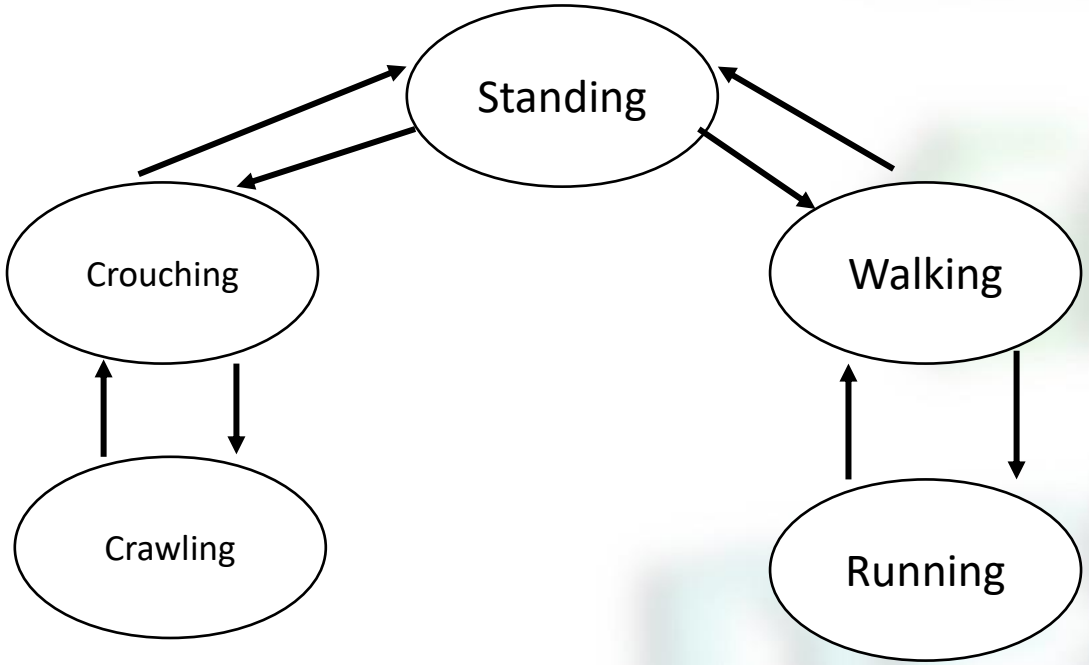
2D Platformer

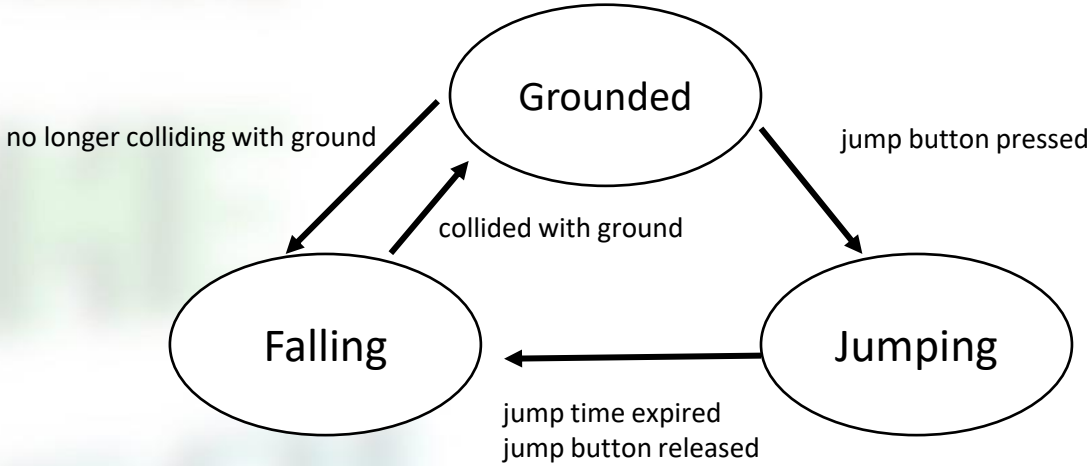
Knox Game Design

November 2026

Walking States



Jumping / Falling States



Block collision

- If in falling state
  - if adding the vertical velocity would cause collision with a block
    - Stop falling / move to grounded state
    - Set vertical velocity to zero
    - Set bottom of the player's collision box to the top of the block it collided with
- If horizontal velocity does not equal zero
  - if adding the horizontal velocity would cause collision with a block
    - If horizontal velocity is negative (moving left)
      - Move player left bounding box (player x position) to right side of block (block x position plus width)
    - If horizontal velocity is positive (moving right)
      - Move player right bounding box (player x position plus width) to left side of block (block x position)
    - Set horizontal velocity to zero

