New UI Widgets Documentation

Release 1.10.3f1

Ilia Novikov

May 12, 2019

Contents

1 New in 1.10 **1**

2 Widgets **3**
- 2.1 Recommended Unity UI documentation . . . 3
- 2.2 Collections . . . 3
- 2.3 Containers . . . 4
- 2.4 Dialogs . . . 4
- 2.5 Input . . . 4
- 2.6 Misc . . . 5

3 Widgets Generation **7**
- 3.1 Requirements . . . 8
- 3.2 Supported types . . . 8
- 3.3 Limitations . . . 9
- 3.4 INotifyPropertyChanged Support . . . 9
- 3.5 Replacing generated code . . . 10
  - 3.5.1 Collections . . . 10
  - 3.5.2 Autocomplete . . . 13

4 Using Widgets **15**
- 4.1 Collections . . . 15
  - 4.1.1 Combobox . . . 15
  - 4.1.2 ListView, TileView and Table . . . 15
  - 4.1.3 Paginator . . . 22
  - 4.1.4 TreeView . . . 23
- 4.2 Containers . . . 27
  - 4.2.1 Accordion . . . 27
  - 4.2.2 Tabs . . . 27
- 4.3 Dialogs . . . 28
  - 4.3.1 DatePicker . . . 28
  - 4.3.2 Dialog . . . 29
  - 4.3.3 FileDialog . . . 31
  - 4.3.4 FolderDialog . . . 33
  - 4.3.5 Notifications . . . 34
  - 4.3.6 Pickers . . . 35
  - 4.3.7 Popup . . . 35
- 4.4 Input . . . 36

4.4.1 Autocomplete . . . 36
4.4.2 Calendar . . . 36
4.4.3 Centered Slider . . . 37
4.4.4 ColorPicker . . . 38
4.4.5 RangeSlider . . . 38
4.4.6 Spinner . . . 39
4.5 Misc . . . 39
4.5.1 ProgressbarDeterminate . . . 39
4.5.2 ProgressbarIndeterminate . . . 40

5 Components **41**
5.1 Bring to Front . . . 41
5.2 Drag and Drop components . . . 41
5.2.1 Drag-and-Drop Support for the Collections . . . 41
5.2.2 Custom Drag Support . . . 42
5.2.3 Custom Drop Support . . . 44
5.3 Draggable . . . 46
5.4 EasyLayout . . . 46
5.5 Layout Switcher . . . 49
5.6 Lightbox . . . 49
5.7 Object Sliding . . . 49
5.8 Resizable . . . 50
5.9 Scrollbar Min Size . . . 50
5.10 ScrollRect Events . . . 50
5.11 Selectable Helper . . . 51
5.12 Sidebar . . . 51
5.13 Splitter . . . 52
5.14 Switch Group . . . 52
5.15 Table Header . . . 52
5.16 Tooltip . . . 53
5.17 TreeView data source . . . 53

6 Styles (Skins) **55**
6.1 Style support for the custom widgets . . . 56

7 Localization Support **57**
7.1 Default ListViewIcons and TreeView . . . 57
7.2 Generated ListView, TreeView, TileView . . . 58
7.3 Tabs . . . 59

8 Data Bind for Unity Support **61**

9 TextMesh Pro Support **63**
9.1 Known problems and solutions . . . 64

10 Support **67**

11 Changelog **69**
11.1 Release 1.10.3 . . . 69
11.2 Release 1.10.2 . . . 69
11.3 Release 1.10.1 . . . 71
11.4 Release 1.10.0 . . . 71
11.5 Release 1.9.3 . . . 72
11.6 Release 1.9.2 . . . 72
11.7 Release 1.9.1 . . . 73

11.8 Release 1.9.0 . . . 73
11.9 Release 1.8.5 . . . 75
11.10 Release 1.8.4 . . . 75
11.11 Release 1.8.3 . . . 75
11.12 Release 1.8.2 . . . 75
11.13 Release 1.8.0 . . . 76
11.14 Release 1.7.4 . . . 76
11.15 Release 1.7.2 . . . 76
11.16 Release 1.7.0 . . . 77
11.17 Release 1.6.5 . . . 77
11.18 Release 1.6.0 . . . 78
11.19 Release 1.5.0 . . . 78
11.20 Release 1.4.2 . . . 78
11.21 Release 1.4.1 . . . 78
11.22 Release 1.4 . . . 78
11.23 Release 1.3 . . . 79
11.24 Release 1.2 . . . 79
11.25 Release 1.1 . . . 79
11.26 Release 1.0 . . . 79

CHAPTER 1

New in 1.10

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

CHAPTER 2

Widgets

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

2.1 Recommended Unity UI documentation

- Canvas
- RectTransform
- Events and Event Triggers
- Mask
- Transitions
- Layout Groups

2.2 Collections

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

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

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

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

2.3 Containers

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

2.4 Dialogs

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

2.5 Input

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

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

2.6 Misc

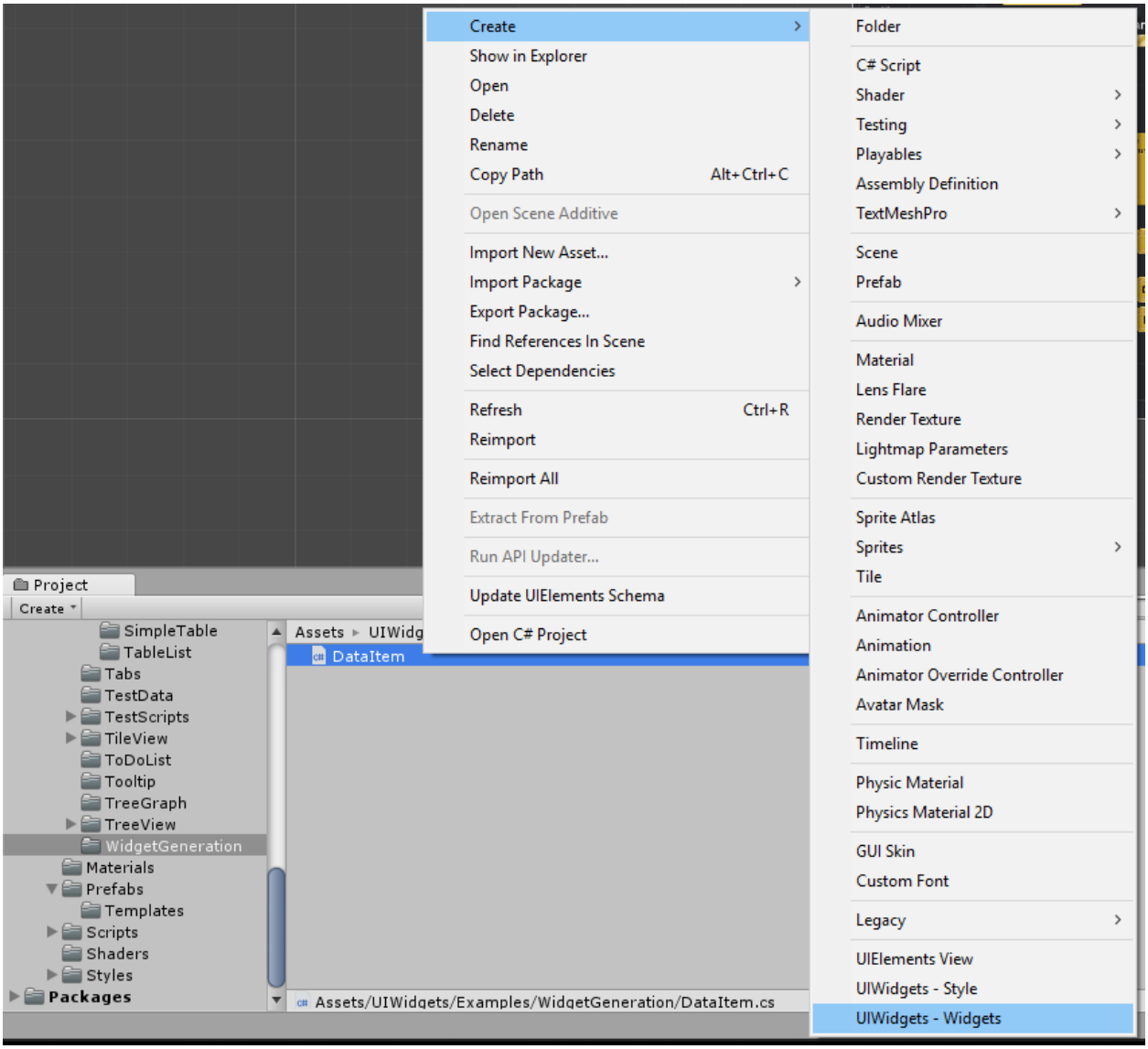
- AudioPlayer
- ProgressbarDeterminate
- ProgressbarIndeterminate
- ScrollRectPaginator
- ScrollRectNumericPaginator

