Unity Tools 2

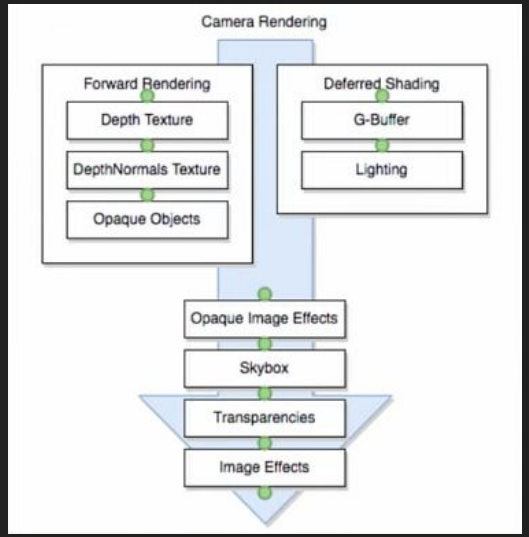
Scripting introduction

Index

- Class 1 recap.
- State of the art: Tool types
- Unity tools scripting: Introduction
  - Methods list
  - GUILayout
- Unity tools: Scripting basics
- Unity tools: Extending the editor
- Exercises:

Unity: Render pipelines

- High customization rendering pipeline (Graphic techniques)
  - Source available in github
- Three types of rendering pipelines
  - Built-in render pipeline
    - Programmer limited
  - Light weight render pipeline
    - Optimized, high performance
    - Forward rendering, mobile devices
    - Low end visuals
  - High definition render pipeline
    - Extended lighting settings
    - High end visuals
    - Target pc and consoles



Resource [GDC]: https://www.youtube.com/watch?time_continue=830&v=zbjkEQMEShM

Unity: Render pipelines

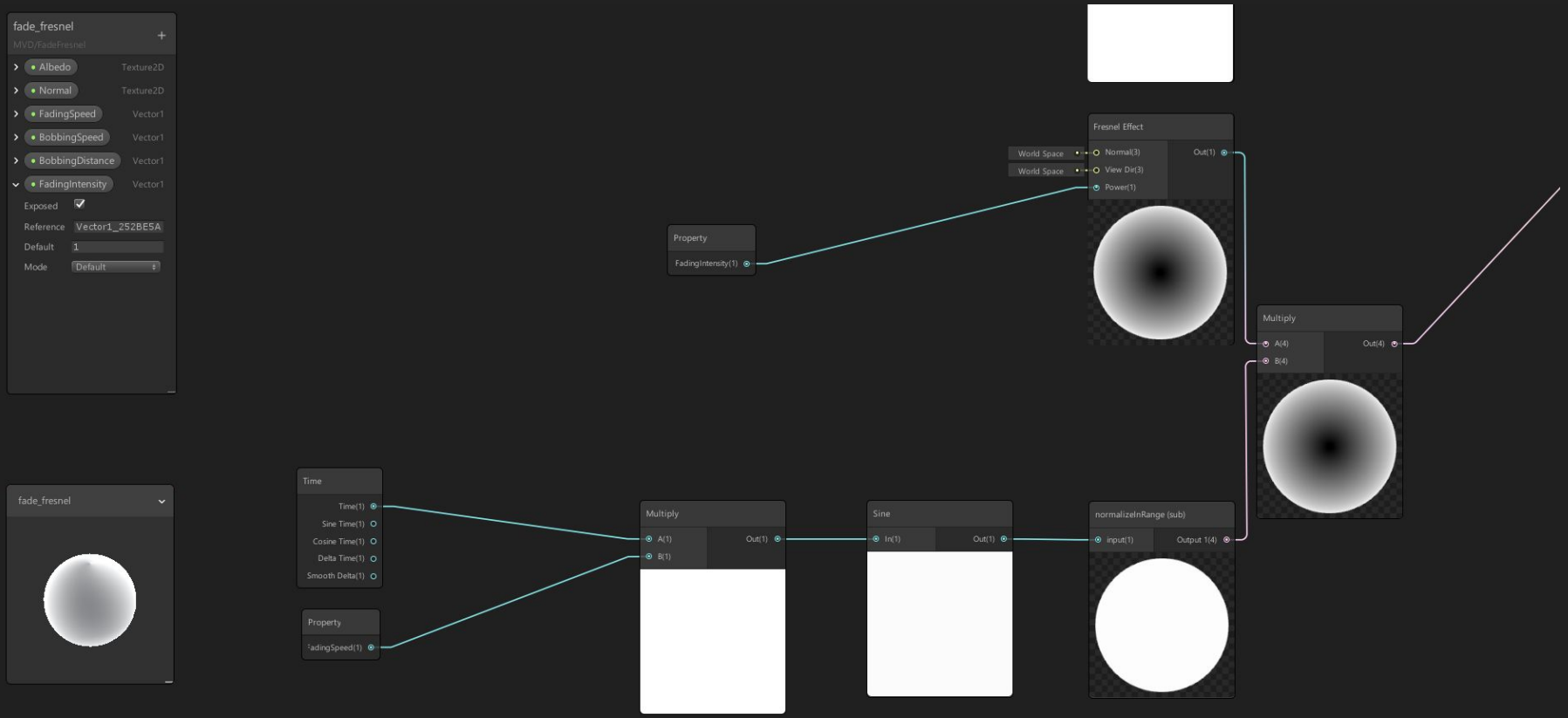
- Available only in 2018.1 and higher
  - HDRP and LWRP packages in the package manager
  - New project -> Templates
  - Scriptable Pipeline repo: https://github.com/Unity-Technologies/ScriptableRenderPipeline

- More information:
  - Unity blogs: SRP overview, LWRP, HDRP
    - https://blogs.unity3d.com/es/2018/02/21/the-lightweight-render-pipeline-optimizing-real-time-performance/
    - https://blogs.unity3d.com/es/2018/03/16/the-high-definition-render-pipeline-focused-on-visual-quality/
  - Source: https://unity.com/srp
  - Keijiro Takahashi: https://github.com/keijiro

