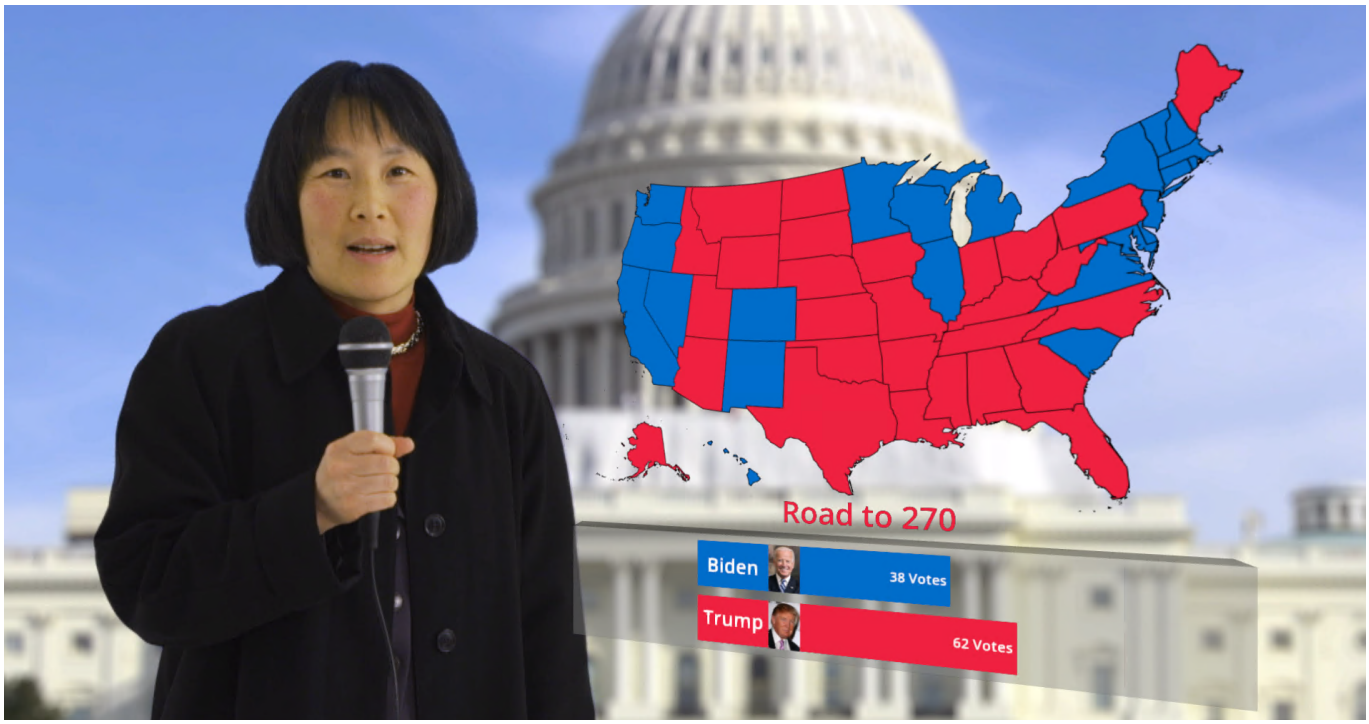
Elections: Map and Progress Bars

The description is relevant for software version 2.9.116.100 and later. The project has been prepared in Godot version 4.0.

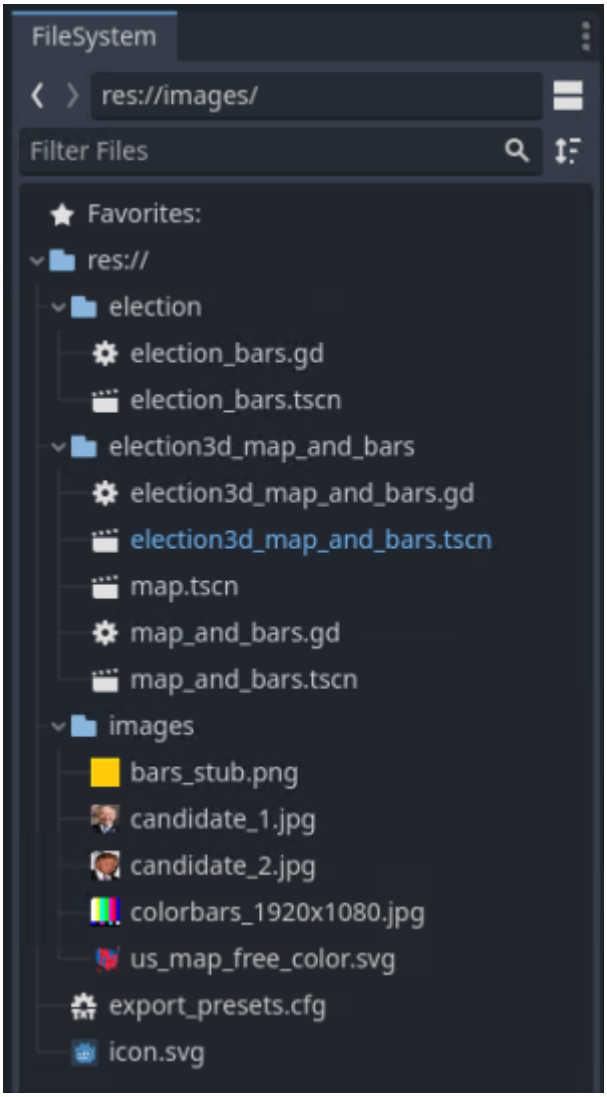
Description of the Godot demo project that implements output of information on the voting process during the correspondent's live feed.

