

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO MÔN HỌC
HỌC PHẦN: CÁC CÔNG NGHỆ LẬP TRÌNH HIỆN ĐẠI

TÊN ĐỀ TÀI BÁO CÁO
TÌM HIỂU CÔNG NGHỆ LẬP TRÌNH GAME GODOT

Họ tên các thành viên nhóm 19 - Lớp DCT1211:

- | | |
|--------------------|------------|
| 1. Huỳnh Quốc Hưng | 3121410239 |
| 2. Phan Văn Quyến | 3121410413 |
| 3. Nguyễn Hữu Hậu | 3121410184 |

Ngày hoàn tất báo cáo: 17/04/2025

Giảng viên hướng dẫn: Thầy Phạm Thị Vương

TP. HỒ CHÍ MINH, tháng 4, năm 2025

TRƯỜNG ĐẠI HỌC SÀI GÒN

NHẬN XÉT CỦA GIẢNG VIÊN

TP.HCM, ngày tháng năm 2025

Giảng viên hướng dẫn

Phạm Thị Vương

MỤC LỤC

| | |
|--|----------|
| LỜI MỞ ĐẦU | 6 |
| CHƯƠNG I: TỔNG QUAN ĐỀ TÀI | 7 |
| 1. Giới thiệu đề tài | 7 |
| 2. Mục tiêu | 7 |
| 3. Lý do chọn đề tài | 7 |
| 4. Phạm vi nghiên cứu | 8 |
| CHƯƠNG II:TÌM HIỂU VỀ GODOT | 8 |
| 1. Lịch sử phát triển | 8 |
| 2. Công nghệ và kiến trúc Godot | 9 |
| 3. Hướng dẫn tải và cài đặt Godot | 10 |
| 4. Ứng dụng Godot trong phát triển game | 14 |
| 4.1. Làm quen với ứng dụng Godot | 14 |
| 4.2. Làm game Platform 2D với Godot | 18 |
| 4.2.1. Thiết kế background | 18 |
| 4.2.2. Thiết kế màn chơi | 22 |
| 4.2.3. Thiết kế nhân vật và hiệu ứng chuyển động | 26 |
| 4.2.4. Thiết kế vật lý cho nhân vật | 31 |
| 4.2.5. Thiết kế logic chuyển động nhân vật | 32 |
| 4.2.6. Thiết kế vật lý cho màn chơi | 33 |
| 4.2.7. Cải tiến hiệu ứng của nhân vật | 35 |
| 4.2.8. Camera động | 39 |
| 4.2.9. Chuyển động mượt cho camera động | 39 |
| 4.2.10. Thêm vật phẩm thu thập | 40 |
| 4.2.11. Sửa lỗi liên quan đến kích thước cửa sổ | 42 |
| 4.2.12. Cải tiến vật phẩm thu thập | 43 |
| 4.2.13. Bộ đếm điểm | 44 |
| 4.2.14. Hiển thị bảng điểm | 46 |
| 4.2.15. Thêm kẻ địch vào game | 49 |
| 4.2.16. Thêm kẻ địch biết di chuyển vào game | 53 |
| 4.2.17. Thêm kẻ địch trên không | 59 |
| 4.2.18. Thêm đích đến vào game | 61 |
| 4.2.19. Tạo màn chơi tiếp theo | 64 |
| 4.2.20. Tạo main menu | 64 |
| 4.2.21. Tạo chuyển cảnh giữa main menu và các màn chơi | 69 |

| | |
|--|-----------|
| 5. Xuất bản game lên Web | 73 |
| CHƯƠNG III: KẾT QUẢ VÀ ĐÁNH GIÁ | 77 |
| 1. Phân tích ưu/nhược điểm của Godot | 77 |
| 2. Ứng dụng thực tế và tiềm năng phát triển | 77 |
| CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN | 78 |
| 1. Tổng kết nghiên cứu | 78 |
| 2. Đề xuất hướng nghiên cứu tiếp theo | 78 |
| CHƯƠNG V: TÀI LIỆU THAM KHẢO | 78 |

LỜI MỞ ĐẦU

Ngày nay, ngành công nghiệp game phát triển mạnh mẽ, kéo theo nhu cầu ngày càng cao về các công cụ hỗ trợ lập trình và thiết kế trò chơi. Trong số các game engine hiện nay, Godot nổi lên như một nền tảng mã nguồn mở đầy tiềm năng, cung cấp một môi trường phát triển linh hoạt, mạnh mẽ và hoàn toàn miễn phí.

Với khả năng hỗ trợ đa nền tảng, giao diện trực quan và hệ thống Scene & Node độc đáo, Godot mang đến cho các nhà phát triển game một phương pháp tiếp cận mới, giúp họ dễ dàng xây dựng trò chơi từ đơn giản đến phức tạp. Bên cạnh đó, việc hỗ trợ nhiều ngôn ngữ lập trình như GDScript, C#, C++ cũng giúp Godot trở thành một lựa chọn hấp dẫn cho cả người mới bắt đầu và lập trình viên chuyên nghiệp.

Với mong muốn tìm hiểu về Godot Engine, nhóm chúng em quyết định thực hiện báo cáo này nhằm nghiên cứu về các tính năng, kiến trúc và ứng dụng của Godot trong phát triển game.

Trong quá trình thực hiện, nghiên cứu và tổng hợp tài liệu sẽ không thể tránh khỏi thiếu sót. Rất mong nhận được ý kiến đóng góp từ thầy cô để báo cáo được hoàn thiện hơn.

CHƯƠNG I: TỔNG QUAN ĐỀ TÀI

1. Giới thiệu đề tài

Ngày nay, với sự phát triển mạnh mẽ của ngành công nghiệp game, nhu cầu về các công cụ hỗ trợ lập trình và phát triển game ngày càng gia tăng. Godot là một game engine mã nguồn mở, cung cấp nhiều tính năng mạnh mẽ, linh hoạt và hoàn toàn miễn phí, giúp lập trình viên phát triển trò chơi dễ dàng trên nhiều nền tảng khác nhau.

Báo cáo này tập trung nghiên cứu về Godot, bao gồm kiến trúc hệ thống, các tính năng nổi bật, quy trình phát triển game và ứng dụng thực tế.

2. Mục tiêu

Mục tiêu của báo cáo bao gồm:

- Tìm hiểu tổng quan về Godot và cách nó hoạt động.
- Phân tích các tính năng cốt lõi như hệ thống Scene & Node, GDScript, vật lý, đồ họa.
- Ứng dụng Godot vào một dự án game mẫu để minh họa quy trình phát triển thực tế.

3. Lý do chọn đề tài

Chúng em chọn Godot Engine làm chủ đề nghiên cứu vì các lý do sau:

- Tính miễn phí và mã nguồn mở: Godot giúp các nhà phát triển tiếp cận một công cụ mạnh mẽ mà không cần lo ngại về chi phí bản quyền.
- Hệ thống thiết kế độc đáo: Godot sử dụng mô hình Scene & Node, giúp tối ưu hóa quy trình phát triển.
- Hỗ trợ đa nền tảng: Game có thể dễ dàng xuất bản trên Windows, macOS, Linux, Android, iOS, HTML5 và nhiều nền tảng khác.
- Cộng đồng phát triển đang mở rộng: Godot đang ngày càng thu hút nhiều lập trình viên, với kho tài liệu phong phú và cộng đồng hỗ trợ tích cực.
- Dễ tiếp cận cho người mới bắt đầu: Với GDScript, Godot cung cấp một cách tiếp cận đơn giản nhưng vẫn đủ mạnh mẽ để phát triển các trò chơi phức tạp.

4. Phạm vi nghiên cứu

Báo cáo này tập trung vào các khía cạnh chính của Godot Engine, bao gồm:

- Cấu trúc và nguyên lý hoạt động của hệ thống Scene & Node.
- Tìm hiểu về hệ thống vật lý, đồ họa, âm thanh.
- Phân tích các ứng dụng thực tế của Godot trong phát triển game 2D.
- Xây dựng một game mẫu để minh họa cách triển khai thực tế với Godot.

CHƯƠNG II: TÌM HIỂU VỀ GODOT

1. Lịch sử phát triển

Godot là một game engine mã nguồn mở, được phát triển từ năm 2007 bởi **Juan Linietsky** và **Ariel Manzur**, hai lập trình viên người Argentina. Ban đầu, Godot được sử dụng làm công cụ nội bộ cho một số công ty phát triển game. Đến năm 2014, nó chính thức được phát hành công khai theo giấy phép MIT License, cho phép lập trình viên sử dụng hoàn toàn miễn phí và tùy chỉnh theo nhu cầu.

Từ đó, Godot đã liên tục phát triển với sự đóng góp từ cộng đồng mã nguồn mở. Một số cột mốc quan trọng:

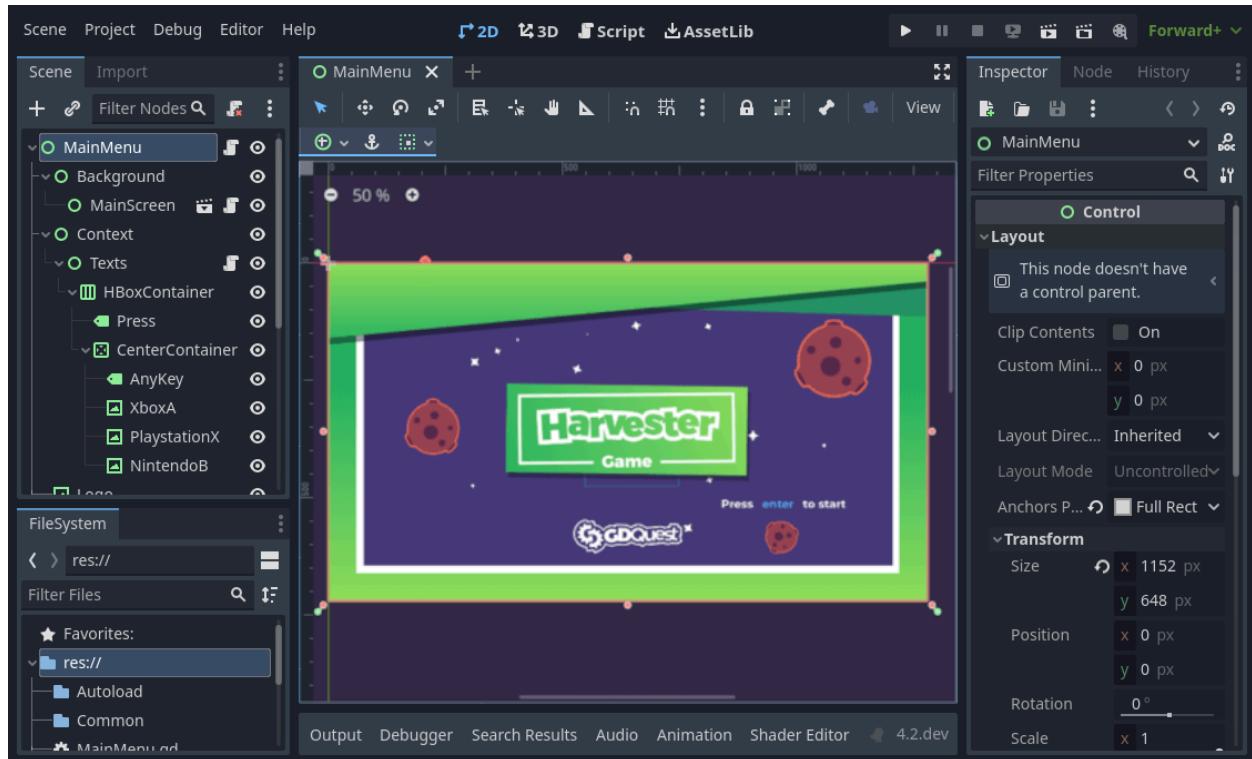
- **Năm 2014:** Godot 1.0 ra mắt, hỗ trợ xuất bản game trên PC, Mobile.
- **Năm 2016:** Godot 2.0 phát hành, cải thiện giao diện người dùng.
- **Năm 2018:** Godot 3.0 giới thiệu hệ thống renderer hoàn toàn mới dựa trên Vulkan (một API điện toán và là đồ họa 3D đa nền tảng).
- **Năm 2023:** Godot 4.0 chính thức ra mắt với cải tiến về hiệu suất, hỗ trợ đa nền tảng tốt hơn.

Hiện nay, Godot được xem là một trong những game engine mã nguồn mở mạnh mẽ, cạnh tranh trực tiếp với Unity và Unreal Engine trong lĩnh vực phát triển game 2D và 3D.

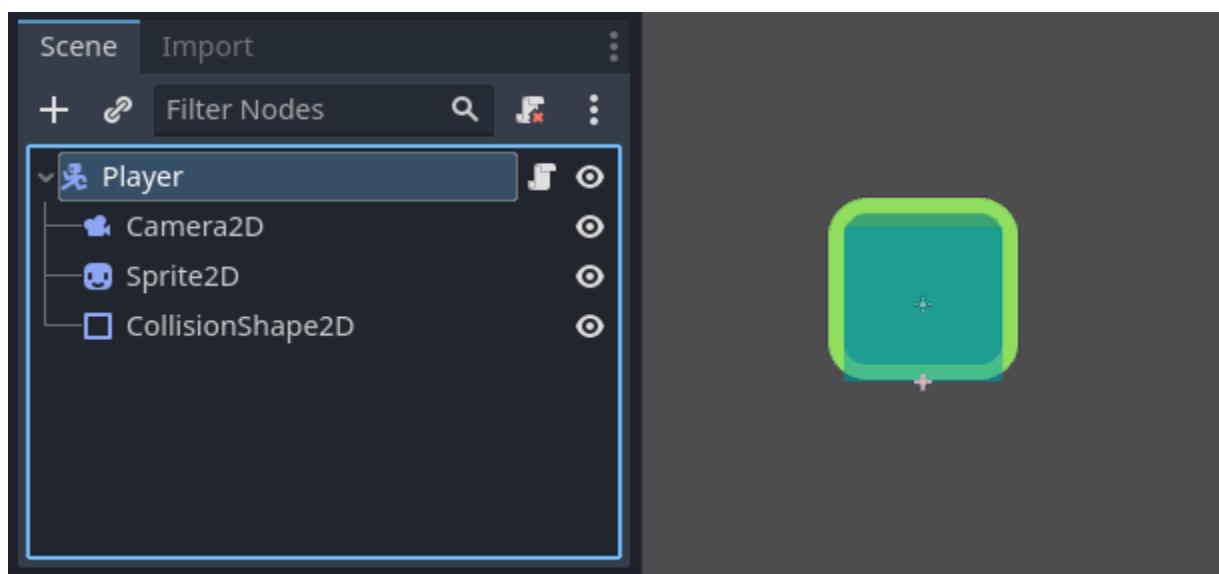
2. Công nghệ và kiến trúc Godot

Godot sử dụng **hệ thống Scene và Node**, giúp quản lý trò chơi một cách logic và dễ dàng mở rộng:

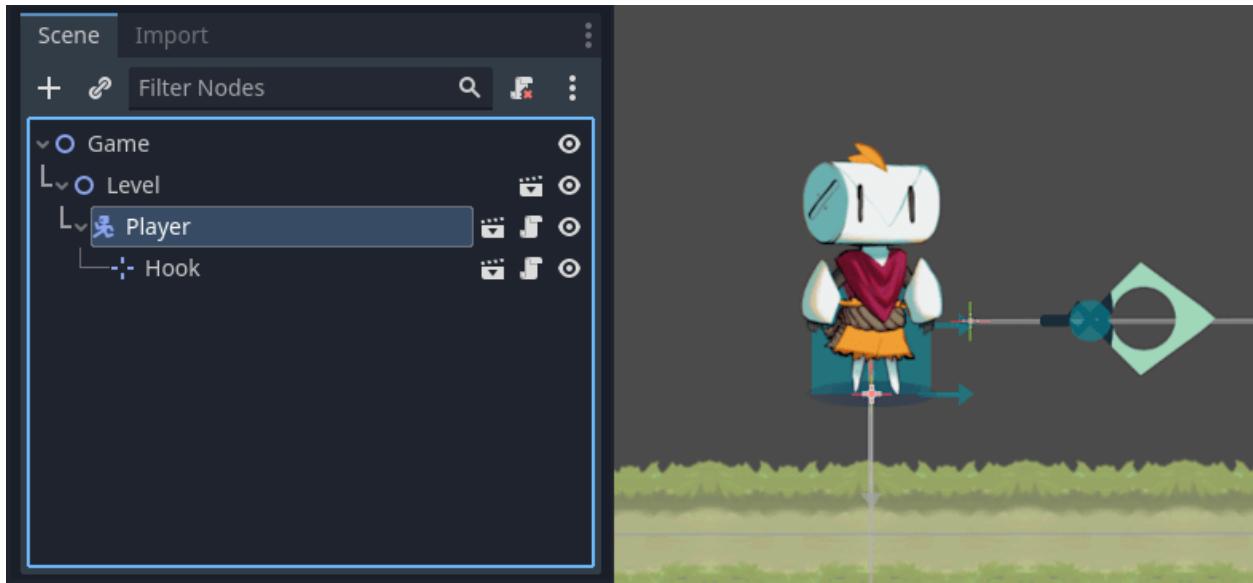
- **Scene**: Là một đơn vị tổ chức chính, Scene có thể là menu, giao diện người dùng, màn chơi,... bất cứ gì ta có thể nghĩ ra và trong Scene có thể chứa nhiều Node.



- **Node**: Là các thành phần cơ bản (như nhân vật, vật thể, âm thanh, UI).



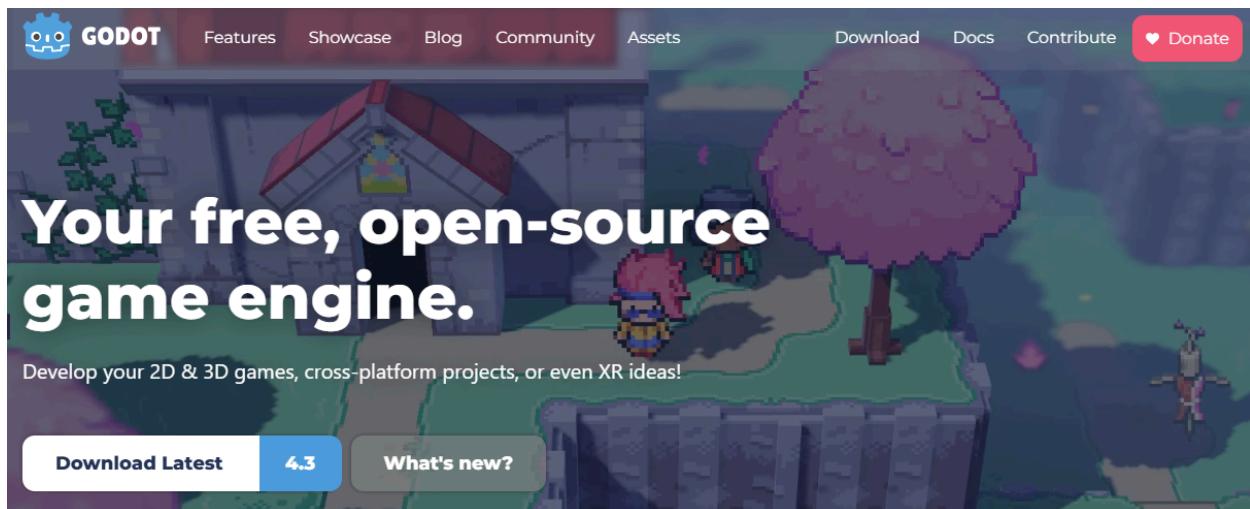
- **Tái sử dụng Scene:** Một Scene có thể được nhúng vào Scene khác, giúp lập trình viên xây dựng game theo mô hình module hóa.



3. Hướng dẫn tải và cài đặt Godot

Cách 1:

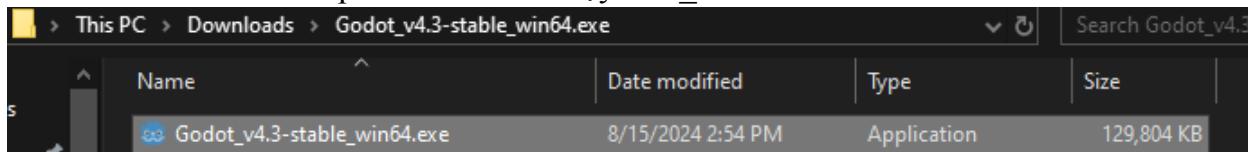
Bước 1: Đi đến trang <https://godotengine.org> và nhấn nút Download Latest



Bước 2: Nhấn nút xanh (Godot Engine) để tải

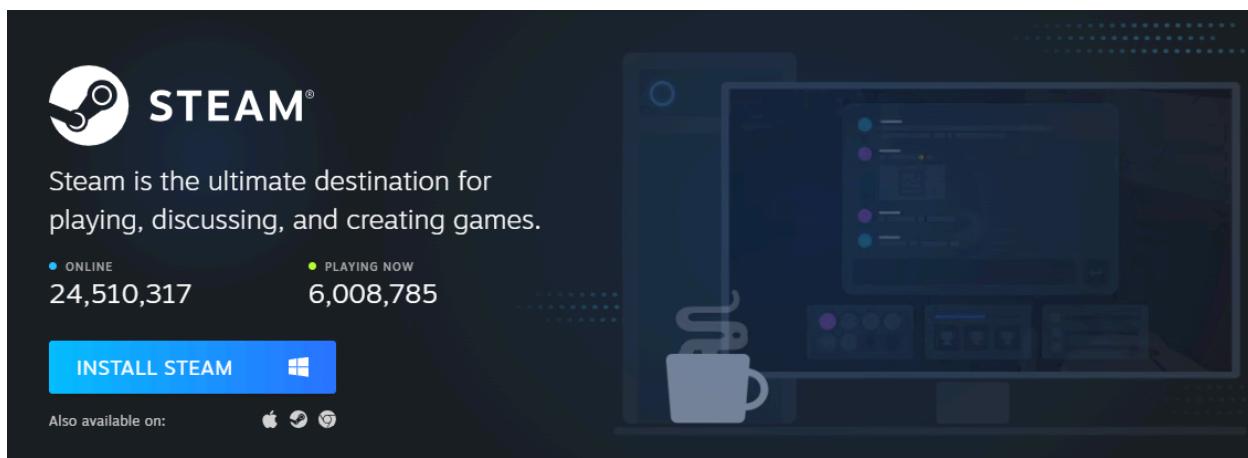


Bước 3: Giải nén file .zip vừa tải về và chạy file _wind64.exe

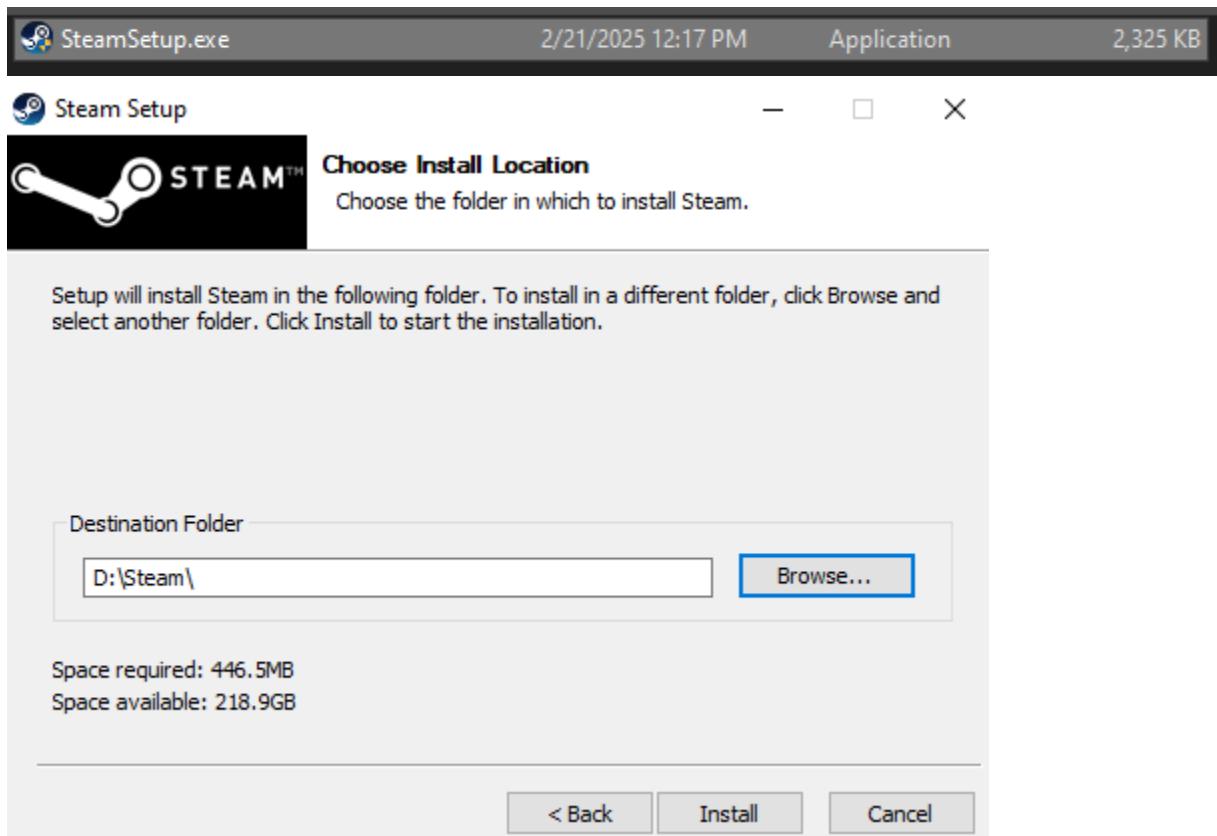


Cách 2:

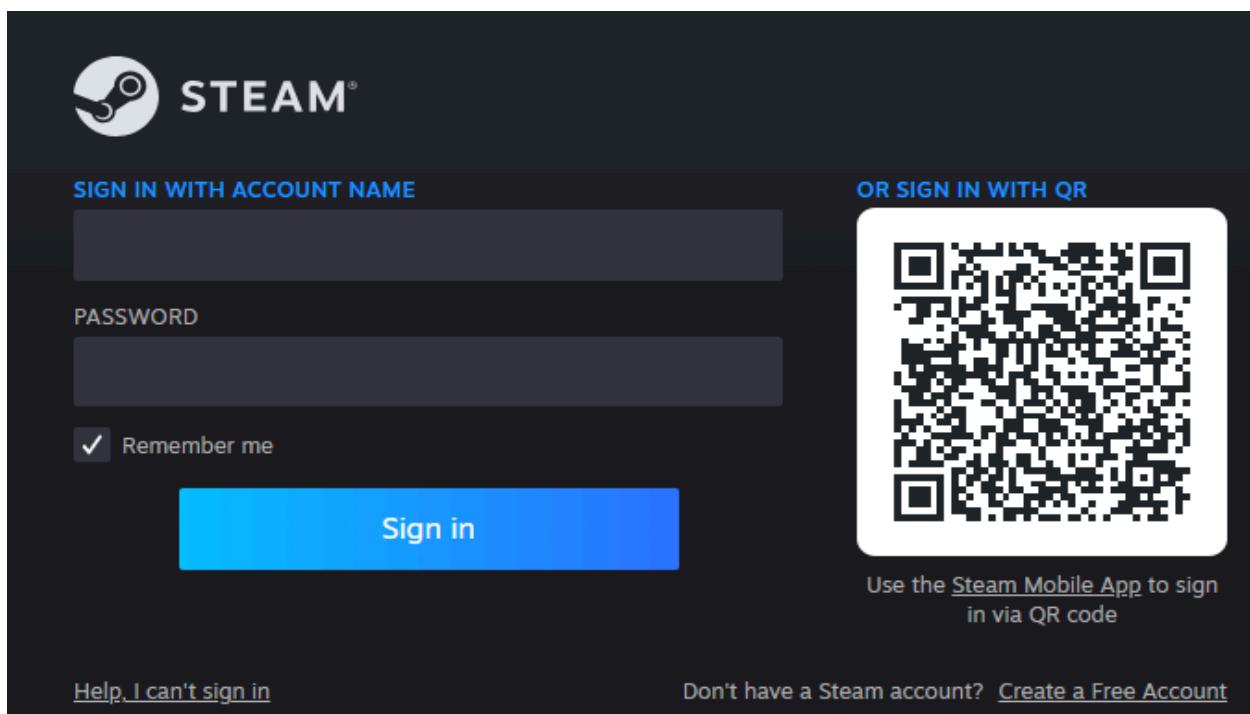
Bước 1: Đi đến trang <https://store.steampowered.com/about/> và cài đặt Steam



