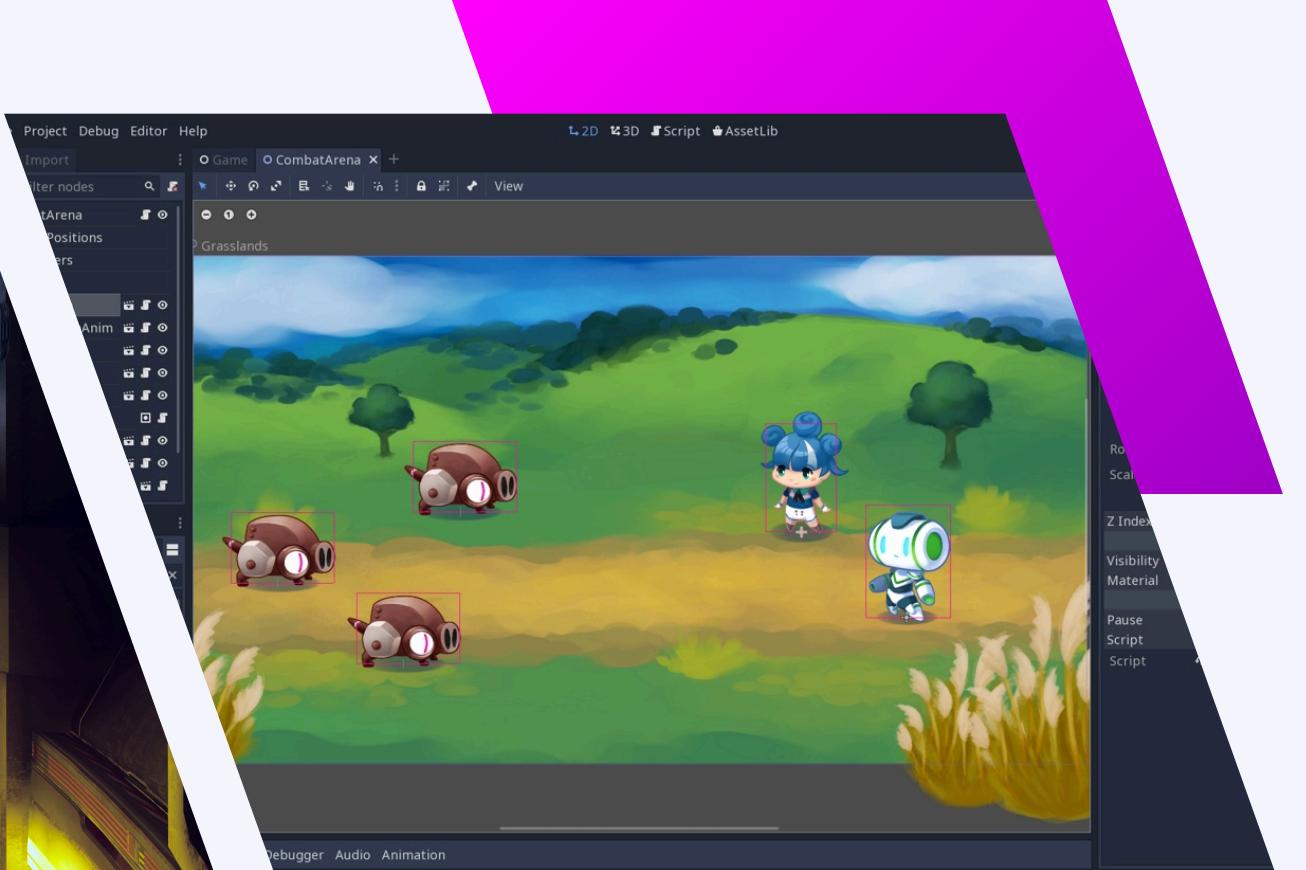
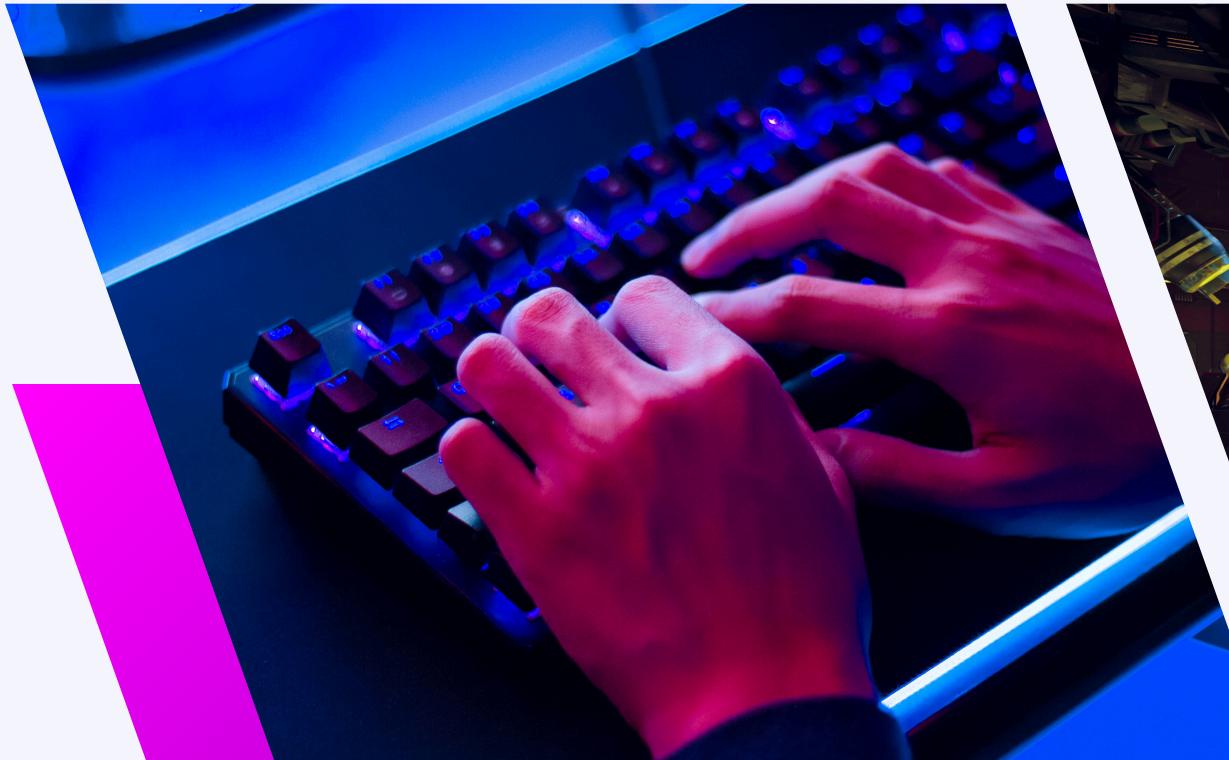


GAME DEVELOPER

College of Creative Design and Entertainment Technology



การพัฒนาเกมด้วยโปรแกรม Godot Engine

GAME ENGINE

Game Engine (เอนจินเกม) คือ ชุดเครื่องมือซอฟต์แวร์ที่ช่วยให้นักพัฒนาสามารถสร้างเกมได้อย่างมีประสิทธิภาพ โดยไม่ต้องเขียนทุกรอบบตั้งแต่เริ่มต้น

คุณสมบัติหลักของ Game Engine

- ระบบกราฟิก (Graphics Engine)
- ระบบฟิสิกส์ (Physics Engine)
- ระบบเสียง (Audio Engine)
- ระบบสคริปต์ (Scripting)
- อินเทอร์เฟซสำหรับออกแบบ (Editor UI)
- ระบบนำเข้าไฟล์ (Asset Management)





GODOT ENGINE

Godot (อวกเสียงว่า กอด็อก) เป็น ชุดพัฒนาเกม ใช้งานได้ฟรี, Open Source, รองรับ 2D/3D

โดยมีระบบ Node ที่ยืดหยุ่นและเรียนรู้ไม่ยาก

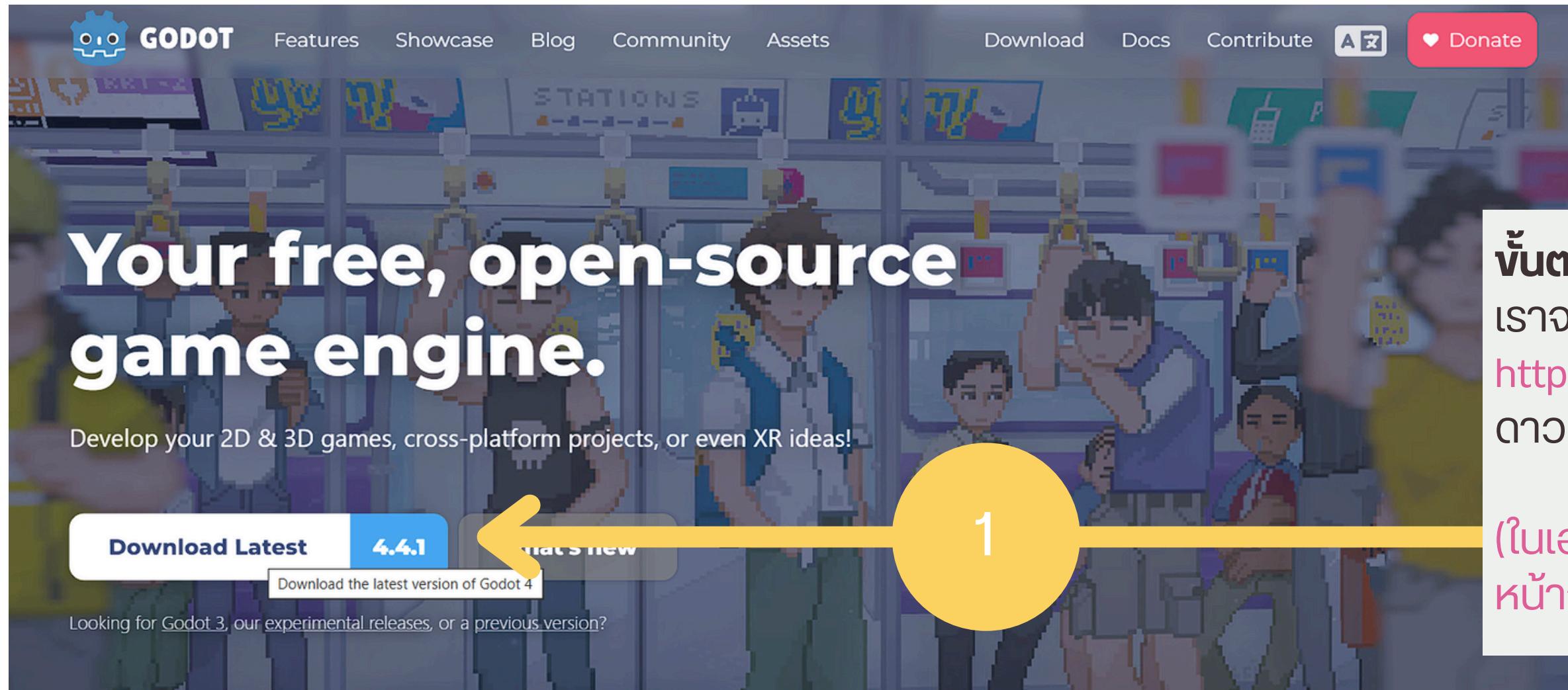
 Godot Engine
<https://godotengine.org> :

Godot Engine - Free and open source 2D and 3D game engine

Your free, open-source game engine. Develop your 2D & 3D games, cross-platform projects, or even XR ideas! ... Looking for Godot 3, our experimental releases, or ...



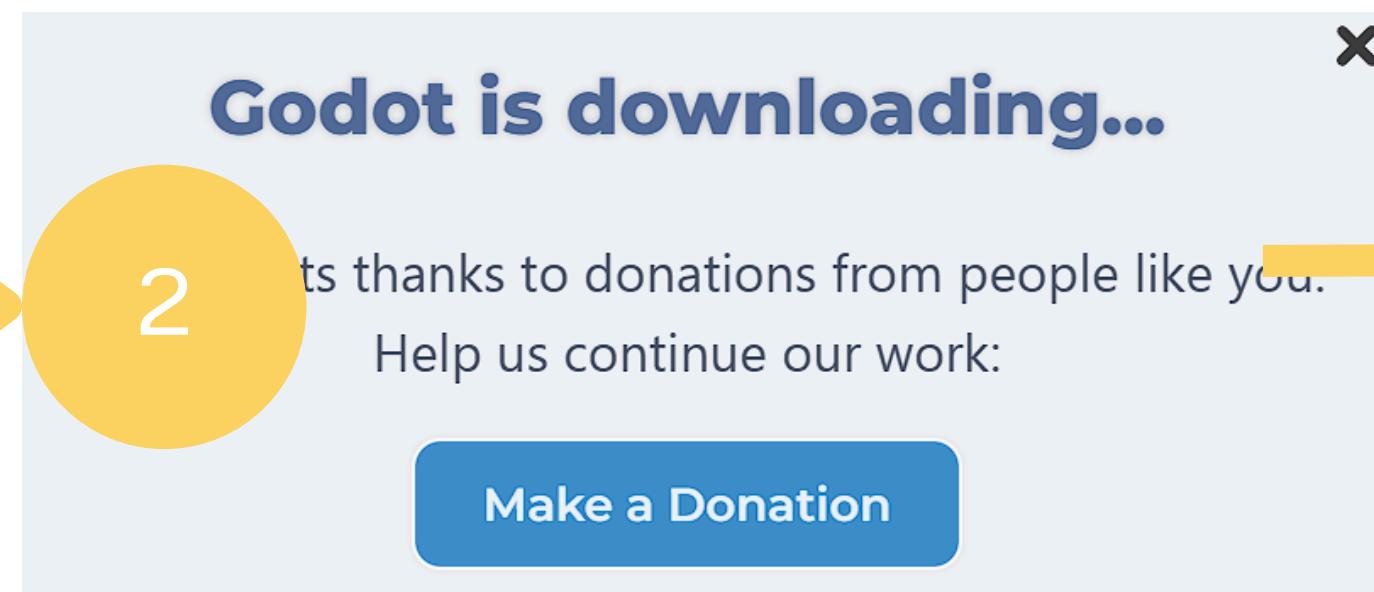
ขั้นตอนที่ 1 เราสามารถค้นหาผ่านเว็บ
браузเซอร์ เช่น Edge หรือ Chrome ด้วยคำว่า
“Godot Engine” เมื่อพบแล้วให้เข้าไปที่ Link
ของเว็บไซต์



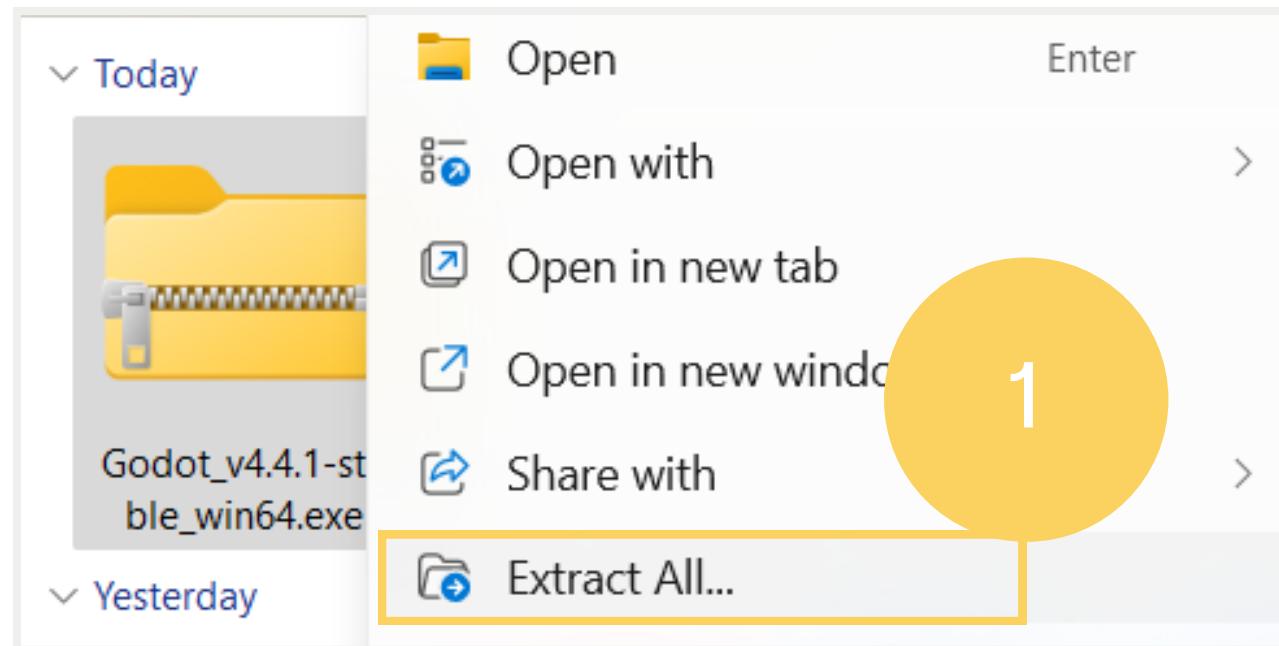
ขั้นตอนที่ 2

เราจะถูกนำทางมาที่เว็บไซต์ <https://godotengine.org/> ซึ่งจะมีเมนูให้เราดาวน์โหลดได้ ให้เลือกเวอร์ชันปัจจุบัน

(ในเอกสารอาจจะมีเลขเวอร์ชันที่แตกต่าง แต่หน้าจอการทำงานไม่แตกต่างกันมาก)



เราจะได้ไฟล์ zip ของตัวโปรเจ็คมาไว้ในเครื่อง

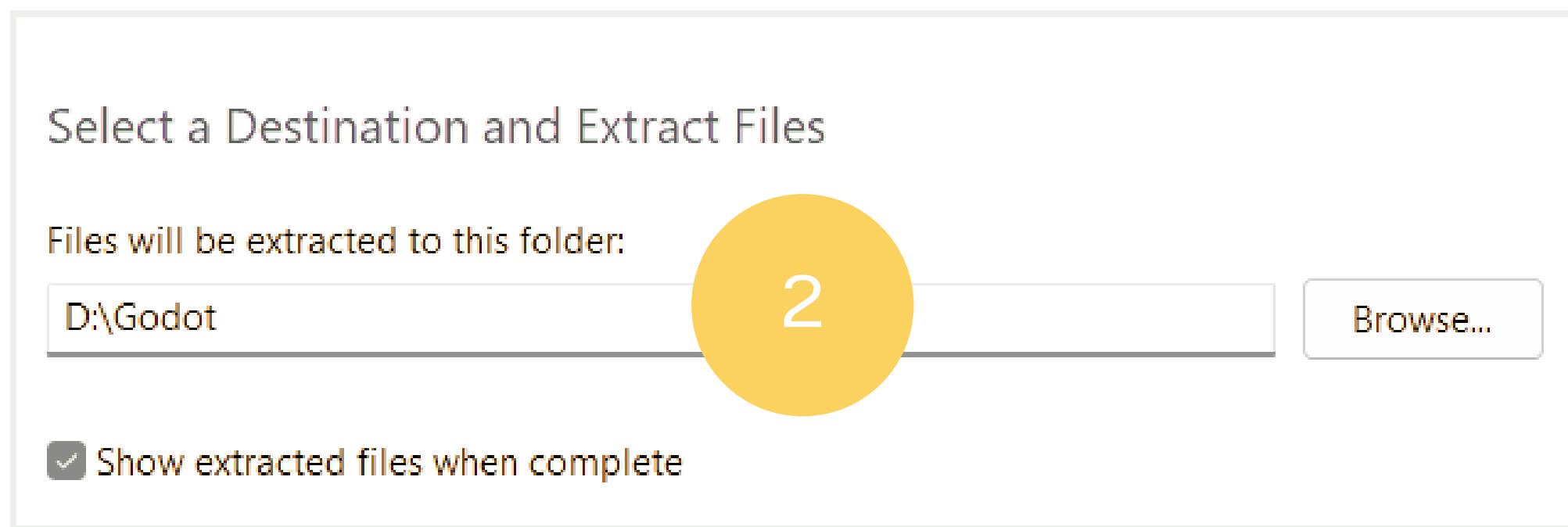


ขั้นตอนที่ 3

ไปกี่ตำแหน่งของไฟล์ที่เราดาวน์โหลด หลังจากนั้นให้ทำการคลิกขวาแล้วเลือก

Extract All

เพื่อแยกไฟล์ zip ที่ดาวน์โหลดมาให้เป็น Folder เมื่อได้ Folder แล้วให้ย้าย Folder ไปยัง Drive หรือตำแหน่งที่เราต้องการจัดเก็บโปรแกรม

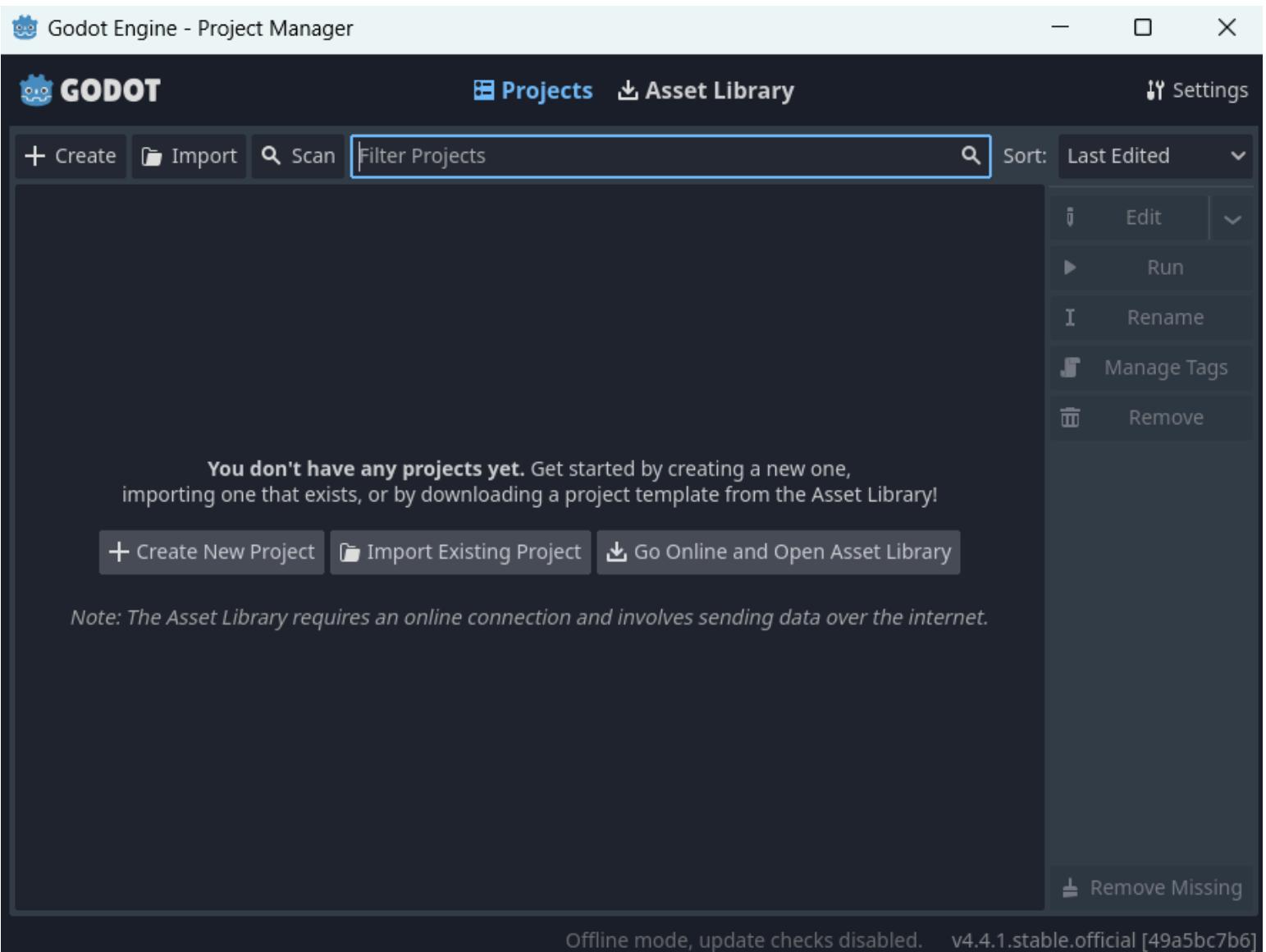


ตัวอย่างในบทเรียนเราจัดเก็บ Folder ตอน Extract ไปไว้ที่ D:\Godot ซึ่งผู้เรียนจะสามารถกำหนดตามหรือจะจัดเก็บตำแหน่งใหม่ก็ได้ตามแต่สะดวก