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

CHAPTER 3

Widgets Generation

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



3.1 Requirements

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

3.2 Supported types

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

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

Graphic types:

- `Sprite`
- `Texture2D`

- `Color`
- `Color32`

3.3 Limitations

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

3.4 INotifyPropertyChanged Support

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

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

   public string Name
   {
      get
      {
         return name;
      }

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

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

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

   ...
}
```

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

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

You can disable this behavior with `ObserveItems` property:

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

3.5 Replacing generated code

Generated code can be freely modified.

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

3.5.1 Collections

Widgets to display collections consist of the three classes:

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

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

Functions to modify in the DefaultItem class:

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

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

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

Original code:

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

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

      ...

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

(continues on next page)

(continued from previous page)

```
        /// <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()
        {
            base.Start();
```

(continues on next page)

(continued from previous page)

```
            if (Number != null)
            {
                Number.onValueChangeInt.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.onValueChangeInt.RemoveListener(UpdateNumber);
            }

            base.OnDestroy();
        }

        ...
    }
}
```

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

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

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

(continues on next page)

(continued from previous page)

```
    ...
}
```

3.5.2 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>
   {
      ...
      /// <summary>
      /// Returns a value indicating whether Input occurs within specified value.
      /// </summary>
```

(continues on next page)

(continued from previous page)

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

CHAPTER 4

Using Widgets

4.1 Collections

4.1.1 Combobox

You should use *ListView* property like `combobox.ListView.SelectedIndex`

4.1.2 ListView, TileView and Table

- All collections support virtualization: gameobjects will be created only for visible items.
- Add `Selectable` component to use keyboard and gamepad navigation.
- Different ListView, TileView and Table can display the same list.

List View Type

Fig. 1: ListView with Fixed Size.

Fig. 2: ListView with Variable Size.

Fig. 3: TileView with Fixed Size.

Fig. 4: TileView with Variable Size.

Settings

- Interactable `bool`

  User can interact with ListView.

- List Type `ListViewType`

  Determines how items are displayed.

  - `ListViewWithFixedSize`

    Works with EasyLayout, Horizontal Layout Group and Vertical Layout Group.

  - `ListViewWithVariableSize`

    Works with EasyLayout, Horizontal Layout Group and Vertical Layout Group.

  - `TileViewWithFixedSize`

    Works with EasyLayout.

  - `TileViewWithVariableSize`

    Works with EasyLayout.

  - `TileViewStaggered`

    Works with EasyLayout.

- Sort `bool`

  If enabled items will be sorted with **SortFunc**. Deprecated, replaced with DataSource.Comparer.

- SortFunc `Func<IEnumerable<TItem>, IEnumerable<TItem>>`

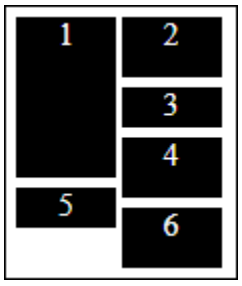


Fig. 5: TileView Staggered.

Not available in the Inspector window Function to sort items. Deprecated, replaced with **DataSource.Comparer**.

- Data Source `ObservableList<TItem>`

  List of the items.

- Multiple Select `bool`

  Allow to select multiple items, otherwise only one.

- SelectedIndex `int`

  Index of the last selected item.

- SelectedIndices `List<int>`

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

- SelectedItem `TItem`

  Not available in the Inspector window Last selected item.

- SelectedItems `List<TItem>`

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

- Direction `ListViewDirection`

  ListView direction.

  - `Horizontal`
  - `Vertical`

- DefaultItem `TComponent`

  A prefab used to display item.

- Container `Transform`

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

- ScrollRect `ScrollRect`

  ScrollRect used by ListView. Required for virtualization support.

- Colors

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

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

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

- Fade Duration `float`

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

- End Scroll Delay `float`

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

- Navigation `bool`

  Allow to use navigation with keyboard or gamepad.

- Looped List `bool`

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

- Is Table `bool`

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

- Set Content Size Fitter `bool`

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

- Scroll Unscaled Time `bool`

  ScrollTo functions will be used unscaled time.

- Scroll Movement `AnimationCurve`

  Animation curve for the ScrollTo functions.

- Center The Items `bool`

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

- Precalculate Item Size `bool`

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

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

- CanSelect `Func<int, bool>`

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

- CanDeselect `Func<int, bool>`

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

- Events
  - OnSelect `UnityEvent<int, ListViewItem>`

    Event raised when item selected.

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

  - OnDeselect `UnityEvent<int, ListViewItem>`

    Event raised when item deselected.

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

  - OnSelectObject `UnityEvent<int>`

    Event raised when item selected.

    Arguments: index of the selected item.

- OnDeselectObject `UnityEvent<int>`

  Event raised when item deselected.

  Arguments: index of the deselected item.

- OnStartScrolling `UnityEvent`

  Event raised when scrolling starts.

- OnEndScrolling `UnityEvent`

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

- onSubmit `UnityEvent`

  Event raised when ListView gameobject has been selected via a "submit" key you specify (default is the return key).

- onCancel `UnityEvent`

  Event raised when ListView gameobject has been deselected.

