ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
KHOA KHOA HỌC MÁY TÍNH
----------o0o----------

PROJECT III

Xây dựng trò chơi 2D Action RPG

GVHD: TS.Phạm Đăng Hải
SVTH: Nông Thanh Huy
MSSV: 20204567

TP. HÀ NỘI, THÁNG 6 NĂM 2024

Mục lục

CHƯƠNG 1: GIỚI THIỆU

I. Tổng quan

Thời đại 4.0 cùng với việc phát triển không ngừng của ngành công nghệ thông tin, các thiết bị công nghệ như máy tính, máy chơi game, điện thoại ngày càng phổ biến. Đi đôi với đó là nhu cầu giải trí ngày càng cao của con người, đặc biệt là thế hệ trẻ. Vì vậy nhu cầu này mở ra thị trường trò chơi trên các thiết bị máy tính, điện thoại.

II. Mô tả đề tài

1. Giới thiệu đề tài

Xây dựng một trò chơi 2D Action RPG sử dụng công cụ Game Engine Unity và công cụ lập trình Visual Studio Code.

2. Mục đích

- Làm quen, nâng cao kinh nghiệm sử dụng công cụ và môi trường lập trình.
- Nâng cao tư duy thiết kế trò chơi và khả năng thiết kế, lập trình phát triển hệ thống trò chơi.
- Hoàn thiện được một dự án trò chơi đơn giản.

3.Yêu cầu

- Thiết kế được kịch bản game đơn giản.
- Xây dựng được hệ thống trò chơi.
- Xây dựng được các tính năng chính của trò chơi.
- Triển khai trò chơi trên nền tảng máy tính.
- Trò chơi được cài đặt dễ dàng và vận hành mượt mà

III. Cơ sở lý thuyết

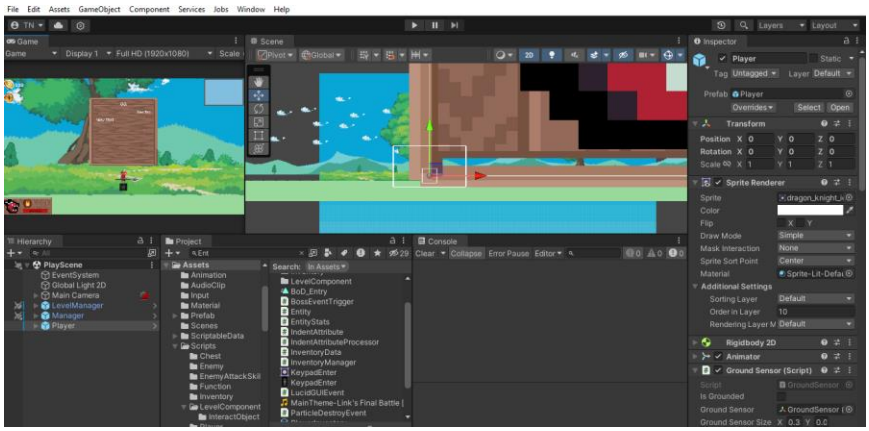
1. Giới thiệu công cụ Unity

- Unity là một game engine đa nền tảng được phát triển bởi Unity Technologies, mà chủ yếu để phát triển video game cho máy tính, consoles và điện thoại. Lần đầu tiên nó được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, đến nay đã mở rộng 27 nền tảng.
- Ưu điểm:
  - Chức năng cốt lõi đa dạng bao gồm: cung cấp công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2D hoặc 3D, công cụ vật lý (tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động, trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng

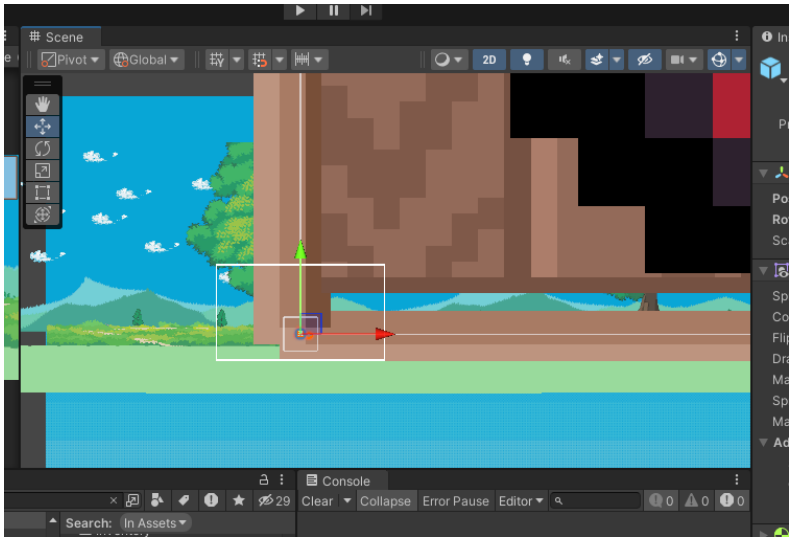
ảnh đồ thị và kết nối mạng. Nhờ có các engine mà công việc làm game trở nên ít tốn kém và đơn giản hơn.

- Hỗ trợ đa nền tảng: Một trong các thế mạnh của Unity3D chính là khả năng hỗ trợ gần như toàn bộ các nền tảng hiện có bao gồm: PlayStation 3, Xbox 360, Wii U, iOS, Android, Windows, Blackberry 10, OS X, Linux, trình duyệt Web và cả Flash. Nói cách khác, chỉ với một gói engine, các studio có thể làm game cho bất kỳ hệ điều hành nào và dễ dàng convert chúng sang những hệ điều hành khác nhau. Đồng thời, đây cũng là giải pháp cho các game online đa nền tảng – có thể chơi đồng thời trên nhiều hệ điều hành, phần cứng khác nhau như Web, PC, Mobile, Tablet….
- Dễ sử dụng: Unity3D được built trong một môi trường phát triển tích hợp, cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi. Do được hướng đến đồng thời cả lập trình viên không chuyên và studio chuyên nghiệp, nên Unity3D khá dễ sử dụng. Hơn nữa, đây là một trong những engine phổ biến nhất trên thế giới, người dùng có thể dễ dàng tìm kiếm kinh nghiệm sử dụng của "tiền bối" trên các forum công nghệ.
- Tính kinh tế cao: Unity Technologies hiện cung cấp bản miễn phí engine Unity3D cho người dùng cá nhân và các doanh nghiệp có doanh thu dưới 100.000 USD/năm. Với bản Pro, người dùng phải trả 1.500 USD/năm – một con số rất khiêm tốn so với những gì engine này mang lại.

2. Một số thành phần trong UnityEditor



- Cửa sổ Scene:

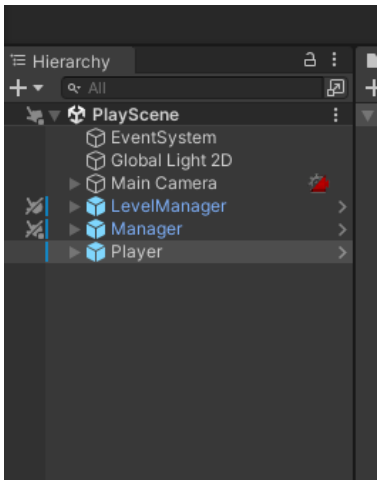


Phần này phần hiển thị các đối tượng trong scene một cách trực quan, có thể lựa chọn các đối tượng, kéo thả, phóng to, thu nhỏ, xoay các đối tượng ...

Phần này có để thiết lập một số thông số như hiển thị ánh sáng, âm anh, cách nhìn 2D hay 3D ...

Khung nhìn Scene là nơi bố trí các GameObject như cây cối, cảnh quan, enemy, player, camera, … trong game. Sự bố trí hoạt cảnh là một trong những chức năng quan trọng nhất của Unity.

- Cửa sổ Hierarchy:



Tab hierarchy là nơi hiển thị các GameObject trong Scene hiện hành. Khi các đối tượng được thêm hoặc xóa trong Scene, tương ứng với các đối tượng đó trong cửa sổ Hierarchy.

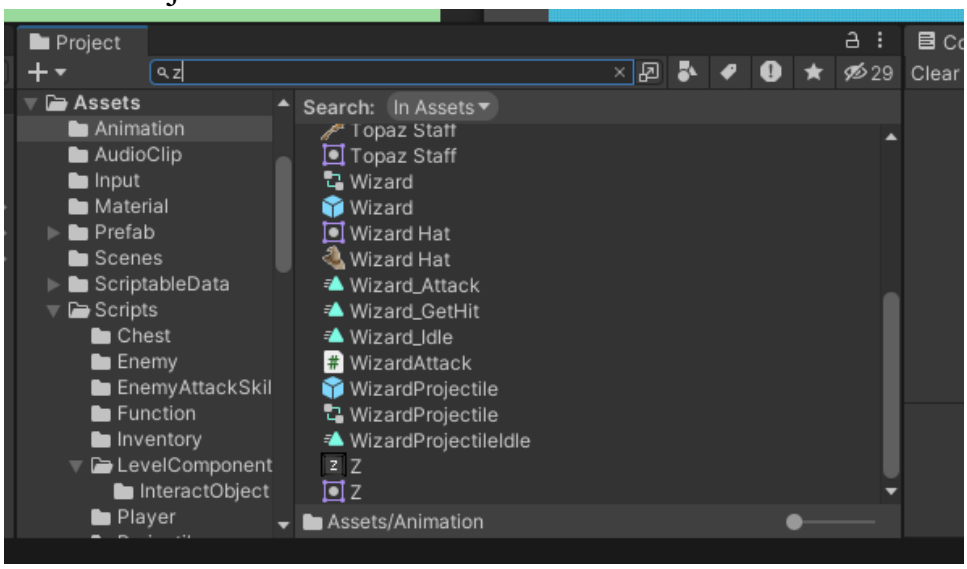
Tương tự trong tab Project, Hierarchy cũng có một thanh tìm kiếm giúp quản lý và thao tác với các Game Object hiệu quả hơn đặc biệt là với các dự án lớn.

- Cửa sổ Game:

Đây là màn hình demo Game, là góc nhìn từ camera trong game.

Thanh công cụ trong cửa sổ game cung cấp các tùy chỉnh về độ phân giải màn hình, thông số (stats), gizmos, tùy chọn bật tắt các component...

- Cửa sổ Project:



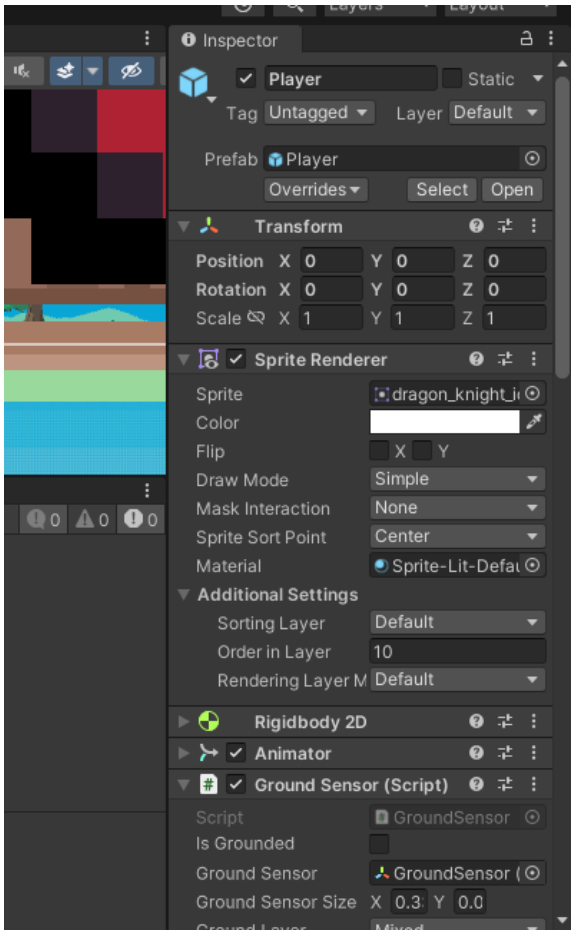
Đây là cửa sổ explorer của Unity, hiển thị thông tin của tất cả các tài nguyên (Assets) trong game của bạn.

Cột bên trái hiển thị assets và các mục yêu thích dưới dạng cây thư mục tương tự như Windows Explorer. Khi click vào một nhánh trên cây thư mục thì toàn bộ nội dung của nhánh đó sẽ được hiển thị ở khung bên phải. Ta có thể tạo ra các thư mục mới bằng cách Right click -> Create -> Folder hoặc nhấn vào nút Create ở góc trên bên trái cửa sổ Project và chọn Folder. Các tài nguyên trong game cũng có thể được tạo ra bằng cách này.

Phía trên cây thư mục là mục Favorites, giúp chúng ta truy cập nhanh vào những tài nguyên thường sử dụng. Chúng ta có thể đưa các tài nguyên vào Favorites bằng thao tác kéo thả.

Đường dẫn của thư mục tài nguyên hiện tại. Chúng ta có thể dễ dàng tiếp cận các thư mục con hoặc thư mục gốc bằng cách click chuột vào mũi tên hoặc tên thư mục.

- Cửa sổ Inspector:



Cửa sổ Inspector hiển thị chi tiết các thông tin về Game Object đang làm việc, kể cả những component được đính kèm và thuộc tính của nó. Bạn có thể điều chỉnh, thiết lập mọi thông số và chức năng của Game Object thông qua cửa sổ Inspector.

Mọi thuộc tính thể hiện trong Inspector đều có thể dễ dàng tuỳ chỉnh trực tiếp mà không cần thông qua một kịch bản định trước. Tuy nhiên Scripting API cung cấp một số lượng nhiều và đầy đủ hơn do giao diện Inspector là có giới hạn.

Các thiết lập của từng component được đặt trong menu. Các bạn có thể click chuột phải, hoặc chọn icon hình bánh răng nhỏ để xuất hiện menu.

Ngoài ra Inspector cũng thể hiện mọi thông số Import Setting của asset đang làm việc như hiển thị mã nguồn của Script, các thông số animation, …

3. Một số khái niệm trong Unity

- GameObject:
  Một đối tượng cụ thể trong game gọi là một game object, có thể là nhân vật, đồ vật nào đó. Ví dụ: cây cối, xe cộ, nhà cửa, người...

- Component:
  Một GameObject sẽ có nhiều thành phần cấu tạo nên nó như là hình ảnh (sprite render), tập hợp các hành động (animator), thành phần xử lý va chạm (collision), tính toán vật lý (physical), mã điều khiển (script), các thành phần khác... mỗi thứ như vậy gọi là một component của GameObject.

- Sprite:
  Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó.

- Animation:
  Là tập một hình ảnh động dựa trên sự thay đổi liên tục của nhiều sprite khác nhau.

