Archivos (parte) de: C:\MiChinchonWeb

Archivo: C:/MiChinchonWeb/scripts/ui/ui_manager.gd

```
extends CanvasLayer
# ui_manager.gd
# Script para gestionar la interfaz de usuario del juego Chinchón

# Señales
signal game_paused         # Emitida cuando se pausa el juego
signal game_resumed        # Emitida cuando se reanuda el juego
signal new_game_requested  # Emitida cuando se solicita iniciar una nueva partida
signal main_menu_requested # Emitida cuando se solicita volver al menú principal
signal settings_changed    # Emitida cuando se cambian las configuraciones

# Referencias a nodos
@onready var pause_menu: Control = $PauseMenu
@onready var game_over_panel: Control = $GameOverPanel
@onready var turn_indicator: Label = $TurnIndicator
@onready var round_indicator: Label = $RoundIndicator
@onready var score_panel: Control = $ScorePanel
@onready var message_label: Label = $MessageLabel
@onready var message_timer: Timer = $MessageTimer
@onready var settings_panel: Control = $SettingsPanel
@onready var confirmation_dialog: ConfirmationDialog = $ConfirmationDialog
@onready var tooltip_panel: Control = $TooltipPanel
@onready var tooltip_label: Label = $TooltipPanel/TooltipLabel

# Variables
var current_player_name: String = ""
var is_paused: bool = false
var last_message: String = ""
var scores: Dictionary = {}
var player_names: Dictionary = {}

# Función de inicialización
func _ready() -> void:
	# Ocultar paneles que no deben estar visibles al inicio
	pause_menu.visible = false
	game_over_panel.visible = false
	settings_panel.visible = false
	tooltip_panel.visible = false
	message_label.visible = false
	
	# Conectar señales internas
	message_timer.connect("timeout", _on_message_timer_timeout)
	
	# Conectar botones del menú de pausa
	var resume_button = pause_menu.get_node("VBoxContainer/ResumeButton")
	var settings_button = pause_menu.get_node("VBoxContainer/SettingsButton")
	var main_menu_button = pause_menu.get_node("VBoxContainer/MainMenuButton")
	
	resume_button.connect("pressed", _on_resume_button_pressed)
	settings_button.connect("pressed", _on_settings_button_pressed)
	main_menu_button.connect("pressed", _on_main_menu_button_pressed)
	
	# Botón de pausa
	var pause_button = $PauseButton
	pause_button.connect("pressed", _on_pause_button_pressed)
	
	# Configurar el panel de juego terminado
	var play_again_button = game_over_panel.get_node("VBoxContainer/PlayAgainButton")
	var exit_button = game_over_panel.get_node("VBoxContainer/ExitButton")
	
	play_again_button.connect("pressed", _on_play_again_button_pressed)
	exit_button.connect("pressed", _on_exit_button_pressed)

# Mostrar mensajes temporales al jugador
func show_message(message: String, duration: float = 2.0) -> void:
	message_label.text = message
	message_label.visible = true
	last_message = message
	
	message_timer.wait_time = duration
	message_timer.start()

# Actualizar indicador de turno
```

```
func update_turn_indicator(player_id: int) -> void:
    var player_name = player_names.get(player_id, "Jugador " + str(player_id + 1))
    current_player_name = player_name
    
    if player_id == 0:
        # Es el turno del jugador actual
        turn_indicator.text = "¡Tu turno!"
    else:
        turn_indicator.text = "Turno de " + player_name
    
    # Destacar visualmente el cambio de turno
    var tween = create_tween()
    tween.tween_property(turn_indicator, "modulate", Color(1, 1, 0.3), 0.3)
    tween.tween_property(turn_indicator, "modulate", Color(1, 1, 1), 0.3)

# Actualizar indicador de ronda
func update_round_indicator(round_num: int, max_rounds: int) -> void:
    round_indicator.text = "Ronda " + str(round_num) + " de " + str(max_rounds)

# Actualizar panel de puntuaciones
func update_scores(player_scores: Dictionary) -> void:
    scores = player_scores
    
    # Limpiar puntuaciones anteriores
    var scores_container = score_panel.get_node("ScoresContainer")
    for child in scores_container.get_children():
        child.queue_free()
    
    # Añadir puntuaciones actualizadas
    for player_id in player_scores.keys():
        var player_name = player_names.get(player_id, "Jugador " + str(player_id + 1))
        var score = player_scores[player_id]
        
        var score_item = HBoxContainer.new()
        
        var name_label = Label.new()
        name_label.text = player_name + ":"
        name_label.size_flags_horizontal = Control.SIZE_EXPAND_FILL
        
        var score_label = Label.new()
        score_label.text = str(score)
        score_label.horizontal_alignment = HORIZONTAL_ALIGNMENT_RIGHT
        
        score_item.add_child(name_label)
        score_item.add_child(score_label)
        
        scores_container.add_child(score_item)
    
    # Hacer visible el panel de puntuaciones
    score_panel.visible = true

# Mostrar panel de fin de juego
func show_game_over(final_scores: Dictionary, winner_id: int = -1) -> void:
    var results_label = game_over_panel.get_node("VBoxContainer/ResultsLabel")
    var winner_label = game_over_panel.get_node("VBoxContainer/WinnerLabel")
    
    # Mostrar puntuaciones finales
    var results_text = "Puntuaciones finales:\n"
    
    # Ordenar jugadores por puntuación (menor a mayor, primero es el ganador)
    var sorted_players = []
    for player_id in final_scores:
        sorted_players.append({"id": player_id, "score": final_scores[player_id]})
    
    sorted_players.sort_custom(func(a, b): return a.score < b.score)
    
    # Generar texto de resultados
    for player in sorted_players:
        var player_name = player_names.get(player.id, "Jugador " + str(player.id + 1))
        results_text += player_name + ": " + str(player.score) + "\n"
    
    results_label.text = results_text
    
    # Mostrar ganador
    if winner_id >= 0:
        var winner_name = player_names.get(winner_id, "Jugador " + str(winner_id + 1))
        winner_label.text = "¡" + winner_name + " gana la partida!"