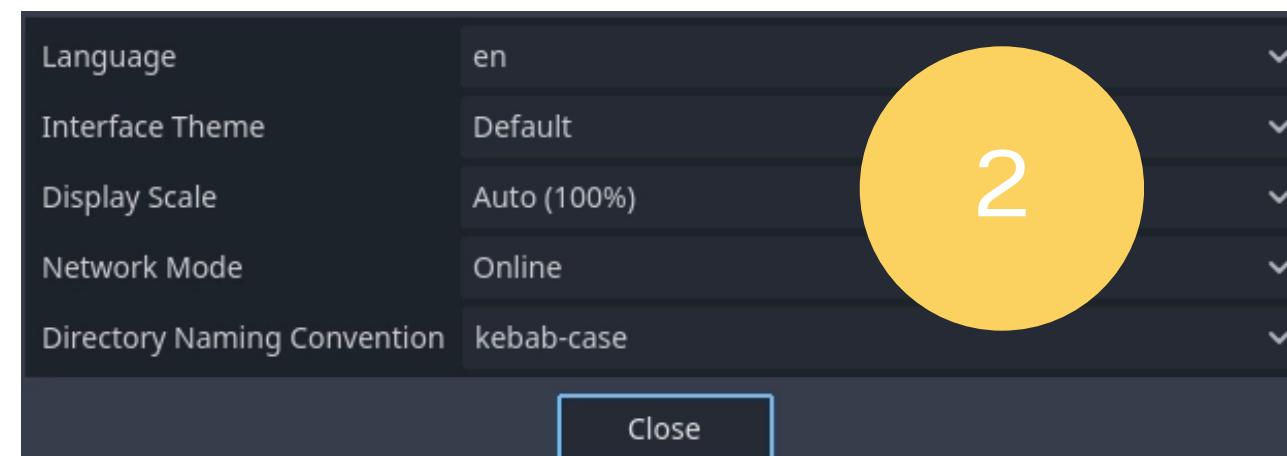
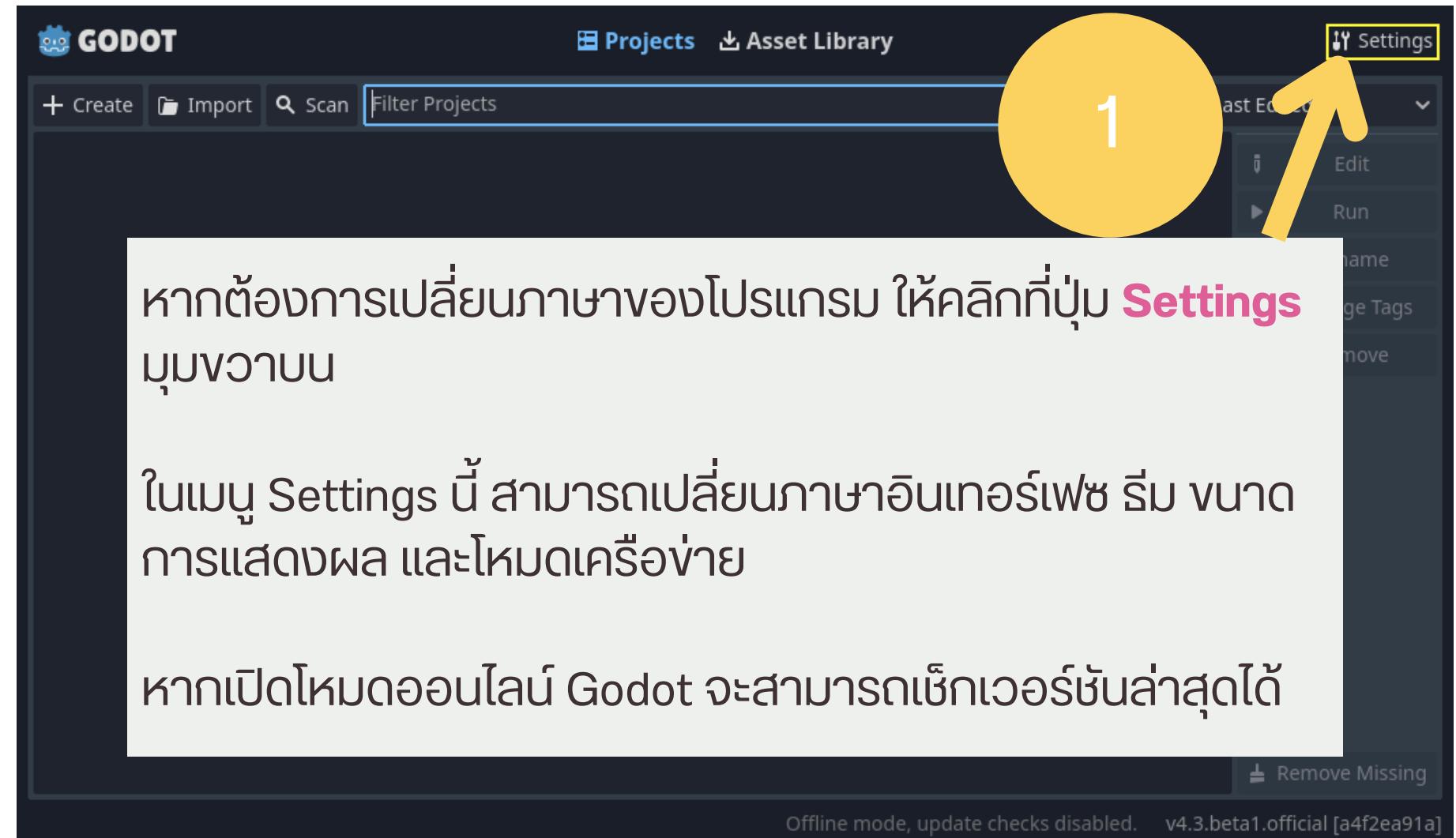


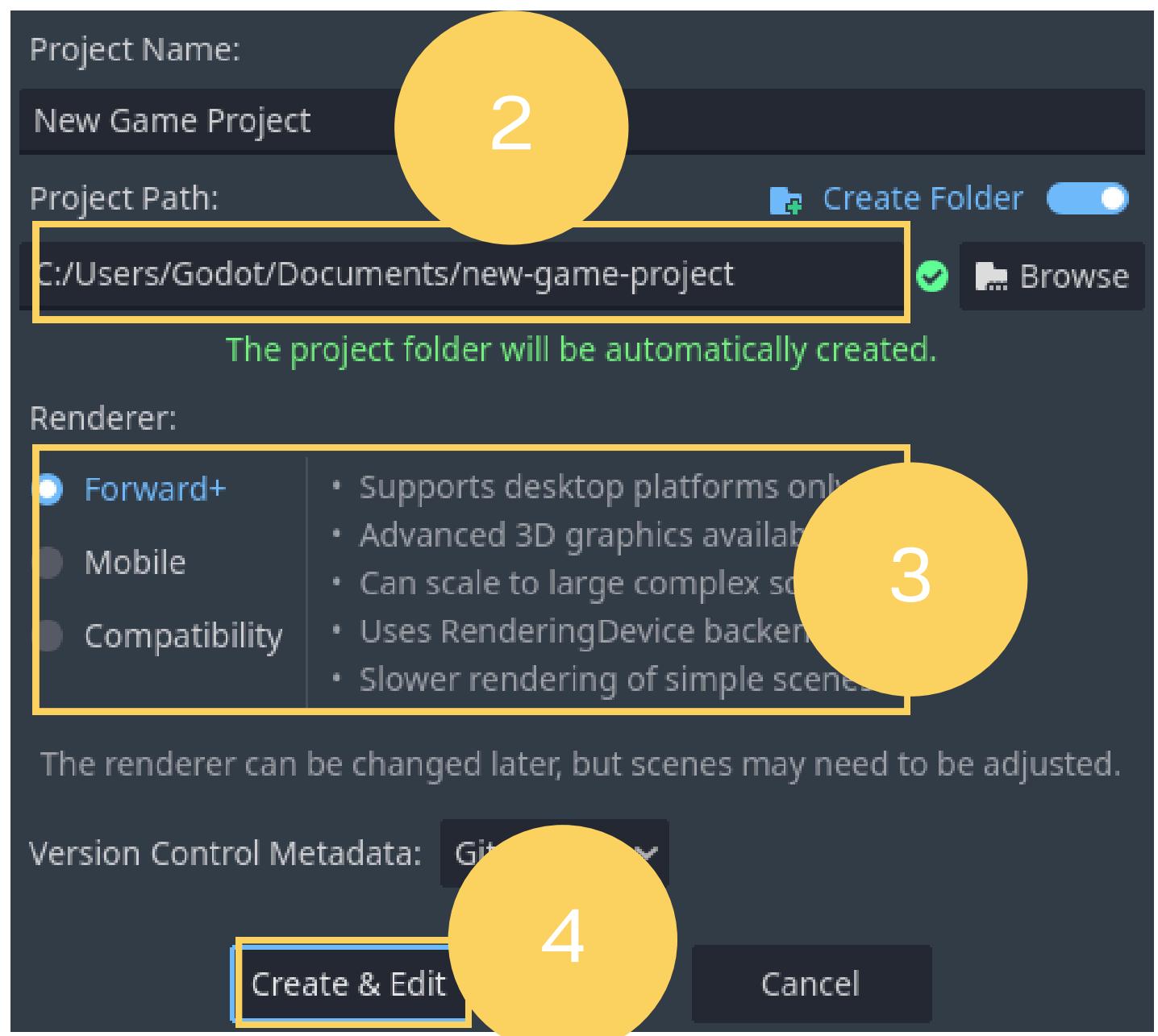
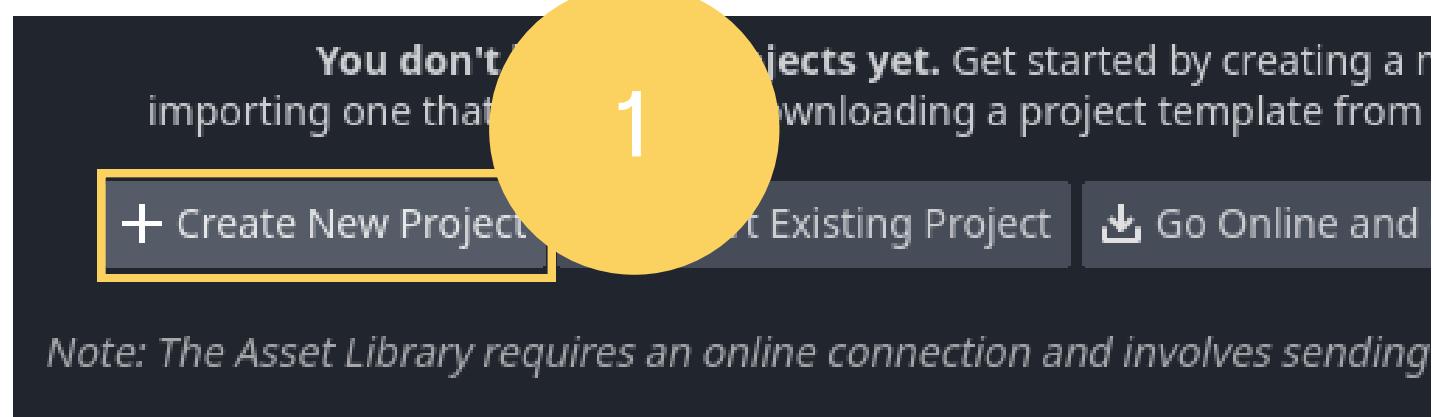
ขั้นตอนที่ 4

หากว่าทุกอย่างพร้อมแล้วทำการเปิดโปรแกรม Godot Engine ตัว Stable ขึ้นมาดังตัวอย่าง



ขั้นตอนที่ 5 หน้าจอการทำงาน Project Manager
เมื่อเปิดโปรแกรม Godot หน้าต่างแรกก็จะเห็นคือ Project Manager ซึ่งใช้ในการ สร้าง ลบ นำเข้า หรือเปิดเล่นโปรเจกต์ เกม





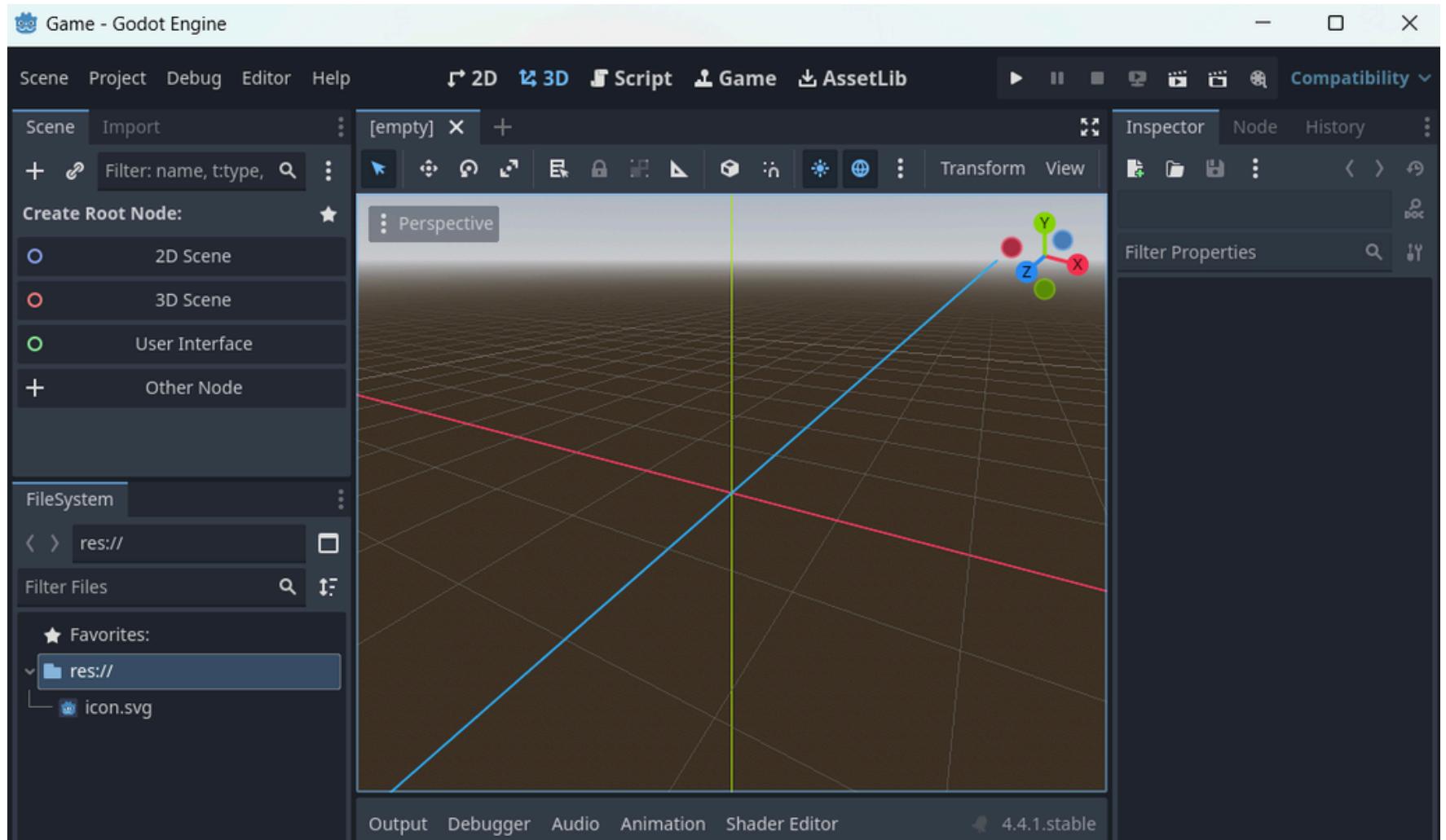
ขั้นตอนที่ 6 การสร้างและนำเข้าโปรเจกต์

Creating and importing projects

การสร้างโปรเจกต์ใหม่:

- คลิกปุ่ม Create ที่มุมบนซ้ายของหน้าต่าง
- ตั้งชื่อโปรเจกต์ จากนั้นคลิกปุ่ม Browse เพื่อเปิดหน้าต่างเลือกโฟลเดอร์ และเลือกโฟลเดอร์ว่างบนคอมพิวเตอร์เพื่อใช้เก็บไฟล์
 - หรือสามารถเปิดใช้ตัวเลือก Create Folder เพื่อให้ระบบสร้างโฟลเดอร์อยู่ใหม่โดยอัตโนมัติตามชื่อโปรเจกต์ โดยจะใช้รูปแบบการตั้งชื่อโฟลเดอร์ตามที่ตั้งค่าไว้ใน Settings
 - หากโฟลเดอร์นั้นว่าง จะมีเครื่องหมายถูกสีเขียวแสดงทางด้านขวา
- เลือกตัวเรนเดอร์ (Renderer) ที่ต้องการใช้ (สามารถเปลี่ยนได้ภายหลัง)
- คลิกปุ่ม Create & Edit เพื่อสร้างโปรเจกต์และเปิดในตัวแก้ไข (Editor)





ขั้นตอนที่ 7 การใช้งาน Editor

Using the Editor

หน้าจอกำหนดราก่อนทำงานของ Godot Engine นั้นจะมีส่วนสำคัญก็คือ FileSystem หรือการจัดการไฟล์ต่างๆ ซึ่งเราสามารถนำเข้า Import ไฟล์ตั้งแต่

- กราฟิก 2 มิติ เช่น ภาพ SpriteSheet, Picture, Image ต่างๆ
- กราฟิก 3 มิติ เช่น ไฟล์แบบจำลอง 3 มิติอย่าง .FBX, .GLB, .OBJ มาใช้ได้
- ไฟล์มีเสียง เช่นไฟล์ mp3, mp4 หรือ ogg สำหรับเล่นเสียง ไปจนถึงภาพเคลื่อนไหว

ขั้นตอนที่ 8 การนำเข้ากราฟิกสำหรับเกม Game Assets

ในบทเรียนนี้เราจะใช้กราฟิกตัวละครจากเว็บไซต์ itch.io ซึ่งเป็นแพลตฟอร์ม Community เกมอันดี (เกนอสระ) ที่ค่อนข้างใหญ่ และมีกราฟิกและเกมที่น่าสนใจให้เราสามารถเข้าไปสร้างรายได้ ก็งสร้างเกม หรือสร้างกราฟิกสำหรับเกม

<https://itch.io/game-assets>
(ต้องสมัครสมาชิก)



The screenshot shows the itch.io website interface with three numbered steps:

1. Click on the "Browse" button in the top navigation bar.
2. Click on the "Assets" tab in the main navigation menu.
3. Use the "Free" filter option in the "Price" dropdown menu to find free assets.

A yellow callout bubble highlights step 3 with the text: "เรามาการ Filter ให้เป็นรูปแบบของ Assets / Free ได้". Another yellow callout bubble highlights the "Free" filter button with the text: "กดที่เมนู Browse → Assets".

The main content area displays various game asset packages, such as "Kenney Game Assets All-in-1", "Combat RPG - 2000K Characters - Animations - Backgrounds - and more!", and "Modern Exteriors".

ขั้นตอนที่ 9 การเตรียมกราฟิก Assets Management System

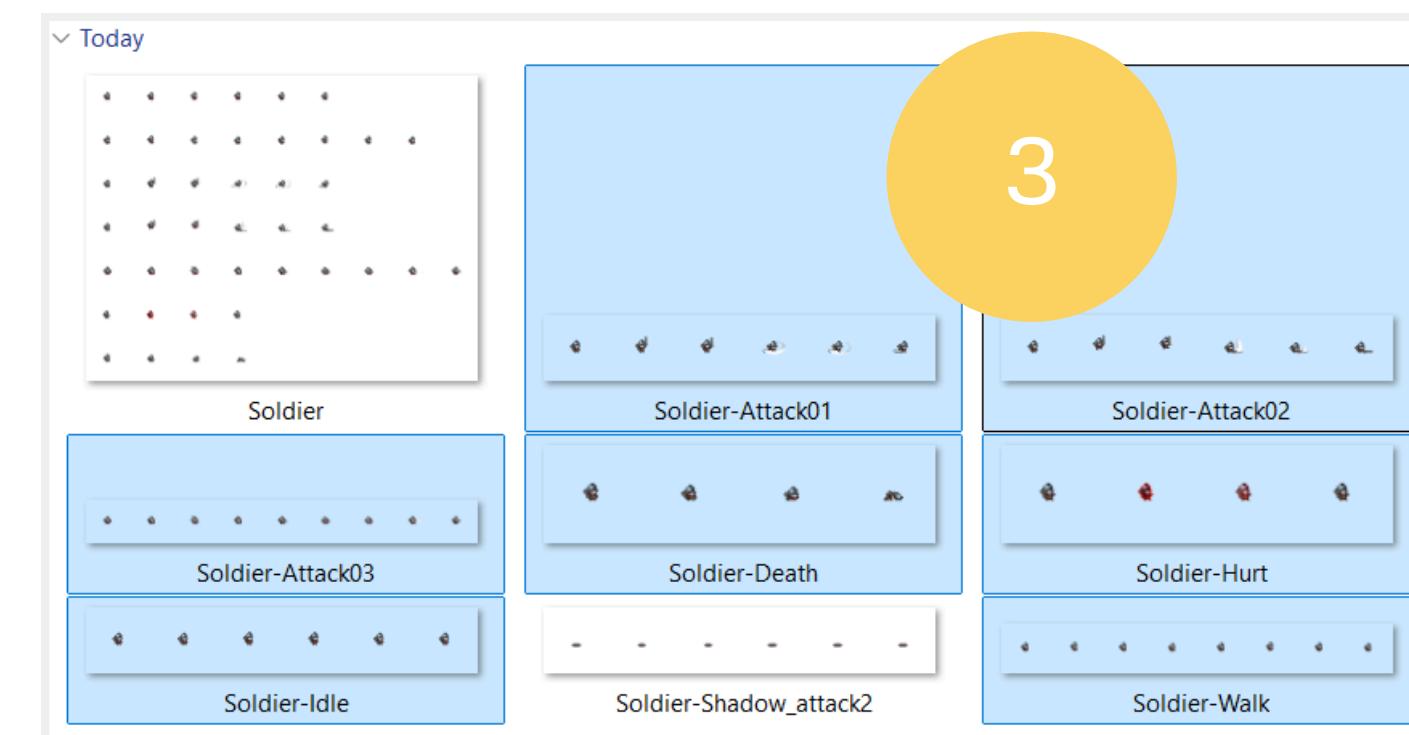
เพื่อความรวดเร็วในการเรียนรู้ ในบทเรียนนี้หากต้องการลองใช้งาน Godot Engine อย่างไวๆ เราสามารถใช้ Assets ที่เตรียมได้ โดยการเข้าดาวน์โหลด กับ **Link Google Drive** ด้านล่าง

The screenshot shows a list of files in a Google Drive folder. The first file, "Tiny RPG Character Asset Pack v1.03b -Free Soldier&O...", is highlighted with a yellow border. A large yellow circle with the number "1" is positioned over the third item in the list.

<https://drive.google.com/drive/folders/1uUa4wWpX1V-VACZXhtTfdCieVjjCn-qJ?usp=sharing>

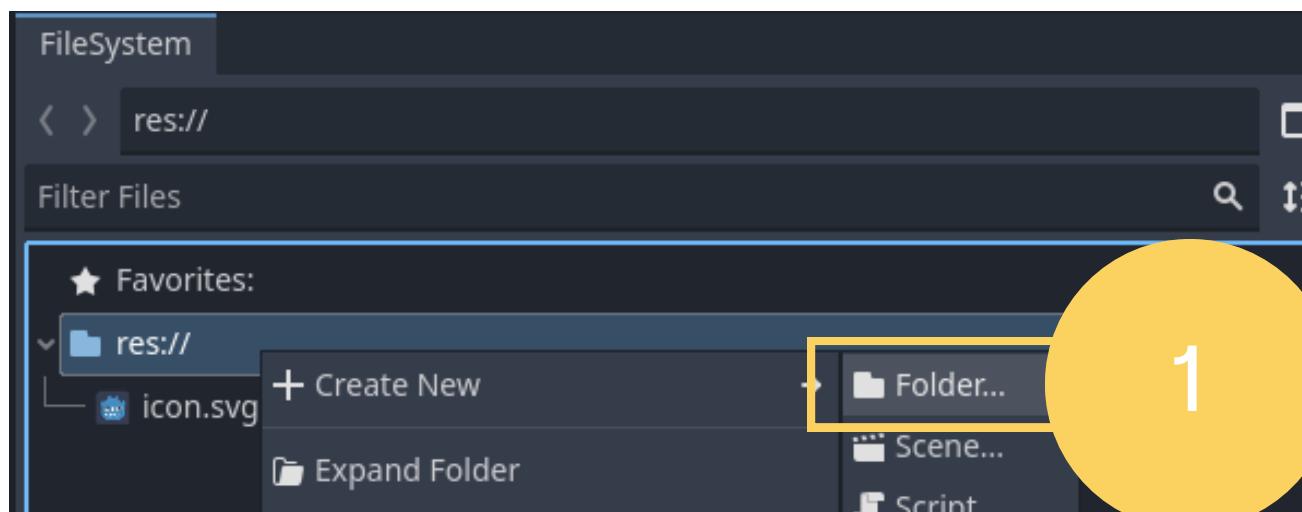
The screenshot shows the Godot Asset Management System interface. On the left, there is a tree view with the following structure:

- Name
- Today
 - Arrow(Projectile)
 - Aseprite file
 - Characters(100x100) **2**



เราจะเลือก Sprite กราฟิกที่

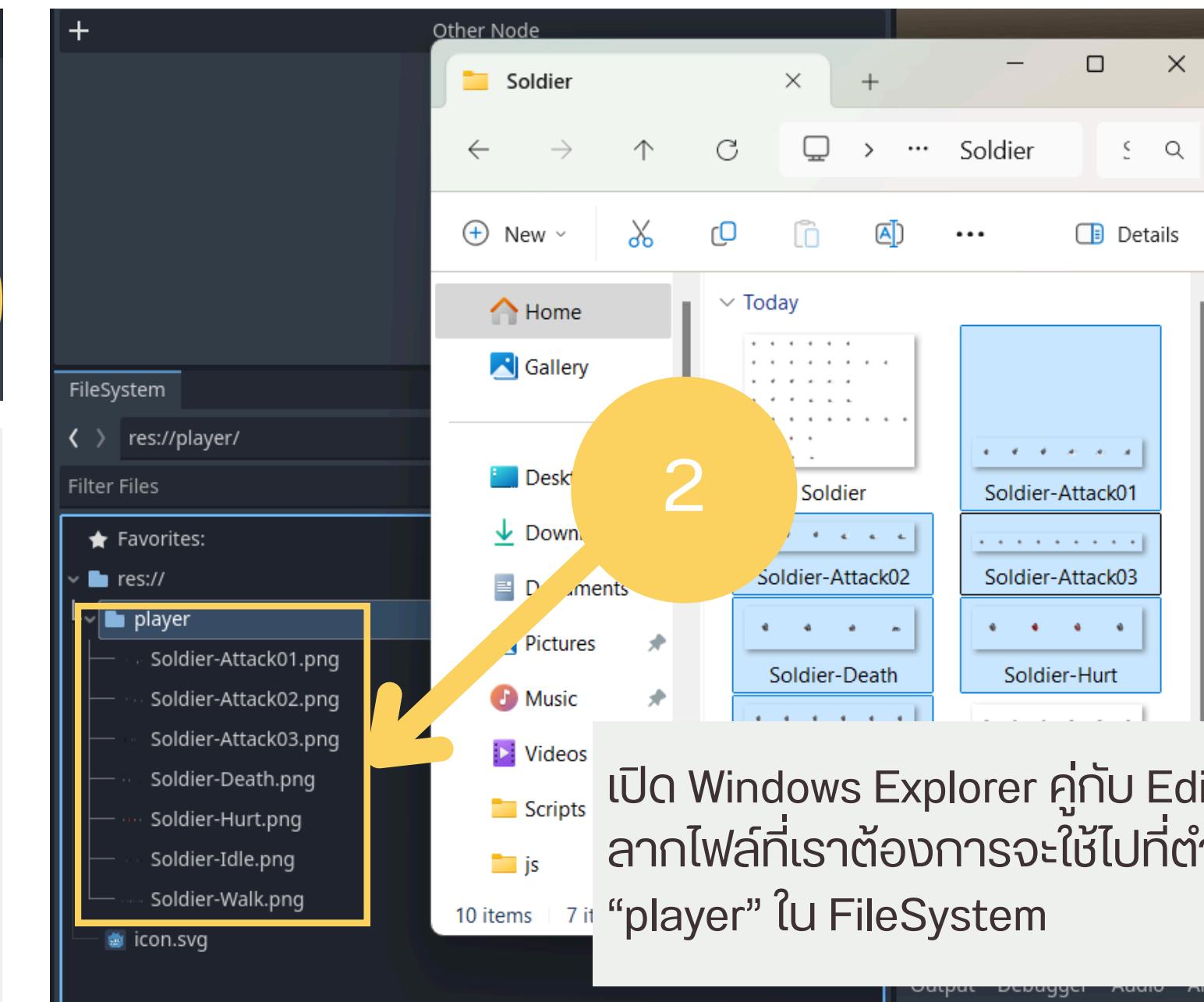
- Soldier-Idle (**สำคัญ**)
- Soldier-Attack01 (**สำคัญ**)
- Soldier-Attack02
- Soldier-Attack03
- Soldier-Hurt
- Soldier-Walk (**สำคัญ**)



