- AudioStreamPlayer.gd

```
extends Node

onready var jump_sound = preload("res://src/Audio/Jump.mp3")
onready var death_sound = preload("res://src/Audio/Game Over.mp3")
onready var startup_sound = preload("res://src/Audio/Startup.mp3")
onready var sound_player = $AudioStreamPlayer

func play_sound(stream: AudioStreamMP3) -> void: #plays audio
	sound_player.stream = stream
	sound_player.stream.set_loop(false) #make sure audio doesn't loop
	sound_player.play()

func _ready():
	pass
```

- Background.gd

```
extends Sprite

var speed = -3 - (Global.elapsed_time / 5.0)
var texture_width: float = 0

func _ready():
	texture_width = abs(texture.get_size().x * scale.x)

func _process(delta: float) -> void:
	if !Global.dead and Global.started and !Global.paused:
		speed = -3 - (Global.elapsed_time / 5.0)
		position.x += speed
		_attempt_reposition()

func _attempt_reposition() -> void:
	if position.x <= -0.5 * texture_width:
		# Don't just reset position texture width, otherwise there will be a gap
		position.x += 2 * texture_width
```

- 'Death Screen'.gd

```
extends RichTextLabel

# Called when the node enters the scene tree for the first time.
```

```
func _ready() -> void:
	pass # Replace with function body.

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
	if Global.dead: #if the player is dead, the death screen shows
		self.visible = true
	else:
		self.visible = false #if not, it doesn't show
```

- Enemy.gd

```
extends KinematicBody2D
class_name Enemy

var velocity = Vector2.ZERO
var speed = -3 - (Global.elapsed_time / 5.0)

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
	add_to_group("ENEMIES") #add all enemies to the group "ENEMIES" so they can be
tracked easily later

func _physics_process(delta: float) -> void:
	if !Global.dead and Global.started:
		velocity.x = speed #ensures constant velocity to the left
		var collision = move_and_collide(velocity) #move and check for collisions
		if collision and collision.collider.name == "Player": #if a collision happens (and it's
with the player, not the tilemap), the player dies
			Global.die()

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
	speed = -3 - (Global.elapsed_time / 5.0)
	if self.global_position.x < -250: #delete once it's entirely out of frame
		queue_free()
```

- Ground.gd

```
extends Node2D

var base_cactus = preload("res://src/Small Cactus.tscn") #load small cactus
var base_large_cactus = preload("res://src/Large Cactus.tscn") #load big cactus
var base_triple_cactus = preload("res://src/Triple Cactus.tscn") #load big cactus
```

```
var base_double_cactus = preload("res://src/Double Cactus.tscn")
var new_enemy_time = 5 #elapsed time to load new enemy

var rng = RandomNumberGenerator.new() #random number generator

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
	if OS.get_name() == "HTML5":
		OS.set_window_maximized(true)
	else:
		GlobalAudioStreamPlayer.play_sound(GlobalAudioStreamPlayer.startup_sound)
	Global.load_score()

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
	if !Global.dead and Global.started and !Global.paused:
		if Global.elapsed_time != 0 and Global.elapsed_time >= new_enemy_time:
#spawn new cactus every set amount of time
			if Global.elapsed_time >= 75:
				new_enemy_time += 1.5
			else:
				new_enemy_time += (4 - Global.elapsed_time/25)
			create_enemy()
	elif (Global.dead):
		if Input.is_action_just_pressed("ui_focus_next"):
			get_parent().add_child(load("res://src/Ground.tscn").instance())
			queue_free()
			Global.reload()

func random_num(minimum: int, maximum: int) -> int: #returns random number between range
(inclusive)
	rng.randomize()
	return rng.randi_range(minimum, maximum)

func spawn_enemy(scene: PackedScene) -> void:
	var new_cactus = scene.instance() #instantiate new cactus and put it right outside the
screen
	new_cactus.global_position = Vector2(1040, 560)
	if random_num(1, 2) == 1:
		new_cactus.set_scale(Vector2(-1,1)) #flip orientation of cactus
	add_child(new_cactus)

func create_enemy() -> void:
	var random_num = random_num(1, 4)
```

```
	if random_num == 1: #spawn large cactus
		spawn_enemy(base_large_cactus)
	elif random_num == 2: #spawn small cactus
		spawn_enemy(base_cactus)
	elif random_num == 3: #spawn triple cactus
		spawn_enemy(base_triple_cactus)
	else: #spawn double cactus
		spawn_enemy(base_double_cactus)
```

