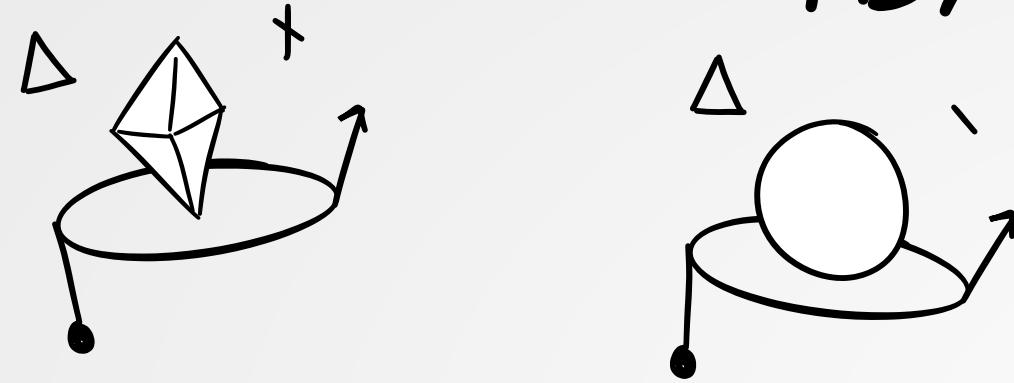


# Cómo implementar niveles

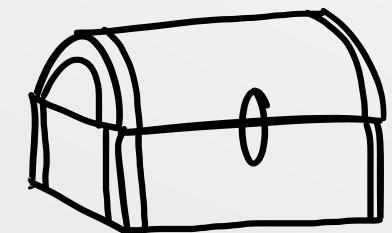
(sin morir en el intento)



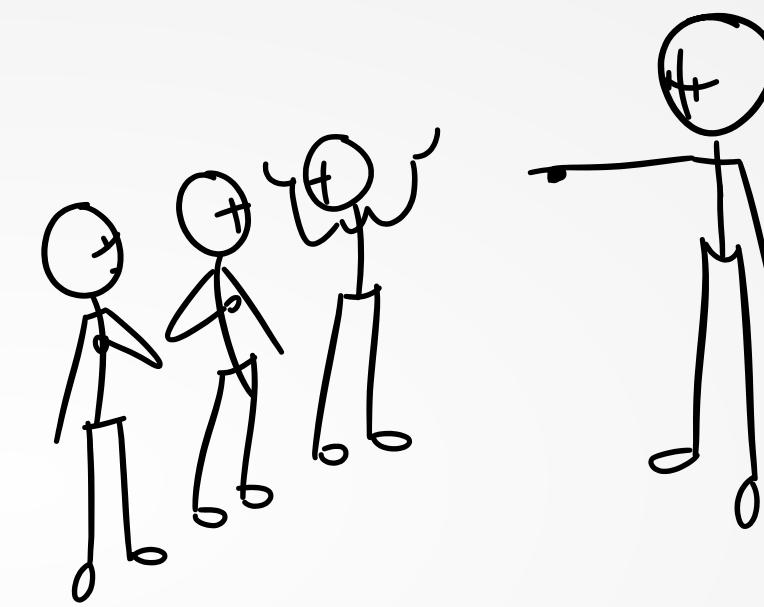
REUSABILIDAD



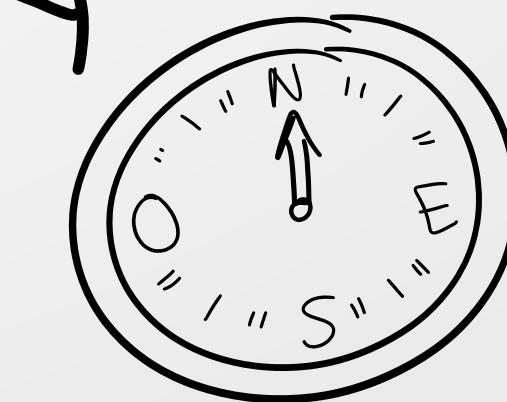
PERSISTENCIA



COMUNICACIÓN

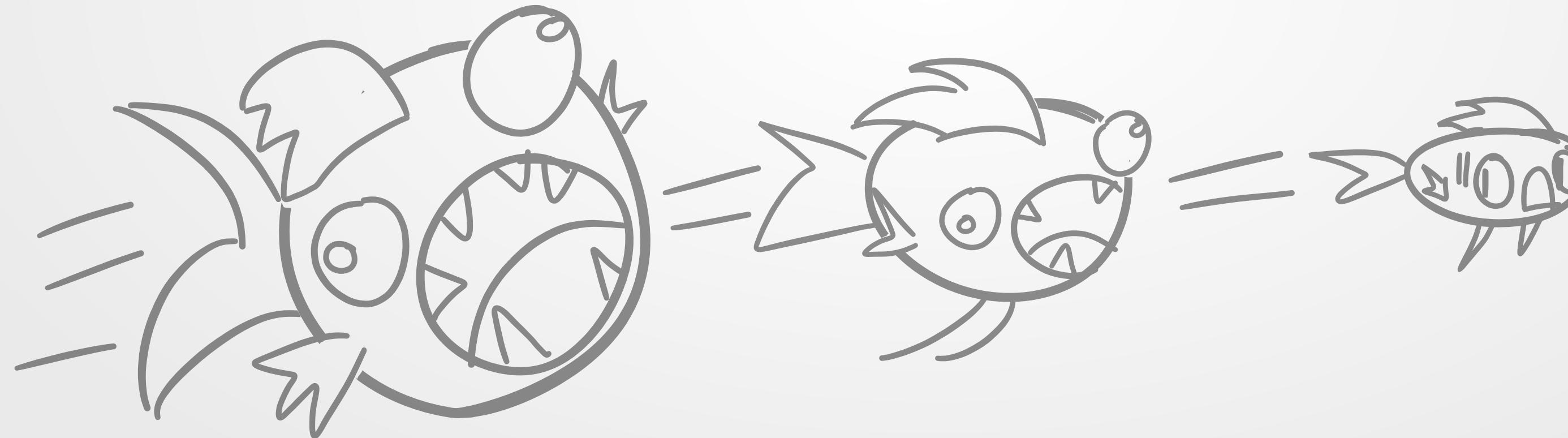


P.O.V Y SCENE



# EDITOR SCRIPTING

Editor → MonoBehaviour → Game Object



# EDITOR SCRIPTING

## Build!

Editor → MonoBehaviour → Game Object



# EDITOR SCRIPTING

```
[CustomEditor(typeof(MiScript))]
public class MiEditor : Editor {
    override void OnInspectorGUI() {
        }
    void OnSceneGUI() {
        }
}
```

*Cosas del inspector*

*Cosas de la escena*

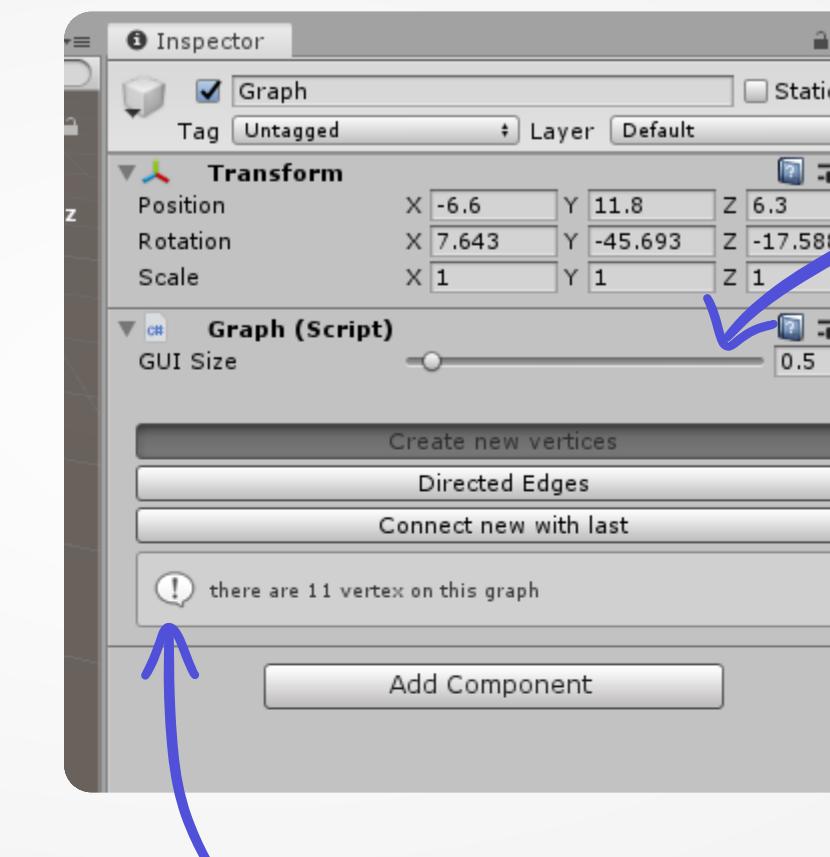
*MonoBehaviour*

*está en UnityEditor namespace*

The image contains two hand-drawn diagrams. The top diagram, labeled 'Cosas del inspector', shows a 3D coordinate system with axes (x, y, z) and three small circles representing selected objects. A wavy line connects this diagram to the 'OnInspectorGUI()' code block. The bottom diagram, labeled 'Cosas de la escena', shows a camera icon with a field of view angle 'α', a pink circle, a stick figure inside an oval with radius 'r', and a 3D coordinate system with axes.

# EDITOR SCRIPTING

OnInspectorGUI()



