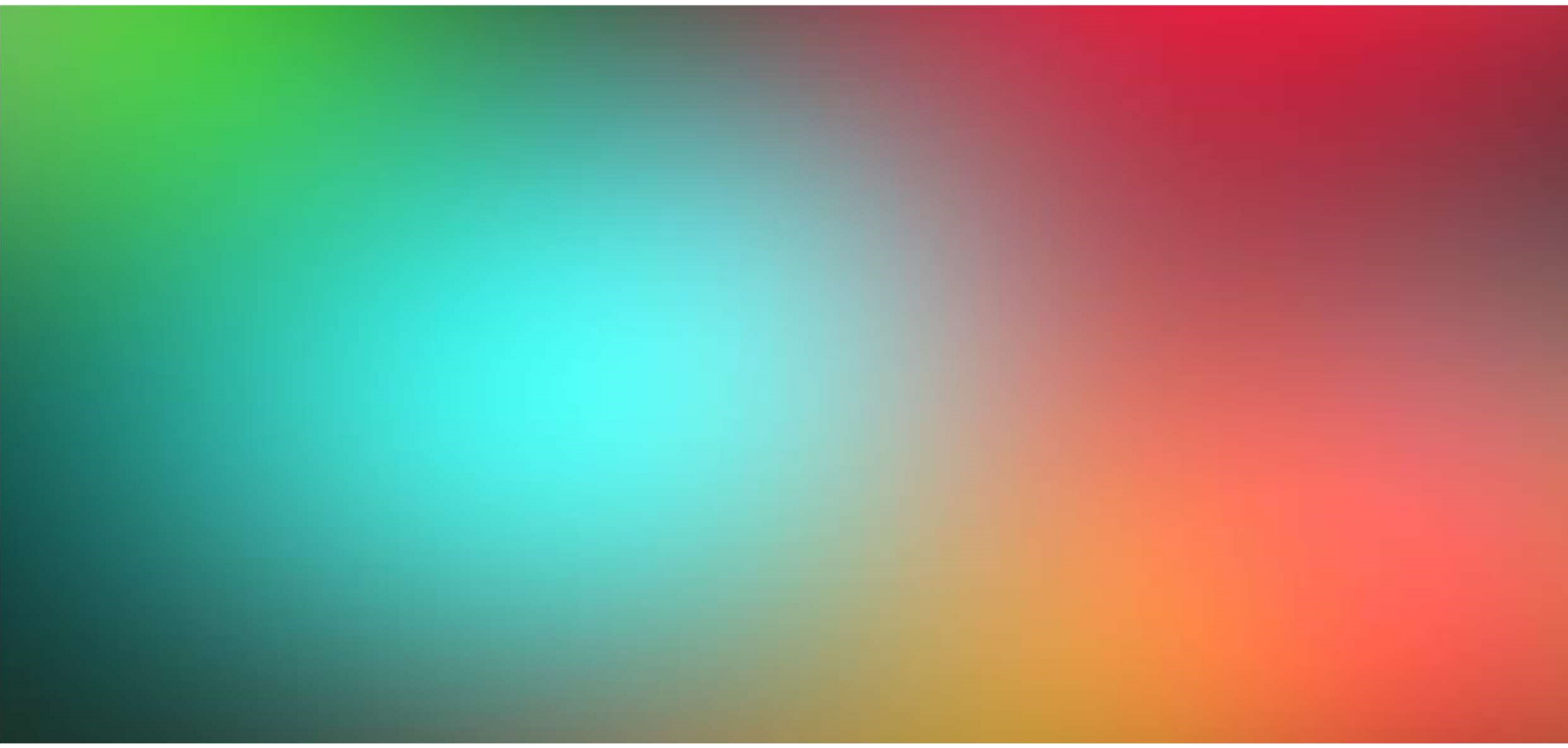

Virtualizing Tree View

Ver.1 30.11.2017

Introduction

Virualizing TreeView is a Unity UI control that can be used to represent hierarchical data. Virtualizing TreeView implements drag & drop, databinding, selection events and are highly customizable. There are also two base classes VirtualizingItemsControl and VirtualizingItemContainer which can be used to implement your own items control. Main advantage of **VirtualizingTreeView** over TreeView is **support for large (1000+) collections of data items**