    else if sorted_players.size() > 0:
        var winner = sorted_players[0]
        var winner_name = player_names.get(winner.id, "Jugador " + str(winner.id + 1))
        winner_label.text = "¡" + winner_name + " gana la partida!"
```

```
    else:
        winner_label.text = "Partida finalizada"

    # Mostrar panel
    game_over_panel.visible = true

# Mostrar/ocultar menú de pausa
func toggle_pause_menu() -> void:
    is_paused = !is_paused

    pause_menu.visible = is_paused

    # Emitir señal correspondiente
    if is_paused:
        emit_signal("game_paused")
    else:
        emit_signal("game_resumed")

# Mostrar panel de configuraciones
func show_settings() -> void:
    settings_panel.visible = true

    # Si está en el menú de pausa, ocultar ese menú temporalmente
    if is_paused:
        pause_menu.visible = false

# Ocultar panel de configuraciones
func hide_settings() -> void:
    settings_panel.visible = false

    # Si estaba en el menú de pausa, volver a mostrarlo
    if is_paused:
        pause_menu.visible = true

    # Emitir señal de configuración cambiada
    emit_signal("settings_changed")

# Mostrar diálogo de confirmación
func show_confirmation(title: String, message: String, confirm_action: Callable) -> void:
    confirmation_dialog.title = title
    confirmation_dialog.dialog_text = message

    # Desconectar conexiones previas
    var confirm_button = confirmation_dialog.get_ok_button()
    if confirm_button.is_connected("pressed", Callable()):
        confirm_button.disconnect("pressed", Callable())

    # Conectar la nueva acción
    confirm_button.connect("pressed", confirm_action, CONNECT_ONE_SHOT)

    # Mostrar diálogo
    confirmation_dialog.popup_centered()

# Mostrar tooltip
func show_tooltip(text: String, position: Vector2) -> void:
    tooltip_label.text = text
    tooltip_panel.position = position
    tooltip_panel.visible = true

    # Asegurar que el tooltip no se salga de la pantalla
    var viewport_size = get_viewport().size
    var panel_size = tooltip_panel.size

    if tooltip_panel.position.x + panel_size.x > viewport_size.x:
        tooltip_panel.position.x = viewport_size.x - panel_size.x

    if tooltip_panel.position.y + panel_size.y > viewport_size.y:
        tooltip_panel.position.y = viewport_size.y - panel_size.y

# Ocultar tooltip
func hide_tooltip() -> void:
    tooltip_panel.visible = false

# Actualizar nombres de jugadores
func set_player_names(names: Dictionary) -> void:
    player_names = names

# Manejadores de eventos internos
func _on_message_timer_timeout() -> void:
    # Ocultar mensaje cuando termina el temporizador
    var tween = create_tween()
```

```
	tween.tween_property(message_label, "modulate", Color(1, 1, 1, 0), 0.5)
	tween.tween_callback(func():
		message_label.visible = false
		message_label.modulate = Color(1, 1, 1, 1)
	)

# Manejadores de eventos de botones
func _on_pause_button_pressed() -> void:
	toggle_pause_menu()

func _on_resume_button_pressed() -> void:
	toggle_pause_menu()  # Ocultar menú de pausa

func _on_settings_button_pressed() -> void:
	show_settings()

func _on_main_menu_button_pressed() -> void:
	show_confirmation(
		"Volver al menú principal",
		"¿Estás seguro de que quieres volver al menú principal?\nPerderás el progreso actual de la partida.",
		func(): emit_signal("main_menu_requested")
	)

func _on_play_again_button_pressed() -> void:
	emit_signal("new_game_requested")

func _on_exit_button_pressed() -> void:
	show_confirmation(
		"Volver al menú principal",
		"¿Estás seguro de que quieres volver al menú principal?",
		func(): emit_signal("main_menu_requested")
	)

# Método de procesamiento para teclas
func _input(event: InputEvent) -> void:
	if event.is_action_pressed("ui_cancel"):  # Tecla Esc
		toggle_pause_menu()
```

Archivo: C:/MiChinchonWeb/scripts/ui/ui_manager.gd.uid

```
uid://b0x2umkqdu3vp
```