

UnityGUI

# This Talk

- What Is UnityGUI?
- Fixed Positioning
- Auto-Layout
- Skinning

# What Is It

- Completely skinnable, has autolayouting
- Immediate mode GUI
- Windows, Buttons, Toggles, Sliders, Scrollviews, etc.

# Fixed positioning

Lot's of Rect() coming up

# Small Example

```
function OnGUI () {  
    GUI.Label (Rect (10,10,120,20), "Hello World!");  
    GUI.Button (Rect (10,30,120,20), "Meet the new GUI");  
}
```

Hello World!

Meet the new GUI

# Meet OnGUI

- Called every frame
  - just like update
- This means you can have other code in there!

# Code Flow

```
function OnGUI () {  
    GUI.Label (Rect (10,10,120,20), "Hello World!");  
    if (GUI.Button (Rect (10,30,120,20), "Start Game"))  
        Application.LoadLevel (1);  
}
```

- The GUI.Button returns true when the user clicks it
- So we can put other stuff in there

# Data Controls

- We just pass in values we want to display as a parameter.
- The function returns the value
  - which may have been modified

```
var playerName = "";  
  
function OnGUI () {  
    playerName = GUI.TextField (Rect (10, 50, 120, 20), playerName);  
}
```

Im typing like mad|

# Conditional GUI

- The OnGUI function gets called like update  
So just throw an if in there

```
var playerName = "";  
var advancedSettings = true;  
  
function OnGUI () {  
    advancedSettings = GUI.Toggle (Rect (10, 10, 120, 20), advancedSettings, "Advanced Settings");  
    if (advancedSettings) {  
        playerName = GUI.TextField (Rect (10, 40, 120, 20), playerName);  
  
        // Just put more GUI here  
    }  
}
```

Advanced Setting

This is a fine day

# More Code Flow

- OnGUI gets called multiple times per frame
  - Mouse & Keyboard events, Layouting  
...and *Display*
- Unless the user has done any action, UnityGUI expects the code flow to be the same.
- This means you should not have other code in there

# Fixed positioning

- Questions?

# Automatic Layout

I hate Rect()

So we've also done an automatic layouting system

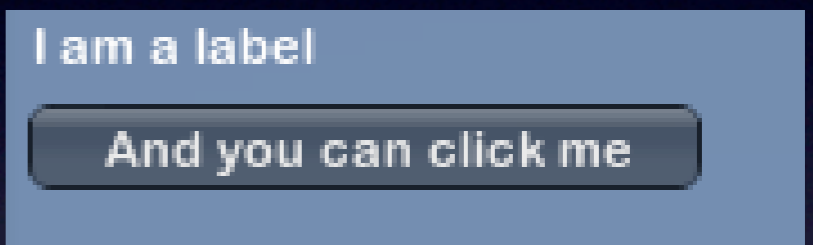
# GUILayout

- Like old-school HTML templates
- Just switch GUI. with GUILayout & remove the Rect parameter
- GUI.Button (Rect (10, 10, 100, 20), "Hello");  
becomes  
GUILayout.Button ("Hello")

# Grouping

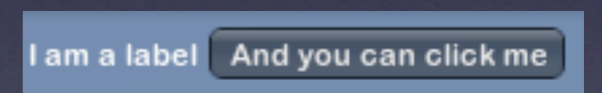
- By default, controls are placed underneath each other

```
function OnGUI () {  
    GUILayout.Label ("I am a label");  
    GUILayout.Button ("And you can click me");  
}
```



- Use BeginHorizontal / EndHorizontal

```
function OnGUI () {  
    GUILayout.BeginHorizontal ();  
    GUILayout.Label ("I am a label");  
    GUILayout.Button ("And you can click me");  
    GUILayout.EndHorizontal ();  
}
```