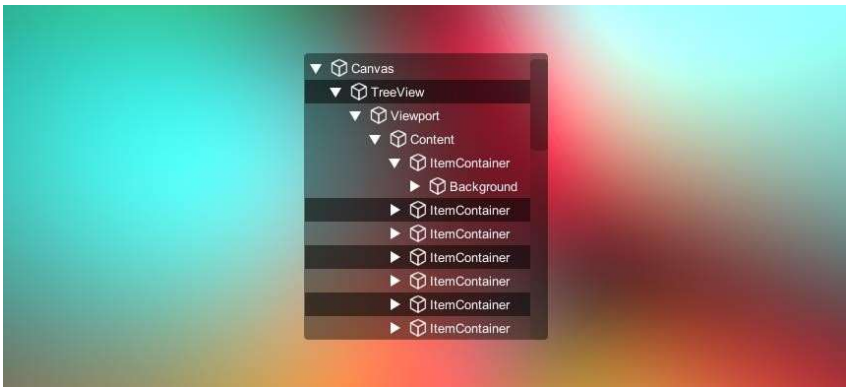

Fig.1 – Virtualizing Tree View Control

Features

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

Package Structure

Package organized as following:

/Scripts/VirtualizingTreeView – implementation scripts

/Prefabs for prefabs

/Sprites tree view graphics

/Demo contains everything related to demoscene

TreeViewDemo

Located in Assets/Battlehub/UIControls/Demo/Scripts/

Minimal setup to get TreeView working:

1) Create data item

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

2) Subscribe to events in Start method

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

3) Unsubscribe OnDestroy

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

4) Implement ItemDataBinding event handler

```
private void OnItemDataBinding(object sender,
                               VirtualizingTreeViewItemDataBindingArgs 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;
```

Virtualizing ScrollRect

Located in Assets/Battlehub/UIControlsVirtualizingTreeView/Scripts/

This class re-use ‘ui containers’ to effectively represent large data item collections. UI containers created for small visible portion of data items.

```
public class VirtualizingScrollRect : ScrollRect
```

Events:

```
//Raised when new portion of data is required
public event DataBindAction ItemDataBinding;
```

Properties:

```
//Gets or sets list of data items.
public IList Items { get; set; }

//Get count of data items.
public int ItemsCount{ get; }
```

Methods:

```
//Returns true if ScrollRect is parent of Transform
public bool IsParentOf(Transform child);

//Insert data item at position determided by index
public void InsertItem(int index, object item, bool raiseDataBindingEvent = true)

//Remove data items with given indices
public void RemoveItems(int[] indices, bool raiseItemDataBindingEvent = true)

//Get ‘ui container’ for data item
public RectTransform GetContainer(object obj)

//Get first visible ‘ui container’
public RectTransform FirstContainer()

//Get last visible ‘ui container’
public RectTransform LastContainer()

//Execute action for each visible ‘ui container’
public void ForEachContainer(System.Action<RectTransform> action)
```

VirtualizingItemsControl

Located in Assets/Battlehub/UIControlsVirtualizingTreeView/Scripts/

This class implements drag&drop and selection functionality. It also serve as base class for items controls such as listbox and treeview.

```
public class VirtualizingItemsControl : MonoBehaviour,
        IPointerDownHandler, IDropHandler
```

Events:

```
//Raised on begin drag
public event EventHandler<ItemArgs> ItemBeginDrag;

//Raised on before drop
public event EventHandler<ItemDropCancelArgs> ItemBeginDrop;

//Raised on after drop
public event EventHandler<ItemDropArgs> ItemDrop;

//Raised when drag & drop operation completed
public event EventHandler<ItemArgs> ItemEndDrag;

//Raised when data items selected or unselected
public event EventHandler<SelectionChangedArgs> SelectionChanged;

//Raised on data item double click
public event EventHandler<ItemArgs> ItemDoubleClick;

//Raised before data items removed
public event EventHandler<ItemsCancelArgs> ItemsRemoving;

//Raised after data items removed
public event EventHandler<ItemsRemovedArgs> ItemsRemoved;
```

Public Fields:

```
//Key codes
public KeyCode MultiselectKey = KeyCode.LeftControl;
public KeyCode RangeselectKey = KeyCode.LeftShift;
public KeyCode SelectAllKey = KeyCode.A;
public KeyCode RemoveKey = KeyCode.Delete;

//Behaviour settings
public bool SelectOnPointerUp = false;
public bool CanUnselectAll = true;
public bool CanEdit = true;
public bool CanDrag = true;
public bool CanReorder = true;

//UI settings
public bool ExpandChildrenWidth = true;
public bool ExpandChildrenHeight;
public float ScrollSpeed = 100;
```

Properties:

```
//Get current active drop target
public object DropTarget { get; }

//Get current drop action. None, SetLastChild, SetPrevSibling, SetNextSibling
public ItemDropAction DropAction { get; }

//Get selected data items count
public int SelectedItemsCount { get; }

//Get or set selected data items
public virtual IEnumerable SelectedItems { get; set; }

//Get or set selected data item
public object SelectedItem { get; set; }

//Get or set selected data item
public int SelectedIndex { get; set; }

//Get or set data items
public IEnumerable Items { get; set; }
```