KNOX GAME DESIGN

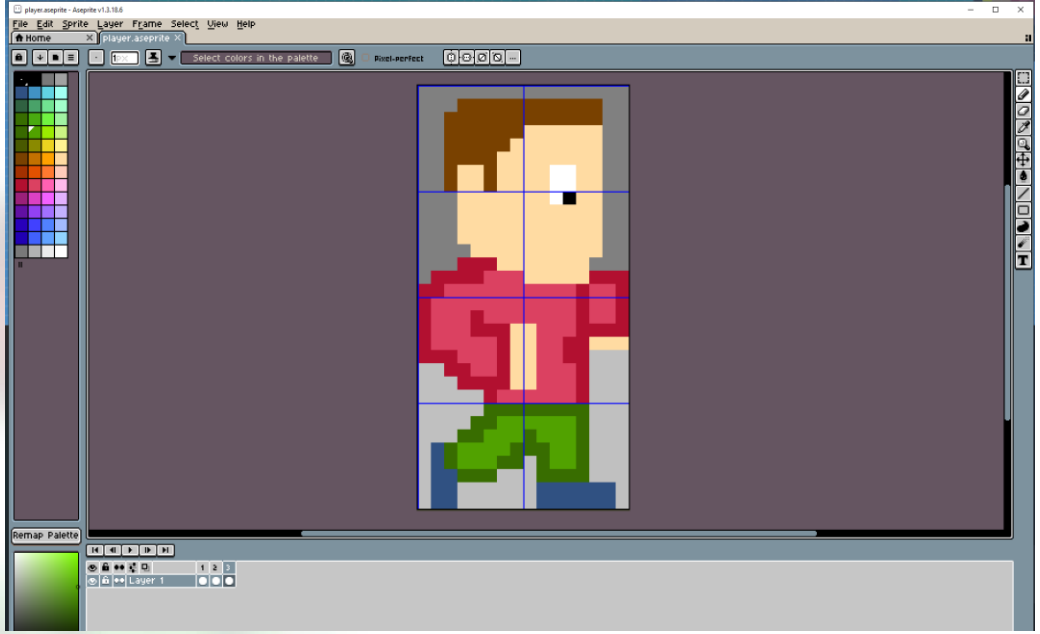
Stomping

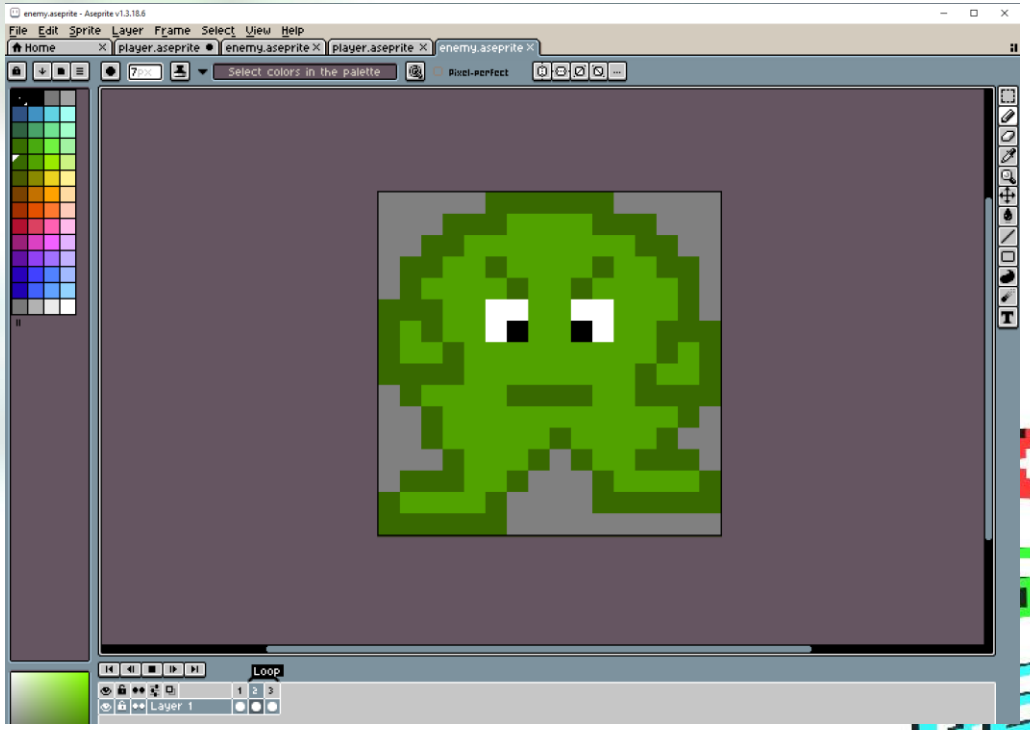
- When collided with enemy
  - If falling (vertical velocity is downward)
    - Then enemy stomped
  - Else
    - Player damaged

