Player Character with Camera and Killzone - Step by Step Tutorial

Overview

This tutorial will help you create a player character system where:

- The camera follows the player upward only (never down)
- A killzone follows the camera's bottom edge
- The player dies when falling below the current view

Step 1: Create the Player Scene

1. Create a new scene in Godot
2. Add a **CharacterBody2D** node as the root
3. Rename it to "Player"
4. Add the following child nodes to Player:
    - **CollisionShape2D** (for player collision)
    - **Sprite2D** or **AnimatedSprite2D** (for player visual)

Player Node Structure:

Player (CharacterBody2D)
├── CollisionShape2D
└── Sprite2D

Step 2: Set Up Player Collision and Visual

1. Select the **CollisionShape2D** node
2. In the Inspector, set the **Shape** property to a new **RectangleShape2D** or **CapsuleShape2D**
3. Adjust the shape size to fit your player sprite
4. Select the **Sprite2D** node
5. Drag your player texture into the **Texture** property

Step 3: Add Player Script

1. Right-click on the **Player** node
2. Select **Attach Script**

3. Save it as `Player.gd`
4. Replace the generated code with the Player script from the code artifact above

Step 4: Add Player to a Group (Important!)

1. Select the **Player** node
2. Go to the **Groups** tab (next to Inspector)
3. Add the player to a group called "player"
4. This helps other systems find the player automatically

Step 5: Create the Main Game Scene

1. Create a new scene
2. Add a **Node2D** as the root and rename it to "Main"
3. Instance your Player scene as a child of Main
4. Add some platform nodes (StaticBody2D with CollisionShape2D and Sprite2D) for testing

Step 6: Set Up the Camera System

1. In your Main scene, add a **Camera2D** node as a child of Main
2. Right-click on the Camera2D node and **Attach Script**
3. Save it as `CameraController.gd`
4. Use the CameraController script from the code artifact above

Configure Camera:

1. Select the Camera2D node
2. In the Inspector, set the **Player Path** to point to your Player node (click the assign button and select the Player)
3. Adjust **Smooth Speed** if desired (higher = faster following)

Step 7: Set Up the Killzone System

1. In your Main scene, add an **Area2D** node as a child of Main
2. Rename it to "Killzone"
3. Right-click and **Attach Script**
4. Save it as `Killzone.gd`
5. Use the Killzone script from the code artifact above

Configure Killzone:

1. Select the Killzone node
2. Set the **Camera Path** to point to your Camera2D node
3. Adjust **Offset Below Camera** if needed (distance below visible area)

Step 8: Set Up Input Map

1. Go to **Project → Project Settings → Input Map**
2. Make sure you have these actions defined:
    - `ui_left` (Left arrow key)
    - `ui_right` (Right arrow key)
    - `ui_accept` (Space bar for jumping)

Step 9: Final Scene Structure

Your Main scene should look like this:

```
Main (Node2D)
├── Player (CharacterBody2D instance)
├── Camera2D
├── Killzone (Area2D)
└── Platforms (your platform nodes)
```

Step 10: Testing and Fine-tuning

1. Run the scene
2. Test player movement with arrow keys
3. Test jumping with spacebar
4. Jump on platforms and notice the camera only goes up
5. Fall below the camera view to test the killzone

Adjustable Parameters:

Player.gd:

- `speed`: How fast the player moves horizontally
- `jump_velocity`: How high the player jumps (negative value)

CameraController.gd:

- `smooth_speed`: How quickly camera follows player
- `player_path`: Reference to player node

Killzone.gd:

- `camera_path`: Reference to camera node
- `offset_below_camera`: Distance below visible area before death

Tips and Troubleshooting

Camera Not Following:

- Make sure the player_path is set correctly
- Check that the player is in the "player" group
- Verify the camera script is attached

Killzone Not Working:

- Ensure the camera_path is set correctly
- Check that the player has a `die()` method
- Make sure the Area2D has monitoring enabled

Player Falls Through Platforms:

- Check that platforms have CollisionShape2D nodes
- Ensure platforms are StaticBody2D or RigidBody2D nodes
- Verify collision layers and masks are set up correctly

Performance Optimization:

- The killzone creates a very wide collision shape (10000 units)
- For better performance in large levels, consider making it narrower
- You can optimize by only updating positions when the camera moves significantly

Advanced Features You Can Add

1. **Smooth Camera Boundaries**: Add left/right limits to camera movement
2. **Death Animation**: Add particle effects or animation before respawning
3. **Checkpoint System**: Save player position at certain points
4. **Sound Effects**: Add audio for jumping, landing, and dying
5. **Multiple Lives**: Track lives before game over

This system provides a solid foundation for platformer games where upward progress is key and falling is dangerous!

```
# Player.gd - Attach this to your player CharacterBody2D node
extends CharacterBody2D

@export var speed = 300.0
@export var jump_velocity = -400.0

# Get the gravity from the project settings to be synced with RigidBody nodes
var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")

func _physics_process(delta):
	# Add the gravity
```

```
	if not is_on_floor():
		velocity.y += gravity * delta

	# Handle jump
	if Input.is_action_just_pressed("ui_accept") and is_on_floor():
		velocity.y = jump_velocity

	# Handle movement
	var direction = Input.get_axis("ui_left", "ui_right")
	if direction != 0:
		velocity.x = direction * speed
	else:
		velocity.x = move_toward(velocity.x, 0, speed)

	move_and_slide()

func die():
	print("Player died!")
	# Reset player position or reload scene
	get_tree().reload_current_scene()


# CameraController.gd - Attach this to a Camera2D node
extends Camera2D

@export var player_path: NodePath
@export var smooth_speed = 5.0

var player: Node2D
var highest_y_position: float

func _ready():
	if player_path:
		player = get_node(player_path)
	else:
		# Try to find player automatically
		player = get_tree().get_first_node_in_group("player")

	if player:
		# Initialize camera position
		global_position = player.global_position
		highest_y_position = player.global_position.y
	else:
		print("Warning: Player not found for camera!")

func _process(delta):
	if not player:
		return
```

```
	# Update highest position if player has climbed higher
	if player.global_position.y < highest_y_position:
		highest_y_position = player.global_position.y

	# Camera follows player horizontally and upward only
	var target_position = Vector2(
		player.global_position.x,
		highest_y_position
	)

	# Smooth camera movement
	global_position = global_position.lerp(target_position, smooth_speed * delta)


# Killzone.gd - Attach this to an Area2D node
extends Area2D

@export var camera_path: NodePath
@export var offset_below_camera = 100.0

var camera: Camera2D
var collision_shape: CollisionShape2D

func _ready():
	# Get camera reference
	if camera_path:
		camera = get_node(camera_path)
	else:
		camera = get_tree().get_first_node_in_group("camera")

	if not camera:
		print("Warning: Camera not found for killzone!")
		return

	# Create collision shape
	collision_shape = CollisionShape2D.new()
	var rect_shape = RectangleShape2D.new()
	rect_shape.size = Vector2(10000, 100)  # Wide killzone
	collision_shape.shape = rect_shape
	add_child(collision_shape)

	# Connect the body_entered signal
	body_entered.connect(_on_body_entered)

func _process(delta):
	if not camera:
		return
```

```
	# Position killzone below camera's bottom edge
	var camera_bottom = camera.global_position.y + get_viewport().size.y / (2 *
camera.zoom.y)
	global_position = Vector2(camera.global_position.x, camera_bottom +
offset_below_camera)

func _on_body_entered(body):
	# Check if the body that entered is the player
	if body.has_method("die"):
		body.die()
```