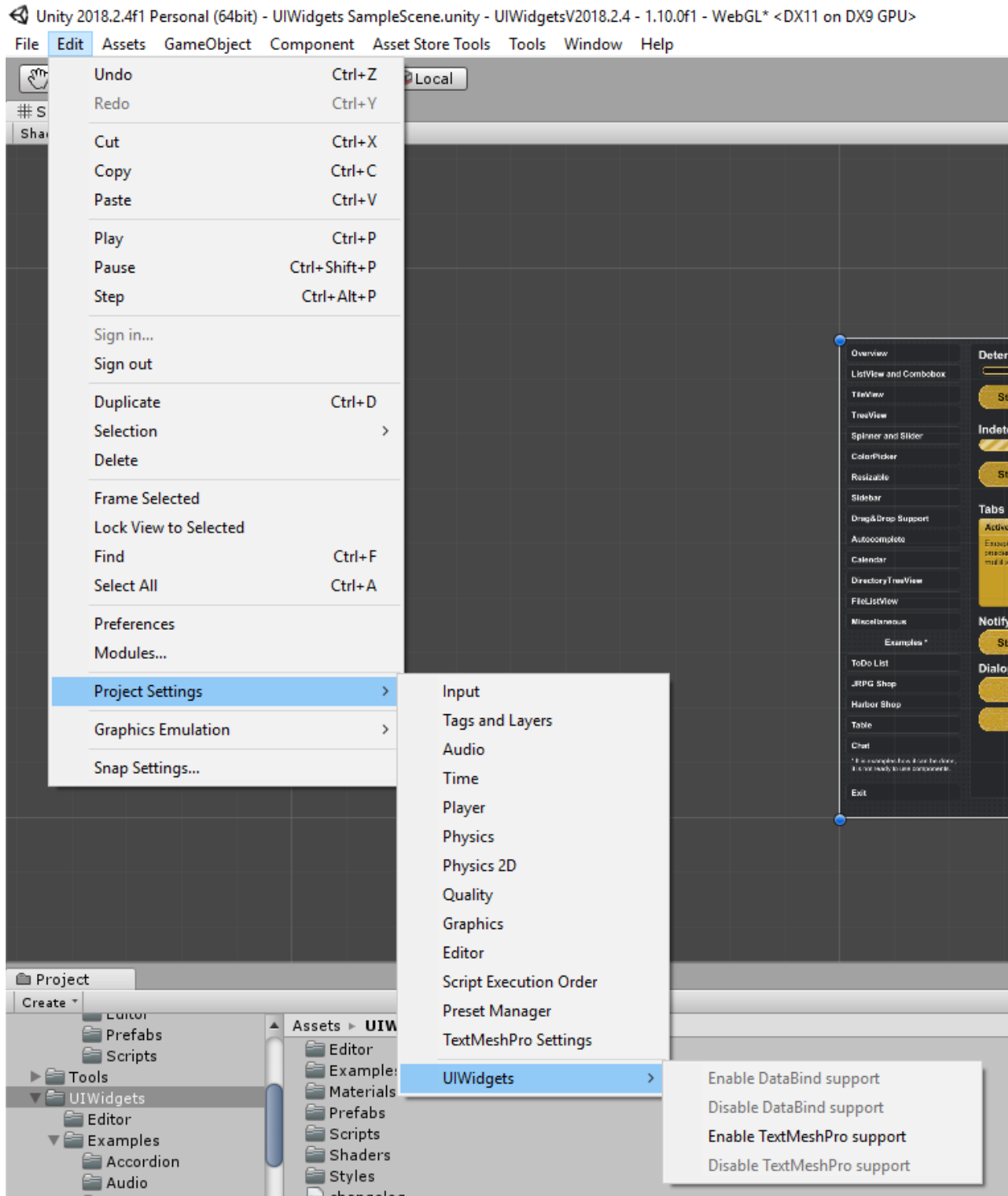
You can enable **TextMesh Pro** support with *Project Settings / New UI Widgets / Enable TextMesh Pro support*. If **TextMesh Pro** not installed option will not be available.

After enabling support:

- widgets created with menu *UI / New UI Widgets /* will use **TextMesh Pro** instead of the default **Text**
- generated widgets will be using TextMesh Pro instead default Text

Disable support with *Project Settings / New UI Widgets / Disable TextMesh Pro support*.