- onItemSelect `UnityEvent`

  Event raised when ListView item gameobject has been selected via a "submit" key you specify (default is the return key).

- onItemCancel `UnityEvent`

  Event raised when ListView item gameobject has been deselected.

- OnUpdateView `UnityEvent`

  Event raised when ListView view was updated.

- OnFocusIn `UnityEvent<BaseEventData>`

  Event raised when ListView gameobject received focus.

- OnFocusOut `UnityEvent<BaseEventData>`

  Event raised when ListView gameobject lost focus.

- OnPointerEnterObject `UnityEvent<int>`

  Event raised when pointer entered ListView item gameobject.

  Arguments: index of the item.

- OnPointerExitObject `UnityEvent<int>`

  Event raised when pointer exited ListView item gameobject.

  Arguments: index of the item.

Get items

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

Set items

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

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

Display same list with ListView, TileView or Table

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

Get last selected index

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

Get selected indices

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

Last selected item

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

Get selected items

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

Delete specified item

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

Delete item by index

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

Clear list

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

Add item

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

Add items

```
var new_items = new List<ListViewIconsItemDescription>()
{
   new_item,
```

(continues on next page)

(continued from previous page)

```
   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);
```

Scroll to item

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

4.1.3 Paginator

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

How to select paginator

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

Settings

- **(optional) Default Page** Template GameObject to display inactive pages
- **(optional) Active Page** Template GameObject to display active page
- **(optional) Prev Page** GameObject, go to the previous page
- **(optional) Next Page** GameObject, go to the next page
- **Fast Drag Distance and Fast Drag Time** Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. To disable set to zero.

- **Force Scroll On Page** Scroll to the nearest page when drag ended if not meet *Fast Drag* condition. Should be used only with touch devices.
- **Animation** Enable animation
- **Movement** Animation curve
- **Unscaled Time** Enable if the animation should use Unscaled Time.

ListViewPaginator specific settings

- **PerPage** 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.

ScrollRectPaginator specific settings

- **Direction** Scroll direction.
- **Page Size Type** If *Page Size Type* = *Auto* page size is equal scroll rect size, if *Page Size Type* = *Fixed* will be used *Page Size* value.
- **Page Size** Size of the page.
- **Page Spacing** Space between pages.

Tile View ScrollRect Fitter

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

4.1.4 TreeView

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

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

Get nodes

```
public TreeView Tree;

ObservableList<TreeNode<TreeViewItem>> nodes;

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

Get selected nodes

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

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

(continues on next page)

(continued from previous page)

```
    // 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);
      }
```

(continues on next page)

(continued from previous page)

```
    });
}
```

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();
}
```

4.2 Containers

4.2.1 Accordion

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,
   },
};
```

4.2.2 Tabs

Select tab

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

Enable tab

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

Disable tab

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

4.3 Dialogs

4.3.1 DatePicker

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

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

      [SerializeField]
      Text Result;

      DateTime currentValue = DateTime.Today;

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

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

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

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

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

(continues on next page)

(continued from previous page)

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

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

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

4.3.2 Dialog

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 parametres
dialog.Show(
    title: "Modal Dialog",
    message: "Simple Modal Dialog.",
    buttons: new DialogActions(){
        // Button name and Func<bool>, return true to close dialog, otherwise false
        {"Close", Dialog.Close},
    },
    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]
```

(continues on next page)

(continued from previous page)

```
    public InputField Username;

    [SerializeField]
    public InputField Password;

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

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

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

        return valid_username && valid_password;
    }
}
```

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 DialogActions(){
            {"Sign in", () => SignInNotify(helper)},
            {"Cancel", Dialog.Close},
        },
        focusButton: "Sign in",
        modal: true,
        modalColor: new Color(0, 0, 0, 0.8f)
    );
}

bool SignInNotify(DialogInputHelper helper)
{
    //if username or password empty than don't close dialog
    if (!helper.Validate())
    {
        return false;
    }
```

(continues on next page)

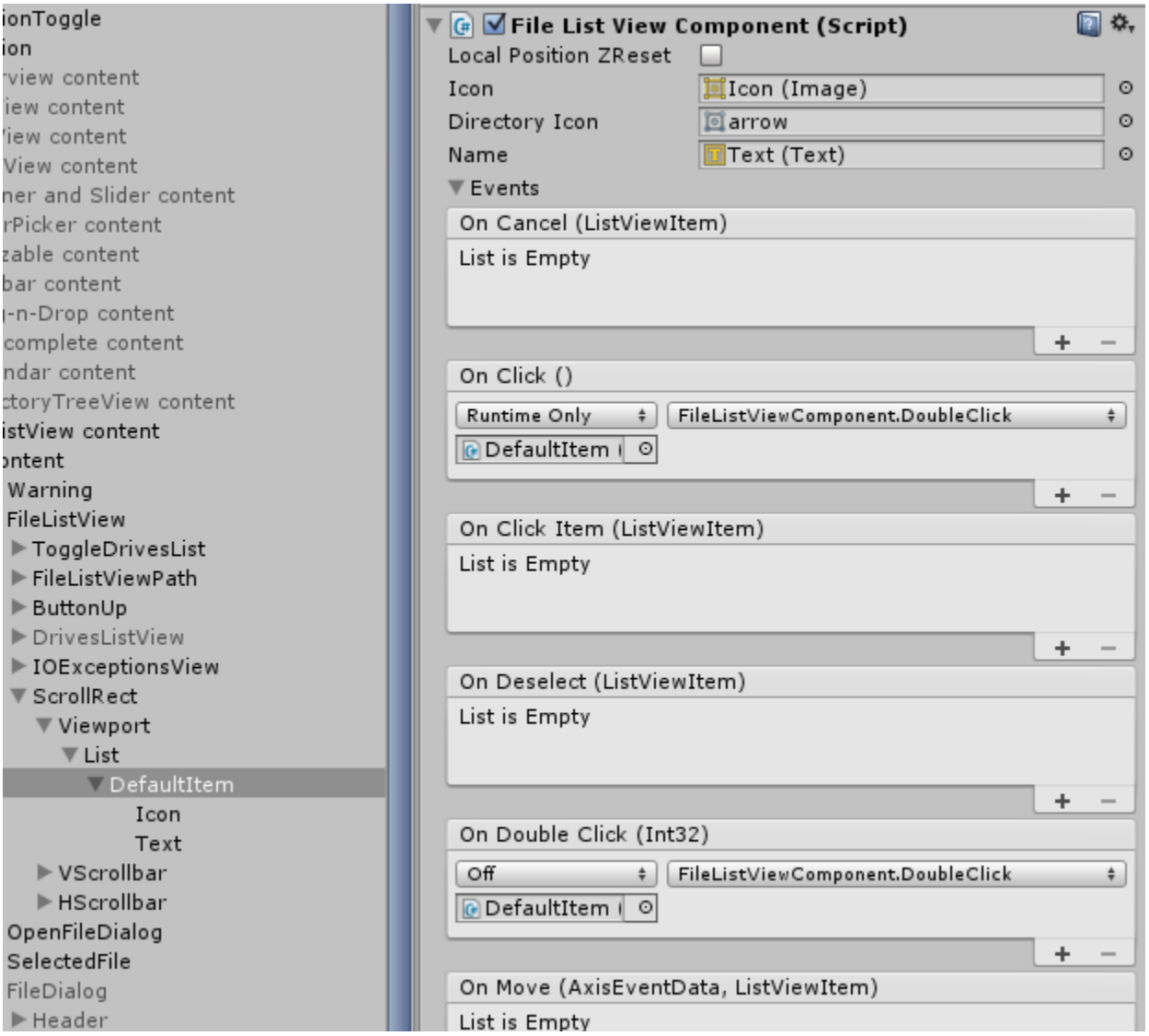
(continued from previous page)

