
New UI Widgets Documentation

Release 1.15.2f1

Ilia Novikov

Dec 12, 2021

CONTENTS

1	Overview	1
1.1	Recommended Unity UI documentation	1
1.2	Collections	1
1.3	Containers	2
1.4	Dialogs	2
1.5	Input	2
1.6	Misc	3
2	Widgets Generation	5
2.1	List of Generated Widgets	6
2.2	Requirements	6
2.3	Supported types	6
2.4	Limitations	7
2.5	Known Problems	7
2.6	Extending and Overriding Classes	7
2.7	INotifyPropertyChanged and IObservable Support	7
2.8	Replacing generated code	9
2.8.1	Collections	9
2.8.2	Autocomplete	12
3	Widgets	15
3.1	Collections	15
3.1.1	AutoCombobox	15
3.1.2	Combobox	15
3.1.3	DirectoryTreeView	16
3.1.4	FileListView	17
3.1.5	Grouped ListView, TileView and Table	18
3.1.6	ListView, TileView and Table	19
3.1.7	ListViewEnum	34
3.1.8	TracksView	35
3.1.9	TreeGraph	37
3.1.10	TreeView	37
3.2	Containers	42
3.2.1	Accordion	42
3.2.2	Tabs	44
3.3	Controls	45
3.3.1	Context Menu	45
3.3.2	Paginator	47
3.3.3	Sidebar	49
3.3.4	SplitButton	50

3.4	Dialogs	51
3.4.1	DatePicker, DateTimePicker, TimePicker	51
3.4.2	Dialog	53
3.4.3	FileDialog	57
3.4.4	FolderDialog	59
3.4.5	Notifications	61
3.4.6	Pickers	66
3.4.7	Popup	67
3.5	Input	68
3.5.1	Autocomplete	68
3.5.2	Calendar	71
3.5.3	Centered Slider	72
3.5.4	Circular Slider	74
3.5.5	ColorPicker	75
3.5.6	DateTime	75
3.5.7	DateScroller, DateTimeScroller, TimeScroller	76
3.5.8	RangeSlider	80
3.5.9	Scale	82
3.5.10	Spinner	84
3.5.11	Time	86
3.5.12	TimeAnalog	87
3.6	Miscellaneous	88
3.6.1	ProgressbarDeterminate	88
3.6.2	ProgressbarIndeterminate	90
3.6.3	Simple Tooltip	90
3.6.4	Tooltip	91
4	Components	95
4.1	Bring to Front	95
4.1.1	Options	95
4.2	Single Line and Multi Line Connectors	95
4.2.1	SingleConnector Options	95
4.2.2	MultipleConnector Options	96
4.2.3	Connector Line	96
4.2.4	ILineBuilder	96
4.3	Distance Lines	96
4.3.1	Options	98
4.4	Drag and Drop components	98
4.4.1	Drag-and-Drop Support for the Collections	98
4.4.2	How Drag&Drop works:	98
4.4.3	Collections Drag Options	99
4.4.4	Collections Drop Options	99
4.4.5	TreeView Drop Options	100
4.4.6	TreeView Node Drop Options	100
4.4.7	Custom Drag Support	100
4.4.8	Custom Drop Support	103
4.4.9	Swapping content between Drag and Drop components	105
4.4.10	Adding limitations to the Drop component	106
4.5	Draggable	106
4.5.1	Options	106
4.5.2	Properties	107
4.5.3	Events	107
4.6	EasyLayout	107
4.6.1	Options	107

4.6.2	Events	111
4.7	Groupable	111
4.7.1	Options	112
4.7.2	Events	112
4.8	Layout Switcher	112
4.8.1	Options	112
4.8.2	Events	113
4.9	Lightbox	113
4.10	ListViewAutoResize	113
4.10.1	Options	113
4.11	Object Sliding	113
4.11.1	Options	113
4.11.2	Helper components	114
4.12	Pinchable	114
4.12.1	Options	114
4.12.2	Events	115
4.13	Resizable	115
4.13.1	Options	115
4.13.2	Events	116
4.13.3	Properties	116
4.13.4	Resize Children With Parent	116
4.14	Resizable Handles	117
4.14.1	Options	117
4.14.2	Events	118
4.15	Rotatable	118
4.15.1	Options	118
4.15.2	Events	118
4.15.3	Properties	119
4.16	Rotatable Handle	119
4.16.1	Options	119
4.16.2	Events	119
4.17	Scrollbar Min Size	119
4.17.1	Options	120
4.18	ScrollRect Events	120
4.18.1	Options	120
4.18.2	Events	120
4.19	ScrollRect Footer	120
4.19.1	Options	120
4.20	ScrollRect Header	121
4.20.1	Options	121
4.21	Selectable Helper	121
4.22	SnapGrid	122
4.22.1	Options	122
4.22.2	SnapGridBase.Border	122
4.22.3	Events	122
4.23	SnapLines	122
4.23.1	Options	123
4.23.2	SnapGridBase.LineX	123
4.23.3	SnapGridBase.LineY	123
4.23.4	Events	123
4.24	Splitter	124
4.24.1	Options	124
4.24.2	Events	124
4.25	Switch Group	124

4.25.1	Options	125
4.26	Table Header	125
4.26.1	Options	125
4.26.2	Get Current Columns Order	126
4.26.3	Change Columns Order	126
4.26.4	Restore Original Columns Order	126
4.26.5	Disable Column	126
4.26.6	Enable Column	126
4.27	TreeView data source	127
5	Effects	129
5.1	Flare Effect	129
5.1.1	Options	129
5.2	Lines Drawer	129
5.2.1	Options	130
5.3	Ring Effect	130
5.3.1	Options	130
5.4	Ripple Effect	130
5.4.1	Options	131
5.5	Snap Grid Drawer	131
5.5.1	Options	131
5.6	Tsunami Effect	131
5.6.1	Options	132
6	Shaders	133
6.1	Gradient Shaders	133
7	Styles (Skins)	135
7.1	Style support for the custom widgets	136
8	Integration	137
8.1	Assembly Definitions	137
8.2	Cursor	137
8.2.1	Cursors Fields	137
8.2.2	Cursors.Cursor Fields	138
8.2.3	UICursor Static Fields	138
8.3	Localization	139
8.3.1	Dialog, Popup Localization	139
8.3.2	Notify Localization	140
8.3.3	Generated Widgets	140
8.4	String Comparison and Culture	141
8.5	Timer and Animations	141
8.6	Unity Update Methods Replacement	141
8.6.1	Interfaces to Replace Unity Update Methods	141
9	Supported Packages	143
9.1	Data Bind for Unity Support	143
9.2	I2 Localization Support	144
9.3	TextMeshPro Support	144
9.3.1	Details	145
9.4	TextMeshPro Converter	145
9.4.1	Modify Code to Adapters	146
10	Known Problems	149

11 Support	151
12 Changelog	153
12.1 Release 1.15.2	153
12.2 Release 1.15.1	154
12.3 Release 1.15.0	154
12.4 Release 1.14.2	155
12.5 Release 1.14.1	156
12.6 Release 1.14.0	156
12.7 Release 1.12.6	157
12.8 Release 1.12.5	157
12.9 Release 1.12.4	158
12.10 Release 1.12.3	158
12.11 Release 1.12.2	158
12.12 Release 1.12.1	158
12.13 Release 1.11.2	159
12.14 Release 1.11.1	160
12.15 Release 1.11.0	160
12.16 Release 1.10.4	161
12.17 Release 1.10.3	161
12.18 Release 1.10.2	162
12.19 Release 1.10.1	163
12.20 Release 1.10.0	163
12.21 Release 1.9.3	164
12.22 Release 1.9.2	165
12.23 Release 1.9.1	165
12.24 Release 1.9.0	166
12.25 Release 1.8.5	167
12.26 Release 1.8.4	167
12.27 Release 1.8.3	167
12.28 Release 1.8.2	168
12.29 Release 1.8.0	168
12.30 Release 1.7.4	169
12.31 Release 1.7.2	169
12.32 Release 1.7.0	169
12.33 Release 1.6.5	170
12.34 Release 1.6.0	170
12.35 Release 1.5.0	171
12.36 Release 1.4.2	171
12.37 Release 1.4.1	171
12.38 Release 1.4	171
12.39 Release 1.3	171
12.40 Release 1.2	171
12.41 Release 1.1	172
12.42 Release 1.0	172

OVERVIEW

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.

1.1 Recommended Unity UI documentation

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

1.2 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**

1.3 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.

1.4 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`.

1.5 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.

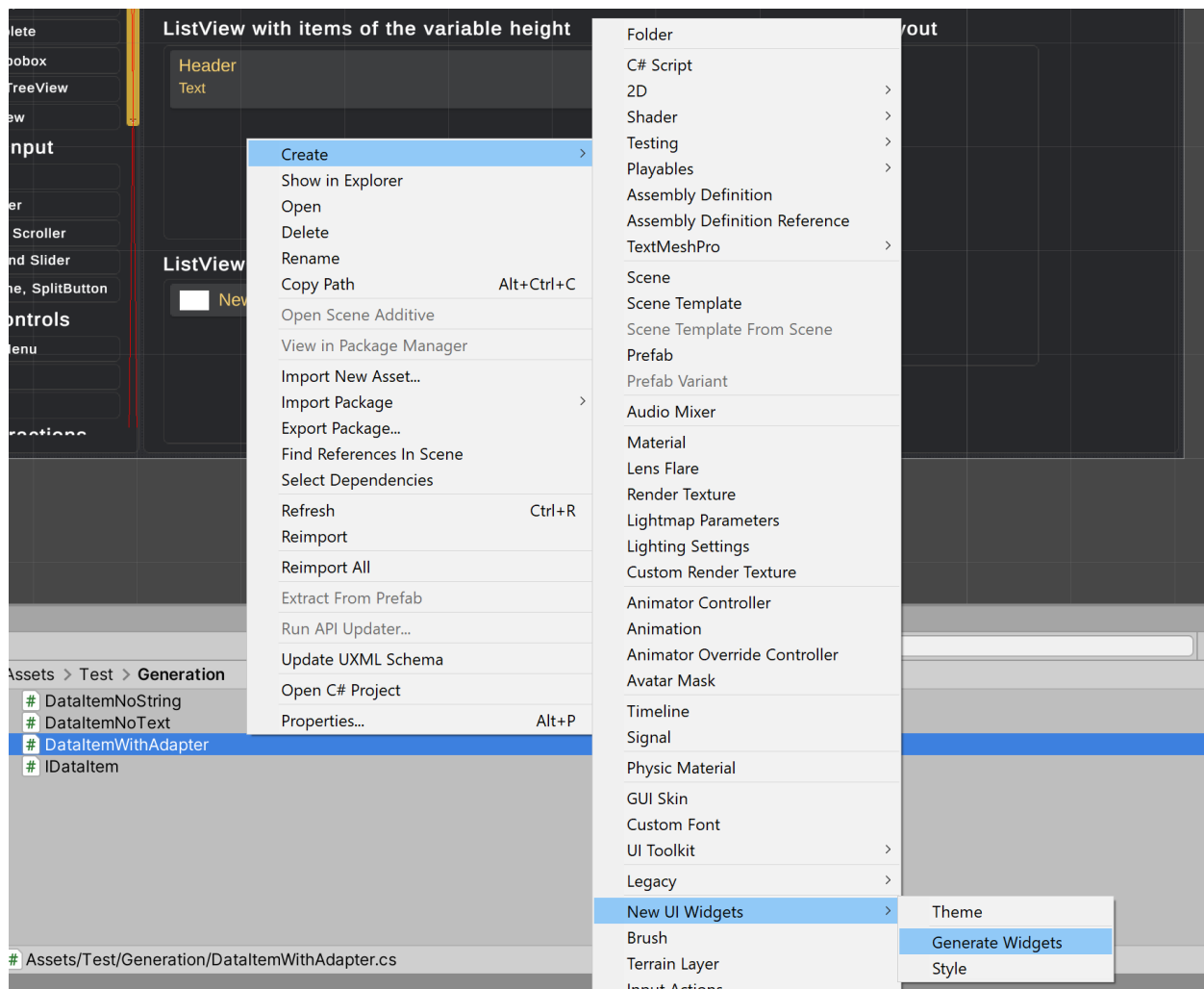
1.6 Misc

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

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

WIDGETS GENERATION

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



2.1 List of Generated Widgets

- *Autocomplete*: requires at least one field or property of `string` type.
- *AutoCombobox*: Combobox with `Autocomplete` to filter and select items by typing; requires at least one field or property of `string` type.
- *Combobox*
- *ComboboxMultiselect*: same Combobox configured to display multiple selected values.
- *DragInfo*: displays content of dragged data
- *ListView*
- *Table*
- *TileView*
- *Tooltip*
- *TreeGraph*
- *TreeView*
- *PickerListView*: Picker to select the value from `ListView`
- *PickerTreeView*: Picker to select the value from `TreeView`

2.2 Requirements

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

To be available in the inspector window data type should have `[System.Serializable]` attribute.

2.3 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`

2.4 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.

2.5 Known Problems

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

Workaround

Specify the type name in the *Data Type* field.

Another way is to change `interface` or `struct` to `class` in the type definition. Then run widgets generation and return type to `interface` or `struct`.

2.6 Extending and Overriding Classes

All generated classes are marked as `partial` to make possible it to split the definition of a class over two or more source files. The recommended way to extending generated class is to create a new source file with class definition and add new methods or overridden methods to it. It will prevent code loss in case of a new run of widgets generation for the same data type.

2.7 INotifyPropertyChanged and IObservable Support

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

If you want to automatically update collections widgets (like `ListView`, `TileView`, `Table`) on item data changes, then you need to add `INotifyPropertyChanged` or `IObservable` implementation to your data type.

Implementation can be added even after widgets generation.

The `IObservable` interface is preferable if you want to reduce memory allocations.

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

    public string Name
    {
        get
        {
            return name;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```
    }

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

public event PropertyChangedEventHandler PropertyChanged;

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

...
}
```

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

    public string Name
    {
        get
        {
            return name;
        }

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

    public event OnChange OnChange;

    protected void Changed(string propertyName)
    {
        OnChange?.Invoke(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";
```


2.8 Replacing generated code

Generated code can be freely modified.

