

BỘ CÔNG THƯƠNG  
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC  
NGÀNH KHOA HỌC MÁY TÍNH  
XÂY DỰNG GAME HEROKID SWORD BẰNG UNITY**

**CBHD: ThS. NGUYỄN XUÂN HOÀNG**

**Sinh viên: Lê Minh Trung Bình**

**Mã sinh viên: 2019602194**

**Lớp: KHMT01**

**Khóa: K14**

LÊ MINH TRUNG BÌNH

KHOA HỌC MÁY TÍNH

*Hà Nội - Năm 2023*

## LỜI MỞ ĐẦU

Ngày nay, ngành công nghiệp game đang phát triển mạnh mẽ và là một trong những ngành công nghiệp lớn nhất trên thế giới. Các game thủ trên toàn thế giới đang yêu cầu những trò chơi chất lượng cao, đồng thời, nhu cầu về các thể loại game cũng ngày càng tăng cao.

Hiện nay, có nhiều công cụ phát triển game 2D như Unity, GameMaker, Construct, Phaser, và nhiều công cụ khác giúp cho việc phát triển game 2D trở nên dễ dàng hơn. Các nhà phát triển game 2D có thể tận dụng các công cụ này để tạo ra những trò chơi thú vị, độc đáo và chất lượng cao.

Trong đồ án tốt nghiệp lần này em sẽ xây dựng game "HeroKid Sword" thuộc thể loại 2D pixel top – down một trong những thể loại game làm mưa làm gió trong cộng đồng người chơi game trên thế giới. Với sự chỉ bảo tận tình của các thầy hướng dẫn và kỹ năng em học được, hy vọng rằng game "HeroKid Sword" sẽ được hoàn thiện và phát triển để có thể đến tay của người chơi và mang lại cho họ những trải nghiệm tuyệt vời.

## LỜI CẢM ƠN

Lời đầu tiên cho phép em gửi lời cảm ơn sâu sắc tới các thầy cô trong khoa Công nghệ thông tin - Trường Đại học Công Nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn những kiến thức, những bài học quý báu và bổ ích . Đặc biệt em xin được bày tỏ sự tri ân và xin chân thành cảm ơn thầy giáo Nguyễn Xuân Hoàng người trực tiếp quản lý, hướng dẫn, chỉ bảo em trong suốt quá trình làm đồ án tốt nghiệp, nghiên cứu và hoàn thành được bài tập lớn này. Sau nữa, em xin gửi tình cảm sâu sắc tới gia đình và bạn bè vì đã luôn bên cạnh khuyến khích, động viên, giúp đỡ cả về vật chất lẫn tinh thần trong suốt quy trình học tập để em hoàn thành tốt việc học tập của bản thân.

Trong quá trình nghiên cứu và làm đồ án tốt nghiệp, do năng lực, kiến thức, trình độ bản thân em còn hạn hẹp nên tránh khỏi những thiếu sót và em mong mọi nhận được sự thông cảm và những góp ý từ quý thầy cô.

Em xin chân thành cảm ơn!

## MỤC LỤC

<b>LỜI MỞ ĐẦU .....</b>	<b>2</b>
<b>LỜI CẢM ƠN .....</b>	<b>3</b>
<b>DANH MỤC HÌNH ẢNH .....</b>	<b>6</b>
<b>PHẦN MỞ ĐẦU .....</b>	<b>8</b>
<b>CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN .....</b>	<b>10</b>
1.1. Giới thiệu tổng quan về game, phân loại game .....	10
1.1.1. Tổng quan về game .....	10
1.1.2. Phân loại game .....	10
1.2. Tổng quan về Unity, Game Engine .....	12
1.2.1. Tổng quan về Unity .....	12
1.2.2. Tổng quan về Game Engine .....	14
1.3. Giới thiệu công nghệ, ngôn ngữ .....	15
1.3.1. Giới thiệu Unity Engine .....	15
1.3.2. Giới thiệu về ngôn ngữ .....	18
<b>CHƯƠNG 2. THIẾT KẾ Ý TƯỞNG GAME .....</b>	<b>21</b>
2.1. Giới thiệu: .....	21
2.2. Thể loại game 2D pixel top-down: .....	21
2.3. Tóm tắt game: .....	22
2.4. Khách hàng mục tiêu: .....	23
2.5. Điểm mạnh của game: .....	24
2.6. Phong cách nghệ thuật game: .....	24
2.7. Thiết bị trải nghiệm game: .....	26
<b>CHƯƠNG 3. XÂY DỰNG PHÁT TRIỂN GAME .....</b>	<b>27</b>
3.1. Thiết kế hệ thống .....	27
3.1.1. Biểu đồ Use Case .....	27
3.1.2. Mô tả các Use Case .....	27
3.1.3. Phân tích các Use Case .....	29
3.2. Thiết kế kịch bản game .....	33
3.2.1. Cách chơi chính: .....	33
3.2.2. Cốt truyện của game: .....	34

3.2.3. Các phần tử của game: .....	34
3.2.4. Các cơ chế của game:.....	38
3.3. Thiết kế giao diện.....	38
3.3.1. Biểu đồ (Flowchart) .....	38
3.3.2. Mô tả .....	39
3.3.3. Giao diện các màn hình .....	40
3.4. Thiết kế âm thanh.....	42
<b>CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ .....</b>	<b>43</b>
4.1. Các kỹ thuật thực hiện.....	43
4.1.1. Kỹ thuật của các đối tượng.....	43
4.1.2. Tạo Animation, Animator cho Player và Enemy .....	55
4.1.3. Thiết kế TileMap, Canvas .....	75
4.1.4. Kỹ thuật xử lý va chạm vật lý (Collision) .....	80
4.2. Cài đặt chương trình và kết quả .....	85
4.2.1. Cài đặt Unity Hub.....	85
4.2.2. Tạo Unity ID .....	85
4.2.3. Import và chạy project .....	86
4.2.4. Kết quả.....	87
<b>KẾT LUẬN .....</b>	<b>90</b>
Kết quả đạt được .....	90
Hạn chế của đề tài .....	90
Hướng phát triển .....	90
<b>TÀI LIỆU THAM KHẢO .....</b>	<b>91</b>

## **DANH MỤC HÌNH ẢNH**

Hình 1: Thiết bị chơi Game Arcade.....	11
Hình 2: Minh họa game Engine. ....	14
Hình 3: Giao diện làm việc Unity Editor. ....	16
Hình 4: Hình ảnh minh họa về game 2D pixel top – down. ....	22
Hình 5: Game Enter the Gungeon. ....	23
Hình 6: Hình ảnh thiết kế game trên Unity. ....	26
Hình 7: Biểu đồ Use Case của game. ....	27
Hình 8: Màn hình hướng dẫn chơi game.....	33
Hình 9: Nhân vật HeroKid. ....	34
Hình 10: Slime.....	35
Hình 11: Nepenthe. ....	35
Hình 12: Các loại Skeleton. ....	35
Hình 13: Objects.....	36
Hình 14: Tilemap Jungle .....	37
Hình 15: Tilemap Overworld.....	37
Hình 16: Tilemap house .....	38
Hình 17: Biểu đồ - Flowchart .....	39
Hình 18: Tấn công Slime .....	40
Hình 19: Tấn công Nepenthe .....	41
Hình 20: Tấn công Skeleton .....	41
Hình 22: Player Controller .....	44
Hình 23: Health Manager.....	47
Hình 24: Slime Script .....	49
Hình 25: Enemy Health Manager.....	52
Hình 26: Các thông số có thể thay đổi ứng với các Enemy khác nhau. ....	55
Hình 27: Màn hình thiết kế hoạt ảnh của Player .....	55
Hình 28: Blend Tree của Player_idle.....	57
Hình 29: Thông số Blend Tree của Player_idle. ....	58
Hình 30: Blend Tree của Attack.....	60
Hình 31: Thông số Blend Tree của Attack. ....	61

Hình 32: Blend Tree của Player_move .....	62
Hình 33: Thông số Blend Tree của Player_move .....	63
Hình 34: Mô hình hoạt ảnh của Player.....	64
Hình 35: Màn hình thiết kế hoạt ảnh của Slime .....	65
Hình 36: Mô hình hoạt ảnh của Slime.....	66
Hình 37: Màn hình thiết kế hoạt ảnh của Nepenthe.....	67
Hình 38: Cây và thông số Blend Tree của Nep_move .....	68
Hình 39: Mô hình hoạt ảnh của Nepenthe .....	69
Hình 40: Màn hình thiết kế hoạt ảnh của Skeletion Red.....	70
Hình 41: Cây và thông số Blend Tree của skeR_move .....	71
Hình 42: Mô hình hoạt ảnh của Skeletion Red.....	72
Hình 43: Màn hình thiết kế hoạt ảnh của Skeletion Shield.....	73
Hình 44: Cây và thông số Blend Tree của ske_move .....	74
Hình 45: Mô hình hoạt ảnh của Skeletion Shield .....	75
Hình 46: Màn hình thiết kế TileMap .....	76
Hình 47: Thông số TileMap ground .....	77
Hình 48: Thông số của TileMap pathways .....	77
Hình 49: Thông số của TileMap solidObjects.....	78
Hình 50: Màn hình thiết kế Canvas .....	79
Hình 51: Thông số thuộc tính tấn công theo hai hướng trái phải.....	80
Hình 52: Thông số thuộc tính tấn công phía trên. ....	81
Hình 53: Thông số thuộc tính tấn công phía dưới. ....	82
Hình 54: Thông số âm thanh ứng với các HitBox. ....	83
Hình 55: Thông shuộc tính vật lý của nhân vật.....	84
Hình 56: Các thuộc tính đi kèm khác .....	85
Hình 57: Màn hình khởi chạy UnityHub.....	86
Hình 58: Màn hình sau khi import và mở Game project trên Unity .....	87
Hình 59: Khu rừng .....	87
Hình 60: Làng .....	88
Hình 61: Hầm ngục .....	89

## **PHẦN MỞ ĐẦU**

### **1. Tên đề tài:**

Xây dựng game HeroKid Sword bằng Unity.

### **2. Lý do chọn đề tài:**

- Ngành công nghiệp game đang phát triển mạnh mẽ và mang lại doanh thu lớn. Xây dựng một trò chơi thành công có thể mang lại lợi nhuận đáng kể cho các nhà phát triển. Với người chơi có thể thỏa sức trải nghiệm những hoạt động mà họ yêu thích, thể hiện khả năng của mình và tương tác với người chơi khác trên toàn thế giới.
- Ngoài ra xây dựng game có thể giúp nâng cao kỹ năng lập trình của những người tham gia. Việc phát triển game thể thao yêu cầu sự phối hợp giữa nhiều kỹ năng như lập trình, thiết kế đồ họa, âm thanh, đồ họa 2D, 3D, và có thể giúp em phát triển các kỹ năng mới một cách hiệu quả.

### **3. Nội dung nghiên cứu:**

Game được phát triển trên Unity Game tập trung chủ yếu vào các kỹ thuật, cách dùng các assets có sẵn để phát triển game.

#### **a) Kiến thức:**

- Cung cấp cho học viên kỹ năng lập trình C# và lập trình Game với Unity Framework.
- Cung cấp các khái niệm cơ bản trong Game Unity, quy trình tạo Game.
- Animation và điều khiển hành động nhân vật, xử lý va chạm.
- Giúp học viên biết đưa game lên Apple Store, CH Play, kiếm tiền từ sản phẩm tạo ra.
- Chia sẻ những kinh nghiệm trong thiết kế và lập trình game di động từ Giảng viên.

#### **b) Kỹ năng:**

- Học được cách xử lý các va chạm vật lý giữa các thành phần đối tượng có trong game.
- Biết các logic các thực thể và ứng dụng thực tế vào game sao cho có thể chân thực nhất.
- Sáng tạo trong khâu thiết kế về mặt thẩm mĩ.
- Hoạt ảnh, âm thanh cài đặt trong trò chơi.

#### **4. Thời gian nghiên cứu:**

Thời gian thực hiện: từ 06/03/2023 đến 06/05/2023

#### **5. Kết quả mong muốn:**

- Hoàn thành được game mình đã đề ra.
- Ứng dụng công nghệ Unity xử lý các bài toán xây dựng lên hệ thống trò chơi hiệu quả. Sử dụng ngôn ngữ C# lập trình lên ứng dụng.
- Xây dựng được trò chơi có tính giải trí cao và hiệu quả giúp người chơi thoải mái sau những ngày làm việc mệt nhọc.
- Chức năng đơn giản dễ sử dụng phù hợp với mọi lứa tuổi.

#### **6. Cấu trúc báo cáo:**

Ngoài các phần khác như mục lục, lời mở đầu, cảm ơn, kết luận,... bản báo cáo đồ án tốt nghiệp sẽ gồm 4 chương :

1. Giới thiệu tổng quan
2. Thiết kế ý tưởng game
3. Xây dựng phát triển game
4. Cài đặt chương trình và kết quả

# CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN

## 1.1. Giới thiệu tổng quan về game, phân loại game

### 1.1.1. Tổng quan về game

Một trò chơi video hay video game là một trò chơi điện tử liên quan đến sự tương tác với giao diện người dùng hoặc thiết bị đầu vào – chẳng hạn như cần điều khiển, bộ điều khiển, bàn phím hoặc thiết bị cảm biến chuyển động - để tạo phản hồi trực quan. Phản hồi này xuất hiện trên thiết bị hiển thị video, chẳng hạn như TV, màn hình, màn hình cảm ứng hoặc tai nghe thực tế ảo. Trò chơi video thường được tăng cường với phản hồi âm thanh qua loa hoặc tai nghe, và đôi khi với các loại phản hồi khác, bao gồm cả công nghệ xúc giác.

Trò chơi video được xác định dựa trên nền tảng của chúng, bao gồm trò chơi arcade, trò chơi trên máy console và trò chơi trên máy tính cá nhân (PC). Gần đây hơn, ngành công nghiệp này đã mở rộng sang lĩnh vực trò chơi di động thông qua điện thoại thông minh và máy tính bảng, hệ thống thực tế ảo và thực tế tăng cường cũng như điều khiển từ xa trên đám mây. Trò chơi video được phân thành nhiều thể loại dựa trên kiểu chơi và mục đích của chúng.

### 1.1.2. Phân loại game

Thường được phân loại như sau:

#### a. Game Arcade

Arcade game hay trò chơi arcade hoặc coin-op game, đây là một thiết bị giải trí hoạt động trên cơ sở người chơi phải bỏ một đồng tiền xu vào mới có thể trải nghiệm các trò chơi trong máy như game video, trò chơi cơ điện, máy pinball, game đổi thưởng, ...Mặt khác địa điểm có thể trải nghiệm Arcade Game chủ yếu ở các nơi công cộng như quán bar, nhà hàng và khu trung tâm thương mại và giải trí.



*Hình 1: Thiết bị chơi Game Arcade.*

Đặc biệt thời kỳ hoàng kim của Arcade Game là ở những năm 70, 80 của thế kỷ trước khi mà thu hút đông đảo người chơi tham gia. Cho đến đầu những thập niên 90 thì sức hút của trò chơi cũng giảm dần. Ở thời điểm hiện tại, với sự phát triển của khoa học công nghệ đã thúc đẩy mọi lĩnh vực đều được hiện đại hóa, trong đó game cũng không phải ngoại lệ.

Do vậy, các trò chơi Arcade Game giờ đây đã được thiết lập để tạo thành Arcade Game online có thể trải nghiệm trên nhiều thiết bị và trở thành trò chơi hấp dẫn trên các thiết bị này là: Arcade game PC, Arcade game iOS, Arcade game Android.

#### b. Game trò chơi trên máy Console

Là một thiết bị máy tính được sử dụng để xuất các tín hiệu về video và hình ảnh của một trò chơi điện tử lên một màn hình hiển thị. Cùng khả năng kết nối đa dạng các thiết bị điều khiển, game console được đánh giá là hệ máy đem lại sự thoải mái nhất trong quá trình trải nghiệm các tựa game nhiều người chơi.

Những sản phẩm video game console đến từ nhiều thương hiệu khác nhau trên thế giới sẽ được nhóm lại dựa trên những điểm chung về tính năng và công nghệ mà nhà phát triển sử dụng, được gọi là một “thế hệ game console”. Thông thường, một thế hệ sẽ chiếm thế “độc tôn” trong khoảng từ 5 đến 7 năm trước khi có sự xuất hiện của một thế hệ tiếp theo.

#### c. Game trên nền tảng máy tính cá nhân (PC Game)

Một trò chơi máy tính cá nhân (PC game) là một trò chơi video được chơi trên một máy tính cá nhân chứ không phải trên một giao diện điều khiển. Trò chơi được điều khiển bằng các thiết bị PC đầu vào như bàn phím, chuột, phím điều khiển vv game PC có thể chơi có hoặc không có kết nối Internet, và đã được đưa ra từ sự ra đời của máy tính cá nhân. Một số lượng lớn các trò chơi có sẵn cho nền tảng PC.

Hầu hết các trò chơi máy tính cá nhân có phần mềm máy tính và phần cứng yêu cầu cụ thể. Để chơi các trò chơi mới nhất, trong hầu hết trường hợp, đồ họa của máy tính thẻ, card âm thanh, xử lý, cung cấp điện và thậm chí cả hệ điều hành có thể cần phải được nâng cấp lên chi tiết kỹ thuật mới nhất.

Đối với người dùng mới, nó cũng giúp để hiểu thuật ngữ máy tính và thuật ngữ được sử dụng trong game máy tính. Với sự ra đời của Internet, máy tính dựa trên trò chơi trực tuyến cũng đã trở nên có sẵn, cho phép nhiều người chơi để chơi với nhau hoặc với nhau.

## 1.2. Tổng quan về Unity, Game Engine

### 1.2.1. Tổng quan về Unity

Unity được biết đến như là một công cụ phát triển trò chơi đa nền tảng, nó được phát triển bởi Unity Technologies. Mục đích sử dụng chủ yếu là để phát triển trò chơi điện tử và mô phỏng cho máy tính, thiết bị di động, bảng điều khiển, ... Nhờ vào tính năng đa nền tảng, Unity là cái tên phổ biến với cả các

nha phat trien game tu do cung nhu trong các studio game. Nó đc dùng nhằm tạo ra những trò chơi như Hearthstone, Cuphead, Pokemon Go, Rimworld cùng vô vàn trò chơi khác nữa.

Lập trình Unity 2D, 3D đc lập trình dựa trên 3 ngôn ngữ chính là C#, UnityScript và Boo. Trong đó, C# là ngôn ngữ chính mà rất nhiều lập trình viên Unity sử dụng ở thời điểm hiện tại. Xuất hiện cách đây khá lâu từ năm 2005, Unity đã có một lượng lớn người dùng cũng như sở hữu một thư viện tài nguyên khổng lồ. Không chỉ có tài liệu tuyệt vời, mà Unity còn có vô vàn video cùng những hướng dẫn trực tuyến đáng ngạc nhiên dành cho người dùng.

Chính vì lý do này, Unity trở thành sự lựa chọn vô cùng sáng suốt cho người mới bước đầu tiếp cận với các công cụ game. Có mặt trong danh sách những công cụ trò chơi điện tử, Unity giữ vai trò tựa như một cổng thông tin tài nguyên và kiến thức, được xây dựng, phát triển chỉ dựa trên cộng đồng rộng lớn của riêng họ.

Unity hỗ trợ mạnh mẽ rất nhiều tính năng, nổi bật là:

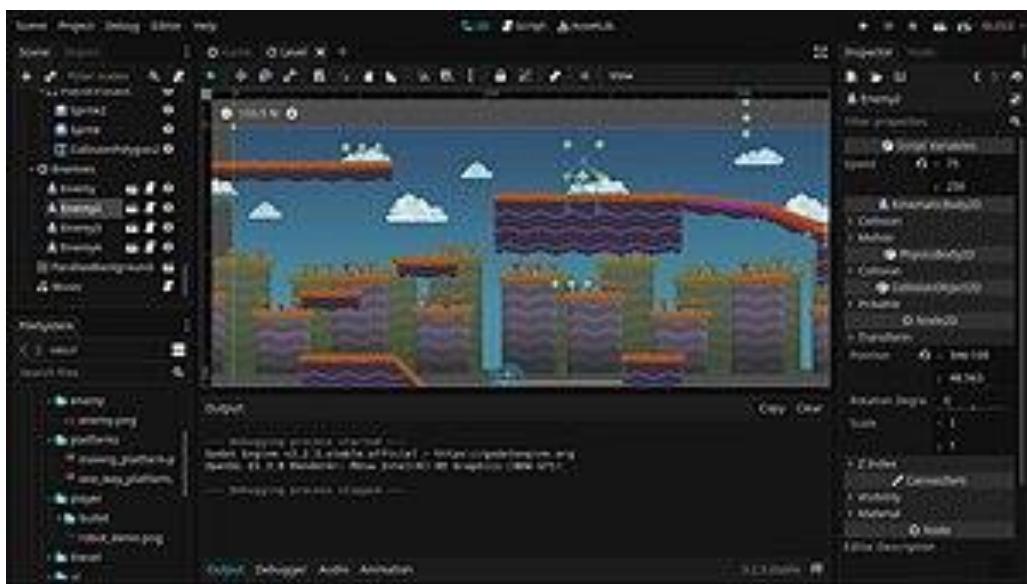
- Unity tạo giao diện UI của game như textbox, drop bar...
- Hiển thị các mô hình 2D, 3D – hệ thống vật lý 3D, 2D.
- Networking: hỗ trợ tạo game cho phép nhiều người chơi tham gia cùng một thời điểm.
- Hỗ trợ nền tảng đặc biệt mới như Virtual reality (VR) – thực tế ảo và Augmented reality (AR) – thực tế tăng cường.
- AI – hỗ trợ bot trong màn hình game và package tạo nên bot trong game.
- Hỗ trợ hiển thị ngôn ngữ bằng nhiều font chữ đặc biệt.