You can import **TextMesh Pro support** package again with *Project Settings / New UI Widgets / Import TextMesh Pro support package* after updating main package to new version.



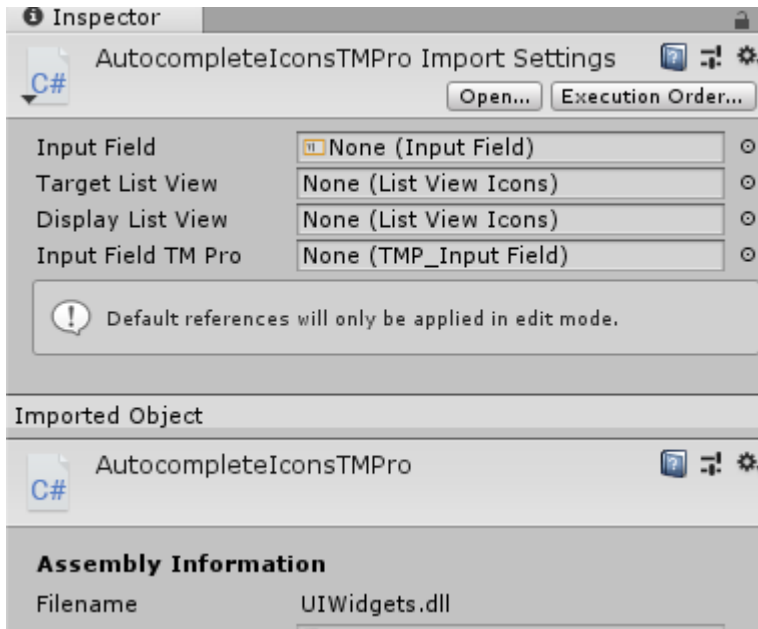
9.1 Known problems and solutions

- Widgets have missing script errors after enabling TextMesh Pro Support

Some Unity versions do not recompile scripts after *Scripting Define Symbols* was changed to enable support.

First try to check “Scripting Define Symbols” from menu Edit/Project Settings/Player - it should contain *UI-WIDGETS_TMPRO_SUPPORT* after enabling TMPPro support, if not then you need to add it.

Then check script file *New UI Widgets / Scripts / ThirdPartySupport / TMPProSupport / Scripts / Autocomplete / AutocompleteIconsTMPPro.cs* - it should be valid MonoBehaviour script like this:



If it display “No MonoBehaviour scripts in the file, ...” then this is a problem with code recompilation.

To fix it you need to modify function `ForceRecompileByLabel` in script “New UI Widgets/Scripts/Utilites/Compatibility.cs”.

Add your Unity version number to line:

```
#if UNITY_2018_1 || UNITY_2018_2
```

like this:

```
#if UNITY_2018_1 || UNITY_2018_2 || UNITY_2019_1
```

And then disable TMPPro support and enable it back to force files recompilation.

CHAPTER 10

Support

You can ask me questions at:

- Forum thread: <https://forum.unity.com/threads/new-ui-widgets.297353/>
- Forum private conversation: <https://forum.unity.com/conversations/add?to=ilih>
- Disqus: <https://ilih.ru/unity-assets/UIWidgets/>
- Email: support@ilih.ru

11.1 Release 1.11.2

- added TracksView to create custom schedule or time-line widgets
- added InputFieldAdapter to improve TextMesh Pro support
- added ListComponentPool
- added SplitButton
- Dialog: added RectTransform content and Action onClose parameters to Show(...) method
- Dialog: added OpenedDialogs property to get list of the opened dialog
- Dialog: DefaultButton replaced with ButtonsTemplates and DialogActions now has option to specify button index for the button template
- Dialog: type of the “buttons” parameter in the Show() method changed to IList<DialogButton>
- Dialog: added “Func<int, bool> onCancel” parameter to the Show() method, called with -1 parameter when dialog closed with top right close button
- DragListener: OnDragListener renamed to DragListener
- DragSupport: added optional DragHandle property, you can use it drag ListView items by specified handle instead of the whole item
- DragSupport: added StartDragEvent and EndDragEvent
- EasyLayout: added ElementsRotate and ElementsRotationStart for Ellipse layout
- ListView: improved navigation support
- ListView: added optional parameter minVisiblePart to IsVisible() method
- ListView: replaced old ListView with ListViewString
- ListView: added Virtualization setting to disable Virtualization
- ListViewDropSupport: added DropPosition parameter