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

2.8.1 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 UIWidgets.TextAdapter 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>
    /// 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;
            protected set;
        }

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

```

(continues on next page)

(continued from previous page)

```

        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;
}

```

(continues on next page)

(continued from previous page)

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

...
}
```

2.8.2 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>
    public class AutocompleteDataItem : UIWidgets.AutoCompleteCustom<UIWidgets.
↳ Examples.WidgetGeneration.DataItem,
        ListViewComponentDataItem, ListViewDataItem>
    {
```

(continues on next page)

(continued from previous page)

```
...
/// <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());
}
}
```


WIDGETS

3.1 Collections

3.1.1 AutoCombobox

Combobox widget combined with Autocomplete widget which allows select item by typing.

Options

- Autocomplete `TAutocomplete`
ListView with items.
- Combobox `TCombobox`
Button to show and hide ListView on click.
- AddItems `bool`
Create a new item and add it to list if item not found with specified input. Requires overridden `TItem` `Input2Item(string input)` method.
- KeepSelection `bool`
Keep selected items for `Autocomplete.DisplayListView`.

3.1.2 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`
The event raised when `ListView` showed.
- `OnHideListView UnityEvent`
The event raised when `ListView` hidden.
- `OnCurrentClick UnityEvent<int, TItem>`
The event raised on click on displayed selected item.

3.1.3 DirectoryTreeView

- All collections widgets support virtualization: gameobjects created only for the visible items.
- Add `Selectable` component to use keyboard and gamepad navigation.
- See also [FolderDialog](#).

Options

Options are almost same as the [TreeView](#).

- `Data Source ObservableList<TreeNode<FileSystemEntry>>`
Not available in the inspector window.
Filled automatically.
- `Root Directory string`
Root directory.
- `Exceptions View IOExceptionsView`
Special component to display IO errors.

Methods

- `TreeNode<FileSystemEntry> ExpandPath(string path, bool scrollToNode = true)`
Expand nodes to the specified path. Returns `null` if node not found.
- `TreeNode<FileSystemEntry> Path2Node(string path)`
Get node of the specified path. Returns `null` if node not found.
- `TreeNode<FileSystemEntry> Path2NearestNode(string path)`
Get exact node or nearest existing parent node of the specified path.
- `void RefreshDirectories()`
Refresh displayed directories according to current state of the file system.

3.1.4 FileListView

- All collections widgets support virtualization: gameobjects created only for the visible items.
- Add `Selectable` component to use keyboard and gamepad navigation.
- See also *FileDialog*.

Options

Options are almost same as the *ListView*, *TileView* and *Table*.

- Data Source `ObservableList<FileSystemEntry>`
Not available in the inspector window.
Filled automatically.
- Current Directory `string`
Current directory. `Application.persistentDataPath` will be used if not specified.
- Directory Patterns `string`
Directory patterns, semicolon used as separator between patterns.
Directory will be displayed if it's match one of the pattern.
Wildcards:
* - Zero or more characters in that position.
? - Zero or one character in that position.
Warning: if directory match two or more patterns it will be displayed two or more times.
- File Patterns `string`
File patterns, semicolon used as separator between patterns.
File will be displayed if it's match one of the pattern.
Wildcards:
* - Zero or more characters in that position.
? - Zero or one character in that position.
Warning: if file match two or more patterns it will be displayed two or more times.
- Button Up `Button`
Button to open parent directory of current directory.
- Button Toggle Drivers `Button`
Button to toggle `DriversList`.
- Path View `FileListViewPath`
Widget to display the current directory.
- Drives List View `DrivesListView`
Widget to display drives list.
- Exceptions View `IOExceptionsView`
Special component to display IO errors.
- Can Display Entry `Func<FileSystemEntry, bool>`
Not available in the inspector window.

Function to check if `FileSystemEntry` should be displayed.

3.1.5 Grouped ListView, TileView and Table

You can create grouped `ListView` with `GroupedList<TItem>` class.

```
public class GroupedItem
{
    public string Name;
    public bool IsGroup = false;
}

public class GroupedItems : GroupedList<GroupedItem>
{
    /// <summary>
    /// Get group for specified item.
    /// </summary>
    /// <param name="item">Item.</param>
    /// <returns>Group for specified item.</returns>
    protected override GroupedItem GetGroup(GroupedItem item)
    {
        var name = item.Name.Length > 0 ? item.Name[0].ToString() : string.Empty;

        foreach (var key in GroupsWithItems.Keys)
        {
            if (key.Name == name)
            {
                return key;
            }
        }

        return new GroupedItem() { Name = name, IsGroup = true, };
    }
}

public class GroupedView : ListViewCustom<GroupedListViewComponent, GroupedItem>
{
    /// GroupedData used to add and remove items instead of the DataSource.
    public GroupedItems GroupedData = new GroupedItems();

    bool isGroupedViewInited;

    public override void Init()
    {
        if (isGroupedViewInited)
        {
            return;
        }

        isGroupedViewInited = true;

        base.Init();

        GroupedData.GroupComparison = (x, y) => x.Name.CompareTo(y.Name);
        GroupedData.Data = DataSource;

        CanSelect = index => !DataSource[index].IsGroup;
    }
}
```

(continues on next page)

(continued from previous page)

```
}  
}
```

3.1.6 ListView, TileView and Table

- All collections widgets support virtualization: gameobjects created only for the visible items.
- Different ListView, TileView and Table can display the same list.
- *Table Header* provides Table specific methods.
- In most cases *ToggleGroup* and *SwitchGroup* components used by widgets under *DefaultItem* hierarchy should be placed outside *DefaultItem* gameobject. And on value changed callbacks should process all items, not only the current one, since invisible items do not receive callbacks because of the virtualization.

List View Type



Fig. 1: ListView with Fixed Size.

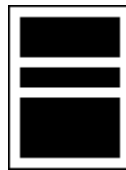


Fig. 2: ListView with Variable Size.

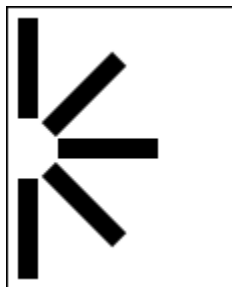


Fig. 3: ListView with Ellipse layout.



Fig. 4: TileView with Fixed Size.



Fig. 5: TileView with Variable Size.

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.

Options

- **Interactable** `bool`
Allow users interact with the ListView.
- **Virtualization** `bool`
Enable virtualization. If enabled `GameObject` instantiated only for the visible items; otherwise for the all items.
- **List Type** `ListViewType`
Determines how items are displayed.
 - `ListViewWithFixedSize`
Works with EasyLayout, Horizontal Layout Group and Vertical Layout Group.

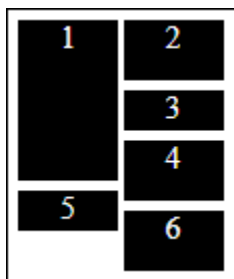


Fig. 6: TileView Staggered.

- 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 *deprecated*
 - If enabled items will be sorted with **SortFunc**. Deprecated, replaced with DataSource.Comparer.
- SortFunc Func<IEnumerable<TItem>, IEnumerable<TItem>> *deprecated*
 - Not available in the Inspector window* Function to sort items. Deprecated, replaced with **Data-Source.Comparer**.
- Data Source ObservableList<TItem>
 - List of the items. It works the same way as `List<T>` with some additions.
 - Not available in the inspector window if type not specified as serializable.
- Multiple Select bool
 - Allow to select multiple items, otherwise only one.
- Range Mode RangeSelectionMode
 - Specify range selection mode (multiple items selection with Shift key).
 - StartFromFirst
 - Select all items from the first selected item to the newly selected item.
 - StartFromLast
 - Select all items from the last selected item to the newly selected item.
- Selected Index int
 - Index of the last selected item.
- Selected Indices List<int>
 - Not available in the Inspector window* List of the selected items indices.
- Selected Item TItem
 - Not available in the Inspector window* Last selected item.
- Selected Items List<TItem>
 - Not available in the Inspector window* List of the selected items.
- Direction ListViewDirection
 - ListView direction.
 - Horizontal

- Vertical
- Default Item `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.
- Allow Coloring `bool`
 - Change colors of the highlighted and selected items.
 - If you want to more precise control on item colors, like different colors depending of item data, then you can override `StateDefault()`, `StateHighlighted()`, `StateSelected()` methods of item component class.
- 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`
Specify time type used by scroll animation.
If enabled then will be used `Time.unscaledTime`; otherwise will be used `Time.time`.
- `Scroll Movement` `AnimationCurve`
Animation curve for the `ScrollTo` functions.
Specify how long scroll animation will be and what speed will it have.
- `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.
- `Auto Scroll Area` `float`
ListView will be automatically scrolled if the pointer in less then a specified distance from the border during drag&drop.
- `Auto Scroll Speed` `float`
Speed of auto-scroll.
- `Can Select` `Func<int, bool>`
The function that determines whether the item with the specified index can be selected. Unselectable items cannot be highlighted and skipped by keyboard and gamepad navigation.
- `Can Deselect` `Func<int, bool>`
The function that determines whether the item with the specified index can be deselected.
- `Scroll Inertia Until Item Center` `bool`
Enable custom scroll inertia.
It is replace `ScrollRect` inertia in such a way so that after the end of scrolling the item will be exactly in the center.
Intended to use with `ListViewEllipse` but works with other types too.
- `Scroll Inertia` `AnimationCurve`
Similar to `Scroll Movement`, but only for the scroll inertia.

Events

- `OnSelect` `UnityEvent<int, ListViewItem>`
The event raised when item selected.
Arguments: index of the selected item and `DefaultItem` instance for the selected item.
- `OnDeselect` `UnityEvent<int, ListViewItem>`
The event raised when item deselected.
Arguments: index of the deselected item and `DefaultItem` instance for the deselected item.
- `OnSelectObject` `UnityEvent<int>`

The event raised when item selected.

Arguments: index of the selected item.

- `OnDeselectObject UnityEvent<int>`

The event raised when item deselected.

Arguments: index of the deselected item.

- `OnStartScrolling UnityEvent`

The event raised when scrolling starts.

- `OnEndScrolling UnityEvent`

The event raised when after **End Scroll Delay** from left last scroll event.

- `onSubmit UnityEvent`

The event raised when ListView gameobject has been selected via a “submit” key you specify (default is the return key).

- `onCancel UnityEvent`

The event raised when ListView gameobject has been deselected.

- `onItemSelect UnityEvent`

The event raised when ListView item gameobject has been selected via a “submit” key you specify (default is the return key).

- `onItemCancel UnityEvent`

The event raised when ListView item gameobject has been deselected.

- `OnUpdateView UnityEvent`

The event raised when ListView view was updated.

- `OnFocusIn UnityEvent<BaseEventData>`

The event raised when ListView gameobject received focus.

- `OnFocusOut UnityEvent<BaseEventData>`

The event raised when ListView gameobject lost focus.

- `OnPointerEnterObject UnityEvent<int>`

The event raised when pointer entered on ListView item gameobject.

Arguments: index of the item.

- `OnPointerExitObject UnityEvent<int>`

The event raised when pointer exited on ListView item gameobject.

Arguments: index of the item.

- `OnDataSourceChanged UnityEvent<ListViewCustom<TComponent, TItem>>`

The event raised when DataSource replaced with the new list.

Arguments: ListView instance.

Items Events

It is `ListView.ItemsEvents` field with list of items events. First argument is item index, second is item component instance, third is event data.

- `PointerClick UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on every pointer click on item component.
- `FirstClick UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on first pointer click with left mouse button on item component.
- `DoubleClick UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on second pointer click with left mouse button on item component.
- `PointerUp UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on pointer up on item component.
- `PointerDown UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on pointer down on item component.
- `PointerEnter UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on pointer enter on item component.
- `PointerExit UnityEvent<int, ListViewItem, PointerEventData>`
The event raised on pointer exit on item component.
- `Move UnityEvent<int, ListViewItem, AxisEventData>`
The event raised on move with keyboard or gamepad on item component.
- `Submit UnityEvent<int, ListViewItem, BaseEventData>`
The event raised on submit on item component.
- `Cancel UnityEvent<int, ListViewItem, BaseEventData>`
The event raised on cancel on item component.
- `Select UnityEvent<int, ListViewItem, BaseEventData>`
The event raised when item component has been selected by `EventSystem`.
- `Deselect UnityEvent<int, ListViewItem, BaseEventData>`
The event raised when item component has been deselected by `EventSystem`.
- `Resize UnityEvent<int, ListViewItem, Vector2>`
The event raised when item component size was changed.

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.
- `DisableRecycling` `bool`
Disable recycling of this instance. Used in Drag&Drop or animations (enable at the start of the animation and disable at the end).
- `GraphicsForeground` `Graphic[]`
References to the foreground objects like Text.
- `GraphicsBackground` `Graphic[]`
References to the background objects.

Methods

- `SetData(TItem item)`
Set data.
- `SelectItem()`
Select current item.
- `DeselectItem()`
Deselect current item.
- `RemoveItem()`
Remove current item from the ListView.DataSource.
- `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 to unload unused resources like sprites.
- `StateDefault()`
Called by ListView when item in the default state.
- `StateSelected()`
Called by ListView when item selected.
- `StateHighlighted()`
Called by ListView when item highlighted.

Auto-Resize DefaultItem instances on ListView Resize Maintaing Aspect Ratio

- DefaultItem.RectTransform anchors should be set to the horizontal or vertical stretch depending on ListView.Direction
- Add Aspect Ratio Fitter to the DefaultItem and set Aspect Mode=Width Controls Height or Height Controls Width depending on ListView.Direction
- Change ListView.ListType to List View With Variable Size or Tile View With Variable Size
- Make sure that ListView.Container.EasyLayout children size set to Do Nothing.

Multiple DefaultItems

ListView has the TemplateSelector property, it allow to use of different templates depending on the item or index.

```
namespace UIWidgets.Examples
{
    using UnityEngine;

    public class TreeViewMultipleDefaultItems : MonoBehaviour,
    ↳ IListViewTemplateSelector<TreeViewSampleComponent, ListNode<ITreeViewSampleItem>>
    {
        [SerializeField]
        protected TreeViewSample TreeView;

        [SerializeField]
        protected TreeViewSampleComponentContinent ContinentTemplate;

        [SerializeField]
        protected TreeViewSampleComponentCountry CountryTemplate;

        protected virtual void Start()
        {
            TreeView.TemplateSelector = this;
        }

        public TreeViewSampleComponent[] AllTemplates()
        {
            return new TreeViewSampleComponent[] { ContinentTemplate, CountryTemplate };
        }

        public TreeViewSampleComponent Select(int index, ListNode<ITreeViewSampleItem>
    ↳ item)
        {
            if (item.Node.Item is TreeViewSampleItemContinent)
            {
                return ContinentTemplate;
            }

            return CountryTemplate;
        }
    }
}
```

IListViewTemplateSelector Interface

Methods:

- TComponent [] AllTemplates()
Get all possible templates.
- TComponent Select(int index, TItem item);
Returns template to use for specified item with index.

DefaultItem Instances

```
// also available .Active and .Cache modes
foreach (var instance in ListView.GetComponentsEnumerator(PoolEnumeratorMode.All))
{
    // do somethind with DefaultItem instance
}
```

Add Item

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

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 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);
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);
```

Adding Callbacks to Custom Events of the Components

```
public class YourListView : ListViewCustom<YourListViewItemComponent,
↳YourListViewItem>
{
    protected override void AddCallback(ListViewItem item)
    {
        base.AddCallback(item);
        item.onDoubleClick.AddListener(ProcessDoubleClick);
    }

    protected override void RemoveCallback(ListViewItem item)
    {
        base.RemoveCallback(item);
        item.onDoubleClick.RemoveListener(ProcessDoubleClick);
    }

    void ProcessDoubleClick(int index)
    {
        Debug.Log("double click: " + DataSource[index]);
    }
}
```

Scroll to Item

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

Disable Items

```
protected virtual void Start()
{
    listView.CanSelect = CanBuy;
}

bool CanBuy(Item item)
{
    return player.Money >= item.Price;
}
```

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>
                    <(listItems);
                    originalItems.OnChange += Filter;
                }

                return originalItems;
            }

            set
            {

```

(continues on next page)

(continued from previous page)

```

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

        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()

```

(continues on next page)

(continued from previous page)

```
{
    base.Init();

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

/// <summary>
/// Process the destroy event.
/// </summary>
protected override void OnDestroy()
{
    if (originalItems != null)
    {
        originalItems.OnChange -= Filter;
    }

    base.OnDestroy();
}
}
```

3.1.7 ListViewEnum

Special *ListView*, *TileView* and *Table* to work with enum. Used in combination with `ListViewEnum<TEnum>`.

ListViewEnum<TEnum> Constructor Arguments

- `listView ListViewEnum`
ListView to display enum values.
- `showObsolete bool = false`
Show obsolete values.
- `long2enum Func<long, TEnum> (optional)`
Custom converter from `long` to `TEnum`, use it to avoid memory allocations by default converter.
- `enum2long Func<TEnum, long> (optional)`
Custom converter from `TEnum` to `long`, use it to avoid memory allocations by default converter.

ListViewEnum<TEnum> Properties

- `Selected TEnum`
Selected value.

Example

