Editor Handles Attributes

version 2.1.0

Introduction

Making a custom editor in unity takes a lot of time. For every class we need to create an editor class and wirte many lines of code.

Example

A class with two points:

MyClass.cs

```
using UnityEngine;

public class MyClass : MonoBehaviour
{
    public Vector3 pointA, pointB;
}
```

To Do List:

1. Draw a line from pointA to pointB
2. Make moveable handles for pointA and pointB
3. Make labels

What we used to do:

MyClassEditor.cs

```
using UnityEngine;
using UnityEditor;

[CustomEditor(typeof(MyClass))]
public class MyClassEditor : Editor
{
    private MyClass myClass;

    private void OnEnable()
    {
        myClass = target as MyClass;
    }

    private void OnSceneGUI()
    {
        Vector3 pointA = myClass.pointA;
        Vector3 pointB = myClass.pointB;
        float pointAHandleSize = HandleUtility.GetHandleSize(pointA) * 0.1f;
        float pointBHandleSize = HandleUtility.GetHandleSize(pointB) * 0.1f;
        Quaternion q = Quaternion.identity;

        Handles.DrawLine(pointA, pointB);
        Handles.Label(pointA, "pointA");
        Handles.Label(pointB, "pointB");

        EditorGUI.BeginChangeCheck();
        pointA = Handles.FreeMoveHandle(pointA, q, pointAHandleSize,Vector3.zero,Handles.SphereHandleCap);
        if (EditorGUI.EndChangeCheck())
        {
            Undo.RecordObject(myClass, "move pointA");
            myClass.pointA = pointA;
        }
        EditorGUI.BeginChangeCheck();
        pointB = Handles.FreeMoveHandle(pointB, q, pointAHandleSize, Vector3.zero, Handles.SphereHandleCap);
        if (EditorGUI.EndChangeCheck())
        {
            Undo.RecordObject(myClass, "move pointB");
            myClass.pointB = pointB;
        }
    }
}
```

Using EHandles:

MyClass.cs

```
using UnityEngine;

public class MyClass : MonoBehaviour
{
    [EHandles.FreeMoveHandle , EHandles.Label , EHandles.DrawLine("pointB")]
    public Vector3 pointA, pointB;
}
```

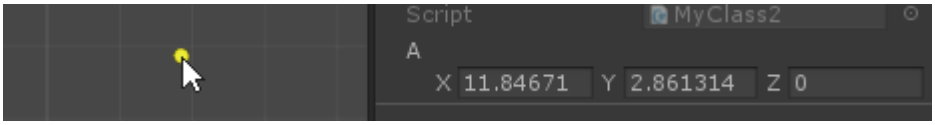
As you can see it save a lot of time.

Topics

How to use?

The EHandles attributes are markers that can be placed above fields.
For example, you can add the FreeMoveHandle attribute above a Vector3 to create a moveable handle in the scene.

```
[EHandles.FreeMoveHandle]
public Vector3 a;
```



Supported Fields
Before adding an attribute to a field make sure it supports your field.
(Check EHandles **Attributes Table**)

Private Fields
It supports private fields with [SerializeField] attribute.

```
[SerializeField, EHandles.FreeMoveHandle]
private Vector3 a;
```

Inside a non MonoBehaviour class or struct
You need to add a System.Serializable attribute above your class or struct.

```
Public class MyClass : MonoBehaviour
{
    public A a;
    public B b;
}

[System.Serializable]
public class A
{
    [EHandles.FreeMoveHandle]
    public Transform tran;
}

[System.Serializable]
public struct B
{
    [EHandles.FreeMoveHandle]
    public Vector3 v;
}
```