- ListViewPaginator: now use ListView.ScrollToAnimatedPosition instead of the own animation
- Notify: fixed incorrect size and rotation of next notification if previous notification was closed during hide animation
- Resizable: added AllowResize property to enable/disable resize without removing component
- ScrollBlock: SetText() renamed to UpdateView()
- ScrollRectPaginator: ForceScrollOnPage replaced with ForcedPosition to support different positions

11.2 Release 1.11.1

- added AutocompleteCombobox
- ListView: fixed scrolling bug with variable size list types
- Notify: renamed AnimationRotate to AnimationRotateVertical, AnimationCollapse to AnimationCollapseVertical
- Notify: added animations AnimationRotateHorizontal, AnimationCollapseHorizontal, AnimationSlideRight, AnimationSlideLeft, AnimationSlideUp, AnimationSlideDown
- Notify: added configurable animations AnimationRotateBase, AnimationCollapseBase, AnimationSlideBase
- Resizable: added OnResize event
- Splitter: added OnResize event
- Tabs: added SelectedTabIndex property

11.3 Release 1.11.0

- added ScrollRectHeader (example of usage in Examples/ListView/ListViewHeader scene)
- added EasyLayoutEllipseScroll
- Combobox: added OnShowListView and OnHideListView events
- EasyLayout: added new layout type Ellipse
- EasyLayout: added new option ResetRotation
- ListView: added DestroyDefaultItemsCache, if enabled instances of the previous DefaultItem will be destroyed when replacing DefaultItem
- ListView: added new ListViewEllipse list type
- Scroller: renamed to ScrollBlock

11.4 Release 1.10.4

- added DateScroller, DateTimeScroller, DateTimeScrollerSeparate, TimeScroller widgets
- added EditorCondition attributes to use with MonoBehaviourConditional and UIBehaviourConditional
- added LayoutElementMax: allow to control the maximum preferred sizes of the LayoutGroup
- added UIFlare shader
- Combobox: added HideAfterItemToggle option
- DateTime: fixed init and time errors

- DatePicker: added DateChangeOnly option to allow to select date on change or on click
- EasyLayout: fixed FitContainer
- ListView: added null value support for the GraphicsForeground and GraphicsBackground properties
- ListView: added AllowColoring option
- ListView: added StateDefault(), StateSelected() and StateHighlighted() functions to the base default item class as addition to coloring functions
- ListView: added loading example with UIFlare shader use

11.5 Release 1.10.3

- added GroupedTileView example
- DragRedirect: improved support for the multiple redirects
- GroupedList: added ItemsPerBlock, EmptyGroupItem, EmptyItem properties for the TileView support
- EasyLayout: added Flex layout type
- EasyLayout: added Staggered layout type
- EasyLayout: renamed Stacking to MainAxis
- ListView: HighlightedBackgroundColor and HighlightedColor now applied automatically after changed
- ListView: fixed scrolling when List Type is fixed, ListScrollValue enabled and DefaultItem have Layout Group
- ListView: fixed rare bug for the ListView with items of the variable sizes.
- ListView: added missing fields in the Inspector window for the simple ListView
- ListView: added TileViewStaggered renderer
- ScrollRectPaginator: fixed displayed buttons at the start
- Style: fixed error when style created not in the folder or outside Assets folder
- TextMesh Pro support: improved support for the Unity 2019.1
- Tooltip: fixed displayed tooltip after parent gameobject was disabled (thanks to Gladyon)
- Widget Generation: fixed bug when type has only one field of the supported types

11.6 Release 1.10.2

- added ScrollbarMinSize component - allow set minimum size of the scrollbar handle
- added DragOneDirection component - it changes drag event to work only with one direction
- added LayoutDropIndicator component to use with TableHeader
- added Project Settings support for Unity 2018.3 and later
- Accordion: fixed problems when content size changed
- Accordion: added ForceOpen() and ForceClose() functions to open and close items without animation
- Accordion: added fields AnimationOpen, AnimationOpenFlexible, AnimationClose, AnimationCloseFlexible to change animations
- AudioPlayer: added setter for Source property
- LayoutSwitcher: added LayoutSelector field to control layout selection