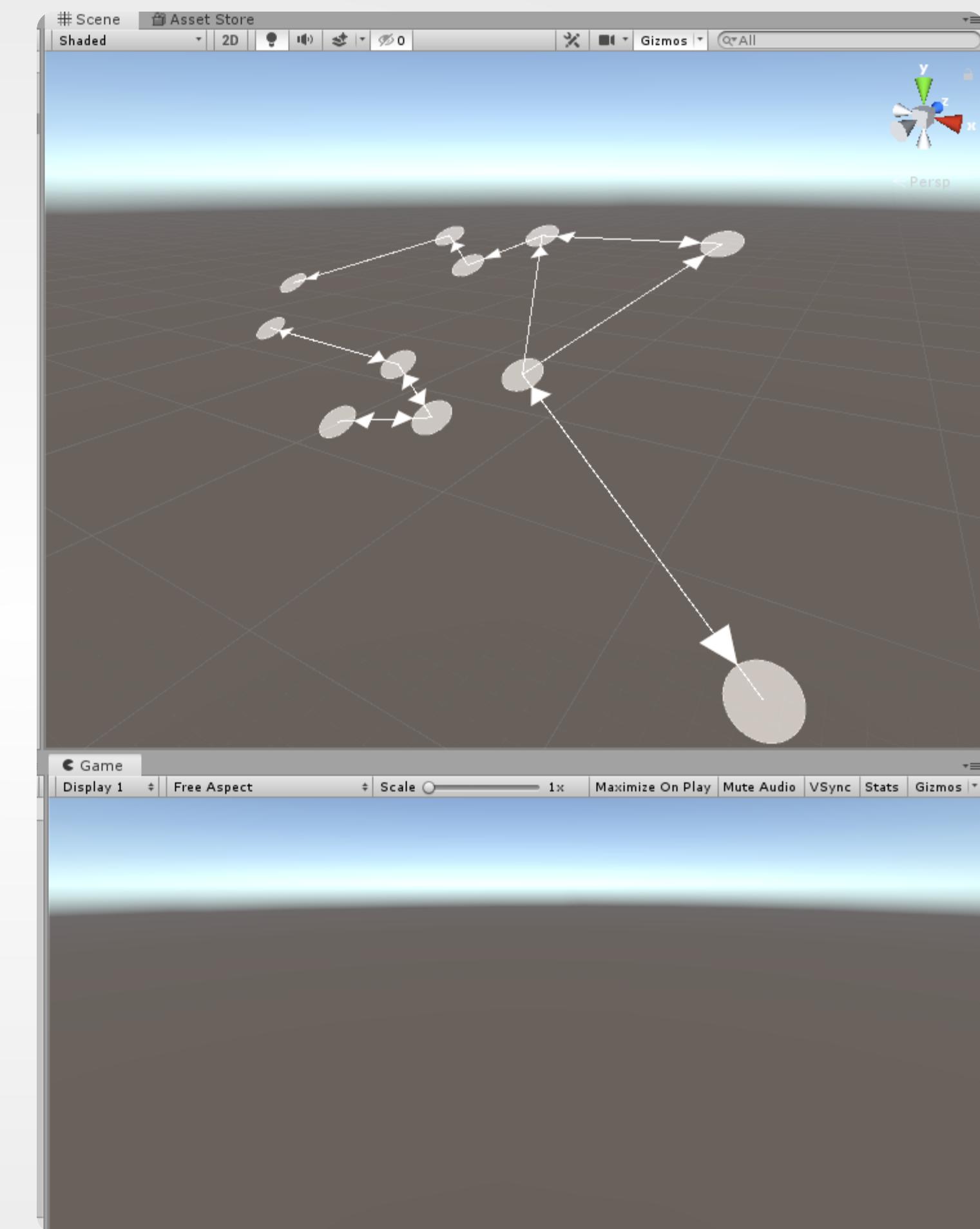
many DEBUGGY

such BUTTON  
very SLIDER

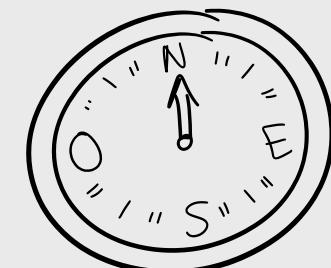
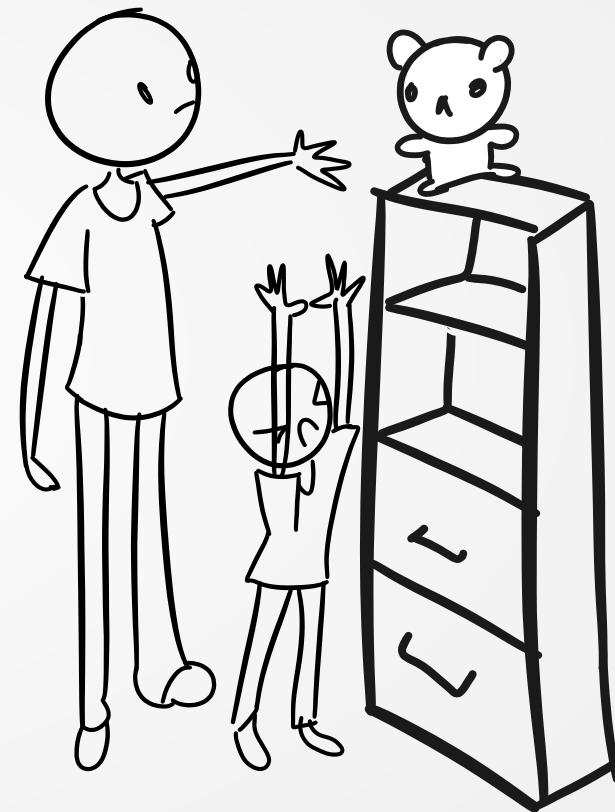
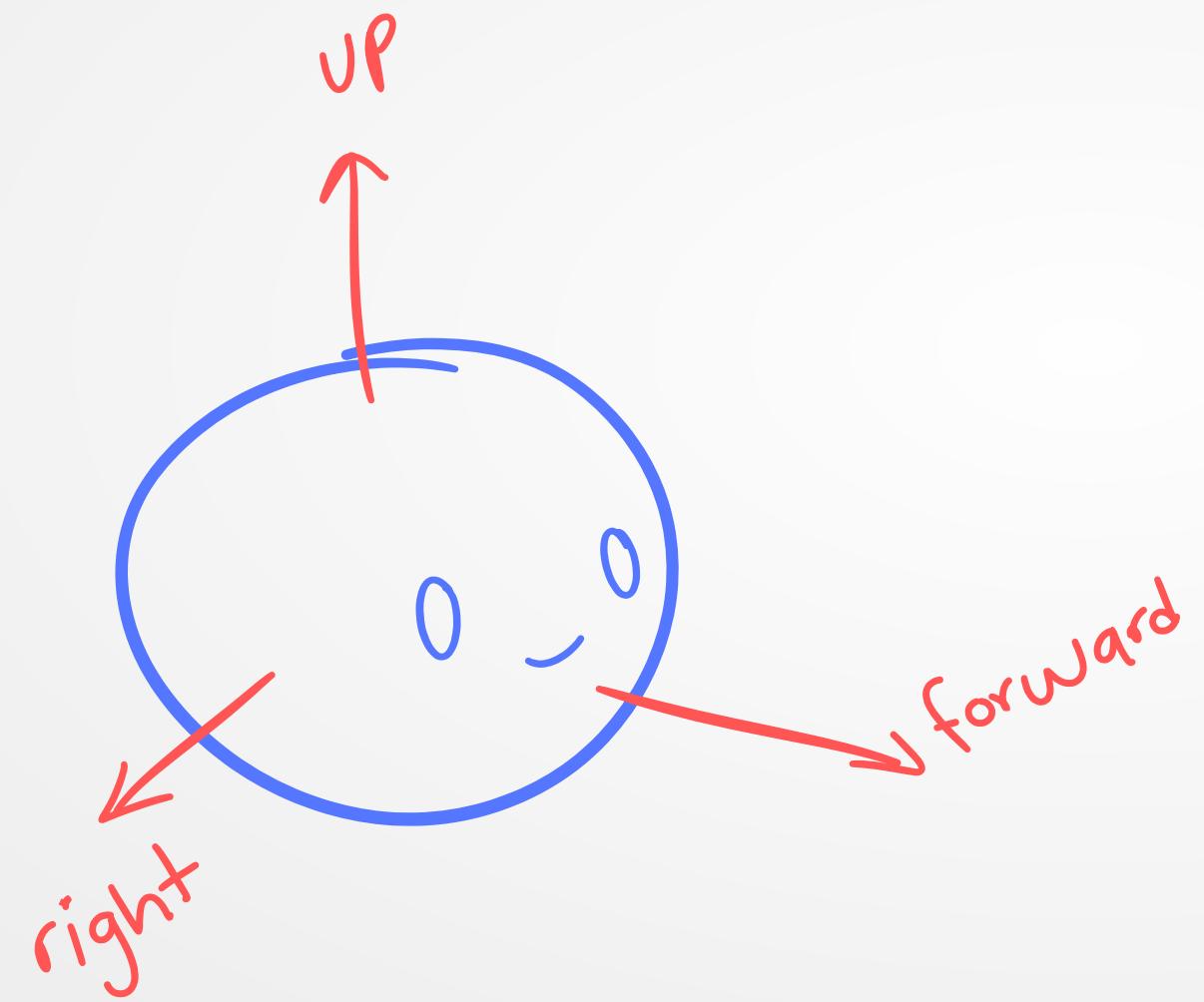


# EDITOR SCRIPTING

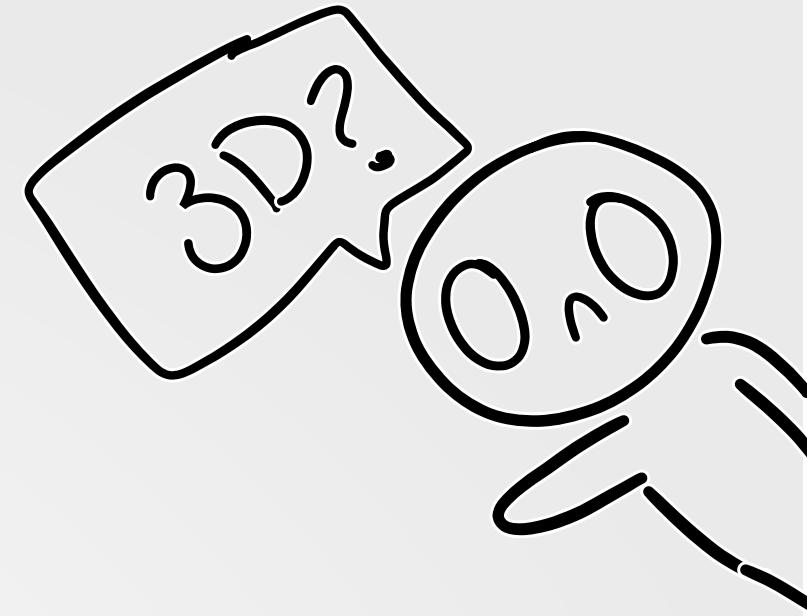
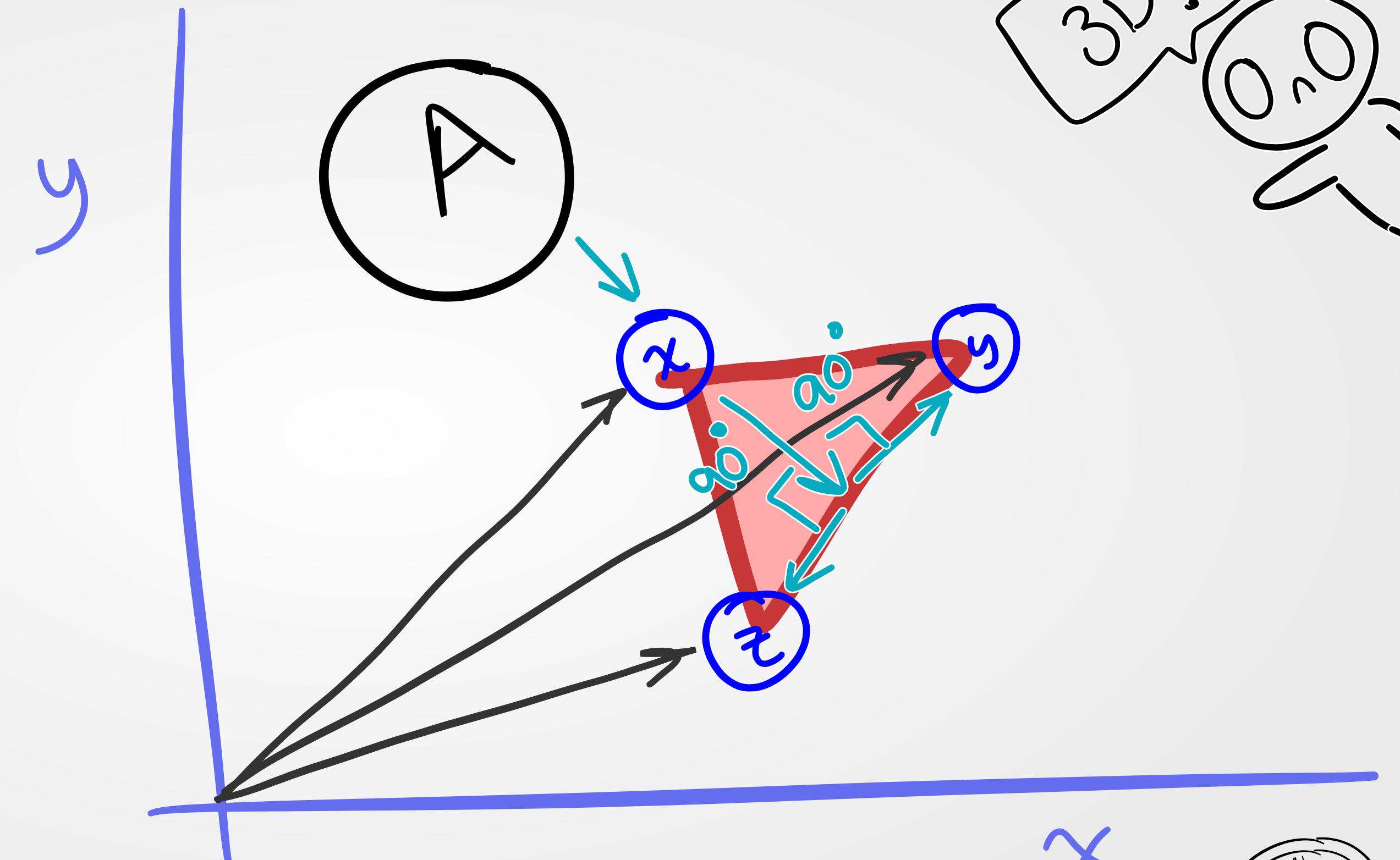
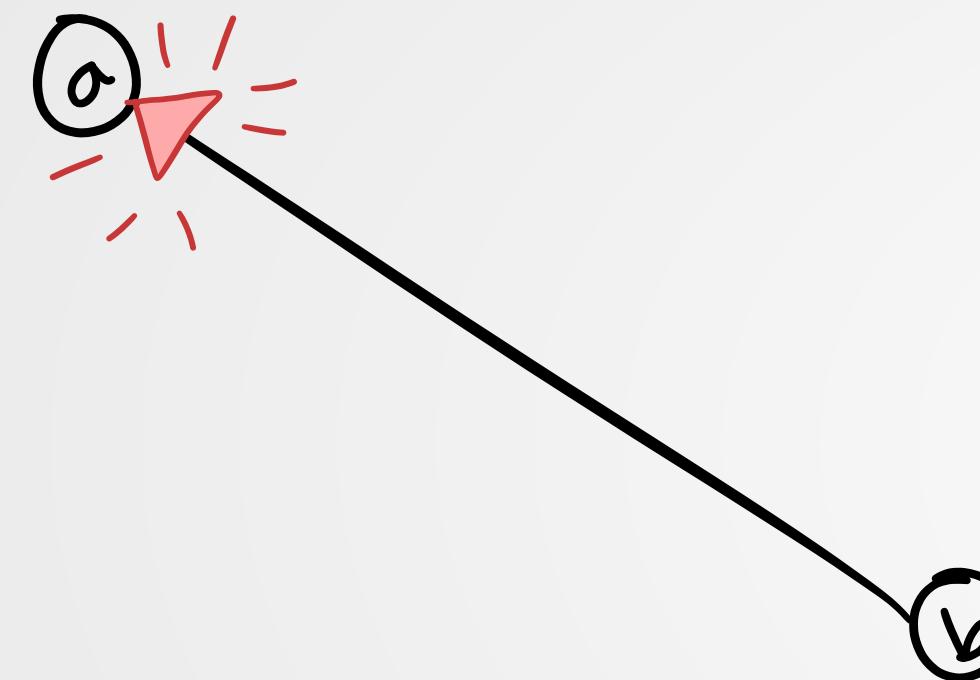
OnSceneGUI ()



# PoV y SCENE



PoV y SCENE

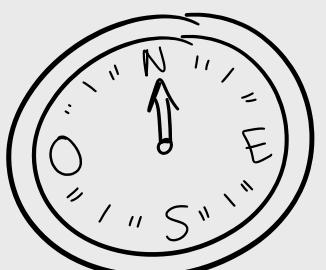


PoV y SCENE

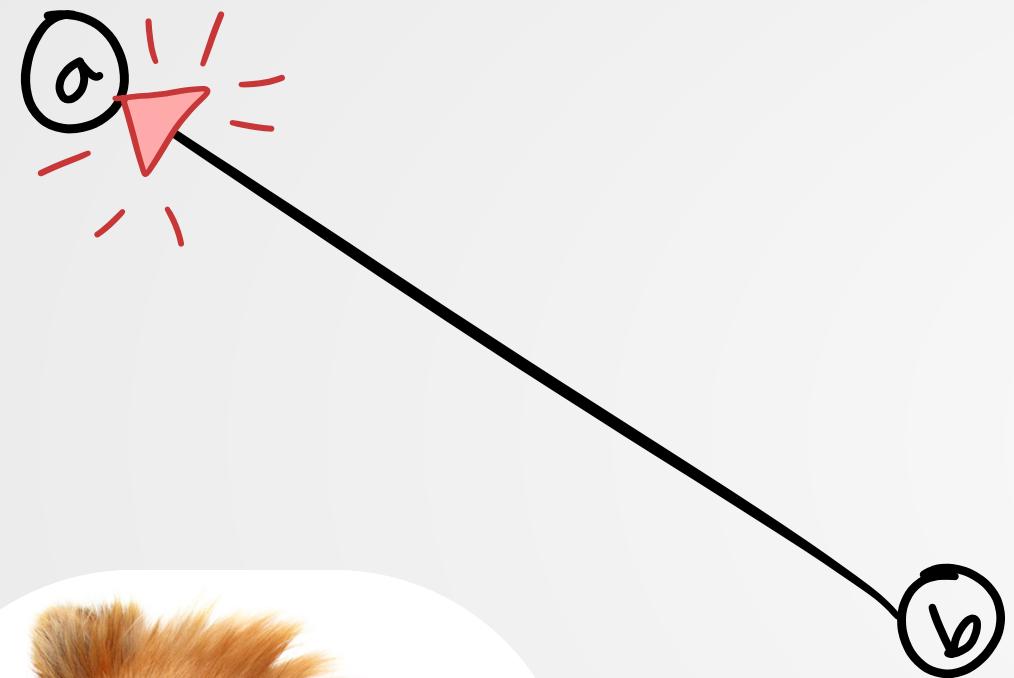
~~HanVes. Matrix~~

$$\begin{bmatrix} P_1 & P_2 & P_3 & P_4 \\ r_1 & r_2 & r_3 & r_4 \\ s_1 & s_2 & s_3 & s_4 \\ \textcircled{\text{O}} & \textcircled{\text{O}} & \textcircled{\text{O}} & 1 \end{bmatrix}$$