Trong tất cả số tính năng cơ bản của Unity, giao diện ứng dụng Editor đc đề cao hơn cả. Unity Editor, với nhiều công cụ giúp hỗ trợ tốt các đầu việc

phát triển game gồm tạo ra mô hình 3D (ánh sáng, vật lý), viết Script... chỉ được thực hiện bằng thao tác đơn giản là kéo và thả hình ảnh. Cũng chính nhờ vào tính năng này mà Unity đã hỗ trợ tốt cho những vị trí khác trong game phải kể đến là Game Designer.

### 1.2.2. Tổng quan về Game Engine

Game Engine được biết đến như là công cụ giúp cho các nhà sáng tạo game có thể phát triển loại hình game của mình một cách đơn giản và ít chi phí. Game Engine đã có sẵn lập trình về game giúp nhà sản xuất có thể dễ dàng tạo ra những thiết kế như khung cảnh, nhân vật, âm thanh, đồ họa; giúp quản lý bộ nhớ.



Hình 2: Minh họa game Engine.

Từ một game Engine có thể phát triển thành rất nhiều loại game khác nhau, từ đó có thể thấy rằng thế giới game ngày càng đa dạng và phong phú hơn. Game Engine hoạt động gồm:

a. Thiết bị đầu vào

Game Engine cung cấp các thiết bị đầu vào như chuột, cảm ứng, gamepad... xử lý dữ liệu đầu vào chủ yếu thông qua events và polling.

Máy tính sẽ ghi lại hành trình nhấn chuột trong events và mã tùy chỉnh của người dùng sẽ được kích hoạt dựa trên những dữ liệu được ghi lại đó. Đôi

với polling, nó được sử dụng để lấy vị trí dữ liệu như tọa độ con trỏ chuột hoặc dựa trên điện thoại di động.

#### b. Đồ họa

Các game thủ thường thấy game có đồ họa 2D và 3D đúng không? Đối với game có đồ họa 3D thường bắt mắt và thu hút hơn 2D bởi thế giới game 3D thường cho người chơi cảm giác chân thực và thích thú hơn.

Game Engine cung cấp cả tính năng tạo game 2D và 3D để các nhà sản xuất thoải mái sáng tạo. Bên cạnh đây game cung cấp một số tính năng có thể tùy chỉnh như hiệu ứng âm thanh, ánh sáng, đổ bóng, pha trộn...

Nhờ đồ họa của game Engine được cung cấp sẵn, các nhà sản xuất đã tiết kiệm nhiều về thời gian, chi phí để tạo ra các chi tiết nhỏ nhưng quan trọng này.

#### c. Công cụ vật lý

Đây là bộ công cụ được ứng dụng để tăng khả năng di chuyển vật lý, hiệu ứng chuyển động một cách chân thực và chính xác. Công cụ này giúp cho các nhân vật trong game di chuyển một cách mượt mà hơn mà không bị thô cứng, người chơi sẽ trải nghiệm cảm giác như mình được hòa nhập thực sự với thế giới game đó.

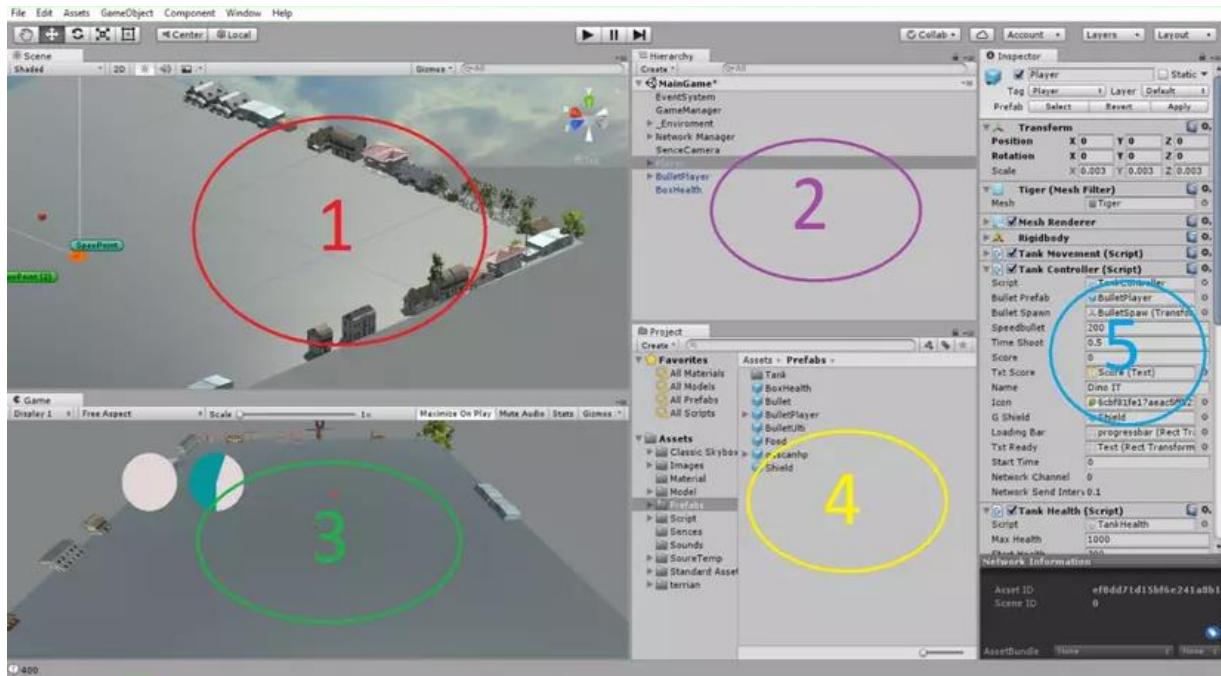
#### d. Trí tuệ nhân tạo

Công nghệ thực sự ngày càng hiện đại và đôi khi nó vượt qua cả trí tưởng tượng của con người. Công nghệ này có nhiệm vụ ghi lại hành trình người chơi sử dụng vũ khí dựa trên những tình huống khác nhau, để từ đó có thể đề xuất cách thức và hướng chơi tiếp theo.

### **1.3. Giới thiệu công nghệ, ngôn ngữ**

#### **1.3.1. Giới thiệu Unity Engine**

Các thành phần trong Unity Editor:



Hình 3: Giao diện làm việc Unity Editor.

### 1. Cửa sổ Sences

Phần này phần hiển thị các đối tượng trong scenes 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 thanh, cách nhìn 2D hay 3D ... Khung nhìn Scene là nơi bố trí các Game Object 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.

### 2. Cửa sổ Hierarchy

Tab hierarchy là nơi hiển thị các Game Object trong Sences hiện hành. Khi các đối tượng được thêm hoặc xóa trong Sences, 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.

### 3. 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...

### 4. 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.

### 5. 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 tùy chỉnh trực tiếp mà không cần thông qua một kịch bản định trước. Tuy nhiên Scripting API cung cấp một số lượng nhiều và dày đặc 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, ...

### **1.3.2. Giới thiệu về ngôn ngữ**

Lập trình Unity 2D, 3D được lập trình dựa trên 3 ngôn ngữ chính là C#, UnityScript và Boo. Trong đó, C# là ngôn ngữ chính mà rất nhiều lập trình viên Unity sử dụng ở thời điểm hiện tại.

C# (hay C sharp) là một ngôn ngữ lập trình đơn giản, được phát triển bởi đội ngũ kỹ sư của Microsoft vào năm 2000, trong đó người dẫn đầu là Anders Hejlsberg và Scott Wiltamuth. C# là ngôn ngữ lập trình hiện đại, hướng đối tượng và nó được xây dựng trên nền tảng của hai ngôn ngữ mạnh nhất là C++ và Java.

C# được thiết kế cho Common Language Infrastructure (CLI), mà gồm Executable Code và Runtime Environment, cho phép chúng ta sử dụng các ngôn ngữ high-level đa dạng trên các nền tảng và cấu trúc máy tính khác nhau. C# với sự hỗ trợ mạnh mẽ của .NET Framework giúp cho việc tạo một ứng dụng Windows Forms hay WPF (Windows Presentation Foundation), . . . trở nên rất dễ dàng.

Các đặc điểm để làm cho C# là ngôn ngữ lập trình chuyên nghiệp được sử dụng rộng rãi:

### 1. C# là ngôn ngữ đơn giản

Ngôn ngữ C# dựng trên nền tảng C++ và Java nên ngôn ngữ C# khá đơn giản. Nếu chúng ta thân thiện với C và C++ hoặc thậm chí là Java, chúng ta sẽ thấy C# khá giống về diện mạo, cú pháp, biểu thức, toán tử và những chức năng khác được lấy trực tiếp từ ngôn ngữ C và C++, nhưng nó đã được cải tiến để làm cho ngôn ngữ đơn giản hơn. Một vài trong các sự cải tiến là loại bỏ các dư thừa, hay là thêm vào những cú pháp thay đổi.

### 2. C# là ngôn ngữ hiện đại

Một vài khái niệm khá mới mẻ như xử lý ngoại lệ, những kiểu dữ liệu mở rộng, bảo mật mã nguồn ...v.v... Đây là những đặc tính được cho là của một ngôn ngữ hiện đại cần có. Và C# chứa tất cả các đặc tính ta vừa nêu trên. Các bạn sẽ dần tìm hiểu được các đặc tính trên qua các bài học trong series này.

### 3. C# là một ngôn ngữ lập trình thuần hướng đối tượng

Lập trình hướng đối tượng (tiếng Anh: Object-oriented programming, viết tắt: OOP) là một phương pháp lập trình có 4 tính chất. Đó là tính trừu tượng (abstraction), tính đóng gói (encapsulation), tính đa hình (polymorphism) và tính kế thừa (inheritance). C# hỗ trợ cho chúng ta tất cả những đặc tính trên. Và để hiểu rõ hơn thì chúng ta sẽ có một chương trình bày về phần này.

Ngoài những đặc điểm trên thì còn một số ưu điểm nổi bật của C#:

- C# có cấu trúc khá gần gũi với các ngôn ngữ lập trình truyền thống, nên cũng khá dễ dàng tiếp cận và học nhanh với C#.
- C# có thể biên dịch trên nhiều nền tảng máy tính khác nhau.

- C# được xây dựng trên nền tảng của C++ và Java nên nó được thừa hưởng những ưu điểm của ngôn ngữ đó.
- C# là một phần của .NET Framework nên được sự chống lưng khá lớn đến từ bộ phận này.
- C# có IDE Visual Studio cùng nhiều plug-in vô cùng mạnh mẽ

## **CHƯƠNG 2. THIẾT KẾ Ý TƯỞNG GAME**

### **2.1. Giới thiệu:**

Trong game HeroKid Sword, người chơi sẽ nhập vai vào một nhân vật, cụ thể là nhân vật HeroKid và đối đầu với các quái vật và kẻ thù khác trong một môi trường 2D được tạo hình bằng pixel.

Trang bị sử dụng trong game là thanh Wood, thanh kiếm của HeroKid để tiêu diệt quái vật.

Người chơi sẽ phải điều khiển di chuyển, tấn công, và sử dụng các kỹ năng của HeroKid để chiến đấu và đánh bại các quái vật, thu thập vật phẩm, nâng cấp nhân vật và trang bị để trở nên mạnh mẽ hơn và vượt qua các thử thách khó khăn hơn.

Với cốt truyện lôi cuốn và âm thanh sống động, người chơi sẽ đi qua các cung bậc cảm xúc khác nhau.

### **2.2. Thể loại game 2D pixel top-down:**

Thể loại của HeroKid Sword là game 2D pixel top-down đòi hỏi người chơi sẽ điều khiển một nhân vật trong một môi trường 2D được tạo thành từ các pixel, và quan sát game từ góc nhìn từ trên xuống (top-down). Nó thường được sử dụng cho các game phiêu lưu, hành động, và thường có phong cách đồ họa retro.

Một số ví dụ về game 2D pixel top-down nổi tiếng có thể kể đến như The Legend of Zelda, Stardew Valley, và Enter the Gungeon.

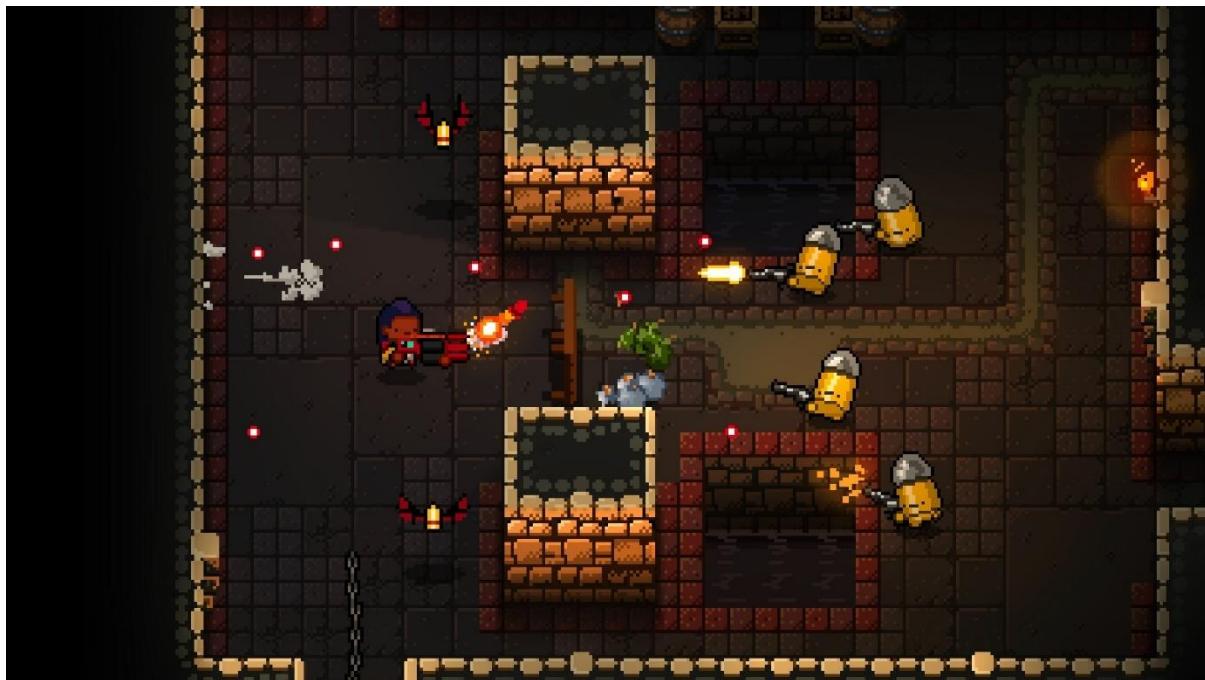


Hình 4: Hình ảnh minh họa về game 2D pixel top – down.

Để đạt được thành tích cao trong game, người chơi cần phải sử dụng các kỹ năng và chiến lược phù hợp, tìm ra đường đi tối ưu và tận dụng các vật phẩm hỗ trợ một cách thông minh. Với đồ họa tuyệt đẹp, âm thanh sống động và gameplay hấp dẫn, game đua xe vượt địa hình là một trong những trò chơi hấp dẫn và thú vị nhất cho những người yêu thích thể loại game thể thao và đua xe.

### 2.3. Tóm tắt game:

Game thường có đồ họa đơn giản nhưng sáng tạo, âm thanh sống động và hệ thống gameplay thú vị, đòi hỏi người chơi phải có kỹ năng và sự nhanh nhẹn trong việc điều khiển nhân vật của mình. Một số game đánh quái nổi tiếng trong thể loại này có thể kể đến như The Binding of Isaac, Enter the Gungeon và Hyper Light Drifter.



Hình 5: Game Enter the Gungeon.

Với những ai yêu thích thể loại game đánh quái và đồ họa pixel đặc trưng, thì nó là một sự lựa chọn tuyệt vời để trải nghiệm. Như đã nói ở trên sự khéo léo trong di chuyển cũng như chiến đấu đánh bại mục tiêu là thứ hấp dẫn ban đầu, người chơi quan tâm đến cách thức chơi đầu tiên.

#### 2.4. Khách hàng mục tiêu:

Game thường hướng đến khách hàng là những người yêu thích các trò chơi hành động phiêu lưu và thử thách bản thân với các trận chiến đối đầu với quái vật và kẻ thù trong một môi trường tạo hình bằng pixel đơn giản nhưng đầy sáng tạo.

Một điểm đặc biệt của game là nó có thể được chơi bởi mọi lứa tuổi từ 5 tuổi trở lên. Vì đồ họa và âm thanh được thiết kế đơn giản và dễ hiểu, nó có thể giúp trẻ em phát triển kỹ năng tư duy, tăng cường khả năng phản xạ và cải thiện sự tập trung.

Ngoài ra, do đặc tính đơn giản và thời lượng chơi không quá dài, thể loại game này còn phù hợp với những người có ít thời gian và muốn tìm kiếm một trò chơi giải trí nhẹ nhàng để giải tỏa căng thẳng và thư giãn.

## **2.5. Điểm mạnh của game:**

- Gameplay thú vị và đầy thử thách.
- Đồ họa pixel đơn giản, sáng tạo và dễ thao tác.
- Thiết kế, tạo hình nhân vật độc lạ, quen thuộc.
- Tại mỗi màn bạn sẽ có 1 nhiệm vụ khác nhau mang đến sự thích thú khi chơi.
- Đồ họa thiết kế đẹp mắt 2D.
- Có đa dạng các màn chơi khác nhau.
- Bạn sẽ được trải qua những trận chiến đầy gian khổ với những thử thách chưa từng có.
- Âm thanh trong game sống động giúp kích thích người chơi. Luật chơi cũng đơn giản kết hợp với các hiệu ứng, bạn chỉ cần chơi thử 1 lần là lần sau sẽ trở thành cao thủ....
- Game mượt mà không bị giật.
- Nhiều kỹ năng và trang bị để mở khóa: Trong quá trình chơi, người chơi có thể thu thập tiền và kinh nghiệm để mở khóa các kỹ năng và trang bị mới cho nhân vật của mình, giúp tăng cường sức mạnh và khả năng chiến đấu của nhân vật

## **2.6. Phong cách nghệ thuật game:**

Phong cách game được thiết kế theo hướng retro và đồ họa pixel, mang lại một cảm giác hoài niệm về những trò chơi điện tử cổ điển từ những năm 80 và 90.

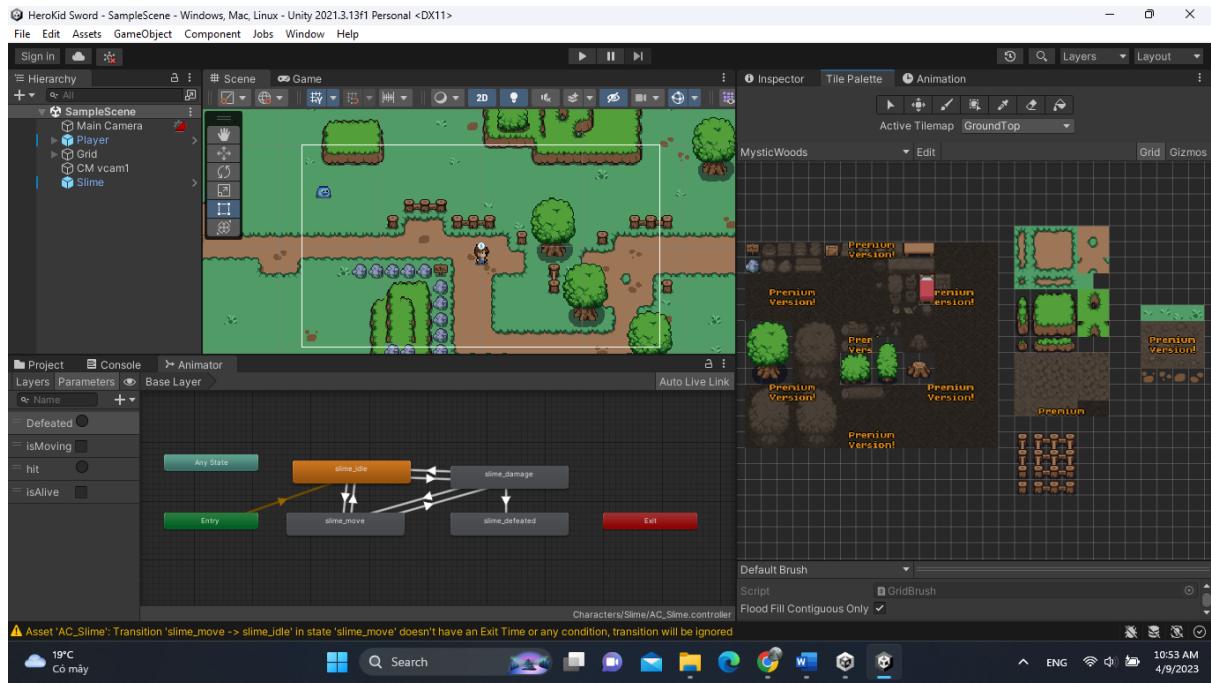
Các game trong thể loại này thường có đường đi hướng trên xuống và được thiết kế với góc nhìn từ trên xuống (top-down), cho phép người chơi quan sát được toàn bộ môi trường xung quanh. Các đối tượng trong trò chơi được thiết kế với đồ họa pixel đơn giản nhưng sáng tạo, giúp người chơi có thể dễ dàng nhận ra và phân biệt chúng.

