New UI Widgets Documentation

Release 1.10.1f1

Ilia Novikov

Nov 12, 2018

Contents

CHAPTER 1

New in 1.10

CHAPTER 2

Widgets

2.1 Collections

- **Combobox** Data type `string`.
- ComboboxIcons
- **ComboboxIconsMultiselect** ComboboxIcons with multiple selection support.
- DirectoryTreeView *
- FileListView *
- **ListView** Data type `string`.
- **ListViewColors** Data type `Color`.
- **ListViewInt** Data type `int`.
- ListViewIcons
- **ListViewHeight** Data type `string`.
- **ListViewPaginator** Paginator for ListView, TileView, and Table.
- TreeView

2.2 Containers

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

2.3 Dialogs

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

2.4 Input

- **Autocomplete** Data type `string`.
- AutocompleteIcons
- ButtonBig
- ButtonSmall
- Calendar
- **CenteredSlider** Horizontal direction.
- **CenteredSliderVertical** Vertical direction.
- ColorPicker
- ColorPickerRange
- ColorPickerRangeHSV
- **ColorsList** Should be used with ColorPicker to save colors.
- **DateTime** Data type `DateTime`.
- **RangeSlider** Data type `int`. Horizontal direction.
- **RangeSliderVertical** Data type `int`. Vertical direction.
- **RangeSliderFloat** Data type `float`. Horizontal direction.
- **RangeSliderFloatVertical** Data type `float`. Vertical direction.
- **Spinner** Data type `int`.
- **SpinnerFloat** Data type `float`.
- Switch
- **Time12** Data type `TimeSpan`. 12-hour format with AM / PM switch.

- **Time24** Data type `TimeSpan`. 24-hour format.

2.5 Misc

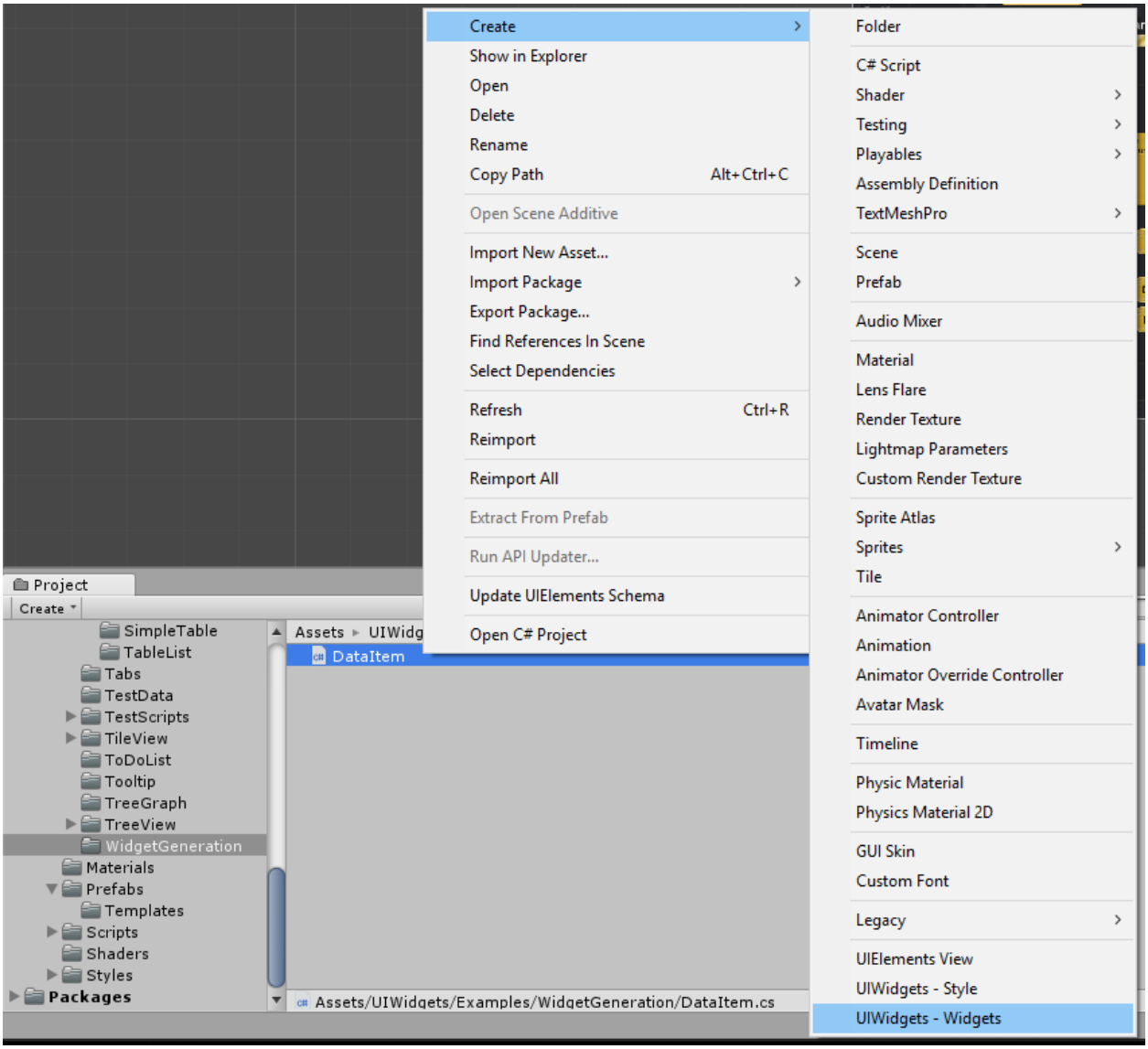
- AudioPlayer
- ProgressbarDeterminate
- ProgressbarIndeterminate
- ScrollRectPaginator
- ScrollRectNumericPaginator

