New UI Widgets Documentation

Release 1.18.6f1

Ilia Novikov

Oct 11, 2025

CONTENTS

CHAPTER

ONE

OVERVIEW

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

Video Tutorials

1.1 Recommended Unity UI documentation

- Working with UI in Unity
- UI Components
- Layout Groups

1.2 Collections

Collections for your custom types can be created by *Widgets Generator*.

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

- *AutocompleteCombobox*
- *AutoComboboxIcons*
- *AutoComboboxString*
- *Combobox*

  Data type `string`.
- *ComboboxInputField*
- *ComboboxEnum*
- *ComboboxEnumMultiselect*
- *ComboboxIcons*
- *ComboboxIconsMultiselect*

  ComboboxIcons with multiple selection support.
- *DirectoryTreeView **
- *FileListView **
- *ListView, TileView and Table (DataGrid)*

Data type `string`.

- *ListViewColors*

  Data type `Color`.

- *ListViewInt*

  Data type `int`.

- *ListViewIcons*
- *ListViewHeight*

  Data type `string`.

- *ListViewPaginator*

  Paginator for the ListView, TileView, and Table.

- *TreeView*

1.3 Containers

- *Accordion*
- *Tabs*

  Tabs buttons displayed on the top side.

- *TabsLeft*

  Tabs buttons displayed on the left side.

- *TabsIcons*

  Tabs buttons with an icon and buttons displayed on the top side.

- *TabsIconsLeft*

  Tabs buttons with an icon and displayed on the left side.

1.4 Controls

- ButtonBig
- ButtonSmall
- *ContextMenu Template*

  Template of the context menu to use by `ContextMenu` component

- *ScrollRectPaginator*

  Paginator for the ScrollRect.

- *ScrollRectNumericPaginator*

  Paginator for the ScrollRect. Navigation can display a number of a page.

- *Sidebar*
- *SplitButton*

  Button with a dropdown for the additional buttons.

1.5 Dialogs

- *DatePicker*

  Data type `DateTime`.

- *DateTimePicker*

  Data type `DateTime`.

- *Dialog Template*

  Template for the custom dialogs.

- *FileDialog **

  Dialog to select the file.

- *FolderDialog **

  Dialog to select the folder.

- *NotifyTemplate*

  Template for the custom notifications.

- *PickerBool*

  Data type `bool`.

- *PickerIcons*

- *PickerInt*

  Data type `int`.

- *PickerString*

  Data type `string`.

- *Popup*

  Template for the custom popup.

- ***TimePicker***
  Data type `TimeSpan`.

1.6 Input

- *Autocomplete*

  Data type `string`.

- *AutocompleteIcons*

- *Calendar*

- *CenteredSlider*

  Horizontal direction.

- *CenteredSliderVertical*

  Vertical direction.

- *CircularSlider*

- *CircularSliderFloat*
- *ColorPicker*
- *ColorPickerRange*
- *ColorPickerRangeHSV*
- *ColorsList*

  Used with ColorPicker to save the selected colors.
- *DateTime*

  Data type `DateTime`.
- *DateTimeScroller*

  Data type `DateTime`.
- *DateTimeScrollerSeparate*

  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.
- *Rating*
- *Scale*
- *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.
- *TimeAnalog*
- *TimeScroller*

1.7 Misc

- *AudioPlayer*
- *Loading Animation*
- *ProgressbarDeterminate*
- *ProgressbarCircular*

  Same as ProgressbarDeterminate, but progress displayed in circle instead of line.
- *ProgressbarIndeterminate*
- *Simple Tooltip*
- *TooltipString*

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

1.8 How to Replace Default Prefabs

- Create a copy of the *New UI Widgets / Assets / UI Themes / PrefabsThemes.asset*
- Replace references with your prefabs
- Set created copy in the **Current** field in the *New UI Widgets / Editor / Widgets References.asset* (available only after any widget was created with the context menu)

CHAPTER

TWO

PROJECT SETTINGS

Settings are located at *Edit / Project Settings… / New UI Widgets*.

2.1 Assembly Definitions

Enable/disable assembly definitions. Enabled by default.
In the case of supported third-party packages without assembly definitions you need to create assembly definitions and specify them as references in the `UIWidgets.asmdef`.

2.2 Instantiate Widgets

Create widgets as copies of prefabs (same as default Unity widgets) or as prefabs references. Create copies by default.

2.3 Widgets Generator: R3 Support

Enable/disable R3.Unity support for the Widgets Generator.
If enabled fields and properties of type `SerializableReactiveProperty<T>` will be correctly processed.

2.4 Styles or Themes

Use *Styles (Legacy)* or *UI Themes* for the widget customization. Themes by default.

2.5 UI Themes: Addressables Support

Enable/disable Addressables support for the *UI Themes*.
Requires installed `Addressables` package (*Window / Package Manage / Unity Registry*).

2.6 Attach Default Theme

Attach default theme to the widgets created from menu.

2.7 Use White Sprite

Sets white sprite for the Image components without sprite.
Prevents rare bugs when such Images are displayed as black.

2.8 TextMeshPro Support

Enable/disable *TextMeshPro Support*. Enabled by default if the TextMeshPro is installed.

2.9 Unity Localization Support

Enable/disable *Unity Localization Support*.

2.10 I2 Localization Support

Enable/disable *I2 Localization Support*.

2.11 Data Bind for Unity Support

Enable/disable *Data Bind for Unity Support*.

2.12 Widgets Generator Settings

Default values for the *Widgets Generator*, you can later change actual values in the generator window.

- *Namespace*

  Namespace for the created scripts.

- *Editor Namespace*

  Namespace for the created editor scripts.

- *Scene Path*

  Path to save created scene, theme and some other files.

- *Scripts Path*

  Path to save created scripts.

- *Prefabs Path*

  Path to save created prefabs.

- *Editor Path*

  Path to save created editor scripts.

Supported Placeholders:

- *DataTypeNamespace*

  Namespace of the your data type.

- *DataTypePath*

  Path to your data type or to the selected object in the Project window.

- *DataTypeName*

Name of your data type.

2.13 UI Themes Settings

- *Attach to UI only*

  Add *ThemeTarget* component only to objects with *RectTransform* component.

- *Attach Default Selectable Colors*

  If enabled, then default *Selectable* colors will be controlled by *Theme*.

- *Wrappers Folder*

  Path to save created wrappers scripts.

- *Wrappers Namespace*

  Namespace for the wrappers scripts.

- *Generate Wrappers*

  If enabled, then automatically generate wrapper scripts for properties which available only via reflection after using the *Theme Attach* command.

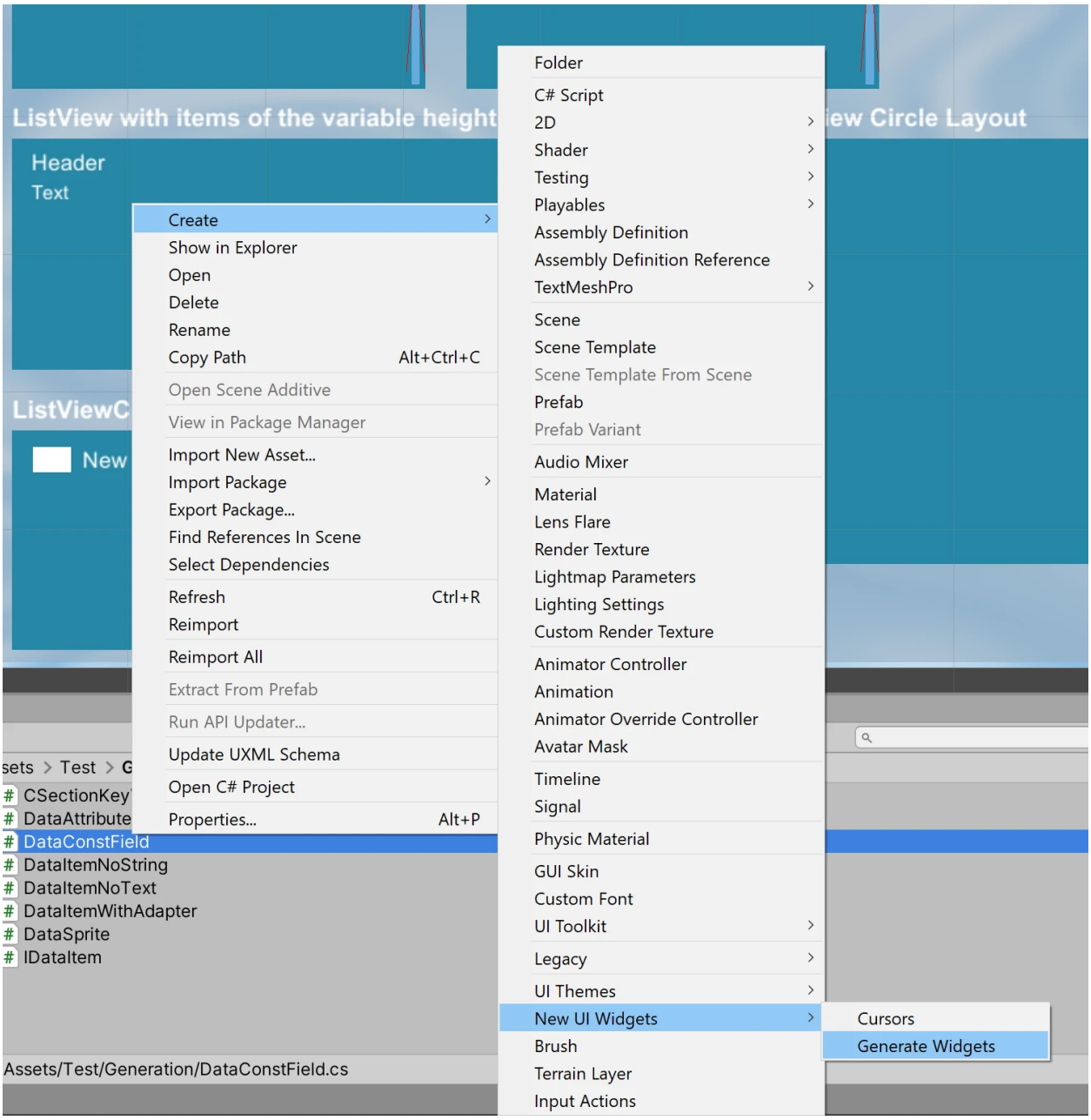
Note: Support is enabled only to installed platforms. Platforms that were added after it requires enabling support again.

CHAPTER

THREE

WIDGETS GENERATOR

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



3.1 List of Generated Widgets

- *Autocomplete*: requires at least one field or property of `string` type.
- *AutoCombobox*: `Combobox` with `Autocomplete` to filter and select items by typing; requires at least one field or property of `string` type.
- *Combobox*
- *ComboboxMultiselect*: same `Combobox` configured to display multiple selected values.
- *DragInfo*: displays content of dragged data
- *ListView*

- *Table (DataGrid)*
- *TileView*
- *Tooltip*
- *TreeGraph*
- *TreeView*
- *PickerListView*: `Picker` to select the value from `ListView`
- *PickerTreeView*: `Picker` to select the value from `TreeView`

3.2 Requirements

The data type should have at least one public field or public readable property of the supported types.
To be available in the inspector window the data type should have `[System.Serializable]` attribute.
Some features of the `Autocomplete`, `AutoCombobox`, and `ListView` require the data type to implement the `IEquatable<T>` interface to work correctly.

3.3 Supported types

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

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

Graphic types:

- `Sprite`
- `Texture`, `Texture2D`
- `Color`
- `Color32`

3.4 Limitations

- Autocomplete

  Requires at least one field or property of the `string` type.

- Table

  Requires at least one field or property of the text type.

3.5 Attributes

- `[GeneratorPaths]`: specify paths for the created scripts, prefabs, scenes
- `[GeneratorNamespace]`: specify namespaces for the created scripts
- `[GeneratorIgnore]`: mark fields or properties that should not be used in widgets
- `[GeneratorAutocomplete]`: mark the field or property that should be used for autocomplete (will be used only first field with this attribute)

```
[GeneratorPaths(
        "Assets/Widgets/Scenes",
        "Assets/Widgets/Scripts",
        "Assets/Widgets/Editor",
        "Assets/Widgets/Prefabs")]
[GeneratorNamespace("Widgets", "Widgets.Editor")]
public class DataSprite
{
        [GeneratorAutocomplete]
        public string Name;

        public string Text;

        public Sprite Icon;

        [GeneratorIgnore]
        public Texture Texture;
}
```

3.6 Known Problems

Widget generator does not work with `struct` or `interface` types inside a namespace with some Unity versions due to bug.

Workaround

Specify the type name in the *Data Type* field.
Another way is to change `interface` or `struct` to `class` in the type definition. Then run widgets generator and return type to `interface` or `struct`.

3.7 Extending and Overriding Classes

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

3.8 INotifyPropertyChanged and IObservable Support

`ObservableList<T>` used by widgets provide support for `INotifyPropertyChanged` and `IObservable` interface of the data type, so widget will be updated if property changed and was raised corresponding event.
If you want to automatically update collections widgets (like ListView, TileView, Table) on item data changes, then you need to add `INotifyPropertyChanged` or `IObservable` implementation to your data type.
Implementation can be added even after widgets generator.
The `IObservable` interface is preferable if you want to reduce memory allocations.

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

   public string Name
   {
      get
      {
         return name;
      }

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

   public event PropertyChangedEventHandler PropertyChanged;

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

   ...
}
```

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

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

(continues on next page)

(continued from previous page)

```
        }

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

    public event OnChange OnChange;

    protected void Changed()
    {
        OnChange?.Invoke(this);
    }

    ...
}
```

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";
```

3.9 Replacing generated code

Generated code can be freely modified.

Important:

Be careful not to overwrite modified scripts if you decide re-run widget generator for the same data type.

3.9.1 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 generator for your type and you will need only to modify created DefaultItem class if it needs at all.

Functions to modify in the DefaultItem class:

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

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

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

Original code:

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

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

      ...

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

      /// <summary>
      /// Sets component data with specified item.
      /// </summary>
      /// <param name="item">Item.</param>
```

(continues on next page)

(continued from previous page)

```
      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)
{
```

(continues on next page)

(continued from previous page)

```
    Item = item;

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

    ...
}
```

3.9.2 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.
```

(continues on next page)

(continued from previous page)

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

         return value.SomeOtherText.ToLower().EndsWith(Query.ToLower());
      }
   }
}
```

CHAPTER

FOUR

WIDGETS

4.1 Collections

4.1.1 AutoCombobox

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

Note:

Difference between Autocomplete, AutoCombobox, and AutocompleteCombobox:
- Autocomplete is InputField with autocomplete feature.
- AutoCombobox is Combobox with the option to select items by typing, with it you can get selected items.
- AutocompleteCombobox is a wrapper for Autocomplete with the ability to select an action when user input is not valid.

Options

- Autocomplete `TAutocomplete`

  ListView with items.

- Combobox `TCombobox`

  Button to show and hide ListView on click.

- AddItems `bool`

  Create a new item and add it to list if item not found with specified input. Requires overrided `TItem Input2Item(string input)` method.

- KeepSelection `bool`

  Keep selected items for Autocomplete.DisplayListView.

Events

- OnItemSelected `UnityEvent<int, TItem>`

  The event raised when item selected.

  Arguments: index of the selected item and selected item.

- OnItemDeselected `UnityEvent<int, TItem>`

  The event raised when item deselected.

  Arguments: index of the deselected item and deselected item. | If an item associated with this index is removed the index can be invalid (`>= DataSouce.Count`) or point to different item. | The item is not valid if list is cleared or selected item was removed.

4.1.2 AutocompleteCombobox

Wrapper for *Autocomplete* with the ability to select an action when user input is not valid.

Note:

Difference between Autocomplete, AutoCombobox, and AutocompleteCombobox:
- Autocomplete is InputField with autocomplete feature.
- AutoCombobox is Combobox with the option to select items by typing, with it you can get selected items.
- AutocompleteCombobox is a wrapper for Autocomplete with the ability to select an action when user input is invalid.

Options

- Autocomplete `AutocompleteString`

  Autocomplete.

- AutocompleteToggle `Button`

  Button to show autocomplete values.

- IfInvalid `InvalidMode`

  Action when user input is invalid: `Ignore`, `FocusInputField`, `ResetInputField`.

MultipleSelect Combobox with Autocomplete

How to combine `ComboboxMultiselect` with `Autocomplete`:

- Create `ComboboxMultiselect` and `Autocomplete` of required type
- move `Autocomplete.InputField` next to `Combobox.Current` and add `LayoutElement` with specified `MinWidth`
- move `DisplayListView` and `TargetListView` next to `Combobox.ListView` and add `LayoutElement` with enabled `ignoreLayout`
- copy `RectTransform` settings and `DataSource` from `Combobox.ListView` to the `DisplayListView` and `TargetListView`
- `TargetListView`: enable `MultipleSelect`

- `Combobox`: specify `TargetListView` as `Combobox.ListView`
- delete or disable the previous `Combobox.ListView`
- `Autocomplete`: `OnOptionSelected` event: add reset `InputField.text`

4.1.3 Combobox

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

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

Options

- Interactable `bool`

  Allow users interact with the Combobox.

- ListView `TListViewCustom`

  ListView with items.

- RepositionListView `bool`

  If enabled ListView is automatically positioned to be completely visible if partially hidden by the bottom or right side of the screen.

- ChangeRounderCorners `bool`

  If enabled then corner radiuses will be changed to match the repositioned ListView.

- ToggleButton `Button`

  Button to show and hide ListView on click.

- Current `TComponent`

  Template to display selected items.

- HideAfterItemToggle `bool`

  Hide ListView right after item selected or deselected.

Events

- OnShowListView `UnityEvent`

  The event raised when ListView showed.

- OnHideListView `UnityEvent`

  The event raised when ListView hidden.

- OnCurrentClick `UnityEvent<int, TItem>`

  The event raised on click on displayed selected item.

4.1.4 ComboboxInputField

Combobox with the ability to select and add new items by typing.

Options

- ListView `TListViewCustom`

  ListView with items.

- ToggleButton `Button`

  Button to show and hide ListView on click.

- Current `TComponent`

  Template to display selected items.

- HideAfterItemToggle `bool`

  Hide ListView right after item selected or deselected.

- Allow New Items `bool`

  Allow to add new items by typing.

- Reset Input `bool`

  Reset `InputField` if item not found and new items not allowed.

Events

- OnShowListView `UnityEvent`

  The event raised when ListView showed.

- OnHideListView `UnityEvent`

  The event raised when ListView hidden.

- OnCurrentClick `UnityEvent<int, TItem>`

  The event raised on click on displayed selected item.

4.1.5 DirectoryTreeView

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

Options

Options are almost same as the *TreeView*.

- Data Source `ObservableList<TreeNode<FileSystemEntry>>`

  Not available in the inspector window.
  Filled automatically.

- Root Directory `string`

  Root directory.

- Exceptions View `IOExceptionsView`

  Special component to display IO errors.

Methods

- `TreeNode<FileSystemEntry> ExpandPath(string path, bool scrollToNode = true)`

  Expand nodes to the specified path. Returns `null` if node not found.

- `TreeNode<FileSystemEntry> Path2Node(string path)`

  Get node of the specified path. Returns `null` if node not found.

- `TreeNode<FileSystemEntry> Path2NearestNode(string path)`

  Get exact node or nearest existing parent node of the specified path.

- `void RefreshDirectories()`

  Refresh displayed directories according to current state of the file system.

4.1.6 FileListView

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

Options

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

- Data Source `ObservableList<FileSystemEntry>`

  Not available in the inspector window.
  Filled automatically.

- Current Directory `string`

  Current directory. `Application.persistentDataPath` will be used if not specified.

- Directory Patterns `string`

  Directory patterns, semicolon used as separator between patterns.
  Directory will be displayed if it's match one of the pattern.
  Wildcards:

* - Zero or more characters in that position.
? - Zero or one character in that position.
Warning: if directory match two or more patterns it will be displayed two or more times.

- File Patterns `string`

  File patterns, semicolon used as separator between patterns.
  File will be displayed if it's match one of the pattern.
  Wildcards:
  * - Zero or more characters in that position.
  ? - Zero or one character in that position.
  Warning: if file match two or more patterns it will be displayed two or more times.

- Button Up `Button`

  Button to open parent directory of current directory.

- Button Toggle Drivers `Button`

  Button to toggle DriversList.

- Path View `FileListViewPath`

  Widget to display the current directory.

- Drives List View `DrivesListView`

  Widget to display drives list.

- Exceptions View `IOExceptionsView`

  Special component to display IO errors.

- Can Display Entry `Func<FileSystemEntry, bool>`

  Not available in the inspector window.
  Function to check if `FileSystemEntry` should be displayed.

4.1.7 Grouped ListView and TileView

You can create grouped ListView with `GroupedList<TItem>` (group items does not exists and will be automatically created) or `LinearGroupedList<TItem>` (group items already exists in `DataSource`).

Grouped ListView

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

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

(continues on next page)

(continued from previous page)

```
        {
            var name = item.Name.Length > 0 ? item.Name[0].ToString() : string.Empty;

            foreach (var group in groups)
            {
                if (group.Name == name)
                {
                                        return group;
                                    }
                            }

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

    bool isGroupedViewInited;

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

        isGroupedViewInited = true;

        base.Init();

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

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

Grouped TileView

```
using UIWidgets;
using UnityEngine;

public class GroupedTileViewSelector : MonoBehaviour, IListViewTemplateSelector
↪<GroupedListViewComponent, GroupedItem>
{
    [SerializeField]
    protected GroupedListViewComponent HeaderTemplate;

    [SerializeField]
    protected GroupedListViewComponent HeaderEmptyTemplate;

    [SerializeField]
    protected GroupedListViewComponent ItemTemplate;
```

(continues on next page)

(continued from previous page)

```
    [SerializeField]
    protected GroupedListViewComponent ItemEmptyTemplate;

    GroupedListViewComponent[] templates;

    public GroupedListViewComponent[] AllTemplates()
    {
        if (templates == null)
        {
            templates = new[] { this.headerTemplate, this.headerEmptyTemplate, this.
↪itemTemplate, this.itemEmptyTemplate, };
        }

        return templates;
    }

    public GroupedListViewComponent Select(int index, GroupedItem item)
    {
        if (item.IsGroup)
        {
            return item.IsEmpty ? headerEmptyTemplate : headerTemplate;
        }

        return item.IsEmpty ? itemEmptyTemplate : itemTemplate;
    }
}

public class GroupedTileView : ListViewCustom<GroupedListViewComponent, GroupedItem>
{
    public GroupedList<Photo> GroupedData = new GroupedList<Photo>((groups, item) =>
    {
        var date = item.Created.Date;

        foreach (var group in groups)
        {
            if (group.Created == date)
            {
                return group;
            }
        }

        return new Photo() { Created = item.Created.Date, IsGroup = true };
    });

    bool isGroupedListViewInited;

    public override void Init()
    {
        if (isGroupedListViewInited)
        {
            return;
```

(continues on next page)

(continued from previous page)

```
        }

        isGroupedListViewInited = true;

        base.Init();

        GroupedData.GroupComparison = (x, y) => x.Created.CompareTo(y.Created);
        GroupedData.EmptyGroupItem = new Photo() { IsGroup = true, IsEmpty = true };
        GroupedData.EmptyItem = new Photo() { IsEmpty = true };
        GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();

        GroupedData.Output = DataSource;
    }

    public override void UpdateItems()
    {
        base.UpdateItems();

        GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();
    }

    public override void Resize()
    {
        base.Resize();

        GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();
    }
}
```

Linear GroupedTileView

```
public class LinearGroupedTileViewSelector : MonoBehaviour, IListViewTemplateSelector
↪<GroupedListViewComponent, GroupedItem>
{
    [SerializeField]
    protected GroupedListViewComponent HeaderTemplate;

    [SerializeField]
    protected GroupedListViewComponent HeaderEmptyTemplate;

    [SerializeField]
    protected GroupedListViewComponent ItemTemplate;

    [SerializeField]
    protected GroupedListViewComponent ItemEmptyTemplate;

    GroupedListViewComponent[] templates;

    public GroupedListViewComponent[] AllTemplates()
    {
```

(continues on next page)

(continued from previous page)

```
      if (templates == null)
      {
         templates = new[] { this.headerTemplate, this.headerEmptyTemplate, this.
↪itemTemplate, this.itemEmptyTemplate, };
      }

      return templates;
   }

   public GroupedListViewComponent Select(int index, GroupedItem item)
   {
      if (item.IsGroup)
      {
         return item.IsEmpty ? headerEmptyTemplate : headerTemplate;
      }

      return item.IsEmpty ? itemEmptyTemplate : itemTemplate;
   }
}

public class LinearGroupedTileView : ListViewCustom<GroupedListViewComponent,␣
↪GroupedItem>
{
   // Real DataSource (use instead of DataSource).
   public ObservableList<GroupedItem> RealDataSource = new ObservableList<GroupedItem>();

   public LinearGroupedList<GroupedItem> GroupedData = new LinearGroupedList<GroupedItem>
↪(x => x.IsGroup);

   bool isGroupedListViewInited;

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

      isGroupedListViewInited = true;

      base.Init();

      GroupedData.EmptyHeaderItem = new GroupedItem() { IsGroup = true, IsEmpty = true };
      GroupedData.EmptyItem = new GroupedItem() { IsEmpty = true };
      GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();

      GroupedData.Input = RealDataSource;
      GroupedData.Output = DataSource;
   }

   public override void UpdateItems()
   {
```

(continues on next page)

(continued from previous page)

```
        base.UpdateItems();

        GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();
    }

    public override void Resize()
    {
        base.Resize();

        GroupedData.ItemsPerBlock = ListRenderer.GetItemsPerBlock();
    }
}
```

4.1.8 ListView, TileView and Table (DataGrid)

Note: **Table** is **ListView** with specific `DefaultItem` and *Table Header* (it also provides **Table** specific methods). Widget with scripts should be created by *Widgets Generator*.

Note: In case of noticeable artifacts when scrolling (these are caused by rounding during rendering), you can increase the font size to reduce the artifacts.

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

List View Type

Fig. 1: ListView with Fixed Size.

Fig. 2: ListView with Variable Size.

Note:

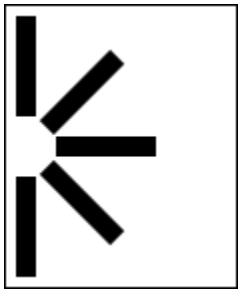


Fig. 3: ListView with Ellipse layout.

Fig. 4: TileView with Fixed Size.

Fig. 5: TileView with Variable Size.

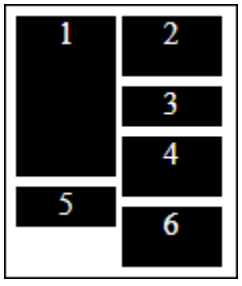


Fig. 6: TileView Staggered.

Requirements for ListView and TileView with Variable Size:
Instance sizes are stored in the `Dictionary<TItem, Vector2>` and different instances of the same item should have the same size.
ListView will not work correctly if sizes are different for the instances of the same item.

EasyLayout Settings for the Ellipse Type

- RectTransform.pivot

  Defines on which side or corner will be the center point.

- EasyLayout.Ellipse settings

  Width and height usually should be specified, set the same value for the circle.

- *Angle Start*

  Base rotation for the first item.

- *Angle Step Auto*

  Should be disabled.

- *Angle Step*

  Angular distance between items.

- *Fill*

  Should be `Arc`.

- *Arc Length*

  Should be **180** if center at the side and **90** if center at the corner.

Options

- Interactable `bool`

  Allow users interact with the ListView.

- Disable ScrollRect `bool`

  If ListView is disabled then ScrollRect and scrollbars will be disabled too.

- Virtualization `bool`

  Enable virtualization. If enabled GameObject instantiated only for the visible items; otherwise for the all items.

- List Type `ListViewType`

  Determines how items are displayed.

  - `ListViewWithFixedSize`

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

  - `ListViewWithVariableSize`

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

  - `ListViewEllipse`

Works with EasyLayout. Axis of rotation is RectTransform.pivot.
It is recommended to use with enabled LoopedList option.

- `TileViewWithFixedSize`

  Works with EasyLayout.

- `TileViewWithVariableSize`

  Works with EasyLayout.

- `TileViewStaggered`

  Works with EasyLayout.

- ChangeLayoutType `bool`

  EasyLayout settings will be changed according current list type if this option enabled.

- StretchToMaxItemSize `bool`

  This option will stretch ListView to fully display the largest item, and works only with `ListViewWithVariableSize`.
  Mainly it is used for the `Combobox.ListView`.

- PrecalculateItemSizes `bool`.

  When enabled, item sizes are calculated at the start, allowing for precise scrolling. Disabling this option will improve performance.
  Disable it if any item data (such as sprites) is loaded asynchronously or via lazy loading.
  Available only for the VariableSize types.

- Sort `bool` *deprecated*

  Available only for `ListViewIcons`. If enabled items will be sorted by name.
  Deprecated, replaced with DataSource.Comparer.

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

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

- Data Source `ObservableList<TItem>`

  List of the items. It works the same way as List<T> with some additions.
  Not available in the inspector window if type not specified as serializable.

- ReversedOrder `bool`

  If enabled first item will be displayed at the end of the list, initial scroll position also will be at the end.
  Can be used for horizontal lists in cultures with right-to-left writing directions.
  Not compatible with `TileViewStaggered`.

- Selected Index `int`

  Index of the last selected item.

- Multiple Select `bool`

  Allow to select multiple items, otherwise only one.

- Multiple Select Require Keys `bool`

  Items selection will work similarly to file explorer if enabled.

- Range Mode `RangeSelectionMode`

  Specify range selection mode (multiple items selection with `Shift` key).

  - `StartFromFirst`

    Select all items from the first selected item to the newly selected item.

  - `StartFromLast`

    Select all items from the last selected item to the newly selected item.

- Selected Indices `List<int>`

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

- Selected Item `TItem`

  Not available in the Inspector window Last selected item.

- Selected Items `List<TItem>`

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

- Direction `ListViewDirection`

  ListView direction.

  - `Horizontal`
  - `Vertical`

- ScrollRect `ScrollRect`

  ScrollRect used by ListView. Required for virtualization support.

- Container `Transform`

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

- SetContentSizeFitter `bool`

  If enable changes ContentSizeFitter settings according to the selected direction.

- Default Item `TComponent`

  A prefab used to display item.

- DestroyDefaultItemsCache `bool`

  If enabled then instances of the previous DefaultItem will be destroyed when `DefaultItem` or `TemplateSelector` are changed.

- Allow Coloring `bool`

  Change colors of the highlighted and selected items.

  If you want to more precise control on item colors, like different colors depending of item data, then you can override `StateDefault()`, `StateHighlighted()`, `StateSelected()`, and `StateDisabled()` methods of `TComponent` class.

- Coloring Striped `bool`

  Use different background colors for the odd and even items.

- Colors

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

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

- Default Color `Color`
- Default Background Color `Color`: only if Coloring Striped disabled
- Default Odd Background Color `Color`: only if Coloring Striped enabled
- Default Even Background Color `Color`: only if Coloring Striped enabled
- Highlighted Color `Color`
- Highlighted Background Color `Color`
- Selected Color `Color`
- Selected Background Color `Color`
- Disabled Color `Color`: multiplier for the specified colors if ListView is not interactable, 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.

- Keep Highlight `bool`

  Keep item highlight on pointer enter until will be selected another gameobject.

- Only One Highlighted `bool`

  Allows only one highlighted item. If disabled then two can be highlighted: first from pointer over, second from navigation by keyboard or gamepad.

- Navigation `bool`

  Allow to use navigation with keyboard or gamepad.
  ListView should have a Selectable component to navigation work properly.

- Looped Navigation `bool`

  Enables/disables navigation from first to last item and vice versa.

- Require Visible `int`

  Number of visible items from current item to border on navigation.

- Style Table `bool`

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

- Header `TableHeader`

  Deprecated field and should be used only if legacy styles are enabled.

- LocalizationSupport `bool`

  If enabled item names will be translated with `Localization.GetTranslation()` method if localization is used.
  Available only for the `ListViewIcons`.

- Can Select `Func<int, bool>`

  The function that determines whether the item with the specified index can be selected. Unselectable items cannot be highlighted and skipped by keyboard and gamepad navigation.

