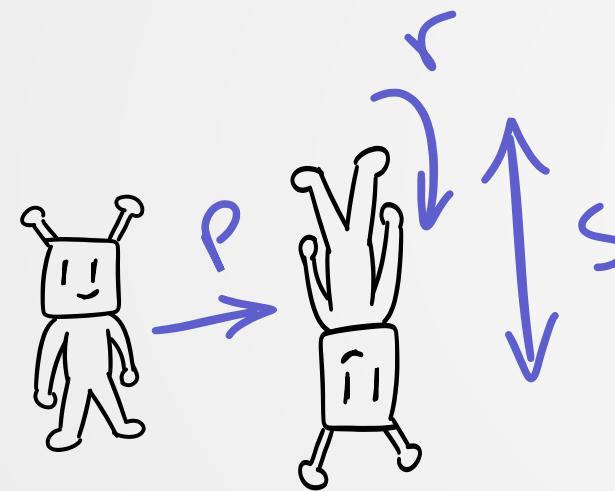
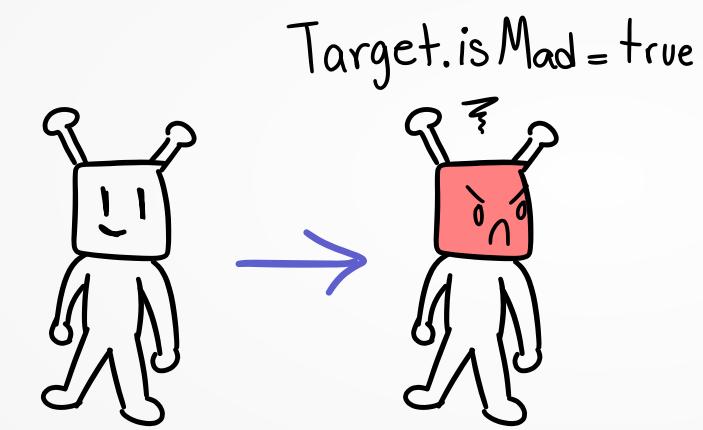


PERSISTENCIA

Undo. Record Object (*, message);



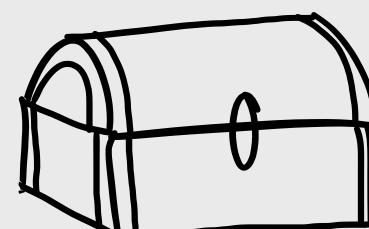
transform



Target



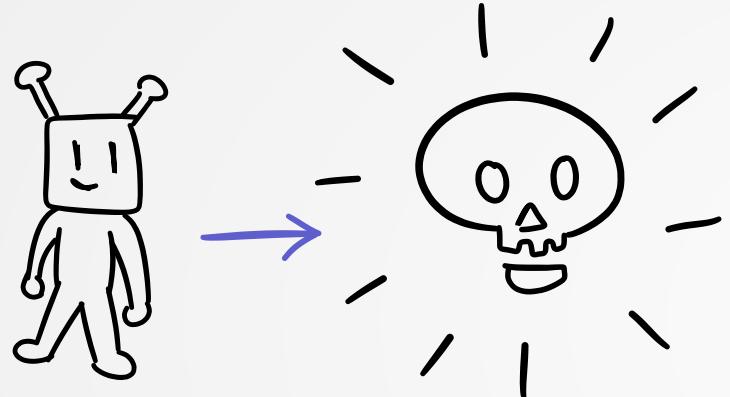
Target.gun; transform;



PERSISTENCIA

No!!