Topics

Attributes

Attribute	Supported Fields	Properties
EHandles.FreeMoveHandle	VGT, VGT[], List<VGT>	handleSize, color
EHandles.Label	VGT, VGT[], List<VGT>	text, color
EHandles.DrawLine	VGT	endPointField, color
EHandles.DrawPolyline	VGT[], List<VGT>	loop, color
EHandles.DrawAAPolyline	VGT[], List<VGT>	loop, width, color
EHandles.DrawAAConvexPolygon	VGT[], List<VGT>	color
EHandles.DrawCircle	VGT, VGT[], List<VGT>, float	radius, radiusField, color, rotation
EHandles.DrawCube	VGT, VGT[], List<VGT>	handleSize, color
EHandles.DrawSphere	VGT, VGT[], List<VGT>	handleSize, color
EHandles.PositionHandle	VGT, VGT[], List<VGT>	...
EHandles.RotationHandle	GT, GT[], List<GT>	...
EHandles.TransformHandle	GT, GT[], List<GT>	...
EHandles.PositionHandlePro (New)	VGT, VGT[], List<VGT>	buttonSize, buttonColor
EHandles.UseLocalSpace (New)	...	transformField
EHandles.UseArrayHotkeys (New)	...	...
EHandles.SceneButton	It only supports Method!	text
EHandles.ExecuteAlways (New)	It only supports Class!	

(V: Vector3, Vector2, Vector3Int, Vector2Int) (G: GameObject) (T: Transform) (VGT: V|G|T) (GT: G|T)

EHandles.FreeMoveHandle

Declarations

```
public FreeMoveHandle()
public FreeMoveHandle(float size)
public FreeMoveHandle(float size, EHandles.Color color)
```

Parameters

size	The size of the handle. (screen space)
color	The color of the handle.

Description

Make a moveable handle in the scene.

```
[EHandles.FreeMoveHandle]
public Vector3 a;

[EHandles.FreeMoveHandle(0.5f)]
public Vector3 b , c , d;

[EHandles.FreeMoveHandle(color = EHandles.Color.yellow)]
public Transform[] Transforms;
```

EHandles.Label

Declarations

```
public Label()
public Label(string text)
```

Parameters

text	The text of the label.

Description

Make a label in the scene.

```
//text = "pointA" (name of current field)
[EHandles.Label]
public Vector3 pointA;

//text = "B"
[EHandles.Label("B")]
public Vector3 pointB;

//p_0, p_1, p_2, ... p_n
[EHandles.Label("p")]
public Vector3[] positions;
```

EHandles.DrawLine

Declarations

```
public DrawLine()
public DrawLine(string endPointSourceField)
public DrawLine(string endPointSourceField , EHandles.Color color)
```

Parameters

endPointField	Name of the end point Field.
color	Color of the line.

Description

Draw a line in the scene.

```
// Draw a line from (a) to (Vector3.zero)
[EHandles.DrawLine]
public Vector3 a;


// Draw a line from (b) to (c)
[EHandles.DrawLine("c")]
public Vector3 b;
public Vector3 c;


// Draw lines from (p0) to (p1) and (p0) to (p2)
[EHandles.DrawLine("p1") , EHandles.DrawLine("p2")]
public Vector3 p0;
public Vector3 p1;
public Vector3 p2;
```

EHandles.DrawPolyline

Declarations

```
public DrawPolyline()
public DrawPolyline(bool loop)
public DrawPolyline(bool loop, EHandles.Color color)
```

Parameters

loop	Connect the start and end positions.
color	Color of the line.

Description

Draw a line going through a list of points.

```
[EHandles.DrawPolyline]
public Vector3[] positions;

[EHandles.DrawPolyline(true , EHandles.Color.yellow)]
public Vector3[] positions_2;
```

EHandles.DrawAAPolyline

Declarations

```
public DrawAAPolyline()
public DrawAAPolyline(float width , bool loop)
public DrawAAPolyline(float width, bool loop , EHandles.Color color)
```

Parameters

width	The width of the line.
loop	Connect the start and end positions.
color	Color of the line.

Description

Draw a anti aliased line going through a list of points.

```
[EHandles.DrawAAPolyline]
public Vector3[] positions;

[EHandles.DrawAAPolyline(width = 3)]
public Vector3[] positions_2;
```

EHandles.DrawAAConvexPolygon

Declarations

```
public DrawAAConvexPolygon()
public DrawAAConvexPolygon(EHandles.Color color)
```

Parameters

color	The Color of the handle.

