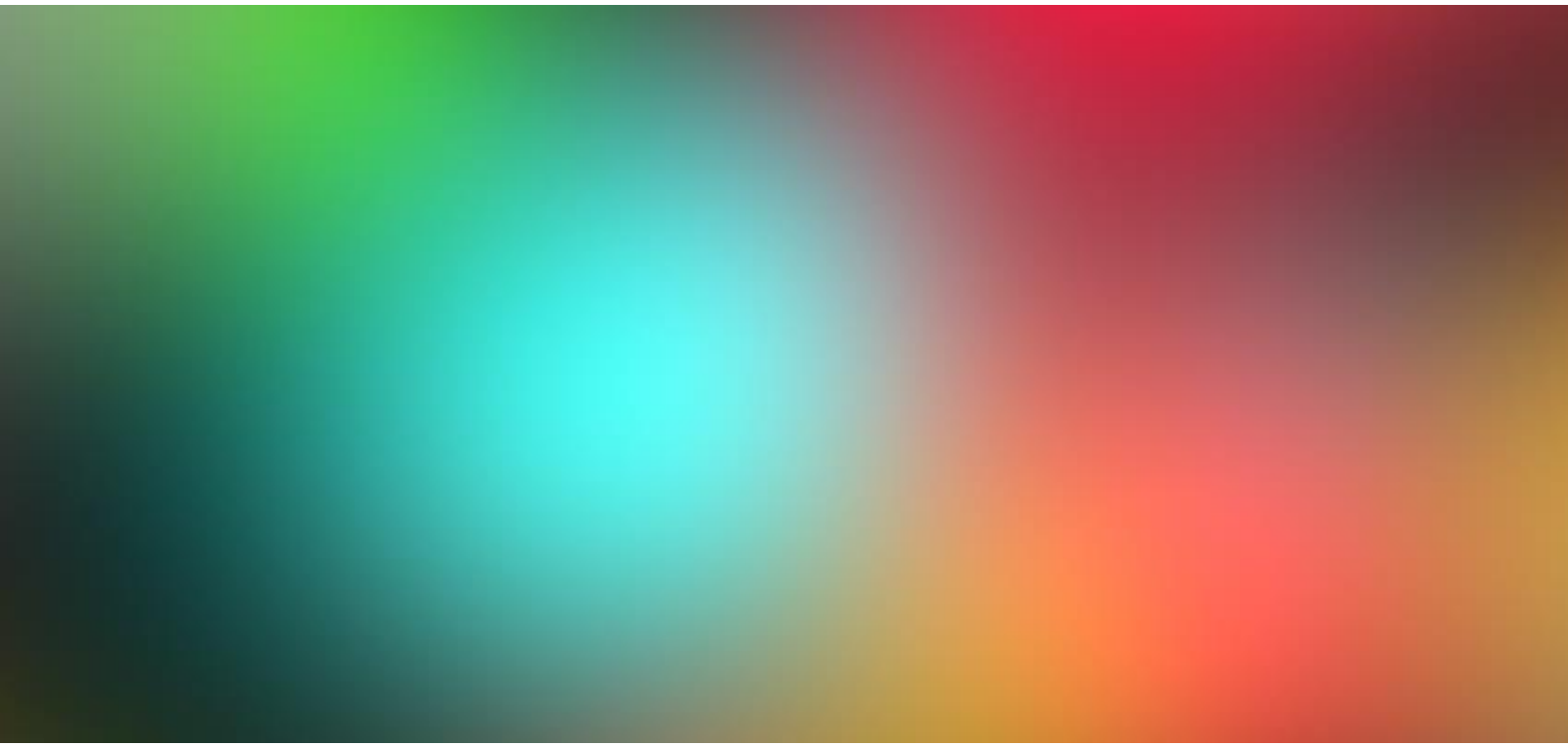

Tree View

Ver.1 14.07.2016

Introduction

TreeView is a Unity UI control that can be used to represent hierarchical data. TreeView implements drag & drop, databinding, selection operations and events and are highly customizable. There are also two base clases ItemsControl and ItemContainer which can be used to implement your own items control.

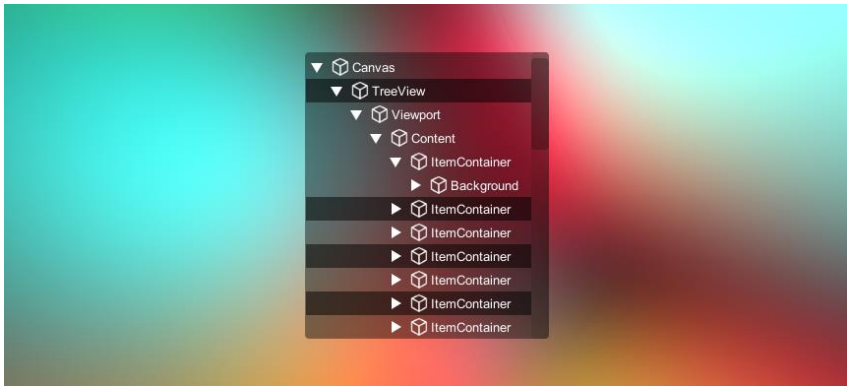

Fig.1 – Tree View Control

Features

- Data Binding,
- Drag & Drop;
- Multiselect & Range Selection
- Auto Scroll;
- Items Removal;
- Highly customizable;