- Pauser.gd

```
extends Node

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
	self.visible = false # Replace with function body.

func _input(event) -> void:
	if !Global.dead and Global.started:
		if event.is_action_pressed("ui_cancel"):
			Global.paused = !Global.paused
			get_tree().paused = !get_tree().paused
			self.visible = !self.visible

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
	if Global.paused:
		Global.pause_time += delta
		Global.current_time += delta
```

- Player.gd

```
extends KinematicBody2D

var velocity = Vector2.ZERO
onready var anim = $AnimationPlayer
const position_x = 0 #starting x position for player

func _physics_process(delta: float) -> void:
	if Global.started and !Global.dead:
		check_jump()
		run_animation()
		apply_gravity(delta)
		if self.global_position.x != position_x:
```

```
                    Global.die()
        else:
            anim.stop() #stop all animations and make sure screen is frozen

func check_jump() -> void:
        if Input.is_action_pressed("move_up") and is_on_floor():
            velocity.y -= Global.gravity * 0.75 #change velocity if player wants to jump
            GlobalAudioStreamPlayer.play_sound(GlobalAudioStreamPlayer.jump_sound)

func apply_gravity(delta: float) -> void:
        velocity.y += Global.gravity * delta #make sure gravity is applied to the player
        velocity = move_and_slide(velocity, Global.FLOOR_NORMAL)

func run_animation() -> void:
        if is_on_floor(): #run different player animations
            anim.play("Run")
        else:
            anim.play("Jump")


# Called when the node enters the scene tree for the first time.
func _ready() -> void:
         pass # Replace with function body.
```

- Score.gd

```
extends RichTextLabel

func _ready() -> void:
        pass # Replace with function body.

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
        if !Global.dead and Global.started:
            Global.current_time += delta
            Global.elapsed_time = int(Global.current_time - Global.pause_time) + 1
        self.text = "Score: " + str(Global.elapsed_time) + "\nHighscore: " + str(Global.highscore)
+ "\nTimes Played: " + str(Global.times_played_again) + "\nTotal Cacti Escaped: " +
str(Global.cactus_escaped) #print score and times played and cactus escaped #print score
```

- 'Starting Screen'.gd

```
extends RichTextLabel
```

```
# Called when the node enters the scene tree for the first time.
func _ready() -> void:
	pass # Replace with function body.

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta: float) -> void:
	if Input.is_action_pressed("ui_accept"): #start game once Space/Enter key is pressed
		Global.started = true
	if !Global.started: #if game hasn't started, show starting screen
		self.visible = true
	else:
		self.visible = false
```

- global.gd

```
extends Node

var dead = false
var elapsed_time = 0 #time spent playing (not paused)
var current_time = 0 #used to calculate elapsed time
var started = false
var gravity = 800.0
var paused = false
var pause_time = 0 #time game spent paused
const FLOOR_NORMAL = Vector2.UP
var score_file = "user://new_score.save" #saves highscores
var highscore = 0
var times_played_again = 0 #how many times you've played the game
var cactus_escaped = 0 #total amount of cacti you've gotten past (only counts if they've left the
screen)
var save_file_data = {"highscore": 0, "times_played_again": 0, "cactus_escaped": 0}

func load_score(): #run when starting, loads high score, times played, and cactus escaped
	var file = File.new()
	if file.file_exists(score_file):
		file.open(score_file, File.READ)
		save_file_data = parse_json(file.get_as_text()) #parse the JSON and make it a
dictionary
		#highscore = file.get_var()
		highscore = save_file_data["highscore"]
		times_played_again = save_file_data["times_played_again"]
		cactus_escaped = save_file_data["cactus_escaped"]
		file.close()
	else:
```

```
                highscore = 0 #set values if no save file found
                times_played_again = 0
                cactus_escaped = 0

func reload(): #reset variables on reload
        dead = false
        paused = false
        started = false
        elapsed_time = 0
        current_time = 0
        pause_time = 0

func save_data():
        save_file_data["highscore"] = highscore #save all of the new data into the save_file_data
dictionary
        times_played_again = times_played_again + 1
        save_file_data["times_played_again"] = times_played_again
        save_file_data["cactus_escaped"] = cactus_escaped
        var file = File.new()
        file.open(score_file, File.WRITE)
        #file.store_var(save_file_data)
        file.store_line(to_json(save_file_data)) #save the dictionary as json file
        file.close()

func die() -> void:
        dead = true
        if (int(elapsed_time) > highscore):
                highscore = int(elapsed_time) #save new highscore
        save_data()
        GlobalAudioStreamPlayer.play_sound(GlobalAudioStreamPlayer.death_sound)
```