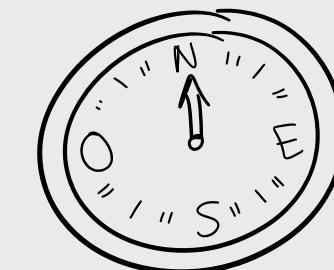
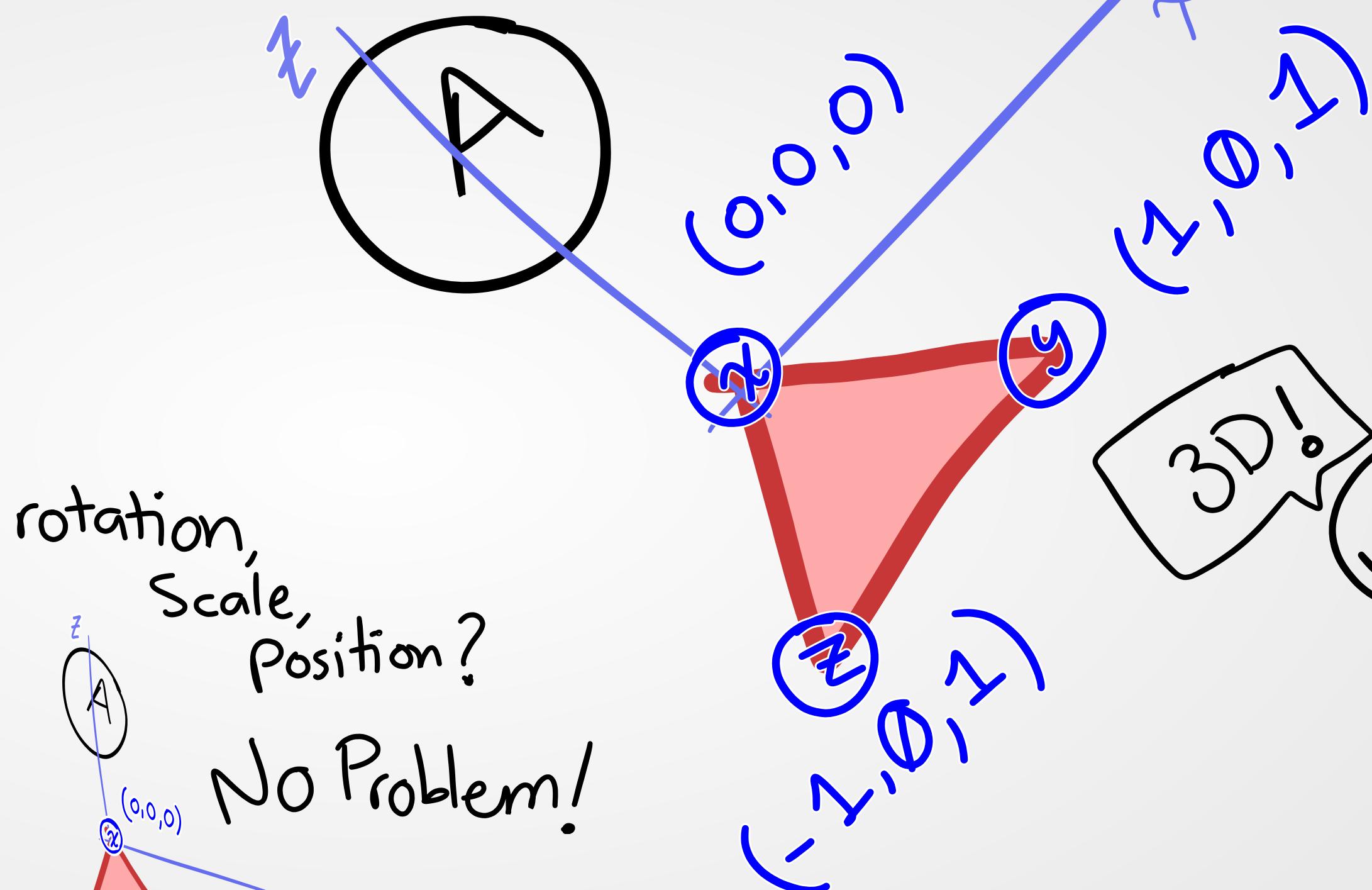
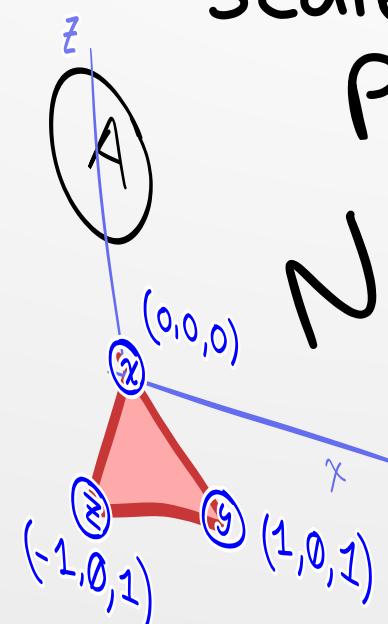
Matrix  $4 \times 4$   
transformation  
matrix



# PoV y SCENE



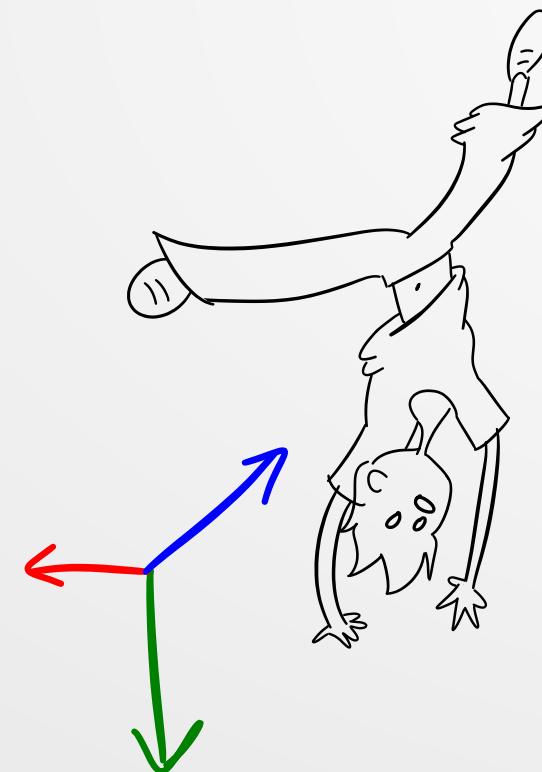
rotation,  
Scale,  
Position?  
No Problem!



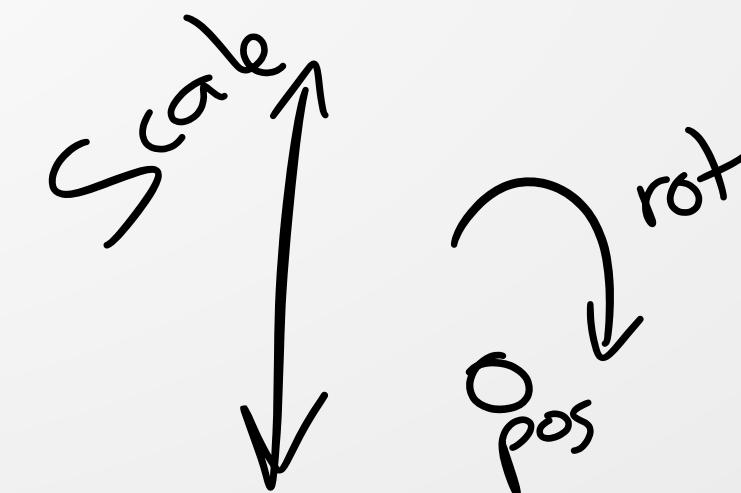
PoV y SCENE

# Cómo Crear una Matriz $4 \times 4$ ?

`transform.localToWorldMatrix`



`Matrix4x4TRS(pos, rot, scale)`



PoV y SCENE

# Cómo crear una Matriz $4 \times 4$ ?

$x$   
►  $y$



Matrix  $4 \times 4$   $x$

Matrix  $4 \times 4$   $y$

$y$   
►  $x$



Matrix  $4 \times 4$   $y$

Matrix  $4 \times 4$   $x$

# PoV y SCENE

## Por qué crear una Matriz 4x4?

```
public class Graph : MonoBehaviour {
    [SerializeField]
    Transform _pov = null;
    public Transform PoV { get { if (!_pov) _pov = transform; return _pov; } }

    public List<Vector3> vertex; ← en coordenadas
    public List<Edge> edges;           locales
    public Vector3ToWorldPoint (Vector3 point) {
        return PoV.TransformPoint(point);
    }

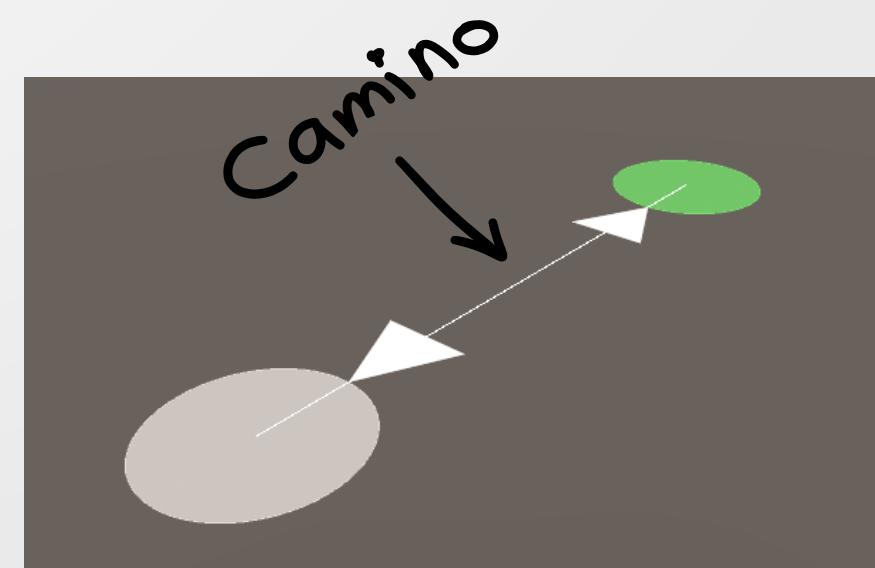
    public Vector3ToLocalPoint (Vector3 point) {
        return PoV.InverseTransformPoint(point);
    }
}
```

Dibuja Caminos entre vértices

```
public static void DrawEdges (Graph target) {
    Matrix4x4 oldMatrix = Handles.matrix;
    Handles.matrix = target.PoV.localToWorldMatrix;

    for (int i=0; i<target.edges.Count; i++) {
        foreach (int link in target.edges[i].links) {
            Handles.DrawLine(target.vertex[link], target.vertex[i]);
            CoolEditor.ArrowHead(target.vertex[link],
                target.vertex[link] - target.vertex[i],
                _buttonSize, _buttonSize);
        }
    }
    Handles.matrix = oldMatrix;
}
```

Handles.DrawLine(target.transform.position + target.vertex[link],  
target.transform.position + target.vertex[i]);



PoV y SCENE

# Por qué crear una Matriz 4x4?

```
public static void ArrowHead (Vector3 position, Vector3 direction,  
                           float size, float offset = 0) {  
    direction.Normalize();  
    Matrix4x4 old = Handles.matrix;  
    Handles.matrix = old *  
        Matrix4x4TRS(position, Quaternion.LookRotation(direction),  
                      Vector3.one);  
  
    Handles.DrawAAConvexPolygon(new Vector3[] {  
        new Vector3(0,0, -offset),  
        new Vector3(size/2f, 0, -size -offset),  
        new Vector3(-size/2f, 0, -size -offset)  
    });  
    Handles.matrix = old;  
}
```



PoV y SCENE

# Devuelve todo a su lugar

```
public static void ArrowHead (Vector3 position, Vector3 direction,
                           float size, float offset = 0) {
    direction.Normalize();

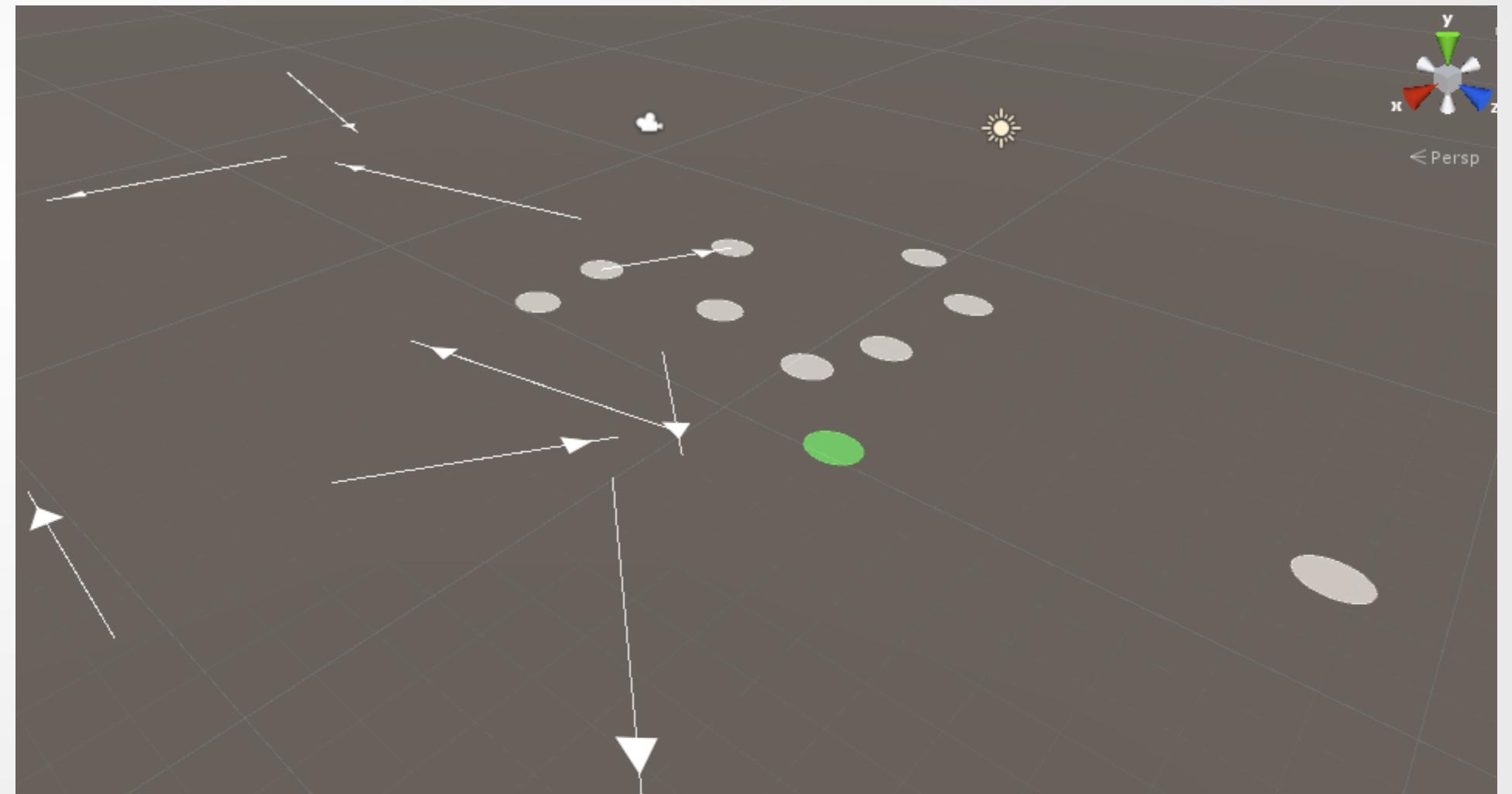
    Matrix4x4 newMatrix = Handles.matrix *
        Matrix4x4TRS(position, Quaternion.LookRotation(direction),
                      Vector3.one);

    using (new Handles.DrawingScope(Handles.color, newMatrix)) {
        Handles.DrawAACConvexPolygon(new Vector3[] {
            new Vector3(0,0, -offset),
            new Vector3(size/2f, 0, -size -offset),
            new Vector3(-size/2f, 0, -size -offset)
        });
    }
}
```