Một số game cùng thể loại này cũng có yếu tố RPG, cho phép người chơi mở khóa các kỹ năng và trang bị mới để tăng cường sức mạnh và khả năng chiến đấu của nhân vật, tạo ra một trải nghiệm chơi game thú vị và đa dạng.

Tâm trạng khi chơi game cổ điển có thể thay đổi tùy thuộc vào cốt truyện và trải nghiệm của người chơi. Thỉnh thoảng, người chơi có thể cảm thấy căng thẳng hoặc hồi hộp trong những tình huống nguy hiểm, nhưng cũng có thể thấy sự hài lòng khi vượt qua các thử thách và hoàn thành nhiệm vụ.

Tóm lại, phong cách game cổ điển mang lại trải nghiệm phiêu lưu thú vị và đầy thử thách cho người chơi, đồng thời cung cấp một thế giới đầy màu sắc để khám phá và tận hưởng.

## 2.7. Thiết bị trải nghiệm game:



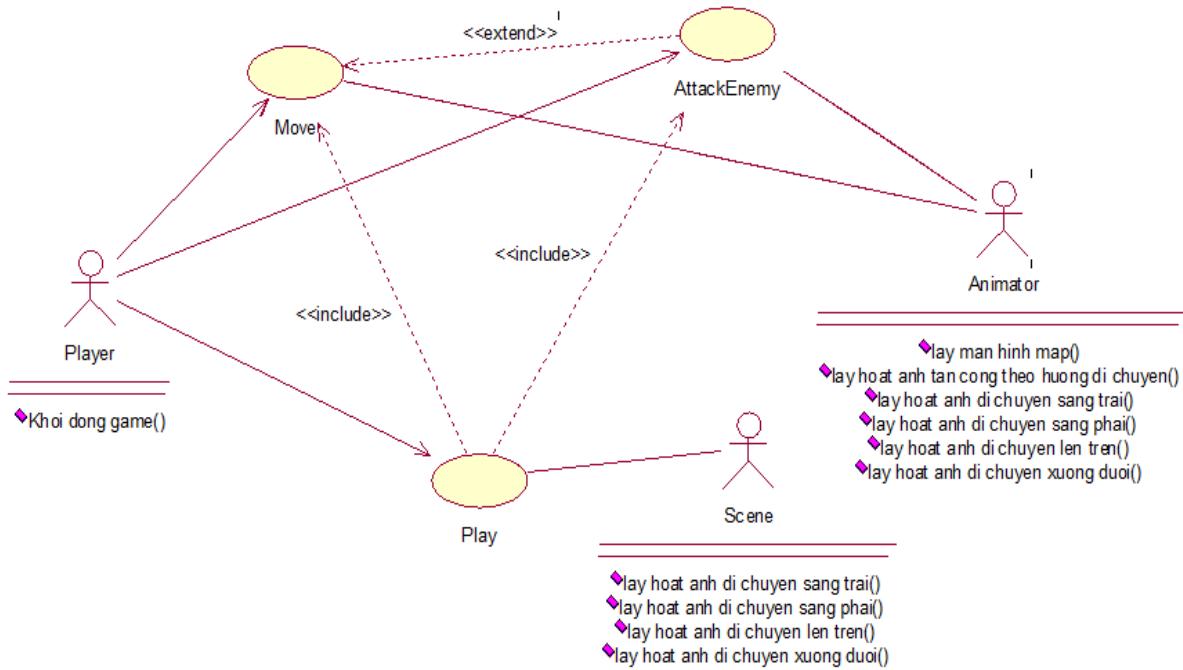
Hình 6: Hình ảnh thiết kế game trên Unity.

- Game được phát triển và xây dựng để chơi trên các PC và laptop.
- Hướng tới chơi game trên các nền tảng còn lại như mobile, ps4, ps5,...

## CHƯƠNG 3. XÂY DỰNG PHÁT TRIỂN GAME

### 3.1. Thiết kế hệ thống

#### 3.1.1. Biểu đồ Use Case



Hình 7: Biểu đồ Use Case của game.

#### 3.1.2. Mô tả các Use Case

##### 1. Use case: Play

- Mô tả: Use case này mô tả hành vi khi người chơi nhấn nút "Play" để bắt đầu trò chơi trong hệ thống game.
- Người tham gia.

- Người chơi: Người sử dụng hệ thống game.
- Luồng chính:
  - Người chơi khởi động hệ thống game.
  - Hệ thống game hiển thị giao diện menu với nút "Play".
  - Người chơi nhấn nút "Play".
  - Hệ thống game chuyển đến màn hình chơi game và bắt đầu trò chơi.
  - Người chơi tiếp tục tương tác và chơi trò chơi theo luồng chơi game chính.
- Luồng thay thế và ngoại lệ:

- Nếu người chơi không tương tác, game tiếp tục hiển thị môi trường và nhân vật ở trạng thái trước đó.
- Yêu cầu phụ:
  - Giao diện người dùng phải hiển thị rõ ràng nút "Play" để người chơi có thể nhận biết và tương tác.

## 2. Use case: Move

- Mô tả: Use case này mô tả quá trình di chuyển của nhân vật trong môi trường game.
- Người tham gia.
 

Người chơi: Tương tác với game thông qua bàn phím hoặc điều khiển.
- Luồng chính:
  - Người chơi khởi động game.
  - Game hiển thị môi trường và nhân vật trên màn hình.
  - Người chơi sử dụng bàn phím hoặc điều khiển để tương tác với game.
  - Game kiểm tra phím di chuyển được nhấn.
  - Game thay đổi tọa độ của nhân vật dựa trên phím di chuyển và các quy tắc di chuyển.
  - Game cập nhật hiển thị môi trường và nhân vật với tọa độ mới.
  - Quá trình di chuyển tiếp tục cho đến khi người chơi kết thúc game hoặc tương tác với các use case khác.
- Luồng thay thế và ngoại lệ:
  - Nếu người chơi không tương tác, game tiếp tục hiển thị môi trường và nhân vật ở trạng thái trước đó.
- Yêu cầu phụ:
  - Game phải theo dõi tọa độ hiện tại của nhân vật.
  - Game phải xử lý các phím di chuyển từ người chơi.
  - Game phải áp dụng quy tắc di chuyển đúng để giới hạn vùng di chuyển của nhân vật.
  - Game phải cập nhật hiển thị môi trường và nhân vật sau khi thay đổi tọa độ.

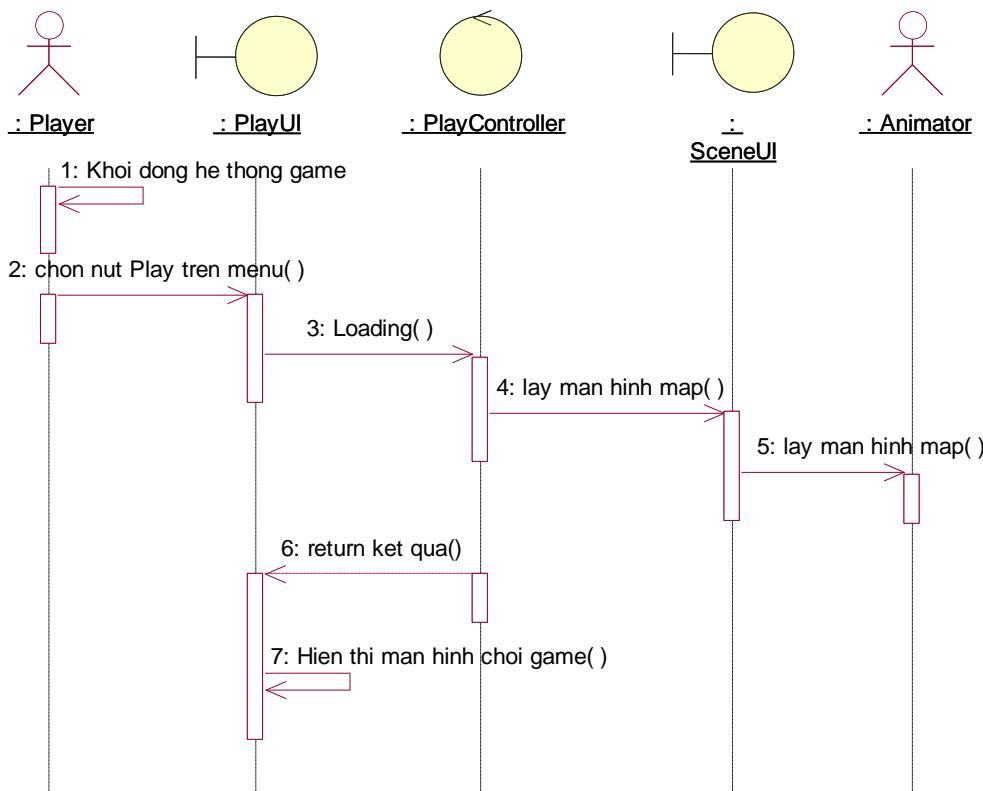
## 3. Use case: AttackEnemy

- Mô tả: Use case này mô tả quá trình tấn công kẻ địch trong môi trường game.
- Người tham gia.

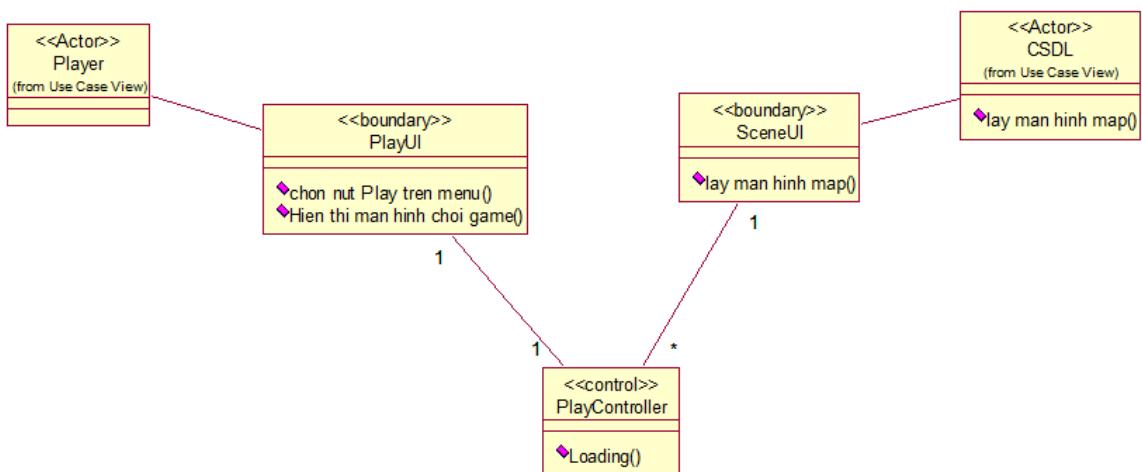
- Nhân vật người chơi: Đại diện cho người chơi trong trò chơi.
- Kẻ địch: Đối thủ hoặc quái vật mà người chơi cần tấn công.
- Luồng chính:
  - Người chơi tiếp cận hoặc gặp gỡ kẻ địch trong trò chơi.
  - Người chơi quyết định tấn công kẻ địch.
  - Trò chơi tính toán sự tấn công dựa trên các yếu tố như sức mạnh, độ chính xác và các yếu tố khác của nhân vật người chơi và kẻ địch.
  - Trò chơi áp dụng kết quả tấn công, gây thiệt hại cho kẻ địch.
  - Trò chơi cập nhật trạng thái của kẻ địch sau khi bị tấn công, bao gồm cập nhật điểm máu (HP) hoặc trạng thái khác.
  - Nếu kẻ địch chưa bị hạ gục, trò chơi cho phép người chơi tiếp tục tương tác hoặc lựa chọn hành động tiếp theo.
- Luồng thay thế và ngoại lệ:
  - Nếu người chơi không quyết định tấn công, trò chơi không thực hiện bất kỳ hành động tấn công nào và tiếp tục trạng thái hiện tại.
  - Nếu kẻ địch đã bị hạ gục sau khi bị tấn công, trò chơi cập nhật trạng thái của kẻ địch và thông báo cho người chơi về kết quả.
- Yêu cầu phụ:
  - Trò chơi phải có hệ thống nhân vật người chơi và kẻ địch.
  - Quá trình tính toán sự tấn công và gây thiệt hại phải được xây dựng dựa trên quy tắc và cân nhắc của trò chơi.
  - Giao diện người dùng phải cung cấp các tùy chọn và nút để người chơi thực hiện tấn công và theo dõi kết quả.

### **3.1.3. Phân tích các Use Case**

1. Use case Play:
- Biểu đồ trình tự:

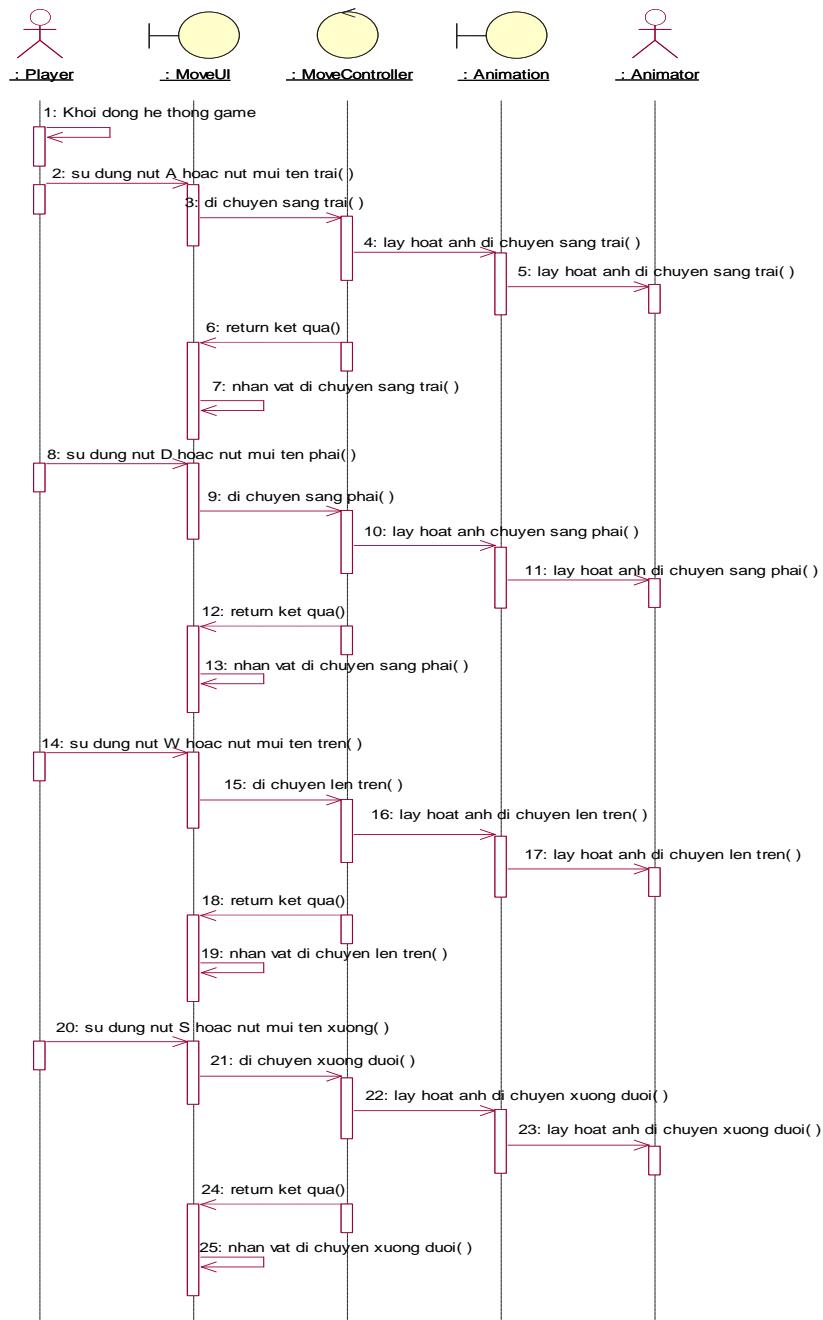


- Biểu đồ VOPC:

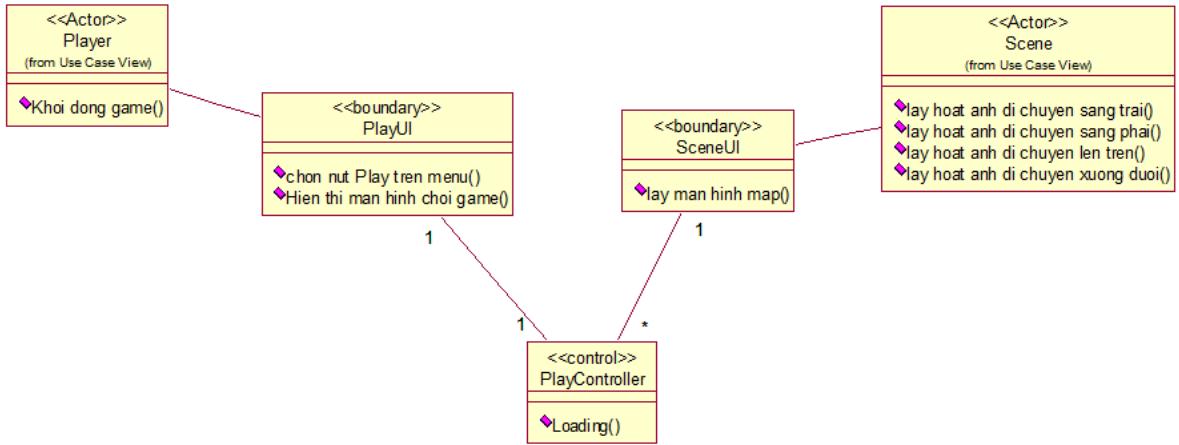


## 2. Use Case Move:

- Biểu đồ trình tự:

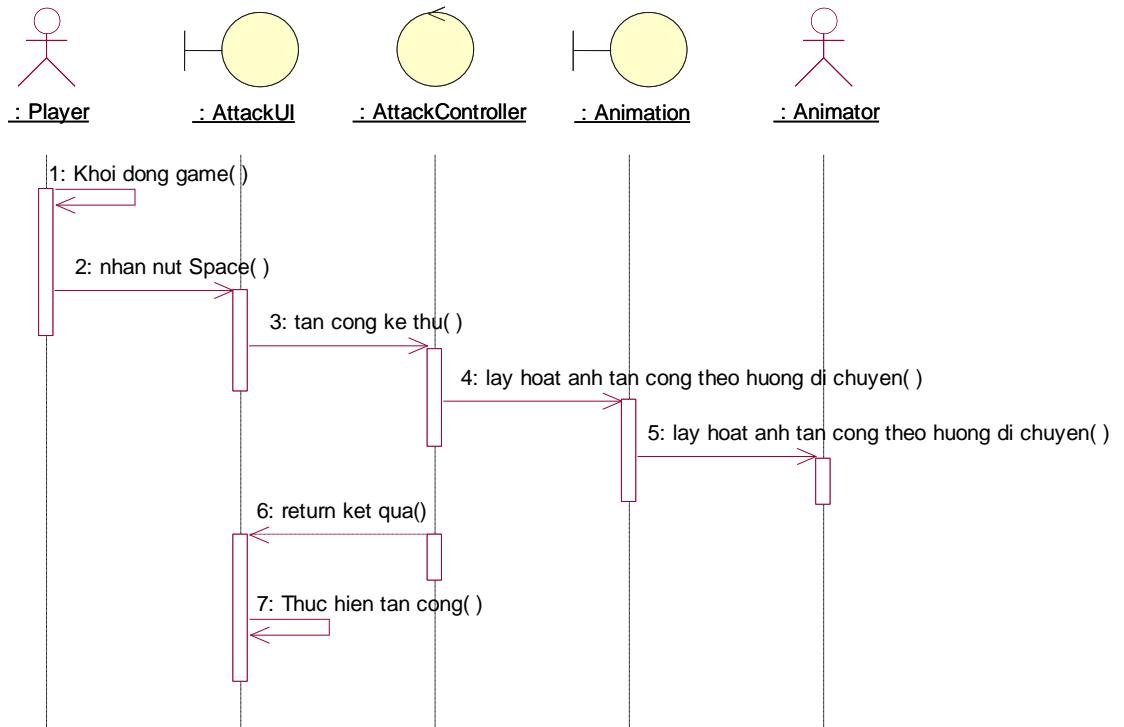


- Biểu đồ VOPC:

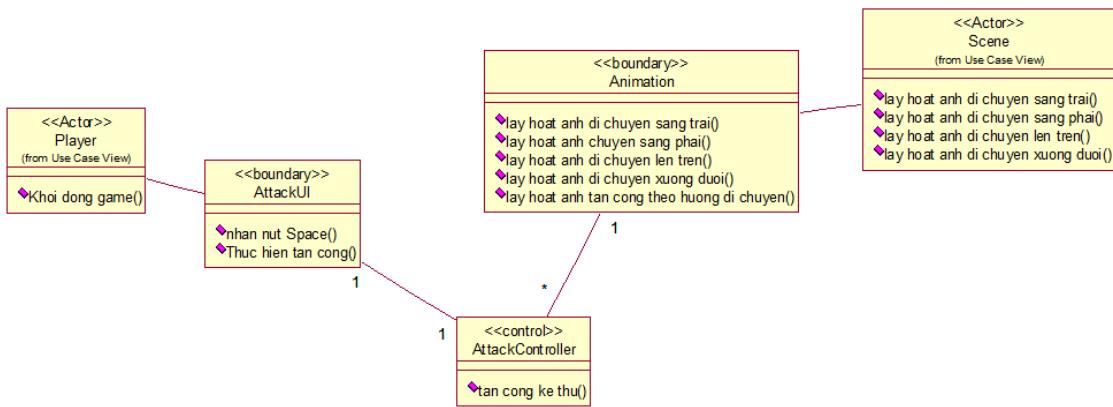


### 3. Use Case AttackEnemy:

- Biểu đồ trình tự:



- Biểu đồ VOPC:



## 3.2. Thiết kế kịch bản game

### 3.2.1. Cách chơi chính:

Trò chơi yêu cầu phần cứng như: Máy tính, bàn phím, loa, chuột, ...