```
    //show notification
    var message = "Sign in.\nUsername: " + helper.Username.text + "\nPassword:
↪<hidden>";
    notifySample.Clone().Show(message, customHideDelay: 3f);

    return true;
}
```

4.3.3 FileDialog

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



Using FileDialog.

```
namespace UIWidgets.Examples
```

(continues on next page)

(continued from previous page)

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

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

      [SerializeField]
      Text Result;

      string currentValue = string.Empty;

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
        }

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

4.3.4 FolderDialog

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

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

                    [SerializeField]
                    Text Result;

                    string currentValue = string.Empty;

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

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

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

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

                    /// <summary>
                    /// Show picker and display selected value.
                    /// </summary>
```

(continues on next page)

(continued from previous page)

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

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

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

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

4.3.5 Notifications

Important: If you want to display more than one notification at the same time, then *notification container* should have *layout group* component like `EasyLayout`.

Minimal code

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

Advanced

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

4.3.6 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");
            }
     }
}
```

4.3.7 Popup

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 parametres
popup.Show(
   title: "Modal popup",
   message: "Simple Modal popup.",
   modal: true,
   modalColor: new Color(0, 0, 0, 0.8f)
);
```

4.4 Input

4.4.1 Autocomplete

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

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

      [SerializeField]
      ListViewIconsItemDescription item;

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

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

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

4.4.2 Calendar

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

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

(continues on next page)

(continued from previous page)

```
    /// <summary>
    /// Test Calendar.
    /// </summary>
    public class TestCalendar : MonoBehaviour
    {
        /// <summary>
        /// Calendart.
        /// </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
            Calendar.Culture = new System.Globalization.CultureInfo("en-US");

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

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

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

4.4.3 Centered Slider

The difference from a simple slider:

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

Set value

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

Set display limits

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

Set value limits

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

4.4.4 ColorPicker

Set color

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

Get color

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

Add listener

```
void Start()
{
   ColorPicker.OnChange.AddListener(ColorChanged);
}

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

4.4.5 RangeSlider

Slider with two handles for minimum and maximum.

Set values

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

Set step

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

Set limits

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

Add listener

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

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

4.4.6 Spinner

Set maximum

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

Set minimun

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

Set value

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

Set step

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

Get value

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

4.5 Misc

4.5.1 ProgressbarDeterminate

Set value

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

Stop animation

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

4.5.2 ProgressbarIndeterminate

Start animation

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

Stop animation

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

CHAPTER 5

Components

5.1 Bring to Front

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

Options

- *With Parents*: bring to front GameObject with parents GameObjects.

5.2 Drag and Drop components

5.2.1 Drag-and-Drop Support for the Collections

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

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

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

Options

- *DragInfo*: component to display the dragged data.
- *DragInfo Offset*: offset from the cursor position.
- *Allow Drop Cursor*: cursor when drop allowed.
- *Allow Drop Cursor Hot Spot*: cursor hot spot.
- *Denied Drop Cursor*: cursor when drop denies.
- *Denied Drop Cursor Hot Spot*: cursor hot spot.
- *Default Cursor Texture*: default cursor.
- *Default Drop Cursor Hot Spot*: cursor hot spot.

5.2.2 Custom Drag Support

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

Methods

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

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

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

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

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();
      }
```

(continues on next page)

(continued from previous page)

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

        base.Dropped(success);
    }

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

    /// <summary>
    /// DragInfo offset.
    /// </summary>
    [SerializeField]
    [FormerlySerializedAs("LocalPosition")]
    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.
```

(continues on next page)

(continued from previous page)

```
        /// </summary>
        protected virtual void HideDragInfo()
        {
            if (DragInfo == null)
            {
                return;
            }

            DragInfo.SetActive(false);
        }
    }
}
```

5.2.3 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.

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

`CanReceiveDrop` function allow to accept only nodes with names ends with *1*.

```
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>
    {
        #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)
        {
            GetComponent<InputField>().text = data.Item.Name;
        }

        /// <summary>
```

(continues on next page)

(continued from previous page)

```
        /// 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

        #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)
        {
            GetComponent<InputField>().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>
        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)
        {
```

(continues on next page)

(continued from previous page)

```
        }

        #endregion
    }
}
```

5.3 Draggable

Dragging gameobject.

Options

- *Handle* (optional): GameObject used to drag current GameObject.
- *Horizontal*: allow horizontal drag movement.
- *Vertical*: allow vertical drag movement.
- *Restriction*:
  - *None*: no restriction.
  - *Strict*: does not allow drag outside parent.
  - *After Drag*: does not allow drag outside parent, applied when drag ended.
- *Curve*: Animation curve used to animate applied **After Drag** restriction.
- *UnscaledTime*: animate applied **After Drag** restriction with unscaled time.

5.4 EasyLayout

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

- Main Axis

  Determine how elements will be placed.

- Layout Type
  - `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.
- Group Position

  Only for the `Compact` and `Grid` layouts.

  Combination of horizonal (`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

  Only for the `Compact` layout.

  Element position in the row (`Left`, `Center`, `Right`).

- Inner Align

  Only for the `Compact` layout.

  Column position relative to the group (`Top`, `Middle`, `Bottom`).

- Compact Constraint

  Only for the `Compact` layout.

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

- Compact Constraint Count

  Only for the `Compact` layout.

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

- Cell Align

  Only for the `Grid` layout.

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

- Grid Constraint

  Only for the `Grid` layout.

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

- Grid Constraint Count

  Only for the `Grid` layout.

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

- Flex Setting

  Only for the `Flex` layout.

  - Wrap

    If disabled elements will all placed onto one line (row or column).

  - Justify 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

    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

    Define how elements are placed out along the cross axis on the line (column or row).

    * `Start`
    * `Center`
    * `End`

- Staggered Settings

  Only for the `Staggered` layout.

  - Fixed Block Count

    Count of the rows or columns.

  - Blocks Count

- Spacing

  Empty space between elements.

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

- Symmetric

  Use symmetric margin.

- Margin

  Empty space from parent edges.

- Skip Inactive

  Do not reserve space for disabled elements.

- Right To Left

  The order of placement of elements.

- Top To Bottom

  The order of placement of elements.

- Children Width
  - `Do nothing`: do not resize elements.