- Key Frame:
  Key Frame hay Frame là một trạng thái của một animation. Có thể được tạo nên từ 1 sprite hay nhiều sprite khác nhau.

- Prefabs:
  Là một khái niệm trong Unity, dùng để sử dụng lại các đối tượng giống nhau có trong game mà chỉ cần khởi tạo lại các giá trị vị trí, tỉ lệ biến dạng và góc quay từ môt đối tượng ban đầu. Ví dụ: Các đối tượng là đồng tiên trong game Mario đều có xử lý giống nhau, nên ta chỉ việc tạo ra một đối tượng ban đầu, các đồng tiền còn lại sẽ sử dụng prefabs. Hoặc khi ta lát gạch cho một cái nền nhà, các viên gạch cũng được sử dụng là prefabs.

- Sounds:
  Âm thanh trong game.

- Script:
  Script là tập tin chứa các đoạn mã nguồn, dùng để khởi tạo và xử lý các đối tượng trong game. Trong Unity có thể dùng C#, Java Script, BOO để lập trình Script.

- Scenes:
  Quản lý tất cả các đối tượng trong một màn chơi của game.

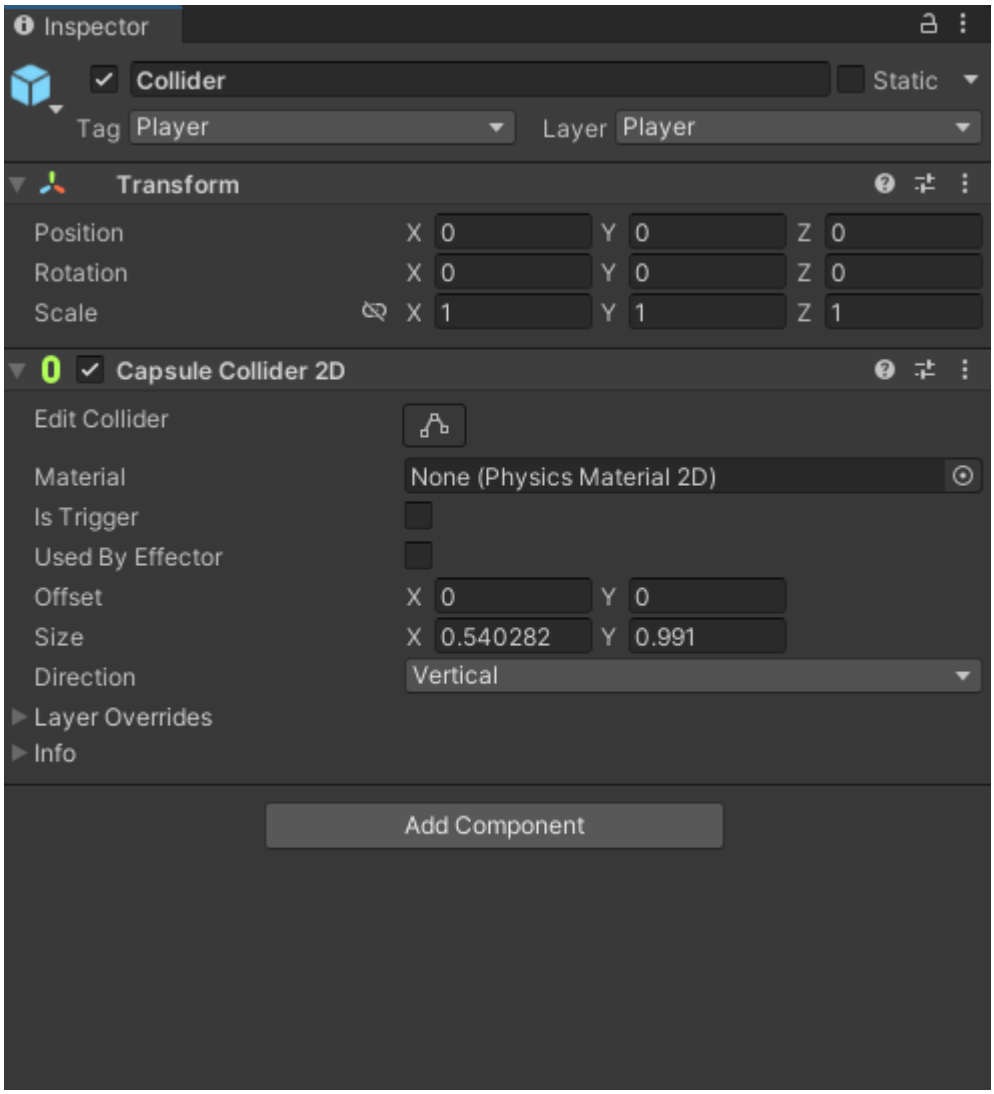
- Assets:
  Bao gồm tất cả những gì phục vụ cho dự án game như sprite, animation, sound, script, scenes…iảm thiểu rất nhiều thời gian cho việc thiết kế và lập trình game

  Assets là tài nguyên xây dựng nên một dự án trên Unity. Những tài nguyên có thể là hình ảnh, âm thanh, mô hình 2D 3D, chất liệu (material), texture vv hoặc cả một project hoàn chỉnh. Các asset do chính những nhà phát triển game tạo ra và có thể được download miễn phí hoặc trả phí trên Unity Asset Store.

- Camera:
  Là một game object đặc biệt trong scene, dùng để xác định tầm nhìn, quan sát các đối tượng khác trong game và hiển thị lên màn hình trò chơi.

- Transform:
  Là 3 phép biến đổi tịnh tiến, quay theo các trục, và phóng to thu nhỏ một đối tượng

- Collider:
  Hệ thống xử lý va chạm của Unity bao gồm 2d và 3d.



IsTrigger cho phép đi xuyên qua và gọi đến các hàm : OnTriggerEnter, OnTriggerStay, OnTriggerExit…

- Rigidbody:
  Hệ thống mô phỏng vật lí trong game.

- Renderer:
  Các SpriteRenderer: thành phần cho phép bạn hiển thị hình ảnh như Sprites để sử dụng trong cả 2D và 3D.

- Instance:
  Là 1 thể hiện của một lớp/script/component.

Có thể một thể hiện của một lớp là một GameObject trong Scene mà lớp đó được gắn vào.

- ScriptableObject:
  Là một container lưu trữ data trong Unity được định nghĩa từ một class.

4. Một số design pattern

Là cách thiết kế các lớp trong Unity:

- Singleton Pattern:
  Mỗi một lớp chỉ có duy nhất một thể hiện và cung cấp quyền truy cập toàn cục vào thể hiện của nó cho các lớp khác.
- Observer Pattern:
  Là một mô hình thiết kế hành vi mà xác định một sự phụ thuộc một-nhiều giữa các đối tượng. Nó cho phép một đối tượng, được gọi là đối tượng chủ đề, thông báo cho các đối tượng khác, được gọi là quan sát viên, về các thay đổi trong trạng thái của nó.
- Command Pattern:
  Là một mô hình thiết kế hành vi, cho phép đóng gói một yêu cầu thành một đối tượng, tách biệt đối tượng yêu cầu với đối tượng thực hiện yêu cầu. Điều này tạo ra mối liên hệ rõ ràng và đặc biệt hữu ích cho chức năng hoàn tác/phục hồi hoặc triển khai hệ thống đầu vào linh hoạt.
- Strategy Pattern:
  Là một mô hình thiết kế hành vi, xác định một họ thuật toán, đóng gói từng thuật toán và làm cho chúng có thể thay thế lẫn nhau. Nó cho phép lựa chọn một thuật toán tại thời điểm chạy, tạo điều kiện linh hoạt và thích nghi trong đối tượng gọi đến nó.
- Factory Pattern:
  Là một mô hình thiết kế tạo đối tượng, cung cấp một giao diện để tạo đối tượng trong một lớp cha, cho phép các lớp con quyết định lớp nào sẽ được khởi tạo.
- Decorator Pattern:
  Là một mô hình thiết kế cấu trúc, cho phép đính kèm các hàm/chức năng mới cho một đối tượng một cách động, mà không cần thay đổi cấu trúc của nó. Mô hình này đặc biệt hữu ích khi bạn cần thêm chức năng vào một đối tượng mà không ảnh hưởng đến các phiên bản khác của cùng một lớp.
- State Pattern:

Là một mô hình thiết kế hành vi, cho phép một đối tượng thay đổi hành vi của mình khi trạng thái nội bộ của nó thay đổi. Mô hình này đặc biệt hữu ích để triển khai các máy trạng thái hữu hạn một cách dễ dàng và dễ bảo trì.

IV. Tổng quan kịch bản trò chơi

1. Giới thiệu trò chơi

Tựa game 2D action RPG màn hình ngang dạng Platformer. Người chơi di chuyển nhân vật trên địa hình, khám phá màn chơi, thu thập vật phẩm và tiêu diệt quái vật trên đường đi. Trò chơi sẽ kết thúc khi người chơi vượt qua các màn chơi nhất định và tiêu diệt được boss.

2. CoreLoop

- Người chơi sẽ thực hiện lặp lại các hoạt động sau:

  Khám phá màn chơi ->Tấn công quái vật,nhặt item -> Lấy kinh nghiệm, nâng cao chỉ số -> Khám phá màn chơi

Người chơi sẽ giành chiến thắng khi tiêu diệt được boss tại màn chơi cuối cùng.

3. Tổng quan các tính năng chính

- Hệ thống điều khiển và hành vi:
  - Điều khiển của người chơi: di chuyển, kỹ năng, tấn công, tương tác với vật phẩm, object, npc.....
  - Hành vi của quái vật: di chuyển tuần tra, phát hiện người chơi, đuổi, tấn công, kỹ năng tấn công,....
- Hệ thống màn chơi:
  - Tải màn chơi ngẫu nhiên:
  - Lượng quái vật được sinh ra tùy theo từng màn chơi. Mỗi màn chơi sẽ có các địa hình khác nhau:có thể có các object và trap
- Hệ thống item-vật phẩm: người chơi có hành trang chứa các vật phẩm, vật phẩm giúp tăng chỉ số sức mạnh cho người chơi và có các hiệu ứng riêng.

  NPC(các đối tượng không phải người chơi): người chơi có thể tương tác để thực thực hiện các chức năng như trao đổi vật phẩm, dịch chuyển, hồi máu....
- Hệ thống trò chơi:

  Hệ thống quản lý trò chơi, quản lý giao diện.....

  Đồ họa: Hoạt ảnh nhân vật, bối cảnh môi trường, animation,UI

Âm thanh: Âm thanh môi trường, âm thanh khi di chuyển,...

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

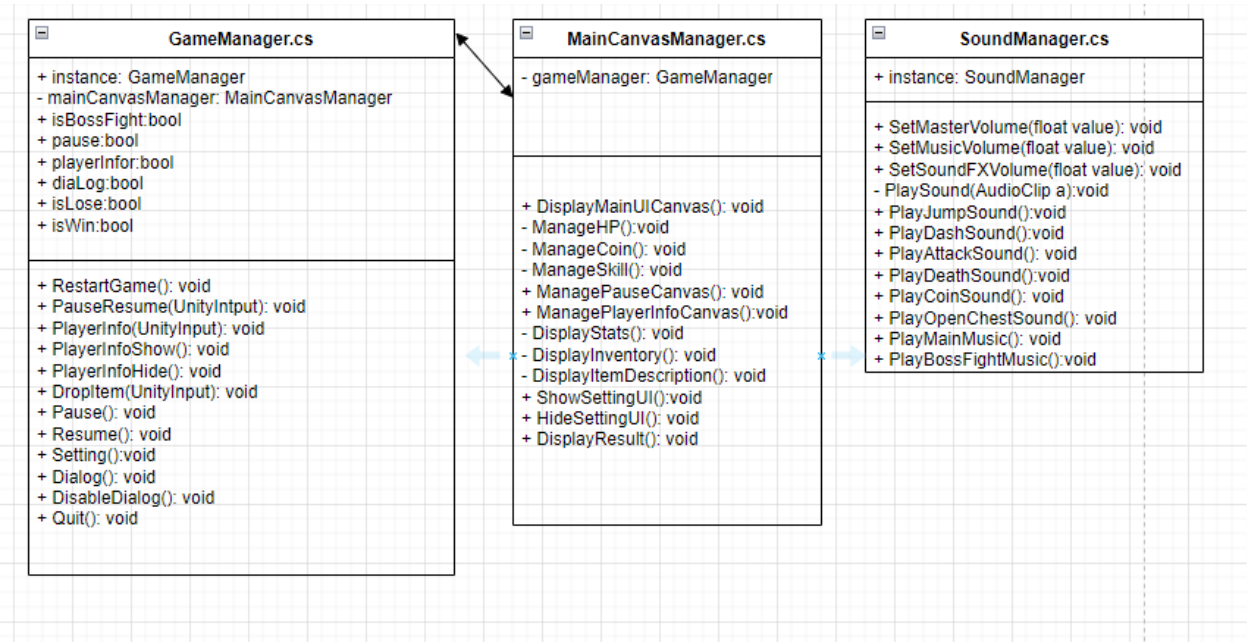
I. Các lớp

STT	Tên lớp	Chức năng
1	Defines.cs	Định nghĩa các chuỗi ký tự dùng trong dự án.
2	GameManager.cs	Quản lý các luồng hay trạng thái của trò chơi: Pause, Setting, win/lose…
3	SoundManager.cs	Hệ thống âm thanh
4	StartScene.cs	Chứa các chức năng tại màn hình chờ
5	DamagePopup.cs	UI hiển thị khi gây sát thương
6	DamagePopupManager.cs	Quản lý UI hiển thị khi gây sát thương
7	DialogManager.cs	UI khi trò chuyện với NPC
8	DisplayHP.cs	UI hiển thị thanh máu
9	InfoUIManager.cs	UI hiển thị thông tin của tương tác
10	MainCanvasManager.cs	Quản lý UI chính của trò chơi
11	Level.cs	Chức năng của màn chơi
12	LevelManager.cs	Quản lý load, tạo và sinh màn chơi
13	SpawnEnemy.cs	Sinh quái vật
14	SpawnManager.cs	Quản lý việc sinh quái vật, item, object
15	DetectArea.cs	Nhận diện người chơi và đưa thông tin cho quái vật.
16	BossEventTrigger.cs	Khởi tạo các thông tin khi vào đánh boss.
17	DropedProp.cs	Các object sinh ra khi quái vật bị tiêu diệt.
18	PlayerPositionTracking.cs	Trả về vị trí của người chơi.
19	TrapDamage.cs	Gây sát thương từ bẫy cho người chơi/quái vật.
20	TrapTrigger.cs	Bẫy trên màn chơi.
21	PlayerInteract.cs	Các object để người chơi tương tác
22	ChestManager.cs	Hành vi của hòm, rương object.
23	TransisGate.cs	Di chuyển qua màn chơi.
24	SaleNPC.cs	Hành vi của NPC bán đồ.
25	RefreshSale.cs	Thuộc hành vi của SaleNPC.cs