GodotDemo_election3d_map_and_bars.zip contains sub folders:

- **Source** - source project files to open in Godot Editor.
- **Compiled** - folder with assembled project in zip format, ready to use with Skylark Godot Engine module.
- **Parts** - additional files required for project operation:
- **toMedia** - files from this folder must be placed in the server's first local media database that has the Godot module installed.
- **toUsrDir** - move files from this folder to `%APPDATA%\Godot\app_userdata\<Project_Name|Skylark_GodotModule_Name>`.



Project Files



The project `election3d_map_and_bars` has 4 scenes:

- **map.tscn** - 2D scene with the SVG map to display the leading candidate for each state.
- **election_bars.tscn** - 2D scene with progress bars visualizing the voting process.
- **map_and_bars.tscn** - 3D scene that combines scenes with the map and progress bars in 3D space, and implements the necessary object animations.
- **election3d_map_and_bars.tscn** - the main project scene displayed at startup. The scene outputs the signal from the correspondent and overlays the map_and_bars.tscn scene.

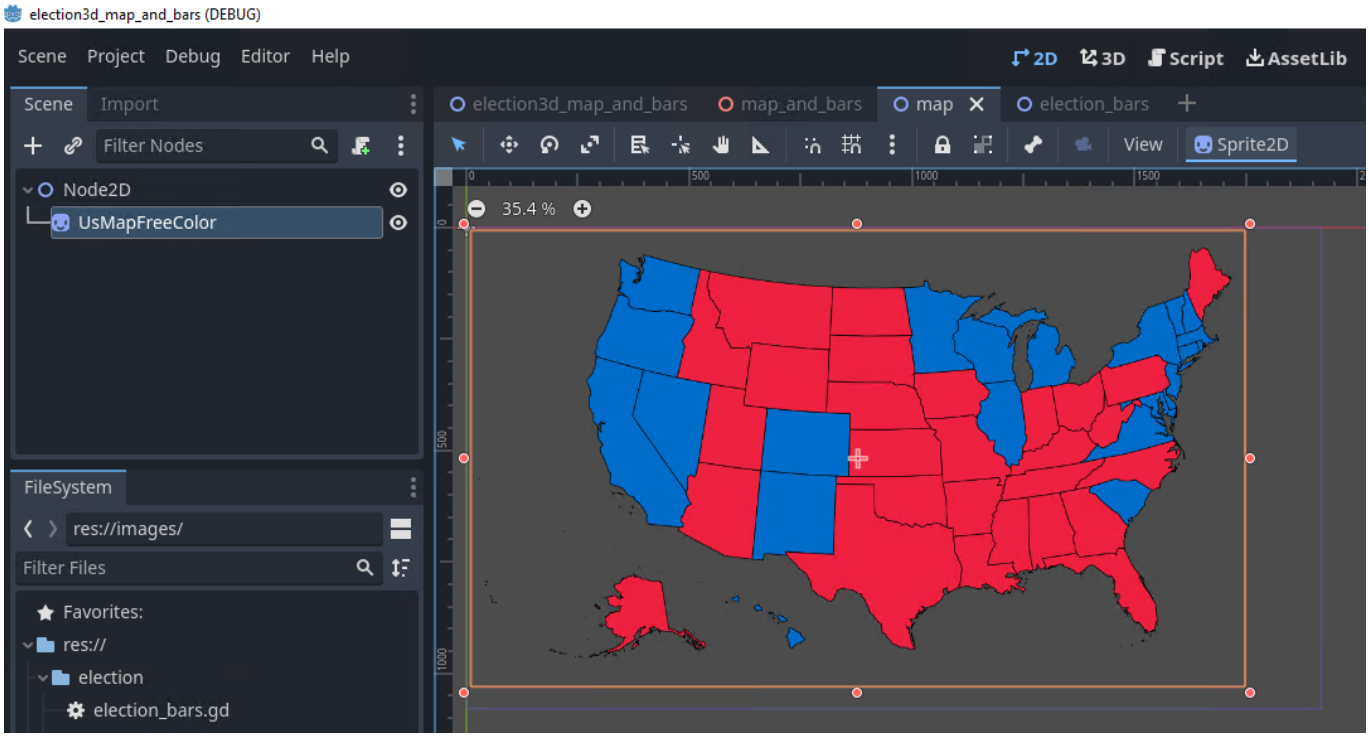
Also, the project includes:

