UNIVERSIDAD DE GRANADA

Ejercicio teórico-práctico 4. Estudio del funcionamiento de una cámara FPS y salto.

Entornos Virtuales

Máster Profesional en Ingeniería Informática

Curso académico 2022/2023

Autor

José Alberto Gómez García

modej@correo.ugr.es

26514779B

En este ejercicio se abordará de forma teórica como implementar una cámara en primera persona que permita al personaje saltar.

En primer lugar, abordaremos el giro de la cámara, lo cual ya se implementó en la práctica 4. Recordemos que necesitamos un nodo como padre de la cámara, de manera que las rotaciones en un eje (X) se hagan en el nodo de la cámara y las rotaciones en el otro eje se hagan en el nodo padre (Y). Para el código que se adjunta, asumiremos que estamos en el nodo de la cámara.

Para ello, en el método *_input* comprobaremos si el evento es de movimiento de ratón y capturaremos las coordenadas del mismo. La coordenada X del ratón se corresponde con la coordenada Y de la cámara y viceversa. Para evitar giros extraños que puedan dar la vuelta al personaje, restringimos el ángulo de giro en X entre dos valores. Para manejar la velocidad, utilizamos un valor de "sensibilidad" en el rango [0,1] .

```
1. var mouseSensitivity = 0.2
2. var min_angle = 90
3. var max_angle = 90
4. onready var nodo_padre = get_parent()
5.
6. func _input(event):
7.   if event is InputEventMouseMotion:
8.       look_rot.y -= (event.relative.x * sensitivity)
9.       look_rot.x -= (event.relative.y * sensitivity)
10.      look_rot.x = clamp(look_rot.x, min_angle, max_angle)
11.
12.      nodo_padre.rotation_degrees.x = look_rot.x
13.      self.rotation_degrees.y = look_rot.y
```

Para calcular la translación del personaje al pulsar la tecla de saltar deberemos tener en cuenta la ecuación de tiro parabólico. Esta ecuación depende de la posición inicial, la velocidad que tenga el objeto, el tiempo transcurrido, el ángulo con el que lanza el objeto y la fuerza de la gravedad.

La fuerza gravedad, el ángulo y velocidad con la que va a saltar son valores constantes que podemos declarar en variables. La velocidad podría ser variable si se pudiera correr o ir más despacio, en cuyo caso tendríamos que, o bien tener varias variables para cada situación y usar una u otra en función de la situación, o bien conseguir la velocidad actual con alguna función de Godot (más recomendable pues al iniciar/parar de esprintar debería acelerarse/frenarse progresivamente). Para simplificar, vamos a asumir que la velocidad es constante. La posición inicial la obtendremos con funciones y variables de Godot Engine, y el tiempo transcurrido saltando lo iremos sumando a partir del "delta" que nos da la función de físicas.

```
1. var gravity = 9.8
2. var jumpSpeed = 10.0
3. var jumpAngle = deg2rad(45.0)
```

```
4. var initialPosition = Vector3()
5. var elapsedTime = 0.0
```

Así pues, dentro de la función de físicas comprobaremos si se ha pulsado la tecla correspondiente a saltar y si estamos en el suelo, para iniciar el salto. En este caso llamaremos a una función encargada de "inicializar el salto". Después, comprobaremos si llevamos más de 0.0 segundos saltando, en cuyo caso llamaremos a la función de "actualizar el salto".

```
1. func _physics_process(delta):
2.     if Input.is_action_pressed("jump") and is_on_floor():
3.         start_jump()
4.
5.     if elapsedTime > 0.0:
6.         elapsedTime += delta
7.         update_jump()
```

La función de "iniciar el salto" actualizaría la posición inicial del sato (que aparentemente se puede obtener de la variable *translation* en Godot Engine 3, y *position* en Godot Engine 4). También sumará algo pequeño al tiempo que estamos saltando para indicar que ha comenzado.

```
1. func start_jump():
2.     initialPosition = nodo_padre.translation
3.     elapsedTime = 0.0000001
```

La función de "actualizar el salto" calculará las nuevas posiciones aplicando las ecuaciones de tiro parabólico, creará el vector de movimiento y se lo sumará a la posición actual. También comprobará si tocamos suelo (para parar el "temporizador" y así decir que no estamos saltando). Nótese que el código es simple, en el sentido de que no se aplica interpolación ni ninguna técnica para suavizar el cambio de posición, sería conveniente tenerlo en cuenta.

```
1. func update_jump():
2.     # Ecuación tiro parabólico
3.     var posX = initialPosition.x + (jumpSpeed *
   cos(jumpAngle)) * elapsedTime
4.     var posY = initialPosition.y + (jumpSpeed *
   sin(jumpAngle)) * elapsedTime - (0.5 * gravity * elapsedTime *
   elapsedTime)
5.
6.     var motion = Vector3(posX, posY, initialPosition.z) -
   initialPosition
7.
8.     nodo_padre.translation += motion
9.     if posY <= initialPosition.y and is_on_floor():
10.             elapsedTime = 0.0
```

Por último, faltaría procesar las teclas WASD dentro de la función de físicas. Esto lo podríamos hacer con el código de la práctica 3 o alguno relativamente similar.

```
1.   if Input.is_action_pressed("fpc_arriba"):
2.         nodo_padre.translate_object_local(Vector3(0,0,-delta))
3.   if Input.is_action_pressed("fpc_abajo"):
4.         nodo_padre.translate_object_local(Vector3(0,0,-delta))
5.   if Input.is_action_pressed("fpc_izq"):
6.         nodo_padre.translate_object_local(Vector3(-delta,0,0))
7.   if Input.is_action_pressed("fpc_der"):
8.         nodo_padre.translate_object_local(Vector3(-delta,0,0))
```

Téngase en cuenta que los ejemplos de código mostrados podrían no ser completamente funcionales, simplemente se adjuntan para aclarar cómo se debería proceder si tuviéramos que implementarlo realmente.