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

CHAPTER 3

Widgets Generation

You can generate widgets for your data type with *Context menu / Create / UIWidgets - Widgets*.



3.1 Requirements

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

3.2 Supported types

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

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

Graphic types:

- `Sprite`
- `Texture2D`
- `Color`
- `Color32`

3.3 Limitations

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

3.4 Replacing generated code

Generated code can be freely modified.

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

3.4.1 Collections

You can replace default widgets used to display item fields with other widgets.

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

Original code:

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

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

      ...

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

(continues on next page)

(continued from previous page)

```
            protected set;
        }

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

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

            ...
        }

        ...
    }
}
```

New code:

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

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

        ...

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

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

(continues on next page)

(continued from previous page)

```
            base.Start();

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

        ...
    }
}
```

3.4.2 Autocomplete

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

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

Original code:

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

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

New code:

```
namespace UIWidgets.Examples.WidgetGeneration.Widgets
{
  /// <summary>
   /// Autocomplete for the DataItem.
   /// </summary>
   public class AutocompleteDataItem : UIWidgets.AutocompleteCustom<UIWidgets.
↪Examples.WidgetGeneration.DataItem,
      ListViewComponentDataItem, ListViewDataItem>
   {
      ...
      /// <summary>
      /// Returns a value indicating whether Input occurs within specified value.
      /// </summary>
      /// <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());
```

(continues on next page)

(continued from previous page)

```
            }
        }
}
```

CHAPTER 4

Using Widgets

4.1 Collections

4.1.1 Combobox

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

4.1.2 ListView, TileView and Table

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

List Types

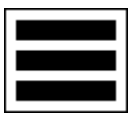


Fig. 1: ListView with Fixed Size

Fig. 2: ListView with Variable Size

Fig. 3: TileView with Fixed Size

Fig. 4: TileView with Variable Size

Get items

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

Set items

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

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

Display same list with ListView, TileView or Table

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

Get last selected index

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

Get selected indices

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

Last selected item

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

Get selected items

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

Delete specified item

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

Delete item by index

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

Clear list

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

Add item

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

Add items

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

Optimization

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

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

(continues on next page)

(continued from previous page)

```
items.Add(new_item);
items.AddRange(new_items);
items.RemoveAt(0);

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

Replace item

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

Sort

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

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

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

Enable permanent sort

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

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

Disable permanent sort

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

Set selected index

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

Or:

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

Behavior is different if you enable `MultipleSelect`:

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

Deselect

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

Or:

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

Scroll to item

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

4.1.3 Paginator

How to select paginator

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

Settings

- **(optional) Default Page** Template GameObject to display inactive pages
- **(optional) Active Page** Template GameObject to display active page
- **(optional) Prev Page** GameObject, go to the previous page
- **(optional) Next Page** GameObject, go to the next page
- **Fast Drag Distance and Fast Drag Time** Scroll to the next or previous page if drag distance more than *Fast Drag Distance* and drag time less than *Fast Drag Time*. To disable set to zero.
- **Force Scroll On Page** Scroll to the nearest page when drag ended if not meet *Fast Drag* condition. Should be used only with touch devices.
- **Animation** Enable animation
- **Movement** Animation curve
- **Unscaled Time** Enable if the animation should use Unscaled Time.

ListViewPaginator specific settings

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

ListViewPaginator works with ListLiew, TileView (in this case PerPage is rows or columns count) and TreeView.

ScrollRectPaginator specific settings

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

Tile View ScrollRect Fitter

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

4.1.4 TreeView

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

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

Get nodes

```
public TreeView Tree;

ObservableList<TreeNode<TreeViewItem>> nodes;