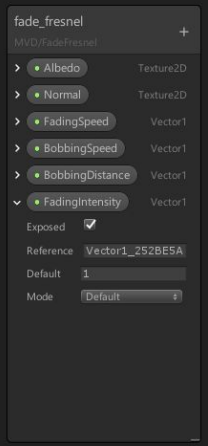
Asset Store: Shadergraph

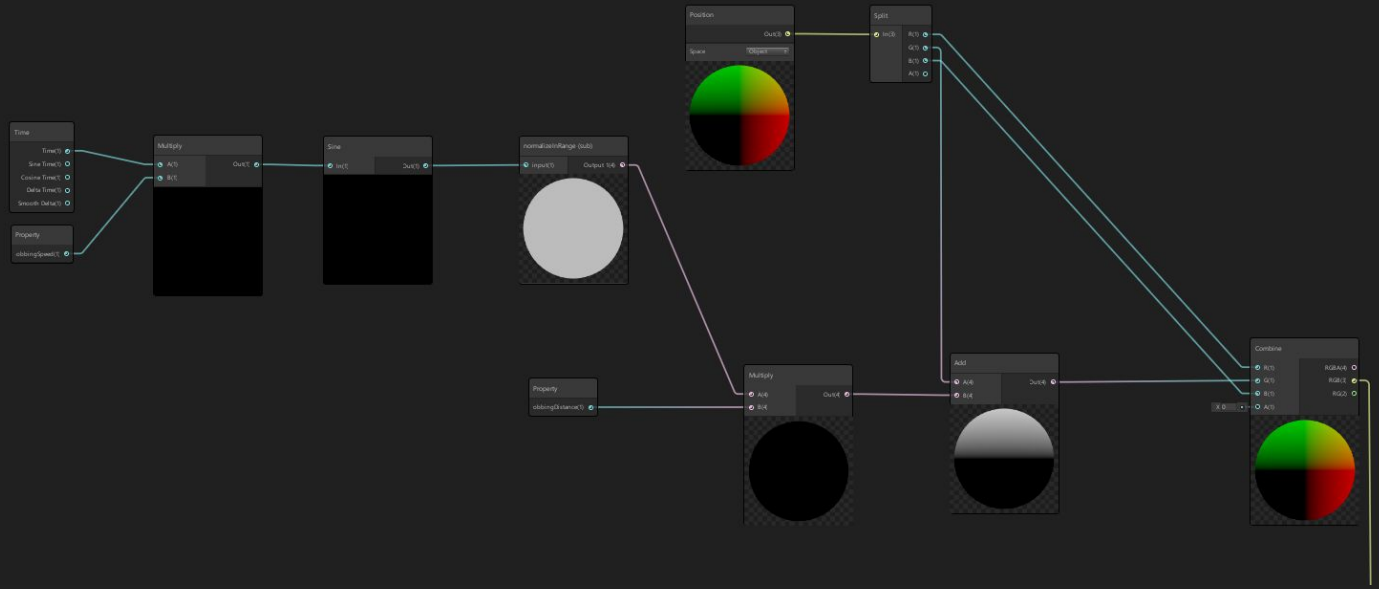
- Create a fade fresnel effect
  - Make it fluctuate over time
  - Allow to change color and speed in inspector
- Create a geometry modification effect
  - Make it translate over time
  - Make it rotate over time
- Create a simple dissolve effect
  - Use a noise texture
  - Dissolve it with a shader parameter

