Universidad de Granada

Doble Grado en Ingeniería Informática y Matemáticas

Informática Gráfica

Informe Práctica 5

Autor:
Jesús Muñoz Velasco

Curso 2025-2026

1. Interacción con el modelo

En primer lugar se ha añadido a cada pieza móvil del modelo los siguientes elementos:

1. Variable `seleccionada`. Esta variable booleana indica si el modelo está seleccionado o no. Inicialmente tendrá el valor `false`.

2. Variable `seleccionar`. Esta cadena (`String`) estará definida para cada pieza móvil con un nombre diferente y servirá para identificarla con una acción definida en el mapa de entrada que se asociará a una tecla (para seleccionar la pieza con el teclado). Se ha seguido la siguiente configuración:

Pieza	Acción mover	Tecla mover	Acción seleccionar	Tecla seleccionar
Brazo_derecho	`mover_brazos`	1	`seleccionar_brazos`	A
Brazo_izquierdo	`mover_brazos`	1	`seleccionar_brazos`	A
Cabeza	`asentir_cabeza`	2	`seleccionar_cabeza`	S
Ojos	`mover_ojos`	3	`seleccionar_ojos`	D
boca	`escalar_boca`	4	`seleccionar_boca`	F
ojo_derecho	`guinar_ojo`	5	`seleccionar_ojo`	G

3. Código de activación de movimiento. Se ha modificado el esquema en cada pieza siguiendo el siguiente modelo:

```
func _process(delta):
        if Input.is_action_just_pressed(activar):
                activa = !activa
        if Input.is_action_just_pressed(seleccionar):
                select()

        if activa and seleccionada:
                ## Activar movimiento
```

4. Función de selección. Además se ha añadido a cada pieza la siguiente función que cambia el material para identificar visualmente la pieza seleccionada:

```
func select() -> void:
        seleccionada = !seleccionada

        if(seleccionada==true):
                material_override = Utilidades.selectedMaterial()
        else:
                material_override=null
```

Donde en `utilidades.gd` se ha añadido el siguiente código:

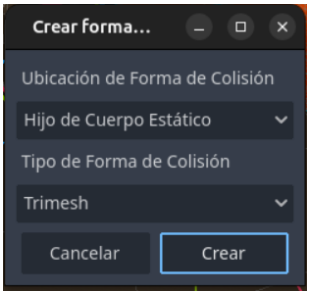
```
var selected_material: StandardMaterial3D = StandardMaterial3D.new()

func _ready() -> void:
        selected_material.albedo_color = Color.ORANGE_RED
        selected_material.cull_mode = BaseMaterial3D.CULL_DISABLED
        selected_material.metallic = 0.0
        selected_material.roughness = 0.5
        selected_material.specular = 0.5
        selected_material.emission_enabled = true
        selected_material.emission = Color(1, 0, 0)
        selected_material.emission_energy = 2.0

func selectedMaterial() -> StandardMaterial3D :
        return selected_material
```

que únicamente establece un material para la selección con cierto brillo.

5. `StaticBody3D` y `CollisionShape3D`. Para cada una de las piezas móviles se ha establecido la siguiente jerarquía en el árbol:

```
<nombre_pieza>
 |-StaticBody3D
    |-CollisionShape3D
```

esto se ha hecho con la opción automática de Godot que es seleccionar el `MeshInstance3D` y en el editor 3D entrar al menú `Malla` y seleccionar la opción `Crear Forma de Colisión...`. En este menú establecemos los siguientes parámetros:



y automáticamente se generará la estructura necesaria para reconocer colisiones.

Una vez hecho todo esto en cada una de las piezas móviles que ya se han mencionado se ha modificado ligeramente el código de la cámara que se adjuntó en la práctica:

```
if event.pressed and (ray.visible): # solo reconoce el rayCast si es visible en
la escena
    ray.global_position = from
    ray.look_at(to)
    ray.target_position = Vector3(0, 0, -1000) # Apunta en su -Z local
    ray.force_raycast_update()
    if ray.is_colliding():
```

```
        var objeto = ray.get_collider()
        var padre : Node = objeto.get_parent()
        if padre and padre.visible:
            print("Seleccionado:", padre.name)
            # Mantengo la utilidad de clicar en el suelo cuando es
            if padre.name == "Suelo":
                Utilidades.crear_cubo_en(ray.get_collision_point())
                print("Creado cubo en:", ray.get_collision_point())
        seleccionar(padre)
    else:
        print("Ningun objeto seleccionado")
```

Además de añadir la siguiente función:

```
func seleccionar(selected : Node) -> void:
        if(selected!=null and selected.has_method("select")): selected.select()
```

De esta forma solo se pueden seleccionar los objetos que sean visibles y solo cuando el `RayCast` esté visible (para poder desactivarlo en los ejercicios adicionales). Finalmente para que los objetos invisibles no interfieran con el resto del proyecto se ha añadido el siguiente script a cada uno de los objetos colisionables (con `StaticBody3D` y `CollisionShape3D`):

```
extends MeshInstance3D

## Hago que no se lean las colisiones de objetos ocultos
var col

func _ready() -> void:
        col = $StaticBody3D/CollisionShape3D

func _process(_delta):
        col.disabled = not visible
```

y en el caso del modelo jerárquico se ha hecho un poco más complejo pero con la misma idea:

```
extends Node3D

## Hago que no se lean las colisiones de objetos ocultos
var hijos : Array = []

func _ready() -> void :
        hijos=[
                $Cabeza/Cara/boca/StaticBody3D/CollisionShape3D,
                $Cabeza/Cara/Ojos/ojo_derecho/StaticBody3D/CollisionShape3D,
                $Cabeza/Cara/Ojos/StaticBody3D/CollisionShape3D,
                $Cabeza/StaticBody3D/CollisionShape3D,
                $Cuerpo/Brazo_derecho/Brazo_derecho/CollisionShape3D,
                $Cuerpo/Brazo_izquierdo/Brazo_izquierdo/CollisionShape3D]

func _process(_delta):
        for hijo in hijos:
                hijo.disabled = not visible
```