Note – this makes it possible to “stomp” an enemy with any portion of the player’s collision box

Sprite Graphics

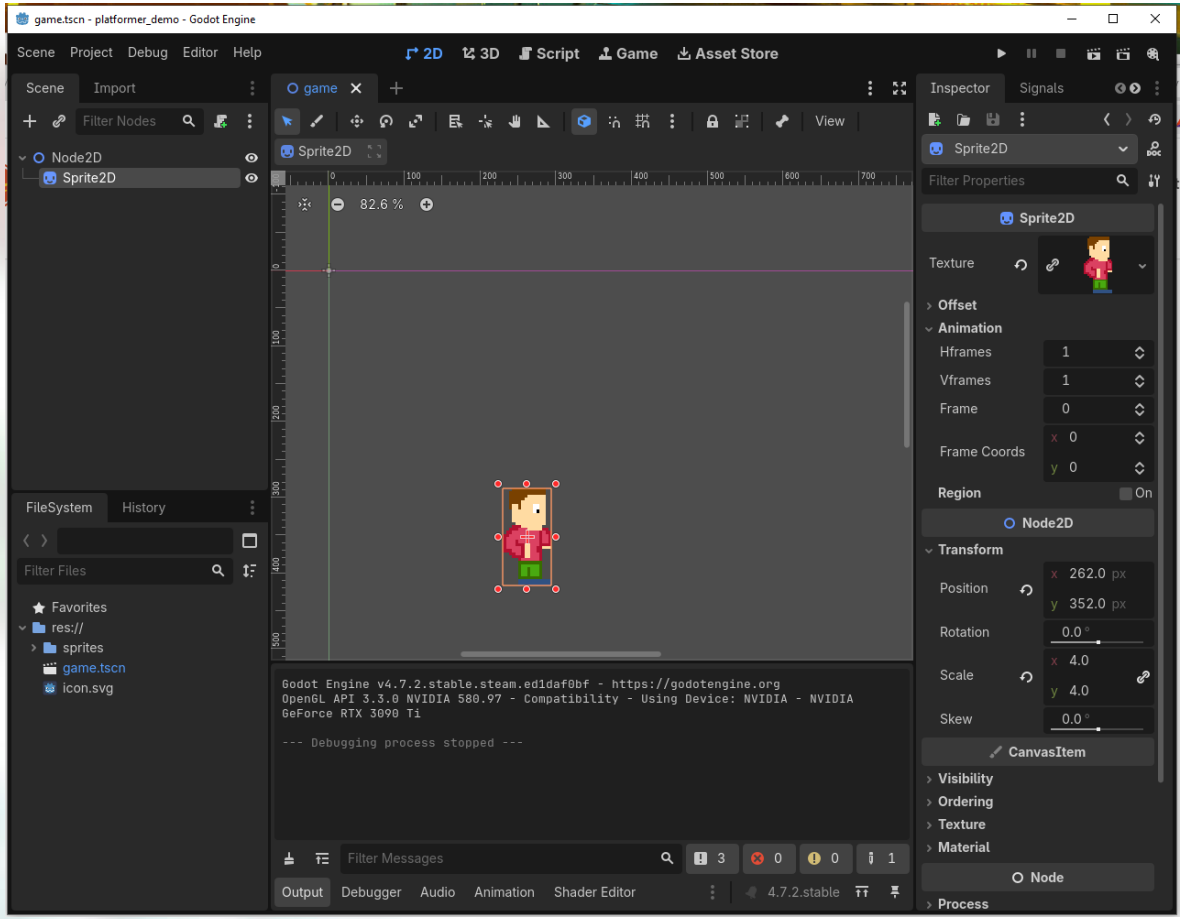
- Aesprite
- Player
  - Size: 16x32 pixels
  - Sprites for each state
    - Standing
    - Walking (3 frames)
    - Jumping
    - Change horizontal direction
    - Dead
- Enemy
  - Size: 16x16
  - Walking (2 frames)
  - Dead
- Blocks
- Export graphics to PNG files