- ListView: added CanSelect(index) and CanDeselect(index) fields
- ListView: added PrecalculateItemSizes, disabling this option increase performance with huge lists of items with variable sizes
- ListView: fixed LimitScrollValue when scroll to end
- ListView: fixed error when drag-and-drop position after the last item
- ObservableList: added INotifyPropertyChanged implementation
- ObservableList: added ObserveItems field
- ObservableList: now allowed null items
- RangeSlider: now correctly works when enabled or disabled inside layout groups
- ResizableHeader: renamed to TableHeader with related class
- TableHeader: no more required IResizableItem implementation for the ListView.DefaultItem
- TableHeader: added GetColumnsOrder() and SetColumnsOrder() functions
- TableHeader: added DropIndicator support
- Sidebar: added prefab and styles support
- Spinner: now use InputField component instead of the inheritance
- Spinner: added TextMesh Pro support
- Switch: SetStatus() now does not invoke events for other Switches in the same group
- TextMesh Pro support: widgets created with default menu “UI / New UI Widgets / ...” if support enabled
- TextMesh Pro support: removed menu “UI / UIWidgets with TextMesh Pro / ...”
- TextMesh Pro support: added menu “Edit / Project / Settings / New UI Widgets / Import TextMesh Pro support package” to import TMPPro prefabs after update to new version
- Widget Generation: added ScriptableObject support
- Widget Generation: added Data Bind support
- Other: fixes related using instantiate with inited complicated widgets
- Other: “UIWidgets” in the menu replaced with “New UI Widgets” to match with the package name
- Other: Time used with animations can be controlled with Utilites.GetTime field (You can use own Time manager instead of the default Time.time)

11.7 Release 1.10.1

- ListView: added ScrollTo(item) and ScrollToAnimated(item) functions
- Paginator: added StopAnimation() function
- ListViewPaginator: fixed direction problem
- TreeView: added ScrollTo(node) and ScrollToAnimated(node) functions
- TreeView: added FindNode() function
- TreeView: now ScrollTo(..) and ScrollToAnimated(...) correctly work with node indentation
- Widget Generation: added interface types support
- Widget Generation: fixed property support

11.8 Release 1.10.0

- Added styles support (Styles folder, new styles can be created from context menu “Create / UIWidget - Style”)
- Added widget generation (context menu “Create / UIWidget - Widgets” on file with item class definition)
- Added DateTime, Time24 and Time12 widgets
- Added DateTimePicker and TimePicker widgets
- Added ColorPickerRangeHSV widget
- Added ColorsList widget to display list of the selected colors, should be used with ColorPicker or ColorPicker-Range.
- Added “Data Bind for Unity” support (requires Unity 5.6 or later)
- Added base ListView Picker class for the custom ListView
- Added base TreeView Picker class for the custom TreeView
- Added base drop support class for the custom TreeView
- Added base drop support class for the custom TreeView node
- Added assembly definitions
- Improvement: Drag can be canceled with Cancel button
- Accordion: added AllItemsCanBeClosed option
- Autocomplete: added GetInputFieldText() function
- Calendar: added DateMin and DateMax properties
- Calendar: added currentDateAsDefault option
- ColorPicker: added Hex block
- ColorPicker: added new palette mode HSVCircle
- ColorPickerRange: DefaultShader replaced with DefaultShaderHorizontal and DefaultShaderVertical
- Connectors: now works correctly with “Screen Space - Camera”
- EasyLayout: reduced memory allocations
- EasyLayout: EasyLayout namespace renamed to EasyLayoutNS to avoid problems with Unity 2018.2 and later
- Interfaces: IItemWidth, IItemHeight, IListViewItemHeight, IListViewItemWidth not used anymore
- ListView: added CenterTheItems property
- ListView: added overridable functions CanBeSelected() and CanBeDeselected()
- ListView: added LoopedList option
- ListView: added Interactable option
- ListView: added IsTable option (required to valid stylization)
- ListView and TileView: ListViewCustomWidth, ListViewCustomHeight, TileViewCustom and TileViewCustomSize replaced with ListViewCustom with List Type option
- ListViewCustomWidth: TItem now does not require IItemWidth implementation
- ListViewCustomHeight: TItem now does not require IItemHeight implementation
- ListViewDropIndicator: added styles support

- ResizableHeader: fixed resize on touch devices
- Sidebar: added OnOpeningStarted and OnClosingStarted, called when appropriated animation started
- other: prefabs in “Sample Assets” folder replaced with scenes
- other: “Standart Assets” folder renamed to “Scripts”
- other: “Sample Assets” folder renamed to “Examples”
- other: removed ListViewGameObjects prefab
- other: removed outdated prefabs and sprites
- other: namespace “UIWidgetsSamples” renamed to “UIWidget.Examples”