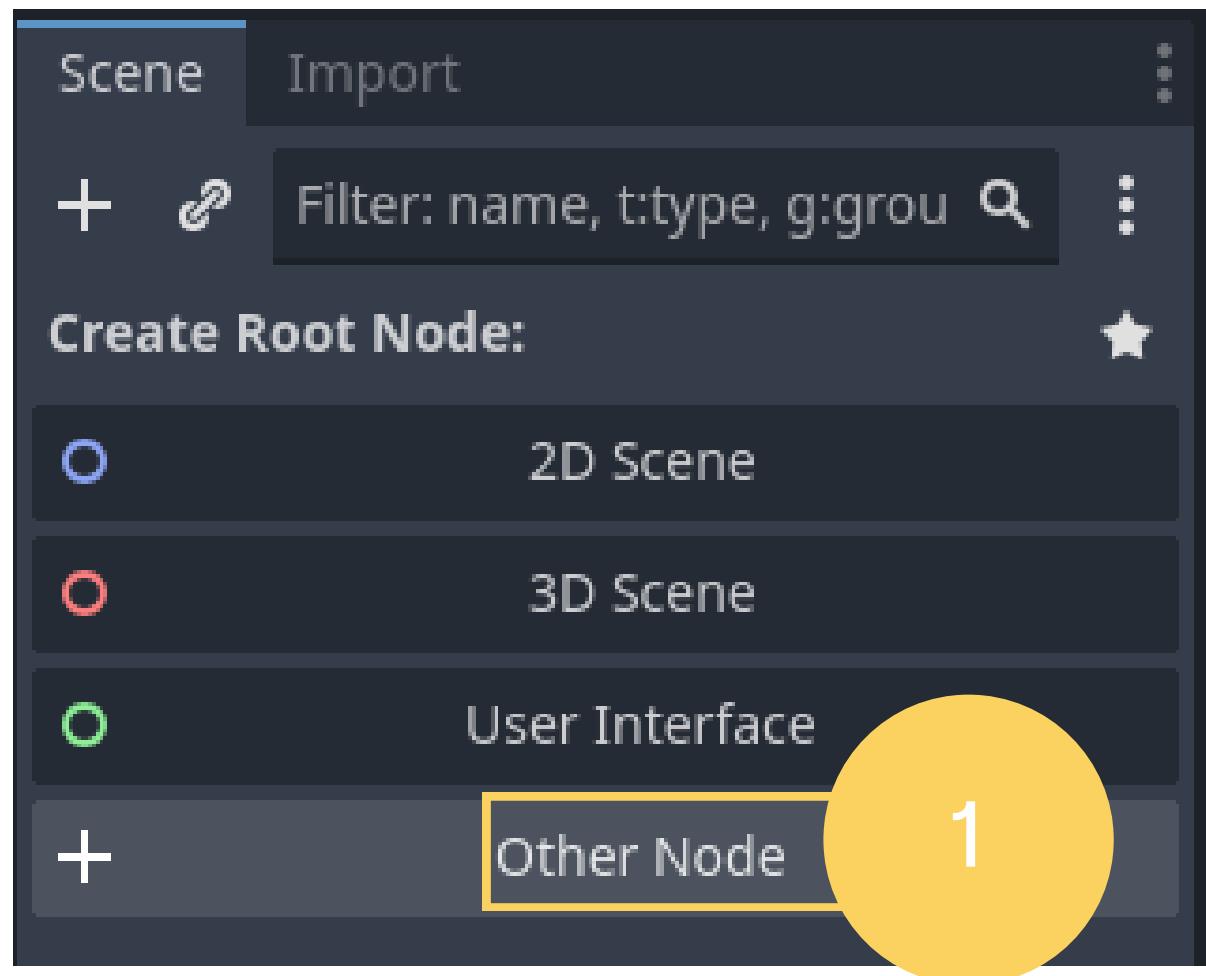
ขั้นตอนที่ 10 สร้าง Folder ใหม่ใน FileSystem

Create new folder

กลับไปที่ Editor ของ Godot Engine ให้ไปที่ FileSystem Panel หลังจากนั้น **คลิกขวา** ที่ res:// หลังจากนั้นเลือก Create New Folder ตั้งชื่อว่า **player** (ตัวเด็กหมด)

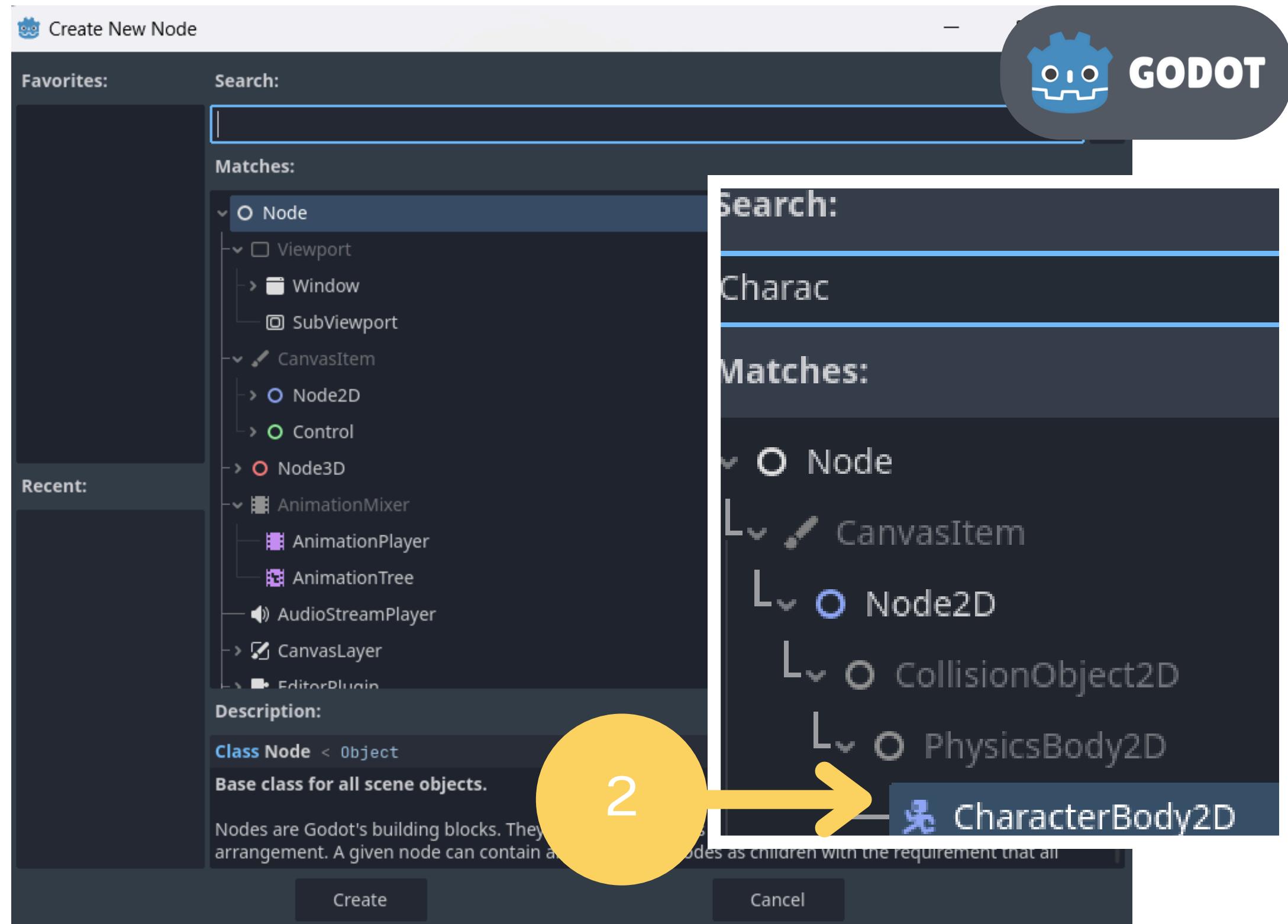


เปิด Windows Explorer คู่กับ Editor หลังจากนั้นให้
ลากไฟล์ที่เราต้องการจะใช้ไปที่ตำแหน่ง folder
“player” ใน FileSystem

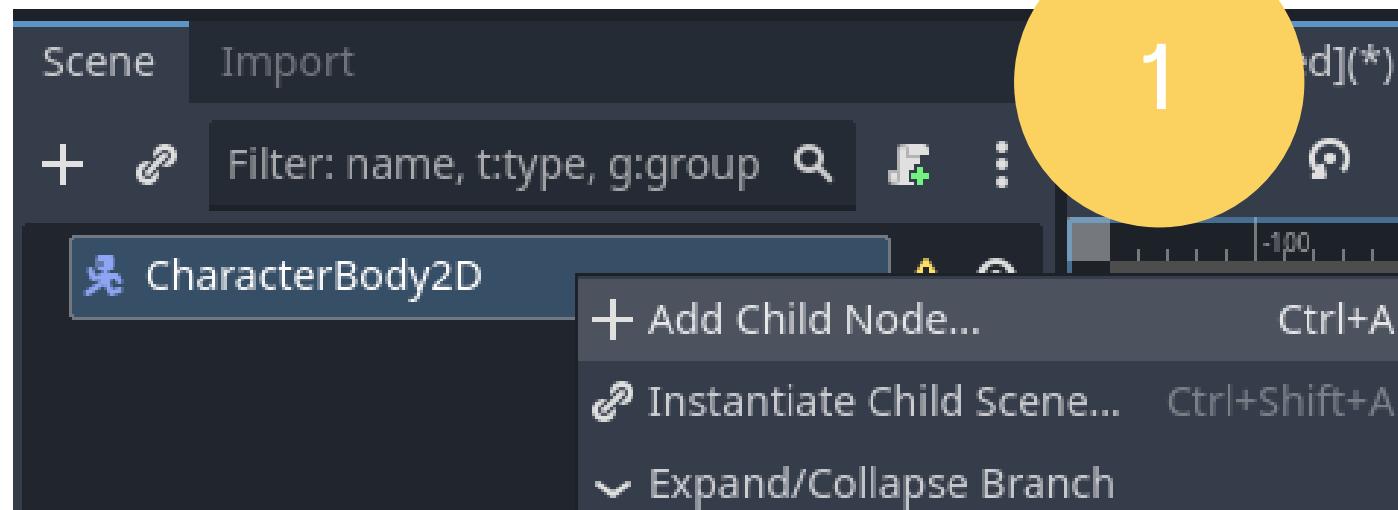


ขั้นตอนที่ 11 การนำเข้าตัวละคร และสร้าง scene Setup Character

ไปที่ส่วน Panel ที่ชื่อว่า “Scene” ให้เราคลิกที่ Other Node เพื่อจะได้กำหนด Node ของ Element บน Godot ให้เป็นตัวละครของเรา



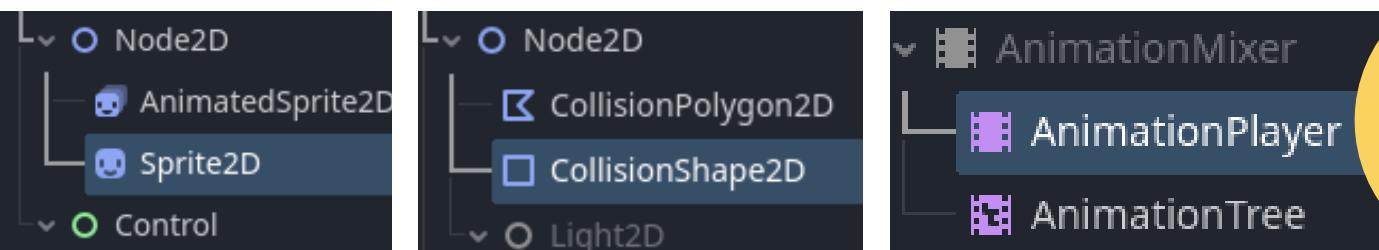
เลือก Node ของเราเป็น CharacterBody2D



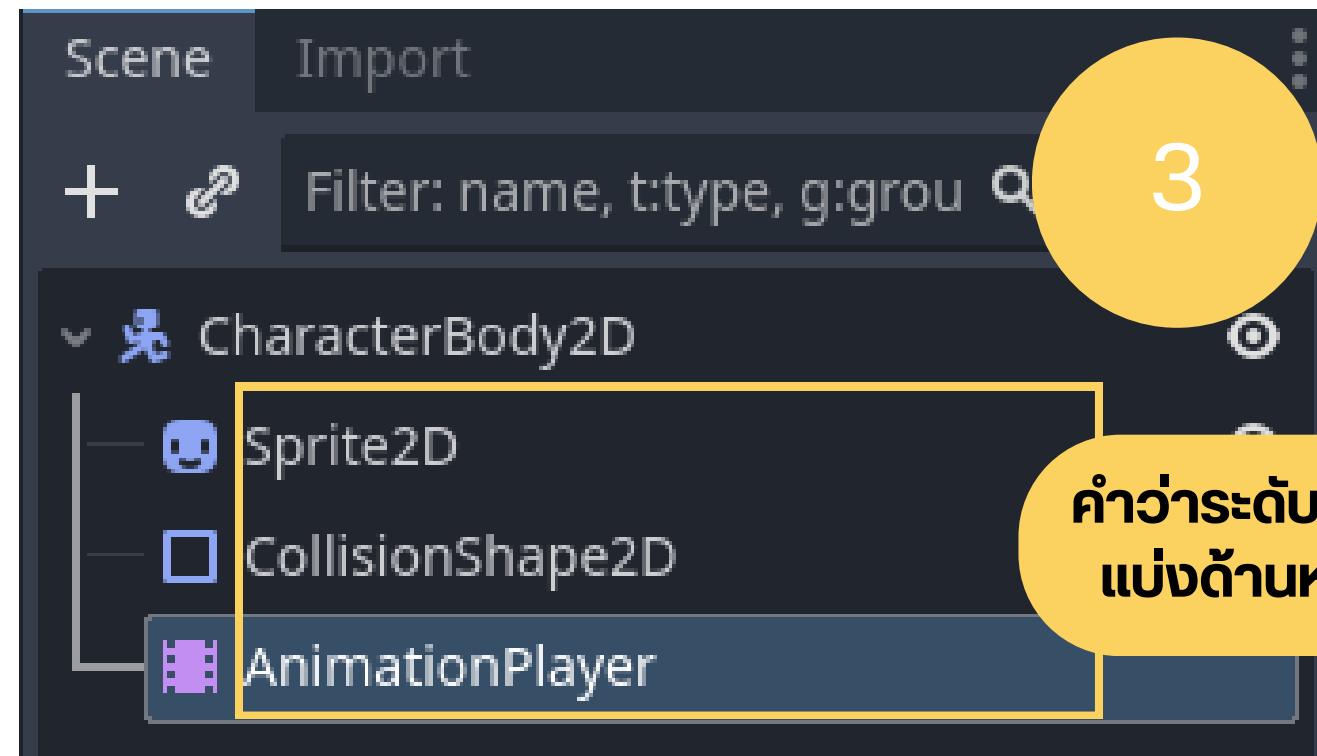
ขั้นตอนที่ 12 การเพิ่ม Child Node หรือโหนดลูก

โครงสร้าง Node ของ Godot Engine บันจะทำงานแบบ Hierarchy ลำดับขั้นคือการสร้าง Parent และ Child ในตัวอย่าง

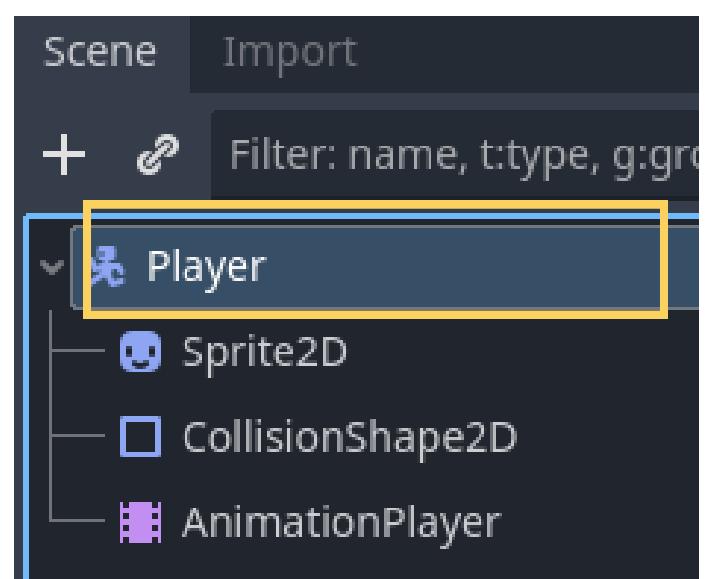
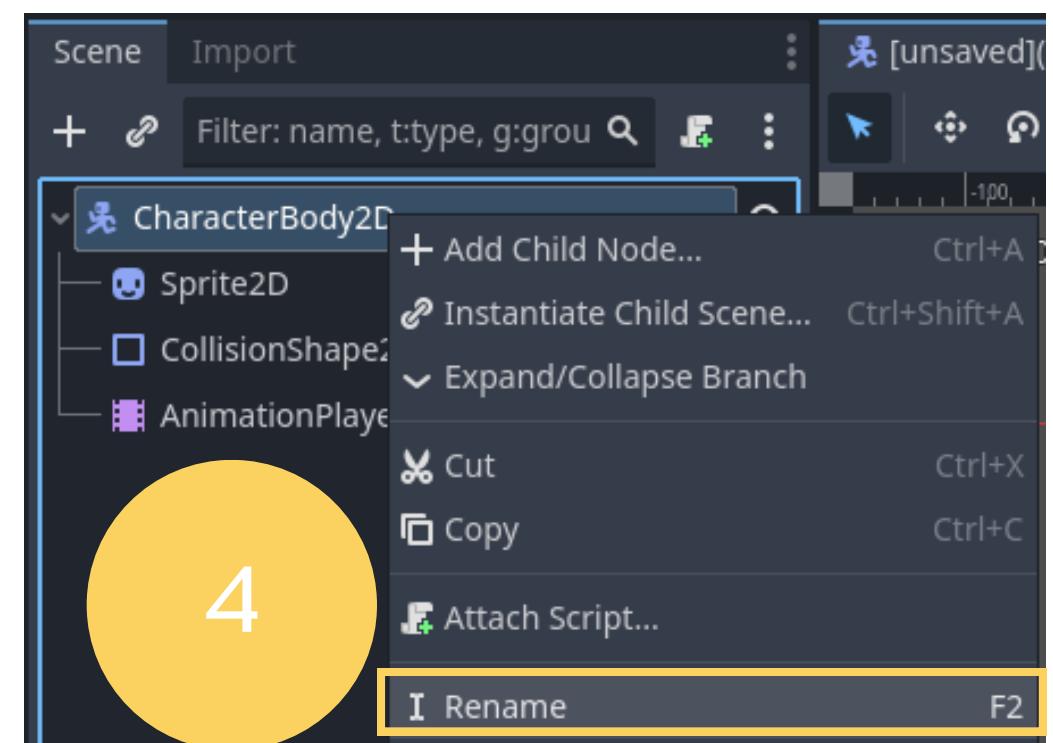
เริ่บโดยการคลิกขวาที่ CharacterBody2D แล้วเพิ่ม
+ Add Child Node

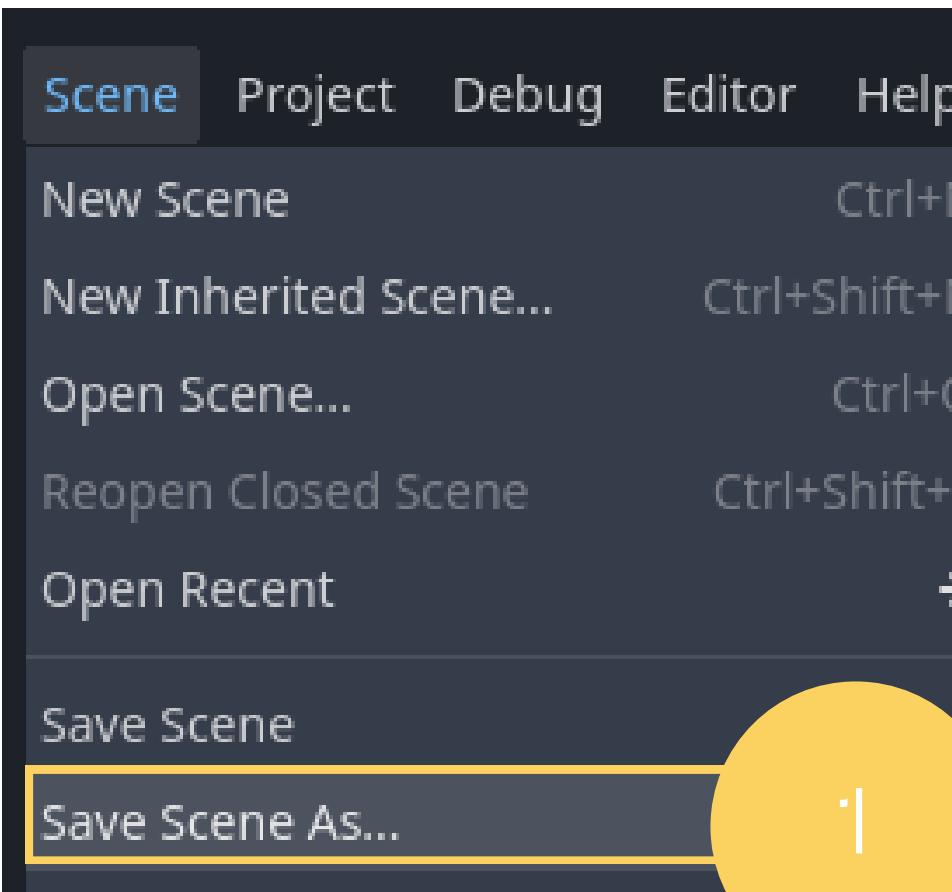


เพิ่ม **Sprite2D, CollisionShape2D, AnimationPlayer**
อยู่เป็นลูกของ CharacterBody2D ในระดับเดียวกัน



เราจะแก้ไข **CharacterBody2D** โดยการเปลี่ยนชื่อของมัน
คลิกขวาแล้ว Rename เป็น **Player**