- Can Deselect `Func<int, bool>`

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

Scroll-related Options

- Retain Scroll Position `bool`

  If enabled ListView will try to retain the scroll position when items are added or deleted from the list.
  This option uses positions of the visible items if all visible items were deleted then the scroll position cannot be corrected.

- Center The Items `bool`

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

- Looped List `bool`

  Is the list looped? The first items will be displayed after the last item and scrolling is infinite. It's recommended to use it without scrollbars.

  Limitations:

  - The `Container.EasyLayout.Margin` for the scrollable axis should be zero
  - `ScrollRect.movementType` = `Unrestricted`
  - supported only ListView (TileView and TreeView are not supported)

- Scroll Unscaled Time `bool`

  Specify time type used by scroll animation.
  If enabled then will be used `Time.unscaledTime`; otherwise will be used `Time.time`.

- Scroll Movement `AnimationCurve`

  Animation curve for the `ScrollToAnimated` methods.
  Specify how long scroll animation will be and what speed will it have.

- Scroll Inertia Until Item Center `bool`

  Enable custom scroll inertia.
  It is replace ScrollRect inertia in such a way so that after the end of scrolling the item will be exactly in the center.
  Intended to use with `ListViewEllipse` but works with other types too.

- Scroll Inertia `AnimationCurve`

  Similar to Scroll Movement, but only for the scroll inertia.

- End Scroll Delay `float`

  The time from last scroll event until **OnEndScrolling** event raising.
  It is better to use UtilitiesScrollRect.TimeToStop() method instead.

- Auto Scroll Area `float`

  ListView will be automatically scrolled if the pointer in less then a specified distance from the border during drag&drop.

- Auto Scroll Speed `float`

  Speed of auto-scroll.

Events

- OnSelect `UnityEvent<int, ListViewItem>`

  The event raised when item selected.

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

- OnDeselect `UnityEvent<int, ListViewItem>`

  The event raised when item deselected.

  Arguments: index of the deselected item and DefaultItem instance for the deselected item.
  If an item associated with this index is removed the index can be invalid (`>= DataSouce.Count`) or point to different item.

- OnSelectObject `UnityEvent<int>`

  The event raised when item selected.

  Arguments: index of the selected item.

- OnDeselectObject `UnityEvent<int>`

  The event raised when item deselected.

  Arguments: index of the deselected item. If an item associated with this index is removed the index can be invalid (`>= DataSouce.Count`) or point to different item.

- OnItemSelected `UnityEvent<int, TItem>`

  The event raised when item selected.

  Arguments: index of the selected item and selected item.

- OnItemDeselected `UnityEvent<int, TItem>`

  The event raised when item deselected.

  Arguments: index of the deselected item and deselected item. | If an item associated with this index is removed the index can be invalid (`>= DataSouce.Count`) or point to different item. | The item is not valid if list is cleared or selected item was removed.

- OnStartScrolling `UnityEvent`

  The event raised when scrolling starts.

- OnEndScrolling `UnityEvent`

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

- onSubmit `UnityEvent`

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

- onCancel `UnityEvent`

  The event raised when ListView gameobject has been deselected.

- onItemSelect `UnityEvent`

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

- onItemCancel `UnityEvent`

  The event raised when ListView item gameobject has been deselected.

- OnUpdateView `UnityEvent`

  The event raised when ListView view was updated.

- OnFocusIn `UnityEvent<BaseEventData>`

  The event raised when ListView gameobject received focus.

- OnFocusOut `UnityEvent<BaseEventData>`

  The event raised when ListView gameobject lost focus.

- OnPointerEnterObject `UnityEvent<int>`

  The event raised when pointer entered on ListView item gameobject.

  Arguments: index of the item.

- OnPointerExitObject `UnityEvent<int>`

  The event raised when pointer exited on ListView item gameobject.

  Arguments: index of the item.

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

  The event raised when DataSource replaced with the new list.

  Arguments: ListView instance.

Items Events

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

- PointerClick `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on every pointer click on item instance.

- FirstClick `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on first pointer click with left mouse button on item instance.

- DoubleClick `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on second pointer click with left mouse button on item instance.

- PointerUp `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on pointer up on item instance.

- PointerDown `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on pointer down on item instance.

- PointerEnter `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on pointer enter on item instance.

- PointerExit `UnityEvent<int, ListViewItem, PointerEventData>`

  The event raised on pointer exit on item instance.

- Move `UnityEvent<int, ListViewItem, AxisEventData>`

The event raised on move with keyboard or gamepad on item instance.

- Submit `UnityEvent<int, ListViewItem, BaseEventData>`

  The event raised on submit on item instance.

- Cancel `UnityEvent<int, ListViewItem, BaseEventData>`

  The event raised on cancel on item instance.

- Select `UnityEvent<int, ListViewItem, BaseEventData>`

  The event raised when item instance has been selected by EventSystem.

- Deselect `UnityEvent<int, ListViewItem, BaseEventData>`

  The event raised when item instance has been deselected by EventSystem.

- Resize `UnityEvent<int, ListViewItem, Vector2>`

  The event raised when item instance size was changed.

- MovedToCache `UnityEvent<ListViewItem>`

  The event raised before item instance recycled.
  Use this event to clean data, unload sprites, stop instance animations.

ListViewComponent Class

Component to display item.

Fields and properties

- Index `int`

  Index of the displayed item. Negative if item not displayed or not used by ListView.

- Owner `ListViewBase`

  Reference to ListView.

- DisableRecycling `bool`

  Disable recycling of this instance. Used in Drag&Drop or animations (enable at the start of the animation and disable at the end).

- GraphicsForeground `Graphic[]`

  References to the foreground objects like Text.

- GraphicsBackground `Graphic[]`

  References to the background objects.

- DisableRecycling `bool`

  If enabled prevent instance recycling until this option is disabled.

Methods

- SetData(`TItem` item)

  Set data to display.
  This method can be called when `Index < 0` if `ListType` is variable size and `PrecalculateItemSizes` enabled to calculate the exact size of the items
  Check that `Index >= 0` if there is need to do some additional actions only if item actually displayed.

- SelectItem()

  Select current item.

- DeselectItem()

  Deselect current item.

- RemoveItem()

  Remove current item from the ListView.DataSource.

- GraphicsColoring(`Color` foregroundColor, `Color` backgroundColor, `float` fadeDuration)

  Called by ListView to set colors for the GraphicsForeground and GraphicsBackground.

- MovedToCache()

  Called by ListView when GameObject moved to cache or recycled to unload unused resources like sprites.

- StateDefault()

  Called by ListView when item in the default state.

- StateSelected()

  Called by ListView when item selected.

- StateHighlighted()

  Called by ListView when item highlighted.

- `Vector2` GetInstanceSize(`int` index)

  Get the size of the instance for the item with the specified index. Used to animate items resize without problems with virtualization.

- SetInstanceSize(`int` index, `Vector2` size)

  Set the size of the instance for the item with the specified index. `UpdateView()` should be called after it to apply changes. Used to animate items resize without problems with virtualization.

- ResetInstanceSize(`int` index)

  Reset the size to the default for the item with the specified index. `UpdateView()` should be called after it to apply changes. Used to animate items resize without problems with virtualization.

Navigation

Navigation requires a `Selectable` component added to the ListView.

If the `Navigation` option is enabled then after ListView is selected you can use the submit key (*Enter* or *Space* key by default) to start navigation between items. Cancel key (*Esc* key) returns the selection to the ListView.

Navigation between items: arrow keys will select the neighbor item (next or previous item, in the case of TileView it can be the neighbor item in any direction).

If there is no item in the arrow key direction then the selected object will be determined by `Selectable` component settings.

With one exception: if the `LoopedNavigation` option is enabled then selection can go from the last item to the first item, and back.

If you need more advanced control then you can override the following methods:

- in the generated `ListViewComponent{DataType}` class (inherited from the `ListViewItem`):

`void OnMove(AxisEventData eventData)`
This method is called by EventSystem for the selected item when the navigation event is raised (arrow key pressed), and it redirects the event to the ListView.

- **in your ListView class:**
  - **`void OnItemMove(int index, ListViewItem item, AxisEventData eventData)`**
    This method processes redirected events and determines which object should be selected
  - `bool Navigate(AxisEventData eventData, int index, bool forward)`
    This method is called by the previous method to select an item if it can be selected

It is recommended to override the `OnItemMove()` method. In this method, you can determine which object should be selected next and specify it with such code:

```
eventData.Use();
eventData.selectedObject = someGameObject;
```

Auto-Resize DefaultItem Instances on ListView Resize and Keep Aspect Ratio

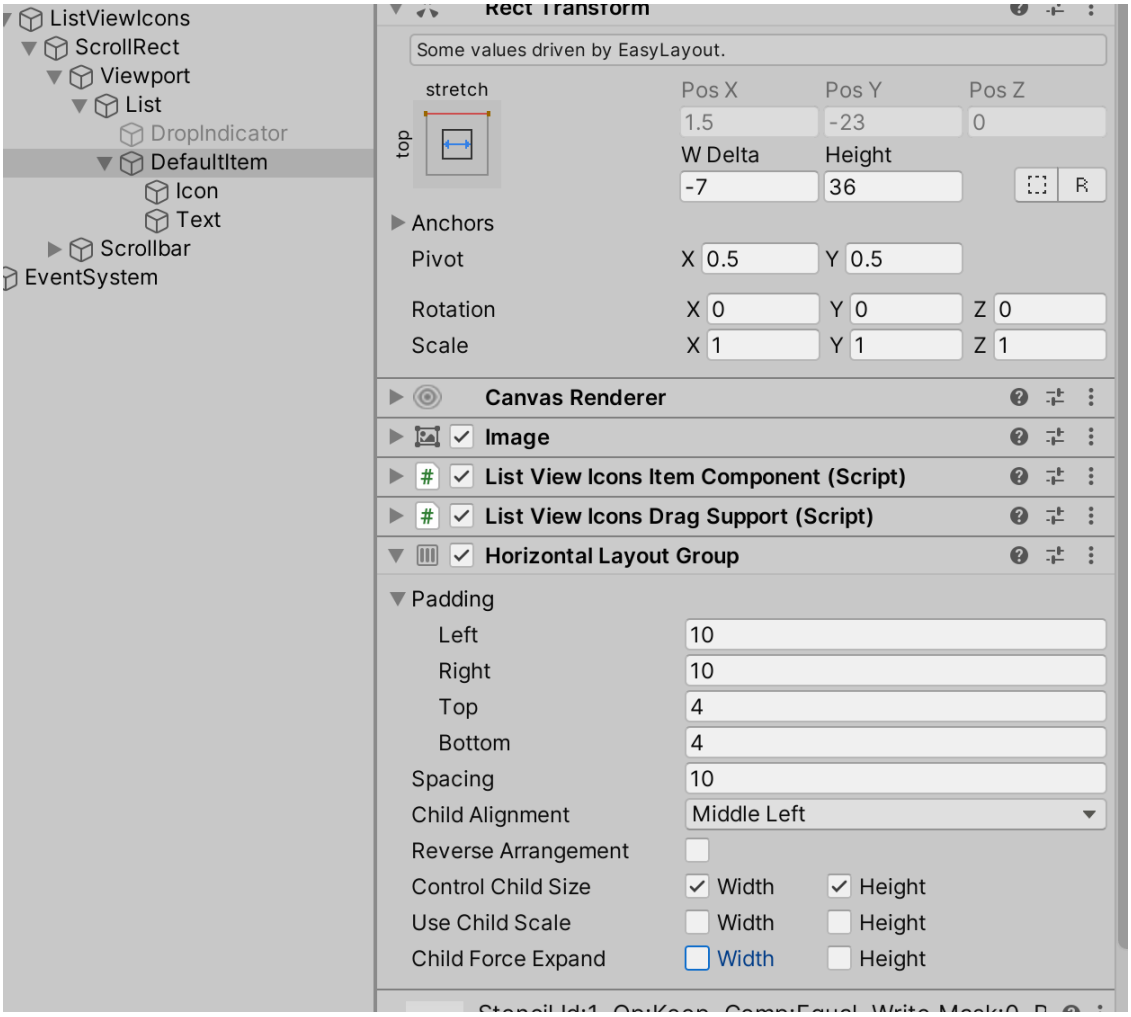
View *ListViewDefaultItemAutoResize*.

ListView with Items of Variable Size

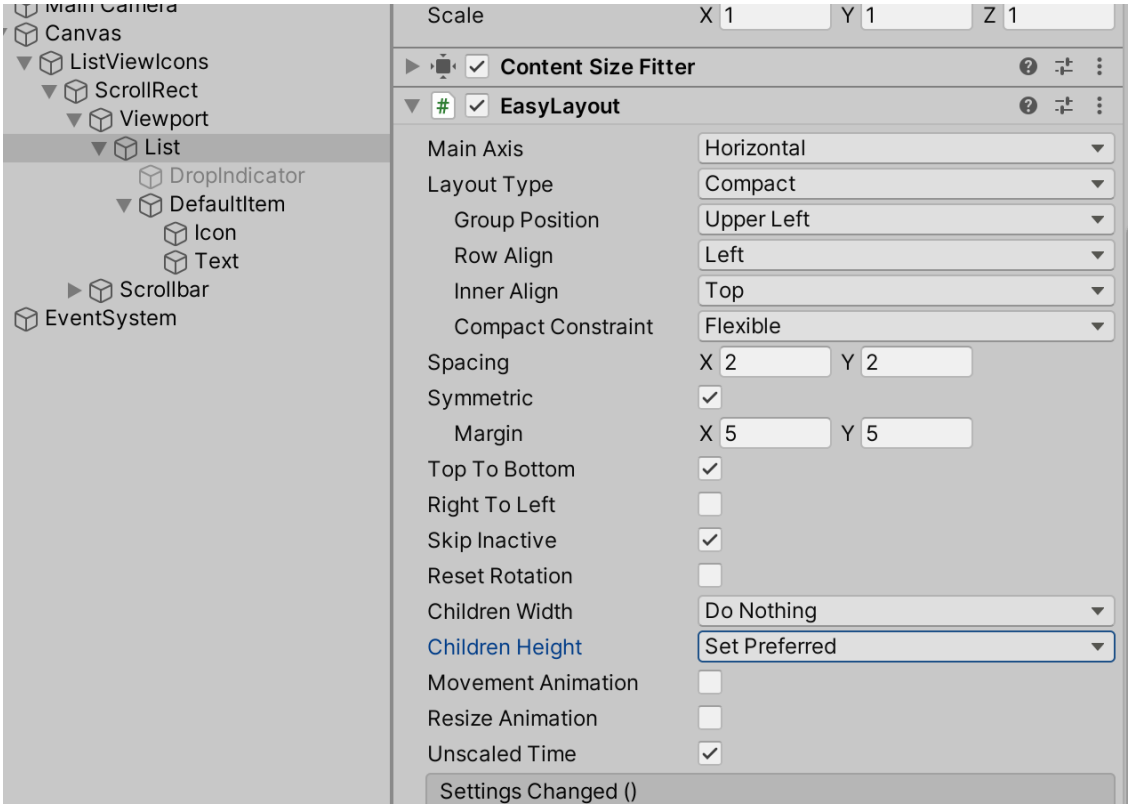
ListView and TileView can display items with different heights *or* widths (it cannot be both at the same time).

1. ***ListView.DefaultItem*: add layout group component (it can be `Horizontal Layout Group`, `Vertical Layout Group`, or `EasyLayout`)**
   - in case of `Horizontal Layout Group` or `Vertical Layout Group`: enable `Height` for `Control Child Size`, specify `Padding` and `Spacing` if needed.
   - in case of `EasyLayout`: change `Children Height` to `Set Preferred`, specify `Margin` and `Spacing` if needed.

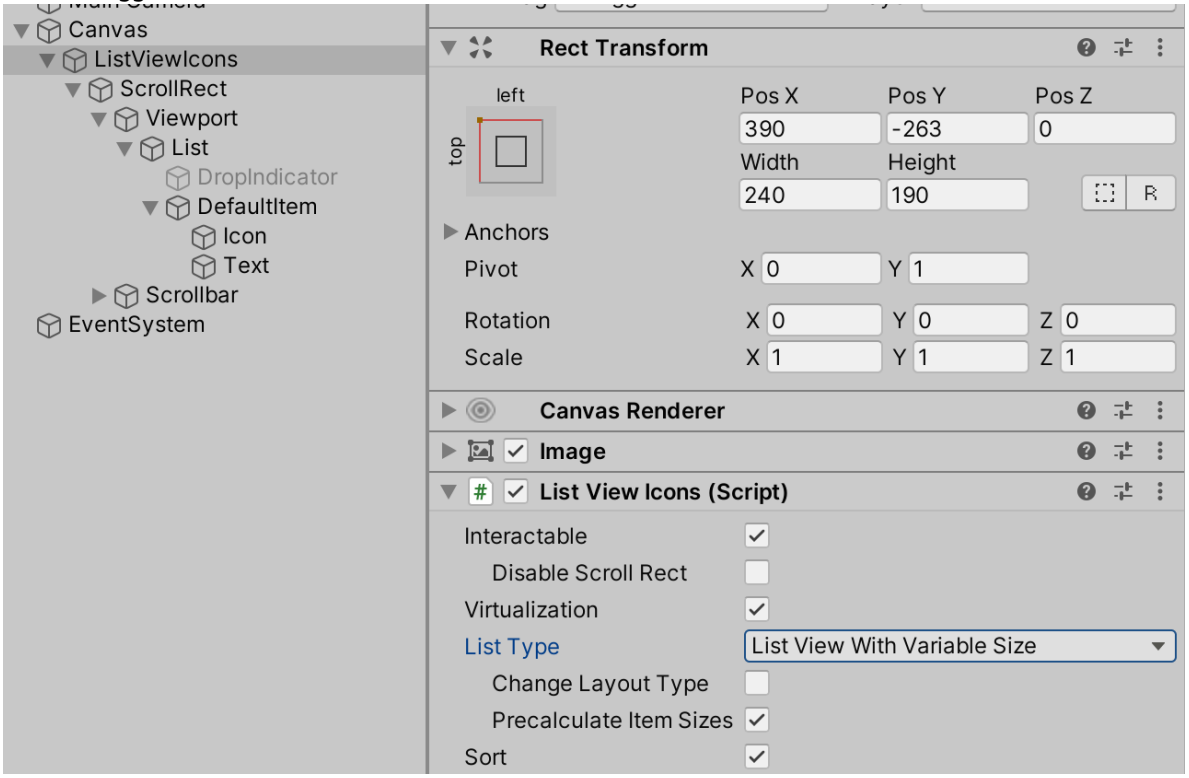
   If the ListView `Direction` is `Horizontal` then use *width* related options instead of *height*. `LayoutElement` component can be to **DefaultItem** to specify minimal width or height and other options.



2. ***ListView.Container*: change `Children Height` to `Set Preferred` in `EasyLayout` component**



3 *ListView*: change `List Type` to `List View With Variable Size`, `Tile View With Variable Size`, or `Tile View Staggered`



Layout group will resize nested game objects and determine the size of each item. `EasyLayout` will resize those items

and `ListView` will correctly process items with different sizes.

Multiple DefaultItems

You can create a component with the `IListViewTemplateSelector<TComponent, TItem>` interface implementation and add this component to the widget.

Note:

ListView also has the `TemplateSelector` property, which can be used to specify template selectors.
But it does not support all functionality (like changing colors and UI Themes support) in the editor.

```
namespace UIWidgets.Examples
{
        using UnityEngine;

        public class GroupMultipleListViewSelector : MonoBehaviour,␣
↪IListViewTemplateSelector<GroupMultipleComponent, GroupMultipleItem>
        {
                [SerializeField]
                public GroupMultipleComponent GroupTemplate;

                [SerializeField]
                public GroupMultipleComponent CheckboxTemplate;

                [SerializeField]
                public GroupMultipleComponent ValueTemplate;

                public GroupMultipleComponent[] AllTemplates()
                {
                        return new[] { GroupTemplate, CheckboxTemplate, ValueTemplate };
                }

                public GroupMultipleComponent Select(int index, GroupMultipleItem item)
                {
                        switch (item.Mode)
                        {
                                case GroupMultipleItem.ItemMode.Group:
                                        return GroupTemplate;
                                case GroupMultipleItem.ItemMode.Checkbox:
                                        return CheckboxTemplate;
                                case GroupMultipleItem.ItemMode.Value:
                                        return ValueTemplate;
                        }

                        throw new ArgumentOutOfRangeException(nameof(item), item.Mode,
↪"Unsupported Item Mode");
                }
        }
}
```

IListViewTemplateSelector Interface

Methods:

- `TComponent[]` AllTemplates()

  Get all possible templates.

- `TComponent` Select(`int` index, `TItem` item);

  Returns template to use for specified item with index.

DefaultItem Instances

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

Add Item

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

Get Items

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

Set Items

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

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

Display Same List with ListView, TileView or Table

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

Get Last Selected Index

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

Get Selected Indices

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

Last Selected Item

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

Get Selected Items

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

Delete Specified Item

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

Delete Item by Index

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

Clear List

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

Add Items

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

Optimization

```
// Use BeginUpdate() and EndUpdate() to keep widget from updating on each change.
// All changes after BeginUpdate() call will be displayed with EndUpdate() call
// or after value returned by BeginUpdate() was disposed.

void AddItems()
{
        var items = listView.DataSource;
        using var _ = 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 at function end
}

void AddItemsV2()
{
        var items = listView.DataSource;
        using (var _ = 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 at end of the 'using' block

        // .., other code
}
```

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 or after value returned by BeginUpdate() was disposed.

Disable Permanent Sort

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

Set Selected Index

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

Or:

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

Behavior is different if you enable `MultipleSelect`:

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

Deselect

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

Or:

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

Adding Callbacks to Custom Events of the Components

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

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

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

Scroll to Item

```
listView.ScrollToAnimated(index);
var index = ListView.DataSource.Count - 1;
// install scroll to item with a specified index
ListView.ScrollTo(index);
// animated scroll (uses ScrollMovement curve)
ListView.ScrollToAnimated(index);
// animated scroll with custom animation curve
var curve = AnimationCurve.EaseInOut(timeStart: 0f, valueStart: 0f, timeEnd: 1f,␣
↪valueEnd: 1f);
ListView.ScrollToAnimated(index, curve, unscaledTime: true);

var pos = ListView.GetItemPosition(index);
// instant scroll to a fixed position
ListView.ScrollToPosition(pos);
// animated scroll (uses ScrollMovement curve)
ListView.ScrollToPositionAnimated(pos);
var curve = AnimationCurve.EaseInOut(timeStart: 0f, valueStart: 0f, timeEnd: 1f,␣
↪valueEnd: 1f);
ListView.ScrollToPositionAnimated(index, curve, unscaledTime: true);
```

Disable Items

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

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

Stop Animations

```
protected virtual void Start()
{
        ListView.ItemsEvents.MovedToCache.AddListener(StopAnimations);
}

void StopAnimations(int index, ListViewItem instance)
{
        instance.StopSelectableAnimations();
        instance.Animator.ResetTrigger("customState");
}
```

Prevent Instance Recycling

You can prevent instance recycling if some action is running (like drag&drop) and the instance should be available until it ends.

```
protected override void InitDrag(PointerEventData eventData)
{
        Instance.DisableRecycling = true;
        // ....
}

public override void Dropped(bool success)
{
        Instance.DisableRecycling = false;
        // ...
}
```

Filter

A ObservableListFilter<T> is available to filter the ObservableList<T>. It accepts the input list and predicate; and provides an output list with items that match the predicate.

```
using UIWidgets;
using UnityEngine;

public class TestFilter : MonoBehaviour
{
        public ListViewIcons ListView;

        public InputFieldAdapter InputField;

        ObservableListFilter<ListViewIconsItemDescription> Filter;

        void Start()
        {
                Filter = new ObservableListFilter<ListViewIconsItemDescription>(ListView.
↪DataSource, Predicate);
                ListView.DataSource = Filter.Output;
                InputField.onValueChanged.AddListener(InputFieldChanged);
        }

        void OnDestroy()
        {
                if (InputField != null)
                {
                        InputField.onValueChanged.RemoveListener(InputFieldChanged);
                }
        }

        void InputFieldChanged(string ignore) => Filter.Refresh();

        bool Predicate(ListViewIconsItemDescription item)
        {
                var name = item.LocalizedName ?? item.Name;
                return UtilitiesCompare.Contains(name, InputField.Value, false);
        }

        public void Add()
        {
                var name = string.Format("Item {0}", Filter.Input.Count.ToString());
                Filter.Input.Add(new ListViewIconsItemDescription() { Name = name });
        }
}
```

Working With Instances

- if you need to change some property, like color, then you need to change it for all instances using the `GetComponentsEnumerator()` method:

> **Note:** ListView controls colors of objects specified in the `Foregrounds`, `Backgrounds`, and `CellsBackgrounds` properties of *ListViewComponent{DataType}.cs* (`TComponent` class) if `Allow Coloring` options is enabled. You may need to remove manually controlled objects from those properties to avoid conflicts.

> **Note:** If you need complex control over item colors in different states (default, highlighted, selected), then you can override `StateDefault()`, `StateHighlighted()`, `StateSelected()`, and `StateDisabled()` methods of *ListViewComponent{DataType}.cs* (`TComponent` class).

```
foreach (var instance in ListView.
↪GetComponentsEnumerator(PoolEnumeratorMode.All))
{
        // if object referenced in the item component class
        instance.TextAdapter.color = Color.red;
        // or find the object by name
        instance.RectTransform.Find("ObjectName").GetComponent<Graphic>().
↪color = Color.red;
}
```

- If you need to change some property of the specific item depending on some value, then you need to modify *ListViewComponent{DataType}.cs* (`TComponent` class):

```
protected virtual void UpdateView()
{
        // ...
        ValueText.color = ValueColor(Item.Value);
}

Color ValueColor(int value)
{
        if (value > 0)
        {
                return Color.green;
        }
        else if (value < 0)
        {
                return Color.red;
        }

        return Color.black;
}
```

- if you need to change some property of the specific item in case of incorrect input, then you need to modify *ListViewComponent{DataType}.cs* (`TComponent` class):

```
InputField InputField;

protected override void Start()
{
        base.Start();

        InputField.onEndEdit.AddListener(ValueChanged);
}

protected override void OnDestroy()
{
        base.OnDestroy();

        if (InputField != null)
        {
                InputField.onEndEdit.RemoveListener(ValueChanged);
        }
}

void ValueChanged(string value)
{
        // check if value is valid
        var valid = !string.IsNullOrEmpty(value);
        if (valid)
        {
                Item.Name = value;
                InputField.textComponent.color = Color.green;
        }
        else
        {
                InputField.textComponent.color = Color.red;
        }
}
```

- if you need to externally control some property of the specific item, like "set color for each item separately," then you need to add a new property to the your *{DataType}.cs* and modify *ListViewComponent{DataType}.cs* (`TComponent` class):

  Your *{DataType}.cs* (`TItem` class):

```
[Serializable]
public class DataType : IObservable
{
        // ... other fields and properties

        [SerializeField]
        Color color = Color.green;

        public Color Color
        {
                get => color;

                set => Change(ref color, value, nameof(Color));
        }
```

ListViewComponent{DataType}.cs:

```
protected virtual void UpdateView()
{
        // ...
        Text.color = Item.Color;
}
```

4.1.9 ListViewEnum

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

ListViewEnum<TEnum> Constructor Arguments

- listView `ListViewEnum`

  ListView to display `enum` values.

- showObsolete `bool` = `false`

  Show obsolete values.

- long2enum `Func<long, TEnum>` (optional)

  Custom converter from `long` to `TEnum`, use it to avoid memory allocations by default converter.

- enum2long `Func<TEnum, long>` (optional)

  Custom converter from `TEnum` to `long`, use it to avoid memory allocations by default converter.

ListViewEnum<TEnum> Properties

- Selected `TEnum`

  Selected value.

Example

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

   ListViewEnum<AdditionalCanvasShaderChannels> Wrapper;

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

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

(continues on next page)

(continued from previous page)

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

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

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

4.1.10 TracksView

Can be used for the schedule-like or timeline-like widgets.

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

Options

- Tracks `ObservableList<Track<TData, TPoint>>`

  List of tracks with items.

- TrackDataView `ScrollRect`

  Used to display tracks items.

- TrackNamesView `ScrollRect`

  Used to display the names of tracks.

- PointNamesView `ScrollBlockBase`

  Used to display the name of points, for example, time or date.

- Items Spacing `float`

  Empty space between items on Y axis.

- Tracks Spacing `float`

Empty space between tracks on Y axis.

- Allow Drag Outside `bool`

  Allow to drag items outside of the TrackDataView.

- Items to Top `bool`

  Push items to the top if there is empty space.

- Compact `bool`

  Compact items position.

- Allow Intersection `bool`

  Allow temporary intersection during drag; overlapped item will be moved to another line after drag.

- Allow Auto Scroll `bool`

  Allow auto-scroll if the cursor is near the border on less the specified distance.

- Auto Scroll Border Distance `float`

  Distance to the border where auto-scroll start working.

- Auto Scroll Speed `float`

  Speed of the auto-scroll.

- Default Item `TDataView`

  A prefab used to display item.

- Default Track Header `TTrackView`

  A prefab used to display track header.

- Default Track Background `TTrackBackground`

  A prefab used to display track background.

- Track Data Dialog `TTrackDataDialog`

  Dialog to add/edit item.

- Track Dialog `TTrackDialog`

  Dialog to add/edit track.

4.1.11 TreeGraph

Options

- Nodes `ObservableList<TreeNode<TItem>>`

  Not available in the inspector window.

- DefaultItem `TComponent`

  A prefab used to display item.

- Direction `TreeGraphDirections`

  Directions: `TopToBottom`, `BottomToTop`, `LeftToRight`, `RightToLeft`.

- Container `RectTransform`

The container of the instantiated gameobjects used to display items.

- Spacing `Vector2`

  Minimal space between items.

- Line Type `ConnectorType`

  Line type: `Straight` or `Rectangular`.

- Line Thickness `float`

  Line thickness.

- Line Margin `float`

  The minimum space from the border before the turn of the line. Supported only by `Rectangular` lines.

4.1.12 TreeView

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

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

Options

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

- Nodes `ObservableList<TreeNode<TItem>>`

  Not available in the inspector window.

- Deselect Collapsed Nodes `bool`

  Deselect nested nodes when parent node collapsed.

- Scroll With Indent `bool`

  Scrolling with node indent in the secondary direction.

- Container Max Size `bool`

  Prevent scrollbar blink caused by virtualization: the container will have the maximum width of all items. By default, the container has the maximum width of only visible items.
  Require `List Type` = `List View with Variable Size`.

Get nodes

```
public TreeView Tree;

ObservableList<TreeNode<TreeViewItem>> nodes;

void Start()
{
   if (Tree.Nodes == null)
   {
      Tree.Nodes = new ObservableList<TreeNode<TreeViewItem>>();
   }

   nodes = Tree.Nodes;
}
```

Get selected nodes

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

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

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

Add listeners

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

   Tree.NodeDeselected.AddListener(ProcessDeselectedNode);
}

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

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

Select node

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

Select node with subnodes

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

Deselect node

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

Deselect node with subnodes

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

Scroll to node

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

Add node

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

Hide nodes

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

Collapse node

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

Expand node

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

Change node name

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

Sort

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

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

public void SortAsc()
{
   using var _ = nodes.BeginUpdate();
   ApplyNodesSort(nodes, comparisonAsc);
}

public void SortDesc()
{
   using var _ = nodes.BeginUpdate();
   ApplyNodesSort(nodes, comparisonDesc);
}

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

Filter nodes

```
public void Filter(string nameContains)
{
   // Maintains performance while items are added/removed/changed
   // by preventing the widgets from drawing
   // until the EndUpdate() method is called
   // or after value returned by BeginUpdate() was disposed.
   using var _ = nodes.BeginUpdate();

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

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()
{
   using var _ = nodes.BeginUpdate();
   nodes.ForEach(SetVisible);
}

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

   node.IsVisible = true;
}
```

Clear nodes

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

Nodes Serialization

You can use helper class `TreeNodeJson<TItem>` for the node serialization and deserialization.

Warning: Unity `JsonUtility` does not support recursive types so it cannot be used. `Newtonsoft.Json` can be used instead.

```
// serialize
var nodes = TreeNodeJson<TreeViewItem>.ConvertNodes(TreeView.Nodes);
var json = JsonConvert.SerializeObject(nodes);

// deserialize
var decoded = JsonConvert.DeserializeObject<TreeNodeJson<TreeViewItem>[]>(json);
TreeView.Nodes = TreeNodeJson<TreeViewItem>.ConvertNodes(decoded);
```

4.2 Containers

4.2.1 Accordion

Options

- Interactable `bool`

  Allow users interact with the Accordion.

