

```
using UnityEditor;
using UnityEngine;
```

```
[CustomEditor(typeof(ProceduralTerrain))]
[CanEditMultipleObjects]
public class ProceduralTerrainEditor : Editor {
```

```
    public override void OnInspectorGUI() {
        serializedObject.Update();
```

```
        EditorGUILayout.PropertyField(serializedObject.FindProperty("TerrainSize"));
        EditorGUILayout.PropertyField(serializedObject.FindProperty("CellSize"));
```

```
        if (GUILayout.Button("Generate"))
            (serializedObject.targetObject as ProceduralTerrain).GenerateTerrain();
```

```
        serializedObject.ApplyModifiedProperties();
```

```
    }
```

```
}
```

To create  
a custom editor  
we must inherit from  
the Editor class

```
using UnityEditor;
using UnityEngine;
```

```
[CustomEditor(typeof(ProceduralTerrain))]
[CanEditMultipleObjects]
public class ProceduralTerrainEditor: Editor {
```

```
    public override void OnInspectorGUI() {
        serializedObject.Update();
```

Make sure we're editing fresh data

```
        EditorGUILayout.PropertyField(serializedObject.FindProperty("TerrainSize"));
        EditorGUILayout.PropertyField(serializedObject.FindProperty("CellSize"));
```

```
        if (GUILayout.Button("Generate"))
            (serializedObject.targetObject as ProceduralTerrain).GenerateTerrain();
```

```
        serializedObject.ApplyModifiedProperties();
```

```
    }
```

```
}
```