- You can nest as deep as you want

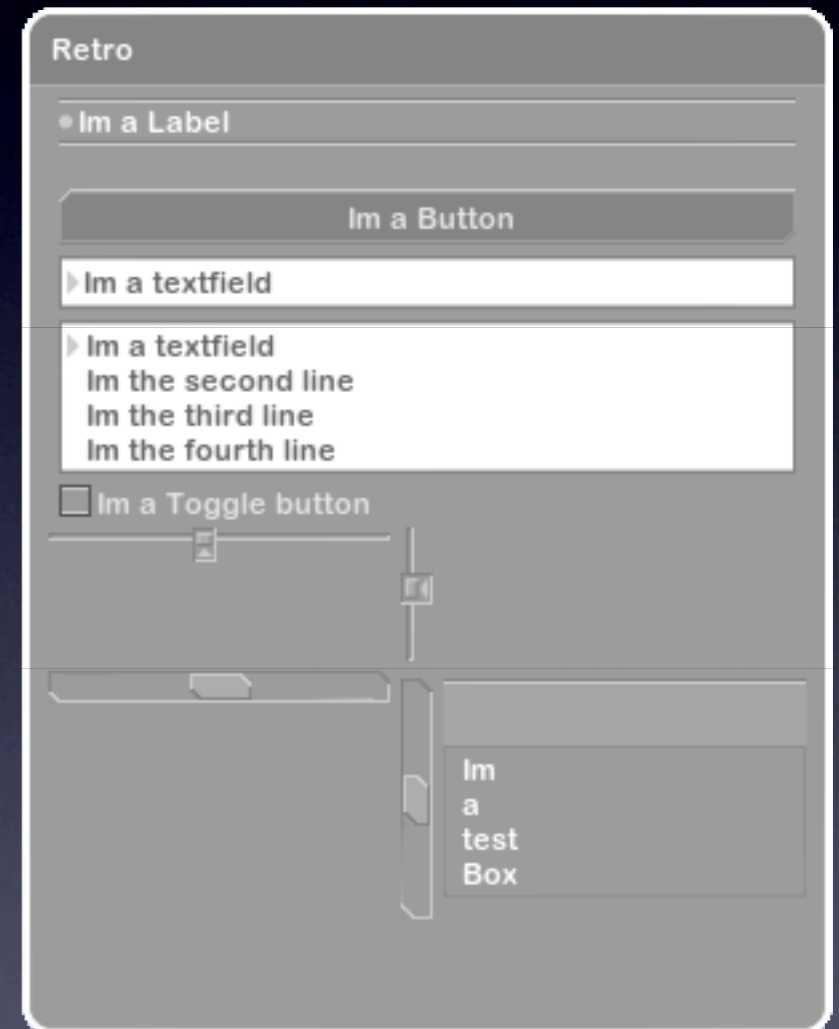
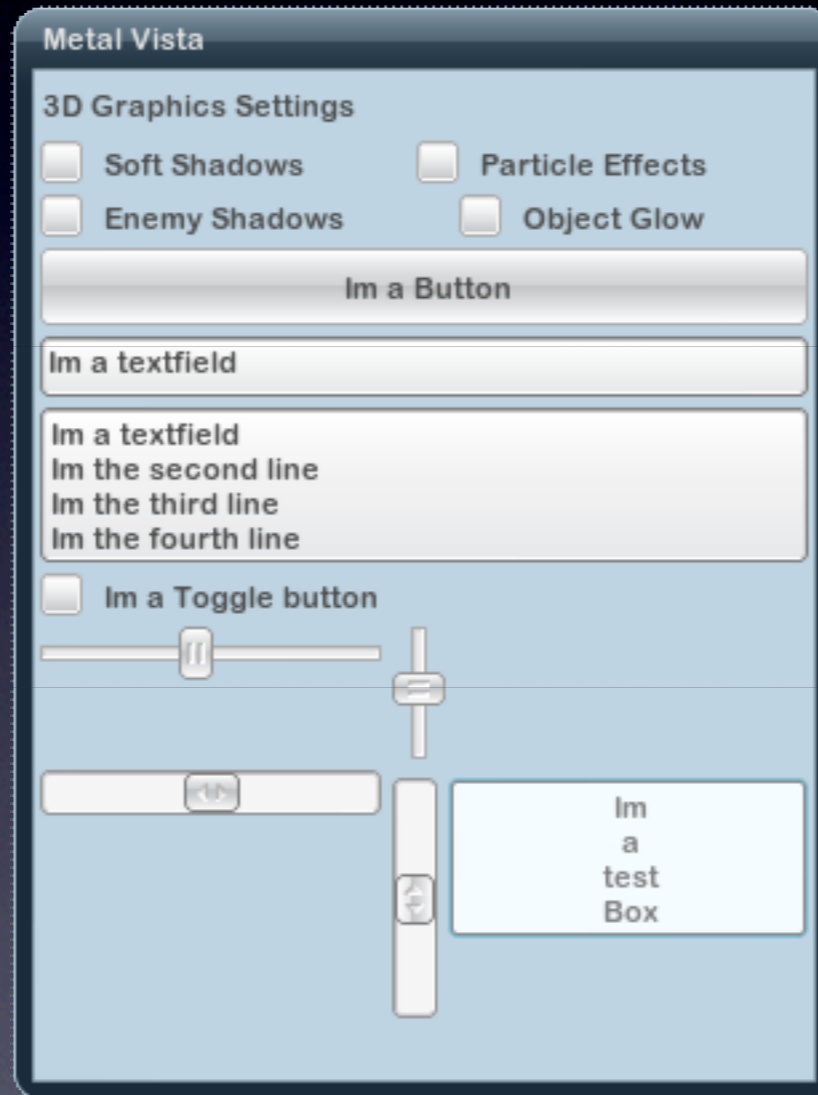
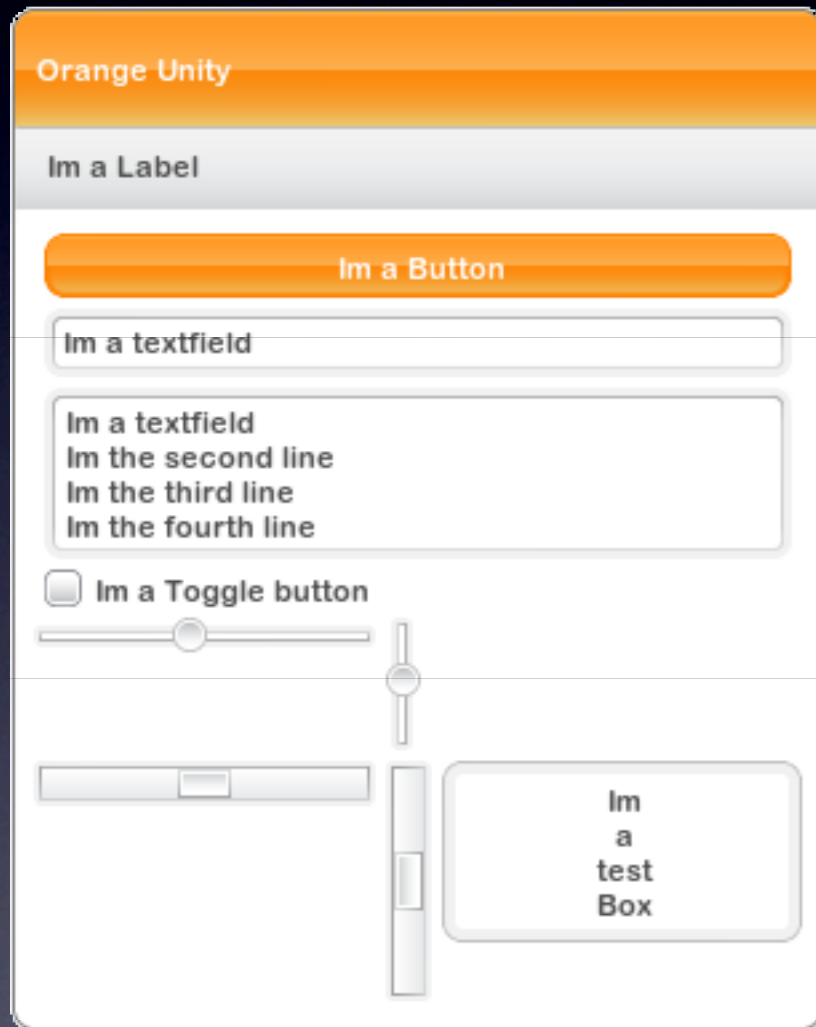
# Size

