Ver.1 14.06.2016

Introduction

Runtime Editor Basics is a set of scripts, which will help you to implement runtime scene/level editor. Package divided into several parts, could be used together or independently.

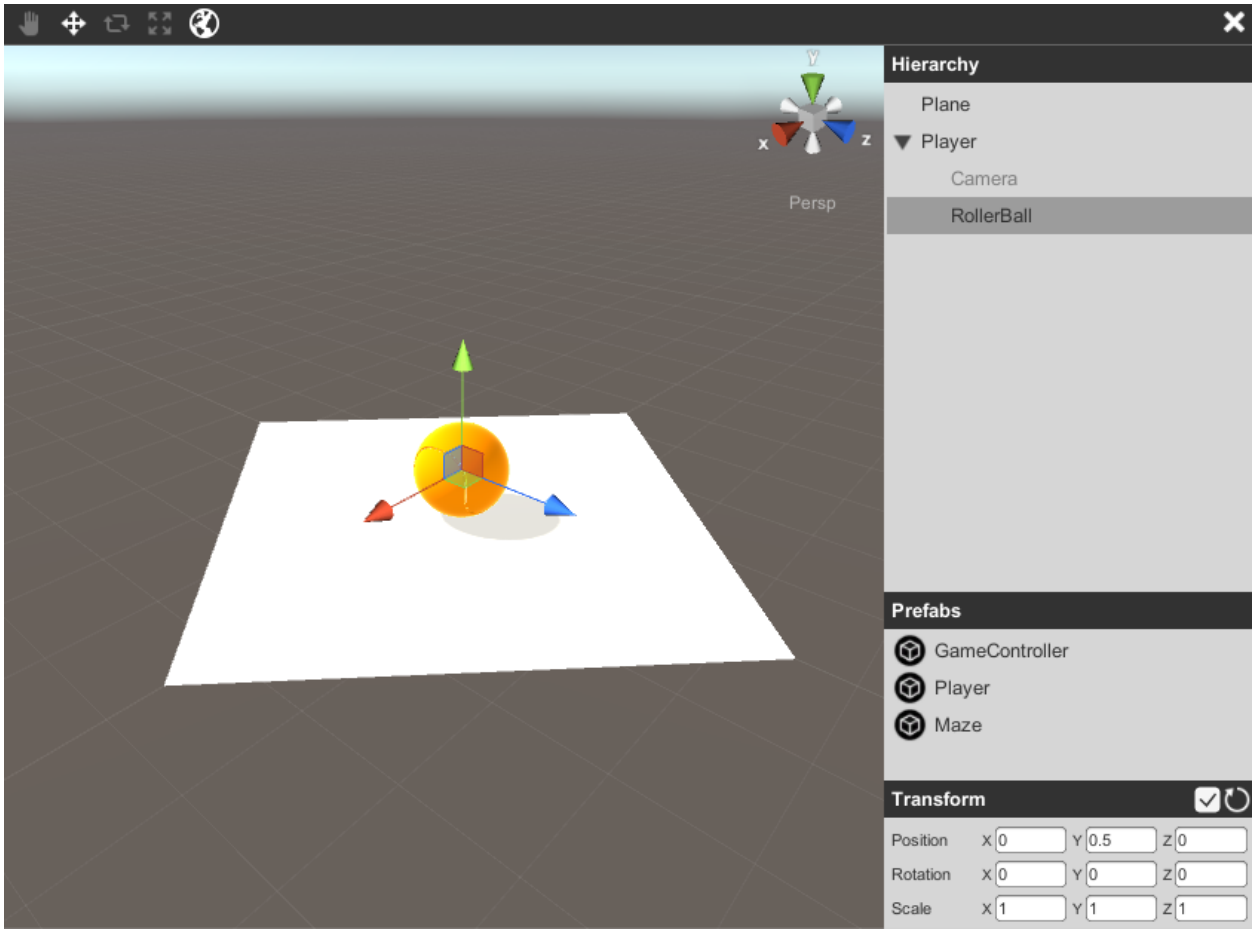

Fig.1 - Runtime Editor

Features

- Positon gizmo,
- Rotation gizmo;
- Scale gizmo;
- Scene gizmo;
- Grid;
- Scene view navigation;
- Orthographic\Perspective view;
- Hierarchy Treeview;
- Preabs Listbox;
- Transform component;
- Global\Local coordinates;
- Assembled editor Prefab;
- Very easy process of exposing to editor gameobjects and prefabs

Package Structure

Runtime Editor located in **Assets/Battlehub/RTEditor folder,**
Gizmos in **Assets/Battlehub/RTHandles,**
TreeView and ListBox in **Assets/Battlehub/UIControls**
Helper classes in **Assets/Battehub/Utils**
Demo scene in **Assets/Battlehub/RTEditor/Demo.unitypackage**