  - `Set Preferred`: set element width to `Preferred Width`.
  - `Set Max From Preferred`: set maximum of the `Preferred Width` from the all elements.
  - `Fit Container`: increase elements width proportionally `Flexible Width` to fit parent width.
  - `Shrink On Overflow`: decrease elements width if summary width more than parent width with margin.
- Children Height

  Similar to `Children Width`
- Settings Changed

  Event, raised after any setting was changed.

5.5 Layout Switcher

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

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

Options

- *Objects*: list of the controlled objects.
- *Default Display Size* (inches): display size to use when actual display size cannot be detected.
- *Layouts*: list of the layouts.
  - *Name*: layout name.
  - *Aspect Ratio*: aspect ratio for this layout.
  - *Max Display Size* (inches): maximum size of the display for this layout (layout will not be used if display size more than specified).

5.6 Lightbox

Lightbox is a component used to display overlay image.

5.7 Object Sliding

Component to drag GameObject horizontally or vertically between specified positions.

Options

- *Positions*: allowed positions for this object.
- *Direction*: slide direction.
  - *Horizontal*
  - *Vertical*
- *Movement*: animation curve.
- *Unscaled Time*: animate with unscaled time.

Helper components

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

- *Object Sliding Horizontal Helper*
  - *Object on Left*: list of the objects on the left side of the current object.
  - *Object on Right*: list of the objects on the right side of the current object.
- *Object Sliding Vertical Helper*
  - *Object on Top*: list of the objects on the top side of the current object.
  - *Object on Bottom*: list of the objects on the bottom side of the current object.

5.8 Resizable

Allow to resize gameobject.

Options

- *Update RectTransform*: change RectTransform size.
- *Update LayoutElement*: change LayoutElement size.
- *Active Region*: distance from border where resize allowed.
- *Min Size*: minimal size.
- *Max Size*: maximum size, not applied if size is zero.
- *Keep Aspect Ratio*: Aspect ratio applied after MinSize and MaxSize, so if default aspect ratio not equal MinSize and MaxSize aspect ratio then real size may be outside limit with one of the axis.
- *Resize Directions*: allowed resize directions.

Events

- `OnStartResize(Resizable)`
- `OnEndResize(Resizable)`

5.9 Scrollbar Min Size

Allow to set minimal scrollbars sizes of the ScrollRect.

Options

- *Horizontal Min Size*: minimal size of the horizontal scrollbar.
- *Vertical Min Size*: minimal size of the vertical scrollbar.

5.10 ScrollRect Events

Provide pull events for the ScrollRect.

Options

- *Required Movement*: required pull distance to raise events.

Events

- `OnPull(PullDirection)`
- `OnPullAllowed(PullDirection)`
- `OnPullCancel(PullDirection)`

- `OnPullUp()`
- `OnPullDown()`
- `OnPullLeft()`
- `OnPullRight()`

5.11 Selectable Helper

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

5.12 Sidebar

Component to drag sidebar from behind the screen.

Options

- *Interactable*: use to enable or disable the ability to drag the sidebar.
- *Curve*: animation curve for the open and close animations.
- *Direction*: sidebar drag direction to open.
- *Animation Type*:
  - *Overlay*
  - *Push*
  - *Scale Down*
  - *Uncover*
  - *Slide Along*
  - *Slide Out*
  - *Resize*
  - *Scale Down and Push*
- *Scale Down Limit*: scale cannot be lower this value for the ScaleDown animation.
- *Is Open*: is sidebar opened?
- *Modal*: is sidebar should be closed with the click outside of the sidebar?
- *ScrollRect Support*: allow to handle children ScrollRect's drag events.
- *Content*: content GameObject.
- *Animate With Layout*: change Content LayoutElement size during animation.
- *Optional Handle*: handle to open and close sidebar.
- *Unscaled Time*: animate with unscaled time.

Events

- `OnOpen()`
- `OnClose()`
- `OnOpeningStarted()`
- `OnClosingStarted()`

5.13 Splitter

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

Options

- *Type*:
  - *Horizontal*: change heights of the gameobjects.
  - *Vertical*: change widths of the gameobjects.
- *Update RectTransform*: change neighboring gameobjects RectTransform size.
- *Update LayoutElement*: change neighboring gameobjects LayoutElement size.
- *Mode*:
  - *Auto*: use previous and next siblings in hierarchy.
  - *Manual*: use specified targets to resize.
- *Left Target*: left (or top) target to resize.
- *Right Target*: right (or bottom) target to resize.

5.14 Switch Group

Same as Toggle Group, but for the Switch widget.

Options

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

5.15 Table Header

Used with ListView on table mode. Allow columns resizing and reorderindg.

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

Options

- *List*: controlled ListView.
- *Allow Resize*: allow to change columns width.
- *Allow Reorder*: allow to change columns order.
- *On Drag Update*: update column width during drag, if disabled column width will be changed after the drag ended.
- *Active Region*: distance from border where resize allowed.
- *Drop Indicator*: indicator to display new column position during column reordering.

5.16 Tooltip

Displaying the tooltip on object focus.

Options

- *Tooltip Object*: GameObject used as tooltip.
- *Bring To Front*: bring to front tooltip object on display.
- *Show Delay*: delay in seconds before tooltip displayed.
- *Unscaled Time*: delay with unscaled time.

Events

- `OnShow`
- `OnHide`

5.17 TreeView data source

Used in editor mode, allow to edit TreeView nodes.

Important: Work only with default TreeView. Custom TreeView not supported.