Godot

- First step is to just get sprite on screen
  - Add child node > Sprite 2D
  - Copy sprites folder into project folder
  - Select texture under Sprite2d and pick the player sprite file
  - Set sprite scale as needed (suggested 4.0)
  - Fix blurry textures – Project Settings > Rendering > Textures > Nearest
  - Set window size (suggest 1024x960) – Project Settings > Display > Window > Viewport Width / Height



KNOX GAME DESIGN

Horizontal Movement

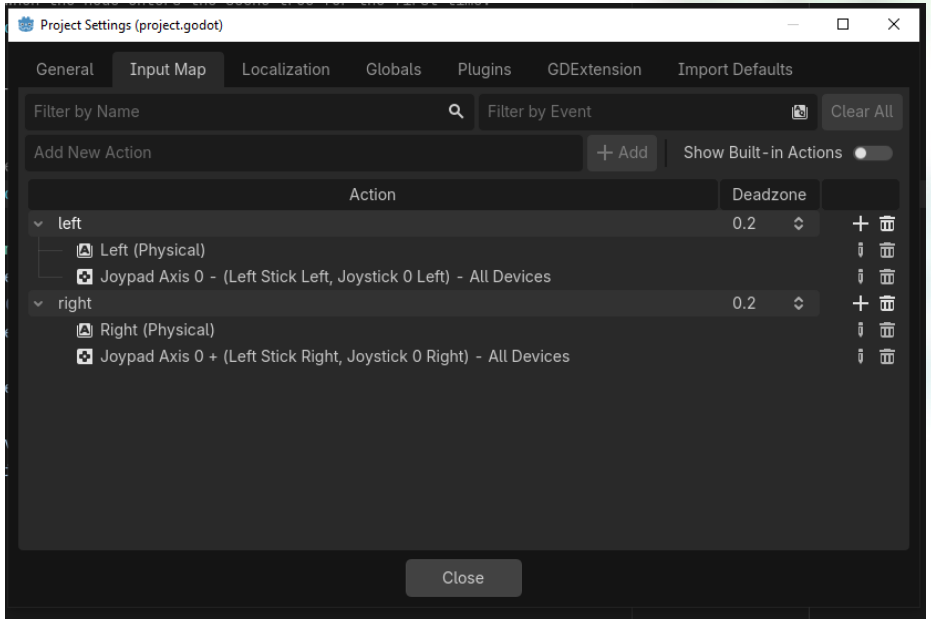
- Create horizontal velocity variable
  - Increase while right pressed
  - Decrease while left pressed
  - Move toward zero during no horizontal input
  - Update player x position on each update by horizontal velocity
  - Set maximum/minimum limit for horizontal velocity
- Suggested max velocity: 4 blocks per second * 8 pixels per block * 4 scale size = 128
- Use *position* property to set position of player

```
1   extends Node2D
2
3   var x
4   var vel_x
5   var GLOBAL_SCALE = 4.0
6
7   # Called when the node enters the scene
8   func _ready() -> void:
9       x = 0
10      vel_x = 32 * GLOBAL_SCALE
11
12  # Called every frame. 'delta' is the el
13  func _process(delta: float) -> void:
14      x += vel_x
15      position = Vector2(x, 0) * delta
16
```