Each folder organized as following:

/Scripts – for runtime scripts
/Scripts/Editor for editor scripts
/Prefabs for prefabs
/Shader/Resources for shaders
/Demo (if present) contains everything related to demoscene

Menu

Runtime editor menu is very simple. There are three menu items:

1. **Create menu item** creates runtime editor using prefab located in Battlehub/RTEditor/Prefabs/RuntimeEditor.prefab
2. **Expose to Editor** makes game object or prefab visible to editor
3. **Hide from Editor** hides game object or prefab from editor

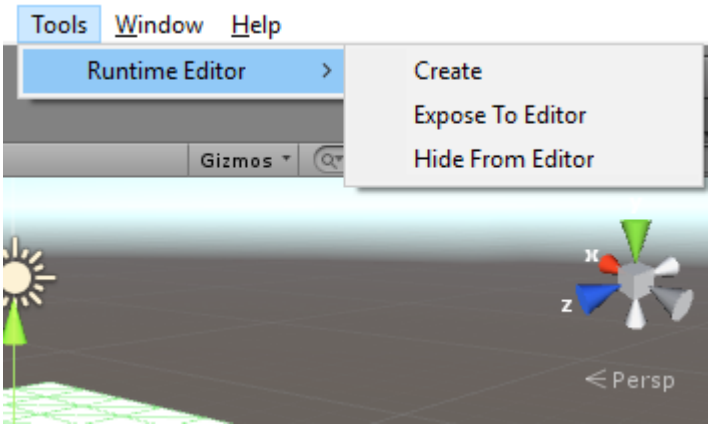

Fig.2 Menu

Gizmos

There are four gizmos included in RuntimeEditorBasics package: Position, Rotation, Scale and Scene gizmo. They behaves almost identical to their equivalents in unity editor. Gizmos, Grid, Rendering classes and all required shaders can be found in **Assets/Battlehub/RTHandles folder.**

Position, Rotation and Scale Editor windows allows you to choose Raycasting Camera, Selection Margin (in screen space coordinates), Target objects, Grid Size, and key which will switch position gizmo to “Snap to Grid mode” (Left Control by default).

Scene Gizmo Editor windows allows you to choose Scene Camera, Pivot Point(to rotate Scene Camera around), Size of Gizmo.

Scene Gizmo could raise following events:

- Orientation Changing;
- Orientation Changed;
- Projection Changed;

Note: Scene gizmo always aligned to the top right corner of the screen

Position Gizmo

The procedure of creation of position gizmo (like all other gizmos) is quite simple:

1) Create Empty **GameObject**;
2) Assign **Assets/Battlehub/RTHandles/Scripts/PositionHandle.cs** script to it.

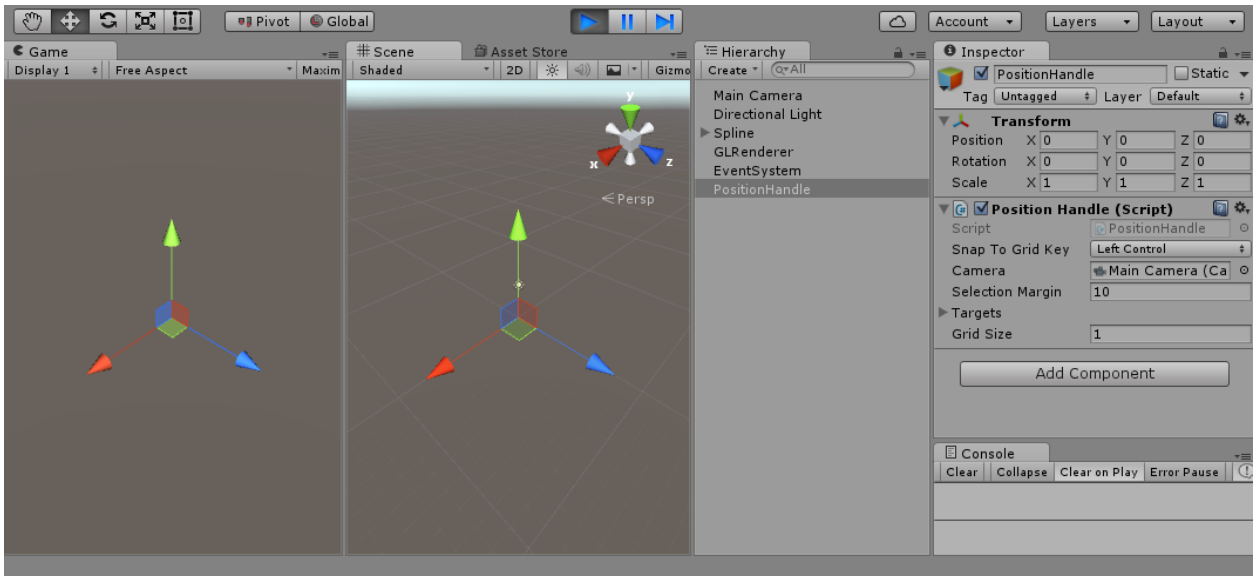

Fig.3 Runtime Position Gizmo

Rotation Gizmo

To use runtime gizmo do following:

1) Create Empty **GameObject**;
2) Assign **Assets/Battlehub/RTHandles/Scripts/RotationHandle.cs** script to it.