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

Get selected nodes

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

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

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

Add listeners

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

   Tree.NodeDeselected.AddListener(ProcessDeselectedNode);
}

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

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

Select node

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

Select node with subnodes

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

Deselect node

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

Deselect node with subnodes

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

Scroll to node

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

Add node

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

Hide nodes

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

Collapse node

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

Expand node

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

Change node name

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

Sort

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
      if (node.Nodes != null)
      {
         ApplyNodesSort(node.Nodes as ObservableList<TreeNode<T>>, comparison);
      }
   });
}
```

Filter nodes

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

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

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

bool SampleFilter(IObservableList<TreeNode<TreeViewItem>> nodes, Func<TreeViewItem,
↪bool> filterFunc)
{
   return nodes.Count(x =>
   {
      var have_visible_children = (x.Nodes==null) ? false : SampleFilter(x.Nodes,␣
↪filterFunc);
      x.IsVisible = have_visible_children || filterFunc(x.Item);
      return x.IsVisible;
   }) > 0;
}
```

Reset filter

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

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

   node.IsVisible = true;
}
```

Clear nodes

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

4.2 Containers

4.2.1 Accordion

Open item

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

Close item

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

Toggle item

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

Set items

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

4.2.2 Tabs

Select tab

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

Enable tab

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

Disable tab

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

4.3 Dialogs

4.3.1 DatePicker

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

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

      [SerializeField]
      Text Result;

      DateTime currentValue = DateTime.Today;

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

4.3.2 Dialog

Minimal code

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

Advanced

```
// create dialog from template
var dialog = dialogPrefab.Clone();
// show dialog with following parametres
```

(continues on next page)

(continued from previous page)

```
dialog.Show(
   title: "Modal Dialog",
   message: "Simple Modal Dialog.",
   buttons: new DialogActions(){
      // Button name and Func<bool>, return true to close dialog, otherwise false
      {"Close", Dialog.Close},
   },
   focusButton: "Close",
   modal: true,
   modalColor: new Color(0, 0, 0, 0.8f)
);
```

Adding new behaviour

1. Create helper component

```
using UnityEngine;
using UnityEngine.UI;

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

   [SerializeField]
   public InputField Password;

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

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

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

      return valid_username && valid_password;
   }
}
```

2. Show dialog.

```
public void ShowDialogSignIn()
{
```

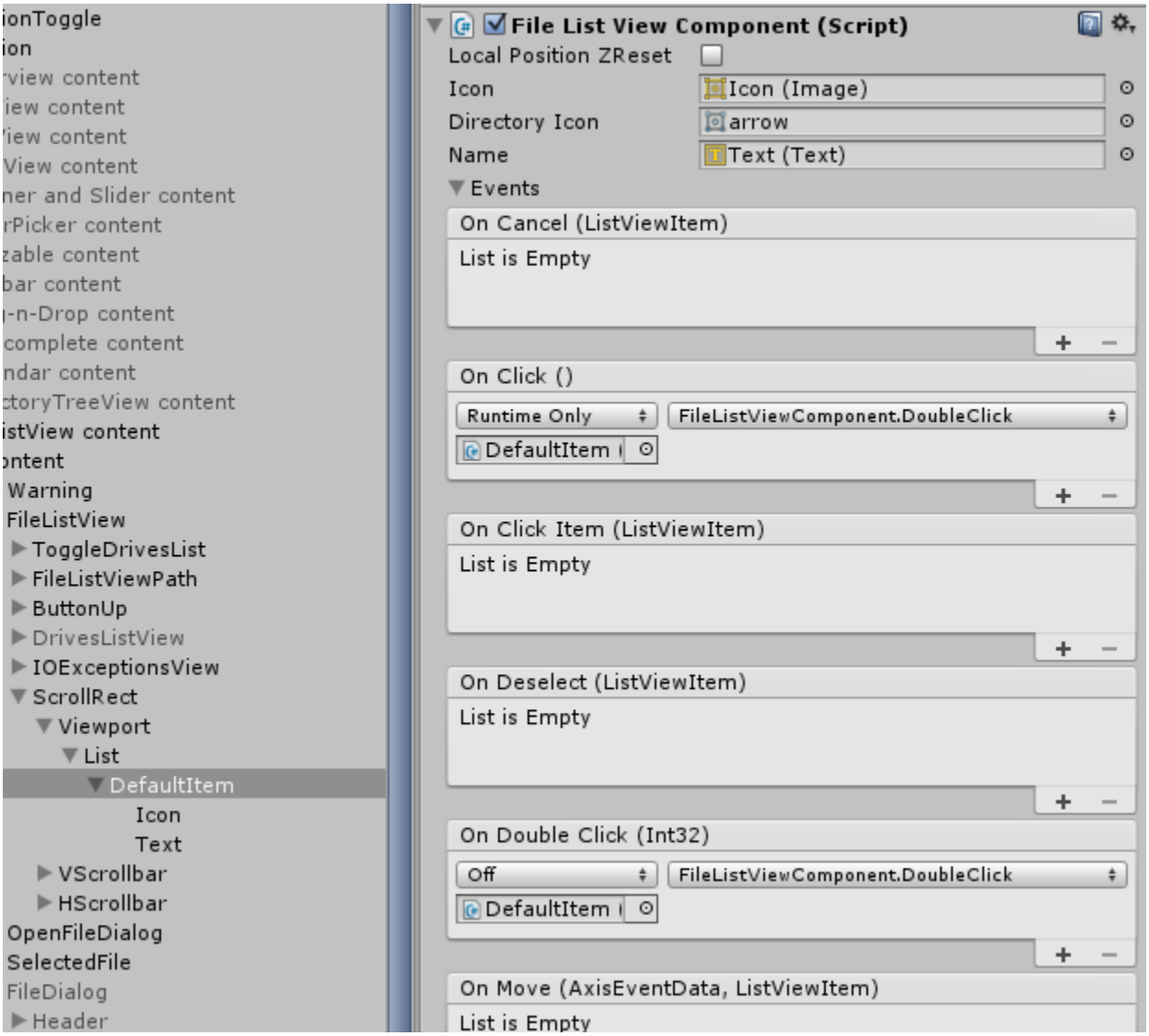
(continues on next page)

(continued from previous page)