- **us_map_free_color.svg** - SVG map of the US territory
- **election3d_map_and_bars.gd** - GD Script file for the election3d_map_and_bars.tscn scene.
- **election_bars.gd** - GD Script file for the election_bars.tscn scene.
- **candidate_1.jpg** - photo of the first candidate.
- **candidate_2.jpg** - photo of the second candidate.
- **bars_stub.png** - stub for easy scaling and positioning of the Sprite3D node displaying progress bars.

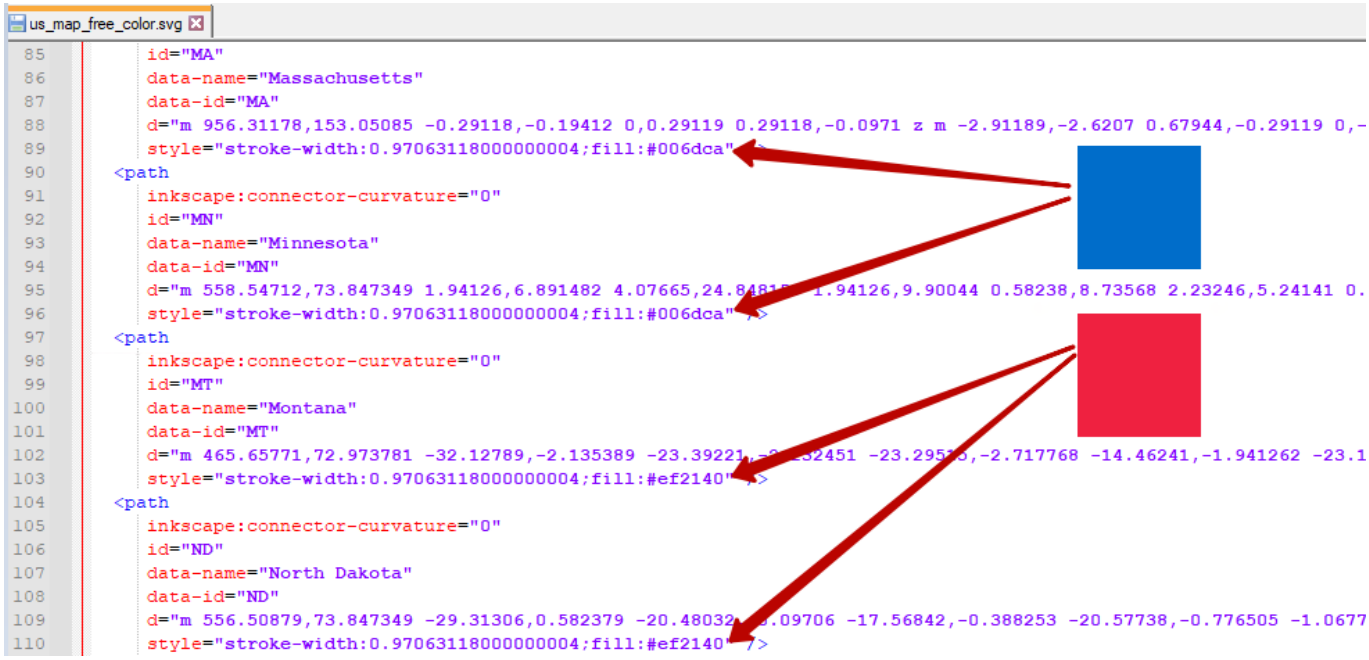
Let’s look closer at the content of each scene.

Map.tscn Scene

The 2D scene contains one Sprite2D object that outputs the map from the us_map_free_color.svg file to the screen.