Description

Draw a convex polygon in the scene.

```
[EHandles.DrawAAConvexPolygon]
public Vector3[] positions;

[EHandles.DrawAAConvexPolygon(EHandles.Color.yellow)]
public Vector3[] positions_2;
```

EHandles.DrawCircle

Declarations

```
public DrawCircle()
public DrawCircle(float radius)
public DrawCircle(float radius , EHandles.Color color)
public DrawCircle(string radiusSourceField)
public DrawCircle(string radiusSourceField , EHandles.Color color)
```

Parameters

radius	Radius of the circle. (world space)
radiusField	Name of the radius field. (world space)
color	Color of the circle.

Description

Draw a circle in the scene.

```
[EHandles.DrawCircle(0.5f)]
public Vector3 b;

public float radius;

[EHandles.DrawCircle("radius")]
public Vector3 c;
```

EHandles.DrawCube

Declarations

```
public DrawCube()
public DrawCube(float size)
public DrawCube(float size, EHandles.Color color)
```

Parameters

size	The size of the handle. (screen space)
color	The color of the handle.

Description

Draw a cube in the scene.

```
[EHandles.DrawCube]
public Vector3 p;
```

EHandles.DrawSphere

Declarations

```
Public DrawSphere()
public DrawSphere(float size)
public DrawSphere(float size, EHandles.Color color)
```

Parameters

size	The size of the handle. (screen space)
color	The color of the handle.

Description

Draw a sphere in the scene.

```
[EHandles.DrawSphere]
public Vector3 p;
```

EHandles.PositionHandle

Make a position handle in the scene.

```
[EHandles.PositionHandle]
public Vector3 a;
```

EHandles.RotationHandle

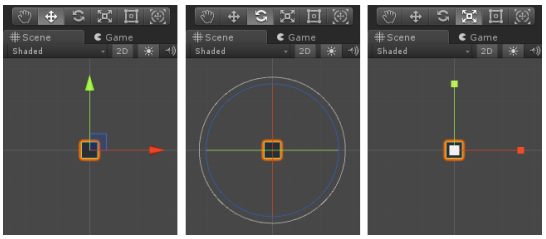
Make a rotation handle in the scene.

```
[EHandles.RotationHandle]
public Transform b;
```

EHandles.TransformHandle

Make a TransformHandle in the scene.

```
[EHandles.TransformHandle]
public Transform tran;
```



Supports: Move tool, Rotate tool, Scale tool

EHandles.PositionHandlePro (New)

Declarations

```
public PositionHandlePro()
public PositionHandlePro(float buttonSize)
public PositionHandlePro(float buttonSize, EHandles.Color buttonColor)
```

Parameters

buttonSize	The size of the button. (screen space)
buttonColor	The color of the button.

Description

Make a selectable position handle. (It can be selected by clicking the sphere button.)

```
[EHandles.PositionHandlePro]
public Vector3 p;
```

EHandles.UseLocalSpace (New)

Declarations

```
public UseLocalSpace()
public UseLocalSpace(string transformField)
```

Parameters

transformField	Name of the transform field.

Description

Transforms position from local space to world space before drawing handles.

```
[EHandles.PositionHandle, EHandles.UseLocalSpace]
public Vector3 localPosition;

public Transform parent;

[EHandles.PositionHandle, EHandles.UseLocalSpace("parent")]
public Vector3 localPosition2;
```

