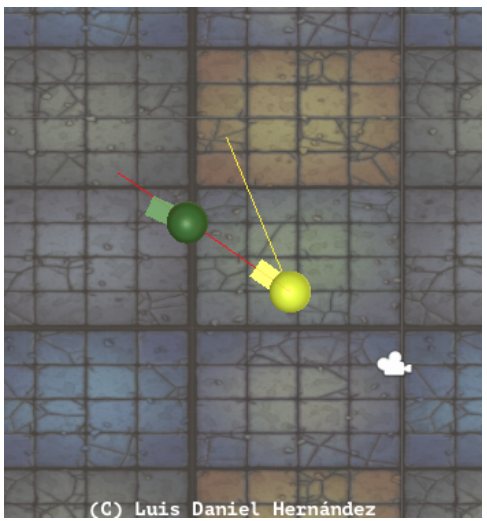
IA en Videojuegos

Tercer Proyecto

L. Daniel Hernández[1]

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

25 de enero de 2022



[1] Erratas 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

La Escena

La escena

La escena de esta sesión constará de los siguientes elementos:

- Un suelo.
- 4 muros que delimitan los límites del suelo.
- Un paseante aleatorio. Cada vez que choque con un muro retornará al centro del escenario.
- Un perseguidor del paseante que se desplaza con aceleración no nula.
- Dos cámaras. Cada uno sigue a cada uno de los personajes.

Desarrollemos las distintas componentes.

El suelo

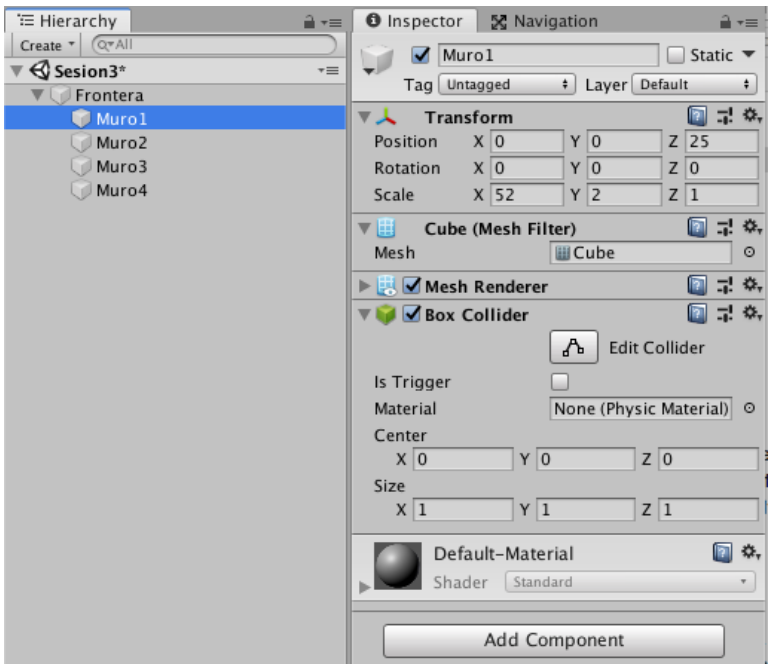
- Se construirá igual que en la segunda sesión.
- Establecer un Quad con:
  - Position (0, 0, 0)
  - Rotation (90, 0, 0)
  - Scale (50, 50, 0)

  Solo se cambiará la escala con respecto a la sesión 2.

Muros frontera

Creación de muros

- Crear un GameObject vacío en la posición (0, 0, 0) y llamarlo `"Frontera"`.
- Crear 4 cubos que sean hijos del GameObject anterior con nombres `"Muro 1"`, `"Muro 2"`, `"Muro 3"` y `"Muro 4"`.
- Dar a cada uno las siguientes posiciones y escalas:
  - Posición (0, 0, 25); Scale (52, 2, 1).
  - Posición (0, 0, -25); Scale (52, 2, 1).
  - Posición (25, 0, 0); Scale (1, 2, 52).
  - Posición (-250, 0, 0); Scale (1, 2, 52).
- Asegurarse de que todos los muros tienen activa la componente `Box Collider`.



Paseante aleatorio

Movimiento lineal

Nuestro objetivo es construir un personaje con el siguiente movimiento: (1) rotar (cambiar la orientación) cada cierto intervalo δt de tiempo y (2) ir en línea recta durante este δt.
El movimiento lineal es un problema resuelto que resolvemos de 3 formas.

