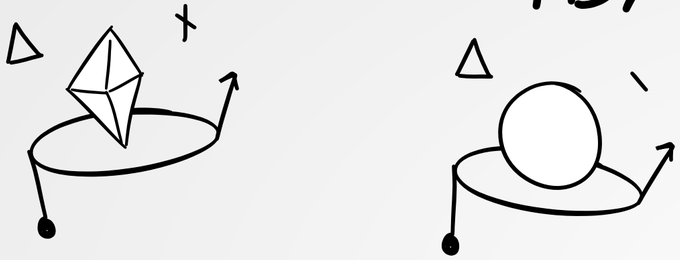
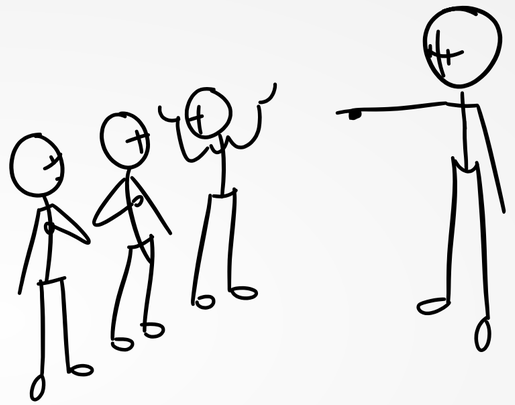
Cómo implementar niveles

(sin morir en el intento)

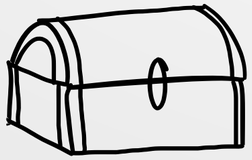
REUSABILIDAD



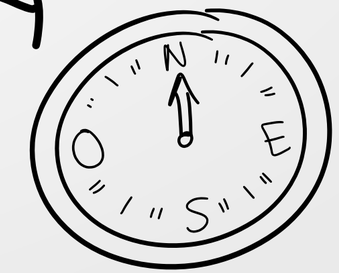
COMUNICACIÓN



PERSISTENCIA

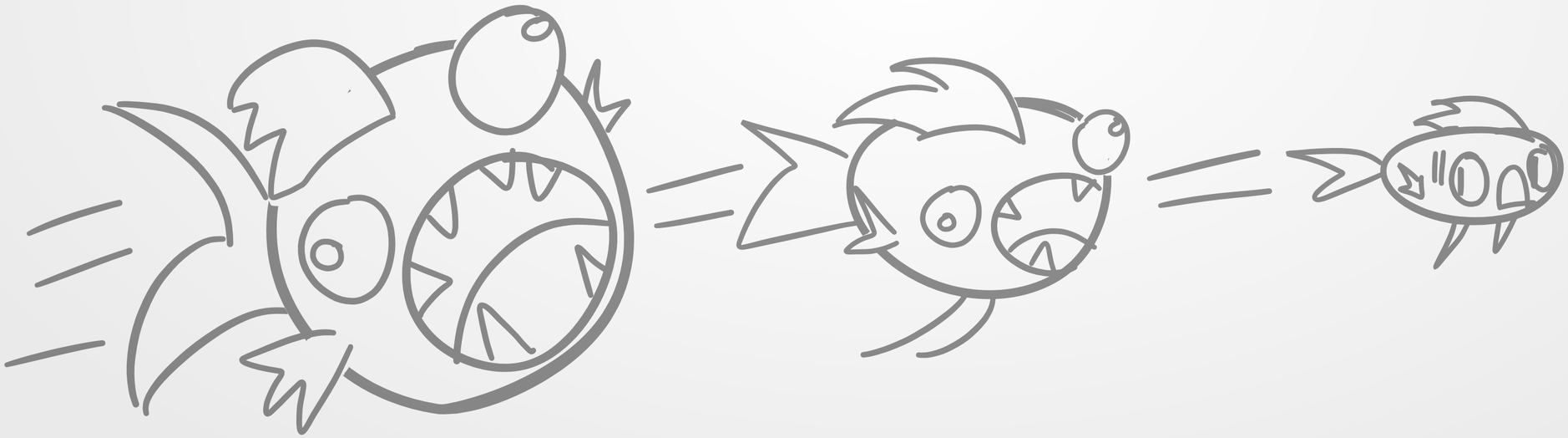


PoV y SCENE



Editor Scripting

Editor → MonoBehaviour → GameObject



Editor Scripting

Build!

Editor → MonoBehaviour → GameObject

Editor Scripting

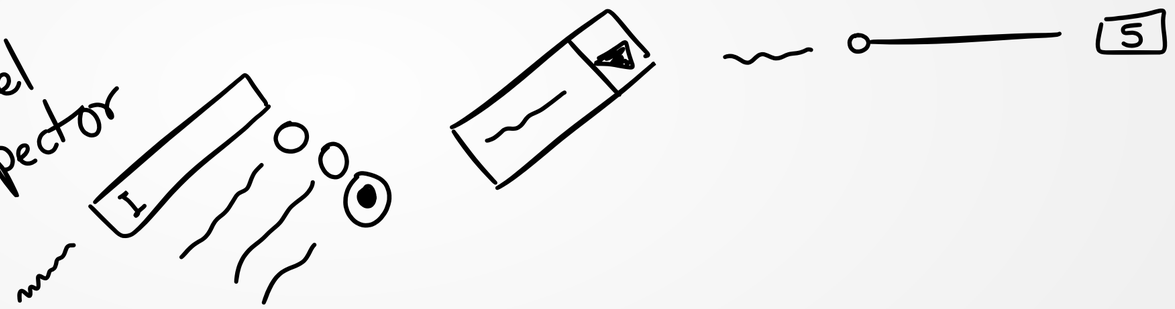
MonoBehaviour

```
[CustomEditor(typeof(MiScript))]
public class MiEditor : Editor {
```

está en UnityEditor namespace

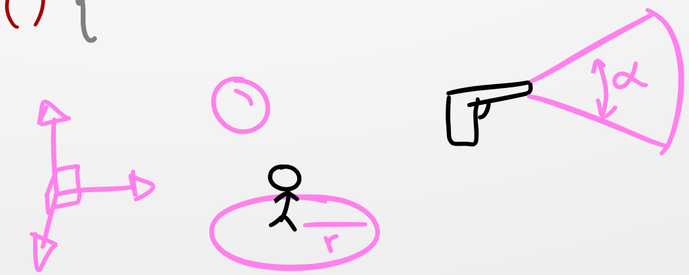
```
    override void OnInspectorGUI() {
```

Cosas del inspector




```
    }
    void OnSceneGUI() {
```

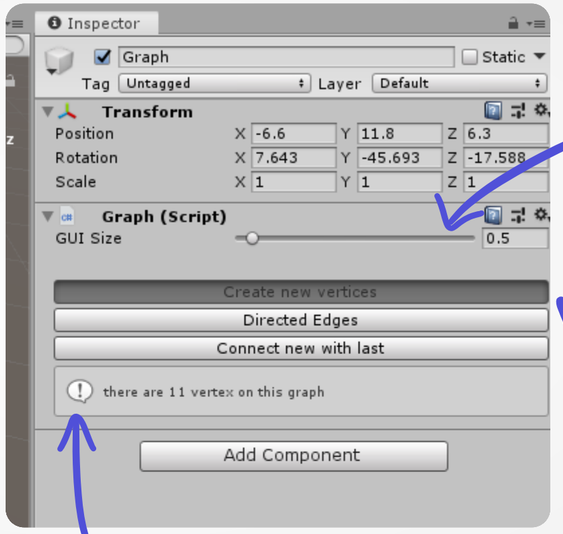
Cosas de la escena




```
    }
}
```

Editor Scripting

OnInspectorGUI()