11.9 Release 1.9.3

- Accordion: now works with content with dynamically change size
- ListView’s, TileView’s, TreeView’s: added GetItemPositionMiddle()
- ListView’s, TileView’s, TreeView’s: added ScrollToPosition()
- ListView’s, TileView’s, TreeView’s: added ScrollToPositionAnimated()
- ResizableHeader: added ColumnEnable, ColumnDisable and ColumnToggle
- ResizableHeader: fixed problem with adding columns
- ResizableHeader: improvements

11.10 Release 1.9.2

- added TreeViewCustomNodeDragSupport
- added ScrollButtons
- Autocomplete: fixed problem with resizing
- Autocomplete: added SearchDelay and MinLength options
- ColorPicker: fixed incorrect display in linear colorspace
- ColorPicker: now click on palette or image will change color
- Draggable: added Horizontal and Vertical options
- Draggable: added Restriction option
- ListViewCustomDragSupport: added DeleteAfterDrop parameter
- ListView’s, TileView’s, TreeView’s: added SetContentSizeFitter parameter
- ListView’s, TileView’s, TreeView’s: added Navigation parameter
- ListView’s, TileView’s, TreeView’s: added IsVisible() function to check if item is visible
- ListView’s, TileView’s, TreeView’s: added animated scrolling to items - ScrollToTime() and ScrollToSpeed()
- ListView’s, TileView’s, TreeView’s: Multiple renamed to MultipleSelect
- RangeSlider: added RangeSliderType; it’s allow or disable handles overlay
- Resizable: fixed error with allowed directions
- Sidebar: added new animation type ScaleDownAndPush

- Spinner: fixed input parsing problem
- Splitter: added Mode option, so you can specify left and right targets, instead using previous and next siblings in hierarchy
- TreeView: added serialization support with `TreeNode<T>.Serialize()` and `TreeNode<T>.Deserialize()`
- TreeView: fixed error when deleting selected node with disabled `DeselectCollapsedNodes`
- TreeView: added `ExpandParentNodes()` and `CollapseParentNodes()` functions
- TreeView's DefaultItem: Filler renamed to Indentation
- Dialog, Notify, Picker, Popup: `Template()` renamed to `Clone()`

11.11 Release 1.9.1

- Fixed CenteredSlider
- Fixed missing links in prefabs
- Fixed demo scene

11.12 Release 1.9.0

- Added AudioPlayer
- Added Calendar
- Added DatePicker
- Added DirectoryTreeView
- Added FileDialog
- Added FileListView
- Added FolderDialog
- Added PickerBool (can be used as Confirmation dialog with Yes/No/Cancel options)
- Accordion: added `ResizeMethod` property
- Accordion: protected `Items` property replaced with public `DataSource` property with type `ObservableList<T>`
- Accordion: added `DisableClosed` option
- ColorPicker: added Image palette, you can use it to get colors from custom Texture2D. The texture must have the Read/Write Enabled flag set in the import settings, otherwise this function will fail.
- ColorPicker: fixed bug with wrong axes with Hue palette
- Drag&Drop: added generic classes `ListViewCustomDragSupport` and `ListViewCustomDropSupport`, using them to add Drag&Drop functionality for own ListView's become more easily. Check `ListViewIconsDragSupport` and `ListViewIconsDropSupport` as reference (ignore `TreeNode` region).
- EasyLayout: fixed "dirty" scene bug when using `FitContainer` or `ShrinkOnOverflow`
- ListView's: `DataSource` can be safely used from other threads
- ListView's: added `GroupedListView` sample
- ListView's: added `.Select(int index, bool raiseEvents)` function, you can use it to select items without raising events