Package Structure

TreeView control located in **Assets/Battlehub/UIControls**

Package organized as following:

/Scripts – for runtime scripts

/Prefabs for prefabs

/Sprites tree view graphics

/Demo contains everything related to demoscene

TreeViewDemo

Located in Assets/Battlehub/UIControls/Demo/Scripts/TreeViewDemo.cs

Minimal setup to get TreeView working:

0) Create data item

```
class MyCustomData
{
    public int childCount;
    public string name;
}
```

1) Subscribe to events in Start method

```
private void Start()
{
    ...
    //subscribe to events
    TreeView.ItemDataBinding += OnItemDataBinding;
    ...
}
```

2) Unsubscribe OnDestroy

```
private void OnDestroy()
{
    ...
    //unsubscribe
    TreeView.ItemDataBinding -= OnItemDataBinding;

 }
```

3) Implement ItemDataBinding event handler

```
private void OnItemDataBinding(object sender, TreeViewItemDataBindingArgs e)
{
    MyCustomData dataItem = e.Item as MyCustomData;
    if (dataItem != null)
    {
        //Display dataItem.name using UI.Text
        Text text = e.ItemPresenter.GetComponentInChildren<Text>(true);
        text.text = dataItem.name;

//And specify whether data item has children (to display expander arrow if needed)
        e.HasChildren = dataItem.childCount > 0;
    }}
```

4) Set Items property

```
    List<MyCustomData> dataItems = ...

    //Bind data items
    TreeView.Items = dataItems;
```

ItemsControl

Located in Assets/Battlehub/UIControls/Scripts/ItemsControl.cs.

Base class for TreeView

```
public class ItemsControl<TDataBindingArgs> : MonoBehaviour, IPointerDownHandler,
       IDropHandler where TDataBindingArgs : ItemDataBindingArgs, new()
    {
        //Drag & Drop Events
        public event EventHandler<ItemDragArgs> ItemBeginDrag;
        public event EventHandler<ItemDropArgs> ItemDrop;
        public event EventHandler<ItemDragArgs> ItemEndDrag;

        //Raise when data for ItemContainer required
        public event EventHandler<TDataBindingArgs> ItemDataBinding;

        //Selection Changed
        public event EventHandler<SelectionChangedEventArgs> SelectionChanged;

        //Item Removed
        public event EventHandler<ItemsRemovedArgs> ItemsRemoved;

        //Key bindings
        public KeyCode MultiselectKey = KeyCode.LeftControl;
        public KeyCode RangeselectKey = KeyCode.LeftShift;
        public KeyCode RemoveKey = KeyCode.Delete;

        //Is Drag & Drop allowed
        public bool CanDrag = true;


        //GameObject with ItemsContainer script (or with ItemsContainer derived class)
        [SerializeField]
        private GameObject ItemContainerPrefab;

        //Layot Panel
        public Transform Panel;

        //Raycasting Camera (used if Canvas.RenderMode == RenderMode.WorldSpace)
        public Camera Camera;

        //Scroll Speed (when item dragged out of ScrollViewer content area)
        public float ScrollSpeed = 100;

        //Set of data items
        public IEnumerable Items { get; set; }

        //items count
        public int ItemsCount { get; }


        //Selected Items count
        public int SelectedItemsCount { get; }

        //Set of selected items
        public IEnumerable SelectedItems { get; set;}

        //First Selected item
        public object SelectedItem { get; set;}

        //Index of first Selected item (-1 if no items selected)
        public int SelectedIndex { get; set;}
```

```
    //Get index of data item
    public int IndexOf(object obj)

    //Get Item Container for dataitem
    public ItemContainer GetItemContainer(object obj)

    //Get Item Container for last dataitem
    public ItemContainer LastItemContainer()

    //Get Item Container by index
    public ItemContainer GetItemContainer(int siblingIndex)

    //Add data item (if you have a collection of items use Items property instead)
    public ItemContainer Add(object item)

    //Insert data item (if you have a collection of items use Items property instead)
    public ItemContainer Insert(int index, object item)

    //Remove data item
    public void Remove(object item)

    //Remove data item by index
    public void RemoveAt(int index)
```

ItemContainer

Located in Assets/Battlehub/UIControls/Scripts/ItemsContainer.cs

Base class for data item representation component

```
[RequireComponent(typeof(RectTransform), typeof(LayoutElement))]
public class ItemContainer : MonoBehaviour, IPointerDownHandler, IPointerUpHandler,
         IPointerEnterHandler, IPointerExitHandler, IBeginDragHandler,
         IDragHandler, IDropHandler, IEndDragHandler
{
    //Is Drag & Drop allowed?
    public bool CanDrag = true;

    //Events
    public static event EventHandler Selected;
    public static event EventHandler Unselected;
    public static event ItemEventHandler PointerDown;
    public static event ItemEventHandler PointerUp;
    public static event ItemEventHandler PointerEnter;
    public static event ItemEventHandler PointerExit;
    public static event ItemEventHandler BeginDrag;
    public static event ItemEventHandler Drag;
    public static event ItemEventHandler Drop;
    public static event ItemEventHandler EndDrag;

    //ItemContainer's LayoutElement
    public LayoutElement LayoutElement { get; }

    //ItemContainer's RectTransform
    public RectTransform RectTransform { get; }

    //Is Item Container selected
    public virtual bool IsSelected { get; set; }

    //Data Item bound to Item Container
    public object Item { get; set; }
```