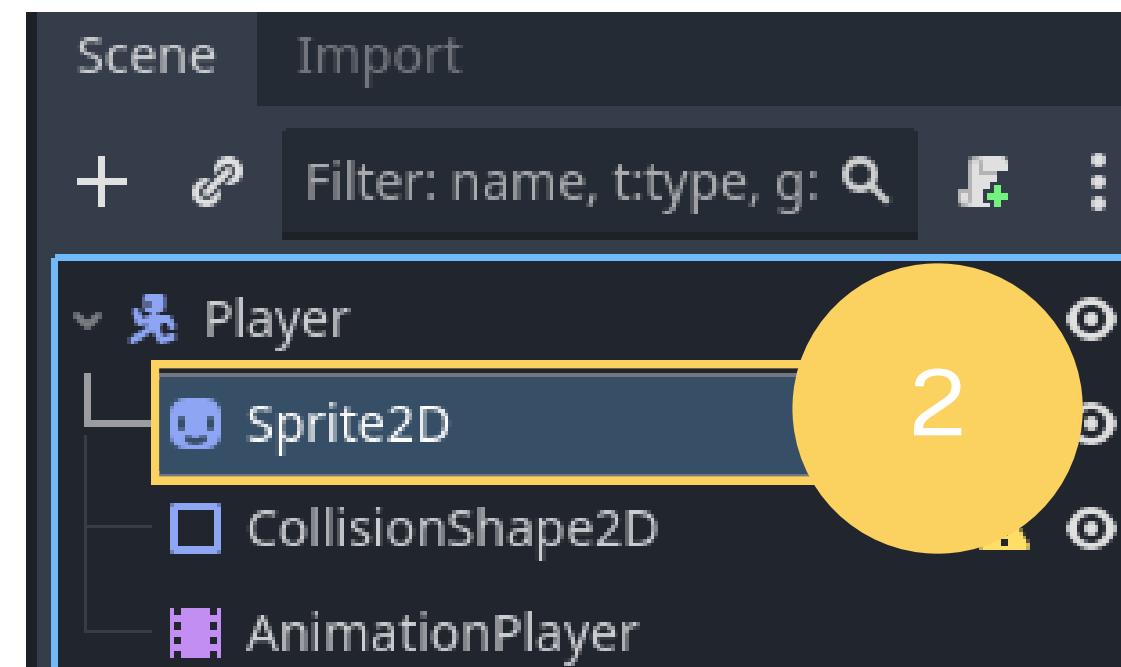




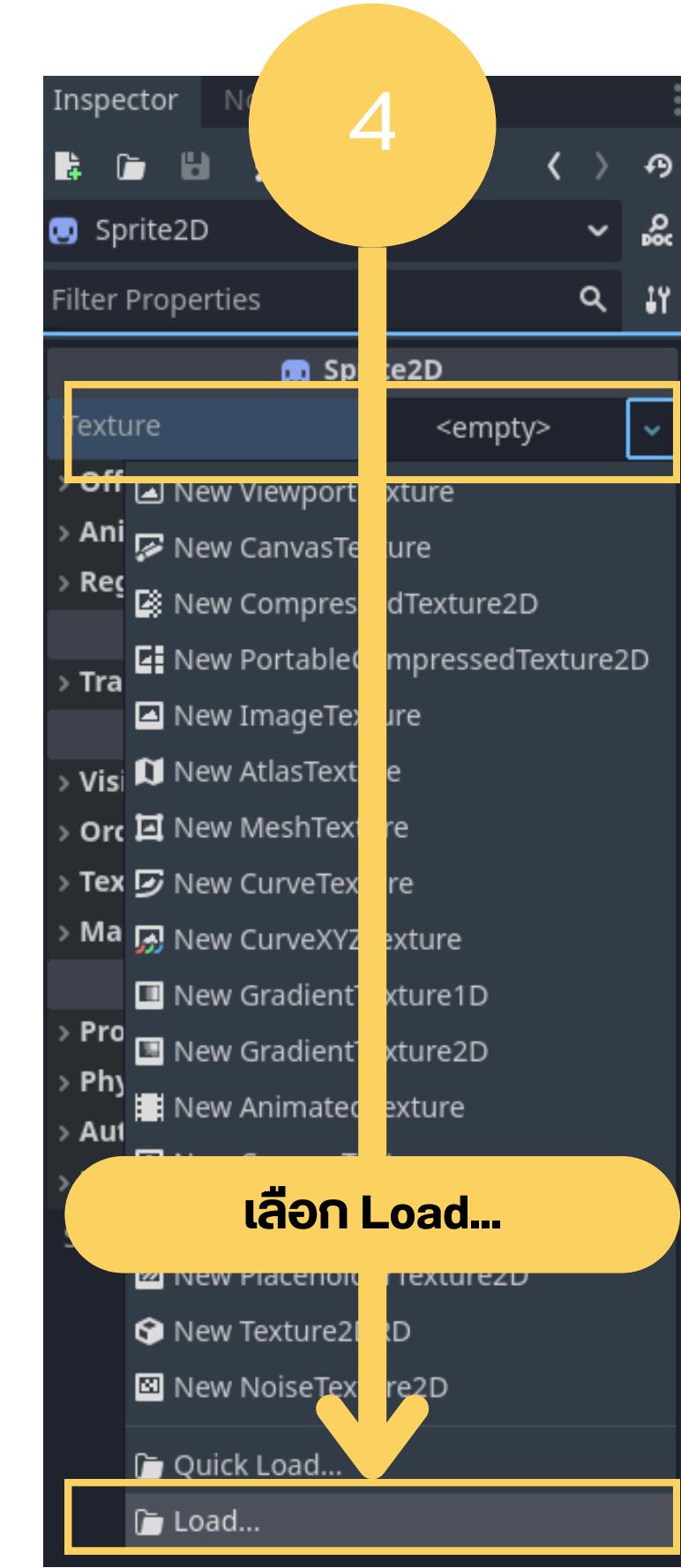
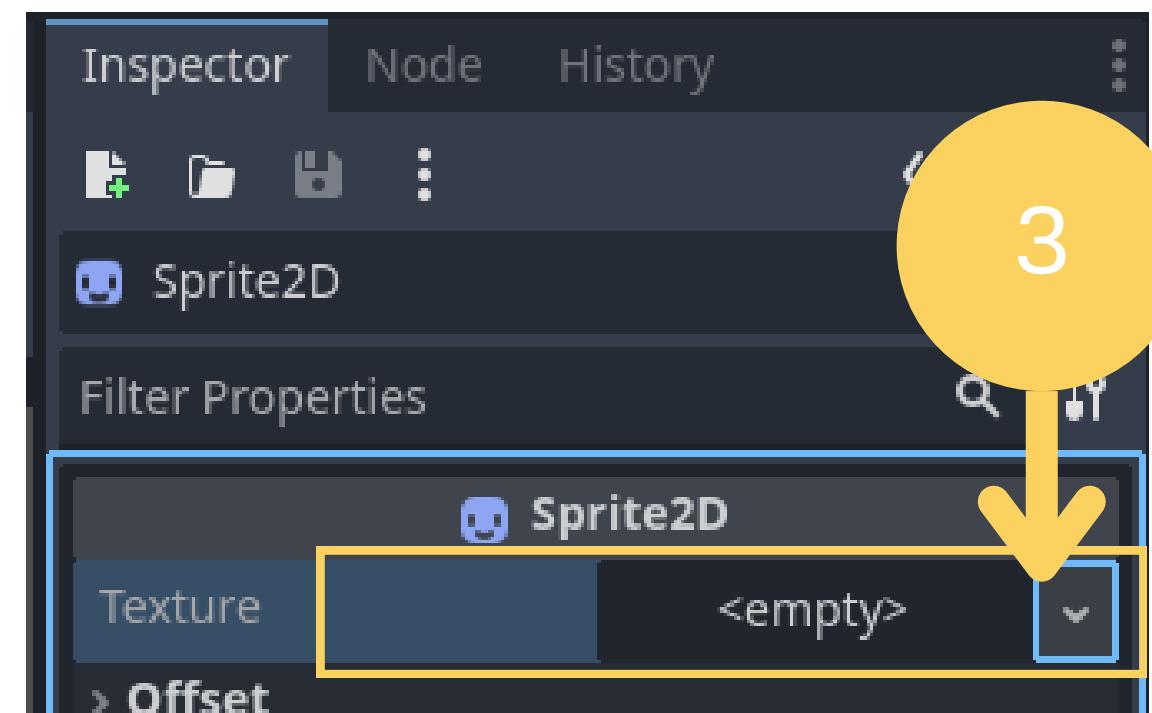
ขั้นตอนที่ 13 บันทึก Scene ใหม่
คลิกที่เมนู Scene หลังจากนั้นเลือกเมนู
ย่ออยค้อ

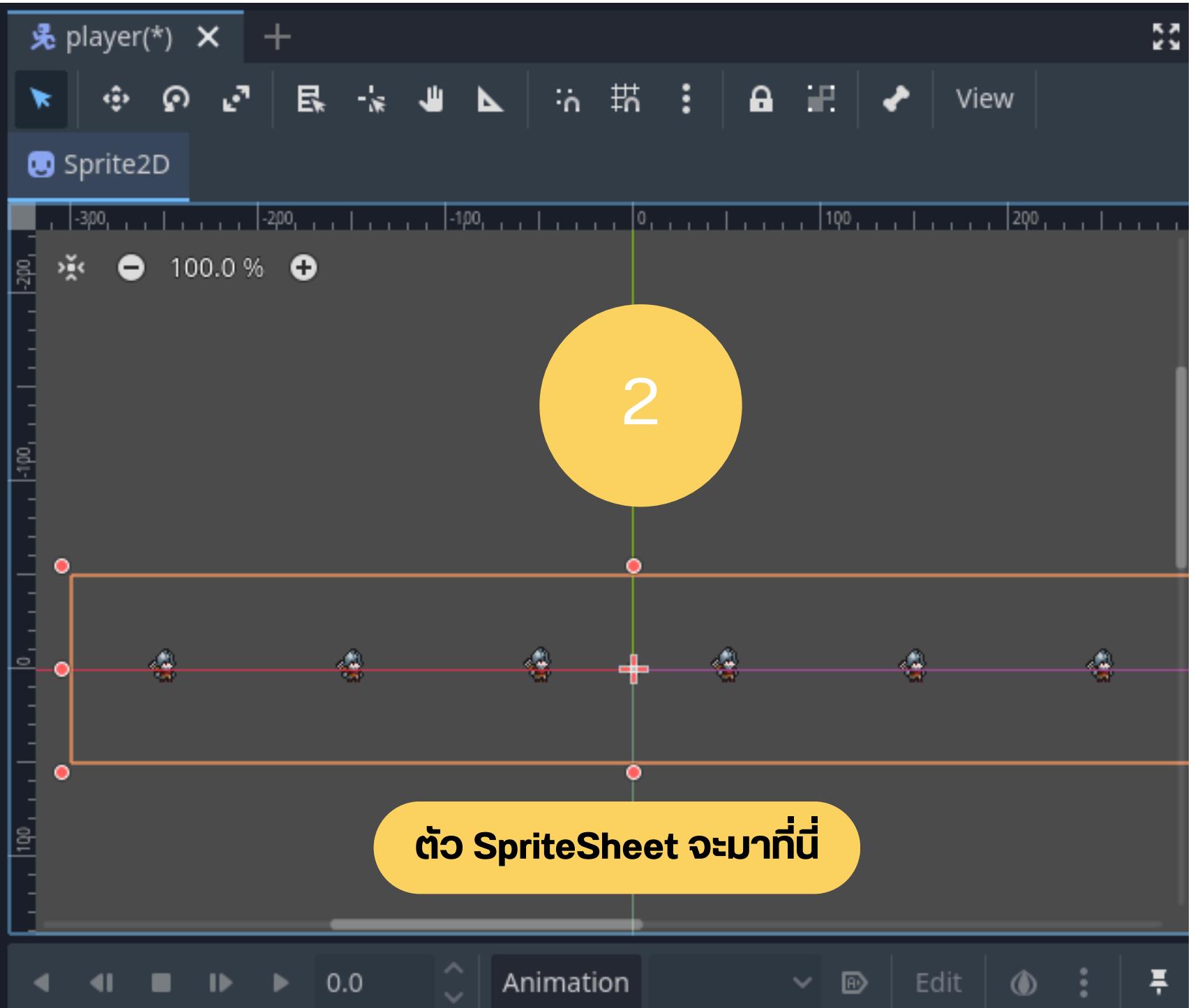
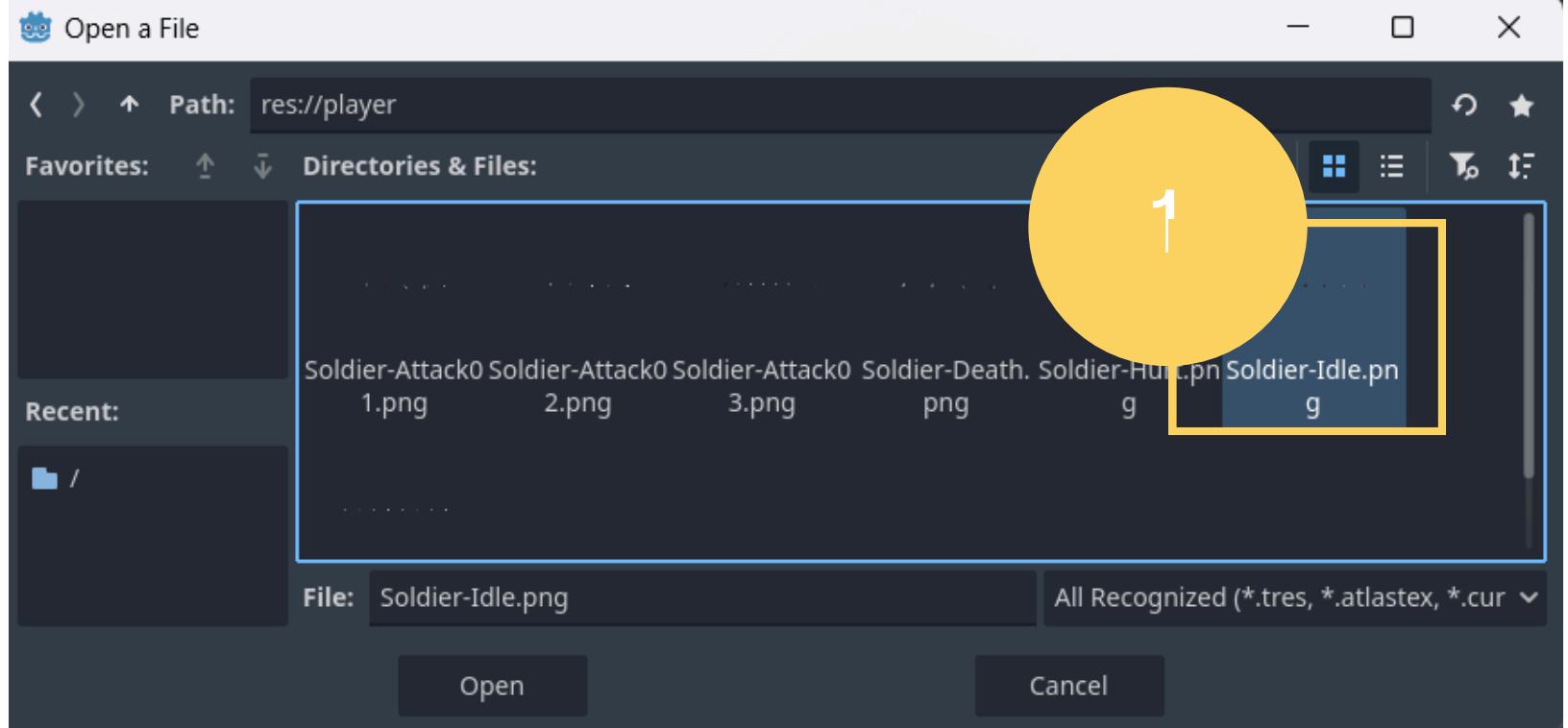
- Save Scene As...
- ตั้งชื่อ Scene ว่า **player.tscn**
- ข้อสังเกตุ player ตัวเล็กหนด

เมื่อบันทึก Scene “**player.tscn**” เสร็จ
เรียบร้อยแล้วให้คลิกที่ **Sprite2D**



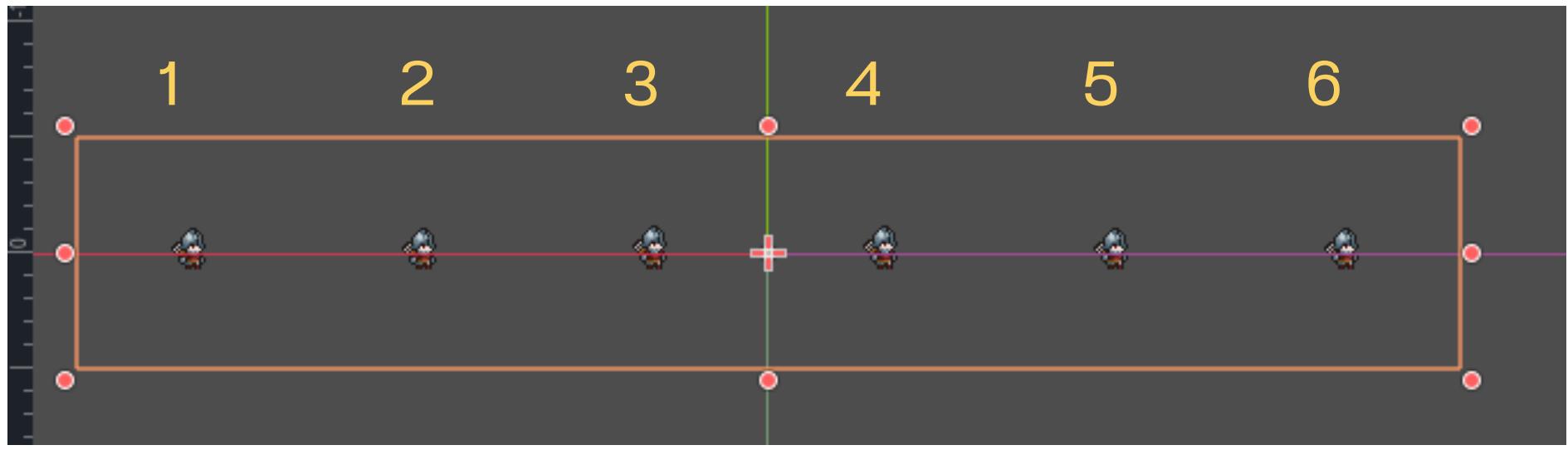
ไปที่ **Inspector** กด Dropdown ที่ **Texture**



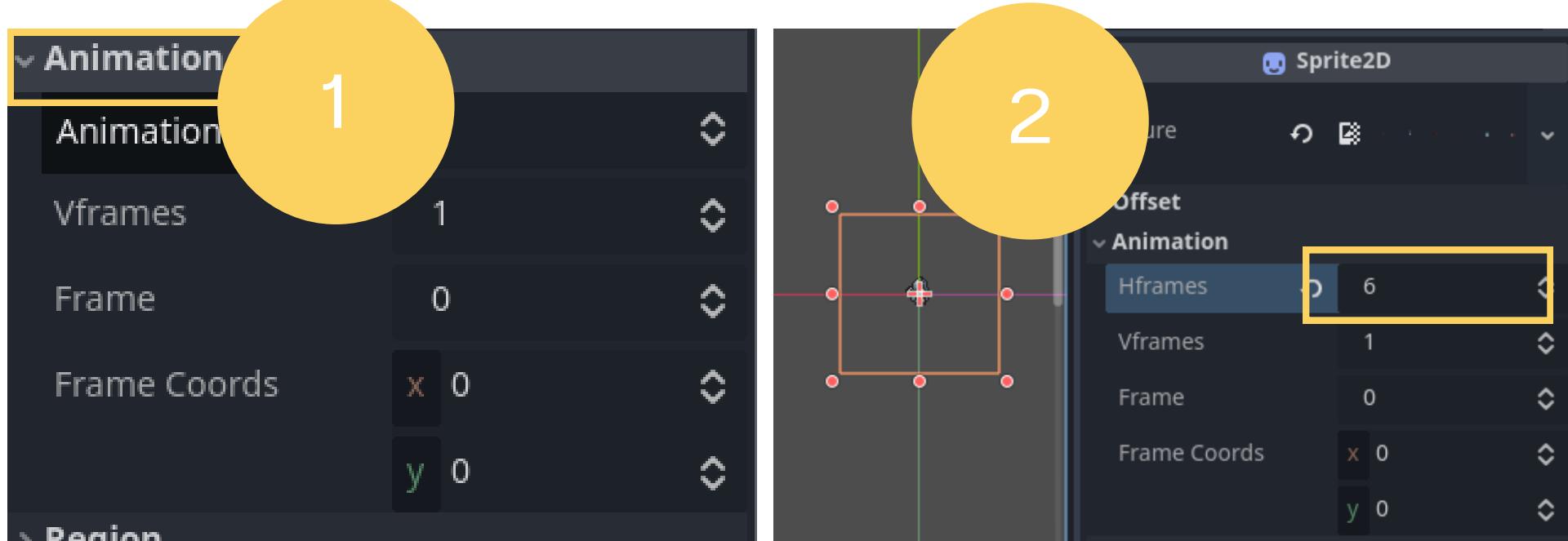


ขั้นตอนที่ 14 การกำหนดอินิเมชันของตัวละคร **Sprite2D and AnimationPlayer**

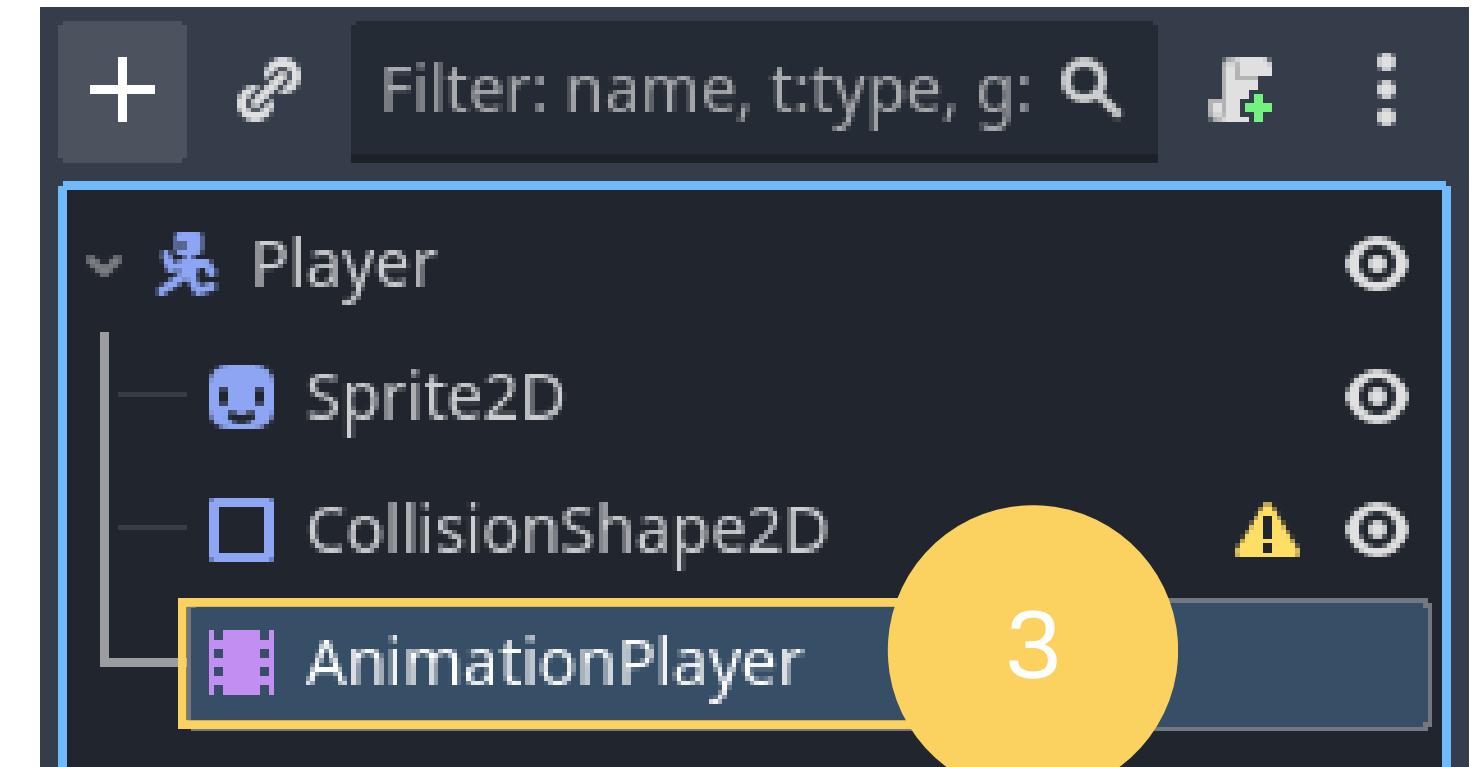
หลังจากกดปุ่ม “Load” เลือกไฟล์ก้ากง Idle ในตัวอย่างคือ Solider-Idle.png เข้าไปไว้บน Scene หากเลือกถูกหน้าจอ Scene ของเราจะปรากฏดังตัวอย่าง คือ ภาพของ SpriteSheet ไปอยู่ใน Panel ของ Editor



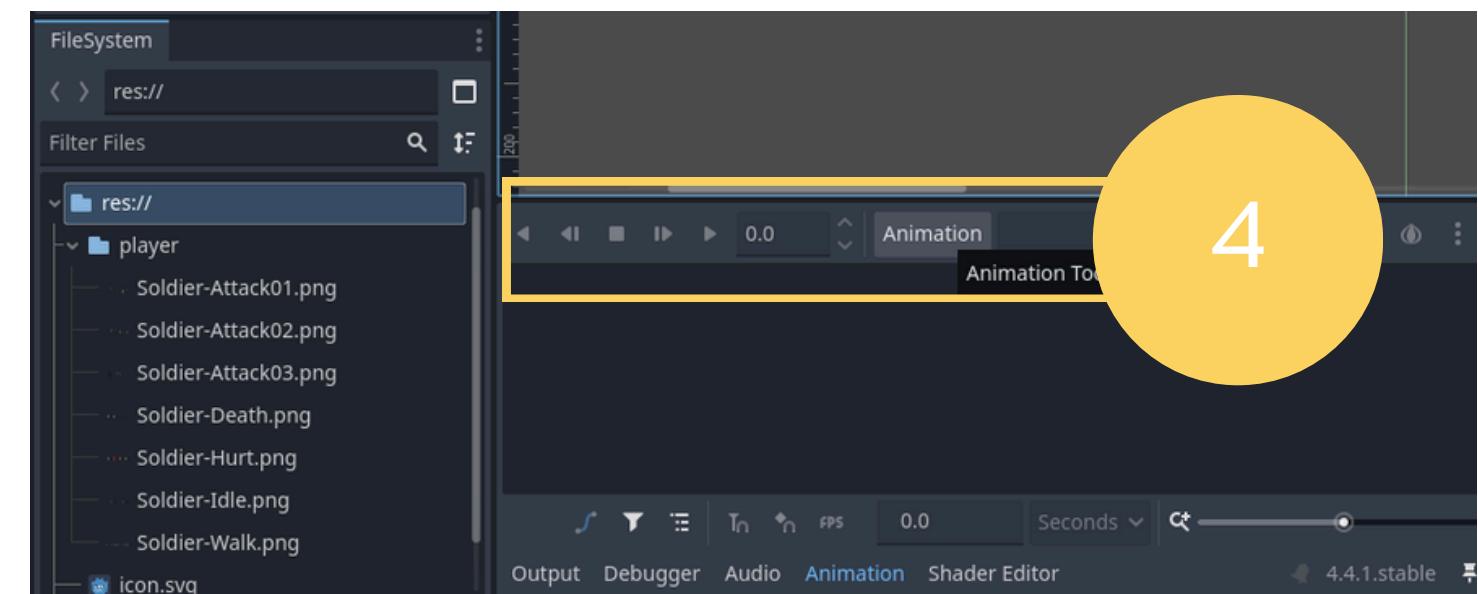
เก็บบิดกีบง่ายของ Godot คือ ให้เราบันจាวนวบ Frame ของ Sprite ได้เลยในตัวอย่างคือ **6 Frames** (Frame คือ ภาพกีบขนาดเท่ากันมาต่อ กัน)



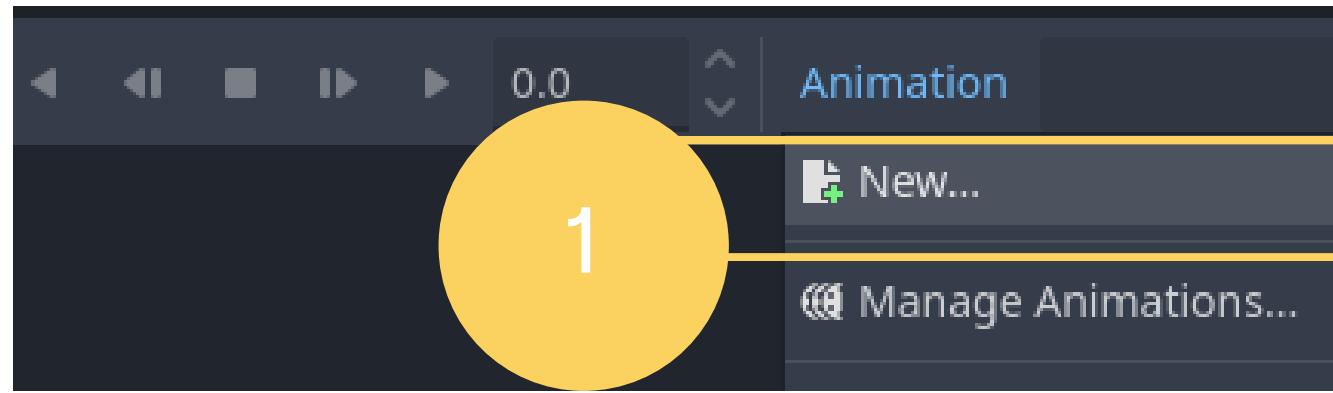
เปิด Sub Panel ของ **Animation** ใน ส่วน **Inspector** หลังจากบันใส่ค่าของ Hframes จาก 1 ให้เป็น 6 (ตามจำนวน Frames ของ SpriteSheet) เราจะพบว่าตัวละครของเราจะถูก切割เป็น 1 Frame ที่มี อีก 5 Frames ถูกซ่อนไว้