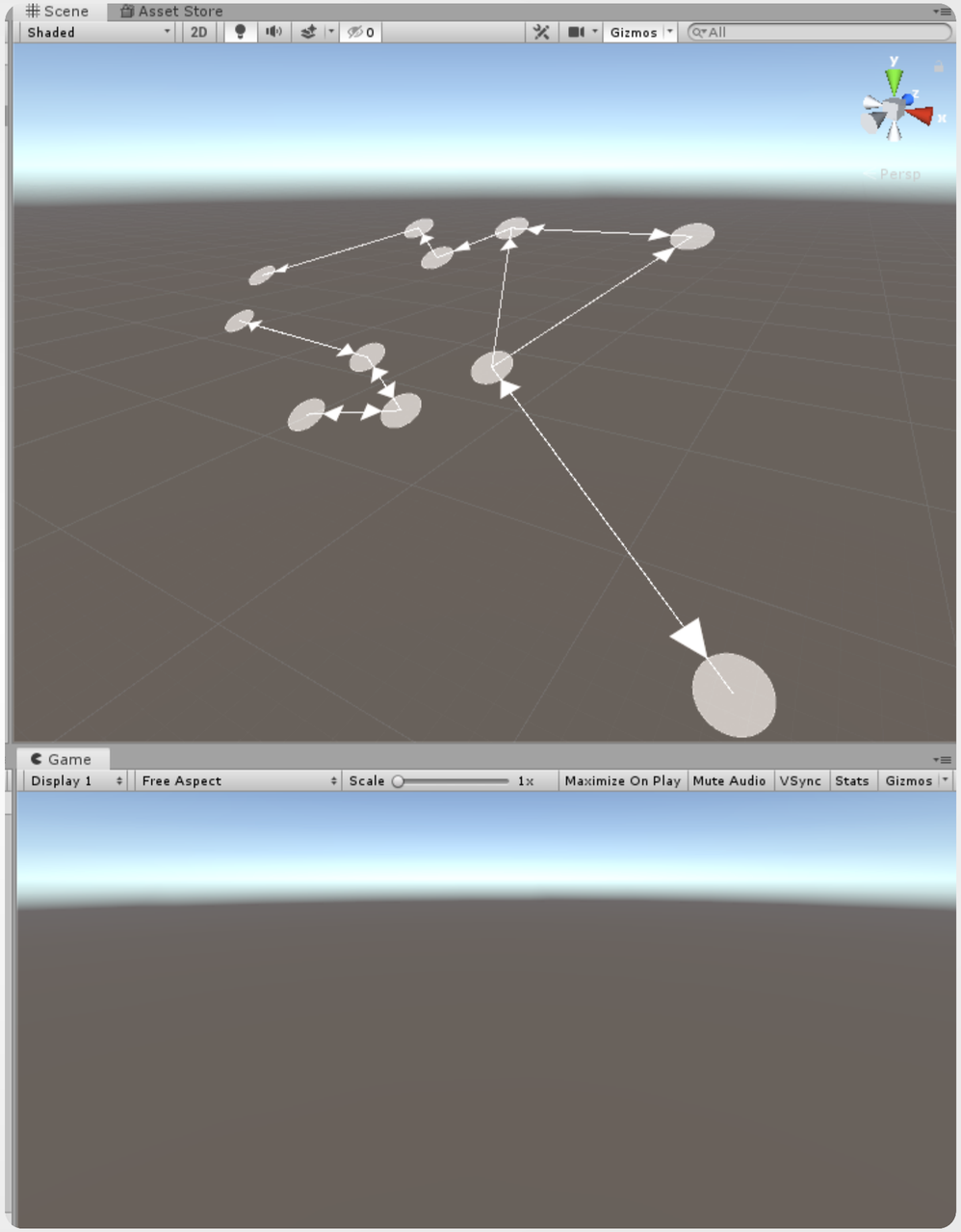
very SLIDER

such BUTTON

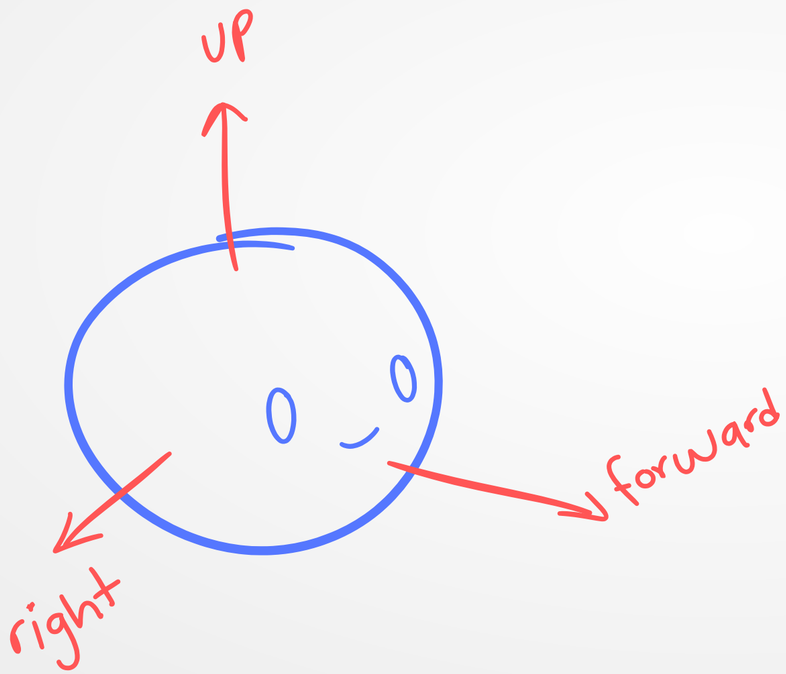
many DEBUGGY

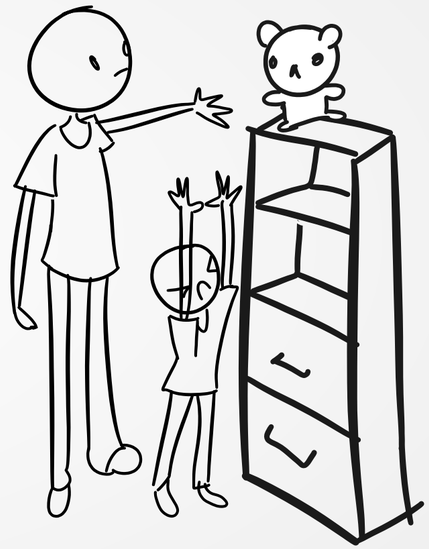
Editor Scripting

OnSceneGUI ()

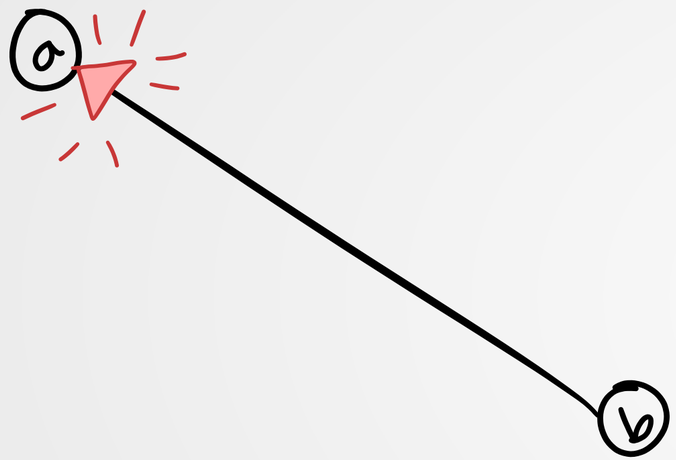


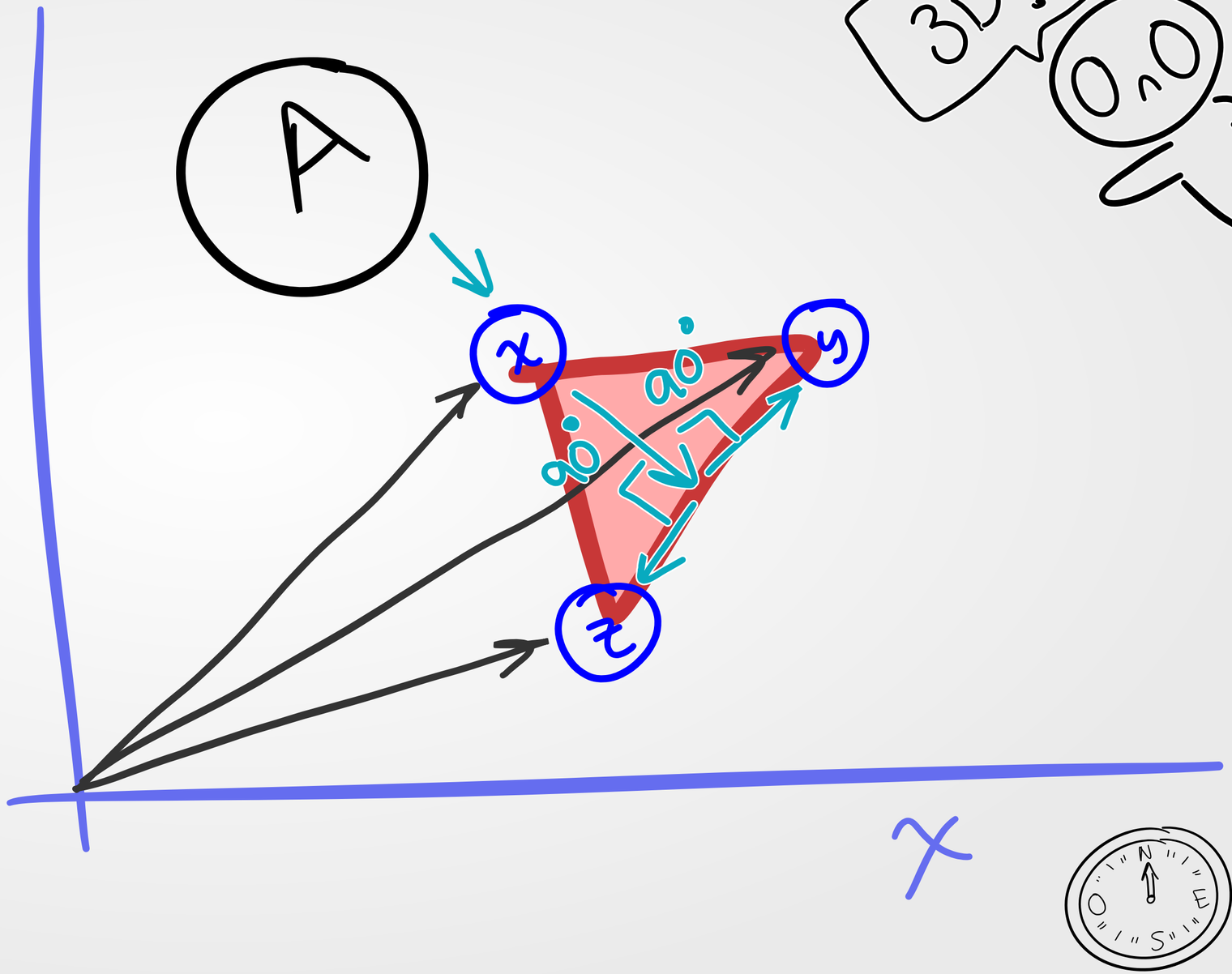
PoV y SCENE





PoV y SCENE





PoV y SCENE

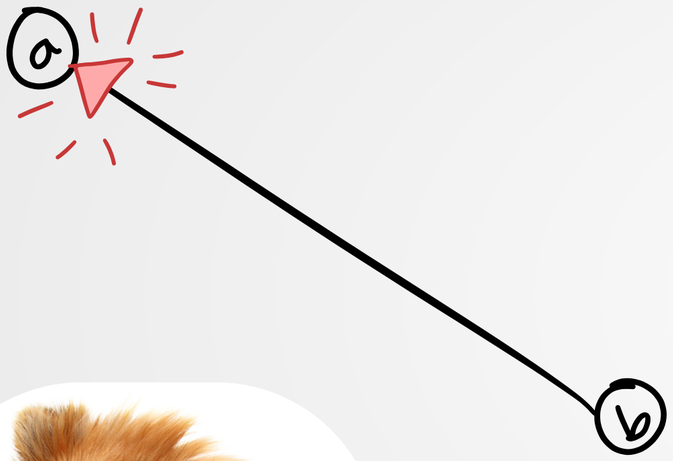
Handles.Matrix

$$\begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ r_1 & r_2 & r_3 & r_4 \\ s_1 & s_2 & s_3 & s_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