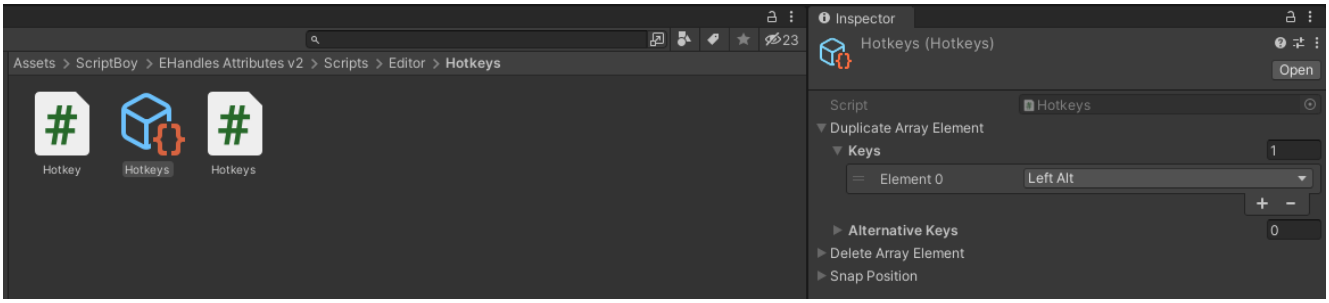
EHandles.UseArrayHotkeys (New)

You can delete or duplicate array elements by holding hotkeys.

```
//Holding D => Delete dragged position
//Holding A => Duplicate dragged position
[EHandles.UseArrayHotkeys]
[EHandles.PositionHandlePro, EHandles.DrawPolyline]
public Vector3[] positions;
```

How to edit hotkeys?

In the Unity menu bar, go to Tools/ScriptBoy/Ehandles/Hotkeys.



EHandles.SceneButton

Declarations

```
Public SceneButton()
public SceneButton (string text)
```

Parameters

text	The text of the button.

Description

Make a button in the scene.

```
[EHandles.SceneButton]
public void Do()
{
   print("void Do()");
}

[EHandles.SceneButton("Button2")]
public void Do_2()
{
   print("void Do_2()");
}
```

Notes

1. It supports Public, NonPublic, Static methods.
2. It does not invoke methods with parameters.
3. It only supports methods of a MonoBehaviour class.

EHandles.ExecuteAlways (New)

By default, It only shows handles of selected objects. You can mark your class to make the handles visible, even if the object is not selected.

```
[EHandles.ExecuteAlways]
public class PositionHandleTest : MonoBehaviour
{
    [EHandles.PositionHandle]
    public Vector3 position;
}
```

How to write a custom attribute? (New)

1. Create a class derive from **EHandles.Attribute**.
(**Recommended File Location:** Assets\ScriptBoy\EHandles Attributes v2\Scripts\Runtime\Attributes\Custom)

```
public class MyHandleAttribute : EHandles.Attribute
{

}
```

2. Create a class derive from **EHandles.AttributeAction<T>** where **T** is **MyHandleAttribute**.
(**Recommended File Location:** Assets\ScriptBoy\EHandles Attributes v2\Scripts\Editor\AttributeAction\Actions\Custom)
(**Note:** The attribute action class file must be in the Assets\ScriptBoy\EHandles Attributes v2\Scripts\Editor)

```
public class MyHandleAction : EHandles.AttributeAction<MyHandleAttribute>
{

}
```

3. Override the **OnSceneGUI(SerializedProperty property)** method.

```
public class MyHandleAction : EHandles.AttributeAction<MyHandleAttribute>
{
    protected override void OnSceneGUI(SerializedProperty property)
    {
        Debug.Log(property.name);
    }
}
```

Now you can use **MyHandleAttribute**.

```
public class Example : MonoBehaviour
{
    [MyHandle]
    public Vector3 v;
}
```

EHandles.Attribute (New)

This is the base class of EHandles attributes. (You can write a custom attribute by deriving from it.)

EHandles.AttributeAction<T> (New)

Description

This is an abstract generic class where T should be a subclass of EHandles.Attribute.
It has an abstract method called OnSceneGUI(SerializedProperty property), It will be called for all fields marked with the T attribute.
Note: There will be only one AttributeAction instance for all T attribute instances.
(e.g. One LableAction instance for all LableAttribute instances)

Parameters

Type	Name	Description
T	attribute	Current attribute.