- Items (DataSource) `ObservableList<AccordionItem>`

  Items.

  `AccordionItem` fields:

  - ToggleObject `GameObject` Click on this object open or close *ContentObject*.
  - ContentObject `GameObject`
  - Open `bool` Default state of the *ContentObject*.

- Only One Open `bool`

  Only one item can be open at the same time.

- All Items Can Be Closed `bool`

  Allow to close all items; otherwise at least one item always will be opened.

- Animate `bool`

  Animate open and close.

- Animation Duration `float`

  Animation Duration.

- Unscaled Time `bool`

  Run animation with unscaled time.

- Direction `AccordionDirection`
  - `Horizontal`
  - `Vertical`
- Resize Method `ResizeMethods`
  - `Size` - change width or height of the ContentObject.
  - `Flexible` - change LayoutElement flexibleWidth or flexibleHeight of the ContentObject.
- Disable Closed `bool`

  Disable closed ContentObjects.

Events

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

AccordionHighlight

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

Open item

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

Close item

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

Toggle item

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

Set items

```
Accordion.DataSource = new ObservableList<AccordionItem>()
{
   new AccordionItem()
   {
      ToggleObject = Header1,
      ContentObject = Content1,
      Open = true,
   },
   new AccordionItem()
   {
      ToggleObject = Header2,
      ContentObject = Content2,
      Open = false,
   },
   new AccordionItem()
   {
      ToggleObject = Header3,
      ContentObject = Content3,
      Open = false,
   },
};
```

4.2.2 Slider Horizontal / Slider Vertical

It is just a `ScrollRect` with *Paginator*.
It can be used for tutorials on mobile devices and can easily fill a screen size.

4.2.3 StackView

It is a container that can have multiple views (items). Current item is displayed at top.

Note: See also *StackViewAnimations*.

Methods

- *void Push(RectTransform newView)*

  Added new view to the top of the stack.
  If newView is already in stack it will be moved to top.
  If newView is current (on top) then nothing will happen.

- *void Replace(RectTransform newView)*

  Replace the current view with a specified one.
  If newView is already in stack it will be moved to top.
  If newView is current (on top) then nothing will happen.

- *void Pop(int total = 1)*

Remove *total* views from top of the stack. The view below the *total* view becomes the new current view.

- *void Clear()*

  Remove all views from stack.

- *void FirstOnly()*

  Remove all views except first from stack.

- *bool Remove(RectTransform view, bool raiseEvents = true)*

  Remove view without any animations or event.
  Return *true* if view was removed; otherwise *false*.

- *bool Contains(RectTransform view)*

  Is StackView contains a specified view?
  Return *true* if StackView contains a specified view; otherwise *false*.

Properties (Read-Only)

- Current *RectTransform*

  Current view.

- Count *int*

  Count of views.

- RunningAnimation *bool*

  true if animation is running; otherwise *false*

Events

- OnBeforeCurrentChanged *UnityEvent<RectTransform>*
- OnAfterCurrentChanged *UnityEvent<RectTransform>*

4.2.4 Tabs

Options

- Container `Transform`

  Container for the tabs buttons.

- DefaultTabButton `Button`

  Button template for the inactive tabs.

- ActiveTabButton `Button`

  Button for the active tab.

- TabObjects `Tab[]`

  Tabs array, contains names and references to the tabs gameobjects.

  `Tab` fields:

  - Name `string`
  - TabObject `GameObject`

- DefaultTabName `string`

  Name of the tab opened by default.

- KeepTabsActive `bool`

  If true does not deactivate hidden tabs.

- ImmediateSelect `bool`

  Open the tab immediately if the tab header is under focus (gameobject selected by EventSystem), useful in a keyboard or gamepad navigation.

- Animation `bool`

  Animate tabs change.

- AnimationDuration `float`

  Animation duration.

- UnscaledTime `bool`

  Run animation with unscaled time.

- CustomAnimation `Func<AnimationData, IEnumerator>`

  Custom animation to use instead of default one.

- CanSelectTab `Func<Tab, bool>`

  Function to check is tab can be selected.

Events

- OnTabSelect `UnityEvent<int>`

  Receive index of the selected tab.

TabsScroll

The component is attached to the Header container. Used to toggle tabs with mouse scroll.

Select tab

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

Enable tab

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

Disable tab

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

4.2.5 TabsSlider

Combination of the `ScrollView` and *Paginator*.

4.3 Controls

4.3.1 Context Menu

To use the menu, you need to add a `ContextMenu` component and `ContextMenu` template. Different menus can use the same template.
Menu items are edited in a separate window which can be opened from the `ContextMenu` component.
In this window, you can specify menu items: name, icon, checkmark, item template, hotkey, and action when the item is clicked.
Initially, two item templates are available: the default template and the delimiter template; a minus sign is used as the key of the delimiter.
Any string can be used as a template key, not just signs.

The keyboard is supported: you can open the menu and navigate between menu items.
Hotkeys work out of the box with both legacy input and a new input system.

Options

- Interactable `bool`

  Allow users interact with the ListView.

- Template `ContextMenuTemplate`

  Context menu template.

- MenuItems `ObservableList<MenuItem>`

  Menu items.

- Is Default `bool`

  Is default menu? Default menu will be opened on context menu key press.

- Navigation `bool`

  Enable keyboard and gamepad navigation.

- Open On Right Button Click `bool`

Open context menu on right mouse button click.

- Open On Context Menu Key `bool`

  Open context menu on context menu key press.

- Submenu Delay `float`

  Delay before open and close sub menu.

- Unscaled Time `bool`

  Use unscaled time.

MenuItem Options

- Visible `bool`

  Is item visible?

- Interactable `bool`

  Is item interactable?

- Icon `Sprite`

  Icon.

- Checked `bool`

  Is item checked?

- Name `string`

  Name.

- HotKey `HotKey`

  HotKey can be enabled with `MenuItem.EnableHotKey()` even if item not used in menu (Supported only of InputSystem enabled).

- Action `UnityEvent<MenuItem>`

  Action on item click.

- Items `ObservableList<MenuItem>`

  Nested items.

Events

- OnOpen `UnityEvent<ContextMenu>`

  The event raised when context menu opened.

  Arguments: opened context menu.

- OnClose `UnityEvent<ContextMenu>`

  The event raised when context menu closed.

  Arguments: closed context menu.

- OnItemSelect `UnityEvent<MenuItem>`

The event raised when menu item selected.

Arguments: selected menu item.

- OnItemDeselect `UnityEvent<ContextMenu>`

  The event raised when menu item deselected.

  Arguments: selected menu item.

ContextMenu for non-UI Gameobjects

You can add the `OpenContextMenu` component with the `ContextMenu` reference to a non-UI game object and the menu will be opened on the right mouse button click.
Or you can open the menu with the script:

```
contextMenu.Open(eventData);
```

4.3.2 Paginator

Important: ScrollRect.Content anchors should at the top left corner.

How to select paginator

- If you need paginator for the TextMeshPro text with `Overflow` = `Page` use `TextMeshProPaginator`
- 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.

Common Options

- Interactable `bool`

  Allow users interact with the Paginator.

- View `PaginatorView`
  - Default Page `ScrollRectPage` *optional*

    Template GameObject to display inactive pages.

  - Active Page `ScrollRectPage` *optional*

    Template GameObject to display active page.

  - Prev Page `ScrollRectPage` *optional*

    GameObject, go to the previous page.

  - Next Page `ScrollRectPage` *optional*

    GameObject, go to the next page.

  - Skip Page `RectTransform` *optional*

    Template to display skipped pages.

  - Pages Container `RectTransform` *optional*

    Container for the created `Default Page` instances.

  - HideIfOnePage `bool`

    Hide active page if has only one page.

  - Visible Pages Count `int`

    Number of visible page buttons.
    The first and last page buttons are always shown.
    Set to 0 to display buttons for all pages.

Common Events

- OnPageSelect `UnityEvent<int>`

ScrollRectPaginator Options

- ScrollRect `ScrollRect`

  ScrollRect to work with.

- Direction `PaginatorDirection`

  Scroll direction.

  - `Auto` detect direction by ScrollRect settings and ScrollRect.content size.
  - `Horizontal` scroll in the horizontal direction
  - `Vertical` scroll in the vertical direction

- Fast Drag Distance `float`

  Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.

- Fast Drag Time `float`

  Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.

- Current Page `int`

  Default page.

- Forced Position `PaginatorPagePosition`

  Automatically scroll to the nearest page after drag ended if not meet *Fast Drag* condition.

  - `None` automatical scroll disabled
  - `OnStart` automatical scroll enabled; page aligned by the left side of the ScrollRect (or the top side if scroll in the vertical direction)
  - `OnCenter` automatical scroll enabled; page aligned by the center side of the ScrollRect
  - `OnEnd` automatical scroll enabled; page aligned by the right side of the ScrollRect (or the bottom side if scroll in the vertical direction)

- Last Page Full Size `bool`

  Change the last page size to full-page size.

- Page Rounding `PaginatorRounding`
  - Floor
  - Round
  - Ceil
- Animation `bool`

  Enable animation.

- Page Size Type `PageSizeType`

  If *Page Size Type* = *Auto* page size is equal to scroll rect size, if *Page Size Type* = *Fixed* will be used *Page Size* value.
  - `Auto`
  - `Fixed`
- Page Size `float`

  Size of the page.

- Page Spacing `float`

  Space between pages.

- Movement `AnimationCurve`

  Animation curve.

- Unscaled Time `bool`

  Run animation with unscaled time.

ListViewPaginator Options

- List View `ListViewBase`
- Per Page `int`

  Items count on one page, for TileView this is rows or columns count per page.

- Fast Drag Distance `float`

  Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.

- Fast Drag Time `float`

  Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. Set zero to disable.

- Current Page `int`

  Default page.

- Forced Position `PaginatorPagePosition`

  Automatically scroll to the nearest page after drag ended if not meet *Fast Drag* condition.
  - `None` automatical scroll disabled

  - `OnStart` automatical scroll enabled; page aligned by the left side of the ScrollRect (or the top side if scroll in the vertical direction)
  - `OnCenter` automatical scroll enabled; page aligned by the center side of the ScrollRect
  - `OnEnd` automatical scroll enabled; page aligned by the right side of the ScrollRect (or the bottom side if scroll in the vertical direction)
- Last Page Full Size `bool`

  Change the last page size to full-page size.
- Page Rounding `PaginatorRounding`
  - Floor
  - Round
  - Ceil
- Animation `bool`

  Enable animation.

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

Animation

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

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

Tile View ScrollRect Fitter

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

4.3.3 Sidebar

Component to drag sidebar from behind the screen.

Options

- Interactable `bool`

  Enable or disable the ability to drag the sidebar.
- Curve `AnimationCurve`

  Animation curve for the open and close animations.
- Direction `SidebarAxis`

  Drag direction to open sidebar.
- Animation Type `SidebarAnimation`
  - `Overlay`
  - `Push`

  - Scale Down
  - Uncover
  - Slide Along
  - Slide Out
  - Resize
  - Scale Down and Push
- Scale Down Limit `float`

  Content scale cannot be lower this value for the ScaleDown animation.
- Is Open `bool`

  Is sidebar opened?
- Modal `bool`

  Is sidebar should be closed with the click outside of the sidebar?
- ScrollRect Support `bool`

  Allow to handle children ScrollRect's drag events.
- Content `RectTransform`

  Cntent GameObject. Required by some animations.
- Animate With Layout `bool`

  Change Content LayoutElement size during animation.
- Optional Handle `GameObject` *optional*

  Handle to open and close sidebar.
- Unscaled Time `bool`

  Run animations with unscaled time.
- Modal Color `Color`

  Modal background color.

Events

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

4.3.4 SplitButton

Button with the additional dropdown list of the buttons.

Options

- Primary Button `Button`

  Primary Button.

- Toggle Button `Button`

  Button to toggle the *Additional Buttons Block*.

- Additional Buttons Block `GameObject`

  Container for the additional buttons.

- Additional Buttons `List<Button>`

  List of the additional buttons.

- Modal Sprite `Sprite`

  Background sprite when additional buttons block displayed.

- Modal Color `Color`

  Background color when additional buttons block displayed.

4.4 Dialogs

Dialogs, Popups, Pickers, Notifications works with templates.
Code usually looks like this:

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

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

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

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

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

4.4.1 ColorPicker Dialog

Note: See *Windows Animations* for open and closed animations.

Options

- Color Picker `ColorPicker`

  ColorPicker.

- Close Button `Button`

  Button to close picker without selecting value.

- Hide on Modal Click `bool`

  Close picker on click on the background if the `modal` option enabled.

- Mode `PickerMode`

  Close picker when value selected or when OK button clicked.

- Ok Button `Button`

  Button to select and close picker.

Code examples

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

    public class TestColorPickerDialog : MonoBehaviour
    {
        [SerializeField]
        ColorPickerDialog InputColor;

        Color currentValue = Color.white;

        async public void Test()
        {
                            var result = await InputColor.Clone().
↪ShowAsync(currentValue);
                            if (result.Success)
                            {
                                    currentValue = result.Value;
                Debug.Log("value: " + currentValue);
                            }
            else
            {
                Debug.Log("canceled");
            }
        }
```

(continues on next page)

(continued from previous page)

```
    }
}
```

4.4.2 DatePicker, DateTimePicker, TimePicker

Nested Widgets Replacement

Nested widgets can be safely replaced with their analogs:

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

Note: See *Windows Animations* for open and closed animations.

DatePicker Options

- CloseButton `Button`

  Button to close picker without selected value.

- HideOnModalClick `bool`

  Close picker on background click outside of picker.

- Mode `PickerMode`

  Picker mode:

  - Close On Select

    Close picker right after value selected.

  - Close On OK

    Close picker on OK click.

- Date Change Only `bool`

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

- OkButton `Button`

  OK button with selected value.

- Calendar `DateBase`

  Reference to the Date widget.

DateTimePicker Options

- CloseButton `Button`

  Button to close picker without selected value.

- HideOnModalClick `bool`

  Close picker on background click outside of picker.

- DateTimeWidget `DateTimeWidget`

  Reference to the DateTime widget.

TimePicker Options

- CloseButton `Button`

  Button to close picker without selected value.

- HideOnModalClick `bool`

  Close picker on background click outside of picker.

- Time `TimeBase`

  Reference to the Time widget.

Minimal Code

```
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 async void TestAsync()
      {
         // create picker by template
```

(continues on next page)

(continued from previous page)

```
        var picker = PickerTemplate.Clone();

        // show picker
        var value = await picker.ShowAsync(currentValue);
        if (value.Success)
        {
            currentValue = value;
            Debug.Log("value: " + value);
        }
        else
        {
            Debug.Log("canceled");
        }
    }

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

4.4.3 Dialog

Note: See *Windows Animations* for open and closed animations.

Options

- Buttons Templates `ReadOnlyCollection<Button>`

  Templates for the buttons.

- Content Root `RectTransform`

  Root gameobject for the content.

- Title Text `Text` (obsolete)

  GameObject to display title. Replaced with the *DialogInfo*.

- Content Text `Text` (obsolete)

  GameObject to display text. Replaced with the *DialogInfo*.

- Icon `Image` (obsolete)

  GameObject to display icon. Replaced with the *DialogInfo*.

- Dialog Info `DialogInfoBase`

  Component to display the dialog info.

- AutoFocus `bool`

  Set focus to the last Selectable object in the Dialog.

- Close Button `Button`

  Button to close dialog.

- Buttons Container `RectTransform`

  Buttons container. If container not specified will be used parent of the button template.

- Hide on Modal Click `bool`

  Close dialog on click on the background if the `modal` option enabled.

Events

- OnAnimationStart `UnityEvent<bool>`

  The event is raised before the animation starts.

  Arguments: true if opening animation; false if closing animation.

Show() Method Parameters

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

- title `string`

  Dialog title.
  Can be changed with `SetInfo()` method.

- message `string`

  Dialog message.

Can be changed with `SetInfo()` method.

- buttons `ButtonsPool`

  Dialog buttons.
  Can be changed with `SetButtons()` method.
  `DialogButton` fields:

  - Label `string`

    Button label.

  - Action `Func<DialogBase, int, bool>`

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

  - Template Index `int`

    Index of the button template.

- focusButton `string`

  Button with focus by default.
  Can be changed with `SetButtons()` or `FocusButton()`.

- position `Vector3?`

  Dialog position.
  Can be changed with `SetPosition()`.

- icon `Sprite`

  Dialog icon.
  Can be changed with `SetInfo()` method.

- modal `bool`

  Modal dialog.
  Can be changed with `SetModal()`.

- modalSprite `Sprite`

  Background image for the modal dialog.
  Can be changed with `SetModal()`.

- modalColor `Color?`

  Background color for the modal dialog.
  Can be changed with `SetModal()`.

- canvas `Canvas`

  Canvas to display dialog. Required if dialog template is prefab.
  Can be changed with `SetCanvas()`.

- content `RectTransform`

  Dialog content. Can be used instead of the *message* and *icon*.
  Can be changed with `SetContent()`.

- onClose `Action`

  Action to run when dialog closed.
  Can be changed with `OnClose` field.

- onCancel `Func<int, bool>`

  Function to run when dialog canceled. Receive dialog instance and -1 as button index, return `true` if dialog should be closed.
  Obsolete, use `Func<DialogBase, int, bool> OnDialogCancel` field instead.

ShowAsync() Method Parameters

All parameters are optional.
`title` and `message` also can be specified with `SetInfo()` to use formatted strings.
Returns index of the clicked button or `-1` in case of `Cancel()` method.

- title `string`

  Dialog title.
  Can be changed with `SetInfo()` method.

- message `string`

  Dialog message.
  Can be changed with `SetInfo()` method.

- buttons `ButtonsPool`

  Dialog buttons.
  Can be changed with `SetButtons()` method.
  `DialogButton` fields:

  - Label `string`

    Button label.

  - Action `Func<DialogBase, int, bool>`

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

  - Template Index `int`

    Index of the button template.

- focusButton `string`

  Button with focus by default.
  Can be changed with `SetButtons()` or `FocusButton()`.

- position `Vector3?`

  Dialog position.
  Can be changed with `SetPosition()`.

- icon `Sprite`

  Dialog icon.
  Can be changed with `SetInfo()` method.

- modal `bool`

  Modal dialog.

Can be changed with `SetModal()`.

- modalSprite `Sprite`

  Background image for the modal dialog.
  Can be changed with `SetModal()`.

- modalColor `Color?`

  Background color for the modal dialog.
  Can be changed with `SetModal()`.

- canvas `Canvas`

  Canvas to display dialog. Required if dialog template is prefab.
  Can be changed with `SetCanvas()`.

- content `RectTransform`

  Dialog content. Can be used instead of the *message* and *icon*.
  Can be changed with `SetContent()`.

- closeOnButtonClick `bool`

  Close dialog on button click.

Minimal code

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

Advanced

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

(continues on next page)

(continued from previous page)

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

Async

```
// create dialog instance
var dialog = dialogPrefab.Clone();
// show dialog with following parameters
var button_index = await dialog.ShowAsync(
   title: "Modal Dialog",
   message: "Simple Modal Dialog.",
   buttons: new DialogButton[]
   {
      "Do Some Action",
      "Do Other Action",
      "Close",
   },
   focusButton: "Close",
   modal: true,
   modalColor: new Color(0, 0, 0, 0.8f)
);

if (button_index == 0)
{
   Debug.Log("Do Some Action");
}
else if (button_index == 1)
{
   Debug.Log("Do Other Action");
}
```

Adding new behaviour

1. Create helper component

```
using UnityEngine;
using UnityEngine.UI;

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

   [SerializeField]
   public InputField Password;

   // Reset values
   public void Refresh()
   {
```

(continues on next page)

(continued from previous page)

```
        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 DialogButton[]
        {
            // on click call SignInNotify
            new DialogButton("Sign in", SignInNotify),

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

bool SignInNotify(DialogBase dialog, int index)
{
    var helper = dialog.GetComponent<DialogInputHelper>();
    if (!helper.Validate())
    {
        return false;
```

(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;
}
```

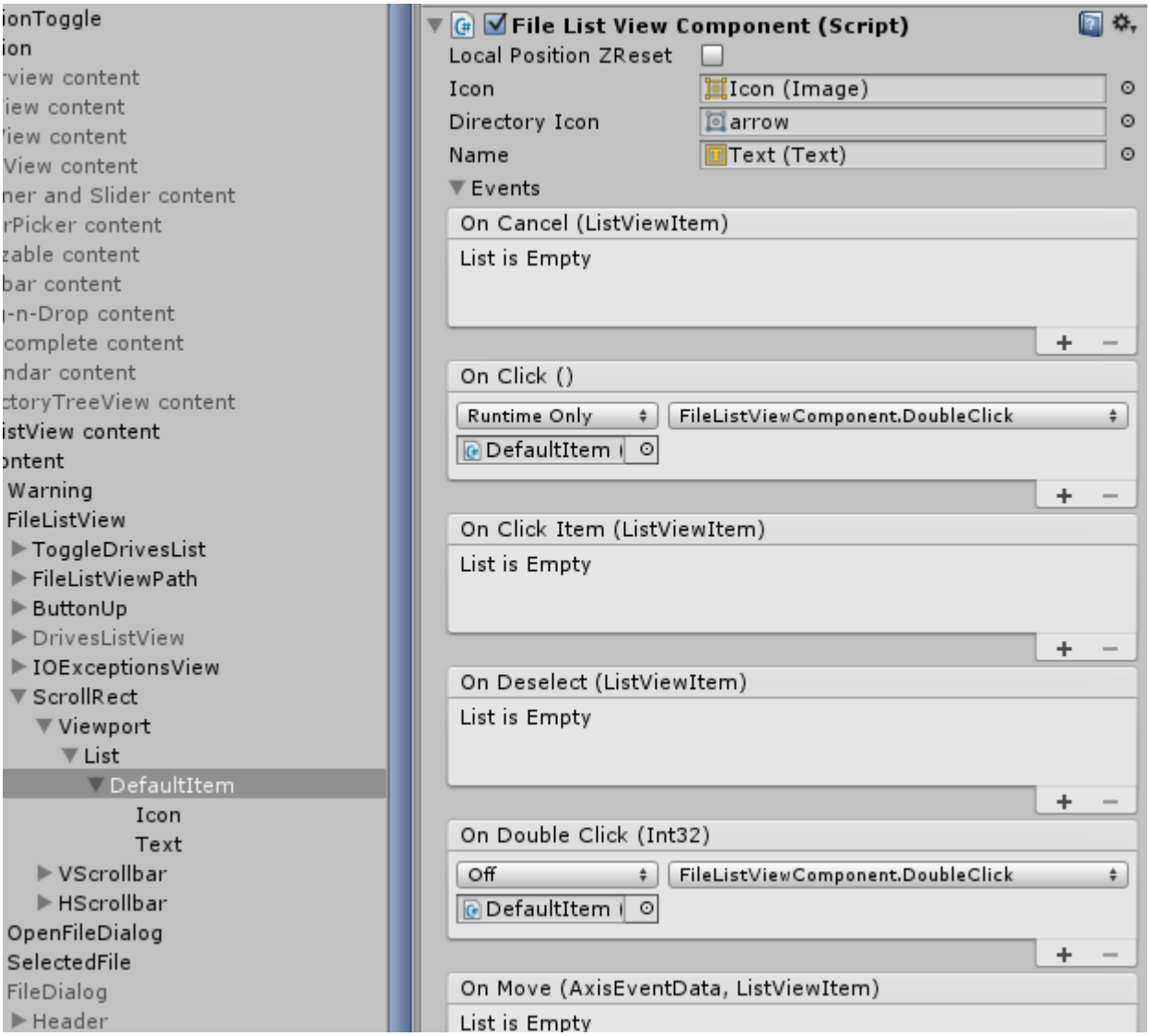
Custom Dialogs

You can create derived class with own methods and fields.

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

4.4.4 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.



Note: See *Windows Animations* for open and closed animations.

Options

- File List View `FileListView`

  FileListView.

- Confirm Dialog `PickerBool`

  Dialog to get confirmation if *Request Confirmation If File Exists* enabled.

- FilenameInput `InputField`

  Input for the filename.

- OkButton `Button`

  Button to close dialog.

- FileShouldExists `bool`

  Selected file should exists.

- Request Confirmation If File Exists `bool`

  Show *Confirm Dialog* if file exists.

Code examples

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

    /// <summary>
    /// Test FileDialog.
    /// </summary>
            public class TestColorPickerDialog : MonoBehaviour
            {
                    [SerializeField]
                    ColorPickerDialog Dialog;

                    [SerializeField]
                    Image Image;

                    async public void Test()
                    {
                            var result = await Dialog.Clone().ShowAsync(Image.color);
                            if (result.Success)
                            {
                                    Image.color = result.Value;
                                    Debug.Log("value: " + result.Value);
                            }
                            else
                            {
                                    Debug.Log("canceled");
                            }
                    }
            }
}
```

4.4.5 FolderDialog

Note: See *Windows Animations* for open and closed animations.

Options

- Directory Tree View `DirectoryTreeView`

  DirectoryTreeView widget.

- Ok Button `Button`

  Button to close dialog.

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

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

              [SerializeField]
              Text Result;

              string currentValue = string.Empty;

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

                      // show picker
                      var value = await picker.Show(currentValue);
          if (value.Success)
          {
             currentValue = value;
             Debug.Log("value: " + value);
          }
          else
          {
             Debug.Log("canceled");
          }
```

(continues on next page)

(continued from previous page)

```
        }

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

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

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

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

4.4.6 Notifications

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

Note: See *Notifications Animations* for open and closed animations.

Options

- Hide Button `Button`

  Button to close notification.

- Animate On Hide Button `bool`

  Run hide animation on HideButton click.

- Text `Text` (obsolete)

  GameObject to display the notification text. Replaced with NotifyInfo.

- Animate On Hide Button `bool`

  If enabled the hide animation will run on `Hide Button` click.

- Hide Delay `float`

  Delay before notification automatically hidden.

- Unscaled Time `bool`

  Delay with unscaled time.

- Slide Up On Hide `bool`

  Start slide up animations after hide current notification. Turn it off if its managed with HideAnimation.

- Notify Info `NotifyInfoBase`

  Component to display the notification message.

- Close Button `Button`

  Button to close notification.

- Buttons Container `RectTransform`

  Buttons container. If container not specified will be used parent of the button template.

Show() Method Parameters

All parameters are optional.
`message` also can be specified with `SetMessage()`
to use formatted strings.

- message `string`

  Notification message.
  Can be changed with `SetMessage()` method.

- customHideDelay `float?`

  Time before notification hidden or `hideAnimation` start running.
  Can be changed with `HideDelay` field.

- container `Transform?`

  Notifications container. Should have `Layout Group` component
  to display multiple notifications.
  Can be changed with `SetContainer()` method.

- showAnimation `Func<TNotification, IEnumerator>`

  Show animation. Can be changed with `ShowAnimation` field.

  > **Warning:** Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- hideAnimation `Func<TNotification, IEnumerator>`

  Hide animation. Can be changed with `HideAnimation` field.

  > **Warning:** Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- slideUpOnHide `bool?`

  Obsolete.
  Use `EasyLayout.MovementAnimation` instead.

- sequenceType `NotifySequence`

  Add notification to sequence and display in order according to the specified `sequenceType`.

- sequenceDelay `float`

  Time between previous notification was hidden and this will be displayed.
  Can be changed with `SequenceDelay` field.

- clearSequence `bool`

  Clear notifications sequence.

- newUnscaledTime `bool?`

  Animations will use unscaled time.

- content `RectTransform`

  Notification content.
  Can be changed with `SetContent()`.

- onReturn `Action`

  Action called when instance return to the cache.
  Can be changed with `OnReturn` field.

- onHide `Action<TNotification>`

  Action called when instance return to the cache.
  Can be changed with `OnNotificationHide` field.

ShowAsync() Method Parameters

All parameters are optional.
`message` also can be specified with `SetMessage()` to use formatted strings.
Returns index of the clicked button or `-1` if notification was hidden after delay or on hide button click.

- message `string`

  Notification message.
  Can be changed with `SetMessage()` method.

- customHideDelay `float?`

  Time before notification hidden or `hideAnimation` start running.
  Can be changed with `HideDelay` field.

- container `Transform?`

  Notifications container. Should have `Layout Group` component to display multiple notifications.
  Can be changed with `SetContainer()` method.

- showAnimation `Func<TNotification, IEnumerator>`

  Show animation. Can be changed with `ShowAnimation` field.

> **Warning:** Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- hideAnimation `Func<TNotification, IEnumerator>`

  Hide animation. Can be changed with `HideAnimation` field.

  > **Warning:** Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- slideUpOnHide `bool?`

  Obsolete. Use `EasyLayout.MovementAnimation` instead.

- sequenceType `NotifySequence`

  Add notification to sequence and display in order according to the specified `sequenceType`.

- sequenceDelay `float`

  Time between previous notification was hidden and this will be displayed.
  Can be changed with `SequenceDelay` field.

- clearSequence `bool`

  Clear notifications sequence.

- newUnscaledTime `bool?`

  Animations will use unscaled time.

- content `RectTransform`

  Notification content.
  Can be changed with `SetContent()`.

- closeOnButtonClick `bool`

  Close notification on button click.

Events

- OnAnimationStart `UnityEvent<bool>`

  The event is raised before the animation starts.

  Arguments: true if opening animation; false if closing animation.