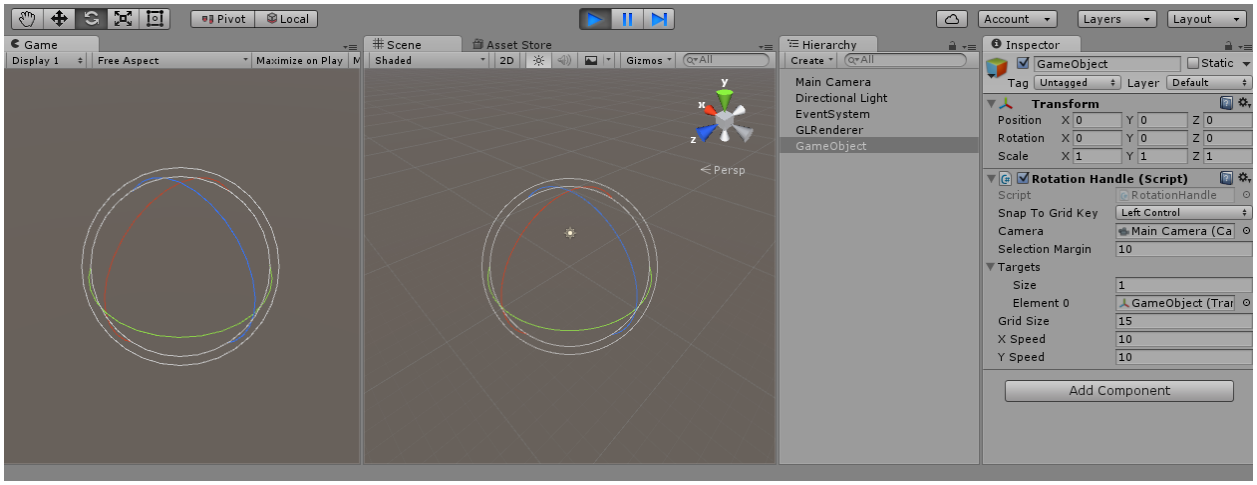

Fig.4 Runtime Rotation Gizmo

Scale Gizmo

To use scale gizmo do following:

1) Create Empty **GameObject**;
2) Assign **Assets/Battlehub/RTHandles/Scripts/ScaleHandle.cs** script to it.

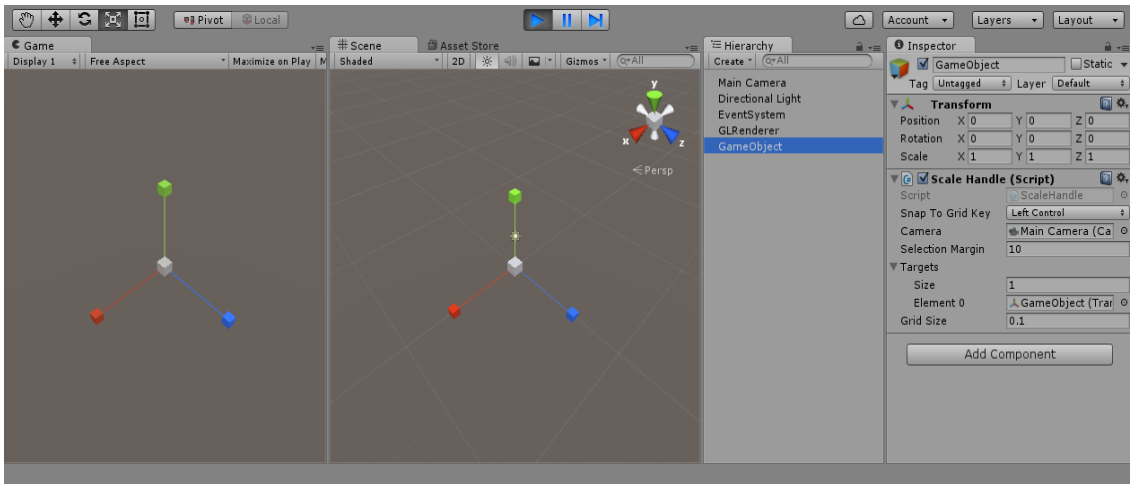

Fig.5 Runtime Scale Gizmo

Scene Gizmo

To use scale gizmo do following:

1) Create Empty **GameObject**;
2) Assign **Assets/Battlehub/RTHandles/Scripts/ScaleHandle.cs** script to it.