```
public class TestListViewEnum : MonoBehaviour
{
    [SerializeField]
    protected ListViewEnum ListView;

    ListViewEnum<AdditionalCanvasShaderChannels> Wrapper;

    protected void Start()
    {
        ListView.OnSelectObject.AddListener(ValueChanged);
        ListView.OnDeselectObject.AddListener(ValueChanged);

        Wrapper = ListView.UseEnum<AdditionalCanvasShaderChannels>(false, x =>
↪ (AdditionalCanvasShaderChannels)x);
    }

    protected void OnDestroy()
    {
        if (ListView != null)
        {
            ListView.OnSelectObject.RemoveListener(ValueChanged);
            ListView.OnDeselectObject.RemoveListener(ValueChanged);
        }
    }

    void ValueChanged(int index)
    {
        Debug.Log(string.Format("selected: {0}", EnumHelper
↪ <AdditionalCanvasShaderChannels>.ToString(Wrapper.Selected)));
    }

    /// <summary>
    /// Select values.
    /// </summary>
    public void Select()
    {
        WrapperWithFlags.Selected = AdditionalCanvasShaderChannels.Normal |
↪ AdditionalCanvasShaderChannels.TexCoord1;
    }
}
```

3.1.8 TracksView

Used for the schedule-like or timeline-like widgets.

Consists on three main blocks: - PointNamesView: used to display the name of points, for example, time or date. - TracksNamesView: used to display the names of tracks. - TrackDataView: used to display tracks items.

Options

- `Tracks ObservableList<Track<TData, TPoint>>`
List of tracks with items.
- `TrackDataView ScrollRect`
Used to display tracks items.
- `TrackNamesView ScrollRect`
Used to display the names of tracks.
- `PointNamesView ScrollBlockBase`
Used to display the name of points, for example, time or date.
- `Items Spacing float`
Empty space between items on Y axis.
- `Tracks Spacing float`
Empty space between tracks on Y axis.
- `Allow Drag Outside bool`
Allow to drag items outside of the `TrackDataView`.
- `Items to Top bool`
Push items to the top if there is empty space.
- `Compact bool`
Compact items position.
- `Allow Intersection bool`
Allow temporary intersection during drag; overlapped item will be moved to another line after drag.
- `Allow Auto Scroll bool`
Allow auto-scroll if the cursor is near the border on less the specified distance.
- `Auto Scroll Border Distance float`
Distance to the border where auto-scroll start working.
- `Auto Scroll Speed float`
Speed of the auto-scroll.
- `Default Item TDataView`
A prefab used to display item.
- `Default Track Header TTrackView`
A prefab used to display track header.
- `Default Track Background TTrackBackground`
A prefab used to display track background.
- `Track Data Dialog TTrackDataDialog`
Dialog to add/edit item.
- `Track Dialog TTrackDialog`

Dialog to add/edit track.

3.1.9 TreeGraph

Options

- `Nodes` `ObservableList<TreeNode<TItem>>`
Not available in the inspector window.
- `DefaultItem` `TComponent`
A prefab used to display item.
- `Direction` `TreeGraphDirections`
Directions: `TopToBottom`, `BottomToTop`, `LeftToRight`, `RightToLeft`.
- `Container` `RectTransform`
The container of the instantiated gameobjects used to display items.
- `Spacing` `Vector2`
Minimal space between items.
- `Line Type` `ConnectorType`
Line type: `Straight` or `Rectangular`.
- `Line Thickness` `float`
Line thickness.
- `Line Margin` `float`
The minimum space from the border before the turn of the line. Supported only by `Rectangular` lines.

3.1.10 TreeView

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

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

Options

Options are almost same as the *`ListView`*, *`TileView`* and *`Table`*.

- `Nodes` `ObservableList<TreeNode<TItem>>`
Not available in the inspector window.
- `Deselect Collapsed Nodes` `bool`
Deselect nested nodes when parent node collapsed.
- `ScrollWithIndent` `bool`

Scrolling with node indent in the secondary direction.

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);  
  
// 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);
        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();
}
```

3.2 Containers

3.2.1 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>`
- `OnStartToggleAnimation` `UnityEvent<AccordionItem>`
- `OnDataSourceChanged` `UnityEvent`

AccordionHighlight

`AccordionHighlight` is a separate component to highlight `ToggleObjects` of the opened item.

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

3.2.2 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.
- `CanSelectTab Func<Tab, bool>`
Function to check is tab can be selected.

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]);
```

3.3 Controls

3.3.1 Context Menu

To use the menu, you need to add a ContextMenu component and ContextMenu template. Different menus can use the same template.

Menu items are edited in a separate window which can be opened from the ContextMenu component.

In this window, you can specify menu items: name, icon, checkmark, item template, hotkey, and action when the item is clicked.

Initially, two item templates are available: the default template and the delimiter template; a minus sign is used as the key of the delimiter.

Any string can be used as a template key, not just signs.

The keyboard is supported: you can open the menu and navigate between menu items.

Hotkeys work out of the box with both legacy input and a new input system.

Options

- **Interactable** `bool`
Allow users interact with the ListView.
- **Template** `ContextMenuTemplate`
Context menu template.
- **MenuItems** `ObservableList<MenuItem>`
Menu items.
- **Is Default** `bool`
Is default menu? Default menu will be opened on context menu key press.
- **Navigation** `bool`
Enable keyboard and gamepad navigation.
- **Open On Right Button Click** `bool`
Open context menu on right mouse button click.
- **Open On Context Menu Key** `bool`
Open context menu on context menu key press.
- **Submenu Delay** `float`
Delay before open and close sub menu.
- **Unscaled Time** `bool`
Use unscaled time.

MenuItem Options

- `Visible bool`
Is item visible?
- `Interactable bool`
Is item interactable?
- `Icon Sprite`
Icon.
- `Checked bool`
Is item checked?
- `Name string`
Name.
- `HotKey HotKey`
HotKey can be enabled with `MenuItem.EnableHotKey()` even if item not used in menu (Supported only if `InputSystem` enabled).
- `Action UnityEvent<MenuItem>`
Action on item click.
- `Items ObservableList<MenuItem>`
Nested items.

Events

- `OnOpen UnityEvent<ContextMenu>`
The event raised when context menu opened.
Arguments: opened context menu.
- `OnClose UnityEvent<ContextMenu>`
The event raised when context menu closed.
Arguments: closed context menu.
- `OnItemSelect UnityEvent<MenuItem>`
The event raised when menu item selected.
Arguments: selected menu item.
- `OnItemDeselect UnityEvent<ContextMenu>`
The event raised when menu item deselected.
Arguments: selected menu item.

3.3.2 Paginator

Important: ScrollRect.Content anchors should be set 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.

Options

- 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.

Animation

Animation work with `AnimationCurve`. Width is the length of the animation in seconds; height is a relative distance (0 is start position; 1 is end position).

`ScrollRectPaginator` use own `Movement` field. `ListViewPaginator` uses `ListView.ScrollMovement` field.

Tile View ScrollRect Fitter

Component to resize `ListView.ScrollRect` to fit the whole number of columns and rows.

3.3.3 Sidebar

Component to drag sidebar from behind the screen.

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.

- `Modal Color` `Color`

Modal background color.

Events

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

3.3.4 SplitButton

Button with the additional dropdown list of the buttons.

Options

- `Primary Button` `Button`
Primary Button.
- `Toggle Button` `Button`
Button to toggle the *Additional Buttons Block*.
- `Additional Buttons Block` `GameObject`
Container for the additional buttons.
- `Additional Buttons` `List<Button>`
List of the additional buttons.
- `Modal Sprite` `Sprite`
Background sprite when additional buttons block displayed.
- `Modal Color` `Color`
Background color when additional buttons block displayed.

3.4 Dialogs

Dialogs, Popups, Pickers, Notifications works with templates.

Code usually looks like this:

```
dialogTemplate.Clone().Show(...)
```

`Clone()` method creates a new instance of the *dialogTemplate* (or takes an instance from the cache if available) and displayed will be this instance, not the original template.

This way, you need only one template to display multiple dialogs at the same time, and also closed dialogs instances are automatically recycled.

But if you have a script outside of the *dialogTemplate* hierarchy and it has reference to the component inside a hierarchy, this reference will never be replaced with the new instance.

The script will be work with *dialogTemplate*, not with actually displayed dialog. To change this behavior, you need to move the script inside the dialog hierarchy.

3.4.1 DatePicker, DateTimePicker, TimePicker

Nested Widgets Replacement

Nested widgets can be safely replaced with their analogs:

- time can be displayed with *Time24*, *Time12*, *TimeAnalog*, *TimeScroller*
- date can be displayed with *Calendar*, *DateScroller*
- datetime can be displayed with *DateTime*, *DateTimeScroller*.

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]
```

(continues on next page)

(continued from previous page)

```
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.
/// </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";
}
}
```

3.4.2 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.
- Close Button `Button`
Button to close dialog.
- Buttons Container `RectTransform`
Buttons container. If container not specified will be used parent of the button template.

Show() Method Parameters

All parameters are optional.

`title` and `message` also can be specified with `SetInfo()` to use formatted strings.

- `title string`
Dialog title.
Can be changed with `SetInfo()` method.
- `message string`
Dialog message.
Can be changed with `SetInfo()` method.
- `buttons ButtonsPool`
Dialog buttons.
Can be changed with `SetButtons()` method.
DialogButton fields:
 - `Label string`
Button label.
 - `Action Func<DialogBase, int, bool>`

Function to run on button click. Receive dialog instance and button index, return `true` to close dialog; otherwise `false`.

– Template Index `int`

Index of the button template.

- `focusButton string`

Button with focus by default.

Can be changed with `SetButtons()` or `FocusButton()`.

- `position Vector3?`

Dialog position.

Can be changed with `SetPosition()`.

- `icon Sprite`

Dialog icon.

Can be changed with `SetInfo()` method.

- `modal bool`

Modal dialog.

Can be changed with `SetModal()`.

- `modalSprite Sprite`

Background image for the modal dialog.

Can be changed with `SetModal()`.

- `modalColor Color?`

Background color for the modal dialog.

Can be changed with `SetModal()`.

- `canvas Canvas`

Canvas to display dialog. Required if dialog template is prefab.

Can be changed with `SetCanvas()`.

- `content RectTransform`

Dialog content. Can be used instead of the *message* and *icon*.

Can be changed with `SetContent()`.

- `onClose Action`

Action to run when dialog closed.

Can be changed with `OnClose` field.

- `onCancel Func<int, bool>`

Function to run when dialog canceled. Receive dialog instance and -1 as button index, return `true` if dialog should be closed.

Obsolete, use `Func<DialogBase, int, bool> OnDialogCancel` field instead.

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
            DialogBase.DefaultClose, // Func<DialogBase, int, bool>, receive dialog_
            ↪instance and button index, 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()
    {

```

(continues on next page)

(continued from previous page)

```

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;
}
}

```

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", SignInNotify),

            // on click close dialog
            new DialogButton("Cancel", DialogBase.DefaultClose),
        },
        focusButton: "Sign in",
        modal: true,
        modalColor: new Color(0, 0, 0, 0.8f)
    );
}

bool SignInNotify(DialogBase dialog, int index)
{
    var helper = dialog.GetComponent<DialogInputHelper>();
    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;
}

```

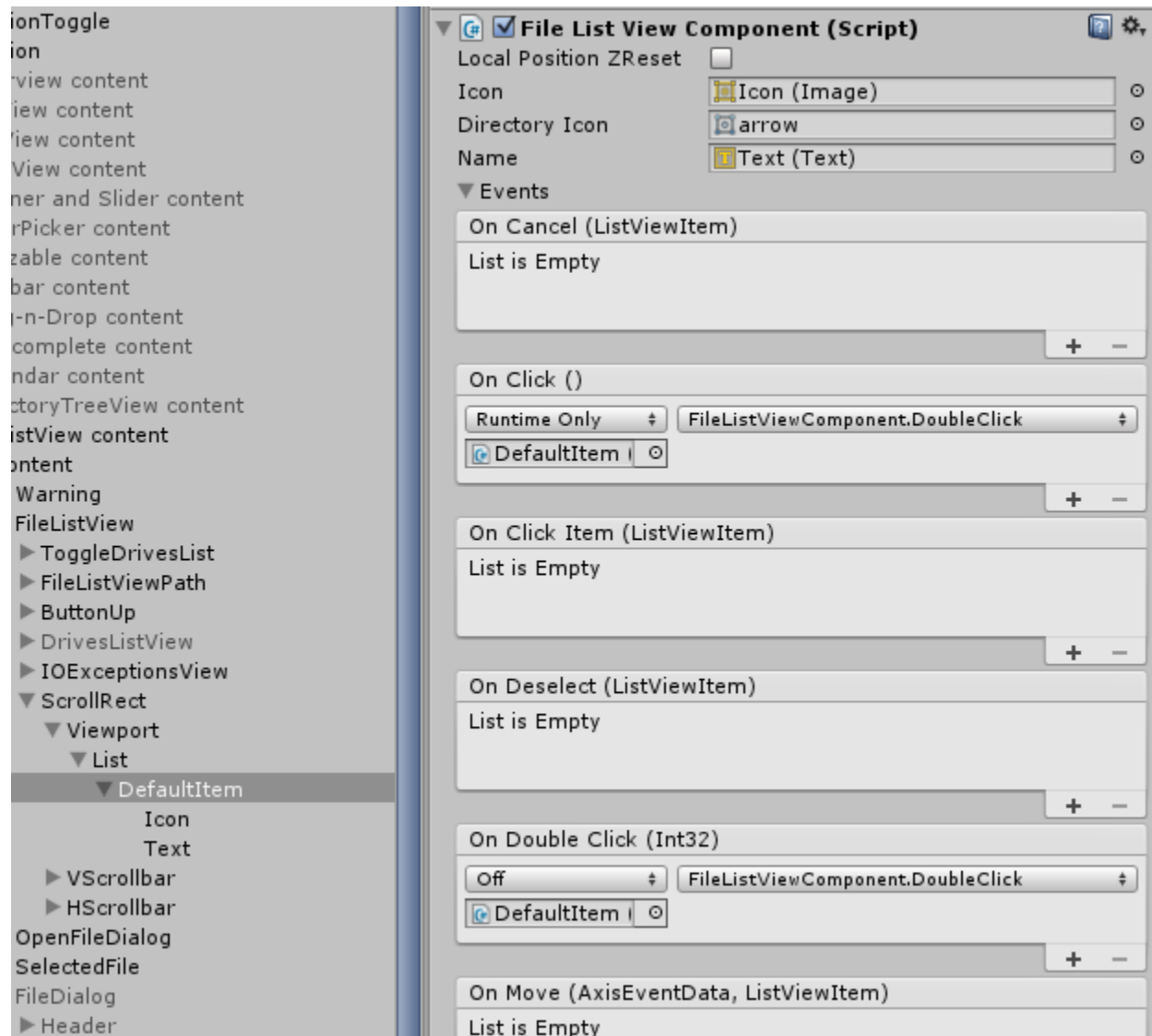

Custom Dialogs

You can create derived class with own methods and fields.

```
public class MyDialog : DialogCustom<MyDialog>
{
    // ...
}
```

3.4.3 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.