CHAPTER 6

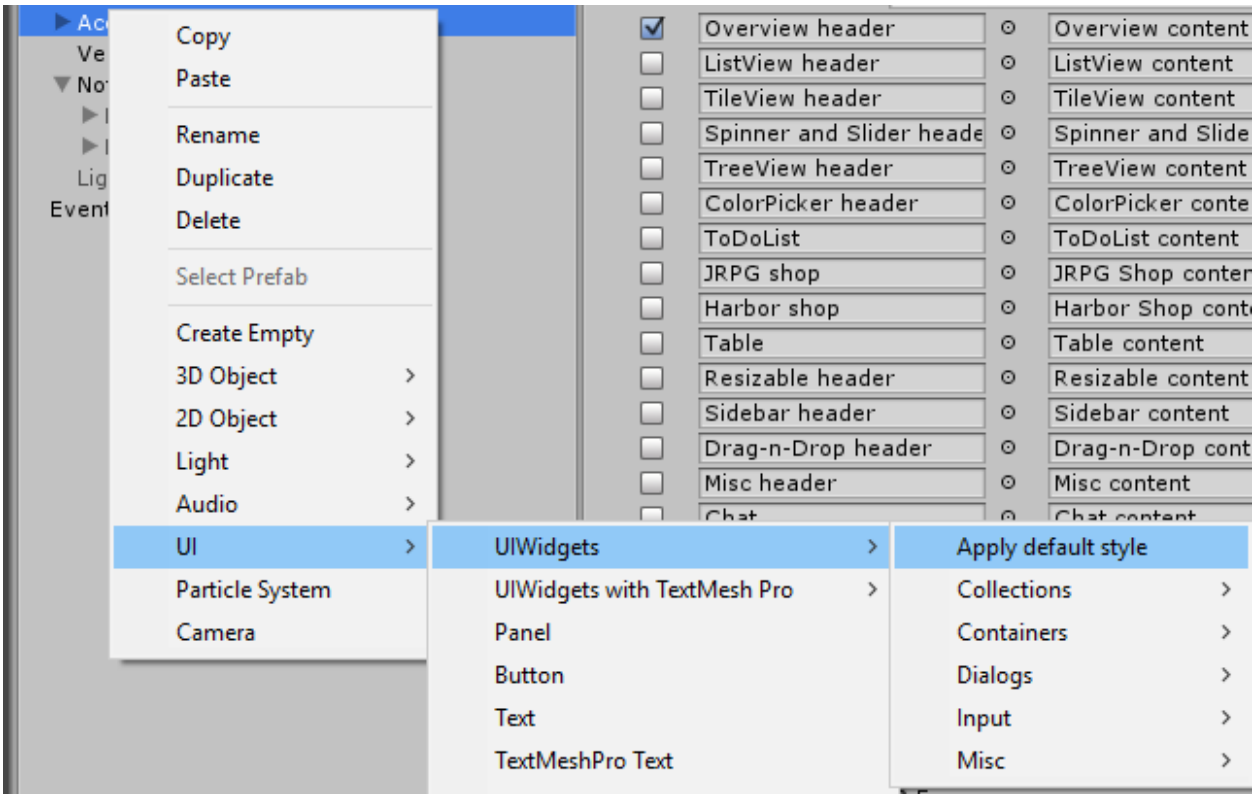
Styles (Skins)

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

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

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

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



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

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

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

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

6.1 Style support for the custom widgets

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

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

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

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

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

CHAPTER 7

Localization Support

7.1 Default ListViewIcons and TreeView

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

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

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

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

         return targetListView;
      }
   }

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

   void OnDestroy()
   {
```

(continues on next page)

(continued from previous page)

```
        LocalizationSystem.OnLanguageChanged -= Localization;
    }

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

7.2 Generated ListView, TreeView, TileView

You can change component class to add localization support.

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

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

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

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

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

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

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

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

7.3 Tabs

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

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

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

CHAPTER 8

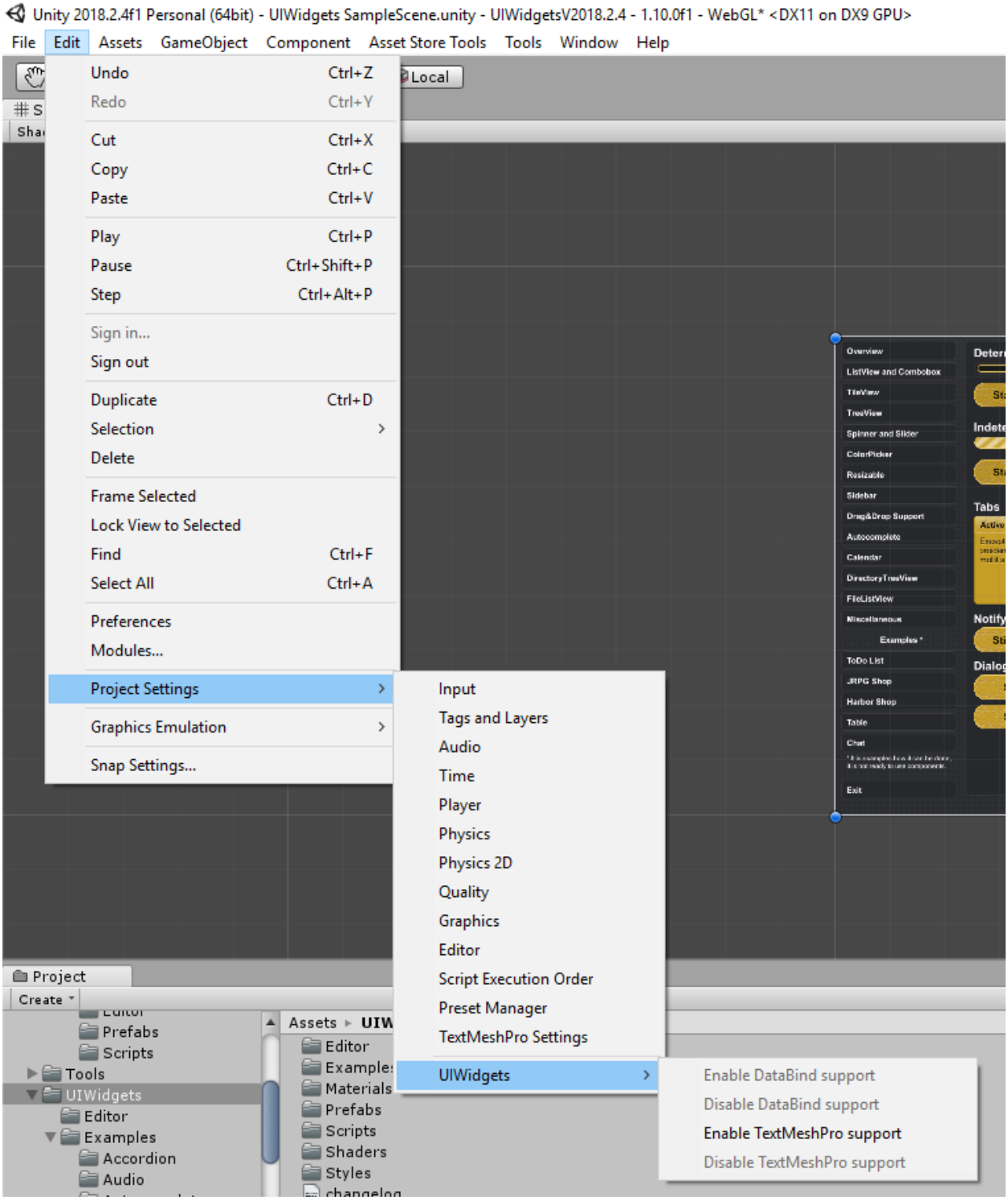
Data Bind for Unity Support

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

After enabling support:

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

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


Unity 2018.2.4f1 Personal (64bit) - UIWidgets SampleScene.unity - UIWidgetsV2018.2.4 - 1.10.0f1 - WebGL* <DX11 on DX9 GPU>
File
Edit
Assets
GameObject
Component
Asset Store Tools
Tools
Window
Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Play Ctrl+P
Pause Ctrl+Shift+P
Step Ctrl+Alt+P
Sign in...
Sign out
Duplicate Ctrl+D
Selection
Delete
Frame Selected
Lock View to Selected
Find Ctrl+F
Select All Ctrl+A
Preferences
Modules...
Project Settings
Graphics Emulation
Snap Settings...
Input
Tags and Layers
Audio
Time
Player
Physics
Physics 2D
Quality
Graphics
Editor
Script Execution Order
Preset Manager
TextMeshPro Settings
UIWidgets
Enable DataBind support
Disable DataBind support
Enable TextMeshPro support
Disable TextMeshPro support
Project
Create
Prefabs
Scripts
Tools
UIWidgets
Editor
Examples
Accordion
Audio
Assets
Editor
Materials
Prefabs
Scripts
Shaders
Styles