`Destroy(Target.gameObject);` X



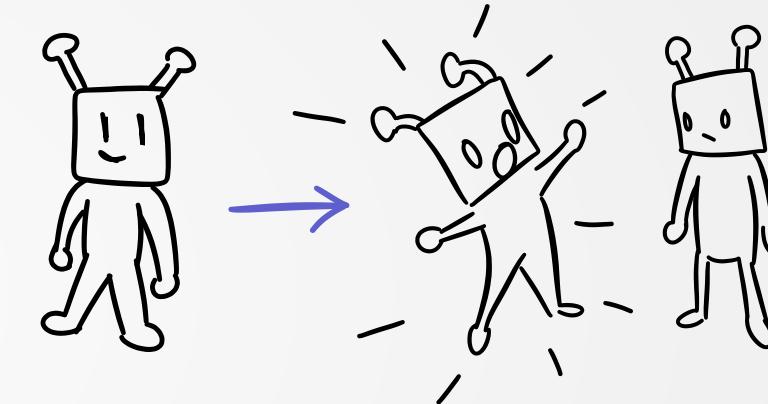
`Undo.DestroyObjectImmediate`

`Util.Destroy(x)`

`UnityEngine.Object`

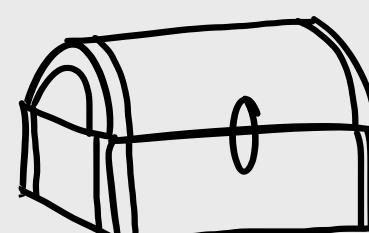
Platform
dependant
Compilation

`c = Instantiate(Target);`



`Undo.RegisterCreatedObjectUndo(c)`

`Util.Instantiate`



PERSISTENCIA

[System.Serializable]

Visible y editable
en el inspector!

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

[System.Serializable]
public class Stats{
    public RenewableStat hp;
    public RenewableStat mp;