KNOX GAME DESIGN

Input

- Use *Input.is_action_pressed(“<direction>”)* to get Input (key/gamepad)
- Add input mappings under Project Settings > Input Map tab
- Set constant acceleration value
- Update velocity based on input, acceleration value, and min/max limits
- Add velocity to player’s position on each update (*_process*)
- Can modify velocity and acceleration values for different terrain (such as sliding on ice)




```gdscript
extends Node2D

var x
var vel_x
var accel_x
var vel_x_max
var GLOBAL_SCALE = 4.0

# Called when the node enters the scene tree
func _ready() -> void:
    x = 0
    vel_x = 0
    accel_x = 4 * GLOBAL_SCALE
    vel_x_max = 4 * 8 * GLOBAL_SCALE

# Called every frame. 'delta' is the elapsed
func _process(delta: float) -> void:

    if (Input.is_action_pressed("left")):
        vel_x -= accel_x
    elif (Input.is_action_pressed("right")):
        vel_x += accel_x
    else:
        if (vel_x > 0):
            vel_x -= accel_x
        elif (vel_x < 0):
            vel_x += accel_x

    x += vel_x
    position = Vector2(x, 0) * delta
```

Prefabs

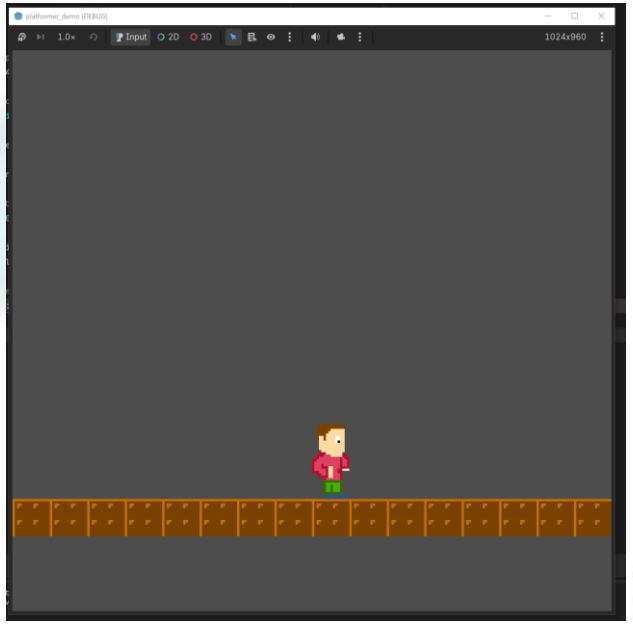
- Block prefab
  - Add new Node2D with Sprite2D as child
  - Set texture, scale
  - Save branch as Scene (block.tscn)
- Player prefab
  - Same as block, save as player.tscn
- GameManager root node
  - Make GameManager its own scene as set as MainScene in project settings
  - Load player and block prefabs
    - *var prefab_player = preload(“res://player.tscn”)*
    - *var prefab_block = preload(“res://block.tscn”)*
  - Instantiate
    - *var player = prefab_player.instantiate()*
    - *var block = prefab_block.instantiate()*
  - Set position with *position* and assign Vector2 value
  - Use *add_child(<object>)* to add the instantiated object to the scene
  - Use a for loop to create a row of blocks

```gdscript
extends Node2D

var prefab_player = preload("res://player.tscn")
var prefab_block = preload("res://block.tscn")
var GLOBAL_SCALE = 4

# Called when the node enters the scene tree for the first time.
func _ready() -> void:
    #create player
    var player = prefab_player.instantiate()
    player.position = Vector2(128 * GLOBAL_SCALE, (192 - 32) * GLOBAL_SCALE)
    add_child(player)

    #create row of blocks
    for i in range(0, 16):
        var block = prefab_block.instantiate()
        block.position = Vector2(i * 16 * GLOBAL_SCALE, 192 * GLOBAL_SCALE)
        add_child(block)
```



KNOX GAME DESIGN

Player Falling state

- Add *JumpState* enum to Player
  - Values: *GROUNDED, JUMPING, FALLING*
- When jumpstate is *FALLING*
  - Add accel_y to vel_y
  - Add vel_y to y
  - Hardcode a y floor value (such as 192)
  - When y is greater than floor value
    - Set jumpstate to *GROUNDED*
    - Set vel_y to zero
    - Move player's y position to floor value minus height

