Jump'n'Run

1. **neues Projekt** erstellen (**Jump_n_Run**) als Kompatible-Version

2. **Kopieren von Vorgabegrafiken** in einen Unterordner **art**
   kenney_platformers…/Base pack/Tiles/tiles_spritesheet.png (Level-Bausteine)
   /Player/p1_walk (animiertes Player)
   /Enemies/slimeWalk1.png + 2 in Unterordner e1_walk (animiertes Monster)

3. **Spieler modellieren**
   Szene => Andere Node => CharacterBody2D (Kräfte + Steuerung + …)
   umbenennen in **Player** (Nodes groß, sonst klein – Namingconvention!)
   StrgA (Node anhängen) => AnimatedSprite2D
   Inspektor => Animation => SpriteFrames => neues SpriteFrame
   evtl. noch einmal anklicken, damit sich das Animationsfenster öffnet
   Player-Bilder aus dem Dateisystem (p1_walk) in das Fenster ziehen
   im Animationsfenster ➢ (Play) um animierte Figur zu sehen
   Player-Szene speichern als **player.tscn**

4. **Main-Szene gestalten**
   Neue Szene mit + hinzufügen
   2D-Szene (Node2D)
   in Main umbenennen und speichern als **main.tscn**
   player.tscn aus Dateisystem auf Main-Node ziehen
   Player in die Mitte des blauen Rechtecks (Bildschirm) ziehen
   ➢ Fehlermeldung => aktuelle als Hauptszene auswählen

5. **Steuerung implementieren**
   player.tscn auswählen und im Kontextmenü von Player => Script hinzufügen
   Vorlage: **Basic Movement** übernehmen => Erstellen
   Script interpretieren und Teile (Gravitation) auskommentieren
   #var gravity und dann den Block mit der roten Zeile

```
7  # Get the gravity from the project settings to be synced with RigidBody nodes.
8  # var gravity = ProjectSettings.get_setting("physics/2d/default_gravity")
9
10 func _physics_process(delta):
11   # Add the gravity.
12   # if not is_on_floor():
13   #   velocity.y += gravity * delta
```

ui steht für User Interface
delta – Zeit seit letztem Aufruf
direktion (+1 oder -1, je nach Pfeiltaste)
move_and_slide – schaut auf delta, Kräfte, Geschwindigkeit und
berechnet die Bewegung des Spielers ➢ (keine Animation)

6. **Animation programmieren**
   Script ergänzen **#startet und pausiert die Animation** ➢ (Rückwärtslaufen)

```
28  if velocity.x != 0:
29    $AnimatedSprite2D.play()
30    if velocity.x > 0:
31      $AnimatedSprite2D.flip_h = false
32    else:
33      $AnimatedSprite2D.flip_h = true
34  else:
35    $AnimatedSprite2D.pause()
```

   weiteres if-Statement: flip_h (horizontale Spiegelung) ➢
   im Inspektor SpeedScale auf 5 ändern (für flüssige Animation) ➢

7. **SpriteSheet splitten**
   Neue Ressourcendatei im Dateisystem anlegen (Typ: TileMap, Name: tiles.tres)
   Dateisystem (alle Ordner schließen) => Kontextmenü => Neue Ressource
   => TileSet (Name: tiles.tres)
   anklicken und aus **art** das tiles_spritesheet.png nach Tiles ziehen und Kacheln erstellen lassen (von Hand (70, 70) als Texture und (2, 2) als Trennung einstellen)
   graue Teile ggf. von Hand aktivieren (durch anklicken)
   (70, 70) auch im Inspektor für TileSize einstellen!

8. **Level bauen** (Hintergrund)
   Neue Szene und TileMap hinzufügen (Node) als **Level1**
   Inspektor => in neues TileSet unsere
   in Tiles (unten) TileSet auswählen und unser tiles.tres laden
   Tile(7,8) in Szene holen (Level1 in Szene aktiv)
   Tile(8,0) und Tile (7,12) ergänzen
   level1.tscn an Main hängen ➢ (Player fällt durch den Boden) ☹

9. **Kollisionen einbauen**
   StrA auf Player => CollisionShape2D
   Inspektor => neue Shape (CapsuleShape2D)
   Nodes gruppieren, damit nicht separat verschiebbar (StrgG oder Menüsymbol)
   CollisionShape für Hindernisse:
   tiles.tres doppelt anklicken
   Inspektor => Physics Layers => Element hinzufügen
   Collision Layer => Hamburger Menü => Ebenenname (**Hintergrund**)
   im Tiles-Editor (Registerkarte Auswählen) vorher evtl. unten TileSet auswähl.
   Tile(7,8) auswählen => Physik => Physics Layer 0 =>
   HamMenü => auf Default zurücksetzen (ganze Kachel)
   ➢ ☺ dann nach links laufen ☹
   Tile(8,0) auswählen => Physik => Physics Layer 0 =>
   HaMe => auf Default zurücksetzen und oberen Punkt löschen
   (rechte Maustaste oder Menüpunkt) (alternativ: Polygon)
   ➢ ☺

10. **Kamera** (Hintergrund bewegt sich)
StrgA an Player => Camera2D ➢ ☺

11. Dekoration

Level1 => Tile(4,3) (ein Schild) einsetzen ➢ (Spieler läuft hinter Schild)
Inspektor => TileMap => Layers => Z Index = -1 ➢ ☺
Tür (Tile(9,6) + Tile(9,5)) am rechten Rand einfügen
StrgA an Level1 Area2D (**Door**) anhängen, daran CollisionShape2D (Rechteck)

12. 2. Level erstellen und betreten

level_1.tscn speichern als level_2.tscn und Node umbenennen (Level2)
Level1 => Script anhängen
Signal und Player bekanntmachen
bei Türberührung (Pfeil nach oben Taste) => Signal senden

```
extends TileMap

signal level2_entered

@onready var player = get_tree().get_root().get_node("Main/Player")

# Called every frame. 'delta' is the elapsed time since the previous frame.
func _process(delta):
  if Input.is_action_just_pressed("ui_up") and $Door2.overlaps_body(player):
    emit_signal("level2_entered")
```

Main => Script anhängen:
Signal mit Funktion _start_Level_2 verbinden
func _start_Level_2 schreiben ➢ ☺

```
func _ready():
  $Level1.connect("level2_entered", _start_level_2)

func _start_level_2():
  $Level1.queue_free()
  add_child(load("res://level_2.tscn").instantiate())
  $Player.global_position = Vector2(307,0)
```

13. Spielende bei Absturz

StrgA an Main => Control anhängen (**UI**)
=> StrA an UI => Label anhängen (Game Over!)
Inspektor => Text = GAME OVER!
=> Theme Overrides => Font Sizes = 80
UI in der Szene ausblenden
Main-Script ergänzen ➢ ☺

```
func _process(delta):
  if !$Player:
    pass
  elif $Player.global_position.y > 600:
    $UI.visible = true
    #$Player.global_position.y = 601
    $Player.queue_free()
```