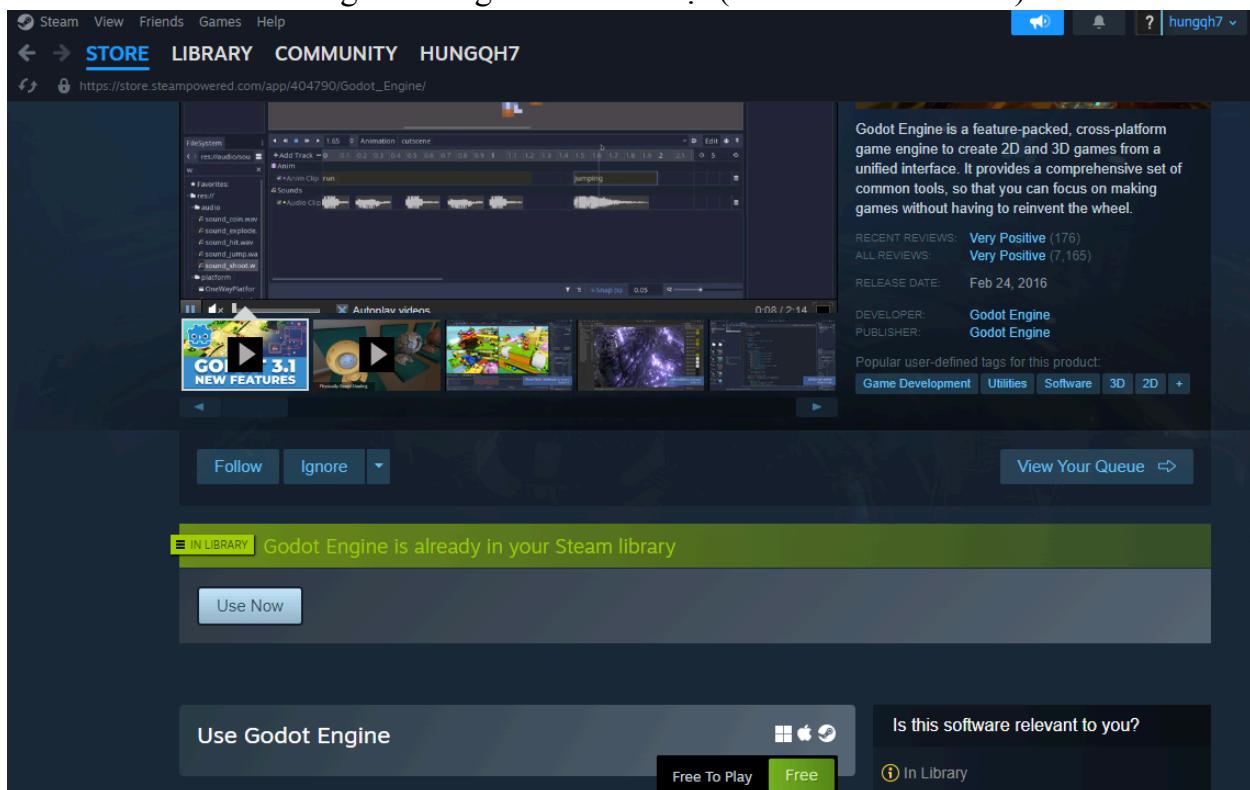
Bước 2: Chạy file .exe và chọn ô đĩa để cài đặt Steam



Bước 3: Khởi động Steam và đăng nhập (đăng ký nếu chưa có tài khoản)



Bước 4: Tìm Godot trong cửa hàng và nhấn cài đặt (nút free màu xanh lá)



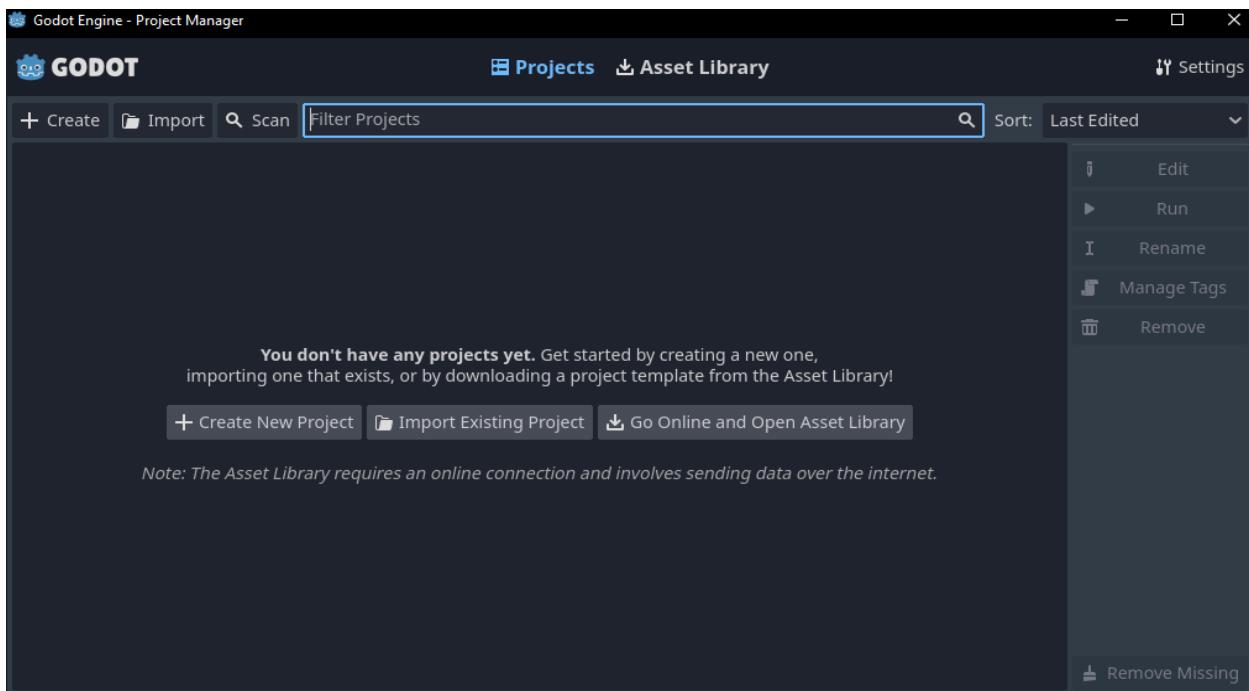
Bước 4: Sau khi chờ cài đặt xong bấm launch để chạy ứng dụng



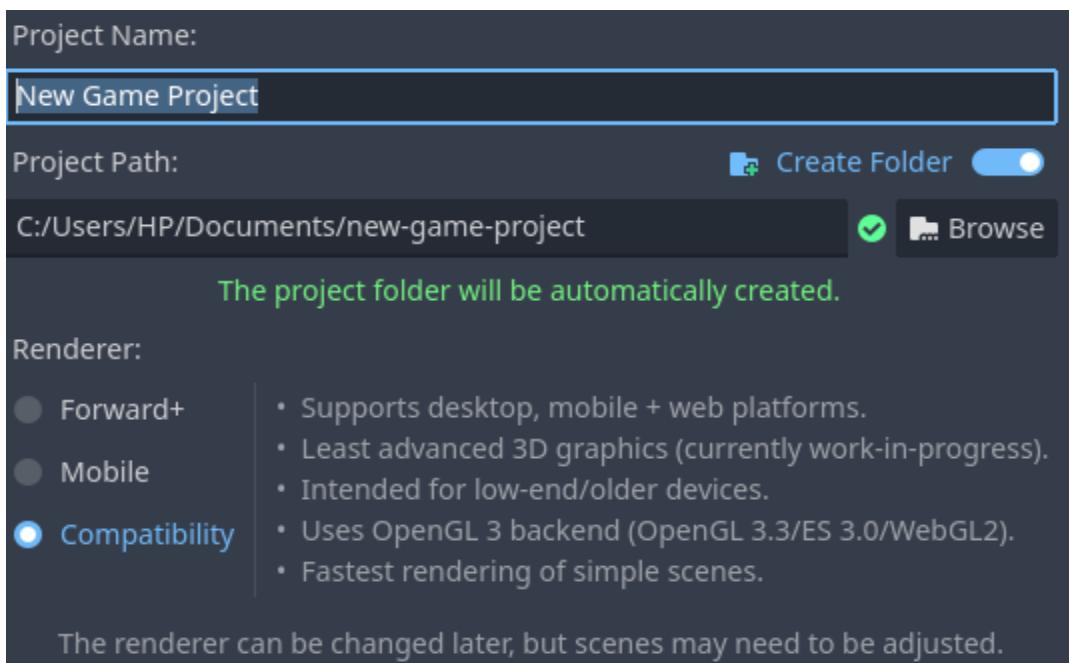
4. Ứng dụng Godot trong phát triển game

4.1. Làm quen với ứng dụng Godot

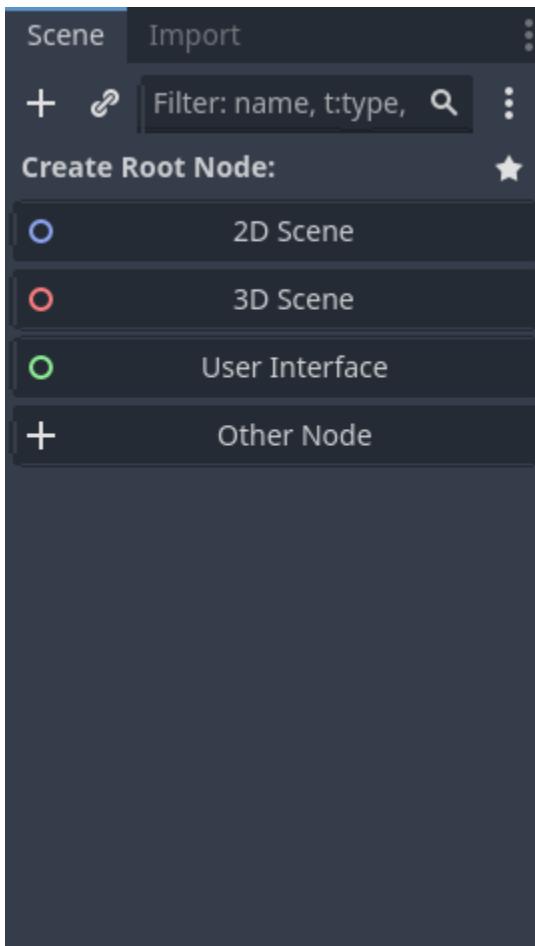
- Giao diện khi mở godot
- Nhấp vào Create New Project để tạo project mới



- Nếu thiết bị cũ, không hỗ trợ Vulkan API, hãy chọn Compatibility khi tạo project nếu có hãy chọn Forward+

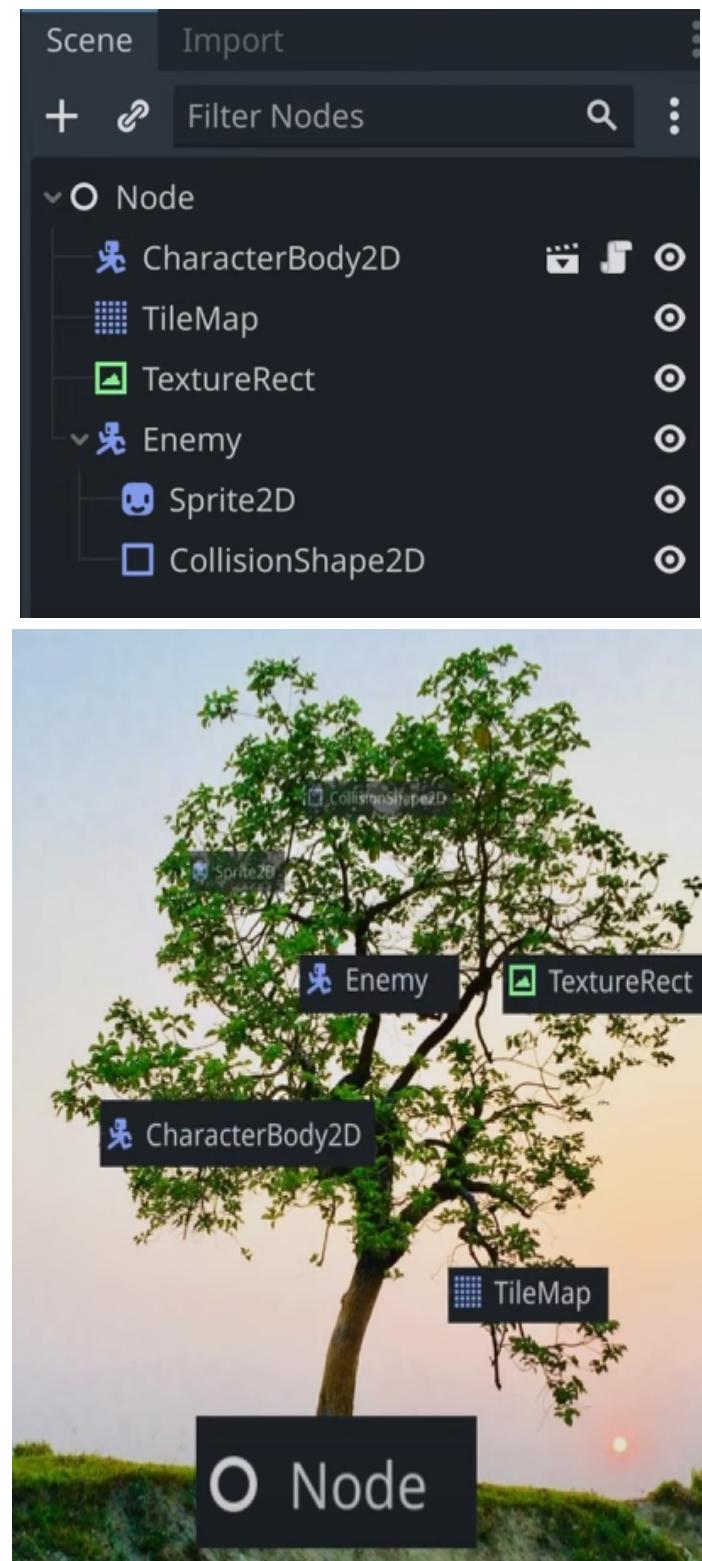


- Nhấn Create & Edit
- Tại giao diện tổng quan của Godot, góc bên trái trên ta sẽ thấy Godot cho những lựa chọn node để bắt đầu tạo game

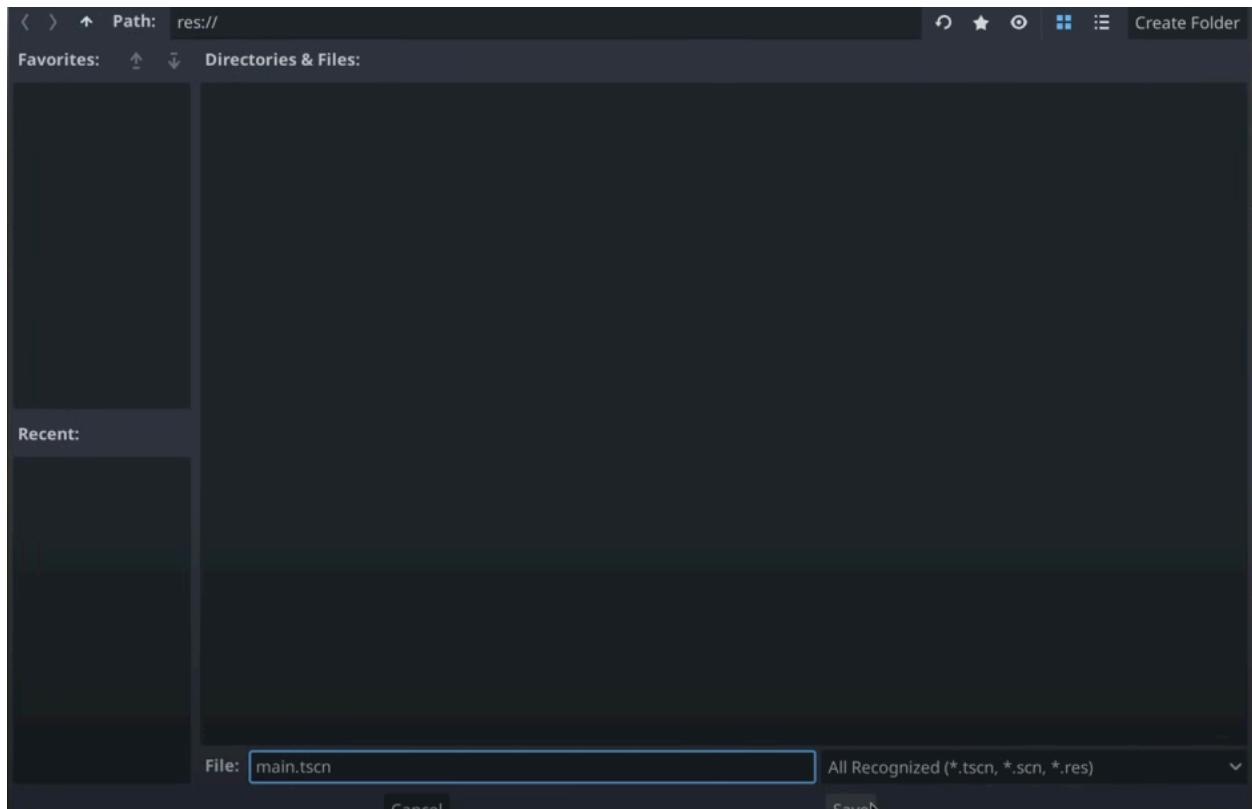


- Chọn Other Node để tạo Node rỗng, ta sẽ dùng Node rỗng này để chứa các thành phần khác nhau hơn, giúp ta quản lý game và các thành phần trong game theo sơ đồ cây

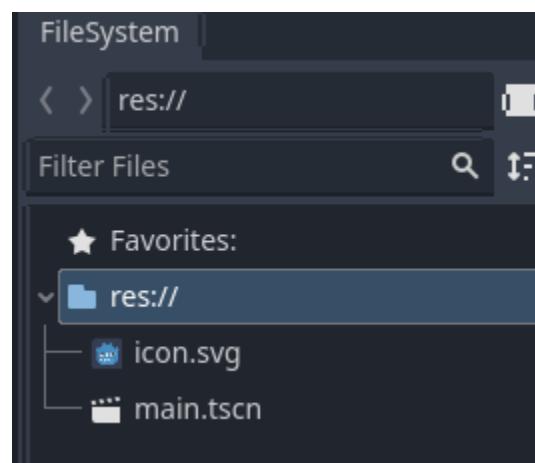
- Ảnh minh họa:



- Nhấn Ctrl + S để lưu và đặt tên file main.tscn, đây là 1 Scene tổng thể của project



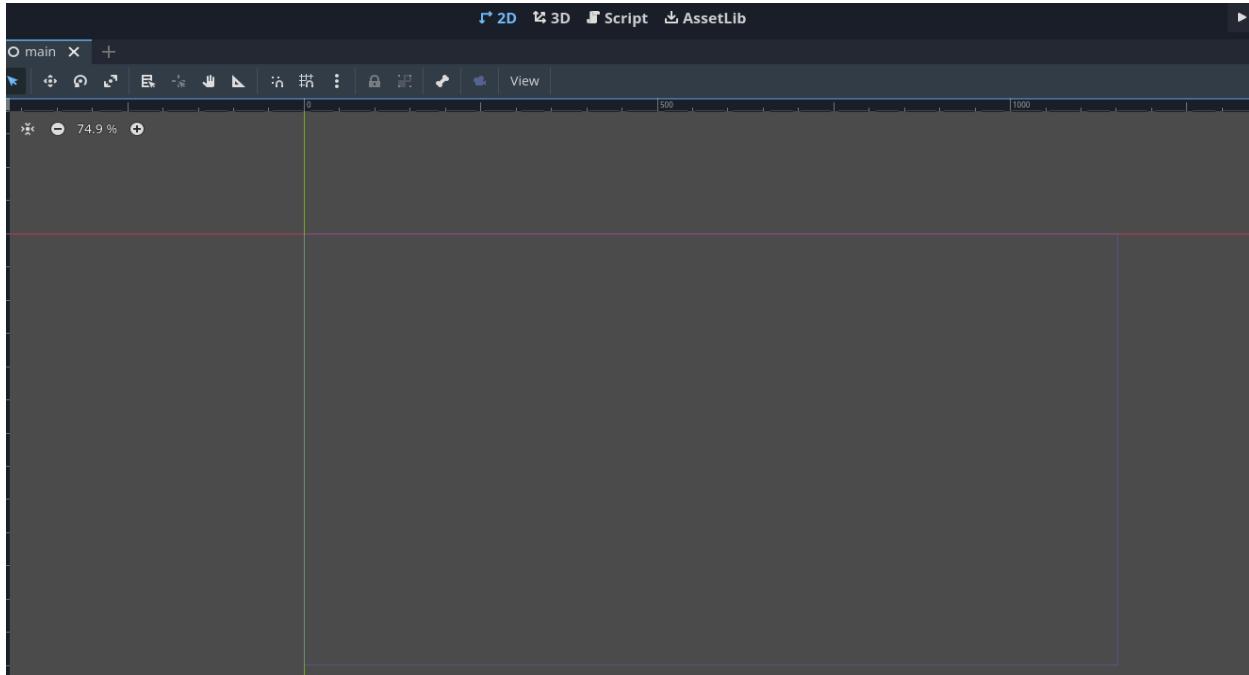
- Sau khi nhấn nút Save thì khu vực trái dưới của màn hình chính ta sẽ thấy Scene (main.tscn) đã được lưu



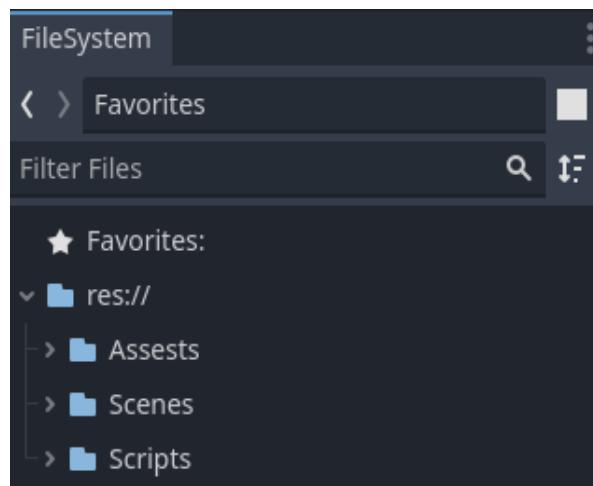
4.2. Làm game Platform 2D với Godot

4.2.1. Thiết kế background

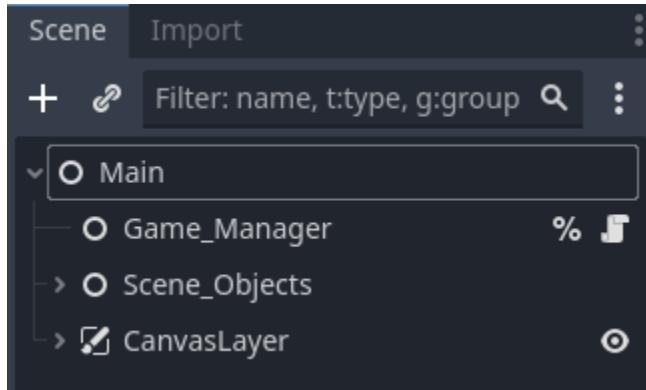
- Đây sẽ là Scene tổng, game sẽ được thiết kế tại đây và nó bao gồm 3 yếu tố:
- + Background: Hình ảnh nền của màn chơi
- + Ground blocks (Tileset): những khối giúp cho nhân vật đứng lên
- + Character (Player): Người chơi hoặc cũng có thể là quái



- Ta sẽ dùng assets miễn phí trên trang itch.io có tên: Pixel Adventure được làm bởi Pixel Frog để thiết kế game (cre: <https://pixelfrog-assets.itch.io/pixel-adventure-1>)
- Trước hết ta cần tạo các thư mục mới (Chuột phải -> New Folder) như hình để dễ quản lý các thành phần (menu góc trái dưới)



- Chọn Background tùy ý trong asset Pixel Adventure và kéo vào thư Assets trong FileSystem
- Tạo các Node rỗng như trong hình để dễ quản lý các thành phần trong menu Scene (menu góc trái trên)

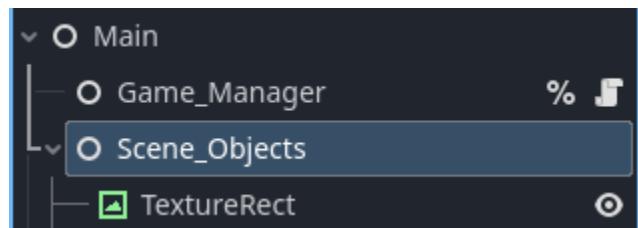


- Để tạo thì chỉ cần nhấn dấu **+** và tìm Node (CanvasLayer cũng tương tự)

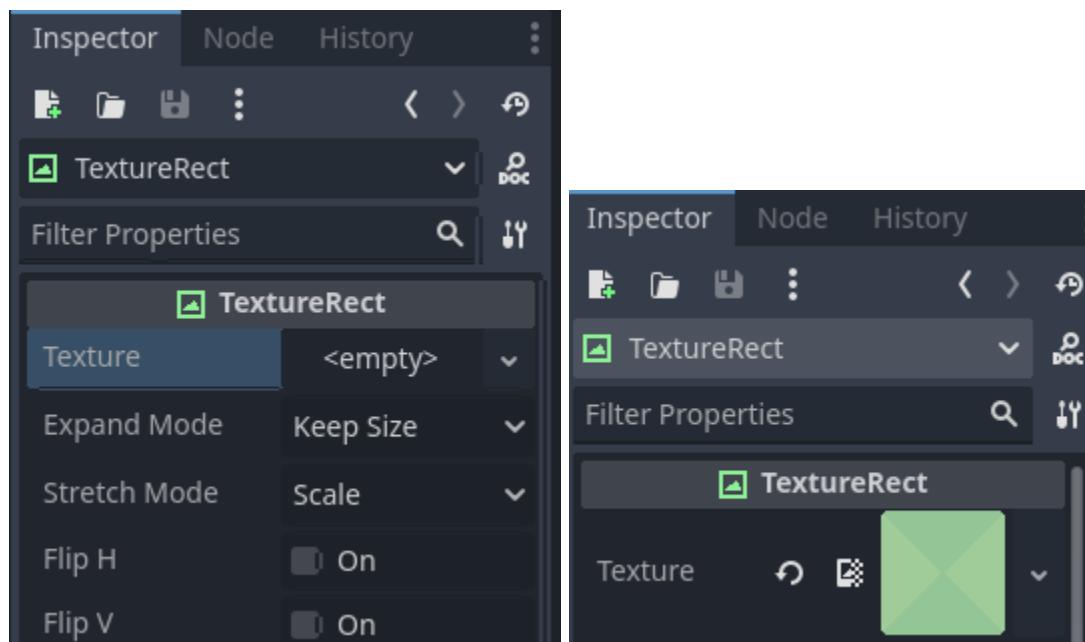


- Sau đó đổi tên các Node (Game_Manger, Scene_Objects, CanvasLayer)
- Giải thích thêm về các Node rỗng này:
 - + Main: là Node tổng chứa toàn bộ mọi thứ trong màn chơi
 - + Game_Manager: là Node chứa logic code, cập nhật những gì diễn ra trong màn chơi
 - + Scene_Objects: là Node chứa toàn bộ những thành phần trong game (nhân vật, thiết kế màn chơi, kẻ địch, vật phẩm,...)
 - + CanvasLayer: là Node chứa những thành phần hiển thị UI như điểm số của người chơi
- **TextureRect** là một node được sử dụng để hiển thị hình ảnh trong giao diện người dùng (UI)

