Informe testing

Universidad Nacional de Colombia

Facultad de Ingeniería

Departamento de Industrial y Sistemas

Presentado por:

Nicolas Quezada Mora - *nquezada@unal.edu.co*

Sharick Yelixa Torres Monroy - *shtorres@unal.edu.co*

Laura Sofia Vargas Rodriguez - *lavargasro@unal.edu.co*

Jeronimo Bermudez Hernandez- jebermudez@unal.edu.co

Profesor:

Oscar Eduardo Alvarez Rodriguez - *oalvarezr@unal.edu.co*

2025

Introducción

En Scrum's Castle el jugador asume el papel de un gerente en un castillo medieval, quien debe encargarse de coordinar un equipo para completar determinadas tareas en el reino. Basándonos en la metodología Scrum, este debe asignar tareas teniendo en cuenta que cada miembro del equipo tiene habilidades diferentes y únicas que le permitirán avanzar más rápido o lento, según sea el caso.

El personaje puede moverse libremente por el castillo, interactuando con objetos, recursos y los demás miembros del equipo. La gestión eficiente y la toma de decisiones estratégicas determinarán el éxito del reino.

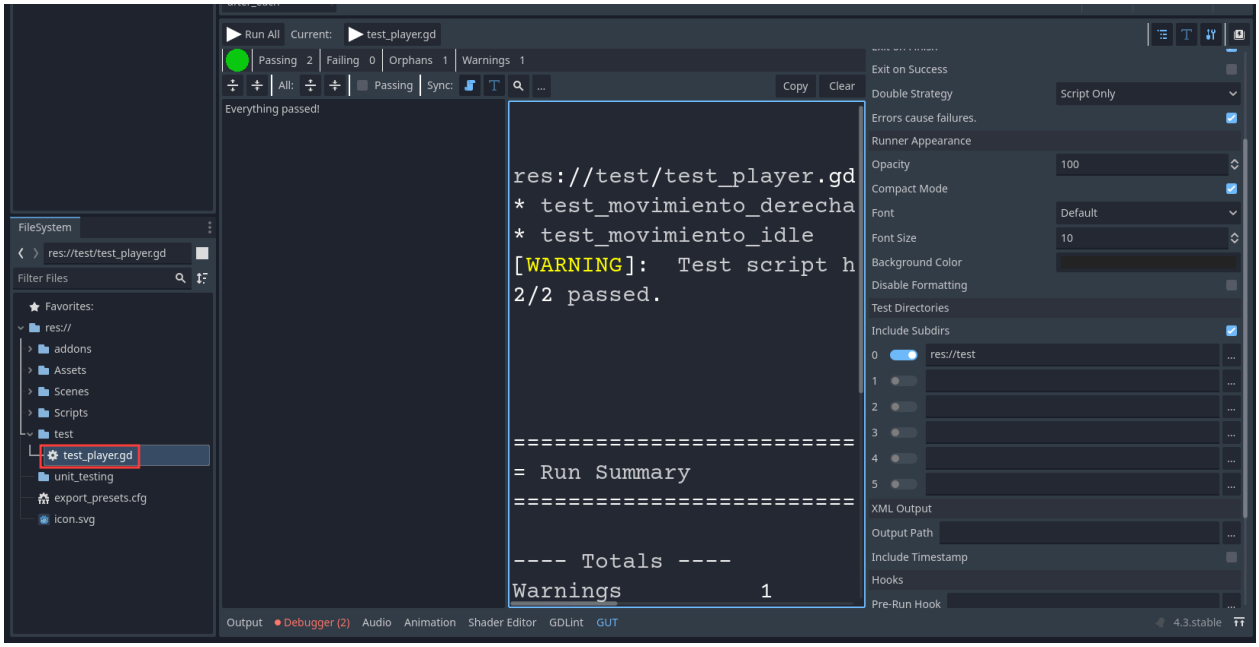
Resumen de los tests

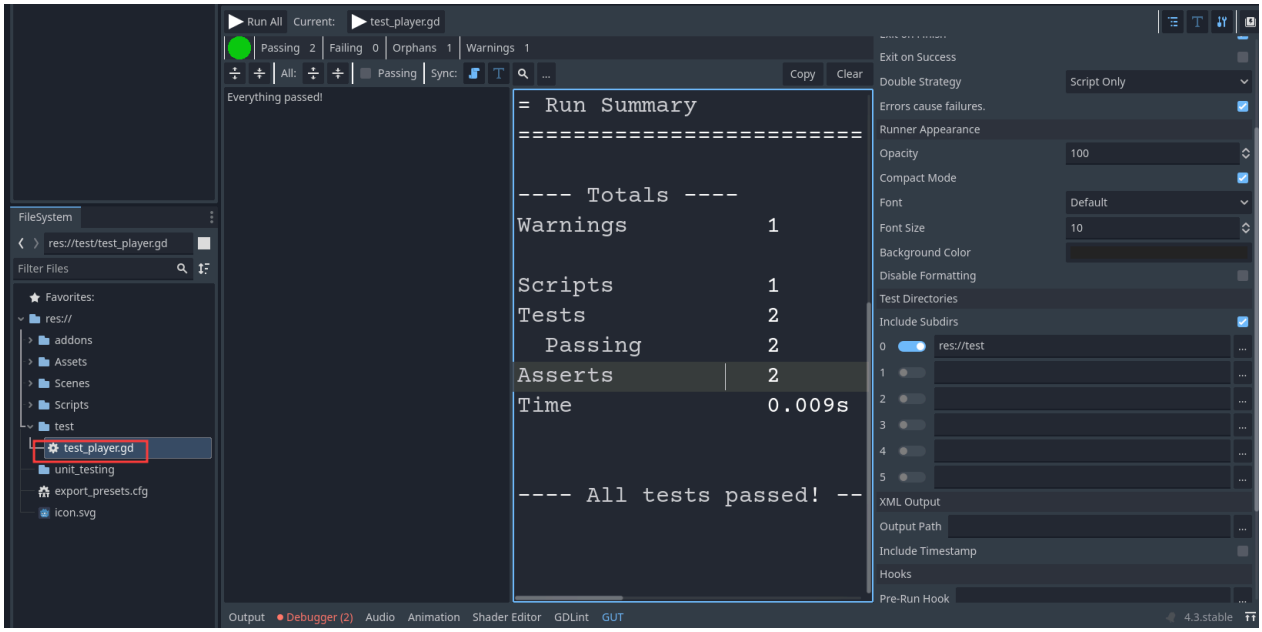
- Nombre del integrante: Nicolas Quezada Mora
- Tipo de prueba realizada: Prueba unitaria (movimiento horizontal y estado idle del jugador)
- Descripción breve del componente probado: Lógica de movimiento del jugador (velocity, animaciones, detección de input).
- Herramienta o framework usado: GUT (Godot Unit Testing Framework)
- Screenshot del código del test:

```gdscript
extends GutTest