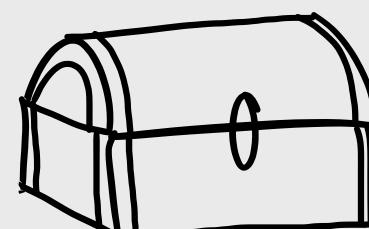
    public int agility;
    public int perception;
    public int attackPower;
    public float attackRadius;
}
```

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

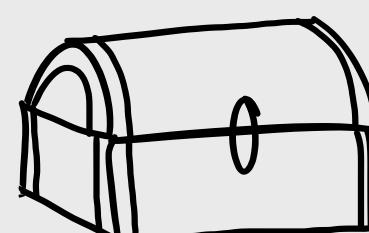
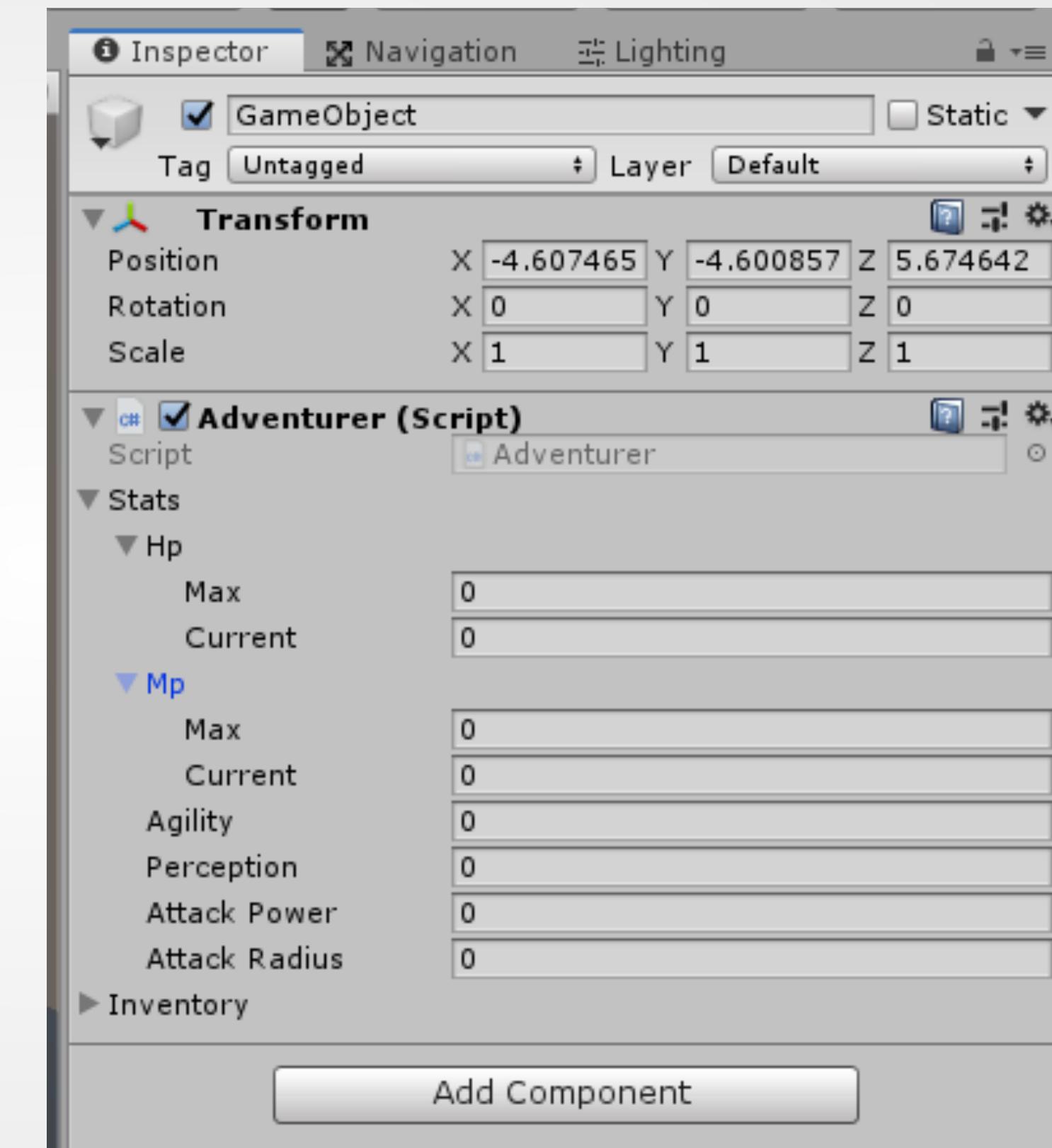
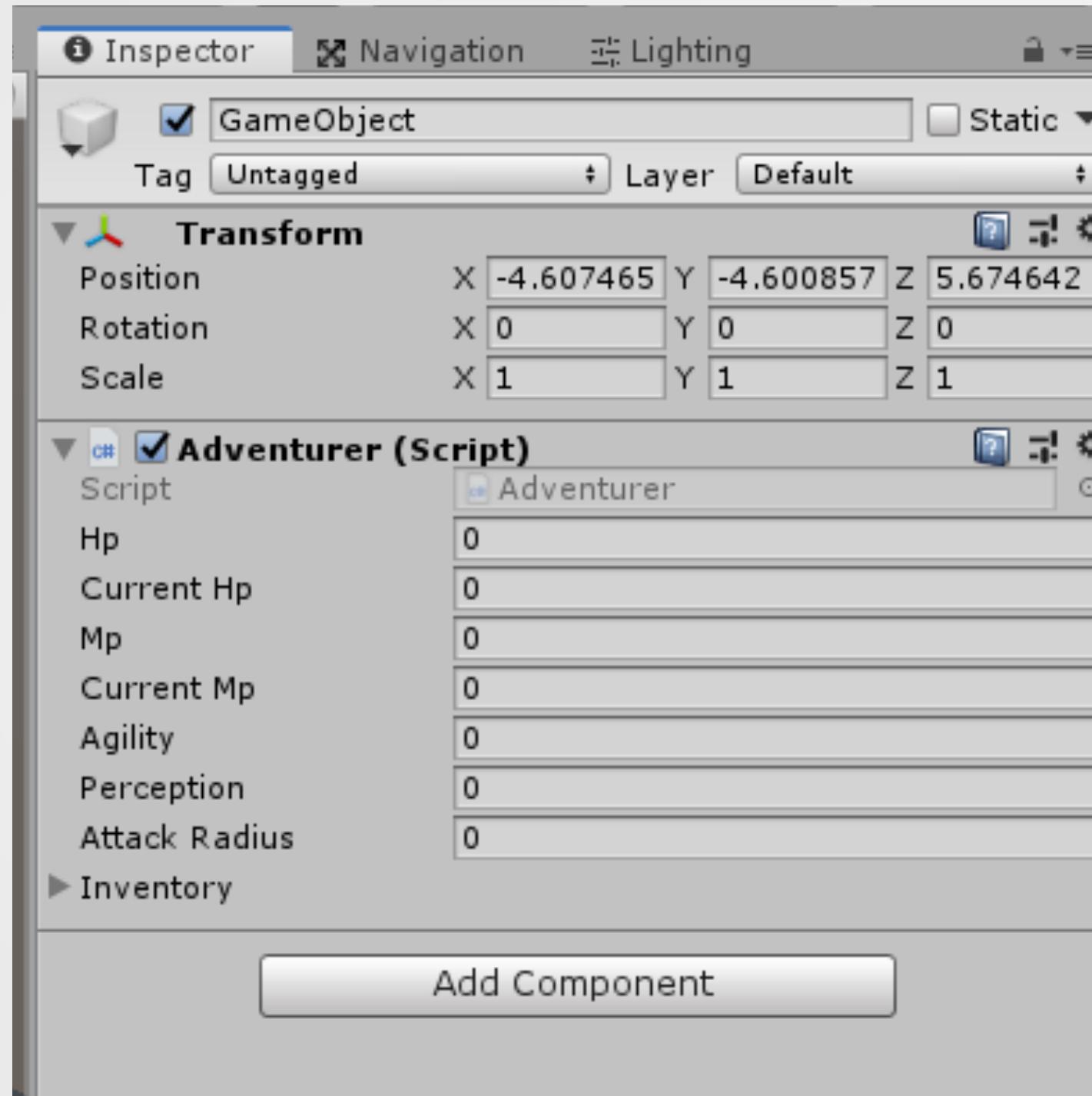
public class Adventurer : MonoBehaviour {