Những nút được sử dụng:

- Phím “A” hoặc phím “mũi tên sang trái”: Nhân vật di chuyển sang bên trái.
- Phím “D” hoặc phím “mũi tên sang phải”: Nhân vật di chuyển sang phải.
- Phím “W” hoặc phím “mũi tên lên trên”: Nhân vật di chuyển lên trên.
- Phím “S” hoặc phím “mũi tên xuống dưới”: Nhân vật di chuyển xuống dưới.
- Phím “Space”(phím cách): Nhân vật sử dụng kiếm để đánh quái.
- Tài liệu hướng dẫn: Nằm ở phần hướng dẫn trong menu chính của trò chơi.



Hình 8: Màn hình hướng dẫn chơi game.

### **3.2.2. Cốt truyện của game:**

HeroKid là một cậu bé dũng cảm, quyết tâm tiêu diệt tất cả các quái vật để bảo vệ thế giới Mystic. Cậu sở hữu một thanh kiếm sắc bén có tên là Wood, và sử dụng kỹ năng của mình để trở thành anh hùng trẻ tuổi nhất trong lịch sử Mystic.

Trên con đường tiêu diệt quái vật, Cậu phải đổi mặt với nhiều thử thách và nguy hiểm khác nhau. Cậu sẽ đi qua nhiều vùng đất khác nhau, chinh phục các thử thách và đánh bại các thủ lĩnh của quái vật. Với sự gan dạ, tài năng và lòng can đảm, HeroKid đã trở thành một anh hùng được tôn vinh trong lòng các cư dân của thế giới Mystic. Hãy dõi theo cuộc hành trình của cậu ta nhé!

### **3.2.3. Các phần tử của game:**

- Thế giới trò chơi rộng lớn, rộng khoảng 10 – 30 lần so với màn hình trò chơi.
- Thiết kế theo một dạng địa hình, 1 khu vực, sau khi kết thúc một địa hình sẽ qua màn và tiếp tục màn chơi mới.
- HeroKid: Nhân vật chính trong game.



*Hình 9: Nhân vật HeroKid.*

- Enemy: Các kẻ thù của nhân vật chính.



*Hình 10: Slime.*

Là sinh vật yếu ớt nhất trong hệ sinh thái của quái vật, tuy vậy nhiều người dân vẫn thường chủ quan vì chất độc của chúng có thể gây chết người.



*Hình 11: Nepenthe.*

Là loài cây ăn thịt người, chúng thường ngủ ở các khu rừng già, bụi đậm. Khi làm chúng thức dậy chúng sẽ tấn công.



*Hình 12: Các loại Skeleton.*

Là loài quái vật xuất hiện mạnh mẽ ở các hầm ngục, chúng tàn bạo hung ác và cực kì nguy hiểm cho thế giới. Tiêu diệt chúng là điều tất yếu.

Có 3 loại là : Skeneton While, Red và cầm khiên, sức mạnh và sức phòng thủ tăng dần.

- Objects: Các đối tượng.



Hình 13: Objects

Các đối tượng như máu, rương, cây năng lượng,... Những đối tượng này thường sử dụng làm đồ trang trí động và vật phẩm.

- Tile Map:



Hình 14: Tilemap Jungle

Các tile đơn giản để thiết kế map, khu vực rừng rậm.



Hình 15: Tilemap Overworld

Cũng có một số tile dùng để xây dựng khu rừng. Ngoài ra có các tile khác như biển, cầu, lâu đài, tòa nhà,...



Hình 16: Tilemap house

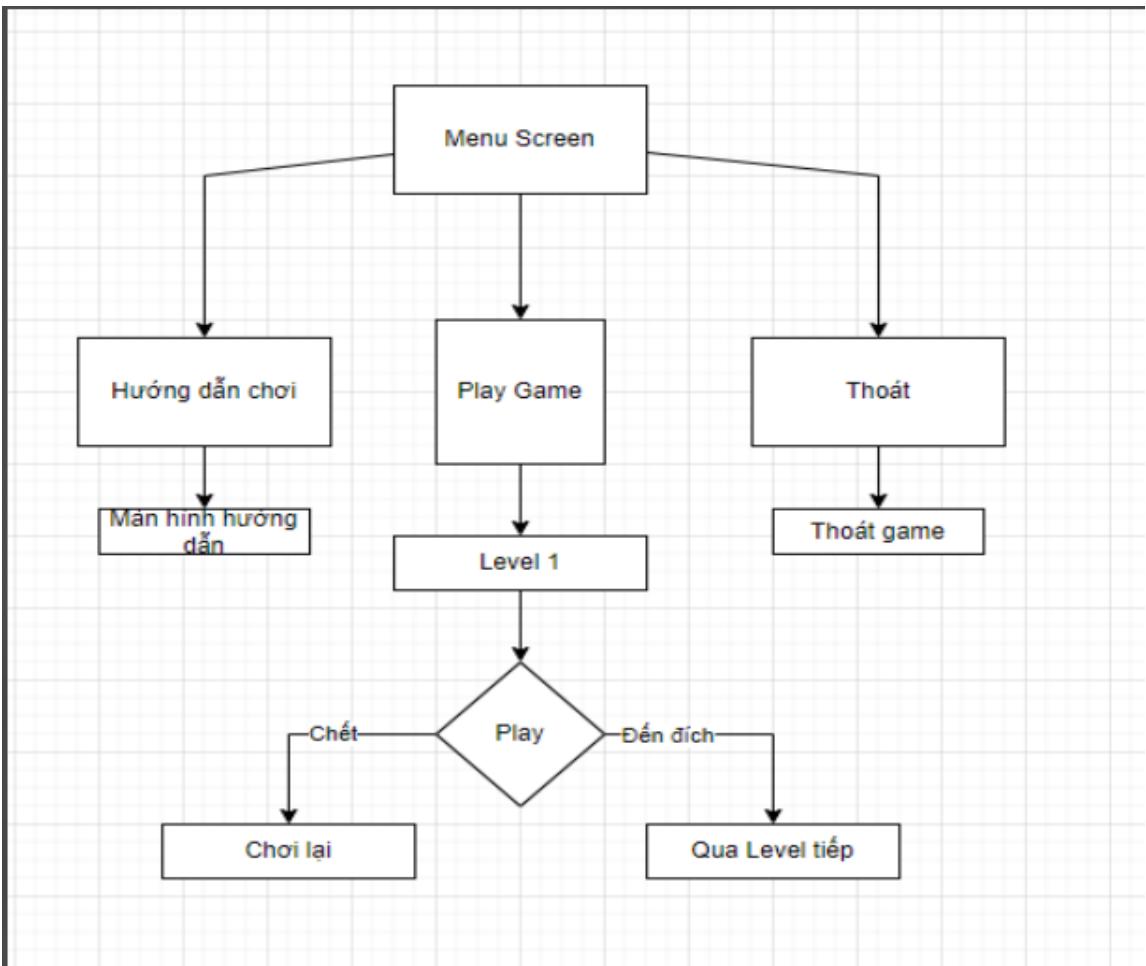
Các tile xây dựng nhà của HeroKid hoặc nhà của dân làng.

#### 3.2.4. Các cơ chế của game:

- Khá đơn giản, tiêu diệt hết lũ quái vật từ cấp thấp đến cấp cao. Đi qua nhiều nơi tìm kiếm các hầm ngục và phá bỏ chúng.
- Tránh để thanh HP về 0, nhân vật sẽ chết.

### 3.3. Thiết kế giao diện

#### 3.3.1. Biểu đồ (Flowchart)



Hình 17: Biểu đồ - Flowchart

### 3.3.2. Mô tả

- Opening Screen: Màn hình này liên quan đến các khoản tín dụng phát triển. Đây là màn hình đầu tiên xuất hiện khi ứng dụng đang tải.
- Start Screen: Màn hình này xuất hiện khi ứng dụng được tải đầy đủ. Đó là chứa thương hiệu dự án, nút bắt đầu và các level chơi. Chọn để bắt đầu thay đổi màn hình tiếp theo.
- Level Screen: Đây là màn hình của level chọn tương ứng hoặc level sau khi qua màn, bao gồm: bản đồ lưới tương tác, trình phát, đối tượng trò chơi, kịch bản không tương tác, GUI hộp hiệu suất (tính năng cấp độ, số máu), tạm dừng và các nút cấu hình. Thoát khỏi ứng dụng trong giai đoạn này, mở lại trong Start Screen.

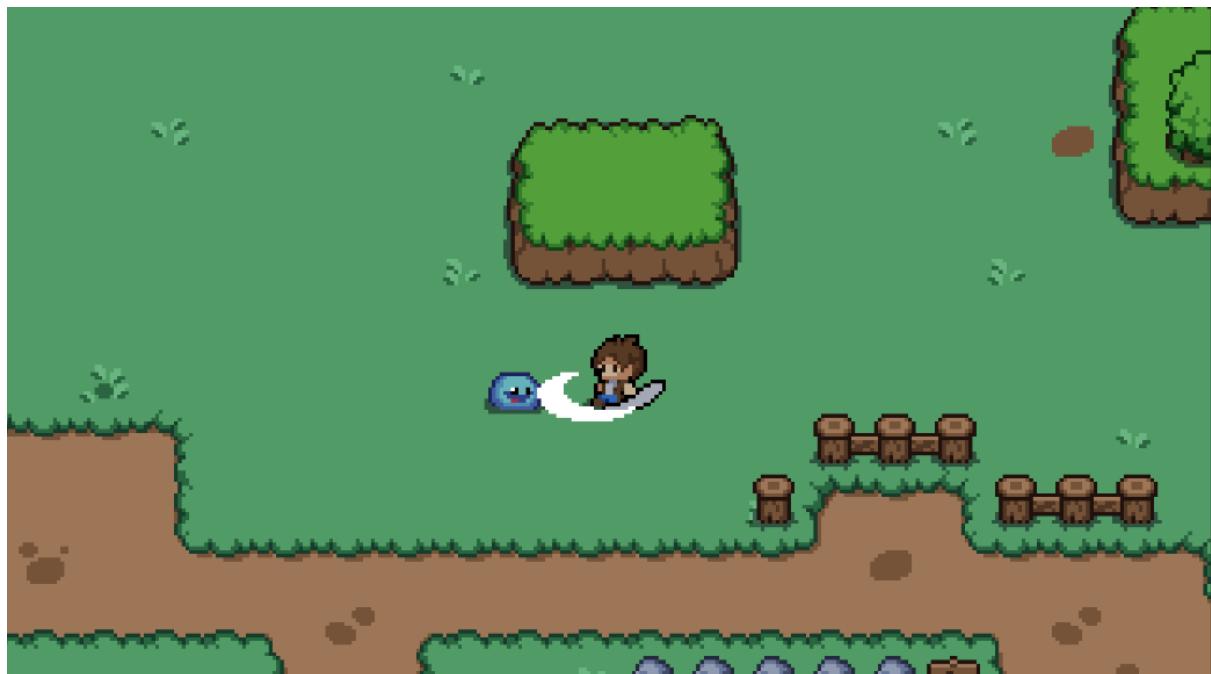
### 3.3.3. Giao diện các màn hình

#### a) Start menu

Màn hình gồm có 3 button có chức năng khác nhau, có hình ảnh và chữ

- Nút “PLAY” để bắt đầu vào game.
- Nút “GAMEPLAY GUIDE” để chuyển sang màn hình hướng dẫn cách chơi game.
- Nút “QUIT” để thoát ra khỏi chế độ chơi.

#### b) Giao diện chơi game



*Hình 18: Tấn công Slime*

Nhân vật tấn công Slime theo hướng trái, khi tấn công nó sẽ chuyển trạng thái từ đứng im sang di chuyển và tiếp cận nhân vật để tấn công.

Khi đánh Slime sẽ bị bật lại một khoảng nhỏ. Sau khi tiêu diệt nó sẽ rớt tiền vàng hoặc trái tim.



Hình 19: Tấn công Nepenthe

Nhân vật tấn công Nepenthe, nó cũng sẽ bị bật lại một khoảng như Slime. Sau khi tiêu diệt nó sẽ rót tiền vàng hoặc trái tim.



Hình 20: Tấn công Skeleton

Nhân vật tấn công Skeleton, nó cũng sẽ bị bật lại một khoảng như các kẻ thù khác. Sau khi tiêu diệt nó sẽ rót tiền vàng hoặc vật phẩm trang bị.

### **3.4. Thiết kế âm thanh**

Sự kiện	Âm thanh
Nhạc nền menu	background.mp3
Nhân vật sử dụng kiếm	sword_basic.mp3
Slime bị nhân vật chém	sword_kill_slime.mp3
Click	click.mp3
SoundEffect	Soundeffect.mav
Nhạc nền khi chơi game	nhacig.mp3
...	...

## CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ

### 4.1. Các kỹ thuật thực hiện

#### 4.1.1. Kỹ thuật của các đối tượng

1) Kỹ thuật của Player.

a) Khai báo biến

```
private Rigidbody2D rb;
private Animator animator;

SpriteRenderer spriteRenderer;

[SerializeField]
private float speed = 150f;

private float attackTime = .25f;
private float attackCounter = .25f;
private bool isAttacking;
```

- Khai báo một biến Rigidbody2D được sử dụng để điều khiển vật lý của nhân vật.
- Khai báo một biến Animator được sử dụng để điều khiển animation của nhân vật.
- Khai báo một biến SpriteRenderer được sử dụng để điều khiển sprite (hình ảnh) của nhân vật.
- Thiết lập giá trị mặc định cho tốc độ, thời gian và trạng thái tấn công của nhân vật.
- Thêm biến isAttacking để đánh dấu xem nhân vật có đang tấn công hay không.

b) Di chuyển

```

rb.velocity = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical")) * speed * Time.deltaTime;
animator.SetFloat("moveX", rb.velocity.x);
animator.SetFloat("moveY", rb.velocity.y);

if(Input.GetAxisRaw("Horizontal") == 1 || Input.GetAxisRaw("Vertical") == 1
|| Input.GetAxisRaw("Horizontal") == -1 || Input.GetAxisRaw("Vertical") == -1)
{
    animator.SetFloat("lastMoveX", Input.GetAxisRaw("Horizontal"));
    animator.SetFloat("lastMoveY", Input.GetAxisRaw("Vertical"));
}

```

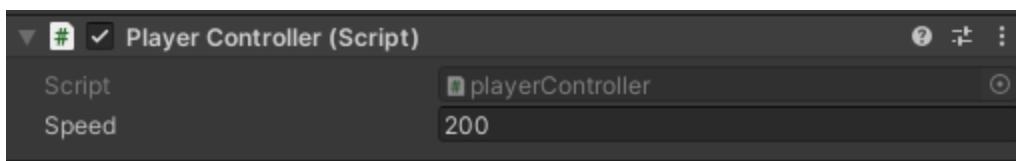
- Cập nhật vận tốc, các tham số cho animation của đối tượng theo trục X và trục Y.
- Các tham số moveX, movY để thay đổi hướng di chuyển của đối tượng.
- Kiểm tra điều kiện để thay đổi trạng thái di chuyển khi chúng ta nhấn các nút di chuyển.
- Hàm hỗ trợ di chuyển: Hỗ trợ di chuyển phía bên trái do hoạt ảnh của nhân vật chỉ có phía bên phải.

```

public void FixedUpdate()
{
    if(rb.velocity.x < 0){
        spriteRenderer.flipX = true;
        // gameObject.BroadcastMessage("IsFacingRight",false);
    }else if(rb.velocity.x > 0){
        spriteRenderer.flipX = false;
        // gameObject.BroadcastMessage("IsFacingRight",true);
    }
}

```

Sau khi chạy ta cũng có thể thay đổi thông số thông qua cửa sổ Inspector:



Hình 21: Player Controller

c) Tân công kẻ thù

## Cập nhật trạng thái tấn công kẻ thù của Player

```
if(isAttacking)
{
    rb.velocity = Vector2.zero;
    attackCounter -= Time.deltaTime;
    if(attackCounter <= 0)
    {
        animator.SetBool("isAttacking", false);
        isAttacking = false;
    }
}

if(Input.GetKeyDown(KeyCode.Space))
{
    attackCounter = attackTime;
    animator.SetBool("isAttacking", true);
    isAttacking = true;
}
```

- Nếu biến boolean isAttacking của đối tượng là true, phương thức sẽ dừng đối tượng và giảm bớt đếm attackCounter theo thời gian thực. Nếu attackCounter đạt giá trị nhỏ hơn hoặc bằng 0, phương thức sẽ tắt animation tấn công và đặt isAttacking về false.
- Nếu người chơi nhấn phím Space, phương thức sẽ thiết lập attackCounter về giá trị attackTime (được khai báo ở đầu script), bật animation tấn công và đặt isAttacking thành true để bắt đầu quá trình tấn công của nhân vật.

## d) HealthManager (Quản lý máu của nhân vật):

- Khai báo các biến liên quan đến trạng thái máu của nhân vật và các biến liên quan đến hiệu ứng flash khi nhân vật chịu sát thương.

```

public int currentHealth = 100;
public int maxHealth = 100;

private bool flashActive;
[SerializeField]
private float flashLength = 0f;
private float flashCounter = 0f;
private SpriteRenderer playerSprite;

```

- Hàm xử lý giảm máu của nhân vật khi bị kẻ thù tấn công.

```

public void HurtPlayer(int damageToGive)
{
    currentHealth -= damageToGive;
    flashActive = true;
    flashCounter = flashLength;
    if(currentHealth <= 0)
    {
        gameObject.SetActive(false);
    }
}

```

- Hiệu ứng flash khi nhân vật bị tấn công:

```

if(flashActive)
{
    if(flashCounter > flashLength *.99f)
    {
        playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,0f);
    }
    else if(flashCounter > flashLength *.82f)
    {
        playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,1f);
    }
    else if(flashCounter > flashLength *.66f)
    {
        playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,0f);
    }
    else if(flashCounter > flashLength *.49f)
    {
        playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,1f);
    }
}

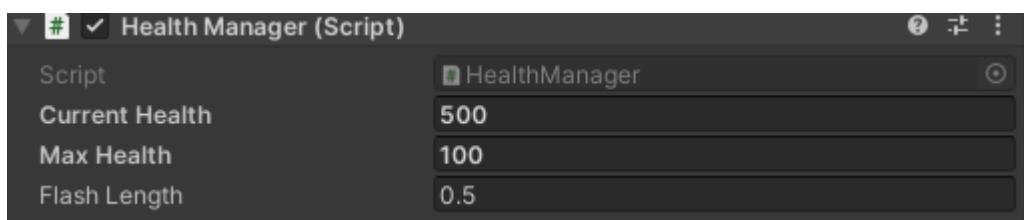
```

```

        else if(flashCounter > flashLength *.33f)
        {
            playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,0f);
        }
        else if(flashCounter > flashLength *.16f)
        {
            playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,1f);
        }
        else if(flashCounter > 0f)
        {
            playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,0f);
        }
        else
        {
            playerSprite.color = new Color(playerSprite.color.r,
playerSprite.color.g, playerSprite.color.b,1f);
            flashActive = false;
        }
        flashCounter -= Time.deltaTime;
    }
}

```

Sau khi code hoàn tất các kĩ thuật chúng ta có thể sửa khác đi để phù hợp với sở thích của chúng ta bằng cách thay đổi thông số thông qua cửa sổ Inspector:



Hình 22: *Health Manager*

## 2) Kĩ thuật của Slime

### a) Trạng thái máu:

Thuộc tính sau được sử dụng để quản lý trạng thái máu và các hiệu ứng liên quan đến máu:

```

public float Health{
    set{

        if(value < health){
            animator.SetTrigger("hit");
        }
        health = value;
        if(health<=0){
            animator.SetBool("isAlive",false);
            Targetable = false;
        }
    }
    get{
        return health;
    }
}

```

b) Trạng thái tấn công:

Thuộc tính sau được sử dụng để quản lý trạng thái của Slime có thể tấn công hay không.

```

public bool Targetable{
    get{
        return targetable;
    }
    set{
        targetable = value;
        rb.simulated = value;
    }
}

```

c) Trạng thái bị tấn công, tiêu diệt:

- Phương thức "Defeated": được sử dụng để thiết lập trạng thái chuyển động của Slime thành trạng thái "Defeated".
- Phương thức "RemoveEnemy": được sử dụng để xóa Slime khỏi trò chơi. Phương thức này thường được gọi khi Slime bị tiêu diệt.
- Phương thức "OnHit(float damage, Vector2 knockback)": được sử dụng để xử lý sự kiện khi Slime bị tấn công bởi một tác nhân nào đó. Phương thức này có hai đối số: "damage" - mức độ sát thương nhận được và "knockback" - lực đẩy gây ra bởi tác nhân tấn công. Phương thức này

giảm mức lượng máu của Slime bằng giá trị "damage" và đẩy Slime bằng lực "knockback" được áp dụng trên Slime. Phương thức này thường được gọi khi Slime bị tấn công bởi một tác nhân nào đó.