- ListView's: added Owner field to ListViewItem (base class for any DefaultItem), it contains link to parent ListView
- ListView's: you can implement IViewData<T> to DefaultItem component class to avoid overriding ListView.SetData() function
- ListView's: added virtual properties Graphic[] GraphicsForeground and Graphic[] GraphicsBackground to ListViewItem, you can them to specify grapgics for coloring, instead overriding coloring functions
- Resizable: mark events as used
- SlideBlock renamed to Sidebar
- Sidebar: added new animation types Overlay (default), Push, Uncover, ScaleDown, SlideAlong, SlideOut, Re-size
- Sidebar: added AnimateWithLayout option for Resize animation, use it if you need more than one Sidebar with Resize on same Content object
- Spinner: added AllowHold option, so you can disable increasing/decreasing value during pointer hold
- Switch: added .SetStatus(bool value), you can change state without raising corresponding events
- TileView's: added TileViewCustomSize
- Tooltip: added UnscaledTime option
- TreeNode: added RootNode property, used to check if nodes belong to same tree
- TreeView's and TreeNode: Nodes type change from IObservableList<TreeNode<TItem>> to ObservableList<TreeNode<TItem>>
- TreeView: added SelectedNodes property
- TreeView: added DeselectCollapsedNodes property, enabled by default
- TreeView: added .Node2Index(TreeNode<TItem> node) function
- TreeView: added .SelectNode(TreeNode<TItem> node) and .SelectNodeWithSubnodes(TreeNode<TItem> node) functions
- TreeViewDataSource: fixed incorrect branch bug (thanks to Heiko Berres)
- ProgressBar: added SpeedType option

11.13 Release 1.8.5

- InputFieldProxy: properties onValueChange, onValueChanged, onEndEdit type changed to UnityEvent<string> and get only.
- ListView: now is possible change DefaultItem in runtime
- ListViewItem: now works without ImageAdvanced
- SlideBlock: added Modal property, if enabled SlideBlock will be closed on click outside SlideBlock
- Tabs: added EnableTab and DisableTab functions

11.14 Release 1.8.4

- Added ColorPickerRange - allow selecting color from a range of two colors.
- Fixed Combobox bug.

11.15 Release 1.8.3

- Added SelectableHelper - allow controlling additional Graphic component according to selection state of current gameobject. So you can control button background color with Button component and Button text color with SelectableHelper
- Added ListViewInt
- Added Picker - base class for creating own pickers
- Added PickerInt, PickerString, PickerIcons
- Added LayoutSwitcher
- SpinnerFloat - added property Culture, specified how the number will be displayed and how input will be parsed
- SpinnerFloat - added field DecimalSeparators, along with decimal separator within Culture determine valid decimal separators for input (Warning: incompatible types with different Unity versions - Unity 4.x use string[] and Unity 5.x use char[])
- Spinner, SpinnerFloat - fixed overflow exception
- Resizable - added corners directions for resize
- ListView's - added FadeDuration for colors change

11.16 Release 1.8.2

- EasyLayout - added Shrink on Overflow option
- EasyLayout - added CompactConstraint and CompactConstraintCount options
- Splitter - fixed problem with using more than one splitter with the same container
- Tabs - added prefab for left side Tabs
- Added ScrollRectRestrictedDrag
- TextMeshPro support available with separate unitypackage
- Beta: Added Connectors. Add SingleConnector or MultipleConnector to empty gameobject

11.17 Release 1.8.0

- Added ScrollRectPaginator
- Added ListViewPaginator
- Added Autocomplete
- Added Popup
- TreeView: added TreeViewDataSource component with nodes editor
- ListView's: added ScrollTo()
- EasyLayout: reduced memory allocation
- EasyLayout: added row/column constraint for Grid layout
- Tabs: added DefaultTabName property
- TreeNode: added Path property - return list of parent nodes

- **TreeViewComponent:** added `OnNodeExpand` property with `Rotate` (rotate toggle) and `ChangeSprite` (change toggle sprite) values
- **Notify and Dialog:** added `Template()` method, now you can use `notifyPrefab.Template().Show(...)` instead `Notify.Template("template name").Show(...)`
- **CenteredSlider:** added `ValueMin`, `ValueMax` and `UseValueLimits`. If `UseValueLimits` enabled then `ValueMin <= Value <= ValueMax`
- **Tabs:** added `TabButtonComponent`, use derived class with overridden `SetButtonData()` to control how tab name will be displayed. For `TabsIcons` you can use `TabIconButton`.
- **Dialog:** added `DialogButtonComponent`, use derived class with overridden `SetButtonName()` to control how button name will be displayed.
- **Dialog:** added `DialogInfoBase`, use derived class with overridden `SetInfo()` to control how info will be displayed.
- **ListView's, TileView:** added `DropIndicator` for Drag-and-Drop
- **TileView:** added `TileViewScrollRectFitter`, `ScrollRect` will be resized to display whole number of items.