    public Stats stats;
    public List<GameObject> inventory;
}

using UnityEngine;
using System.Collections;
using System.Collections.Generic;

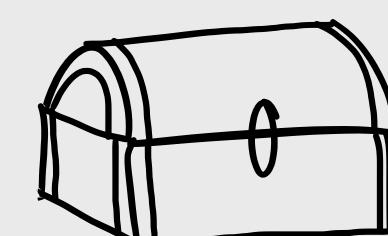
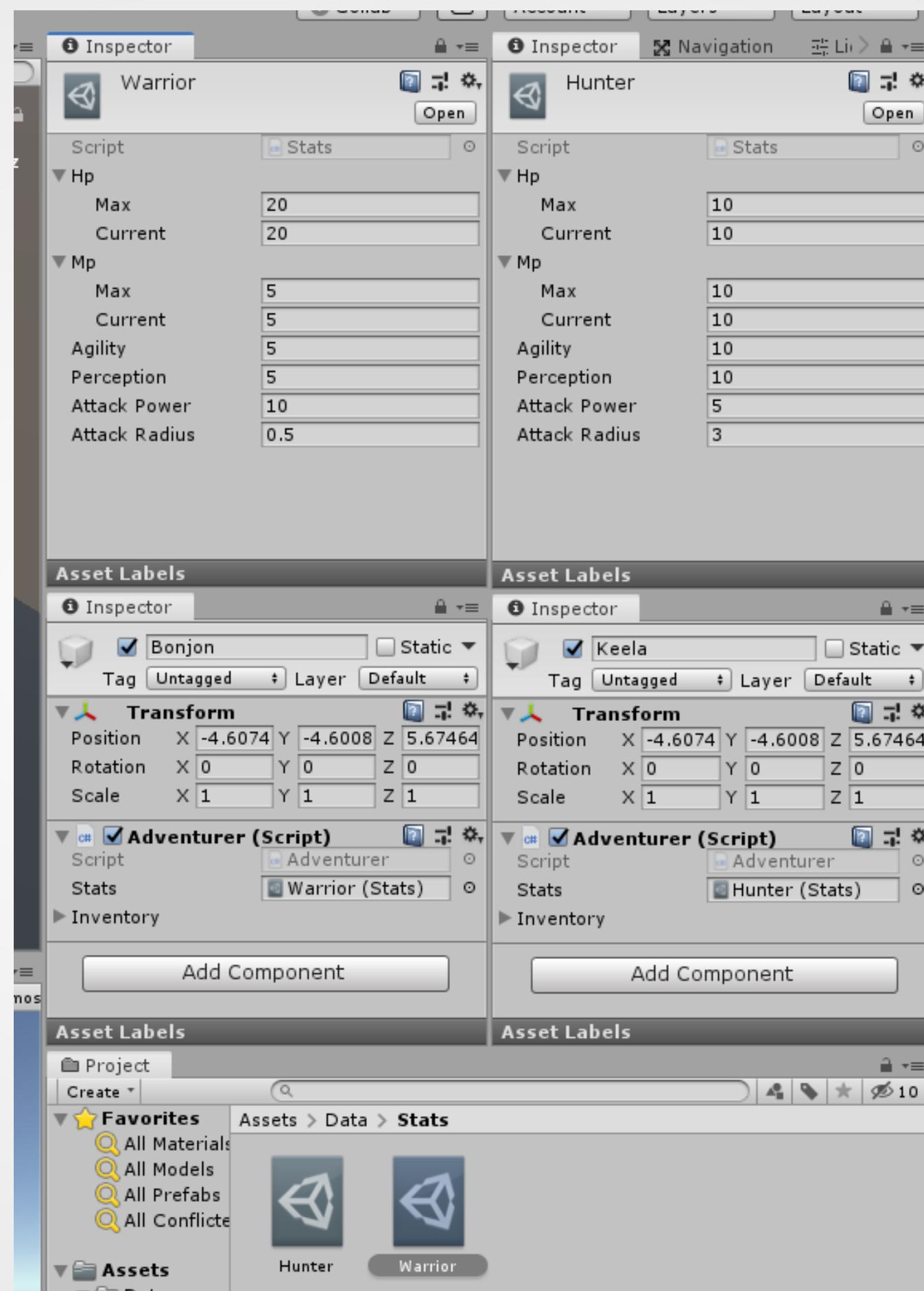
[System.Serializable]
public class RenewableStat {
    public int max = 10;
    public int current = 0;
}
```