transformation matrix

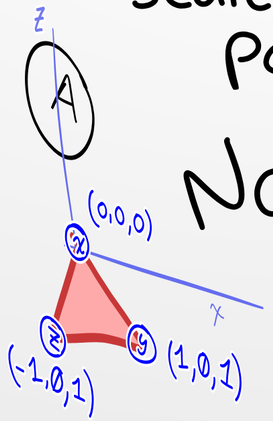
Matrix4x4

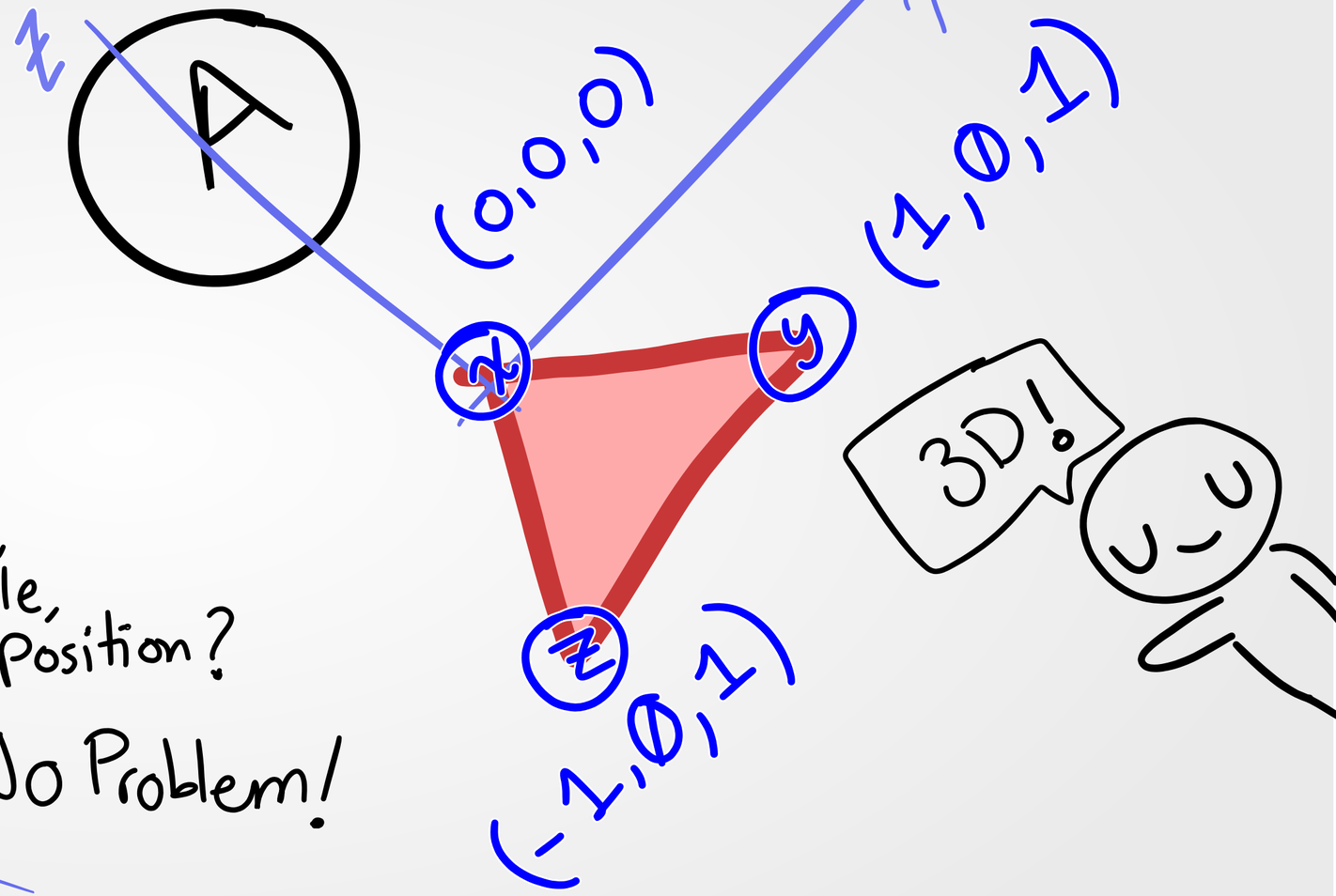
PoV y SCENE



rotation,
Scale,
Position?

No Problem!

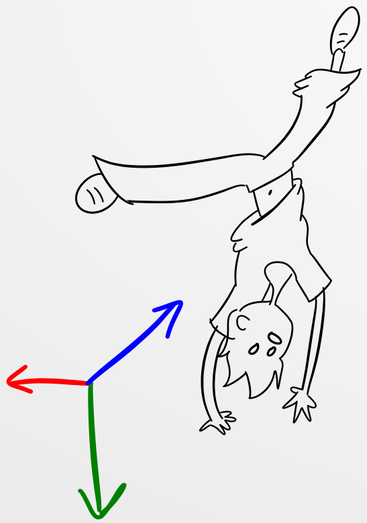




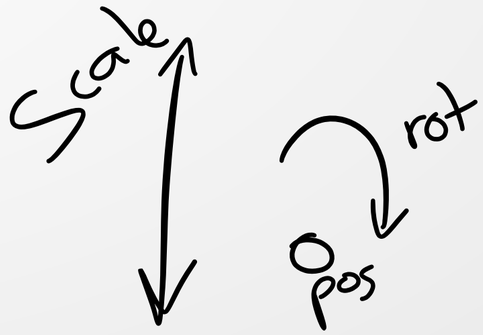
P.V y SCENE

Cómo crear una Matriz4x4?

`transform.localToWorldMatrix`



`Matrix4x4.TRS(pos, rot, scale)`



P.V y SCENE

Cómo crear una Matriz 4x4?

x
y

x * y

Matrix 4x4 x Matrix 4x4 y

y
x

y * x

Matrix 4x4 y Matrix 4x4 x

PoV y SCENE

Por qué crear una Matriz4x4?

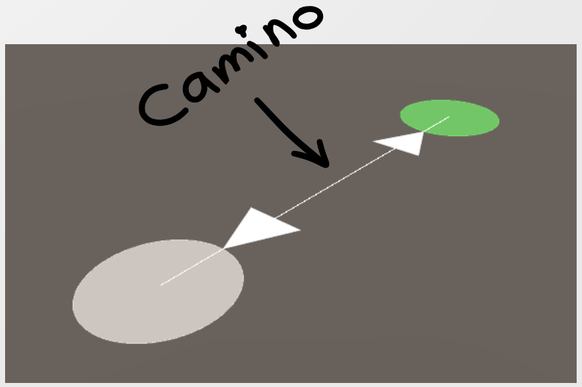
```
public class Graph : MonoBehaviour {
    [SerializeField]
    Transform _pov = null;
    public Transform PoV { get { if (!_pov) _pov = transform; return _pov; } }

    public List<Vector3> vertex;
    public List<Edge> edges;

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

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

en coordenadas locales

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;
}
```

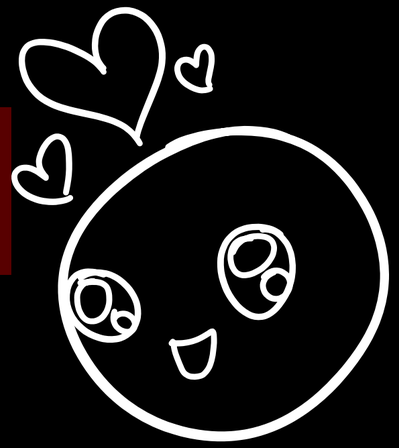
```
Handles.DrawLine(target.transform.position + target.vertex[link],
                 target.transform.position + target.vertex[i]);
```



P.V y SCENE

Por qué crear una Matriz4x4?