- By default, Unity will determine size for you
- You can add overrides to this:  
    GUILayout.Button (“Hi”, *options*)
- *options* can be any number of:
  - GUILayout.Height (50)
  - GUILayout.MinWidth (150)
  - Add as many as you want - separate by comma

# GUILayout

- Questions?

# Skinning



# Content and Style

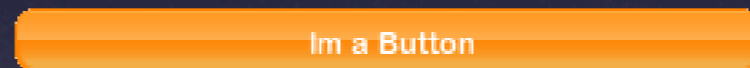
- GUI controls are made up of 2 things

Style

Content



Im a button



- A GUI skin is a collection of styles - one for each control, as well as a bunch of custom ones

# Creating GUISkin

- Create one from Assets->Create->Gui Skin
- Create a variable and assign it in your script:

```
var mySkin : GUISkin;  
function OnGUI () {  
    GUI.skin = mySkin;  
    GUILayout.Button ("Please Click me");  
}
```

# Using Custom styles

- All controls have an optional *style* parameter at the end:

```
GUILayout.Button ("Hi there", "MyCoolStyle")
```

- Finds "MyCoolStyle" in your skin and uses that to render.
- Example: make a toggle look a normal button:

```
myVar = GUILayout.Toggle ("Hi there", myVar, "button");
```

# Skinning

- Questions?