Exercise 1: Fading fresnel

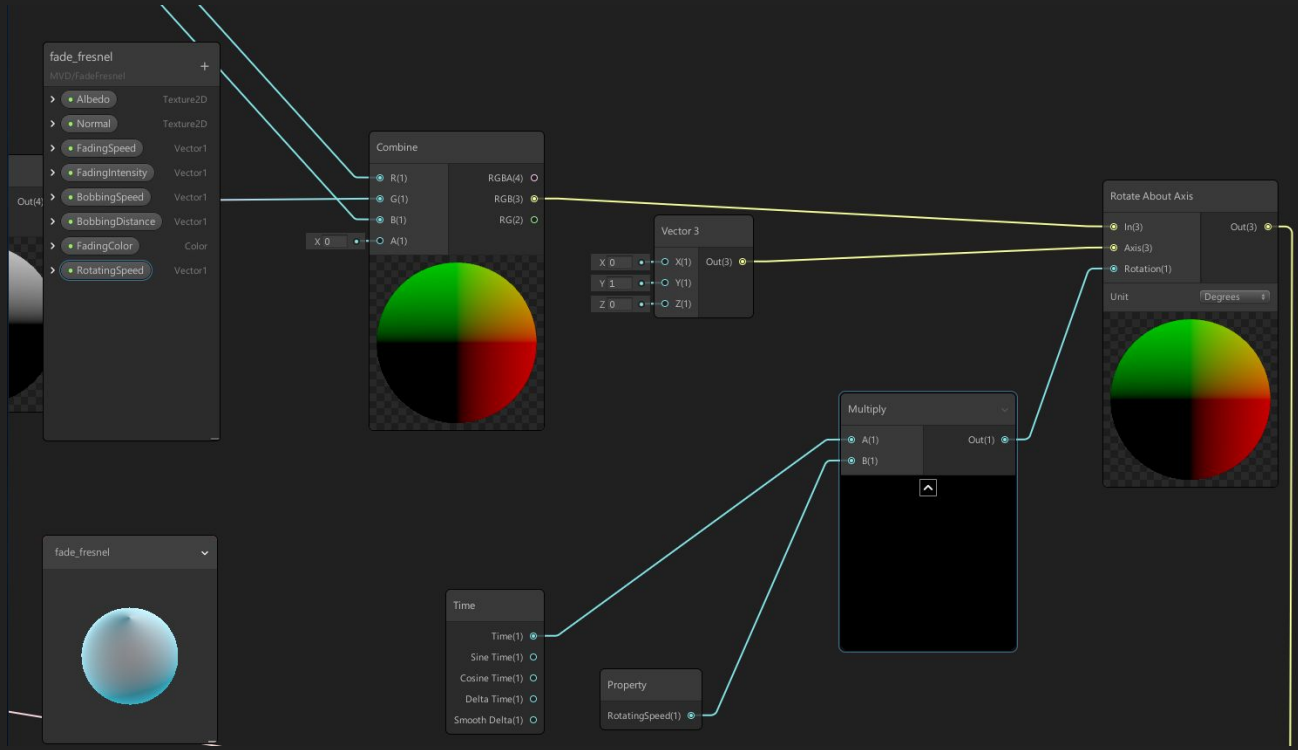


Exercise 2: Translation

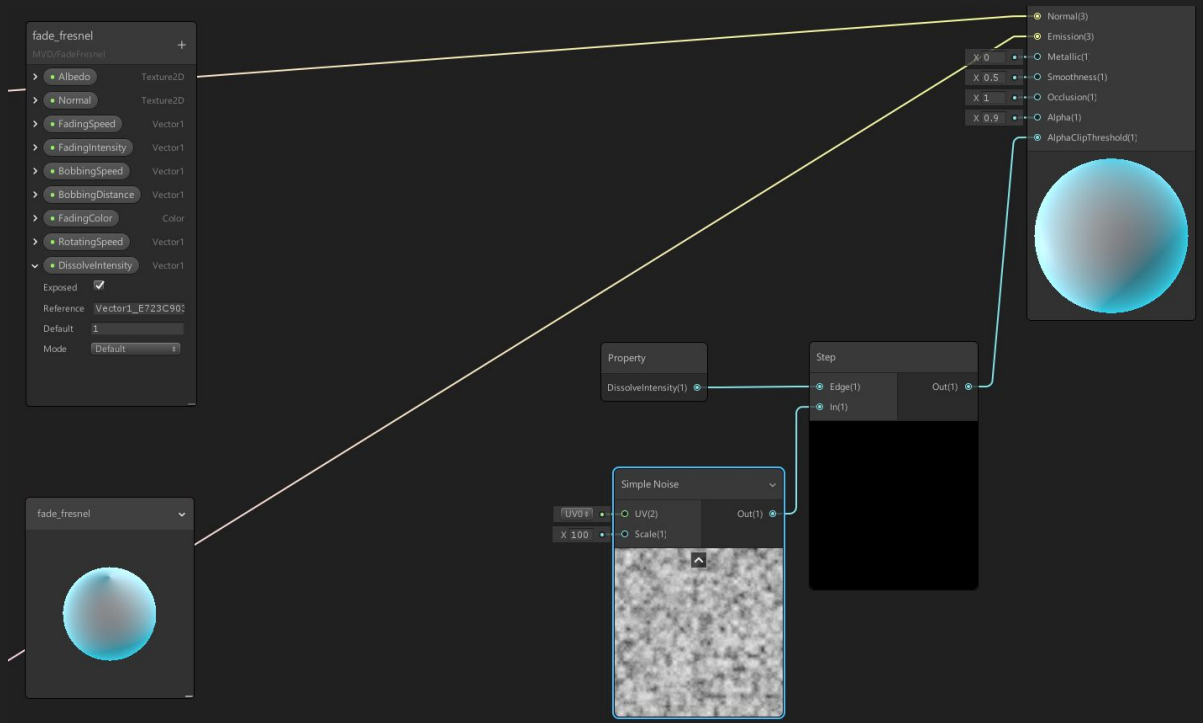




Exercise 2: Rotation



Exercise 3: Dissolve



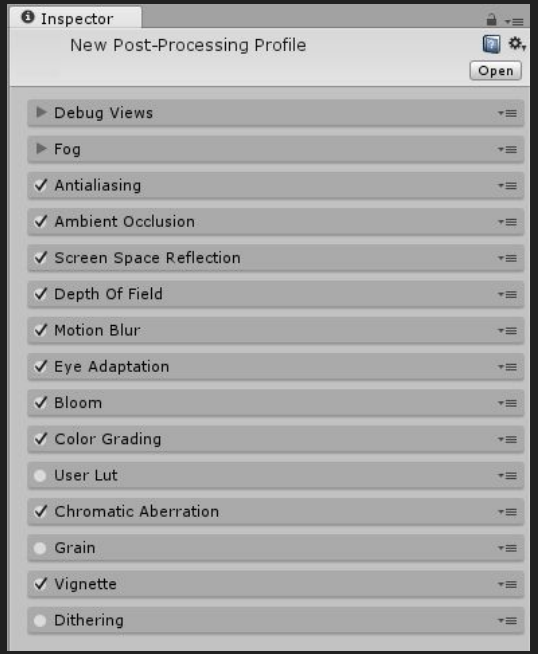
Asset Store: Shadergraph Library

- Github project by Unity Technologies

- Shadergraph code is exposed, can be customized to your needs
- On alpha stage, only works with custom pipeline renders.

Asset Store: Post processing stack

- Process of applying full screen filters
- It improves drastically the visual results
- Many different types of post processing
- Sample given:
  - Viking village
  - Adams demo.
  - Book of the dead demo. (HDRP)



- Apply it always at the end! Settings might change.
- Don’t do bloom abuse!!

Asset Store: Localization tools

- Lean Localization
  - Not useful for our bounds
  - Professional studio oriented.
  - Localization is not only text, also includes sound.
- Very important in some countries (e.g china)
- It can lead to very important problems in the last stage of the project.
  - Texts with different length, UI problems
  - Voice dialogs not fitting properly with the game's narrative.
  - Problems on videogame publishing (e.g china)