- Phương thức "OnHit(float damage)": tương tự như phương thức "OnHit(float damage, Vector2 knockback)", nhưng không có đối số "knockback". Phương thức này chỉ giảm mức lượng máu của Slime bằng giá trị "damage". Phương thức này thường được sử dụng khi Slime không cần phải bị đẩy đi sau khi bị tấn công.

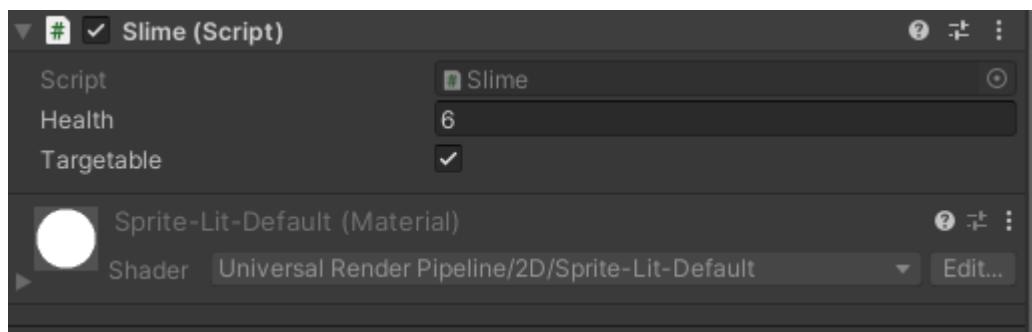
```
public void Defeated(){
    animator.SetTrigger("Defeated");
}

public void RemoveEnemy(){
    Destroy(gameObject);
}

public void OnHit(float damage, Vector2 knockback){
    Health -= damage;
    rb.AddForce(knockback);
}

public void OnHit(float damage){
    Health -= damage;
}
```

Sau khi chạy ta vẫn có thể thay đổi thông số thông qua cửa sổ Inspector:



Hình 23: Slime Script

### 3) Kĩ thuật của các Enemy khác

#### a) EnemyHealthManager (Quản lý máu của kẻ thù)

- Khai báo các biến và hàm bắt đầu, liên quan đến trạng thái máu của kẻ thù và các biến liên quan đến hiệu ứng flash khi kẻ thù chịu sát thương.

```
public int currentHealth;
public int maxHealth;

private bool flashActive;
[SerializeField]
private float flashLength = 0f;
private float flashCounter = 0f;
private SpriteRenderer enemySprite;
void Start()
{
    enemySprite = GetComponent<SpriteRenderer>();
}
```

- Hiệu ứng flash khi kẻ thù bị nhân vật tấn công:

```
if(flashActive)
{
    if(flashCounter > flashLength *.99f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,0f);
    }
    else if(flashCounter > flashLength *.82f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,1f);
    }
    else if(flashCounter > flashLength *.66f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,0f);
    }
    else if(flashCounter > flashLength *.49f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,1f);
    }
    else if(flashCounter > flashLength *.33f)
    {
```

```

        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,0f);
    }
    else if(flashCounter > flashLength *.16f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,1f);
    }
    else if(flashCounter > 0f)
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,0f);
    }
    else
    {
        enemySprite.color = new Color(enemySprite.color.r,
enemySprite.color.g, enemySprite.color.b,1f);
        flashActive = false;
    }
    flashCounter -= Time.deltaTime;
}

```

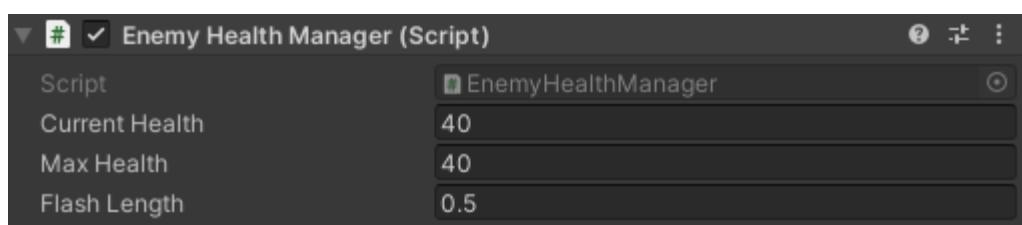
- Hàm xử lý giảm máu của kẻ thù khi bị nhân vật tấn công. Đến khi nào máu của kẻ thù nhỏ hơn hoặc bằng 0 thì sẽ bị tiêu diệt.

```

public void HurtEnemy(int damageToGive)
{
    currentHealth -= damageToGive;
    flashActive = true;
    flashCounter = flashLength;
    if(currentHealth <= 0)
    {
        Destroy(gameObject);
    }
}

```

Sau khi chạy ta vẫn có thể thay đổi thông số thông qua cửa sổ Inspector:



## Hình 24: Enemy Health Manager

### b) Di chuyển của Enemy.

Tùy thuộc vào kẻ thù chúng ta có thể thay đổi theo từng kẻ thù ko nhất thiết phải theo mặc định.

- Phát hiện nhân vật di chuyển tới và đuổi theo nhân vật:

```
public void FollowPlayer()
{
    animator.SetBool("isMoving", true);
    animator.SetFloat("moveX", (target.position.x -
transform.position.x));
    animator.SetFloat("moveY", (target.position.y - transform.position.y));
    transform.position = Vector3.MoveTowards(transform.position,
target.transform.position, speed * Time.deltaTime);

}
```

- Kẻ thù di chuyển trở về vị trí ban đầu: Khi nhân vật đi khỏi tầm nhìn hoặc 1 khoảng được cài đặt trước thì kẻ thù sẽ tự động quay trở về vị trí ban đầu mà kẻ thù được chỉ định.

```
public void GoHome()
{
    animator.SetFloat("moveX", (homePos.position.x -
transform.position.x));
    animator.SetFloat("moveY", (homePos.position.y -
transform.position.y));
    transform.position =
Vector3.MoveTowards(transform.position, homePos.position, speed *
Time.deltaTime);
    if(Vector3.Distance(transform.position, homePos.position) == 0 )
    {
        animator.SetBool("isMoving", false);
    }
}
```

- Kẻ thù không va chạm: Khi kẻ thù va chạm với vật thể khác có tag "MyWeapon", phương thức này sẽ được gọi. Nó tính toán khoảng cách giữa vị trí của kẻ thù và vị trí của vật thể khác, sau đó di chuyển để nó không chạm vào vật thể đó.

```
private void OnTriggerEnter2D(Collider2D other)
```

```

    {
        if(other.tag == "MyWeapon")
        {
            Vector2 difference = transform.position -
other.transform.position;
            transform.position = new Vector2(transform.position.x +
difference.x, transform.position.y + difference.y);
        }
    }
}

```

- Xử lý di chuyển tới nhân vật:

```

void Update()
{
    if(Vector3.Distance(target.position, transform.position) <= maxRange
&& Vector3.Distance(target.position, transform.position) >= minRange)
    {
        FollowPlayer();
    }
    else if(Vector3.Distance(target.position, transform.position) >=
maxRange)
    {
        GoHome();
    }
}

```

- c) Gây sát thương cho nhân vật:

- Hàm OnCollisionEnter2D xử lý sự kiện va chạm với đối tượng có tag là "Player". Nó gọi hàm HurtPlayer của thành phần HealthManager đích để trừ đi điểm máu của người chơi.

```

private void OnCollisionEnter2D(Collision2D other)
{
    if(other.collider.tag == "Player")
    {
        //Destroy(other.gameObject);
        //other.gameObject.SetActive(false);
        other.gameObject.GetComponent<HealthManager>().HurtPlayer(damageTo
Give);
        //reloading = true;

    }
}

```

- Hàm OnCollisionStay2D xử lý sự kiện vật thể đang va chạm với đối tượng có tag là "Player". Nó thiết lập biến isTouching là true.

```
private void OnCollisionStay2D(Collision2D other)
{
    if(other.collider.tag == "Player")
    {
        isTouching = true;
    }

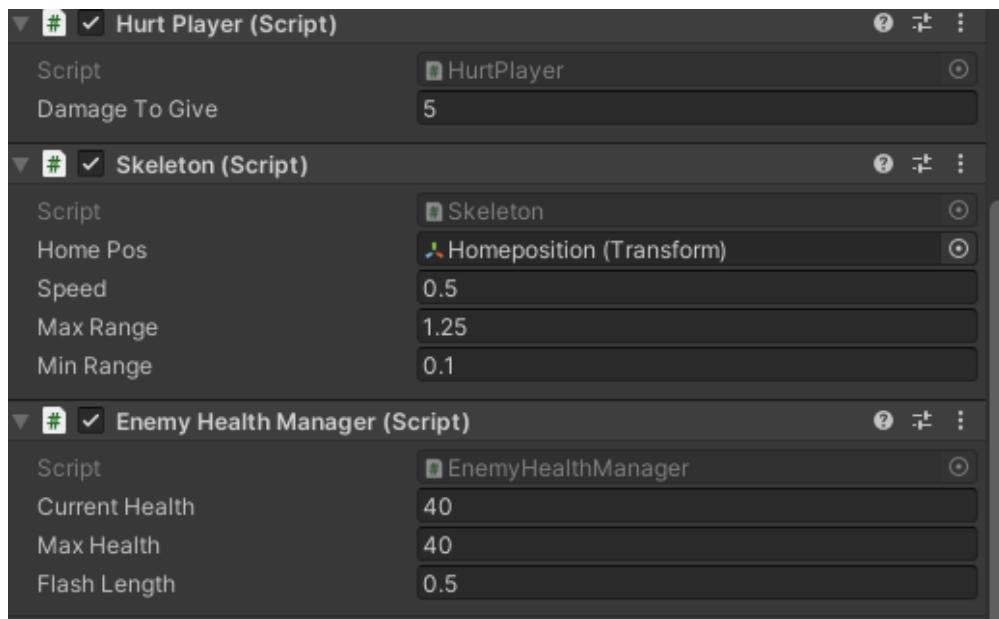
}
```

- Hàm OnCollisionExit2D xử lý sự kiện kết thúc va chạm với đối tượng có tag là "Player". Nó thiết lập biến isTouching là false và đặt lại thời gian chờ waitToHurt là 2 giây.

```
private void OnCollisionExit2D(Collision2D other)
{
    if(other.collider.tag == "Player")
    {
        isTouching = false;
        waitToHurt = 2f;
    }

}
```

Sau khi chạy ta vẫn có thể thay đổi thông số thông qua cửa sổ Inspector:

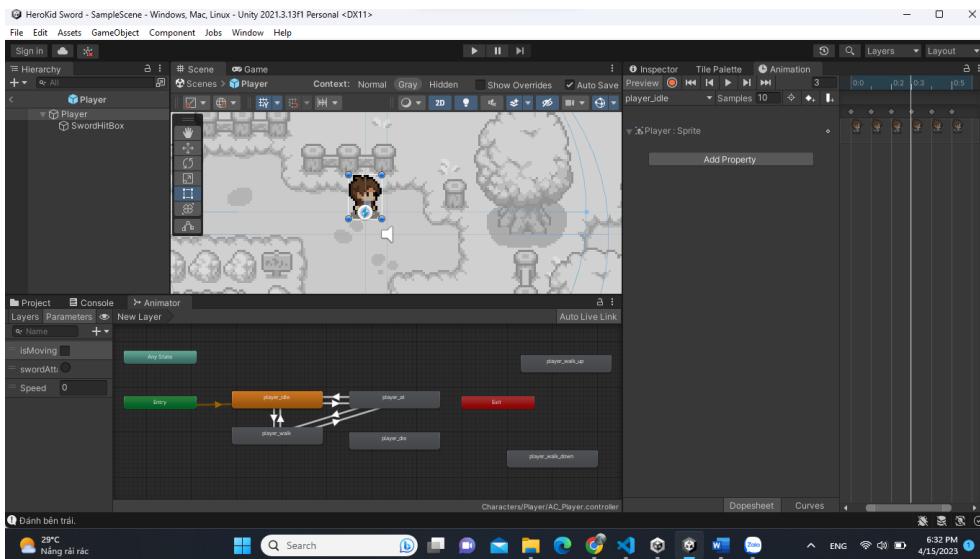


Hình 25: Các thông số có thể thay đổi ứng với các Enemy khác nhau.

#### 4.1.2. Tạo Animation, Animator cho Player và Enemy

A) Player:

Màn hình thiết kế animator và animation của Unity đối với đối tượng nhân vật chính:



Hình 26: Màn hình thiết kế hoạt ảnh của Player

a) Player\_ilde

- Là hoạt ảnh khi nhân vật đứng nghỉ ko di chuyển:



Mỗi hoạt ảnh đều khác nhau theo từng điểm ảnh, mắt thường sẽ khó nhìn thấy sự khác biệt

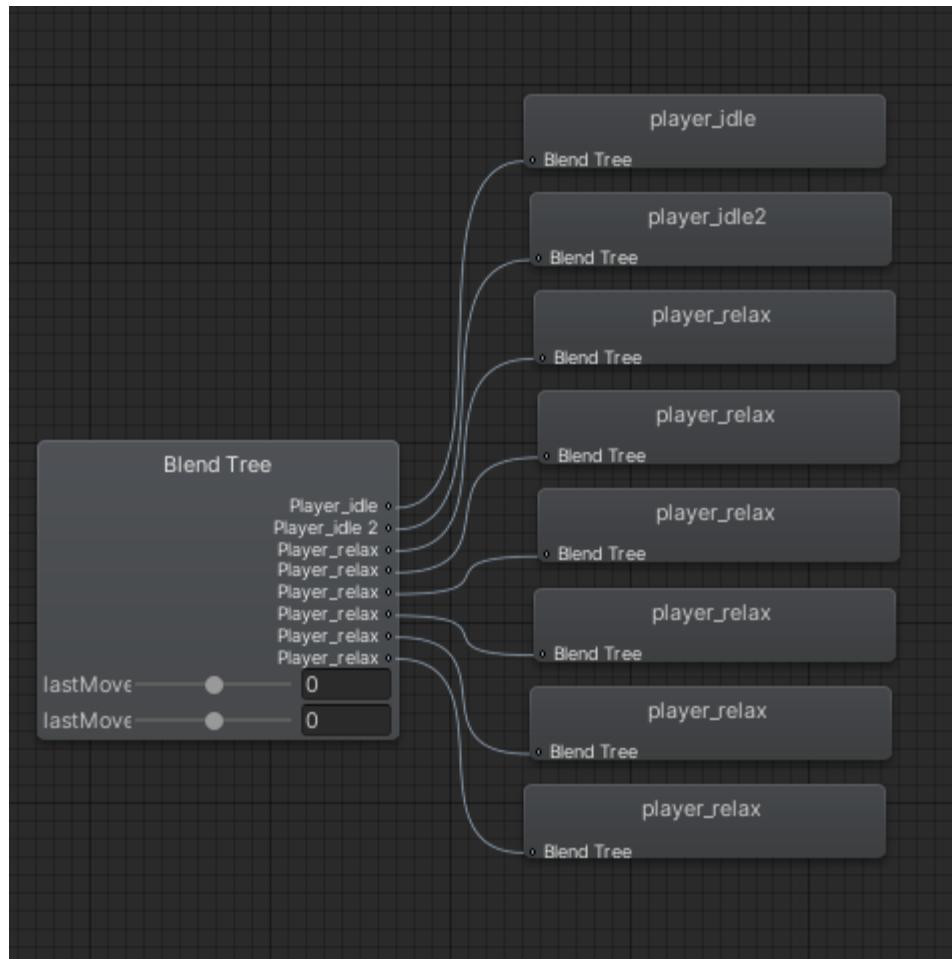


Nhân vật đứng nghỉ trước mặt



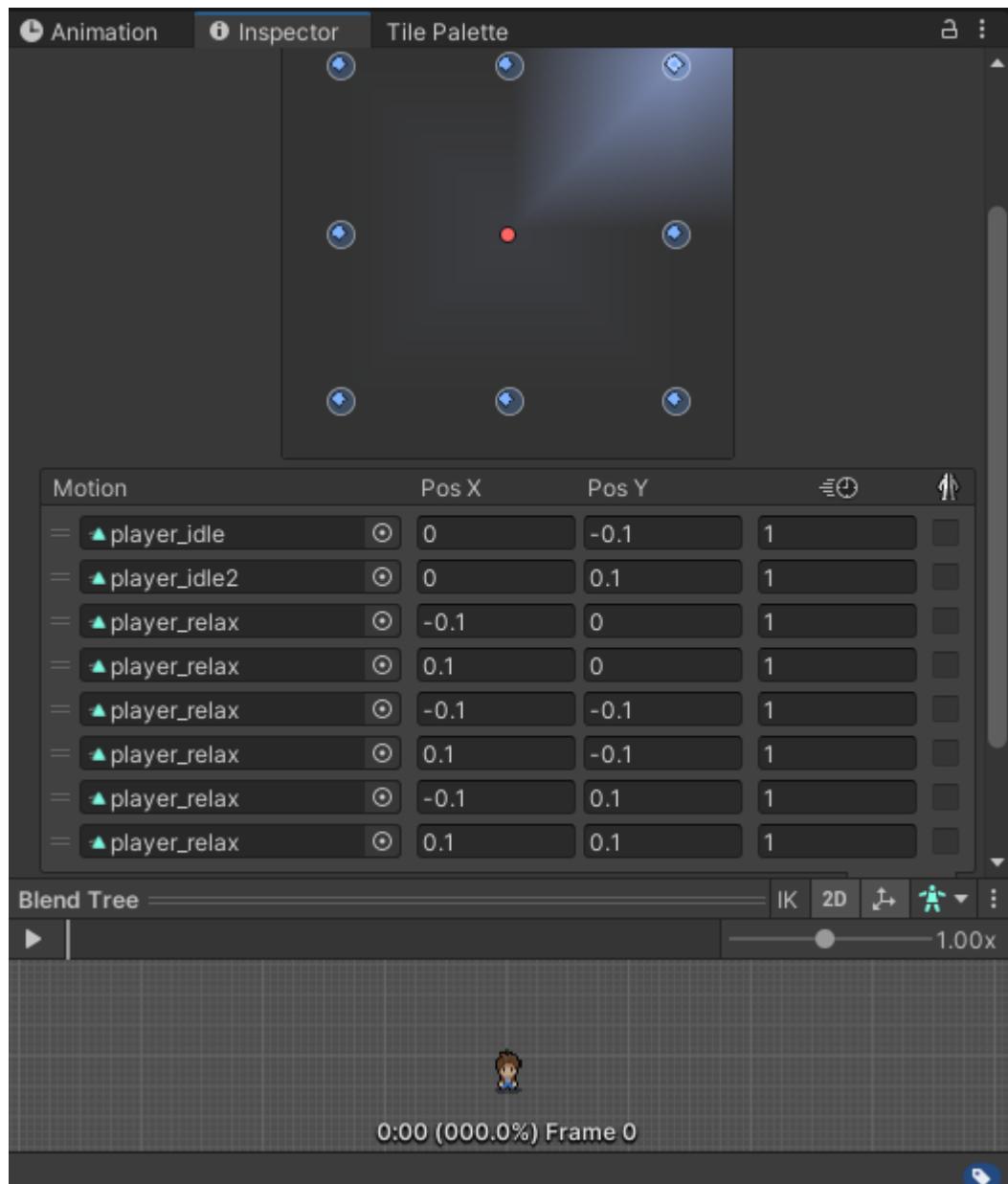
Nhân vật đứng nghỉ sau lưng:

- Kết hợp các hành động lại bằng Blend Tree:



Hình 27: Blend Tree của Player\_idle

Ở đây sẽ có 4 trạng thái nghỉ khác nhau bao gồm các phần hoạt ảnh đã nói ở trên. Được kết nối vào cây tạo trạng thái cho nhân vật.



Hình 28: Thông số Blend Tree của Player\_idle.

Chỉnh sửa thông số cho đúng với thực tế mà mình đặt tên cũng như vị trí đúng cho các hoạt động đứng nghỉ của nhân vật.