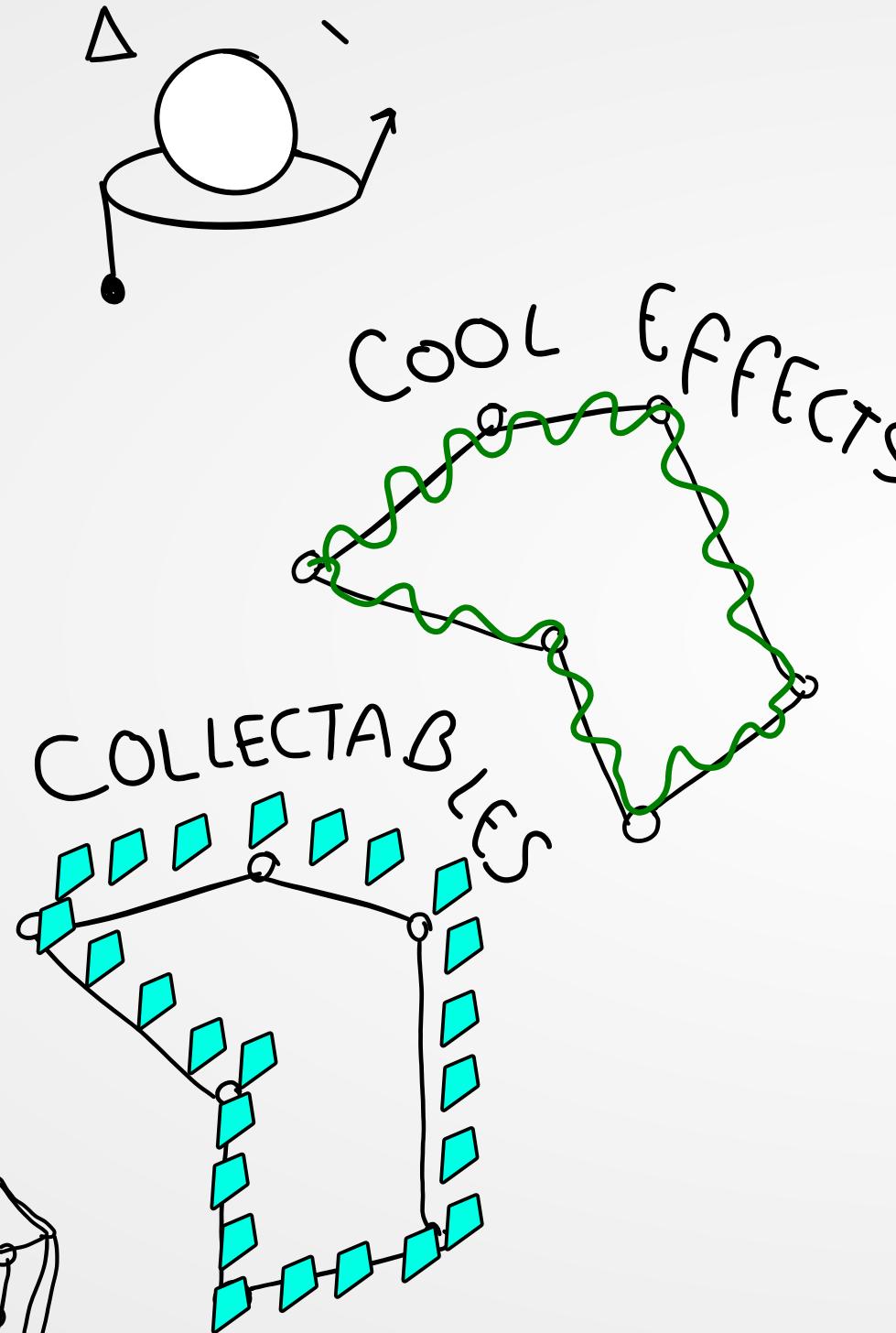
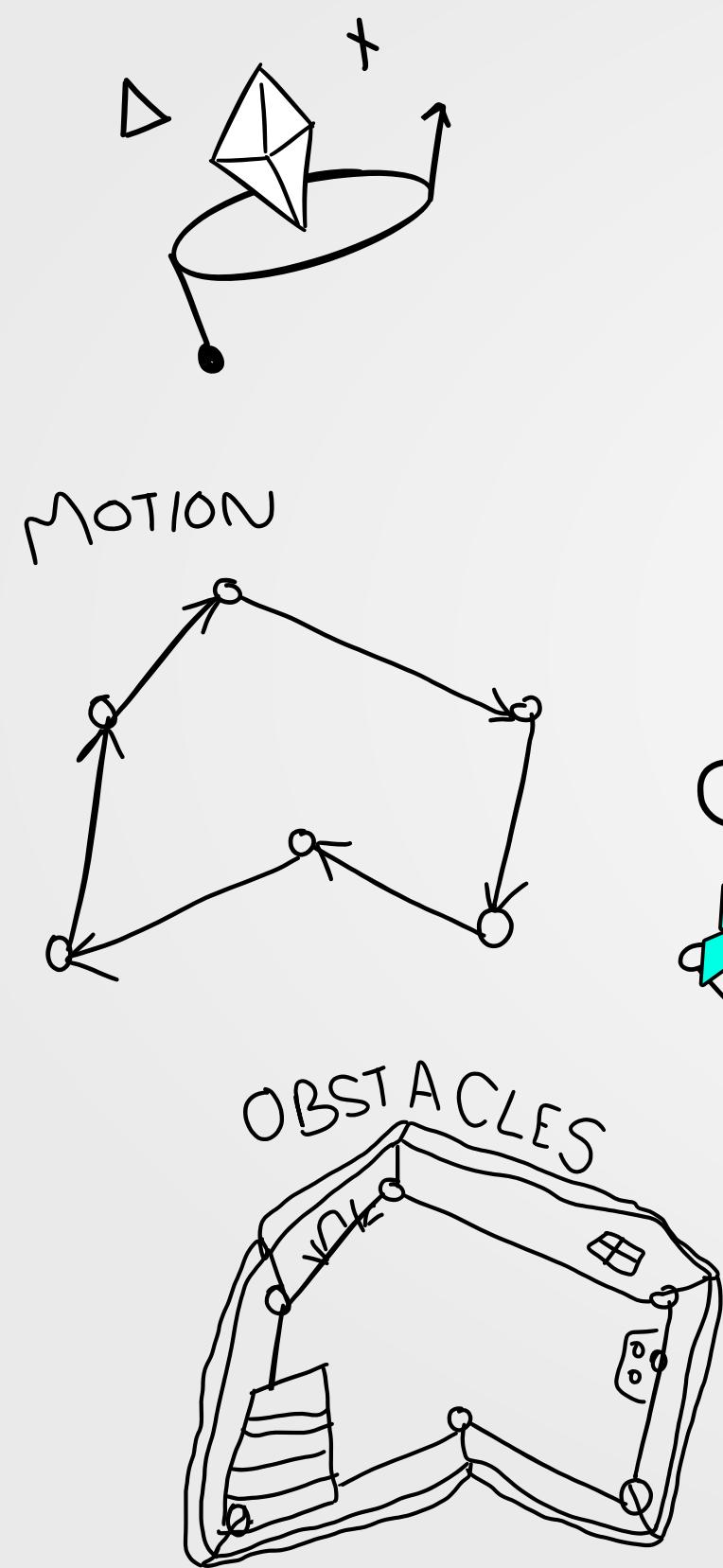
```
public static void ArrowHead (Vector3 position, Vector3 direction,
                           float size, float offset = 0) {
    direction.Normalize();
    Matrix4x4 old = Handles.matrix;
    Handles.matrix = old *
        Matrix4x4TRS(position, Quaternion.LookRotation(direction),
                      Vector3.one);

    Handles.DrawAACConvexPolygon(new Vector3[] {
        new Vector3(0,0, -offset),
        new Vector3(size/2f, 0, -size -offset),
        new Vector3(-size/2f, 0, -size -offset)
    });

    // Handles.matrix = old;
}
```