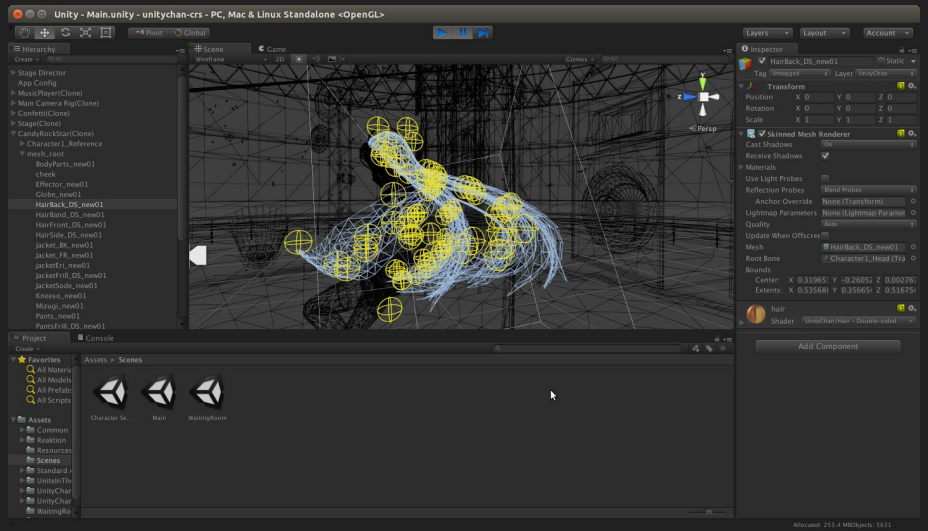
Resource: https://assetstore.unity.com/packages/tools/localization/lean-localization-28504

Which kind of tools do we need in a videogame development process?

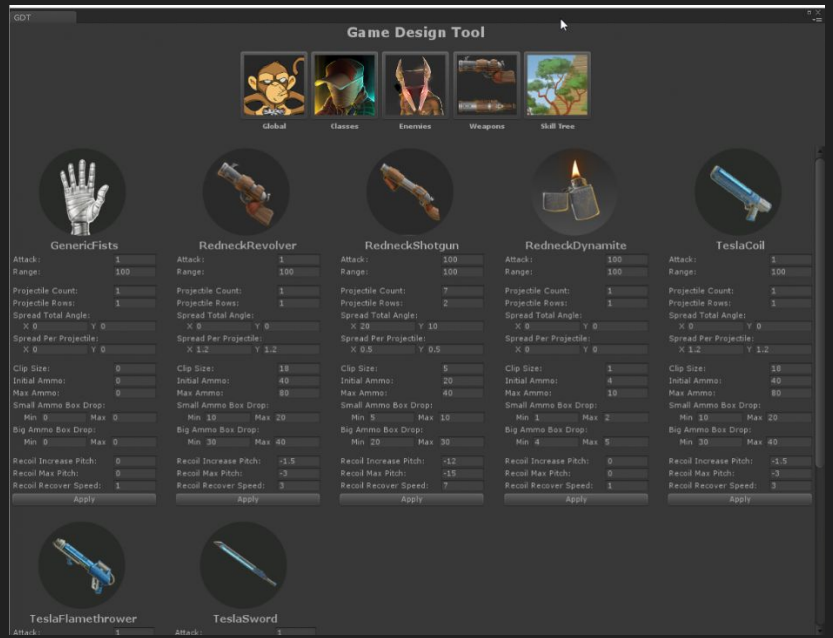
Unity Tools: Editor vs Game

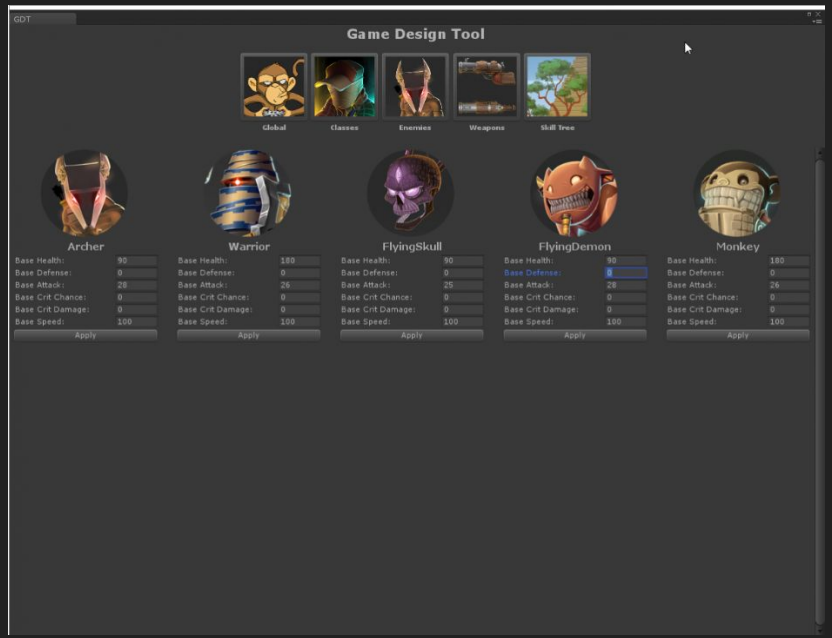
We need to distinguish between

- Tools for the player
  - Editor within the games build.
    - e.g Super mario level editor
  - Customization cvars
- Tools for the developer
  - Third party external tools
    - Tilemap level editor
  - Editor extension tools
    - Inspector customization..
  - Distributed in packages