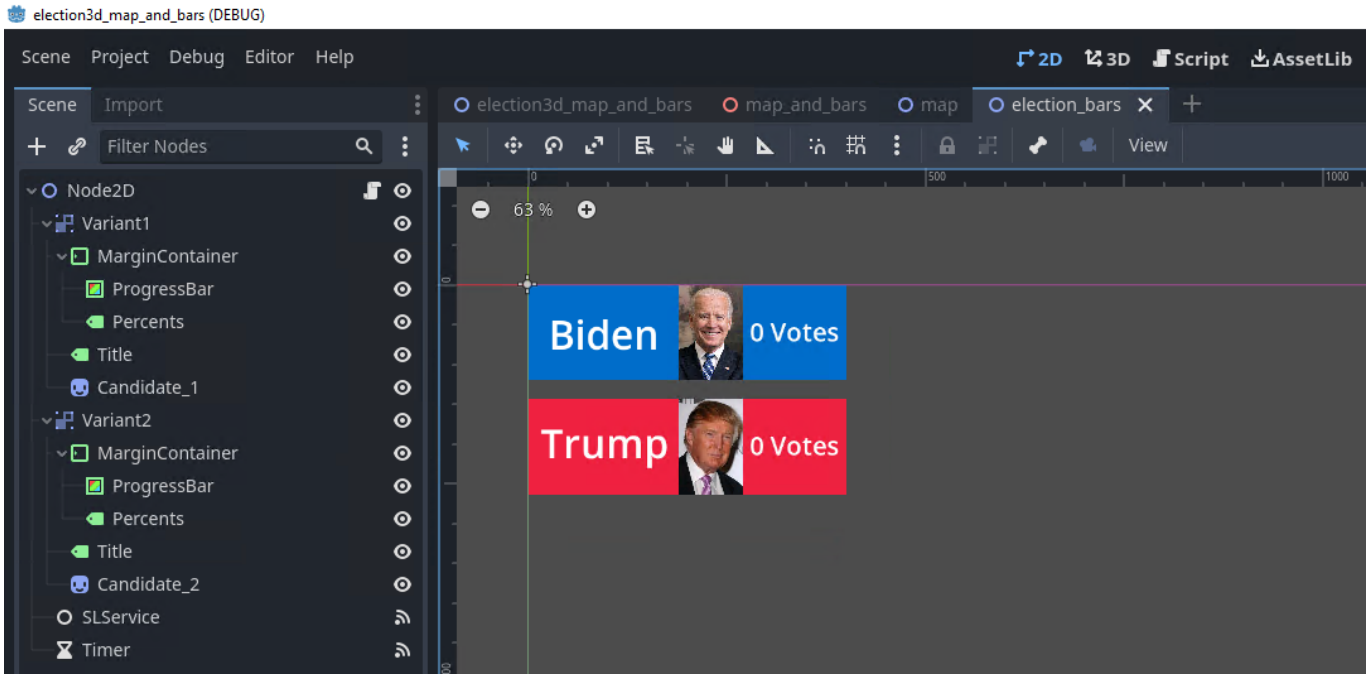


The map has SVG format, which allows you to promptly change the colors directly in the text editor.



Election_bars.tscn Scene

The 2D scene with progress bars visualizing the voting process contains more elements.



- Variant1 and Variant2 (CanvasGroup) nodes combine all the children of the tree.
- The MarginContainer node is used to scale the progress bar itself and move the text. Total width of the element is 1000px. The left part contains candidates' names and their photos, so the useful area for visualizing the progress bar would be 600 px. To set the initial zero position, use the Margin Right = 600 setting.
- Candidate_1 and Candidate_2 (Sprite2D) nodes are used to output the candidate's photo.
- The SLService node is used to retrieve display data from Skylark software.
- The Timer node is used to launch the operation of reading data from a file, which allows the data to be updated in the moment of displaying the composition with the set 5-second interval.

The scene root node has a script attached that provides the file reading and progress bar scaling functionality.