```
    var dialog = dialogSignIn.Clone();
    var helper = dialog.GetComponent<DialogInputHelper>();
    helper.Refresh();

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

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

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

    return true;
}
```

4.3.3 FileDialog

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



Using FileDialog.

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

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

        [SerializeField]
        Text Result;
```

(continues on next page)

(continued from previous page)

```
        string currentValue = string.Empty;

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

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

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

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

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

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

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

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

4.3.4 FolderDialog

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

(continues on next page)

(continued from previous page)

```
using UnityEngine;
using UnityEngine.UI;

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

		[SerializeField]
		Text Result;

		string currentValue = string.Empty;

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

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

4.3.5 Notifications

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

Minimal code

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

Advanced

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

4.3.6 Pickers

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

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

(continues on next page)

(continued from previous page)

```
            int currentValue = 0;

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

                    // set values from template
                    picker.ListView.DataSource = PickerTemplate.ListView.DataSource.
↪ToObservableList();
                    // or set new values
                    //picker.ListView.DataSource = Enumerable.Range(1, 100).
↪ToObservableList();

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

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

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

4.3.7 Popup

Minimal code

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

Advanced

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

(continues on next page)

(continued from previous page)

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

4.4 Input

4.4.1 Autocomplete

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

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

      [SerializeField]
      ListViewIconsItemDescription item;

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

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

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

4.4.2 Calendar

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

```
namespace UIWidgets.Examples
{
   using UnityEngine;