var player  # Cambié "Player" a "player" para mantener la consistencia con el resto del script.

func before_each():
    # Instanciar el nodo jugador con el script a probar.
    player = preload("res://Scripts/player.gd").new()
    add_child(player)

    # Crear un nodo AnimatedSprite2D simulado, ya que el script lo requiere.
    var anim_sprite = AnimatedSprite2D.new()
    anim_sprite.name = "AnimatedSprite2D"
    player.add_child(anim_sprite)

    # Configurar animaciones dummy para evitar errores al llamar a play().
    var frames = SpriteFrames.new()
    frames.add_animation("Idle")
    frames.add_animation("Walk")
    anim_sprite.frames = frames

func after_each():
    player.queue_free()
```

```gdscript
func test_movimiento_derecha():
    # Simular input: presionar "ui_right"
    Input.action_press("ui_right")
    # Llamar al _physics_process con delta = 1/60
    player._physics_process(1/60)
    # Se espera que la dirección sea (1,0), por lo que la velocidad debe ser igual a move_speed en x.
    var expected_velocity = Vector2(player.move_speed, 0)
    assert_eq(player.velocity, expected_velocity, "La velocidad debe ser igual a move_speed en dirección derecha")
    # Liberar el input simulado
    Input.action_release("ui_right")

func test_movimiento_idle():
    # No se simula ningún input; se espera estado idle.
    player._physics_process(1/60)
    var expected_velocity = Vector2.ZERO
    assert_eq(player.velocity, expected_velocity, "La velocidad debe ser cero cuando no hay input")