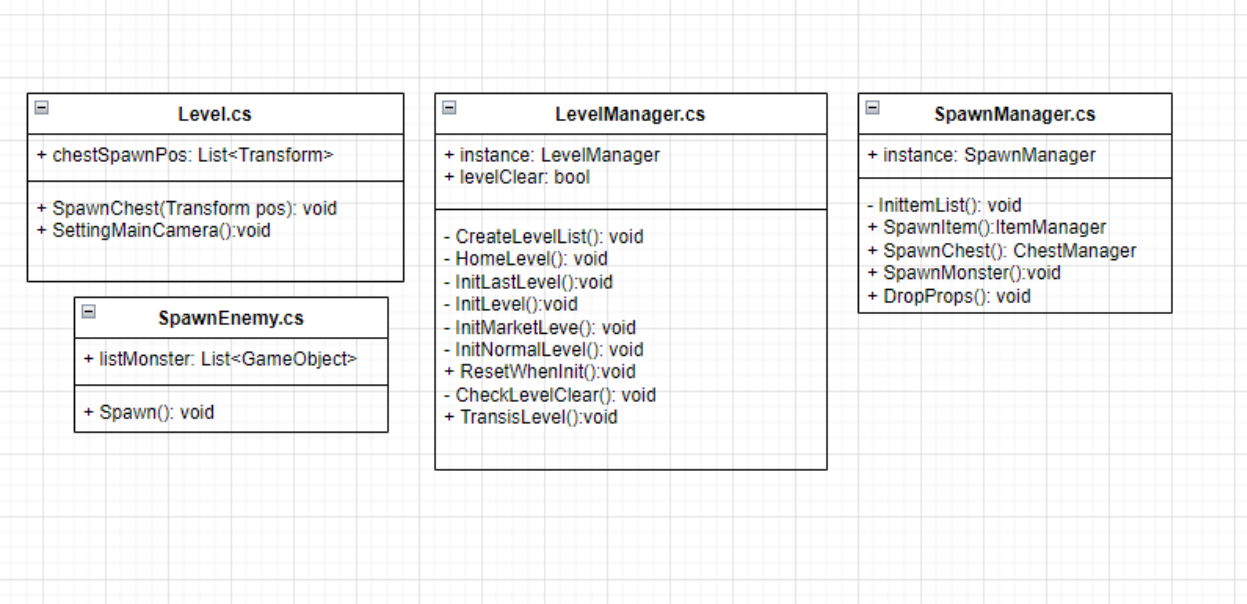
26	Enity.cs	Định nghĩa thông tin thực thể:HP,DEF, ATK….
27	EntityStats.cs	Là 1 ScriptableObject lưu dữ liệu của Entity.
28	Monster_Behavior.cs	Hành vi của quái vật.
29	BoDBehavior.cs	Hành vi của Boss "BoD".
30	AttackSkill.cs	Các đòn tấn công, kĩ năng của quái vật.
31	DashAttack.cs	
32	NormalAttack.cs	
33	WizardAttack.cs	
34	BoD_Teleport.cs	
35	BoDMagicAttack.cs	
36	PlayerStats.cs	Thông tin, chỉ số của người chơi.
37	PlayerMovement.cs	Hành vi của người chơi khi điều khiển.
38	OneWayCheck.cs	Hàm kiểm tra các địa hình.
39	ProjectileBehavior.cs	Hành vi của attack object.
40	Damage.cs	Gây sát thương khi tấn công.
41	DamagePlayer.cs	
42	DamageEnemy.cs	
43	GroundSensor.cs	Nhận diện mặt đất của Entity.
44	ParticleDestroyEvent.cs	Hành vi của hiệu ứng.
45	IdleTransis.cs	Hỗ trợ animation của Player.
46	Transis1.cs	
47	Transis2.cs	
48	ItemStats.cs	Một ScriptableObject lưu giữ thông tin của vật phẩm.
49	ItemManager.cs	Hành vi của vật phẩm.
50	InventoryData.cs	Một ScriptableObject lưu giữ thông tin hành trang của người chơi.
51	InventoryManager.cs	Quản lý hành trang.

II. Biểu đồ lớp

1. Quản lý trò chơi

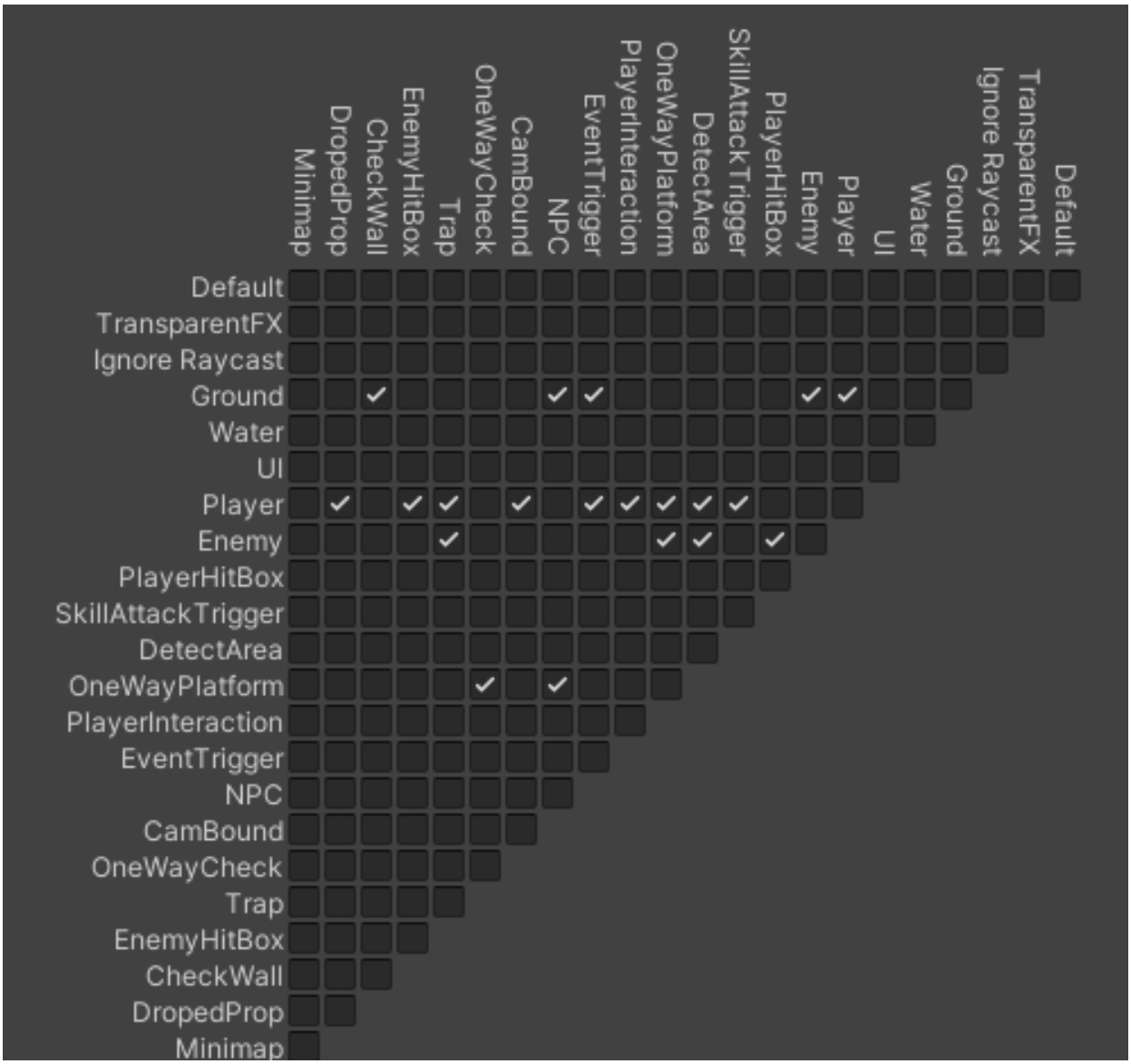


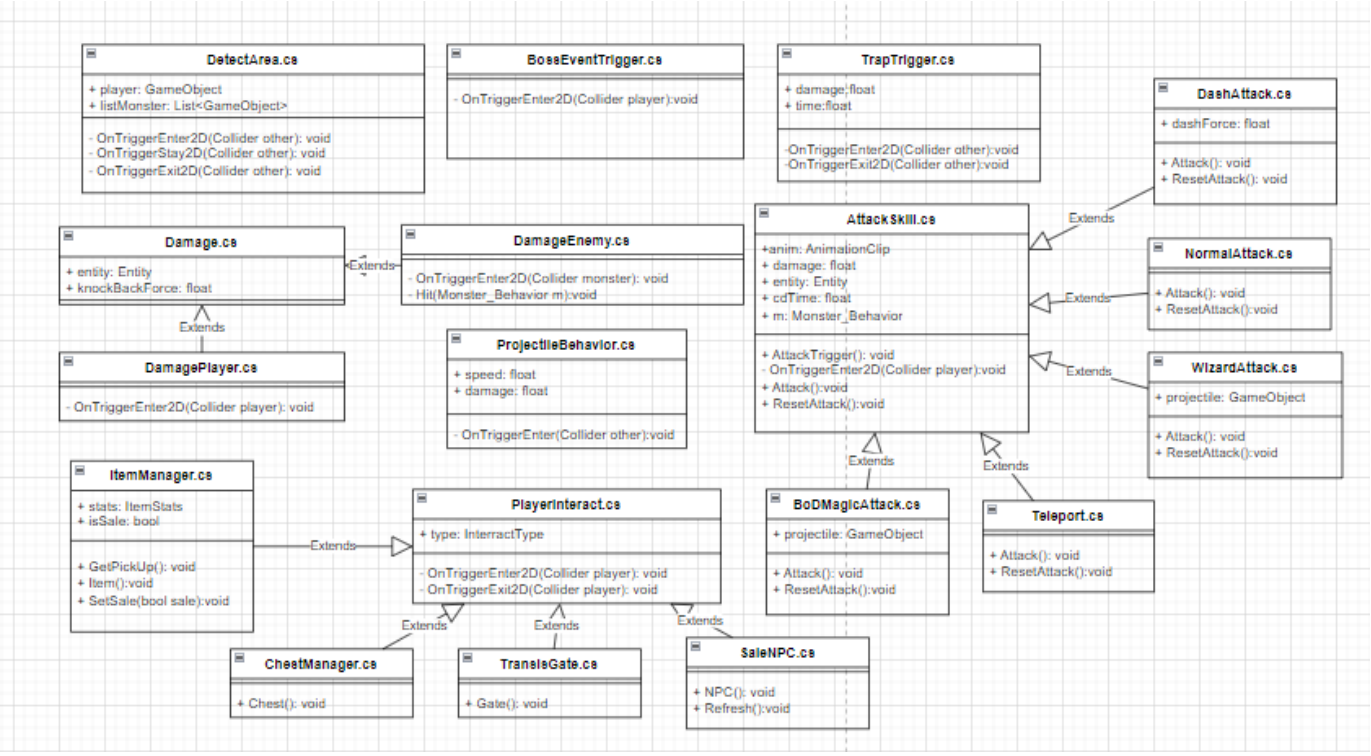
2. Quản lý màn chơi



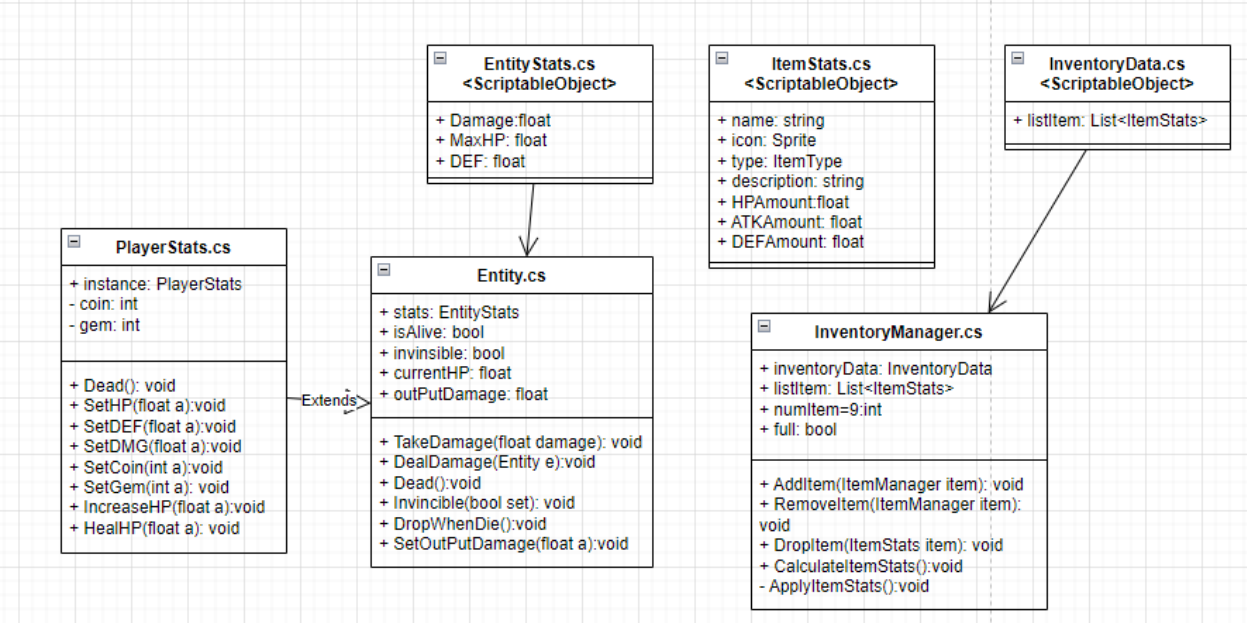
3. Các lớp Collider

Ma trận những collider có thể va chạm/trigger với nhau:

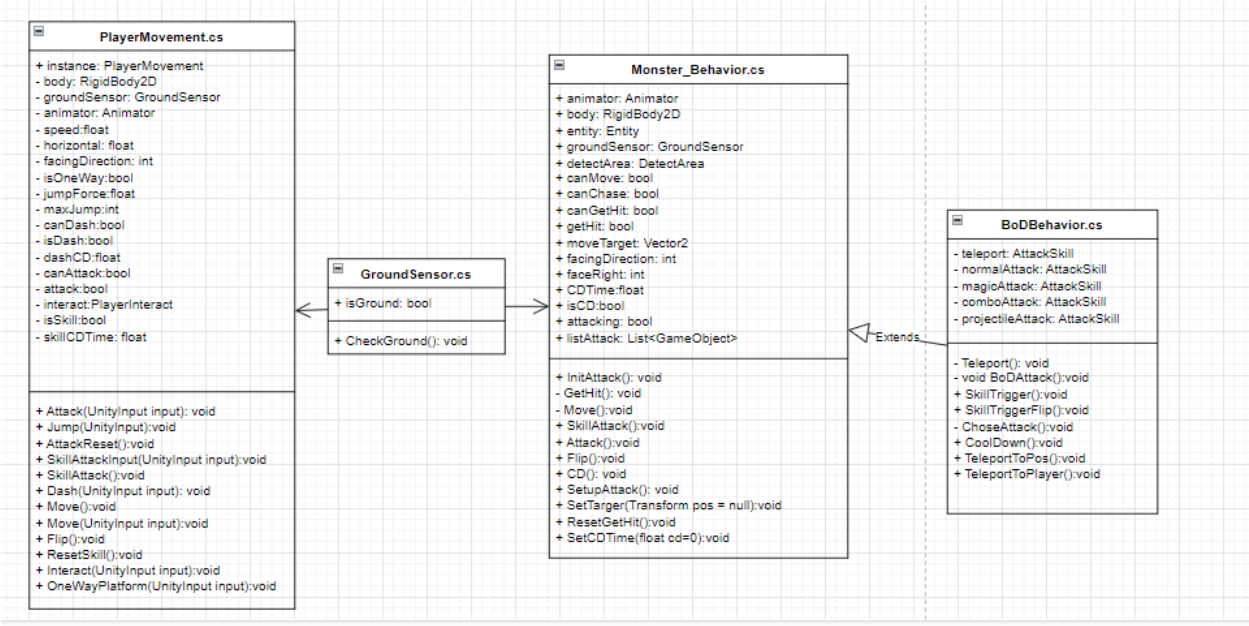

Default
TransparentFX
Ignore Raycast
Ground
Water
UI
Player
Enemy
PlayerHitBox
SkillAttackTrigger
DetectArea
OneWayPlatform
PlayerInteraction
EventTrigger
NPC
CamBound
OneWayCheck
Trap
EnemyHitBox
CheckWall
DropedProp
Minimap



4. Entity, Item

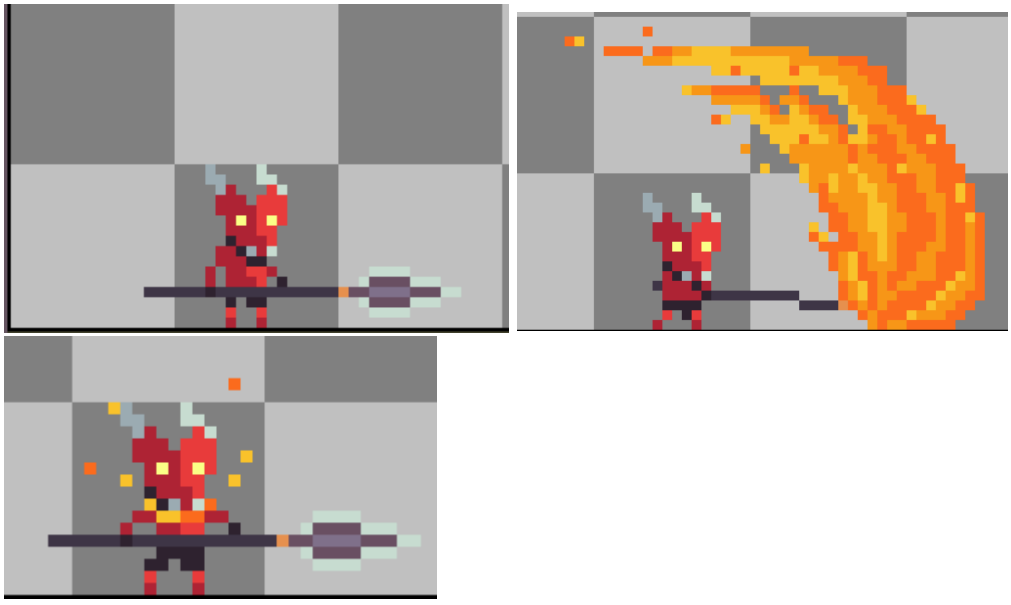


5. Điều khiển, hành vi



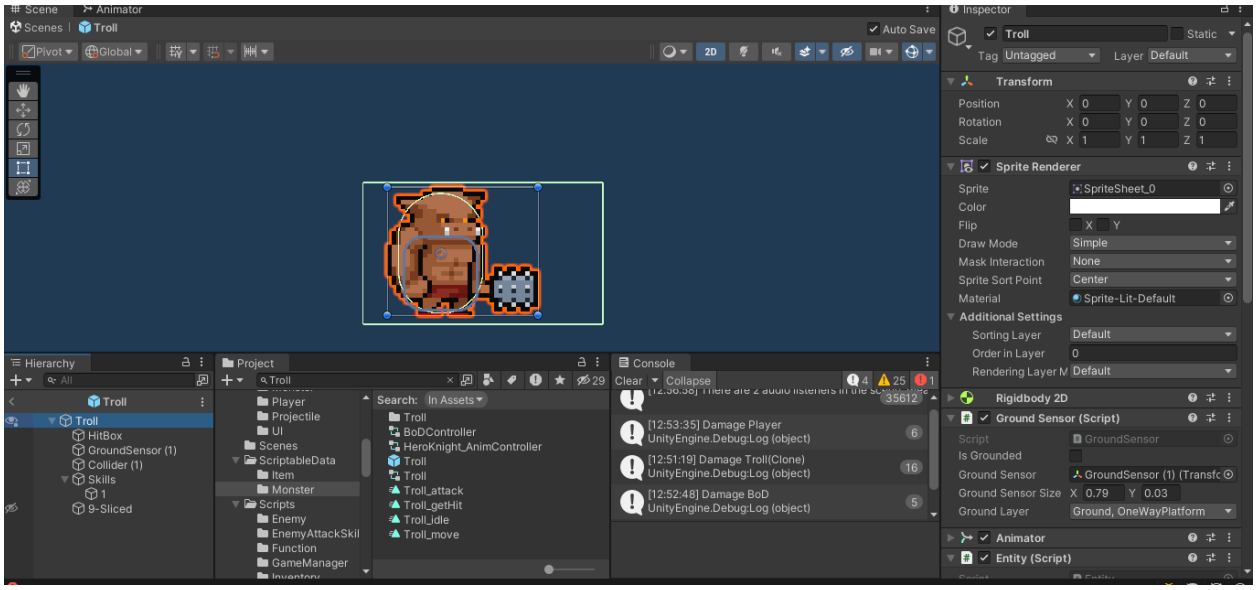
CHƯƠNG 3: PLAYER, QUÁI VẬT VÀ CÁC OBJECTS

I. Player

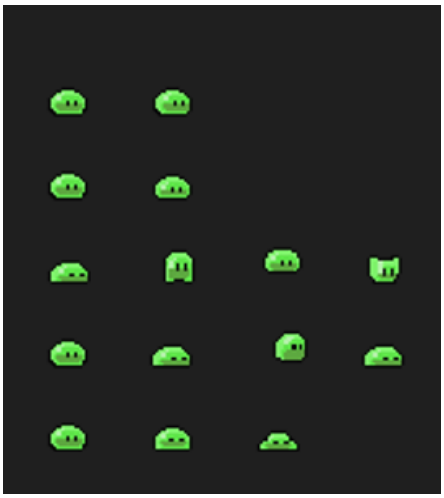


II. Quái vật

Mỗi loại quái vật sẽ được thiết kế và lưu dưới dạng Prefab, chúng sẽ có chỉ số được load từ một ScriptableObject Entity riêng.


Scene
Animator
Scenes
Troll
Auto Save
Pivot
Global
2D
Inspector
Troll
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Sprite Renderer
Sprite
SpriteSheet_0
Color
Flip
Draw Mode
Simple
Mask Interaction
None
Sprite Sort Point
Center
Material
Sprite-Lit-Default
Additional Settings
Sorting Layer
Default
Order in Layer
Rendering Layer M
Hierarchy
Troll
HitBox
GroundSensor (1)
Collider (1)
Skills
1
9-Sliced
Project
Troll
Player
Projectile
UI
Scenes
ScriptableData
Item
Monster
Scripts
Enemy
EnemyAttackSkil
Function
GameManager
Search:
In Assets
Troll
BoDController
HeroKnight_AnimController
Troll
Troll
Troll_attack
Troll_getHit
Troll_idle
Troll_move
Console
Clear
Collapse
[12:53:35] Damage Player
UnityEngine.Debug:Log (object)
[12:51:19] Damage Troll(Clone)
UnityEngine.Debug:Log (object)
[12:52:48] Damage BoD
UnityEngine.Debug:Log (object)
Rigidbody 2D
Ground Sensor (Script)
Script
GroundSensor
Is Grounded
Ground Sensor
GroundSensor (1) (Transfc
Ground Sensor Size
Ground Layer
Ground, OneWayPlatform
Animator
Entity (Script)

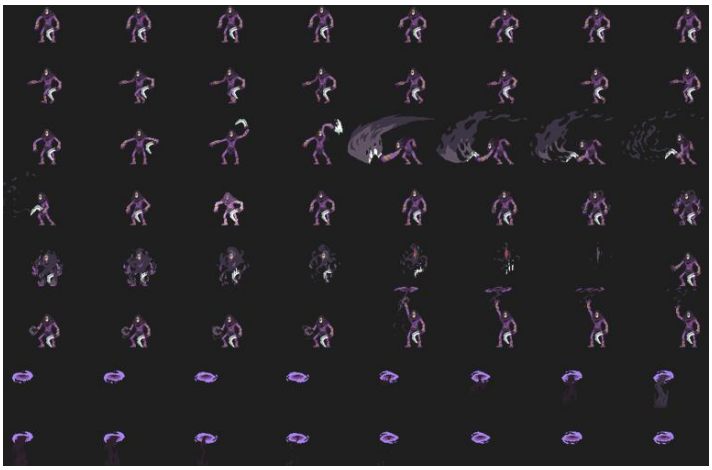
- Slug:



- Troll:

- Wizard:

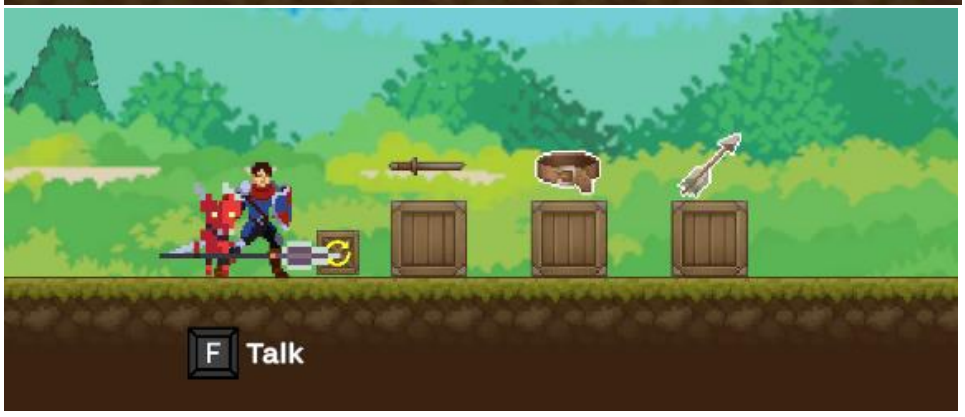
- Boss “BringerOfDeath”:



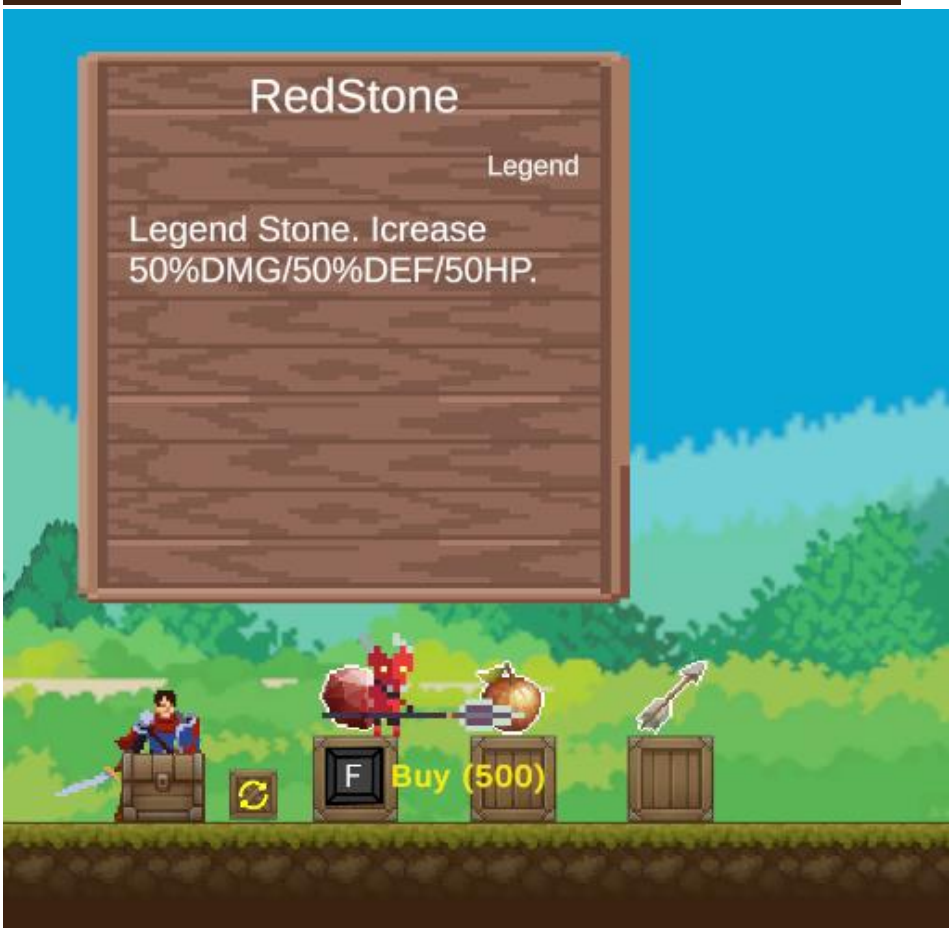
III. NPC

Hiện tại chỉ có 1 NPC duy nhất

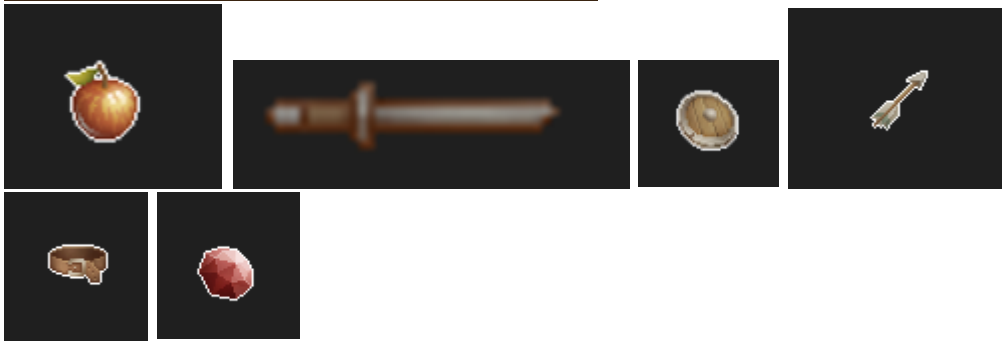
- SaleNPC: Trao đổi bán vật phẩm




F Refresh(100)


