

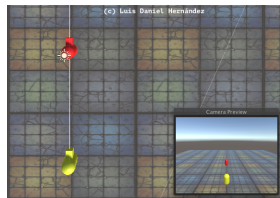
IA en Videojuegos

Segundo Proyecto

L. Daniel Hernández¹

Departamento de Ingeniería de la Información y las
Comunicaciones
Universidad de Murcia

25 de enero de 2022



¹Erratas y sugerencias a ldaniel@um.es. Algunas imágenes pueden ser de internet. Si tuvieran derechos de autor/distribución conocidos, por favor, avisen.

Índice de Contenidos

- 1 Crear materiales con textura
- 2 Editar un Prefab
- 3 Vigilar a objetos
- 4 Varias cámaras en la escena
- 5 Gizmos

Crear materiales con textura

Importando la textura

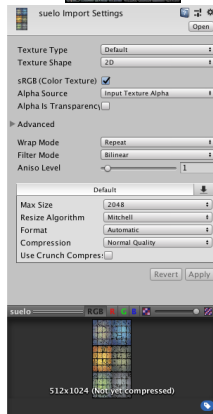
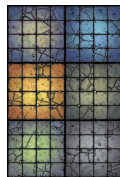
- Descargar cualquier imagen de un suelo con baldosas de distintos colores. A la derecha un ejemplo.
- Guarda el fichero en una nueva carpeta llamada Textures.

Las texturas son imágenes y vídeos que cubren a los objetos. Esta imagen la usaremos para el suelo, así que acabamos de crear una textura.

- Pincha sobre la imagen en el proyecto. En la ventana de inspección tendrás los "Parámetros de importación" que indican cómo la imagen será importada en el proyecto. Deja los valores por defecto.



Textures

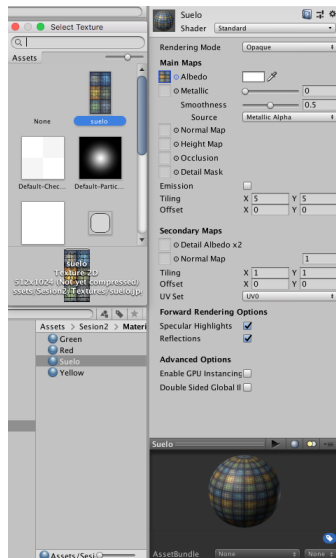


Materiales con textura

- Crea un material nuevo. Llámalo Suelo.
- Pincha sobre Albedo y selecciona la textura que has guardado en la etapa anterior.
- Indica en Tiling los valores $X=5$ e $Y=5$.



Textures



Editar un Prefab

Editar un Prefab en modo Prefab

- Haz doble click sobre el Prefab del suelo (que llamaremos Floor).

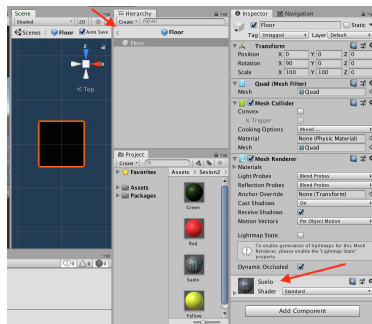
Se entrará en modo de edición del Prefabs. La ventana de jerarquías cambiará a modo de edición de Prefabs.

- Selecciona la carpeta Materiales, donde tienes el nuevo material Suelo construido en el paso anterior.

- Arrastra el material Suelo (que se encuentra en la ventana del proyecto) al objeto Floor (que se encuentra en la ventana de edición de Prefabs). Es la misma acción que se realizó cuando el prefab Floor, antes de ser prefab, era un objeto de la escena (ver Sesión 1).

- Sal del modo edición haciendo click sobre el símbolo < junto al nombre del prefab en la ventana Hierarchy.


Como ejemplo sencillo aplicaremos el nuevo material al prefab Floor.