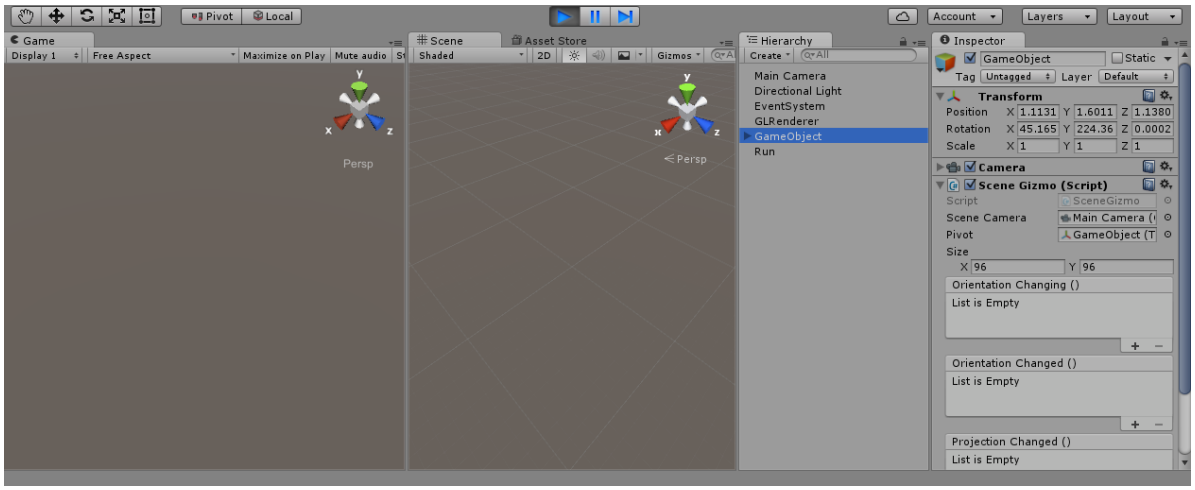

Fig.6 Runtime Scene Gizmo

Grid

To use grid do following:

1) Create Empty **GameObject**;
2) Assign **Assets/Battlehub/RTHandles/Scripts/Scripts/Grid.cs** script to it

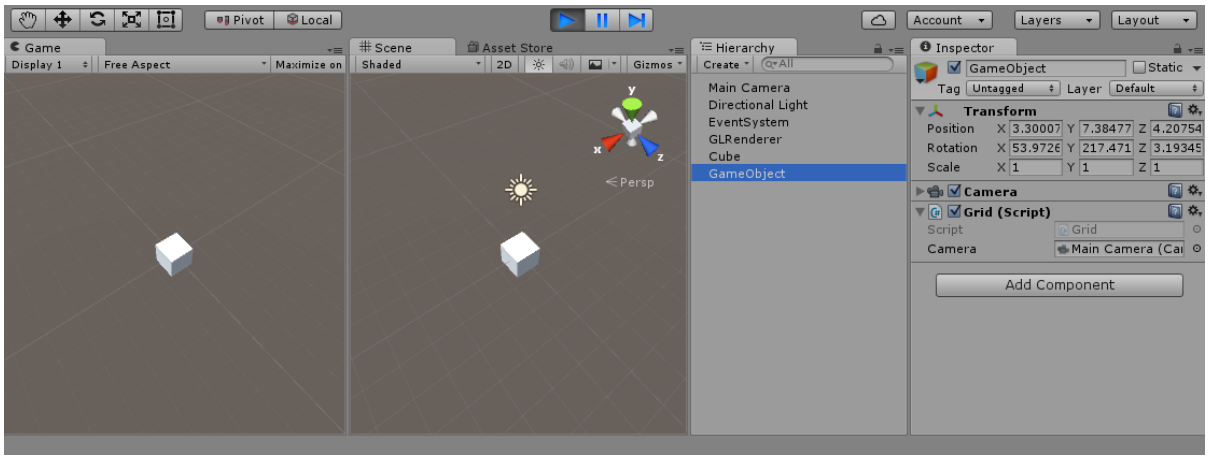

Fig.7 Runtime Grid

IGL

Located in Assets/Battlehub/RTHandles/Scripts/GLRenderer.cs

Implement this interface to make object available to GLRenderer.

```
public interface IGL {
      void Draw();
}
```

GLRenderer

Located in Assets/Battlehub/RTHandles/Scripts/GLRenderer.cs

GLRenderer is a singleton used by GLCamera script to render all registered objects. Register object for rendering by calling `public void Add(IGL gl)` method. Cancel object rendering by calling `public void Remove(IGL gl)` method