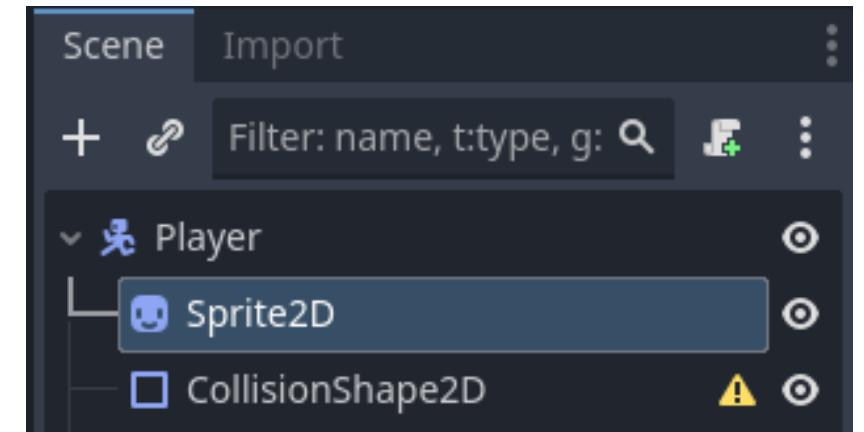
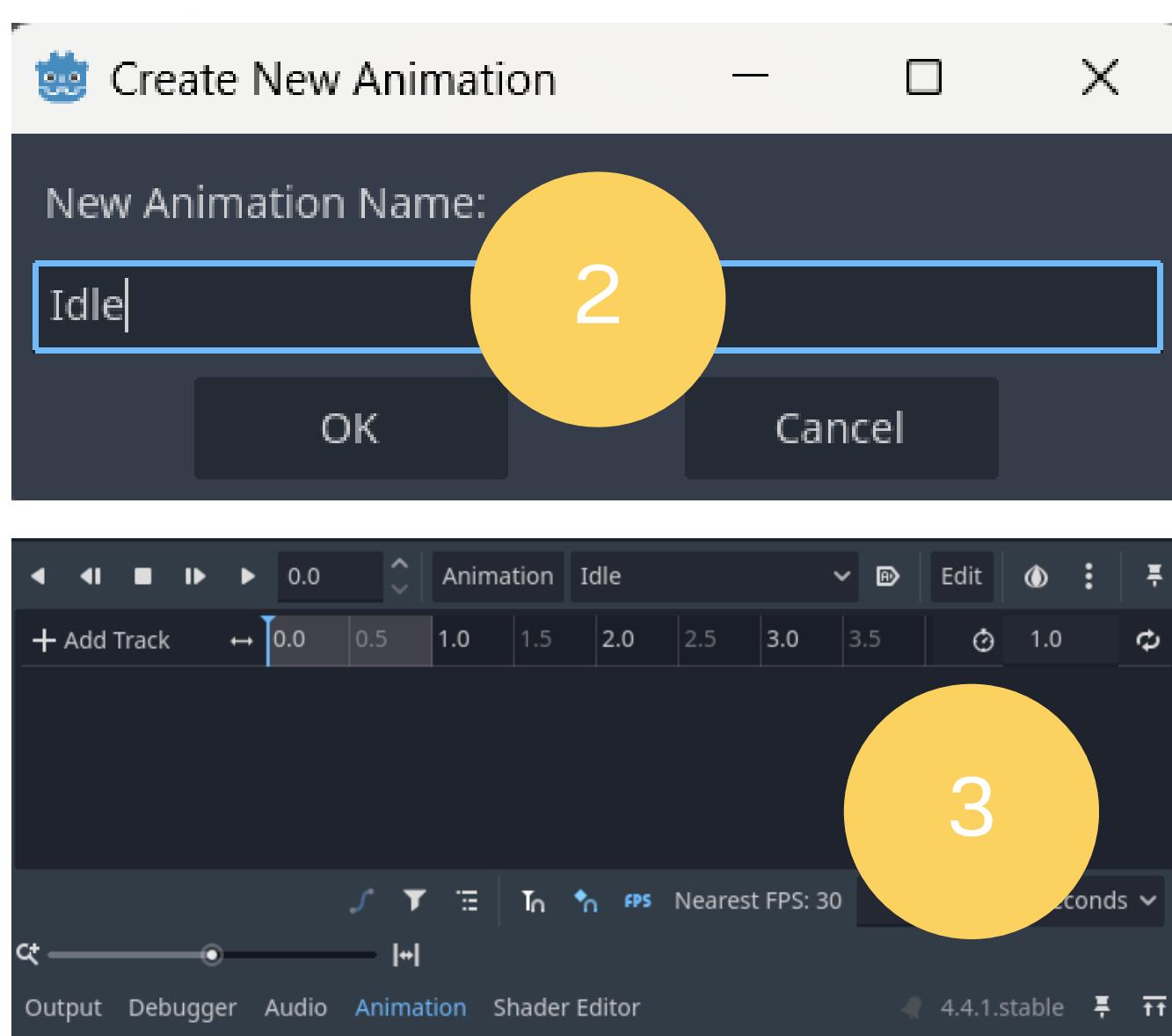
ต้องมาคลิกที่ **AnimationPlayer** uu Panel Scene



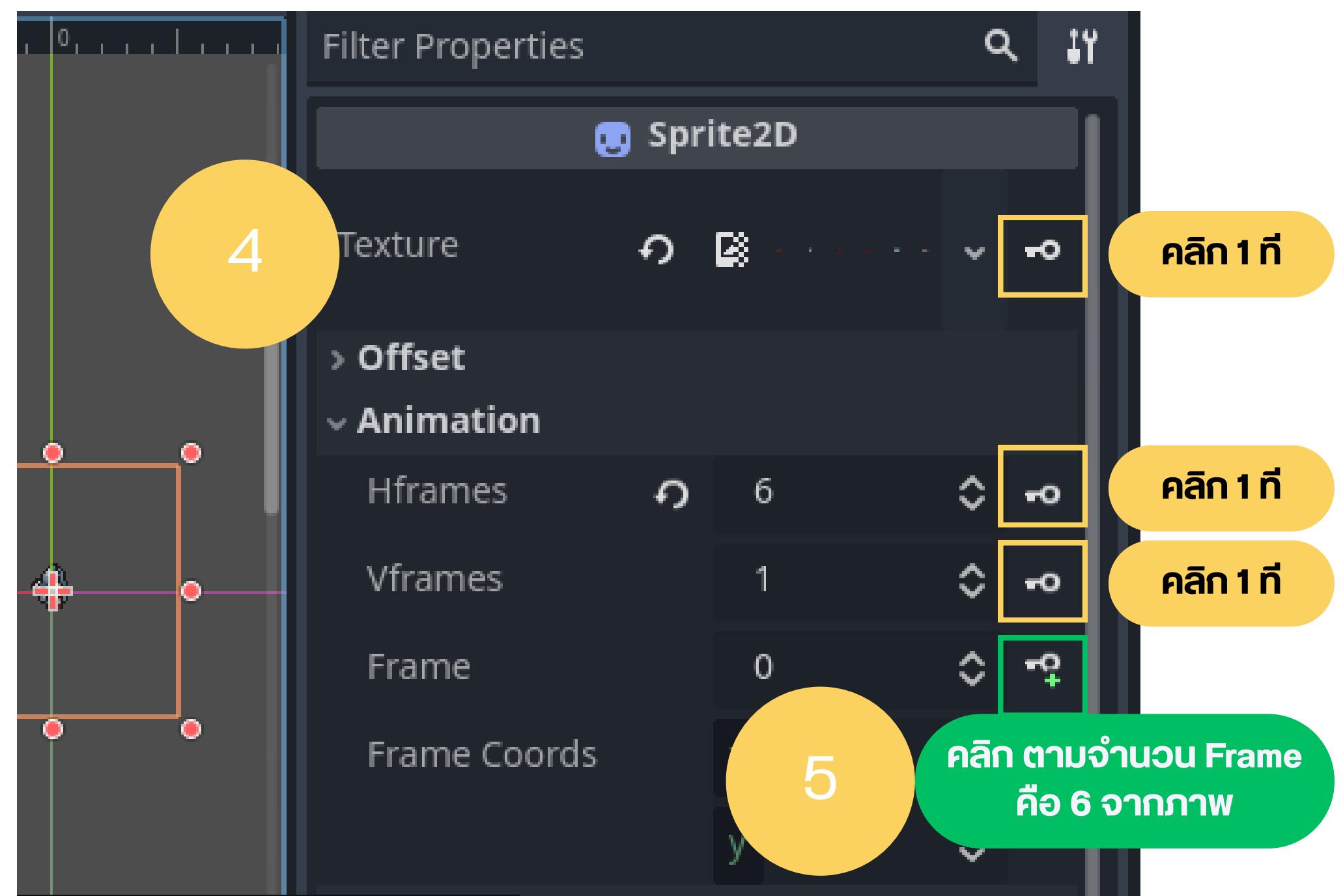
ไปที่ Panel ชื่อ Animation ทำการเลือก New Animation ขึ้นมาใหม่

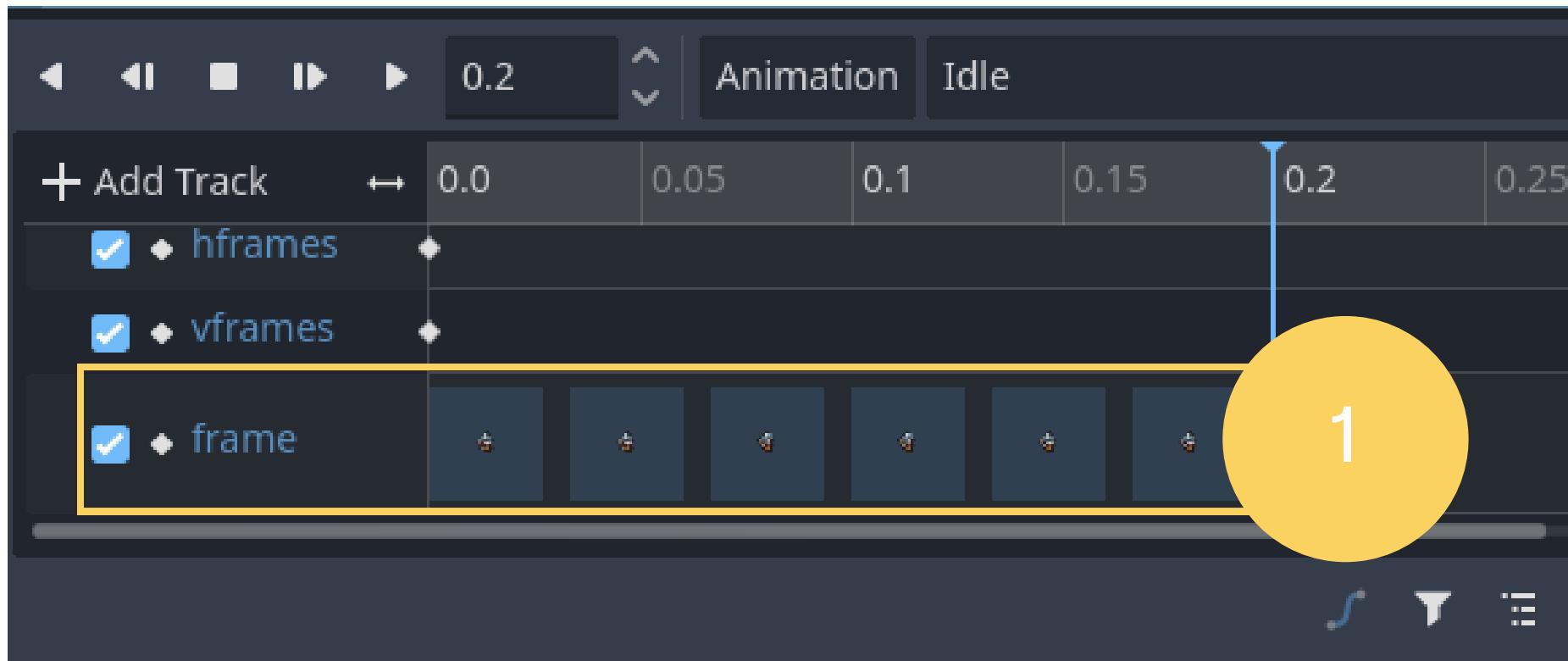


คลิกที่ New หลังจากนั้นจากนั้นให้ไปดูที่ Inspector ของ Sprite2D เม้นจะเป็นสัญลักษณ์ของรูป **ลูกกุญแจ** นั่นคือปุ่มการบันทึก **Keyframe**



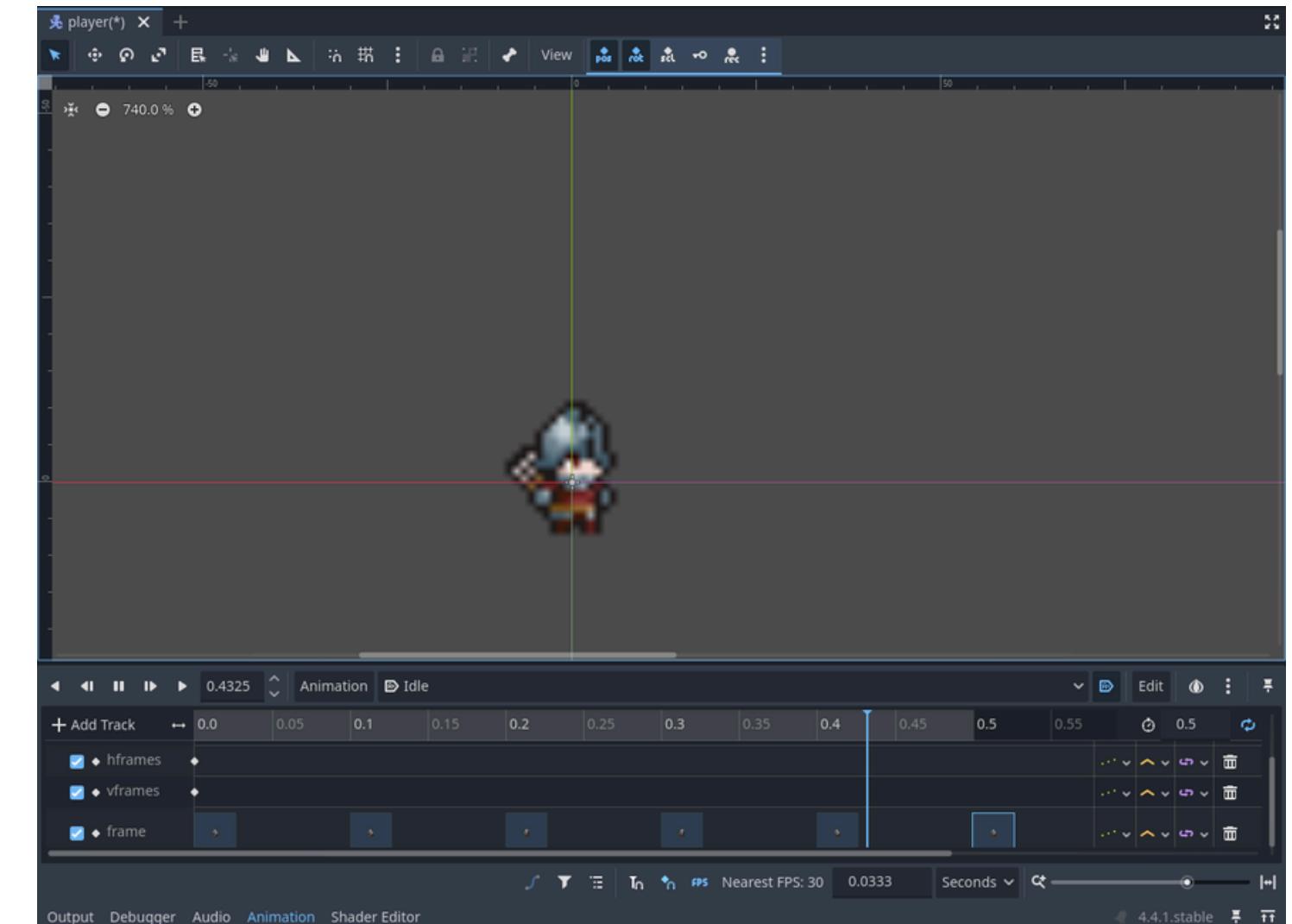
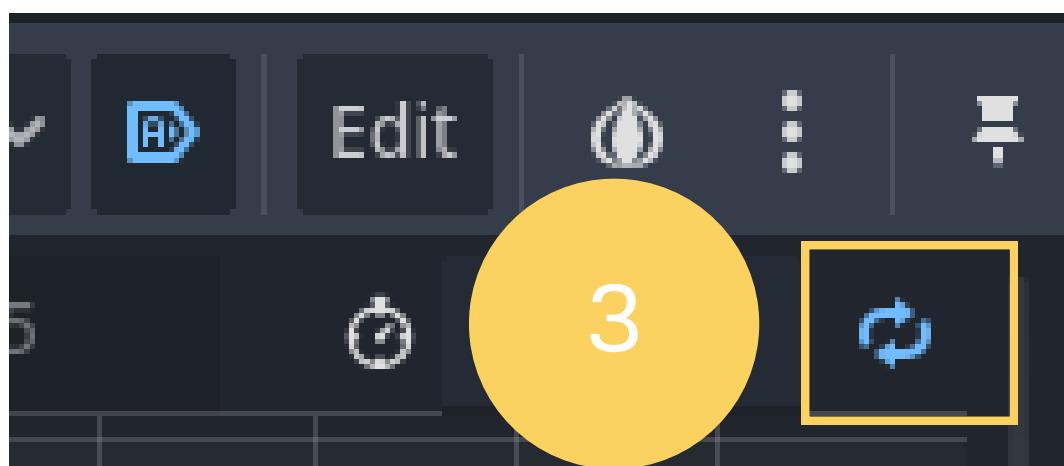
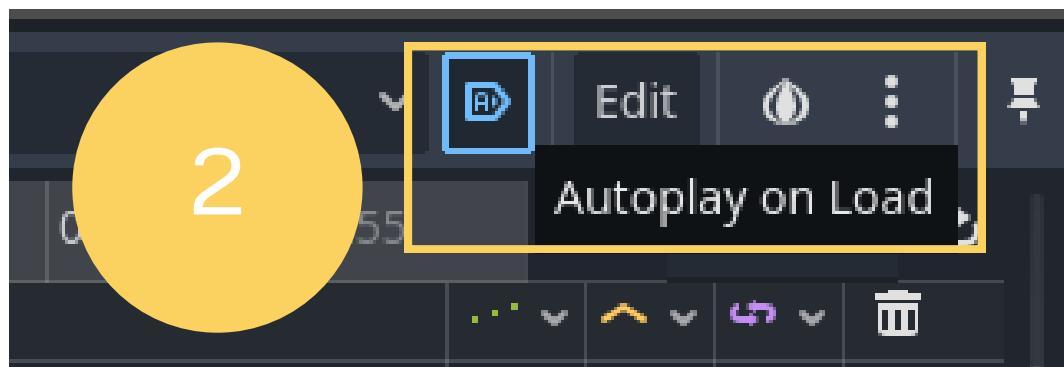
คลิกที่ Sprite2D หลังจากนั้นจากนั้นให้ไปดูที่ Inspector ของ Sprite2D เม้นจะเป็นสัญลักษณ์ของรูป **ลูกกุญแจ** นั่นคือปุ่มการบันทึก **Keyframe**



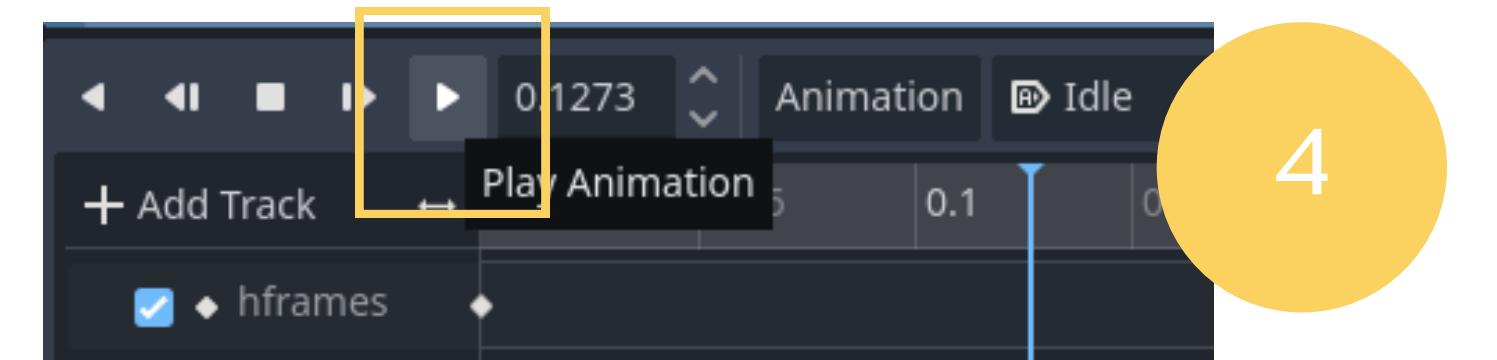


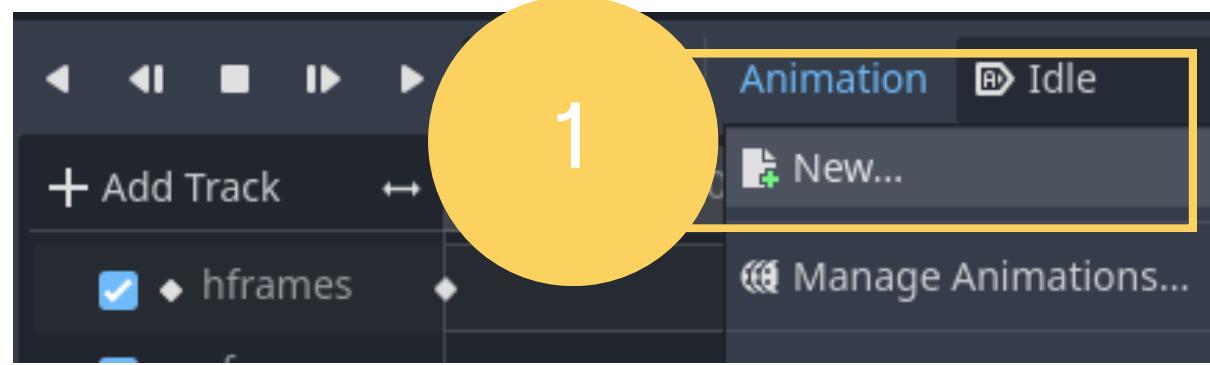
เมื่อเพิ่ม Keyframes ครบหมดแล้วสังเกตการเปลี่ยนแปลงที่เราได้ และอย่าลืมกดปุ่ม Autoplay on Load เพื่อให้มันเป็น Loop

ส่วนของการปรับ Loop อยู่ที่ไอคอนรูป Loop



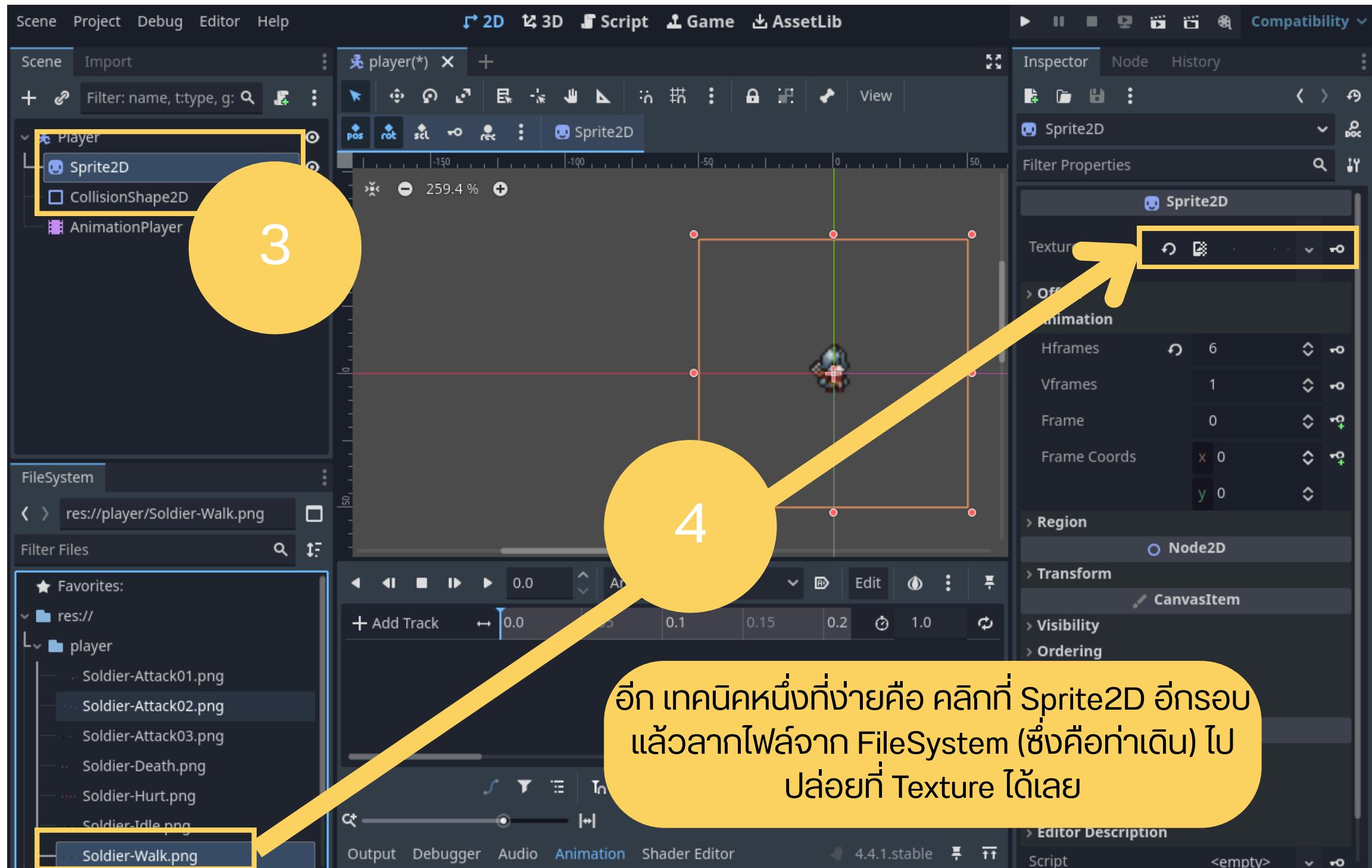
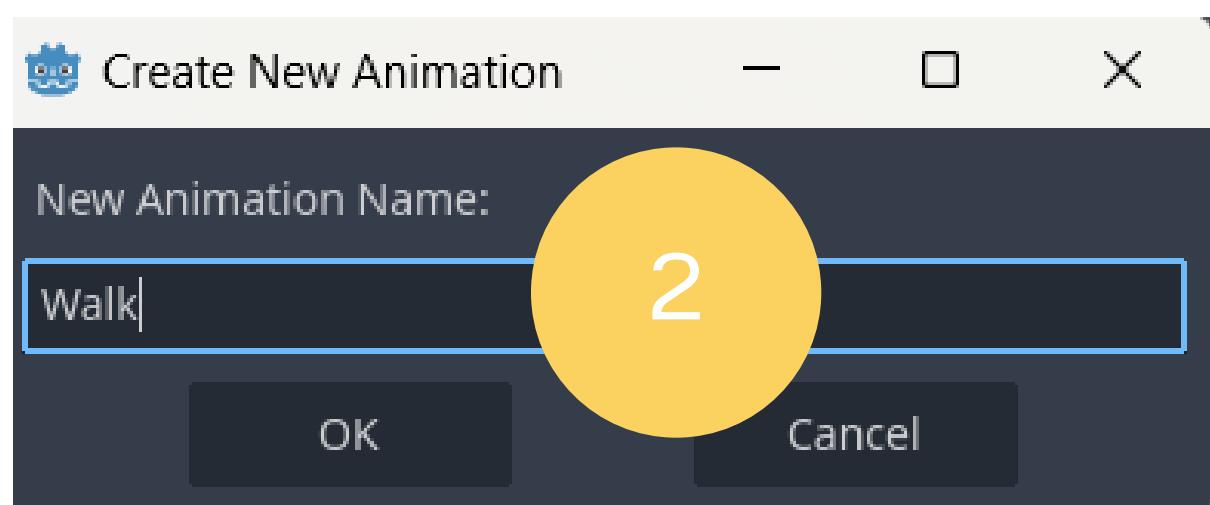
ทดสอบปรับ ระยะ ความห่างของ Sprite Frame และทำการทดสอบการ ขับโดยการกด Play ที่ Animation Panel ดูว่า มันเคลื่อนที่เหมาะสมหรือไม่