Minimal code

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

Advanced

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

Notification with Buttons

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

```
[SerializeField]
protected Notify NotificationTemplate;

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

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

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

(continues on next page)

(continued from previous page)

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

Async Notification with Buttons

```
[SerializeField]
protected Notify NotificationTemplate;

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

        var instance = NotificationTemplate.Clone();
        instance.SetButtons(actions);
        var button_index = await instance.ShowAsync("Notification with buttons. Hide␣
↪after 5 seconds.",
                customHideDelay: 5f, closeOnButtonClick: false);

        while (button_index == 1)
        {
                Debug.Log("click notification button");
                button_index = await instance;
        }

        if (button_index == 0)
        {
                Debug.Log("close notification");
                instance.Hide();
        }
        else
        {
                Debug.Log("hide button");
                instance.Hide();
        }
}
```

Default Hide Animations

Warning: Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

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

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

Default Show Animations

Warning: Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

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

- **ShowAnimationRotateHorizontal**
  Rotate notification on X axis.
- **ShowAnimationRotateVertical**
  Rotate notification on Y axis.
- **ShowAnimationCollapseHorizontal**
  Resize width of the notification.
- **ShowAnimationCollapseVertical**
  Resize height of the notification.
- **ShowAnimationSlideRight**
  Slide notification from right.

- **ShowAnimationSlideLeft**
  Slide notification from left.
- **ShowAnimationSlideUp**
  Slide notification from top.
- **ShowAnimationSlideDown**
  Slide notification from bottom.

Configurable Hide Animations

Warning: Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- **HideAnimationRotateBase**

  Arguments:

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

- **HideAnimationCollapseBase**

  Arguments:

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

- **HideAnimationSlideBase**

  Arguments:

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

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

Configurable Show Animations

Warning: Obsolete. It is recommended to use *Notifications Animations* for open and closed animations.

- **ShowAnimationRotateBase**

  Arguments:

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

- **ShowAnimationCollapseBase**

  Arguments:

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

- **ShowAnimationSlideBase**

  Arguments:

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

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

Custom Notifications

You can create derived class with own methods.

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

4.4.7 Picker

Base class for the custom pickers.

Note: See *Windows Animations* for open and closed animations.

Options

- AutoFocus `bool`

  Set focus to the last Selectable object in the Picker.

- Close Button `Button`

  Button to close picker.

- Hide on Modal Click `bool`

  Close picker on click on the background if the `modal` option enabled.

Events

- OnAnimationStart `UnityEvent<bool>`

  The event is raised before the animation starts.

  Arguments: true if opening animation; false if closing animation.

Show() Method Parameters

All parameters are optional.

- defaultValue `TValue`

  Default value.</param>

- onSelect `Action<TValue>`

  Callback with selected value.

- onCancel `Action`

  Callback when picker closed without any value selected.

- modalSprite `Sprite`

  Background image for the modal dialog.
  Can be changed with `SetModal()`.

- modalColor `Color?`

  Background color for the modal dialog.
  Can be changed with `SetModal()`.

- canvas `Canvas`

  Canvas. Can be changed with `SetCanvas()`.

ShowAsync() Method Parameters

All parameters are optional.
Returns `TPicker.Result` with selected value or success mark.

- defaultValue `TValue`

  Default value.</param>

- modalSprite `Sprite`

  Background image for the modal dialog.
  Can be changed with `SetModal()`.

- modalColor `Color?`

  Background color for the modal dialog.
  Can be changed with `SetModal()`.

- canvas `Canvas`

  Canvas. Can be changed with `SetCanvas()`.

TPicker.Result Fields

- Value `TValue`

  Selected value or a default value if nothing is selected.

- Success `bool`

  `true` if the value was selected; `false` if the picker was canceled or closed without a value chosen.

Example

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

	public class PickerIntTest : MonoBehaviour
	{
		[SerializeField]
		PickerInt PickerTemplate;

		int currentValue = 0;

		async public void TestAsync()
		{
			// create picker instance
			var picker = PickerTemplate.Clone();

			// copy values
			picker.ListView.DataSource = PickerTemplate.ListView.DataSource;

			// show picker
			var value = await picker.ShowAsync(currentValue);
			if (value.Success)
			{
				currentValue = value;
				Debug.Log("value: " + value);
			}
			else
			{
				Debug.Log("canceled");
			}
		}

		/// <summary>
		/// Show picker with callbacks and log selected value.
		/// </summary>
		public void TestCallbacks()
		{
			// create picker instance
			var picker = PickerTemplate.Clone();

			// copy values
```

(continues on next page)

(continued from previous page)

```
                        picker.ListView.DataSource = PickerTemplate.ListView.DataSource;

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

                void ValueSelected(int value)
                {
                        currentValue = value;
                        Debug.Log(string.Format("value: {0}", value));
                }

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

4.4.8 Popup

Note: See *Windows Animations* for open and closed animations.

Options

- Title Text `Text` (obsolete)

  GameObject to display title. Replaced with the *DialogInfo*.

- Content Text `Text` (obsolete)

  GameObject to display text. Replaced with the *DialogInfo*.

- Icon `Image` (obsolete)

  GameObject to display icon. Replaced with the *DialogInfo*.

- Info `DialogInfoBase`

  Component to display the popup info.

- AutoFocus `bool`

  Set focus to the last Selectable object in the Popup.

- CloseButton `Button`

  Button to close popup.

- Hide on Modal Click `bool`

  Close popup on click on the background if the `modal` option enabled.

Events

- OnAnimationStart `UnityEvent<bool>`

  The event is raised before the animation starts.

  Arguments: true if opening animation; false if closing animation.

Show() Method Parameters

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

- title `string`

  Popup title.
  Can be changed with `SetInfo()` method.

- message `string`

  Popup message.
  Can be changed with `SetInfo()` method.

- position `Vector3?`

  Popup position.
  Can be changed with `SetPosition()`.

- icon `Sprite`

  Popup icon.
  Can be changed with `SetInfo()` method.

- modal `bool`

  Modal popup.
  Can be changed with `SetModal()`.

- modalSprite `Sprite`

  Background image for the modal popup.
  Can be changed with `SetModal()`.

- modalColor `Color?`

  Background color for the modal popup.
  Can be changed with `SetModal()`.

- canvas `Canvas`

  Canvas to display popup. Required if popup template is prefab.
  Can be changed with `SetCanvas()`.

- content `RectTransform`

  Dialog content. Can be used instead of the *message* and *icon*.
  Can be changed with `SetContent()`.

- onClose `Action`

  Action to run when dialog closed.

Can be changed with `OnClose` field.

Minimal code

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

Advanced

```
// create popup instance
var popup = popupTemplate.Clone();
// show popup with following parameters
popup.Show(
   title: "Modal popup",
   message: "Simple Modal popup.",
   modal: true,
   modalColor: new Color(0, 0, 0, 0.8f)
);
```

Async

```
var popup = popupTemplate.Clone();
await popup.ShowAsync();
```

4.5 Input

4.5.1 Autocomplete

Note:

Difference between Autocomplete, AutoCombobox, and AutocompleteCombobox:

- Autocomplete is InputField with autocomplete feature.

- AutoCombobox is Combobox with the option to select items by typing, with it you can get selected items.

- AutocompleteCombobox is a wrapper for Autocomplete with the ability to select an action when user input is not valid.

Options

- Input Field `InputField`

  Input field.

- Target List View `TListView`

  ListView to display available values.

- Display List View `TListView`

  Selected value will be added to this ListView.

- Allow Duplicate `bool`

  TargetListView can have duplicate items if this option is enabled.
  Data type must implement `IEquatable<T>` interface for this option to work correctly.

- Data Source `List<TValue>`

  List of the all values.

- Filter `AutocompleteFilter`

  Filter settings.

  - `Startswith`

    Value should starts with the specified input.

  - `Contains`

    Value should contains with the specified input.

- Case Sensitive `bool`

  Is filter case sensitive?

- Delimiter Chars `char[]`

  Delimiter chars to split input to the words.

- Input Type `AutocompleteInput`

  Filter with the current word or the whole input.

  - `Word`
  - `AllInput`

- Result `AutocompleteResult`

  What to do with input after value selected.

  - `Append`
  - `Replace`

- Min Length `int`

  Minimal length of the input to start search.

- Search Delay `float`

  The delay in seconds between when a keystroke occurs and when a search is performed.

- Unscaled Time `bool`

  Delay with unscaled time.

- ResetListViewSelection `bool`

  Deselect selected items in the DisplayListView.

- AllowCancelOnDeselect `Func<BaseEventData, AutocompleteCustom<TValue, TListViewComponent, TListView>, bool>`

  Allow to cancel DisplayListView close on deselect event.

Events

- OnOptionSelected `UnityEvent`
- OnOptionSelectedItem `UnityEvent<TValue>`
- OnItemNotFound `UnityEvent<string>`
- OnCancelInput `UnityEvent`
- OnSearchCompleted `UnityEvent`
- OnShowOptions `UnityEvent<AutocompleteCustom<TValue, TListViewComponent, TListView>>`
- OnHideOptions `UnityEvent<AutocompleteCustom<TValue, TListViewComponent, TListView>>`

```
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;
      }
   }
}
```

4.5.2 Calendar

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

Options

- Interactable `bool`

  Is interactable?

- Date `DateTime`

  Current date.

- Date Min `DateTime`

  Minimal date.

- Date Max `DateTime`

  Maximum date.

- Format `string`

  Format of the specified `DefaultDateMin` and `DefaultDateMax`.

- First Day Of Week `DayOfWeek`

  First day of the week.

- Container `RectTransform`

  Container for the dates.

- Calendar Date Template `CalendarDateBase`

  Template for the date.

- Header Container `RectTransform`

  Container for the day of weeks.

- Calendar Day Of Week Template `CalendarDayOfWeekBase`

  Template for the day of week.

- Date Text `Text`

  Text to display the current date.

- Date Format `string`

  Date format for the `Date Text`.

- Month Text `Text`

  Text to display the current month.

- Month Format `string`

  Date format for the `Month Text`.

- Month Button `Button`

  Button to open `Month Selector`

- Month Selector `MonthSelector`

  Show list of available months.

- Year Button `Button`

  Button to open `Year Selector`

- Year Selector `YearSelector`

  Show list of available years.

Events

- OnDateChanged `UnityEvent<DateTime>`
- OnDateClick `UnityEvent<DateTime>`

```
namespace UIWidgets.Examples
{
   using UnityEngine;

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
        }

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

4.5.3 Centered Slider

The differences from a default slider:

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

Options

- Interactable `bool`

  Allow users interact with the Slider.

- Value `int`

  Current value.

- Use Value Limits `bool`

  Value cannot exceed the specified limits.

- Limit Min `int`

  Minimal limit of the value.

- Limit Max `int`

  Maximum limit of the value.

- Value Min `int`

  Minimal value.

- Value Max `int`

  Maximal value.

- Step `int`

  Value step.

- Whole Number Of Steps `bool`

  Whole number of steps for the value.

- Handle `RangeSliderHandle`

  Handle to drag.

- UsableRangeRect `RectTransform`

Usable range.

- FillRect `RectTransform`

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

- ScrollMode `CenteredSlider.ScrollModes`

  Change the value on the mouse scroll.

  - `Ignore`

    Do not change the value on the mouse scroll.

  - `UpIncrease`

    Increase the value on the mouse wheel up and decrease on the wheel down.

  - `UpDecrease`

    Decrease the value on the mouse wheel up and increase on the wheel down.

Events

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

Set value

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

Set display limits

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

Set value limits

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

4.5.4 Circular Slider

Options

- Interactable `bool`

  Is interactable?

- Handle `DragListener`

  Handle to drag.

- Arrow `RectTransform`

  Arrow.

- Value `int`

  Current value.

- Min Value `int`

  Minimal value.

- Max Value``int``

  Maximal value.

- Step `int`

  Value step.

- Start Angle `float`

  Angle for the Min Value.

- ScrollMode `CircularSlider.ScrollModes`

  Change the value on the mouse scroll.

  - `Ignore`

    Do not change the value on the mouse scroll.

  - `UpIncrease`

    Increase the value on the mouse wheel up and decrease on the wheel down.

  - `UpDecrease`

    Decrease the value on the mouse wheel up and increase on the wheel down.

Events

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

Set value

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

Set value limits

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

4.5.5 ColorPicker

Options

- Interactable `bool`

  Allow users interact with the ColorPicker.

- RGBPalette `ColorPickerRGBPalette`

  Palette (Image and Slider) to select color by RGB.

- RGBBlock `ColorPickerRGBBlock`

  Sliders and InputFields to select color by RGB.

- HSVPalette `ColorPickerHSVPalette`

  Palette (Image and Slider) to select color by HSV.

- HSVBlock `ColorPickerHSVBlock`

  Sliders and InputFields to select color by HSV.

- ABlock `ColorPickerABlock`

  Slider and InputField to select color transparency.

- HexBlock `ColorPickerHexBlockBase`

  InputField to select color by hex value.

- ColorView `ColorPickerColorBlock`

  Block to display selected color with transparency.

- ImagePalette `ColorPickerImagePalette`

  Image to select color from sprite.

- InputMode `ColorPickerInputMode`

  Input block to display: `None`, `HSV`, `RGB`.

- PaletteMode `ColorPickerPaletteMode`

  Palette block to display and it's mode: `None`, `Red`, `Green`, `Blue`, `Hue`, `Saturation`, `Value`, `HSVCircle`, `Image`.

- Color `Color`

  Selected color.

Events

- OnChange `UnityEvent<Color32>`

  The event raised when color changed.

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);
```

4.5.6 ColorPickerRange

Allow to select color in specified range of two colors.
Two versions: `ColorPickerRange` and `ColorPickerRangeHSV`.
HSV version can help to avoid getting *dirty* colors in between.

Options

- Slider `Slider`

  Slider to change color.

- SliderBackground `Image`

  Image to display color gradient using the specified shader.

- DefaultShaderHorizontal `Shader`

  Shader to display color gradient if slider is horizontal.

- DefaultShaderVertical `Shader`

  Shader to display color gradient if slider is vertical.

- ColorLeft `Color`

  Color on the left side (or bottom if slider is vertical).

- ColorRight `Color`

  Color on the right side (or top if slider is vertical).

- Color `Color`

  Selected color.

Events

- OnChange `UnityEvent<Color32>`

  The event raised when color changed.

Set color

```
ColorPickerRange.Color = Color.cyan;
```

Get color

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

Add listener

```
void Start() => ColorPickerRange.OnChange.AddListener(ColorChanged);

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

4.5.7 DateTime

Nested widgets can be safely replaced with their analogs:

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

DateScroller Options

- Current Date Time As Default `bool`
  - Default Date Time `DateTime` (`string` in Inspector window)
  - Format `string`

    Format to parse **Default Date Time**.
- Calendar `DateBase`

  Widget to select date.
- Time `TimeBase`

  Widget to select time.
- Is Scroll Blocks Used `bool`

  Is Calendar and Time widgets are *scrollers*? Required for the styles support.

Events

- OnDateTimeChanged `UnityEvent<DateTime>`

  The event raised when date changed.

  Arguments: selected datetime.

4.5.8 DateScroller, DateTimeScroller, TimeScroller

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

Note: DateTime Formats Strings

You can use *ScrollBlockResizer* to display the specified amount of items according to the widget height.

DateScroller Options

- Interactable `bool`

  User can interact with this widget.

- Current Date As Default `bool`
  - Default Date `DateTime` (`string` in Inspector window)
- Default Date Min `DateTime` (`string` in Inspector window)

  Minimal selectable date.

- Default Date Max `DateTime` (`string` in Inspector window)

  Maximum selectable date.

- Format `string`

  Format to parse **Default Date**, **Default Date Min**, and **Default Date Max**.

- Independent scroll `bool`

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

- Years `bool`

  Display years scroller.
  - Years Scroller `Scroller`
  - Years Step `int`
  - Years Format `string`
- Months `bool`

  Display moths scroller.
  - Months Scroller `Scroller`
  - Months Step `int`
  - Months Format `string`

- Days `bool`

  Display days scroller.

  - Days Scroller `Scroller`
  - Days Step `int`
  - Days Format `string`

- Events
  - OnDateChanged `UnityEvent<DateTime>`

    The event raised when date changed.

    Arguments: selected datetime.

  - OnDateClick `UnityEvent<DateTime>`

    The event raised when date setted or changed.

    Arguments: selected datetime.

DateTimeScroller Options

Same settings as DateScroller with addition:

- Hours `bool`

  Display hours scroller.

  - Hours Scroller `Scroller`
  - Hours Step `int`
  - Hours Format `string`

    Used if **AMPM** disabled.

  - Hours AMPM Format `string`

    Used if **AMPM** enabled.

- Minutes `bool`

  Display minutes scroller.

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

- Seconds `bool`

  Display seconds scroller.

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

- AMPM `bool`

  Display AMPM scroller.

  - AMPM Scroller `Scroller`

  - AMPM Format `string`

TimeScroller Options

- Interactable `bool`

  User can interact with this widget.

- Current Time As Default `bool`
  - Time Text `TimeSpan` (`string` in Inspector window)
- Default Time Min `TimeSpan` (`string` in Inspector window)

  Minimal selectable time.

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

  Maximum selectable time.

- Format `string`

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

- Independent scroll `bool`

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

- Hours `bool`

  Display hours scroller.
  - Hours Scroller `Scroller`
  - Hours Step `int`
- Minutes `bool`

  Display minutes scroller.
  - Minutes Scroller `Scroller`
  - Minutes Step `int`
- Seconds `bool`

  Display seconds scroller.
  - Seconds Scroller `Scroller`
  - Seconds Step `int`
- AMPM `bool`

  Display AMPM scroller.
  - AMPM Scroller `Scroller`
- Events
  - OnTimeChanged `UnityEvent<TimeSpan>`

    The event raised when time changed.

    Arguments: selected time.

```
namespace UIWidgets.Examples
{
   using UnityEngine;

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

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

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

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

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

Customization

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

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

```
public class ScrollBlockCustomization : MonoBehaviour
{
   [SerializeField]
```

(continues on next page)

(continued from previous page)

```
    ScrollBlock YearsScrollBlock;

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

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

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

4.5.9 RangeSlider

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

Options

- Type `RangeSliderType`

  Type of the slider.

  - `AllowHandleOverlay`

    Handles can intersects. Value scale is constant.

  - `DisableHandleOverlay`

    Handles can not intersects. Value scale is variable.

- Value Min `int/float`

  Minimal value.

- Value Max `int/float`

  Maximal value.

- Step `int/float`

  Step of the value.

- Limit Min `int/float`

  Value cannot be less that this.

- Limit Max `int/float`

Value cannot be more that this.

- Handle Min `RangeSliderHandle`

  Handle to change the minimal value.

- Handle Max `RangeSliderHandle`

  Handle to change the maximal value.

- UsableRangeRect `RectTransform`

  Usable range.

- FillRect `RectTransform`

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

- Whole Number Of Steps `bool`

  Whole number of steps for the value.

Events

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

Set values

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

Set step

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

Set limits

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

Add listener

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

void SliderChanged(int min, int max)
{
   if (slider.WholeNumberOfSteps)
```

(continues on next page)

(continued from previous page)

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

4.5.10 Rating

Options

- Interactable `bool`

  User can interact with this widget.

- Value `int`

  Default rating value.

- Value Max `int`

  Maximal rating value.

- Star Empty `RatingStar`

  Template of an empty start.

- Star Full `RatingStar`

  Template of a full start.

- Color Min `Color`

  Color for the lowest rating.

- Color Max `Color`

  Color for the highest rating.

- Lerp Mode `ColorLerpMode`

  Color lerp mode: `RGB` or `HSV`.

Events

- OnChange `UnityEvent<int>`

  The event raised when rating changed.

  Arguments: rating.

4.5.11 Scale

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

Options

- Container `RectTransform`

  Marks container.

- Main Line `Image`

  Main line.

- Show Current Value `bool`

  Show marks for the current values.

- Current Mark Template `ScaleMarkTemplate`

  Template for the current mark.

- Show Min Value `bool`

  Show mark for the min value.

- Min Mark `ScaleMarkTemplate`

  Minimum mark.

- Show Max Value `bool`

  Maximum mark.

- Scale Marks `List<ScaleMark>`

  Marks templates.

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

  Fill output list with values where marks should be displayed.

ScaleMark

- Step `float`

  Value difference between marks.

- Template `ScaleMarkTemplate`

  Mark template.

SliderScale and RangeSliderScale Components

- Scale `Scale`

  Scale gameobject.

- Format `string`

  Format to display mark value.

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

- Formatter `Func<float, string>`

  Custom formatter to use instead of format string.

CenteredSliderScale Component

- Scale `Scale`

  Scale gameobject.

- Format `string`

  Format to display mark value.

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

- Formatter `Func<float, string>`

  Custom formatter to use instead of format string.

- Negative Step Rate `float`

  Multiplicator for marks at negative side of the scale.

- Positive Step Rate `float`

  Multiplicator for marks at positive side of the scale.

4.5.12 Spinner

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

Note: In case of TextMeshPro error `IndexOutOfRangeException: Index was outside the bounds of the array` you need to reduce the font size or increase RectTransform size.

Options

- Interactable `bool`

  User can interact with this widget.

- Value Min `int/float`

  Minimal value.

- Value Max `int/float`

  Maximal value.

- Step `int/float`

  Step of the value.

- SpinnerValue `int/float`

  Current value.

- Validation `SpinnerValidation`

  Validate value on specified event.

  - `OnKeyDown`

    Value checked on every key down event.
    Some value ranges cannot processed correctly with `OnKeyDown` validation.
    For example `2..10` because to enter `10` you need to enter `1` and `1` is not valid value.

  - `OnEndInput`

    Value checked when editing has ended.

- ScrollMode `Spinner.ScrollModes`

  Change the value on the mouse scroll.

  - `Ignore`

    Do not change the value on the mouse scroll.

  - `UpIncrease`

    Increase the value on the mouse wheel up and decrease on the wheel down.

  - `UpDecrease`

    Decrease the value on the mouse wheel up and increase on the wheel down.

- AllowHold `bool`

  Change value on button hold.

- HoldStartDelay `float`

  Delay of hold in seconds to start change value.

- HoldChangeDelay `float`

  Delay of hold in seconds between each change value.

- Plus Button `ButtonAdvanced`

  Button to increase value.

- Minus Button `ButtonAdvanced`

Button to decrease value.

Events

- onPlusClick `UnityEvent`
- onMinusClick `UnityEvent`

Spinner Events

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

SpinnerFloat Options

- Format `string`

  Value format.
- Decimal Separators `char[]`

  Decimal separators.
- Number Style `NumberStyles`

  Style of the number.

SpinnerFloat Events

- onValueChangeFloat `UnityEvent<float>`
- onEndEditFloat `UnityEvent<float>`

Set maximum

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

Set minimun

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

Set value

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

Set step

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

Get value

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

4.5.13 SpinnerVector3

Combination of three *Spinner* to represent `Vector3`.

Options

- Interactable `bool`

  User can interact with this widget.

- Value `Vector3`

  Minimal value.

- SpinnerX `SpinnerFloat`

  SpinnerX for the `Vector3.x`.

- SpinnerY `SpinnerFloat`

  Spinner for the `Vector3.y`.

- SpinnerZ `SpinnerFloat`

  Spinner for the `Vector3.z`.

Events

- OnValueChanged `UnityEvent<Vector3>`

4.5.14 Switch

Analog of the default **Toggle**, but changes mark position instead of changing checkmark visibility.

Options

- IsOn `bool`

  Is on?

- Group `SwitchGroup`

  Switch group. Only one `Switch` in the same group can be on.

- Direction `SwitchDirection`

  Mark animation direction: `LeftToRight`, `RightToLeft`, `BottomToTop`, `TopToBottom`

- Mark `RectTransform`

  Animated mark.

- MarkGraphic `Graphic`

  Mark graphic.

- Background `Graphic`

  Background graphic.

- MarkOnColor `Color`

  Color of the MarkGraphic when `Switch` is on.

- MarkOffColor `Color`

  Color of the MarkGraphic when `Switch` is off.

- BackgroundOnColor `Color`

  Color of the Background when `Switch` is on.

- BackgroundOffColor `Color`

  Color of the Background when `Switch` is on.

- AnimationDuration `float`

  Animation duration.

- AnimationCurve `AnimationCurve`

  Animation curve.

- UnscaledTime `bool`

  Animate using unscaled time.

Events

- OnValuedChanged `UnityEvent<bool>`

  Event on value changed.

4.5.15 Time

Time24 has 24-hour format.
Time12 has 12-hour format with AM/PM toggle.

Options

- Interactable `bool`

  User can interact with this widget.

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

  Minimal selectable time.

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

  Maximum selectable time.
- Input Hours Adapter `InputFieldAdapter`

  InputField for the hours.
- Input Minutes Adapter `InputFieldAdapter`

  InputField for the minutes.
- Input Seconds Adapter `InputFieldAdapter`

  InputField for the seconds.
- Button Hours Increase `ButtonAdvanced`

  Button to increase hours.
- Button Hours Decrease `ButtonAdvanced`

  Button to decrease hours.
- Button Minutes Increase `ButtonAdvanced`

  Button to increase minutes.
- Button Minutes Decrease `ButtonAdvanced`

  Button to decrease minutes.
- Button Seconds Increase `ButtonAdvanced`

  Button to increase seconds.
- Button Seconds Decrease `ButtonAdvanced`

  Button to decrease seconds.
- Allow Hold `bool`

  Allow button hold after `Hold Start Delay` to increase/decrease time with each `Hold Change Delay`.
- Hold Start Delay `float`

  Seconds from button press to start increase/decrease on hold.
- Hold Change Delay `float`

  Seconds to single increase/decrease during hold.
- AMPM Button `Button`

  Button to toggle AM/PM.
- AMPM Text Adapter `TextAdapter`

  Text to display AM/PM.

Events

- OnTimeChanged `UnityEvent<TimeSpan>`

  The event raised when time changed.

  Arguments: selected time.

4.5.16 TimeAnalog

Options

- Interactable `bool`

  User can interact with this widget.

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

  Minimal selectable time.

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

  Maximum selectable time.

- Slider `CircularSlider`

  Time slider.

- Step `int`

  Time step at minutes.

- AMPM Button `Button`

  Button to toggle AM/PM.

- AMPM Text `TextAdapter`

  Text to display AM/PM.

- Hours Labels `List<GameObject>`

  Hours labels, required for the styles support.

Events

- OnTimeChanged `UnityEvent<TimeSpan>`

  The event raised when time changed.

  Arguments: selected time.

4.6 Miscellaneous

4.6.1 Audio Player

Plays `AudioClip`.

Options

- Progress `Slider`

  Slider to display and change progress.
- PlayButton `Button`

  Button to start play.
- PauseButton `Button`

  Button to pause.
- StopButton `Button`

  Button to stop.
- ToggleButton `Button`

  Button to start/stop.
- Source `AudioSource`

  Audio source used to play `AudioClip`.

4.6.2 Loading Animation

Displays the rotating arc of the circle.

Options

- Progressbar `ProgressbarDeterminate`

  Circular progress bar.
- Value Min `int`

  Minimum arc length in degrees (range 0..360).
- Value Max `int`

  Maximum arc length in degrees (range 0..360).
- ValueSpeed `int`

  Rate of arc change in degrees.
- RotateSpeed `float`

  Arc rotation speed in degrees

4.6.3 ProgressbarDeterminate

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

Options

- Max `int`

  Maximum value of the progress.

- Value `int`

  Current value of the progress.

- Full Bar Mask `Image`

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

- Full Bar Border `Image`

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

- Text Type `ProgressbarTextTypes`

  How to progress should be displayed as text.

  - `None`

    Does not display text.

  - `Percent`

    Show progress as percent like *15%*

  - `Range`

    Show progress as text like *15 / 100*

- Speed `float`

  Animation speed in the seconds.

- Speed Type `ProgressbarSpeedType`

  Specifies how speed should be interpreted.

  - `TimeToValueChangedOnOne`

    Speed is time to change progress on 1.

  - `ConstantSpeed`

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

  - `ConstantTime`

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

- Unscaled Time `bool`

  Run animation with unscaled time.

- Text Func `Func<ProgressbarDeterminateBase, string>`

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

- Background `Image`

Background image.

- Empty Bar `Image`

  Empty bar image.

- Full Bar Image `Image`

  Full bar Image.

- Empty Bar Text `Text`

  Text to display progress.

- Full Bar Text `Text`

  Text to display progress.

Set value

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

Stop animation

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

4.6.4 ProgressbarIndeterminate

Options

- Direction `ProgressbarDirection`

  Animation direction.

  - `Horizontal`
  - `Vertical`

- Bar `RawImage`

  Image to animate. Use texture type `texture` and set *Wrap Mode* to `repeat`.

- Border `Image`

  Border image.

- Mask `Image`

  Mask.

- Speed `float`

  Animation speed.

- Unscaled Time `bool`

  Run animation with unscaled time.

Start animation

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

Stop animation

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

4.6.5 Simple Tooltip

Displays the tooltip when cursor over gameobject or gameobject get focus.
`SimpleTootip` cannot be used by multiple gameobjcts unlike *Tooltip*.

Options

- Tooltip Object `GameObject`

  GameObject used as tooltip.

- Bring To Front `bool`

  Bring tooltip object to front.

- Show Delay `float`

  Delay in seconds before tooltip displayed.

- Unscaled Time `bool`

  Delay with unscaled time.

Events

- OnShow `UnityEvent`
- OnHide `UnityEvent`

4.6.6 Tooltip

Displays the generic tooltip when cursor over gameobject or gameobject get focus.
Different gameobjects can use the same tooltip gameobject.
Tooltip for custom type can be created by *Widgets Generator*.
Tooltip is automatically updated if custom types implements `IObservable` or `INotifyPropertyChanged` interface.

Using Tooltip

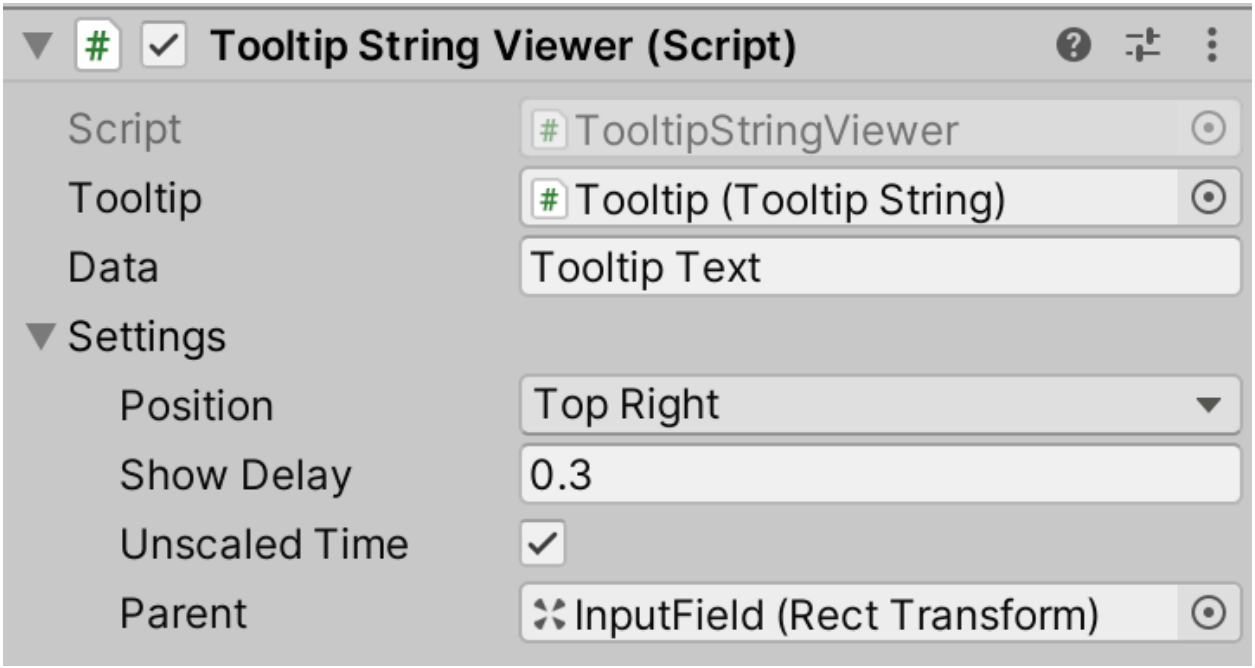
Add tooltip to gameobject:

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

Remove tooltip:

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

Also tooltip can be added with Tooltip Viewer component.



Destroy Tooltip Viewer component to remove tooltip.

Tooltip Fields and Properties

- `TData` CurrentData
- `GameObject` CurrentTarget

Tooltip Methods

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

Tooltip Events

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

Tooltip Settings

- Position `TooltipPosition`

  Tooltip position relative to target gameobject.
- Delay `float`

  Delay before tooltip displayed.
- Parent `RectTransform`

  Tooltip parent.
- UnscaledTime `bool`

  Delay specified in unscaled time.

TooltipPosition

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

- Bottom Left
- Bottom Center
- Bottom Right

Tooltip Viewer Fields

- Tooltip `TTooltip`
- Data `TData`

  Data to display.
- Settings `TooltipSettings`

  Tooltip display settings.

Tooltip Code Example

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

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

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

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

Tooltip Viewer Code Example

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

CHAPTER

FIVE

COMPONENTS

5.1 Collections Related

5.1.1 AutocompleteDataSource

Load lines from file and set them as Autocomplete.DataSource.

Options

- File `TextAsset`

  File with lines.

- CommentsStartWith `List<string>`

  Ignore lines that start with specified strings.

5.1.2 ListViewDefaultItemAspectRatio

Automatically changes item instance sizes on `ListView` resize to maintain aspect ratio.
Changes width if `ListView` is horizontal, otherwise height. Other dimensions of the item (height or width) should be set to stretch.
Add this component to the `ListView` game object to use.

Options

- Aspect Ratio `float`

  Aspect ratio of item (`width / height`).

5.1.3 ListViewDefaultItemAutoResize

Automatically changes item instances size on `ListView` resize, used when only one item should be visible. Add this component to the `ListView` game object to use.

Options

- Size Difference `Vector2`

  Difference in size between ListView and DefaultItem.
  Use the right mouse button and select the "Reset" component to update the `Size Difference` value after the DefaultItem size is changed in the editor.

5.1.4 ListViewAutoResize

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

Options

- MaxSize `float`

  Maximum size.

- UpdateRectTransform `size`

  Set RectTransform size.

5.1.5 ListViewStringDataFile

Load lines from file and set them as ListViewString.DataSource.

Options

- File `TextAsset`

  File with lines.

- CommentsStartWith `List<string>`

  Ignore lines that start with specified strings.

- Unique `bool`

  Allow only unique lines.

- AllowEmptyItems `bool`

  Allow empty strings.

- CreateNewList `bool`

  Create a new list or use `DataSource`.

5.1.6 Table Header

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

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

Options

- Interactable `bool`

  Allow interaction.

- List `ListViewBase`

  Controlled ListView.

- Allow Resize `bool`

  Allow to change columns width.

- Allow Reorder `bool`

  Allow to change columns order.

- On Drag Update `bool`

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

- Active Region `float`

  Distance from border where resize allowed.

- Cursors `Cursors`

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

- FillContextMenu `bool`

  Fill ContextMenu. Requires the `ContextMenu` component.

- ContextMenuNames `List<string>`

  Names for the ContextMenu items. They should be in the same order as columns.

- Drop Indicator `LayoutDropIndicator`

  Indicator to display new column position during column reordering.

- Drag Button `PointerEventData.InputButton`

  The button that should be pressed to process the drag event.

Cet Current Columns Order

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

Change Columns Order

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

Restore Original Columns Order

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

Disable Column

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

Enable Column

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

Add/Remove Column at Runtime