Inherited Members Properties

Type	Name	Description
FieldInfo	fieldInfo	Current field.
Type	fieldType	Current field type.
Transform	transform	The transform of the current script.

Inherited Members Methods

```
bool TryGetPosition(SerializedProperty property, out Vector3 position)
bool TryGetPositions(SerializedProperty property, out Vector3[] positions, bool loop)
void SetPosition(SerializedProperty property, Vector3 position)
```

EditMode (Removed)

In the old version you could enable/disable EHandles by adding a boolean named editMode.

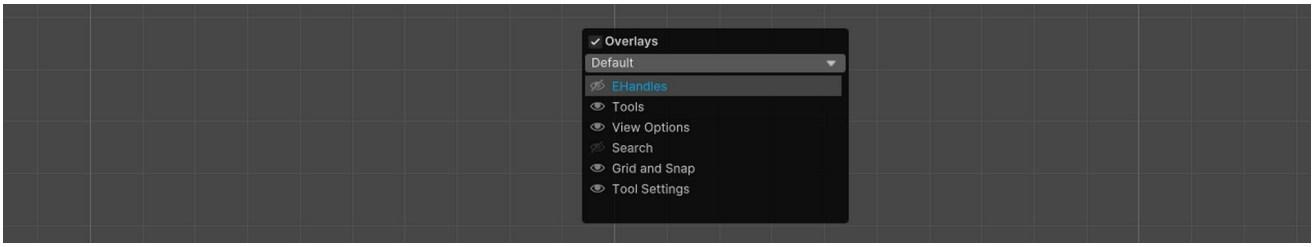
```
public class Example : MonoBehaviour
{
    //It must be here (above other fields)
    [SerializeField] private bool editMode;

    [EHandles.FreeMoveHandle]
    public Vector3 a;
    [EHandles.FreeMoveHandle]
    public Vector3 b;
}
```

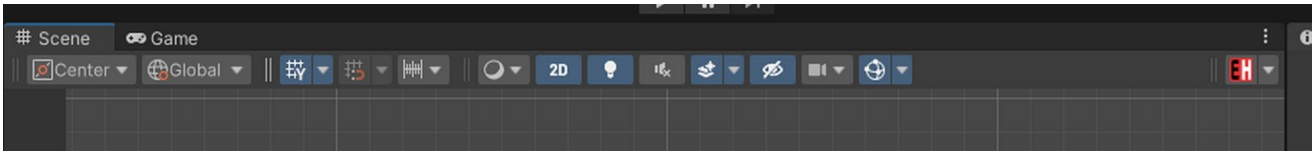
Alternative Features

1. EHandles Toolbar Overlay
2. EHandles Settings

EHandles Toolbar Overlay (New)

In the scene window, press the **Space** button, Then enable the EHandles overlay.





Buttons

: You can enable/disable EHandles by clicking this button.

: Click this button to open more options.

Note: It's only available for Unity 2021.2 or newer.

EHandles Settings (New)

You can edit settings if you go to Tools/ScriptBoy/Ehandles/Settings.

Properties

☑ **Is Enabled:** Is EHandles enabled ?
☑ **ShowHandles:** Enable/Disable the visibility of handles.
☑ **ShowSceneButtons:** Enable/Disable the visibility of scene buttons.

☑ **OptimizeSceneButtons (New) :** By default, It checks all scripts in the scene!
Now, It only checks classes marked with EHandles.ExecuteAlways.
(Reload your scene once you edit this option)

☑ **CheckPropertyType:** You can disable this to allow all types.
(It checks if the property type is allowed or not before checking its attribute.)

AllowedPropertyTypes: Edit supported field types.
(**By default, It only supports these types:** Vector3, Vector2, Vector3Int, Vector2Int, ObjectReference, Generic)

✉ Links & Contact Info

https://youtube.com/EHandlesAttributes

ScriptBoyTools@outlook.com

Have Fun!
Script Boy
;)