DOCUMENTACIÓN

Clase: Inteligencia Artificial	Práctica: 4 [Generación procedural]
Alumno: Miguel Rodríguez Gallego	

Motivos de mi práctica

Quise hacer una práctica que llegase a los mínimos de forma adecuada y ya explayarme en estudiar más la generación procedural cuando tenga más tiempo, ya que es un tema super interesante.

Sin embargo, por estas razones escogí la generación procedural por perlin noise.

Proceso de creación

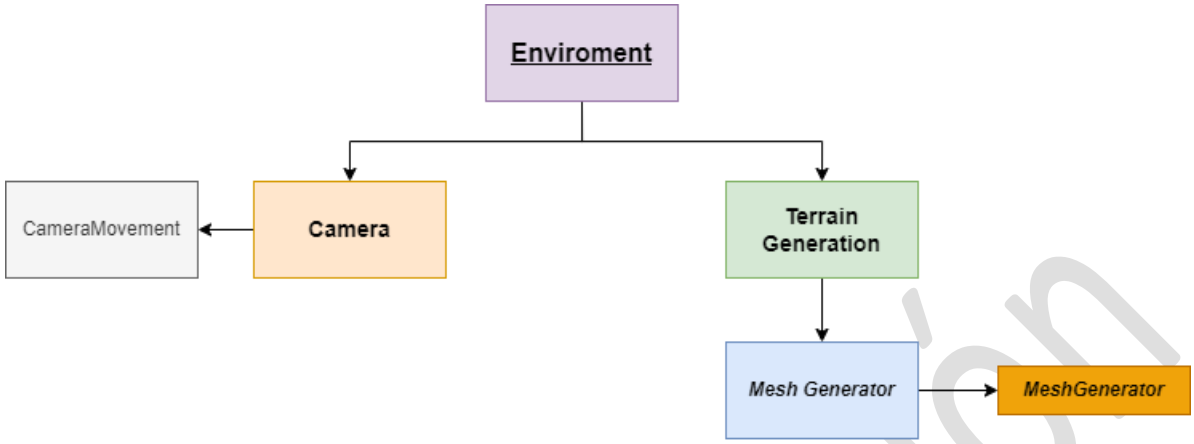
Comencé recogiendo los scripts del movimiento de la cámara y la base en general de organización de archivos de anteriores prácticas…

Luego procedí a informarme sobre el tema de generación procedural simple, por lo que acudí a un tutorial de Brackeys de generación para entender como hacerlo por Perlin Noise de manera fácil, rápida y eficiente (https://www.youtube.com/watch?v=64NblGkAabk&t=579s).

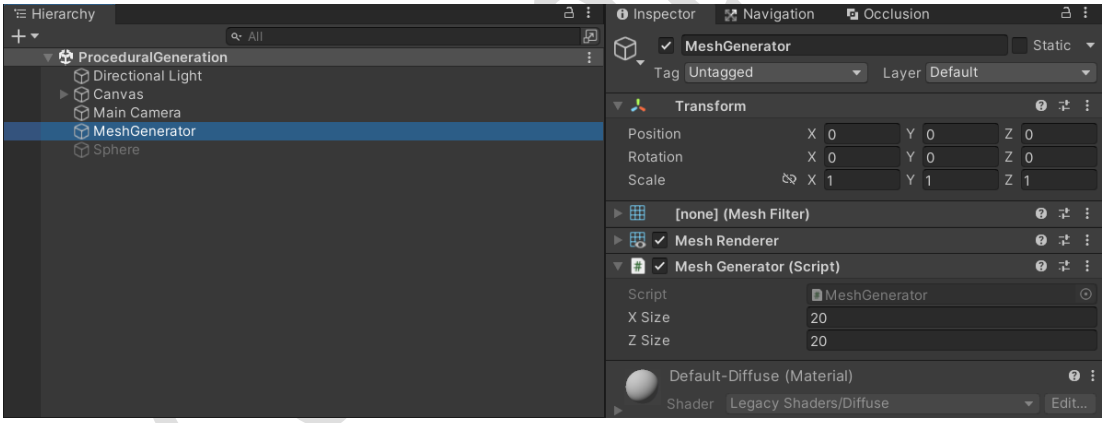
Dificultades y como se resolvieron

Intenté hacer que fuese interactuable poniendo un mesh collider, pero no fui capaz en ningún momento, y probé diferentes maneras, sin embargo, ninguna dio un resultado óptimo por lo que al final lo dejé en un terreno simple.

Diagrama de clases



- Camera
  - CameraMovement: Poder mover la cámara cuando presionas espacio y tmb pausarla con la misma tecla para que salga el cursor y puedas interactuar con el ambiente.
- Mesh Generator
  - MeshGenerator: Crea la malla en el objeto en el que está colocado.




```
7   Mesh mesh;
8
9   Vector3[] vertices;
10  int[] triangles;
11
12  // Mesh terrain size values
13  [SerializeField] int xSize = 20;
14  [SerializeField] int zSize = 20;
15
16  /// <summary>
17  ///     Get mesh references and initialize procedural world generation
18  /// </summary>
    Mensaje de Unity | 0 referencias
19  void Start()
20  {
21      mesh = new Mesh();
22      GetComponent<MeshFilter>().mesh = mesh;
23
24      CreateShape();
25      UpdateMesh();
26  }
```

```
27
28      /// <summary>
29      ///     Create mesh triangle values
30      /// </summary>
        1 referencia
31      void CreateShape()
32      {
33          vertices = new Vector3[(xSize + 1) * (zSize + 1)];
34
35          // Triangle vertices positions
36          for (int i = 0, z = 0; z <= zSize; z++)
37          {
38              for (int x = 0; x <= xSize; x++)
39              {
40                  float y = Mathf.PerlinNoise(x * .3f, z * .3f) * 2f;
41                  vertices[i] = new Vector3(x, y, z);
42                  i++;
43              }
44          }
45
46          triangles = new int[xSize * zSize * 6];
47
48          int vert = 0;
49          int tris = 0;
50
51          // Set triangle length => Update triangle points
52          for (int z = 0; z < zSize; z++)
53          {
54              for (int x = 0; x < xSize; x++)
55              {
56                  triangles[tris + 0] = vert + 0;
57                  triangles[tris + 1] = vert + xSize + 1;
58                  triangles[tris + 2] = vert + 1;
59                  triangles[tris + 3] = vert + 1;
60                  triangles[tris + 4] = vert + xSize + 1;
61                  triangles[tris + 5] = vert + xSize + 2;
62
63                  vert++;
64                  tris += 6;
65              }
66              vert++;
67          }
68      }
```

```
70      /// <summary>
71      ///     Calculate mesh
72      /// </summary>
        1 referencia
73      void UpdateMesh()
74      {
75          mesh.Clear();
76
77          mesh.vertices = vertices;
78          mesh.triangles = triangles;
79
80          mesh.RecalculateNormals();
81      }
```

Autocrítica

Creo que ha sido mi práctica más vaga con diferencia, pero estoy priorizando el tiempo para el resto de las asignaturas pendientes también, que tengo poco tiempo este curso.

Entonces, estoy contento de que funcione, pero seguro que intentaré ahondar y mejorar este concepto en el momento que tenga más tiempo en un mes y pico, porque es muy interesante.