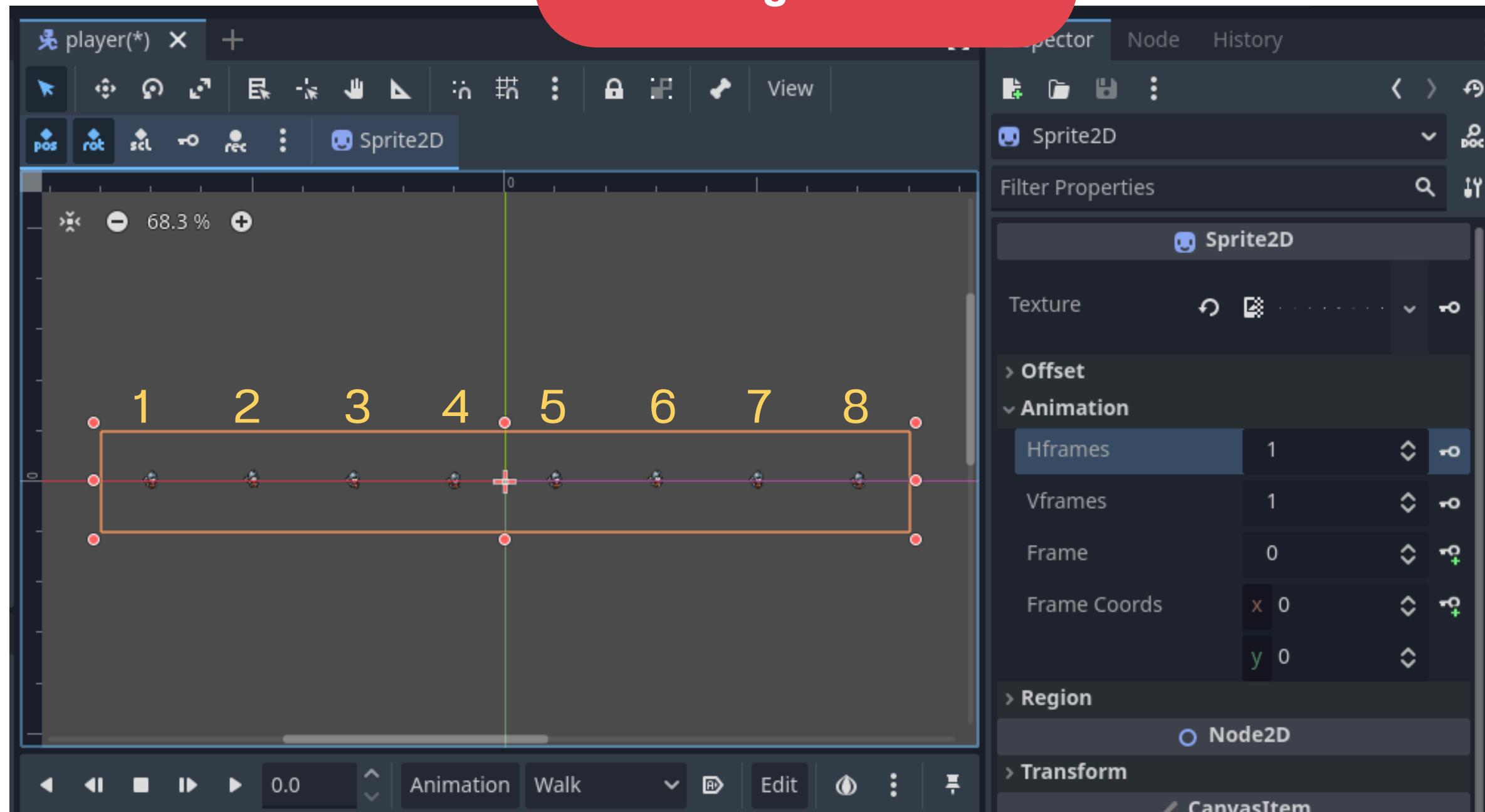


ขั้นตอนที่ 15 การเพิ่มชุด Animation ใหม่ Add new Animation

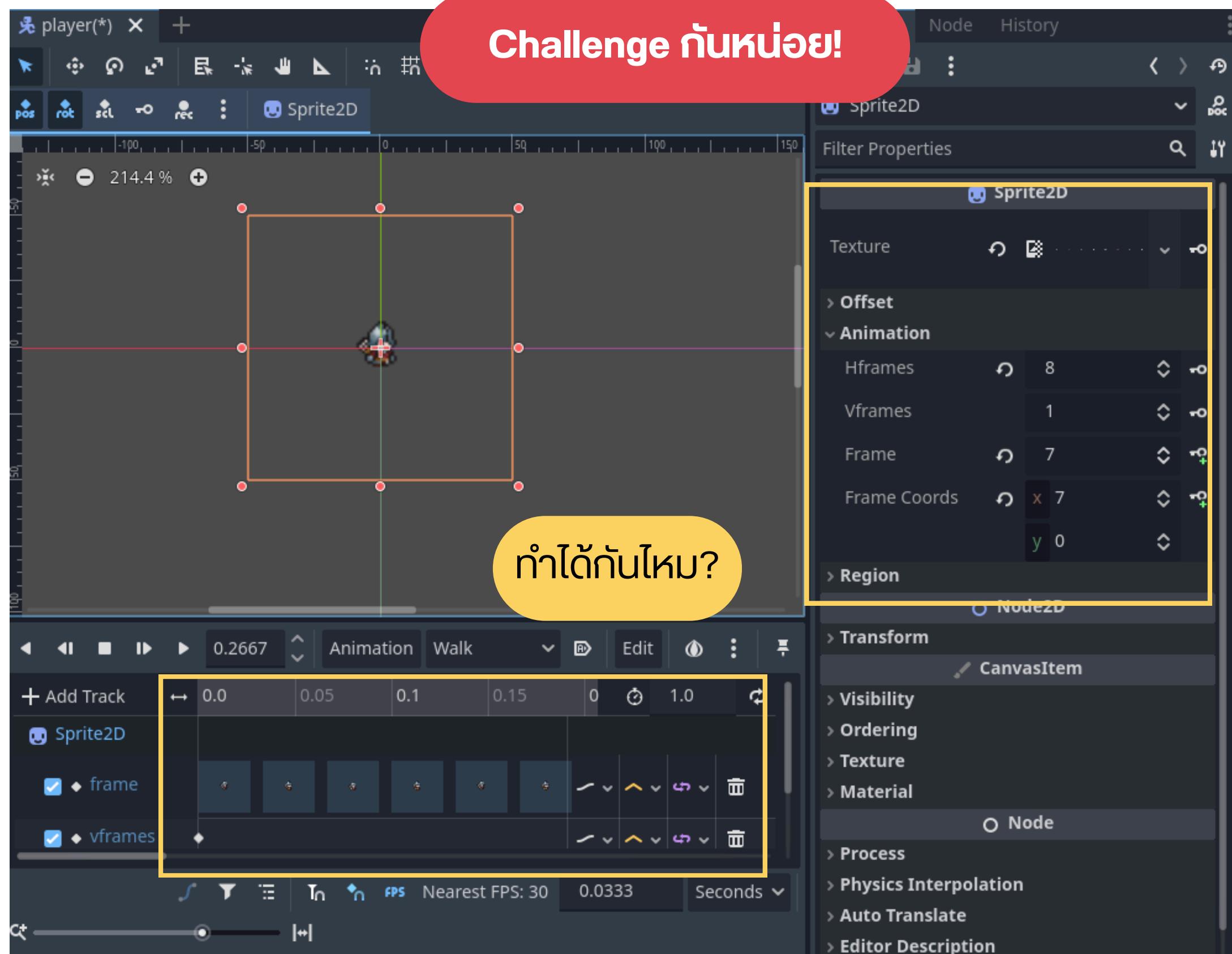
หากว่าเราพอใจกับ animation **Idle** ของเราแล้ว
ในขั้นตอนต่อไปคือการสร้าง New Animation
ใหม่ตั้งชื่อว่า **Walk** (ระวังตัวอักษรเล็กหรือใหญ่)

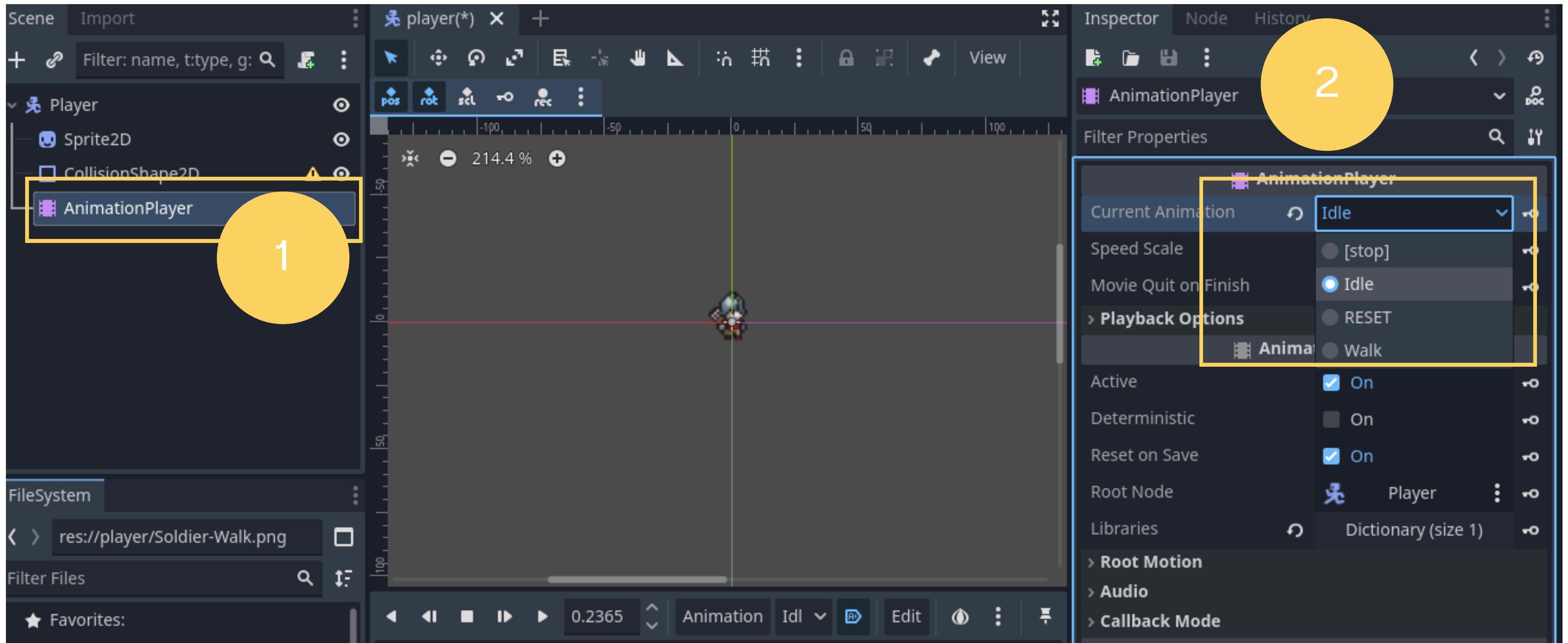


Challenge กันหน่อย!



ลองกวน Step ของขั้นตอน 14-15 ดูสิว่าเราทำได้ไหม? ระวังเรื่องของจำนวน
Frames ล่ะ

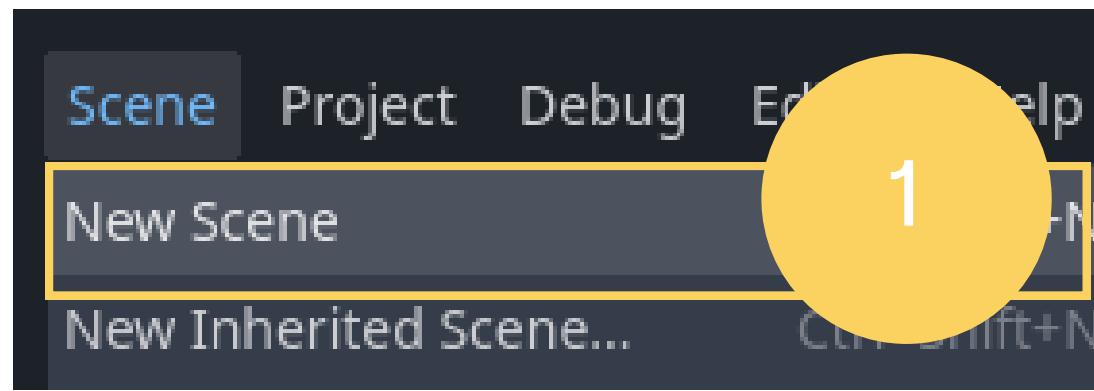




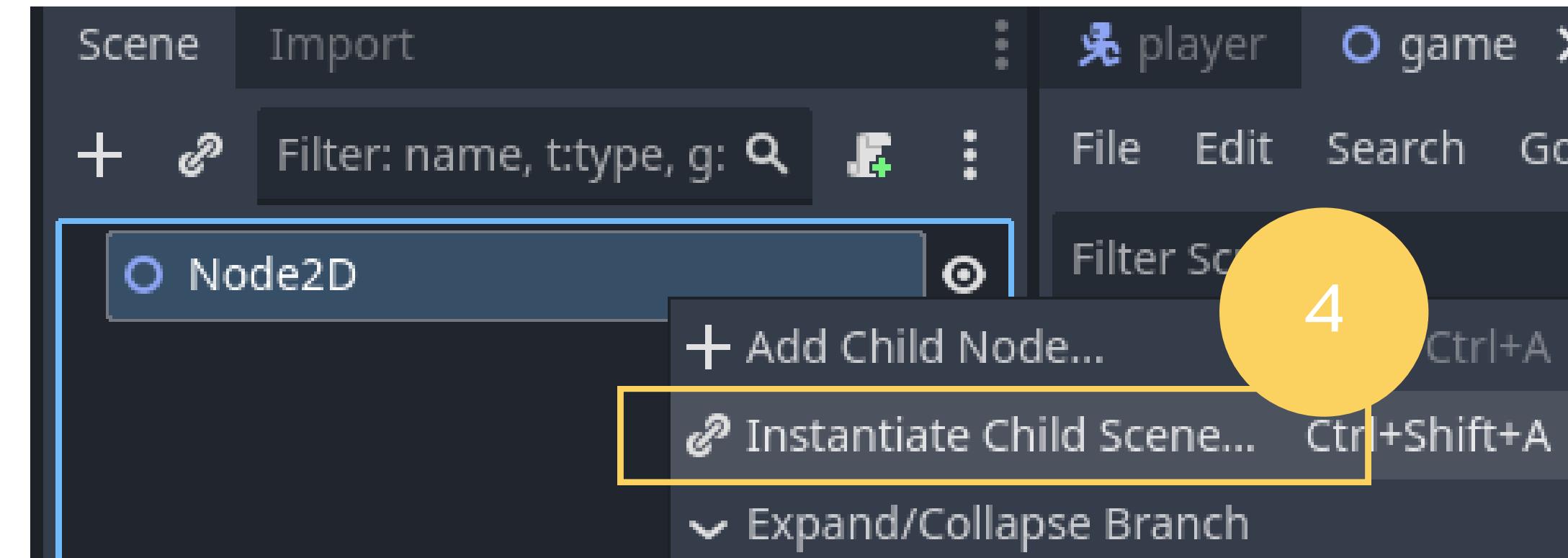
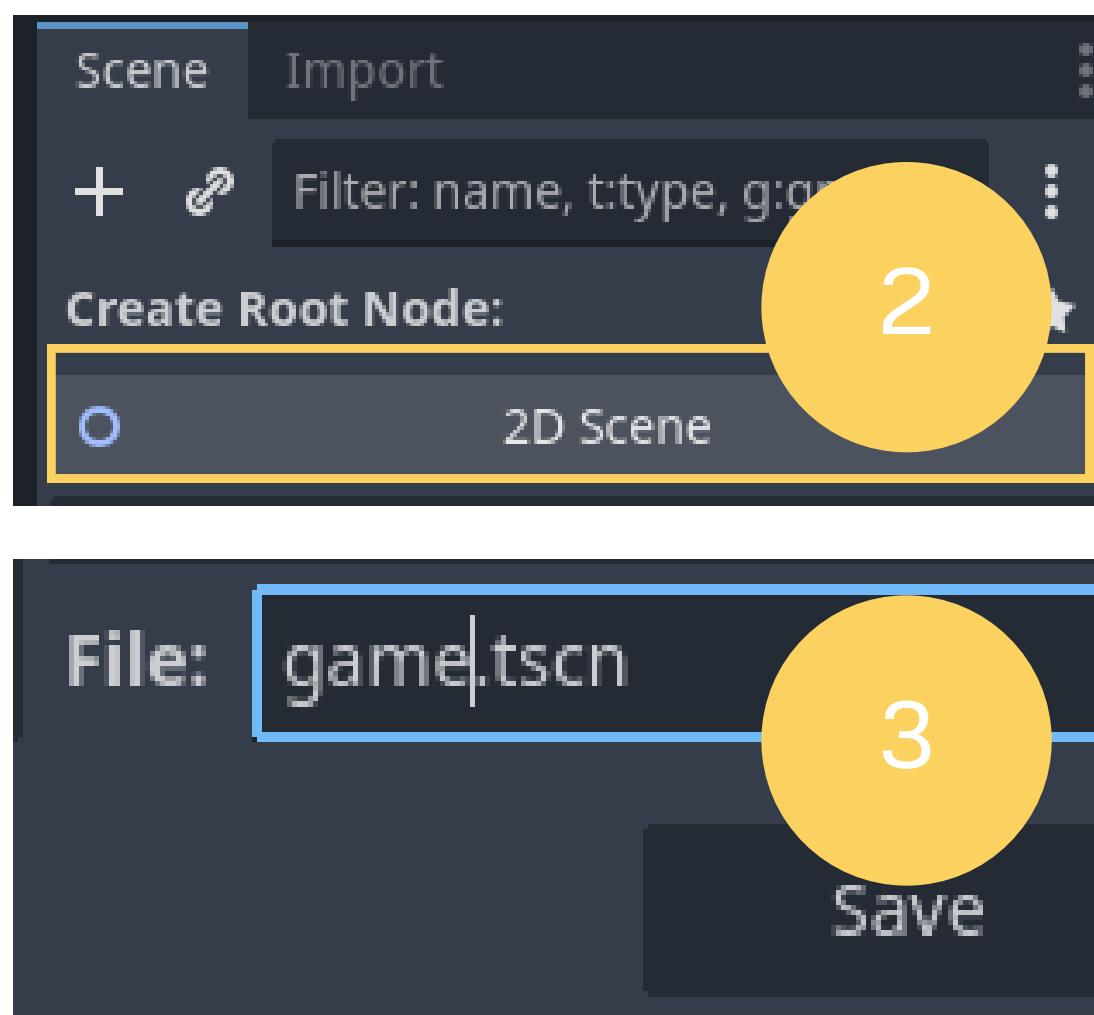
ขั้นตอนที่ 16 ตั้งค่า Current Animation

Set Current Animation

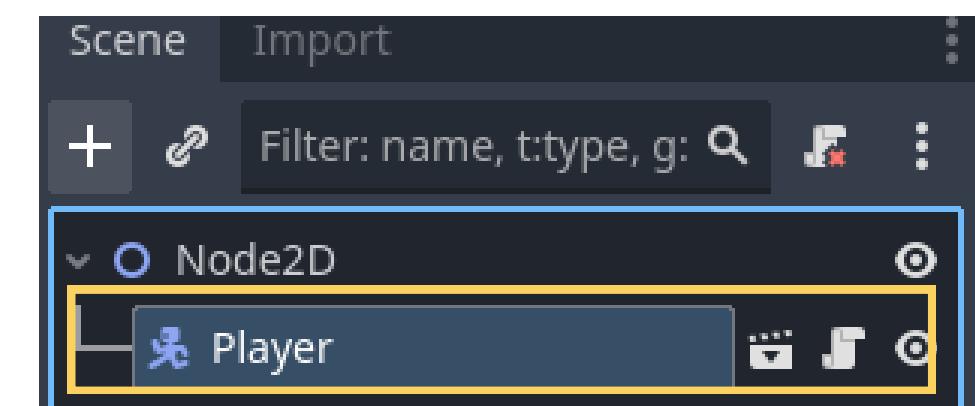
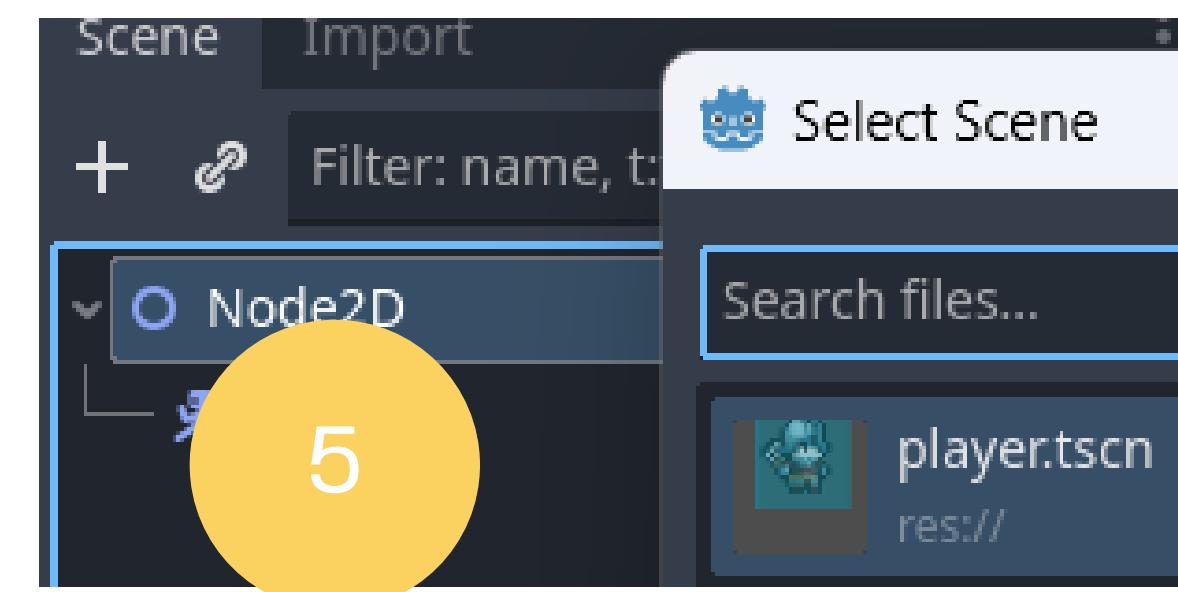
คลิกที่ AnimationPlayer บน Panel Scene หลังจากนั้นไปที่ Inspector ปรับส่วนของ Current Animation ให้เลือก DropDownList เป็น Idle เป็น State พื้นฐานหลักของเรา