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

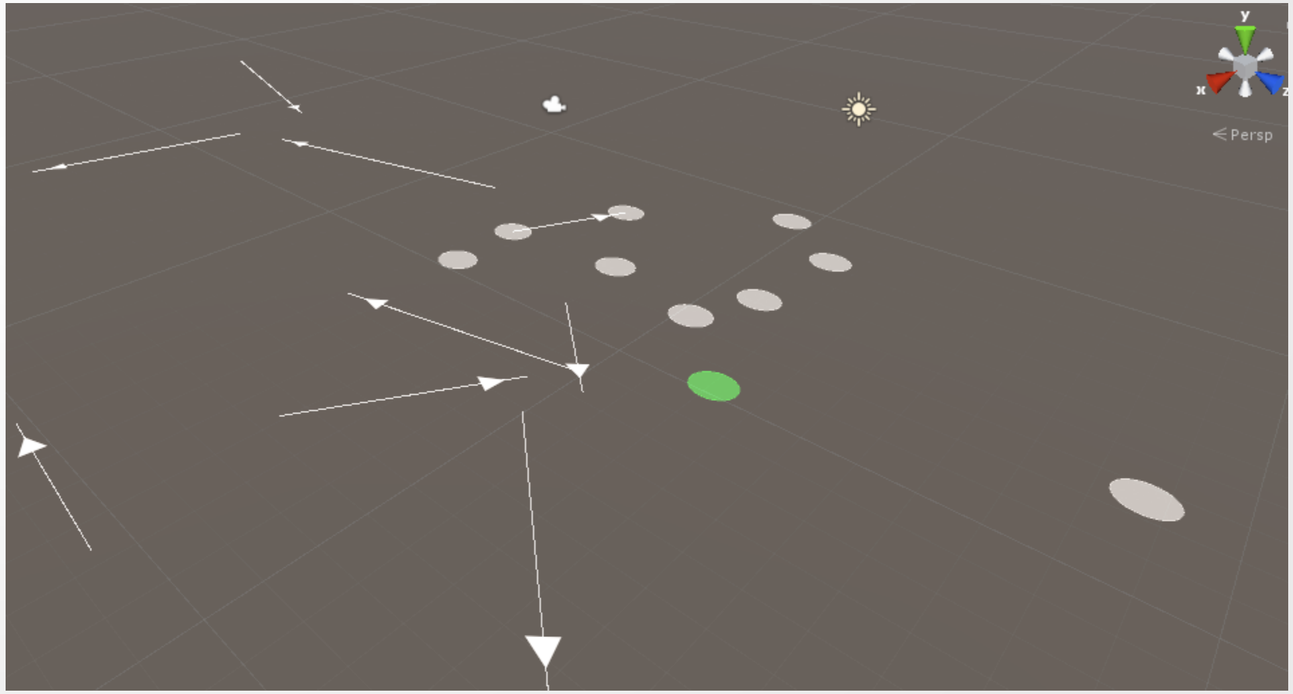


P.V y SCENE

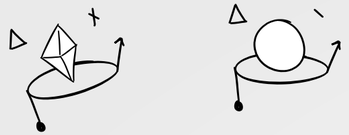
Devuelve todo a su lugar

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

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

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

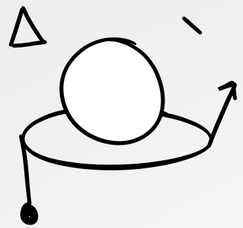
```
public static void ArrowHead (Vector3 position, Vector3 direction,
                              float size, float offset = 0) {
    direction.Normalize();
    Matrix4x4 old = Handles.matrix;
    Handles.matrix = old *
        Matrix4x4.TRS(position, Quaternion.LookRotation(direction),
         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



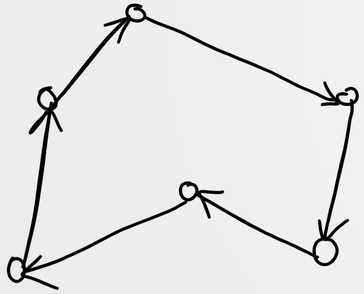
Reusabilidad



REUSABILIDAD

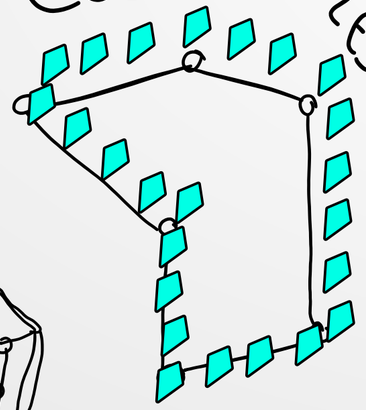


MOTION

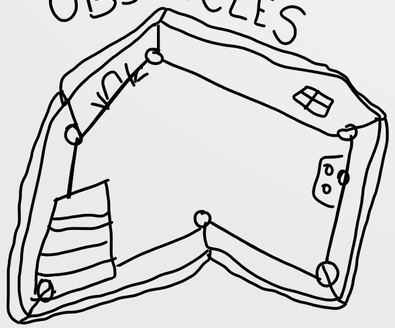


COOL EFFECTS

COLLECTABLES



OBSTACLES



Data Structure

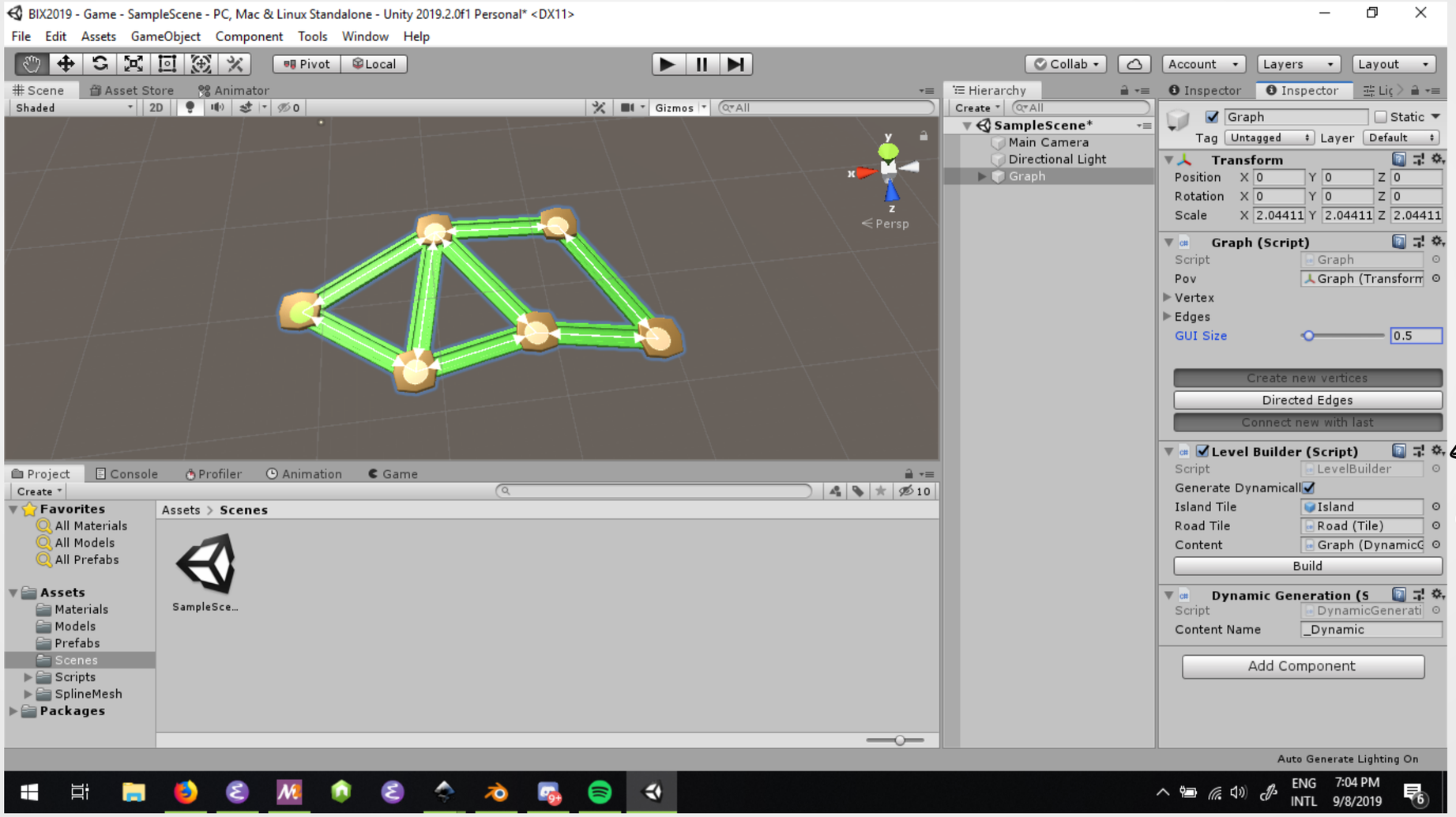
knows Nothing

Graph : MonoBehaviour

talks to

Motion : MonoBehaviour

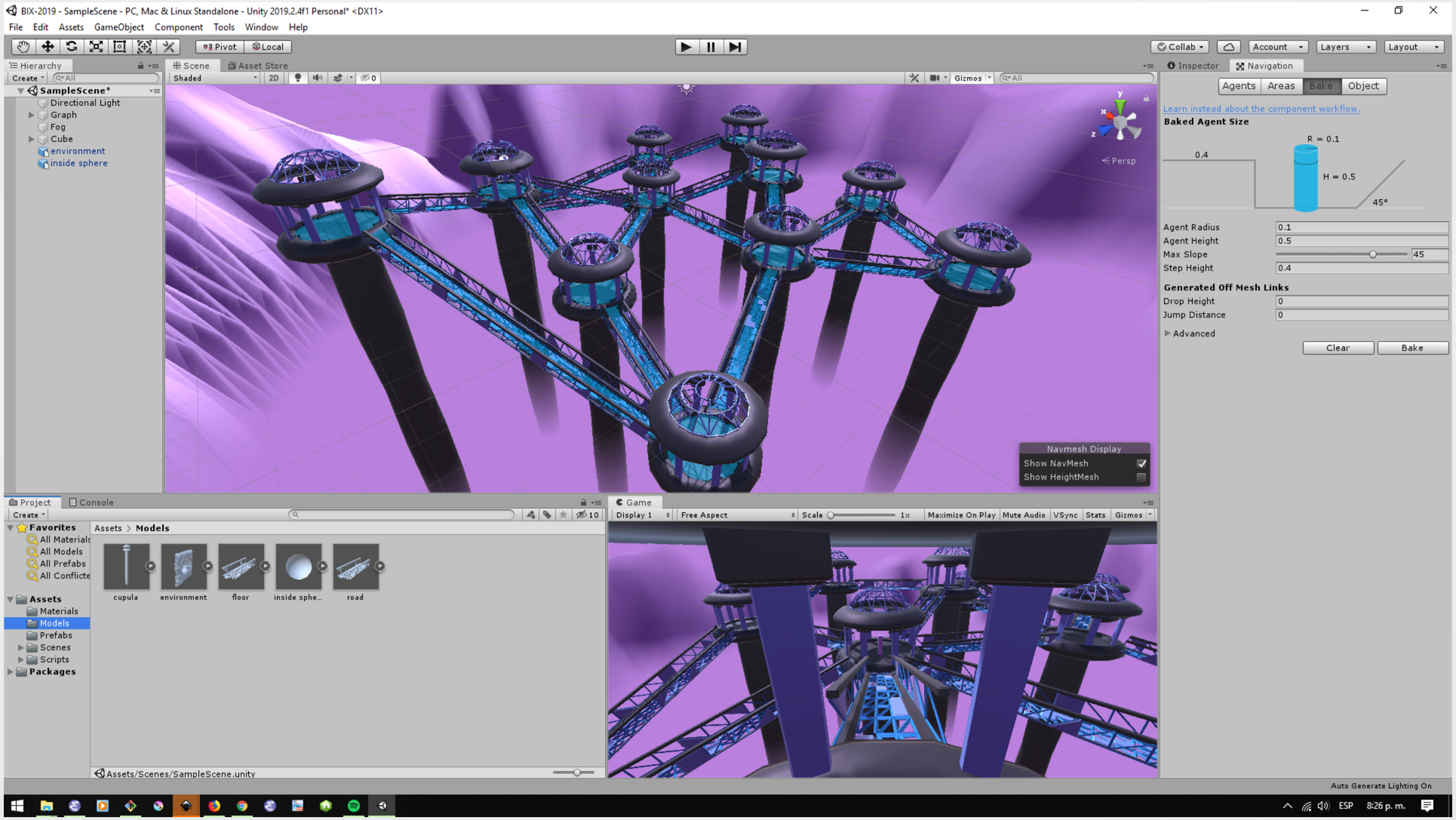
Reusabilidad



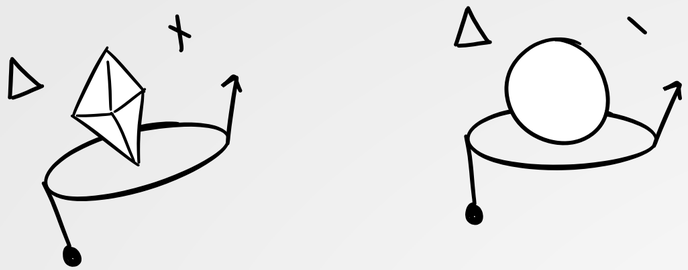
info sobre el grafo

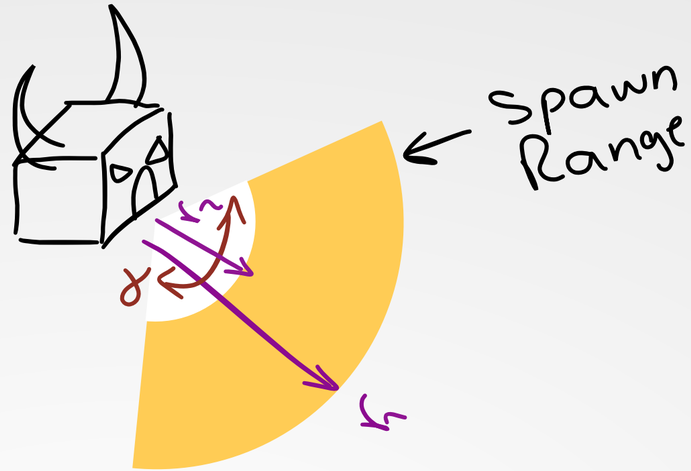
Construye cosas con esa información

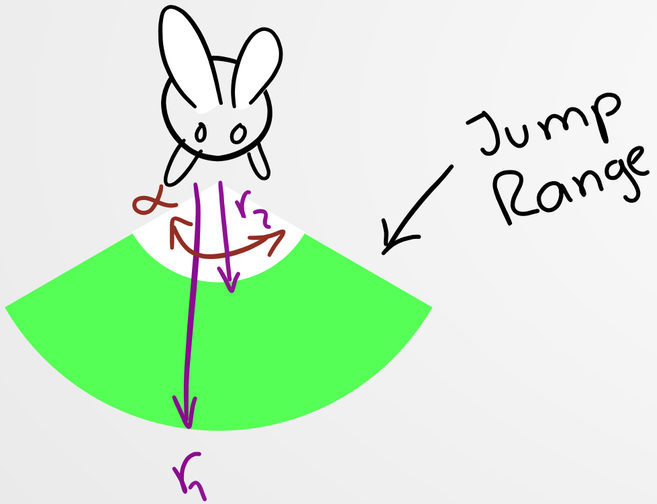
Reusabilidad

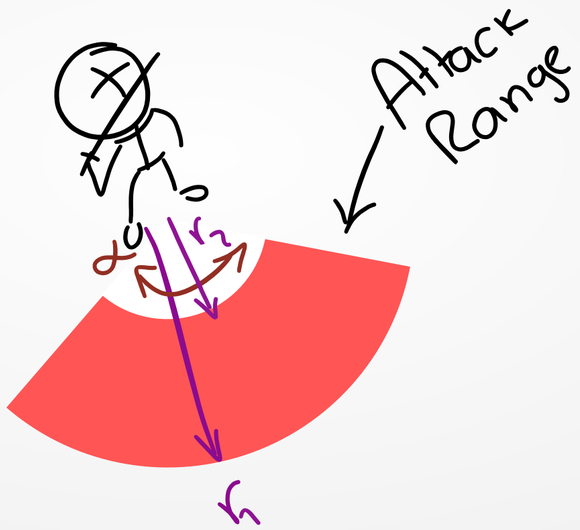


REUSABILIDAD



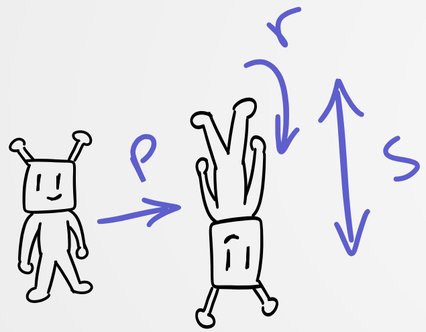




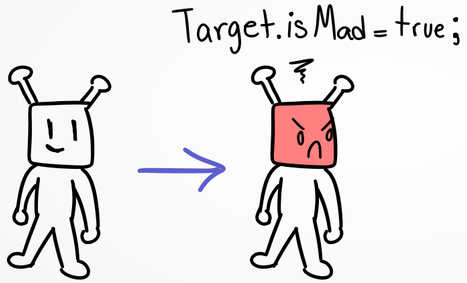


Persistencia

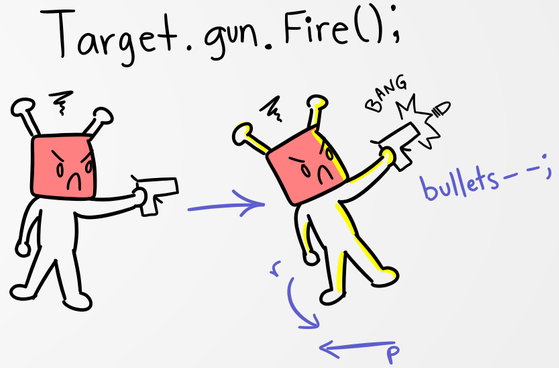
```
Undo.RecordObject(*, message);
```



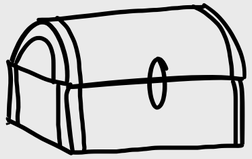
transform



Target



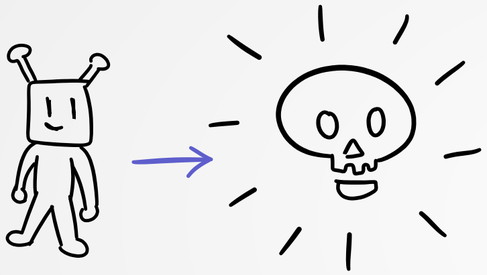
Target.gun; transform;



PERSISTENCIA

No!!