PERSISTENCIA



PERSISTENCIA



PERSISTENCIA

The screenshot shows the Unity Editor interface with two open scripts:

- Stats.cs** (Git:master C#/1 +4 yas):

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

[System.Serializable]
[CreateAssetMenu(menuName = "Stats")]
public class Stats : ScriptableObject {
    public RenewableStat hp;
    public RenewableStat mp;

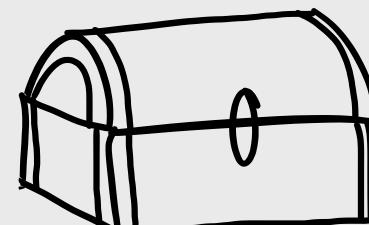
    public int agility;
    public int perception;
    public int attackPower;
    public float attackRadius;
}
```
- Adventurer.cs** (Git:master C#/1 +2 yas):

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class Adventurer : MonoBehaviour {
    public Stats stats;
    public List<GameObject> inventory;

    // ...
}
```

The Unity Editor interface includes a toolbar at the top with icons for Record, Play, Prefab, and Create. Below the toolbar is a docked panel titled "Code" containing icons for various tools likeinkscape 0.92.4, GIMP, Chromium, Inkscape, Krita, Neko, Tabbar, Visual Studio Code, and others. At the bottom of the screen, there's a docked panel titled "Code" with icons for Blender 2.9, VLC media player, Unity Hub, Spotify, Tor Browser, Free Download, DOSBox 0.74-3, curl-7.65.3..., Discord, Start Tor Browser, Minecraft, EZBlocker, and oculus-go... . The status bar at the bottom right shows the time as 2:38 PM and the date as 9/15/2019.



PERSISTENCIA



Dictionary X

Interface X

private *
variables

Si no se ve, no se guarda*

PERSISTENCIA

guarda variables privadas
y las muestra en
el inspector

[SerializeField]