RedStone
Legend
Legend Stone. Icrease
50%DMG/50%DEF/50HP.
F Buy (500)

IV. Item

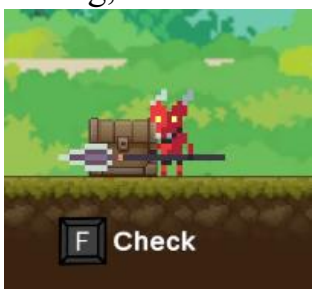


Name	ATK Amount	DEF Amount	HP Amount	Rate
Apple	0	0	30	Common
WoodenSword	10	0	0	Common
WoodenShield	0	20	0	Common
Arrow	30	0	0	Rare
Belt	0	20	30	Rare
RedStone	50	50	50	Legend

V. Các objects khác

- Cổng dịch chuyển:

- Rương, hòm:

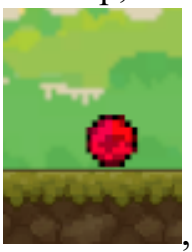




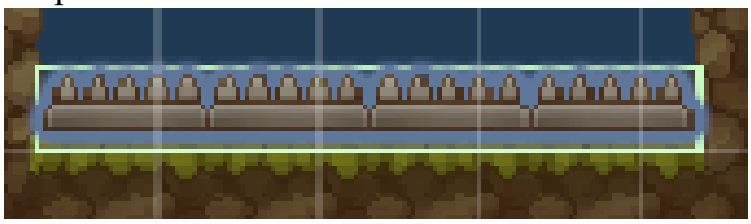
- Projectile:
  - Fireball:

- WizardProjectile: 
- BoDProjectile: 

- HP drop, coin:

 , 

- Trap:



CHƯƠNG 4: THIẾT KẾ TRÒ CHƠI

I. Hệ thống quản lý trò chơi

- GameManager sẽ đảm nhiệm việc quản lý các sự kiện cũng như các trạng thái của trò chơi như Pause(tạm dừng), Mở hành trang, Play…
    - Tại mỗi trạng thái sẽ gọi đến một UI riêng.
- Kích hoạt các event như: Lose: khi HP nhân vật về 0, Win: khi tiêu diệt được Boss
- Camera: Tại mỗi màn chơi sẽ có giới hạn vùng chiều của camera. camera chỉ chiếu ở trong khu vực màn chơi.
    - Ở màn chơi bình thường: Sẽ đi theo nhân vật

- Ở màn chơi boss: Ngay khi player kích hoạt đánh boss, camera sẽ khóa vùng chiếu tại vị trí trung tâm và người chơi cũng không thể đi ra khu vực này khi vẫn đang đánh boss.

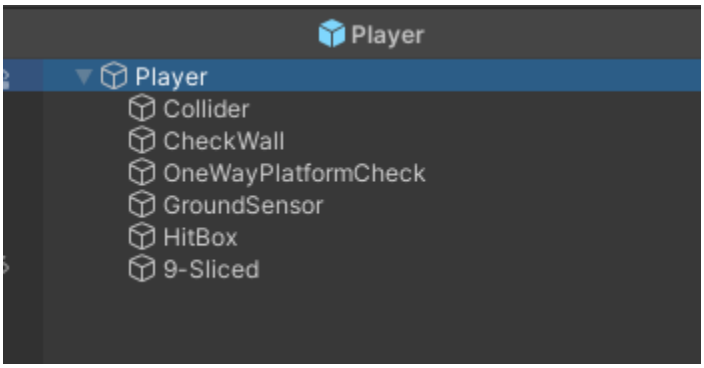
II. Hệ thống nhận điều khiển

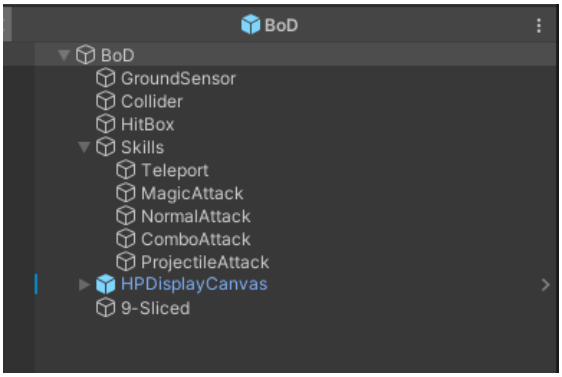
Sử dụng hệ thống Input của Unity để nhận điều khiển từ người chơi.

4 Arrow Key: 

- Nhân vật player:
  - Di chuyển: Right and Left Arrow Key
  - Nhảy: Z
  - Lướt: X
  - Tấn công bình thường: C
  - Kỹ năng: A
  - Pause: Escape
  - Mở hành trang: Tab
  - Tương tác:F
  - Di chuyển xuống dưới các OneWayPlatform: Down Arrow Key
- UI:
  - Di chuyển giữa các nút: 4 Arrow Key
  - Chọn: Enter
  - Quay lại/Resume : Escape
  - Bỏ item ra đất: F

III. Xây dựng nhân vật





(*9-sliced là sprite renderer trên minimap)*

- Player và quái vật đều được lưu dưới dạng Prefab và có các GameObject và điểm chung sau:
  - Collider: để tạo vùng collider của nhân vật
  - RigidBody2D: Áp dụng vật lý với nhân vật
  - GroundSensor: Nhận diện nhân vật có ở mặt đất hay không để triển khai các hành vi
  - HitBox: Vùng gây sát thương của nhân vật. Khi vùng này chạm vào collider sẽ gây sát thương lên nhân vật có collider đó.
- Cơ chế tấn công:
  - Animation trong Unity là mỗi chuỗi các Sprite và ta có thể ghi lại vị trí cũng như các thuộc tính của 1 GameObject nằm trong GameObject có chứa Animation đó.
  - Tại các sprite tấn công trong animation tấn công, ghi lại vị trí và kích thước của HitBox của nhân vật, sau đó có thể ẩn HitBox này đi khi chuyển qua Sprite khác.

IV. Hệ thống hành vi

1. Projectiles

Projectile không được áp dụng vật lý vì vậy collider của nó chính là HitBox.
Có 3 loại projectile:

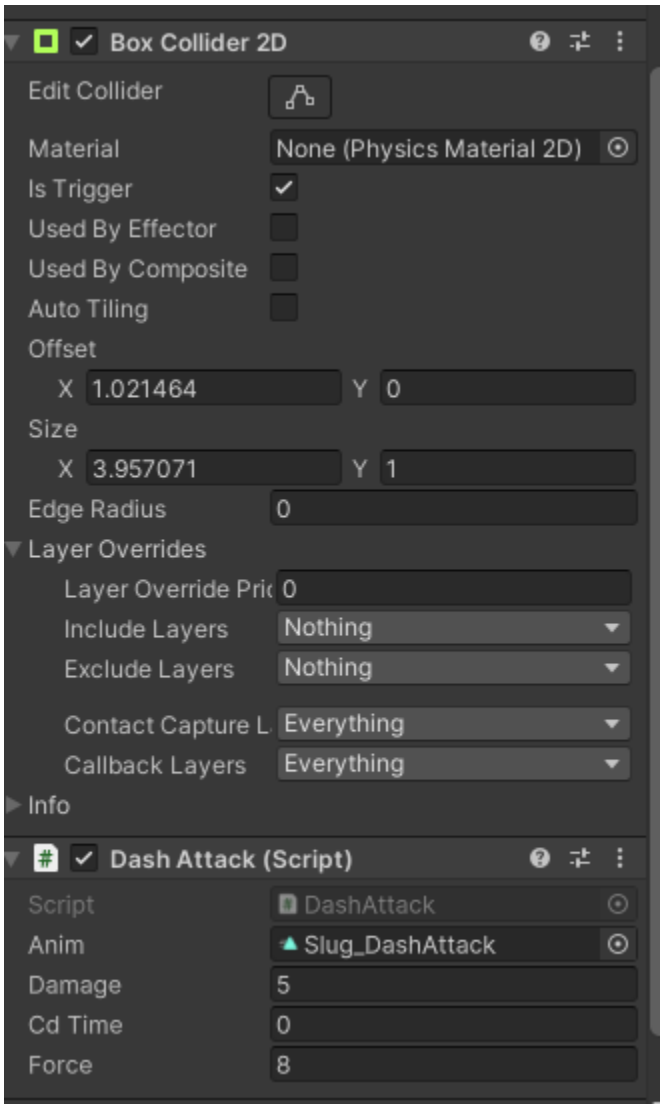
- WizardProjectile:
  Chọn mục tiêu là vị trí của Player khi được sinh. Di chuyển theo hướng từ vị trí gốc đến vị trí đích xa hơn nữa đến một vị trí nhất định sẽ biến mất. Nếu trúng Player sẽ gây sát thương.
- FireBall: Projectile skill của Player. Bay thẳng theo hướng di chuyển của Player. Gây sát thương cho quái trên đường đi.
- BoDProjectile: Xuất hiện bên trên Player và gây sát thương nếu collider của nó trigger với collider của Player.

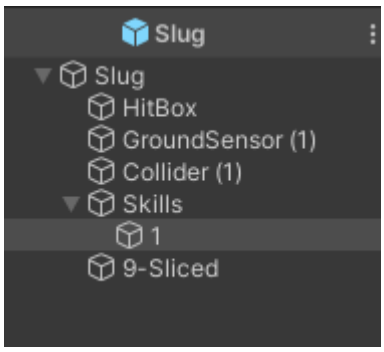
2. Quái vật

Tùy loại quái vật được thiết kế mà có thể di chuyển, truy đuổi Player.

Quái vật bình thường sẽ có 4 hành vi chính:

- Tuần tra: Bao gồm đứng chờ và di chuyển xung quanh khu vực của DetectObject(GameObject có chứa DetectArea.cs).
- Truy đuổi: Có thể có hoặc không. Khi Player di chuyển vào khu vực DetectObject mà có quái vật ở đó, DetectObject sẽ trigger tới quái vật chuyển sang trạng thái truy đuổi Player.
- Bị tấn công: Có thể có hoặc không. Khi bị Player tấn công, quái vật sẽ bị đẩy lùi và không làm gì cả, thời gian của hành vi này thường nhỏ và nếu người chơi dừng tấn công, quái vật có thể tấn công lại ngay lập tức.
- Tấn công: Khi Player trong tầm tấn công của quái vật, quái vật sẽ chuyển sang trạng thái tấn công ngay lập tức.
  Quái vật sẽ bao gồm các GameObject con có chứa lớp AttackSkill bên trong nó. Mỗi GameObject có chứa AttackSkill sẽ có một khu vực kích hoạt tấn công riêng tùy thiết kế mỗi loại quái.





Đối với Boss, chúng sẽ có hành vi riêng và trong màn chơi đánh boss, sẽ có nhiều GameObject tại màn chơi để Boss có thể thực hiện các hành vi khác nhau. Hành vi chính của Boss: Liên tục tấn công với các Skill, sau khi tấn công sẽ có khoảng thời gian chờ để tấn công tiếp.

Hành vi quái:

Tên	Patrol	Truy đuổi	Bị tấn công	Tấn công
Troll	Có	Có	Có	Đánh một đòn về phía trước
Slug	Có	Có	Không	Lướt đến tấn công người chơi
Wizard	Không	Không	Có	Tầm tấn công rộng, có thể coi như một tháp trụ. Tấn công bắn 3 Wizard Projectile về người chơi.

- BoD:
    - Teleport: Dịch chuyển đổi vị trí về 2 vị trí mặc định trên màn chơi sau mỗi lần tấn công.
    - MagicAttack: Tấn công giống như Wizard.
    - BoDMagicAttack: Tấn công bằng BoDProjectile.
    - NormalAttack: Dịch chuyển đến gần người chơi và tấn công như Troll.
    - ComboAttack: Dùng 3 đòn NormalAttack liên tục

3. Player

Hành vi của Player phụ thuộc vào điều khiển của người chơi:

Hành vi	Mô tả
Đứng yên(Idle)	Khi người chơi không điều khiển và Player ở mặt đất.
Nhảy	Nhảy lên không trung.
Nhảy lần 2	Nhảy lần nữa sau khi nhảy lần 1.
Lướt(Dash)	Phóng về phía trước.
Tấn công, Tấn công liên tục	Tấn công về phía trước.
Kỹ năng(SkillAttack)	Player sẽ bắn 1 quả Fireball về phía trước. Sau khi thực hiện sẽ cần chờ 1 khoảng thời gian mới có thể thực hiện tiếp.
Tương tác	Khi đi vào khu vực của vật thể tương tác. Người chơi có thể điều khiển Player tương tác với vật thể

	và thực hiện chức năng của vật thể. Ví dụ: Di chuyển qua màn, Nhặt đồ, Mở rương…
Di chuyển	Di chuyển sang 2 bên trái phải.
Di chuyển xuống dưới khi đứng trên Platform	Nhân vật có thể di chuyển xuống dưới các OneWayPlatform.