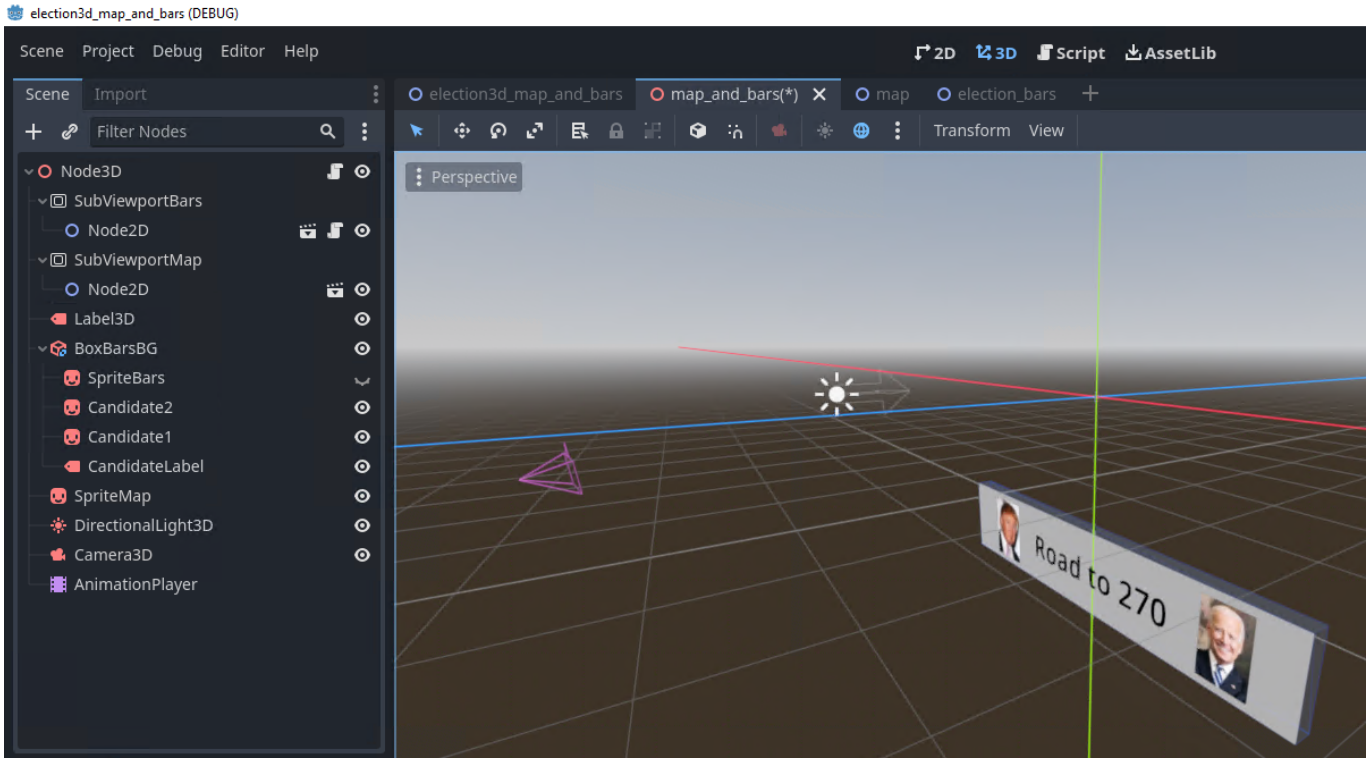
election_bars.gd

```
extends Node2D

# The directory with user files is located at:
''%APPDATA%\Godot\app_userdata\<Project_Name|Skylark_GodotModule_Name>'
'.
var votes_path = "user://votes.txt"

# Called when the node enters the scene tree for the first time.
func _ready():
        $Timer.start() # Launch timer
        load_votes) # Initial data load from the file
        print(OS.get_user_data_dir()) # Output of data on the current
user directory

# The function is called upon retrieving actions from Skylark SL Neo by
the SLService node or manually.
```