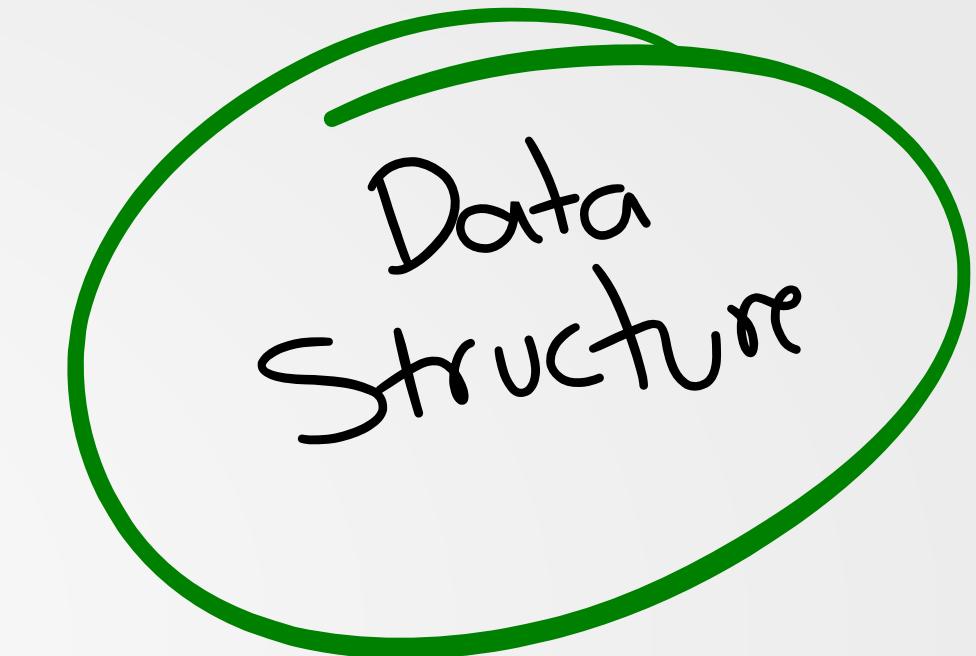


# REUSABILIDAD

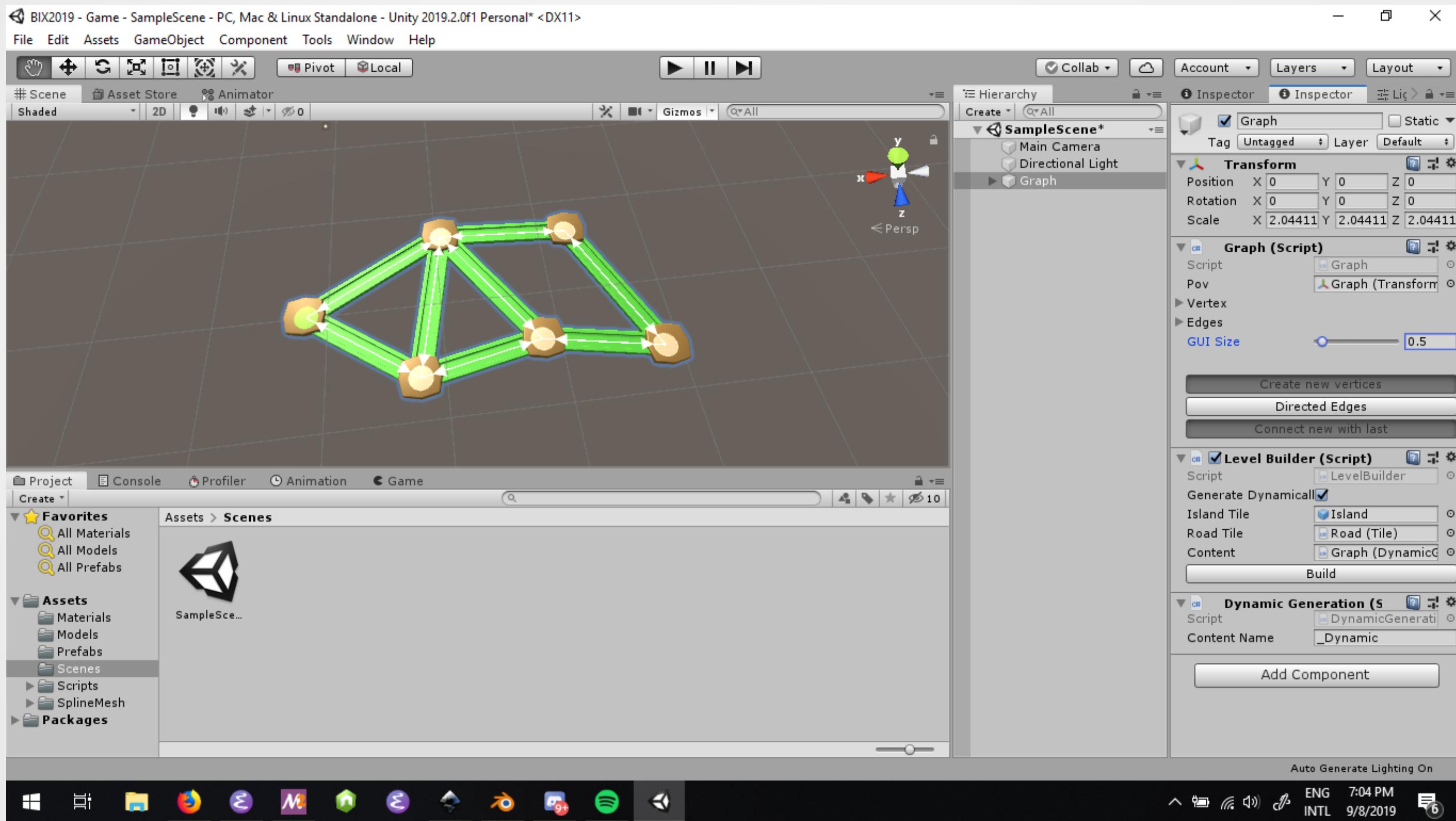


Motion : MonoBehaviour

Graph : MonoBehaviour



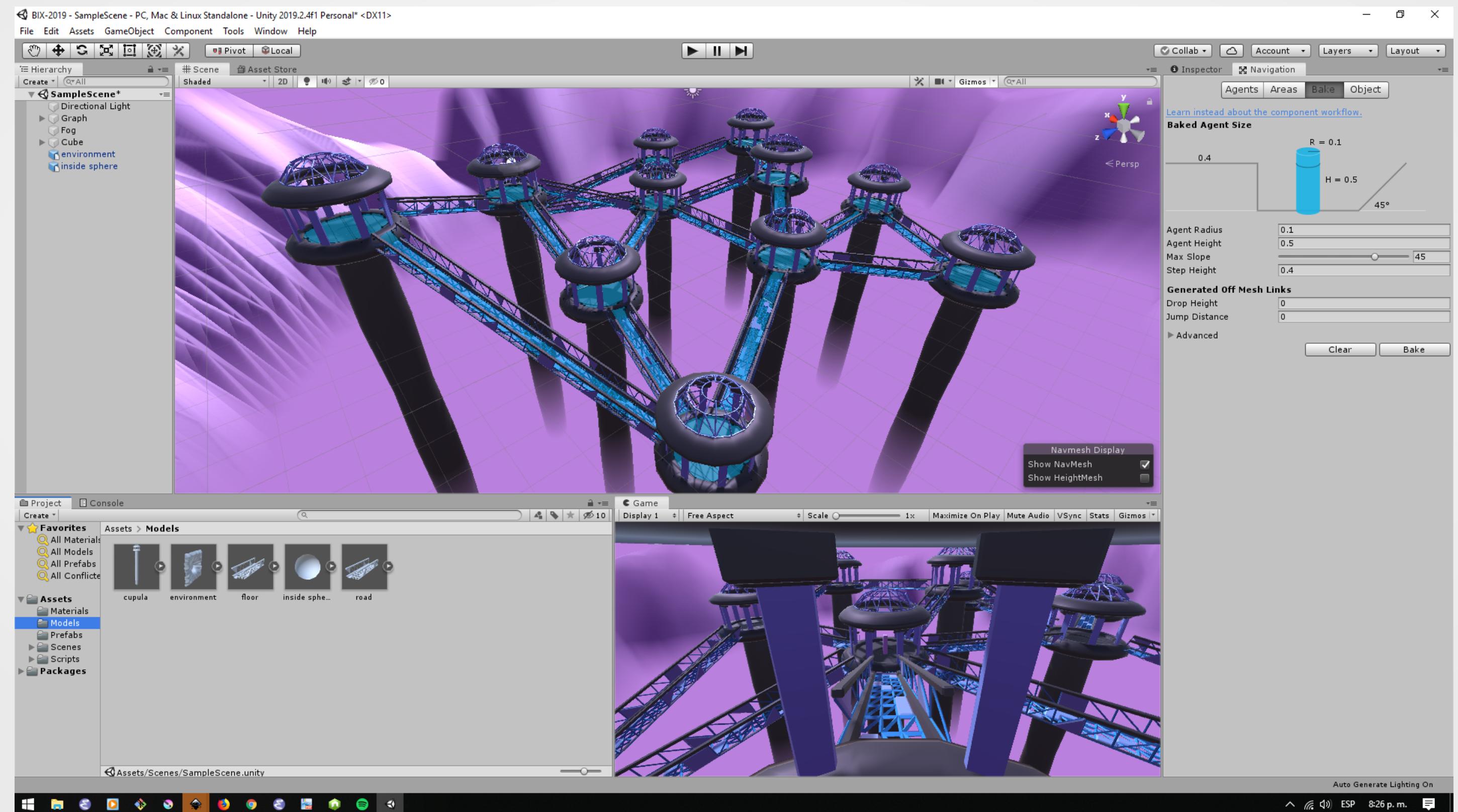
# REUSABILIDAD



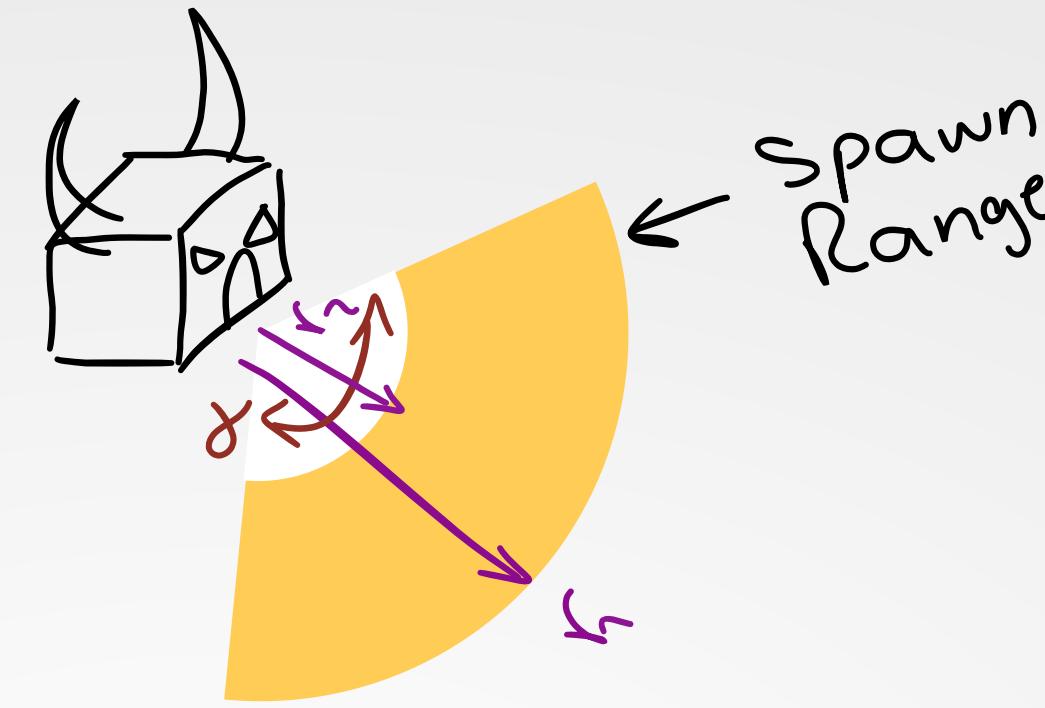
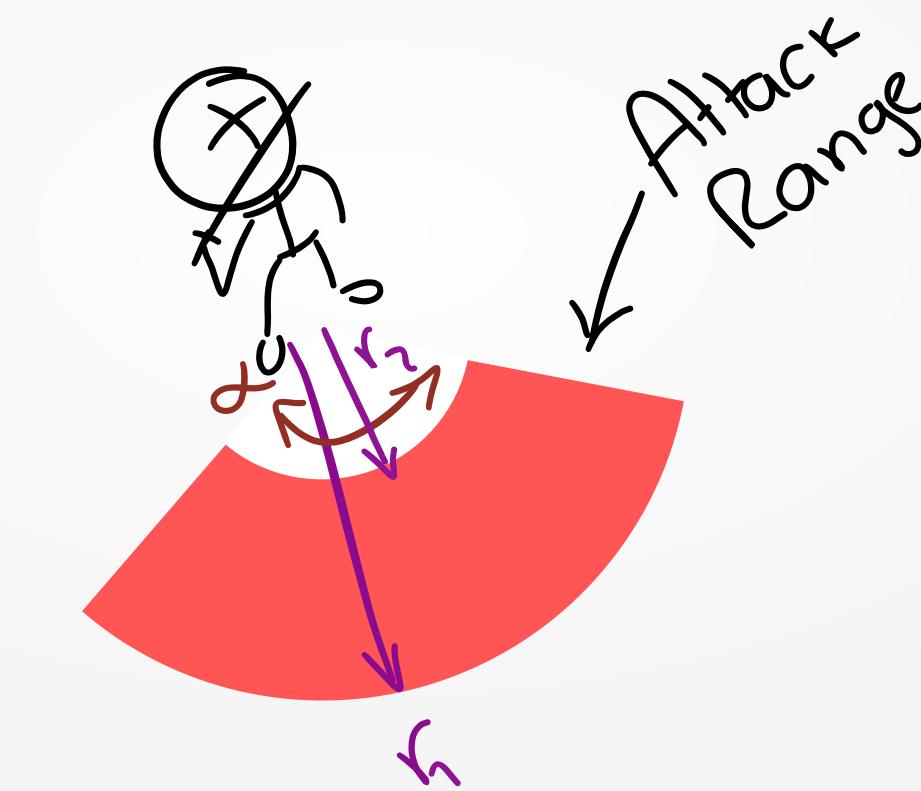
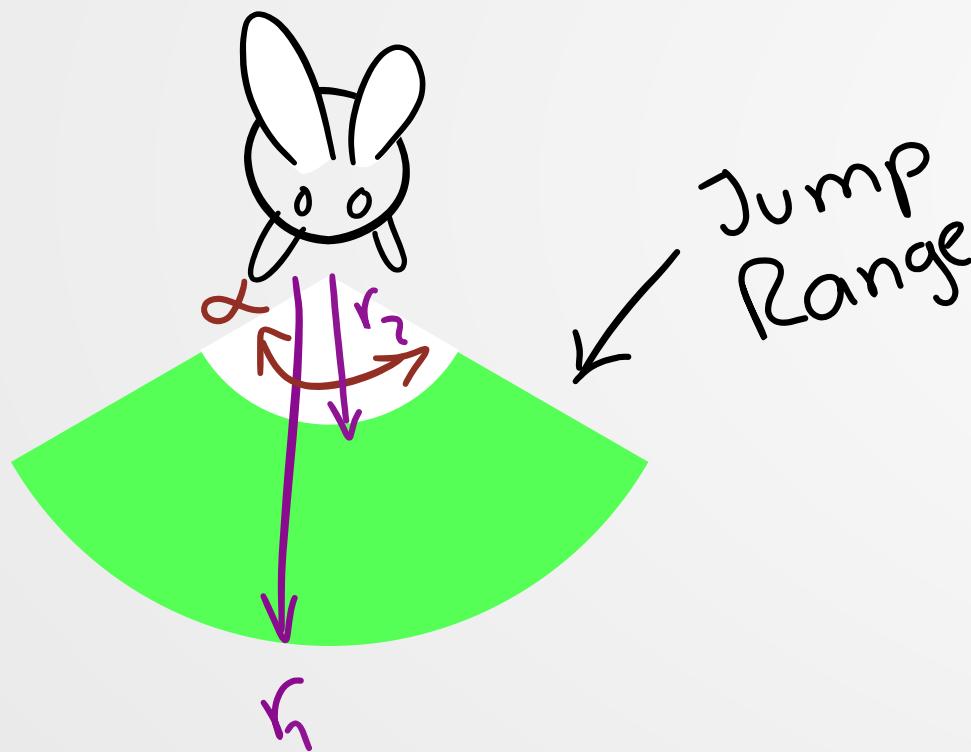
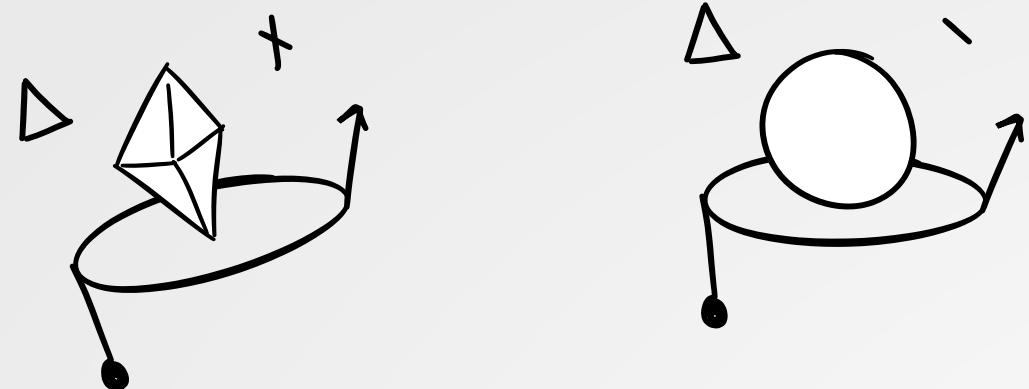
Construye cosas con esa información

info sobre el grafo

# REUSABILIDAD

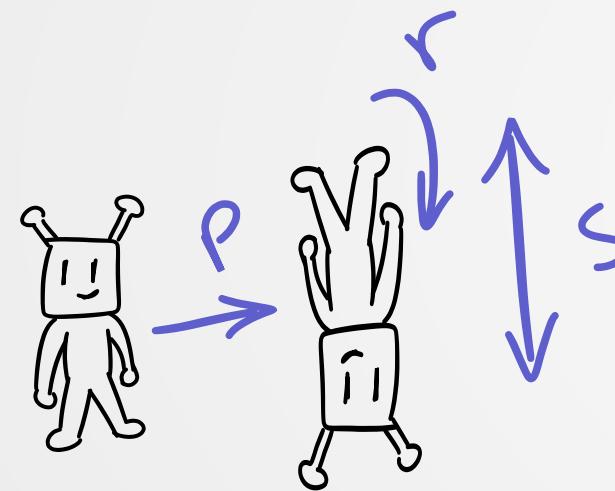


# REUSABILIDAD

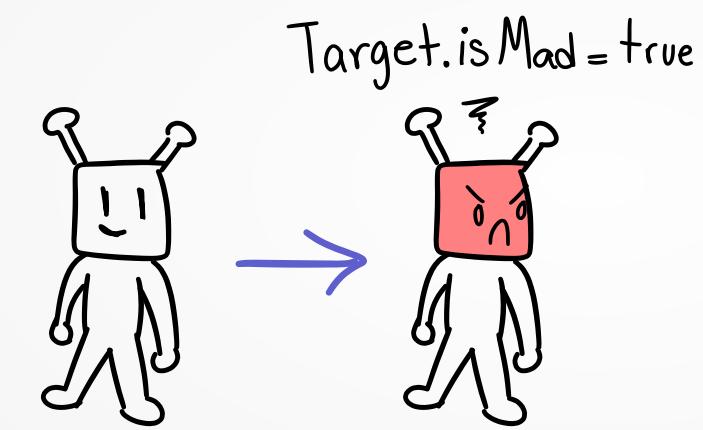


# PERSISTENCIA

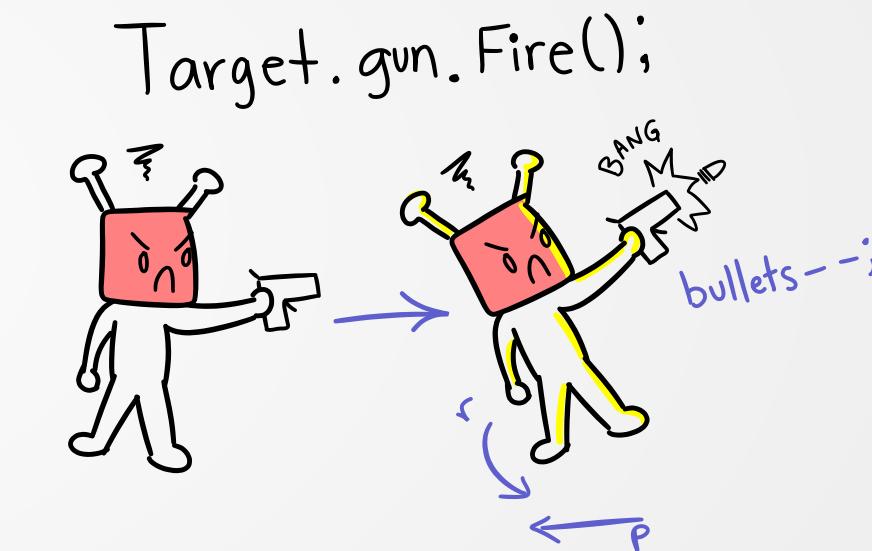
Undo. Record Object (\*, message);



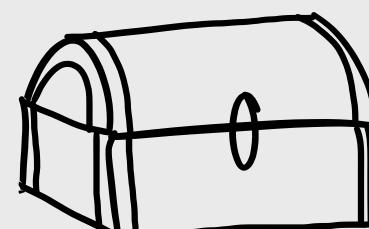
transform



Target



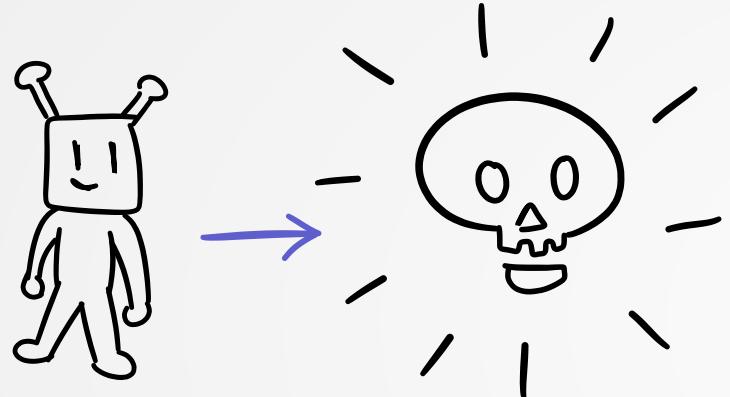
Target.gun; transform ;



# PERSISTENCIA

No!!