ItemDropMarker

Located in Assets/Battlehub/UIControls/Scripts/ItemDropMarker.cs

Item Drop Marker is used to highlight item drop location.

ItemDropMarker could be in one of the states specified by ItemDropAction enum.

```
public enum ItemDropAction
{
   None,
   SetLastChild,
   SetPrevSibling,
   SetNextSibling
}
```

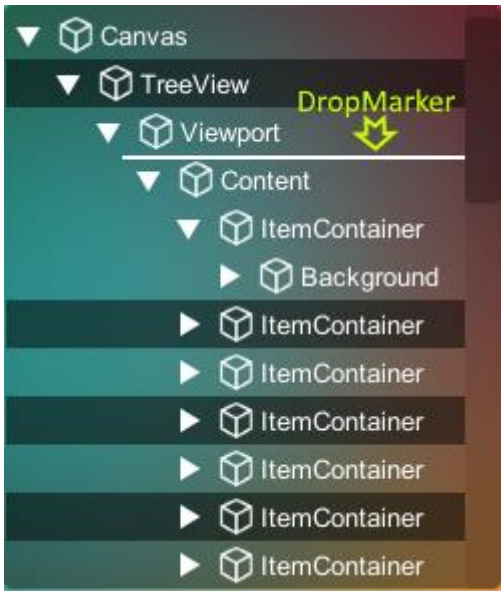

Fig.11 ItemDropMarker

TreeView

Located in Assets/Battlehub/UIControls/Scripts/TreeView.cs

Prefab Assets/Battlehub/UIControls/Prefabs/TreeView.prefab

TreeView supports multiselection, drag & drop and delete item's operation

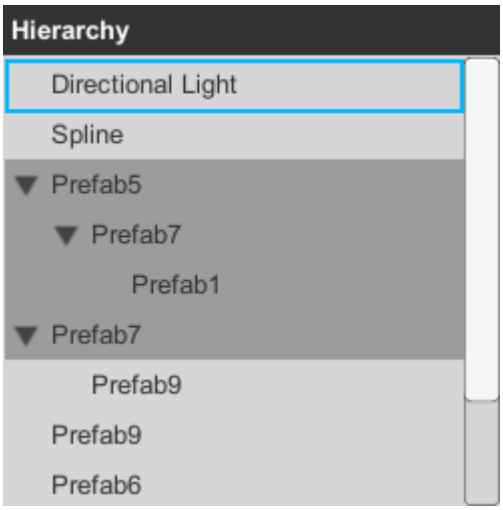

Fig.12 TreeView

```
public class TreeView : ItemsControl<TreeViewItemDataBindingArgs>
{
        //Raised when item is expanded
        public event EventHandler<ItemExpandingArgs> ItemExpanding;

        //Indent between treeview levels
        public int Indent = 20;

        //Add child data item to tree view
        public void AddChild(object parent, object item)

        //Change parent of item
        public void ChangeParent(object parent, object item)

        //Expand TreeViewItem
        public void Expand(TreeViewItem item)

        //Collapse TreeViewItem
        public void Collapse(TreeViewItem item)
```

TreeViewItem

Located in Assets/Battlehub/UIControls/Scripts/TreeViewItem.cs

Prefab Assets/Battlehub/UIControls/Prefabs/TreeViewItem.prefab

```
public class TreeViewItem : ItemContainer
{
    //Raised when item's parent changed
    public static event EventHandler<ParentChangedEventArgs> ParentChanged;

    //Accumulated indent
    public int Indent { get; }

    //Parent TreeViewItem
    public TreeViewItem Parent { get; set; }

    public override bool IsSelected { get; set; }

    //Whether tree view item can be expanded (if true expander arrow is visible)
    public bool CanExpand { get;set;}

    //Is tree view item expanded
    public bool IsExpanded { get; set; }

    //Whether tree view item has children
    public bool HasChildren { get; }

    //Is tree view item is descendant of other tree view item;
    public bool IsDescendantOf(TreeViewItem parent)

    //Returns first child if exists
    public TreeViewItem FirstChild()

    //Returns next child if exists
    public TreeViewItem NextChild(TreeViewItem currentChild)

    //Returns last child
    public TreeViewItem LastChild()
```

Limitations and Issues

- **Not tested with collections more than 100 elements.**
- **Does not support data item virtualization**

Support

If you have any questions, suggestions, you want to talk or you have some issues please send mail to Vadim.Andriyanov@outlook.com or Battlehub@outlook.com.