Methods:

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

//Get index of data item
public virtual void SetIndex(object obj, int newIndex)

//Get last item container data
public ItemContainerData LastItemContainerData()

//Get visible item container for data item
public VirtualizingItemContainer GetItemContainer(object item)

//Get item container data for data item
public ItemContainerData GetItemContainerData(object item)

//Get item container data with siblingIndex
public ItemContainerData GetItemContainerData(int siblingIndex)

//Methods for external item drag and drop opertion
public void ExternalBeginDrag(Vector3 position)
public void ExternalItemDrag(Vector3 position)
public void ExternalItemDrop()

//Remove selected items
public void RemoveSelectedItems()

//Add data item to the end
public ItemContainerData Add(object item)

//Insert data item at index
(if you insert more than one item consider using Items property)
public virtual ItemContainerData Insert(int index, object item)

//Move data item
public void SetNextSibling(object sibling, object nextSibling)
public void SetPrevSibling(object sibling, object prevSibling)

//Remove data item
public virtual void Remove(object item)
//Get data item at index
 public object GetItemAt(int index)
```

VirtualizingTreeViewDropMarker。

Located in Assets/Battlehub/UIControls/Scripts/ VirtualizingTreeView

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
}
```

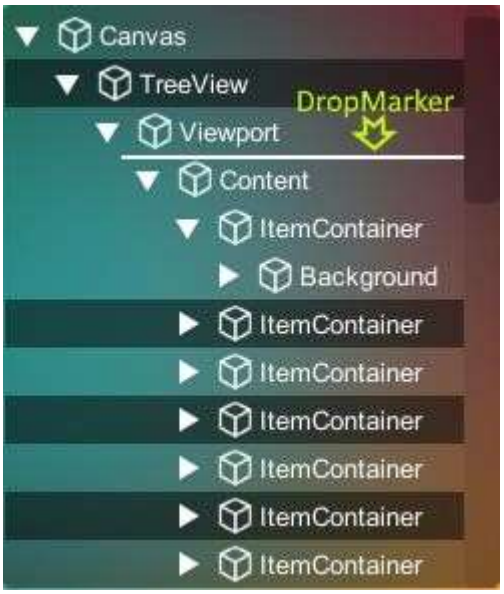

Fig.11 ItemDropMarker

VirtualizingTreeView

Located in Assets/Battlehub/UIControls/VirtualizingTreeView/

Assets/Battlehub/UIControls/Prefabs/VirtualizingTreeView.prefab

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

```
public class VirtualizingTreeView :
VirtualizingItemsControl<VirtualizingTreeViewItemDataBindingArgs>
```

Events:

```
//Raised when item expanding and children data is required to be supplied
public event EventHandler<VirtualizingItemExpandingArgs> ItemExpanding;
```

Methods:

```
//Add child data item
public void AddChild(object parent, object item);

//Remove child data item. (isLastChild sholud be set to true for last child)
public void RemoveChild(object parent, object item, bool isLastChild)

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

//Returns true if data item is expanded
public bool IsExpanded(object item)

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


//Collapse data item
public void Collapse(object item)
```

VirtualizingTreeViewItem

Located in Assets/Battlehub/UIControls/VirtualizingTreeView/

Assets/Battlehub/UIControls/Prefabs/VirtualizingTreeViewItem.prefab

```
public class VirtualizingTreeViewItem : VirtualizingItemContainer
```

Properties:

```
//Get TreeView item indent
public float Indent { get; }

//Get or set data item
public override object Item { get; set; }

//Get or set tree view item data (state of ui container)
public TreeViewItemContainerData TreeViewItemData { get; set; }


//Get or set tree view item parent data (state of parent ui container)
public TreeViewItemContainerData Parent { get; set; }


//Gets or sets whether item can be expanded (this property is driven by tree view.
If you set this property you might not get desired results)
public bool CanExpand { get; set; }

//Expand or collapse TreeViewItem
public bool IsExpanded { get; set; }


//Returns true if TreeViewItem has children
public bool HasChildren { get; }
```

Methods:

```
//Get first child of TreeViewItem
public TreeViewItemContainerData FirstChild();

//Get next child
public TreeViewItemContainerData NextChild(TreeViewItemContainerData currentChild);

//Get last child
public TreeViewItemContainerData LastChild();

//Get last descendant
public TreeViewItemContainerData LastDescendant()
```

Limitations and Issues

- **Has problems with horizontal scrollbar (Auto Hide And Expand visibility is not currently supported)**

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.