Práctica 5: Interacción con ratón y selección de objetos

UNIVERSIDAD DE GRANADA

Autor: Lucas Hidalgo Herrera

Grado: Doble Grado en Ingeniería Informática y Matemáticas

Asignatura: Informática Gráfica

Fecha: 5 de enero de 2026

Índice General

1 Interacción con el modelo jerárquico **2**
1.1 Funciones útiles implementadas 2
1.2 Interacción de las teclas 3
1.3 Interacción con el ratón 4

1 Interacción con el modelo jerárquico

Una vez hemos realizado toda la práctica solo nos queda por realizar el último ejercicio, aquel en el que debemos añadir una nueva funcionalidad a los botones del ratón. Además, como se nos comenta en el guion de prácticas vamos a implementar una serie de eventos con las flechas del teclado de nuestro ordenador; en mi caso, usaremos la tecla H (*KEY_H*) y la tecla L (*KEY_L*).

Por último, antes de comenzar el desarrollo del código implementado, me gustaría aclarar que aparecen comentarios de *DEBUG* en el mismo debido a que, bajo mi opinión, es mucho más descriptivo el proceso que se ha seguido.

1.1 Funciones útiles implementadas

Para facilitar la lectura del código y evitar tener que copiar código múltiples veces he realizado una serie de funciones que se verán reflejadas en el código 1. Las funciones creadas son:

- *select_figure(figure)*: dado un nodo del árbol de escena, guarda a ese nodo como seleccionado en una variable global.
- *select_figure_by_name(name)*: dado el nombre de una malla del árbol de escena, busca el nodo del árbol de escena con ese nombre y usa la función anterior para seleccionarlo.
- *select_figure_at_mouse(from,to)*: dada la proyección actual del origen del nodo *RayCast3D* y el punto destino del ratón sobre la escena, en caso de colisionar con alguna malla de la escena, la selecciona usando la primera de las funciones.

Listing 1: Script de Camera3D

```
@onready var ray = \$RayCast3D # nodo 'RayCast3D' hijo de la camara

## ------------------------------------------------ ##
## Funcion que selecciona el nodo pasada como argumento
##  figure : figura pasada como argumento

func select_figure(figure):
    # Seleccionar nuevo
    selected_figure = figure
    if selected_figure:
        print("Seleccionado: ", selected_figure.name)

## ------------------------------------------------ ##
## Funcion que selecciona mesh instance cuyo nombre coincide con el
    pasado como argumento
##  name : nombre de la mesh instance
func select_figure_by_name(name):
    var root = get_tree().current_scene
    var node = root.find_child(name, true, false)
    if node:
        select_figure(node)
    else:
```

```
        print("ERROR: No se encontro figura: ", name)

## ------------------------------------------------ ##
## Funcion que selecciona el mesh instance clicada con el raton
##  name : nombre de la mesh instance
func select_figure_at_mouse(from, to):
    print("=== RAYCAST DEBUG ===")
    print("From: ", from)
    print("To: ", to)

    ray.global_position = from
    ray.look_at(to)
    ray.target_position = Vector3(0, 0, -1000)
    ray.force_raycast_update()

    print("Ray enabled: ", ray.enabled)
    print("Ray colliding: ", ray.is_colliding())

    if ray.is_colliding():
        var figura = ray.get_collider().get_parent()
        select_figure(figura)
    else:
        print("NO HAY COLISION")
```

1.2 Interacción de las teclas

Tal y como hemos comentado en la sección 1, vamos a implementar una funcionalidad por cada una de las teclas comentadas. La estructura del código es conocida, pues ya se usó en prácticas anteriores para mostrar aristas.

El código se encuentra en la figura 2. En este código, se refleja el trato de un evento de teclado que se haya producido en el desarrollo de la simulación; en nuestro caso, actualmente nos interesa filtrar los eventos que hayan ocurrido por alguna de las teclas que he comentado anteriormente. De hecho, sólo vamos a tomar el evento en el que la tecla deje de estar pulsada; de esta manera, conseguimos la sensación de un selector dentro de la simulación.

Listing 2: Gestión de eventos de teclado

```
func _unhandled_key_input(key_event):
    # Cambiar parte con flechas
    if key_event.keycode == KEY_H and not key_event.pressed:
        current_figure_index = (current_figure_index + 1) % figures
           .size()
        select_figure_by_name(figures[current_figure_index])
        print("Parte: ", figures[current_figure_index])

    if key_event.keycode == KEY_L and not key_event.pressed:
        current_figure_index = (current_figure_index - 1 + figures.
           size()) % figures.size()
        select_figure_by_name(figures[current_figure_index])
        print("Parte: ", figures[current_figure_index])
```

1.3 Interacción con el ratón

Para finalizar la práctica, sólo queda comentar la implementación seguida en la interacción con el ratón, su código aparece en la figura 3. En él, se tratan dos eventos del ratón, cuando se realiza un "clic" sobre uno de los objetos de la escena y cuando, una vez realizado el "clic" se mueve el ratón:

- **Selector de figuras**: la implementación de este evento se ve en la primera de las condiciones, donde usando la función *select_figure_at_mouse* de la figura 1 conseguimos nuestro objetivo. Cabe recalcar que el evento que hemos filtrado es producido por pulsar un botón del ratón y que este botón sea el botón izquierdo.

- **Movimiento tras selección**: una vez hayamos seleccionado un objeto con cualquiera de las opciones posibles, moviendo el ratón conseguimos que se mueva la figura a lo largo del plano $Y = 0$. El movimiento se corresponde con el movimiento del ratón en la pantalla, es decir, intenta simular que la propia pantalla es ese plano usando como componente x del movimiento la componente x del ratón en la pantalla y como componente z la componente y del ratón en la pantalla. Ahora bien, para que no sea muy brusco este movimiento, se multiplica por un factor de 0.01. Por último, para darle algo de sentido al movimiento con la escena, he añadido un limitador de posición, es decir, sólo se puede mover la figura seleccionada en un entorno del centro según la norma del máximo; esto se realiza con la función *clamp()*.

Listing 3: Gestión de eventos de ratón

```
func _input( event ):
    var mouse_pos = get_viewport().get_mouse_position()
    var from = project_ray_origin( mouse_pos )
    var to = from + project_ray_normal( mouse_pos ) * 1000.0

    if event is InputEventMouseButton and event.pressed \
            and event.button_index == MOUSE_BUTTON_LEFT:
        select_figure_at_mouse(from,to)

    if event is InputEventMouseMotion and selected_figure:
        var move = Vector3((event.relative.x * 0.01), 0, (event
            .relative.y * 0.01))
        selected_figure.position += move
        selected_figure.position.x = clamp(selected_figure.
            position.x, -10.0, 10.0)
        selected_figure.position.z = clamp(selected_figure.
            position.z, -10.0, 10.0)
```