`Destroy(Target.gameObject);` X



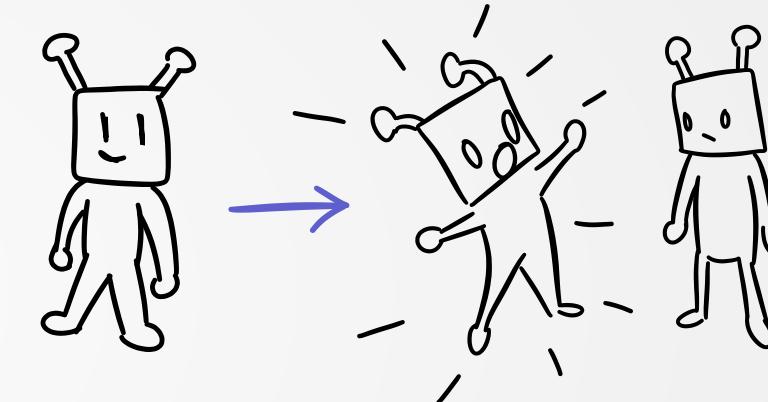
`Undo.DestroyObjectImmediate`

`Util.Destroy(x)`

`UnityEngine.Object`

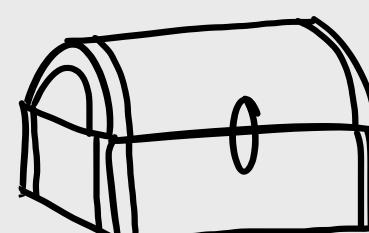
Platform  
dependant  
Compilation

`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;

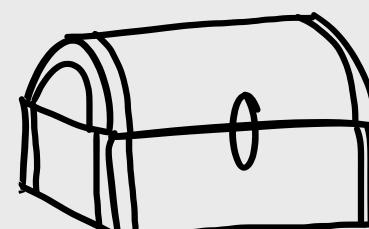
    public int agility;
    public int perception;
    public int attackPower;
    public float attackRadius;
}
```

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

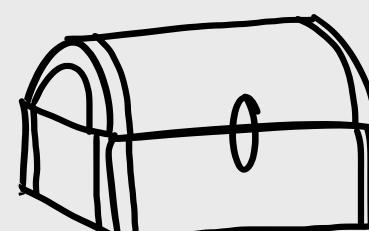
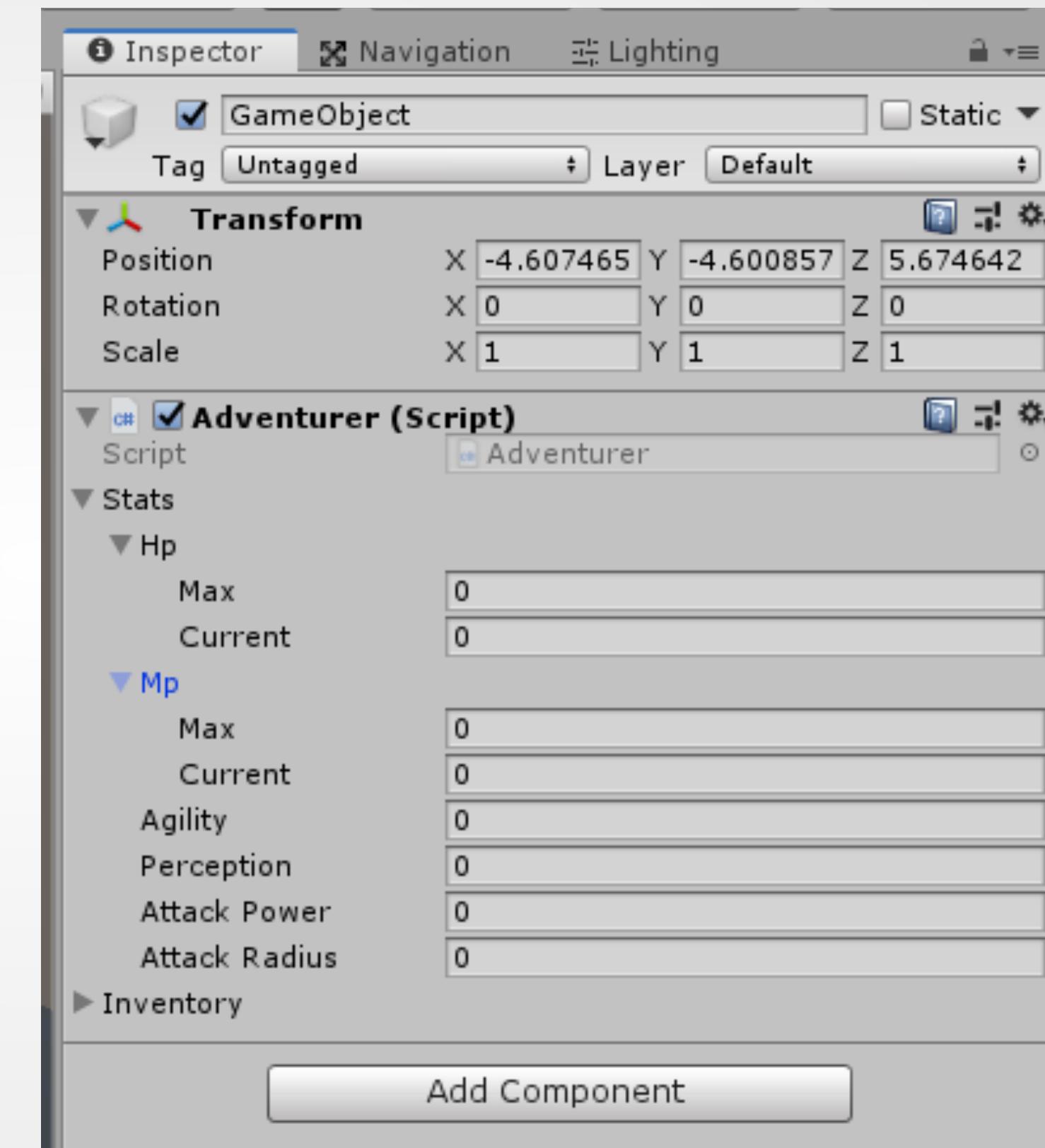
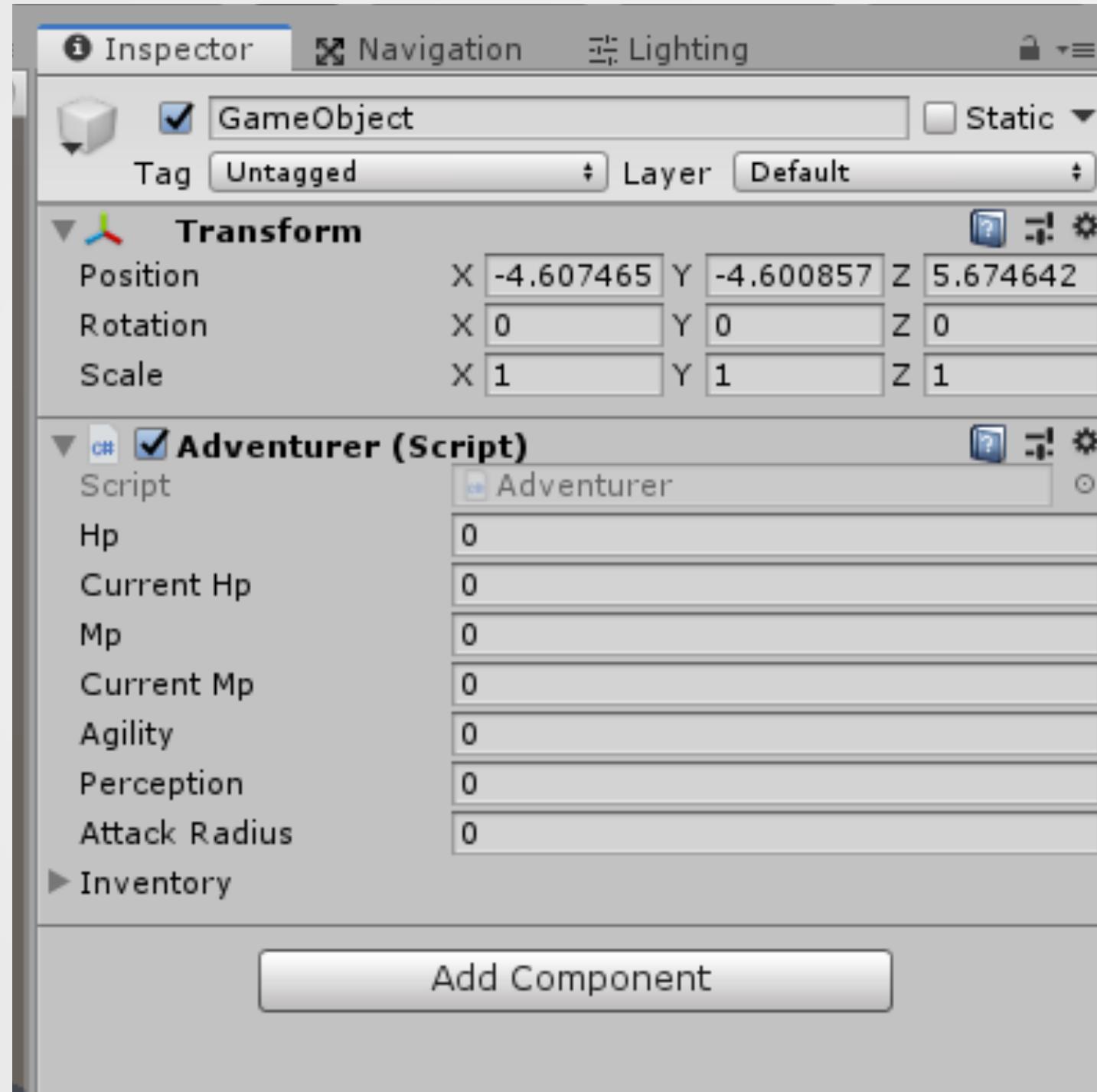
public class Adventurer : MonoBehaviour {
    public Stats stats;
    public List<GameObject> inventory;
}

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

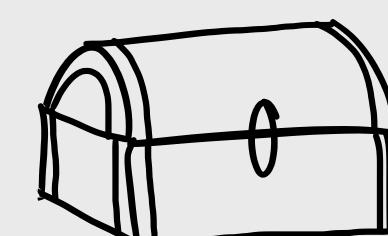
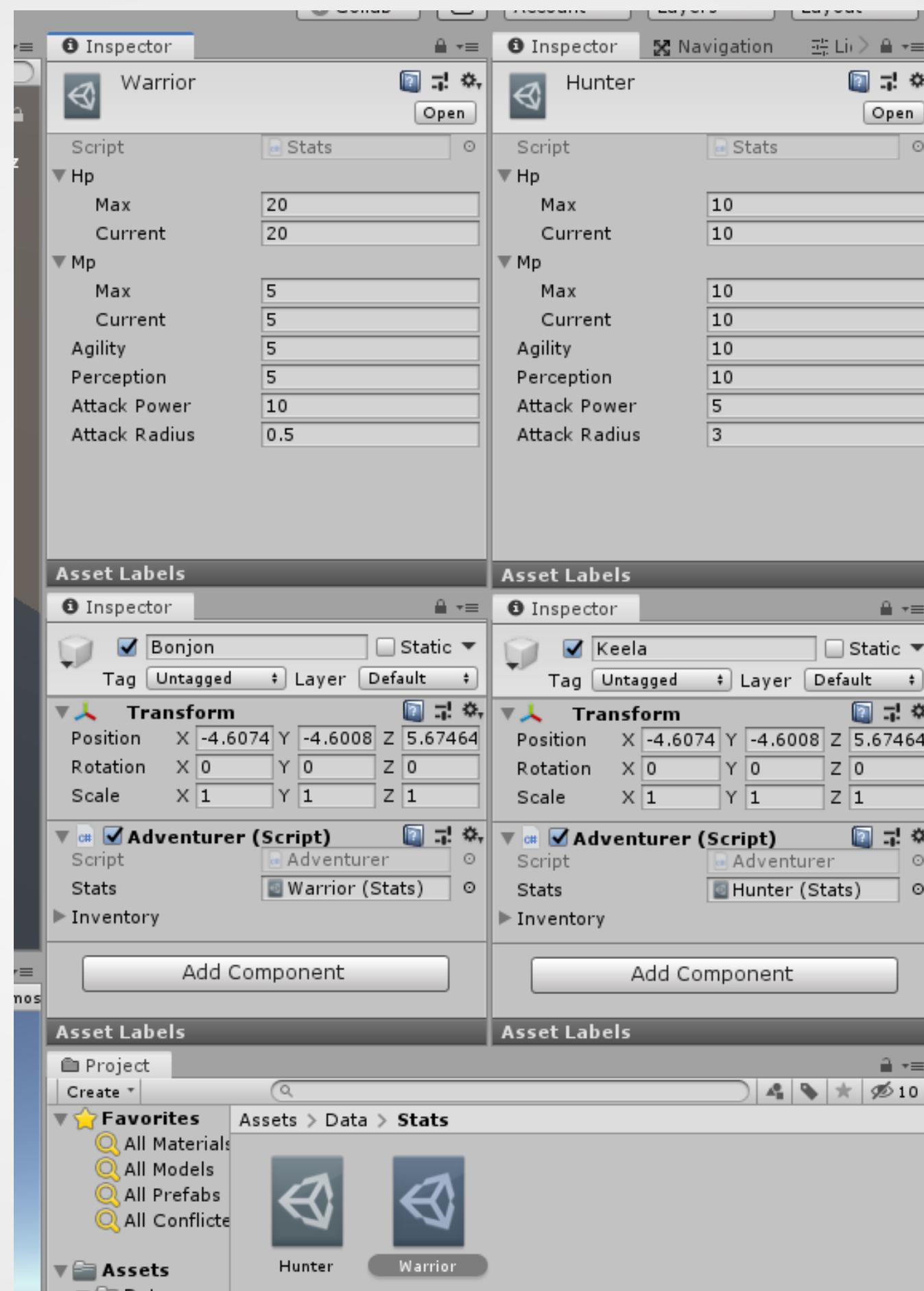
[System.Serializable]
public class RenewableStat {
    public int max = 10;
    public int current = 0;
}
```



# 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;

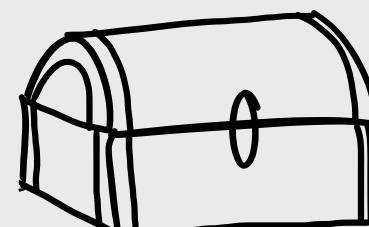
    public int agility;
    public int perception;
    public int attackPower;
    public float attackRadius;
}
```
- Adventurer.cs** (Git:master C#/1 +2 yas):

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

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

    // ...
}
```

The Unity Editor background features a dark theme with a green and yellow geometric pattern. Various application icons are visible in the dock at the bottom.