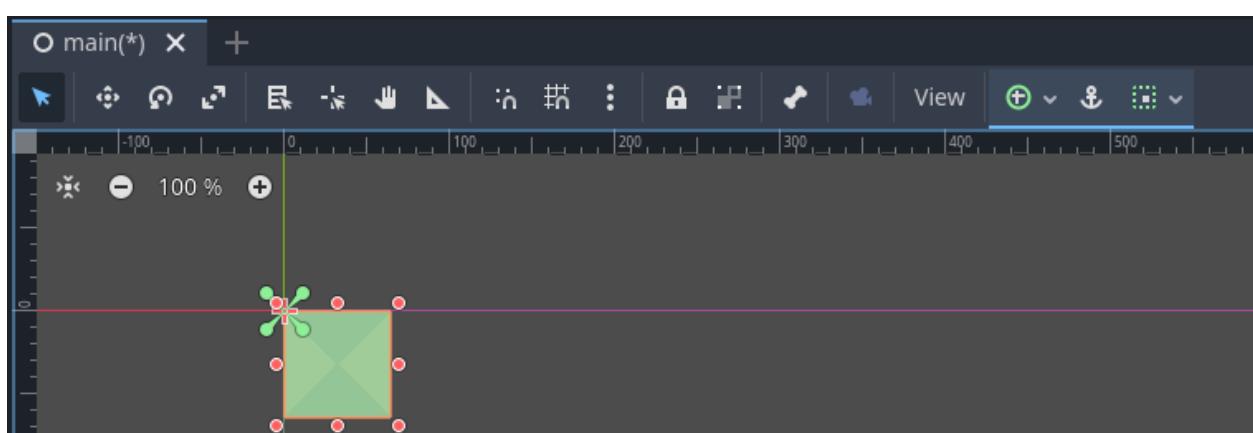
- Chuột phải vào Scene_Objects (hoặc Ctrl+A) và tìm TextureRect và nhập Create
- TextureRect sẽ là Node để chứa background



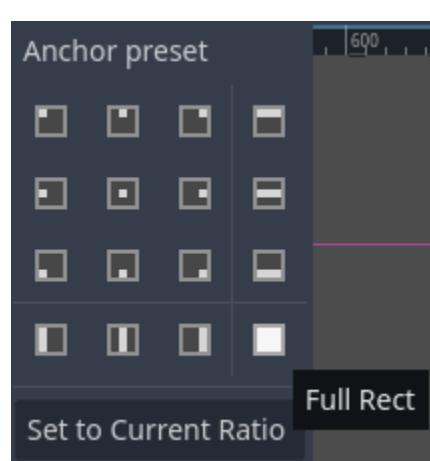
- Kéo thả Background từ FileSystem vào mục Texture trong menu Inspector (menu góc phải trên)



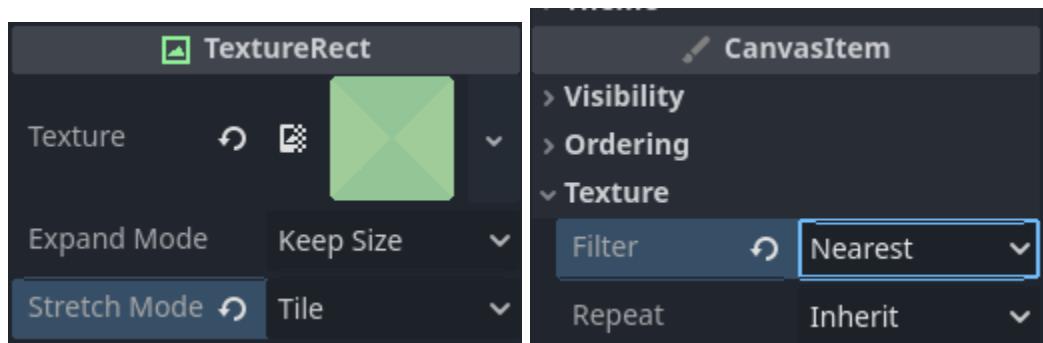
- Ta sẽ thấy được khôi background giữa màn hình xám



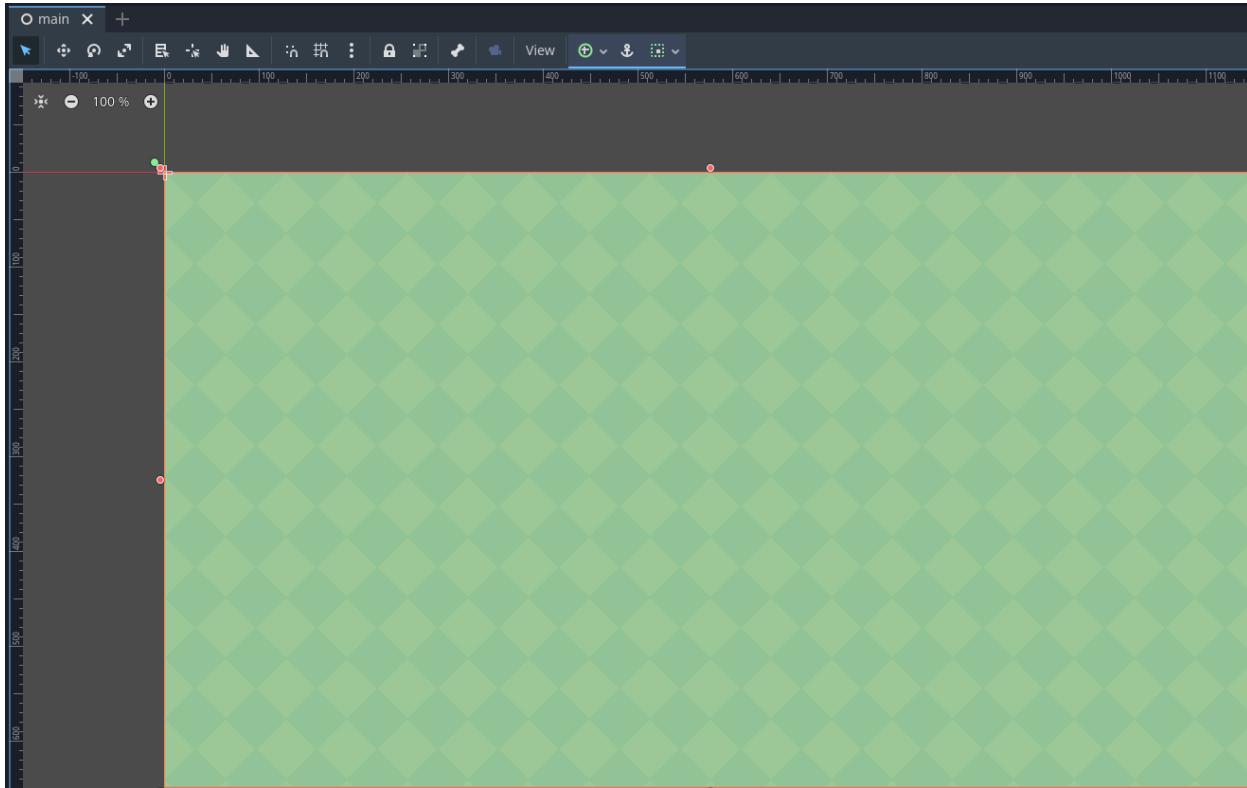
- Tuy nhiên việc chỉnh sửa kích thước background bằng tay sẽ không hay khi game có sự thay đổi về kích thước màn hình
- Để làm cho Background có tính linh hoạt ta nhấp vào biểu tượng  và chọn Full Rect để kích thước Background tự động tương thích với toàn màn hình



- Lúc này Background sẽ to ra và lấp đầy màn hình nhưng hình ảnh quá mờ và bị kéo giãn, để chỉnh sửa lại ta vào menu Inspector (menu bên phải) chọn Texture -> Filter -> Nearest và TextureRect -> Stretch Mode -> Tile

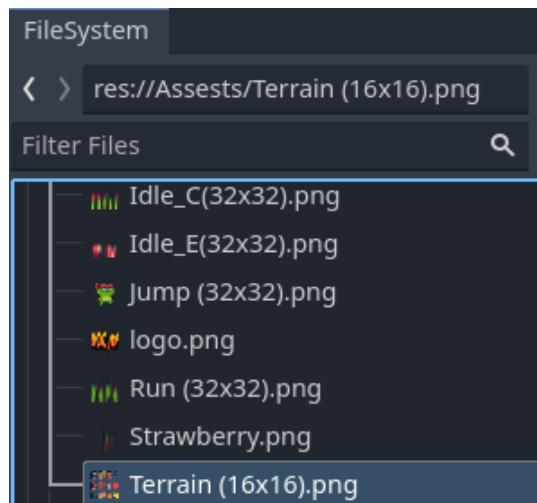


- Kết quả background:

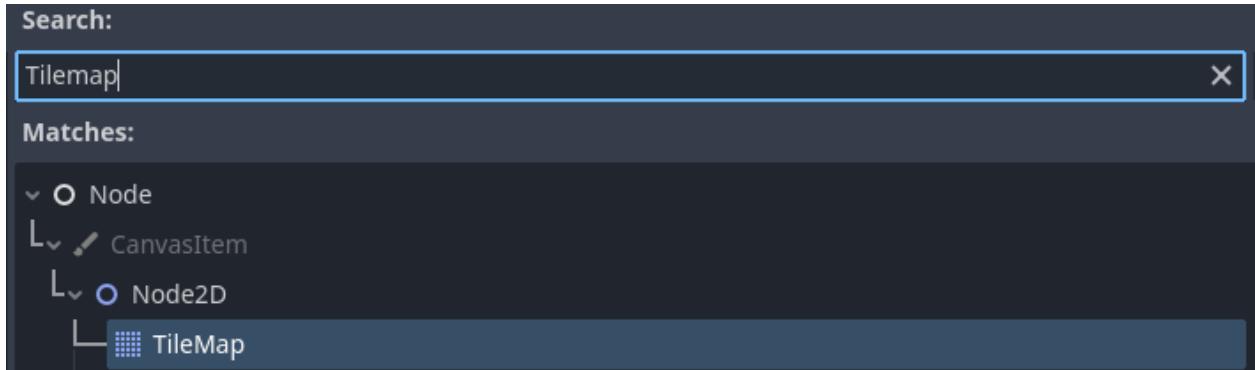


4.2.2. Thiết kế màn chơi

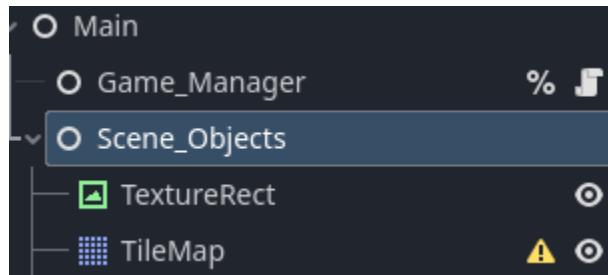
- **TileSet** là một tập hợp các tiles (ô vuông) có thể sử dụng để vẽ bản đồ trong TileMap. Nó giống như một bộ sưu tập các viên gạch có thể đặt vào bản đồ để tạo môi trường game
- Ta cần Tileset để tạo môi trường cho nhân vật đứng lên, kéo file Terrain.png vào thư mục Assests trong FileSystem



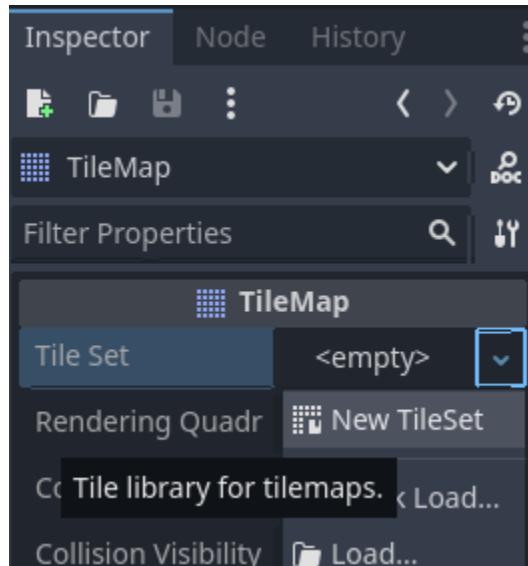
- **TileMap** là một Node dùng để vẽ bản đồ bằng cách sử dụng các tile từ TileSet
- Chuột phải vào Scene_Objects chọn **+ Add Child Node...** để thêm Node con
- Tìm và chọn TileMap



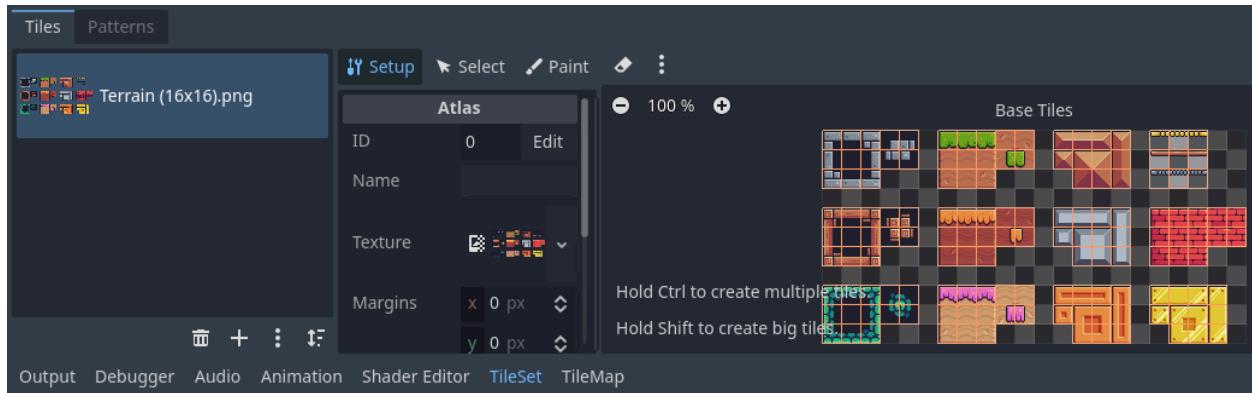
- Ta có kết quả như sau:



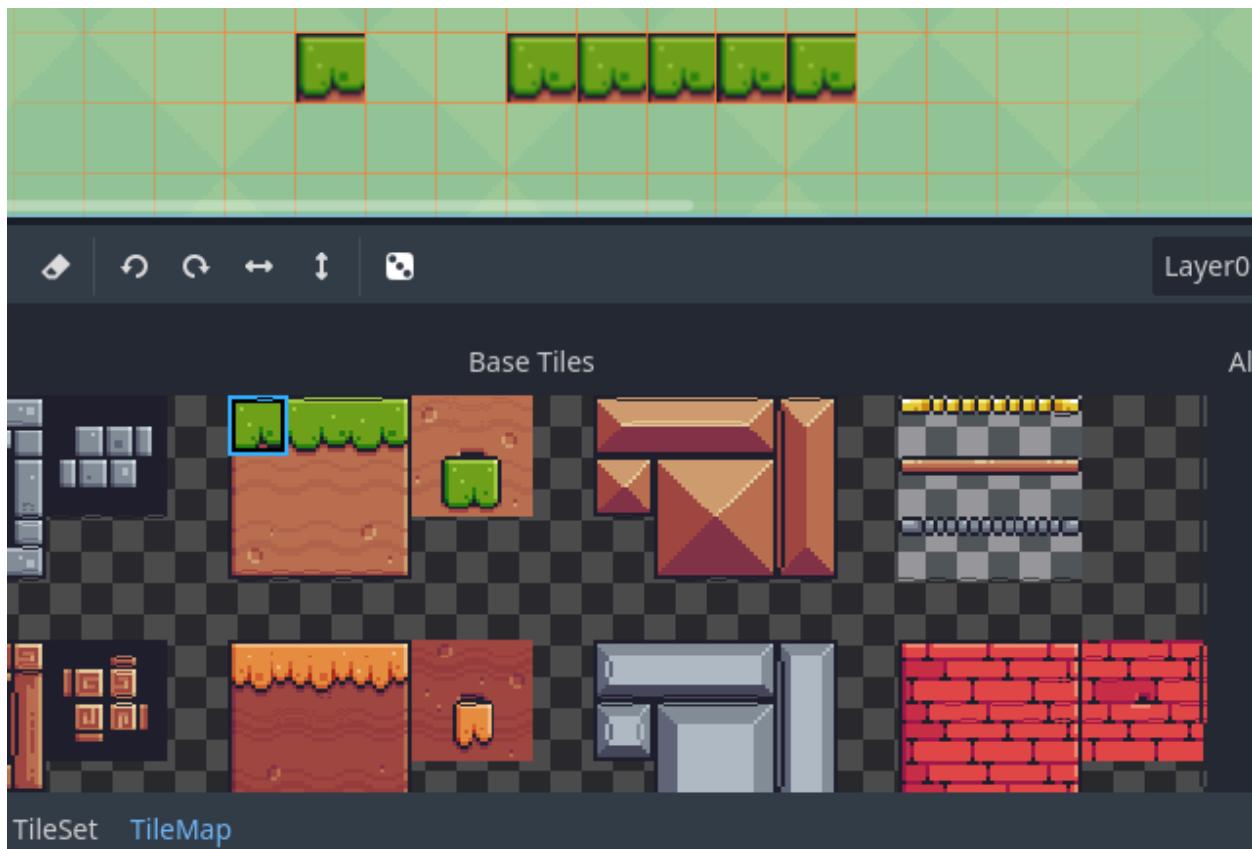
- Chọn New TileSet ở menu Inspector (menu bên phải)



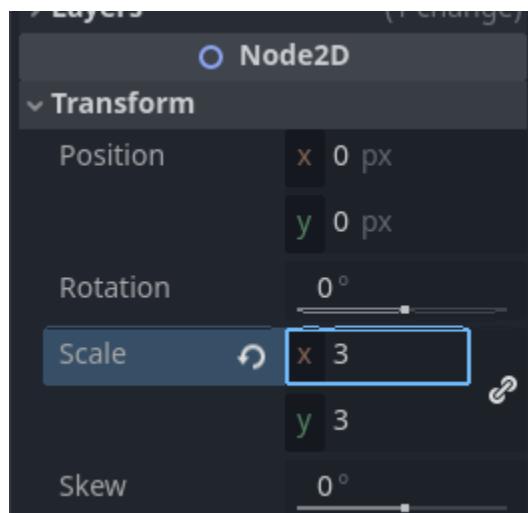
- Kéo Terrain.png vào menu dưới giữa màn hình để tiến hành thiết kế các khối để làm game



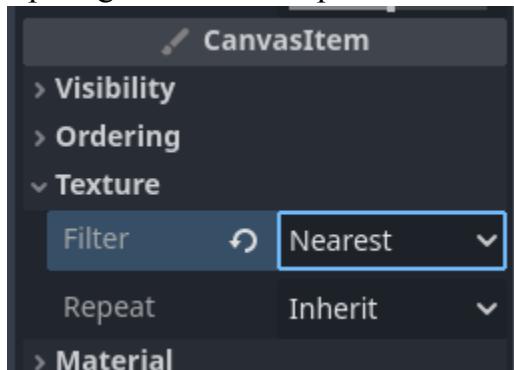
- Chọn TileMap, bắt đầu chọn các khối tùy ý và thiết kế màn chơi bằng cách nhấp chuột phải hoặc giữ chuột phải kéo để vẽ màn chơi (chuột phải hoặc kéo giữ chuột phải để xoá hoặc Ctrl + Z)



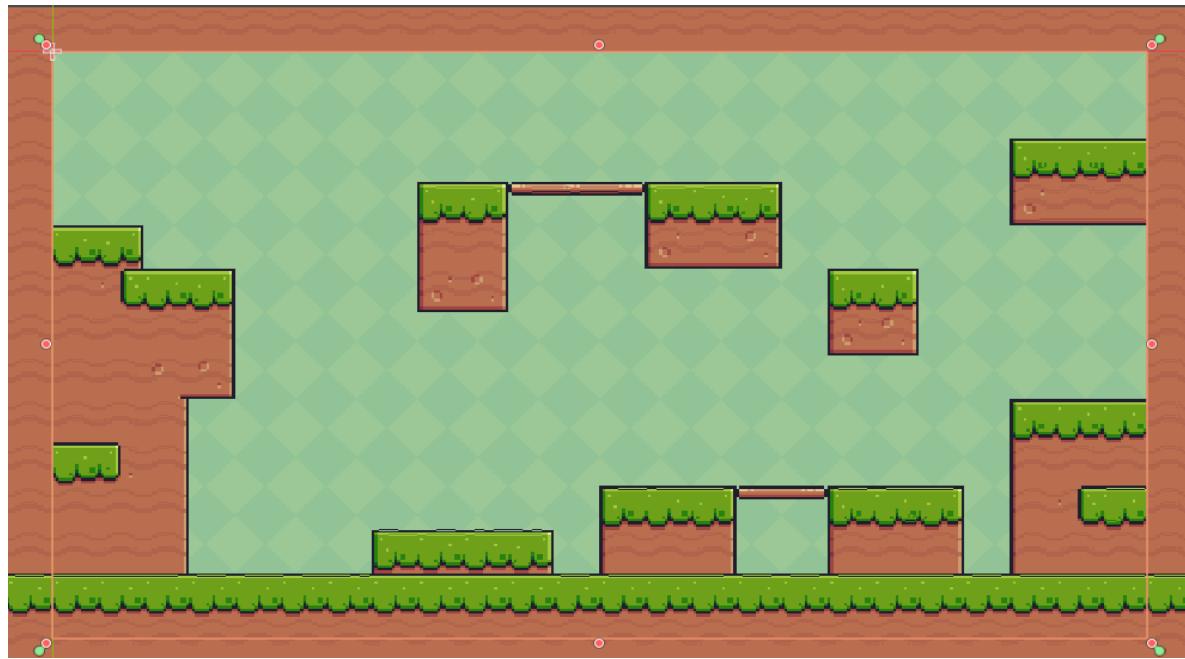
- Lúc này các khối nhìn khá nhỏ, muốn phóng to chỉ cần vào menu Inspector chọn Transform -> Scale của x và y = 3



Hình ảnh sẽ bị vỡ do phóng to chỉ cần nhấp vào: Texture -> Filter -> Nearest



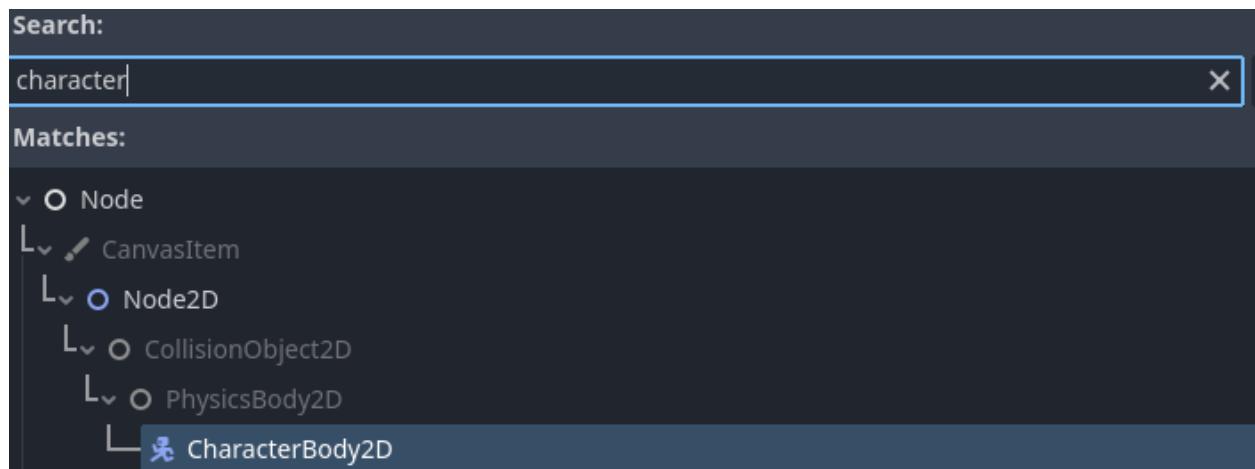
- Đây là 1 màn chơi minh họa:



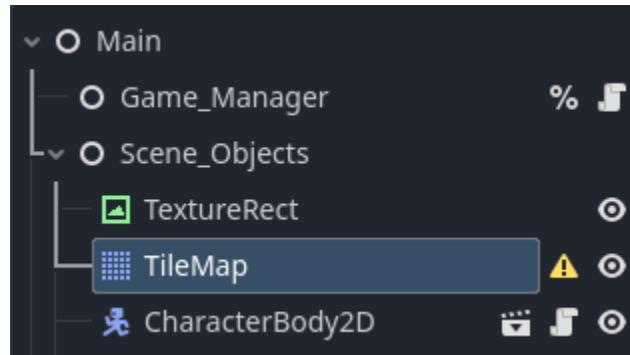
4.2.3. Thiết kế nhân vật và hiệu ứng chuyển động

- **CharacterBody2D** là một node trong Godot dành riêng cho các nhân vật di chuyển trong game 2D. Nó giúp dễ dàng xử lý va chạm, trọng lực, ma sát, và di chuyển mà không cần tự viết lại các thuật toán vật lý phức tạp

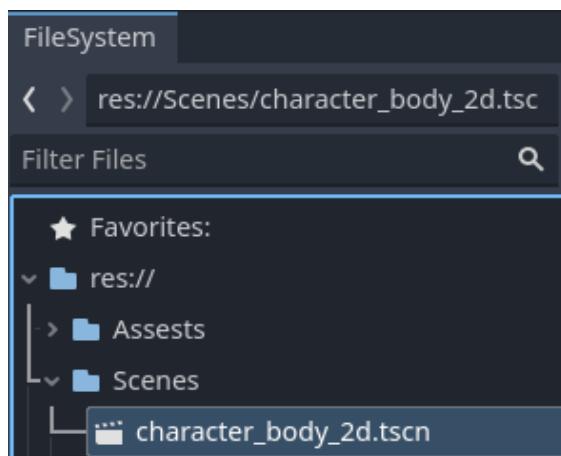
- Chuột phải vào Scene_Objects chọn **+ Add Child Node...** Ctrl+A để thêm Node con
- Tìm và chọn Node CharacterBody2D



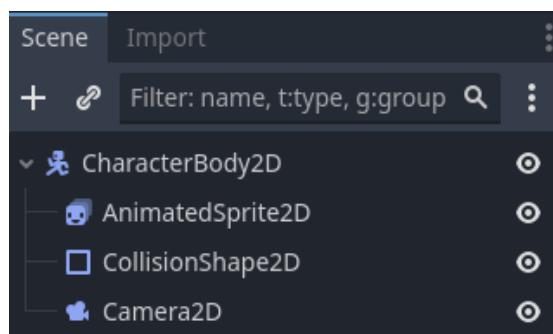
- Ta có kết quả như sau:



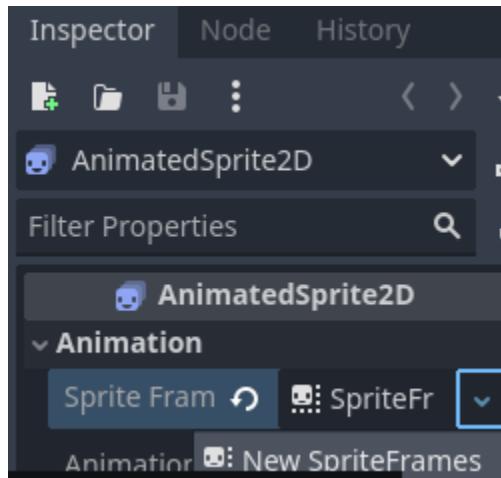
- Trong 1 nhân vật bao gồm: Sprite (hình ảnh) và Collider (vật lý của nhân vật)
- Trước khi thiết kế các thành phần bên trong ta cần lưu nhân vật thành Scene độc lập để khi chỉnh sửa tránh ảnh hưởng Scene tổng
- Chuột phải vào CharacterBody2D chọn **Save Branch as Scene...** và lưu vào trong thư mục Scenes trong FileSystem



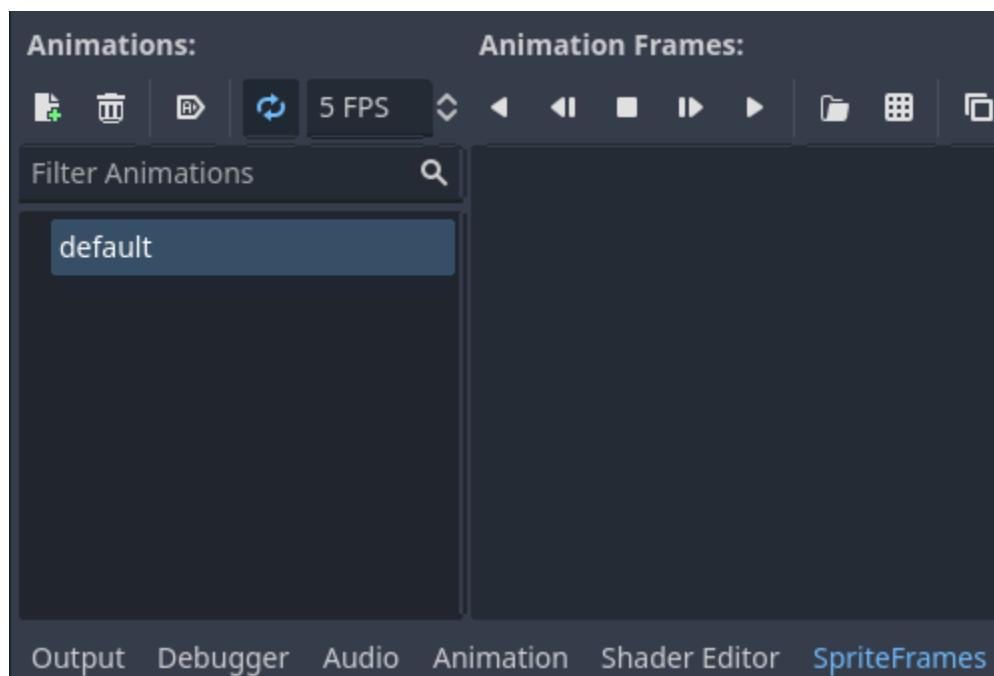
- Do nhân vật là 1 Scene độc lập nên khi nhấp đôi đúp để chọn thì ta thấy trong menu Scene sẽ hiển thị mỗi CharacterBody2D và các thành phần bên trong, khi cập nhật thì nhân vật trong Scene tổng của game cũng sẽ được cập nhật



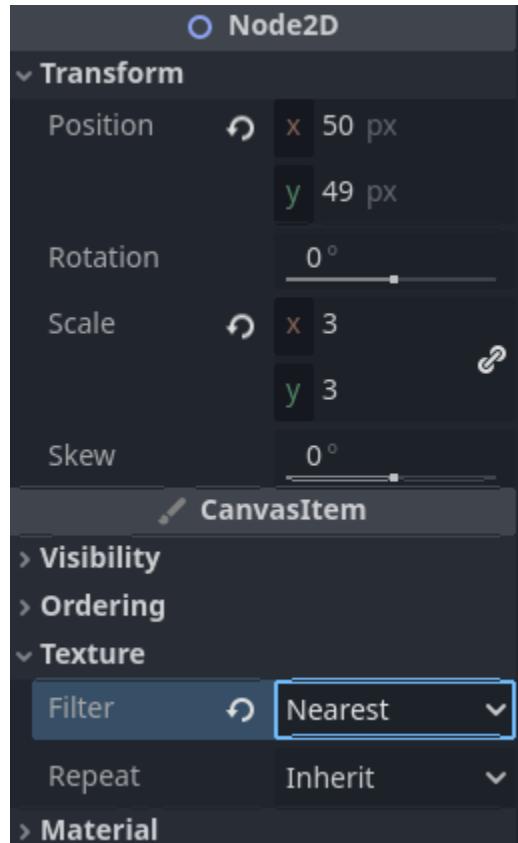
- **AnimatedSprite2D** là một node trong Godot dùng để hiển thị hình ảnh động (sprite animation) trong game 2D. Nó giúp nhân vật hoặc vật thể có thể chuyển động mượt mà bằng cách hiển thị nhiều khung hình (frames) liên tiếp.
- Tại menu Inspector chọn New SpriteFrame



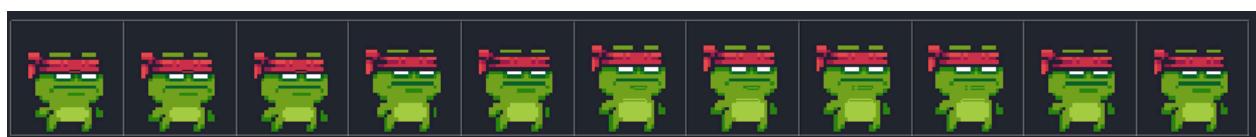
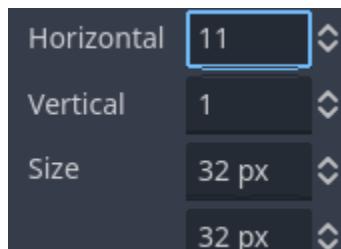
- Hiệu ứng của nhân vật đưa chia thành nhiều khung hình, khi game chạy sẽ lặp các khung hình đó liên tục tạo thành hiệu ứng
- Kéo file hiệu ứng vào thư mục Assets và chọn SpriteFrames để mở menu bên dưới giữa màn hình



- Chính Scale x và y = 3 và Texture -> Nearest để hình ảnh nhân vật không bị nhòe



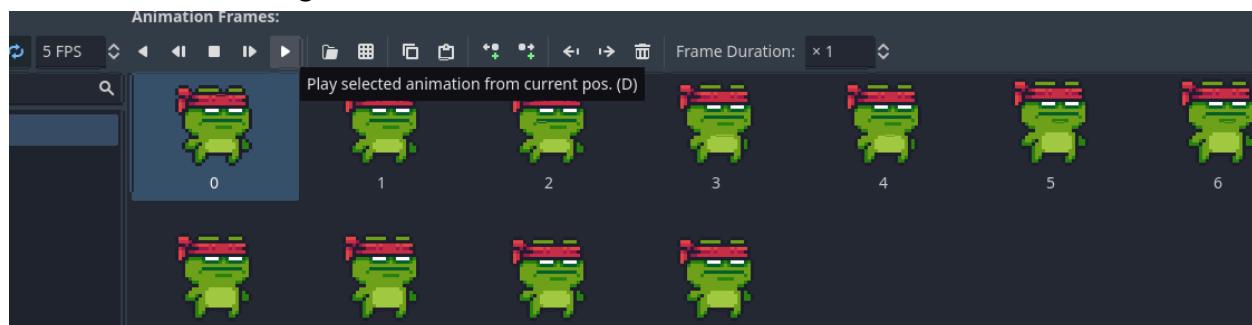
- Nhấp vào biểu tượng -> chọn file hiệu ứng trước đó đã thêm -> Chia hiệu ứng thành từng khung hình



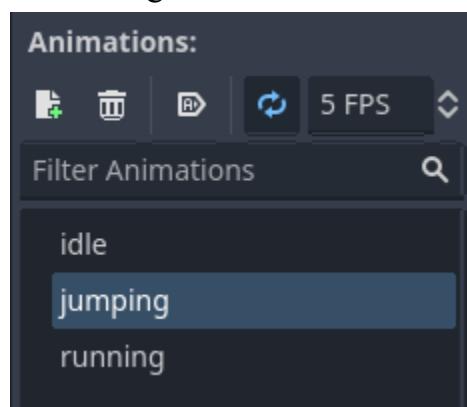
- Kéo giữ chuột trái để chọn từng khung hình và xác nhận



- Có thể chạy xem trước bằng phím D, có thể chỉnh FPS (khung hình mỗi giây) tăng lên để hiệu ứng nhân vật nhanh hơn



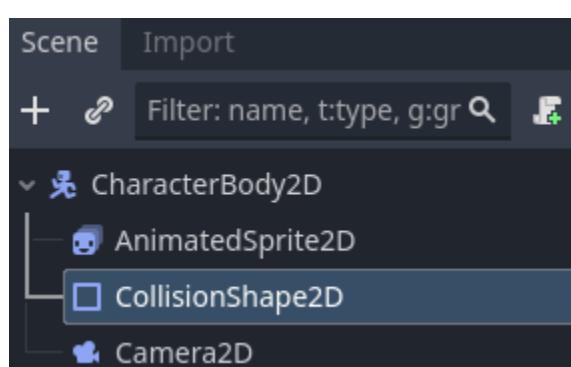
- Nhấn nút để thêm hiệu ứng



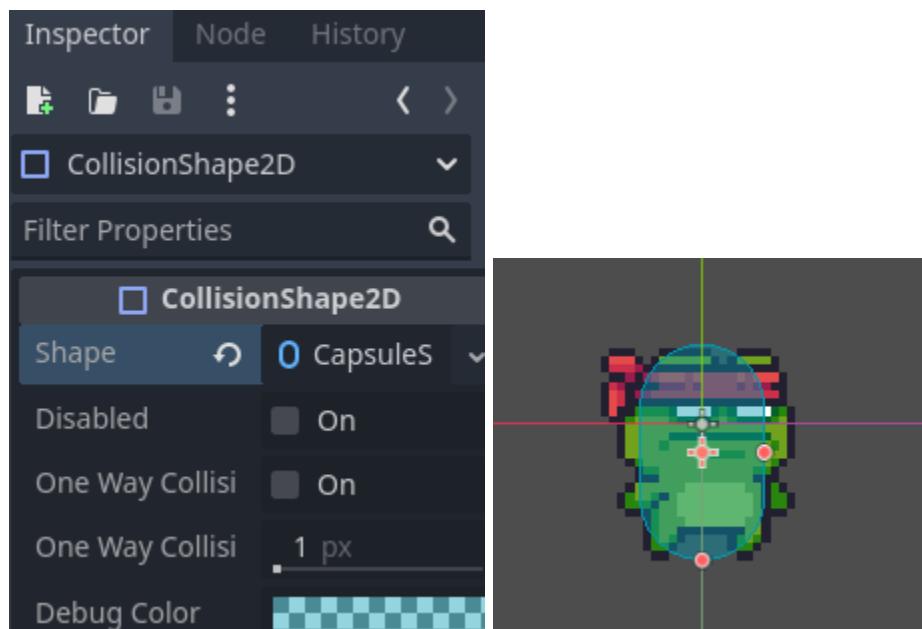
- Thực hiện việc chia khung hình tương tự cho hiệu ứng nhảy và chạy
- Nhấp vào biểu tượng đối với hiệu ứng nhân vật đứng im (idle) để hiệu ứng này được mặc định chạy khi bật game
- Có thể thấy nhân vật có hiệu ứng khi đứng im, hiệu ứng chạy và nhảy cần được chỉnh sửa trong Script của nhân vật

4.2.4. Thiết kế vật lý cho nhân vật

- **CollisionShape2D** là một node trong Godot dùng để xác định vùng va chạm của một đối tượng trong game 2D. Nó không hiển thị trên màn hình nhưng giúp kiểm soát va chạm giữa các đối tượng khi kết hợp với các node vật lý
- Sau khi đã có nhân vật, ta cần tạo hệ thống vật lý cho nhân vật tương tác với môi trường
 - Chuột phải vào CharacterBody2D chọn **+ Add Child Node...** Ctrl+A
 - Tìm và chọn CollisionShape2D

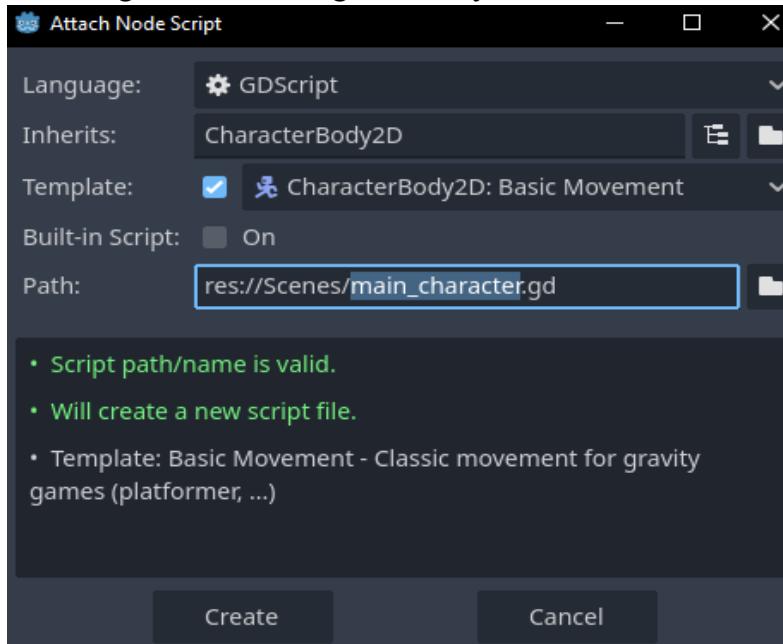


- Trong menu Inspector chọn Shape -> CapsuleShape2D và chỉnh sửa hình vuông tùy ý, đây sẽ là Hitbox của nhân vật



4.2.5. Thiết kế logic chuyển động nhân vật

- Để nhân vật có thể di chuyển trong màn chơi, ta cần cho nhân vật 1 Script sử dụng ngôn ngữ GDScript bằng cách nhập vào biểu tượng  góc trái trên màn hình sau khi chọn CharacterBody2D
- Hệ thống sẽ tự tạo 1 logic di chuyển cơ bản cho nhân vật, có thể chỉnh sửa tùy ý

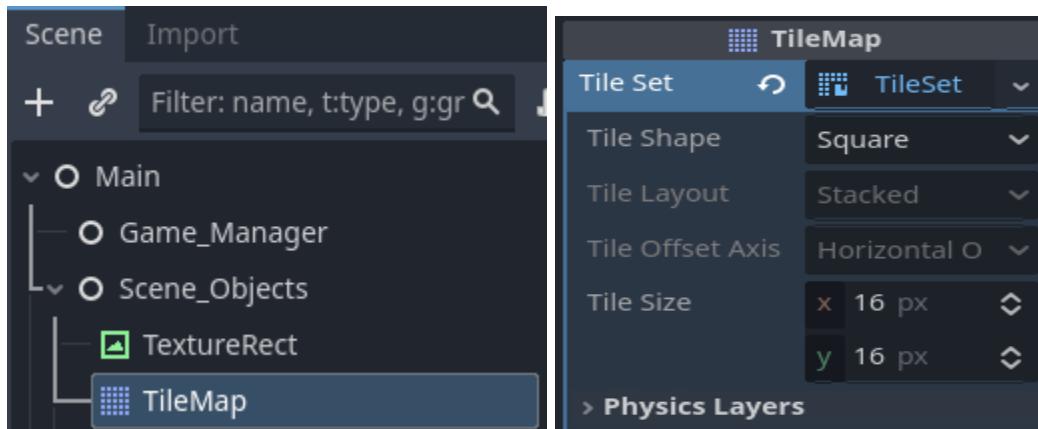


```
1  extends CharacterBody2D
2
3
4  const SPEED = 300.0
5  const JUMP_VELOCITY = -400.0
6
7
8  func _physics_process(delta: float) -> void:
9    # Add the gravity.
10   if not is_on_floor():
11     velocity += get_gravity() * delta
12
13   # Handle jump.
14   if Input.is_action_just_pressed("ui_accept") and is_on_floor():
15     velocity.y = JUMP_VELOCITY
16
17   # Get the input direction and handle the movement/deceleration.
18   # As good practice, you should replace UI actions with custom gameplay actions.
19   var direction := Input.get_axis("ui_left", "ui_right")
20   if direction:
21     velocity.x = direction * SPEED
```

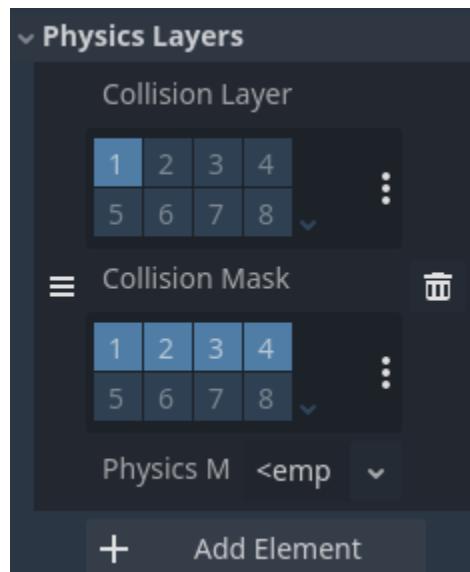
- Lúc này khi chạy game, nhân vật sẽ rời ra ngoài màn chơi, vẫn để là nhân vật đã có logic di chuyển cùng hitbox nhưng màn chơi thì chưa có hệ thống vật lý để va chạm với hitbox nhân vật dẫn tới việc nhân vật rời xuyên màn chơi

4.2.6. Thiết kế vật lý cho màn chơi

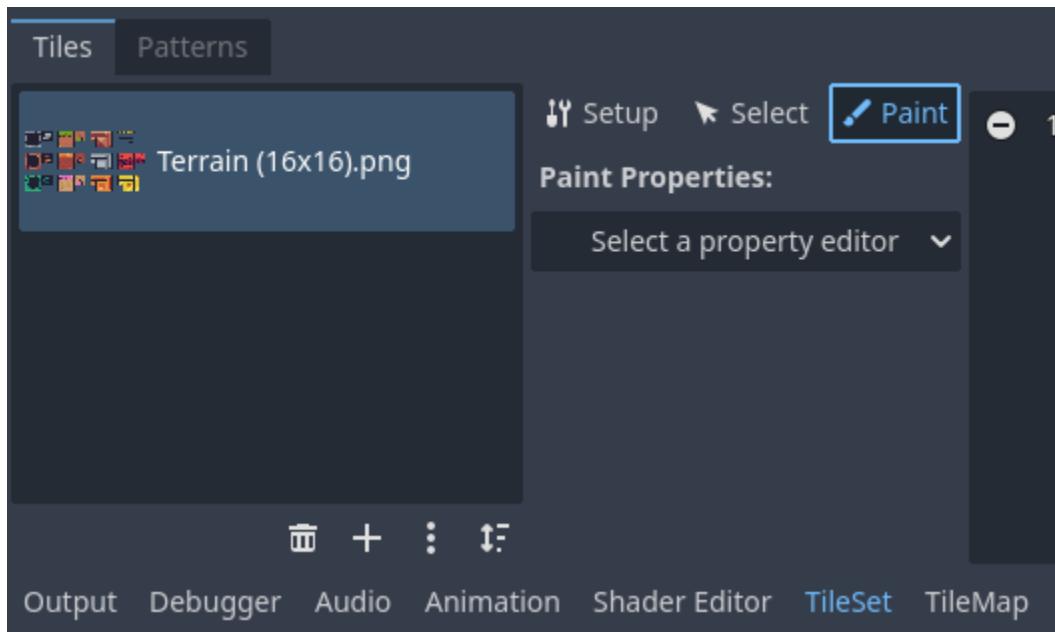
- Chọn TileMap ở menu Scene (góc trái trên) và nhấp vào TileSet ở menu Inspector (góc phải trên)



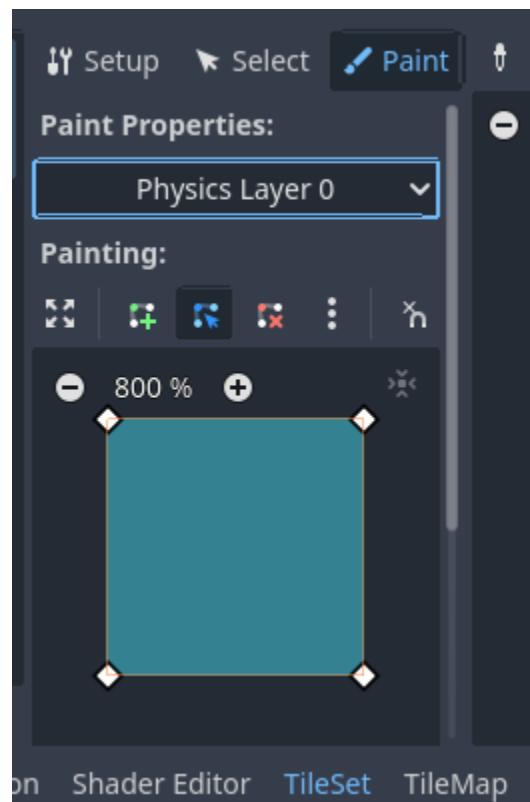
- Chọn Physics Layers (Lớp vật lý) và nhấn Add Element để thêm lớp vật lý cho màn chơi
- Màn chơi sẽ ở lớp 1 (Collision Layer) và tương tác với các thành phần khác ở các lớp khác trong game (Vd: nhân vật ở lớp 2, vật phẩm thu thập ở lớp 3, v.v...)



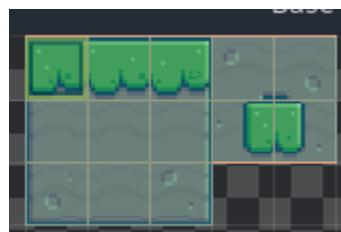
- Nhập vào TileSet, chọn cọ vẽ Paint (Cọ vẽ)



- Trong Paint Properties chọn lớp vật lý có tên Physic Layer 0 (lớp vật lý vừa được tạo ở trên)



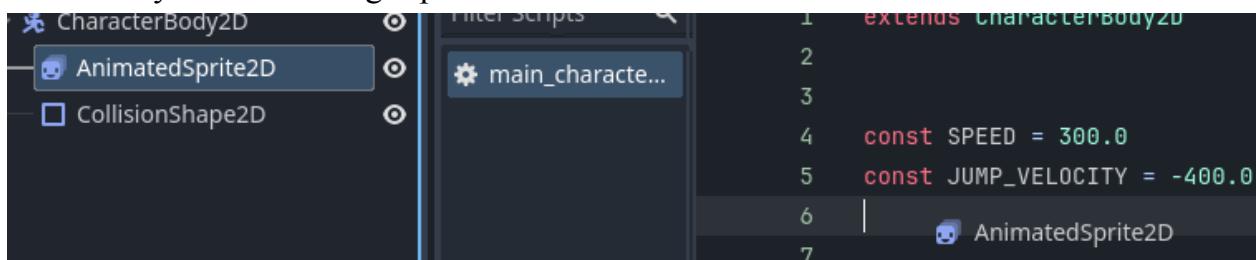
- Áp dụng lớp vật lý cho các khối được dùng trong game với chuột trái



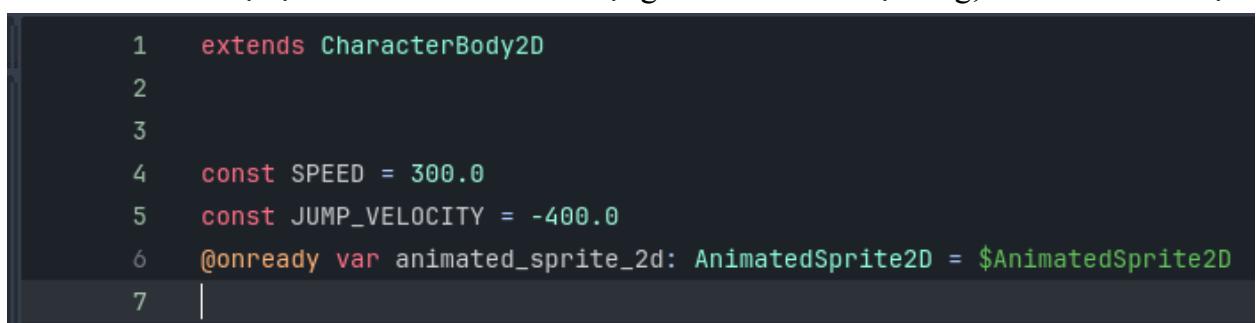
- Lúc này khi chạy game thì ta thấy nhân vật đã tiếp đất, giữa màn chơi và nhân vật có va chạm vật lý với nhau

4.2.7. Cài tiến hiệu ứng của nhân vật

- Ta cần giải quyết 3 vấn đề nếu muốn cài tiến hiệu ứng
 1. Làm sao xoay nhân vật dựa theo hướng di chuyển?
 2. Làm sao chuyển đổi giữa các hiệu ứng?
 3. Làm sao chỉnh sửa logic của nhảy/chạy tùy ý?
- Để giải quyết câu hỏi đầu tiên, ta cần kéo AnimatedSprite2D vào trong Script và lưu ý trước khi thả giữ phím Ctrl

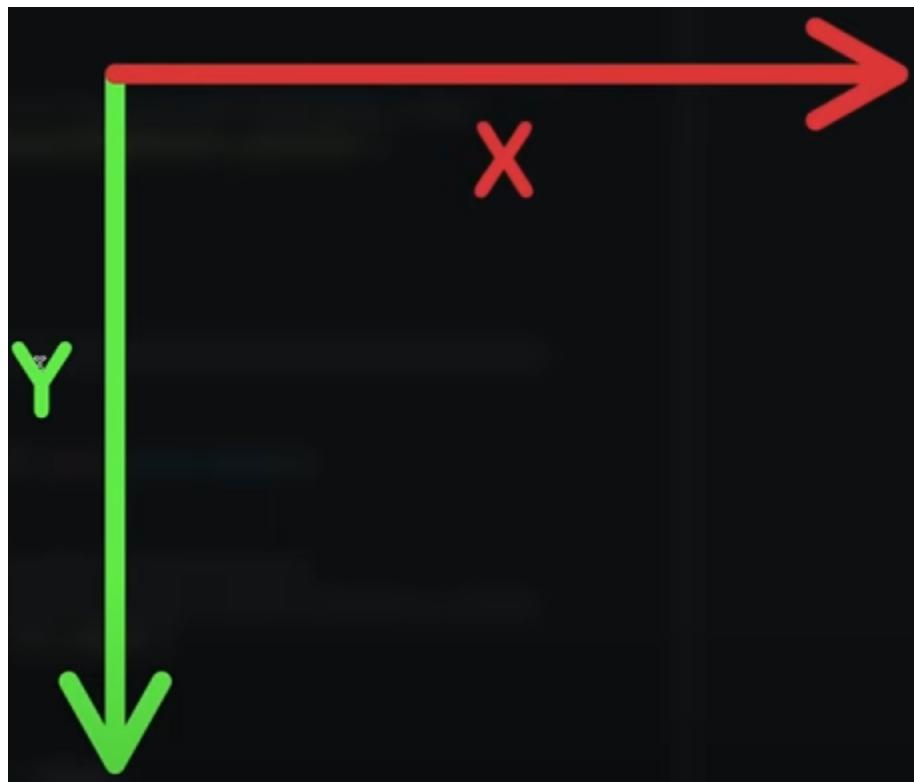


- Godot sẽ tự tạo cho ta 1 biến để sử dụng để chỉnh sửa hiệu ứng, hình ảnh nhân vật