```
Destroy(Target.gameObject);
```



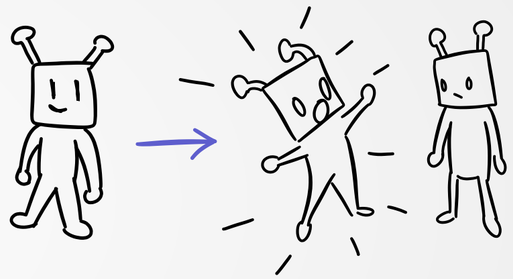
Undo.DestroyObjectImmediate

Util.Destroy(x)

UnityEngine.Object

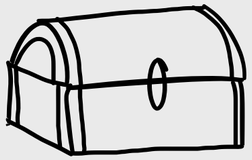
Platform dependant compilation

```
c = Instantiate(Target);
```



Undo.RegisterCreatedObjectUndo(c)

Util.Instantiate



Persistencia

[System.Serializable]

Visible y editable en el inspector!

```cs
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;
}
```

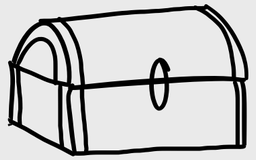
Stats.cs All L6 Git:master (C#//l +5 yas)

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

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

    // ...
}
```

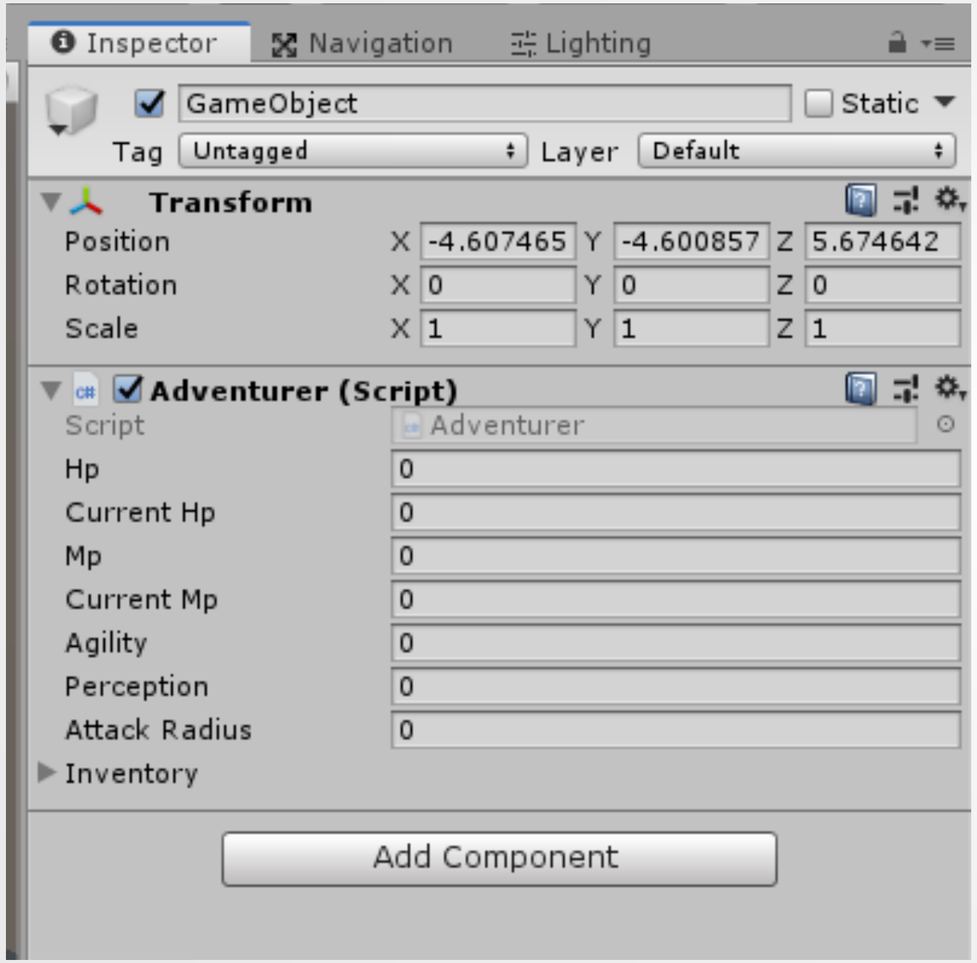
Adventurer.cs All L9 Git:master (C#//l +2 yas)