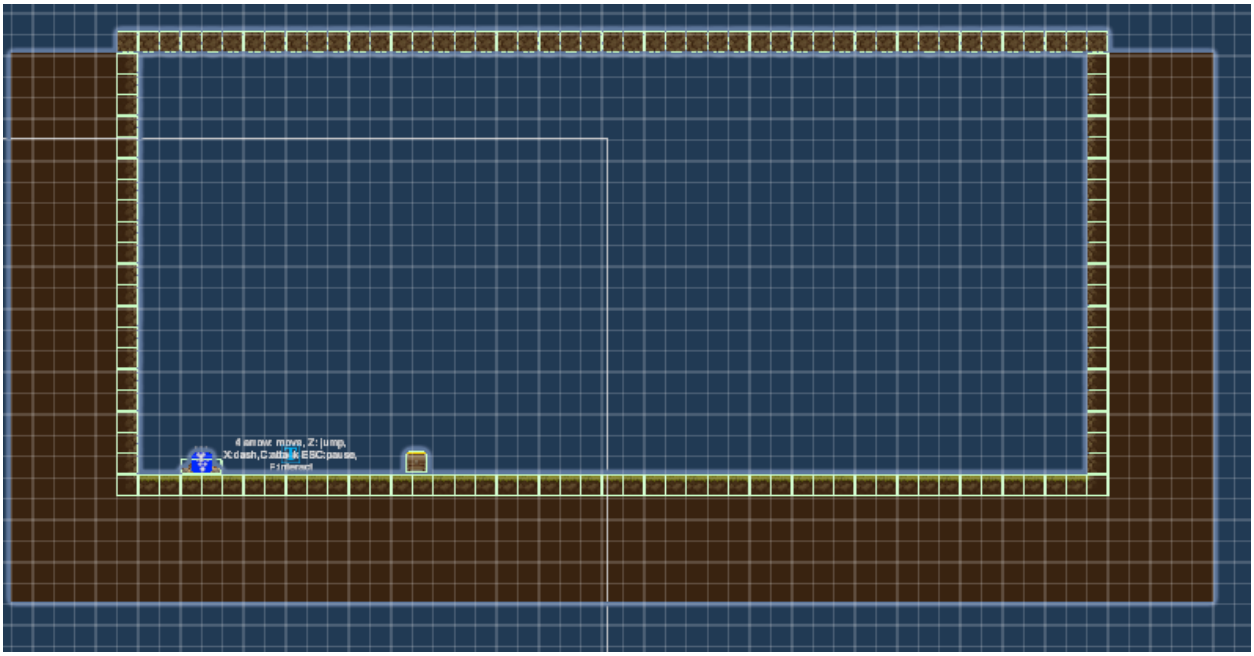
V. Thiết kế màn chơi

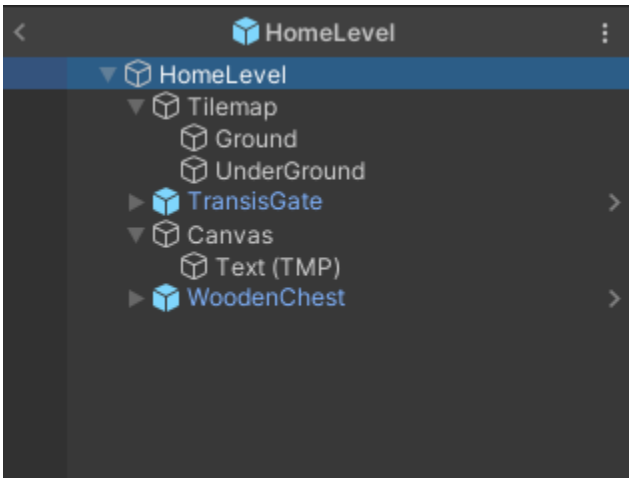
1. Màn chơi

Trên màn chơi sẽ có các collider màu xanh để định nghĩa mặt đất và ngăn không cho các thực thể ra ngoài màn chơi.

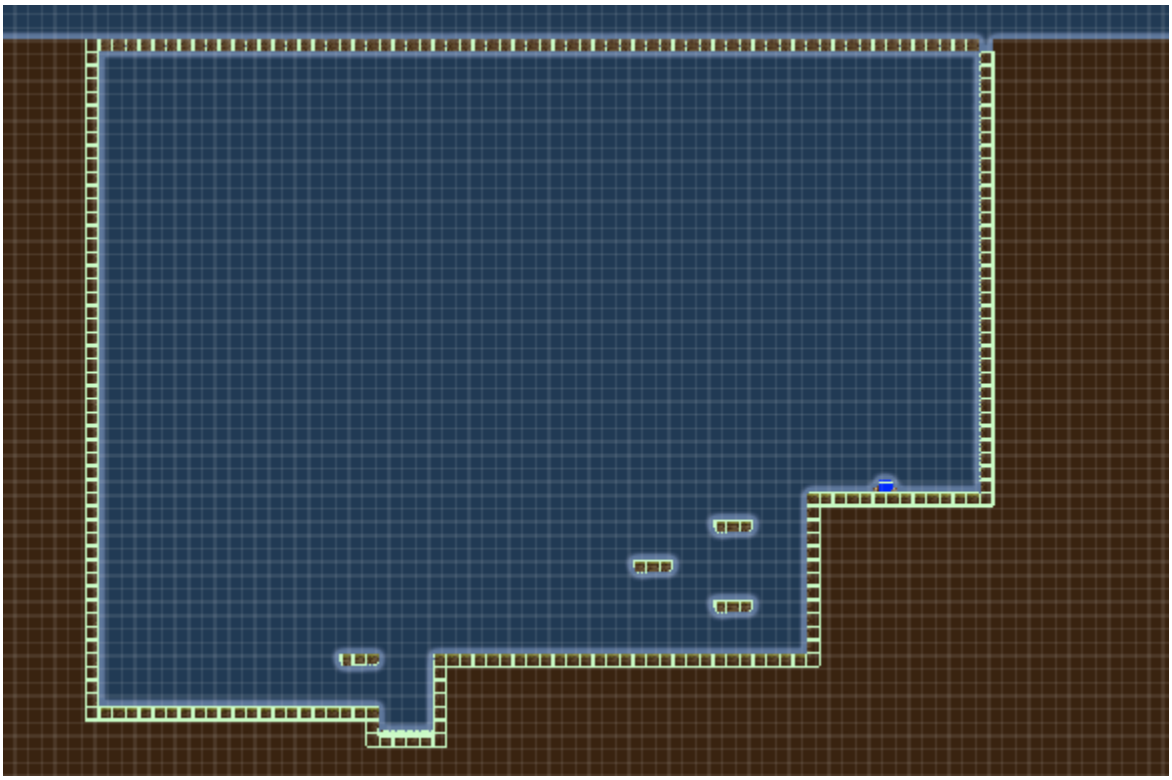
Mỗi màn chơi được lưu dưới dạng các prefab.

- HomeLevel:

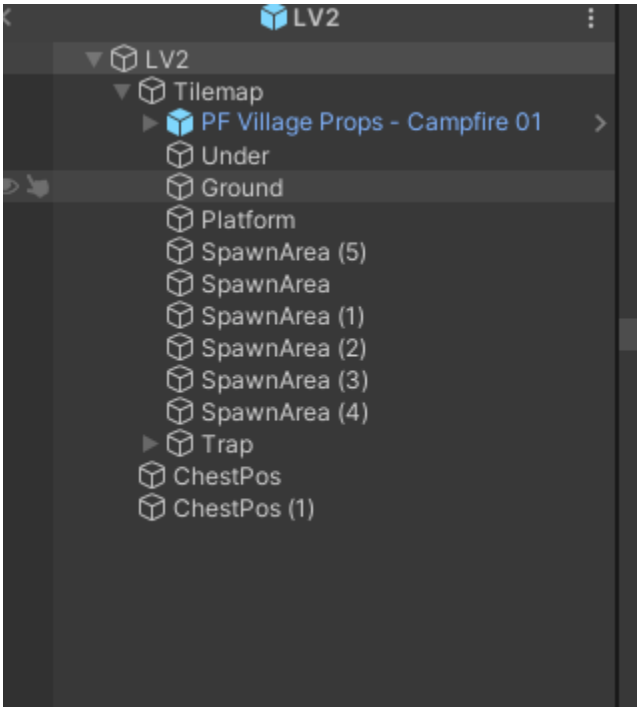


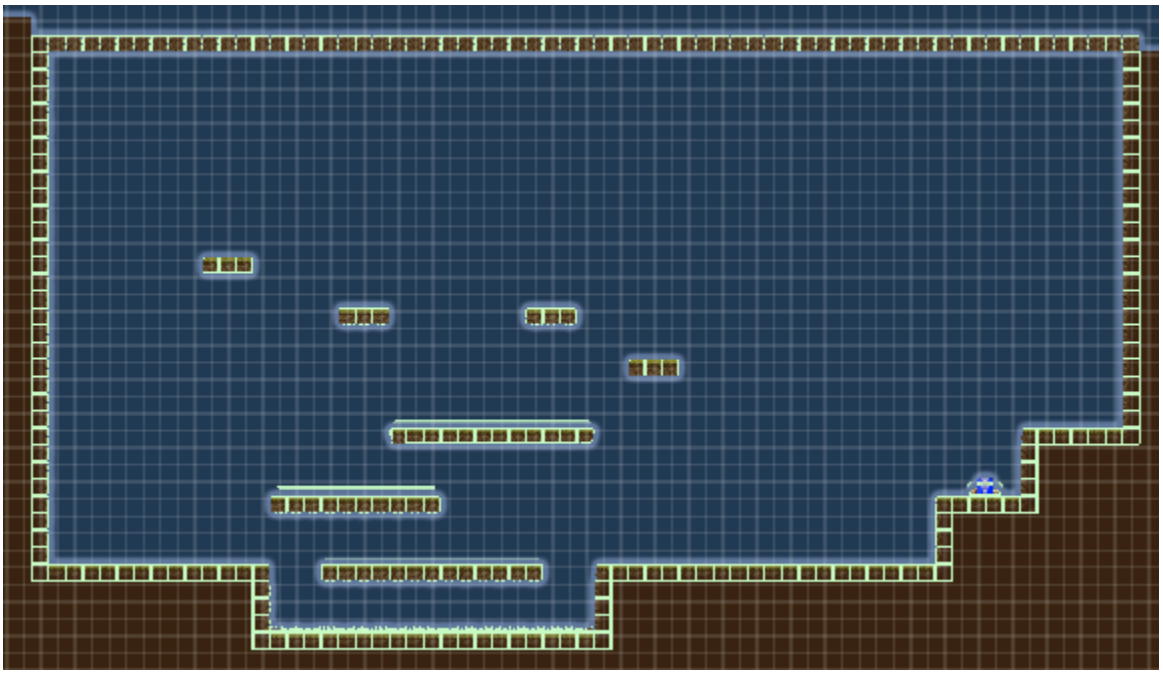

HomeLevel
HomeLevel
Tilemap
Ground
UnderGround
TransisGate
Canvas
Text (TMP)
WoodenChest

- Level:

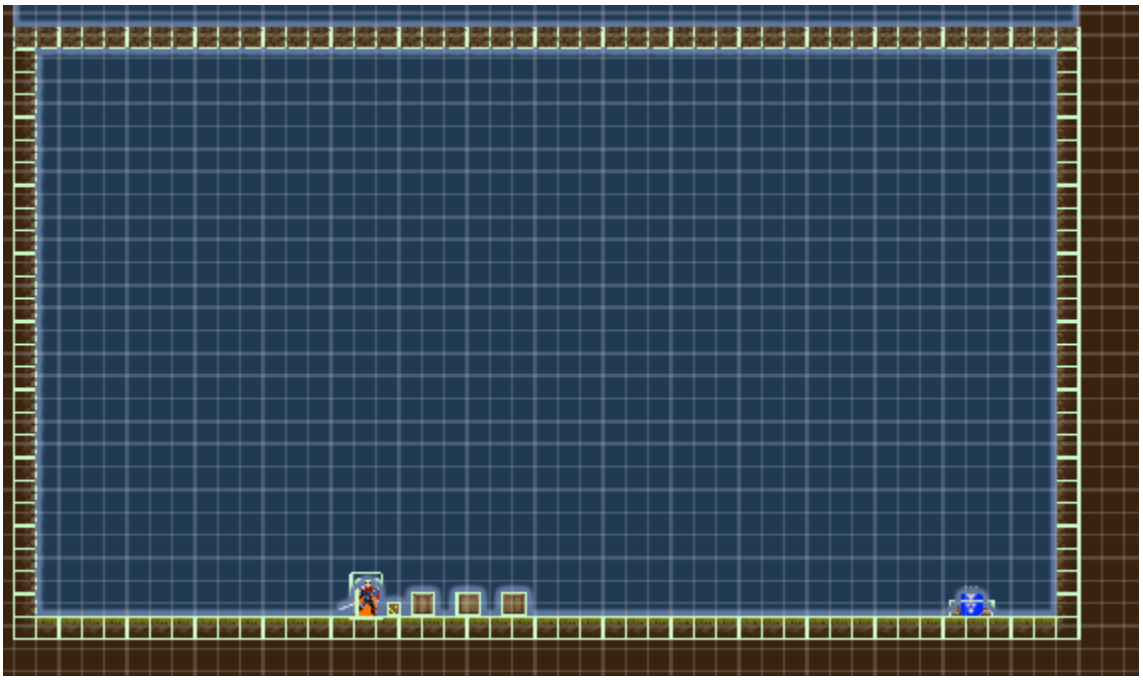



LV1
LV1
Tilemap
PF Village Props - Campfire 01
Under
Platform
Ground
SpawnArea
SpawnArea
SpawnArea (1)
Trap
Spike
Spike (1)
Spike (2)
Spike (3)
ChestPos
ChestPos (1)

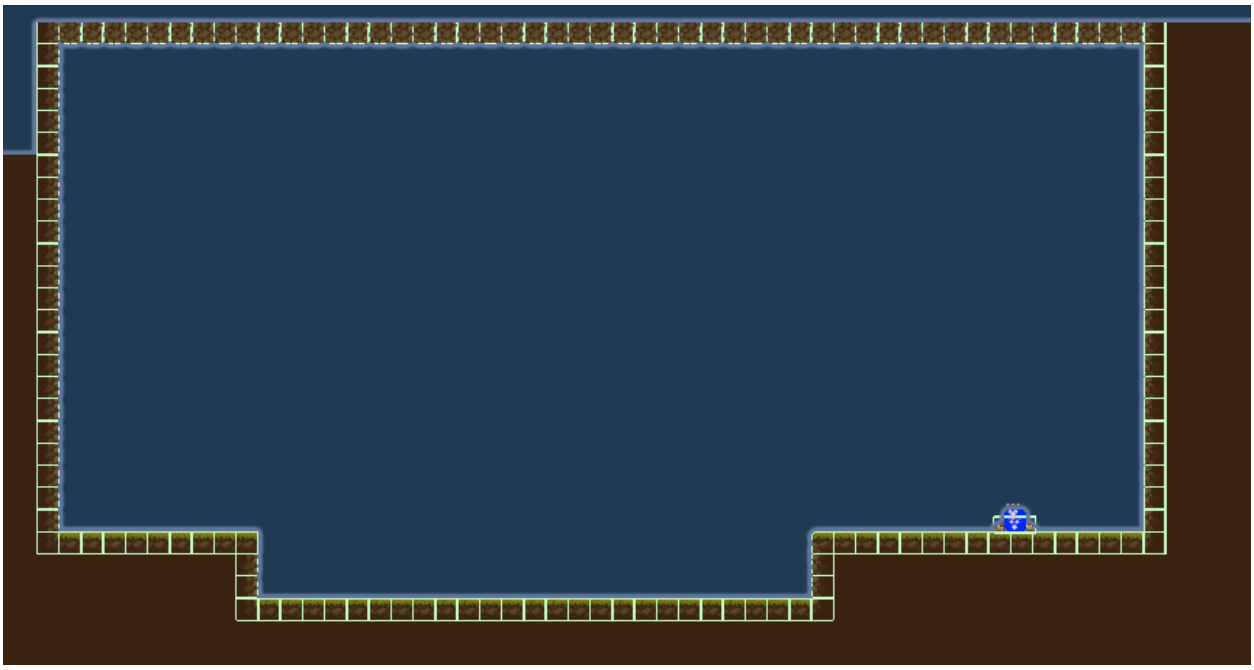

LV2
LV2
Tilemap
PF Village Props - Campfire 01
Under
Ground
Platform
SpawnArea (5)
SpawnArea
SpawnArea (1)
SpawnArea (2)
SpawnArea (3)
SpawnArea (4)
Trap
ChestPos
ChestPos (1)