```

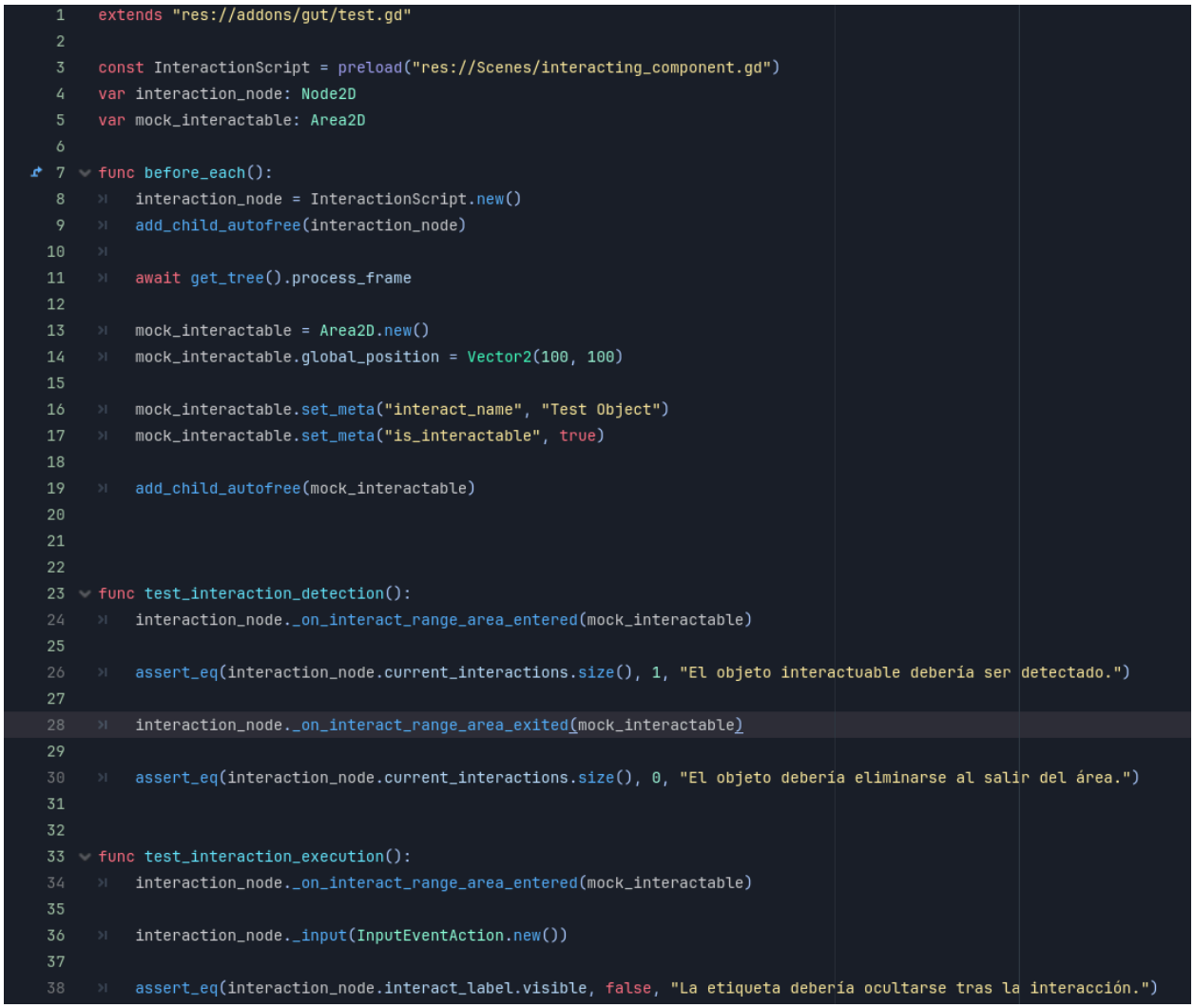
- Resultado de la ejecución:





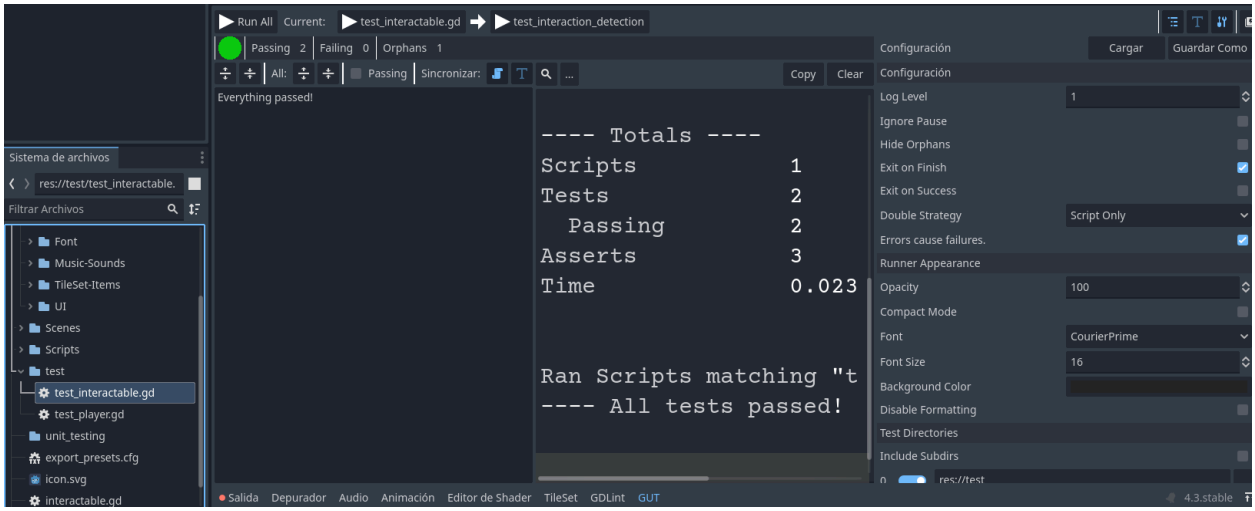

```
res://test/test_player.gd
* test_movimiento_derecha
* test_movimiento_idle
[WARNING]:  Test script has 3 unfreed children.  Increase log level for more details.
2/2 passed.

==============================================
= Run Summary
==============================================

---- Totals ----
Warnings          1

Scripts           1
Tests             2
  Passing         2
Asserts           2
Time              0.009s


---- All tests passed! ----
```