11.18 Release 1.7.4

- Added `Switch`
- **Resizable:** added `KeepAspectRatio` property
- **Tabs:** added `SelectedTab` property
- **Tabs:** added `OnTabSelect` event
- **Known problems:** `Accordion` with `EasyLayout` and `Canvas.PixelPerfect` enabled in `Unity 5.3` cause error "Trying to add (Layout Rebuilder for) {ObjectName} (UnityEngine.RectTransform) for layout rebuild while we are already inside a layout rebuild loop. This is not supported." in some cases. Workaround - use `Vertical` or `Horizontal Layout Group` instead `EasyLayout`.

11.19 Release 1.7.2

- Fixed errors in `WinStore` builds.
- **IDropSupport:** added `DropCanceled` method.
- **DragSupport:** added `DragPoint` property (empty gameobject on cursor/touch position), you can use it to attach custom gameobject with information about draggable object.
- **ListViewIconsDragSupport, TreeViewNodeDragSupport:** show information about draggable object.
- **Tabs:** added `Tabs` with icons.

11.20 Release 1.7.0

- Added `Drag and Drop` support.
- **ComboboxCustom and ComboboxIcons:** Added `Multiselect` support.
- **ResizableHeader:** Added `drag column` support.
- **TreeViewItem:** Added `Tag` property.
- **SlideBlock:** Optional support for children `ScrollRect`.
- **Accordion:** Added `Direction`.

- **Accordion:** Added support Horizontal Layout Group and Vertical Layout Group (Content Objects should have LayoutElement component).
- **ListViews:** Added limited support Horizontal Layout Group and Vertical Layout Group (you cannot change ListView direction in runtime).
- **ObservableList:** Added events OnCollectionChange (raised when items added, removed or replaced) and OnCollectionItemChange (raised when item in collection raise OnChange or PropertyChanged events).
- **ObservableList:** Added Comparison, ResortOnCollectionChanged, ResortOnCollectionItemChanged properties.
- **TreeNode:** Added Parent property. Now you can remove node from tree using Node.Parent = null or move node to another subtree Node.Parent = AnotherNode.

11.21 Release 1.6.5

- Added Resizable.
- Added Splitter.
- Added SlideBlock.
- Added ScrollRectEvents component with PullUp, PullDown, PullLeft, PullRight events (use it for refresh or load more options).
- **ListViewCustom:** Removed properties SelectedComponent and SelectedComponents.
- **ObservableList:** Now you can disable items observe in constructor.
- **ListViewItem:** Added MovedToCache function, called when item moved to cache, you can use it to free used resources.
- Added Table sample (ListViewCustom + ResizableHeader + Tooltip).
- **TileView** sample - added Resizable for TileView and TileViewItems and toggle direction.
- Bug fixes.
- Optimization.

11.22 Release 1.6.0

- **ColorPicker**
- For **ListView**, **ListViewIcons**, **ListViewCustom**, **ListViewCustomHeight**, **TileView** added support for **ObservableList**
- **Items** property marked obsolete but can be used.
- Added optional sequence parameters for **Notify** - notifications can be showed one by one, not only all at once like before.
- For **ListViewIcons** items and **TreeView** nodes added field **LocalizedName**, so now can be easily added localization support.
- **EasyLayout - Control Width, Max Width, Control Height, Max Height** replaced with “Children Width” and “Children H
 - Do Nothing
 - Set Preferred - Set width/height to preferred, like Control Width/Height

- Set Max from Preferred - Set width/height to maximum preferred width/height of items, like Max Width/Height
 - Fit Container - similar to “Child Force Expand” from Horizontal/Vertical Layout Group
- ListViewCustomHeight - implementation of IListViewItemHeight for components now optional, but you still can implement it for optimization purpose.

11.23 Release 1.5.0

- Added TileView
- Added TreeView
- Added ResizableHeader
- Direction option for ListView’s
- Value option for ListViewIcons items

11.24 Release 1.4.2

- Added ListViewCustomHeight (support items of variable heights)

11.25 Release 1.4.1

- Added CenteredSlider.

11.26 Release 1.4

- Added RangeSlider
- Added Accordion
- Bugfixes. Thanks to Nox from Purple Pwny Studios (<http://purplepwny.com>) for helping fix a mobile combobox bug.

11.27 Release 1.3

- Added ListViewIcons
- Added ComboboxIcons
- Added ListViewCustom
- Added ComboboxCustom

11.28 Release 1.2

- Added Dialog
- Added Draggable

11.29 Release 1.1

- Added Notify
- Added EasyLayout

11.30 Release 1.0

- Initial release