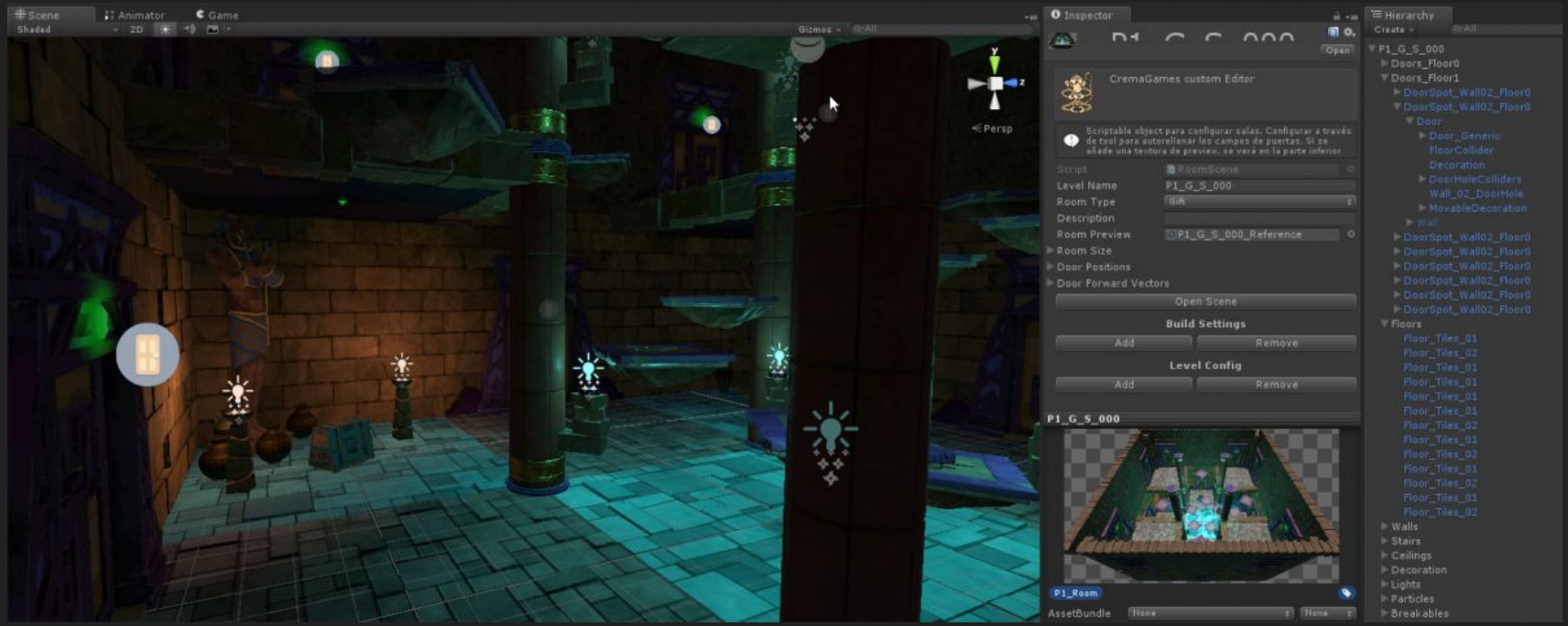


Unity Tools: Game design tools



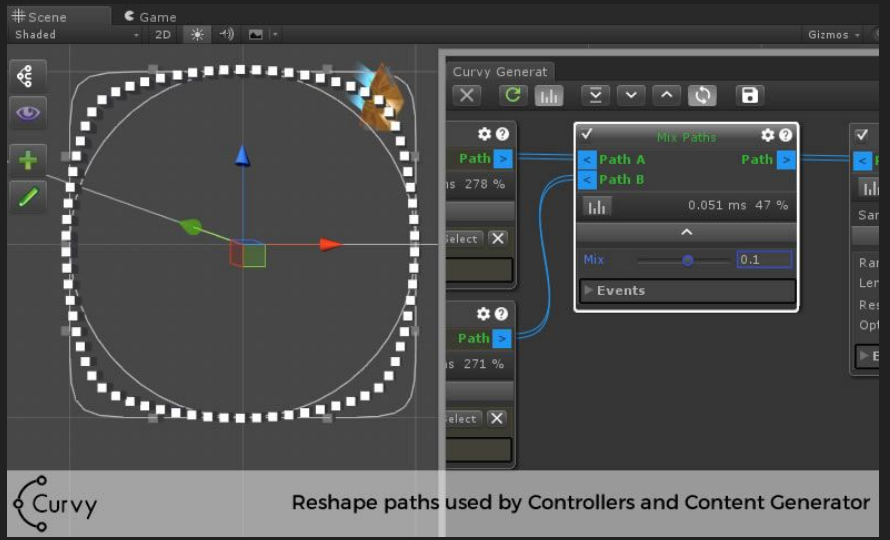


Unity Tools: Level editors



Unity Tools: Utilities

- Prop placement
  - Very useful to place props
  - Save lot of time from manual process
  - Procedural placement is more natural
- Spline tools
  - Very useful to place elements along paths
  - High level of control



- Snapping is very important (metrics)
- Hotkeys will increase your speed performance.

Unity Tools: Visual debug

- Gizmos
  - Very helpful to debug and visualize results.
  - Can be customized at runtime
- Accesible at Unity scriptable API
  - OnDrawGizmos
  - OnDrawGizmosSelected
  - Many more ...

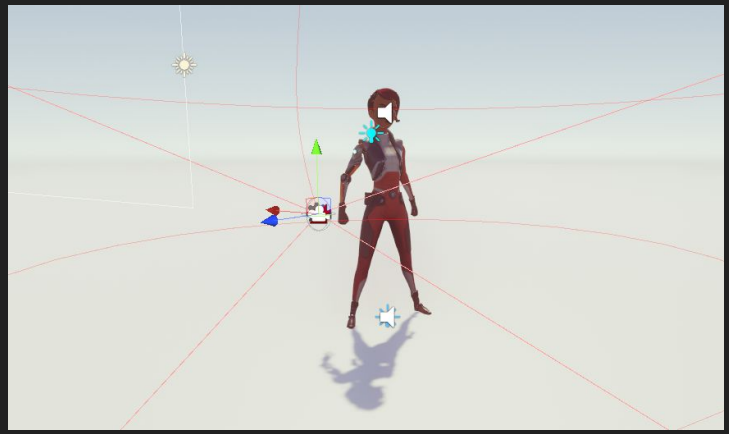
To understand what your team need,
don’t ask them.

Go to their desks and watch them work!

Unity Tools: Player setup

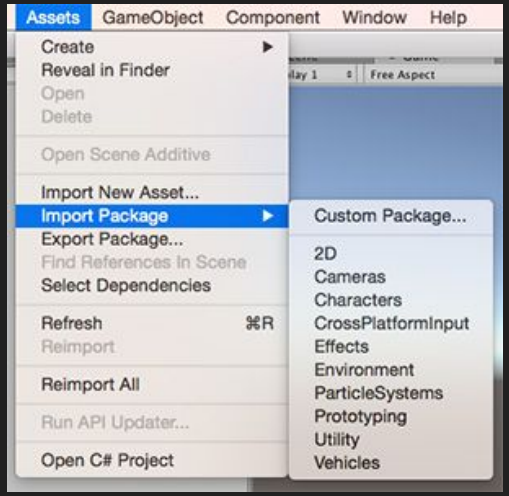
- First step: Setup reference dependency
  - Setup the camera rig (use given prefab)
  - Adjust camera rig position relative to the player
  - Setup scene layers.