   /// <summary>
   /// Test Calendar.
```

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

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

4.4.3 Centered Slider

The difference from a simple slider:

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

Set value

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

Set display limits

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

Set value limits

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

4.4.4 ColorPicker

Set color

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

Get color

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

Add listener

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

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

4.4.5 RangeSlider

Slider with two handles for minimum and maximum.

Set values

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

Set step

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

Set limits

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

Add listener

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

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

4.4.6 Spinner

Set maximum

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

Set minimun

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

Set value

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

Set step

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

Get value

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

4.5 Misc

4.5.1 ProgressbarDeterminate

Set value

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

Stop animation

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

4.5.2 ProgressbarIndeterminate

Start animation

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

Stop animation

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

CHAPTER 5

Components

5.1 Common components

- **Draggable** Dragging gameobject.
- **Resizable** Resizing gameobject.
- **Splitter** Resize neighboring gameobject on drag. Should be used with layout group.
- **Resizable header** Used with ListView on table mode. Allow columns resizing and reorderindg.
- **Sidebar** Object to drag from behind the screen.
- **Tooltip** Displaying the tooltip on object focus.
- **Switch Group** Same as Toggle Group, but for Switch.
- **Bring to Front** Should be used with Dialog or Draggable objects.
- **ScrollRectEvents** Provide pull events for ScrollRect.
- **EasyLayout** Layout group.
- **TreeView data source** To use in editor mode, allow editing TreeView nodes.
- **Selectable Helper** Selectable works only with one Graphic component, Selectable Helper allows to control more Graphic components.

5.2 Widget specific components

- **Drag and Drop components** Different components used with different widgets. Drag components attached to collections DefaultItem. Drop components to collections widget and TreeView.DefaultItem.

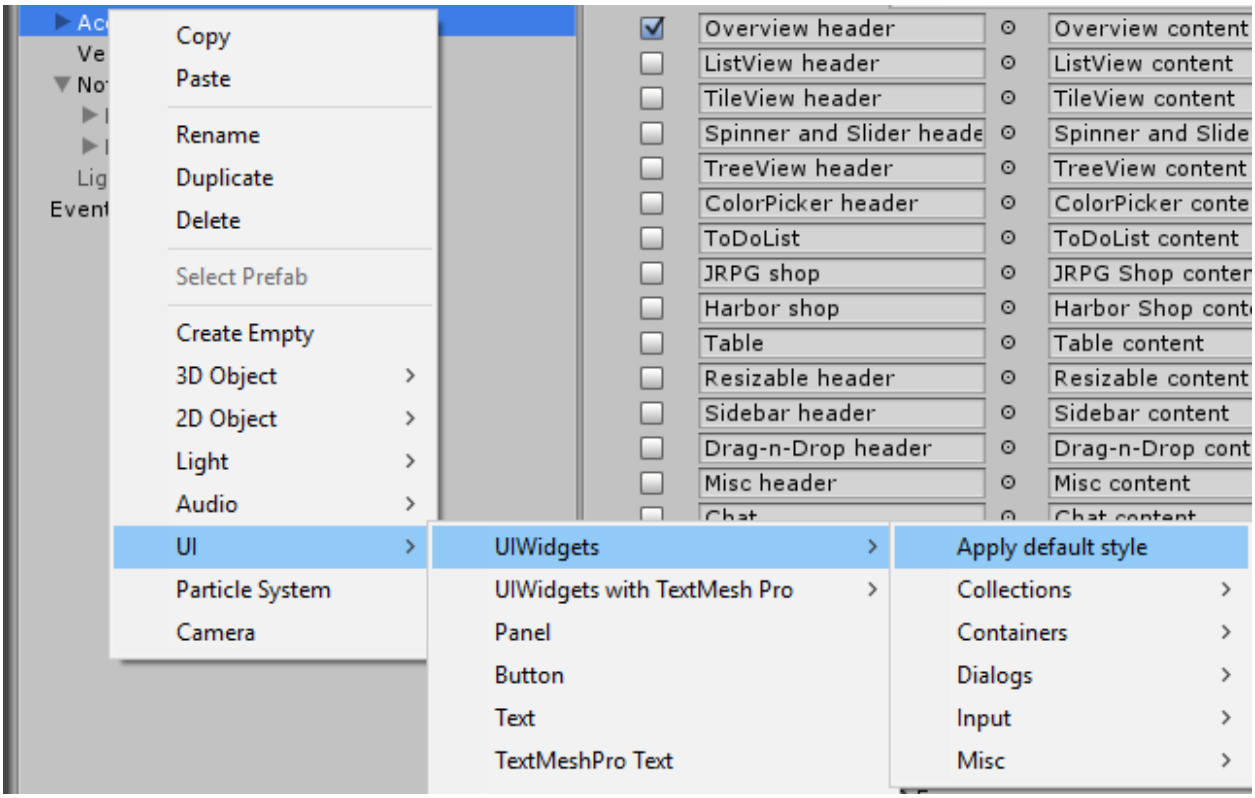
CHAPTER 6

Styles

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

New style can be created with menu *Assets / Create / UIWidgets - 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 / UIWidgets / Apply default style*.



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

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

CHAPTER 7

Localization Support

7.1 Default ListViewIcons and TreeView

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

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

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

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

         return targetListView;
      }
   }

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

   void OnDestroy()
```

(continues on next page)