- Nombre del integrante: Jerónimo Bermúdez Hernández
- Tipo de prueba realizada: Prueba unitaria (módulo de interacción con objetos)
- Descripción breve del componente probado: Área de detección de interacción, trigger de eventos, etiqueta contextual.
- Herramienta o framework usado: GUT (Godot Unit Testing Framework)
- Screenshot del código del test:



```
extends "res://addons/gut/test.gd"

const InteractionScript = preload("res://Scenes/interacting_component.gd")
var interaction_node: Node2D
var mock_interactable: Area2D

func before_each():
    interaction_node = InteractionScript.new()
    add_child_autofree(interaction_node)

    await get_tree().process_frame

    mock_interactable = Area2D.new()
    mock_interactable.global_position = Vector2(100, 100)

    mock_interactable.set_meta("interact_name", "Test Object")
    mock_interactable.set_meta("is_interactable", true)

    add_child_autofree(mock_interactable)



func test_interaction_detection():
    interaction_node._on_interact_range_area_entered(mock_interactable)

    assert_eq(interaction_node.current_interactions.size(), 1, "El objeto interactuable debería ser detectado.")

    interaction_node._on_interact_range_area_exited(mock_interactable)

    assert_eq(interaction_node.current_interactions.size(), 0, "El objeto debería eliminarse al salir del área.")


func test_interaction_execution():
    interaction_node._on_interact_range_area_entered(mock_interactable)

    interaction_node._input(InputEventAction.new())

    assert_eq(interaction_node.interact_label.visible, false, "La etiqueta debería ocultarse tras la interacción.")
```

- Resultado de la ejecución:



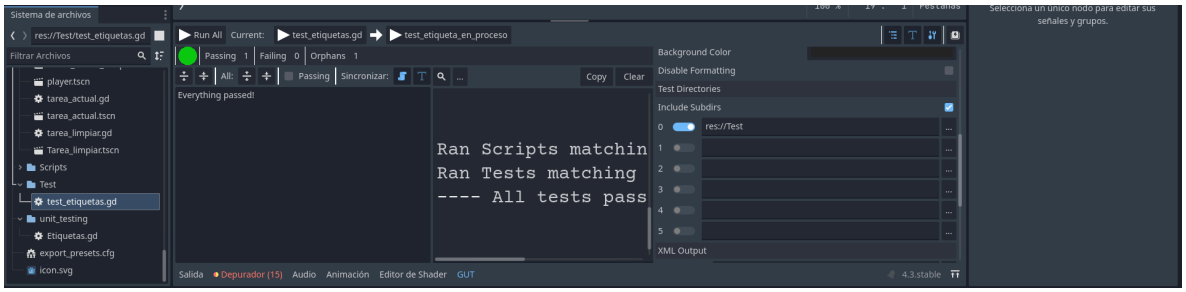
- Nombre del integrante: Laura Sofia Vargas Rodriguez
- Tipo de prueba realizada: Prueba unitaria (Verificación de cambio de estado en UI)
- Descripción breve del componente probado: Label el cual muestra el estado actual de una tarea, Método update_label_state() que actualiza texto y color según el estado
- Herramienta o framework usado: GUT (Godot Unit Testing Framework)
- Screenshot del código del test:

```gdscript
extends "res://addons/gut/test.gd"

var target_node: Node

func before_each() -> void:
    target_node = autofree(load("res://unit_testing/Etiquetas.gd").new())
    add_child(target_node)
    await get_tree().process_frame

func after_each() -> void:
    pass

func test_etiqueta_en_proceso() -> void:
    var estado_test = "En proceso"

    target_node.update_label_state(estado_test)
    await get_tree().process_frame


    var label = target_node.etiquetas_estado
    assert_eq(label.text, "En proceso", "Texto correcto en estado 'En proceso'")
```

```gdscript
    assert_eq(label.text, "En proceso", "Texto correcto en estado 'En proceso'")
    assert_eq(
        label.get_theme_color("font_color"),
        Color.GRAY,
        "Color debe ser gris para 'En proceso'"
    )

func test_etiqueta_completado() -> void:
    target_node.update_label_state("Completado")
    await get_tree().process_frame

    var label = target_node.etiquetas_estado
    assert_eq(label.text, "Completado", "Texto para estado completado")
    assert_eq(label.get_theme_color("font_color"), Color.GREEN)

func test_etiqueta_fallido() -> void:
    target_node.update_label_state("Fallido")
    await get_tree().process_frame

    var label = target_node.etiquetas_estado
    assert_eq(label.text, "Fallido", "Texto para estado fallido")
```

```gdscript
    var label = target_node.etiquetas_estado
    assert_eq(label.text, "Fallido", "Texto para estado fallido")
    assert_eq(label.get_theme_color("font_color"), Color.RED)

func test_estado_por_defecto() -> void:
    target_node.update_label_state("Desconocido")
    await get_tree().process_frame

    var label = target_node.etiquetas_estado
    assert_eq(label.get_theme_color("font_color"), Color.WHITE, "Color por defecto debe ser blanco")