Static X
get{} set{} X

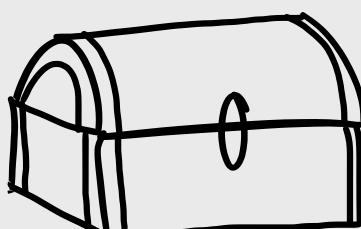
dictionary X

interfaces X

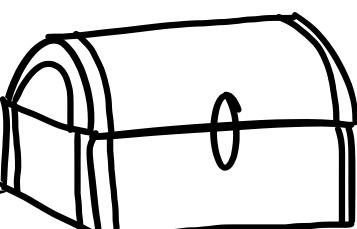
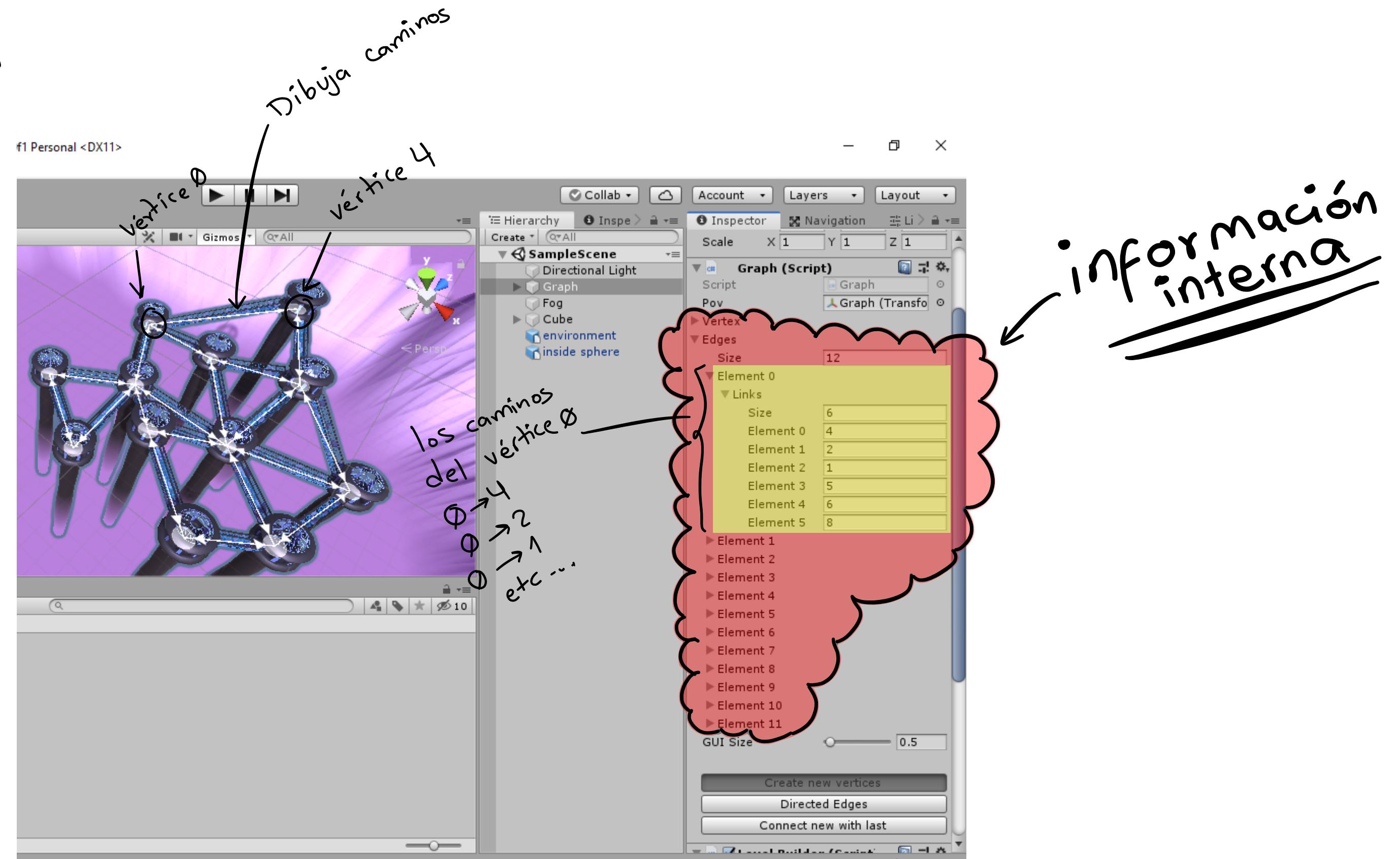
Esconde cosas visibles
en el inspector

[HideInInspector]

(pero, si eran visibles,
las guarda)



PERSISTENCIA



PERSISTENCIA

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class Graph : MonoBehaviour {
    public event System.Action onGraphModified;