```
var order = tableHeader.GetColumnsOrder();
tableHeader.RestoreColumnsOrder();

// add new column to the header
new_column_header.SetParent(tableHeader.transform);
new_column_header.SetSiblingIndex(...);
order.Insert(..., ...);

// or remove column
Destroy(tableHeader.transform.GetChild(index));
order.RemoveAt(...);
tableHeader.Refresh()

// new DefaultItem with another set of cells
listView.DefaultItem = newDefaultItem;

// modify order with new column index or deleted column index and set it back
tableHeader.SetColumnsOrder(order);
```

5.1.7 TileViewScrollRectFitter

Resizes the ScrollRect to fit a whole number of items.
Used together with the *ListView or TileView*.

5.1.8 TreeView DataSource

Used in editor mode, allow to edit TreeView nodes.

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

5.1.9 TreeView Toggle Animation

Helper generic script to animate collapse and expand nodes.

Options

- TreeView `TTreeView`

  Target TreeView.

- Mode `ModeType`

  Animation mode.

  - `ConstantTime`
  - `ConstantSpeed`

- Time `float`

  Time in seconds to expand or collapse all nested nodes.

- Speed `float`

  Animation speed in points per second.

- Unscaled Time `bool`

  Run animation with unscaled time.

5.2 Interactions

5.2.1 Bring to Front

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

Options

- With Parents `bool`

  Bring to front GameObject with parents GameObjects.

5.2.2 Drag and Drop

Drag-and-Drop Support for the Collections

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

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

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

How Drag&Drop works

There are two components: one to process drag and another to process drop.

The Drag component is inherited from `DragSupport<TItem>` and is attached to a game object with data to drag (like ListView.DefaultItem, item in inventory). It's used: - to receive data from the game object - to show draggable data - to process results (like removing the dropped item from the original ListView).

The Drop component implements the `IDropSupport<TItem>` interface (it can implement multiple interfaces with different types) and is attached to the game object which can receive data (like ListView, inventory or inventory cell, terrain). It is used: - to check if a drop is possible when the pointer over the game object (like container has enough space or data meets some condition, show a DropIndicator if the drop is possible) - to process drop when the pointer is released (like adding an item to the container) - to process canceled drop when the pointer leaves the game object (like hide DropIndicator)

The Drag component looks for the target under the pointer with the Drop component that can accept a `TItem` and call `bool CanReceiveDrop(TItem data, PointerEventData eventData)` to check if the target can receive the dragged item. On pointer release called `Drop(TItem data, PointerEventData eventData)` for the drop component if found and then `Dropped(bool success)` for the drag component.

Common Drag Options

- Allow Drag `bool`

  Allow drag.

- Cursors `Cursors`

  Custom cursors to show the allowed and denied drop states.

- Event System Select `bool`

  Makes the draggable object selected by EventSystem.

- Handle `DragSupportHandle` *optional*

  Custom handle to drag, it not specified will be dragged by current instance.

- Drag Button `PointerEventData.InputButton`

Pressed pointer button to drag object.

- Drag Delay `float`

  How many seconds must pass from the click to the start of dragging.

- Redirect Drag To ScrollRect `bool`

  Redirects the drag events to the parent ScrollRect if the drag starts before DragDelay time is passed.

- Unscaled Time `bool`

  If enabled Drag Delay time will counted using `Time.unscaledTime`; otherwise will be used `Time.time`

Collections Drag Options

- ListView `TListView` *optional*

  ListView instance.
  Not available for TreeView.

- DragInfo `TComponent` *optional*

  Component to display the dragged data.

- DragInfo Offset `Vector3`

  Offset from the cursor position for the DragInfo.

- Delete After Drop `bool`

  Delete item from collection after drop.
  Not available for TreeView.

Collections Drop Options

- Drop Position `NearestType`

  Drop position.

  - `Auto` insert dropped item to the nearest position.
  - `Before` insert dropped item before item under pointer.
  - `After` insert dropped item after item under pointer.

- Drop Indicator `ListViewDropIndicator`

  Indicator to display position where dropped item will be inserted.

- Delete Node After Drop `bool`

  Delete dropped node from TreeView.
  Not available for TreeView.

- Receive Items `bool`

  Receive dropped items.

- Receive Nodes `bool`

  Receive dropped nodes.

TreeView Drop Options

- Drop Position `NearestType`

  Drop position.

  - `Auto` insert dropped item to the nearest position
  - `Before` insert dropped item before item under pointer
  - `After` insert dropped item after item under pointer

- Drop Indicator `ListViewDropIndicator`

  Indicator to display position where dropped item will be inserted.

- Receive Items `bool`

  Receive dropped items.

- Receive Nodes `bool`

  Receive dropped nodes.

TreeView Node Drop Options

- Drop Indicator `ListViewDropIndicator`

  Indicator to display position where dropped item will be inserted.

- Delete Node After Drop `bool`

  Delete dropped node from TreeView.

- Receive Items `bool`

  Receive dropped items.

- Reorder Area `float`

  Distance in percent of height from border to add dropped node before/after instead of drop as sub-node. Allowed value range is 0f..0.5f

Custom Drag Support

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

Methods

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

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

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

   /// <summary>
```

(continues on next page)

(continued from previous page)

```
    /// 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();
        }

        /// <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);
```

(continues on next page)

(continued from previous page)

```
    }

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

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

    /// <summary>
    /// Start this instance.
    /// </summary>
    protected virtual void Start()
    {
        if (DragInfo != null)
        {
            DragInfo.SetActive(false);
        }
    }

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

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

        DragInfo.SetActive(true);

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

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

(continues on next page)

(continued from previous page)

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

Custom Drop Support

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

Methods

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

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

`CanReceiveDrop` function allows to accept only nodes with names ends with *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>
    {
        /// <summary>
        /// InputField.text value before drop.
        /// Can be used to swap content with drag source.
        /// </summary>
        public string OriginalData;

        #region IDropSupport<string>

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

(continues on next page)

(continued from previous page)

```
            OriginalData = input.text;
            input.text = data;
        }

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

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

        #endregion

        #region IDropSupport<TreeNode<TreeViewItem>>

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

        /// <summary>
        /// Determines whether this instance can receive drop with the specified data and␣
↪eventData.
        /// </summary>
        /// <returns>true if this instance can receive drop with the specified data and␣
↪eventData; otherwise, false.</returns>
        /// <param name="data">Data.</param>
        /// <param name="eventData">Event data.</param>
        public bool CanReceiveDrop(TreeNode<TreeViewItem> data, PointerEventData eventData)
        {
```

(continues on next page)

(continued from previous page)

```
            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
    }
}
```

Swapping content between Drag and Drop components

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

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

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

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

                // replace dragged text with drop target text
```

(continues on next page)

(continued from previous page)

```
            GetComponent<InputField>().text = (target as InputFieldDropSupport).
↪OriginalData;
         }
         else
         {
            Dropped(false);
         }

         ResetCursor();
      }
   }
}
```

Adding limitations to the Drop component

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

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

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

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

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

5.2.3 DragOneDirection

Modifies the drag event to work in only one direction. Used with nested ScrollRects.

Options

- MinDistance `float`

  Minimal drag distance to determine drag direction.

- DragButton `PointerEventData.InputButton`

  The button that should be pressed to process the drag event.

5.2.4 Drag Redirect

Use it to drag multiple objects simultaneously or increase drag area for the object that should have small visible size.

Options

- Redirect To `GameObject`

  GameObject to receive drag events.

- Mark as Used `bool`

  Mark drag events as used after they redirected.

- Min Distance `Vector2`

  Required distance to redirect `OnDrag` and `OnEndDrag` events.

5.2.5 Draggable

The Draggable component is used to change position, like dragging a window by header. It shouldn't be confused with *Drag and Drop*.

Options

- Interactable `bool`

  Allow interaction.

- Handle `GameObject` *optional*

  GameObject used to drag current GameObject.

- Horizontal `bool`

  Allow horizontal drag movement.

- Vertical `bool`

  Allow vertical drag movement.

- Restriction `DraggableRestriction`:
  - `None`: no restriction.

  - `Strict`: does not allow drag outside the parent.
  - `After Drag`: does not allow drag outside the parent, applied after drag ended.
- Curve `AnimationCurve`

  Animation curve used to animate applied **After Drag** restriction.
- Unscaled Time `bool`

  Run animation with unscaled time.
- Snap Grids `List<SnapGridBase>`

  Allow snapping the `RectTransform` position to the nearest line.
  See *SnapGrid* and *SnapLines*.
- Snap Distance `Vector2`

  Maximum distance to lines where snapping is available.

Properties

- Target `RectTransform`

  Target to drag; the self is by default.

Events

- OnStartDrag `UnityEvent<Draggable>`
- OnDrag `UnityEvent<Draggable>`
- OnEndDrag `UnityEvent<Draggable>`
- OnSnap = `UnityEvent<Draggable, SnapGridBase.Result>`
- OnEndSnap = `UnityEvent<Draggable, SnapGridBase.Result>`
- OnTargetChanged `UnityEvent<Draggable>`

5.2.6 Groupable

Allows to select a group of the gameobjects; and then resize, rotate, align all of them simultaneously.
Can select only elements with the same parent as the Groupable component.

Shared components settings between Groupable and the selected elements:

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

Options

- Interactable `bool`

  Allow interaction.

- Highlight Template `RectTransform` *optional*

  Template to highlight selected gameobjects.

- Selection Mode `Groupable.Mode`

  Selection mode.

  - `Contains`

    Selects only gameobjects fully inside the selection area.

  - `Overlaps`

    Selects gameobjects inside the selection area or partially overlaps the selection area.

- Group Rotation `bool`

  If enabled selected gameobjects will be rotated as part of the group; otherwise each separately.

Events

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

5.2.7 Object Sliding

Component to drag GameObject horizontally or vertically between specified positions.

Options

- Interactable `bool`

  Allow interaction.

- Positions `List<float>`

  Allowed positions for this object.

- Direction `ObjectSlidingDirection`

  Slide direction.

  - `Horizontal`
  - `Vertical`

- Movement `AnimationCurve`

  Animation curve.

- Unscaled Time `bool`

  Animate with unscaled time.

Helper components

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

- Object Sliding Horizontal Helper
  - Object on Left `List<RectTransform>`

    List of the objects on the left side of the current object.
  - Object on Right `List<RectTransform>`

    List of the objects on the right side of the current object.
- Object Sliding Vertical Helper
  - Object on Top `List<RectTransform>`

    List of the objects on the top side of the current object.
  - Object on Bottom `List<RectTransform>`

    List of the objects on the bottom side of the current object.

5.2.8 Pinchable

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

Options

- Interactable `bool`

  Allows users interaction.
- AllowDrag `bool`

  Allows drag.
- AllowResize `bool`

  Allows resize.
- AllowRotate `bool`

  Allows rotation.

Events

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

5.2.9 Resizable

Allows resizing gameobject by size or scale.

Options

- Interactable `bool`

  Allow users to change the size of the GameObject.

- Resize Directions `Resizable.Directions`

  Allowed resizing directions.

- Type `ResizeType`

  Resize type.

  - `Size`

    Resize by changing size of the gameobject.

  - `Scale`

    Resize by changing scale of the gameobject.

- Include Corners `bool`

  Allow resize when cursor in the one of the corners. Should be disabled to use together with *Rotatable* component.

- Integer Size `bool`

  If enabled size is rounded to the integer number. Reason: size can be float number if gameobject is rotated.

- Update RectTransform `bool`

  Change RectTransform size.

- Update LayoutElement `bool`

  Change LayoutElement size.

- Active Region `float`

  Distance from border where resize allowed.

- Min Size `Vector2`

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

- Max Size `Vector2`

  Maximum size in points, for the `Scale` type limits is checked against *width * scale.x* and *height * scale.y*.
  Not applied if size is zero.

- Stop At Parent Borders `bool`

  If enabled then the component cannot be resized to exceed the parent borders.

- Keep Aspect Ratio `bool`

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

- Cursors `Cursors`

  Custom cursors to the allowed resize state.

- Snap Grids `List<SnapGridBase>`

  Allow snapping the `RectTransform` position to the nearest line.
  See *SnapGrid* and *SnapLines*.

- Snap Distance `Vector2`

  Maximum distance to lines where snapping is available.

Events

- OnStartResize `UnityEvent<Resizable>`
- OnResize `UnityEvent<Resizable>`
- OnEndResize `UnityEvent<Resizable>`
- OnResizeDelta = `UnityEvent<Resizable, Resizable.Regions, Vector2>`
- OnResizeDirectionsChanged `UnityEvent<Resizable>`
- OnTargetChanged `UnityEvent<Resizable>`

Properties

- Target `RectTransform`

  Target to resize; the self is by default.

Resize Children With Parent

There are a few ways to resize children with parent:

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

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

  Video about anchors.

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

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

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

5.2.10 Resizable Handles

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

Options

- Interactable `bool`

  Allow users to change the size of the GameObject.

- Own Handles `bool`

  If enabled you can specify your own handles for the current component.

  If disabled you can specify Handles Source for current component, this allows you to create a single set of handles instead of duplicate them for each component.

  Handles should be acquired with `GetSourceHandles()` and returned with `ReleaseSourceHandles()` functions.

- Handles Source `ResizableHandles`

  Handles source to use if Own Handles disabled.

- Top Left `DragListener` *optional*

  Top left handle.

- Top Center `DragListener` *optional*

  Top center handle.

- Top Right `DragListener` *optional*

  Top right handle.

- Middle Left `DragListener` *optional*

  Middle left handle.

- Middle Right `DragListener` *optional*

  Middle right handle.

- Bottom Left `DragListener` *optional*

  Bottom left handle.

- Bottom Center `DragListener` *optional*

  Bottom center handle.

- Bottom Right `DragListener` *optional*

  Bottom right handle.

- HandleState `Func<ResizableHandles, BaseEventData, bool, bool>` *optional*

  Return handle state (enabled/disabled) on select/deselect event (got or lost focus).

  Use case: show Rotatable and Resizable handles only if target (or one of handles) is selected, otherwise deselect.

Events

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

5.2.11 Rotatable

Allows rotating gameobject around its pivot.

Options

- Interactable `bool`

  Allow users to change the rotation of the GameObject.
- Rotate Directions `Rotatable.Directions`

  Allowed corners to apply the rotation.
- Active Region `float`

  Distance from border where rotation allowed.
- Limit Rotation `bool`

  Allows rotating objects only with the specified angles range.
  - Angle Min `float`

    Allowed value is in range [-180..180].
  - Angle Max `float`

    Allowed value is in range [-180..180].
- Angle step `float`

  Allowed value is in range [0..180). Set 0 to disable.
- Cursors `Cursors`

  Custom cursors to show the allowed rotation state.

Events

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

Properties

- Target `RectTransform`

  Target to rotate; the self is by default.

5.2.12 Rotatable Handle

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

Options

- Interactable `bool`

  Allow users to change the rotation of the GameObject.

- Own Handle `bool`

  If enabled you can specify your own handle for the current component.
  If disabled you can specify Handle Source for current component, this allows you to create a single handle instead of duplicate it for each component.
  Handle should be acquired with `GetSourceHandle()` and returned with `ReleaseSourceHandle()` functions.

- Handle Source `RotatableHandle`

  Handle source to use if Own Handle disabled.

- Handle `DragListener` *optional*

  Handle.

- HandleState `Func<RotatableHandle, BaseEventData, bool, bool>` *optional*

  Return handle state (enabled/disabled) on select/deselect event (got or lost focus).
  Use case: show Rotatable and Resizable handles only if target (or one of handles) is selected, otherwise deselect.

Events

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

5.3 Layout

5.3.1 EasyLayout

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

Options

- Main Axis `Axis`

  Determine how elements will be placed (at horizonal or vertical direction first).

- Layout Type `LayoutTypes`
  - `Compact`: Compactly places the elements.
  - `Grid`: Places elements in the grid. Cell size is not fixed and depend on elements sizes in the same row and column.
  - `Flex`: Places elements like CSS flexbox layout.
  - `Staggered`: Places elements one-by-one to the shortest column or row depending on the main axis.
  - `Ellipse`: Places elements one-by-one on the border of the ellipse or the circle starting from `Angle Start` and `Angle Step` distance between items.
  - `Hex`: Places elements in the hexagonal grid.
- Group Position `Anchors`

  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 `HorizontalAligns`

  Only for the `Compact` layout.

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

- Inner Align `InnerAligns`

  Only for the `Compact` layout.

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

- Compact Constraint `CompactConstraints`

  Only for the `Compact` layout.
  - `Flexible`: Rows and columns count depends on the parent size.
  - `Max Column Count`
  - `Max Row Count`
- Compact Constraint Count `int`

  Only for the `Compact` layout.

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

- Cell Align `Anchors`

  Only for the `Grid` layout.

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

- Grid Constraint `GridConstraints`

  Only for the `Grid` layout.
  - `Flexible`: Rows and columns count depends on the parent size.

  - `Fixed Column Count`
  - `Fixed Row Count`

- Grid Constraint Count `int`

  Only for the `Grid` layout.

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

- Flex Setting `EasyLayoutFlexSettings`

  Only for the `Flex` layout.

  - Wrap `bool`

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

  - Justify Content `EasyLayoutFlexSettings.Content`

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

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

  - Align Content `EasyLayoutFlexSettings.Content`

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

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

  - Align Items `EasyLayoutFlexSettings.Items`

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

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

- Staggered Settings `EasyLayoutStaggeredSettings`

Only for the `Staggered` layout.

- Fixed Block Count `bool`

  Count of the rows or columns.

- Blocks Count `int`

- Ellipse Settings `EasyLayoutEllipseSettings`

  Only for the `Ellipse` layout.

  Set equal width and height for the circle layout.

  RectTransform pivot is used as the center of the ellipse.

  - Width Auto `bool`

    RectTransform width is used as the width of the ellipse.

  - Width `float`

    Ellipse width if `Width Auto` disabled.

  - Height Auto `bool`

    RectTransform height is used as the height of the ellipse.

  - Height `float`

    Ellipse height if `Height Auto` disabled.

  - Angle Start `float`

    Position of the first element in the degrees.

  - Angle Step Auto `bool`

    Are elements placed with equal angular distance or specified `Angle Step`?

  - Angle Step `float`

    Elements placed with specified angular distance between neighbour elements.

  - Fill `EllipseFill`

    Determines how to calculate the distance between elements if `Angle Step Auto` enabled.

    * `Closed`: angular distance is 360 degrees divided into the elements count; distance is the same between the first and last elements.
    * `Arc`: angular distance is arc length divided into the elements count minus one

  - Arc Length `float`

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

    Can be more than 360 degrees.

  - Align `EllipseAlign`

    Determines how elements are placed on the ellipse border.

    * `Outer`: right borders of the elements are placed on the ellipse border.
    * `Center`: center of the elements are placed on the ellipse border.
    * `Inner`: left borders of the elements are placed on the ellipse border.

  - ElementsRotate `bool`

Rotate elements according to position or not.

  - ElementsRotationStart `float`

    Initial rotation of the elements.

- Hex Settings `EasyLayoutHexSettings`
  - Orientation `OrientationMode`
    - `FlatTop` Flat-top orientation.
    - `PointyTop` Pointy-top orientation.
  - Coordinates `CoordinatesMode`
    - `Read` Read coordinates from the `HexCoordinates` component and place components in the grid according to those coordinates.
    - `Write` Automatically places components to the grid and writes calculated coordinates to the `HexCoordinates` component.
  - Shoves Odd `bool` If enabled shoves odd rows to the bottom (if `FlatTop`) or right (if `PointyTop`); otherwise, shoves even rows.
  - Constraint `HexConstraints`
    - `Flexible` No constraints.
    - `FixedColumnCount` Constraint the number of columns to a specified number.
    - `FixedRowCount` Constraint the number of rows to a specified number.
    - `CellsPerRow` Constraint the cells per row to a specified number.
    - `CellsPerColumn` Constraint the cells per column to a specified number.
  - ConstraintCount `int` Number for the specified constraint.
  - `Decrease Shoved` Shoved rows or columns will have 1 cell less than the specified constraint (only for the `CellsPerRow` and `CellsPerColumn` constraint).
- Spacing `Vector2`

  Empty space between elements.

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

  For `Hex` layout it is recommended to have following ratio:

  - `FlatTop`: `Y` should be `X` / 2
  - `PointyTop`: `Y` should be `X` * 2
- Symmetric `bool`

  Use symmetric margin.
- Margin `Vector2`

  Empty space from parent edges.
- Skip Inactive `bool`

  Do not reserve space for disabled elements.
- Right To Left `bool`

  The order of placement of elements.

- Top To Bottom `bool`

  The order of placement of elements.

- Reset Rotation `bool`

  Reset rotation of the elements to 0.

- Movement Animation `bool`

  Animate elements repositioning.

- Movement Animate All `bool`

  Animate all elements if enabled; otherwise new elements will not be animated.

- Movement Curve `AnimationCurve`

  Movement animation curve.

- Resize Animation `bool`

  Animate elements resizing.

- Resize Animate All `bool`

  Animate all elements if enabled; otherwise new elements will not be animated.

- Resize Curve `AnimationCurve`

  Resize animation curve.

- IgnoreLayoutElementSizes `bool`

  ILayoutElement options will be ignored. Increases performance without side effects if `Children Width` and `Children Height` are not controlled.

- Children Width `ChildrenSize`
  - `Do nothing`: do not resize elements.
  - `Set Preferred`: set element width to `Preferred Width`.
  - `Set Max From Preferred`: set maximum of the `Preferred Width` from the all elements.
  - `Fit Container`: change children size in range from minimal to preferred to fit container.
  - `Set Preferred and Fit Container`: set children size to preferred, then increase size proportionally `Flexible Width` to fit parent width if required.
  - `Shrink On Overflow`: decrease elements width if summary width more than parent width including margin.

- Children Height `ChildrenSize`

  Similar to `Children Width`

Events

- Settings Changed `UnityEvent`

  Event, raised after any setting was changed.

5.3.2 EasyLayoutEllipseScroll

Scroll for the `EasyLayout` with `Ellipse` layout type.

Options

- IsHorizontal `bool`

  Is scroll horizontal or vertical?
- DragSensitivity `float`
- ScrollSensitivity `float`
- ScrollValue `float`

  Scroll position.
- Inertia `bool`
- TimeToStop `float`

  Time until inertia stopped.
- UnscaledTime `bool`

  Animate inertia scroll with unscaled time.
- DragButton `PointerEventData.InputButton`

  The button that should be pressed to process the drag event.

Events

- OnScrollEvent `UnityEvent`
- ScrollVelocity `UnityEvent`

5.3.3 Hexagonal Grid Coordinates

Each game object under the `EasyLayout` control with the `Hex` layout has a `HexCoordinates` component.

It provides access to the coordinates of the object in the grid in the different coordinate systems: `OffsetCoordinates` and `CubeCoordinates`.

Also, this component provides coordinates for layout in case `Coordinates` is `Read` mode.

See more in *the guide <https://www.redblobgames.com/grids/hexagons/>* about hexagonal grids.

Note: You can use `HexCoordinatesDebug` component with `TextAdapter` component to display cells coordinates for the debug.

Options

- Row `int`
- Column `int`

Events

- On Coordinates Changed `UnityEvent`

  The event is raised when coordinates are changed.

Properties

- Offset `OffsetCoordinates`

  Readonly.
- Cube `CubeCoordinates`

  Readonly.

5.3.4 HexLayoutBuilder

This component simplifies the creation of custom grids.

- TemplateFlatTop `HexCoordinates`

  Cell template for the `FlatTop` orientation.
- TemplatePointyTop `HexCoordinates`

  Cell template for the `PointyTop` orientation.
- Blocks `List<Block>`

  If the `EasyLayout.HexSettings.Orientation` is `FlatTop` then each block describes a column. Otherwise, each block describes a row.
- Instances `IReadOnlyDictionary<CubeCoordinates, HexCoordinates>`

  Provides access to the cells by coordinates.

Block

- Start `int`

  Index of the first cell.
- Cells `int`

  Cells in on block (row or column).

5.3.5 LayoutElementMax

Allows to control the maximum preferred sizes of the `LayoutElement`.

Options

- ignoreLayout `bool`

  Should this RectTransform be ignored by the layout system?

- layoutPriority `int`

  The Priority of layout this element has.

- MaxWidth `float`

  Maximum preferred height.

- MaxHeight `float`

  Maximum preferred width.

5.3.6 LayoutPlaceholder

Used as replacement for the open *Combobox* if it is under `LayoutGroup` control.

Usage

```
var placeholder = LayoutPlaceholder.Create(transform as RectTransform);
placeholder.Show();

// ....

placeholder.Hide();
```

5.3.7 Layout Switcher

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

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

Options

- Objects `List<RectTransform>`

  List of the controlled objects.

- Default Display Size (inches) `float`

  Display size to use when actual display size cannot be detected.

- Layouts `List<UILayout>`

  List of the layouts.

- Name `string`

  Layout name.

- Aspect Ratio `Vector2`

  Aspect ratio for this layout.

- Max Display Size (inches) `float`

  Maximum size of the display for this layout (layout will not be used if display size more than the specified one).

Events

- LayoutChanged `UnityEvent<UILayout>`

5.3.8 LimitMaxSize

Limits RectTransform width or height if its size is relative to the parent.

Options

- LimitWidth `bool`
- MaxWidth `float`

  Maximum allowed width.

- LimitHeight `bool`
- MaxHeight `float`

  Maximum allowed height.

5.3.9 RectTransform Cover

Changes the RectTransform size (while preserving its ratio) to the smallest possible size to fill the parent, leaving no empty space. It is recommended to add a Mask component to the parent.

Same as `AspectRatioFitter` with `Aspect Mode == Envelope Parent`, but the aspect ratio is taken from ILayoutElement's preferred width and height instead of defined as property.

5.4 Event Listeners

5.4.1 ClickListener

Events

- ClickEvent `UnityEvent<PointerEventData>`

  The event on pointer click.

- DownEvent `UnityEvent<PointerEventData>`

  The event on pointer down.

- UpEvent `UnityEvent<PointerEventData>`

  The event on pointer up.

- DoubleClickEvent `UnityEvent<PointerEventData>`

  The event on pointer double click with left mouse button.

5.4.2 InputFieldListener

Used by *Autocomplete*.

Events

- onSelect `UnityEvent<BaseEventData>`

  The event on game object select.

- onDeselect `UnityEvent<BaseEventData>`

  The event on game object deselect.

- OnMoveEvent `UnityEvent<AxisEventData>`

  The event on arrow keys press.

- OnSubmitEvent `UnityEvent<BaseEventData, bool>`

  The event on tab key or enter key press, second argument is enter key pressed.

5.4.3 ResizeListener

Events

- OnResize `UnityEvent`

  The event raised when `RectTransform` size changed.

- OnResizeNextFrame `UnityEvent`

  The event raised on next frame after `RectTransform` size changed.

5.4.4 ScrollListener

Events

- ScrollEvent `UnityEvent<BaseEventData>`

  The event on pointer scroll.

5.4.5 SelectListener

Events

- onSelect `UnityEvent<BaseEventData>`

  The event on game object select.

- onDeselect `UnityEvent<BaseEventData>`

  The event on game object deselect.

5.4.6 TransformListener

Events

- OnTransformChanged `UnityEvent`

  The event raised when transform.hasChanged enabled.

5.5 Mobile-Specific Components

5.5.1 SafeArea

Change RectTransform size to fit Screen.safeArea. RectTransform should be direct child of root Canvas game object.

Events

- OnScreenChange `UnityEvent<SafeArea.Borders>`

5.5.2 Swipe

Provide swipe events.

Options

- Unscaled Time `bool`

  Use unscaled time.

- Max Time `float`

  If dragged longer than the specified time then it is not swipe event.

- Required Distance `float`

  Minimum distance to be swiped.

- Min Distance `float`

  Minimum distance at X or Y axis to be swiped at those axes.

Events

- OnSwipe `UnityEvent<Swipe.Direction>`

5.6 ScrollRect Related

5.6.1 Scrollbar Min Size

Allow to set minimal scrollbars sizes of the ScrollRect.

Options

- Horizontal Min Size `float`

  Minimal size of the horizontal scrollbar.

- Vertical Min Size `float`

  Minimal size of the vertical scrollbar.

5.6.2 ScrollRectAutoScroll

Allows scrolling content during *drag & drop* when the pointer is in less than a specified distance from the border. Options ——-

- Area `float`

  ScrollRect will be automatically scrolled if the pointer in less then a specified distance from the border during drag&drop.

- Speed `float`

  Speed of auto-scroll.

- Unscaled Time `bool`

  Specify time type used by scroll animation. If enabled then will be used `Time.unscaledTime`; otherwise will be used `Time.time`.

5.6.3 ScrollRectContentResize

Resizes `ScrollRect.content` children's game objects to match `ScrollRect` size. Used to resize carousel slides to fill the full screen.

5.6.4 ScrollRect DragSensitivity

Allows to change ScrollRect *Drag Sensitivity* similar to *Scroll Sensitivity*.

- 1f is the default drag speed
- more than 1 to increase (2f is two time faster)
- less than 1 to decrease (0.5f is two time slower)
- negative to drag in a reverse direction

5.6.5 ScrollRect Events

Provide pull events for the ScrollRect.

Options

- Thresholds `PullThreshold`

  Separate thresholds values for each pull direction to raise events.

Events

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

5.6.6 ScrollRect Footer

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

Options

- ScrollRect `ScrollRect`

  ScrollRect.

- Block `RectTransform`

  Actual footer block.

- IsHorizontal `bool`

  ScrollRect direction.

- DisplayType `ScrollRectHeaderType`

Display type.

- `Reveal`

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

- `Resize`

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

- MinSize `float`

  Minimal size of the footer.

5.6.7 ScrollRect Header

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

Options

- ScrollRect `ScrollRect`

  ScrollRect.

- Block `RectTransform`

  Actual header block.

- IsHorizontal `bool`

  ScrollRect direction.

- DisplayType `ScrollRectHeaderType`

  Display type.

  - `Reveal`

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

  - `Resize`

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

- MinSize `float`

  Minimal size of the header.

5.6.8 ScrollRectRestrictedDrag

Limit drag distance.

Options

- MaxDrag `Vector2`

  Maximum allowed drag distance.

5.7 SnapGrid

5.7.1 SnapGrid

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

Options

- Snap Border Inside `SnapGridBase.Border`

  Allow snapping to the inner side of the border.

- Snap Border Outside `SnapGridBase.Border`

  Allow snapping to the outer side of the border.

- Padding `Vector2`

  Padding from borders.

- Step `Vector2`

  Size of the grid cells.

- Spacing `Vector`

  Empty space between cells.

- Snap To Spacing `bool`

  Allow spacing to inner sides of the spacing lines.

SnapGridBase.Border

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

Events

- OnLinesChanged `UnityEvent`

  Raised when lines changed.

5.7.2 SnapGridDetector

Detects *SnapGrid* under cursor during the drag, allowing snap to the automatically detected grid instead of the specified manually.
Used together with the `ISnapGridSupport``components, like :doc:`resizable`, :doc:`draggable`, and ``DropRectTransform`.

Options

- Mode `SnapGridDetector.Modes`
  - Add
  - Replace
- DragButton `PointerEventData.InputButton`

  The button that should be pressed to process the drag event.

5.7.3 SnapLines

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

Options

- Snap Border Inside `SnapGridBase.Border`

  Allow snapping to the inner side of the border.
- Snap Border Outside `SnapGridBase.Border`

  Allow snapping to the outer side of the border.
- Lines X `ObservableList<SnapGridBase.LineX>`

  Lines on X axis.
- Lines Y `ObservableList<SnapGridBase.LineY>`

  Lines on Y axis.