CHAPTER 9

TextMesh Pro Support

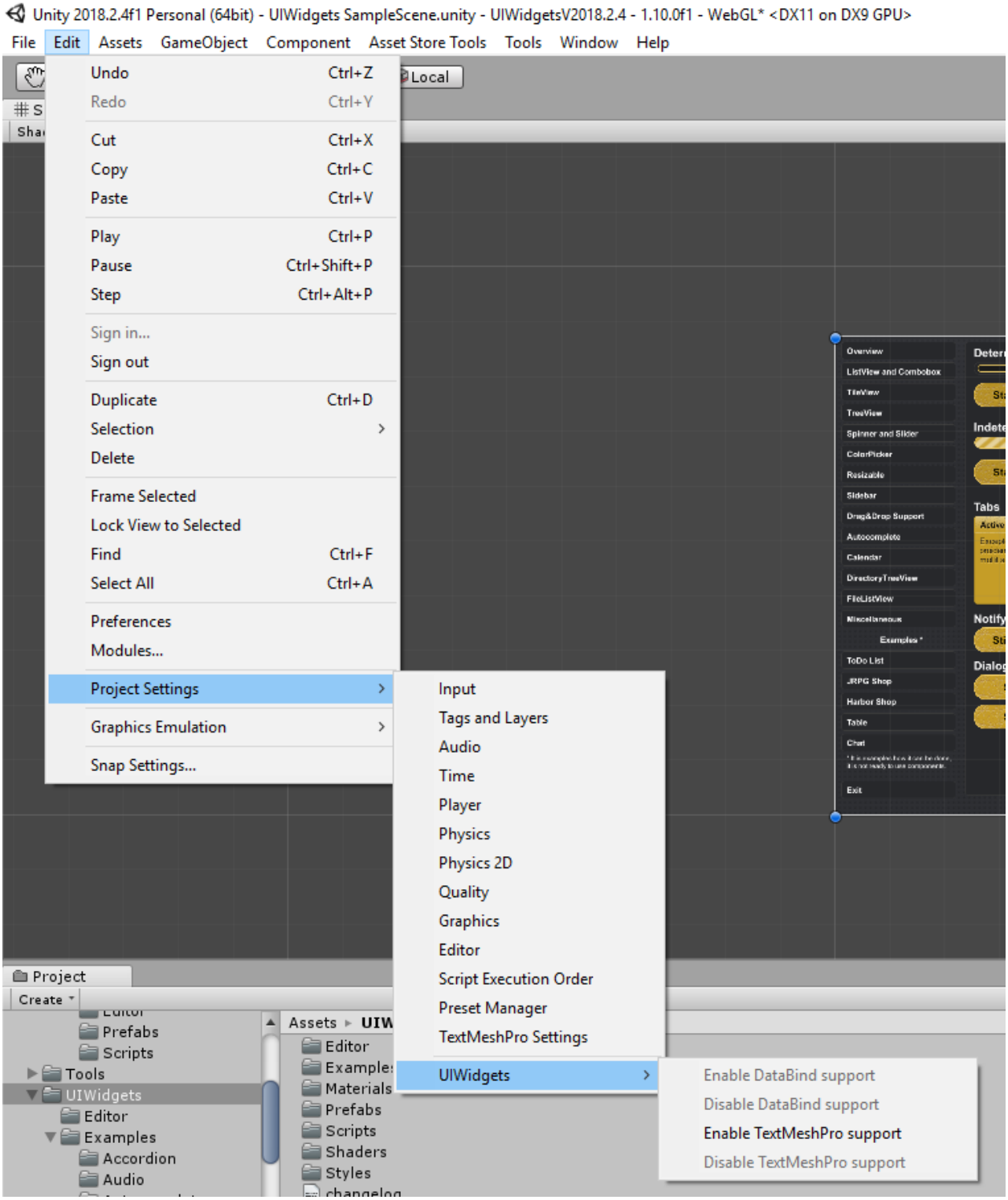
You can enable **TextMesh Pro** support with *Project Settings / New UI Widgets / Enable TextMesh Pro support*. If **TextMesh Pro** not installed option will not be available.

After enabling support:

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

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

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



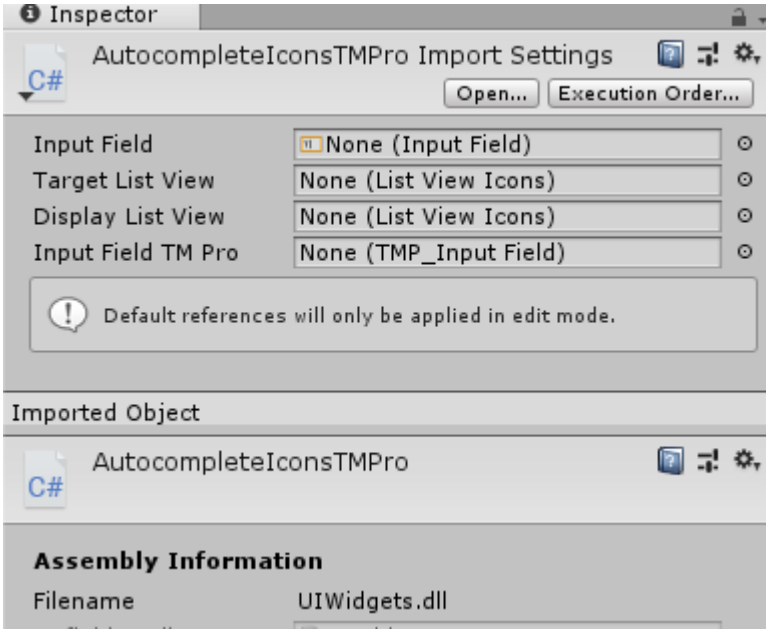
9.1 Known problems and solutions

- Widgets have missing script errors after enabling TextMesh Pro Support

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

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

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



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

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

Add your Unity version number to line:

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

like this:

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

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

CHAPTER 10

Support

You can ask me questions at:

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

CHAPTER 11

Changelog

11.1 Release 1.10.3

- added GroupedTileView example
- 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
- DragRedirect: improved support for the multiple redirects
- 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 have only one field of the supported types

11.2 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 INofityPropertyChanged 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 invocate 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 TMPro prefabs after update to new version
- Widget Generation: added ScriptableObject support
- Widget Generation: added Data Bind support
- Other: fixes related using instantiate with inited complicated widgets
- Other: "UIWidgets" in the menu replaced with "New UI Widgets" to match with the package name
- Other: Time used with animations can be controlled with Utilites.GetTime field (You can use own Time manager instead of the default Time.time)

11.3 Release 1.10.1

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

11.4 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"