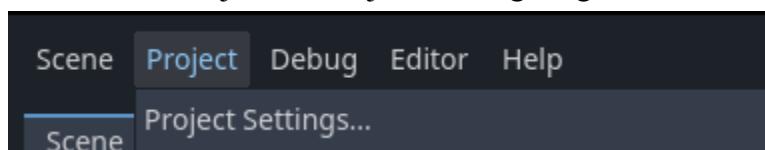


* Lưu ý

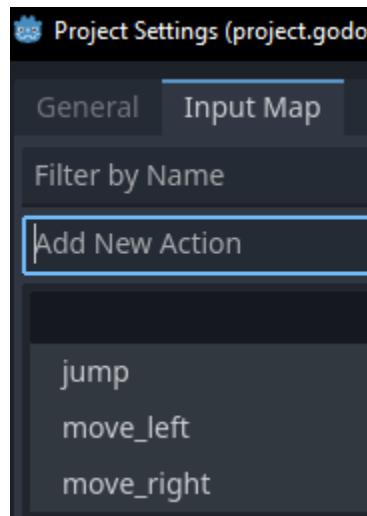
- Trong Godot thì trục tung (X) và trục hoành (Y) sẽ có hướng như trong hình và tăng dần theo hướng mũi tên



- Tham khảo thêm syntax của GDScript tại trang
https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html
- Trước tiên, ta cần vào Project -> Project Settings ở góc trái trên màn hình



- Vào mục Input Map và thêm vào 3 biến hành động: jump (nhảy), move_left (sang trái) và move_right (sang phải)



- Nhấn dấu **+** cạnh các hành động vừa được thêm và gán nút vào các biến đó



- Nhập lệnh có logic như sau để xoay nhân vật dựa theo hướng di chuyển

```

19  >|  var direction := Input.get_axis("move_left", "move_right")
20  >|  if direction:
21  >|  >|  velocity.x = direction * SPEED
22  >|  else:
23  >|  >|  velocity.x = move_toward(velocity.x, 0, SPEED)
24  >|
25  >|  # Direction
26  >|  if direction > 0:
27  >|  >|  animated_sprite_2d.flip_h = false
28  >|  elif direction < 0:
29  >|  >|  animated_sprite_2d.flip_h = true
30  >|  move_and_slide()

```

- Chạy game và kiểm tra kết quả, nhân vật sẽ xoay theo hướng chỉ định
- Đến vấn đề thứ 2, ta cần sửa code với logic như sau

```

8 > func _physics_process(delta: float) -> void:
9  >|  #Animations
10 >|  if (velocity.x > 1 || velocity.x < -1 ):
11  >|  >|  animated_sprite_2d.animation = "running"
12 >|  else:
13  >|  >|  animated_sprite_2d.animation = "idle"
14  >|
15  >|  # Add the gravity.
16 >|  if not is_on_floor():
17  >|  >|  velocity += get_gravity() * delta
18  >|  >|  animated_sprite_2d.animation = "jumping"
19

```

- Chạy game và kiểm tra kết quả, nhân vật sẽ chạy hiệu ứng theo logic đã cài đặt
- Đối với đề thứ 3, ta cần cài đặt logic như sau

```

4  const SPEED = 300.0
5  const JUMP_VELOCITY = -600.0

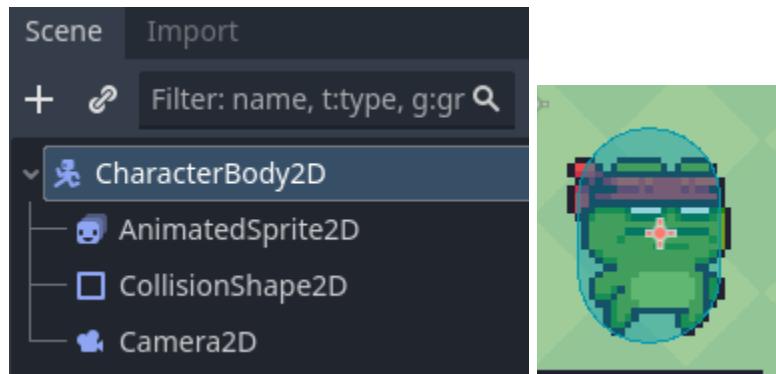
29 >|  else:
30  >|  >|  velocity.x = move_toward(velocity.x, 0, 12)

```

- Kết quả khi chạy sẽ thấy nhân vật có thể nhảy cao hơn và có hiệu ứng trượt nhẹ khi từ từ ngừng di chuyển về phía trước

4.2.8. Camera động

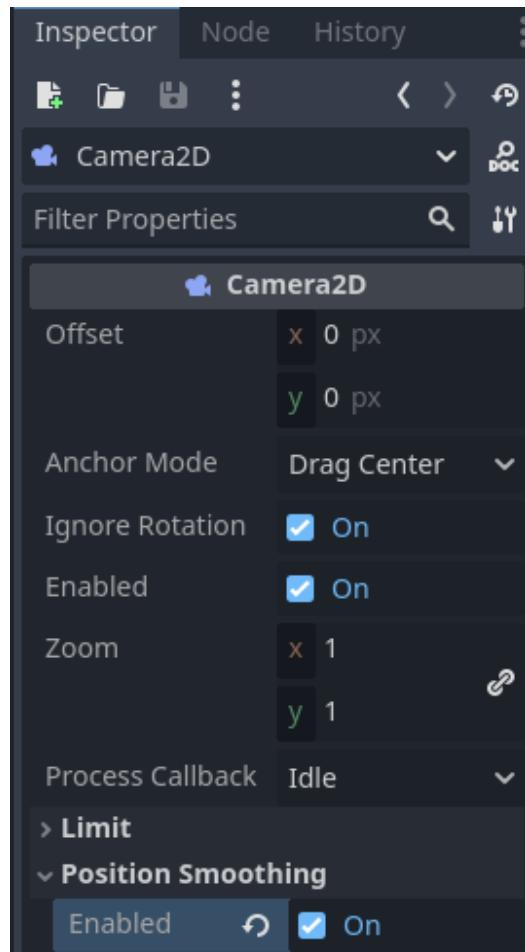
- Thêm node Camera2D là con của CharacterBody2D và đặt camera vào giữa nhân vật



- Chạy game để kiểm tra camera di chuyển theo nhân vật

4.2.9. Chuyển động mượt cho camera động

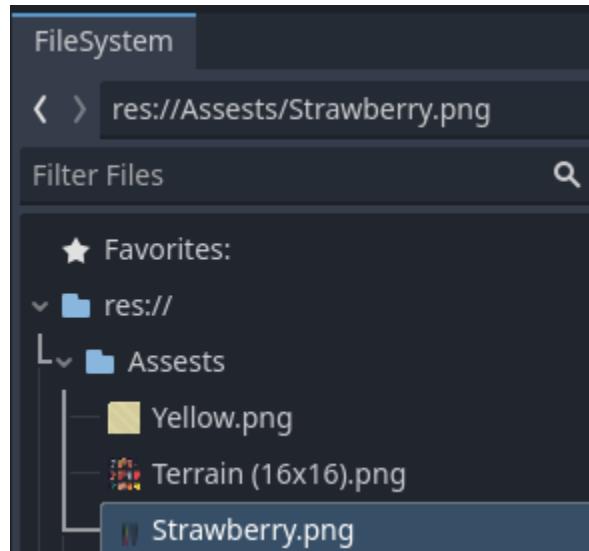
- Tại menu Inspector chọn Position Smoothing -> Enabled: On



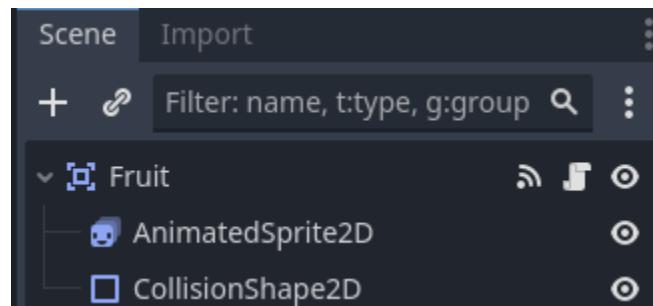
- Chạy game để kiểm tra camera di chuyển theo nhân vật một cách mượt mà

4.2.10. Thêm vật phẩm thu thập

- Kéo file hiệu ứng vật phẩm thu thập vào FileSystem



- **Area2D** là một node đặc biệt trong Godot dùng để phát hiện va chạm không vật lý trong không gian 2D. Nó không bị ảnh hưởng bởi trọng lực hoặc lực tác động, mà chỉ dùng để kiểm tra xem có thứ gì đi vào hoặc đi ra khỏi nó.
- Tạo node mới tên Area2D sửa tên thành Fruit (tên vật phẩm thu thập) và Node con AnimatedSprite2D, CollisionShape2D và thực hiện gán hiệu ứng, hình ảnh, vật lý tương tự như thiết kế nhân vật

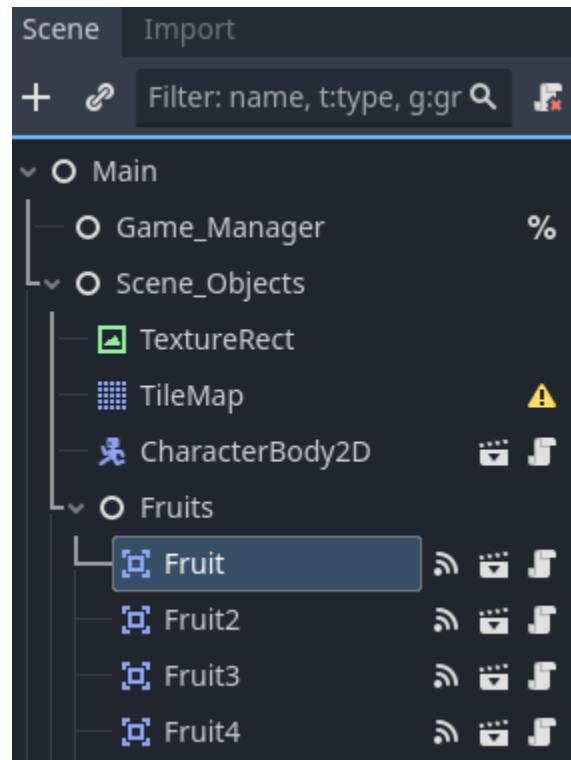


- Lưu vật phẩm thành 1 Scene riêng để dễ chỉnh sửa và sử dụng, tránh ảnh hưởng Scene tổng (tương tự nhân vật)

- Sau khi tạo xong ta có thể tái sử dụng vật phẩm thu thập, kéo thả Scene Fruit vào màn chơi số lượng vật phẩm tùy thích từ File System

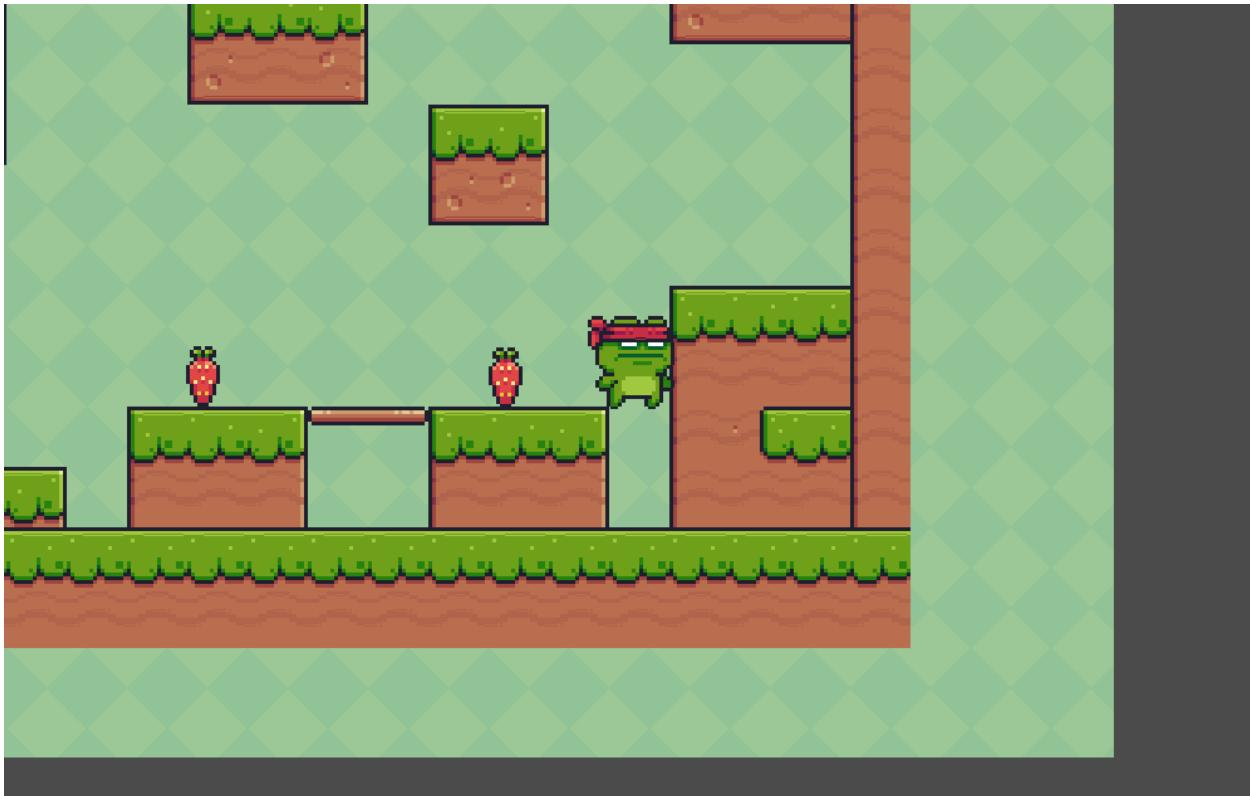


- Có thể gom các vật phẩm thành 1 Node rỗng để cấu trúc cây nhìn gọn gàng hơn

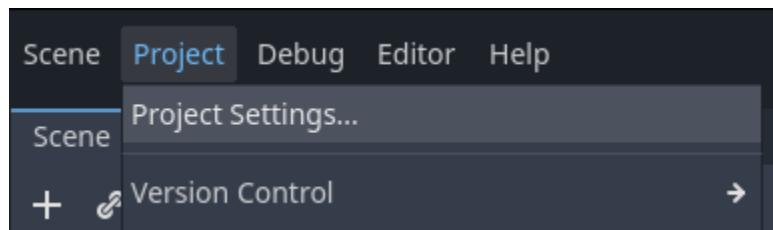


4.2.11. Sửa lỗi liên quan đến kích thước cửa sổ

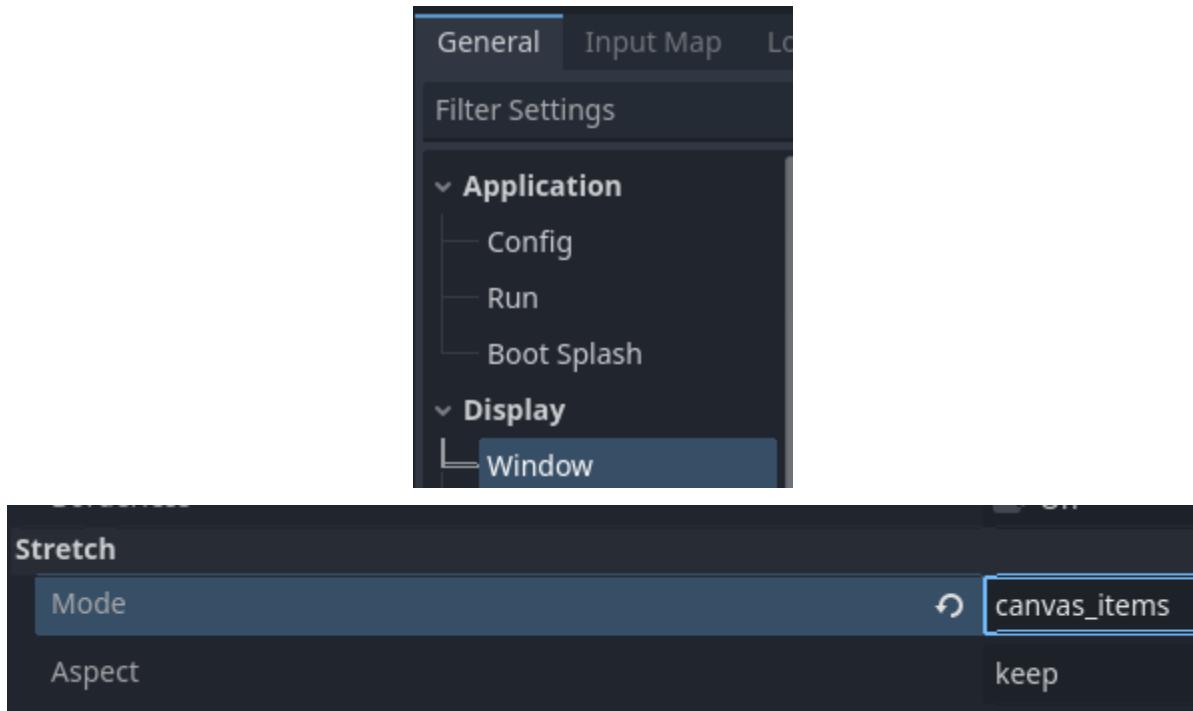
- Trước khi cải thiện logic của vật phẩm, ta cần xử lý vấn đề khi thay đổi kích thước cửa sổ background sẽ bị tràn ra ngoài



- Vào Project -> Project Settings

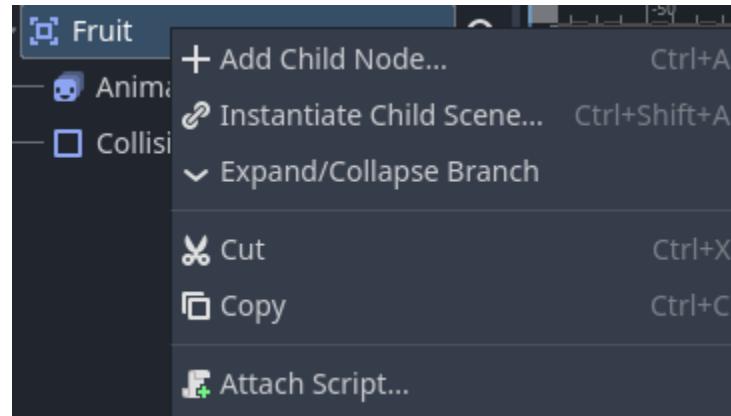


- Chọn General -> Display -> Window
- Chọn Stretch -> Mode -> canvas_items

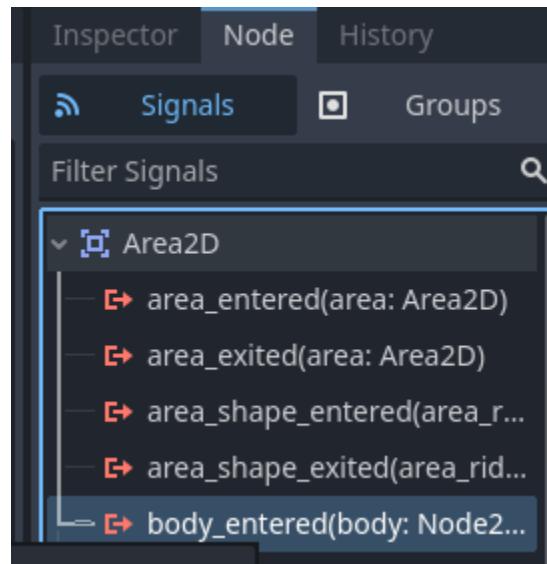


4.2.12. Cài tiến vật phẩm thu thập

- Thêm Script cho vật phẩm bằng cách chuột phải -> chọn Attach Script



- Tại menu Node (góc phải trên) chọn tín hiệu body_entered



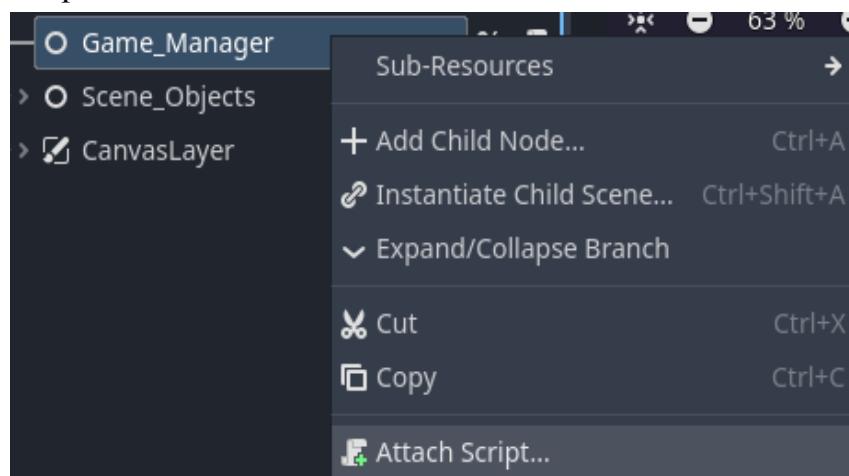
- Chính sửa logic code như sau

```
→ 14 func _on_body_entered(body: Node2D) -> void:
15   if(body.name == "CharacterBody2D"):
16     queue_free()
```

- Chạy game và kiểm tra vật phẩm biến mất khi chạm vào

4.2.13. Bộ đếm điểm

- Tạo Node rỗng đặt tên Game_Manager và đính kèm Script với chuột phải -> chọn Attach Script



```

File Edit Search Go To Debug
Filter Scripts ⚙️
fruit.gd
game_manager.gd
main_character.gd
1   extends Node
2
3
4   # Called when the node has been added to a scene.
5   func _ready():
6       pass # Replace with your code here.
7
8
9   # Called every frame. You can set several scripts to run
10  func _process(delta):
11      pass

```

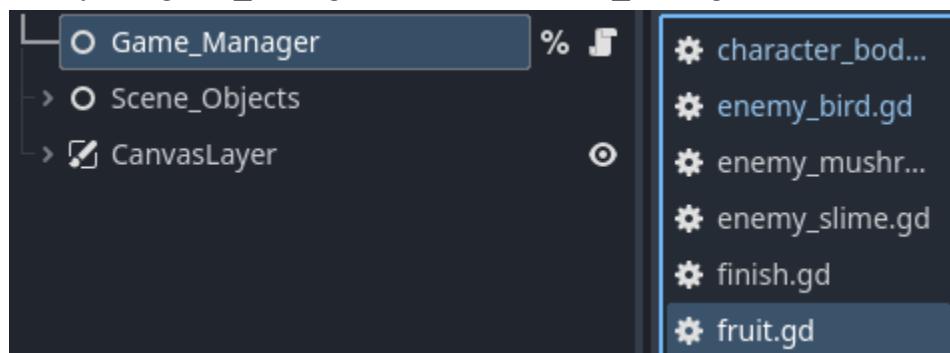
- Chính sửa logic code như sau

```

1   extends Node
2
3   var points = 0
4
5   func add_point():
6       points += 1
7       print(points)
8

```

- Để gọi được Script này trong Script của vật phẩm ta chuột phải vào node Game_Manager và c
- chọn **% Access as Unique Name**
- Kéo node Game_Manager vào Script vật phẩm và giữ Ctrl khi thả để tạo ra biến @onready var game_manager: Node = %Game_Manager



- Chính sửa logic lại như sau

```

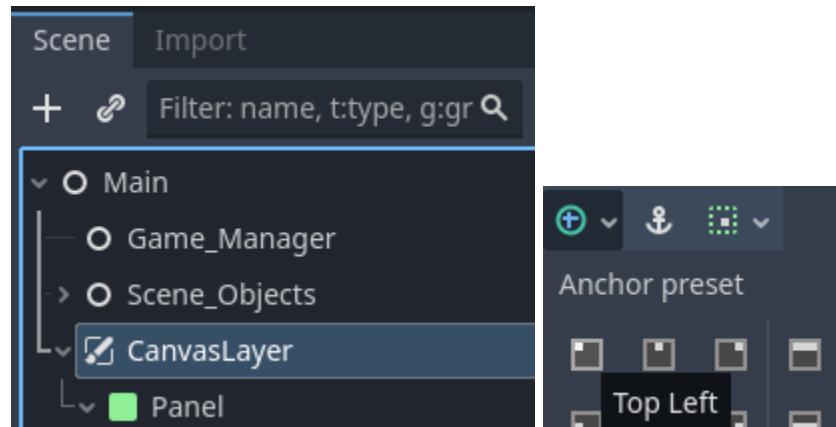
15  func _on_body_entered(body: Node2D) -> void:
16    if(body.name == "CharacterBody2D"):
17      queue_free()
18      game_manager.add_point()
19

```

- Khi chạy game ta sẽ thấy mỗi khi nhặt vật phẩm thì biến đếm sẽ được xuất tại màn hình debug bên dưới

4.2.14. Hiển thị bảng điểm

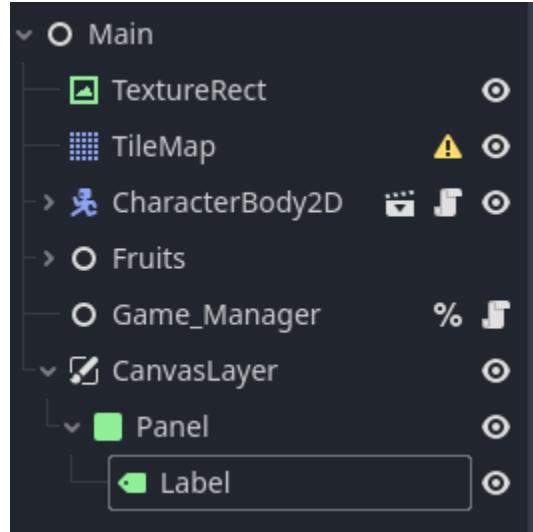
- **CanvasLayer** là một node đặc biệt trong Godot Engine, dùng để vẽ UI hoặc các đối tượng 2D mà không bị ảnh hưởng bởi camera hoặc scene chính
- **Panel** là một node giao diện trong Godot, thường được sử dụng để tạo nền hoặc khung cho UI. Nó có thể chứa các thành phần khác như, văn bản (Label), thanh máu, v.v..
- Thêm Node con Panel -> chọn Anchor -> Top Left



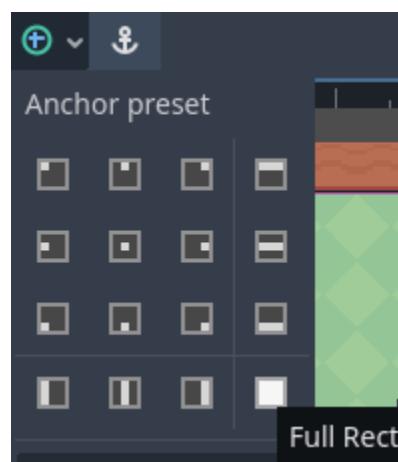
- Chính sửa vị trí bảng điểm



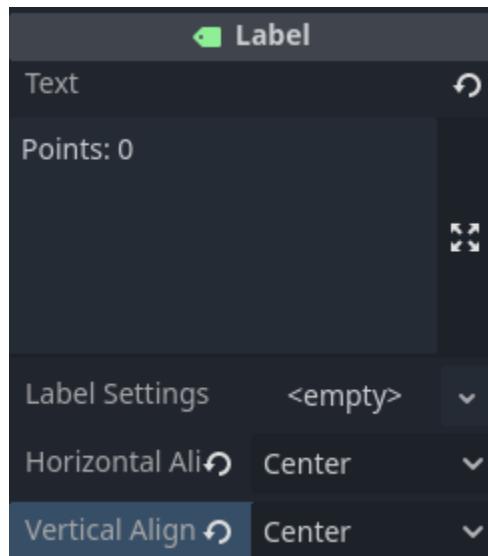
- **Label** là một node UI (Control) trong Godot dùng để hiển thị văn bản tĩnh trên màn hình. Nó không thể chỉnh sửa
- Thêm Node con Label vào Panel



