Unity Editor Customization

(should have chosen a smaller topic)

A way too fast view of Unity Editor Customization

(should have chosen a smaller topic)

If you missed the talk...

- It was fast
- These slides are an over-simplification
- YMMV: the talk is a starting-point, not an end-point

Things to show..

..auto run on import..

..in-scene context sensitivity..

[live demo goes here]

HOW?

- Gizmos (“visualize in 3D”)
- Handles (“interact in-scene”)
- Editors (“customize Inspector”)
- EditorWindows (“windows”)

...but

- Gizmos (“visualize in 3D”)
  - RUNS IN GAME-MODE ONLY
  - NO INTERACTION
- Handles (“interact in-scene”)
  - EDITOR CLASSES ONLY

First: Handles

Summary for Custom Handles:

1. Calculate a distance

2. Draw

3. Draw-with-context

```
public virtual void OnSceneGUI()
```

Phase I
(init + store)

- foreach(... handle you want for this object...)
  - newID = GUIUtility.GetControlID (FocusType.Passive));
  - if (newID < 0) *// not allowed right now*
    - continue;
  - thingsByHandleID.Add(newID ...);

Phase 2
(measure + report)

- foreach(... handle you want for this object...)
  - if (e.type == EventType.layout)
    - HandleUtility.AddControl(newID, handle.PixelDistanceFromMouseToHandle());

Phase 3
(draw it!)

- foreach(... handle you want for this object...)
  - handle.DrawAt(tipOfHandle);
  - // shift this if mouse is dragging

Phase 4
(retrieve it)

- selectedHandle = thingsByHandleID [HandleUtility.nearestControl];

Phase 5
(deal with mouse)

- if (e.isMouse)

```
{
   switch (e.type)
   {
   ...
```

5: mouseDown

- switch (e.type)
  - case EventType.mouseDown:

```
                {
                    currentHandleBeingDraggedControlID =
HandleUtility.nearestControl;
                    currentHandleDragOffset = Vector3.zero;
                    HandleUtility.AddDefaultControl
(HandleUtility.nearestControl);

                    OnHandleGrabbed( selectedHandle );
                }
```

6: mouseDrag

- switch (e.type)
  - case EventType.mouseDrag:
    - // check it’s one of OUR handles
    - // choose a 3D position for the 2D mouse position
    - // update our private “render offset for this Handle”
  - HandleUtility.Repaint();

7: mouseUp

- switch (e.type)
  - case EventType.mouseUp:

```
                {
                    OnHandleReleased( selectedHandle,
selectedHandle.position + currentHandleDragOffset );

                    currentHandleBeingDraggedControlID =
-1;
                    HandleUtility.AddDefaultControl (-1); //
removes it
                }
```

Second: Customize your class

Custom Gizmos:

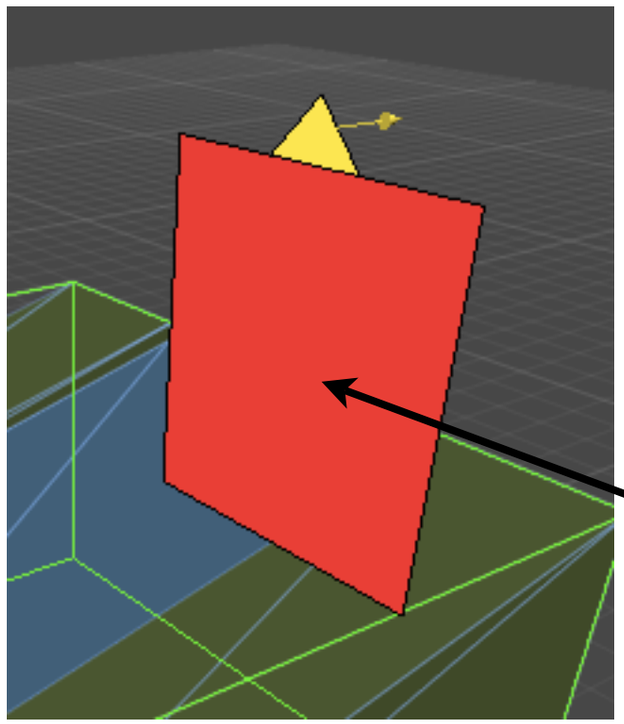
```
void OnDrawGizmos()
{
    Gizmos.color = socket.gizmoColour;
    Gizmos.DrawWireCube(Vector3.zero,
        new Vector3(socket.diameterApproximately,
        socket.diameterApproximately,
        0.1f*socket.diameterApproximately)
    );
}
```

I weird thing...