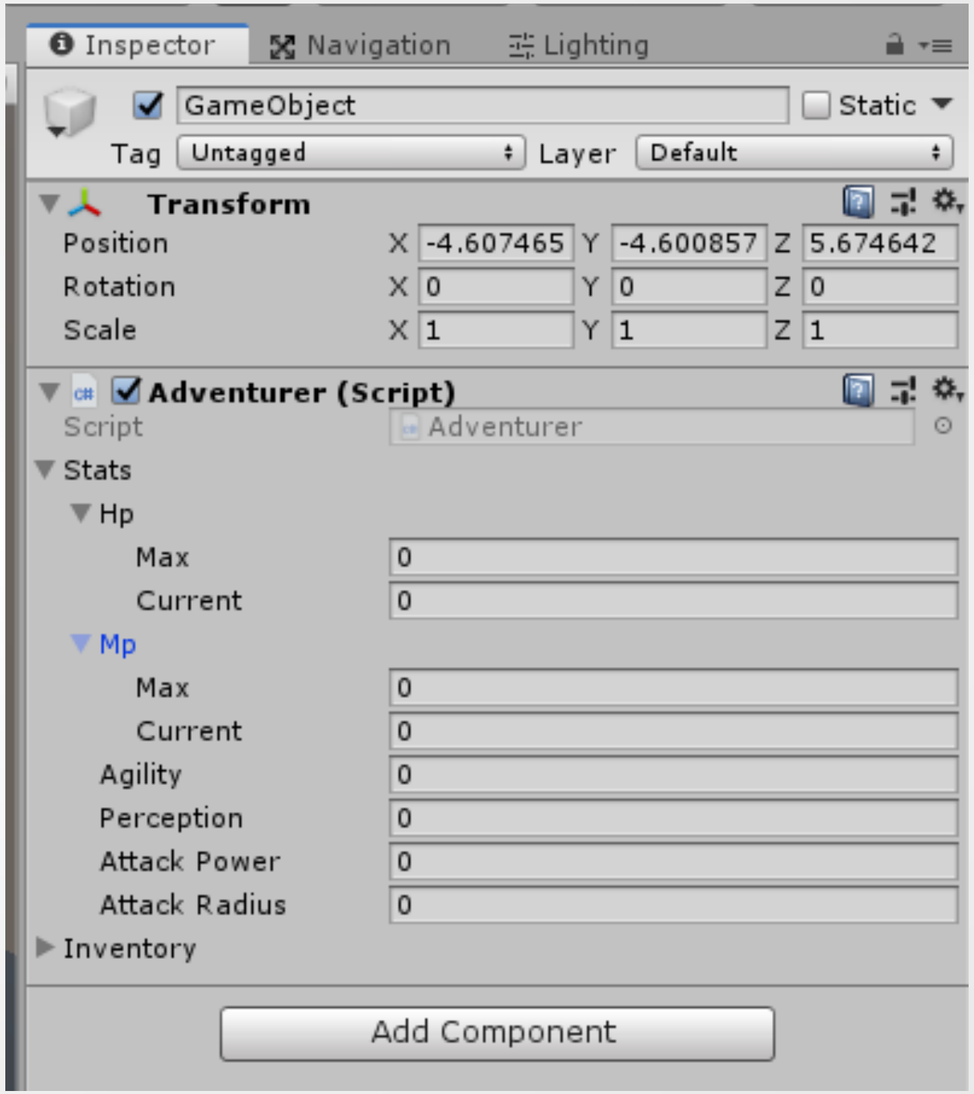
```cs
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

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

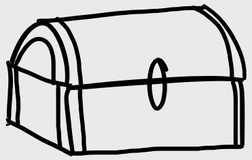
RenewableStat.cs All L1 Git-master (C#//l +5 yas)



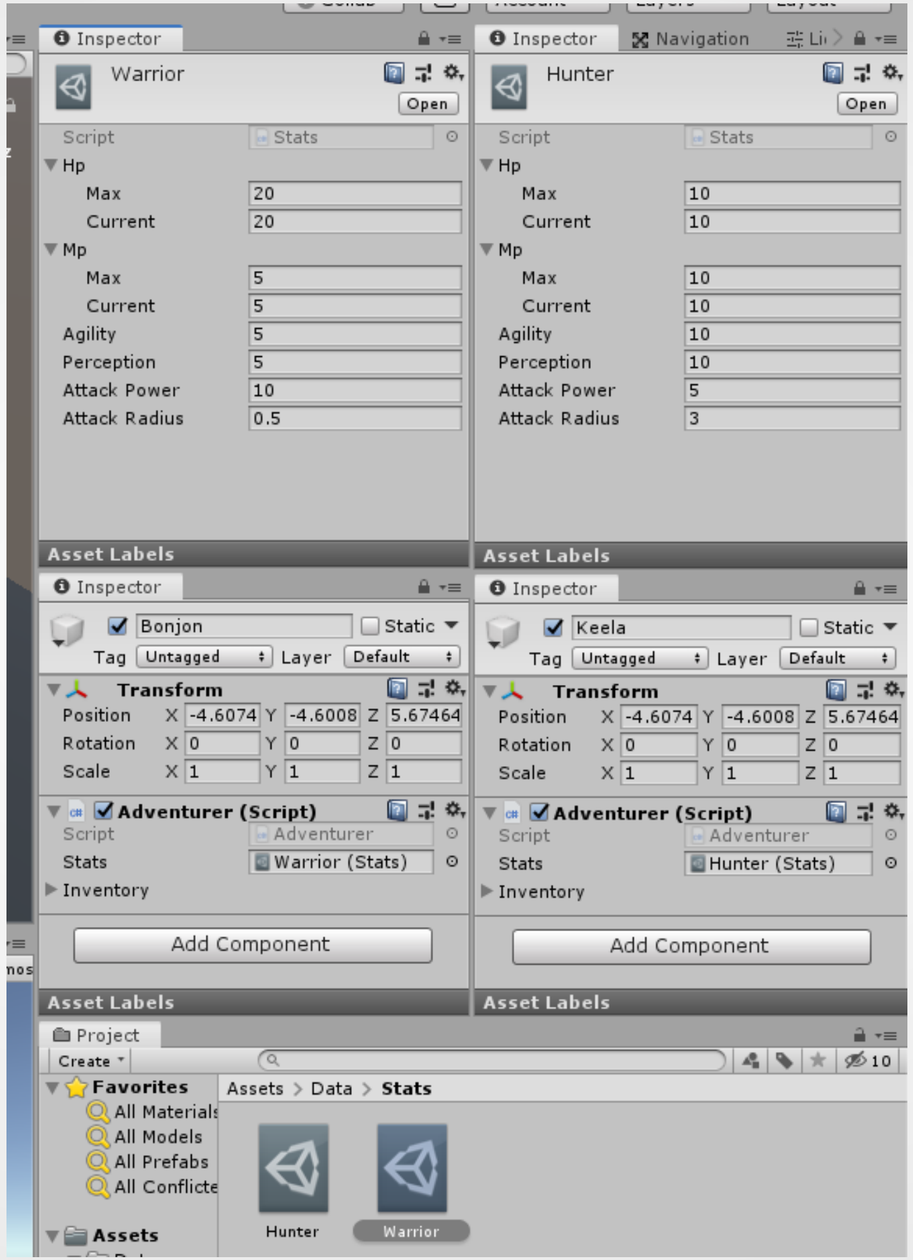
PERSISTENCIA

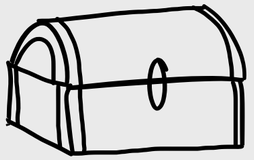






PERSISTENCIA





Persistencia

```cs
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;
}
```

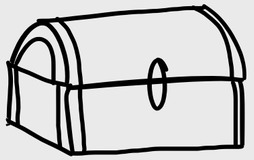
-\--- Stats.cs All L16 Git:master (C#//1 +4 yas)

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

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

    // ...
}
```

-\--- Adventurer.cs All L9 Git:master (C#//1 +2 yas)

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

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

-\--- RenewableStat.cs All L1 Git-master (C#//1 +5 yas)



PERSISTENCIA

Dictionary X

Interface X

private * variables

Si no se ve, no se guarda*

PERSISTENCIA

guarda variables privadas y las muestra en el inspector

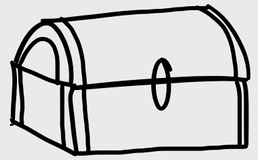
[SerializeField]