(continued from previous page)

```
    {
        LocalizationSystem.OnLanguageChanged -= Localization;
    }

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

7.2 Generated ListView, TreeView, TileView

You can change component class to add localization support.

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

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

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

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

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

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

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

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

7.3 Tabs

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

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

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

CHAPTER 8

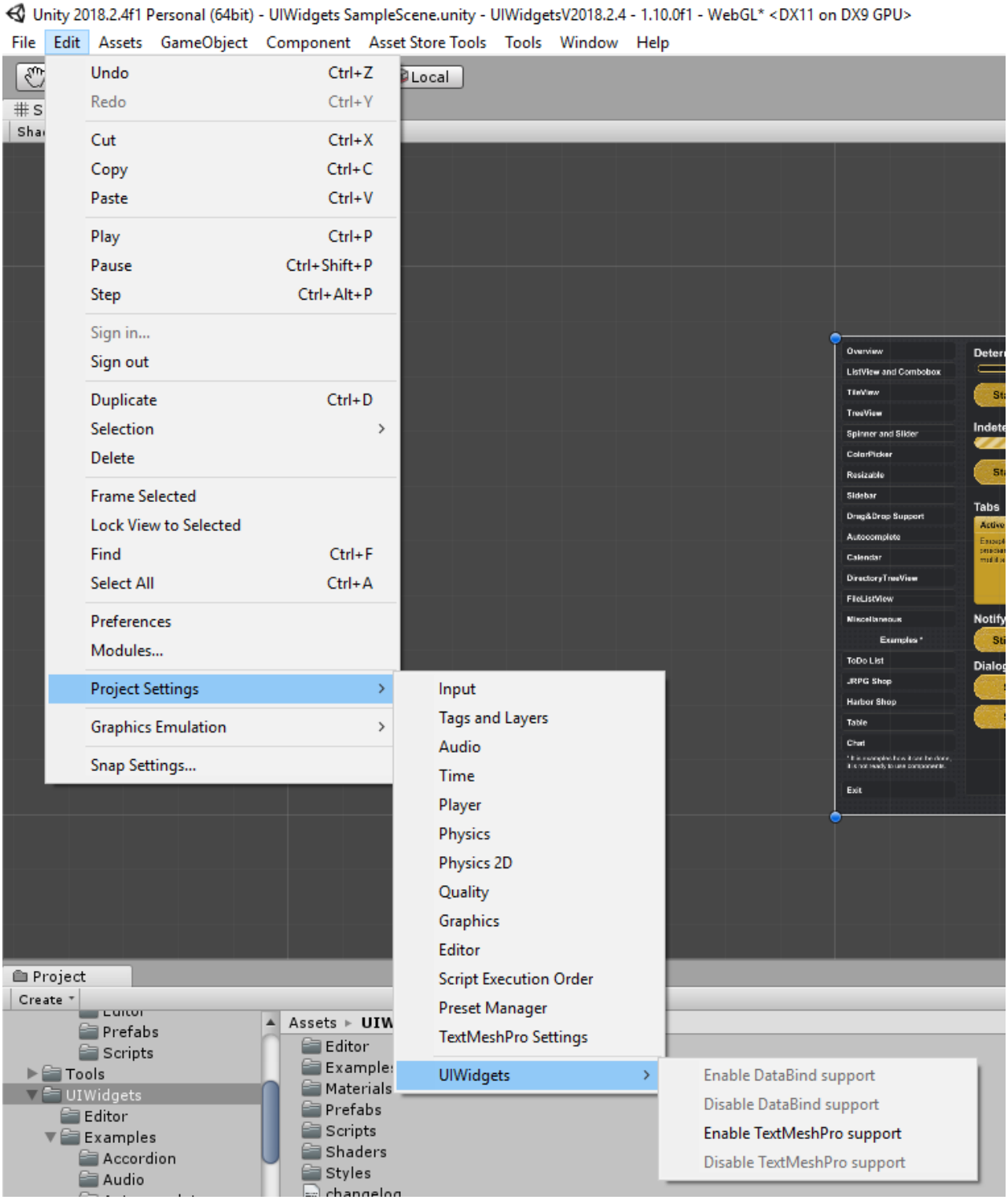
Data Bind for Unity Support

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

After enabling support:

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

Disable support with *Project Settings / UIWidgets / Disable DataBind Pro support*.


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

CHAPTER 9

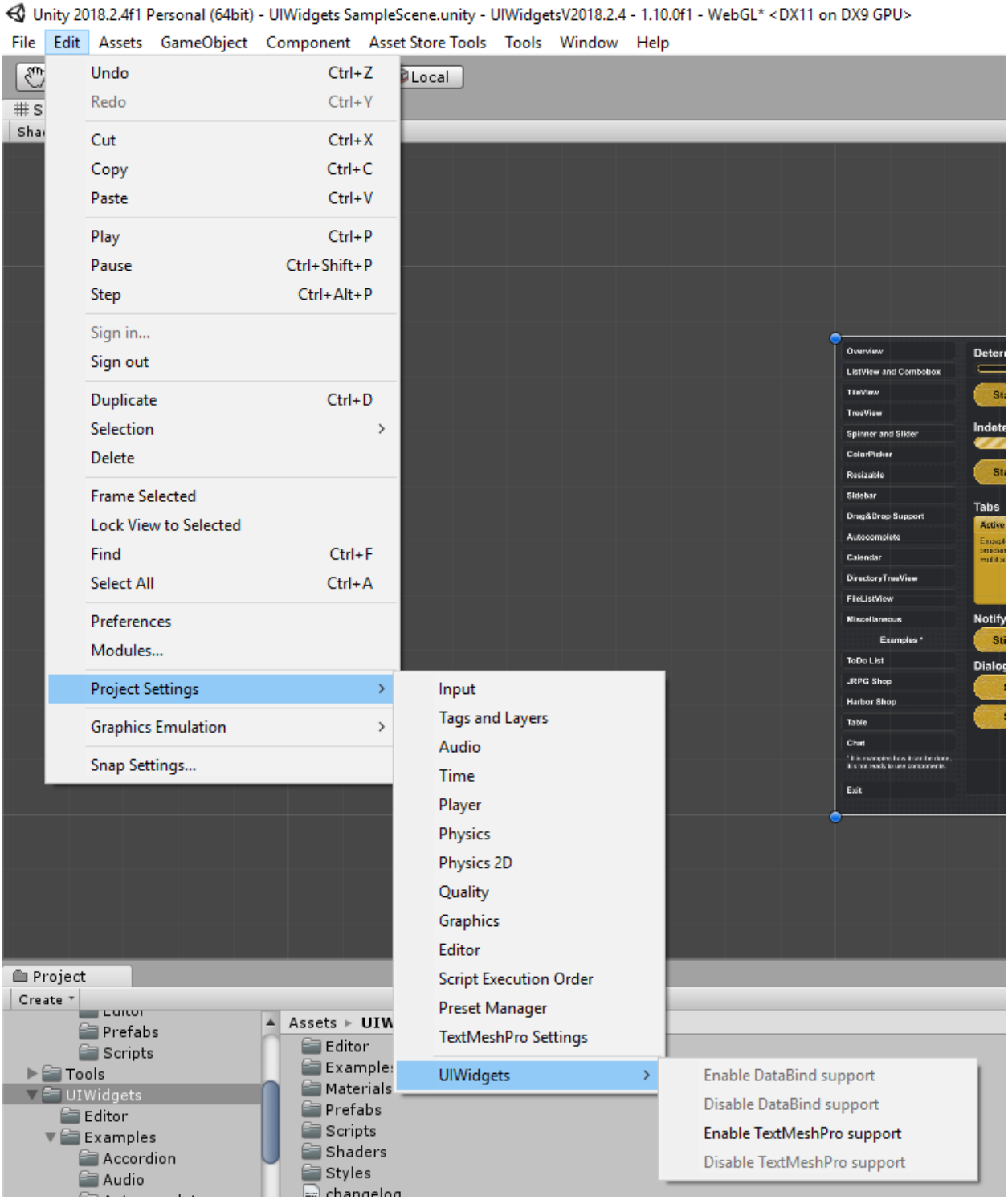
TextMesh Pro Support

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

After enabling support:

- default widgets with TextMesh Pro can be created with menu *UI / UIWidgets with TextMesh Pro*
- generated widgets will be using TextMesh Pro instead default Text

Disable support with *Project Settings / UIWidgets / Disable TextMesh Pro support*.


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

CHAPTER 10

Support

You can ask me questions at:

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

CHAPTER 11

Changelog

11.1 Release 1.10.0

- Added styles support (Styles folder, new styles can be created from context menu “Create / UIWidget - Style”)
- Added widget generation (context menu “Create / UIWidget - Widgets” on file with item class definition)
- Added DateTime, Time24 and Time12 widgets
- Added DateTimePicker and TimePicker widgets
- Added ColorPickerRangeHSV widget
- Added ColorsList widget to display list of the selected colors, should be used with ColorPicker or ColorPickerRange.
- Added “Data Bind for Unity” support (requires Unity 5.6 or later)
- Added base ListView Picker class for the custom ListView
- Added base TreeView Picker class for the custom TreeView
- Added base drop support class for the custom TreeView
- Added base drop support class for the custom TreeView node
- Added assembly definitions
- Improvement: Drag can be canceled with Cancel button
- Accordion: added AllItemsCanBeClosed option
- Autocomplete: added GetInputFieldText() function
- Calendar: added DateMin and DateMax properties
- Calendar: added currentDateAsDefault option
- ColorPicker: added Hex block
- ColorPicker: added new palette mode HSVCircle

- ColorPickerRange: DefaultShader replaced with DefaultShaderHorizontal and DefaultShaderVertical
- Connectors: now works correctly with "Screen Space - Camera"
- EasyLayout: reduced memory allocations
- EasyLayout: EasyLayout namespace renamed to EasyLayoutNS to avoid problems with Unity 2018.2 and later
- Interfaces: IItemWidth, IItemHeight, IListViewItemHeight, IListViewItemWidth not used anymore
- ListView: added CenterTheItems property
- ListView: added overridable functions CanBeSelected() and CanBeDeselected()
- ListView: added LoopedList option
- ListView: added Interactable option