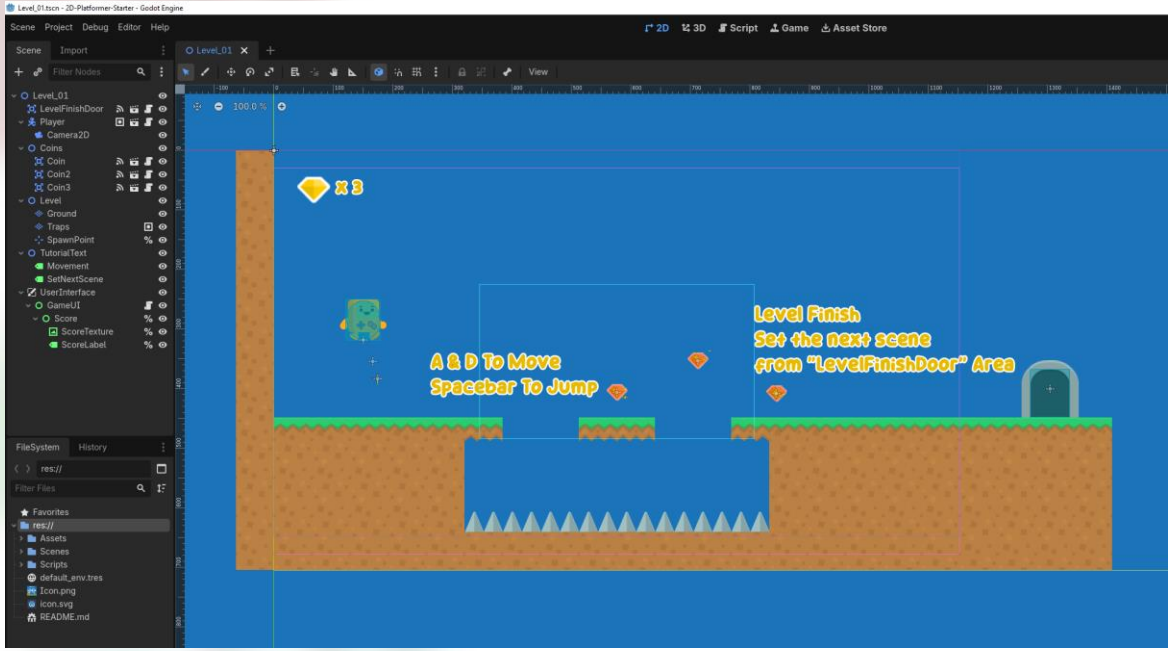
Player Jumping state

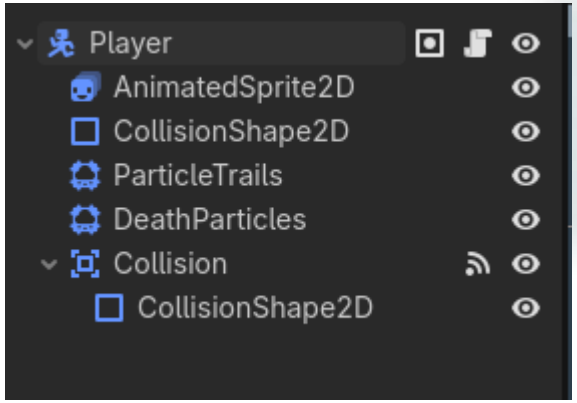
- Add *jump* input in Project settings (Space, X button)
- If *jumpstate* is GROUNDED AND jump button pressed
  - Set *jumpstate* to JUMPING
- If *jumpstate* is JUMPING
  - set jump velocity to constant jump velocity (4 blocks * 8 pixels)
  - if jump button NOT pressed, set *jumpstate* to FALLING
- Add *jump_time* and *jump_time_max* variables
  - Increment *jump_time* by delta if in JUMPING state
  - if *jump_time* is greater than *jump_time_max*, then set *jumpstate* to FALLING
  - Set *jump_time* back to zero on next jump