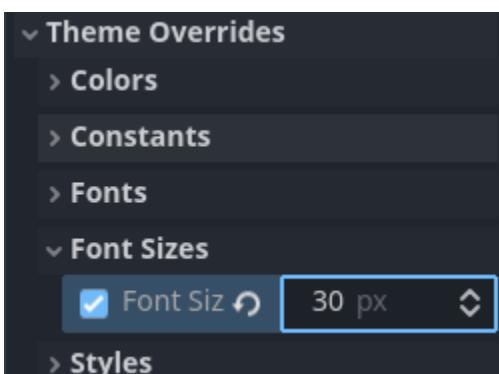
- Chọn biểu tượng -> Full Rect



- Nhập text ở menu Inspector, và chọn Center cho cả Horizontal và Vertical Alignment



- Chọn Theme Overrides -> Font Sizes = 30



- Chuột phải Label -> Access as Unique Name
- Kéo Label vào Script của node Game_Manger và giữ phím Ctrl trước khi thả

```

TreeMap          fruit.gd
CharacterBody2D  game_manager...
Fruits
Game_Manager    %
CanvasLayer
Panel
  Points_Label %

```

```

3 var points = 0
4 @onready var points_label: Label = %Points_Label
5
6 func add_point():
7   points += 1
8   print(points)
9
10 # Called when the node enters the scene tree for t
11 func ready() -> void:

```

- Sửa logic Script của Game_Manager

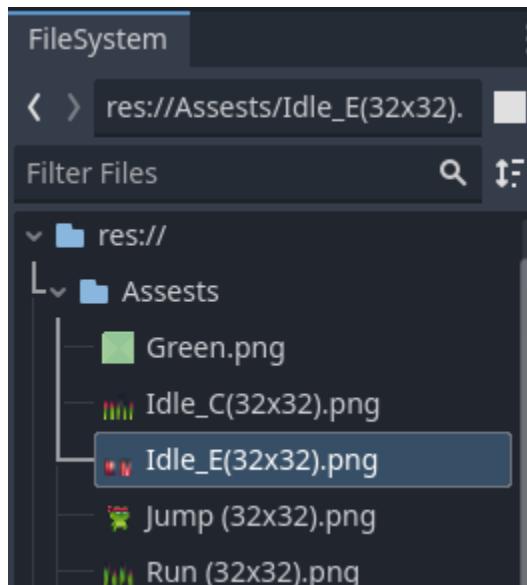
```

3  var points = 0
4  @onready var points_label: Label = %Points_Label
5
6  func add_point():
7      points += 1
8      print(points)
9      points_label.text = "Points: " + str(points)

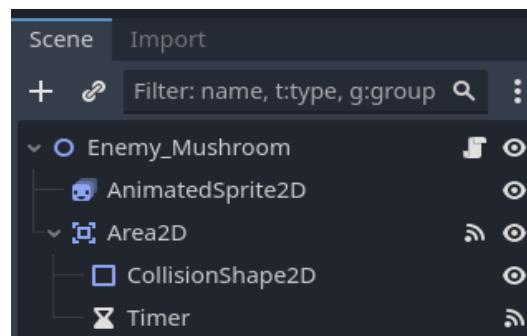
```

4.2.15. Thêm kẻ địch vào game

- Sử dụng assets miễn phí <https://pixelfrog-assets.itch.io/pixel-adventure-2>, giải nén và thêm hiệu ứng hình ảnh kẻ địch vào File System



- **Node2D** là một node cơ bản trong Godot, dùng để quản lý các đối tượng trong không gian 2D
- Thêm node Node2D và Node con AnimatedSprite2D + Area2D + CollisionShape2D + Timer

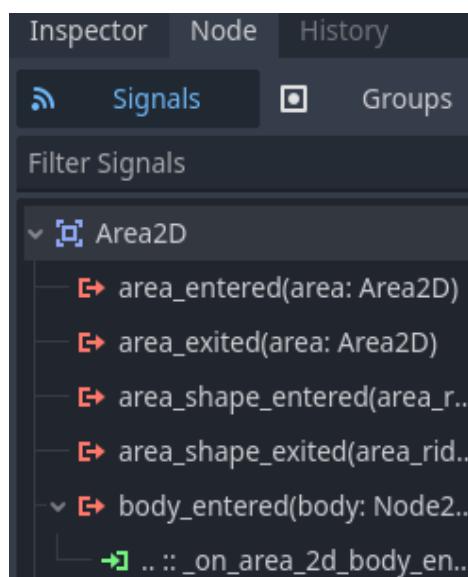


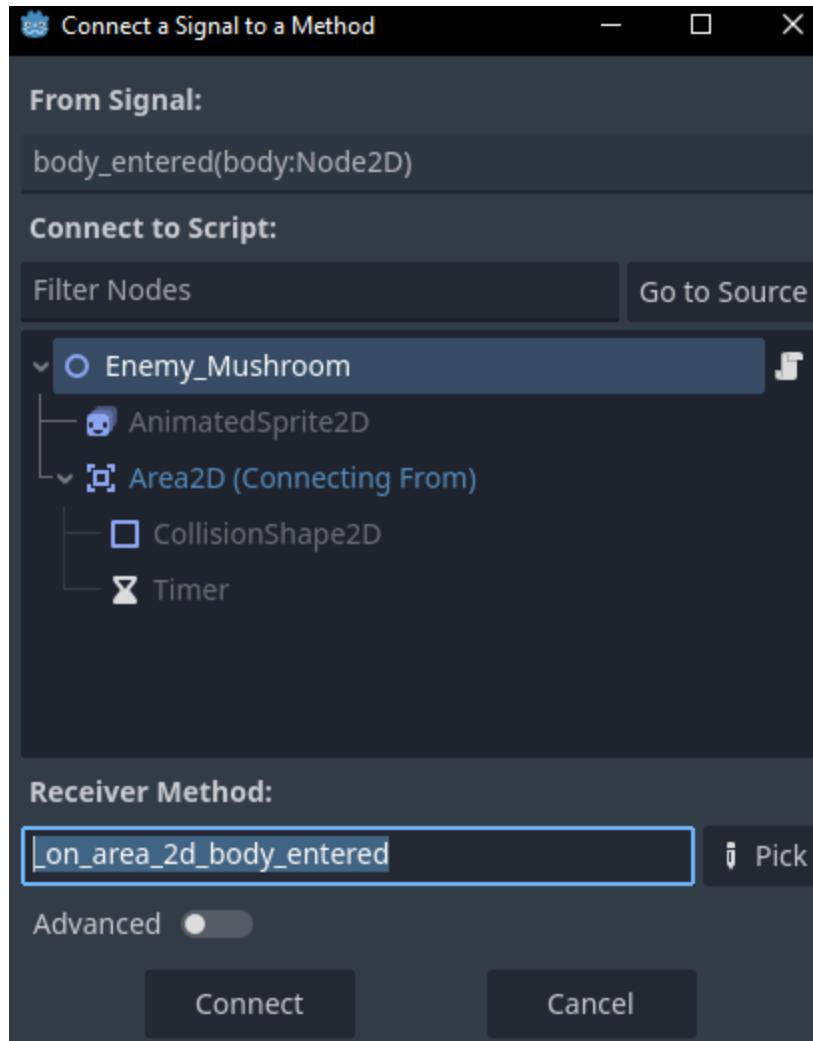
- Thiết kế kẻ địch tương tự như thiết kế nhân vật, các bước tạo lớp vật lý, thêm hiệu ứng nhân vật đều giống với thiết kế nhân vật
- Sau khi thiết kế xong chỉ cần lưu kẻ địch thành 1 Scene và kéo scene kẻ địch vào trong màn chơi



4.2.17. Cài tiền kẻ địch và hiệu ứng người chơi trong game

- Thêm Script cho kẻ địch
- Trong node Area2D và chọn tín hiệu body_entered ở menu Node và kết nối tới Script kẻ địch





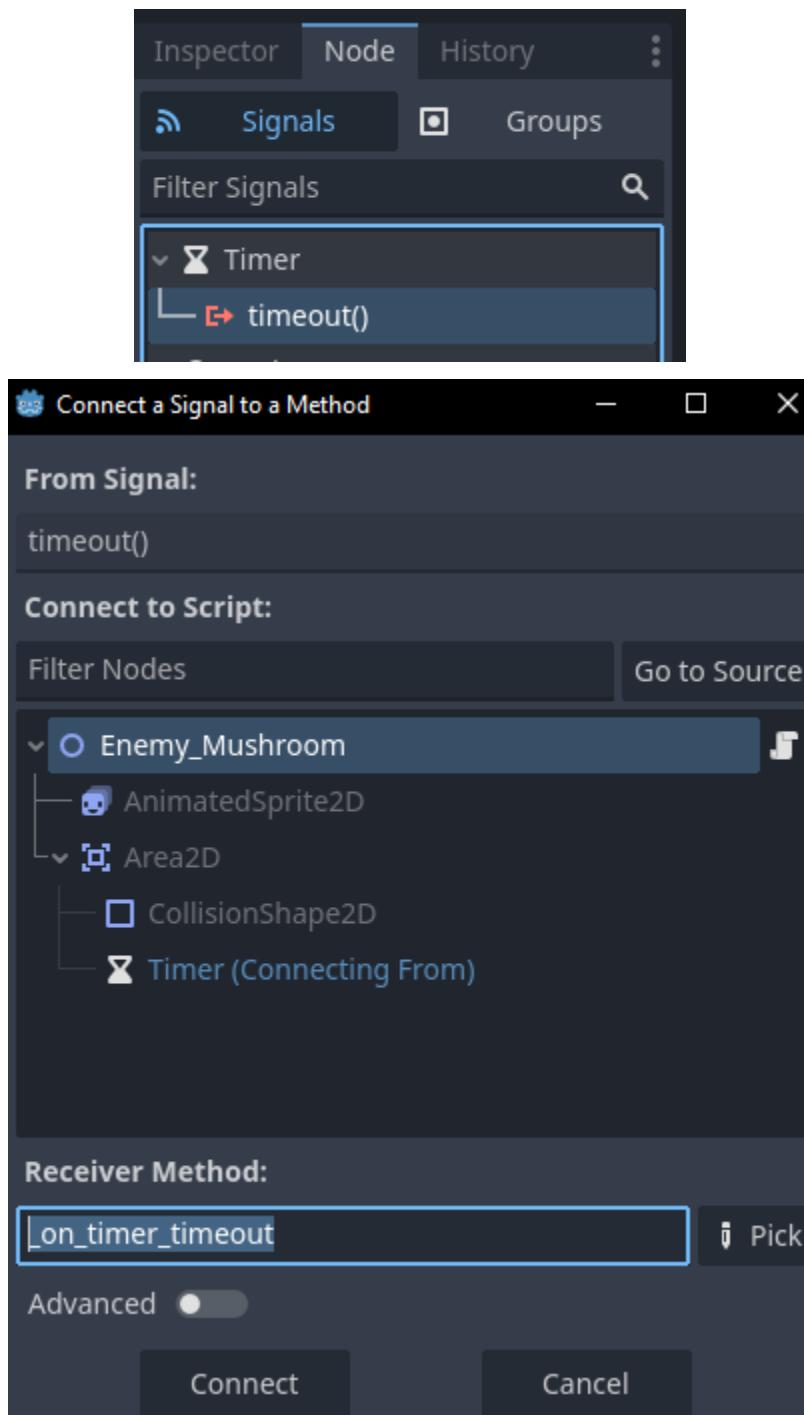
- Thêm logic code cho nhân vật (người chơi) như sau

```
8 func jump():
9     > velocity.y = JUMP_VELOCITY
10
```

- Kéo Node Timer từ menu Scene vào script của kẻ địch và giữ Ctrl trước khi thả

```
2
3 @onready var timer: Timer = $Area2D/Timer
4
```

- Chọn tín hiệu Timeout menu bên phải kết nối tới script của kẻ địch



- Chính sửa logic code của kẻ địch như sau

```

→ 14  func _on_area_2d_body_entered(body: Node2D) -> void:
15    if(body.name == "CharacterBody2D"):
16      #Enemy y position - Player y position
17      var y_delta = position.y - body.position.y
18      if(y_delta > 30):
19        print("Destroy enemy")
20        queue_free()
21        body.jump()
22      else:
23        print("You die")
24        Engine.time_scale = 0.5
25        body.get_node("CollisionShape2D").queue_free()
26        timer.start()

→ 28  func _on_timer_timeout() -> void:
29    Engine.time_scale = 1.0
30    get_tree().reload_current_scene()
31

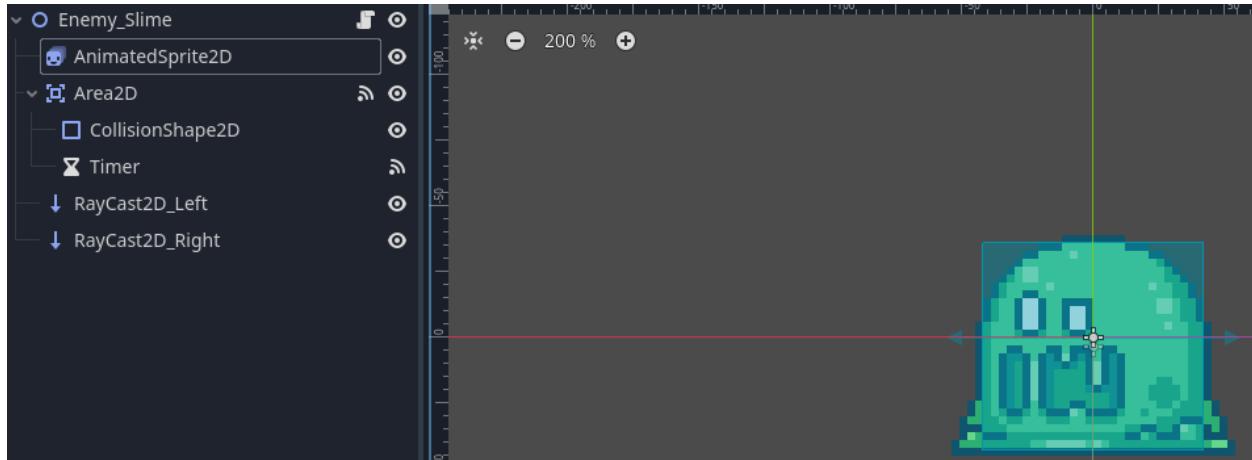
```

- Sau khi chạy game ta nhận thấy khi nhảy lên đầu kẻ địch thì kẻ địch biến mất và nhân vật ta tự bật nhảy, nếu chạm bên cạnh kẻ địch thì nhân vật rơi xuống màn chơi và game tự bắt đầu lại

4.2.16. Thêm kẻ địch biết di chuyển vào game

- Thực hiện lại các bước tạo ra 1 kẻ địch như trên, đính kèm Script + tín hiệu và logic Script giống y như trên
- **RayCast2D** là một node trong Godot dùng để bắn một tia vô hình từ một điểm gốc theo một hướng nhất định. Nó giúp phát hiện vật thể va chạm trên đường đi của tia

- Thêm RayCast2D làm node con của kẻ địch (nhớ thay đổi tên thành RayCast2D_Left và RayCast2D_Right để dễ phân biệt)



- Kéo và giữ Ctrl trước khi thả 2 Node RayCast2D vào trong file script cùng Node AnimatedSprite2D

```

4  @onready var ray_cast_2d_left: RayCast2D = $RayCast2D_Left
5  @onready var ray_cast_2d_right: RayCast2D = $RayCast2D_Right
6  @onready var animated_sprite_2d: AnimatedSprite2D
7

```

- Chỉnh sửa logic code như sau để kẻ địch có thể di chuyển và quay mặt sau khi va chạm tường

```

8  const SPEED = 300.0
9  var direction = 1

```

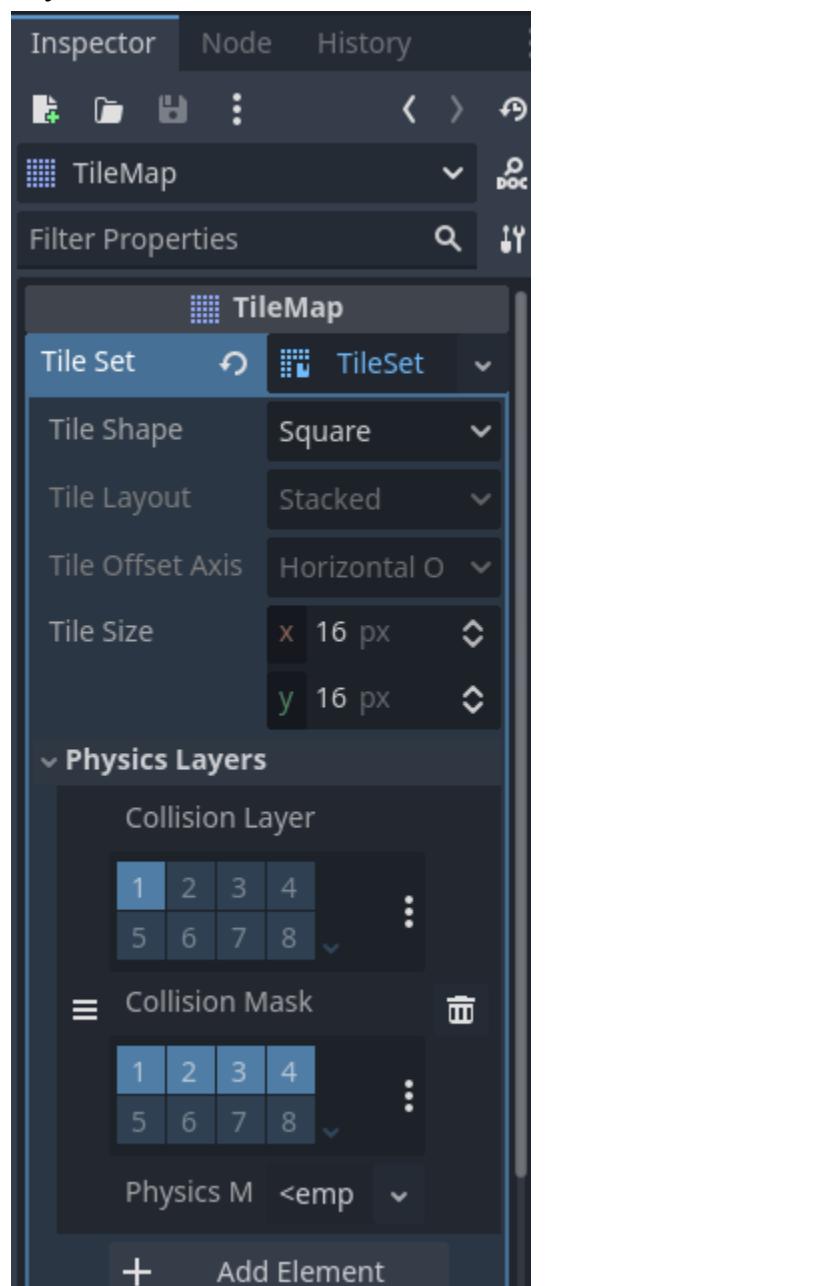
```

16  func _process(delta: float) -> void:
17    if ray_cast_2d_right.is_colliding():
18      direction = -1
19      animated_sprite_2d.flip_h = false
20    if ray_cast_2d_left.is_colliding():
21      direction = 1
22      animated_sprite_2d.flip_h = true
23    position.x += direction * SPEED * delta

```

- Lúc này nếu chạy game ta sẽ thấy lỗi kẻ địch không kết liễu người chơi mà quay mặt lại khi va chạm với người chơi, điều này là do khi mới khởi tạo mặc định các lớp vật lý của bản đồ, nhân vật, kẻ địch đều là lớp đầu tiên (Layer 1) do đó ta cần chỉnh sửa các lớp vật lý sao cho hợp lý

- Trước tiên là lớp vật lý của bản đồ



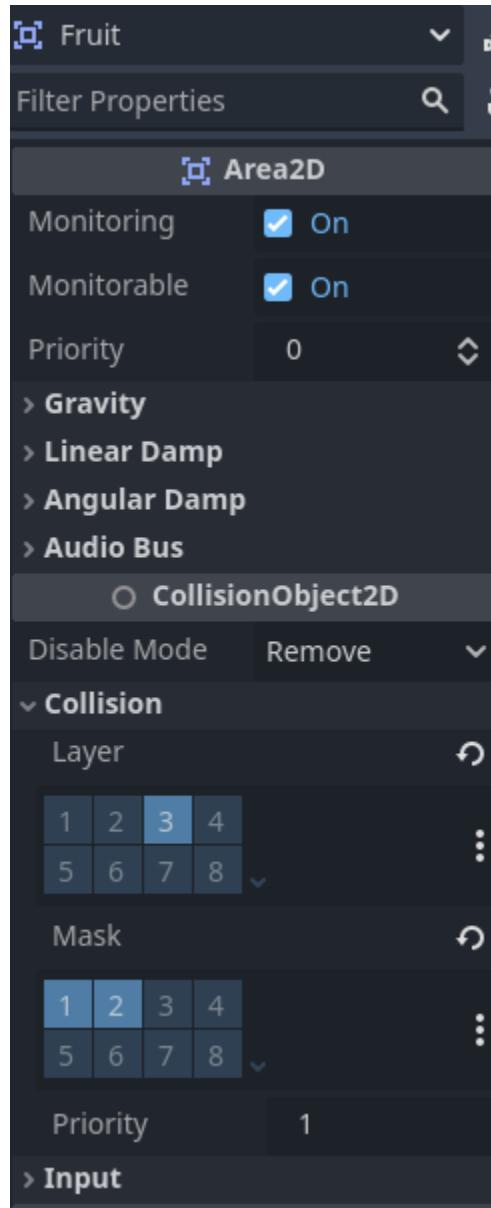
- Bản đồ sẽ nằm ở Layer 1 và có thể tương tác với mọi Layer khác

- Tiếp theo là lớp vật lý người chơi,



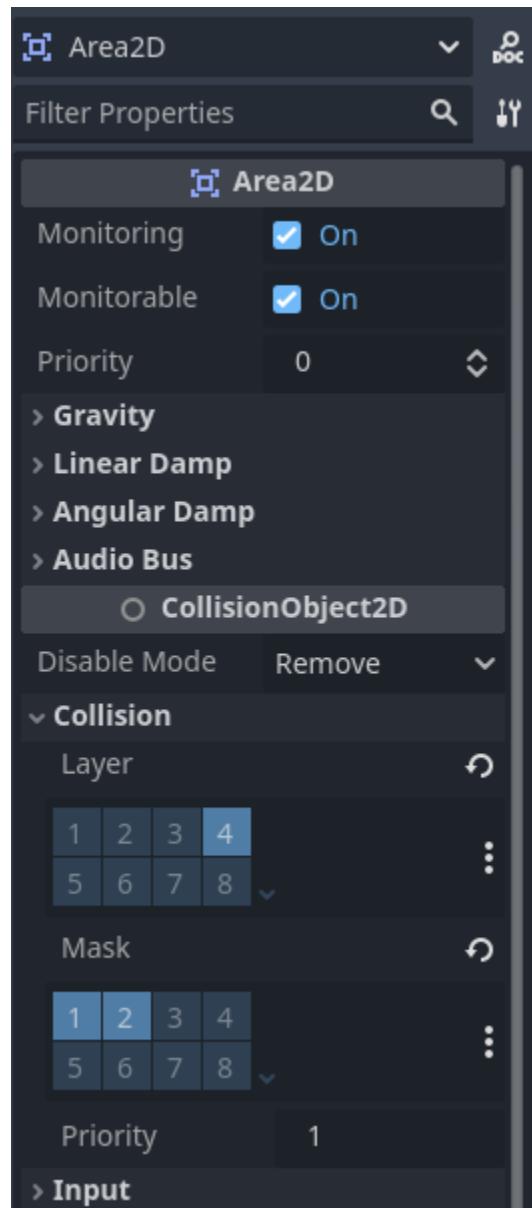
- Người chơi sẽ nằm ở Layer 2 cho phép người chơi va chạm, đứng được trên bัน đồ ở Layer 1

- Đối với vật phẩm thu thập



- Vật phẩm thu thập sẽ nằm ở Layer 3 và cho phép tương tác với Layer 1 (Map) và Layer 2 (Người chơi)

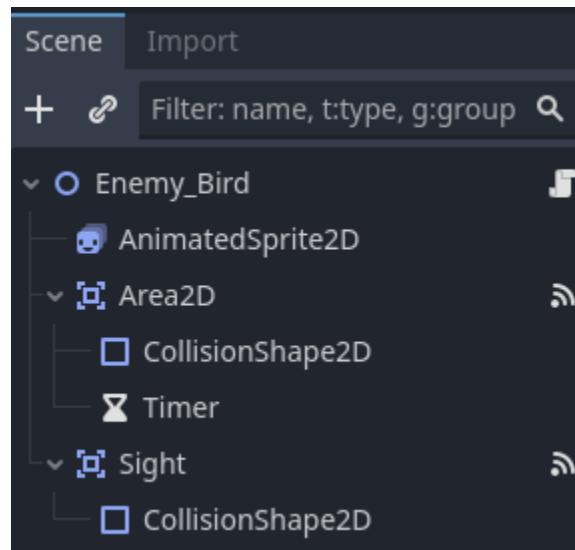
- Vẽ kẻ địch và



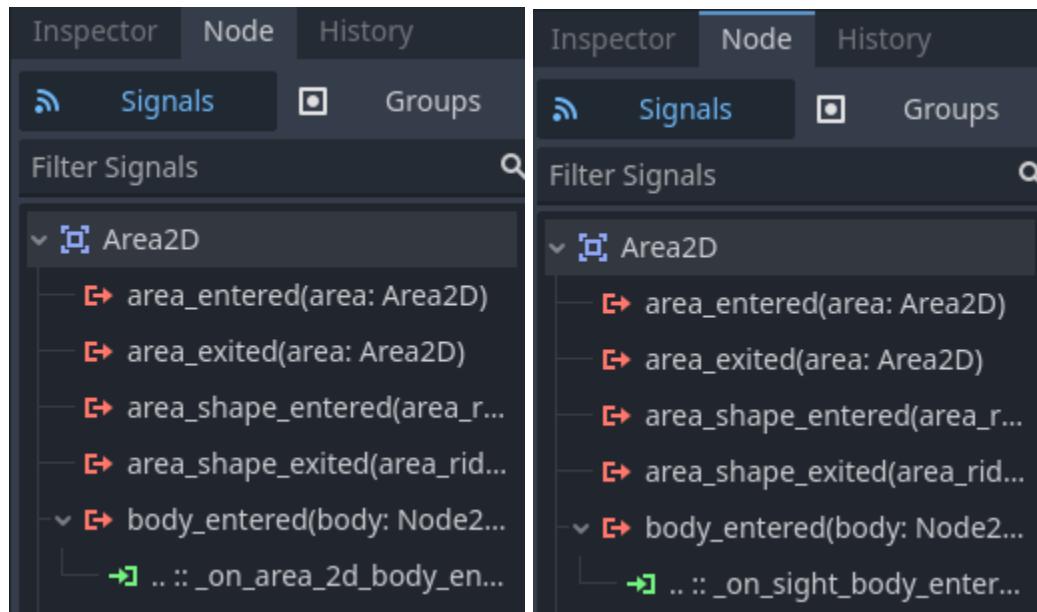
- Kẻ địch sẽ nằm ở Layer 4 và cho phép tương tác với Layer 1 (Map) và Layer 2 (Người chơi)
- Sau khi chạy game thì ta có kết quả như mong muốn, kẻ địch sẽ không quay mặt sau khi chạm vào người chơi