- ListView: added IsTable option (required to valid stylization)
- ListView and TileView: ListViewCustomWidth, ListViewCustomHeight, TileViewCustom and TileViewCustomSize replaced with ListViewCustom with List Type option
- ListViewCustomWidth: TItem now does not require IItemWidth implementation
- ListViewCustomHeight: TItem now does not require IItemHeight implementation
- ListViewDropIndicator: added styles support
- ResizableHeader: fixed resize on touch devices
- Sidebar: added OnOpeningStarted and OnClosingStarted, called when appropriated animation started
- other: prefabs in "Sample Assets" folder replaced with scenes
- other: "Standart Assets" folder renamed to "Scripts"
- other: "Sample Assets" folder renamed to "Examples"
- other: removed ListViewGameObjects prefab
- other: removed outdated prefabs and sprites
- other: namespace "UIWidgetsSamples" renamed to "UIWidget.Examples"

11.2 Release 1.9.3

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

11.3 Release 1.9.2

- added TreeViewCustomNodeDragSupport
- added ScrollButtons
- Autocomplete: fixed problem with resizing
- Autocomplete: added SearchDelay and MinLength options
- ColorPicker: fixed incorrect display in linear colorspace
- ColorPicker: now click on palette or image will change color
- Draggable: added Horizontal and Vertical options
- Draggable: added Restriction option
- ListViewCustomDragSupport: added DeleteAfterDrop parameter
- ListView's, TileView's, TreeView's: added SetContentSizeFitter parameter
- ListView's, TileView's, TreeView's: added Navigation parameter
- ListView's, TileView's, TreeView's: added IsVisible() function to check if item is visible
- ListView's, TileView's, TreeView's: added animated scrolling to items - ScrollToTime() and ScrollToSpeed()
- ListView's, TileView's, TreeView's: Multiple renamed to MultipleSelect
- RangeSlider: added RangeSliderType; it's allow or disable handles overlay
- Resizable: fixed error with allowed directions
- Sidebar: added new animation type ScaleDownAndPush
- Spinner: fixed input parsing problem
- Splitter: added Mode option, so you can specify left and right targers, instead using previous and next siblings in hierarchy
- TreeView: added serialization support with TreeNode<T>.Serialize() and TreeNode<T>.Deserialize()
- TreeView: fixed error when deleting selected node with disabled DeselectCollapsedNodes
- TreeView: added ExpandParentNodes() and CollapseParentNodes() functions
- TreeView's DefaultItem: Filler renamed to Indentation
- Dialog, Notify, Picker, Popup: Template() renamed to Clone()

11.4 Release 1.9.1

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

11.5 Release 1.9.0

- Added AudioPlayer
- Added Calendar
- Added DatePicker
- Added DirectoryTreeView
- Added FileDialog
- Added FileListView
- Added FolderDialog
- Added PickerBool (can be used as Confirmation dialog with Yes/No/Cancel options)
- Accordion: added ResizeMethod property
- Accordion: protected Items property replaced with public DataSource property with type ObservableList<T>
- Accordion: added DisableClosed option
- ColorPicker: added Image palette, you can use it to get colors from custom Texture2D. The texture must have the Read/Write Enabled flag set in the import settings, otherwise this function will fail.
- ColorPicker: fixed bug with wrong axes with Hue palette
- Drag&Drop: added generic classes ListViewCustomDragSupport and ListViewCustomDropSupport, using them to add Drap&Drop functionality for own ListView's become more easily. Check ListViewIconsDragSupport and ListViewIconsDropSupport as reference (ignore TreeNode region).