- Custom player: Hints
  - Keep every functionality separated.
    - E.G: Player input, player controller.
  - Build a FSM machine for player logic
  - Use scriptable objects to maintain states.
    - Each state as scripted object
  - If needed, built different controllers and different prefabs. (e.g my player can transform into an animal)



Unity Tools: Distribution

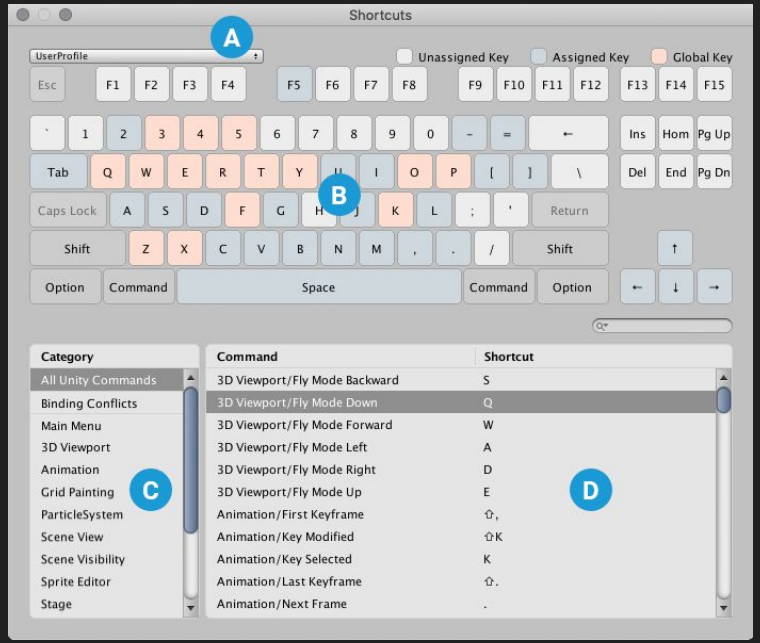
- **Asset packages**
  - Packages are collections of files and data in Unity.
  - They are stored and compressed in a single file
  - They can be exported and imported
- Packages types:
  - Standard Assets (built in packages)
  - Custom package (our case)
- Documentation
  - Document your plugin functionality
  - Show some examples (demo)



Unity tools: Hotkeys

They are more important than what you think!

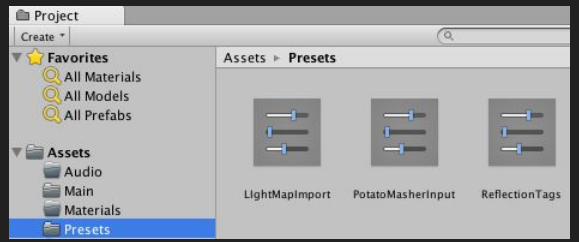
- View and manage unity shortcuts
- Most important unity shortcuts
  - Classic undo/redo/copy/paste
  - N - New gameobject
  - F - Focus selected game object
  - Q/W/E/R - Transforms
  - …



- Full hotkey list also at:
  https://blogs.unity3d.com/2011/08/24/unity-hotkeys-keyboard-shortcuts-in-unity/

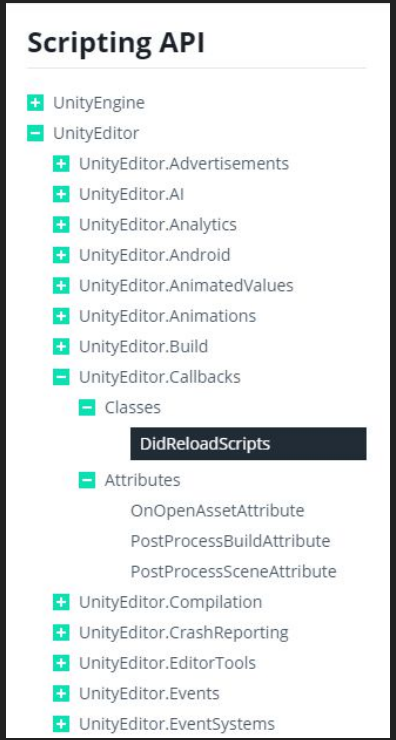
Unity tools: Presets

- To reuse property settings within multiple components
- Support for default settings.
- Also can be used as import settings.
- More information at: https://docs.unity3d.com/Manual/Presets.html



Unity tools: Scripting Introduction

- Resources available
  - Unity scripting API: https://docs.unity3d.com/ScriptReference/
  - Code exposed in custom tools
    - Directly accessible from project folder
    - External compiled .dll (.net refactor)
    - Editor tools must be placed at Editor folder.
  - External resource pages
    - https://code.tutsplus.com/tutorials/how-to-add-your-own-tools-to-unitys-editor--active-10047
    - https://docs.unity3d.com/Manual/BuildPlayerPipeline.html
- Include necessary: using UnityEditor;



Unity tools: Scripting Introduction

Attribute directives

- Markers that can be placed above a class, property or function
- Method of associating metadata or declarative information to code chunks
- Three types of attributes
  - Engine Attributes, listed at https://docs.unity3d.com/ScriptReference/AddComponentMenu.html
  - Engine Editor Attributes, listed at https://docs.unity3d.com/ScriptReference/CallbackOrderAttribute.html
  - .NET attributes, listed at https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/attributes/

- You can also create your own C# attributes
  - Custom attribute creation: https://developerhandbook.com/c-sharp/create-custom-csharp-attributes/