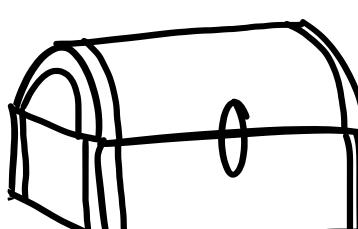
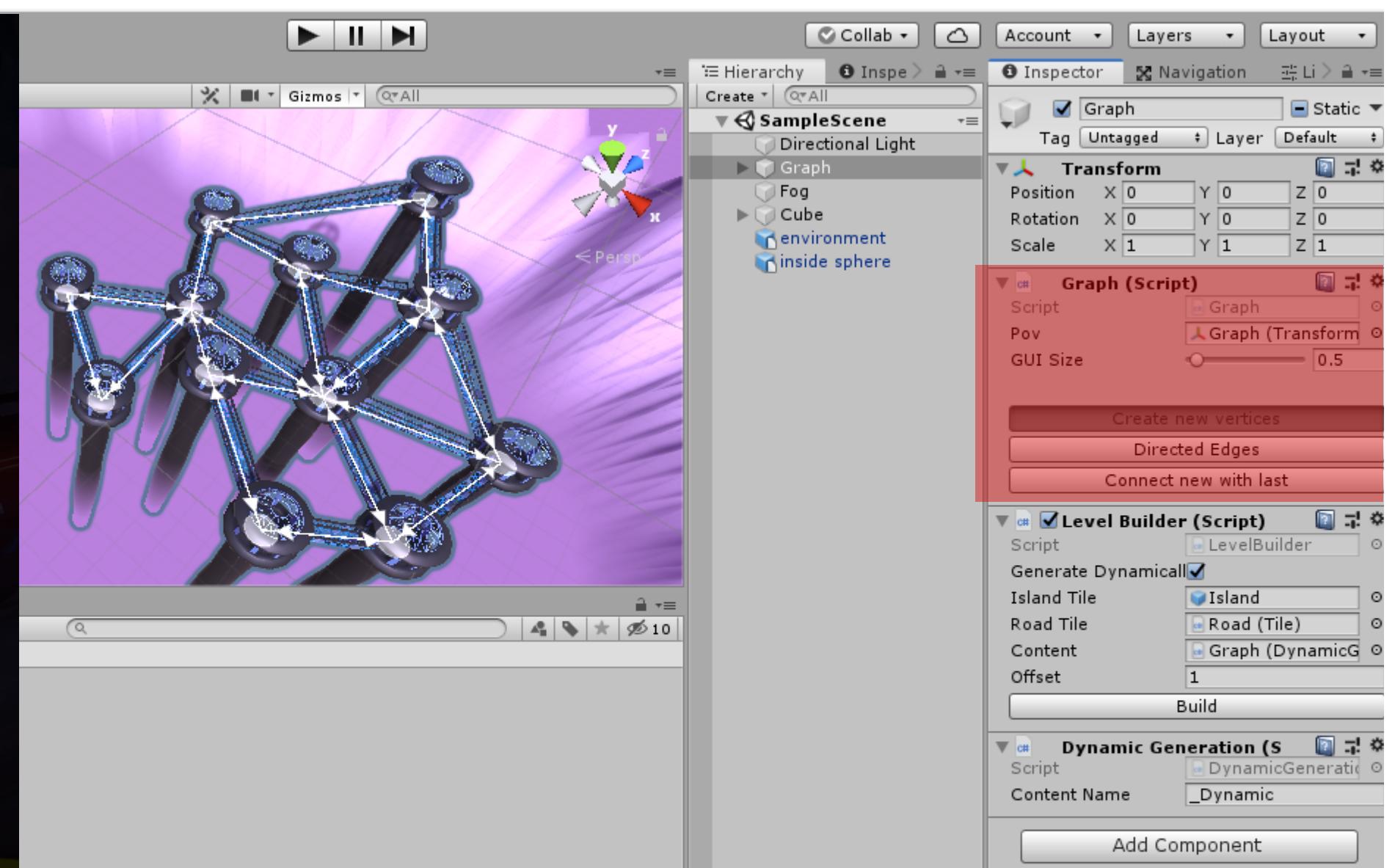
    [SerializeField]
    Transform _pov = null;