11.5 Release 1.9.3

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

11.6 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 targers, instead using previous and next siblings in hierarchy
- TreeView: added serialization support with TreeNode<T>.Serialize() and TreeNode<T>.Deserialize()
- TreeView: fixed error when deleting selected node with disabled DeselectCollapsedNodes
- TreeView: added ExpandParentNodes() and CollapseParentNodes() functions
- TreeView's DefaultItem: Filler renamed to Indentation
- Dialog, Notify, Picker, Popup: Template() renamed to Clone()

11.7 Release 1.9.1

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

11.8 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 Drap&Drop functionality for own ListView's become more easily. Check ListViewIconsDragSupport and ListViewIconsDropSupport as reference (ignore TreeNode region).
- EasyLayout: fixed "dirty" scene bug when using FitContainer or ShrinkOnOverflow
- ListView's: DataSource can be safely used from other threads
- ListView's: added GroupedListView sample
- ListView's: added .Select(int index, bool raiseEvents) function, you can use it to select items without raising events
- ListView's: added Owner field to ListViewItem (base class for any DefaultItem), it contains link to parent ListView
- ListView's: you can implement IViewData<T> to DefaultItem component class to avoid overriding ListView.SetData() function
- ListView's: added virtual properties Graphic[] GraphicsForeground and Graphic[] GraphicsBackground to ListViewItem, you can them to specify grapgics for coloring, instead overriding coloring functions
- Resizable: mark events as used
- SlideBlock renamed to Sidebar
- Sidebar: added new animation types Overlay (default), Push, Uncover, ScaleDown, SlideAlong, SlideOut, Resize
- Sidebar: added AnimateWithLayout option for Resize animation, use it if you need more than one Sidebar with Resize on same Content object
- Spinner: added AllowHold option, so you can disable increasing/decreasing value during pointer hold
- Switch: added .SetStatus(bool value), you can change state without raising corresponding events
- TileView's: added TileViewCustomSize
- Tooltip: added UnscaledTime option
- TreeNode: added RootNode property, used to check if nodes belong to same tree
- TreeView's and TreeNode: Nodes type change from IObservableList<TreeNode<TItem>> to ObservableList<TreeNode<TItem>>
- TreeView: added SelectedNodes property
- TreeView: added DeselectCollapsedNodes property, enabled by default
- TreeView: added .Node2Index(TreeNode<TItem> node) function
- TreeView: added .SelectNode(TreeNode<TItem> node) and .SelectNodeWithSubnodes(TreeNode<TItem> node) functions
- TreeViewDataSource: fixed incorrect branch bug (thanks to Heiko Berres)
- ProgressBar: added SpeedType option

11.9 Release 1.8.5

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

11.10 Release 1.8.4

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

11.11 Release 1.8.3

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

11.12 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 ScrollRectRestictedDrag
- TextMeshPro support available with separate unitypackage
- Beta: Added Connectors. Add SingleConnector or MultipleConnector to empty gameobject

11.13 Release 1.8.0

- Added ScrollRectPaginator
- Added ListViewPaginator
- Added Autocomplete
- Added Popup
- TreeView: added TreeViewDataSource component with nodes editor
- ListView's: added ScrollTo()
- EasyLayout: reduced memory allocation
- EasyLayout: added row/column constraint for Grid layout
- Tabs: added DefaultTabName property
- TreeNode: added Path property - return list of parent nodes
- TreeViewComponent: added OnNodeExpand property with Rotate (rotate toggle) and ChangeSprite (change toggle sprite) values
- Notify and Dialog: added Template() method, now you can use notifyPrefab.Template().Show(...) instead Notify.Template("template name").Show(...)
- CenteredSlider: added ValueMin, ValueMax and UseValueLimits. If UseValueLimits enabled then ValueMin <= Value <= ValueMax
- Tabs: added TabButtonComponent, use derived class with overridden SetButtonData() to control how tab name will be displayed. For TabsIcons you can use TabIconButton.
- Dialog: added DialogButtonComponent, use derived class with overridden SetButtonName() to control how button name will be displayed.
- Dialog: added DialogInfoBase, use derived class with overridden SetInfo() to control how info will be displayed.
- ListView's, TileView: added DropIndicator for Drag-and-Drop
- TileView: added TileViewScrollRectFitter, ScrollRect will be resized to display whole number of items.

11.14 Release 1.7.4

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

11.15 Release 1.7.2

- Fixed errors in WinStore builds.
- IDropSupport: added DropCanceled method.

- DragSupport: added DragPoint property (empty gameobject on cursor/touch position), you can use it to attach custom gameobject with information about draggable object.
- ListViewIconsDragSupport, TreeViewNodeDragSupport: show information about draggable object.
- Tabs: added Tabs with icons.

11.16 Release 1.7.0

- Added Drag and Drop support.
- ComboboxCustom and ComboboxIcons: Added Multiselect support.
- ResizableHeader: Added drag column support.
- TreeViewItem: Added Tag property.
- SlideBlock: Optional support for children ScrollRect.
- Accordion: Added Direction.
- Accordion: Added support Horizontal Layout Group and Vertical Layout Group (Content Objects should have LayoutElement component).
- ListViews: Added limited support Horizontal Layout Group and Vertical Layout Group (you cannot change ListView direction in runtime).
- ObservableList: Added events OnCollectionChange (raised when items added, removed or replaced) and OnCollectionItemChange (raised when item in collection raise OnChange or PropertyChanged events).
- ObservableList: Added Comparison, ResortOnCollectionChanged, ResortOnCollectionItemChanged properties.
- TreeNode: Added Parent property. Now you can remove node from tree using Node.Parent = null or move node to another subtree Node.Parent = AnotherNode.

11.17 Release 1.6.5

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

11.18 Release 1.6.0

- ColorPicker
- For ListView, ListViewIcons, ListViewCustom, ListViewCustomHeight, TileView added support for ObservableList
- Items property marked obsolete but can be used.
- Added optional sequence parameters for Notify - notifications can be showed one by one, not only all at once like before.
- For ListViewIcons items and TreeView nodes added field LocalizedName, so now can be easily added localization support.
- **EasyLayout - Control Width, Max Width, Control Height, Max Height replaced with "Children Width" and "Children H**
  - Do Nothing
  - Set Preferred - Set width/height to preferred, like Control Width/Height
  - Set Max from Preferred - Set width/height to maximum preferred width/height of items, like Max Width/Height
  - Fit Container - similar to "Child Force Expand" from Horizontal/Vertical Layout Group
- ListViewCustomHeight - implementation of IListViewItemHeight for components now optional, but you still can implement it for optimization purpose.

11.19 Release 1.5.0

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

11.20 Release 1.4.2

- Added ListViewCustomHeight (support items of variable heights)

11.21 Release 1.4.1

- Added CenteredSlider.

11.22 Release 1.4

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

11.23 Release 1.3

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

11.24 Release 1.2

- Added Dialog
- Added Draggable

11.25 Release 1.1

- Added Notify
- Added EasyLayout

11.26 Release 1.0

- Initial release