Unity tools: Attributes

Engine Attributes

- AddComponentMenu: https://docs.unity3d.com/ScriptReference/AddComponentMenu.html
- ContextMenu/ItemAttribute..:
- DisallowMultipleComponent: https://docs.unity3d.com/ScriptReference/DisallowMultipleComponent.html
- ExecuteAlways: https://docs.unity3d.com/ScriptReference/ExecuteAlways.html
- ExecuteInEditMode: https://docs.unity3d.com/ScriptReference/ExecuteInEditMode.html
- HeaderAttribute: https://docs.unity3d.com/ScriptReference/HeaderAttribute.html
- HelpURLAttribute: https://docs.unity3d.com/ScriptReference/HelpURLAttribute.html
- HideInInspector: https://docs.unity3d.com/ScriptReference/HideInInspector.html
- RangeAttribute: https://docs.unity3d.com/ScriptReference/RangeAttribute.html
- RequireComponent: https://docs.unity3d.com/ScriptReference/RangeAttribute.html
- SelectionBaseAttribute: https://docs.unity3d.com/ScriptReference/SelectionBaseAttribute.html

Unity tools: Attributes

Engine Attributes

- Serializefield: https://docs.unity3d.com/ScriptReference/SerializeField.html
- SpaceAttribute: https://docs.unity3d.com/ScriptReference/SpaceAttribute.html
- TextArea: https://docs.unity3d.com/ScriptReference/TextAreaAttribute.html
- Tooltip: https://docs.unity3d.com/ScriptReference/TooltipAttribute.html

Unity tools: Attributes

Editor Attributes

- CallbackOrder: https://docs.unity3d.com/ScriptReference/CallbackOrderAttribute.html
- CanEditMultipleObjects: https://docs.unity3d.com/ScriptReference/CanEditMultipleObjects.html
- CustomEditor: https://docs.unity3d.com/ScriptReference/CustomEditor.html
- CustomPropertyDrawer: https://docs.unity3d.com/ScriptReference/CustomPropertyDrawer.html
- DrawGizmo: https://docs.unity3d.com/ScriptReference/DrawGizmo.html
- InitializeOnLoadAttribute: https://docs.unity3d.com/ScriptReference/InitializeOnLoadAttribute.html
- MenuItem: https://docs.unity3d.com/ScriptReference/MenuItem.html
- PreferenceItem

Unity tools: Attributes

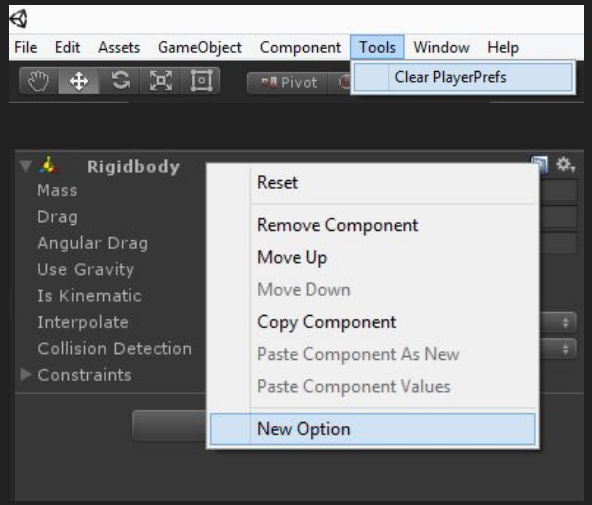
Miscelaneous Attributes

- Callbacks
  - OnOpenAssetAttribute: https://docs.unity3d.com/ScriptReference/Callbacks.OnOpenAssetAttribute.html
  - PostProcessSceneAttr:https://docs.unity3d.com/ScriptReference/Callbacks.PostProcessSceneAttribute.html
  - DidReloadScripts: https://docs.unity3d.com/ScriptReference/Callbacks.DidReloadScripts.html

- Editor tools:
  - EditorTool: https://docs.unity3d.com/ScriptReference/EditorTools.EditorToolAttribute.html

Unity Tools: Scripting basics

- Editor Tools work with context areas as seen before.
- The main context areas are:
  - Edit
  - Assets
  - Gameobjects
  - Components
  - Tools
  - Window
  - Help..



- We can create custom access tools for this context areas.
  - Menus to be accessed from toolbar or quick actions
  - Hotkeys to access this tools

Unity Tools: Scripting basics

- There are special paths that act as context menus, (toolbar and shortcut acces)
- Hotkeys are defined in the Command title, adding blank space before e.g ‘Tools/New Option %#a’

Command	Description
`[MenuItem("Assets/Load Additive Scene")]`	Available under the “Assets” menu, as well using right-click inside the project view.
`[MenuItem("Assets/Create/New Option")]`	Items will be listed when clicking on the “Create” button in the project view (useful when adding new types that can be added to the project)
`[MenuItem("CONTEXT/Rigidbody/New Option")]`	Available by right-clicking inside the inspector of the component.

Resource: https://unity3d.com/es/learn/tutorials/topics/interface-essentials/unity-editor-extensions-menu-items

Unity Tools: Scripting basics

- MenuCommand
  - Sometimes is necessary to have a reference to the component within Contextmenus
  - Passing MenuCommand as parameter to that function will give us a reference to the object.

Command	Description
`[MenuItem("Assets/ProcessTexture", true)]`	We can also block submenu actions by passing a boolean parameter
`[MenuItem("NewMenu/Option1", false, 1)]`	Priority within the menu list can be set by adding a numerical parameter.
`[ContextMenuItem("Randomize Name", "Randomize")]`	Customize attribute actions by allowing submenus context on them.