### b) Attack

- Là hoạt ảnh khi nhân vật tấn công kẻ thù.



Nhân vật tấn công cơ bản, đánh bên trái, bên phải.

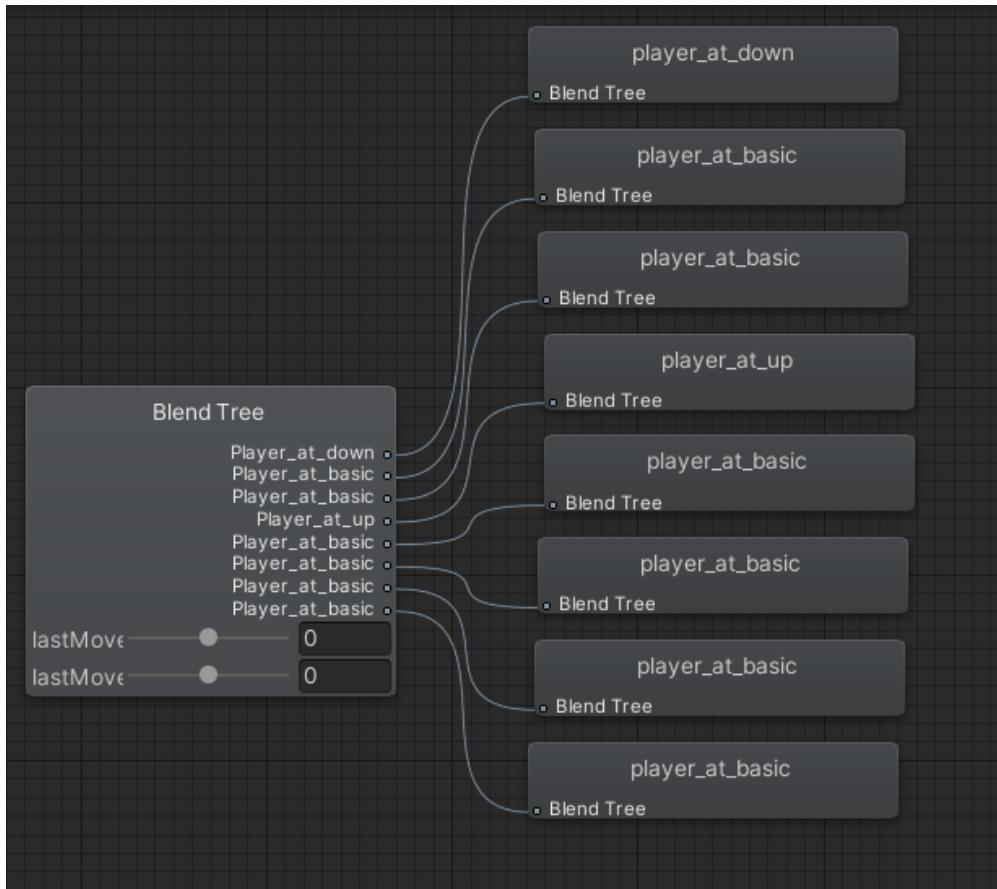


Nhân vật tấn công đằng sau.



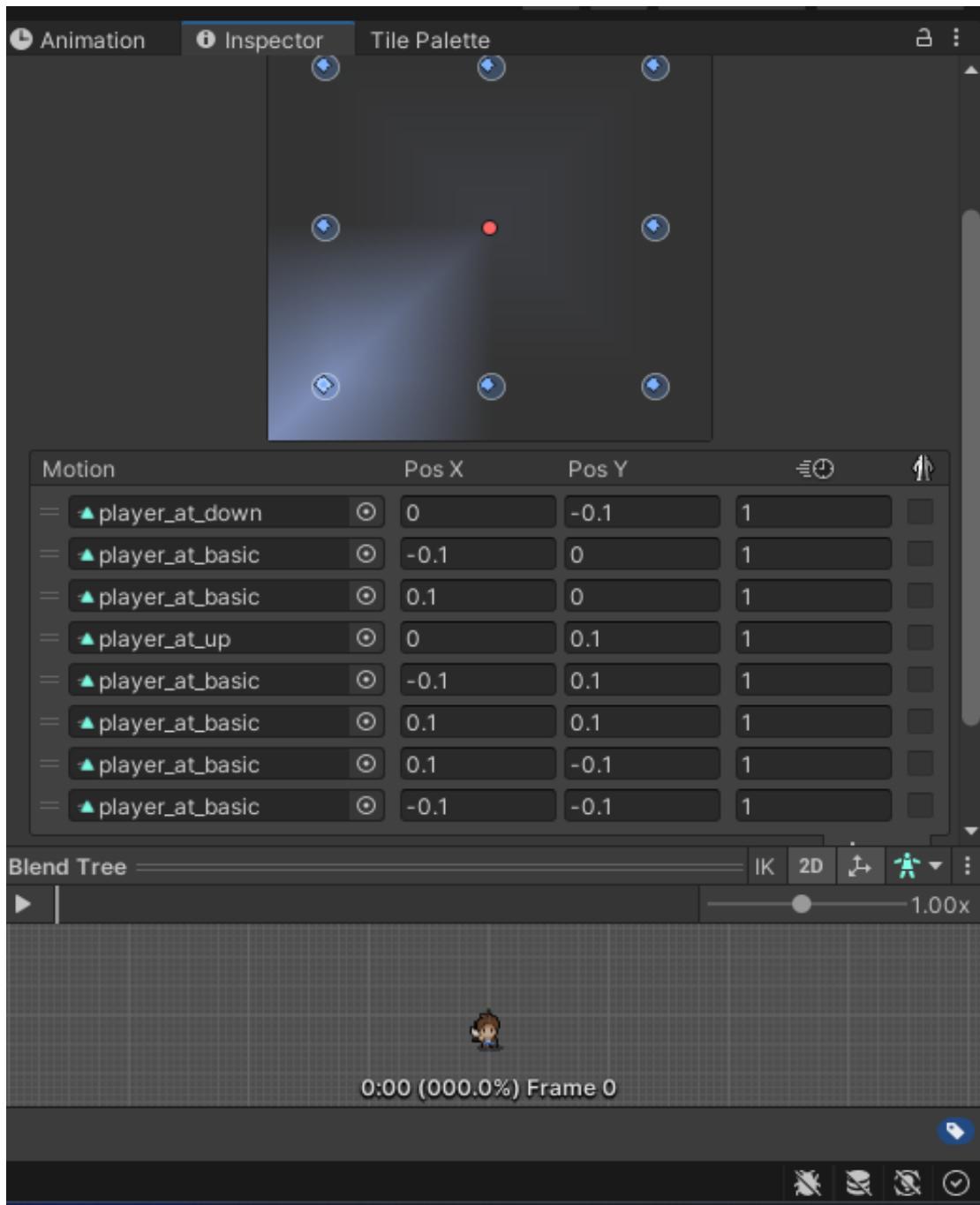
Nhân vật tấn công phía trước.

- Kết hợp các hành động lại bằng Blend Tree:



Hình 29: Blend Tree của Attack.

Có 4 trạng thái tấn công khác nhau bao gồm các phần hoạt ảnh đã nói ở trên. Được kết nối vào cây tạo trạng thái cho nhân vật tấn công kẻ thù.



Hình 30: Thông số Blend Tree của Attack.

Chỉnh sửa thông số cho đúng với thực tế mà mình đặt tên cũng như vị trí đúng cho các hoạt động tấn công của nhân vật.

### c) Player\_move

- Là hoạt ảnh khi nhân vật di chuyển:



Nhân vật di chuyển cơ bản theo hướng sang phải, sang trái.

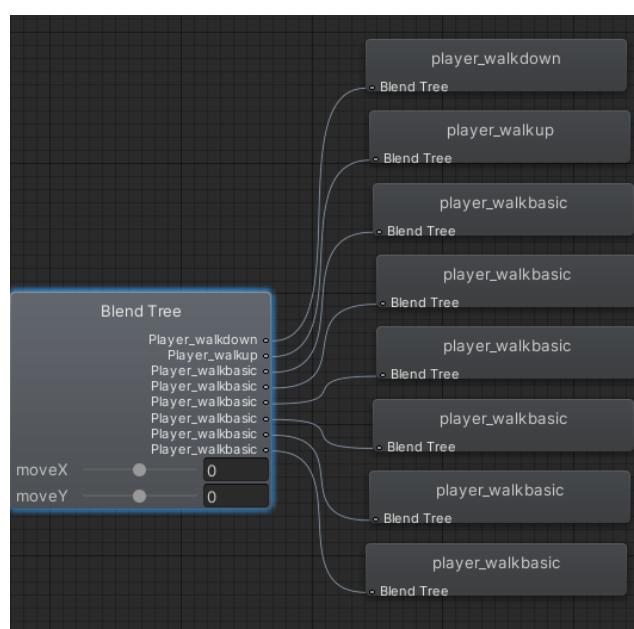


Nhân vật di chuyển lên trên.



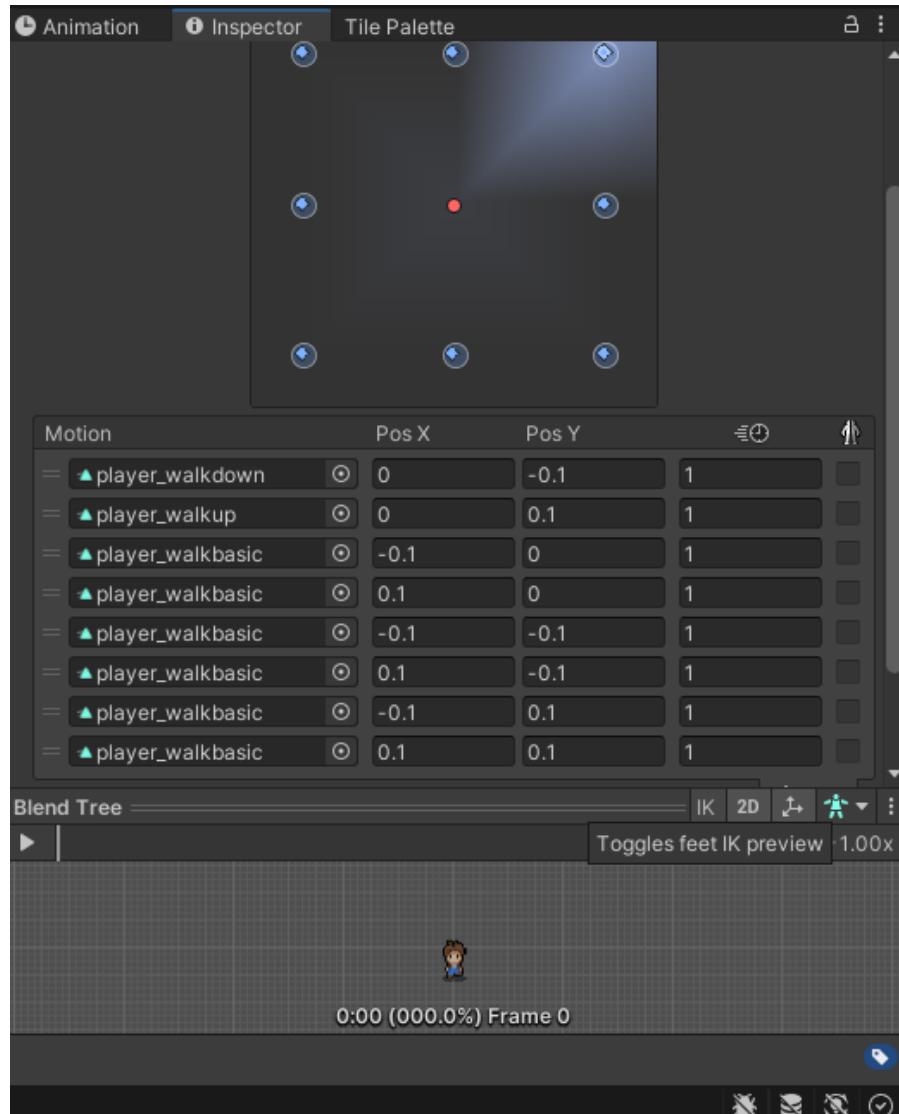
Nhân vật di chuyển xuống dưới

- Kết hợp các hành động lại bằng Blend Tree:



Hình 31: Blend Tree của Player\_move.

Có 4 trạng thái di chuyển khác nhau bao gồm các phần hoạt ảnh đã nói ở trên. Được kết nối vào cây tạo trạng thái cho nhân vật di chuyển.



Hình 32: Thông số Blend Tree của Player\_move.

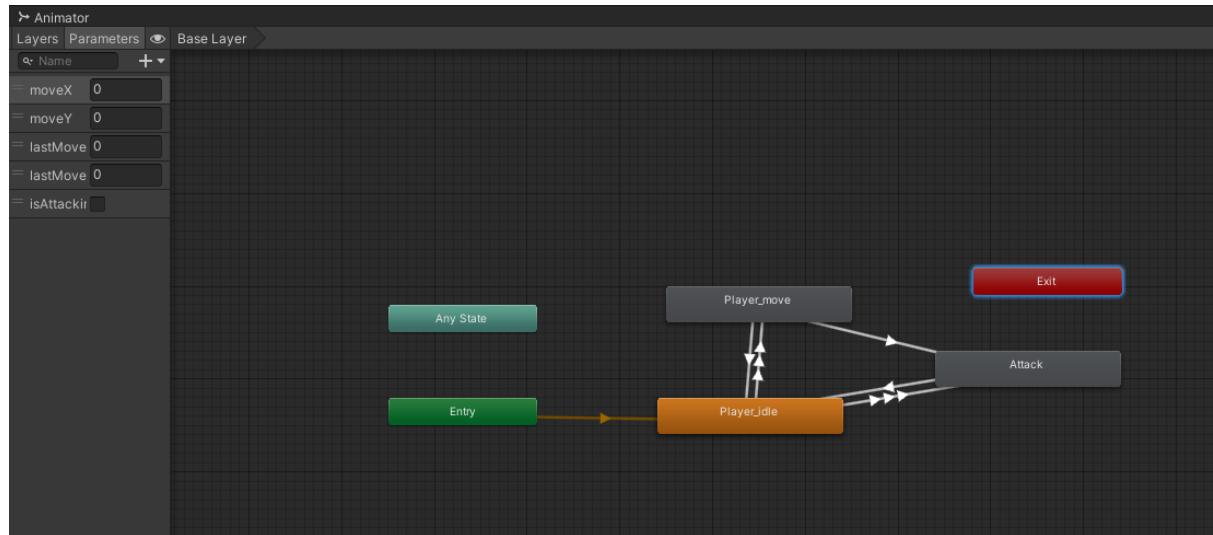
Chỉnh sửa thông số cho đúng với thực tế mà mình đặt tên cũng như vị trí đúng cho các hoạt động di chuyển của nhân vật.

d) player\_die

Hoạt ảnh nhân vật chết: Lượng máu của nhân vật về 0 và tiến vào trạng thái chết.



⇒ Kết nối các hoạt ảnh cho ra kiểu di chuyển của nhân vật:



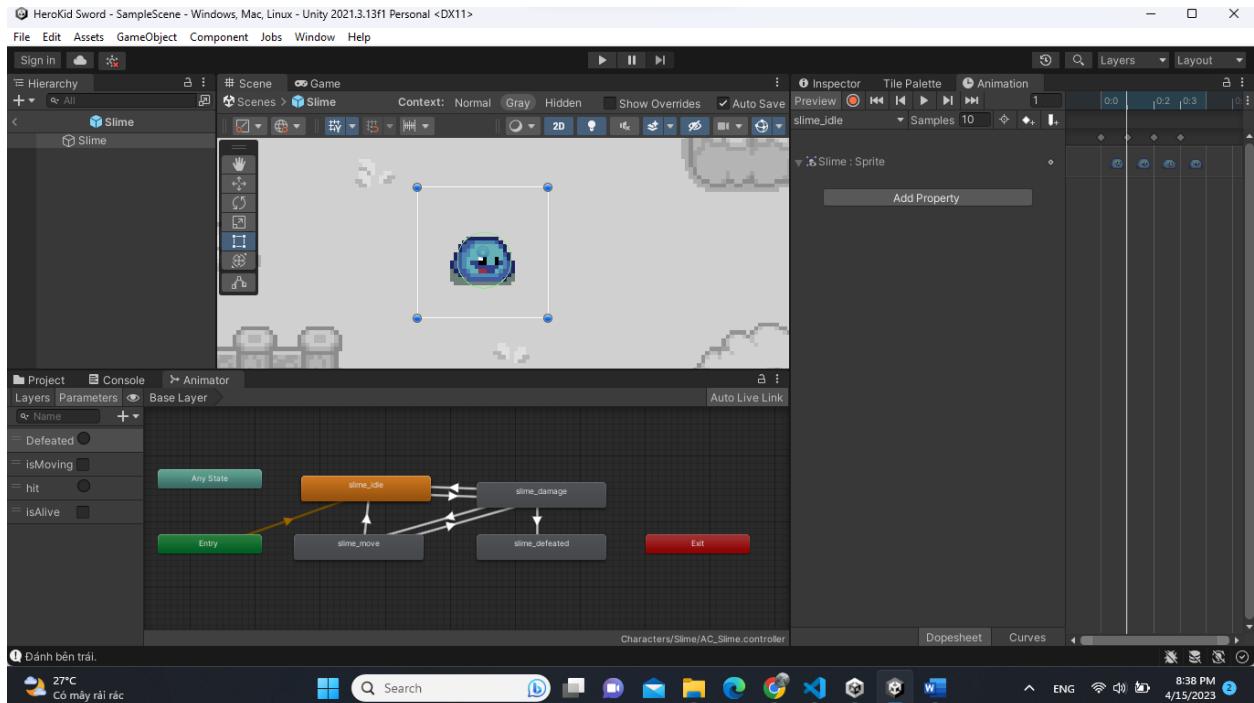
Hình 33: Mô hình hoạt ảnh của Player.

Mô hình đầy đủ của Player. Khi đã có những hoạt ảnh cần thiết sẽ kết nối chúng với nhau theo trạng thái mong muốn.

## B) Enemy

Nhìn chung khá giống với thiết kế nhân vật Hero ở trên.

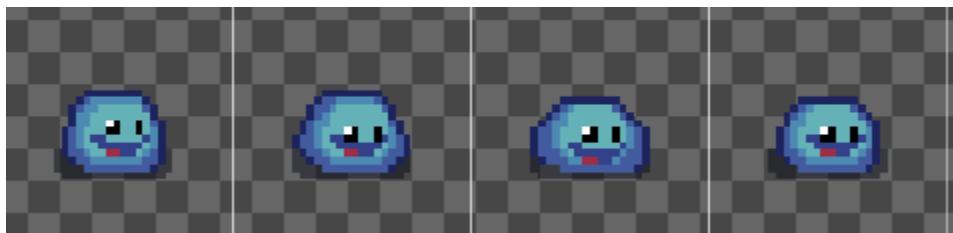
1) Thiết kế của Slime:



Hình 34: Màn hình thiết kế hoạt ảnh của Slime

a) slime\_idle

Là hoạt ảnh slime đứng nghỉ.



Mỗi hình ảnh ứng với mỗi vị trí, khi kết hợp các hình theo trình tự và chạy chúng sẽ ra một chuyển động đứng nghỉ.

b) slime\_move



Là hoạt ảnh slime di chuyển, slime di chuyển rất đơn giản chỉ nhảy lên nhảy xuống cũng như tấn công nhân vật.

c) slime\_hit



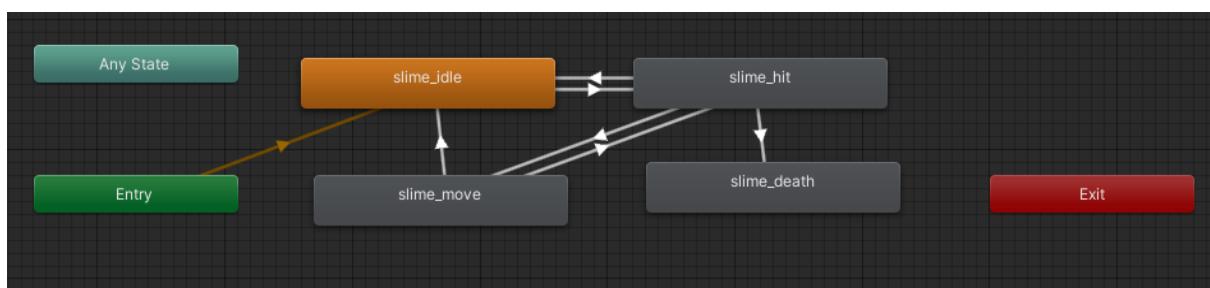
Là hoạt ảnh slime bị nhân vật tấn công.

d) slime\_death



Là hoạt ảnh slime bị nhân vật kết liễu.