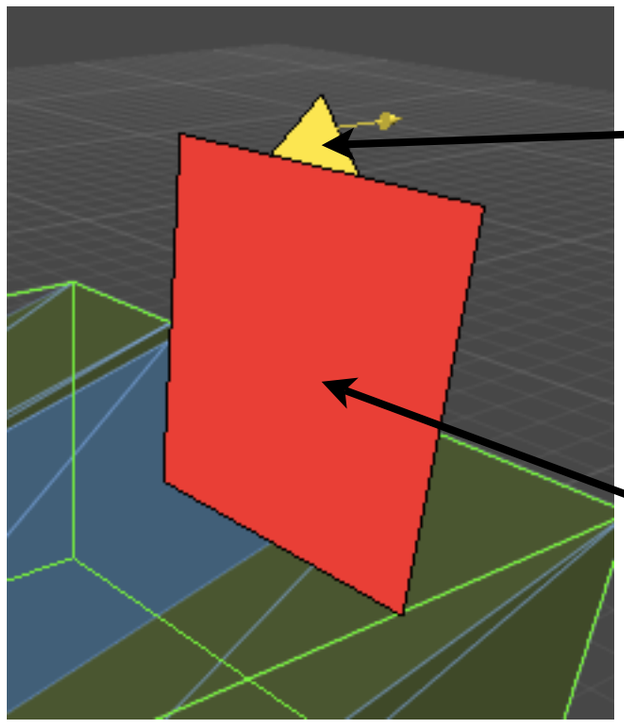
- Handles rendering API is missing some of the features of Gizmos
- Handles can't draw triangles :(
- ...or can it?

Handles.

DrawSolidRectangleWithOutline()



Rectangle



Also a rectangle

Rectangle

Third: Customize your Inspector

In theory...

```
using UnityEditor;
```

```
using UnityEditor;
 [CustomEditor(typeof(SnapPiece))]
```

```
using UnityEditor;
 [CustomEditor(typeof(SnapPiece))]

public class EditorSnapPiece : Editor
{
```

```
using UnityEditor;
 [CustomEditor(typeof(SnapPiece))]

public class EditorSnapPiece : Editor
{
        public override void OnInspectorGUI()
        {
```

```
using UnityEditor;
 [CustomEditor(typeof(SnapPiece))]

public class EditorSnapPiece : Editor
{
        public override void OnInspectorGUI()
        {
```

... use standard Unity GUI, GUILayout, etc.

... use EditorGUI, EditorGUILayout **where they help**, or ignore them

In practice...

(weep)

Serialization

(weep)

Serialization

(weep)

Dictionary

Serialization

(weep)

Undo

Dictionary

Serialization

Generics

(weep)

Undo

Dictionary

Serialization

Generics

(weep)

Undo

Dictionary

DLL's broken

Serialization

Generics

SerializedProperty

(weep)

Undo

Dictionary

DLL's broken

Serialization

Generics

SerializedProperty

(weep)

Unity API's broken

Undo

Dictionary

DLL's broken

Serialization

Generics

SerializedProperty

(weep)

Unity API's broken

Settings

Undo

Dictionary

DLL's broken

Dragon1: Serialization

Serialization

- Read Lucas Meijer's blog post. http://blogs.unity3d.com/2014/06/24/serialization-in-unity/

Serialization

- Read Lucas Meijer's blog post. http://blogs.unity3d.com/2014/06/24/serialization-in-unity/
- tl;dr:
  - ...

Serialization runs ALL THE TIME

To check, must “start/stop scene”

...and: must “save, quit, re-open Unity”

Use ISerialization

callbacks ... or die

Constructors ARE CALLED MORE THAN ONCE

...short version:

Serialization

- Inspectors: use SerializedProperty
- ...and then don’t worry about it.

But...

Serialization

- SerializedProperty is missing support for some common use-cases.

Dragon2: Undo

Undo

- All (game) code you've written to date is now wrong, will silently break your project
- Required methods won't compile if included in (game) classes
- Requires Serialization to be RUNNING and HAVE NO ERRORS (or silently fails)

Undo

- ..and:

Undo

- It’s broken in current Unity.
- (Transform. Fixed in next update, maybe)

Serialization

SerializedProperty

Undo

DLL's broken

Unity API's broken

?

Generics

Dictionary

Settings

Serialization

SerializedProperty

Undo

DLL's broken

Unity API's broken

Generics

Dictionary

Settings

Serialization

SerializedProperty

Undo

DLL's broken

Unity API's broken

Generics

Dictionary

Settings

Serialization

SerializedProperty

Undo

DLL's broken

Unity API's broken

Generics

Dictionary

Settings

Conclusions

- Steep learning "cliff"
- Well worth the effort
- Many easy wins / low-hanging fruit along the way
- Avoid Serialization, avoid Generics, until you have the time to learn these in DETAIL

Questions?

More?

@t_machine_org