Code examples

```
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;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```

        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";
    }
}

```

3.4.4 FolderDialog

Options

- Directory Tree View `DirectoryTreeView`
`DirectoryTreeView` widget.
- 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
    {

```

(continues on next page)

(continued from previous page)

```
[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)
{
    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";
}
}
```

3.4.5 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.
- `Slide Up On Hide bool`
Start slide up animations after hide current notification. Turn it off if its managed with `HideAnimation`.
- `Notify Info NotifyInfoBase`
Component to display the notification message.
- `Close Button Button`
Button to close notification.
- `Buttons Container RectTransform`
Buttons container. If container not specified will be used parent of the button template.

Show() Method Parameters

All parameters are optional.

`message` also can be specified with `SetMessage ()` to use formatted strings.

- `message string`
Notification message.
Can be changed with `SetMessage ()` method.
- `customHideDelay float?`
Time before notification hidden or `hideAnimation` start running.
Can be changed with `HideDelay` field.
- `container Transform?`
Notifications container. Should have `Layout Group` component to display multiple notifications.
Can be changed with `SetContainer ()` method.

- `showAnimation Func<TNotification, IEnumerator>`
Show animation.
Can be changed with `ShowAnimation` field.
- `hideAnimation Func<TNotification, IEnumerator>`
Hide animation.
Can be changed with `HideAnimation` field.
- `slideUpOnHide bool?`
Start slide up animations after hide current notification.
Can be changed with `SlideUpOnHide` field.
- `sequenceType NotifySequence`
Add notification to sequence and display in order according specified `sequenceType`.
- `sequenceDelay float`
Time between previous notification was hidden and this will be displayed.
Can be changed with `SequenceDelay` field.
- `clearSequence bool`
Clear notifications sequence.
- `newUnscaledTime bool?`
Animations will use unscaled time.
- `content RectTransform`
Notification content.
Can be changed with `SetContent()`.
- `onReturn Action`
Action called when instance return to the cache.
Can be changed with `OnReturn` field.

Minimal code

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

Advanced

```
var notification = notificatetionTemplate.Clone();  
// show notification  
notification.Show(  
    // Show notification with following text  
    message: "Simple Notification.",  
    // Hide it after 4.5 seconds  
    customHideDelay = 4.5f,
```

(continues on next page)

(continued from previous page)

```
// Run specified animation on hide
hideAnimation = NotificationBase.AnimationCollapseVertical,
// without SlideUpOnHide
slideUpOnHide = false
);
```

Notification with Buttons

Notifications can have buttons with custom actions. Buttons callback receive notification instance and button index, return true to close notification; otherwise false.

```
[SerializeField]
protected Notify NotificationTemplate;

/// <summary>
/// Show notification.
/// </summary>
public void ShowNotify()
{
    var actions = new NotificationButton[]
    {
        new NotificationButton("Close", NotificationClose),
        new NotificationButton("Log", NotificationClick),
    };

    var instance = NotificationTemplate.Clone();
    instance.Show("Notification with buttons. Hide after 5 seconds.", customHideDelay: 5f);
    instance.SetButtons(actions);
}

bool NotificationClose(NotificationBase notification, int index)
{
    Debug.Log("close notification");
    return true;
}

bool NotificationClick(NotificationBase notification, int index)
{
    Debug.Log("click notification button");
    return false;
}
```

Default Hide Animations

Note: Hide Animation is coroutine that accepts NotificationBase 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.

Default Show Animations

Note: Show Animation is coroutine that accepts `NotificationBase` instance and play show animation for this instance. You can specify any custom coroutine.

- **ShowAnimationRotateHorizontal** Rotate notification on X axis.
- **ShowAnimationRotateVertical** Rotate notification on Y axis.
- **ShowAnimationCollapseHorizontal** Resize width of the notification.
- **ShowAnimationCollapseVertical** Resize height of the notification.
- **ShowAnimationSlideRight** Slide notification from right.
- **ShowAnimationSlideLeft** Slide notification from left.
- **ShowAnimationSlideUp** Slide notification from top.
- **ShowAnimationSlideDown** Slide notification from bottom.

Configurable Hide Animations

- **HideAnimationRotateBase**

Arguments:

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

- **HideAnimationCollapseBase**

Arguments:

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

- **HideAnimationSlideBase**

Arguments:

- **NotificationBase notification** 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.

```
NotificationTemplate.Clone().Show(
    "Notification message.",
    customHideDelay: 3f,
    hideAnimation: x => NotificationBase.HideAnimationSlideBase(x, true, -1f, 200f,
↳ true)
);
```

Configurable Show Animations

- **ShowAnimationRotateBase**

Arguments:

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

- **ShowAnimationCollapseBase**

Arguments:

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

- **ShowAnimationSlideBase**

Arguments:

- **NotificationBase notification** 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.

```
NotificationTemplate.Clone().Show(
    "Notification message.",
    customHideDelay: 3f,
    showAnimation: x => NotificationBase.ShowAnimationSlideBase(x, true, -1f, 200f,
↳ true)
);
```

Custom Notifications

You can create derived class with own methods.

```
public class MyNotify : NotificationCustom<MyNotify>
{
    // ...
}
```

3.4.6 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();

            // 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");
        }
    }
}
```

3.4.7 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.
- `CloseButton Button`
Button to close popup.

Show() Method Parameters

All parameters are optional.

`title` and `message` also can be specified with `SetInfo()` to use formatted strings.

- `title string`
Popup title.
Can be changed with `SetInfo()` method.
- `message string`
Popup message.
Can be changed with `SetInfo()` method.
- `position Vector3?`
Popup position.
Can be changed with `SetPosition()`.
- `icon Sprite`
Popup icon.
Can be changed with `SetInfo()` method.
- `modal bool`
Modal popup.
Can be changed with `SetModal()`.
- `modalSprite Sprite`
Background image for the modal popup.
Can be changed with `SetModal()`.
- `modalColor Color?`
Background color for the modal popup.

Can be changed with `SetModal()`.

- `canvas Canvas`

Canvas to display popup. Required if popup template is prefab.

Can be changed with `SetCanvas()`.

- `content RectTransform`

Dialog content. Can be used instead of the *message* and *icon*.

Can be changed with `SetContent()`.

- `onClose Action`

Action to run when dialog closed.

Can be changed with `OnClose` field.

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

3.5 Input

3.5.1 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.
- **ResetListViewSelection** `bool`
Deselect selected items in the DisplayListView.
- **AllowCancelOnDeselect** `Func<BaseEventData, AutocompleteCustom<TValue, TListViewComponent, TListView>, bool>`
Allow to cancel DisplayListView close on deselect event.

Events

- `OnOptionSelected` `UnityEvent`
- `OnOptionSelectedItem` `UnityEvent<TValue>`
- `OnItemNotFound` `UnityEvent<string>`
- `OnCancelInput` `UnityEvent`
- `OnSearchCompleted` `UnityEvent`

```
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;
        }
    }
}
```

3.5.2 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>`
- `OnDateClick UnityEvent<DateTime>`

```
namespace UIWidgets.Examples
{
    using UnityEngine;

    /// <summary>
```

(continues on next page)

(continued from previous page)

```
/// 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);

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

        // change culture (display days and months in english)
        Calendar.Culture = new System.Globalization.CultureInfo("en-US");

        // change culture (display days and months in french)
        Calendar.Culture = new System.Globalization.CultureInfo("fr-FR");

        // 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();
    }
}
```

3.5.3 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.
- `Value Max` `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

- `OnValueChanged` `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;
```

3.5.4 Circular Slider

Options

- `Interactable bool`
Is interactable?
- `Handle DragListener`
Handle to drag.
- `Arrow RectTransform`
Arrow.
- `Value int`
Current value.
- `Min Value int`
Minimal value.
- `Max Value ``int```
Maximal value.
- `Step int`
Value step.
- `Start Angle float`
Angle for the Min Value.

Events

- `OnValueChanged UnityEvent<int>`
- `OnChange UnityEvent`

Set value

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

Set value limits

```
slider.MinValue = 100;  
slider.MaxValue = 200;
```

3.5.5 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);  
}
```

3.5.6 DateTime

Nested widgets can be safely replaced with their analogs:

- time can be displayed with *Time24*, *Time12*, *TimeAnalog*, *TimeScroller*
- date can be displayed with *Calendar*, *DateScroller*

DateScroller Options

- Current Date Time As Default `bool`
 - Default Date Time `DateTime` (string in Inspector window)
 - Format `string`
Format to parse **Default Date Time**.
- Calendar `DateBase`
Widget to select date.
- Time `TimeBase`
Widget to select time.
- Is Scroll Blocks Used `bool`
Is Calendar and Time widgets are *scrollers*? Required for the styles support.

Events

- OnDateTimeChanged `UnityEvent<DateTime>`
The event raised when date changed.
Arguments: selected datetime.

3.5.7 DateScroller, DateTimeScroller, TimeScroller

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

Note: [DateTime Formats Strings](#)

DateScroller Options

- 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>`
The event raised when date changed.
Arguments: selected datetime.
- **OnDateClick** `UnityEvent<DateTime>`
The event raised when date setted or changed.
Arguments: selected datetime.

DateTimeScroller Options

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 Options

- 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>
 - The 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;

        /// <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);
        }
    }
}
```

Customization

ScrollBlock has `OnItemChanged(int index, ScrollBlockItem item)` event. You can subscribe to this event to customize items depending on index or value.

- selected item has `Index = 0`
- items before it have a negative index
- items after it have a positive index
- step of the index is 1.

```
public class ScrollBlockCustomization : MonoBehaviour
{
    [SerializeField]
    ScrollBlock YearsScrollBlock;

    protected void Start()
    {
        YearsScrollBlock.OnItemChanged += ItemChanged;
    }

    protected void OnDestroy()
    {
        if (YearsScrollBlock != null)
        {
            YearsScrollBlock.OnItemChanged -= ItemChanged;
        }
    }

    protected void ItemChanged(int index, ScrollBlockItem item)
    {
        item.Text.Bold = index == 0;
        item.Text.fontSize = index == 0 ? 20 : 14;
    }
}
```

3.5.8 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 than this.
- **Limit Max** `int/float`
Value cannot be more than 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

- **OnValueChanged** `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));
    }
}
```

3.5.9 Scale

Scale for the sliders: default *Slider*, *RangeSlider* (Disable Handle Overlay is not supported), *CenteredSlider*. To use add the appropriate *SliderScale* / *RangeSliderScale* / *CenteredSliderScale* component to the *Slider*, then create and specify *Scale* gameobject.

Options

- Container *RectTransform*
Marks container.
- Main Line *Image*
Main line.
- Show Current Value *bool*
Show marks for the current values.
- Current Mark Template *ScaleMarkTemplate*
Template for the current mark.
- Show Min Value *bool*
Show mark for the min value.
- Min Mark *ScaleMarkTemplate*
Minimum mark.
- Show Max Value *bool*

Maximum mark.

- `Scale Marks List<ScaleMark>`

Marks templates.

- `MarkValuesGenerator` `Action<float min, float max, float step, List<float> output>`

Fill output list with values where marks should be displayed.

ScaleMark

- `Step float`

Value difference between marks.

- `Template ScaleMarkTemplate`

Mark template.

SliderScale and RangeSliderScale Components

- `Scale Scale`

Scale gameobject.

- `Format string`

Format to display mark value.

- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>
- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-numeric-format-strings>

- `Formatter Func<float, string>`

Custom formatter to use instead of format string.

CenteredSliderScale Component

- `Scale Scale`

Scale gameobject.

- `Format string`

Format to display mark value.

- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/standard-numeric-format-strings>
- <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-numeric-format-strings>

- `Formatter Func<float, string>`

Custom formatter to use instead of format string.

- `Negative Step Rate float`

Multiplicator for marks at negative side of the scale.

- `Positive Step Rate float`

Multiplicator for marks at positive side of the scale.

3.5.10 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`
Value checked on every key down event.
Some value ranges cannot be processed correctly with `OnKeyDown` validation.
For example `2..10` because to enter `10` you need to enter `1` and `1` is not a valid value.
 - `OnEndInput`
Value checked when editing has ended.
- `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

- `onValueChangedInt` `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 minimum

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

Set value

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

Set step

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

Get value

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

3.5.11 Time

Time24 has 24-hour format.

Time12 has 12-hour format with AM/PM toggle.

Options

- **Interactable** `bool`
User can interact with Time widget.
- **Current Time As Default** `bool`
 - **Time** `TimeSpan` (string in Inspector window)
- **Time Min** `TimeSpan` (string in Inspector window)
Minimal selectable time.
- **Time Max** `TimeSpan` (string in Inspector window)
Maximum selectable time.
- **Input Hours Adapter** `InputFieldAdapter`
InputField for the hours.
- **Input Minutes Adapter** `InputFieldAdapter`
InputField for the minutes.
- **Input Seconds Adapter** `InputFieldAdapter`
InputField for the seconds.
- **Button Hours Increase** `ButtonAdvanced`
Button to increase hours.
- **Button Hours Decrease** `ButtonAdvanced`
Button to decrease hours.
- **Button Minutes Increase** `ButtonAdvanced`
Button to increase minutes.
- **Button Minutes Decrease** `ButtonAdvanced`
Button to decrease minutes.

- `Button Seconds Increase ButtonAdvanced`
Button to increase seconds.
- `Button Seconds Decrease ButtonAdvanced`
Button to decrease seconds.
- `Allow Hold bool`
Allow button hold after `Hold Start Delay` to increase/decrease time with each `Hold Change Delay`.
- `Hold Start Delay float`
Seconds from button press to start increase/decrease on hold.
- `Hold Change Delay float`
Seconds to single increase/decrease during hold.
- `AMPM Button Button`
Button to toggle AM/PM.
- `AMPM Text Adapter TextAdapter`
Text to display AM/PM.

Events

- `OnTimeChanged UnityEvent<TimeSpan>`
The event raised when time changed.
Arguments: selected time.

3.5.12 TimeAnalog

Options

- `Interactable bool`
User can interact with Time widget.
- `Current Time As Default bool`
 - `Time TimeSpan (string in Inspector window)`
- `Time Min TimeSpan (string in Inspector window)`
Minimal selectable time.
- `Time Max TimeSpan (string in Inspector window)`
Maximum selectable time.
- `Slider CircularSlider`
Time slider.
- `Step int`
Time step at minutes.
- `AMPM Button Button`

Button to toggle AM/PM.

- `AMPM Text TextAdapter`

Text to display AM/PM.

- `Hours Labels List<GameObject>`

Hours labels, required for the styles support.

Events

- `OnTimeChanged UnityEvent<TimeSpan>`

The event raised when time changed.

Arguments: selected time.

3.6 Miscellaneous

3.6.1 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();
```

3.6.2 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();
```

3.6.3 Simple Tooltip

Display the tooltip when cursor over gameobject or gameobject get focus.
`SimpleTooltip` cannot be used by multiple gameobjects unlike *Tooltip*.

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.

Events

- `OnShow UnityEvent`
- `OnHide UnityEvent`

3.6.4 Tooltip

Display the generic tooltip when cursor over gameobject or gameobject get focus.

Different gameobjects can use the same tooltip gameobject.

Tooltip for custom type can be created with [Widgets Generation](#).

Tooltip is automatically updated if custom types implements `IObservable` or `INotifyPropertyChanged` interface.

Using Tooltip

Add tooltip to gameobject:

```
Tooltip.Register(  
    TargetGO, // gameobject to add tooltip  
    Data, // data to display  
    new TooltipSettings(TooltipPosition.TopCenter, delay = 0.3f, unscaledTime = true)  
);
```

Remove tooltip:

```
Tooltip.Unregister(TargetGO);
```

Also tooltip can be added with Tooltip Viewer component.