4.2.17. Thêm kẻ địch trên không

- Tạo kẻ địch tương tự mục 4.2.15 (sử dụng hình ảnh khác trong assets) có cấu trúc như sau:



- Trong Node Area2D và Sight ta sẽ vào menu Node chọn tín hiệu body_entered



- Ta chỉnh sửa logic Script như sau:

```

1  extends Node2D
2
3  @onready var timer: Timer = $Area2D/Timer
4  @onready var animated_sprite_2d: AnimatedSprite2D = $AnimatedSprite2D
5
6  const SPEED = 150.0
7  var is_following = false
8  var player: Node2D = null
9
10 # Called every frame. 'delta' is the elapsed time since the previous frame.
11 func _process(delta: float) -> void:
12     if is_following and player != null:
13         var direction_to_player = (player.position - position).normalized()
14         position += direction_to_player * SPEED * delta
15         if direction_to_player.x > 0:
16             animated_sprite_2d.flip_h = true
17         else:
18             animated_sprite_2d.flip_h = false

```

```

→ 20 func _on_sight_body_entered(body: Node2D) -> void:
21     if body.name == "CharacterBody2D":
22         is_following = true
23         player = body
24
→ 25 func _on_area_2d_body_entered(body: Node2D) -> void:
26     if body.name == "CharacterBody2D":
27         var y_delta = position.y - body.position.y
28         if y_delta > 30:
29             print("Destroy enemy")
30             queue_free()
31             body.jump()
32         else:
33             print("You die")
34             Engine.time_scale = 0.5
35             body.get_node("CollisionShape2D").queue_free()
36             timer.start()

```

```

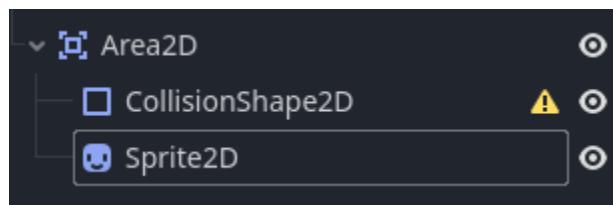
38     func _on_timer_timeout() -> void:
39         Engine.time_scale = 1.0
40         get_tree().reload_current_scene()

```

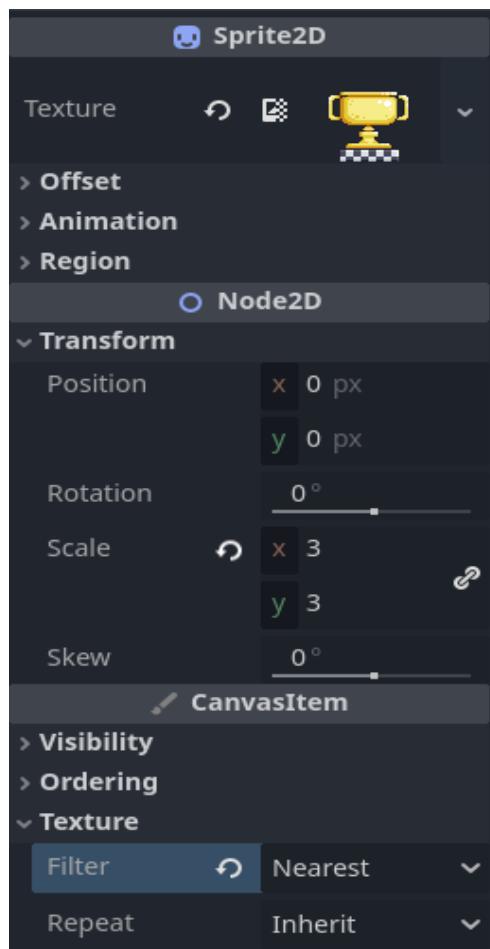
- Ta sẽ có kẻ địch biết bay và đuổi theo người chơi khi trong tầm (Node Sight)

4.2.18. Thêm đích đến vào game

- Để tạo được đích đến, ta có thể làm tương tự như vật phẩm thu thập (4.2.11)
- Tạo node Area2D có node con là CollisionShape2D và Sprite2D



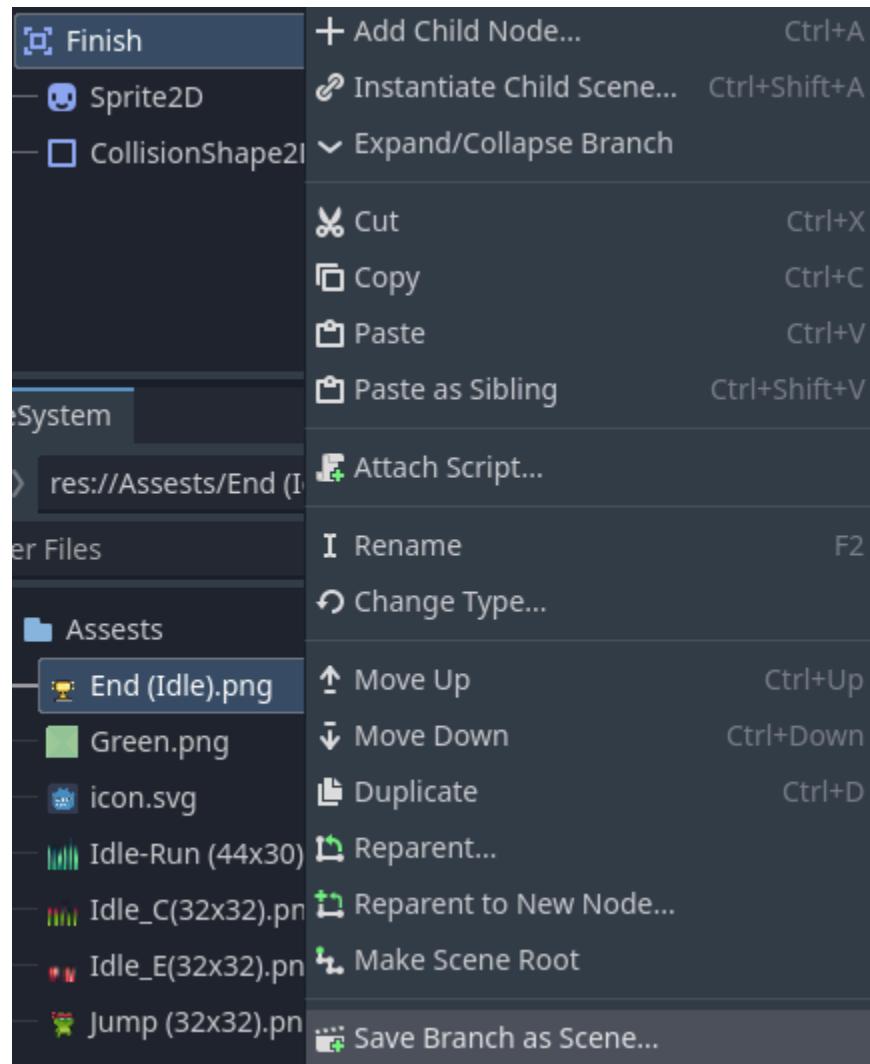
- Kéo hình ảnh đích đến vào trong menu bên phải của Sprite2D và chỉnh sửa thông số như sau



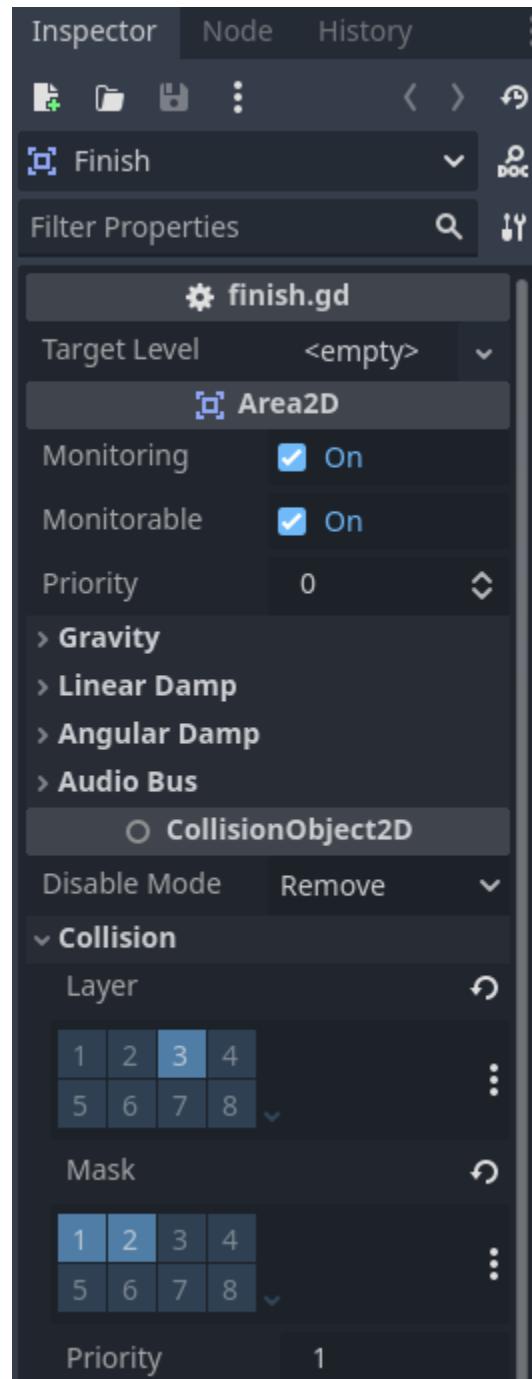
- Đối với CollisionShape2D ta chọn hình vuông và chỉnh sửa để bao quanh hình ảnh



- Tương tự như vật phẩm thu thập, ta cần chọn Save Branch as Scene để dễ sử dụng



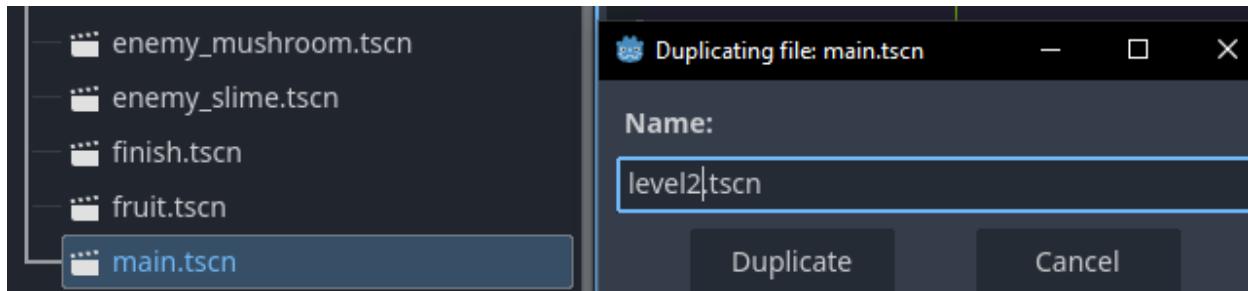
- Chọn lớp vật lý tương tự vật phẩm thu thập



- Đích đến hiện tại chưa có chức năng gì và sẽ được cải tiến ở các bước sau

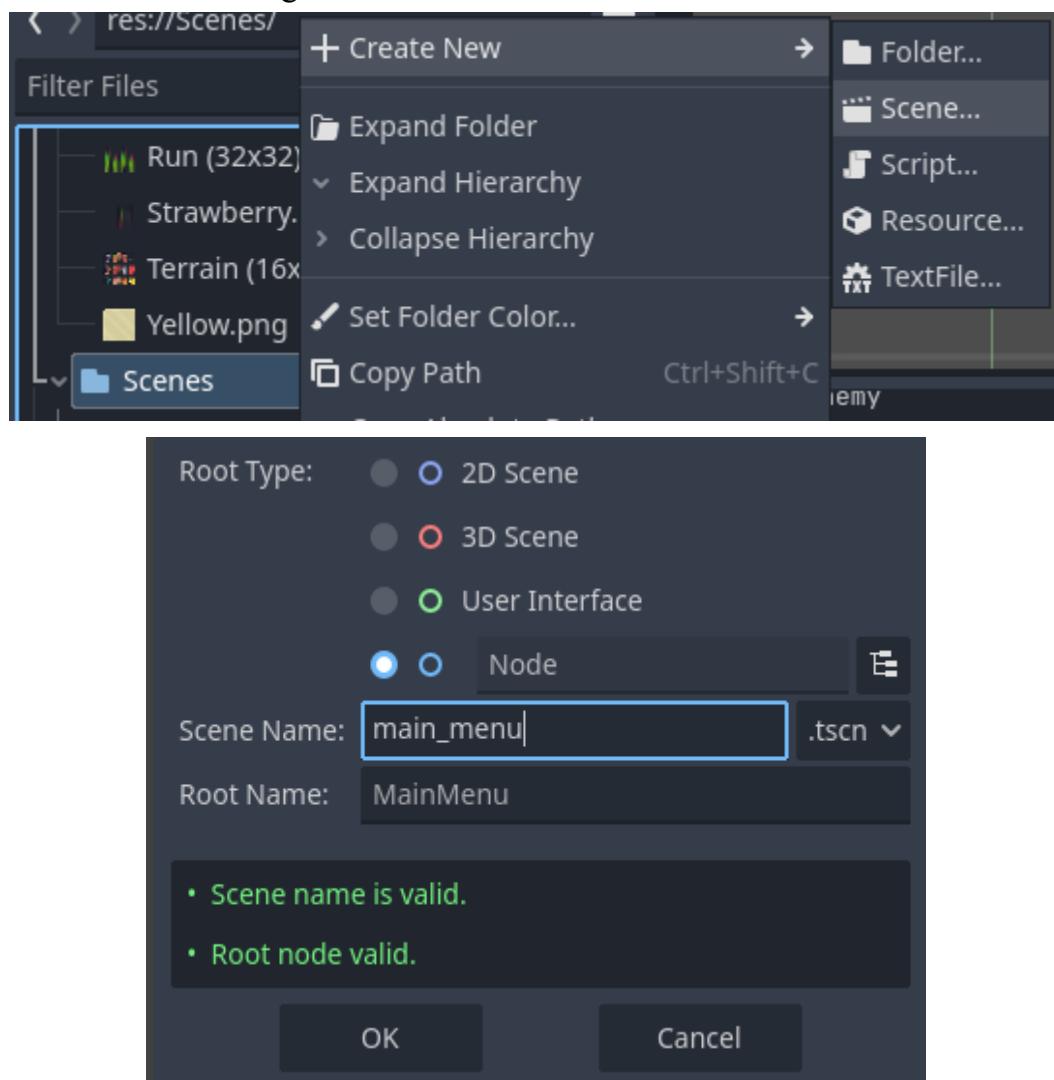
4.2.19. Tạo màn chơi tiếp theo

- Dựa vào tất cả kiến thức từ 4.2.1 đến 4.2.18 ta có thể tạo thêm màn chơi nhưng để tiết kiệm thời gian ta có thể copy scene main (nhấn Ctrl + D) và chỉnh sửa để tạo thành màn chơi mới

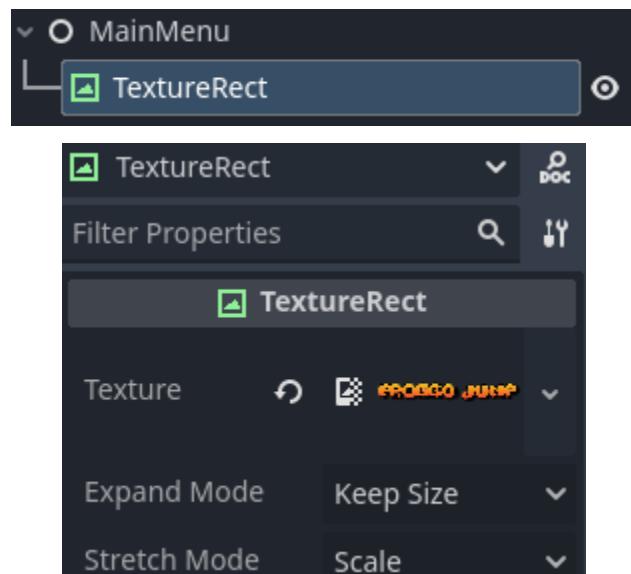


4.2.20. Tạo main menu

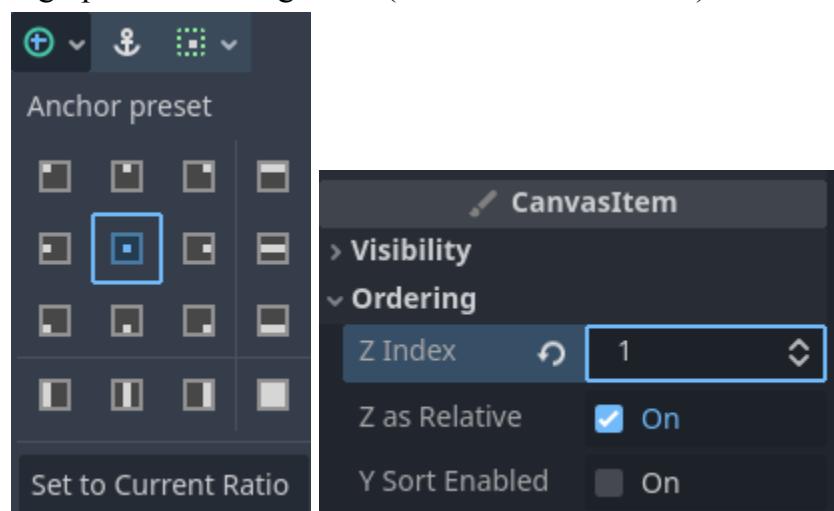
- Tạo Scene mới trong thư mục Scenes



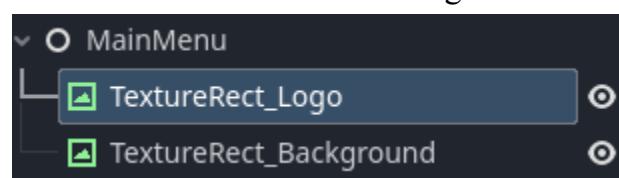
- Tạo node con TextureRect và kéo logo vào menu bên phải



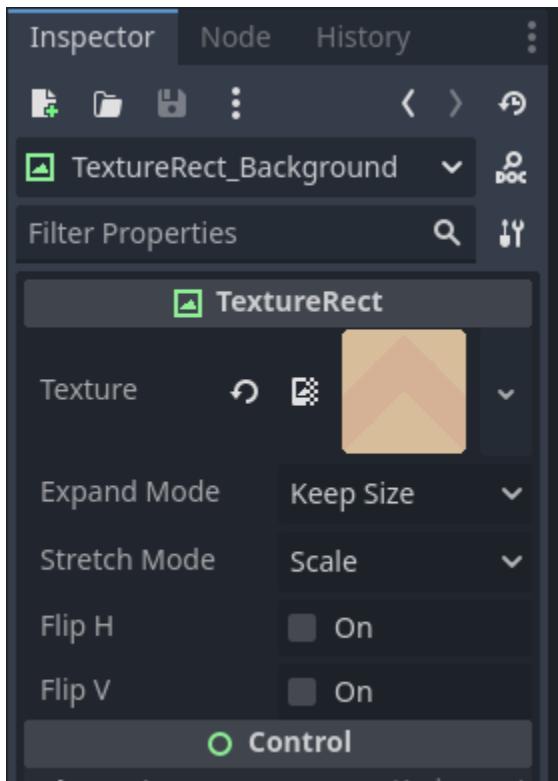
- Tại thanh công cụ giữa chọn Anchor preset -> Center, chọn Z Index = 1 để ưu tiên hiển thị logo phía trên background (mặc định Z Index = 0)



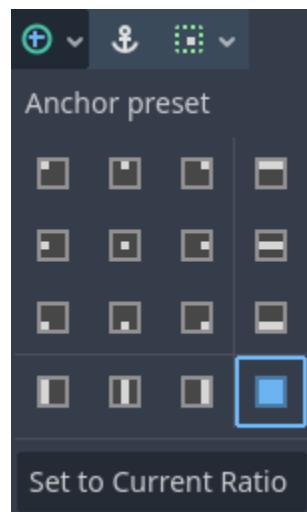
- Tạo thêm 1 node con TextureRect để kèm background sau logo



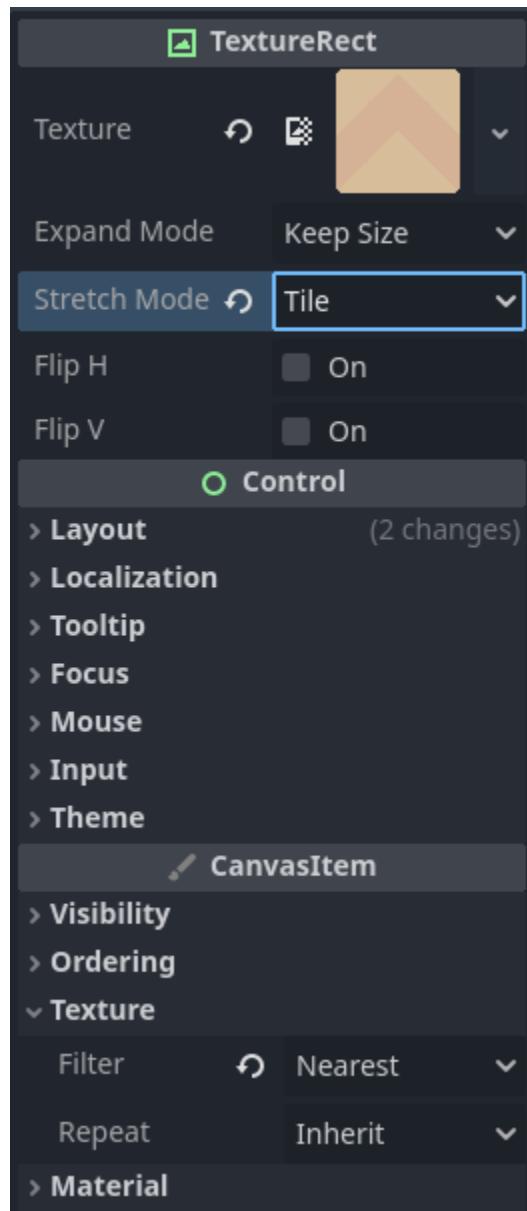
- Kéo ảnh vào menu bên phải của node TextureRect_Background



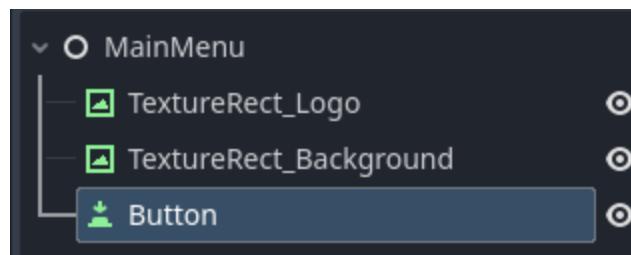
- Chọn Anchor preset -> FullRect



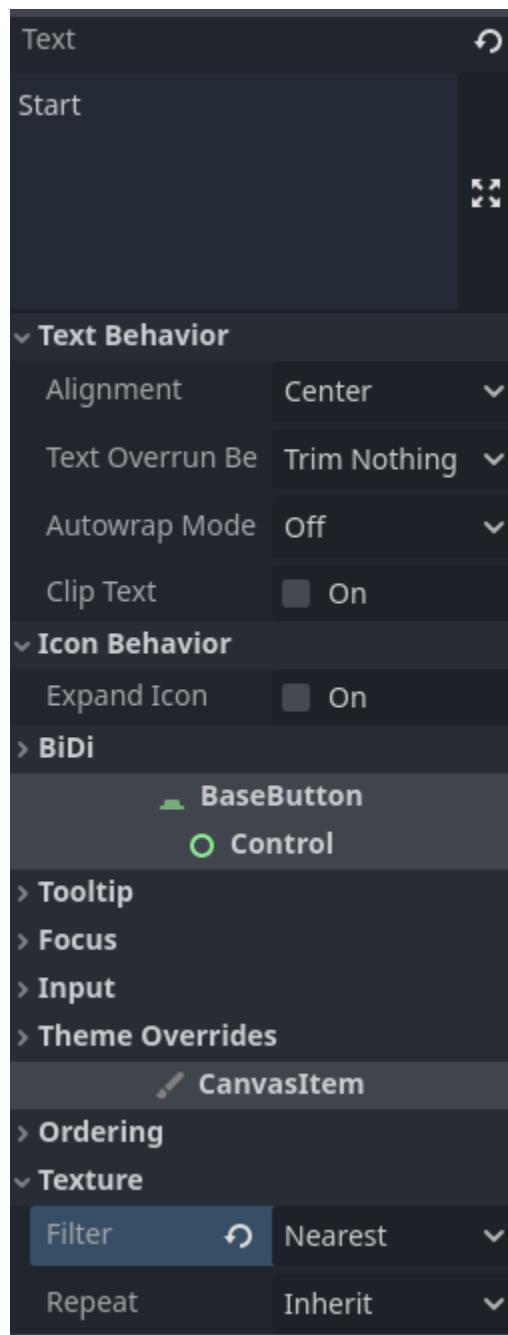
- Trong Stretch Mode -> chọn Tile và Texture -> Filter -> Nearest để tránh hình bị vỡ khi phóng to



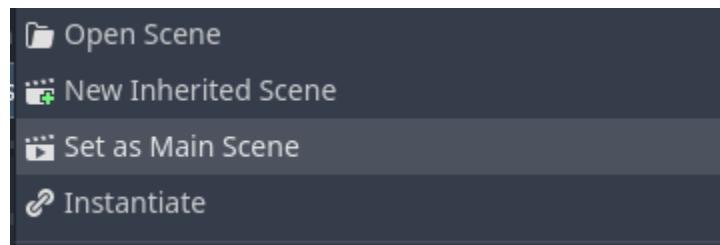
- Tạo Node con Button để có nút bắt đầu game trong menu



- Chính sửa Node Button như sau

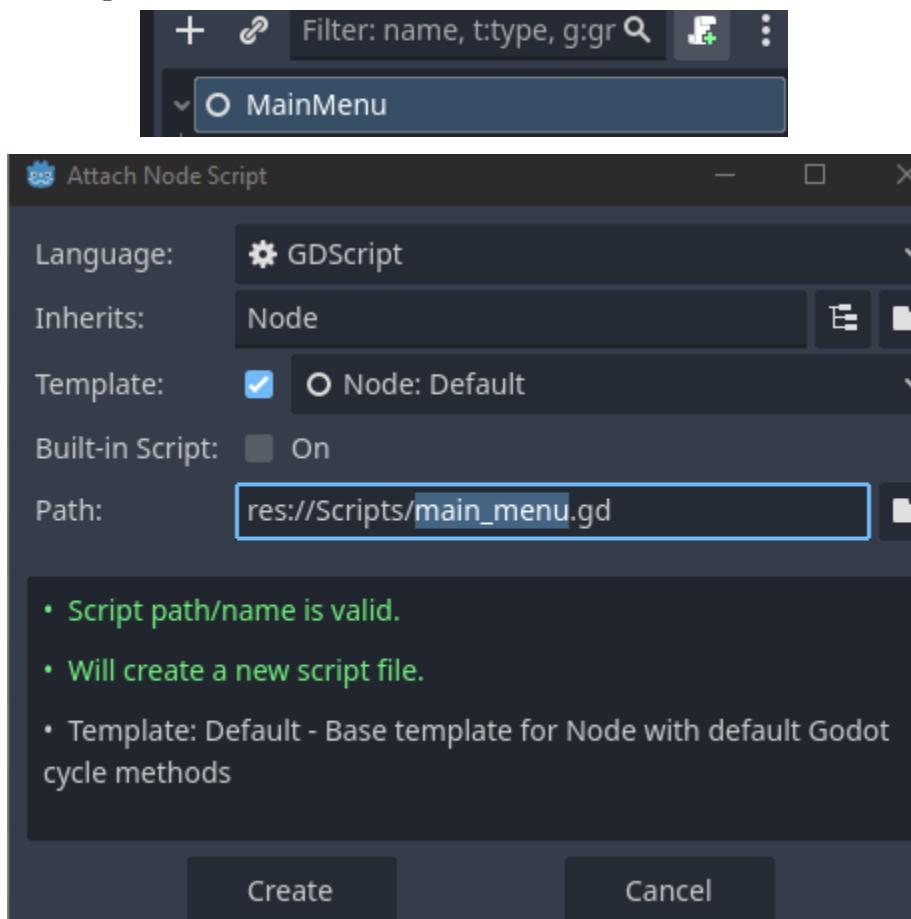


- Chọn main menu làm scene chính, mỗi khi chạy game thì main menu sẽ luôn được ưu tiên chạy trước
- (Chuột phải **main_menu.tscn**) -> Set as Main Scene)