# 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

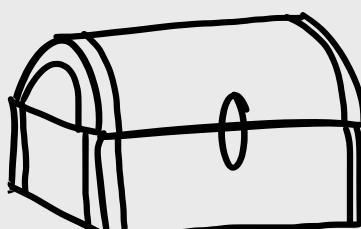
dictionary X

interfaces X

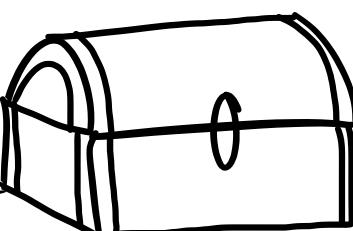
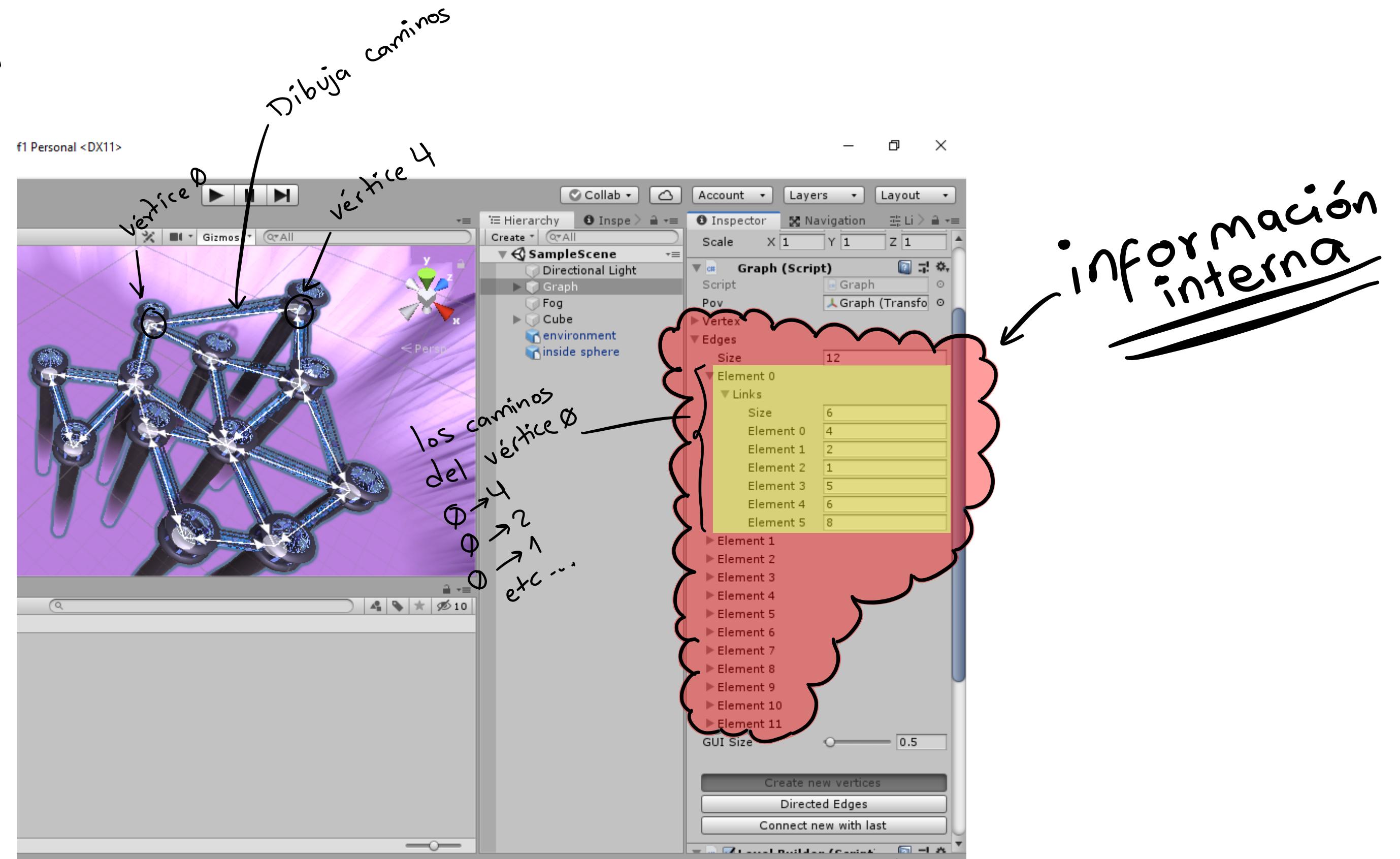
Esconde cosas visibles  
en el inspector

[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;

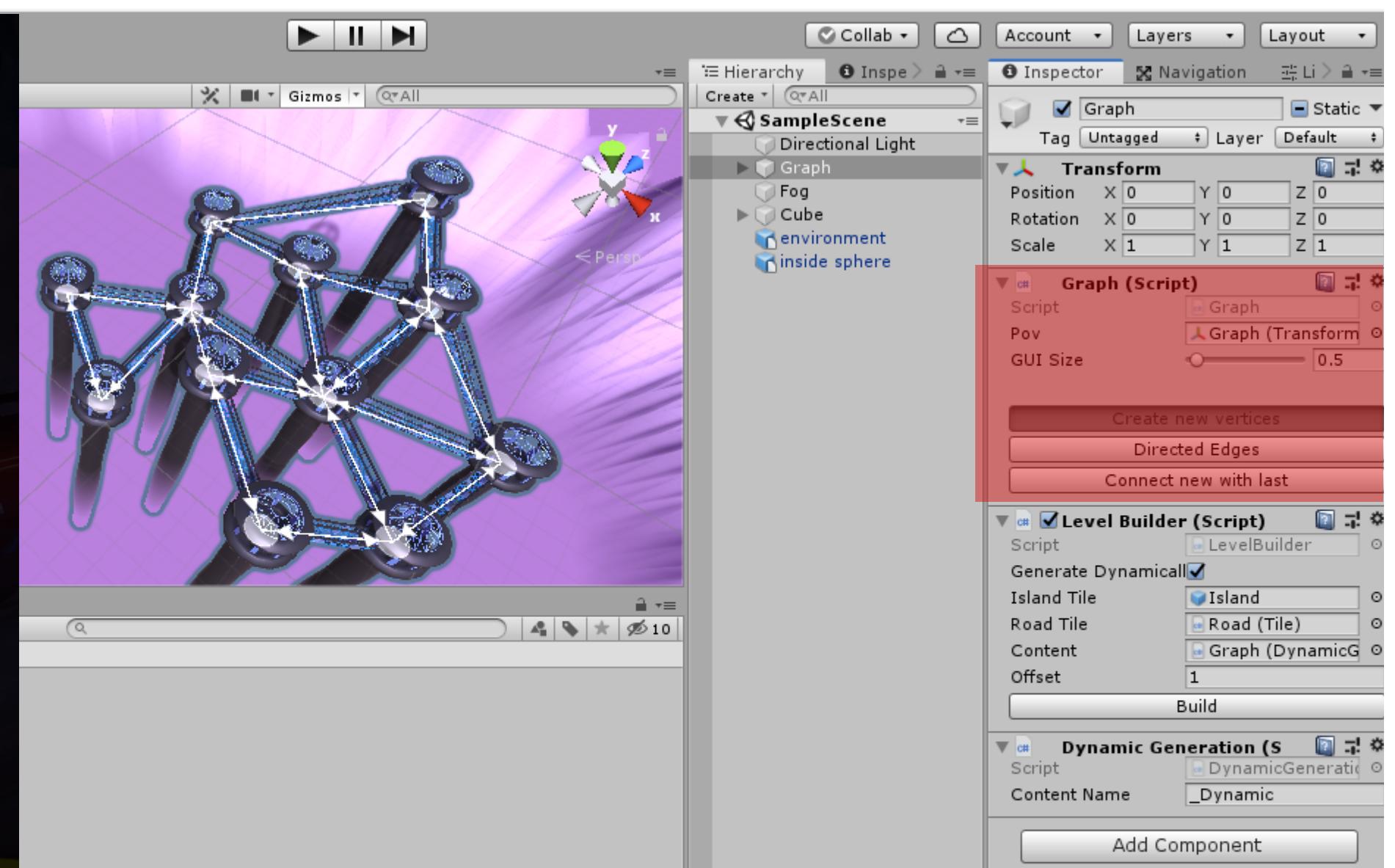
    [SerializeField]
    Transform _pov = null;

    [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;

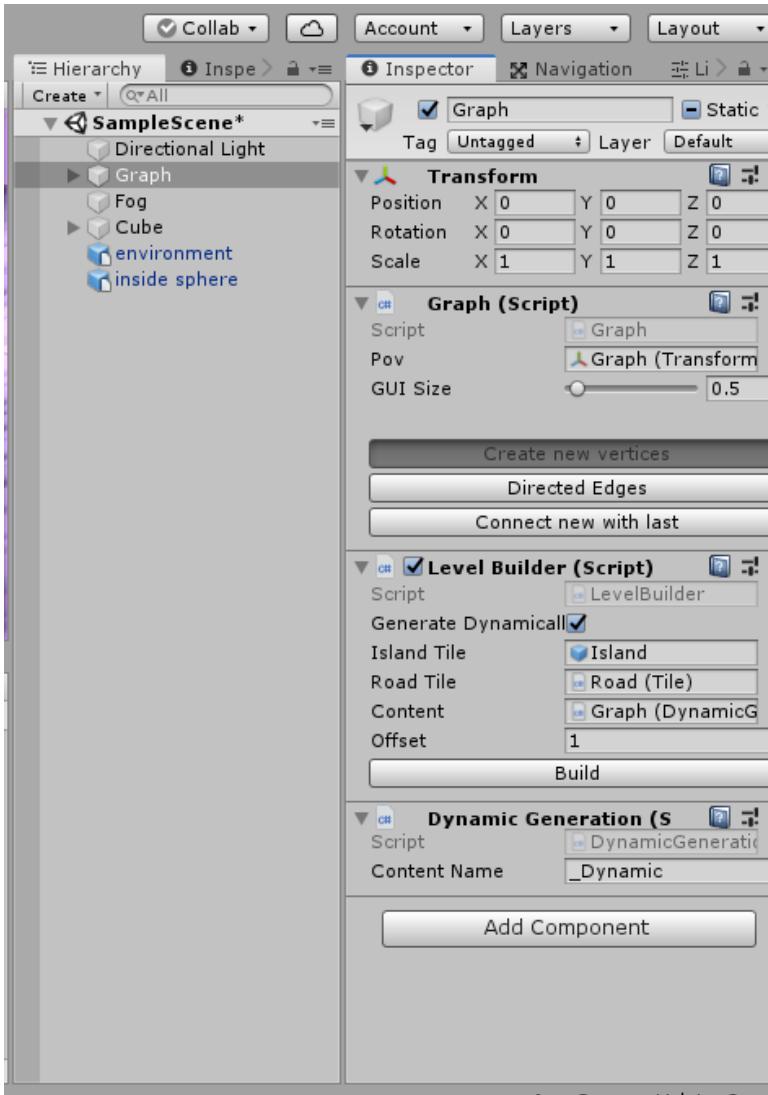
    [SerializeField]
    Tile _roadTile;

    [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;

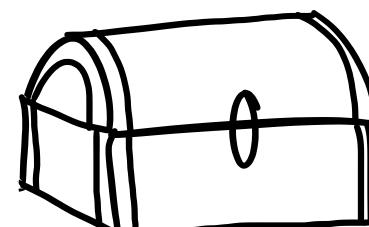
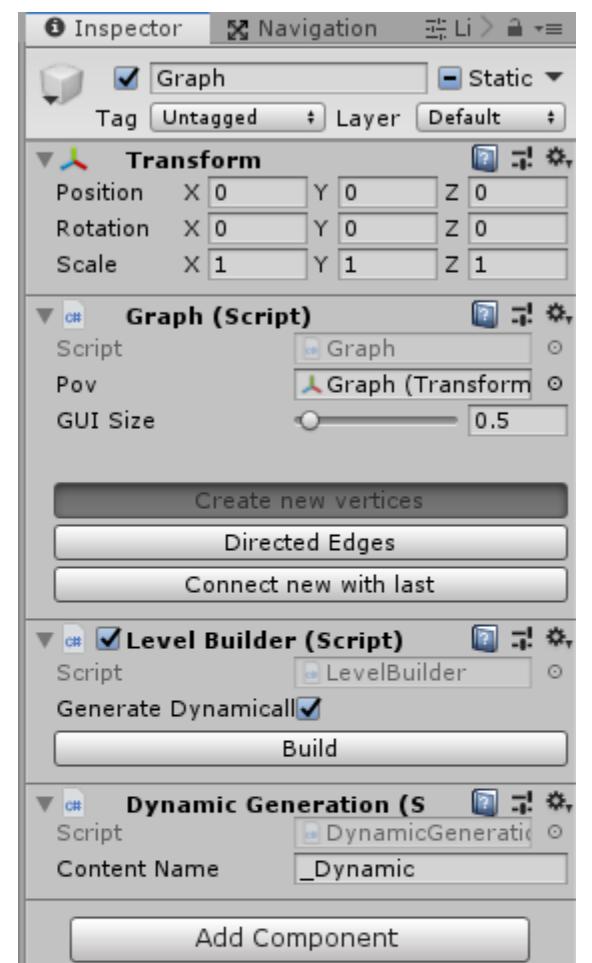
    [HideInInspector]
    [SerializeField]
    Tile _roadTile;