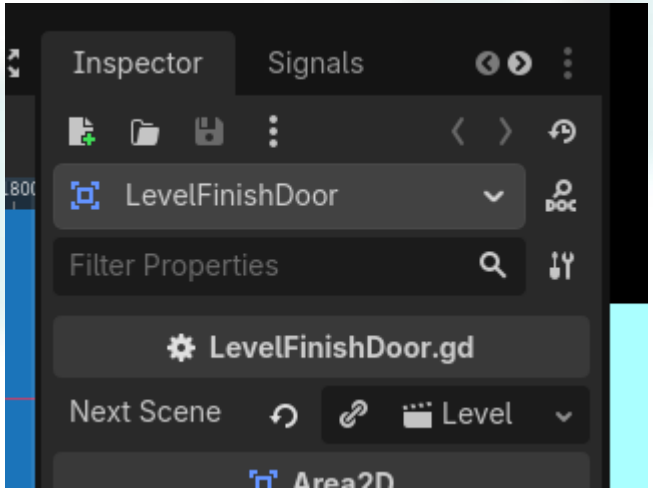
KNOX GAME DESIGN

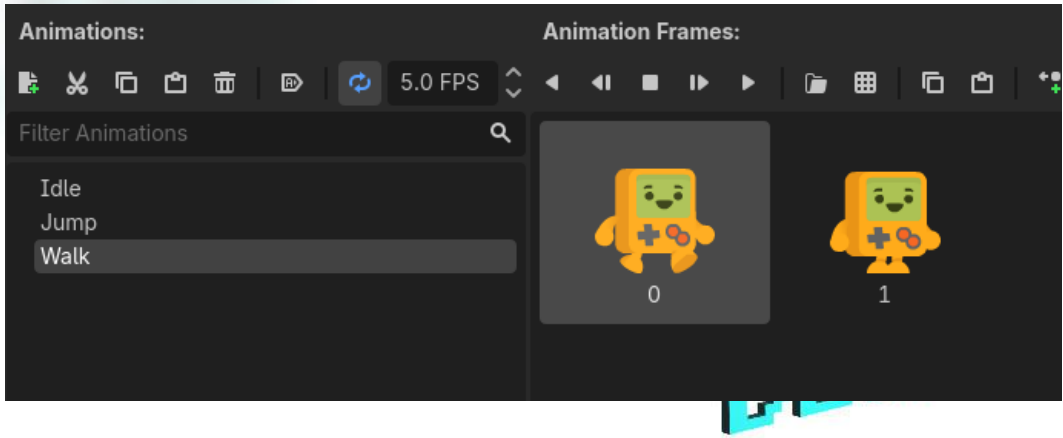
2D Platformer Starter Kit

- Prefabs
  - player.tscn
  - Coin.tscn
  - LevelFinishDoor.tscn
- Player
  - extends: CharacterBody2D
  - children: AnimatedSprite2D, CollisionShape2D
- Door takes a Level scene as an Inspector parameter for next level