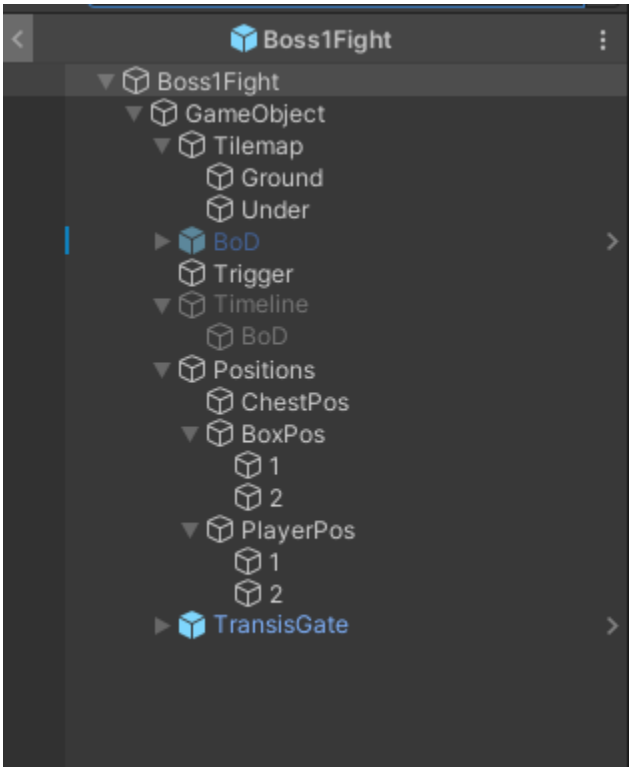


- MarketLevel:



- BossLevel:





2. Quản lý màn chơi
   - Màn chơi đầu tiên mà người chơi vào đó là HomeLevel.

- Mỗi màn chơi sẽ có một cổng dịch chuyển, cổng này chỉ có thể tương tác khi người chơi đã tiêu diệt hết quái. Khi người chơi tương tác sẽ di chuyển đến màn chơi tiếp theo.
- Load màn chơi: Khi người chơi bắt đầu vào màn chơi, hệ thống quản lý màn chơi sẽ lấy ngẫu nhiên một số n màn chơi trong kho dữ liệu màn chơi. Sau đó sẽ load ngẫu nhiên màn chơi trong n (số n do người tạo chọn trước) màn chơi trên. Khi người chơi vượt qua 1 màn chơi, hệ thống sẽ load ngẫu nhiên 1 màn trong (n-1) màn chơi còn lại. Sẽ có mốc màn chơi để dịch chuyển đến khu mua sắm(có các NPC). Cứ như vậy cho đến màn chơi cuối là boss.
- Trên màn chơi có thể sẽ có các trap object: khi người chơi hoặc quái đi vào sẽ mất máu.
- Trên màn chơi cũng có các địa điểm sinh ra rương báu tùy vào người tạo màn chơi.
- Sinh quái vật: tại các vị trí tùy người tạo màn chơi.
- Detect người chơi(hỗ trợ cho hành vi của quái vật):
  Tại bất kỳ vị trí nào trên màn chơi mà người chơi và quái vật có thể di chuyển đến, tại đó sẽ được đặt một object có thể nhận diện người chơi giúp cho hành vi của quái vật. Tùy loại quái vật mà chúng có thể nhận diện được và đuổi người chơi chỉ khi cả 2 cùng đứng trong object này.
- Đối với màn chơi boss: Có thể sẽ có thêm các object hỗ trợ cho hành vi của chúng.

VI. Hệ thống tương tác các object

Khi người chơi di chuyển vào khu vực của vật thể, collider của người chơi sẽ trigger với collider của vật thể, từ đó sẽ hiện lên thông tin của vật thể(tùy loại mà có cách hiện khác nhau), đồng thời người chơi có thể tương tác với vật thể đó để thực hiện chức năng của nó.

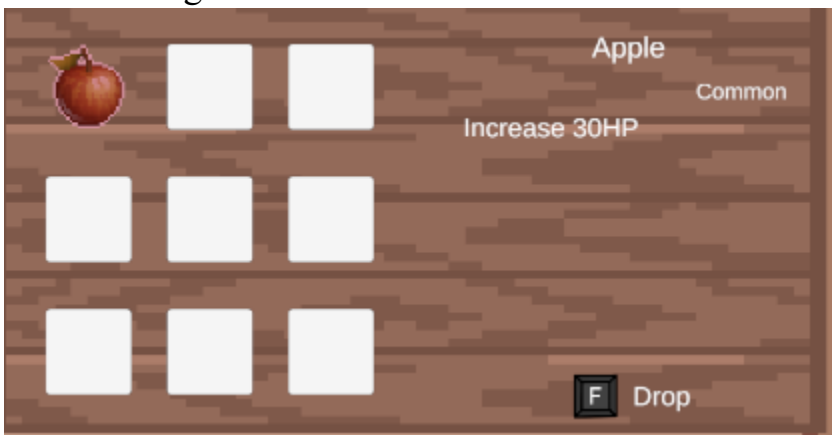
- Chest: mở hòm, tạo ra item.
- Cổng dịch chuyển: Di chuyển đến màn chơi khác.
- NPC: hiển thị Dialog nói chuyện, trao đổi item….
  - NPC bán đồ: NPC này sẽ sinh ra item có giá cả, người chơi chỉ có thể tương tác khi chỉ số coin đủ mua.
- Item: thêm vào inventory.

Ngoài ra còn có các vật thể sẽ tương tác luôn với người chơi(có thể trigger hoặc không cần):

- HP drop: hồi máu cho người chơi.
- Gold coin: cộng vào chỉ số coin.

VII. Hệ thống hành trang

- Khi người chơi tương tác với một item, nếu hành trang chưa đầy(9 items) item sẽ được thêm vào hành trang. Hành trang sẽ có phương thức để cộng các chỉ số của item vào chỉ số của người chơi.
- Người chơi cũng có thể bỏ item ra đất, lúc này hành trang sẽ tính toán lại chỉ số cho người chơi.



VIII. Animation

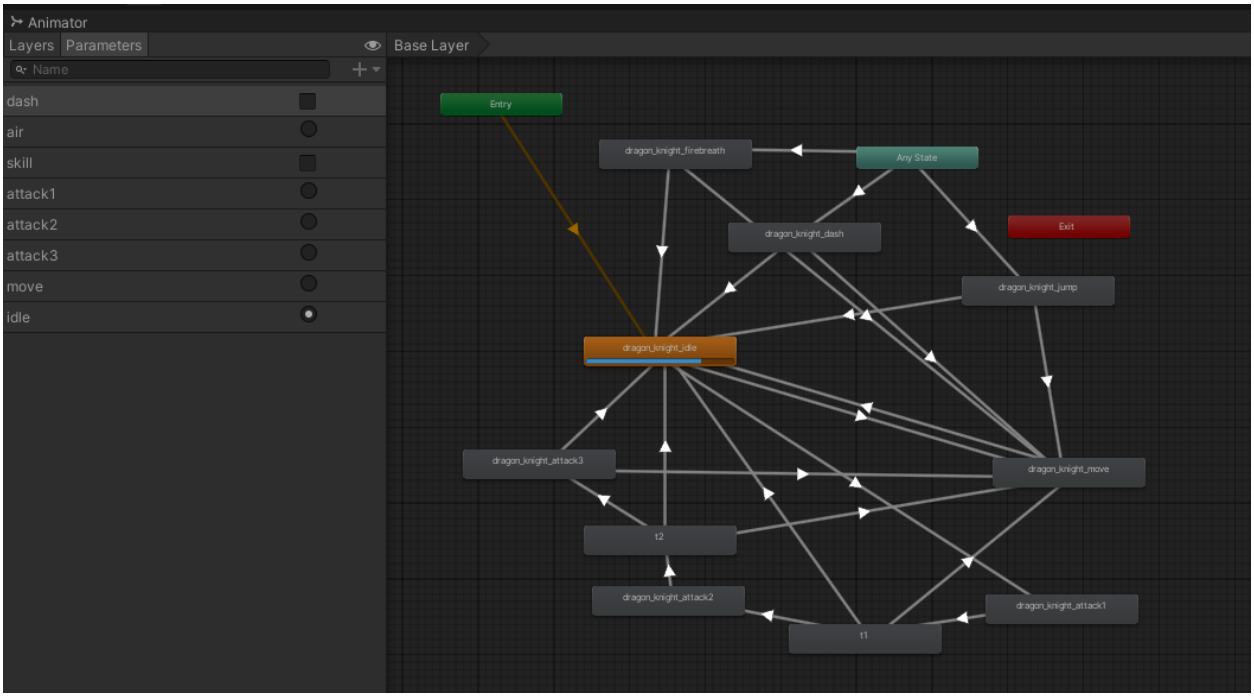
Sử dụng Animator để quản lý và chuyển giao giữa các trạng thái.

1. Player

Animation của Player dựa và điều khiển nhập vì thế việc lập trình có chút phức tạp.

Player có các 9 animation: Idle, Run, Jump, Dash, 3 Normal Attack, Skill.

AnimatorController của Player sẽ có các tham số sau để chuyển giao animation:



2. Quái vật

Quái vật có các state và chỉ thực hiện khi trigger xảy ra.

- Slug: Idle, Run, DashAttack.
- Troll: Idle, Run, NormalAttack, GetHit.
- Wizard: Idle, GetHit, MagicAttack.

Đối với Boss:

- BoD: Idle, Teleport, NormalAttack, ComboAttack, MagicAttack, BoDProjectileAttack.

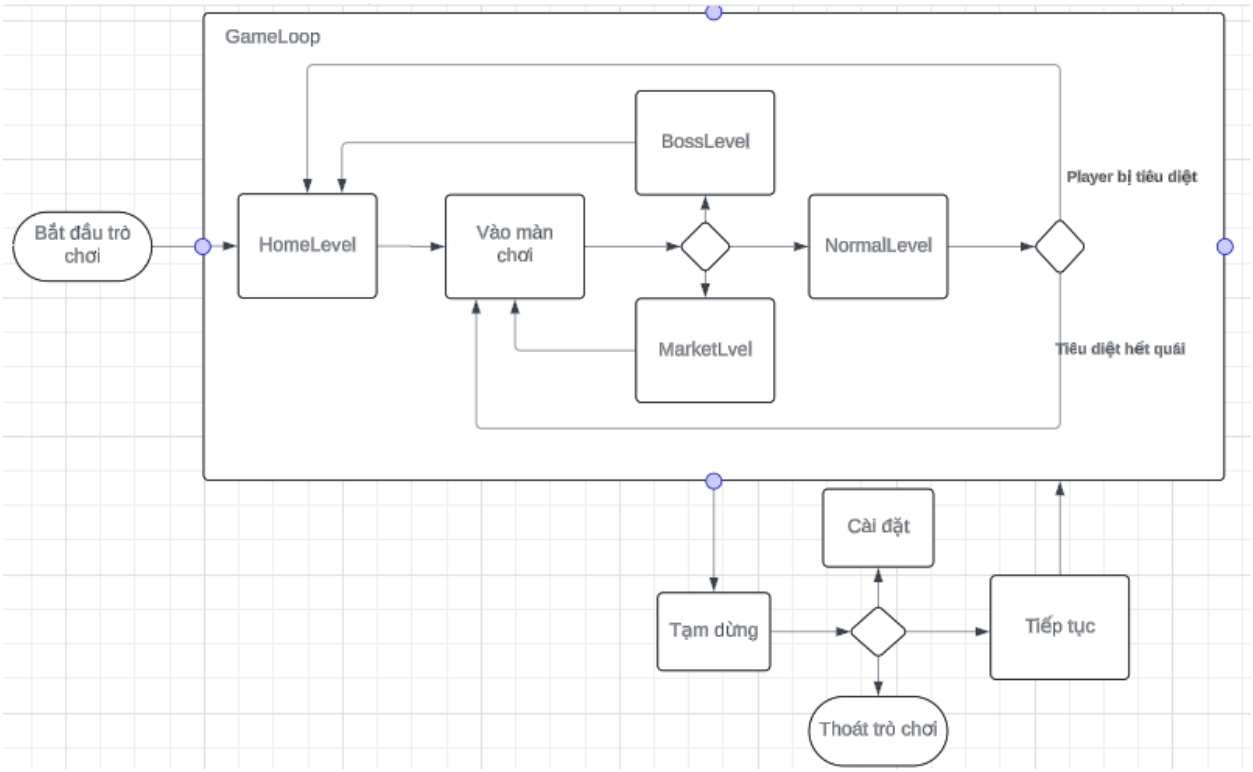
3. NPC

Chỉ có Idle.

4. Projectile

Chỉ có Idle.