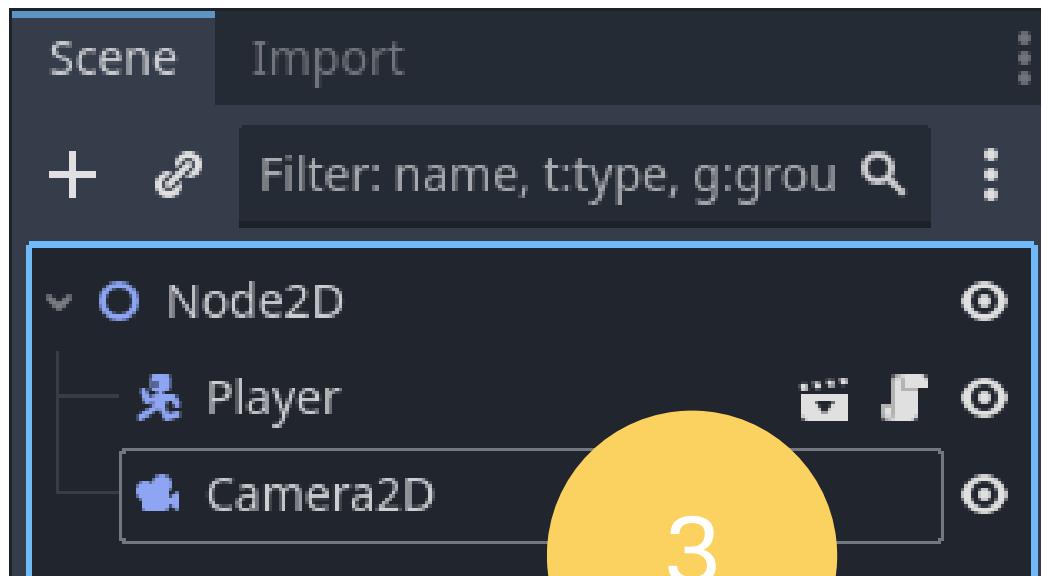
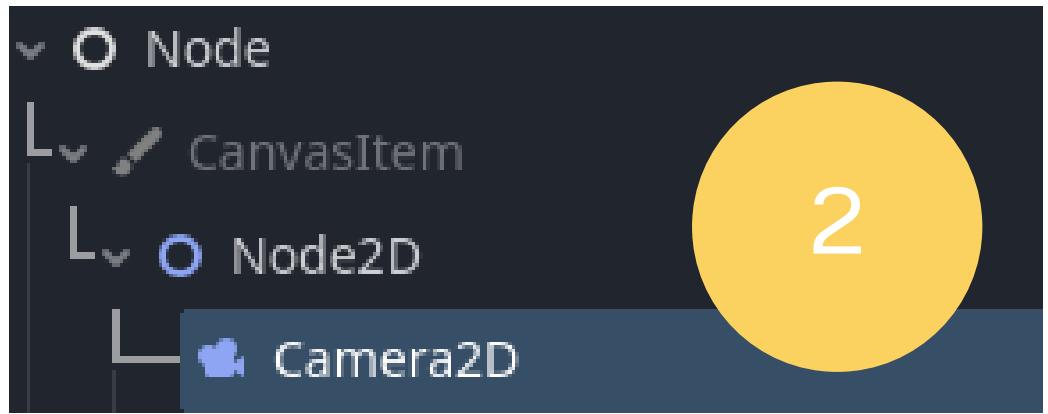
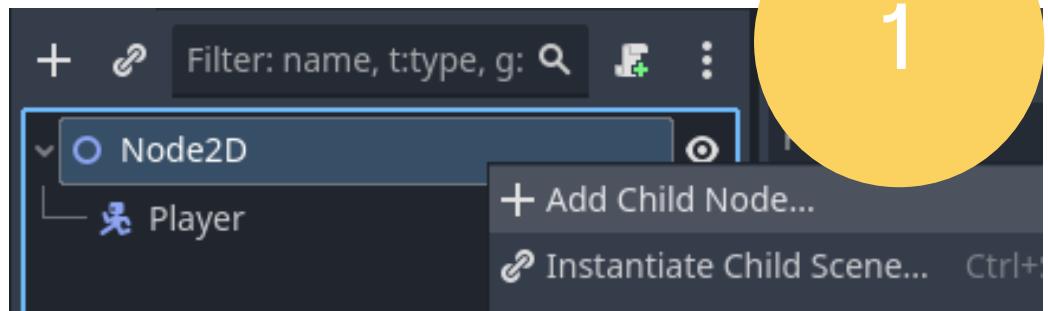
ทำการ **New Scene** ใหม่ขึ้นมาตั้งชื่อ Scene ว่า **game.tscn** สำหรับแสดงผลเกมของเรา ทำการเลือก Node เป็น **2D Scene**



คลิกขวาที่ **Node2D** หลังจากนั้นเลือก **Instantiate Child Scene...** เพื่อเอา **player.tscn** เข้ามาในจานเกมของเรา



จะเห็นว่าตอนนี้ Player จะเป็น Child Node ของ Node2D เรียบร้อยแล้ว



ขั้นตอนที่ 17 เพิ่มกล้องให้เกม Add Camera2D

พระเอกหลักของการสร้างเกมคือกล้องที่เราต้องแสดงผลตัวละครให้ปรากฏในชีบ ต้องมีการติดตาม ต้องมีการเคลื่อนไหวตามตัวละคร เป็นไปตามกฎของพิสิกส์

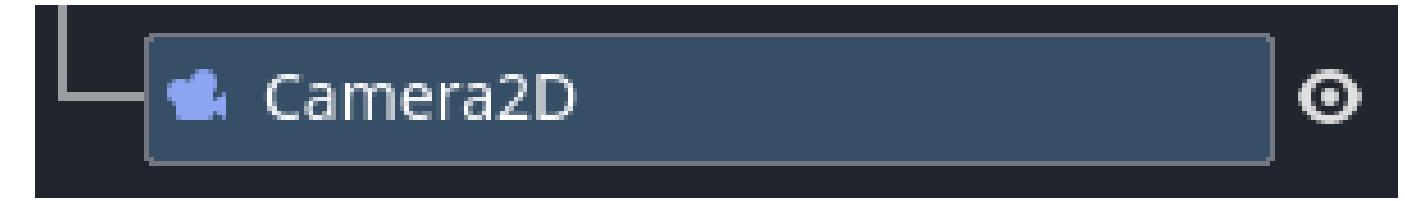
คลิกขวาที่ Node2D

เลือก **+ Add Child Node...**

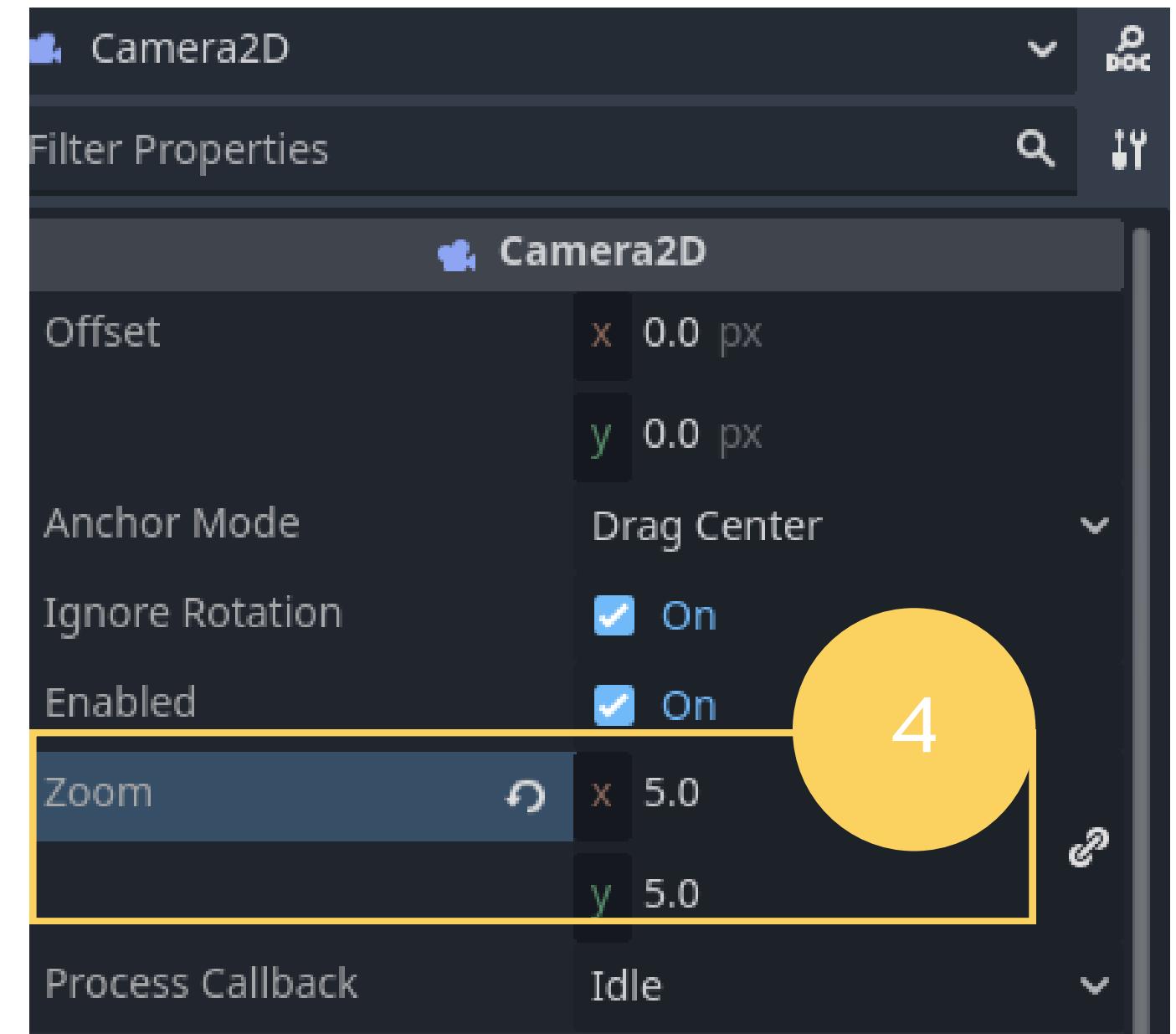
หลังจากนั้น

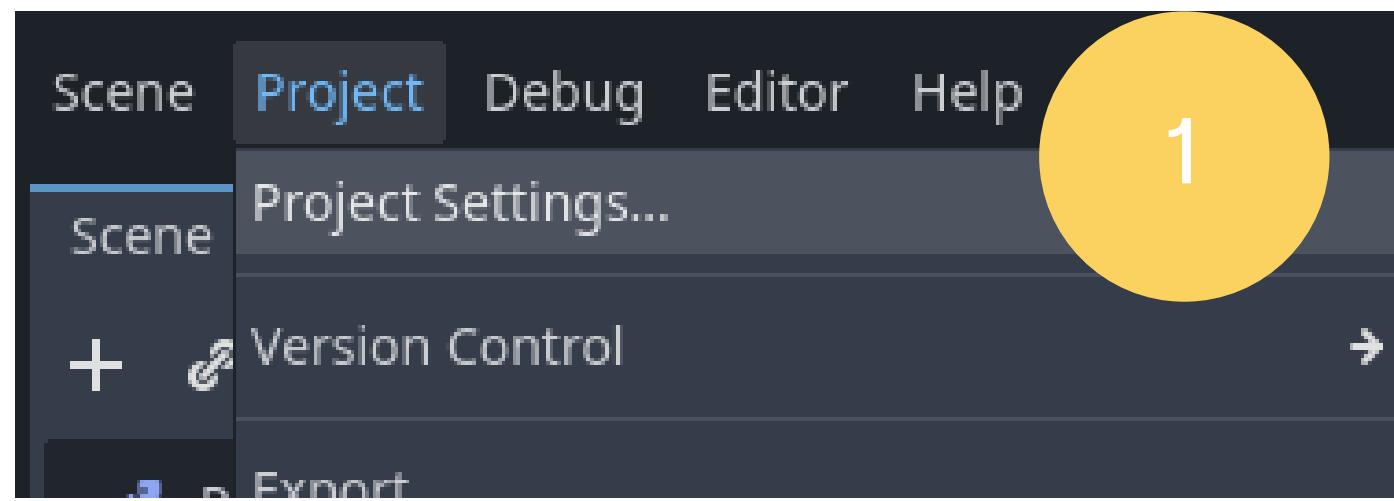
เลือก **Node2D → Camera2D**

วิเคราะห์และตรวจสอบว่าตอนนี้ Camera2D เป็นลูกของ Node2D หรือ ยัง ตามภาพประกอบ



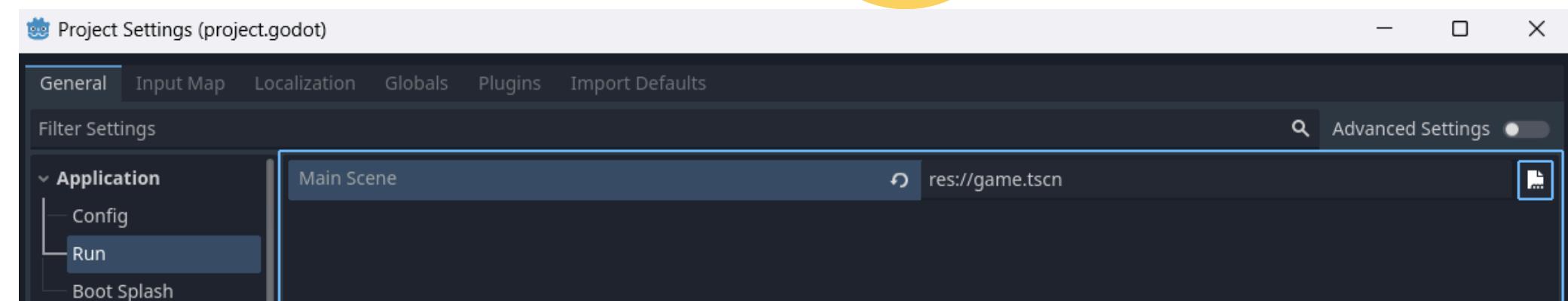
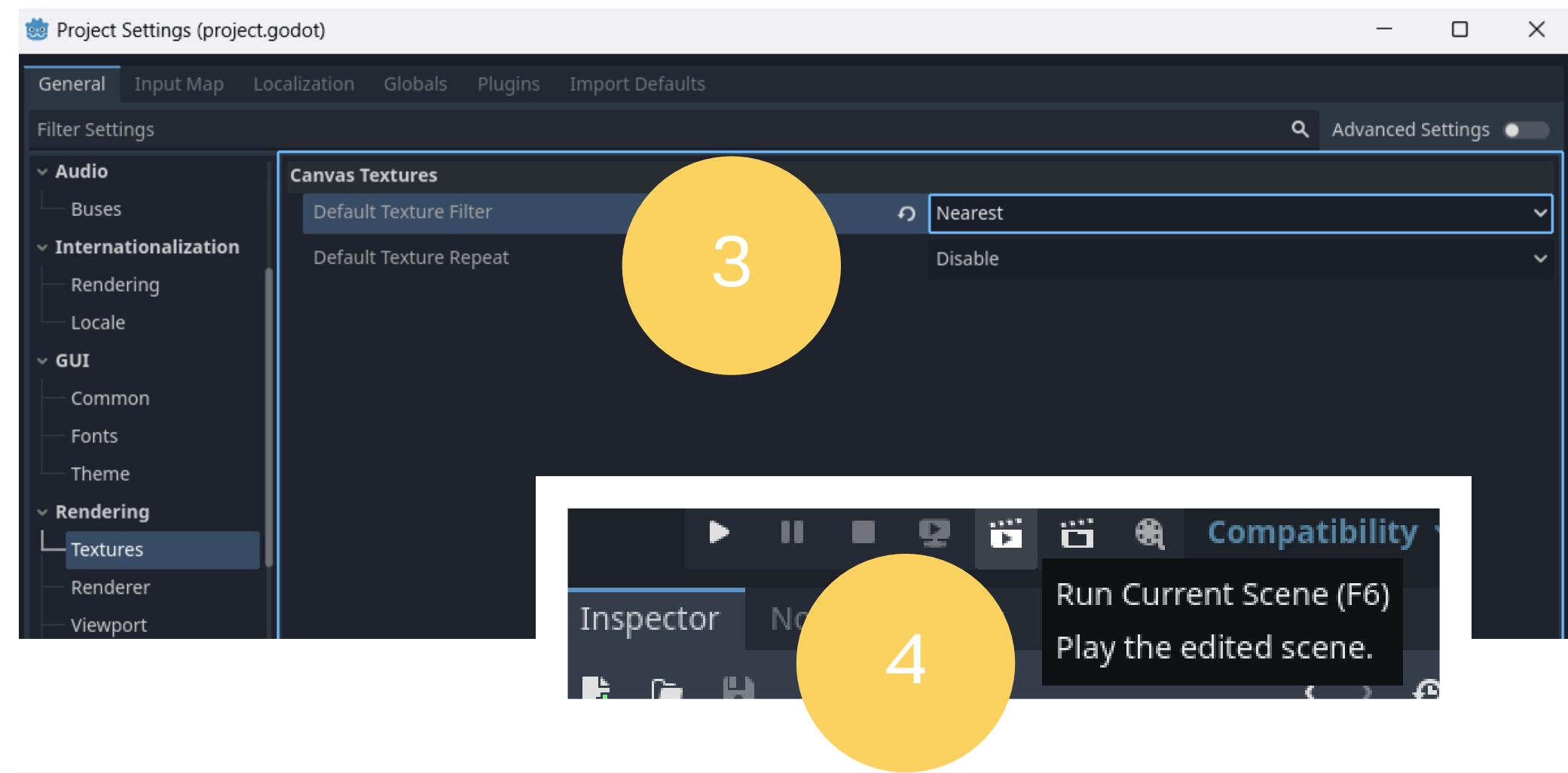
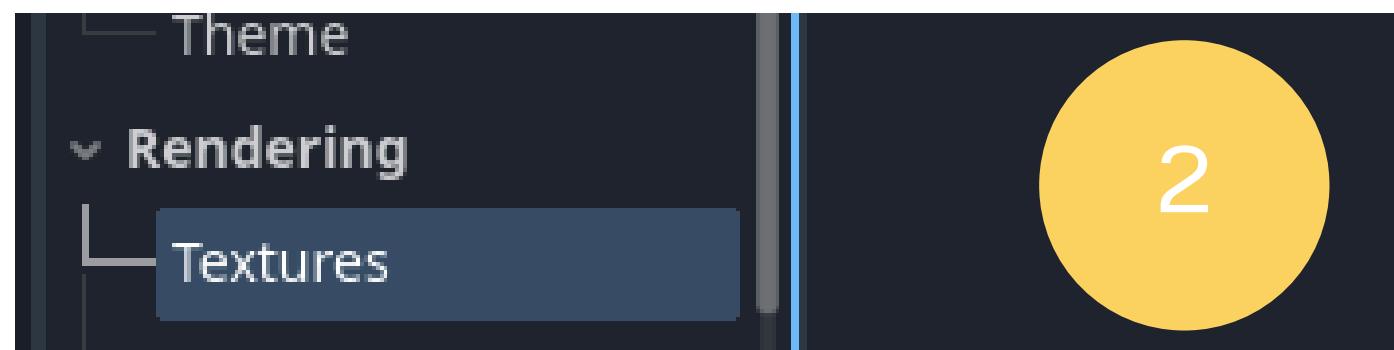
คลิกที่ **Camera2D** ตั้งค่าใน Inspector ส่วนของ Zoom ให้เป็น **x** ที่ 5.0 และ **y** ที่ 5.0 เพื่อการแสดงผล





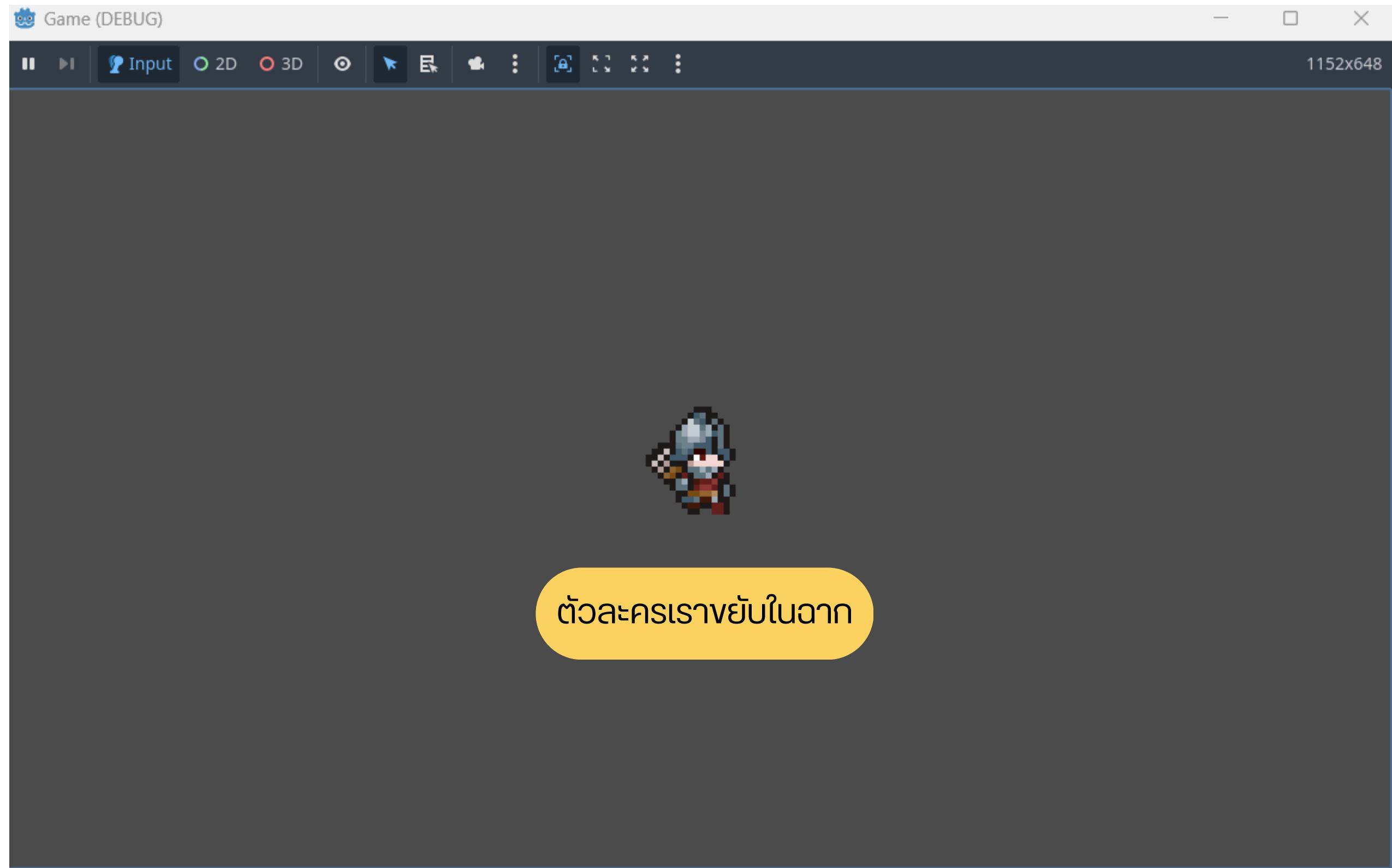
ขั้นตอนที่ 18 ปรับคุณภาพความคมชัดของ Texture Project Settings... → Texture, Filter

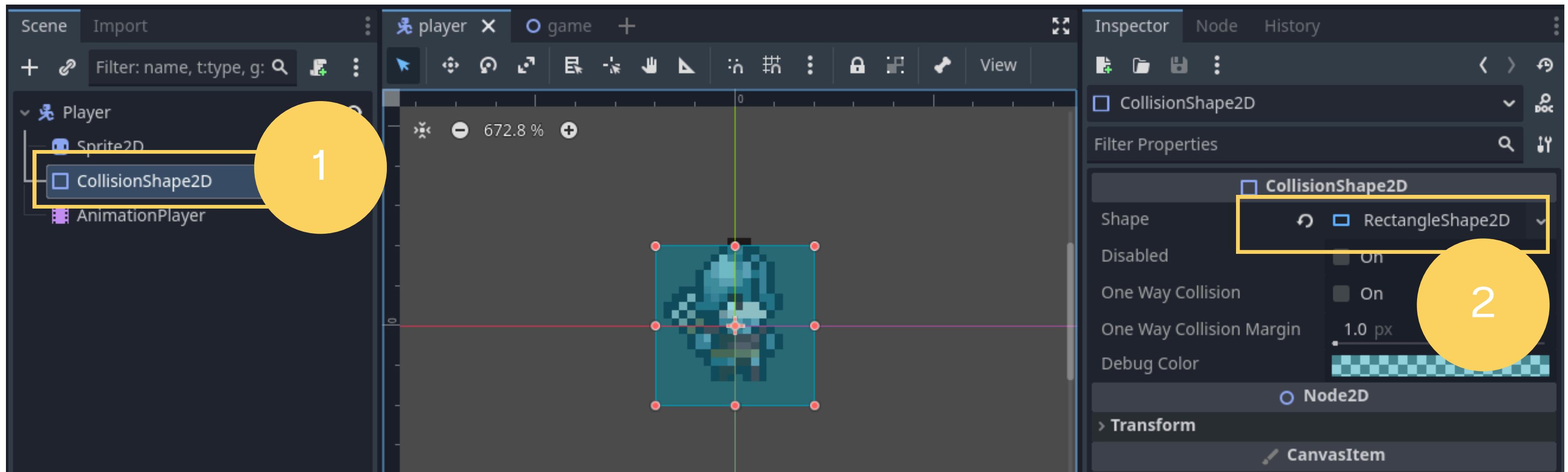
คลิกที่เมนู **Project** หลังจากนั้นเลือก **Project Settings...** ระบบจะเปิดหน้าต่างส่วนของ Project Settings ขึ้นมา ให้เราเลือกหมวดของ **Rendering** → **Textures** แล้วเลือก Canvas Textures เลือก Default Texture Filter ปรับค่า เป็น **Nearest**



ไปที่ แท็บของ **Application-> Run** ให้เลือก Main Scene เป็น **game.tscn** เป็น Scene แรกสำหรับเริ่มต้น

หากปรับค่า Default Texture Filter เป็น Nearest และตั้งค่า **Application->Run** ส่วนของ Main Scene เป็น **game.tscn** เรียบร้อยแล้ว ให้กด **Close** เลื่อนตำแหน่งของโปรแกรมไปที่มุมขวา เลือกปุ่ม Run เลือก **Run Current Scene**