Editing a Prefab in Prefab Mode

Vigilar a objetos

Enlazar objetos con variables

Algunos objetos del juego necesitan seguir la pista a otros. P.e. para perseguir a un enemigo se necesita su posición. Unity proporciona varias formas de hacerlo. Una de ellas es usar scripts con **variables globales que almacenen la referencia del gameobject**.  **Controlling GameObjects using components**

- Partir de la escena con dos prefabs: Floor (el suelo) y Player (el jugador, controlado por un script).
- Crea un nuevo objeto de juego: Seek. Haz que sea una modificación de Player, sin el script Controller y con una textura diferente (p.e. de color amarillo).
- Añadir a Seek la componente-script Seek.cs (ver transparencia siguiente)
- Arrastrar el objeto de la escena Player al parámetro Target que aparece en la componente Seek (Script) (ventana inspector).
- Establece las velocidades 2 y 8 para los objetos Seek y Player, respectivamente.
- Antes de ejecutar el juego prepara las cámaras (siguiente sección).

Enlazar objetos con variables

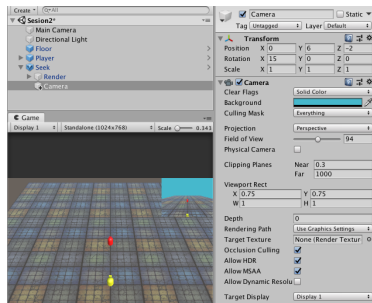
```
1 public class Seek : MonoBehaviour
2 {
3     public Transform target; // Referencia al Transform de un objeto externo.
4     public float velocity = 2;
5
6     void Update()
7     {
8         // Diferencia de posiciones entre el objetivo y este objeto.
9         Vector3 newDirection = target.position - transform.position;
10
11        // Mirar en la dirección del vector calculado.
12        transform.LookAt(transform.position + newDirection);
13
14        // Avanzar de acuerdo a la velocidad establecida
15        transform.position += newDirection * velocity * Time.deltaTime;
16    }
17 }
```

Varias cámaras en la escena

Dos cámaras en nuestra escena

- Con el suelo centrado en la escena, establece los siguientes parámetros en la cámara principal: Position (0, 20, -17), Rotation (54, 0, 0).
- Cómo crear una cámara en tercera persona:
 - Crea una nueva cámara y añádela en la jerarquía del objeto Seek de la escena.
 - Establece los siguientes parámetros: Posición(0, 6, -4), Rotation(25, 0, 0).
- Cómo mostrar simultáneamente las capturas de las dos cámaras de la escena:
 - Superposición de capturas: En el parámetro Depth establece -1 para la cámara principal y 0 para la cámara de Seek.
 - Dónde mostrar: En Viewport Rect cambiar los valores para la cámara de Seek: X=0.75 y Y=0.75

Ejemplo sencillo de dos cámaras: una para la escena y otra de tercera persona.



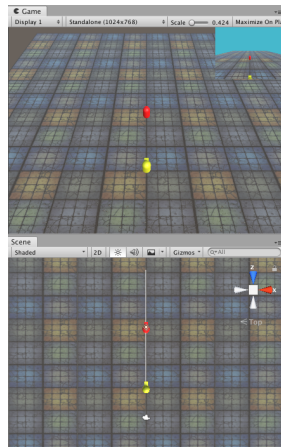
Cameras

Gizmos

Más información sobre la escena

Un gizmo es cualquier gráfico superpuesto en la vista escena y referente a un objeto. Son gizmos los ejes cartesianos o los rayos de luz. Podemos crear gizmos asociando a nuestros objetos scripts con el método `OnDrawGizmos()`. Son muy útiles para la depuración.

- Edita el script `Seek.cs` y añade el código que se indica en la siguiente transparencia.
- Pulsa sobre el eje Y del gizmo del sistema de coordenadas. Obtendrás una vista superior de la escena.
Observa la línea blanca que “sale” del objeto `Seek`, es el gizmo que acabamos de crear.
- Ejecuta el juego. Tendrás 3 perspectivas de lo que ocurre: esta nueva vista superior (en la escena) y las dos perspectivas cónicas de las dos cámaras.



Gizmos

Mostrar un gizmo

```
1 public class Seek : MonoBehaviour
2     ....
3
4     private void OnDrawGizmos()
5     // El gizmo: una línea en la dirección del objetivo
6     {
7         // Origen de la línea
8         Vector3 from = transform.position;
9
10        // Destino de la línea
11        Vector3 to = transform.localPosition +
12            (target.position - transform.position) * velocity;
13
14        // Elevación para no tocar el suelo
15        Vector3 elevation = new Vector3(0, 1, 0);
16
17        from = from + elevation;
18        to = to + elevation;
19
20        Gizmos.DrawLine(from, to);
21    }
22 }
```