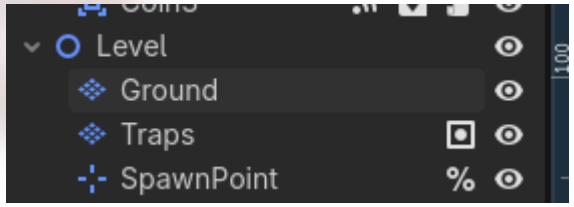




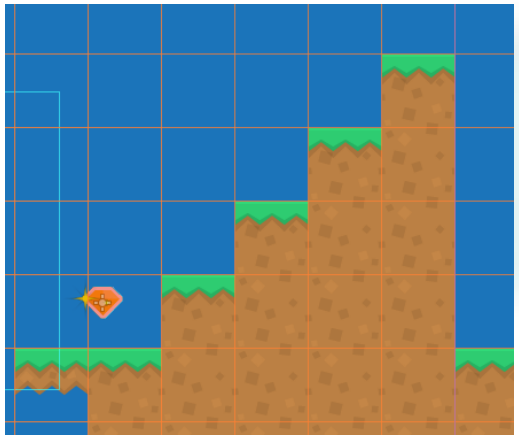


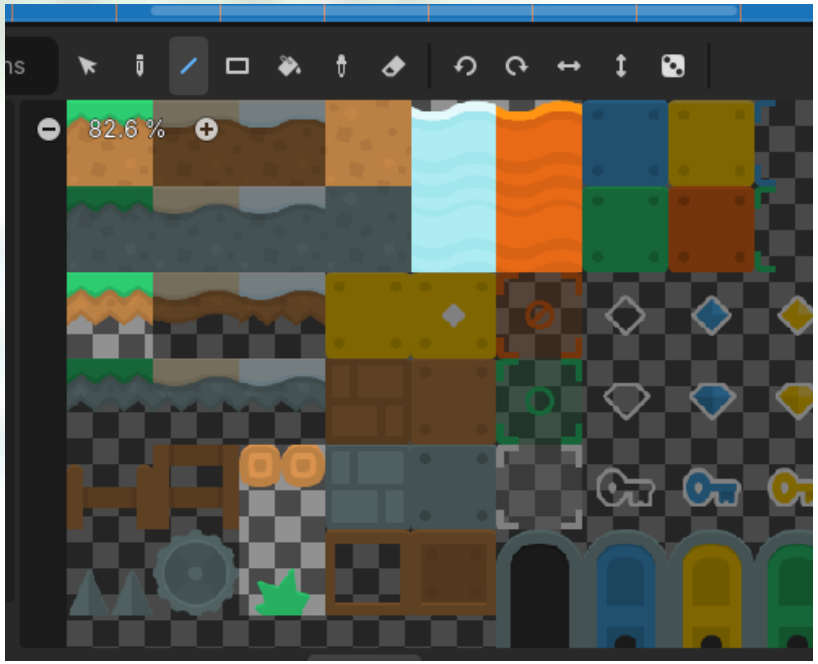


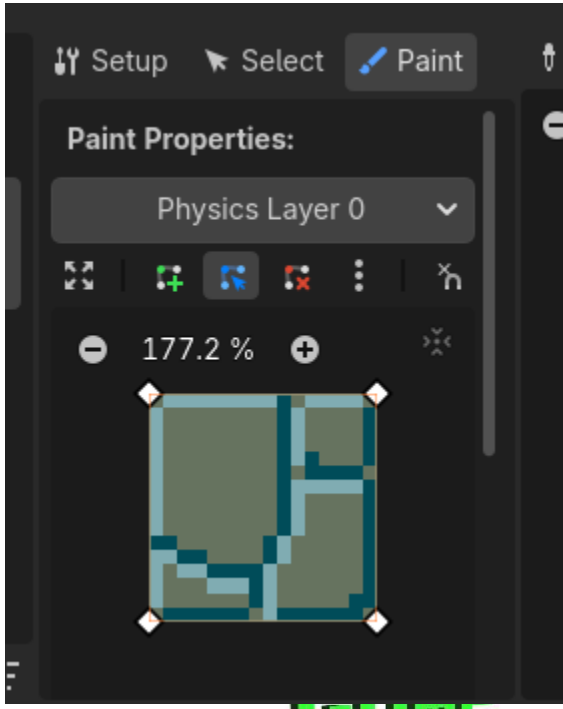
2D Platformer Starter Kit – tile map



- Level
  - Ground
  - Traps
  - SpawnPoint
- Add new tiles with Paint Tool, Line Tool
- TileSet > Paint > Properties > Physics Layer 0
  - Otherwise, player will fall through tile

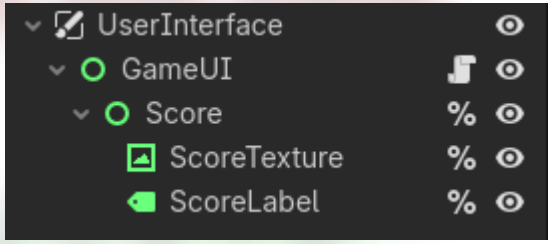




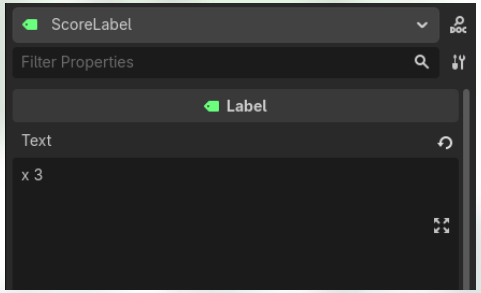


2D Platformer Starter Kit - UI

- ScoreTexture for sprites
- ScoreLabel for displaying text
  - Set with .text property








```
extends Control

@onready var score_texture = %Score/ScoreTexture
@onready var score_label = %Score/ScoreLabel

func _process(_delta):
    # Set the score label text to the score variabl
    score_label.text = "x %d" % GameManager.score
```

KNOX GAME DESIGN