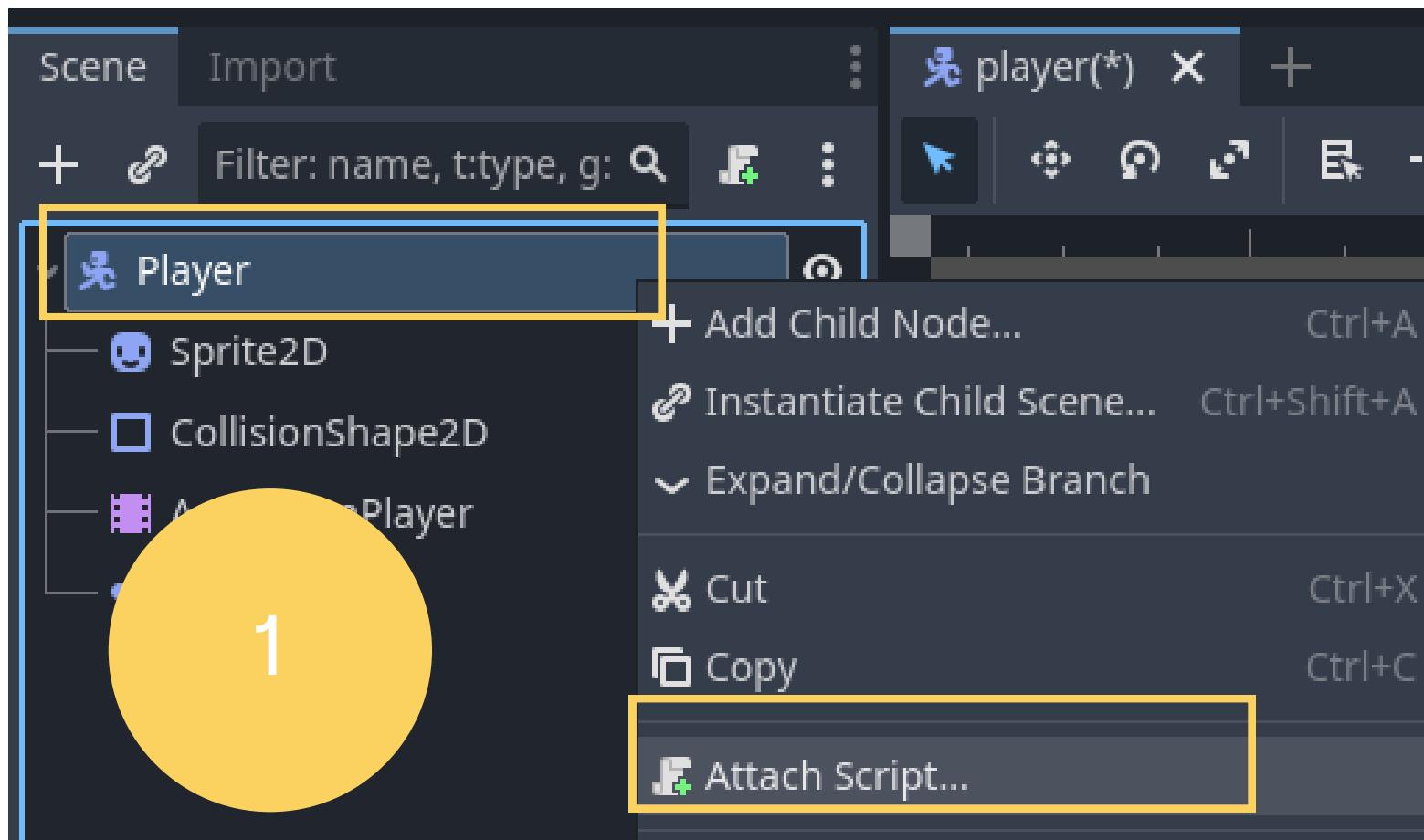
ขั้นตอนที่ 19 การใส่ขอบเขตให้กับตัวละคร Collision Detection (CollisionShape2D)

หลักการ Collision Detection (การตรวจจับการชน) คือกระบวนการที่ใช้ในเกมเพื่อระบุว่า วัตถุ 2 ชิ้นหรือมากกว่านั้นมีการสัมผัส ชน หรือกับซ้อนกันหรือไม่ เช่น: ตัวละครเดินชน ก้าวแพง กระซูนโดนศัตรู ตัวละครเก็บเหรียญได้

ในตัวอย่างคลิกที่ CollisionShape2D บน Panel Scene และไปที่ Inspector หลังจาก บันเลือก **Shape** เลือกเป็น **RectangleShape2D** ก็จะเห็นระยะของ Collision ของตัวละครของเราปรากฏใน Editor Panel เป็นกรอบสีฟ้า

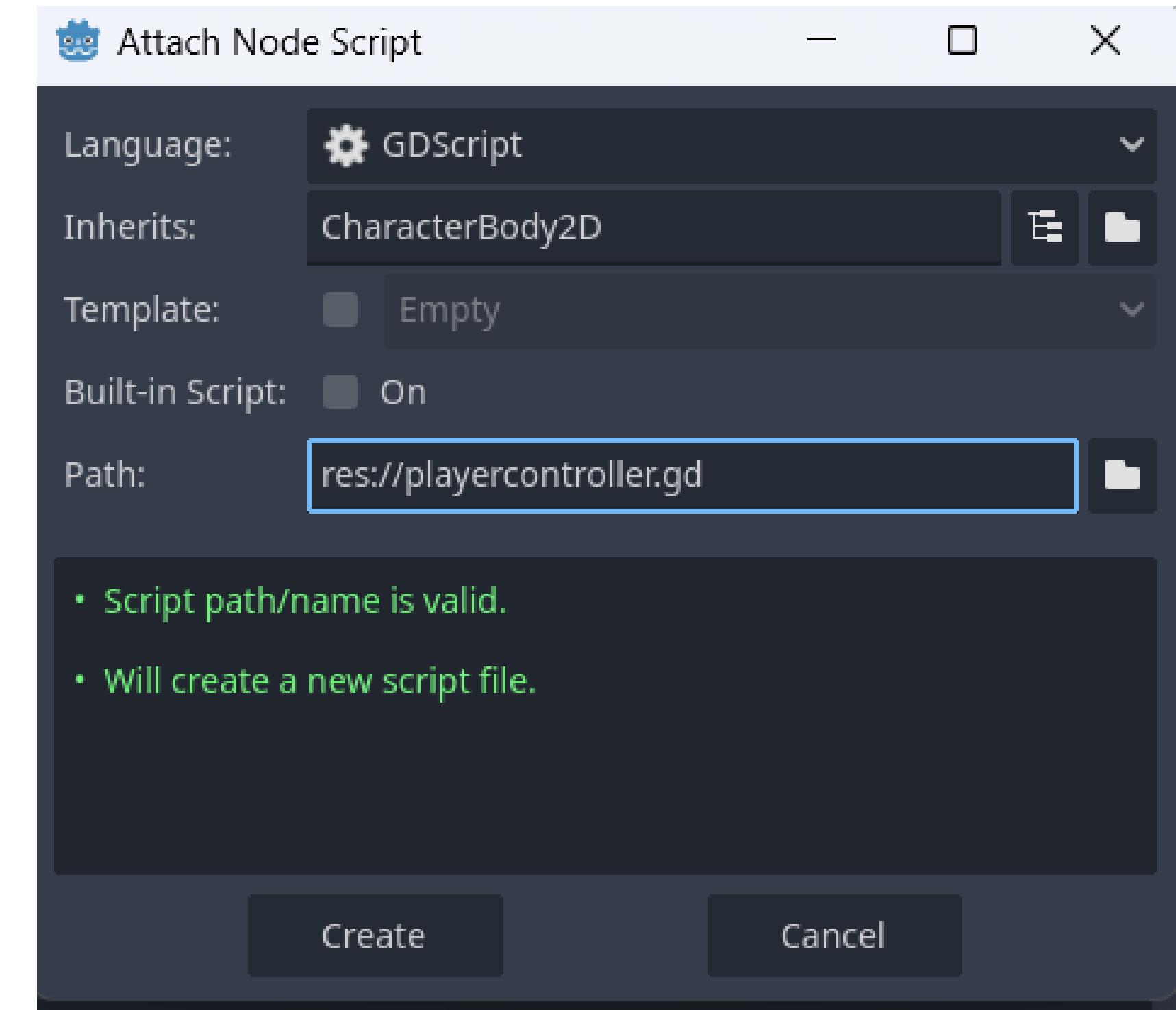
GD Script เขียนโปรแกรม

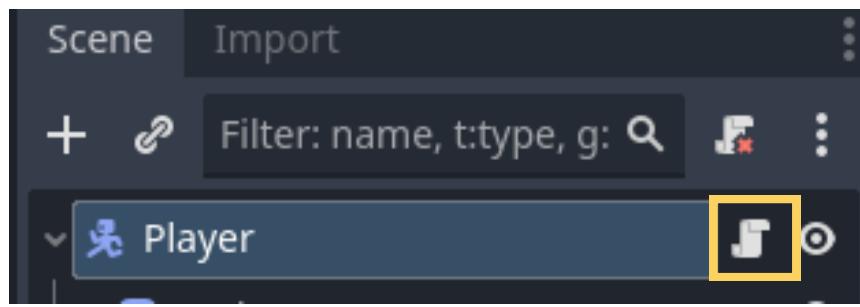
GDScript คือ ภาษาสคริปต์เฉพาะของ Godot Engine ที่ออกแบบมาเพื่อเขียนโค้ดควบคุมเกม มีไวยากรณ์คล้าย Python ใช้งานง่าย อ่านเข้าใจเร็ว และทำงานร่วมกับระบบ Node ของ Godot ได้อย่างลื่นไหล



ขั้นตอนที่ 20 เขียนโปรแกรมกัน
Coding for Game

คลิกขวาที่ Node **Player** หลังจากนั้นเลือก **Attach Script...**
ตั้งชื่อ **Script** ว่า **playercontroller.gd**





เราสามารถพิมพ์คำสั่ง
(Source Code) ได้บน
หน้าต่าง Panel Editor

```
player(*) x +  
File Edit Search Go To Debug  
Filter Scripts  playercontroller.gd(*)  
1 extends CharacterBody2D  
2 # ความเร็วในการเคลื่อนที่ของตัวละคร  
3 @export var speed: float = 200.0
```

ความเร็วในการเคลื่อนที่ของตัวละคร
@export var speed: float = 200.0

```
1 extends CharacterBody2D  
2 # ความเร็วในการเคลื่อนที่ของตัวละคร  
3 @export var speed: float = 200.0  
4 func
```

เทคนิค: ไม่ต้องรีบ พิมพ์เพิ่มไปจะมี Helper ของ Code มาช่วย

```
func _physics_process(delta: float) -> void:  
    var direction = Vector2.ZERO  
    # ตรวจสอบปุ่มกดลูกศร  
    if Input.is_action_pressed("ui_right"):  
        direction.x += 1  
    if Input.is_action_pressed("ui_left"):  
        direction.x -= 1  
    if Input.is_action_pressed("ui_down"):  
        direction.y += 1  
    if Input.is_action_pressed("ui_up"):  
        direction.y -= 1
```

กำหนดการเคลื่อนที่เป็นแนวทแยงได้ราบรื่น
if direction != Vector2.ZERO:
 direction = direction.normalized()

ตั้งค่าความเร็วให้กับตัวละคร
velocity = direction * speed

สั่งให้ตัวละครเคลื่อนที่ตามพิสิกส์
move_and_slide()

player X +

File Edit Search Go To Debug

Filter Scripts 

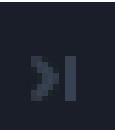
* playercontroller.gd(*)

```
1  extends CharacterBody2D
2  # ความเร็วในการเคลื่อนที่ของตัวละคร
3  @export var speed: float = 200.0
4  func _physics_process(delta: float) -> void:
5      var direction = Vector2.ZERO
6      # ตรวจสอบปุ่มกดลูกศร
7      if Input.is_action_pressed("ui_right"):
8          direction.x += 1
9      if Input.is_action_pressed("ui_left"):
10         direction.x -= 1
11     if Input.is_action_pressed("ui_down"):
12         direction.y += 1
13     if Input.is_action_pressed("ui_up"):
14         direction.y -= 1
15     # ทำให้การเคลื่อนที่เป็นแนวทแยงได้ร้าบเรียบ
16     if direction != Vector2.ZERO:
17         direction = direction.normalized()
18
19
20     # ตั้งค่าความเร็วให้กับตัวละคร
21     velocity = direction * speed
22
23     # สั่งให้ตัวละครเคลื่อนที่ตามพิกัด
24     move_and_slide()
25
```

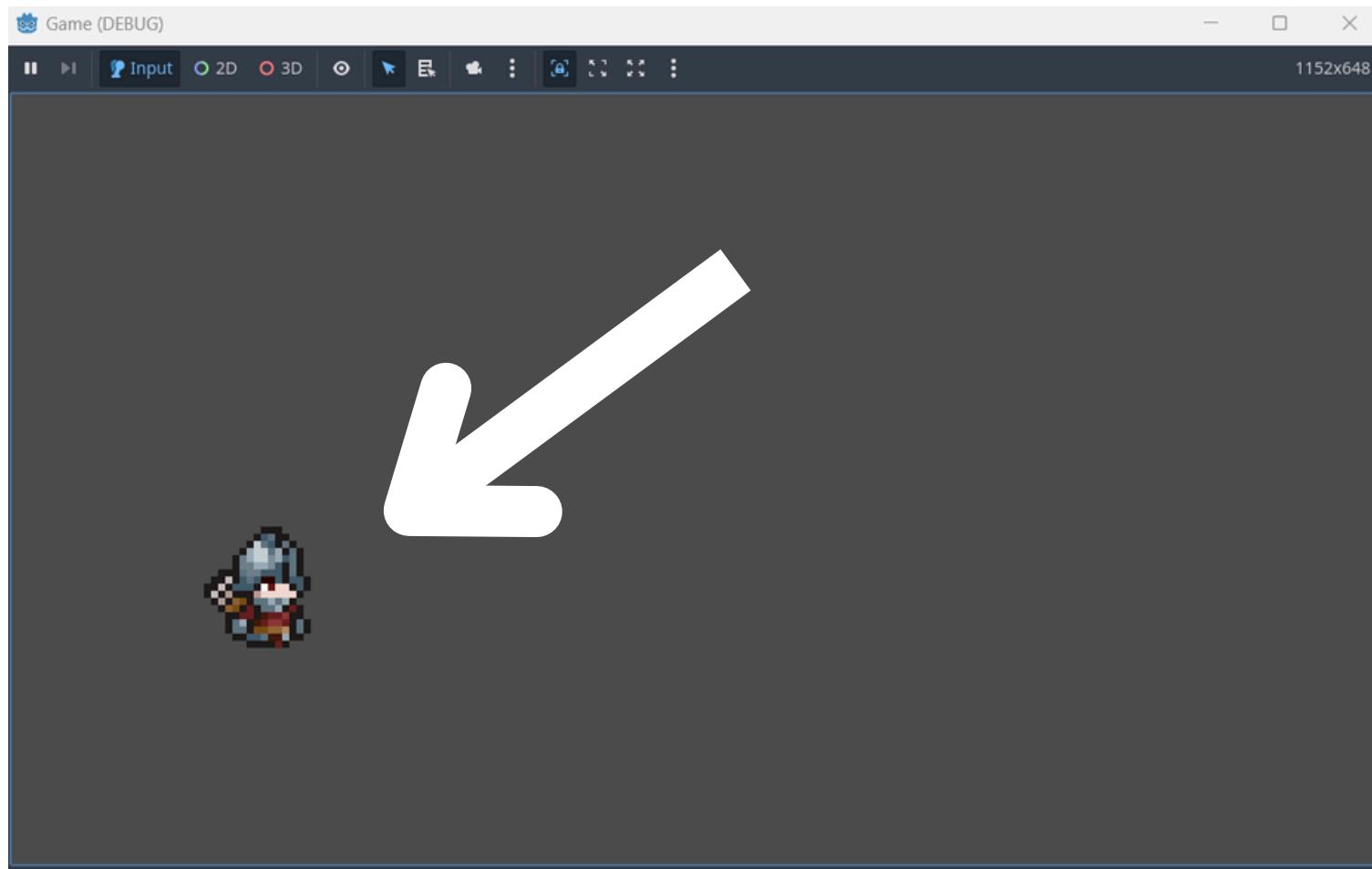
playercontroller.gd(*)  

Filter Methods 

_physics_process

เก็บบีโคيا Copy ไปวางให้ค่อยๆ พิมพ์ แล้วใช้ปุ่ม **Tab** ไปกีลส่วน กีลระยะ สังเกตุการเยื่องให้ดี ด้วยเครื่องหมาย 

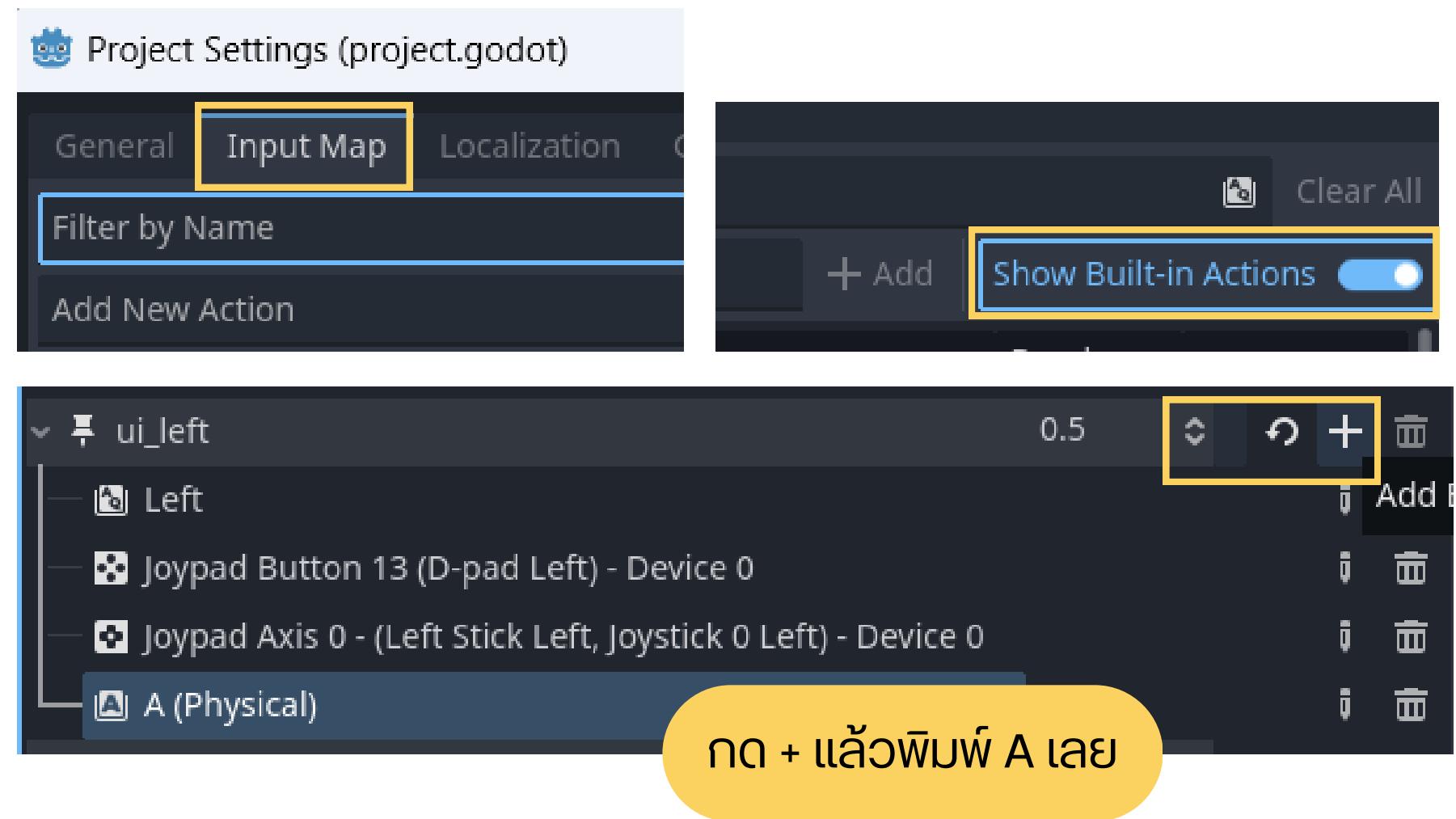
1 @export var speed: float = 200.0
2
3
4 func _physics_process(delta: float) -> void:
5 var direction = Vector2.ZERO
6  Tab 1 ครั้งเยื่อง 1 ครั้ง
7 if Input.is_action_pressed("ui_right"):
8  direction.x += 1 Tab 2 ครั้งเยื่อง 2 ครั้ง
9 if Input.is_action_pressed("ui_left"):
10 direction.x -= 1
11
12 if Input.is_action_pressed("ui_down"):
13 direction.y += 1
14 if Input.is_action_pressed("ui_up"):
15 direction.y -= 1
16
17 # ทำให้การเคลื่อนที่เป็นแนวทแยงได้ร้าบเรียบ
18 if direction != Vector2.ZERO:
19 direction = direction.normalized()
20
21 # ตั้งค่าความเร็วให้กับตัวละคร
22 velocity = direction * speed
23
24 # สั่งให้ตัวละครเคลื่อนที่ตามพิกัด
25 move_and_slide()



ตัวละครเคลื่อนที่ไปมาในจากเกมของเราได้แล้วเหลือแค่:

- อยากรกดปุ่ม WASD แทนลูกศร
- และอยากรีบันหันซ้ายหรือขวา ตาม Direction ของ Horizontal แทนระบบหน้าหลัง X

ตั้งค่า Input ใน Project > **Project Settings > Input Map:**
เพิ่ม action: ui_up, ui_down, ui_left, ui_right
แล้วผูกกับปุ่มลูกศรบนคีย์บอร์ด กด **Show Built-in Action**



กำเนิดนี้องกันกับ WSD ก็สมบูรณ์แล้ว



GODOT

```
if direction.x != 0:  
    $Sprite2D.flip_h = direction.x < 0
```

เพิ่บคำสั่งนี้ไว้ใต้ move_and_slide() เยื่องให้
ดูก็ต้องก็สามารถกำหนดให้ตัวละครของเราหัน
ซ้าย ขวา ไปตาม Direction ได้แล้ว

```
23     # สั่งให้ตัวละครเคลื่อนที่ตามพิกัด  
24     move_and_slide()  
25  
26     if direction.x != 0:  
27         $Sprite2D.flip_h = direction.x < 0
```

The screenshot shows the Godot Engine's integrated development environment (IDE). The top bar includes tabs for 'player', 'game', and '+'. The menu bar has 'File', 'Edit', 'Search', 'Go To', and 'Debug'. Below the menu is a 'Filter Scripts' search bar with the result 'playercontroller.gd'. The main code editor displays the following script:

```
5     var direction = Vector2.ZERO  
6     # ตรวจสอบปุ่มกดลูกศร  
7     if Input.is_action_pressed("ui_right"):  
8         direction.x += 1  
9     if Input.is_action_pressed("ui_left"):  
10        direction.x -= 1  
11    if Input.is_action_p  
12        direction.y += 1  
13    if Input.is_action_p  
14        direction.y -= 1  
15  
16    # สำหรับการเคลื่อนที่เป็นแนวพื้น  
17    if direction != Vect  
18        direction = dire  
19  
20        # ตั้งค่าความเร็วให้กับตัวละคร  
21        velocity = direction  
22  
23        # สั่งให้ตัวละครเคลื่อนที่ตามพ  
24        move_and_slide()  
25  
26    if direction.x != 0:  
27        $Sprite2D.flip_h
```

The bottom right corner of the code editor has a large white arrow pointing left, indicating the direction of the character in the game window.

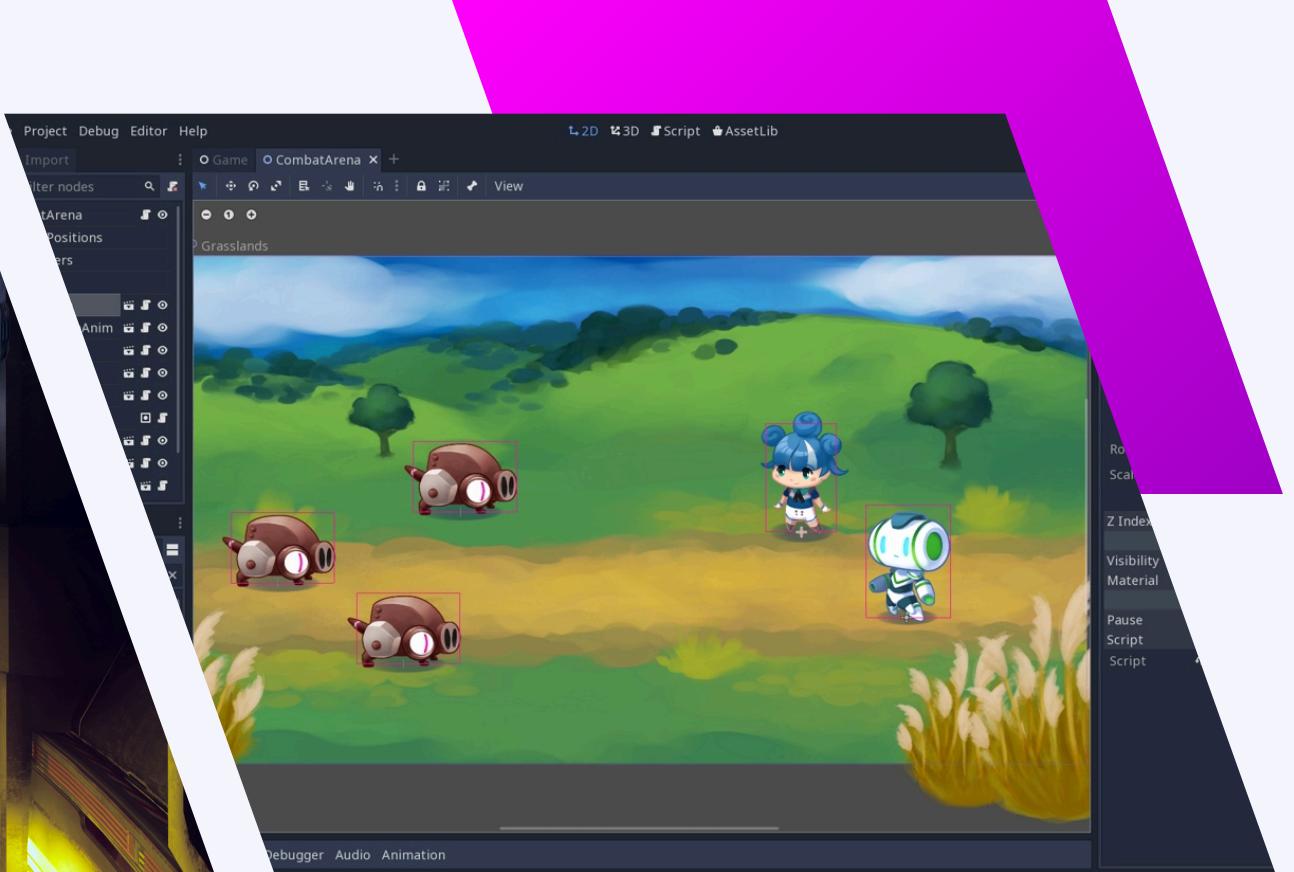
Source Code และ Project สำหรับ Run บนกวนก็งบารียน
ดาวน์โหลดได้ที่

Godot Engine 4.0.1-stable-0-g49a5bc7b6 - <https://godotengine.org>
OpenGL API - 3.3 Mobile GL ES 2.0 Context 24.9.1.240813 - Compatibility - Using Device: ATI Technologies Inc. - AMD Radeon (TM) Graphics

<https://github.com/banyapon/antdpugodotcourse/tree/main/chapter1>

GAME DEVELOPER

College of Creative Design and Entertainment Technology



การพัฒนาเกมด้วยโปรแกรม Godot Engine