- static X
- get{} set{} X
- dictionary X
- interfaces X

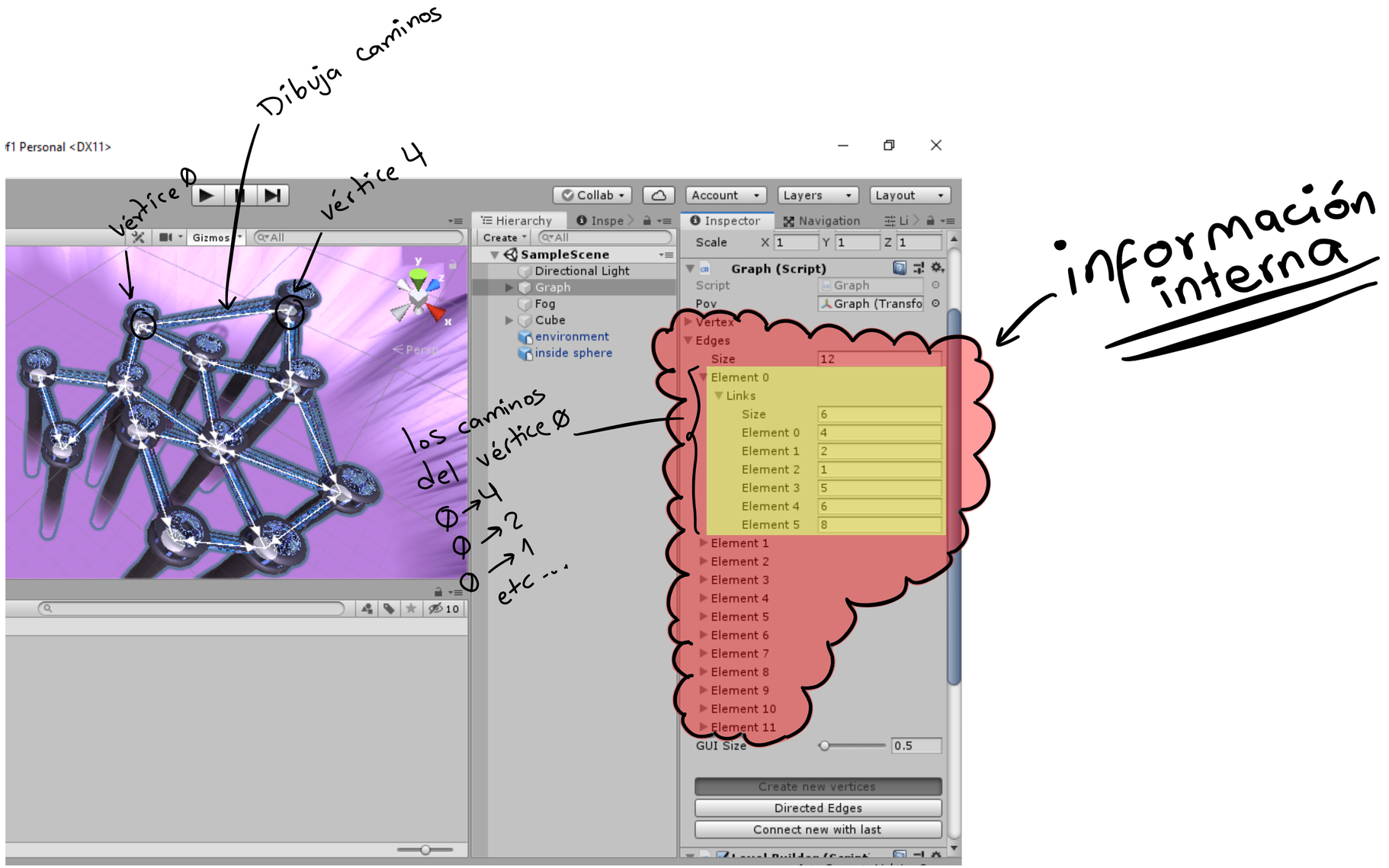
Esconde cosas visibles en el inspector

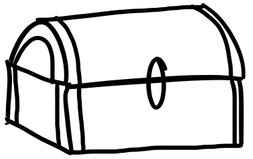
[HideInInspector]

(pero, si eran visibles, las guarda)



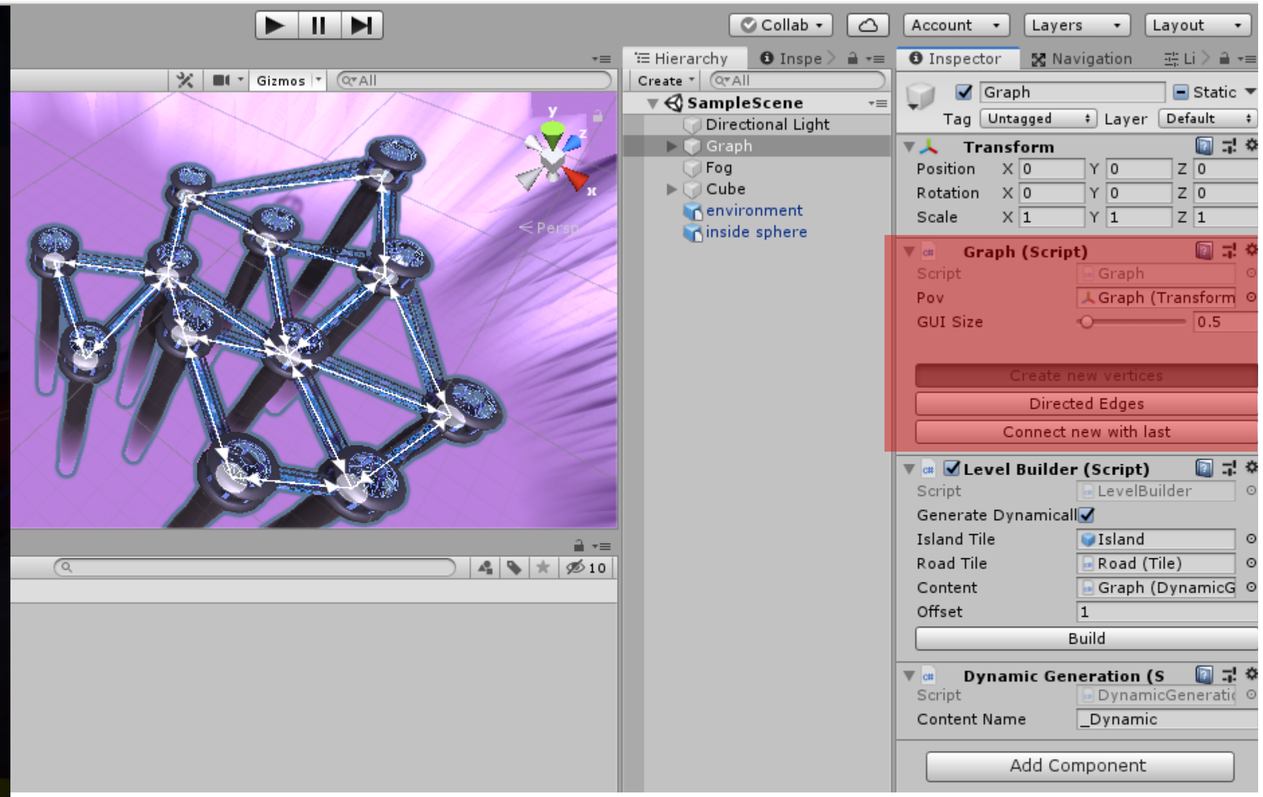
PERSISTENCIA

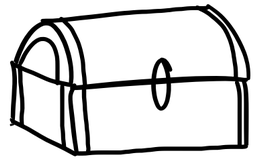




Persistencia

```
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;
```

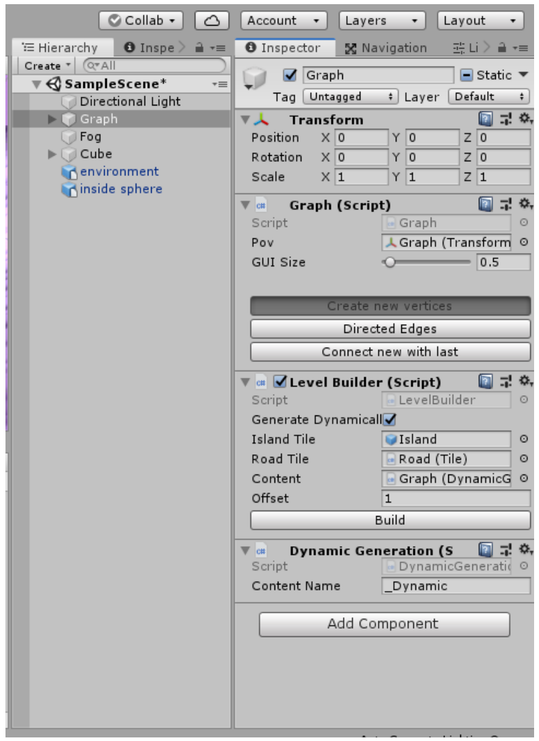




PERSISTENCIA

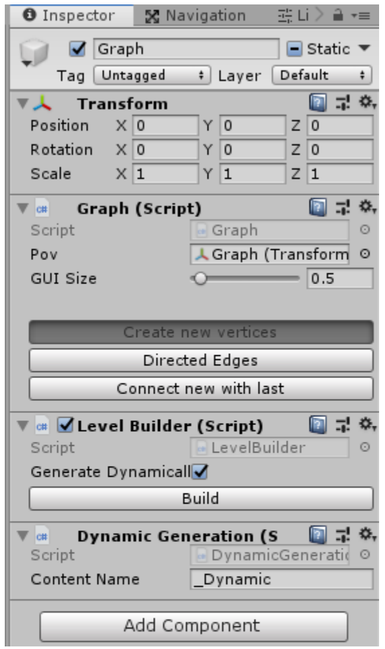
① Mostrar y Configurar

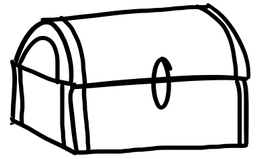
```
[ExecuteInEditMode]
public class LevelBuilder : MonoBehaviour {
    Graph _g;
    Graph _Graph { get { if (!_g) _g = GetComp

    public bool generateDynamically = false;

    [SerializeField]
    GameObject _islandTile;

    [SerializeField]
    Tile _roadTile;

    [SerializeField]
    DynamicGeneration _content;

    [SerializeField]
    float _offset = 1;

    void OnEnable () {
        _Graph.onGraphModified += GraphModifie
    }

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



② Ocultar y crear prefab

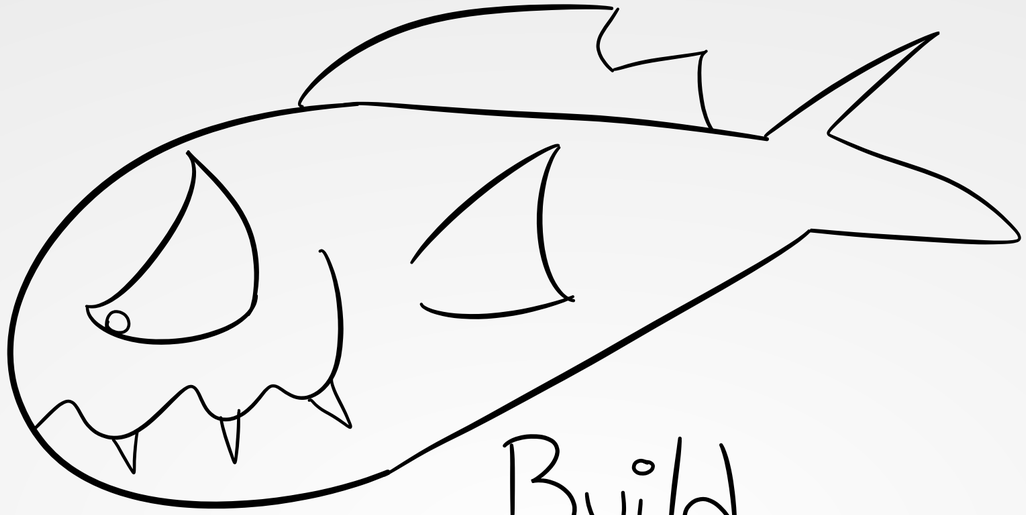
```
[ExecuteInEditMode]
public class LevelBuilder : MonoBehaviour {
    Graph _g;
    Graph _Graph { get { if (!_g) _g = GetComp