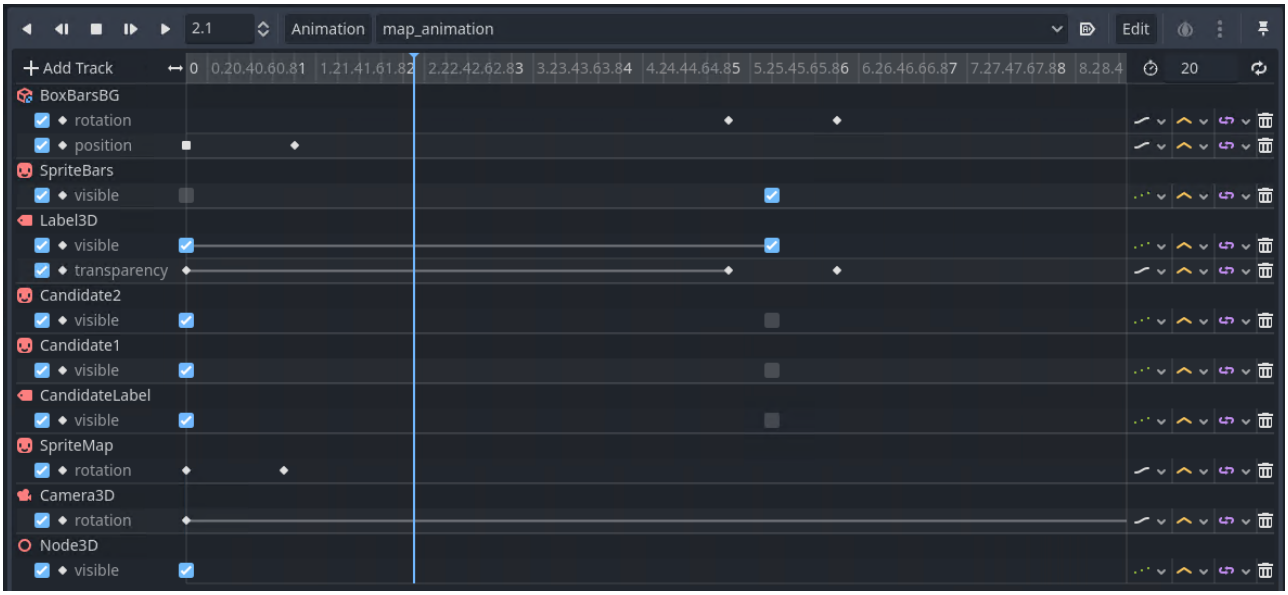
```
# params[0] - votes for the first candidate, params[1] - votes for the
second candidate
func _on_sl_service_action_received(name, params):
	if name == "vote":
		if params.size() == 2:
			# If the necessary number of parameters is
passed, the value is written to the Label node, and the formula is used
to calculate a new value for Margin Right of the MarginContainer node.
$Variant1/MarginContainer/Percents.set_text(params[0]+" Votes ")
$Variant1/MarginContainer.add_theme_constant_override("margin_right",
(100-int(params[0]))*600/100)
$Variant2/MarginContainer/Percents.set_text(params[1]+" Votes ")
$Variant2/MarginContainer.add_theme_constant_override("margin_right",
(100-int(params[1]))*600/100)

# Function for reading votes from a text file. Supposedly, the first
line of the file contains a numeric value of votes for the first
candidate, and the second line for the second candidate, respectively.
func load_votes():
	var votes = []

	if FileAccess.file_exists(votes_path):
		print("file found")
		var file = FileAccess.open(votes_path, FileAccess.READ)

		while not file.eof_reached(): # iterate through all
lines until the end of file is reached
			votes.append(file.get_line())
		file.close()

		_on_sl_service_action_received("vote", votes)

	else:
		print("file not found")

	return

# The function is called at the end of the specified interval in the
Timer node.
func _on_timer_timeout():
	load_votes()
```

Map_and_bars.tscn Scene

The 3D scene that provides map and progress bars layout in space, as well as appearing, showing and hiding animations.



- The SubViewportBars (SubViewport) node provides capture of election_bars.tscn scene.
- The SubViewportMap (SubViewport) node provides capture of map.tscn scene.
- The Label3D (Label3D) node is used to output the composition's slogan - Road to 270.
- The BoxBarsBG (CSGBox3D) node is used as a 3D semitransparent backing to output the candidate information.
- The SpriteBars (Sprite3D) node is used to overlay the texture retrieved from SubViewportBars (initially, the bars_stub.png image is used as a texture for this element in the project, while the Viewport texture is overlaid using a program method from GDScript).
- The Candidate2 and Candidate2 (Sprite3D) nodes are used to output the candidates' photos.
- The CandidateLabel node is used to output the composition's slogan - Road to 270.
- The SpriteMap node is used to overlay the texture retrieved from SubViewportMap (Viewport texture is overlaid using a program method from GDScript).
- The DirectionalLight3D node is used to create the object lighting.
- The Camera3D node is used to create a perspective view of the scene objects.
- The AnimationPlayer node is used to execute object animation parameters in the scene. The animation is 20 seconds long.