```

- Resultado de la ejecución:




```
res://Test/test_etiquetas.gd
* test_etiqueta_en_proceso
1/1 passed.
==============================================
= Run Summary
==============================================
---- Totals ----
Scripts          1
Tests            1
```

```
 Passing        1
Asserts         2
Time          0.052s


Ran Scripts matching "test_etiquetas.gd"
Ran Tests matching "test_etiqueta_en_proceso"
---- All tests passed! ----
```

- Nombre del integrante: Sharick Yelixa Torres Monroy
- Tipo de prueba realizada: Prueba unitaria (Temporizador de tareas)
- Descripción breve del componente probado: Temporizador (actualización de fotogramas, detención automática, cuenta regresiva).
- Herramienta o framework usado: GUT (Godot Unit Testing Framework)
- Screenshot del código del test:

```gdscript
extends GutTest

var timer

func before_each():
    timer = preload("res://unit_testing/timer.gd").new()
    add_child(timer)

func after_each():
    timer.queue_free()

func test_timer_estado_inicial():
    assert_eq(timer.tiempo_restante, 0, "El tiempo restante debe ser 0 al inicio")
    assert_false(timer.timer_iniciado, "El temporizador no debería estar iniciado")

func test_iniciar_timer():
    timer.iniciar_timer(2)  # 2 minutos
    assert_true(timer.timer_iniciado, "El temporizador debe iniciarse")
    assert_eq(timer.tiempo_restante, 120, "Debe establecer el tiempo correctamente en segundos")

func test_timer_decrementa():
    timer.iniciar_timer(1)  # 1 minuto
    timer._process(1.5)  # Simulamos 1.5 segundos de delta
    assert_eq(timer.tiempo_restante, 59, "El temporizador debe haber disminuido en 2 segundos")
```