SnapGridBase.LineX

- X `float`

  Position on X axis.

- Snap Left `bool`

  Allow snapping by left side of the `RectTranform` (right of the line).

- Snap Right `bool`

  Allow snapping by right side of the `RectTranform` (left of the line).

SnapGridBase.LineY

- Y `float`

  Position on Y axis.

- Snap Top `bool`

  Allow snapping by top side of the `RectTranform` (bottom of the line).

- Snap Bottom `bool`

  Allow snapping by bottom side of the `RectTranform` (top of the line).

Events

- OnLinesChanged `UnityEvent`

  Raised when lines changed.

5.8 Windows

5.8.1 Notifications Animations

Animations to show and hide *Notifications*.
Add this component to the *Notifications* to use.

Note: See *Windows Animations* on how to create custom animations.

Options

- Unscaled Time `bool`

  Animation with use unscaled time if enabled.

- Disable Interactable `bool`

  Blocks all interactions with dialog while the animation is running.
  User cannot press any button, change the values of other widgets, and so on.

- Show `NotificationAnimations.ShowMode`

  Animation to show notification.

  - `Rotate` - change X or Y rotation axis of the notification
  - `Explode` - change width or height of the notification
  - `FadeIn` - change transparency of the notification
  - `SlideRight` - move notification from the right
  - `SlideLeft` - move notification from the left
  - `SlideUp` - move notification from the top
  - `SlideDown` - move notification from the bottom

- ShowHorizontal `bool`

  Animate in horizontal direction (only for the `Rotate` and `Explode` animations).

- ShowAnimationCurve `AnimationCurve`

  Animation curve.

- Hide `NotificationAnimations.HideMode`

  Animation to hide notification.

  - `Rotate` - change X or Y rotation axis of the notification
  - `Collapse` - change width or height of the notification
  - `FadeIn` - change transparency of the notification
  - `SlideRight` - move notification to the right
  - `SlideLeft` - move notification to the left
  - `SlideUp` - move notification to the top
  - `SlideDown` - move notification to the bottom

- HideHorizontal `bool`

  Animate in horizontal direction (only for the `Rotate` and `Collapse` animations).

- HideAnimationCurve `AnimationCurve`

  Animation curve.

5.8.2 Windows Animations

Animations to show and hide windows (*Dialog*, *Picker*, *Popup*, *Notifications*).
Add this component to the window object to use.
Notifications also has other animations: *Notifications Animations*.

Options

- Unscaled Time `bool`

  Animation with use unscaled time if enabled.

- Animation Curve `AnimationCurve`

  Curve used to animate values.

- Animate Alpha `bool`

  Animate window transparency.

- Animate Scale `bool`

  Animate window scale.

- Animate Anchors `bool`

  Animate window `RectTransform` anchors.

- Anchor X `float`

  Target `RectTransform` anchor X value.

- Anchor Y `float`

  Target `RectTransform` anchor Y value.

- Disable Interactable `bool`

  Blocks all interactions with dialog while the animation is running.
  User cannot press any button, change the values of other widgets, and so on.

Custom Animations

You can create your animations by implementing the `IWindowAnimations` interface.

```
public interface IWindowAnimations
{
        bool Enabled
        {
                get;
        }

        IEnumerator Open();

        IEnumerator Close();
}
```

5.9 ButtonAdvanced

The `Button` component with exposed events on pointer enter/exit/down/up.
Legacy. Not recommended to use. Better use separate listeners with the default `Button` component.

5.10 Events

- onPointerEnter `UnityEvent<PointerEventData>`
- onPointerExit `UnityEvent<PointerEventData>`
- onPointerDown `UnityEvent<PointerEventData>`
- onPointerUp `UnityEvent<PointerEventData>`

5.11 CalendarMultipleDate

Replacement for the default `CalendarDate` component to use together with *CalendarMultipleDates* component.

5.11.1 Options

- Dates `CalendarMultipleDates`

  Component with list of the selected dates.

5.12 CalendarMultipleDates

Provides list of the selected dates for the *Calendar*, used together with *CalendarMultipleDate*.

5.12.1 Options

- DataSource `ObservableList<DateTime>`

  Selected dates.

5.13 ColorsList

Helper to add current color from the *ColorPicker* or *ColorPickerRange* to the `ListViewColors`.

5.13.1 Options

- ColorPicker `ColorPicker`
- ColorPickerRange `ColorPickerRange`

  Used only if ColorPicker not specified.
- ListView `ListViewColors`
- AddButton `Button`

  Button to add color.

5.14 ComponentPool

Generic object pool.
Legacy. Not recommended to use. Better use ObjectPool instead.

5.15 Single Line and Multi Line Connectors

Draw a line from current gameobject to the specified targets.

5.15.1 SingleConnector Options

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

  Builder to draw custom lines.

5.15.2 MultipleConnector Options

- Material `Material`
- Color `Color`
- Raycast Target `bool`
- Sprite `Sprite`
- Lines `ObservableList<ConnectorLine>`

  Lines list.
- Builder `ILineBuilder`

  Builder to draw custom lines.

5.15.3 Connector Line

- Target `RectTransform`
- Start `ConnectorPosition`

  Start point of the line: `Top`, `Bottom`, `Left`, `Right`, `Center`.
- End `ConnectorPosition`

  End point of the line: `Top`, `Bottom`, `Left`, `Right`, `Center`.
- Type `ConnectorType`

  Line type: `Straight` or `Rectangular`.
- Arrow `ConnectorArrow`

  Arrow type: `None`, `Forward`, `Backward`. Multiple types can be selected.
- Thickness `float`

  Line thickness.
- Margin `float`

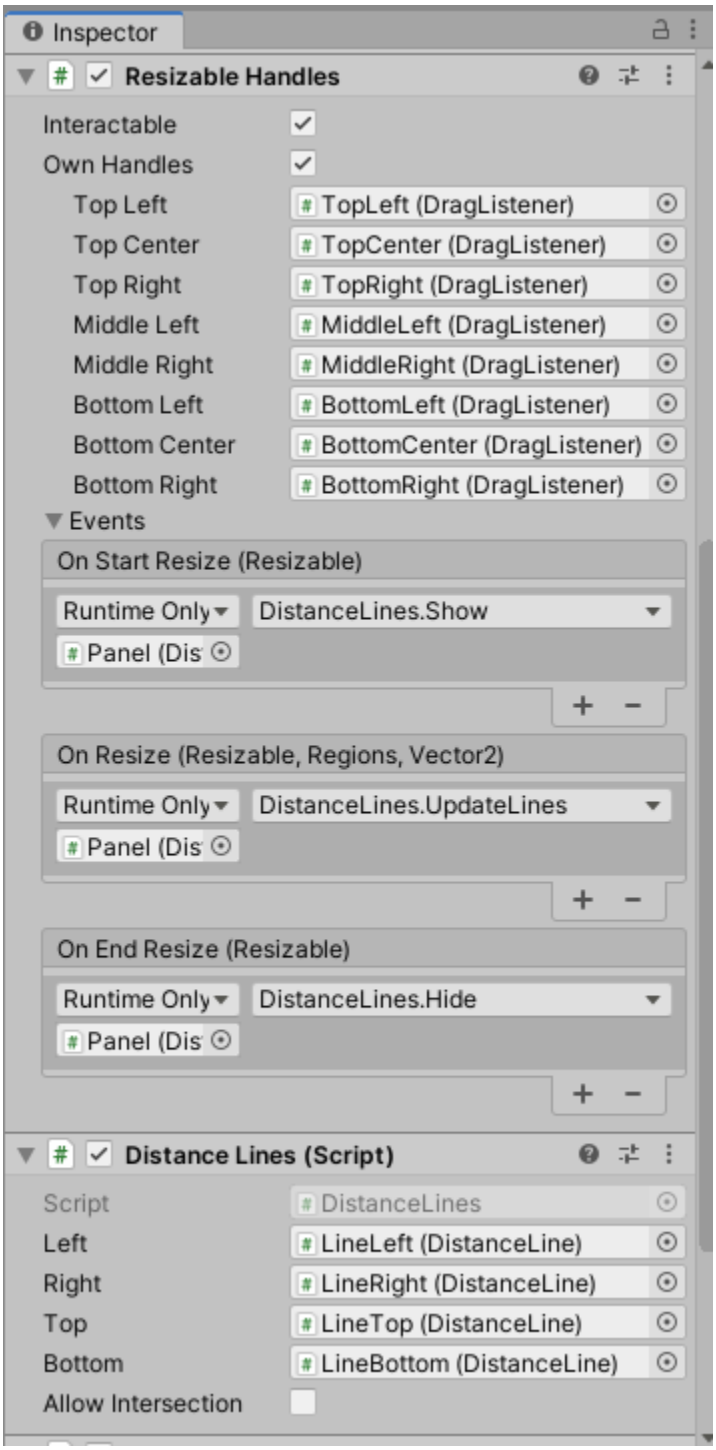
  The minimum space from the border before the turn of the line. Supported only by `Rectangular` lines.

5.15.4 ILineBuilder

Interface to build connectors mesh with a single method:
`int Build(ConnectorBase connector, RectTransform source, ConnectorLine line, VertexHelper vh, int index)`

5.16 Distance Lines

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



5.16.1 Options

- Left `DistanceLine` *optional*

  Line from the left border of the parent.

- Right `DistanceLine` *optional*

  Line from the right border of the parent.

- Top `DistanceLine` *optional*

  Line from the top border of the parent.

- Bottom `DistanceLine` *optional*

  Line from the bottom border of the parent.

- Allow Intersection `bool`

  Allow lines intersection.
  If disabled lines are drawn from parent border to the nearest Target border; otherwise from parent border to the same Target border.

5.17 IOExceptionsView

Handle IO exceptions: catch exceptions and display the following errors.

5.17.1 Options

- ErrorArgument `GameObject`

  Error in case of the `ArgumentNullException` or `ArgumentException`.

- ErrorLongPath `GameObject`

  Error in case of the `PathTooLongException`.

- ErrorUnauthorizedAccess `GameObject`

  Error in case of the `UnauthorizedAccessException`.

- ErrorSecurity `GameObject`

  Error in case of the `SecurityException`.

- ErrorDirectoryNotFound `GameObject`

  Error in case of the `DirectoryNotFoundException`.

- ErrorIO `GameObject`

  Error in case of the `IOException`.

```
var result = ExceptionsView.Execute<ObservableList<TreeNode<FileSystemEntry>>>
↪(FillDrivesList);

protected virtual void FillDrivesList(ObservableList<TreeNode<FileSystemEntry>> list)
{
        foreach (var drive in Directory.GetLogicalDrives())
        {
```

(continues on next page)

(continued from previous page)

```
                var item = new FileSystemEntry(drive, drive, false);
                list.Add(new TreeNode<FileSystemEntry>(item, null));
        }
}
```

5.18 Lightbox

Lightbox is a component used to display overlay image.

5.19 ModalHelper

Creates a fullscreen background for the modal widgets. You can specify background sprite, color, and action on click.

```
modalID = ModalHelper.Open(this, background_sprite, background_color, onclick);
//...
ModalHelper.Close(modalID);
```

5.20 OpenContextMenu

Opens *Context Menu* by clicking on a non-UI gameobject.
Requires `PhysicsRaycaster` on main camera for the 3D objects.
Requires `PhysicsRaycaster2D` on main camera for the 2D objects.

5.20.1 Options

- Interactable `bool`

  Allow users interact with this widget.

- Menu `ContextMenu`

  Menu to open.

- PointerButton `PointerEventData.InputButton`

  The pointer button is used to open the menu.

5.21 ScrollBlock

This component allows to display infinite list of strings.
It is used by *DateScroller, DateTimeScroller, TimeScroller*.

Each item represents an integer index which can be converted to the string representation with the `Value` property.
Item at center always have index `0`, items before it have indices with step `-1`, items after it have indices with step `+1`.

5.21.1 Options

- Value `Func<int, string>`

  Convert integer value to the string representation.

- Increase `Action`

  Increase value by 1.

- Decrease `Action`

  Decrease value by 1.

- AllowIncrease `Func<bool>`

  Check if the value can be increased. Values higher than the current one will not be displayed.

- AllowDecrease `Func<bool>`

  Check if the value can be decreased. Values lower than the current one will not be displayed.

- IsInteractable `Func<bool>`

  Is `ScrollBlock` interactable?

5.21.2 Usage

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

	public class MinutesScroll : MonoBehaviour
	{

		TimeSpan Time = new TimeSpan(12, 10, 20);

		void Start()
		{
			MinutesScrollBlock.Value = Value;
			MinutesScrollBlock.Decrease = DecreaseMinutes;
			MinutesScrollBlock.Increase = IncreaseMinutes;
		}

		string Value(int steps)
		{
			// date used only for convenient conversion of minutes to string
			var date = new DateTime(2000, 1, 2);
			date += IncreaseMinutes(steps) - date.TimeOfDay;

			return date.ToString("mm");
		}

		string IncreaseMinutes(int steps) => Time + new TimeSpan(0, steps, 0);
		string DecreaseMinutes(int steps) => IncreaseMinutes(-steps);
```

(continues on next page)

(continued from previous page)

```
        }
}
```

5.22 ScrollBlockResizer

This component resizes *ScrollBlock* items to display the specified amount of items according to the `ScrollBlock` height.

Intended to be used with *DateScroller, DateTimeScroller, TimeScroller*.

5.22.1 Options

- Blocks `List<ScrollBlock>`

  ScrollBlocks with items to resize.

- Highlight `RectTransform`

  Background highlight for the central ScrollBlock item.

- Visible Items `int`

  Amount of the visible items.

5.23 ScrollButtons

Buttons to scroll `ScrollRect` content on press or hold.

- ScrollButtonLeft `RectTransform`

  Button to scroll on left.

- ScrollButtonRight `RectTransform`

  Button to scroll on right.

- ScrollButtonTop `RectTransform`

  Button to scroll on top.

- ScrollButtonBottom `RectTransform`

  Button to scroll on bottom.

- ScrollSensitivityRateOnClick `float`

  Scroll on press is `ScrollRect.scrollSensitivity` * `Rate` per click.

- ScrollSensitivityRateHold `float`

  Scroll on hold is `ScrollRect.scrollSensitivity` * `Rate` per second.

- Animate `bool`

  Animate scroll on click or scroll immediately if disabled.

- Curve `AnimationCurve`

  Scroll animation curve.

- UnscaledTime `bool`

  Animate using unscaled time.

5.24 Selectable Helper

`Selectable` works only with one `Graphic` component, `SelectableHelper` and `SelectableHelperList` allows to work with additional `Graphic` components.

5.25 Selectable Marker

Shows the specified marker for the currently selected game object. You can use multiple markers at the same time.
For the widgets with multiple `Selectable` components like `Spinner` you may want to use the marker for the root object instead of for each `Selectable`
To do this you need to implement the `ISelectableMarkerTarget` interface or add the `SelectableMarkerTarget` component to the root object of the widget.

5.25.1 Options

- Marker `RectTransform`

  Use an empty object with a nested actual marker.

- Children Only `bool`

  Show markers only for the nested game objects.

- RequireMarker `Predicate<GameObject>`

  Check if a marker should be displayed for the specified object. It has more priority than the `ChildrenOnly` option.

5.26 Slider Scroll

This component allows to change value of the `Slider` widget on scroll.

5.26.1 Options

- ScrollMode `ScrollModes`
  - Ignore
  - `UpIncrease` Increase value on scroll up.
  - `UpDecrease` Decrease value on scroll up.
- Step `float`

  The slider value changes on a step with each scroll event.

5.27 Splitter

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

Note:

You can use `minWidth` and `minHeight` properties of the `LayoutElement` to set minimal size of the objects.
And use the `SplitterMaxSize` component to set the maximum size of the objects. Total maximum size should be less than the total size of the objects.

5.27.1 Options

- Interactable `bool`

  Allow users to interact with the splitter.

- Type `SplitterType`
  - `Horizontal`: change heights of the game objects.
  - `Vertical`: change widths of the game objects.
- Update RectTransform `bool`

  Change RectTransform size of the left and right game objects.

- Update LayoutElement `bool`

  Change LayoutElement size of the left and right game objects.

- Drag Button `PointerEventData.InputButton`

  The button that should be pressed to process the drag event.

- Integer Size `bool`

  If enabled size is rounded to the integer number.

- Mode `SplitterMode`
  - `Auto`: use previous and next siblings in hierarchy.
  - `Manual`: use specified targets to resize.
- Previous Object `RectTransform`

  Left (or top) object to resize.

- Next Object `RectTransform`

  Right (or bottom) object to resize.

- Cursors `Cursors`

  Cursors to use.

5.27.2 Events

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

5.28 StackViewAnimations

This component provides animations for the *StackView*.

5.29 Options

- Push New Settings *Settings*

  Animation settings for a new view when used *StackView.Push()* method.

- Push Current Settings *Settings*

  Animation settings for a current view when used *StackView.Push()* method.

- Pop Current Settings *Settings*

  Animation settings for a current view when used *StackView.Pop()* method.

- Pop Previous Settings *Settings*

  Animation settings for a previous view (it became a current view) when used *StackView.Pop()* method.

- Replace New Settings *Settings*

  Animation settings for a new view when used *StackView.Replace()* method.

- Replace Current Settings *Settings*

  Animation settings for a current view when used *StackView.Replace()* method.

5.30 Settings

- Curve *AnimationCurve*
- Axis *Axis*
  - Horizontal
  - Vertical

5.31 Switch Group

Same as Toggle Group, but for the Switch widget.

5.31.1 Options

- Allow Switch Off `bool`

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

CHAPTER

SIX

HELPERS

6.1 Async Helpers

Using async will simplify the code and helps get rid of callbacks, especially useful in case of multiple nested callbacks.

For this reason, helper scripts have been added to make it easier to implement async support for the own scripts or widgets.

The target script should implement an `IAwaitable<TResult>` or `IAwaitable` interface.

Script example:

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

   public class ConfirmExample : MonoBehaviour, IAwaitable<bool>
   {
      [SerializeField]
      protected Text Message;

      [SerializeField]
      protected Button ButtonOk;

      [SerializeField]
      protected Button ButtonCancel;

      event Action<bool> EvOnComplete;

      public event Action<bool> OnComplete
      {
         add => EvOnComplete += value;
         remove => EvOnComplete -= value;
      }

      public Awaiter<bool> GetAwaiter() => new Awaiter<bool>(this);
```

(continues on next page)

(continued from previous page)

```
        protected virtual void Start() => AddListeners();

        protected virtual void OnDestroy()
        {
            RemoveListeners();
            Cancel();
        }

        void AddListeners()
        {
            ButtonOk.onClick.AddListener(Confirm);
            ButtonCancel.onClick.AddListener(Cancel);
        }

        void RemoveListeners()
        {
            ButtonOk.onClick.RemoveListener(Confirm);
            ButtonCancel.onClick.RemoveListener(Cancel);
        }

        public void Confirm() => Complete(true);

        public void Cancel() => Complete(false);

        void Complete(bool result)
        {
            gameObject.SetActive(false);
            EvOnComplete?.Invoke(result);
        }

        public ConfirmExample Open(string message)
        {
            Message.text = message;
            gameObject.SetActive(true);
            EventSystem.current.SetSelectedGameObject(ButtonOk.gameObject);

            return this;
        }
    }
}
```

Using:

```
namespace UIWidgets.Examples
{
    using UnityEngine;

    public class TestConfirm : MonoBehaviour
    {
        [SerializeField]
        public ConfirmExample Confirm;
```

(continues on next page)

(continued from previous page)

```
    public async void Test()
    {
        if (await Confirm.Open("Quit?"))
        {
            Application.Quit();
        }
    }
}
```

6.2 DataLoader

Helper class to simplify loading and caching data. Intended to use with ListView items to avoid problems with reusable instances.

6.3 Usage

```
public class ListViewImage : ListViewCustom<ImageItem, string>
{
        public DataLoader<string, Sprite> Loader;

        protected override void InitOnce()
        {
                base.InitOnce();

                Loader = new DataLoader<string, Sprite>(async request =>
                {
                        var web_request = UnityWebRequest.Get(request);

                        await web_request.SendWebRequest();

                        if (web_request.result != UnityWebRequest.Result.Success)
                        {
                                Debug.LogWarning(string.Format("UnityWebRequest Failed.␣
↪URI: {0}; Result: {1}; Code: {2}; Content: {3}",
                                        request, web_request.result, web_request.
↪responseCode, web_request.downloadHandler.text));
                                return new Tuple<bool, Sprite>(false, null);
                        }

                        var texture = DownloadHandlerTexture.GetContent(web_request);
                        var sprite = Sprite.Create(texture, new Rect(0, 0, texture.width,
↪ texture.height), new Vector2(0.5f, 0.5f));
                        return new Tuple<bool, Sprite>(true, sprite);
                });
        }
}
```

(continues on next page)

(continued from previous page)

```
public class ImageItem : ListViewItem, IViewData<string>
{
        public Image Image;

        protected string Item;

        protected DataLoader<string, Sprite> Loader => (Owner as ListViewImage).Loader;

        public void SetData(string item)
        {
                Item = item;
                UpdateView(item);
        }

        async void UpdateView(string url)
        {
                // reset image or show "loading" placeholder
                Image.sprite = null;

                // load sprite
                var (success, sprite) = await Loader.GetAsync(url);

                // if sprite was not loaded
                if (!success)
                {
                        return;
                }

                // check if item is still same (item will be different if ImageItem␣
↪instance was recycled)
                if (Item != url)
                {
                        // do nothing
                        return;
                }

                Image.sprite = sprite;
        }
```

6.4 HierarchyPosition

It is used to save and restore the game object's position in the hierarchy, for example when the object is moved to the bottom of the root canvas to display it on top of all other objects.

```
HierarchyPosition position;

void Show()
{
        var canvas = UtilitiesUI.FindTopmostCanvas(transform);
        if (canvas != null)
```

(continues on next page)

(continued from previous page)

```
        {
                position = HierarchyPosition.SetParent(transform, canvas);
        }

        gameObject.SetActive(true);
}

void Hide()
{
        Position.Restore();

        gameObject.SetActive(false);
}
```

6.5 Stable Sort

This class provides stable sort for the `IList<T>`, that is, if two elements are equal, their order will be preserved. A default `List<T>.Sort()` implementation performs an unstable sort: order might not preserved for elements that are equal.

```
StableSort.Sort(list, (a, b) => a.Field.CompareTo(b.Field), reverse: false);
```

CHAPTER

SEVEN

EFFECTS

7.1 Border Effect

Shader effect. Add the `BorderEffect` component to the game object with the `Graphic` component.

Note: The `Mask` component should be placed before this component.

7.1.1 Options

- Border Color `Color`

  Color of the border.

- Transparent Background `bool`

  Change color of the `Graphic` component to transparent.

- Horizontal Borders `Vector2`

  Left (x) and right (y) border width in pixels.

- Vertical Borders `Vector2`

  Top (x) and bottom (y) border width in pixels.

7.2 Flare Effect

Shader effect. Add the `FlareEffect` component to the game object with the `Graphic` component.

7.2.1 Options

- Global Space `bool`

  If enabled flare is displayed in global space: different `Graphic` components will have the same flare if the settings are the same.
  Works only in Render Mode = `Screen Space - Overlay`.

- Color `Color`

  Color of the flare.

- Size `float`

Size of the flare in range [0..1].

- Speed `float`

  Speed of the flare relative to the `Graphic` size (or `Canvas` if global space used): how much times it will move from left to right in one second.

- Start Position `float`

  Start position of the flare in range [0..1].

- Angle `float`

  Flare angle in range [-1..1].

7.3 Grayscale Effect

Display `Graphic` at grayscale. Shader effect.

7.3.1 Options

- Rate `ColorRate`

  Color multiplier to make `Graphic` grayscale.

  - Red `float`
  - Green `float`
  - Blue `float`

- Grayscale Enabled `bool`

  Enable/disable effect.

7.4 Lines Drawer

Draw straight lines on X or Y axis.

Note: The `Mask` component should be placed before this component.

7.4.1 Options

- Line Color `Color`

  Line color.

- Line Thickness `float`

  Line thickness.

- Transparent Background `bool`

  Change color of the `Graphic` component to transparent.

- LinesX `ObservableList<float>`

Position on X axis where vertical line should be drawn in range [0..width], 0 at left.

- LinesY `ObservableList<float>`

  Position on Y axis where horizontal line should be drawn in range [0..height], 0 at bottom.

7.5 Ring Effect

Draw ring or circle. Shader effect. Add the `RingEffect` component to the game object with the `Graphic` component.

Note: The `Mask` component should be placed before this component.

7.5.1 Options

- Ring Color `Color`

  Color of the ring.

- Thickness `float`

  Ring thickness.

- Padding `float`

  Padding from border.

- Transparent Background `bool`

  Change color of the `Graphic` component to transparent.

7.6 Ripple Effect

Draw ripples on the click position. Maximum 10 ripples per game object. Shader effect. Add the `RippleEffect` component to the game object with the `Graphic` component.

7.6.1 Options

- Start Color `Color`

  Initial color of the ripple.

- End Color `Color`

  End color of the ripple.

- Speed `float`

  Growth speed of the ripple.

- Max Size `float`

  Maximum size of the ripple in range [0..1].

7.7 Rounded Corners

Shader effect. Add the `RoundedCorners` or `RoundedCornersX4` (each corner can have its radius) component to the game object with the `Graphic` component.

Note: The `Mask` component should be placed before this component.

7.7.1 Options

- Radius `float` (`BorderRadius` in case of `RoundedCornersX4`)

  Corners radius.

- Border Width `float`

  Border width (cannot be more than Radius).

- Border Color `Color`

  Color of the border.

7.8 Snap Grid Drawer

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

Note: Requires `SnapGrid` or `SnapLines` components.

7.8.1 Options

- Line Color `Color`

  Line color.

- Line Thickness `float`

  Line thickness.

- Transparent Background `bool`

  Change color of the `Graphic` component to transparent.

- Include Borders `bool`

  Draw borders if borders enabled in `SnapGrid` or `SnapLines` components.

7.9 Tsunami Effect

RectTransform size is changed from `MinSize` to `MaxSize` depending of distance from the game object to the pointer.

7.9.1 Options

- MinSize `Vector2`

  Minimal size of the component.

- MaxSize `Vector2`

  Maximum size of the component.

- Distance `float`

  Effect distance.

CHAPTER

EIGHT

SHADERS

8.1 Gradient Shaders

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

- UIGradientHLineHSV

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

- UIGradientHLineRGB

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

- UIGradientVLineHSV

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

- UIGradientVLineRGB

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

- UIGradientPlaneHSV

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

- UIGradientPlaneRGB

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

CHAPTER

NINE

UI SCALING

UI can have missing line or lines with different width. It can be caused by those reasons:

- Game window Scale is not an integer.

  Make sure that the Game window Scale is 1x (2x in case of HighDPI display).

  In any case, the Scale should be an integer, otherwise, lines will have different thickness.

  For example, it can be downscaled from 1920x1080 (game window render size) to 1280x720 (if the scale is 0.66): then some 1-pixel lines will be lost, but some still be visible, because the visible screen size is not enough to display them all.

  Similar will if the game render size is 1280x720 and the scale is 1.5: all lines will be visible, but some lines still be thicker than others.

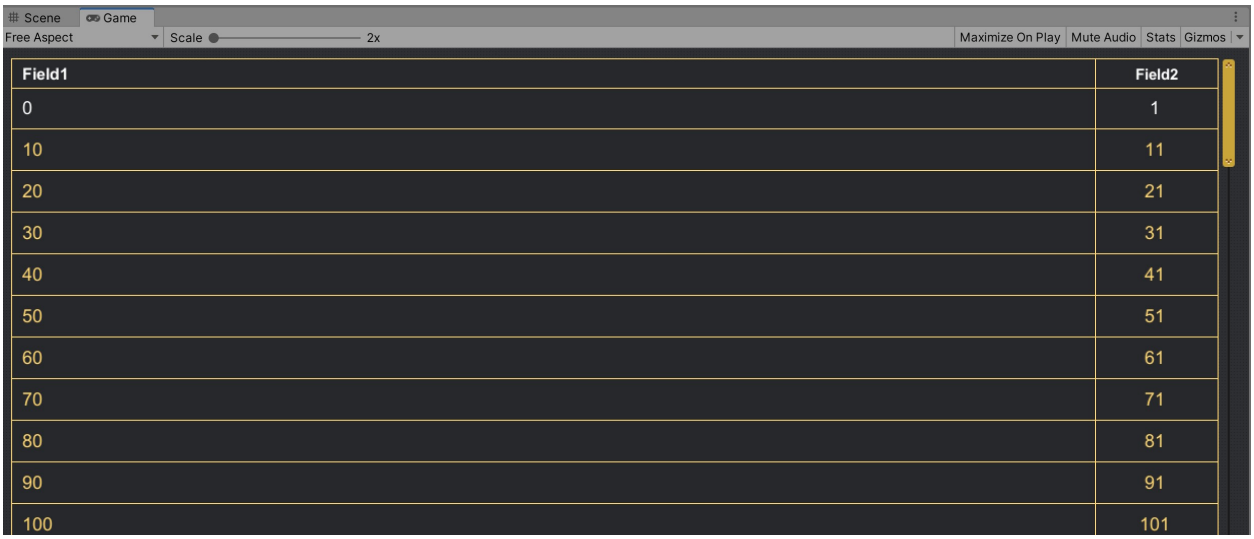


Fig. 1: Scale = 2x

- Game window render size does not match with Canvas Scaler settings

  UI rendered in size specified by Canvas Scaler settings, then scaled to match the Game window render size and then multiplied on Game window Scale.

  Check Canvas Scaler settings:

  - *UI Scale Mode* = *Constant Pixel Size* then should be no problem is the scale is an integer

  - *UI Scale Mode* = *Scale With Screen Size* then make sure Reference Resolution is equal to Game window render size or make integer multiplier.

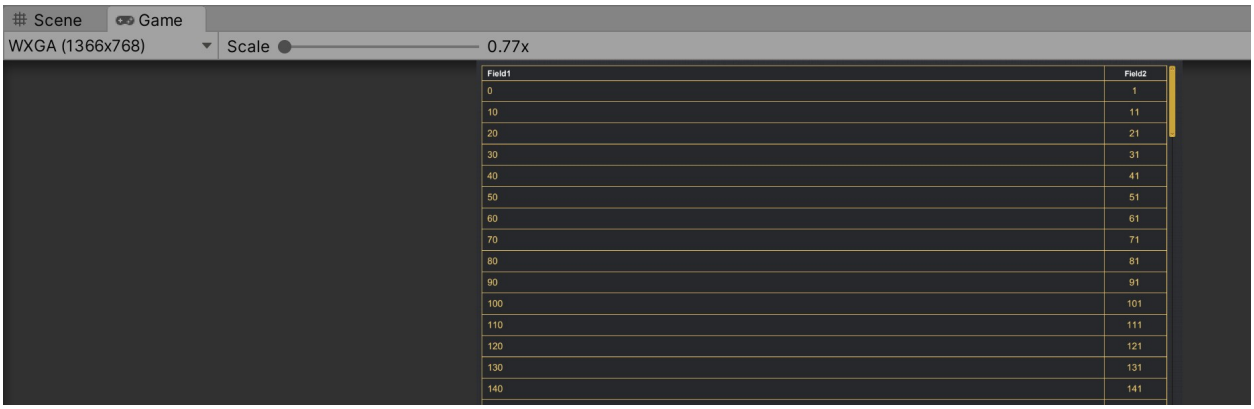


Fig. 2: Scale = 0.77x

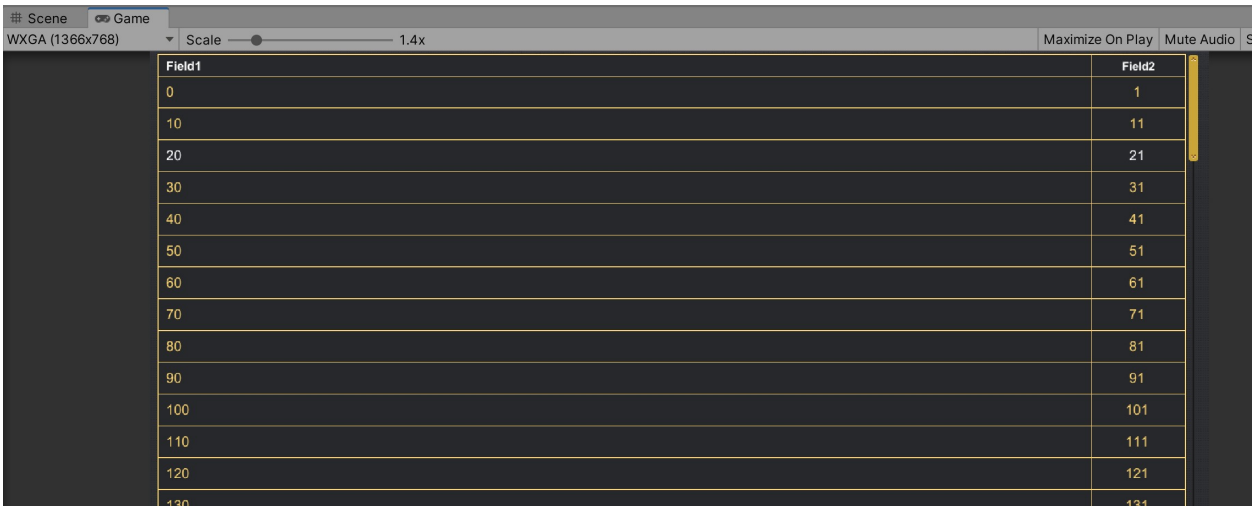


Fig. 3: Scale = 1.7x

For example Reference Resolution = 1920x1080 and Game window render size = 3840x2160 -> multiplier = 2 is okay

If Reference Resolution = 1920x1080 and Game window render size = 2560x1440 -> multiplier = 1.33 will be problem with lines thickness.