Example usage:

```
private void OnEnable() {
    if (GLRenderer.Instance != null) {
        GLRenderer.Instance.Add(this);
    }
}

private void OnDisable(){
    if (GLRenderer.Instance != null) {
        GLRenderer.Instance.Remove(this);
    }
}
```

GLCamera

Located in Assets/Battlehub/RTHandles/Scripts/GLCamera.cs

Attach this script to any camera and GL graphics will be renderer with this camera

```
[ExecuteInEditMode]
public class GLCamera : MonoBehaviour
{
    private void OnPostRender()
    {
        if(GLRenderer.Instance != null)
        {
            GLRenderer.Instance.Draw();
        }
    }
}
```

RuntimeHandles

Located in Assets/Battlehub/RTHandles/Scripts/RuntimeHandles.cs

This class contains rendering code and helper methods for all gizmos and grid.

Get Screen Scale factor for given world position and camera:

```
public static float GetScreenScale(Vector3 position, Camera camera)
```

Draw Position Gizmo with given world position, rotation and selectedAxis:

```
public static void DoPositionHandle(Vector3 position, Quaternion rotation,
       RuntimeHandleAxis selectedAxis = RuntimeHandleAxis.None)
```

Draw Rotation Gizmo with given world position, rotation and selectedAxis:

```
public static void DoRotationHandle(Quaternion rotation, Vector3 position,
      RuntimeHandleAxis selectedAxis = RuntimeHandleAxis.None)
```

Draw Rotation Gizmo with given world position, rotation, scale and selectedAxis:

```
public static void DoScaleHandle(Vector3 scale, Vector3 position,
      Quaternion rotation,
      RuntimeHandleAxis selectedAxis = RuntimeHandleAxis.None)
```

Draw Scene Gizmo with given world position, rotation. Selection specified using Vector3 selection arg, where Vector3.zero means “not selected”. xAlpha, yAlpha and zApha args are used to fade in and fade out corresponding axes. gizmoScale = 1 is used to render Scene Gizmo in 96x96 rectangle

```
public static void DoSceneGizmo(Vector3 position, Quaternion rotation,
      Vector3 selection, float gizmoScale,
      float xAlpha = 1.0f, float yAlpha = 1.0f, float zAlpha = 1.0f)
```

Get far plane for grid rendering camera

```
public static float GetGridFarPlane()
```

Draw Grid

```
public static void DrawGrid()
```

BaseHandle

Located in Assets/Battlehub/RTHandles/Scripts/BaseHandle.cs

Base class for object manipulation gizmos. Contains basic HitTesting method, Target objects position synchronization method and basic drag and drop method.

RuntimeTool And PivotRotation

Located in Assets/Battlehub/RTHandles/Scripts/BaseHandle.cs

These enumerations defined as following:

```
public enum RuntimeTool
{
    None,
    Move,
    Rotate,
    Scale,
    View,
}

public enum RuntimePivotRotation
{
    Local,
    Global
}
```

RuntimeTools

Located in Assets/Battlehub/RTHandles/Scripts/BaseHandle.cs

This static class is used for the following purposes:

- Specify Current Tool (other tools will be blocked)
- Pivot Rotation determines rotation of coordinate system in which gizmo will be renderer (local or global)
- Determine whether input should be locked (IsLocked)
- Determine whether gizmo drag and drop operation in progress (IsDragDrop)
- Determine whether scene gizmo is selected or not;
- Raise ToolChanged and PivotRotationChanged events

```
public class RuntimeTools {
   public static event RuntimeToolChanged ToolChanged;
   public static event RuntimePivotRotationChanged PivotRotationChanged;
   public static bool IsLocked { get; set;}
   public static bool IsDragDrop { get; set; }
   public static bool IsSceneGizmoSelected { get; set; }
   public static RuntimeTool Current { get; set; }
   public static RuntimePivotRotation PivotRotation { get; set; }
}
```

RuntimeEditor UI

Located in Assets/Battlehub/RTEditor/Prefabs/RuntimeEditor.prefab is assembled runtime scene editor with all functionality available in package.

