UNIVERSIDAD DE GRANADA

Ejercicio teórico-práctico 3. Estudio de un sistema ECS en Godot Engine.

Entornos Virtuales

Máster Profesional en Ingeniería Informática

Curso académico 2022/2023

Autor

José Alberto Gómez García

modej@correo.ugr.es

26514779B

El escenario que se plantea es un juego que podría ser de tipo “shooter”. En el que el jugador controla un personaje que tiene una determinada cantidad de vida, armas y munición, y propiedades especiales.

Mis cuatro componentes son *HEALTH, STAMINA, POSITION* y *ABLE_TO_CROUCH,* los cuales representan la salud de un personaje, la resistencia que posee, la posición dentro del mundo, y si el personaje puede agacharse o no.

La salud se podría representar computacionalmente como un valor entero, o si se quiere complicar un poco el sistema, podría ser un vector de 3 componentes (o más), entre las que tendríamos un entero para la salud actual, otro entero para la salud máxima (podríamos tener objetos que la modificaran), un flotante para la tasa de regeneración de salud por segundo, etc. Con la resistencia usaríamos el mismo esquema que con la salud. Para la posición del jugador utilizaríamos un vector de 3 valores flotantes que actúen como coordenadas, y para comprobar si se puede o no agachar bastaría con emplear un valor booleano.

Mis entidades son dos. El personaje principal que controla el jugador estaría representado por una escena en la que tendríamos un KineticBody, con su correspondiente forma de colisión, un rotation helper, una cámara y el modelo del arma, esencialmente el esquema descrito en la práctica 5 y un arma. Los NPC, enemigos u objetivos de práctica serían objetos RigidBody con una forma de colisión y un modelo del arma (en caso de llevarla). Todos los objetos tendrían asignado un identificador, por ejemplo, de texto, en función del tipo de elemento. El personaje podría tener su UID de usuario (y el texto sería algo así como player_84934893), y cada NPC tendría otro ID en función del nivel, orden de generación, o demás.

Los sistemas que se proponen son *recuperar estadísticas básicas* y *agacharse.* Estos sistemas serán representados computacionalmente como funciones (*func* en Godot), y serán estas funciones quieren implementarán la lógica necesaria para llevar a cabo dichas acciones.

- El método de recuperar estadísticas básicas comprobaría si el personaje está en combate o no. De no estar en combate, irá actualizando la salud del personaje. La resistencia puede recuperarla siempre que no esté corriendo, volando o realizando alguna acción de este estilo, ya que de estar llevando a cabo estas acciones la estará consumiendo.
- El método de agacharse se limitará a modificar el modelo del personaje (dimensiones menores en altura, y puede que algo mayores en anchura) y establecer una velocidad menor que la velocidad que se tiene al andar. Nótese que es una simplificación pues en un juego real deberíamos tener en cuenta animaciones, ruido producido al andar, etc.

Por una parte, tendríamos los componentes, cuyo código podríamos tener especificado en scripts de la siguiente manera:

```
class HealthStats:
    var max_health = 100
    var current_health = 100
    var regen_per_sec = 5.0

class StaminaStats:
    var max_stamina = 100
    var current_stamina = 100
    var regen_per_sec = 10.0
```

```
class Position:
    var pos: Vector3(x,y,z)

class Crouch:
    var ABLE_TO_CROUCH = true
```

En los nodos del personaje y NPCs tendríamos la lógica necesaria para añadir y eliminar los grupos a los que pertenece, también se mandaría añadir la referencia al script del autoloader que mencionaré a continuación (con una señal podríamos notificar que les cree los valores iniciales asociados a las distintas propiedades). Tendría los grupos básicos por defecto e iría añadiendo y perdiendo "propiedades" en función a determinados eventos. Un ejemplo para el caso del personaje principal, asumiendo que las teclas son un placeholder, podría ser el siguiente:

```
func _ready():
    add_to_group("HEALTH")
    add_to_group("STAMINA")
    add_to_group("POSITION")
    ScriptMaestro.entidades.add(self)

func _input(event):
     if event.is_action_pressed("random_key"):
        add_to_group("CROUCH")
      if event.is_action_pressed("other_random_key"):
         remove_from_group("CROUCH")
```

Como parte del autoloader tendríamos los scripts que incorporan las mecánicas en sí mismas del juego (podríamos tener una por fichero, o agrupadas por temática), y otro script para gestionar la aplicación de dichas mecánicas, el cual tendría que contener una lista de identificadores de los objetos que pueden tener mecánicas (personaje principal y NPCs).

Un script que incorpore la función de agacharse podría ser algo así:

```
func crouch(nodo_a_aplicar):
    var nodo = get_node(nodo_a_aplicar)
    nodo.scale.y = clamp(nodo.scale.y - 0.4, 0.5, 1.0)
    nodo.scale.z = clamp(nodo.scale.z + 0.2, 0.8, 1.5)
```

El script de aplicación de dichas mecánicas (en el autoloader) sería algo similar a lo siguiente:

```
1. var entidades: []
2. func _process(delta):
3.     for e in entidades:
4.         if e.is_in_group("CROUCH"):
5.             MecanicaAgachar.crouch(e)
6.         # Por ejemplo, habría una lógica más compleja.
7.         if e.is_in_group("HEALTH"):
8.             MecanicasSalud.health_regen(e)
9.         if e.is_in_group("STAMINA"):
10.             MecanicasStamina.stamina_regen(e)
```