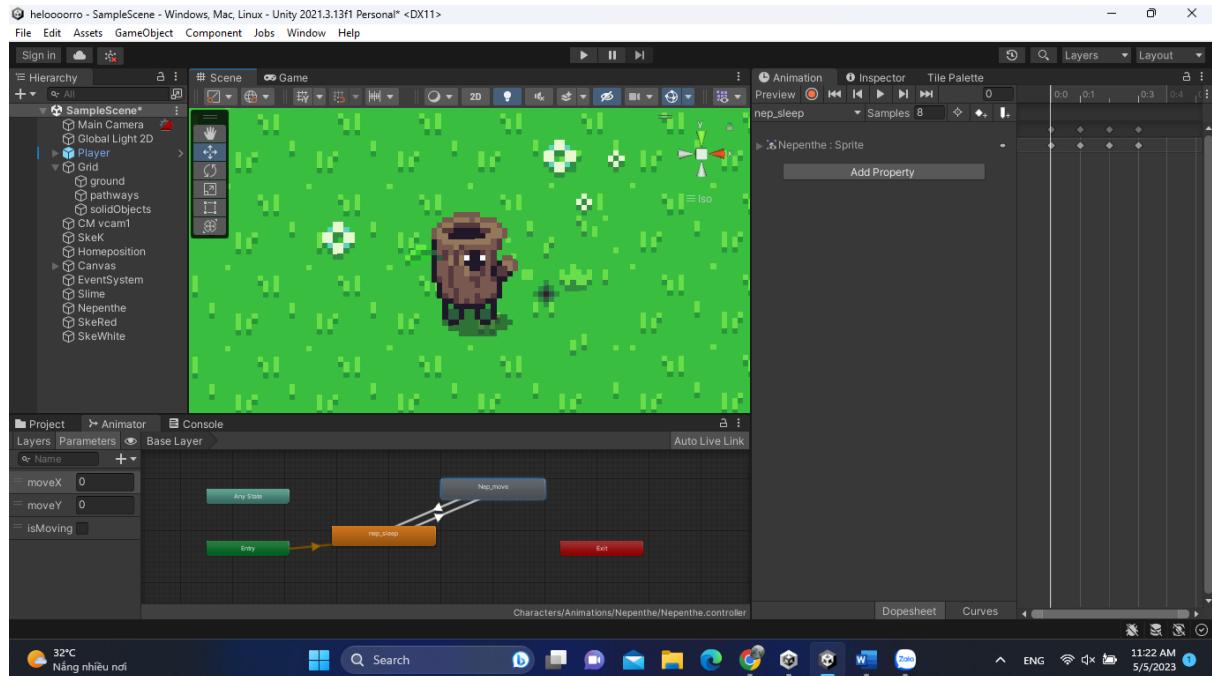
⇒ Kết nối các hoạt ảnh cho ra kiểu di chuyển của Slime:



Hình 35: Mô hình hoạt ảnh của Slime.

Mô hình đầy đủ của Slime. Khi đã có những hoạt ảnh cần thiết sẽ kết nối chúng với nhau theo trạng thái mong muốn.

## 2) Thiết kế của Nepenthe



Hình 36: Màn hình thiết kế hoạt ảnh của Nepenthe

### a) Nep\_move

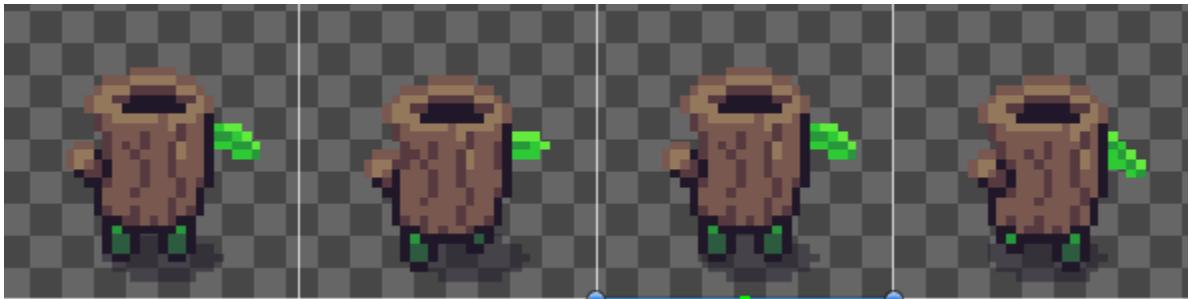
- Là hoạt ảnh Nepenthe di chuyển:



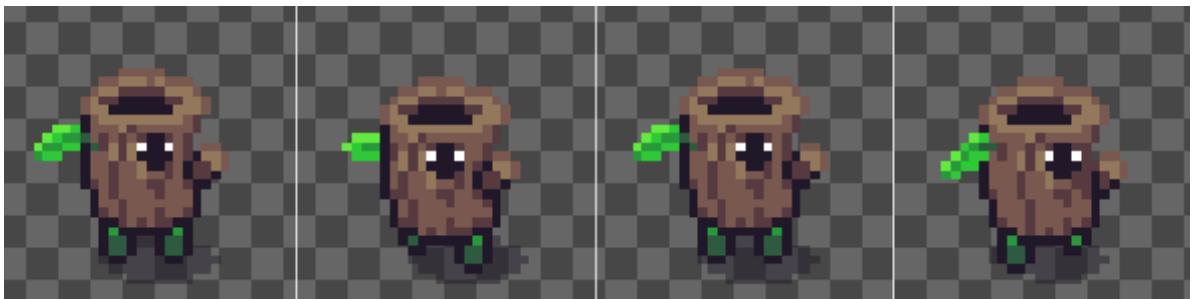
Nepenthe di chuyển sang trái.



Nepenthe di chuyển sang phải.

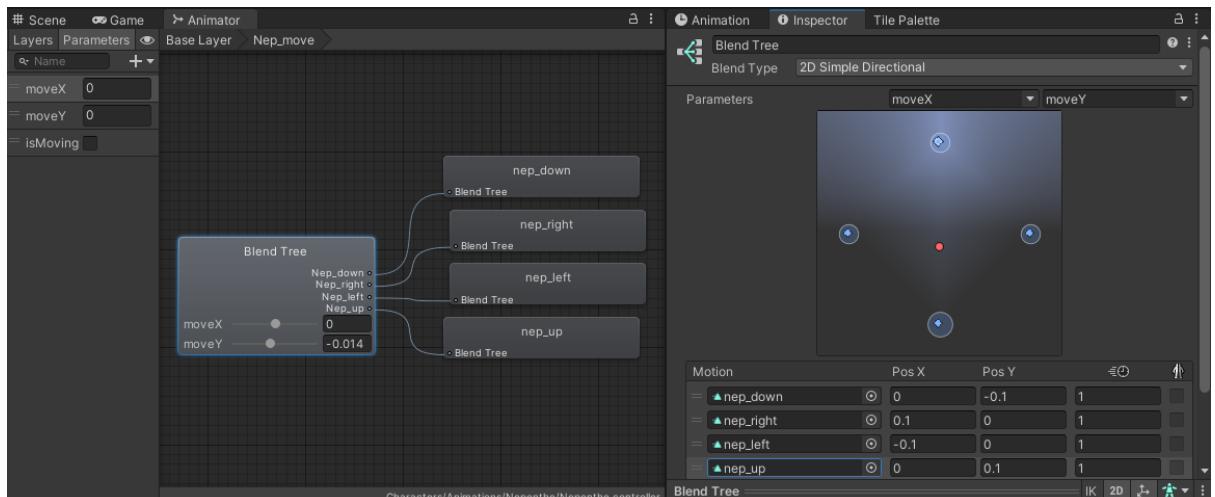


Nepenthe di chuyển lên trên.



Nepenthe di chuyển xuống dưới.

- Kết hợp các hành động lại bằng Blend Tree:



Hình 37: Cây và thông số Blend Tree của Nep\_move

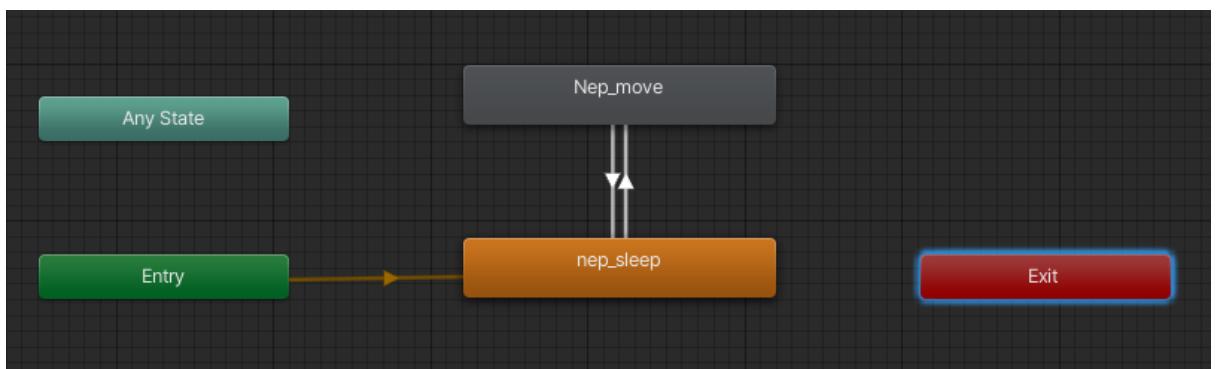
Bao gồm cả hai cửa sổ thao tác, phần này giúp thay đổi thông số dễ dàng hơn, chỉnh thông số vị trí ứng với các hoạt ảnh đã tạo trước đó.

b) nep\_sleep



Nepenthe bước vào trạng thái ngủ của mình.

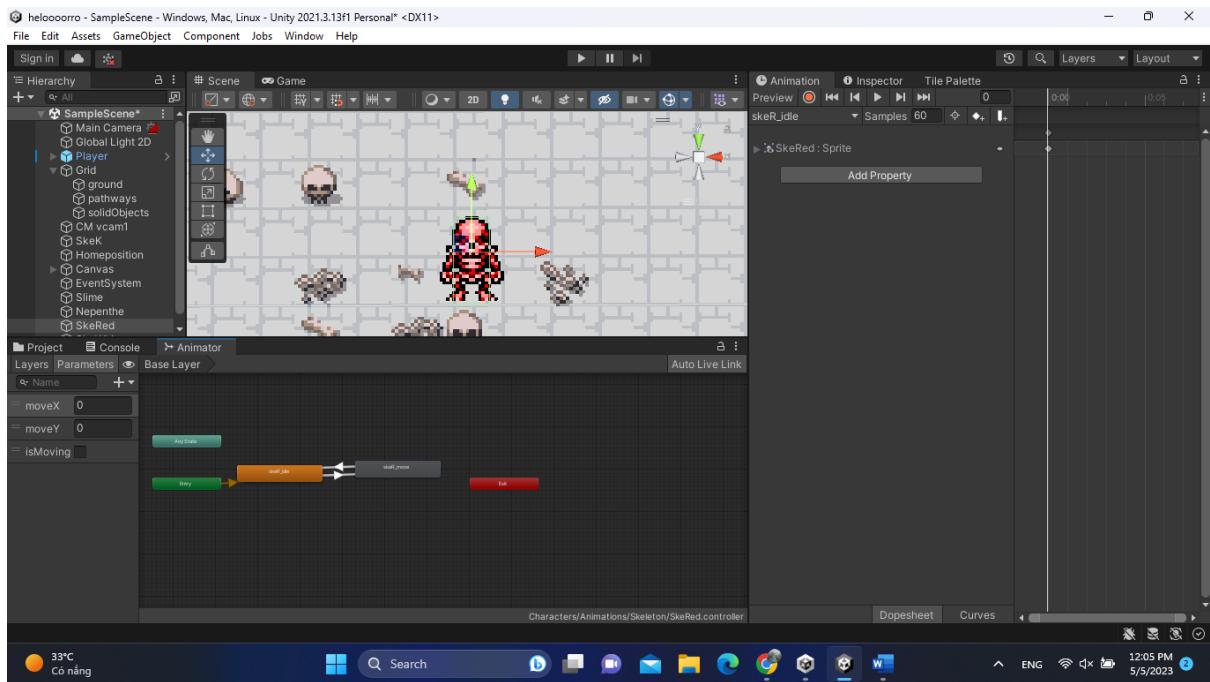
⇒ Kết nối các hoạt ảnh cho ra kiểu di chuyển của Nepenthe:



Hình 38: Mô hình hoạt ảnh của Nepenthe

Mô hình đầy đủ của Nepenthe. Khi đã có những hoạt ảnh cần thiết sẽ kết nối chúng với nhau theo trạng thái mong muốn.

### 3) Thiết kế Skeleton Red



Hình 39: Màn hình thiết kế hoạt ảnh của Skeleton Red

### a) skeR\_move

- Là các hoạt ảnh di chuyển của bộ xương:



Bộ xương di chuyển xuống dưới:



Bộ xương di chuyển lên trên.

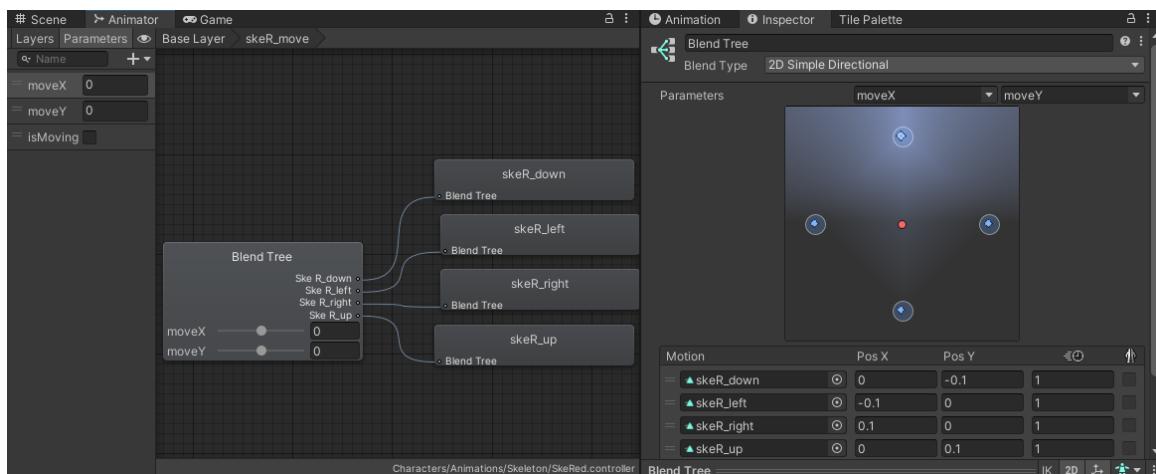


Bộ xương di chuyển sang trái.



Bộ xương di chuyển sang phải.

- Kết hợp các hành động lại bằng Blend Tree:



Hình 40: Cây và thông số Blend Tree của skeR\_move

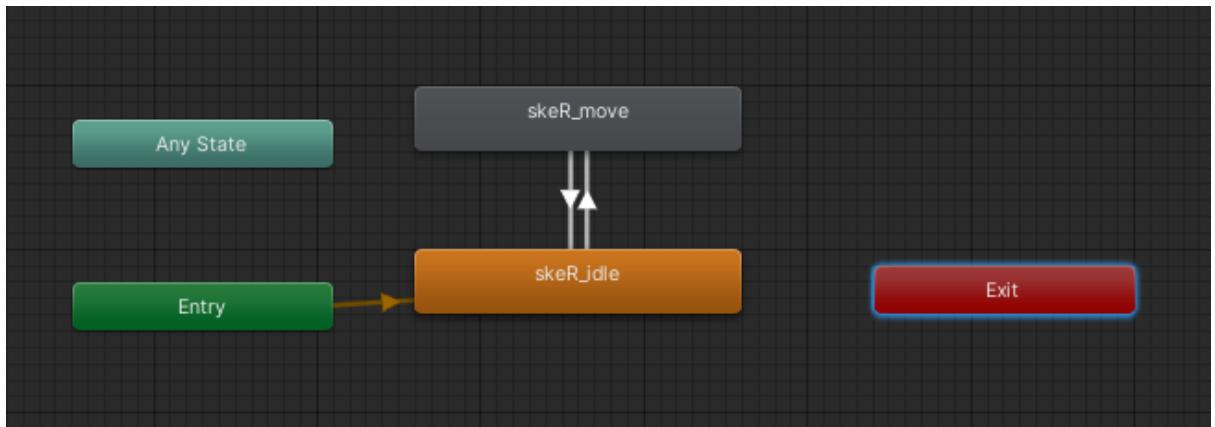
Bao gồm cả hai cửa sổ thao tác, phần này giúp thay đổi thông số dễ dàng hơn, chỉnh thông số vị trí ứng với các hoạt ảnh đã tạo trước đó.

b) skeR\_idle

Là hoạt ảnh khi bộ xương đứng nghỉ ko di chuyển:



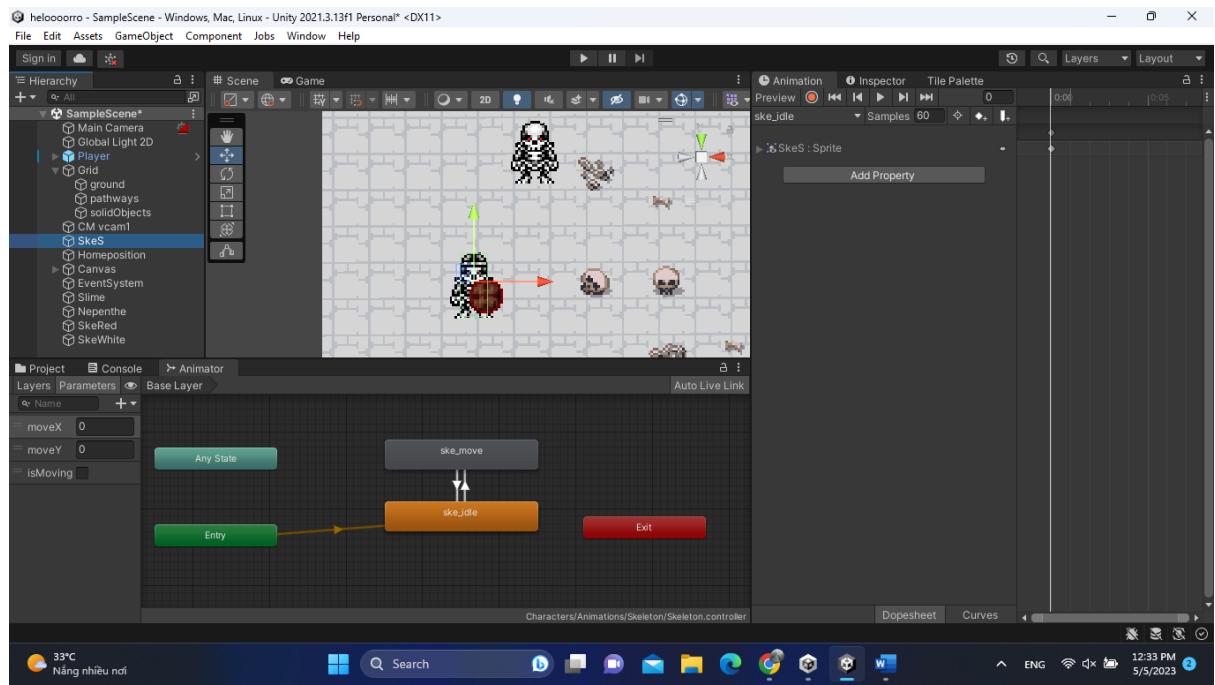
⇒ Kết nối các hoạt ảnh sẽ cho ra cách di chuyển của bộ xương.



Hình 41: Mô hình hoạt ảnh của Skeletion Red

Mô hình đầy đủ của Skeletion Red được đặt ở cửa sổ Animator. Khi đã có những hoạt ảnh cần thiết sẽ kết nối chúng với nhau theo trạng thái mong muốn.

#### 4) Thiết kế Skeletion Shield



Hình 42: Màn hình thiết kế hoạt ảnh của Skeleton Shield

### a) ske\_move

- Là các hoạt ảnh di chuyển của bộ xương:



Bộ xương di chuyển xuống dưới.



Bộ xương di chuyển lên trên.

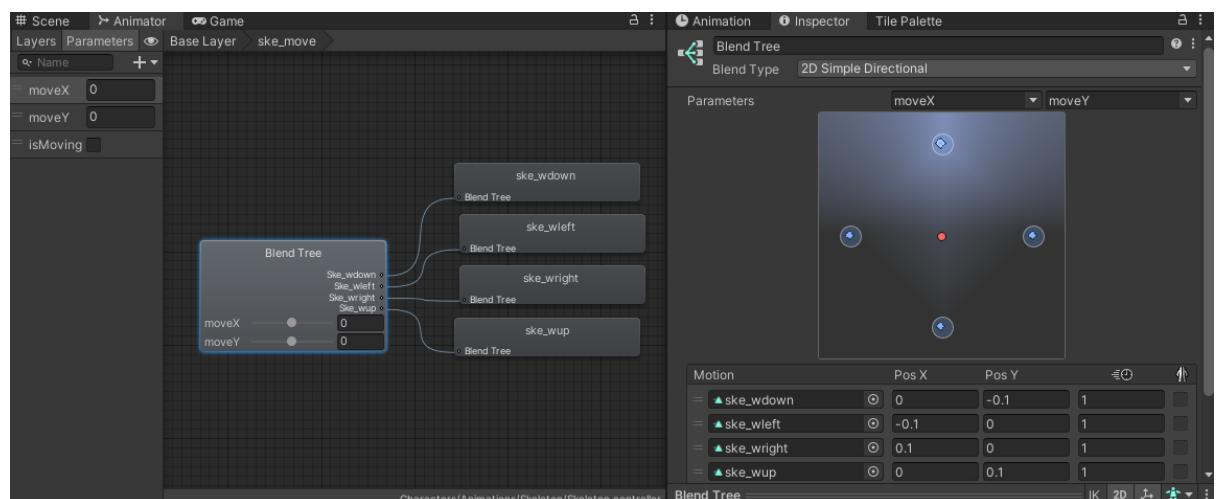


Bộ xương di chuyển sang trái.