Runtime editor UI consists of following components:

- SceneView & ViewportFitter
- Header with Tools Panel and Close button
- Hierarchy TreeView
- Prefabs ListBox
- Transform panel

Scene navigation and shortcuts are almost the same as in unity editor except following:

- Shift+D is used to duplicate objects;
- Right mouse button is used to pan (do the same as middle mouse button)

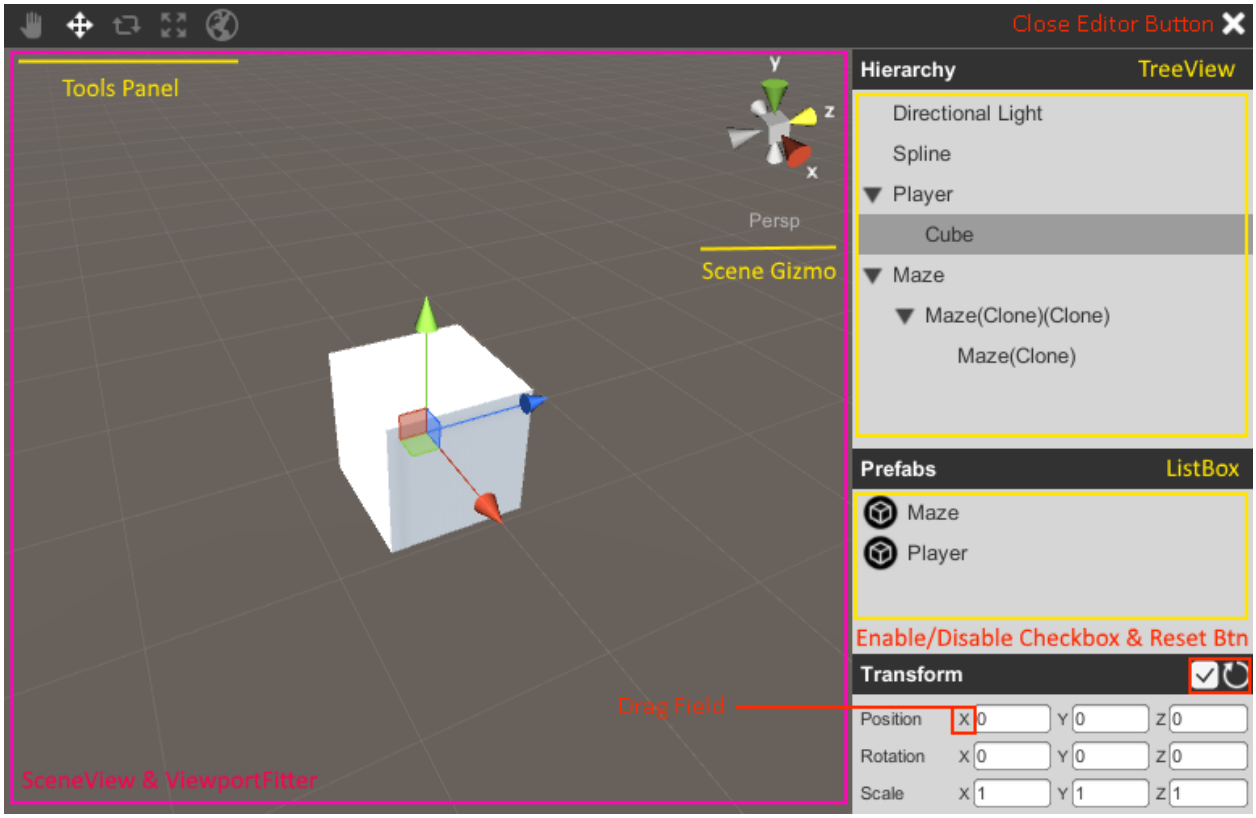

Fig.8 Runtime Editor UI

Warning: This UI is very basic, no dockable panels, no inspector and other windows available in unity editor, etc. This is not replacement of original unity editor. This is runtime editor basics, please do not expect too much.

RuntimeEditor

Located in Assets/Battlehub/RTEditor/Scripts/RuntimeEditor.cs

This class is responsible for wiring up hiding and showing editor components, scene view raycasting and selection.

Important fields:

```
//prefabs will be available in runtime prefabs panel
public GameObject[] Prefabs;
public GameObject Grid         // grid game object
public GameObject SceneGizmo   // scene gizmo
public GameObject EditButton   // button which opens editor
public GameObject CloseButton; // close editor button
public GameObject EditorRoot;  // major editor ui
public Camera SceneCamera;     // camera used for raycasting

//script responsible for scene navigation
public RuntimeSceneView SceneView;

//key codes
public KeyCode MultiselectKey = KeyCode.LeftControl;
public KeyCode RangeSelectKey = KeyCode.LeftShift;
public KeyCode DuplicateKey = KeyCode.D;
public KeyCode DuplicateKey2 = KeyCode.LeftShift;

//raycast layer for scene view selection
private LayerMask m_raycastLayerMask = 1 << 31;
private int m_raycastLayer = 31;
public int RaycastLayer { get; set; }

//whether runtime editor opened visible, active and enabled.
private bool m_isOn;
public bool IsOn { get; set; }

//editor instance
private static RuntimeEditor m_instance;
public static RuntimeEditor Instance { get { return m_instance; }}
```

RuntimeToolsPanel

Located in Assets/Battlehub/RTEditor/Scripts/RuntimeTools.cs.