Unity Tools: Editor extensions

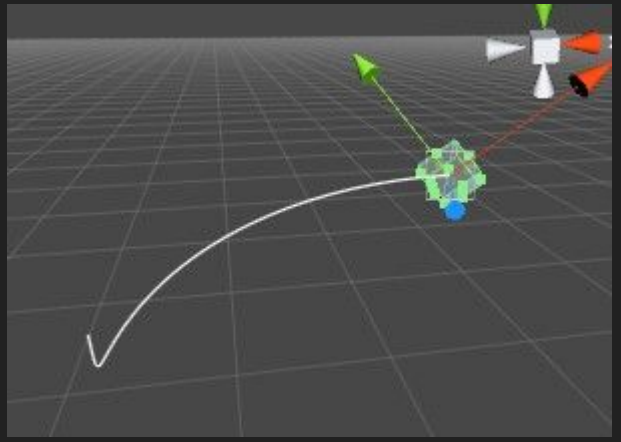
- Add menu items (hotkeys)
- Related menu classes
- Priority and order

```
// Add a new menu item that is accessed by right-clicking on an asset in the project view

[MenuItem("Assets/Load Additive Scene")]
private static void LoadAdditiveScene()
{
    var selected = Selection.activeObject;

EditorApplication.OpenSceneAdditive(AssetDatabase.GetAssetPath(selected));
}


}
```

```
[MenuItem("Assets/ProcessTexture")]
private static void DoSomethingWithTexture()
{
}

// Note that we pass the same path, and also pass "true" to the second argument.
[MenuItem("Assets/ProcessTexture", true)]
private static bool NewMenuOptionValidation()
{
    // This returns true when the selected object is a Texture2D (the menu item will be disabled otherwise).
    return Selection.activeObject.GetType() == typeof(Texture2D);
}
```

Resource: https://unity3d.com/es/learn/tutorials/topics/interface-essentials/unity-editor-extensions-menu-items?playlist=17117

Unity Tools: Handles & gizmos



- Handles
  - Custom 3D gui controls in the scene view
  - You can define your own custom handles
  - Very useful to realtime control groups and entities on the scene view.
  - Focused in object manipulation
  - Scripting API, many mathematical functions already in.
  - Reference: https://docs.unity3d.com/ScriptReference/Handles.html

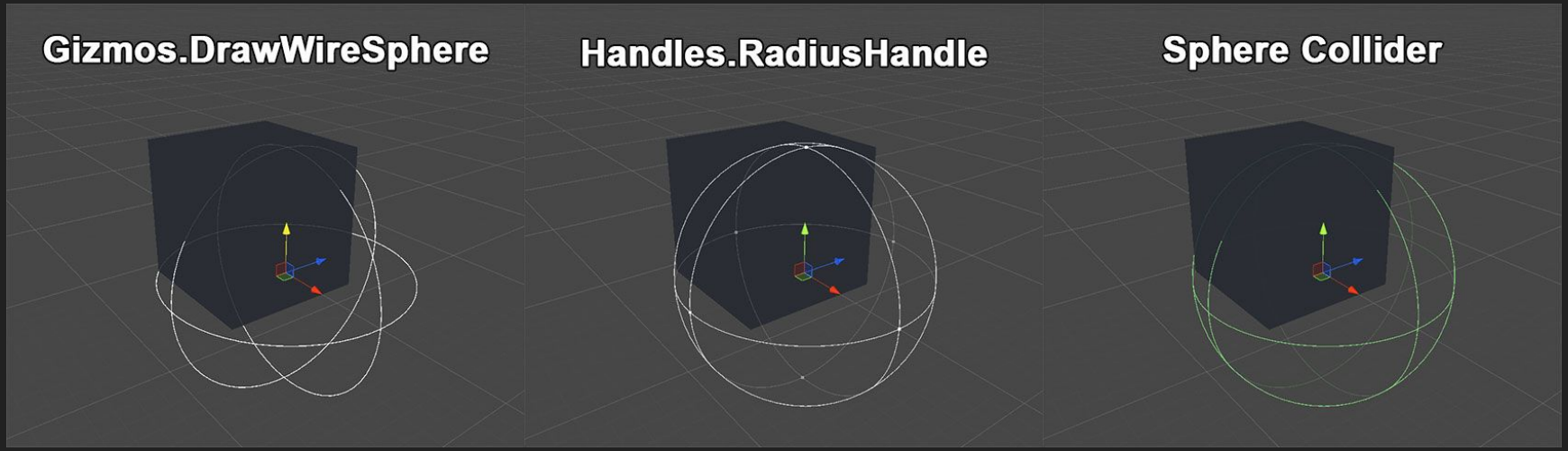
- Gizmos
  - Gizmos are collection of built-in handles
  - These are normally primitive shapes to manipulate objects
  - Reference: https://docs.unity3d.com/ScriptReference/Gizmos.html

Unity Tools: Methods list

- Editor reserved method names, used for different purposes.

Methods	Description
OnDrawGizmos	Draw gizmo on scene
OnDrawGizmoSelected	Draw gizmo when object selected on scene
OnInspectorGUI	Draws custom component properties
OnSceneGUI	Draws GUI on scene
OnGUI	Draw custom window GUI
OnValidate	When a property of a component is changed.
DrawDefaultInspector	Draws default inspector GUI
Repaint	Used to repaint the inspector GUI

Unity Tools: Inspector tools



Unity Tools: Inspector tools

- Allow you to customize your script interface [CustomEditor(typeof(LevelScript))]
- Make every script so that artists and other team members can easily understand it and use it.
- Easily extend with DrawDefaultInspector()

```
using UnityEngine;
using System.Collections;
using UnityEditor;

[CustomEditor(typeof(LevelScript))]
public class LevelScriptEditor : Editor
{
    public override void OnInspectorGUI()
    {
        LevelScript myTarget = (LevelScript)target;
        myTarget.experience = EditorGUILayout.IntField("Experience",
myTarget.experience);
        EditorGUILayout.LabelField("Level",myTarget.Level.ToString());
    }
}
```

```
using UnityEngine;
using System.Collections;
using UnityEditor;

[CustomEditor(typeof(SomeScript))]
public class SomeScriptEditor : Editor
{
    public override void OnInspectorGUI()
    {
        DrawDefaultInspector();

        EditorGUILayout.HelpBox("This is a help box", MessageType.Info);
    }
}
```

Resource: https://unity3d.com/es/learn/tutorials/topics/interface-essentials/drawdefaultinspector-function?playlist=17117

Resources

- https://unity3d.com/es/learn/tutorials/topics/interface-essentials/building-custom-inspector?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/interface-essentials/drawdefaultinspector-function?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/interface-essentials/adding-buttons-custom-inspector?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/interface-essentials/unity-editor-extensions-menu-items?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/scripting/introduction-editor-scripting?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/scripting/creating-spline-tool?playlist=17117
- https://unity3d.com/es/learn/tutorials/topics/scripting/getting-started-ik?playlist=17117