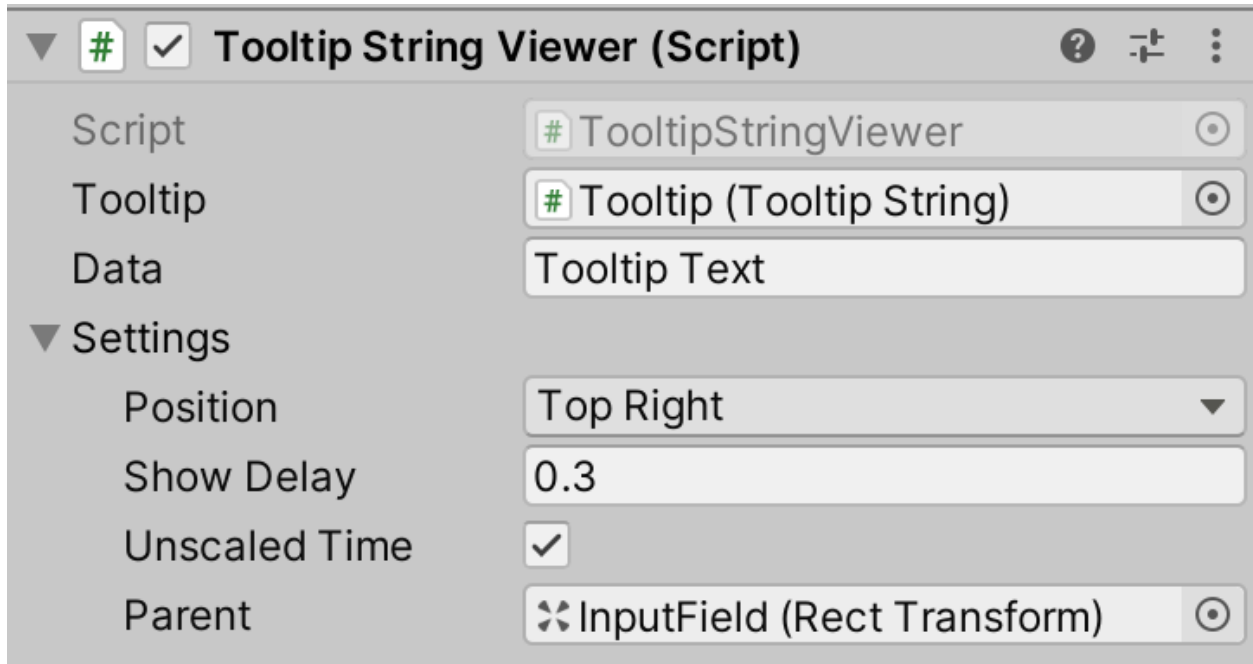
Destroy Tooltip Viewer component to remove tooltip.

Tooltip Fields and Properties

- `TData CurrentData`
- `GameObject CurrentTarget`

Tooltip Methods

- `Register(GameObject target, TData data, TooltipSettings settings)`
- `Unregister(GameObject target)`
- `Show(GameObject target)`
- `Hide()`
- `TData GetData(GameObject target)`
- `bool UpdateData(GameObject target, TData data)`



- `TooltipSettings GetSettings(GameObject target)`
- `UpdateSettings(GameObject target, TooltipSettings settings)`

Tooltip Events

- `OnShow UnityEvent<TTooltip, GameObject>`
- `OnHide UnityEvent<TTooltip, GameObject>`

Tooltip Settings

- **Position** `TooltipPosition`
Tooltip position relative to target gameobject.
- **Delay** `float`
Delay before tooltip displayed.
- **Parent** `RectTransform`
Tooltip parent.
- **UnscaledTime** `bool`
Delay specified in unscaled time.

TooltipPosition

- Top Left
- Top Center
- Top Right
- Middle Left
- Middle Center
- Middle Right
- Bottom Left
- Bottom Center
- Bottom Right

Tooltip Viewer Fields

- Tooltip TTooltip
- Data TData
Data to display.
- Settings TooltipSettings
Tooltip display settings.

Tooltip Code Example

```
namespace UIWidgets
{
    /// <summary>
    /// Tooltip string.
    /// </summary>
    public class TooltipString : Tooltip<string, TooltipString>
    {
        /// <summary>
        /// Text.
        /// </summary>
        public TextAdapter Text;

        /// <summary>
        /// Item.
        /// </summary>
        public string Item
        {
            get;
            protected set;
        }

        /// <inheritdoc/>
        protected override void SetData(string data)
        {
            Item = data;
        }
    }
}
```

(continues on next page)

(continued from previous page)

```
        UpdateView();
    }

    /// <inheritdoc/>
    protected override void UpdateView()
    {
        Text.text = Item;
    }
}
```

Tooltip Viewer Code Example

```
namespace UIWidgets
{
    /// <summary>
    /// TooltipString viewer.
    /// </summary>
    public class TooltipStringViewer : TooltipViewer<string, TooltipString>
    {
    }
}
```

COMPONENTS

4.1 Bring to Front

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

4.1.1 Options

- With Parents `bool`

Bring to front GameObject with parents GameObjects.

4.2 Single Line and Multi Line Connectors

Draw a line between current gameobject and specified targets.

4.2.1 SingleConnector Options

- Material `Material`
- Color `Color`
- Raycast Target `bool`
- Sprite `Sprite`
- Line `ConnectorLine`
- Builder `ILineBuilder`

Builder to draw custom lines.

4.2.2 MultipleConnector Options

- `Material` `Material`
- `Color` `Color`
- `Raycast Target` `bool`
- `Sprite` `Sprite`
- `Lines` `ObservableList<ConnectorLine>`
Lines list.
- `Builder` `ILineBuilder`
Builder to draw custom lines.

4.2.3 Connector Line

- `Target` `RectTransform`
- `Start` `ConnectorPosition`
Start point of the line: Top, Bottom, Left, Right, Center.
- `End` `ConnectorPosition`
End point of the line: Top, Bottom, Left, Right, Center.
- `Type` `ConnectorType`
Line type: Straight or Rectangular.
- `Thickness` `float`
Line thickness.
- `Margin` `float`
The minimum space from the border before the turn of the line. Supported only by Rectangular lines.

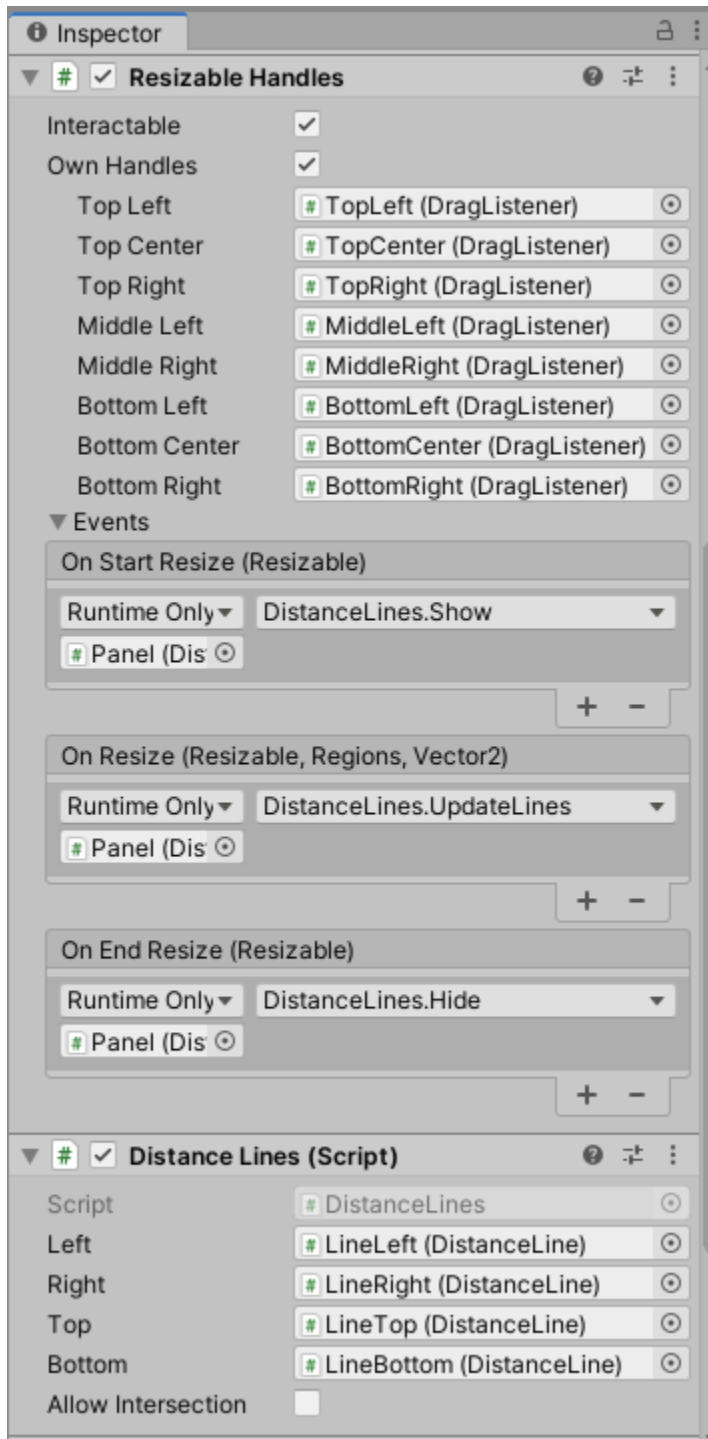
4.2.4 ILineBuilder

Interface to build connectors mesh with a single method:

```
int Build(ConnectorBase connector, RectTransform source, ConnectorLine line,
VertexHelper vh, int index)
```

4.3 Distance Lines

`Show()`, `UpdateLines()`, `Hide()` methods can be attached to appropriate events like `OnStartResize`, `OnResize`, `OnEndResize` for ease of use.



4.3.1 Options

- `Left DistanceLine` *optional*
Line from the left border of the parent.
- `Right DistanceLine` *optional*
Line from the right border of the parent.
- `Top DistanceLine` *optional*
Line from the top border of the parent.
- `Bottom DistanceLine` *optional*
Line from the bottom border of the parent.
- `Allow Intersection` `bool`
Allow lines intersection.
If disabled lines are drawn from parent border to the nearest Target border; otherwise from parent border to the same Target border.

4.4 Drag and Drop components

4.4.1 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 (`ListView`, `TreeView`) 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 gameobjects to disable drag-and-drop functionality.

4.4.2 How Drag&Drop works:

- there are two components: one process drag and another process drop
- Drop component implements `IDropSupport<TItem>` interface (it can implement multiple interfaces with different types). It is used to check if the dragged item can be received and process drop (for example add item to the `ListView`)
- Drag component is inherited from `DragSupport<TItem>` and attached to a game object with data to drag (for the `ListView`, it's attached to `ListView.DefaultItem`)
- when dragging, it looks for target with drop component that can accept a `TItem` and call `bool CanReceiveDrop(TItem data, PointerEventData eventData)` to check if target can receive dragged item
- on end drag called `Drop(TItem data, PointerEventData eventData)` for the drop component and then `Dropped(bool success)` for the drag component

4.4.3 Collections Drag Options

- `Allow Drag` `bool`
Allow drag.
- `Handle DragSupportHandle` *optional*
Custom handle to drag, if not specified will be dragged by current instance.
- `ListView TListView` *optional*
ListView instance.
Not available for TreeView.
- `DragInfo TComponent` *optional*
Component to display the dragged data.
- `DragInfo Offset` `Vector3`
Offset from the cursor position for the DragInfo.
- `Delete After Drop` `bool`
Delete item from collection after drop.
Not available for TreeView.
- `Cursors` `Cursors`
Custom cursors to show the allowed and denied drop states.

4.4.4 Collections Drop Options

- `Drop Position` `NearestType`
Drop position.
 - `Auto` insert dropped item to the nearest position.
 - `Before` insert dropped item before item under pointer.
 - `After` insert dropped item after item under pointer.
- `Drop Indicator` `ListViewDropIndicator`
Indicator to display position where dropped item will be inserted.
- `Delete Node After Drop` `bool`
Delete dropped node from TreeView.
Not available for TreeView.
- `Receive Items` `bool`
Receive dropped items.
- `Receive Nodes` `bool`
Receive dropped nodes.

4.4.5 TreeView Drop Options

- Drop Position `NearestType`
Drop position.
 - Auto insert dropped item to the nearest position
 - Before insert dropped item before item under pointer
 - After insert dropped item after item under pointer
- Drop Indicator `ListViewDropIndicator`
Indicator to display position where dropped item will be inserted.
- Receive Items `bool`
Receive dropped items.
- Receive Nodes `bool`
Receive dropped nodes.

4.4.6 TreeView Node Drop Options

- Drop Indicator `ListViewDropIndicator`
Indicator to display position where dropped item will be inserted.
- Delete Node After Drop `bool`
Delete dropped node from TreeView.
- Receive Items `bool`
Receive dropped items.
- Reorder Area `float`
Distance in percent of height from border to add dropped node before/after instead of drop as sub-node. Allowed value range is 0f..0.5f

4.4.7 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.

```

(continues on next page)

(continued from previous page)

```

/// </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;
    }
}

```

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]

```

(continues on next page)

(continued from previous page)

```
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()
{
    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);
}
}
```

4.4.8 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 allows to accept only nodes with names ends with *I*.

```
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.
        /// </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>
    }
}
```

(continues on next page)

(continued from previous page)

```

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

#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
}

```


4.4.9 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>
    [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);
            }

            ResetCursor();
        }
    }
}
```

4.4.10 Adding limitations to the Drop component

In this example, ListViewIcons will receive drag-and-drop data only if DataSource.Count less than MaxQuantity.

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

    public class ListViewIconsDropSupportLimitedQuantity : ListViewIconsDropSupport
    {
        [SerializeField]
        public int MaxQuantity = 10;

        public override bool CanReceiveDrop(ListViewIconsItemDescription data, ↩
        ⇨ PointerEventData eventData)
        {
            // disable drop if quantity limit reached
            if ((MaxQuantity >= 0) && (ListView.DataSource.Count >= MaxQuantity))
            {
                return false;
            }

            return base.CanReceiveDrop(data, eventData);
        }
    }
}
```

4.5 Draggable

Dragging gameobject.

4.5.1 Options

- Interactable bool
Allow interaction.
- 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 the parent.
 - After Drag: does not allow drag outside the parent, applied after drag ended.
- Curve AnimationCurve

Animation curve used to animate applied **After Drag** restriction.

- `Unscaled Time bool`
Run animation with unscaled time.
- `Snap Grids List<SnapGridBase>`
Allow snapping the `RectTransform` position to the nearest line.
See *SnapGrid* and *SnapLines*.
- `Snap Distance Vector2`
Maximum distance to lines where snapping is available.

4.5.2 Properties

- `Target RectTransform`
Target to drag; the self is by default.

4.5.3 Events

- `OnStartDrag UnityEvent<Draggable>`
- `OnDrag UnityEvent<Draggable>`
- `OnEndDrag UnityEvent<Draggable>`

4.6 EasyLayout

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

4.6.1 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`: change children size in range from minimal to preferred to fit container.
- `Set Preferred and Fit Container`: set children size to preferred, then increase size proportionally Flexible Width to fit parent width if required.
- `Shrink On Overflow`: decrease elements width if summary width more than parent width including margin.

- `Children Height` `ChildrenSize`

Similar to `Children Width`

4.6.2 Events

- `Settings Changed` `UnityEvent`

Event, raised after any setting was changed.

4.7 Groupable

Allows to select a group of the gameobjects; and then resize, rotate, align all of them simultaneously.

Can select only elements with the same parent as the Groupable component.