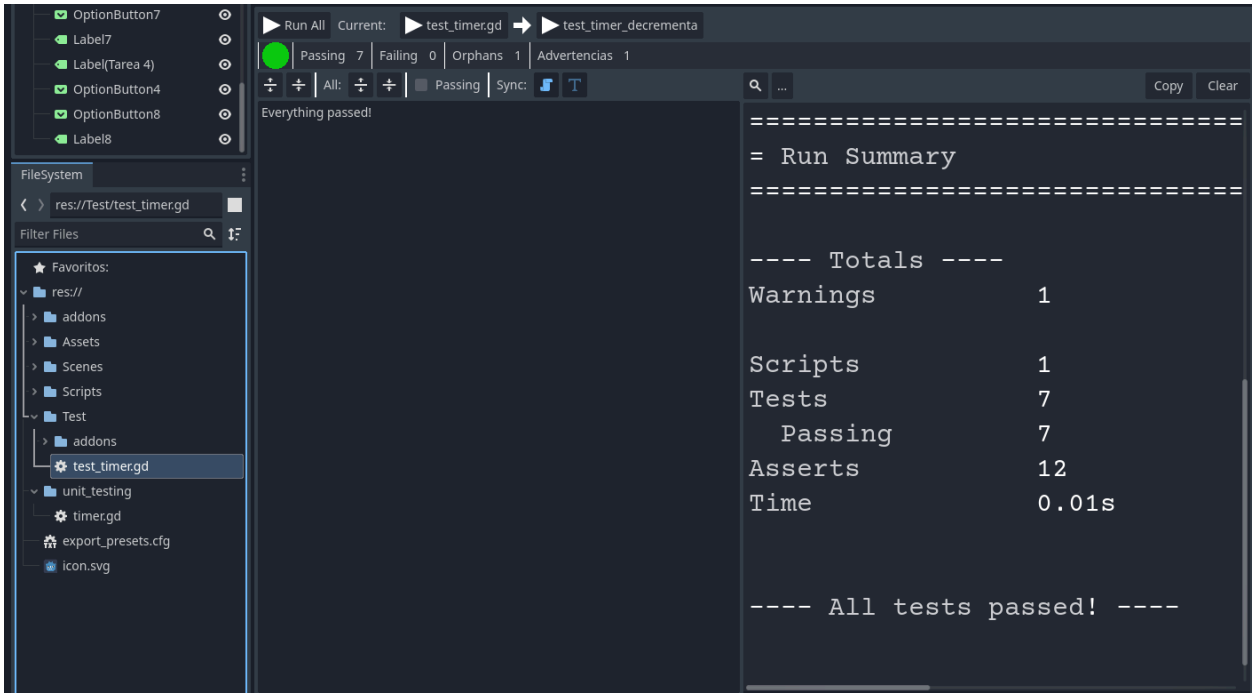
```gdscript
func test_timer_llega_a_cero():
    timer.iniciar_timer(1)
    for i in range(60):  # Simula 60 ticks de 1 segundo
        timer._process(1.0)
    timer._process(0.1) # Simula un último frame para asegurar la actualización
    assert_eq(timer.tiempo_restante, 0, "El tiempo debe ser 0 después de 60 segundos")
    assert_false(timer.timer_iniciado, "El temporizador debe detenerse al llegar a 0")

func test_detener_timer():
    timer.iniciar_timer(5)
    timer.detener_timer()
    assert_false(timer.timer_iniciado, "El temporizador debe estar detenido")
    assert_eq(timer.tiempo_restante, 300, "El tiempo restante no debe cambiar al detenerse")

func test_timer_acumulador_fraccional():
    timer.iniciar_timer(1)  # 1 minuto
    timer._process(0.5)  # Simulamos 0.5 segundos
    assert_eq(timer.tiempo_restante, 60, "El tiempo no debe decrementar con delta < 1.0")
    timer._process(0.5)  # Simulamos otros 0.5 segundos (total 1.0)
    assert_eq(timer.tiempo_restante, 59, "El tiempo debe decrementar en 1 segundo")
```

```gdscript
func test_timer_multiples_llamados_process():
    timer.iniciar_timer(1)  # 1 minuto
    timer._process(0.3)  # 0.3 segundos
    timer._process(0.7)  # 0.7 segundos (total 1.0)
    assert_eq(timer.tiempo_restante, 59, "El tiempo debe decrementar en 1 segundo")
```

- Resultado de la ejecución:

```
res://Test/test_timer.gd
* test_timer_initial_state
* test_iniciar_timer
* test_timer_decrementa
* test_timer_llega_a_cero
* test_detener_timer
* test_timer_acumulador_fraccional
* test_timer_multiple_process_calls
[WARNING]:  Test script has 8 unfreed children.  Increase log level for more details.
7/7 passed.


==============================================
= Run Summary
==============================================

---- Totals ----
Warnings          1

Scripts           1
```

```
Tests           7
  Passing       7
Asserts         12
Time            0.011s


---- All tests passed! ----
```



Lecciones aprendidas y dificultades

- Es muy complicado hacer pruebas sobre código, sobre todo sobre el código ajeno, se resalta la importancia del desarrollo orientado a pruebas.

- Fue un poco difícil generar un test en una nueva plataforma como lo es godot y tomó más tiempo del que se creía

- Dado que por el momento estamos trabajando sin control de versiones (por la plataforma que estamos empleando), fue bastante difícil trabajar sobre el código de mis compañeros. La realización de test no fue tan intuitiva como me esperaba, pero definitivamente es importante para verificar que todo funcione de manera adecuada.