- EasyLayout: fixed "dirty" scene bug when using FitContainer or ShrinkOnOverflow
- ListView's: DataSource can be safely used from other threads
- ListView's: added GroupedListView sample
- ListView's: added .Select(int index, bool raiseEvents) function, you can use it to select items without raising events
- ListView's: added Owner field to ListViewItem (base class for any DefaultItem), it contains link to parent ListView
- ListView's: you can implement IViewData<T> to DefaultItem component class to avoid overriding ListView.SetData() function
- ListView's: added virtual properties Graphic[] GraphicsForeground and Graphic[] GraphicsBackground to ListViewItem, you can them to specify grapgics for coloring, instead overriding coloring functions
- Resizable: mark events as used
- SlideBlock renamed to Sidebar
- Sidebar: added new animation types Overlay (default), Push, Uncover, ScaleDown, SlideAlong, SlideOut, Resize
- Sidebar: added AnimateWithLayout option for Resize animation, use it if you need more than one Sidebar with Resize on same Content object
- Spinner: added AllowHold option, so you can disable increasing/decreasing value during pointer hold
- Switch: added .SetStatus(bool value), you can change state without raising corresponding events
- TileView's: added TileViewCustomSize

- Tooltip: added UnscaledTime option
- TreeNode: added RootNode property, used to check if nodes belong to same tree
- TreeView's and TreeNode: Nodes type change from IObservableList<TreeNode<TItem>> to ObservableList<TreeNode<TItem>>
- TreeView: added SelectedNodes property
- TreeView: added DeselectCollapsedNodes property, enabled by default
- TreeView: added .Node2Index(TreeNode<TItem> node) function
- TreeView: added .SelectNode(TreeNode<TItem> node) and .SelectNodeWithSubnodes(TreeNode<TItem> node) functions
- TreeViewDataSource: fixed incorrect branch bug (thanks to Heiko Berres)
- ProgressBar: added SpeedType option

11.6 Release 1.8.5

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

11.7 Release 1.8.4

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

11.8 Release 1.8.3

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

- Spinner, SpinnerFloat - fixed overflow exception
- Resizable - added corners directions for resize
- ListView's - added FadeDuration for colors change

11.9 Release 1.8.2

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

11.10 Release 1.8.0

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

- TileView: added TileViewScrollRectFitter, ScrollRect will be resized to display whole number of items.

11.11 Release 1.7.4

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

11.12 Release 1.7.2

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

11.13 Release 1.7.0

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

11.14 Release 1.6.5

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

11.15 Release 1.6.0

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

11.16 Release 1.5.0

- Added TileView

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

11.17 Release 1.4.2

- Added ListViewCustomHeight (support items of variable heights)

11.18 Release 1.4.1

- Added CenteredSlider.

11.19 Release 1.4

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

11.20 Release 1.3

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

11.21 Release 1.2

- Added Dialog
- Added Draggable

11.22 Release 1.1

- Added Notify
- Added EasyLayout

11.23 Release 1.0

- Initial release