Shared components settings between Groupable and the selected elements:

- `Resizable.KeepAspectRatio`
- `Rotatable.LimitRotation`
- `Rotatable.AngleMin`
- `Rotatable.AngleMax`
- `Rotatable.AngleStep`

4.7.1 Options

- `Interactable` `bool`
Allow interaction.
- `Highlight Template` `RectTransform` *optional*
Template to highlight selected gameobjects.
- `Selection Mode` `Groupable.Mode`
Selection mode.
 - `Contains`
Selects only gameobjects fully inside the selection area.
 - `Overlaps`
Selects gameobjects inside the selection area or partially overlaps the selection area.
- `Group Rotation` `bool`
If enabled selected gameobjects will be rotated as part of the group; otherwise each separately.

4.7.2 Events

- `OnStartSelection` `UnityEvent<Groupable>`
- `OnSelection` `UnityEvent<Groupable>`
- `OnEndSelection` `UnityEvent<Groupable>`

4.8 Layout Switcher

Allows creating different layouts with the same `GameObjects` for different screen sizes and aspect ratios. Used when anchors, pivots and layout groups not enough to create a layout with different aspect ratios support.

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

4.8.1 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 the specified one).

4.8.2 Events

- `LayoutChanged` `UnityEvent<UILayout>`

4.9 Lightbox

Lightbox is a component used to display overlay image.

4.10 ListViewAutoResize

Auto-resizes `ListView` or `TileView` according to item counts until specified maximum size reached. The component implements the `ILayoutElement` interface, so it can be used with `LayoutGroup`.

4.10.1 Options

- `MaxSize` `float`
Maximum size.
- `UpdateRectTransform` `size`
Set `RectTransform` size.

4.11 Object Sliding

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

4.11.1 Options

- `Interactable` `bool`
Allow interaction.
- `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.

4.11.2 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.

4.12 Pinchable

Allows drag/resize/rotate gameobject with multi-touches.

4.12.1 Options

- `Interactable` `bool`
Allows users interaction.
- `AllowDrag` `bool`
Allows drag.
- `AllowResize` `bool`
Allows resize.
- `AllowRotate` `bool`
Allows rotation.

4.12.2 Events

- `OnStartPinch` `UnityEvent<Pinchable>`
- `OnPinch` `UnityEvent<Pinchable>`
- `OnEndPinch` `UnityEvent<Pinchable>`

4.13 Resizable

Allows resizing gameobject by size or scale.

4.13.1 Options

- `Interactable` `bool`
Allow users to change the size of the `GameObject`.
- `Resize Directions` `Resizable.Directions`
Allowed resizing directions.
- `Type` `ResizeType`
Resize type.
 - `Size`
Resize by changing size of the gameobject.
 - `Scale`
Resize by changing scale of the gameobject.
- `Include Corners` `bool`
Allow resize when cursor in the one of the corners. Should be disabled to use together with *Rotatable* component.
- `Integer Size` `bool`
If enabled size is rounded to the integer number. Reason: size can be float number if gameobject is rotated.
- `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 in points, for the `Scale` type limits is checked against *width * scale.x* and *height * scale.y*.
- `Max Size` `Vector2`

Maximum size in points, for the `Scale` type limits is checked against *width * scale.x* and *height * scale.y*.

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.

- `Cursors` `Cursors`

Custom cursors to the allowed resize state.

- `Snap Grids` `List<SnapGridBase>`

Allow snapping the `RectTransform` position to the nearest line.

See [SnapGrid](#) and [SnapLines](#).

- `Snap Distance` `Vector2`

Maximum distance to lines where snapping is available.

4.13.2 Events

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

4.13.3 Properties

- `Target RectTransform`

Target to resize; the self is by default.

4.13.4 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.

4.14 Resizable Handles

Helper component with handles to resize for the *Resizable*.

4.14.1 Options

- `Interactable` `bool`
Allow users to change the size of the `GameObject`.
- `Own Handles` `bool`
If enabled you can specify your own handles for the current component.
If disabled you can specify `Handles Source` for current component, this allows you to create a single set of handles instead of duplicate them for each component.
Handles should be acquired with `GetSourceHandles()` and returned with `ReleaseSourceHandles()` functions.
- `Handles Source` `ResizableHandles`
Handles source to use if `Own Handles` disabled.
- `Top Left DragListener` *optional*
Top left handle.
- `Top Center DragListener` *optional*
Top center handle.
- `Top Right DragListener` *optional*
Top right handle.
- `Middle Left DragListener` *optional*
Middle left handle.
- `Middle Right DragListener` *optional*
Middle right handle.
- `Bottom Left DragListener` *optional*
Bottom left handle.
- `Bottom Center DragListener` *optional*
Bottom center handle.
- `Bottom Right DragListener` *optional*
Bottom right handle.

4.14.2 Events

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

4.15 Rotatable

Allows rotating gameobject around its pivot.

4.15.1 Options

- `Interactable` `bool`
Allow users to change the rotation of the GameObject.
- `Rotate Directions` `Rotatable.Directions`
Allowed corners to apply the rotation.
- `Active Region` `float`
Distance from border where rotation allowed.
- `Limit Rotation` `bool`
Allows rotating objects only with the specified angles range.
 - `Angle Min` `float`
Allowed value is in range [-180..180].
 - `Angle Max` `float`
Allowed value is in range [-180..180].
- `Angle step` `float`
Allowed value is in range [0..180). Set 0 to disable.
- `Cursors` `Cursors`
Custom cursors to show the allowed rotation state.

4.15.2 Events

- `OnStartRotate` `UnityEvent<Rotatable>`
- `OnRotate` `UnityEvent<Rotatable>`
- `OnEndRotate` `UnityEvent<Rotatable>`

4.15.3 Properties

- `Target RectTransform`
Target to rotate; the self is by default.

4.16 Rotatable Handle

Helper component with handle to rotate for the *Rotatable*.

4.16.1 Options

- `Interactable bool`
Allow users to change the rotation of the `GameObject`.
- `Own Handle bool`
If enabled you can specify your own handle for the current component.
If disabled you can specify Handle Source for current component, this allows you to create a single handle instead of duplicate it for each component.
Handle should be acquired with `GetSourceHandle()` and returned with `ReleaseSourceHandle()` functions.
- `Handle Source RotatableHandle`
Handle source to use if `Own Handle` disabled.
- `Handle DragListener optional`
Handle.

4.16.2 Events

- `OnStartRotate UnityEvent<Rotatable>`
- `OnRotate UnityEvent<Rotatable>`
- `OnEndRotate UnityEvent<Rotatable>`

4.17 Scrollbar Min Size

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

4.17.1 Options

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

4.18 ScrollRect Events

Provide pull events for the ScrollRect.

4.18.1 Options

- `Thresholds PullThreshold`
Separate thresholds values for each pull direction to raise events.

4.18.2 Events

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

4.19 ScrollRect Footer

Footer for the ScrollRect; visible when scrolled to the bottom.

4.19.1 Options

- `ScrollRect ScrollRect`
ScrollRect.
- `Block RectTransform`
Actual footer block.
- `IsHorizontal bool`
ScrollRect direction.

- `DisplayType ScrollRectHeaderType`

Display type.

- `Reveal`

Show block when scrolled to the bottom and hide on scroll up.

- `Resize`

Resize block from current size at the bottom to the minimal size on scroll up.

- `MinSize float`

Minimal size of the footer.

4.20 ScrollRect Header

Header for the `ScrollRect`; visible when scrolled to the top.

4.20.1 Options

- `ScrollRect ScrollRect`

`ScrollRect`.

- `Block RectTransform`

Actual header block.

- `IsHorizontal bool`

`ScrollRect` direction.

- `DisplayType ScrollRectHeaderType`

Display type.

- `Reveal`

Show block when scrolled to the top and hide on scroll down.

- `Resize`

Resize block from current size at the top to the minimal size on scroll down.

- `MinSize float`

Minimal size of the header.

4.21 Selectable Helper

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

4.22 SnapGrid

Allow snapping the `RectTransform` position or size to the nearest line. Does not work on its own, should be used together with *Resizable*, *Draggable*, or `DropRectTransform`

4.22.1 Options

- `Snap Border Inside SnapGridBase.Border`
Allow snapping to the inner side of the border.
- `Snap Border Outside SnapGridBase.Border`
Allow snapping to the outer side of the border.
- `Padding Vector2`
Padding from borders.
- `Step Vector2`
Size of the grid cells.
- `Spacing Vector`
Empty space between cells.
- `Snap To Spacing bool`
Allow spacing to inner sides of the spacing lines.

4.22.2 SnapGridBase.Border

- `Left bool`
- `Right bool`
- `Top bool`
- `Boottom bool`

4.22.3 Events

- `OnLinesChanged UnityEvent`
Raised when lines changed.

4.23 SnapLines

Allow snapping the `RectTransform` position or size to the nearest line. Does not work on its own, should be used together with *Resizable*, *Draggable*, or `DropRectTransform`

4.23.1 Options

- `Snap Border Inside SnapGridBase.Border`
Allow snapping to the inner side of the border.
- `Snap Border Outside SnapGridBase.Border`
Allow snapping to the outer side of the border.
- `Lines X ObservableList<SnapGridBase.LineX>`
Lines on X axis.
- `Lines Y ObservableList<SnapGridBase.LineY>`
Lines on Y axis.

4.23.2 SnapGridBase.LineX

- `X float`
Position on X axis.
- `Snap Left bool`
Allow snapping by left side of the `RectTranform` (right of the line).
- `Snap Right bool`
Allow snapping by right side of the `RectTranform` (left of the line).

4.23.3 SnapGridBase.LineY

- `Y float`
Position on Y axis.
- `Snap Top bool`
Allow snapping by top side of the `RectTranform` (bottom of the line).
- `Snap Bottom bool`
Allow snapping by bottom side of the `RectTranform` (top of the line).

4.23.4 Events

- `OnLinesChanged UnityEvent`
Raised when lines changed.

4.24 Splitter

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

4.24.1 Options

- `Interactable` `bool`
Allow users to interact with the splitter.
- `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.

4.24.2 Events

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

4.25 Switch Group

Same as `Toggle Group`, but for the `Switch` widget.

4.25.1 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.

4.26 Table Header

Used with `ListView` on table mode. Allows to resize and reorder columns.

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

4.26.1 Options

- Interactable `bool`

Allow interaction.

- 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.

- Cursors `Cursors`

Custom cursors to show the allowed column resize state, allowed, and denied drop states.

4.26.2 Cet Current Columns Order

```
// index is the original position of the column  
// value is the current position of the column  
var order = tableHeader.GetColumnsOrder();
```

4.26.3 Change Columns Order

```
var order = new List<int>(2, 1, 0);  
tableHeader.SetColumnsOrder(order);
```

4.26.4 Restore Original Columns Order

```
tableHeader.RestoreColumnsOrder();
```

4.26.5 Disable Column

```
var column = 0;  
tableHeader.ColumnDisable(column);
```

4.26.6 Enable Column

```
var column = 0;  
tableHeader.ColumnEnable(column);
```

Add/Remove Column at Runtime

```
var order = tableHeader.GetColumnsOrder();  
tableHeader.RestoreColumnsOrder();  
  
// add new column to the header  
new_column_header.SetParent(tableHeader.transform);  
new_column_header.SetSiblingIndex(...);  
order.Insert(..., ...);  
  
// or remove column  
Destroy(tableHeader.transform.GetChild(index));  
order.RemoveAt(...);  
tableHeader.Refresh()  
  
// new DefaultItem with another set of cells  
listView.DefaultItem = newDefaultItem;  
  
// modify order with new column index or deleted column index and set it back  
tableHeader.SetColumnsOrder(order);
```

4.27 TreeView data source

Used in editor mode, allow to edit TreeView nodes.

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

5.1 Flare Effect

Shader based. Create material with *Custom / New UI Widgets / UIFlareTransparent* or *Custom / New UI Widgets / UIFlareGlobal* shader and set it to `Image` component.

Note: *UIFlareGlobal* works only in Render Mode = Screen Space - Overlay.

5.1.1 Options

- Flare Color `Color`
Color of the flare.
- Flare Size `float`
Size of the flare in range [0..1].
- Flare Speed `float`
Speed of the flare relative to the image size: how much times it will move from left to right in one second.
- Flare Delay `float`
Delay between appearances of the flare. Examples:
 - 0 - no delay
 - 1 - delay is travel time from left to right
 - 2 - delay is doubled travel time from left to right

5.2 Lines Drawer

Draw straight lines on X or Y axis.

Note: Requires enabled `TexCoord1` at `Canvas.AdditionalShaderChannells`.

5.2.1 Options

- `Line Color Color`
Line color.
- `Line Thickness float`
Line thickness.
- `Transparent Background bool`
Change color of the `Graphic` component to transparent.
- `LinesX ObservableList<float>`
Position on X axis where vertical line should be drawn in range `[0..width]`, 0 at left.
- `LinesY ObservableList<float>`
Position on Y axis where horizontal line should be drawn in range `[0..height]`, 0 at bottom.

5.3 Ring Effect

Draw ring or circle. Shader based effect. To use add `RingEffect` component to gameobject with `Graphic` component.

Note: Requires enabled `TexCoord1` at `Canvas.AdditionalShaderChannells`.

5.3.1 Options

- `Ring Color Color`
Color of the ring.
- `Thickness float`
Ring thickness.
- `Padding float`
Padding from border.
- `Transparent Background bool`
Change color of the `Graphic` component to transparent.

5.4 Ripple Effect

Draw ripples on the click position. Maximum 10 ripples per gameobject. Shader based effect. To use add `RippleEffect` component to gameobject with `Graphic` component.

Note: Requires enabled `TexCoord1` at `Canvas.AdditionalShaderChannells`.

5.4.1 Options

- `Start Color Color`
Initial color of the ripple.
- `End Color Color`
End color of the ripple.
- `Speed float`
Growth speed of the ripple.
- `Max Size float`
Maximum size of the ripple in range [0..1].

5.5 Snap Grid Drawer

Draw straight lines on X or Y axis, lines position provided by `SnapGrid` or `SnapLines` components.

Note: Requires enabled `TexCoord1` at `Canvas.AdditionalShaderChannells`.

5.5.1 Options

- `Line Color Color`
Line color.
- `Line Thickness float`
Line thickness.
- `Transparent Background bool`
Change color of the `Graphic` component to transparent.
- `Include Borders bool`
Draw borders if borders enabled in `SnapGrid` or `SnapLines` components.

5.6 Tsunami Effect

`RectTransform` size is changed from `MinSize` to `MaxSize` with distance from gameobject to the pointer. To use add `TsunamiEffect` component.

5.6.1 Options

- `MinSize Vector2`
Minimal size of the component.
- `MaxSize Vector2`
Maximum size of the component.
- `Distance float`
Effect distance.

SHADERS

6.1 Gradient Shaders

Those are shaders used by ColorPicker. Use ColorHSV.ShaderColor to set colors for the HSV shaders.

- UIGradientHLineHSV

The horizontal gradient between the two colors. HSV color model.

- UIGradientHLineRGB

The horizontal gradient between the two colors. RGB color model.

- UIGradientVLineHSV

The vertical gradient between the two colors. HSV color model.

- UIGradientVLineRGB

The vertical gradient between the two colors. RGB color model.

- UIGradientPlaneHSV

The plane gradient between the four colors, each color in the own corner. HSV color model.

- UIGradientPlaneRGB

The plane gradient between the four colors, each color in the own corner. RGB color model.