This class is responsible for visual state and Tools panel input.

```
public Toggle ViewToggle;
public Toggle MoveToggle;
public Toggle RotateToggle;
public Toggle ScaleToggle;
public Toggle PivotRotationToggle;
```

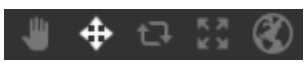

Fig.9 Tools Panel

ExposeToEditor

Located in Assets/Battlehub/RTEditor/Scripts/ExposeToEditor.cs.

Attach this script to any GameObject you want to be exposed to RuntimeEditor

```
public delegate void ExposeToEditorChangeEvent<T>(
         ExposeToEditor obj, T oldValue, T newValue);
public delegate void ExposeToEditorEvent(ExposeToEditor obj);

[DisallowMultipleComponent]
public class ExposeToEditor : MonoBehaviour
{
    public static event ExposeToEditorEvent NameChanged;
    public static event ExposeToEditorEvent TransformChanged;
    public static event ExposeToEditorEvent Awaked;
    public static event ExposeToEditorEvent Started;
    public static event ExposeToEditorEvent Enabled;
    public static event ExposeToEditorEvent Disabled;
    public static event ExposeToEditorEvent Destroyed;
    public static event ExposeToEditorChangeEvent<ExposeToEditor> ParentChanged;

    //disable object in awake method. May be useful for objects with Rigidbody
    component attached
    public bool DisableOnAwake = false;

    //only GameObjects with ExposeToEditor component considered as children
    public int ChildCount { get }
    public ExposeToEditor GetChild(int index);
    public ExposeToEditor[] GetChildren();
    public ExposeToEditor Parent
```

HierarchyItem

Located in Assets/Battlehub/RTEditor/Scripts/HierarchyItem.cs

This class is used to create links between ExposeToEditor objects in hierarchy. Suppose you have two objects exposed to editor. One of them is grandparent of another. Object between them will be HierarchyItem. HierarchyItem will track all hierarchy changes between ExposeToEditor objects. HierarchyItems are created automatically. You don't have to assign them to GameObjects.

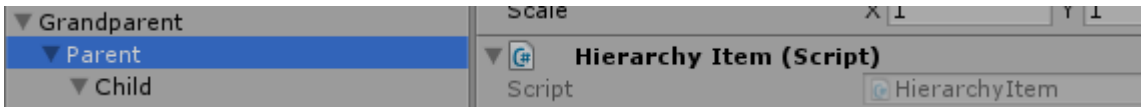

Fig.10 Hierarchy Item

RuntimeSceneView

Located in Assets/Battlehub/RTEditor/Scripts/RuntimeSceneView.cs.

Responsible for scene view navigation

```
//is pointer over scene view?
public bool IsPointerOver { get; }


// eye cursor (visible when alt key pressed and pointer is over scene view)
public Texture2D ViewTexture;

// hand cursor (visible when View Tool selected)
public Texture2D MoveTexture;

// camera used for raycasting
public Camera Camera;

// pivot transform (to rotate camera around)
public Transform Pivot;


// lock user input
public void LockInput();

// unlock user input
public void UnlockInput();
```

MouseOrbit

Located in Assets/Battlehub/RTEditor/MouseOrbit.cs.

Responsible for camera rotation around pivot.

Slightly modified version of this script.

RuntimeSelection

Located in Assets/Battlehub/RTEditor/Scripts/RuntimeSceneView.cs

This static class is limted equivalent of UnityEditor.Selection class

```
public delegate void RuntimeSelectionChanged(Object[] unselectedObjects);
public static class RuntimeSelection {
  //this event raised when selection changed
  public static event RuntimeSelectionChanged SelectionChanged;
  public static GameObject activeGameObject { get; set;}
  public static Object activeObject { get; set; }
  //unordered selection of objects including prefabs
  public static Object[] objects {set; set }
  //unordered selection of game objects including prefabs
  public static GameObject[] gameObjects { get; }
  public static Transform activeTransform;
  //change selection without SelectionChanged event
  public static void Select(Object activeGameObject, Object[] selection)
```

RuntimeHierarcy

Located in Assets/Battlehub/RTEditor/Scripts/RuntimeHierarchy.cs

This class manages hierarchy of exposed to editor GameObjects, feed them to the treeview, and handles RuntimeSelection and TreeView events.

```
public class RuntimeHierarchy : MonoBehaviour
{
   //Prefab with TreeView script
   public GameObject TreeViewPrefab;

   //Filtering by type criteria
   public System.Type TypeCriteria = typeof(GameObject);

   //Colors for disabled and enabled GameObjects
   public Color DisabledItemColor = new Color(0.5f, 0.5f, 0.5f);
   public Color EnabledItemColor = new Color(0.2f, 0.2f, 0.2f);
```