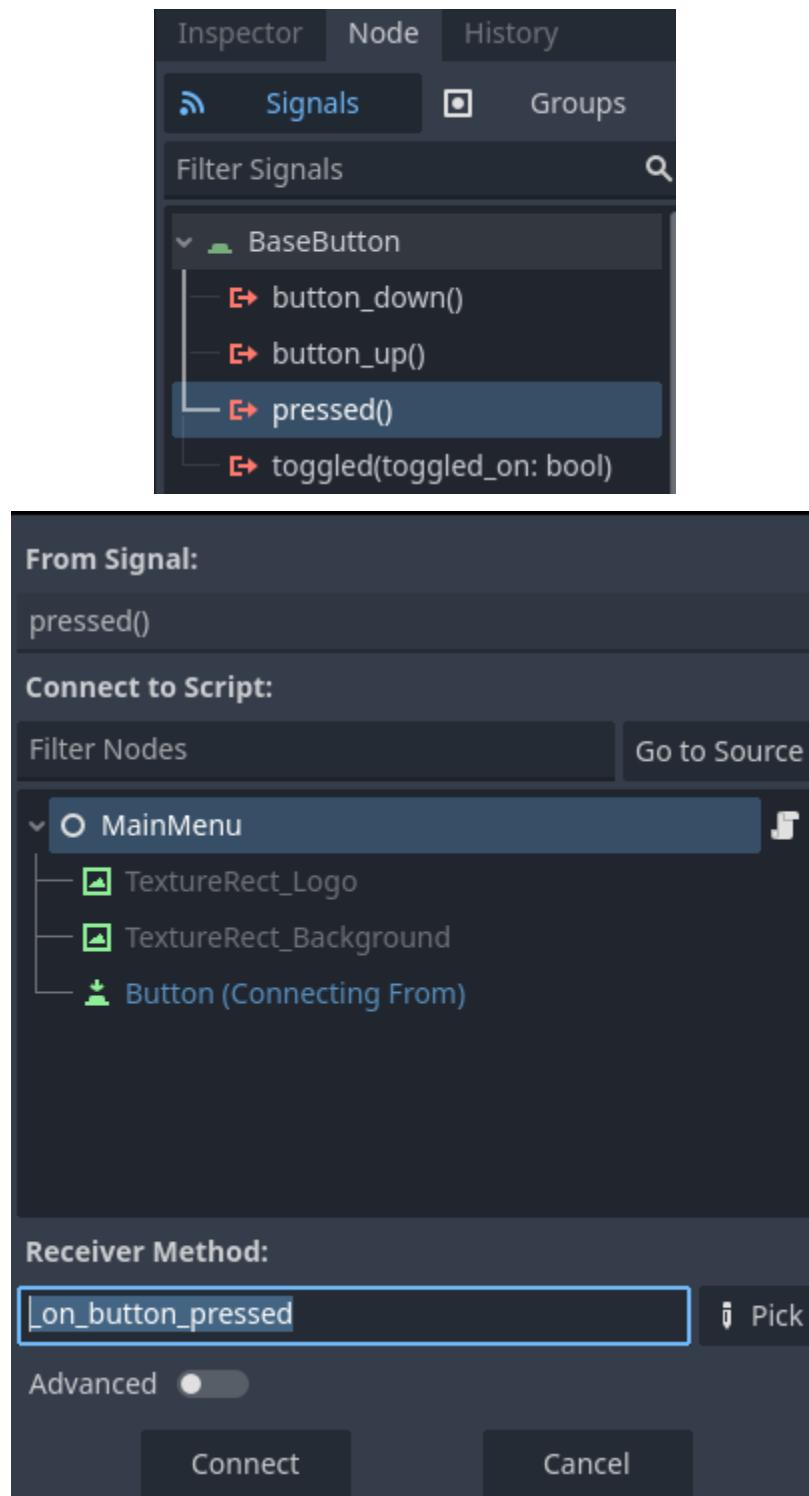


4.2.21. Tạo chuyển cảnh giữa main menu và các màn chơi

- Tạo file script cho main menu



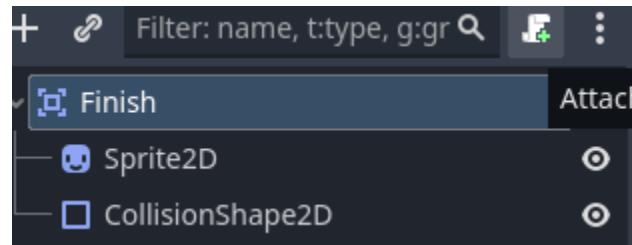
- Chọn Button và tại menu Node chọn tín hiệu pressed() và kết nối tới node main menu



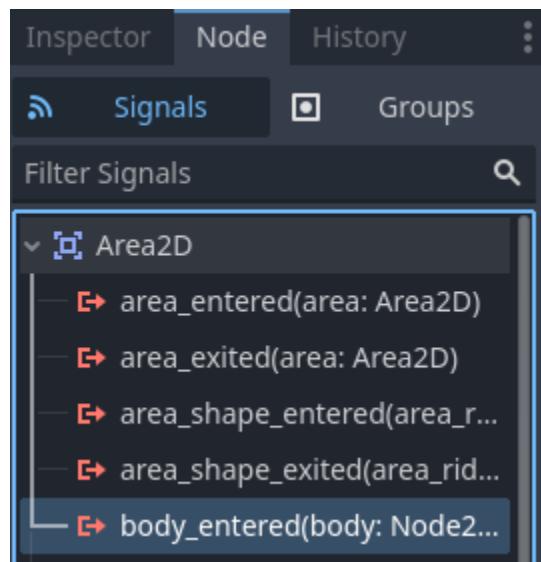
- Chính sửa logic code tín hiệu như sau

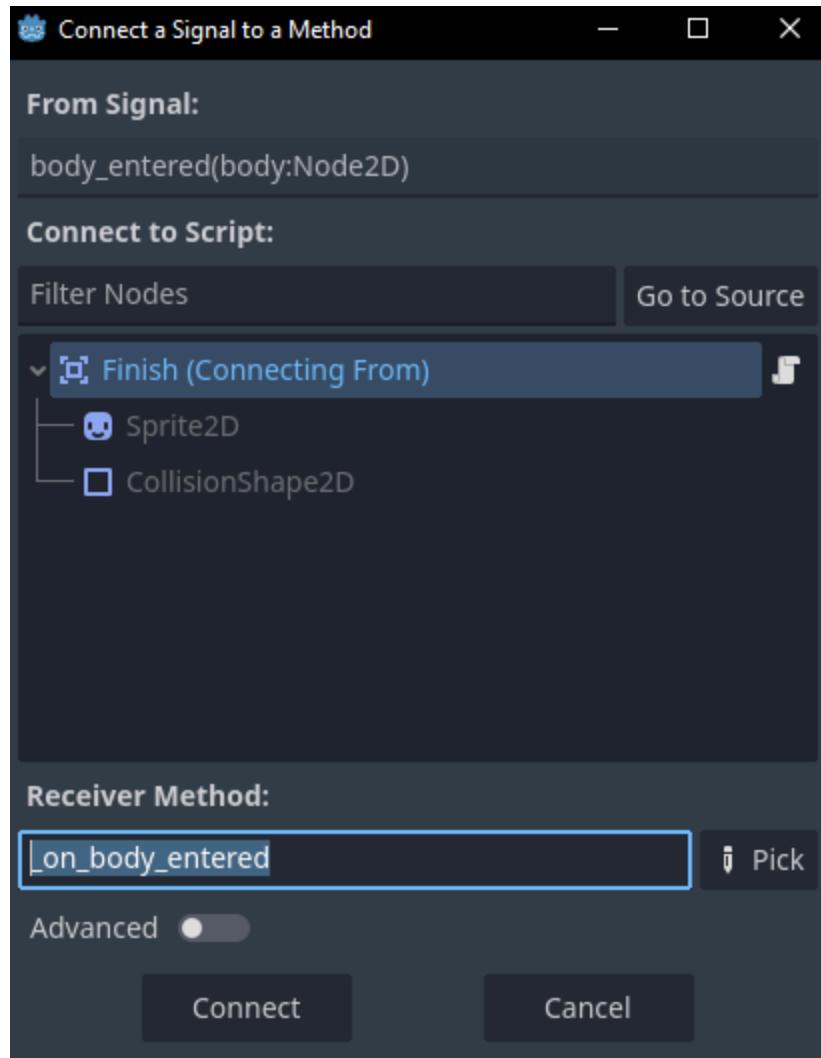
```
11 func _on_button_pressed() -> void:
12     get_tree().change_scene_to_file("res://Scenes/level1.tscn")
13
```

- Khi chạy game nhấn Start sẽ dẫn ta tới level 1
- Để màn chơi chuyển từ màn 1 sang màn 2 và về menu ta cần sửa logic script cho node Area2D đích đến (mục 4.2.18)
- Thêm Script cho đích đến



- Chọn tín hiệu body_entered tại menu Node

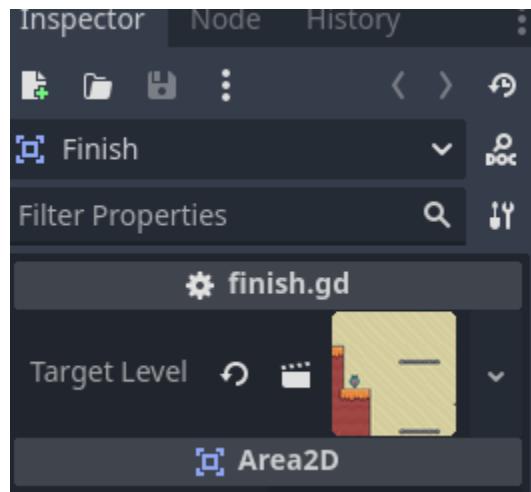




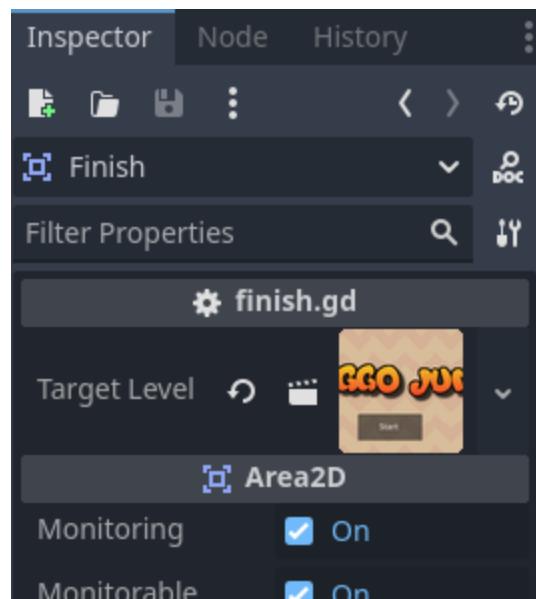
- Sửa logic code như sau, code sẽ lấy scene hiện tại và dẫn tới scene tiếp theo dựa theo cài đặt ở bước tiếp theo

```
1  extends Area2D
2
3  @export var target_level: PackedScene
4
5  func _on_body_entered(body: Node2D) -> void:
6    if(body.name == "CharacterBody2D"):
7      get_tree().change_scene_to_packed(target_level)
8
```

- Trong scene level 1, click chọn đối tượng đích đến và nhìn menu bên phải
- Kéo scene level 2 vào trong mục Target Level

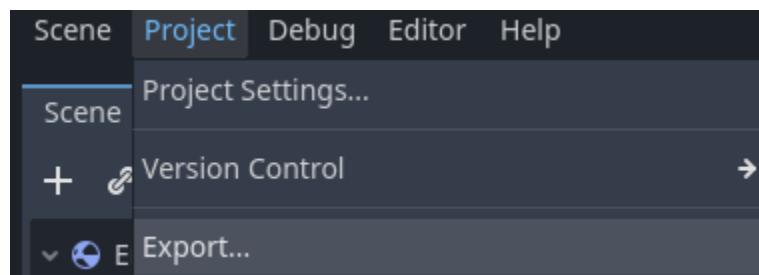


- Làm tương tự với đích đến trong level 2 dẫn về main menu

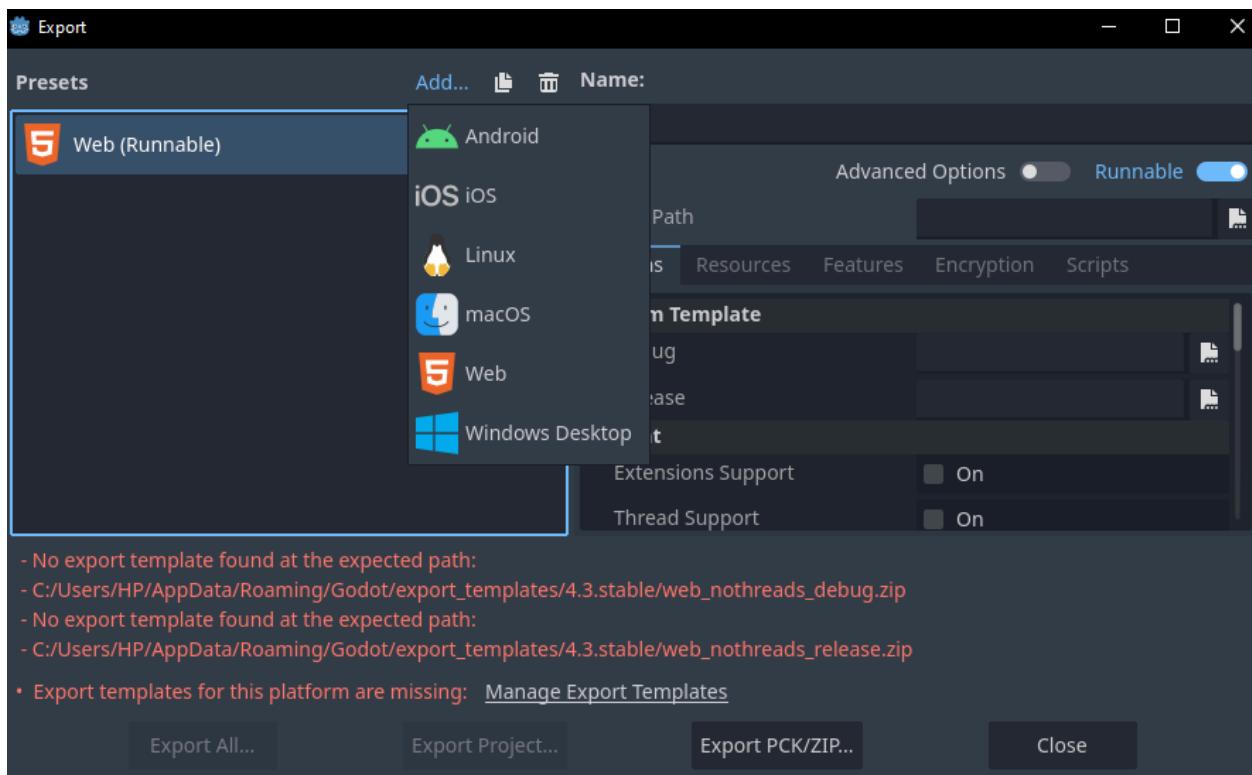


5. Xuất bản game lên Web

- Ta cần xuất game bằng cách nhấn vào Project trên thanh công cụ phía trên -> Export

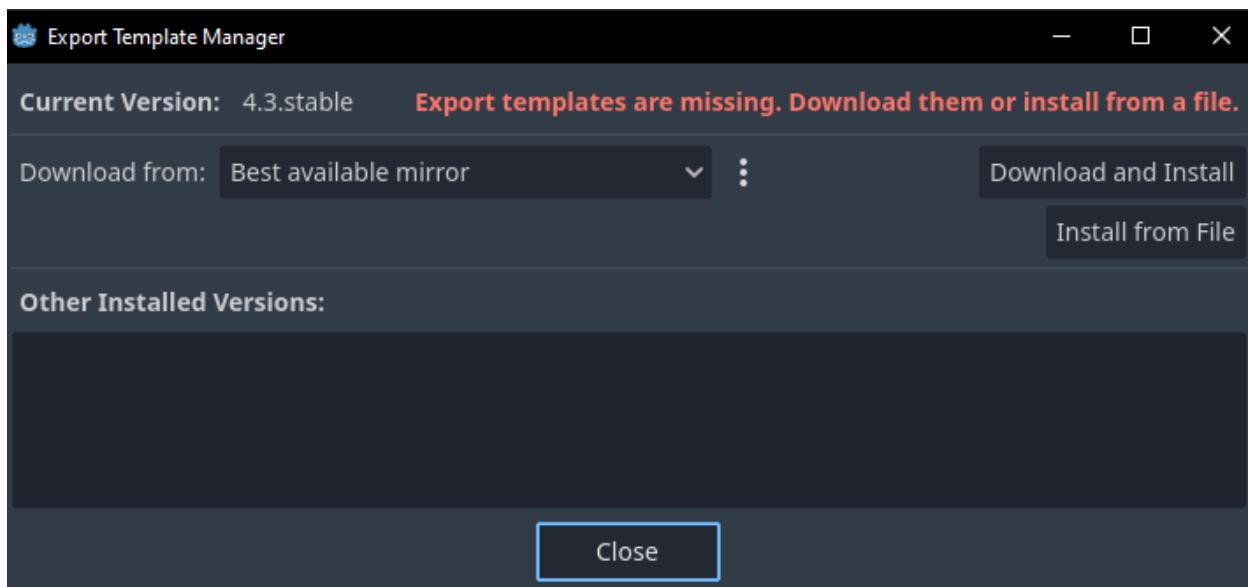


- Nhấn Add và chọn Web

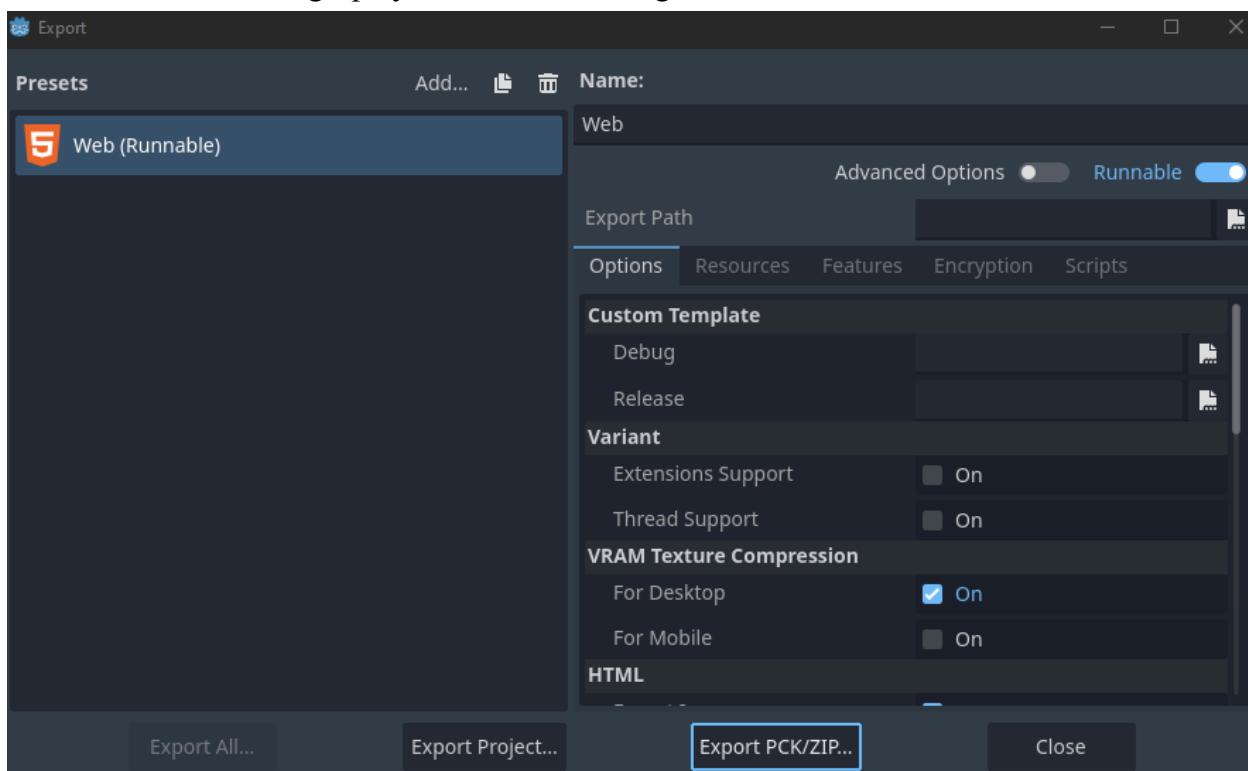


- Lúc này hệ thống sẽ báo lỗi rằng ta chưa có Export Templates -> Nhấn vào Manage Export Templates

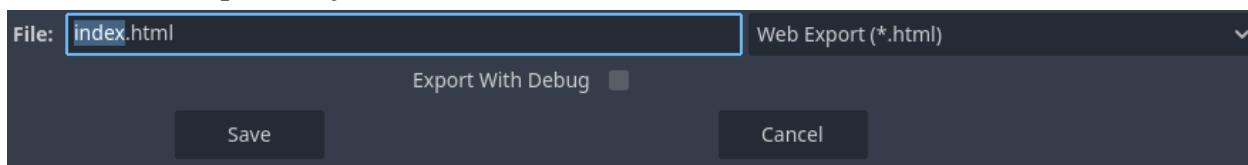
- Nhấn Download and Install



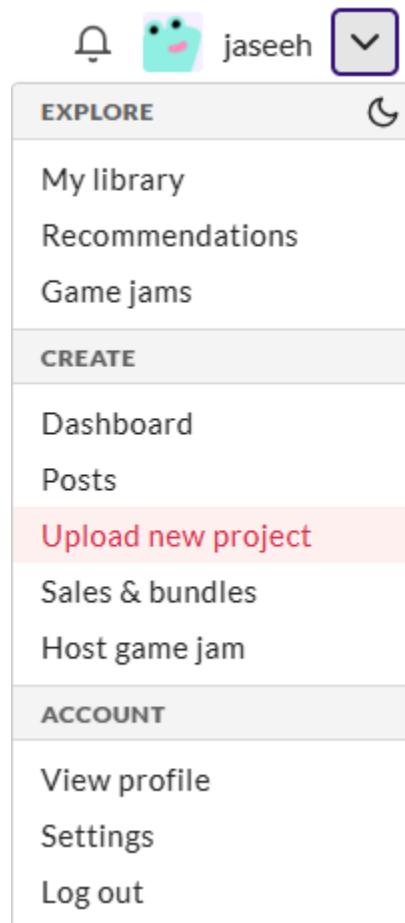
- Sau khi tải xong, quay lại menu sẽ không còn báo lỗi



- Chọn Export Project, đặt tên file index.html và chọn thư mục xuất



- Chuyển thư mục thành file zip
- Vào trang <https://itch.io> tạo tài khoản và chọn Upload new project tại menu góc phải trên



- Điền thông tin cần thiết và chọn Upload File

Uploads

[Upload files](#)

or

Choose from Dropbox

[Add External file](#) ?

File size limit: 1 GB. [Contact us](#) if you need more space

TIP Use **butler** to upload files: it only uploads what's changed, generates patches for the [itch.io app](#), and you can automate it. [Get started!](#)

- Loại project ta chọn HTML

Kind of project

HTML – You have a ZIP or HTML file that will be played in the browser

TIP You can add additional downloadable files for any of the types above

- Bấm Save & view page ở cuối trang
- Itch.io cũng sẽ cung cấp cho ta đường link đến web của game trên thanh địa chỉ

CHƯƠNG III: KẾT QUẢ VÀ ĐÁNH GIÁ

1. Phân tích ưu/nhược điểm của Godot

- **Ưu điểm:**
 - + Mã nguồn mở và hoàn toàn miễn phí
 - + Không cần trả phí bản quyền khi phát hành game
 - + Hệ thống Scene và Node trực quan
 - + Dễ dàng tổ chức và tái sử dụng các thành phần trong game
 - + Tích hợp sẵn công cụ chỉnh sửa không cần cài đặt phần mềm ngoài để chỉnh sửa giao diện UI, tạo animation, v.v.
 - + Có hỗ trợ kéo-thả giúp giảm bớt việc viết code thủ công
 - + Hiệu suất tốt, nhẹ, phù hợp cho game 2D
 - + Hỗ trợ xuất game đa nền tảng Windows, macOS, Linux, Android, iOS, HTML5
 - + Nhiều tài liệu hướng dẫn, diễn đàn hỗ trợ từ cộng đồng lập trình viên, cập nhật thường xuyên và ngày càng cải thiện
- **Nhược điểm:**
 - + 3D chưa thực sự mạnh mẽ so với Unity và Unreal
 - + Hệ sinh thái chưa phổ biến bằng Unity nên ít cơ hội việc làm hơn
 - + Thiếu tài nguyên & asset store chuyên nghiệp

2. Ứng dụng thực tế và tiềm năng phát triển

- Phát triển mạnh trong cộng đồng mã nguồn mở
- Ngày càng có nhiều lập trình viên đóng góp phát triển engine
- Cải tiến đồ họa và 3D trong tương lai

CHƯƠNG IV: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Tổng kết nghiên cứu

- Godot Engine là một công cụ phát triển game mạnh mẽ, đặc biệt là trong lĩnh vực game 2D. Với hệ thống Scene và Node trực quan, ngôn ngữ GDScript dễ học, và khả năng xuất game đa nền tảng, Godot trở thành một lựa chọn hấp dẫn cho lập trình viên indie
- Dù còn một số hạn chế như hệ sinh thái chưa rộng lớn và 3D chưa mạnh bằng Unity, nhưng với sự phát triển không ngừng của cộng đồng mã nguồn mở, Godot đang dần hoàn thiện hơn

2. Đề xuất hướng nghiên cứu tiếp theo

- Phát triển game online với Godot
- Khám phá sâu hơn về hệ thống 3D của Godot
- Nâng cao hiệu suất game 2D

CHƯƠNG V: TÀI LIỆU THAM KHẢO

<https://www.gdquest.com/tutorial/godot/> (Godot tutorials)

<https://docs.godotengine.org/en/stable/index.html> (Godot docs)

<https://youtu.be/LOhfqjmasi0?si=jYvcrR84G6hhYr29> (Godot2D của Brackeys)

<https://www.youtube.com/watch?v=5V9f3MT86M8>

https://youtu.be/zL_a0Ei6Vs?si=EaQpT93JDtSj8-I

<https://youtu.be/kBzV7vgdQfU?si=DZSb5szwpDgOjf7t> (Series Godot2D của Coco Code)