- *UI Scale Mode* = *Constant Physical Size*, similar to *Scale With Screen Size* but relies on display DPI, so it scale can be different for different devices.

The Game window render size can be changed with a dropdown left from the Scale (Free Aspect on the screenshot).

CHAPTER

TEN

CUSTOMIZATION

10.1 Styles (Legacy)

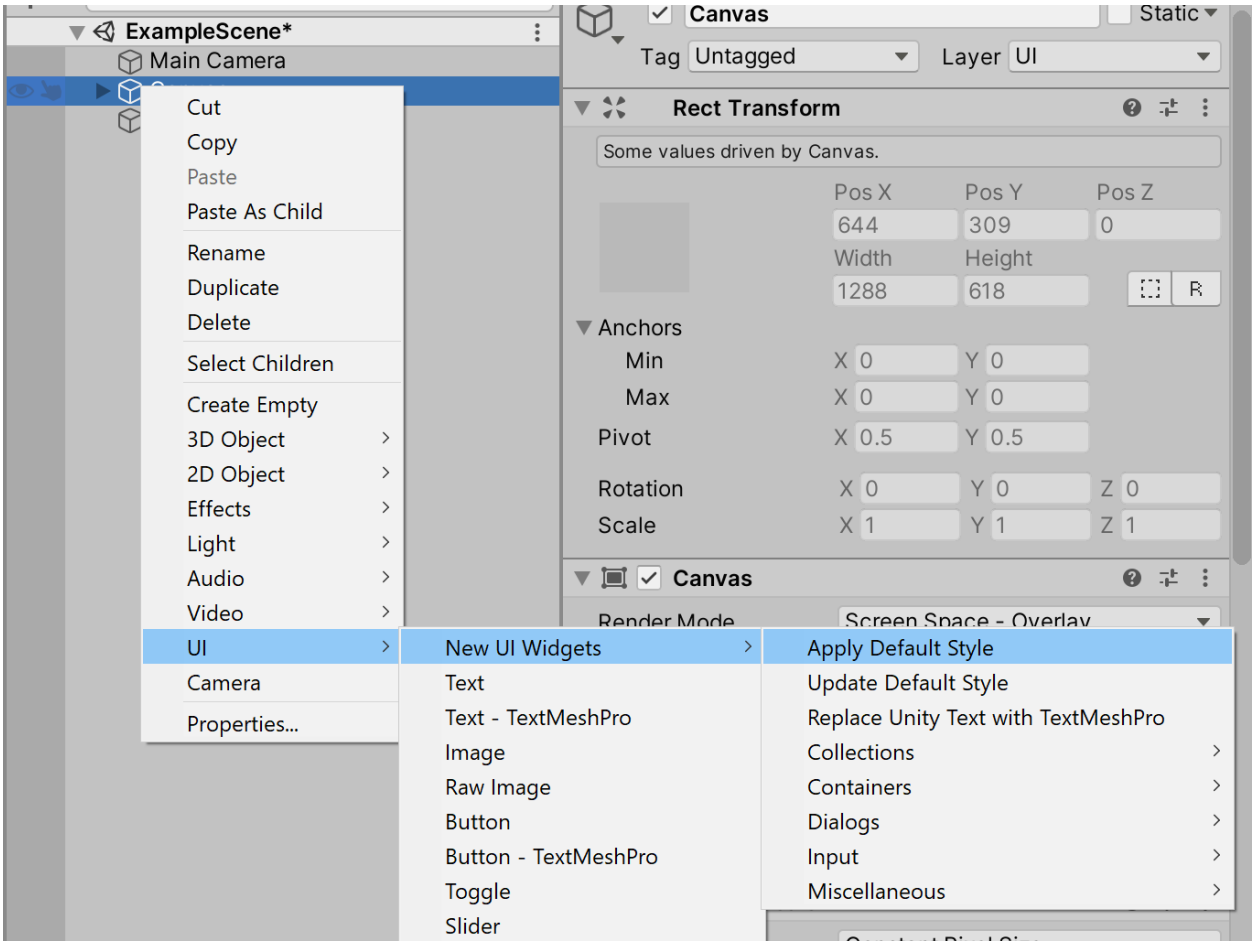
> **Warning:** Styles are obsolete and no longer supported. They are replaced with *UI Themes*.

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

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

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

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



You can change widgets settings and then save them to the style with *UI / New UI Widgets / Update Default Style*.

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

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

Note: For the style support for the nested widgets (for example, Switch or Spinner in the ListView.DefaultItem), you should add the `StyleSupportAny` component to the gameobject of the parent widget and specify nested widgets at the `Objects` field.

10.1.1 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
```

(continues on next page)

(continued from previous page)

```
{
   public virtual bool SetStyle(Style style)
   {
      style.Collections.MainBackground.ApplyTo(GetComponent<Image>());

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

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

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

Note: Widgets created by *Widgets Generator* already have style support.

10.2 UI Themes

UI Themes is a tool for customizing the appearance of widgets and centralized customization management.
Easy to integrate and use with already existing interfaces.
Full Documentation

CHAPTER

ELEVEN

INTEGRATION

11.1 Assembly Definitions

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

Why no assembly definitions by default:

- changes in the *.asmdef* files are lost with the package update
- supported third-party packages do not use assembly definitions, so they cannot be referenced to be used in a separate assembly
- version defines works only for Unity packages, so only direct references to assembly definitions are available

11.1.1 Recommended Settings

- asmdef for the runtime should be created in the *New UI Widgets* folder
- asmdef for the editor should be created in the *New UI Widgets / Editor* folder
- references to `TextMeshPro` and `InputSystem` should be added if you use them

11.2 Cursor

- Cursors

  A scriptable object that contains a list of different cursors types.
  Asset can be created with the *Context menu / Create / New UI Widgets / Cursors*.

- UICursor

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

- CursorDPISelector

  This component selects the most appropriate `Cursors` asset by `Screen.dpi` from the available cursors list and sets it as default cursors (`UICursor.Cursors`).
  Components like Resizable have the `Cursors` field, so they can have custom cursors to use instead of the default one.

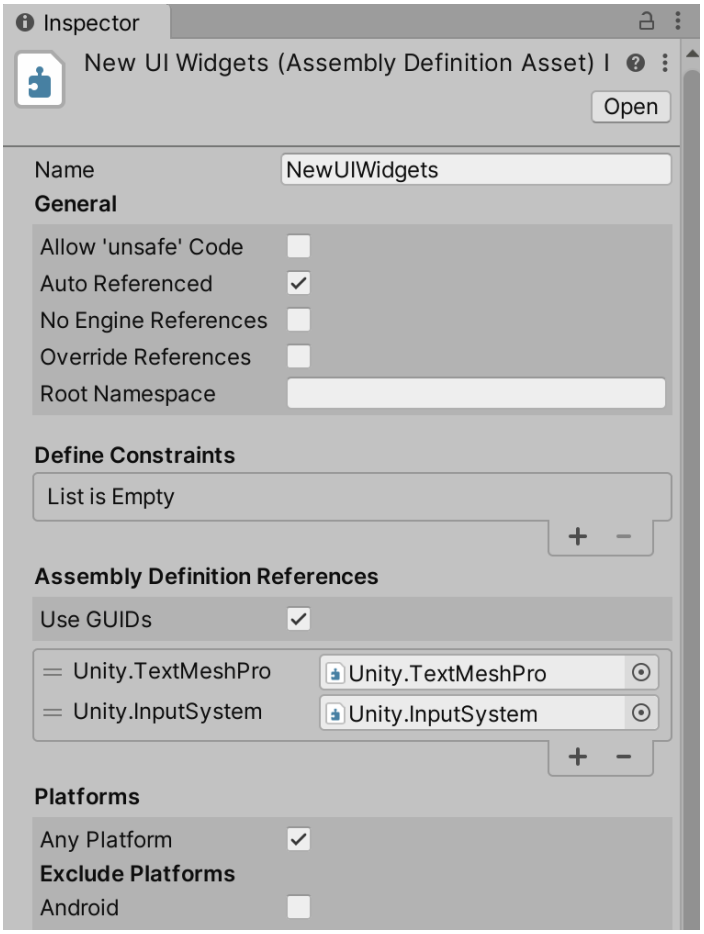


Fig. 1: Runtime

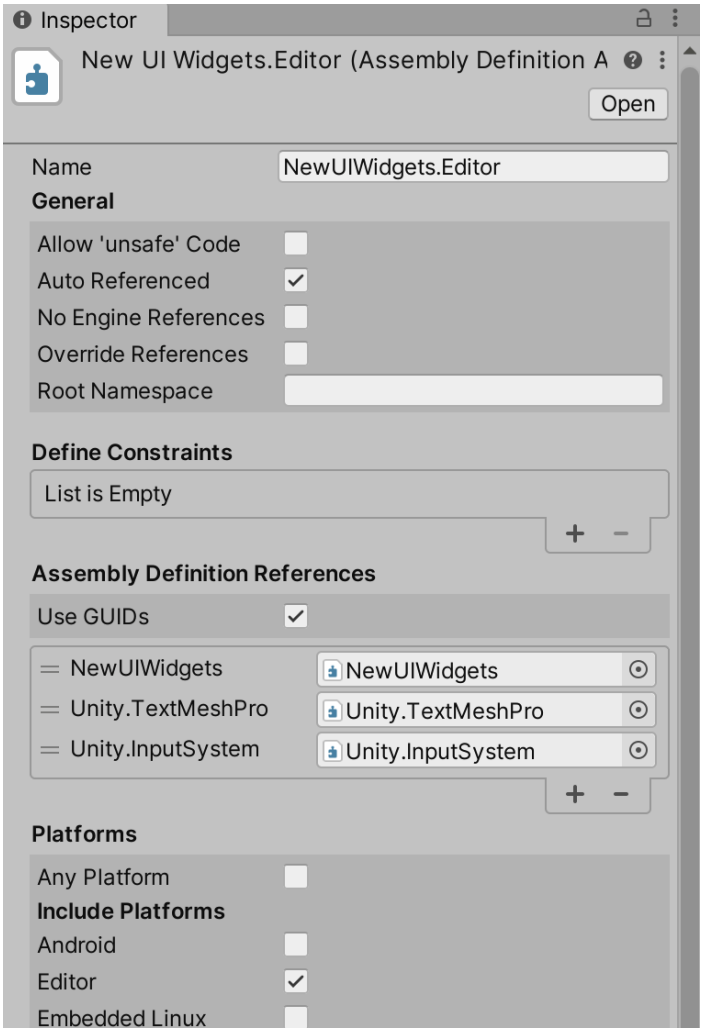

Inspector
New UI Widgets.Editor (Assembly Definition A
Open
Name
NewUIWidgets.Editor
General
Allow 'unsafe' Code
Auto Referenced
No Engine References
Override References
Root Namespace
Define Constraints
List is Empty
Assembly Definition References
Use GUIDs
NewUIWidgets
NewUIWidgets
Unity.TextMeshPro
Unity.TextMeshPro
Unity.InputSystem
Unity.InputSystem
Platforms
Any Platform
Include Platforms
Android
Editor
Embedded Linux

Fig. 2: Editor

11.2.1 Cursors Fields

- Default `Cursors.Cursor`

  Default cursor.

- Allowed `Cursors.Cursor`

  Cursor for the allowed actions.

- Denied `Cursors.Cursor`

  Cursor for the not allowed actions

- NorthSouthArrow `Cursors.Cursor`

  North <-> South arrow.

- EastWestArrow `Cursors.Cursor`

  East <-> West arrow.

- NorthEastSouthWestArrow `Cursors.Cursor`

  NorthEast <-> SouthWest arrow.

- NorthWestSouthEastArrow `Cursors.Cursor`

  NorthWest <-> SouthEast arrow.

- NorthWestRotateArrow `Cursors.Cursor`

  North <-> West arrow.

- NorthEastRotateArrow `Cursors.Cursor`

  North <-> East arrow.

- SouthWestRotateArrow `Cursors.Cursor`

  South <-> West arrow.

- SouthEastRotateArrow `Cursors.Cursor`

  South <-> East arrow.

11.2.2 Cursors.Cursor Fields

- Texture `Texture2D`

  Cursor texture.

- Hotspot `Vector2`

  Cursor hot spot.

11.2.3 UICursor Static Fields

- Cursors `Cursors`

  Different default cursors.

- CanSet `Func<Component, bool>`

  Is can the specified component set the cursor?
  `true` if cursor does not have an owner or the owner is the same.

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

  Set the cursor and current owner.
  The cursor will be changed only if `CanSet(owner)` returns `true`.

- Reset `Action<Component>`

  Reset cursor and its owner to the default.

11.3 Input

The common way to interact with the interface is through clicks, but the same interaction with other objects occurs through clicks.
And most often problem is when the same click is processed by UI and another object simultaneously.

There are a couple of solutions to this problem.

If you are using the `OnMouseX()` method then you should use `EventSystem.current.IsPointerOverGameObject()` to check if a click should be processed.

```
protected void OnMouseDown()
{
        if (EventSystem.current.IsPointerOverGameObject())
        {
                Debug.Log(Time.frameCount + " ignore click because over UI");
                return;
        }

        Debug.Log(Time.frameCount + " OnMouseDown");
}
```

But a better way is to implement the `IPointerClickHandler` interface. In this case, you do not need additional checks, but you should add the `PhysicsRaycaster` component to the camera.

```
public class Test3DClick : MonoBehaviour, IPointerClickHandler
{
    public void OnPointerClick(PointerEventData eventData)
    {
        Debug.Log(Time.frameCount + " OnPointerClick");
    }
}
```

The second method is recommended because it uses the same EventSystem as the UI objects.

Table 1: MonoBehaviour Methods and Corresponding EventSystem Interfaces

MonoBehaviour Method	Interface
OnMouseDown	IPointerDownHandler
OnMouseDrag	IBeginDragHandler, IDragHandler, IEndDragHandler
OnMouseEnter	IPointerEnterHandler
OnMouseExit	IPointerExitHandler
OnMouseOver	IPointerMoveHandler
OnMouseUp	IPointerUpHandler
OnMouseUpAsButton	IPointerClickHandler

11.4 Localization

Most widgets have localization support, exceptions are:

- AutocompleteString
- ComboboxString
- ListViewString

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

Example for the **I2 Localization**:

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

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

   return result;
}

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

11.4.1 Dialog, Popup Localization

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

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

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

   var instance = DialogTemplate.Clone();
   instance.DialogInfo.LocalizationSupport = true;
   instance.Show(
      buttons: actions,
      focusButton: "Close",
      modal: false,
      onCancel: DialogClose);
   instance.SetInfo("Welcome, {0}", new object[] { "username", }, "Value 1: {0}\nValue␣
↪2: {1}", new object[] { "argument 1", "argument 2" });
}

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

11.4.2 Notify Localization

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

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

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

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

11.4.3 Generated Widgets

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

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

   public string Name;
}
```

11.5 String Comparison and Culture

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

- UtilitiesCompare.Culture `CultureInfo`

  Culture used to compare strings, by default used `CultureInfo.InvariantCulture`.

- UtilitiesCompare.OptionsCaseSensitive `CompareOptions`

  Options to compare strings with case sensitive.

- UtilitiesCompare.OptionsCaseIgnore `CompareOptions`

  Options to compare strings with case ignore.

11.6 Timer and Animations

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

You can also specify your timer instead of the default one. To do this, you need to create a type with the implementation of the `ITime` interface and set it as `WidgetsTime.Instance`:

11.7 Unity Update Methods Replacement

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

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

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

11.7.1 Interfaces to Replace Unity Update Methods

- `IUpdatable` replace `Update()` method

  Methods:

  - `RunUpdate()`

- `ILateUpdatable` replace `LateUpdate()` method

  Methods:

  - `RunLateUpdate()`

- `IFixedUpdatable` replace `FixedUpdate()` method

  Methods:

  - `RunFixedUpdate()`

CHAPTER

TWELVE

SUPPORTED PACKAGES

12.1 Data Bind for Unity Support

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

After enabling support:

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

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

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

Note: If you enabled *Assembly Definitions* then you need to create an assembly definition for the **Data Bind** if not exist and specify it as a reference in the `UIWidgets.asmdef`.

12.2 I2 Localization Support

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

Localization Support Details

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

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

Note: If you enabled *Assembly Definitions* then you need to create an assembly definition for the **I2 Localization** if not exist and specify it as a reference in the `UIWidgets.asmdef`.

12.3 TextMeshPro Support

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

After enabling support:

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

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

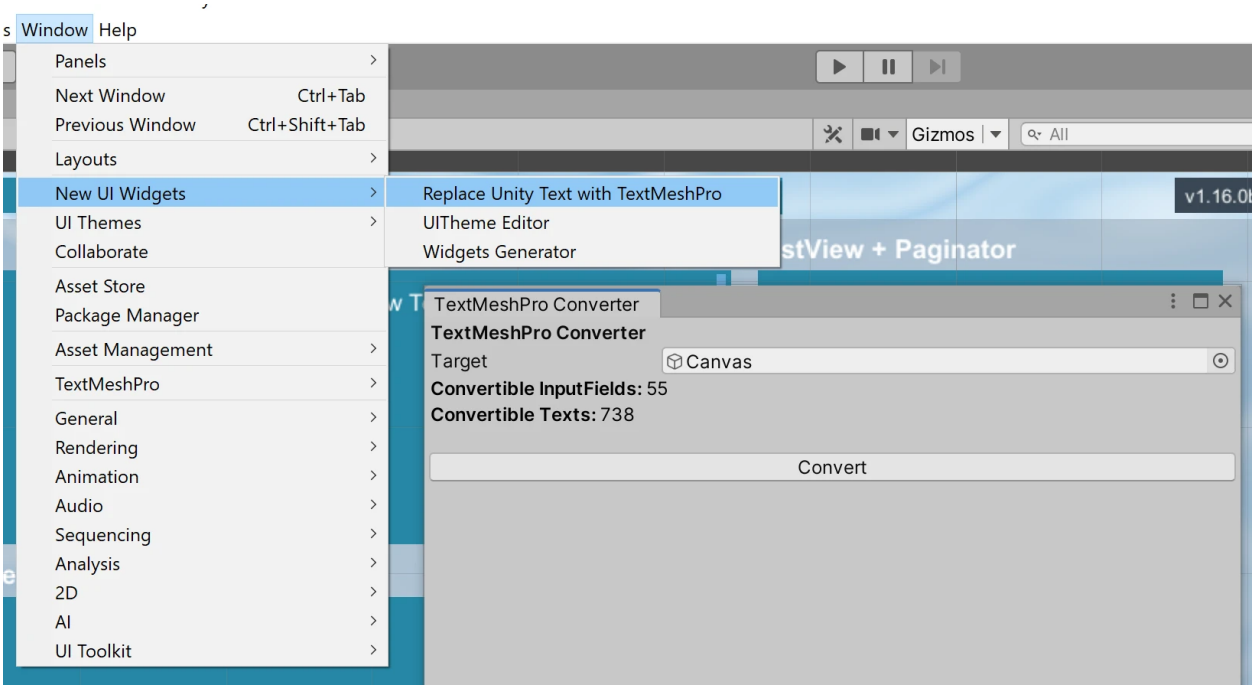
Note: Support is enabled only to installed platforms. Platforms that were added after it requires enabling support again.

12.3.1 Details

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

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

12.4 TextMeshPro Converter



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

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

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

Limitations:

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

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

[SerializeField]
Graphic SecondName; // can be converted
```

(continues on next page)

(continued from previous page)

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

Solutions:

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

12.4.1 Modify Code to Adapters

Original script:

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

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

Modification:

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

   [SerializeField]
   TextAdapter NameAdapter;

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

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

   #if UNITY_EDITOR
   protected virtual void OnValidate()
   {
      Upgrade();
   }
   #endif
}
```

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

12.5 Unity Localization Support

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

Localization Support Details

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

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

CHAPTER

THIRTEEN

CONTROLLERS SUPPORT

13.1 Collections

Widget	Keyboard	Mouse	Touch	Gamepad
AutocompleteCombobox	Yes	Only to select values, InputField requires keyboard		
AutoCombobox	Yes	Only to select values, InputField requires keyboard		
Combobox	Yes	Yes	Yes	Yes
ComboboxInputField	Yes	Only to select values, InputField requires keyboard		
ListView	Yes	Yes	Yes	Yes
TreeView	Yes	Yes	Yes	Yes

13.2 Containers

Widget	Keyboard	Mouse	Touch	Gamepad
Accordion	Yes	Yes	Yes	Yes
Slider	Yes	Yes	Yes	Yes
Tabs	Yes	Yes	Yes	Yes

13.3 Controls

Widget	Keyboard	Mouse	Touch	Gamepad
ContextMenu	Yes	Yes	No	No
Paginator	Yes	Yes	Yes	Yes
Sidebar	No	Yes	Yes	No
SplitButton	Yes	Yes	Yes	Yes

13.4 Dialogs

Widget	Keyboard	Mouse	Touch	Gamepad
Dialog	Yes	Yes	Yes	Yes
Picker	Yes	Yes	Yes	Yes
Popup	Yes	Yes	Yes	Yes

13.5 Input

Widget	Keyboard	Mouse	Touch	Gamepad
Autocomplete	Yes	Only to select values, InputField requires keyboard		
Calendar	Yes	Yes	Yes	Yes
CenteredSlider	Yes	Yes	Yes	Yes
CircularSlider	No	Yes	Yes	No
ColorPicker	Yes	Yes (InputField requires keyboard)		
ColorPickerRange	Yes	Yes	Yes	Yes
DateTime	Yes	Yes (InputField requires keyboard)		
DateScroller	No	Yes	Yes	No
DateTimeScroller	No	Yes	Yes	No
RangeSlider	Yes	Yes	Yes	Yes
Rating	Yes	Yes	Yes	Yes
Scale	Not interactable			
Spinner	Yes	Yes (InputField required keyboard)		
Switch	Yes	Yes	Yes	Yes
Time	Yes	Yes (InputField required keyboard)		
TimeAnalog	No	Yes	Yes	No
TimeScroller	No	Yes	Yes	No

13.6 Miscellaneous

Widget	Keyboard	Mouse	Touch	Gamepad
AudioPlayer	Yes	Yes	Yes	Yes
Loading Animation	Not interactable			
Progressbar	Not interactable			
Tooltip	Not interactable			

13.7 Components

Component	Keyboard	Mouse	Touch	Gamepad
TableHeader	Sort only	Yes	Yes	Sort only
Drag-and-Drop	No	Yes	Yes	No
Draggable	No	Yes	Yes	No
Groupable	No	Yes	Yes	No
Object Sliding	No	Yes	Yes	No
Pinchable	No	Yes	Yes	No
Resizable	No	Yes	Yes	No
Rotatable	No	Yes	Yes	No
ScrollBlock	No	Yes	Yes	No
Splitter	No	Yes	Yes	No
Lightbox	No	Yes	Yes	No

CHAPTER

FOURTEEN

KNOWN PROBLEMS

14.1 Missing References or Scripts

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

14.2 TextMeshPro Support are Disabled After the Platform Switch

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

14.3 Newly Created Widgets are White

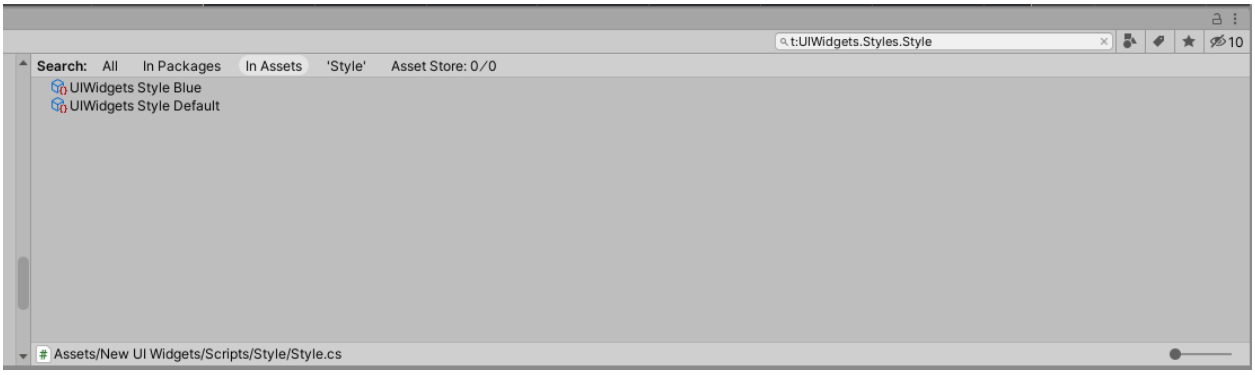
It happens because of the empty style used as default and it automatically applied to newly created widgets.
Please open *New UI Widgets/Styles/UIWidgets Style Default* and check it settings (it should not be all white color or null), and set it as default.
If UIWidgets Style Default values are all white color or null, then try to import package again, sometimes import works incorrectly.

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

14.4 ListView Item Highlight or Selection Goes to Next Items Automatically

Reason are navigation events raised by a gamepad or joystick, sometimes unintentionally because of sticks drift.
Solutions for the different input modules:

- Standalone Input Module:
  - open *Project Settings / Input Manager*



  - find **Horizontal** and **Vertical** records with `Type` = `Joystick Axis` (there are two of each, another one for keyboard)
  - rename those **Horizontal** and **Vertical** records to names not used by `Standalone Input Module`
- Input System UI Input Module:
  - open *Project Settings / Input System Package*
  - add keyboard, mouse, and other required devices to the `Supported Devices`

14.5 Input System Limitations

Input System is supported, but its limitations are still applied.

Limitations effects:

> After enabling, the UI will not react to a pointer's position until the position is changed.

It will affect ListView, TileView, Table and others: items under the cursor will not be properly highlighted when scrolling.

> The new input system cannot yet feed text input into uGUI and TextMesh Pro input field components. This means that text input ATM is still picked up directly and internally from the Unity native runtime.

It will affect all widgets that use InputField like Autocomplete, AutoCombobox, Spinner, etc…

14.6 Dragged Objects Lagged Behind the Cursor

This happens because the cursor is rendered by the system (hardware cursor). But the game window displays frames with some lag because of enabled VSync and QualitySettings.maxQueuedFrames (frames buffer). So you see the actual cursor position and game screen that match the cursor position 1-3 frames before.

Solutions: use software cursor, it will have input lag, but there will be no difference in cursor and draggable object positions.

Add such a script at the start to change the cursor to software mode. You need a cursor image to do this, you can copy and edit "cursor_arrow_minus.png" to remove the minus sign.