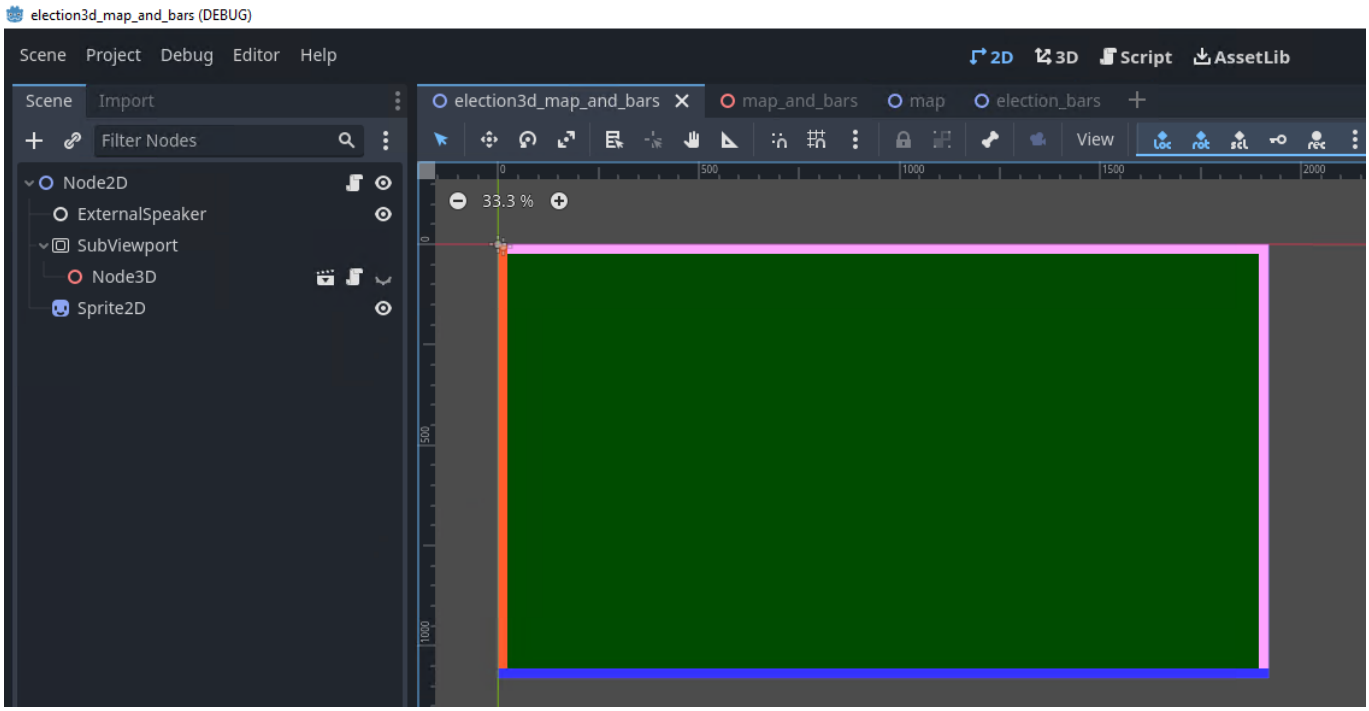
The root node of the scene has a script attached that ensures the textures are properly assigned.

map_and_bars.gd

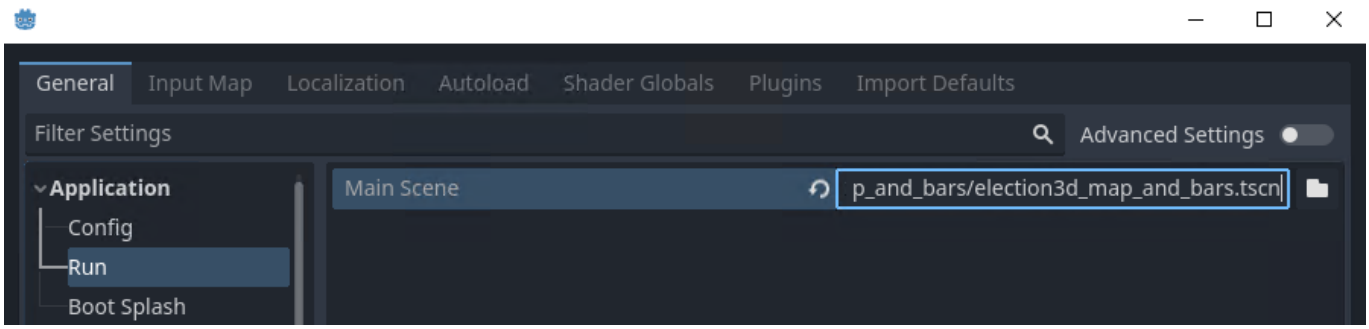
```
extends Node3D

# Called when the node enters the scene tree for the first time.
func _ready():

	var texture = $SubViewportMap.get_texture() # Retrieving the texture from Viewport
	$SpriteMap.texture = texture # Assign a texture to the Sprite3D node, to output the render result of the scene loaded to Viewport

	var texture2 = $SubViewportBars.get_texture()
	$BoxBarsBG/SpriteBars.texture = texture2
```