    [HideInInspector]
    [SerializeField]
    DynamicGeneration _content;

    [HideInInspector]
    [SerializeField]
    float _offset = 1;

    void OnEnable () {
        _Graph.onGraphModified += GraphModifiedHandler;
    }

    public void GraphModifiedHandler () {
        if (generateDynamically) {
            Build();
        }
    }
}
```



# COMUNICACIÓN



UnityEditor



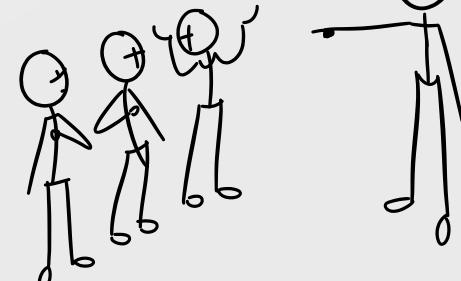
Build



GameObject

Cosas que  
el .exe necesita

estado del  
Componente



# COMUNICACIÓN

On Inspector GUI

Variables en la misma clase

On Scene GUI

```
[CustomEditor(typeof(Graph))]
public class GraphEditor : Editor {
    Graph Target { get => (Graph) target; }
    static float _buttonSize = 0.5f;
    public static int selected = 0;

    public static bool directedEdges = false;
    public static bool continuousConnection = false;
    public static bool createNewVertices = true;
```

Set On Inspector GUI

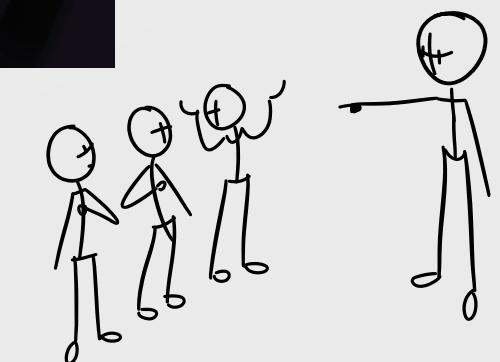
```
createNewVertices =
    GUILayout.Toggle(createNewVertices,
                   "Create new vertices", "Button");
directedEdges =
    GUILayout.Toggle(directedEdges,
                   "Directed Edges", "Button");
continuousConnection =
    GUILayout.Toggle(continuousConnection,
                   "Connect new with last", "Button");
```

Read On Scene GUI

```
if (continuousConnection && selected >= 0) {
    target.Connect(selected, target.vertex.Count - 1,
                    directedEdges);
}

if (target.IsConnected(selected, clicked)) {
    Undo.RecordObject(target, "disconnected two vertices");
    target.Disconnect(selected, clicked, directedEdges);
}
else if (selected < 0)
```

```
if (createNewVertices) {
    UpdateVertexCreation(target);
    CoolEditor.HideTool();
} else if (selected < 0)
```



# COMUNICACIÓN



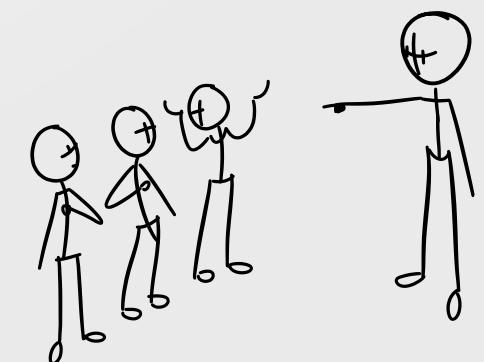
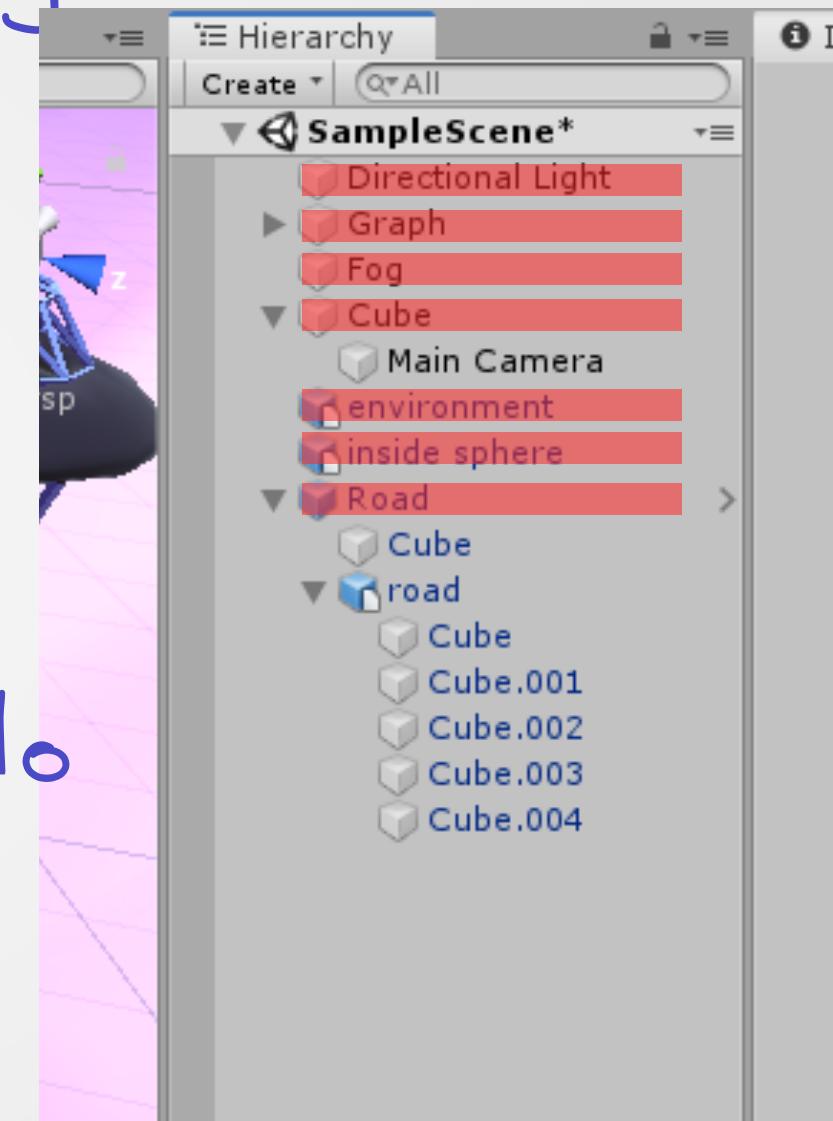
Target.gameObject.

GetRootGameObjects

OnEnable()

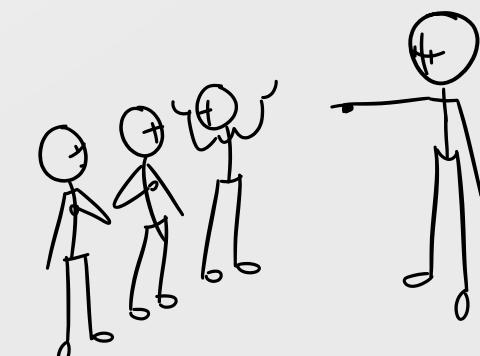
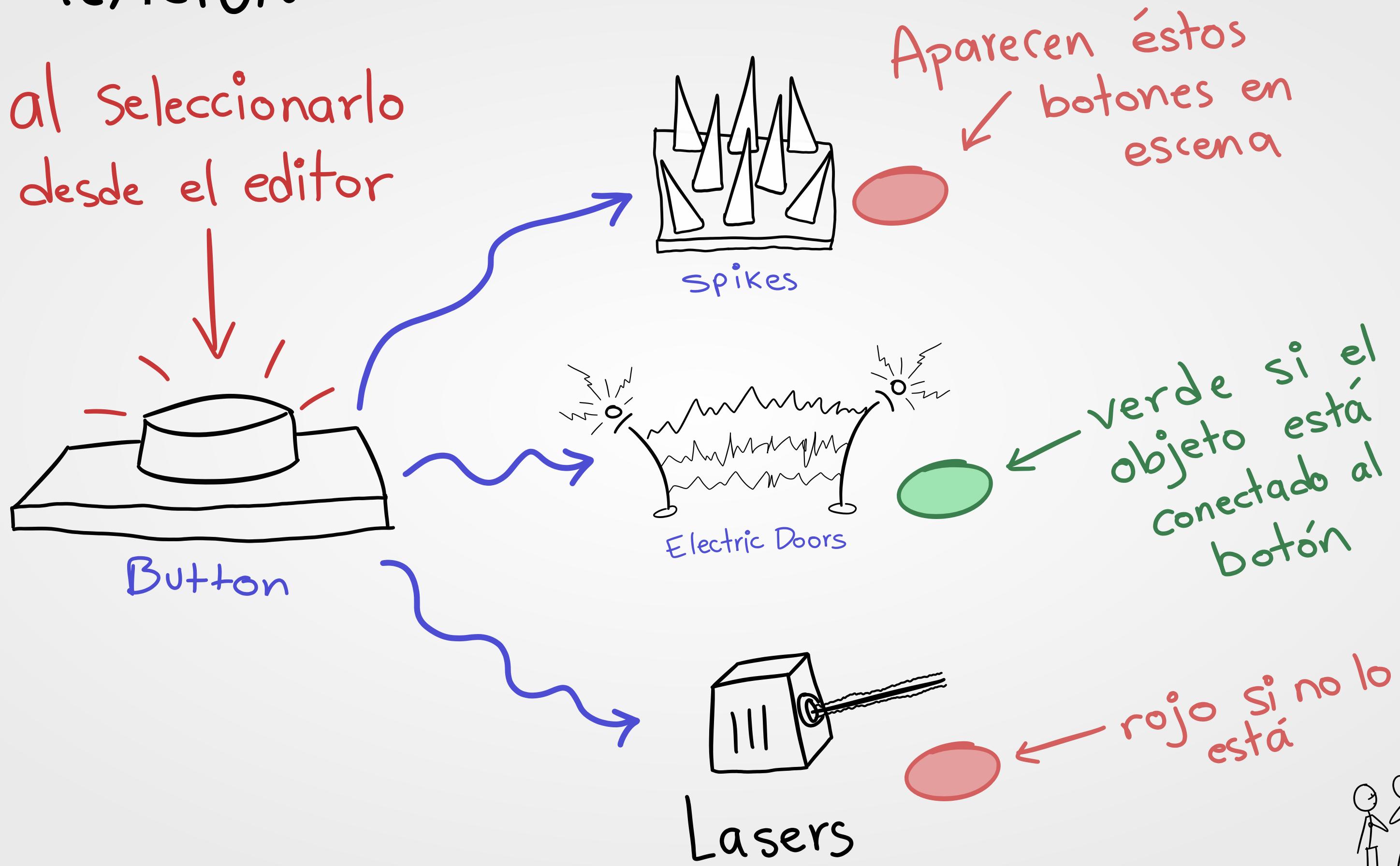
Cuando un objeto  
acaba de ser seleccionado

Game Object[]

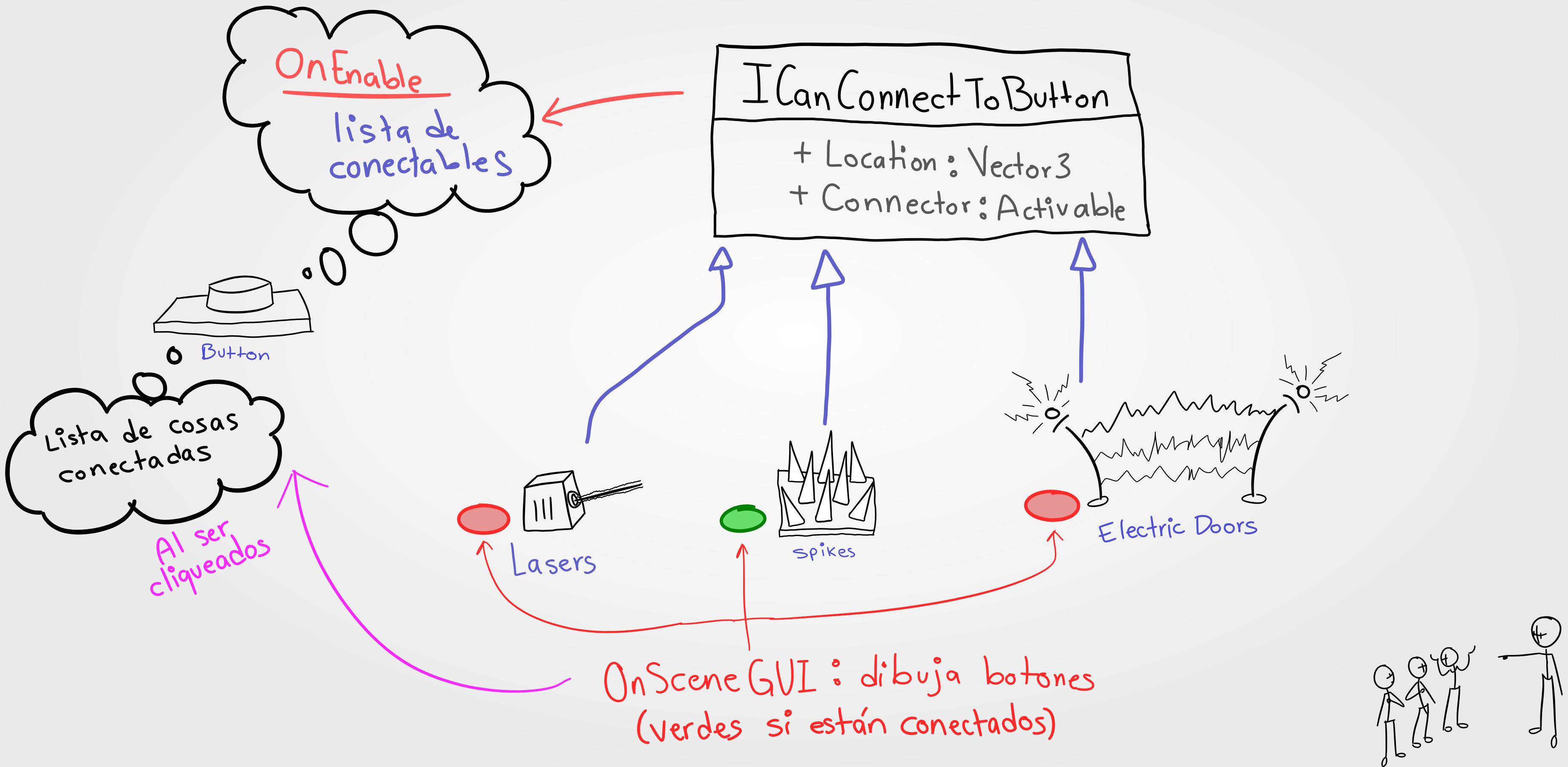


# COMUNICACIÓN

al seleccionarlo  
desde el editor



# COMUNICACIÓN



# Gracias!!

- ▶ Ruth García
- ▶ fb.com/pr00thmatic
- ▶ github.com/pr00thmatic
- ▶ youtube.com/pr00thmatic