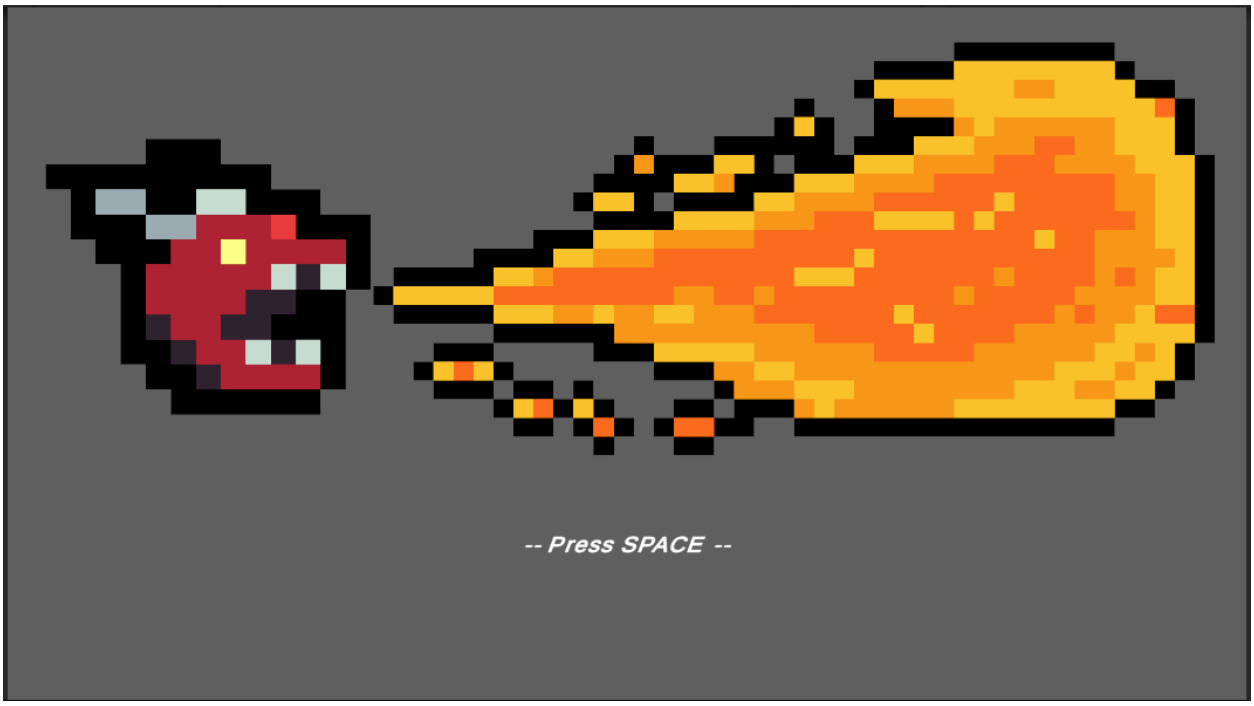
IX. Kịch bản trò chơi



- Khi người chơi khởi động trò chơi, trò chơi sẽ hiện lên logo sau đó tiến vào màn hình chờ.
- Tại màn hình chờ, người chơi sẽ thực hiện thao tác nhất định để tiến vào màn hình chính của trò chơi.
- Sau khi nhận thao tác từ người chơi tại màn hình chờ, hệ thống sẽ tiến hành khởi tạo tài nguyên quản lý và đưa người chơi vào màn hình chính của trò chơi và tải lên HomeLevel.
- Người chơi tiến hành vào màn chơi sau khi tương tác với cổng dịch chuyển.
- Hệ thống quản lý màn chơi sẽ tải lên màn chơi, sinh quái.
- Người chơi tiến hành khám phá và thực hiện CoreLoop như đã nêu ở Chương I:
    - Nếu HP người chơi về 0, trò chơi sẽ hiện thông báo “Bạn đã thua” và đưa người chơi về HomeLevel.
    - Nếu người chơi tiêu diệt được Boss, trò chơi sẽ hiện thông báo “Win” và đưa người chơi về HomeLevel.

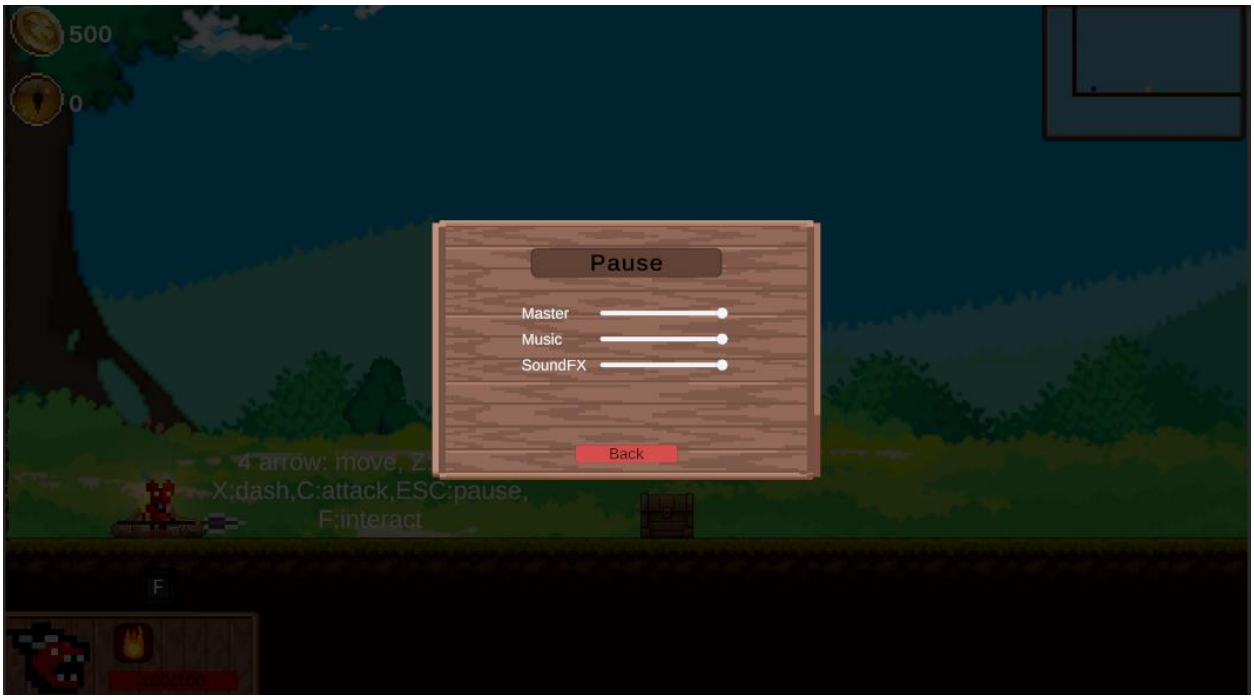
CHƯƠNG 5: USER INTERFACE(UI)

- Màn hình chờ:



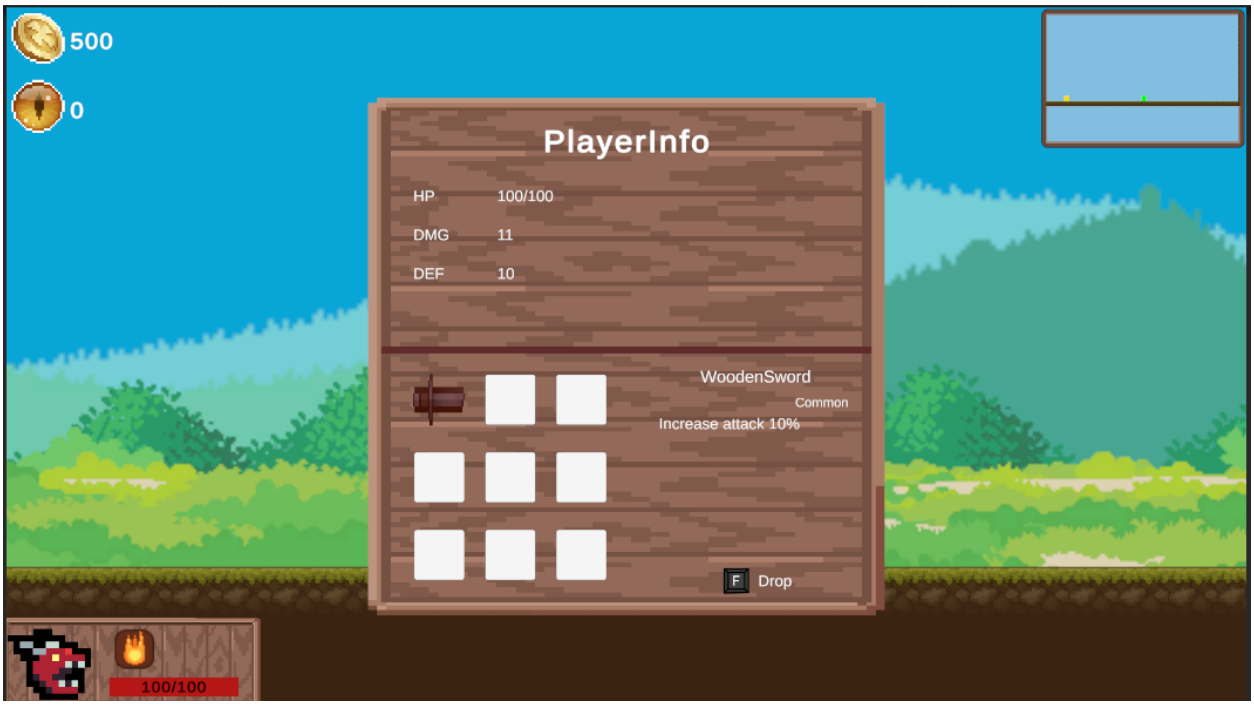
- Màn hình tạm dừng:

- Màn hình cài đặt:



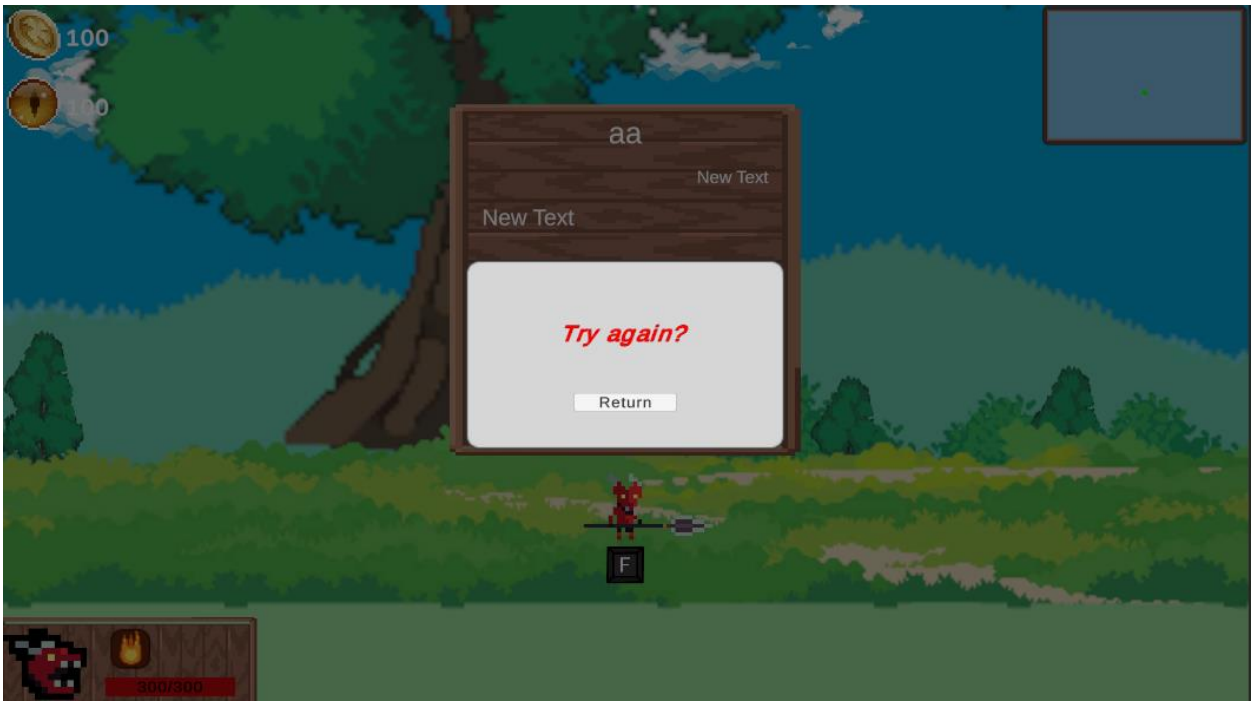
- Màn hình chính:

- Màn hình mở hành trang:



- UI thông tin:

- Màn hình thua cuộc



- Màn hình chiến thắng:

CHƯƠNG 6: Âm thanh

Sử dụng AudioSource để tạo âm thanh trong trò chơi và AudioMixer để quản lý âm lượng âm thanh. Âm lượng có thể được chỉnh thông qua Setting trên UI.

Tên	Loại	Mô tả
Minecraft-Sweden	Background Music	Phát trong quá trình chơi
Mini-Boss	Background Music	Phát khi đánh boss
attack	FX	Phát khi bị tấn công trúng
Minecraft - OpenChest	FX	Phát khi mở rương
jump	FX	Phát khi nhân vật nhảy
dash	FX	Phát khi nhân vật lướt
death	FX	Phát khi quái vật bị tiêu diệt
coin	FX	Phát khi coin drop từ quái vật

CHƯƠNG 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

I. Kết luận

- Kết quả:
  - Thiết kế được một kịch bản trò chơi đơn giản.
  - Hoàn thành được các tính năng cơ bản của trò chơi.
  - Xây dựng được bản thử nghiệm của trò chơi.
  - Nâng cao tư duy lập trình trò chơi và khả năng sử dụng công cụ.
- Hạn chế:
  - Chưa có thiết kế cũng như nguồn tài nguyên đồ họa cho trò chơi
  - Chưa có kịch bản trò chơi hoàn thiện cũng như kịch bản để phát triển thành một sản phẩm có khả năng thương mại.
  - Trò chơi còn thiếu tính năng:
    - Save/Load
    - Cốt truyện
    - Cần bổ sung thêm nội dung: Màn chơi, quái, vật phẩm….

II. Hướng phát triển

Hoàn thiện và phát triển các tính năng:

- Hoàn thiện hệ thống skill cho người chơi.
- Hoàn thiện hệ thống gây sát thương, tính toán chỉ số cho nhân vật, quái vật, vật phẩm, độ khó tăng dần.

- Xây dựng hệ thống tăng cường chỉ số cho người chơi: Là cơ chế tăng sức mạnh vĩnh viễn cho người chơi. Việc tăng cường sức mạnh thông thường trong màn chơi(ví dụ mua đồ để tăng chỉ số) sẽ bị reset mỗi khi người chơi kết thúc 1 lượt chơi. Cơ chế vĩnh viễn này sẽ tăng chỉ số vĩnh viễn, không bị reset trừ khi xóa dữ liệu. Người chơi có thể mở khóa sau một vài lần chơi. Cơ chế này nhằm giúp giảm độ khó cũng như tạo hứng thú cho người chơi.
- Bổ sung màn chơi, loại quái vật, vật phẩm và các loại tấn công của quái.
- Xây dựng cốt truyện.
- Bổ sung các nhân vật, hệ thống nhân vật.
- Cải thiện đồ họa

Reference

[1] https://docs.unity.com/

[2] https://www.linkedin.com/pulse/top-7-design-patterns-every-unity-game-developer-should-charles-hache/