Máster Universitario en Ingeniería Informática

ENTORNOS VIRTUALES

Ejercicio téorico-práctico 3:
Estudio de un sistema ECS en Godot

UNIVERSIDAD DE GRANADA

Carlos Morales Aguilera
carlos7ma@correo.ugr.es

Curso Académico 2020-2021

Índice

1. Escenario **2**

2. Componentes **2**

3. Entidades **2**

4. Sistemas **2**

5. Métodos **2**

6. Pseudo Algoritmo **3**

7. Singleton **3**

1. Escenario

El escenario propuesto es un deportista que practica deporte al aire libre, aunque se detallará con mayor precisión en los siguientes apartados, la idea es representarlo mediante un objeto que se puede mover por el escenario y a su vez se ve afectado por los efectos del deporte (adelgazar).

2. Componentes

Mis tres componentes son: **Forma**, **Posición** y **Rotación**.

Estos tres componentes se representan por vectores 3d donde la forma representa los valores X, Y y Z de escalado del cubo, la posición la ubicación en las 3 dimensiones y la rotación como un valor X ya que solo se pretende que rote en torno a este eje.

3. Entidades

Para las entidades, he opetado por utilizar una representación muy simple, siendo la de números enteros (**int**), ya que con un número identificativo es más que suficiente para poder representar un deportista de forma inequívoca.

Mis entidades se representan computacionalmente como: **int**.

4. Sistemas

Los sistemas propuestos para el ejercicio son: **Correr** y **Adelgazar**.

Mis sistemas se representan computacionalmente como: **func** en Godot, las cuales poseen toda la funcionalidad para definir las diferentes acciones.

5. Métodos

Los métodos y su justificación son las siguientes:

- **Correr**: Al correr, lo ideal es que se mueva el corredor, simplemente desplazándose hacia delante en el eje que se estime oportuno, por ejemplo el x, a su vez, con el fin de no perderse en el mapa, se introduce un cierto grado de giro en el eje x para que de vueltas circulares. (Afecta por lo tanto a los componentes de Posición y Rotación).
- Sistema de **Adelgazar**: Se entiende que cuando el deportista se esta moviendo (aunque exagerando para el escenario evidentemente) este adelgaza de forma leve, por lo que este método se encarga de modificar la forma del deportista de forma que se reduzca en los ejes necesarios.

6. Pseudo Algoritmo

Nota: He intentado parecerme lo máximo posible a la sintaxis de godot en el pseudo algoritmo, pero ante la falta de documentación de ciertos elementos, el código seguramente no sea ejecutable ni reproducible, por lo que es más bien orientativo.

Tal y como se comenta, se asumen los aspectos de creación, eliminación y asignación de componentes a entidades.

El pseudo algoritmo sería el siguiente:

```
# Componente de forma del deportista
class Forma:
    var figura: Vector3()

# Componente de posicion del deportista
class Posicion:
    var pos: Vector3()

# Componente de rotacion del deportista
class Rotacion:
    var grado: float

# Sistema de correr
func correr(delta):
    for ent in entidades:
        var posicion = ent.get(Posicion)
        # Se mueve mas rapido de lo que adelgaza como es logico
        posicion.pos.x += 10*delta

        var rotacion = ent.get(Rotacion)
        # Gira relativamente rapido pero sin llegar a producir circulos constantes
        rotacion.grado += 7*delta

# Sistema de broncearse
func adelgazar(delta):
    for ent in entidades:
        var forma = ent.get(Forma)
        # No sobrepasar un limite de adelgazar
        if forma.figura.x > 0.5:
            forma.figura.x -= delta
        if forma.figura.z > 0.5:
            forma.figura.z -= delta
```

7. Singleton

Si observamos la Documentación Oficial de Godot, podemos observar que un objeto **Singleton** posee las siguientes características:

- Siempre están cargados, no importa qué escena esté en ejecución.

- Puedes almacenar variables globales como la información del jugador.
- Puedes manejar el cambio de escenas y las transiciones entre escenas.
- Actúa como un singleton, ya que GDScript no soporta variables globales por diseño.

Sabiendo esto, necesitaremos la funcionalidad **Autoload** de Godot para cargar nuestro nodo como un objeto **Singleton**, con el fin de poder *autocargar* la funcionalidad implementada cada vez que lo deseemos.

Para ello, una vez utilizada la herramienta tal y como se explica en la documentación anteriormente mencionada, podríamos llamar al sistema de adelgazar de una forma tan sencilla como:

```
# Suponemos la llamada desde otro script
func _process(delta):
    # Se detecta automaticamente
    correr(delta)
```