Election3d_map_and_bars.tscn Scene

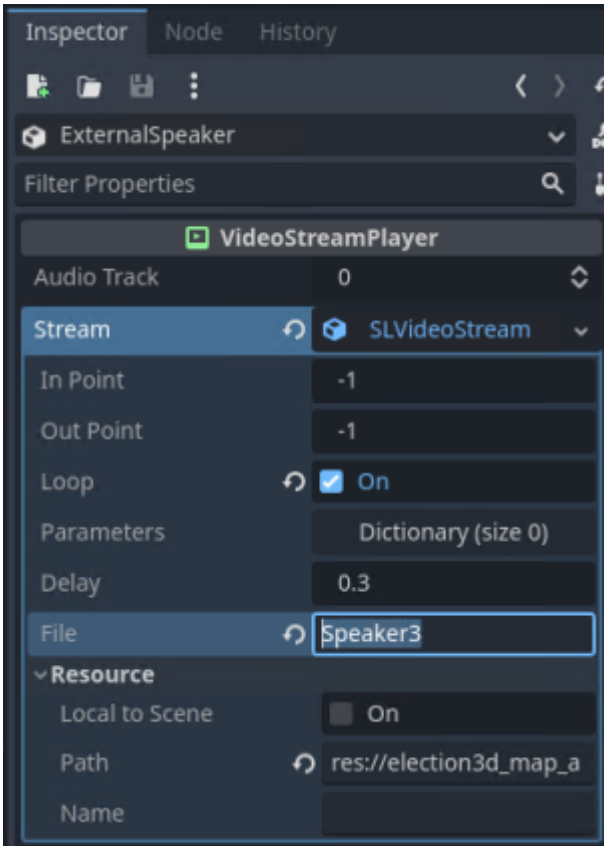
The project's main scene, providing a 3D scene overlay on the signal with the correspondent.



Launch of this scene is set in the project settings: `Project→Project Settings→General→Application→Run→Main Scene`.



- The ExternalSpeaker (SLVideoStreamPlayer) node provides playback of the Speaker3 file from the SL NEO server media database. This is where you can specify a LIVE clip to play live streams from the server input.



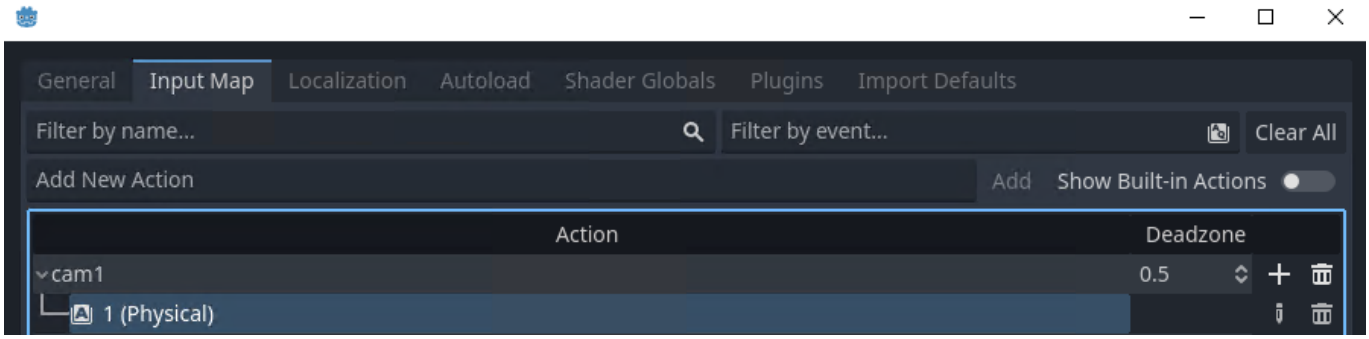
- The SubViewport (SubViewport) node provides capture of the map_and_bars.tscn scene.
- The Sprite2D node is used to overlay the texture retrieved from SubViewport (Viewport texture is overlaid using a program method from GDScript).

The scene root node has a script attached that ensures that textures are displayed correctly, and animations are triggered by pressing a button on the keyboard.

election3d_map_and_bars.gd

```
extends Node2D

# Called when the node enters the scene tree for the first time.
func _ready():
	var texture = $SubViewport.get_texture() # Retrieving the texture from Viewport
	$Sprite2D.texture = texture # Assign a texture to the Sprite2D node, to output the render result of the scene loaded to Viewport

# Called when user input is detected in the Godot window
func _input(event):
	if event is InputEventKey:
		print(event)

	# If the combination corresponding to the cam1 Action is pressed
	if event.is_action_pressed("cam1"):
$SubViewport/Node3D/AnimationPlayer.play("map_animation") # Launches “map_animation” in the AnimationPlayer node from the scene loaded to SubViewport.
```

The “1” key is assigned in the project settings to launch the cam1 Action.



The project’s program logic only serves to demonstrate functionality. In real-world projects, keystroke monitoring is added to ensure the correct order and timeliness of switches.