GAM250: Advanced Games Programming

4: Tool Development

Learning outcomes

- **Understand** the importance of tools in Game Development
- **Implement** Scriptable Objects to create custom assets
- **Implement** Property Inspectors and Wizards with Unity Editor Scripts

Attributes

Attributes in C#

- These are classes that allow you to add metadata to other classes
- We can then extract these attributes and use them in our own code
- There are a few of these defined in the C# language including
  - Obsolete - Mark a method as obsolete
  - Conditional - Mark a method as optionable based on a flag
  - AttributeUsage - Defines how a custom attribute can be used
- We can also create our own custom attributes by creating a class that inherits from **Attribute**

Using Attributes

- To use an attribute you **decorate** a method, class or variable with a set of square brackets with the name of the attribute

```
[Obsolete]
void OldUpdateMethod(int x,int y)
```

- Please note attributes can decorate classes, methods or variables

Unity Attributes

- Unity has a number of inbuilt Attributes that can be used to mark your code (or inherit from to make new Attributes)
  - ContexMenu - Markup a function - Adds a command to the context menu which can be selected in the inspector.
  - SerializeField - Markup a Variable - Forces Unity to serialize a variable which will then be displayed in the inspector
  - CustomEditor - Markup a class which inherits from Editor - This specifies that this editor class will be an editor for a certain script.
  - DrawGizmo - Markup a function - This method should have same signature as DrawGizmo, will draw a gizmo in the editor.

`https://docs.unity3d.com/400/Documentation/ScriptReference/20_class_hierarchy.Attributes.html`

Property Drawer

- Property Drawers allow you to customize how a script displays its variables in the inspector
- You have to create a class that inherits from **PropertyDrawer**, then override the **OnGUI** function
- We then have complete control of how the property is drawn using Immediate Mode GUI (drawing via code)

PropertyDrawer

Property Drawer Tips

- Create an Editor Folder to hold the PropertyDrawer Classes
- Create an Attribute Folder to hold any Custom Attributes
- All GUI code uses the IMGUI (aka Immediate Mode GUI), you have to code your GUI by hand

Property Drawer Live Coding

https://docs.unity3d.com/Manual/editor-PropertyDrawers.html https://docs.unity3d.com/Manual/GUIScriptingGuide.html https://blogs.unity3d.com/2015/12/22/going-deep-with-imgui-and-editor-customization/

ScriptableObjects

Scriptable Objects

- Is a class which allows you to store data separate from Script instances
- You have to inherit from the **ScriptableObject** class
- All properties are serialized and can be stored in a asset
- This means that you can add an instance of your ScriptableObject to a script and then assign the asset to it
- This is great for storing stats, weapons etc
- TIP: You can decorate your Scriptable Object with a CreateAssetMenu Attribute. This will allow you to create the Scriptable Object via a menu item.

`https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/scriptable-objects`

Scriptable Objects Live Coding

Wizard

More on Editor Scripts

- All editor scripts should be placed in an Editor Folder
- You have to add a using statement for **UnityEditor** to the top of the source file
- You can create custom Inspectors for Scripts using Property Drawer or you can take complete control and create a Custom editor
- You can also add custom menu items, this then allows you to create Wizards for common tasks

Immediate Mode GUI

- This is a separate system from Unity's Game Object based UI System
- You write the GUI entirely in code, this code has to be in the OnGUI (or equivalent) function or be in a function that is called by OnGUI
- All logic for the controls have to go into this function, e.g to detect button presses
- You can customise the look n' feel of the editor using GUI Styles and Skins

Wizard Live Coding

Exercise

Exercise 1 - Scriptable Objects

- Create a Scriptable Object which represents a Weapon for the ship
  - This should contain a Prefab for the bullet that is spawned
  - A speed value
  - A damage value
- Replace the original bullet with an instance of the Scriptable Object
- Create an Asset for this weapon
- **Stretch Goal:** Add a list of weapons that allow the player to cycle through these
- **Stretch Goal:** Create an enemy Scriptable Object

Exercise 2 - Wizard

- Create a Wizard which allows you to create a weapon
- **Stretch Goal:** Create an enemy wizard

Exercise 3 - Custom UI

- Create a Custom UI for the weapon
- **Stretch Goal:** Create a Custom UI for the enemy