- En la sesión 1: dado un nuevo vector dirección calcular la nueva posición.
- En la sesión 2: dado una posición dirigirse a esa dirección.
- En esta sesión: cambiar la posición en la dirección de la orientación del personaje.

La orientación del GameObject es el valor `transform.forward`.

```
1 public float velocity = 4;     // Máxima velocidad

3 void Update()
4 {
5   transform.position += transform.forward * velocity * Time.deltaTime;
6 }
```

 Transform.forward

Rotación Aleatoria con Movimiento Lineal

```
1  // Control del tiempo
2  public float deltaTime = 0.25f; // Intervalo de tiempo.
3  private float time = 0f;       // Tiempo acumulado entre frames.

5  // Control de la orientación
6  public float maxAngle = 360; // Una circunferencia
7  private float angle = 0f;      // Angulo de orientación actual.
8  private float newAngle = 0f; // Nuevo ángulo (será aleatorio)

10 void Update()
11 {
12   time += Time.deltaTime;  // Tiempo acumulado transcurrido

14   if (time >= deltaTime)  // Si se alcanzó el intervalo temporal ...
15   {
16     newAngle = (Random.value - Random.value) * maxAngle;  // cambiar el ángulo de orientación.
17     time = 0;                              // Vuelta a empezar
18   }
19    // Linearly interpolates between a and b by t. Especial para ángulos.
20   angle = Mathf.LerpAngle(angle, newAngle, Time.deltaTime);    // Cambio suave

22   // The rotation as Euler angles in degrees. Rotación para esa orientación.
23   transform.eulerAngles = new Vector3(0, angle, 0);

25   // Movimiento lineal
26   transform.position += transform.forward * velocity * Time.deltaTime; }
```

 Rotación y Orientación en Unity  Transform.eulerAngles

MUY IMPORTANTE PARA LA PRÁCTICA: para cada personaje deberás tener su atributo `angle` y aplicar tú el correspondiente giro.

Corrutinas

Rotación con corrutinas

- Antes has visto cómo realizar una rotación aleatoria transcurrido cierto tiempo.
- La rotación aleatoria también se puede hacer con corrutinas (función que puede parar su ejecución, devolver el control a Unity y continuar por donde se quedó en el siguiente frame o en el tiempo que se establezca.)
- El código es el siguiente:


Coroutines

```
1  IEnumerator NewDirection()
2  {
3    while (true)
4    {
5      // yield, cede el control a Unity. Rotomará el control pasado 0.25f
6      yield return new WaitForSeconds(0.25f); // Cambiar orientación
7
8      newAngle = (Random.value - Random.value) * maxAngle;
9    }
10 }
```

- La corrutina se invoca con `StartCoroutine()` en el método `Start()`.

```
1  void Start()
2  {
3    StartCoroutine("NewDirection");   // Inicia la corrutina
         NewDirection
4  }
```

Rotación aleatoria con corrutinas y movimiento lineal

- Copia el código de la transparencia anterior en el fichero nuevo `Wander.cs`.
- Añade le siguiente código:

```
1  public float velocity = 4;     // Máxima velocidad
2  public float maxAngle = 360; // Una circunferencia
3  private float angle = 0f;    // Angulo de orientación actual.
4  private float newAngle = 0f; // Nuevo ángulo (será aleatorio)
5
6
7  void Update()
8  {
9    angle = Mathf.LerpAngle(angle, newAngle, Time.deltaTime);
10   transform.eulerAngles = new Vector3(0, angle, 0);
11   transform.position +=
12                       transform.forward * velocity * Time.deltaTime;
13 }
```

pues ya no se necesita las variables referente a los tiempos.

StartCoroutines() Start() Coroutines (Scripts)

¿Qué debe hacer para poder cambiar el tiempo desde el editor de Unity?

Personaje Wander

- Un personaje Wander es aquel que tiene rotación aleatoria con y movimiento lineal.
- Crea un personaje igual que el personaje Seek de la sesión anterior.
- Llama al personaje Wander.
- Añade el script `Wander.cs` (creado según la transparencia anterior) al personaje Wander que acabas de construir.
- Ejecuta el juego y veras un personaje que se mueve de forma aleatorio por el escenario.
- Tiene el problema de que atraviesa los muros, así que le vamos a darle percepción (tacto) en las siguientes transparencias.