```
using UnityEngine;

public class SoftwareCursor : MonoBehaviour
```

(continues on next page)

(continued from previous page)

```
{
    [SerializeField]
    Texture2D cursor;

    [SerializeField]
    Vector2 cursorHotspot = Vector2.zero;

    public void Start()
    {
        Cursor.SetCursor(cursor, cursorHotspot, CursorMode.ForceSoftware);
    }
}
```

14.7 Errors After New Input System Enabled

There can be errors if *Input System* was enabled in *Player Settings* after the package was installed.

Errors will be like: `Assets\New UI Widgets\Scripts\Utilities\CompatibilityInput.cs(8,20): error CS0234: The type or namespace name 'InputSystem' does not exist in the namespace 'UnityEngine' (are you missing an assembly reference?)`

Solution:

1. Open *Package Manager* and install the *Input System* package if it is not installed (yes, Unity allows to enable *Input System* even if the package is not installed);
2. Find assembly definitions in the *New UI Widgets* folder (type `t:asmdef` in the project window search);
3. Add a reference to the `Unity.InputSystem` assembly and press *Apply* for each *UIWidgets* assembly.

Warning: Do not select all *UIWidgets* assemblies to add a reference to all of them simultaneously because Unity will keep only the first two original references and remove all others.

CHAPTER

FIFTEEN

SUPPORT

You can ask me questions at:

- Unity Discussions: https://discussions.unity.com/t/new-ui-widgets/567076
- Email: support@ilih.name

CHAPTER

SIXTEEN

CHANGELOG

16.1 Release 1.18.6

- Unity 6.3 support
- fixed error when an empty TextMeshPro package version 5.0.0 is installed
- fixed conversion from ColorHSV to Color (completely black and white colors were impossible)
- ListViewEnum: now you can specify a list of enums to use with "Wrapper = ListView.UseEnum(new [] { EnumType.Value0, EnumType.Value3, });"
- Paginator: fixed the problem with Forced Position = Center on the first and last page

16.2 Release 1.18.5

- fixed error on first time installation ("UITHEMES_INSTALLED" is not added to the "Scripting Define Symbols")

16.3 Release 1.18.4

- ColorPicker: fixed a bug related to linear color space handling
- ListView: fixed a bug where an incorrect number of items were displayed when using large margins
- ListView: fixed a bug in Variable Size mode when the data type does not implement IComparable
- ListView: fixed a bug in Variable Size mode where rows or columns were miscalculated
- ListView: fixed a bug with ReversedOrder and asymmetric margins

16.4 Release 1.18.3

- Unity 6.1 and 6.2 support
- added implementation of the IEquatable<T> interface for all data types
- added ObservableData and ObservableDataWithPropertyChanged - base classes for the observable types
- added CalendarSpinner and DateSpinnerPicker prefabs (year can be changed with Spinner)

- added CalendarYearList and DateYearListPicker prefabs (similar to Windows calendar), replacing previous Calendar and DatePicker
- AutoCombobox: fixed disabled InputField when no selected item
- DateTime and DateTimePicker prefabs: now CalendarYearList is used by default
- ListView: fixed bug when deselect events are called with SelectedIndices in an invalid state
- Picker: added OnClose callback
- Popup: OnClose callback now is called after the close animation ends
- Tabs: added Animation, AnimationDuration, UnscaledTime, and CustomAnimation options
- TracksView: fixed bug with undisplayed items (thanks to cookiedrugs)
- WindowAnimations: fixed bug with incorrect anchorMax of the RectTransform
- WindowAnimations: one frame delay in animations is replaced with the OnAnimationStart event in the widgets that use this component

16.5 Release 1.18.2

- added FileTable prefab
- added StackView container
- added Unity Localization Support (will be used default table specified in the "String Database / Default Table Reference" in the localization settings)
- fixed the error when InputSystem is not installed
- Dialog: added parent argument to the Show() and ShowAsync() methods
- EasyLayout: the SettingsChanged event is now called if values changed in the Inspector window
- EasyLayout: fixes for the Flex layout
- IViewData<T>: added the OnItemChanged property
- ListView: added Flex layout support for the TileView with Fixed Size
- ListView: fixed bug with KeepScrollAtBottom when item is larger than ListView container
- ListViewComponent: added the OnItemChanged property
- Paginator: the VisiblePagesCount even values are rounded up to an odd number except 0
- Project Settings: now can change default namespaces and paths for the Widgets Generator
- RounderCorners and RounderCornersX4: now correctly works with the Mask component if Mask is placed after the RounderCorners component
- Shaders: fixed bug in VR mode
- Widgets Generator: added DataBind support for the Item property of the created ListViewComponent and TreeViewComponent
- Widgets Generator: fixed bug with get-only properties

16.6 Release 1.18.1

- ContextMenu: now the Visible and Interactable options of menu items are enabled by default
- Notification: added the AnimateOnHideButton option, if enabled the hide animation will run on HideButton click
- Notification Animations: the ShowDuration and HideDuration options are obsolete, use the ShowAnimationCurve and HideAnimationCurve options instead
- Notification Animations: removed the ShowAnimateOthers and HideAnimateOthers options, you can use the EasLayout.MovementAnimation option instead
- ScrollRectPaginator: now the ActivePage can work even if DefaultPage is not specified
- UIThemes Support: added support for default/active icons for the TabsIcons
- Widgets Generator: added support for the R3.SerializableReactiveProperty<T> fields and properties

16.7 Release 1.18.0

- Dialog, Picker, Popup: added animations support, for the existing dialogs/pickers/popups you can add the WindowAnimations component or create a custom animation component with IWindowAnimations implementation
- EasyLayout: removed incorrect warning in some cases
- Notification: added the NotificationsAnimations component for open/close animations, it has more priority than ShowAnimation and HideAnimation fields and arguments, you can create a custom animation component with the IWindowAnimations implementation
- TableHeader: added the FillContextMenu and ContextMenuNames options to automatically fill ContextMenu to enable/disable columns
- UtilitiesTime static fields were replaced with WidgetsTime.Instance
- Widgets Generator: added ContextMenu to enable/disable table columns
- reduced amount of static fields

16.8 Release 1.17.16

- Effects: fixed bug with zero vertices (like Text component without any text)
- ListView: added the KeepScrollAtBottom option, which keeps the scroll at the bottom on DataSource changes (only if the scroll is already at the bottom)
- ObservableListFilter: Output now uses the same comparison as Input
- Widgets Generator: added option to specify paths for the created scripts, prefabs, scenes, etc
- Widgets Generator: added GeneratorPaths attribute to specify paths directly in the data type
- Widgets Generator: added GeneratorNamespace attribute to specify namespaces for the created scripts directly in the data type
- shaders: fixed bug related to HDR
- shaders: fixed bug related to VR

16.9 Release 1.17.15

- InputAdapter: added workaround to properly update layout because TMP_InputField does not call LayoutRebuilder.MarkLayoutForRebuild() when it should be
- ListView: the RetainScrollPosition option now correctly works with the enabled ReversedOrder option
- Project Settings: removed performance impact on enter/exit play mode
- Rating: fixed error on Value = 0
- Rating: now interactable works properly
- SafeArea: added OnScreenChange event with information about borders
- Swipe: added OnTap event
- UIThemes: added the "Attach UI Only" option in "Project Settings... / UI Themes", if enabled ThemeTarget component will be added only to game objects with RectTransform component

16.10 Release 1.17.14

- COMPATIBILITY-BREAKING CHANGES: methods Updater.RunOnce(Action) and Updater.RunOnceNextFrame(Action) replaced with methods Updater.RunOnce(UnityEngine.Object owner, Action) and Updater.RunOnceNextFrame(UnityEngine.Object owner, Action)
- added ListViewDefaultItemAutoResize: automatically changes item instances size on ListView resize, used when only one item should be visible
- InputFieldAdapter: added the SetTextWithoutNotify(string text) method
- EasyLayout: added warning when one of the objects in the same row has relative width (or one of the objects in the same column has relative height)
- EasyLayout: added IgnoreLayoutElementSizes option: if enabled ILayoutElement options will be ignored, increases performance without side effects if ChildrenWidth and ChildrenHeight are not controlled
- EasyLayout: added Hex layout
- ScrollBlock: added ScrollBlockResizer component to resize ScrollBlock items to display specified item count independent of size
- TreeView: fixed bug in the TreeViewDropSupport components
- UIThemes: now support addressable assets, support can be enabled in "Project Settings... / UI Themes"
- UIThemes: Theme: added the AddressableSupport option, addressable assets can be preloaded with PreloadAddressable() or PreloadAddressable(VariationId) methods
- UIThemes: now requires references to the Unity.Addressables and Unity.ResourceManager in the UIThemes assembly definition
- UIThemes: now requires reference to Unity.Addressables.Editor in the UIThemes.Editor assembly definition

16.11 Release 1.17.13

- improved Assembly Definitions support
- ListView: reduced the SetData() calls for the item instances to improve performance
- ListView: fixed bug with DataSource.Clear()
- TreeView: improved RetainScrollPosition support

16.12 Release 1.17.12

- added ScrollRectAutoScroll component: allows scrolling content during drag&drop when the pointer is in less than a specified distance from the border
- ListView: fixed ReversedOrder for the horizontal direction
- Drag: added the DragDelay option - how many seconds must pass from the click to the start of dragging
- Drag: added the RedirectDragToScrollRect option - redirects the drag events to the parent ScrollRect if the drag starts before DragDelay time is passed

16.13 Release 1.17.11

- ListView: scroll no more "jump" when dragging backward and item size is changed
- Widgets Generator: fixed bug occurring when the default theme is not specified

16.14 Release 1.17.10

- ListView: added the RetainScrollPosition option, if enabled ListView will try to retain the scroll position when items are added or deleted from the list
- ListView: improved the LoopedList option support
- ListView, TiveView: fixed missing items in some cases if the list type is of variable size
- ListViewPaginator: fixed animation bug
- ObservableList: OnCollectionChange event is no more raised in cases like "list[index] = list[index]" (for reference type only)
- Widgets Generator: fixed script error

16.15 Release 1.17.9

- COMPATIBILITY-BREAKING CHANGES: ActivateInputField() method renamed to Activate() in the IInputFieldProxy interface
- added DataLoader class to simplify loading and caching data
- Accordion: added Interactable option
- CenteredSlider: added Interactable and PointerButton options
- ColorPicker: added Interactable option
- Combobox: added Interactable option
- Combobox: added RepositionListView option, if enabled ListView is automatically positioned to be completely visible if partially hidden by the bottom or right side of the screen
- Combobox: added ChangeRounderCorners option, if enabled then corner radiuses will be changed to match the repositioned ListView
- FileListViewPath: added Interactable, DisabledColor, and PointerButton options
- ListView: added LoopedNavigation option, enabled by default, if enables/disables navigation from first to last item and vice versa
- ListView: added MultipleSelectRequireKeys option, disabled by default, items selection will work similarly to file explorer if enabled
- ListView: added RequireVisible option, number of visible items from current item to border on navigation
- ListView: scroll no more "jump" when scrolling backward and item size is changed
- ListView: LoopedList option now works correctly with items of variable sizes
- ListViewPaginator: fixed bug with empty ListView and incorrectly detected page in some cases
- OpenContextMenu: added Interactable and PointerButton options
- Paginator: added Interactable option
- SliderScroll: no more change Slider value if it is not interactable
- Tabs: added TabsScroll component to toggle tabs with mouse scroll
- TreeNode: fixed incorrect deserialization
- UI Themes: fixed ThemeTarget bug caused by removed component

16.16 Release 1.17.8

- COMPATIBILITY-BREAKING CHANGES: `public virtual void Init()` methods are split into two methods `public void Init()` and `protected virtual void InitOnce()` to avoid excessive flag check. To fix broken code you need to replace `public override void Init()` with `protected override void InitOnce()`.
- COMPATIBILITY-BREAKING CHANGES: UIThemes ThemesReferences and ReferencesGUIDs scripts moved from the Editor folder to the Scripts folder
- added SelectableMarker: shows the specified marker for the currently selected game object.
- added TextMeshProPaginator for the TMP_Text (use with text Overflow = Page)

- AutoCombobox: added DataSource, SelectedIndex, SelectedIndices, SelectedItem, and SelectedItems properties (proxies for the corresponding ListView properties)
- Combobox: added DataSource, SelectedIndex, SelectedIndices, SelectedItem, and SelectedItems properties (proxies for the corresponding ListView properties)
- EasyLayout: added MovementAnimateAll and ResizeAnimateAll options - animate movement/resize for all elements if enabled; otherwise new elements will not be animated.
- ListView: CenterTheItems renamed to the AlignCenter
- Notification: SlideUpOnHide now is obsolete, use the EasyLayout "Movement Animation" option instead
- TreeNodes: added alwaysIncludeNested parameter to the Filter and FilterNodes method
- Paginators: added the HideIfOnePage option - an active page will be hidden if Pages <= 1

16.17 Release 1.17.7

- COMPATIBILITY-BREAKING CHANGES: now BeginUpdate() method of IObservableList<T>, ObservableList<T>, and GroupedList<T> returns ListUpdater instead of void
- Calendar: now the OutOfRangeDate option is properly used instead of the OtherMonth
- Calendar: now the date is selectable
- Dialog: added AutoFocus option - set focus to the last Selectable object in the Dialog
- DragSupport: fixed bug with OnDestroy()
- IObservableList<T>, ObservableList<T>, GroupedList<T>: now BeginUpdate() method returns disposable struct to automatically call EndUpdate() at the end of the block ("using var _ = DataSource.BeginUpdate();");
- Notifications: added FadeIn and FadeOut animations (requires CanvasGroup component)
- ListView: fixed navigation for the ListViewEllipse type
- Picker: added AutoFocus option - set focus to the last Selectable object in the Picker
- Popup: added AutoFocus option - set focus to the last Selectable object in the Popup
- Resizable: added the IncludeOuterRegion option to enable/disable resize outside of the target (disabling it restores behavior before v1.17.3)
- Resizable: improved compatibility with other draggable components like ScrollRect
- Resizable: ResizableRoot can be manually added to the parent object instead of the canvas
- Resizable: ResizableRoot is no longer required if IncludeOuterRegion is disabled
- TableHeader: fixed bug with destroyed cells
- small optimizations and code improvements
- UI Themes: "Remove All ThemeTargets" renamed to "Detach Theme"
- UI Themes: "Remove ThemeTargets with Default Theme" renamed to "Detach Default Theme"
- UI Themes: added "Attach to the Scene" button to the Theme editor, it will add/replace all themes in the active scene with the current one
- Settings: auto-attach theme now is disabled by default
- Widgets Generator: the created scene now uses a separate copy of the base theme
- Widgets Generator: fixed duplication of foregrounds list in DefaultItem if all of them are null

16.18 Release 1.17.6

- added SplitterMaxSize component to limit the maximum size of the Splitter targets (set value to 0 to disable)
- added SliderScroll component for the Slider widget with ScrollMode option to change the value on the mouse scroll
- CircularSlider: added ScrollMode option to change the value on the mouse scroll
- CenteredSlider: added ScrollMode option to change the value on the mouse scroll
- Drag: fixed the AutoScroll error
- ObservableList: added UseStableSort option, enabled by default
- ObservableList: added optional "bool stable" parameter to the Sort() methods
- SnapGrid: fixed problem with the Canvas scale factor
- small optimizations and code improvements

16.19 Release 1.17.5

- AutoCombobox: added RequireSelectedItem option to enable/disable selection of the first item
- Spinner: added ScrollMode option to change the value on the mouse scroll
- TileView: fixed problem with Variable Size mode when the same item is added to the list multiple times
- Widgets Generation: fixed script error with the "InstantiatePrefabs" property

16.20 Release 1.17.4

- fixed problems during the first installation
- added RectTransformCover: changes the RectTransform size (while preserving its ratio) to the smallest possible size to fill the parent, leaving no empty space
- Combobox: fixed problem with LayoutGroup
- DragSupport: removed FillRaycasts method and RaycastResults field
- ListView: added StretchToMaxItemSize option to stretch ListView to fully display the largest item, works only with ListType = ListViewWithVariableSize
- Project Settings: added "Use White Sprite" option to set sprite for the Image components without sprite, prevents rare bug when such Images are displayed as black
- Resizable: fixed behavior when the cursor is over a hidden part
- TableHeader: fixed the cell sizes jump when enabling column

16.21 Release 1.17.3

- Assembly Definitions: improved support when reinstalling the package
- Combobox: fixed problem with undesired focus on the toggle button
- BorderEffect: BorderColor alpha is now properly supported
- LinesDrawer, SnapGridDrawer: LineColor alpha is now properly supported
- Resizable: added the StopAtParentBorders option to prevent resizing outside of the parent borders.
- Resizable: now resize working if the cursor is outside of the target in ActiveRegion distance (previously only when it was inside)
- UIThemes: fixed build error for Unity 2021.3 versions

16.22 Release 1.17.2

- added Nodes.Filter() extension method
- ColorHSV: added LerpHSVAlternative() method to lerp Hue using the shortest distance
- ColorPickerDialog: value type changed from Color32 to Color
- LayoutSwitcher: fixed Updater-related bug
- Shaders: added lerpHSVAlternative() function to lerp Hue using the shortest distance
- TreeView: fixed Editor error when opening ScrollSettings
- Widgets Generator: the UpdateView() method of ListViewComponent is now virtual

16.23 Release 1.17.1

- fixed Style.cs bug when installing the package the first time
- added new Container prefabs: SlidersHorizontal, SlidersVertical, and TabsSliders
- added SpinnerVector3
- added ColorPickerDialog
- added shader functions: RGBtoHSV(float4 colorRGB), RGBtoHSV(float3 colorRGB), HSVtoRGB3(float3 colorHSV), lerpHSV(float4 colorRGB0, float4 colorRGB1, float s), lerpHSV3(float3 colorRGB0, float3 colorRGB1, float s)
- EasyLayout: fixed movement and resize animations
- Combobox: now ListView position and size do not change with canvas resize
- Combobox: improved gamepad and keyboard support
- ListViewPaginator: now ListView can be changed in runtime and can be null
- ListViewPaginator, ScrollRectPaginator: added SetSharedDefaultPages() and SetSharedSkipPages() methods
- ScrollRectPaginator: now ScrollRect can be changed in runtime and can be null
- Spinner: added ValueLimits option

16.24 Release 1.17.0

- COMPATIBILITY-BREAKING CHANGES: usages of `ReadOnlyCollection<T>` replaced with `IReadOnlyList<T>`
- fixed bug when using TMPro Converter on objects with `ThemeTarget` component
- added `RounderCorners` and `RounderCornersX4` effects
- added `StableSort` helper
- TMPro Support: added DefaultFont option in Project Settings, used by TMPro Converter
- UI Themes: added commands *"Find Options"* and *"Find And Create Options"* to use with existing `ThemeTarget` components
- UI Themes: font size by default changed to 24
- UI Themes: colorMultiplier by default changed to 1
- UI Themes: commands *"... Create Options"* now set the current value for all variations if the option was created

16.25 Release 1.16.5

- Unity 2023.2 support
- fixed tooltip sprites
- fixed Widgets Generator window error when all fields are deselected
- ListViewItem: replaced Graphic[] with List<Graphic> field properties (GraphicsForeground -> Foregrounds; Foreground -> foregrounds; graphicsBackground -> backgrounds; GraphicsBackground -> Backgrounds; cells-GraphicsBackground -> cellsBackgrounds)
- ListViewDropIndicator: fixed wrong position
- UIThemes: fixed bug when properties controlled by the owner were changed by Theme
- UIThemes: added Selectable.colorMultiplier support
- UIThemes: added Text.fontSize support

16.26 Release 1.16.4

- fixed assembly definitions error
- UI Themes: fixed error caused by a missing folder in the package (since Unity does not include an empty folder in the package)
- Paginator: added PageRounding option to determine how the current page is calculated

16.27 Release 1.16.3

- Autocomplete: fixed missing ListView in some cases
- Combobox: fixed the button position on the hierarchy (thanks to Antuan Johnson)
- Notification: templates buttons now are properly hidden
- Resizable: now correctly works with non-one scale
- SelectableHelper and SelectableHelperList: added Interactable and WatchInteractable properties

16.28 Release 1.16.2

- Different Draggable components: added DragButton option

16.29 Release 1.16.1

- added ObservableListFilter
- added prefabs for the default Unity widgets to use with the default theme
- added "Editor / Widgets References.asset" to replace default prefabs (available only after any widget was created with context menu)
- fixed error when replacing Unity Text with TMPro Text on the ThemeTarget component
- Autocomplete: added OnShowOptions and OnHideOptions events, use them to change the position of the shared DisplayListView
- ContextMenu: now shows actions count in the items editor
- ContextMenu: now parent items are highlighted when the submenu is opened
- ContextMenu: fixed position when open using the context menu key
- ListViewString: fixed sort (thanks to RickSaada1)
- ScrollRectPaginator: fixed bug with the wrong page count
- UIThemes: added option to specify folder, and namespace for wrappers, and enable generate wrappers in Project Settings
- UIThemes: ThemeTargets Search window: search is now performed on all opened scenes, not only active
- UIThemes: ThemeTargets Search window: added search on all scenes and prefabs
- UIThemes: ThemeTargets Search window: search results preserved after assembly reload
- UIThemes: added context menu "Remove ThemeTargets with Default Theme"
- UIThemes: added variations reorder
- UIThemes: added Theme.IsActiveProperty(name) method to control available properties
- UIThemes: white sprite can be marked with the "ui-themes-white-sprite" label
- UIThemes: fixed options reordering when filter enabled
- UIThemes: fixed variations delete

16.30 Release 1.16.0

- now the oldest supported version is Unity 2020.3
- Unity 2023.1 support
- added UIThemes as a replacement to the legacy Styles (Styles can be enabled in "Project Settings / New UI Widgets")
- Assembly Definitions are automatically created (this behavior can be disabled in "Project Settings / New UI Widgets")
- added SelectableHelperList, an equivalent of SelectableHelper with multiple TargetGraphics
- added a workaround to avoid the string comparison bug in WebGL
- fixed Domain Reload support
- added option to choose between creating widgets copies or prefabs from the menu
- AutocompleteStringCombobox: fixed value selection bug
- Combobox: now properly closed after clicking on the selected item
- Connectors: added line arrows
- Connectors: fixed bugs with Canvas Screen Space = Camera and canvas is rotated or have non-1 scale
- Dialog, Picker, Popup, Notification, ModalHelper: fixed bug with cache containing destroyed game objects
- EasyLayout: bug fixes
- ListView: added OnItemSelected and OnItemDeselected events
- ListView: added CreateTemplateSelector() method to replace TemplateSelector setter to support UI Themes
- ListView: fixed events error
- ListView: added notification if item type is not [Serializable]
- ListView: added ColoringStriped, DefaultEvenBackgroundColor, DefaultOddBackgroundColor
- Paginator: added VisiblePagesCount and SkipPage options
- Paginator: now correctly updated when using the scrollbar
- ScrollBlock: added Scroll(steps, AnimationCurve) method
- Styles are now obsolete and replaced with UI Themes
- Switch: fixed toggle animation
- TimeScroller: added Scroll(TimeSpan, AnimationCurve) method
- Tooltip: fixed bug when the tooltip was not hidden in some cases

16.31 Release 1.15.10

- Unity 2022.2 support
- domain reload support: fixed null reference exception
- Autocomplete: now input with tags is correctly parsed
- Combobox: now correctly updated when item properties changed
- Combobox: position in hierarchy correctly restored after ListView closed
- Combobox: fixed use ListView.Select()/Deselect() with raiseEvents = false
- EasyLayout: fixed Grid layout bug
- ListView: fixed GetComponentsEnumerator() return not all instances
- ListView: minor fixes
- ListViewPaginator: fixed LoopedList support
- ResizableHandles: added HandlesState field to control handles visibility on select/deselect events
- RotatableHandles: added HandleState field to control handles visibility on select/deselect events
- TreeView: fixed ContainerMaxSize, now the size is correct if TreeView has collapsed nodes

16.32 Release 1.15.9

- ListView: fixed undisplayed properties in the Inspector window
- Rating: fixed Interactable does not work correctly when disabled
- TreeView: added ContainerMaxSize option to prevent scrollbar blink caused by virtualization: the container will have the maximum width of all items. By default, the container has the maximum width of only visible items. Require ListType = List View with Variable Size.

16.33 Release 1.15.8

- added Rating widget (Text can be replaced with an Image or any other Graphic component)
- added async helpers scripts
- DatePicker, PickerInt, PickerString, and custom PickerListView: added an optional OK button and Mode option to choose between “close on select” and “close on OK click”
- Dialog, Notification, Picker, Popup: added OnBaseInstanceOpen and OnBaseInstanceClose static events
- Dialog, Notification, Picker: added OnInstanceOpen and OnInstanceClose static events for the custom types
- Popup: added ShowAsync() method to use with async/await
- Styles: fixed missing font in some Unity versions
- TileView: added LinearGroupedTileView example
- TracksView: added Timeline
- menu “New UI Widgets/Dialogs” renamed to “New UI Widgets/Dialogs Templates”

16.34 Release 1.15.7

- added LoadAnimation widget
- Dialog, Lightbox, Picker, Popup: added HideOnModalClick option
- added workaround to avoid ReSharper RRSRP-489023 bug
- Combobox: fixed bug when items were removed but still displayed as selected
- ListView: added ChangeLayoutType option: if enabled changes EasyLayout.LayoutType to match ListType.
- ListView, TreeView: now the deselect events invoked for the removed indices/nodes

16.35 Release 1.15.6

- now the oldest supported version is Unity 2018.4
- added Grayscale effect
- added LocalizationSupport option to disable translation for widgets with localization support
- added LimitMaxSize script to limit size when using anchors stretch
- added ProgressbarCircular prefab and menu option
- added SafeArea script to resize RectTransform to fit the safe area
- added Swipe script
- Dialog, Notification, Picker: added ShowAsync() method to use with async/await
- Dialog, Notification, Picker: added IsDestroyed property to check if is instance destroyed
- Dialog, Notification, Picker: now destroying instances will raise cancel or hide events
- Effects (derived from UVEffect): improved filled image type support
- Calendar: add OtherMonthWeekend and OutOfRangeDate colors to the Date component
- ContextMenu: fixed bug occurring with opened "ContextMenu Items Editor" window in play mode
- ListView, TreeView: fixed incorrect drop indicator position in some cases
- ListView: added GetInstanceSize(), SetInstanceSize(), ResetInstanceSize() methods to animate items resize without problems with virtualization
- ListView: fixed wrong drop position and indicator if enabled CenterTheItems
- Ring Effect: added Fill option
- TreeView Drop Support: added AutoDropPosition and DropPosition options
- TreeView: added AllowToggle option
- TreeView: added TreeViewToggleAnimation script to animate node toggle
- Widgets Generator: fixed bug with "const" fields
- Widgets Generator: now you can select the fields that will be used in the widgets, including the field for the autocomplete; it also can be done with [GeneratorIgnore] and [GeneratorAutocomplete] attributes

16.36 Release 1.15.5

- improved Unity 2021.3 LTS support
- COMPATIBILITY-BREAKING CHANGES: ListView: methods ComponentCreated, ComponentDestroyed, ComponentActivated, ComponentCached changed to public
- ListView: added SetSharedTemplates() method
- Widgets Generation: fixed bug when data type has a parameterless constructor

16.37 Release 1.15.4

- added ScrollRectDragSensitivity
- added UtilitiesScrollRect: get time for ScrollRect stop by inertia
- ListView: added ReversedOrder option (items displayed from end to start)
- ListView: added OnlyOneHighlighted option
- ScrollRectHeader, ScrollRectFooter: added Visible option to show and hide header (or footer)
- ScrollRectHeader, ScrollRectFooter: added layout support if DisplayType is Reveal
- ScrollRectFooter: added ChangeLayout option
- ScrollRectPaginator: added RoundingError option to avoid excess last page
- Switch: added AnimationCurve option
- Widgets Generation: fixed enum related bug
- Widgets Generation: improved support of latest Unity versions (2022.1.0+)
- Style: fixed "Create Style" bug

16.38 Release 1.15.3

- COMPATIBILITY-BREAKING CHANGES: LateUpdateAdd and LateUpdateRemove methods of IUpdaterProxy renamed to AddLateUpdate and RemoveLateUpdate, added RemoveRunOnce and RemoveRunOnceNextFrame methods
- ContextMenu: added helper script OpenContextMenu to open the menu by clicking on non-UI gameobject, requires PhysicsRaycaster and/or PhysicsRaycaster2D on the main camera
- DateTimeScroller and DateScroller: fixed AMPM change on hours scroll if IndependentScroll enabled
- EasyLayout: added optional movement and resize animation support; warning: can decrease performance
- Effects: RingEffect, RippleEffect, LinesDrawer, SnapGridDrawer no more requires enabled TexCoord1 channel on Canvas
- ListView: fixed item instance visibility if ListViewItem.DisableRecycling enabled
- ListView: added OnNavigate event; called after navigating to the other item instance with keyboard or gamepad
- ListView: added ItemsEvents.MovedToCache event
- ListView: FixHighlightItemUnderPointer option now obsolete

- ListView: added KeepHighlight option to keep item highlight on pointer enter until will be selected another gameobject
- ListView: fixed wrong events processing order in some cases
- ListViewItem: added StopSelectableAnimations() method to ListViewItem class
- Notification: added OpenedNotifications, AllNotifications and InactiveNotifications properties
- Picker: added OpenedPickers, AllPickers, and InactivePickers properties
- Spinner: fixed bug with Unity Text and OnKeyDown validation
- Tabs: added EventSystemSelectActiveHeader option
- Tabs: added ImmediateSelect option
- Tabs: added NextTab() and PreviousTab() methods
- TimeScroller: added SingleAMPM property to disable multiple AM PM options in scroll block
- TreeNode: added HasNodes and HasVisibleNodes properties
- TreeView: added ToggleOnNavigate option, if enabled expand node on move right event and collapse node on move left event
- TreeView: added ToggleOnSubmitCancel option, if enabled expand node on submit event and collapse node on cancel event

16.39 Release 1.15.2

- added Updater static class to control scripts updates IUpdatable.RunUpdate() without reflection instead of the default MonoBehaviour.Update()
- added SnapGrid: sticks draggable or resizable UI game objects to the nearest grid lines
- added LinesDrawer and SnapGridDrawer effects
- COMPATIBILITY-BREAKING CHANGES: MonoBehaviour.Update() replaced with IUpdatable.RunUpdate()
- Autocomplete: added AllowCancelOnDeselect to cancel DisplayListView close on deselect event.
- Autocomplete: added OnSearchCompleted event
- Autocomplete: added ResetListViewSelection option
- AutoCombobox: fixed InputField display bug
- AutoCombobox: fixed coloring bug
- AutoCombobox: added KeepSelection option (set Autocomplete.DisplayListView selected items)
- Connectors: fixed incorrect positions when CanvasMode is WorldSpace and its scale is not 1
- Dialog: added ButtonsContainer option
- ListView: fixed bug with incorrect item sizes when using variable size type
- ListView: fixed highlighting bug
- ListView: fixed wrong background color for the last items in table mode
- ListView: added RangeMode property to determine which element is the start when selecting a range with the Shift key.
- ListView: fixed instance recycling if ListViewItem.IsDragged enabled

- ListViewItem: IsDragged renamed to DisableRecycling
- ListViewString: now sort can be disabled with EnableSort in the Inspector window
- Notification: added ButtonsContainer option
- Paginator: added SetPage method to change current page without animation
- Sidebar: added ModalColor field
- shaders: now should support stereo instanced rendering and SRP batcher (thanks to David Watt)
- Tooltip: added generic Tooltip
- TreeView: added FindNodes method
- Widgets Generation: added Tooltip generation

16.40 Release 1.15.1

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

16.41 Release 1.15.0

- added ListViewEnum with ListViewEnum<T> wrapper to work with any enums
- reduced memory allocations
- all classes with INotifyPropertyChanged support now also implements IObservable which works without memory allocations
- Autocomplete, Combobox: added field ParentCanvas, it used as ListView parent on open
- Combobox: toggle-button is now full width
- ContextMenu: now works correctly with all canvas render modes
- DateScroller/DateTimeScroller/TimeScroller: ScrollBlock replaced with ScrollBlockBase
- Dialog: added InactiveDialogs and AllDialogs properties to get access to the template instances
- Dialog: DialogButton now support callback with Func<DialogBase dialog, int buttonIndex, bool closeDialog> type

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

16.42 Release 1.14.2

- added CircularSlider widget
- added SliderScale widget
- added TimeAnalog widget
- Accordion: added OnStartToggleAnimation and OnDataSourceChanged events
- Accordion: added AccordionHighlight component
- AutoComplete: added OnItemNotFound and OnCancelInput events
- AutoCombobox: added AddItems option (requires overrided Input2Item method)

- CenteredSlider: event OnValuesChange renamed to OnValueChanged
- Connectors: fixed bug related to "Scale With Screen Size"
- Connectors: added rectangular lines support
- Cursor: fixed flickering
- DatePicker and DateTimePicker: fixed initial date
- DirectoryTreeView: nested nodes are automatically loaded on expand from script
- DirectoryTreeView: added ExpandPath(), Path2Node(), Path2NearestNode(), RefreshDirectories() methods
- ListView: fixed highlight coloring on navigation
- ListView: fixed unstoppable auto scroll bug
- ListView Drop Support: added ReceiveOnlyEmptyNode option
- RangeSlider: event OnValuesChange renamed to OnValuesChanged
- Resizable: fixed position change
- ScrollBlock: added AllowIncrease and AllowDecrease fields
- Spinner: added SetValue() method to change value without OnChangeEvent invocation
- Styles: fixed error when creating a new style
- Styles: added PixelsPerUnitMultiplier property to the Image styles
- TreeGraph: small performance improvement
- TreeGraph: added LineThickness, LineType, LineMargin options
- TreeView: added ScrollWithIndent option
- Widgets Generation: improved localization support

16.43 Release 1.14.1

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

16.44 Release 1.14.0

- added localizations integration support
- added I2 Localization support
- added ContextMenu
- added Input System support
- added UtilitiesCompare class
- added ScrollRectFooter
- added AutoComboboxIcons prefab
- Dialog, Picker, Popup: added CloseButton property

- EasyLayout: added SetPreferredAndFitContainer option for the Children Size
- ListView: added Header property
- ListViewPaginator: added LoopedList support
- Notification: added "content" and "onReturn" parameters to the Show() method
- Style: fixed unchangeable settings after "Apply Fast Settings" use
- Style: added "Update Default Style" option, which is opposite of the "Apply Default Style", it gets style settings from widgets and saves them to the current style
- Tabs: added CanSelectTab field to check if tab can be selected with a button click
- TabsCustom: TabButton class changed to the generic class TabButton<T>
- Widgets Generation: generated classes are partial now
- Widgets Generation: added AutoCombobox widget
- Utilities: most functions moved to the new Utilities* classes

16.45 Release 1.12.6

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

16.46 Release 1.12.5

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

16.47 Release 1.12.4

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

16.48 Release 1.12.3

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

16.49 Release 1.12.2

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

16.50 Release 1.12.1

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

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

16.51 Release 1.11.2

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

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

16.52 Release 1.11.1

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

16.53 Release 1.11.0

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

16.54 Release 1.10.4

- added DateScroller, DateTimeScroller, DateTimeScrollerSeparate, TimeScroller widgets
- added EditorCondition attributes to use with MonoBehaviourConditional and UIBehaviourConditional
- added LayoutElementMax: allow to control the maximum preferred sizes of the LayoutGroup
- added UIFlare shader
- Combobox: added HideAfterItemToggle option
- DateTime: fixed init and time errors
- DatePicker: added DateChangeOnly option to allow to select date on change or on click
- EasyLayout: fixed FitContainer
- ListView: added null value support for the GraphicsForeground and GraphicsBackground properties
- ListView: added AllowColoring option
- ListView: added StateDefault(), StateSelected() and StateHighlighted() functions to the base default item class as addition to coloring functions
- ListView: added loading example with UIFlare shader use

16.55 Release 1.10.3

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

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

16.57 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

16.58 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”

16.59 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

16.60 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()

16.61 Release 1.9.1

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

16.62 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

16.63 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

16.64 Release 1.8.4

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

16.65 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

16.66 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

16.67 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.

16.68 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.

16.69 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.

16.70 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.

16.71 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.

16.72 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 Height" with options:**
  - 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.

16.73 Release 1.5.0

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

16.74 Release 1.4.2

- Added ListViewCustomHeight (support items of variable heights)

16.75 Release 1.4.1

- Added CenteredSlider.

16.76 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.

16.77 Release 1.3

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

16.78 Release 1.2

- Added Dialog
- Added Draggable

16.79 Release 1.1

- Added Notify
- Added EasyLayout

16.80 Release 1.0

- Initial release