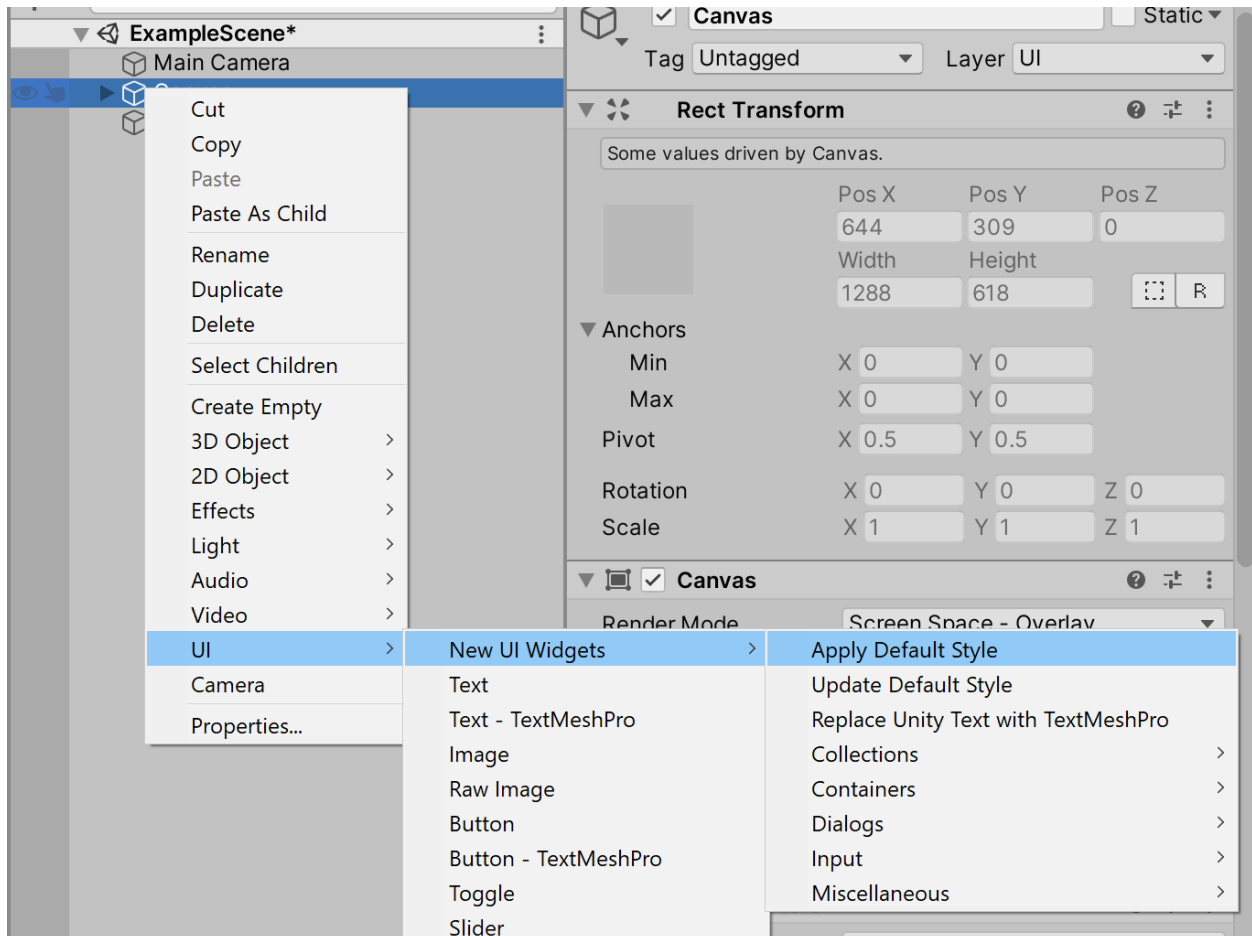
STYLES (SKINS)

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*.



You can change widgets settings and then save them to the style with *UI / New UI Widgets / Update 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: For the style support for the nested widgets (for example, Switch or Spinner in the `ListView.DefaultItem`), you should add the `StyleSupportAny` component to the gameobject of the parent widget and specify nested widgets at the `Objects` field.

7.1 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 children gameobjects was processed; otherwise false.
    }

    public virtual bool GetStyle(Style style)
    {
        style.Collections.MainBackground.GetFrom(GetComponent<Image>());

        return true; // true if children gameobjects was processed; otherwise false.
    }
}
```

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

INTEGRATION

8.1 Assembly Definitions

The package does not have assembly definitions, but you can add them with all required references.

Why no assembly definitions by default: - supported third-party packages do not use assembly definitions, so they cannot be referenced to be used in a separate assembly - version defines works only for Unity packages, so only direct references to assembly definitions are available - changes in the .asmdef files are lost with the package update

8.2 Cursor

- Cursors

A scriptable object that contains a list of different cursors types.

Asset can be created with the *Context menu / Create / New UI Widgets / Cursors*.

- UICursor

Static class, wrapper for the `Cursor.SetCursor()` to avoid cursor conflicts between different widgets and components. For example `:doc:/components/resizable`` component should not change the cursor if currently controlled by the `:doc:/components/drag-and-drop`` component.

- CursorDPISelector

This component selects the most appropriate `Cursors` asset by `Screen.dpi` from the available cursors list and sets it as default cursors (`UICursor.Cursors`).

Components like `Resizable` have the `Cursors` field, so they can have custom cursors to use instead of the default one.

8.2.1 Cursors Fields

- `Default Cursors.Cursor`

Default cursor.

- `Allowed Cursors.Cursor`

Cursor for the allowed actions.

- `Denied Cursors.Cursor`

Cursor for the not allowed actions

- `NorthSouthArrow Cursors.Cursor`

North <-> South arrow.

- `EastWestArrow Cursors.Cursor`

East <-> West arrow.

- `NorthEastSouthWestArrow Cursors.Cursor`

NorthEast <-> SouthWest arrow.

- `NorthWestSouthEastArrow Cursors.Cursor`

NorthWest <-> SouthEast arrow.

- `NorthWestRotateArrow Cursors.Cursor`

North <-> West arrow.

- `NorthEastRotateArrow Cursors.Cursor`

North <-> East arrow.

- `SouthWestRotateArrow Cursors.Cursor`

South <-> West arrow.

- `SouthEastRotateArrow Cursors.Cursor`

South <-> East arrow.

8.2.2 Cursors.Cursor Fields

- `Texture Texture2D`

Cursor texture.

- `Hotspot Vector2`

Cursor hot spot.

8.2.3 UICursor Static Fields

- `Cursors Cursors`

Different default cursors.

- `CanSet Func<Component, bool>`

Is can the specified component set the cursor?

`true` if cursor does not have an owner or the owner is the same.

- `Set Action<Component, Cursors.Cursor>`

Set the cursor and current owner.

The cursor will be changed only if `CanSet (owner)` returns `true`.

- `Reset Action<Component>`

Reset cursor and its owner to the default.

8.3 Localization

Most widgets have localization support, exceptions are:

- AutocompleteString
- ComboboxString
- ListViewString

Integration with custom localization system can done with `UIWidgets.l10n.Localization` class.

Example for the **I2 Localization**:

```
protected virtual void Start()
{
    Localization.GetTranslation = I2Translation;
    Localization.GetCountryCode = I2CountryCode; // used by Calendar and similar_
    ↪widgets
    I2.Loc.LocalizationManager.OnLocalizeEvent += Localization.LocaleChanged;
}

public static string I2Translation(string input)
{
    var result = I2.Loc.LocalizationManager.GetTranslation(input);
    if (result == null)
    {
        return input;
    }

    return result;
}

public static string I2CountryCode()
{
    return I2.Loc.LocalizationManager.CurrentLanguageCode;
}
```

8.3.1 Dialog, Popup Localization

Dialog and Popup widgets requires enabled `LocalizationSupport` in `DialogInfoBase` component.

Formatted strings can be used with the `SetInfo` method:

```
public void Dialog()
{
    var actions = new DialogButton[]
    {
        new DialogButton("OK", DialogClose),
        new DialogButton("Cancel", DialogClose),
    };

    var instance = DialogTemplate.Clone();
    instance.DialogInfo.LocalizationSupport = true;
    instance.Show(
        buttons: actions,
        focusButton: "Close",
        modal: false,
```

(continues on next page)

(continued from previous page)

```
        onCancel: DialogClose);
    instance.SetInfo("Welcome, {0}", new object[] { "username", }, "Value 1: {0}\nValue 2: {1}", new object[] { "argument 1", "argument 2" });
}

bool DialogClose(int buttonIndex)
{
    return true;
}
```

8.3.2 Notify Localization

Notify widget requires enabled `LocalizationSupport` in `NotifyInfoBase` component.

Formatted strings can be used with the `SetMessage` method:

```
public void NotificationFormatted()
{
    var instance = NotificationTemplate.Clone();

    instance.NotifyInfo.LocalizationSupport = true;
    instance.Show(customHideDelay: 0f);
    instance.SetMessage("Welcome, {0} {1}", "FirstName", "LastName");
}
```

8.3.3 Generated Widgets

The easiest way to add localization support is to implement property returning a localized string in the data class. Widgets are automatically updated on locale changes.

```
public class Item
{
    public string LocalizedName
    {
        get
        {
            return I2.Loc.LocalizationManager.GetTranslation(Name);
        }
    }

    public string Name;
}
```

8.4 String Comparison and Culture

All widgets and components use `UtilitiesCompare` to compare strings. You can change comparison settings with the following fields:

- `UtilitiesCompare.Culture` `CultureInfo`
Culture used to compare strings, by default used `CultureInfo.InvariantCulture`.
- `UtilitiesCompare.OptionsCaseSensitive` `CompareOptions`
Options to compare strings with case sensitive.
- `UtilitiesCompare.OptionsCaseIgnore` `CompareOptions`
Options to compare strings with case ignore.

8.5 Timer and Animations

All widgets and components with animations have option `UnscaledTime`. The animation will be run with `Time.unscaledTime` if this option enabled.

You can also specify own timer instead of the default one. To do this, you need to set the following fields:

- `UtilitiesTime.GetTime` `Func<bool, float>`
Accept the time type, `true` if unscaled time. Returns the current time in seconds since the start of the game.
- `UtilitiesTime.GetDeltaTime` `Func<float>`
Accept the time type, `true` if unscaled time. Returns the current time in seconds since the last frame.

8.6 Unity Update Methods Replacement

Update manager is used to optimize the performance of `Update()`, `LateUpdate()`, `FixedUpdate()` calls and same calls required only for one frame.

You can replace the update manager with a custom one which implements `IUpdaterProxy` interface:

```
Updater.Proxy = custom_updater;
```

8.6.1 Interfaces to Replace Unity Update Methods

- `IUpdatable` replace `Update()` method
Methods:
– `RunUpdate()`
- `ILateUpdatable` replace `LateUpdate()` method
Methods:
– `RunLateUpdate()`
- `IFixedUpdatable` replace `FixedUpdate()` method

Methods:

- `RunFixedUpdate()`

SUPPORTED PACKAGES

9.1 Data Bind for Unity Support

You can enable [Data Bind for Unity](#) support with *Project Settings... / New UI Widgets / DataBind support / Enable*. 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 / DataBind Support / Disable*.



Note: Support is enabled only to installed platforms. Platforms that were added after it requires enabling support again.

9.2 I2 Localization Support

You can enable **I2 Localization** support with *Project Settings... / New UI Widgets / I2 Localization Support / Enable*. If **I2 Localization** not installed option will not be available.

Localization Support Details

Disable support with *Project Settings... / New UI Widgets / I2 Localization Support / Disable*.



Note: Support is enabled only to installed platforms. Platforms that were added after it requires enabling support again.

9.3 TextMeshPro Support



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

After enabling support:

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

You can disable support the same way with *Project Settings... / New UI Widgets / TextMeshPro Support / Disable*.

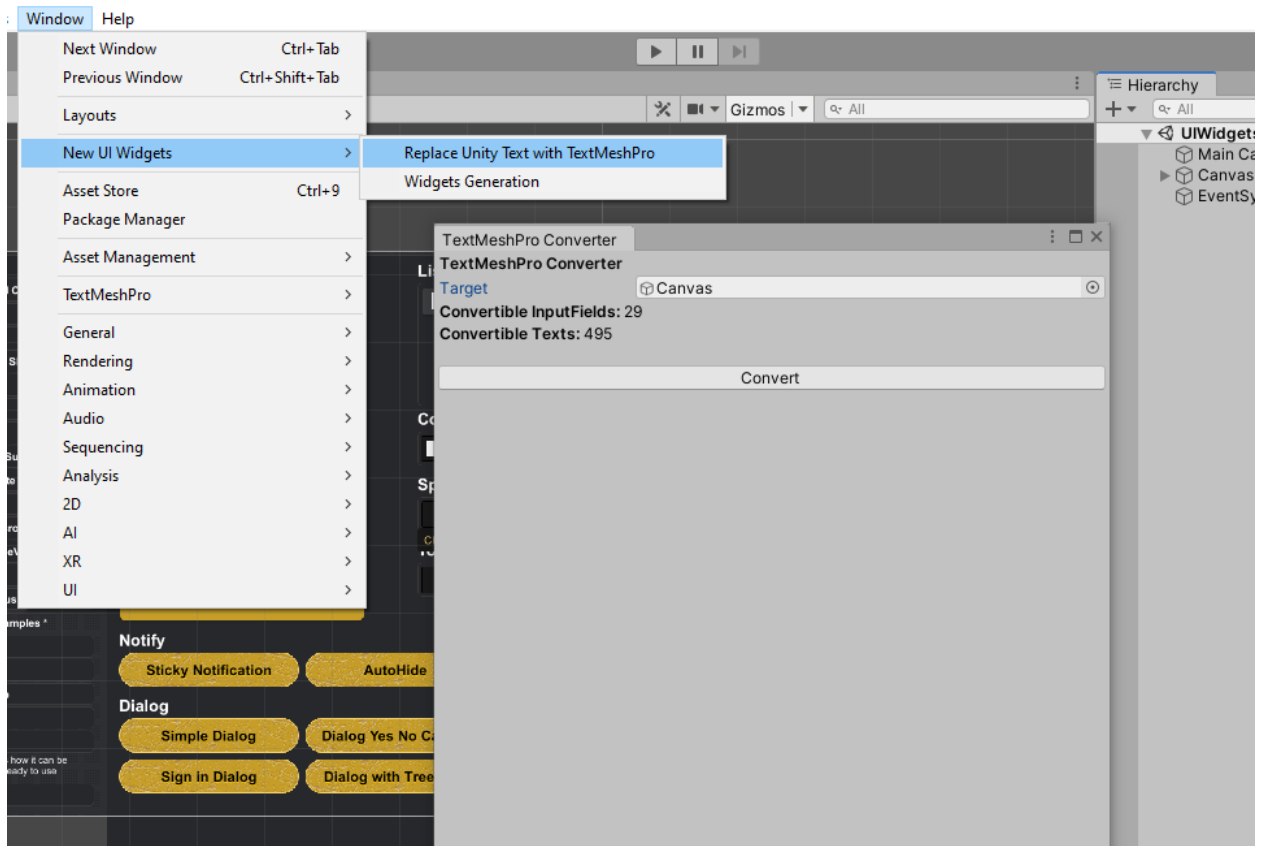
Note: Support is enabled only to installed platforms. Platforms that were added after it requires enabling support again.

9.3.1 Details

TextMeshPro support is enabled by adding `UIWIDGETS_TMPRO_SUPPORT` directive to the *Scripting Define Symbols* in the *Player Settings* and forced scripts recompilation.

Starting with version *1.12* TextMeshPro support is done with `TextAdapter` and `InputFieldAdapter` components. Those are adapters for the actual Unity and TextMeshPro components. This allows replacing Text components without any code changes.

9.4 TextMeshPro Converter



This is a tool to convert existing UI at the scene from default `Text` and `InputField` to the **TextMeshPro** equivalent components.

Converter available with the context menu *UI/New UI Widgets/Replace Unity Text with TextMeshPro* or with *Window/New UI Widgets/Replace Unity Text with TextMeshPro*.

Scripts references to `Text` and `InputField` components will be automatically replaced if type of reference is common base type like `Graphic` or `MonoBehaviour`; otherwise those components will not be converted.

Limitations:

- If you have any scripts with the serialized fields of type `Text` or `InputField` with specified components, then those components will not be converted.