Percepción

Algo sobre el motor físico

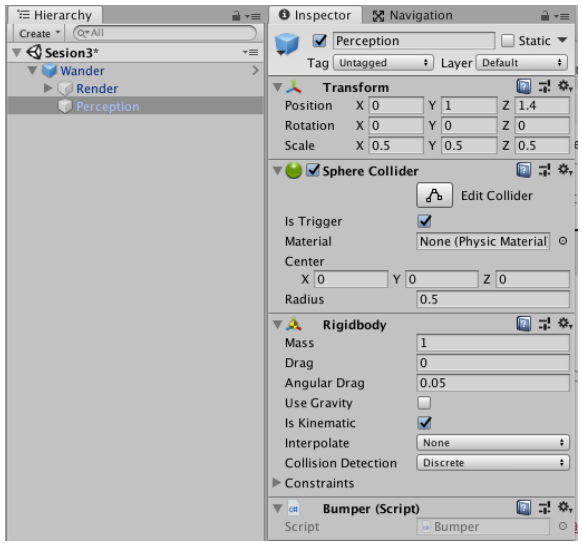
- Con la componente `RigidBody` los objetos se consideran en el motor físico y reaccionan a las colisiones solo si tiene una componente `Collider`.
- Se pueden controlar las colisiones con los métodos `OnCollisionXXX()` de `RigidBody`.
- Si se quiere usar el motor físico para detectar si un collider toca a otro (sin comportamiento de objeto sólido) se deberá:
  - En la componente `Rigidbody` marcar `Is Kinematic`. El objeto de esta componente no se moverá con el motor físico.
  - En la componente `Collider` marca `Is Trigger`. Se producirá un evento (trigger) cada vez que este collider choque con otro.
  - Se podrán controlar los triggers mediante Scripts con los métodos `OnTriggerXXX()` de `Collider`

 Rigidbody  Rigidbody (Script)

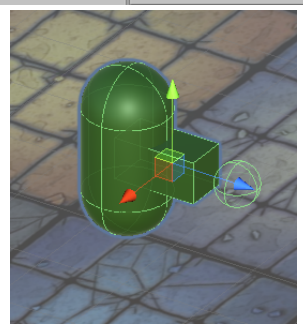
 Collider  Box Collider  Collider (Script)

Responde: ¿Qué consecuencias tiene que un objeto tenga collider pero no rigidbody? Un objeto de juego solo debe tener un Rigidbody ¿dónde debe colocarse?

Simular el tacto



- Crea un GameObject vacío en una posición que se encuentre "delante de la nariz" del personaje Wander.
- Añádele las componentes `Sphere Collider` y `Rigidbody` activando `Is Trigger` y `Is Kinematic`.
- Crea el script `Bumper.cs`.
  La función del script es colocar al personaje en el centro de la escena cada vez que el collider esférico (el sensor) detecte un choque con los muros de la frontera. Consta solo de la siguiente función:



```
1 void OnTriggerEnter(Collider other)
2 {
3   if (other.transform.root.name.Equals("Frontera")) // Si choca
4     transform.root.position = Vector3.zero;  // Reset posición
5 }
```

Añade este script al GameObject sin textura y vuelve a ejecutar el juego.

OnTriggerEnter(Collider) Transform

Perseguidor

Movimiento acelerado

- Crea el script `SeekAceleration.cs` con el siguiente contenido y se lo añades a un nuevo personaje que llamarás SeekAcc.

```
1  public Transform target;   // Comenta adecuadamente.
2  public float maxAceleration = 2;
3  public float maxVelocity = 4;
4  private Vector3 velocity = Vector3.zero;

6  void Update()
7  {
8    Vector3 newDirection = target.position - transform.position;

10   transform.LookAt(transform.position + newDirection);

12   velocity += newDirection * maxAceleration * Time.deltaTime;

14   if (velocity.magnitude > maxVelocity)
15     velocity = velocity.normalized * maxVelocity;

17   transform.position += velocity * Time.deltaTime;
18 }
```

- Asigna en el inspector la referencia del personaje Wander como `Target`.
- En tiempo de ejecución modifica `maxXXX`. Justifica el comportamiento.
- Añade `OnDrawGizmos()` y prográmalo para que dibuje una línea en la dirección de la velocidad actual y otra en la dirección de la orientación. ¿Son las mismas? Ponte de pie y simula ese movimiento.

Cámaras

Dos cámaras en nuestra escena

Esta es una tarea que resulta muy simple y se deja como ejercicio:

- Añade una cámara al perseguidor y que se vea en la esquina superior derecha de la ventana del juego.
- Pon la cámara principal de la escena para que persiga al paseante.

Si tiene alguna duda consulte la sesión 2.