    [HideInInspector]
    public List<Vector3> vertex = new List<Vector3>();
    [HideInInspector]
    public List<Edge> edges = new List<Edge>();

    public Vector3 ToWorldPoint (Vector3 point) {
        return PoV.TransformPoint(point);
    }

    public Vector3 ToLocalPoint (Vector3 point) {
        return PoV.InverseTransformPoint(point);
    }

    public void SetVertex (int index, Vector3 pos) {
        vertex[index] = pos;
    }
}
```

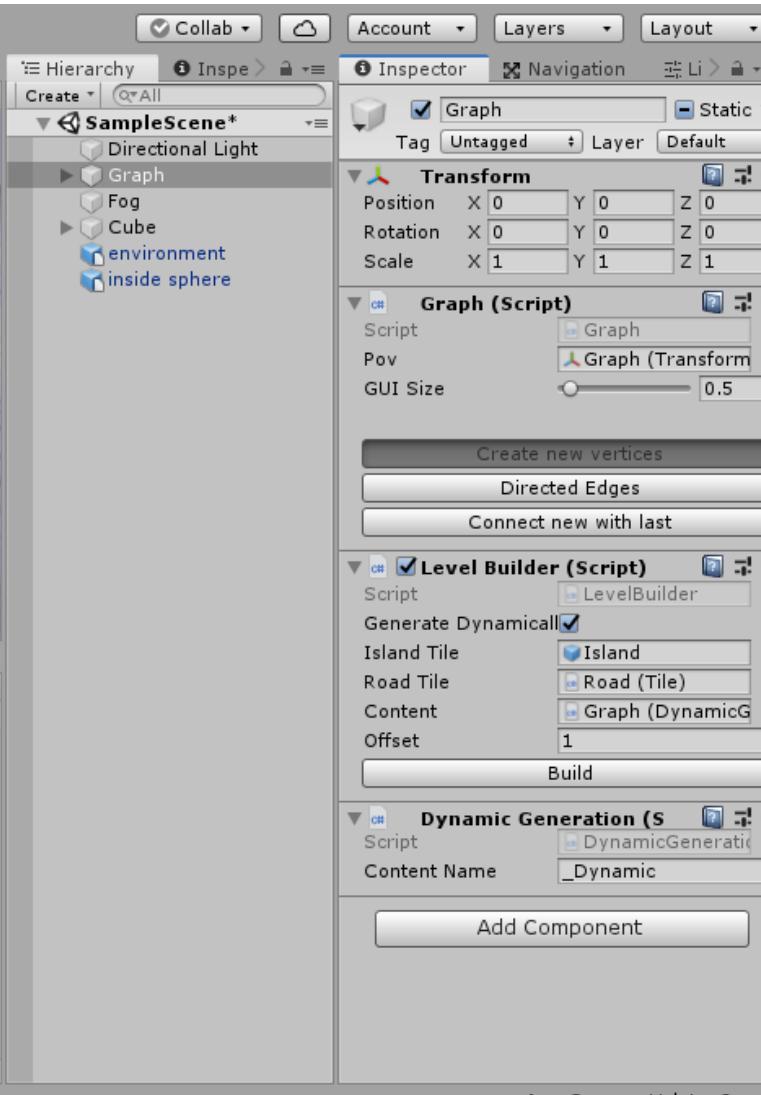


PERSISTENCIA

1

Mostrar y
Configurar

```
[ExecuteInEditMode]
public class LevelBuilder : MonoBehaviour {
    Graph _g;
    Graph _Graph { get { if (!_g) _g = GetComponent<Graph>(); return _g; } }
    public bool generateDynamically = false;
    [SerializeField] GameObject _islandTile;
    [SerializeField] Tile _roadTile;
    [SerializeField] DynamicGeneration _content;
    [SerializeField] float _offset = 1;
    void OnEnable () {
        _Graph.onGraphModified += GraphModifiedHandler;
    }
    public void GraphModifiedHandler () {
        if (generateDynamically) {
            Build();
        }
    }
}
```



2

Ocultar y
crear prefab

```
[ExecuteInEditMode]
public class LevelBuilder : MonoBehaviour {
    Graph _g;
    Graph _Graph { get { if (!_g) _g = GetComponent<Graph>(); return _g; } }
    public bool generateDynamically = false;
    [HideInInspector]
    [SerializeField]
    GameObject _islandTile;
    [HideInInspector]
    [SerializeField]
    Tile _roadTile;
    [HideInInspector]
    [SerializeField]
    DynamicGeneration _content;
    [HideInInspector]
    float _offset = 1;
    void OnEnable () {
        _Graph.onGraphModified += GraphModifiedHandler;
    }
    public void GraphModifiedHandler () {
        if (generateDynamically) {
            Build();
        }
    }
}
```