```
[SerializeField]
Text Name; // cannot be converted

[SerializeField]
Graphic SecondName; // can be converted

[SerializeField]
TextAdapter ThirdName; // can be converted
```

Solutions:

- manually change type to the `TextAdapter` or `InputFieldAdapter` and add the corresponding component to the referenced `GameObject`
- modify code to automatically replace components with adapters

9.4.1 Modify Code to Adapters

Original script:

```
class SomeComponent : MonoBehaviour
{
    [SerializeField]
    Text Name;

    public void SomeMethod()
    {
        Name.text = "value";
    }
}
```

Modification:

```
class SomeComponent : MonoBehaviour, IUpgradeable
{
    [SerializeField]
    [System.Obsolete("Replaced with NameAdapter.")]
    Text Name;

    [SerializeField]
    TextAdapter NameAdapter;

    public void SomeMethod()
    {
        NameAdapter.text = "value";
    }

    public virtual void Upgrade()
    {
        Utilities.GetOrAddComponent(Name, ref NameAdapter);
    }

    #if UNITY_EDITOR
    protected virtual void OnValidate()
    {
```

(continues on next page)

(continued from previous page)

```
{  
    Upgrade();  
}  
#endif  
}
```

Note: If you **undo** conversion you can see warnings like *Not found any Text/TextField/TextFieldExtended component*. This is happening because the newly added TMPro components were deleted and the old default components are not yet restored. In such cases, those warnings should be ignored.

KNOWN PROBLEMS

- **Missing references or scripts**

Sometimes newly created widgets have missing references, or scripts are missing after the update.
Please try to import package again.

- **TextMeshPro support are disabled after the platform switch**

In some cases TextMeshPro support can be disabled after the platform switch because of the missing directive in *Scripting Define Symbols* for the current platform.
Like an upgrade to the new Unity version with the newly added platform and then switch to it.
You need to enable *TextMeshPro Support* again **without** saving the scene to avoid references lost.

- **TextMeshPro related errors after upgrade**

Package upgrade remove TMPro reference from assembly definitions.
Disable TMPro support and enabling it back solves the problem.

- **Newly created widgets are white**

It happens because of the empty style used as default and it automatically applied to newly created widgets.

Please open *New UI Widgets/Styles/UIWidgets Style Default* and check it settings (it should not be all white color or null), and set it as default.

If UIWidgets Style Default values are all white color or null, then try to import package again, sometimes import works incorrectly.

You can use “t:UIWidgets.Styles.Style” to find all styles and check which one is used by default.



- **Random errors of type `*MissingReferenceException`: The object of type ‘...’ has been destroyed but you are still trying to access it* in 2019.3 or later**

These errors can be related to *Enter Play Mode* settings with disabled *Reload Domain* and static variables in your code.

If *Reload Domain* is disabled, then static variables are not reset to default values on entering *Play Mode*.

You should add special code to reset static variables to solve the problem or enable *Reload Domain*.

Example:

```
static ObservableList<int> Numbers = new ObservableList<int>();

[RuntimeInitializeOnLoadMethod(RuntimeInitializeLoadType.
↳ SubsystemRegistration)]
protected static void StaticInit()
{
    Numbers = new ObservableList<int>();
}
```

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

CHANGELOG

12.1 Release 1.15.2

- added Updater static class to control scripts updates `IUpdatable.RunUpdate()` without reflection instead of the default `MonoBehaviour.Update()`
- added SnapGrid: allow to stick draggable or resizable UI game objects to the nearest grid lines
- added LinesDrawer and SnapGridDrawer effects
- COMPATIBILITY-BREAKING CHANGES: `MonoBehaviour.Update()` replaced with `IUpdatable.RunUpdate()`
- Autocomplete: added `AllowCancelOnDeselect` to cancel `DisplayListView` close on deselect event.
- Autocomplete: added `OnSearchCompleted` event
- Autocomplete: added `ResetListViewSelection` option
- AutoCombobox: fixed `InputField` display bug
- AutoCombobox: fixed coloring bug
- AutoCombobox: added `KeepSelection` option (set `Autocomplete.DisplayListView` selected items)
- Connectors: fixed incorrect positions when `CanvasMode` is `WorldSpace` and its scale is not 1
- Dialog: added `ButtonsContainer` option
- ListView: fixed bug with incorrect item sizes when using variable size type
- ListView: fixed highlighting bug
- ListView: fixed wrong background color for the last items in table mode
- ListView: added `RangeMode` property to determine which element is the start when selecting a range with the Shift key.
- ListView: fixed instance recycling if `ListViewItem.IsDragged` enabled
- ListViewItem: `IsDragged` renamed to `DisableRecycling`
- ListViewString: now sort can be disabled with `EnableSort` in Inspector window
- Notification: added `ButtonsContainer` option
- Paginator: added `SetPage` method to change current page without animation
- Sidebar: added `ModalColor` field
- shaders: now should support stereo instanced rendering and SRP batcher (thanks to David Watt)
- Tooltip: added generic Tooltip

- TreeView: added FindNodes method
- Widgets Generation: added Tooltip generation

12.2 Release 1.15.1

- COMPATIBILITY-BREAKING CHANGES: cursors fields at components (Resizable, Rotatable, Splitter, TableHeader, *DragSupport) are no more used and replaced with Cursors asset and CursorsDPISelector component (recommended to have only one CursorsDPISelector component at the scene)
- Accordion: added Curve property to use in animations
- ContextMenu: fixed HotKey null bug
- ListView: added OnComponentCreated, OnComponentEnabled, OnComponentDisabled, OnComponentDestroyed events
- ListView: renamed StopScrollAtItemCenter to ScrollInertiaUntilItemCenter and StopScrollInertia to ScrollInertia
- TreeView: fixed node remove bug when different nodes using the same item
- TreeView: fixed multiple selection bug when selecting a collapsed node
- TreeViewNodeDropSupport: added “Expand Node On Hold” option with customizable delay
- UICursors: static methods replaced to fields so they can be replaced

12.3 Release 1.15.0

- added ListViewEnum with ListViewEnum<T> wrapper to work with any enums
- reduced memory allocations
- all classes with INotifyPropertyChanged support now also implements IObservable which works without memory allocations
- Autocomplete, Combobox: added field ParentCanvas, it used as ListView parent on open
- Combobox: toggle-button is now full width
- ContextMenu: now works correctly with all canvas render modes
- DateScroller/DateTimeScroller/TimeScroller: ScrollBlock replaced with ScrollBlockBase
- Dialog: added InactiveDialogs and AllDialogs properties to get access to the template instances
- Dialog: DialogButton now support callback with Func<DialogBase dialog, int buttonIndex, bool closeDialog> type
- EasyLayout: small improvements
- EasyLayout: Filter property is obsolete and replaced with ShouldIgnore
- Input System support: fixed bug on mobile devices
- ListView: added property TemplateSelector, its allow to use of different templates (not only DefaultItem) depending on the item
- ListView: added GetDebugInfo and PrintDebugInfo methods

- ListView: added “AnimationCurve animation, bool unscaledTime, Action after = null” parameters to the ScrollTo*Animated methods
- ListView: StopScrollAtItemCenter and StopScrollInertia properties
- ListView: fixed problem with not displayed items for ListType with variable sizes
- ListView: added GetComponentEnumerator to iterate through DefaultItems instances as allocation free replacement of the ForEachComponent method
- ListView: added support of Container with custom scale
- ListView: fixed AutoScroll bug
- Notification: NotificationButton now support callback with Func<NotificationBase notification, int buttonIndex, bool closeNotification> type
- Paginators: added OnMovement event
- Popup: added content and onClose parameters to the Show method, added SetContent() method, added OnClose field, works the same way as dialog
- Resizable: added UseCanvasScaler option, if enabled ActiveRegion will be changed according to the CanvasScaler settings
- ScrollBlock: added OnItemChanged event to customize items depending on index and value
- ScrollBlock: added ScrollBlockBase and ScrollBlockCustom<T> classes
- TracksViewBase: ScrollBlock replaced with ScrollBlockBase
- TreeView: fixed drop support bug
- TreeView prefabs: toggle arrow is now nested
- UICursor: added Replacement function to replace cursor (can be used to replace cursor on High DPI screens)
- Utilities: more functions moved to the new UtilitiesUI and UtilitiesRectTransform classes

12.4 Release 1.14.2

- added CircularSlider widget
- added SliderScale widget
- added TimeAnalog widget
- Accordion: added OnStartToggleAnimation and OnDataSourceChanged events
- Accordion: added AccordionHighlight component
- AutoComplete: added OnItemNotFound and OnCancelInput events
- AutoCombobox: added AddItems option (requires overridden Input2Item method)
- CenteredSlider: event OnValuesChange renamed to OnValueChanged
- Connectors: fixed bug related to “Scale With Screen Size”
- Connectors: added rectangular lines support
- Cursor: fixed flickering
- DatePicker and DateTimePicker: fixed initial date
- DirectoryTreeView: nested nodes are automatically loaded on expand from script

- DirectoryTreeView: added ExpandPath(), Path2Node(), Path2NearestNode(), RefreshDirectories() methods
- ListView: fixed highlight coloring on navigation
- ListView: fixed unstopable auto scroll bug
- ListView Drop Support: added ReceiveOnlyEmptyNode option
- RangeSlider: event OnValuesChange renamed to OnValuesChanged
- Resizable: fixed position change
- ScrollBlock: added AllowIncrease and AllowDecrease fields
- Spinner: added SetValue() method to change value without OnChangeEvent invocation
- Styles: fixed error when creating a new style
- Styles: added PixelsPerUnitMultiplier property to the Image styles
- TreeGraph: small performance improvement
- TreeGraph: added LineThickness, LineType, LineMargin options
- TreeView: added ScrollWithIndent option
- Widgets Generation: improved localization support

12.5 Release 1.14.1

- EasyLayout: reduced memory allocations
- Widgets Generation: fixed type name error
- Widgets Generation: fixed missing reference

12.6 Release 1.14.0

- added localizations integration support
- added I2 Localization support
- added ContextMenu
- added Input System support
- added UtilitiesCompare class
- added ScrollRectFooter
- added AutoComboboxIcons prefab
- Dialog, Picker, Popup: added CloseButton property
- EasyLayout: added SetPreferredAndFitContainer option for the Children Size
- ListView: added Header property
- ListViewPaginator: added LoopedList support
- Notification: added “content” and “onReturn” parameters to the Show() method
- Style: fixed unchangeable settings after “Apply Fast Settings” use

- Style: added “Update Default Style” option, which is opposite of the “Apply Default Style”, it gets style settings from widgets and saves them to the current style
- Tabs: added CanSelectTab field to check if tab can be selected with a button click
- TabsCustom: TabButton class changed to the generic class TabButton<T>
- Widgets Generation: generated classes are partial now
- Widgets Generation: added AutoCombobox widget
- Utilities: most functions moved to the new Utilities* classes

12.7 Release 1.12.6

- ListViewItem: added ToggleOnClick and ToggleOnSubmit fields
- Widgets Generation fixes

12.8 Release 1.12.5

- added UIFlareGlobal shader: flare at global space
- added Ripple effect
- UIWidgets extensions methods moved to UIWidgets.Extensions namespace
- EasyLayout extensions methods moved to EasyLayoutNS.Extensions namespace
- shaders: replaced properties names with properties IDs
- Dialog: Show() arguments can later be changed with other methods: SetInfo(), SetButtons(), FocusButton(), SetPosition(), SetContent(), SetCanvas(), SetModal().
- EasyLayout: added GetElementPosition to get position in group
- InputFieldExtended: fixed bug with Value property (thanks to RickSaada1)
- ListView: added ItemsEvents field
- ListViewItem: now foreground and background graphics are serialized properties
- Notify: added buttons support with SetButtons(IList<DialogButton> buttons) method
- ProgressBarIndeterminate: fixed bar jump at the start
- TableHeader: fixed bug with ColumnToggle (thanks to jbw)
- UIFlare shaders: added flare delay property

12.9 Release 1.12.4

- Unity 4.6+ and Unity 5.x no more supported, now the oldest supported version is 2017.4
- fixed SendMessage warnings in Unity 2019.3 and later versions
- assembly definitions removed because all changes in .asmdef files are deleted on package update
- ListView: DefaultItem no more disabled by default in Editor mode
- ListViewDragSupport: added auto-scroll when the drag is near the border
- Notify: now you can create derived classes with NotificationCustom<T>
- TreeView Drag&Drop: now nodes can be reordered

12.10 Release 1.12.3

- added Pinchable component: drag, rotate, resize multi-touch support
- added ListViewAutoResize component: auto-resize ListView or TileView according to items counts until specified maximum size reached
- [Serializable] attribute of TreeNode<TItem> class not available for Unity 2020.1 and later versions
- ListView: added DisableScrollRect property to disable ScrollRect if ListView is not Interactable
- ListView and TreeView Drag&Drop: added Interactable support

12.11 Release 1.12.2

- added DistanceLines component
- added UI Cursor settings component
- Dialog: fixed buttons order
- DirectoryTreeView: fixed drives list
- ListViewPaginator, ScrollRectPaginator: fixed LastPageFullSize option
- ListView: now resize of disabled ListView processed correctly

12.12 Release 1.12.1

- added converter from Unity Text to TextMeshPro text
- added IUpgradeable interface to improve compatibility between versions
- added Groupable component
- added UIFlareTransparent shader
- added ResizableHandles component
- added Rotatable component
- added RotatableHandle component

- deleted a lot of lambda functions
- other lambda functions replaced with local functions
- renamed classes *Utilites to *Utilities
- improved performance with Asset Pipeline V2
- Combobox: fixed navigation support
- Draggable: added Target property to drag the specified target instead of self
- DragSupport: added AllowDrag field
- DropSupport: added ReceiveItems and ReceiveNodes fields for the base classes
- ListView: not selectable items are no more highlighted and navigated
- ListViewPaginator, ScrollRectPaginator: added LastPageFullSize option to change the last page size to full-page size
- Resizable: AllowResize renamed to Interactable
- Resizable: added resize type to change between size and scale
- Resizable: added Target property to resize the specified target instead of self
- ScrollRectEvents: RequiredMovement replaced with Thresholds to support separate thresholds for each pull direction
- Splitter: AllowResize renamed to Interactable
- Widgets Generation: added option to manually specify the type name if the type cannot be detected from the MonoScript

12.13 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

12.14 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

12.15 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

12.16 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

12.17 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

12.18 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 Utilities.GetTime field (You can use own Time manager instead of the default Time.time)

12.19 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

12.20 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 ColorPickerRange.
- 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”

12.21 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

12.22 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()`

12.23 Release 1.9.1

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

12.24 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 graphics 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

12.25 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

12.26 Release 1.8.4

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

12.27 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

12.28 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

12.29 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.

12.30 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.

12.31 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.

12.32 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.

12.33 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.

12.34 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.

12.35 Release 1.5.0

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

12.36 Release 1.4.2

- Added ListViewCustomHeight (support items of variable heights)

12.37 Release 1.4.1

- Added CenteredSlider.

12.38 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.

12.39 Release 1.3

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

12.40 Release 1.2

- Added Dialog
- Added Draggable

12.41 Release 1.1

- Added Notify
- Added EasyLayout

12.42 Release 1.0

- Initial release