    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 += GraphModifie
```





COMUNICACIÓN

UnityEditor



Build

GameObject

Cosas que el .exe necesita

estado del Componente

Build

X

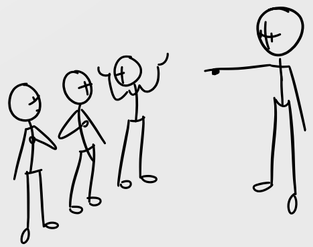
Editor Scripting

estado del editor

toolboxes

preferencias del editor

X



COMUNICACIÓN

OnInspectorGUI

↕ Variables en la misma clase

OnSceneGUI

```
[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 OnInspectorGUI

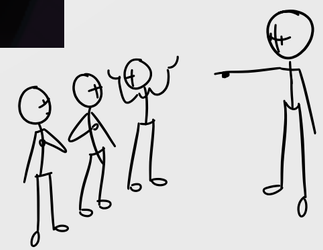
```
createNewVertices =
    GUILayout.Toggle(createNewVertices,
                     "Create new vertices", "Button");
directedEdges =
    GUILayout.Toggle(directedEdges,
                     "Directed Edges", "Button");
continuousConnection =
    GUILayout.Toggle(continuousConnection,
             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

Read OnSceneGUI

```
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);
```

```
if (createNewVertices) {
    UpdateVertexCreation(target);
    CoolEditor.HideTool();
```

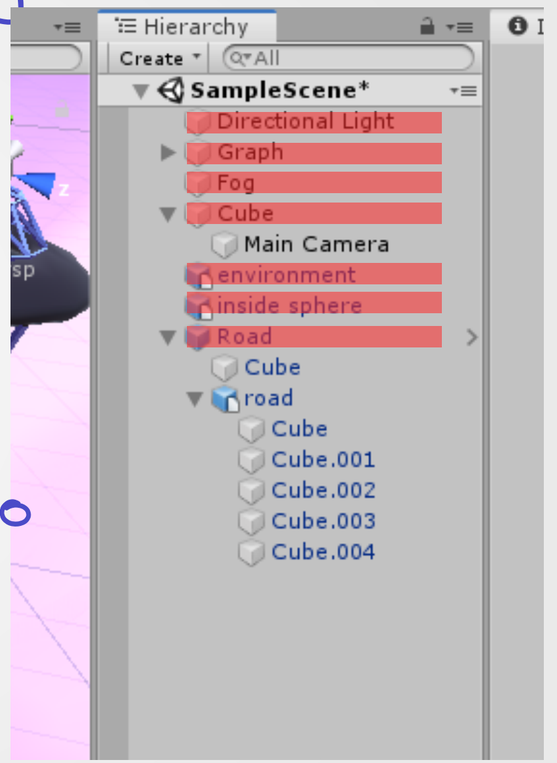


COMUNICACIÓN

Target.gameObject.

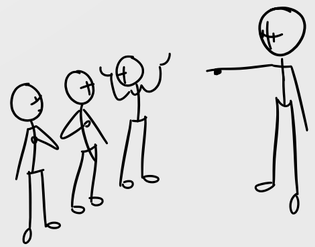
GetRootGameObjects

GameObject[]



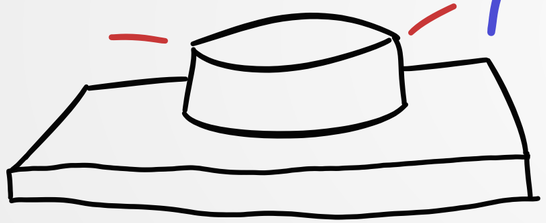
OnEnable()

Cuando un objeto acaba de ser seleccionado

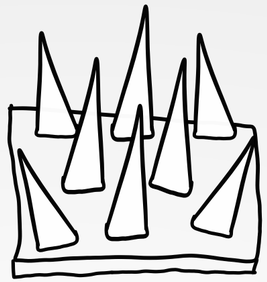


COMUNICACIÓN

al seleccionarlo desde el editor

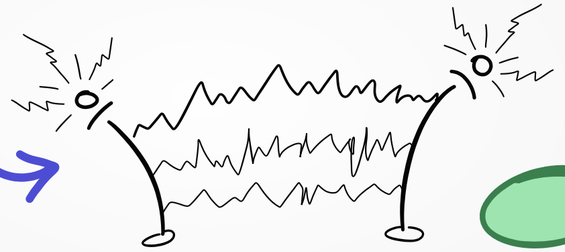


Button



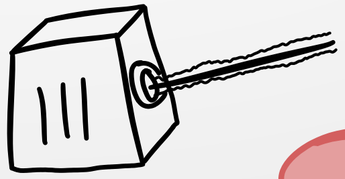
spikes

Aparecen éstos botones en escena



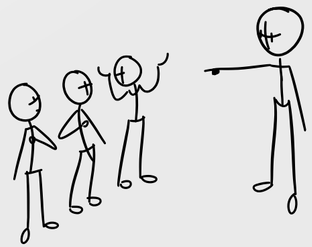
Electric Doors

verde si el objeto está conectado al botón

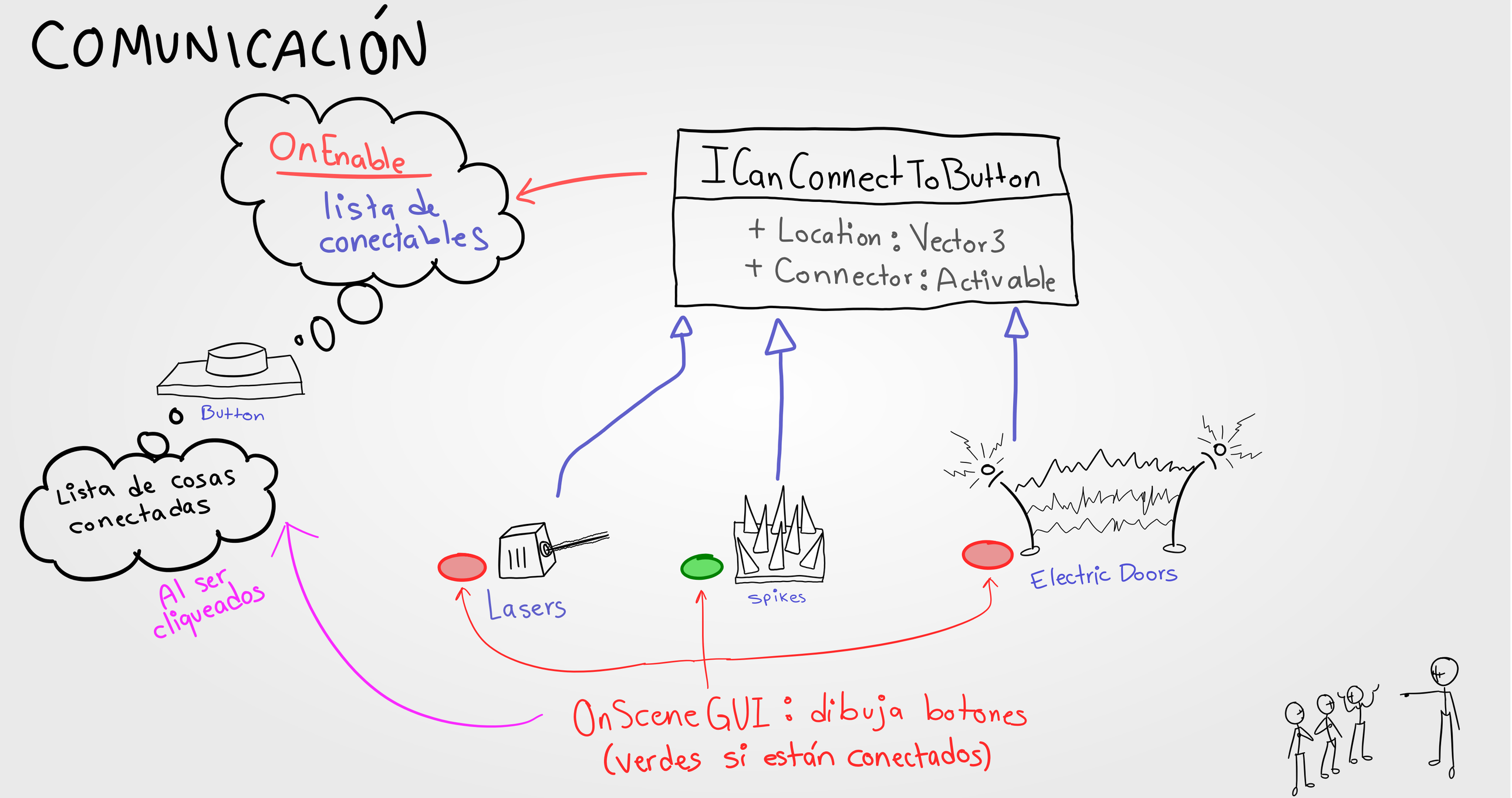


Lasers

rojo si no lo está



COMUNICACIÓN



Gracias!!

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