RuntimePrefabs

Located in Assets/Battlehub/RTEditor/Scripts/RuntimePrefabs.cs

This class manages hierarchy of exposed to editor Prefabs, feed them to the ListBox, and handles RuntimeSelection and ListBox events

```
public class RuntimePrefabs : MonoBehaviour
{
    //Prefab with ListBox script
    public GameObject ListBoxPrefab;

    //Filtering by type criteria
    public Type TypeCriteria = typeof(GameObject);

    //RuntimeEditor reference
    public RuntimeEditor Editor;
```

TransformComponent

Located in Assets/Battlehub/RTEditor/Scripts/TransformComponent.cs.

The main purpose of this class is to expose GameObject transform component

```
public class TransformComponent : MonoBehaviour
{
    public Toggle EnableDisableToggle;
    public GameObject TransformComponentUI;
    public InputField PositionX;
    public InputField PositionY;
    public InputField PositionZ;
    public InputField RotationX;
    public InputField RotationY;
    public InputField RotationZ;
    public InputField ScaleX;
    public InputField ScaleY;
    public InputField ScaleZ;
    public Button Reset;
```

UIControls

There are two major UI controls included in package: TreeView and ListBox;

These controls 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.

ItemsControl

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

Base class for TreeView and ListBox

```
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 (for TreeViewItem and for ListBoxItem).

```
[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
}
```

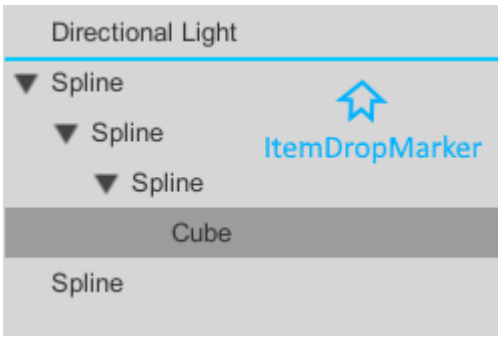

Fig.11 ItemDropMarker

ListBox

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

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

See **Assets/Battlehub/RTEditor/Scripts/RuntimePrefabs.cs** for usage example.

ListBox has same funtionality as ItemsControl and defined as following:

```
public class ListBox : ItemsControl { }
```

ListBox supports multiselection, drag & drop and delete item's operation. But in prefabs panel all these functions disabled.

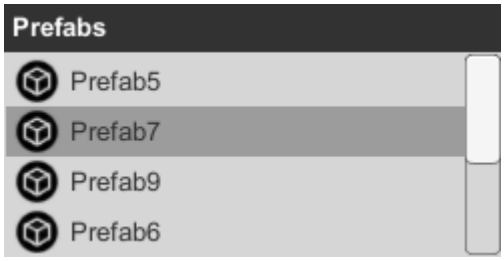

Fig.12 ListBox

ListBoxItem

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

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

ListBoxItem use Toggle to implement selected and unselected visual state

```
public override bool IsSelected
{
        get { return base.IsSelected; }
        set
        {
            if (base.IsSelected != value)
            {
                m_toggle.isOn = value;
                base.IsSelected = value;
            }
        }
 }

 protected override void AwakeOverride()
 {
        m_toggle = GetComponent<Toggle>();
        m_toggle.interactable = false;
        m_toggle.isOn = IsSelected;
  }
```

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

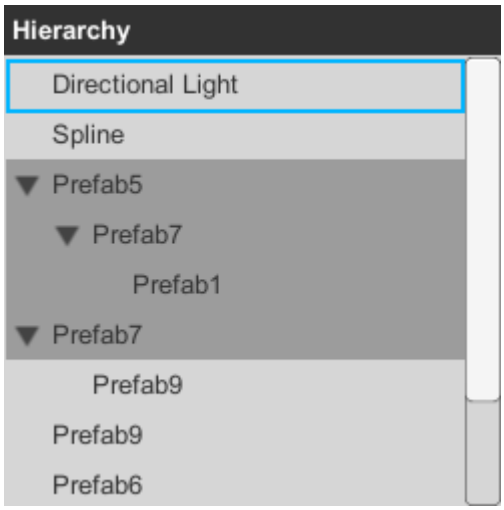

Fig.12 TreeView

```
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

This Runtime Editor is very basic, no dockable panels, no inspector and other windows available in unity editor, etc. There is lack of following important functions:

- **Undo/Redo functionality,**
- **Save/Load,**
- **Play/Edit mode (always play mode)**

Furthermore API (if can be called so) is completely different from API located in UnityEditor namespace.

This is not replacement of original unity editor, this is Runtime Editor Basics. Please do not expect too much.

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.