Bộ xương di chuyển sang phải.

- Kết hợp các hành động lại bằng Blend Tree:



Hình 43: Cây và thông số Blend Tree của ske\_move

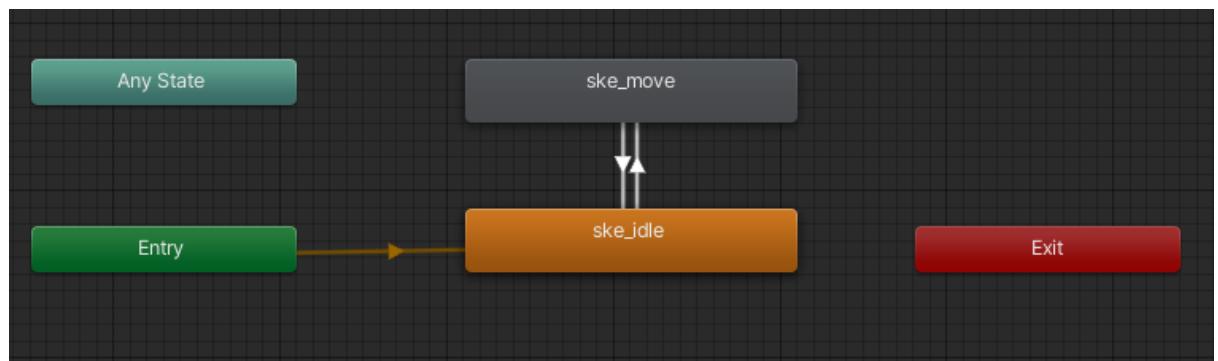
Bao gồm cả hai cửa sổ thao tác, phần này giúp thay đổi thông số dễ dàng hơn, chỉnh thông số vị trí ứng với các hoạt ảnh đã tạo trước đó.

b) ske\_idle



Là hoạt ảnh khi bộ xương đứng nghỉ ko di chuyển:

⇒ Kết nối các hoạt ảnh sẽ cho ra cách di chuyển của bộ xương.

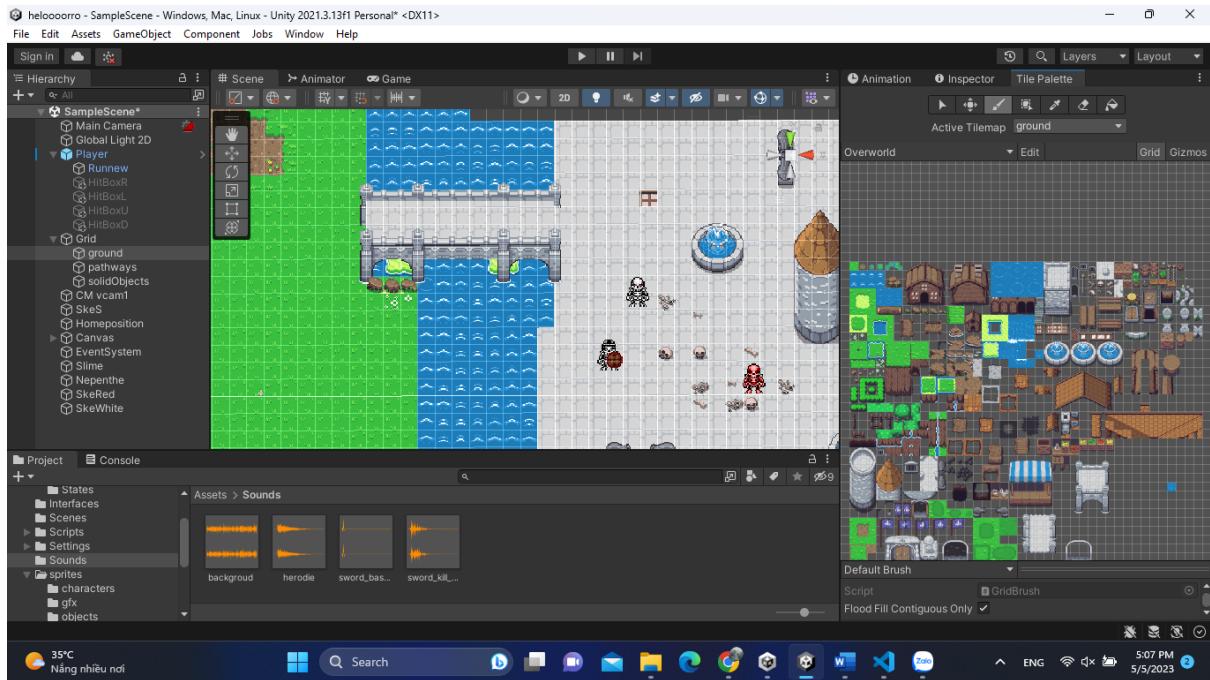


Hình 44: Mô hình hoạt ảnh của Skeleton Shield

Mô hình đầy đủ của Skeleton Shield được đặt ở cửa sổ Animator. Khi đã có những hoạt ảnh cần thiết sẽ kết nối chúng với nhau theo trạng thái mong muốn.

#### 4.1.3. Thiết kế TileMap, Canvas

a) TileMap:

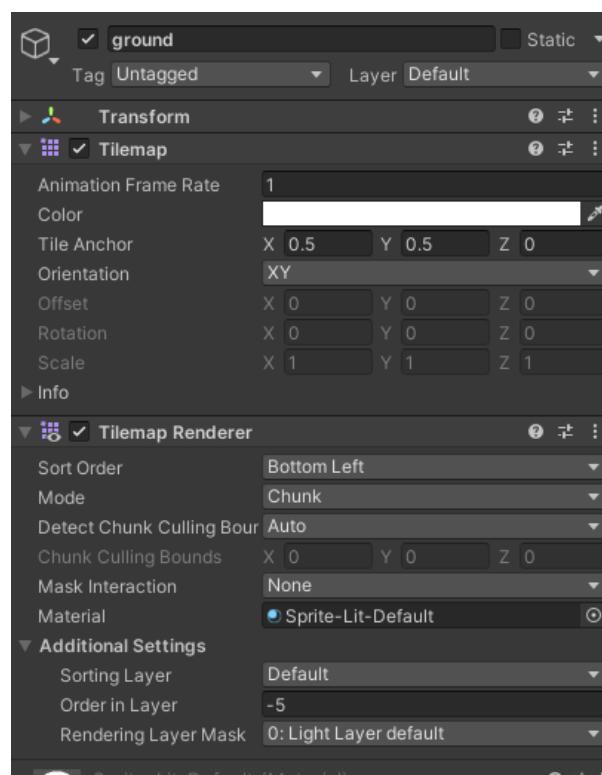


Hình 45: Màn hình thiết kế TileMap

Thiết kế này có 3 tầng TileMap khác nhau bao gồm:

- Tầng ground (đất): đây là tầng dùng để tạo đất, bao phủ toàn bộ khu vực chơi game. Tầng này có thể chứa các tileset như cỏ, đá, bụi cây,...

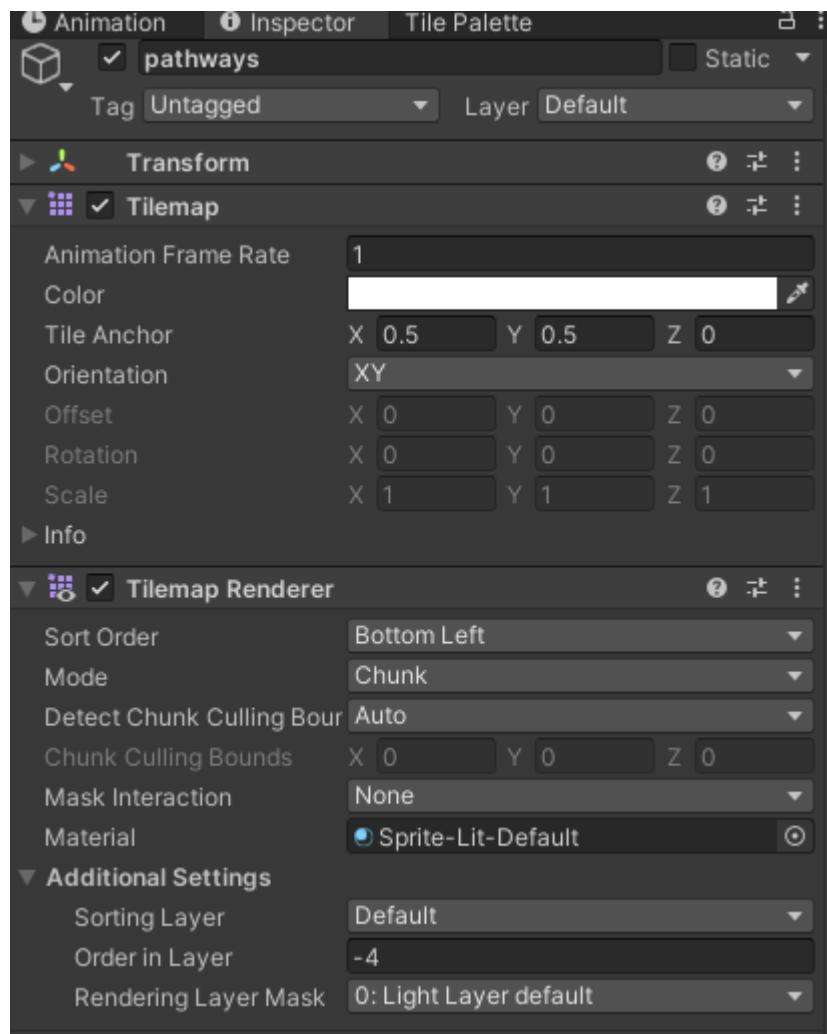
Order in Layer gán bằng -5.



Hình 46: Thông số TileMap ground

- Tầng pathways (đường đi): đây là tầng dùng để tạo các con đường đi trong game. Tầng này có thể chứa các tileset như đường đi, bậc thang, cầu,...

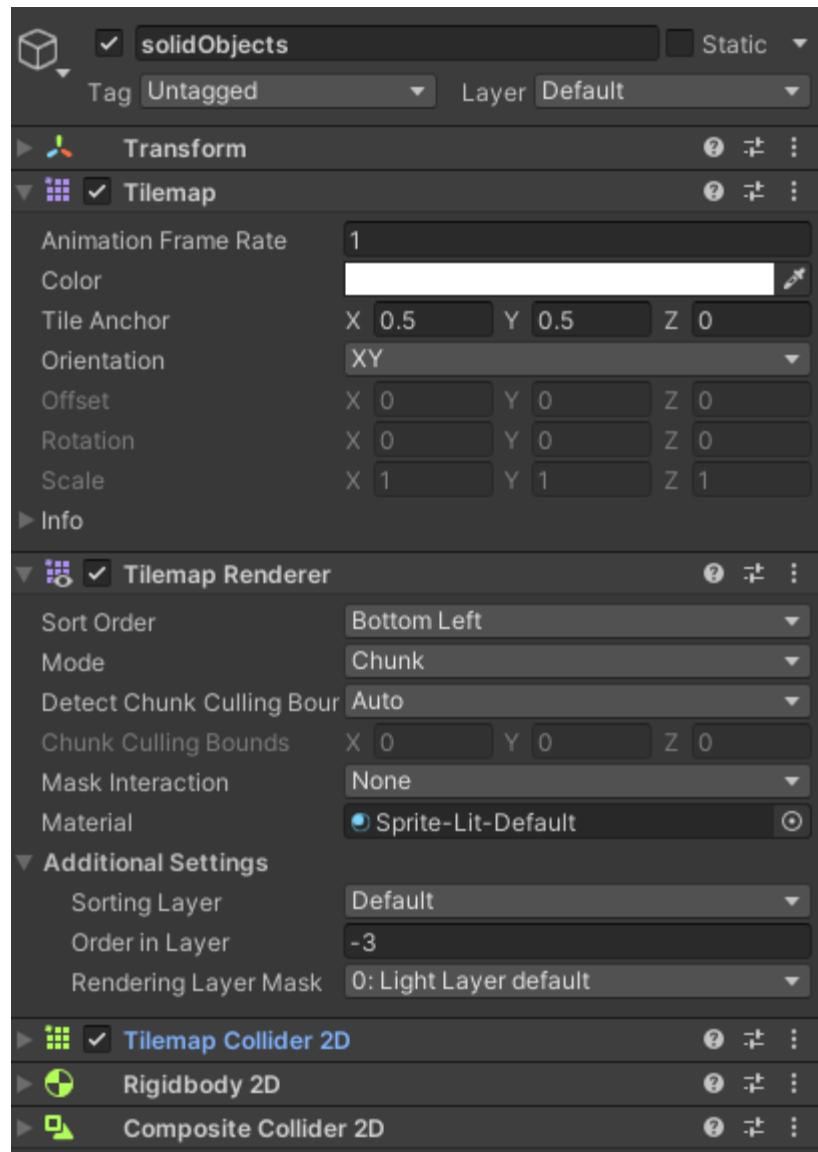
Order in Layer gán bằng -4.



Hình 47: Thông số của TileMap pathways

- Tầng solidObjects (vật thể cố định): đây là tầng dùng để tạo các vật thể như tường, tảng đá, cây cối... những vật thể này sẽ không thể di chuyển được.

Order in Layer gán bằng -3.



Hình 48: Thông số của TileMap solidObjects

- Tại TileMap này có thêm Tilemap Collider 2D và Rigidbody 2D để thực hiện xử lý va chạm với các game object khác trong scene.

Tilemap Collider 2D được sử dụng để định nghĩa hình dạng va chạm của TileMap. Nó giúp định nghĩa vùng không gian 2D mà các đối tượng khác không thể xuyên qua hoặc va chạm với các tile trong TileMap.

Rigidbody 2D giúp xử lý động lực và vật lý trên TileMap. Nó cung cấp các thuộc tính như khối lượng, tốc độ và lực hấp dẫn để giả lập các tác động vật lý

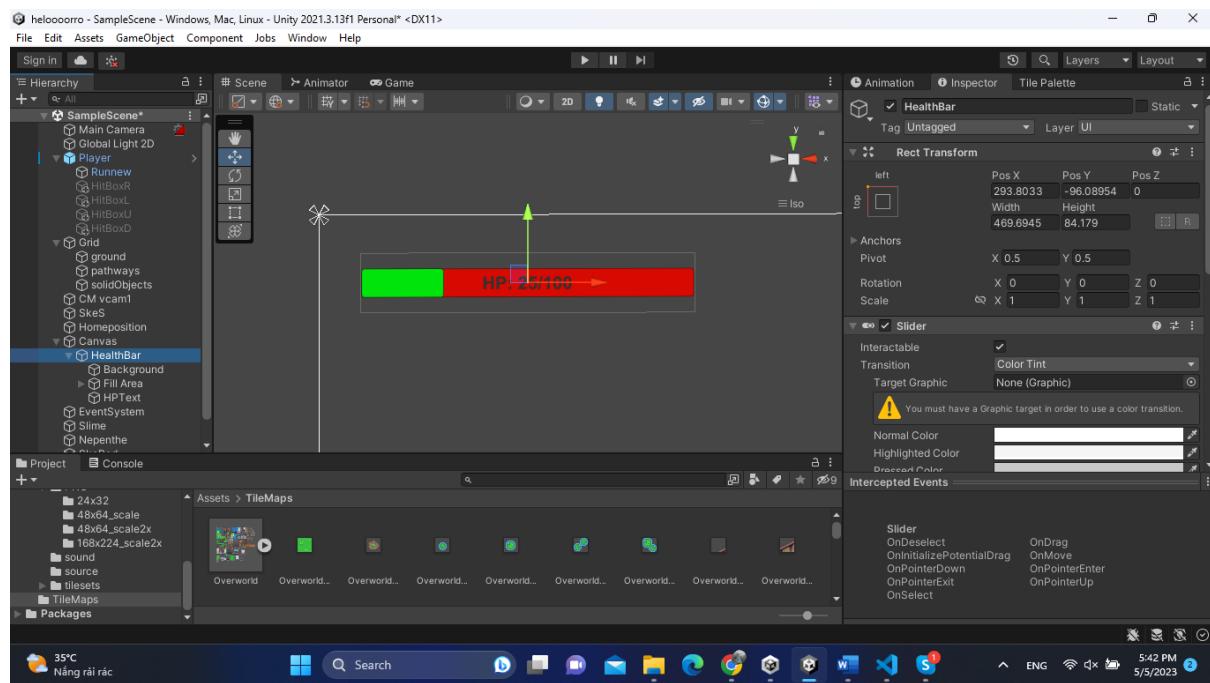
như lực đẩy, lực kéo và trọng lực. Rigidbody 2D được sử dụng để giúp xử lý va chạm giữa TileMap và các đối tượng di động như nhân vật chính, kẻ địch hoặc các vật phẩm trong game.

Các tầng TileMap này có thể được sử dụng để tạo môi trường chơi game đa dạng và phong phú hơn. Nó cũng giúp cho việc xử lý va chạm giữa các đối tượng trong game được dễ dàng hơn.

### b) Canvas

Trong canvas của game này được sử dụng Health Bar là một thành phần của giao diện người dùng (UI) trong trò chơi. Nó được sử dụng để hiển thị lượng máu còn lại của người chơi hoặc các nhân vật khác trong trò chơi.

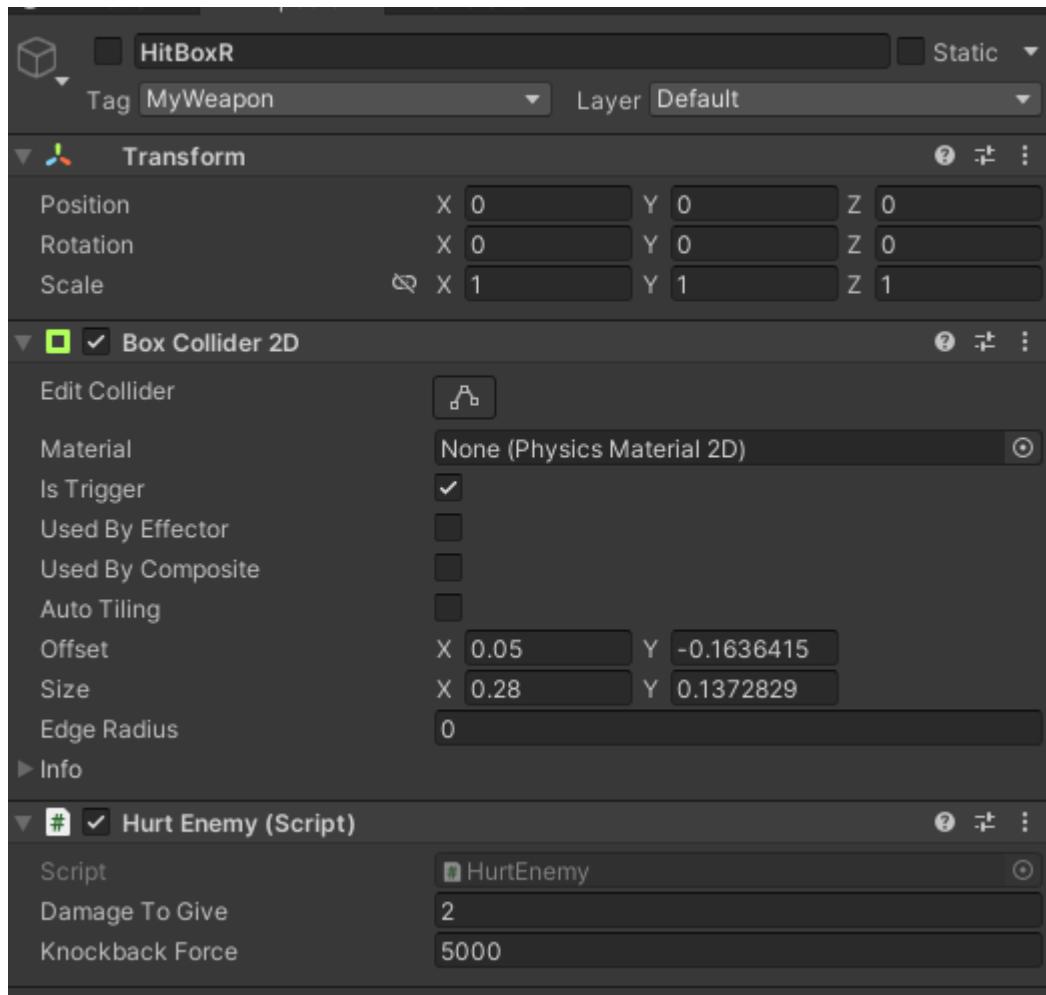
Health Bar được đặt ở một góc trên màn hình để người chơi có thể theo dõi thông tin về sức khỏe của nhân vật của mình trong khi chơi.



Hình 49: Màn hình thiết kế Canvas

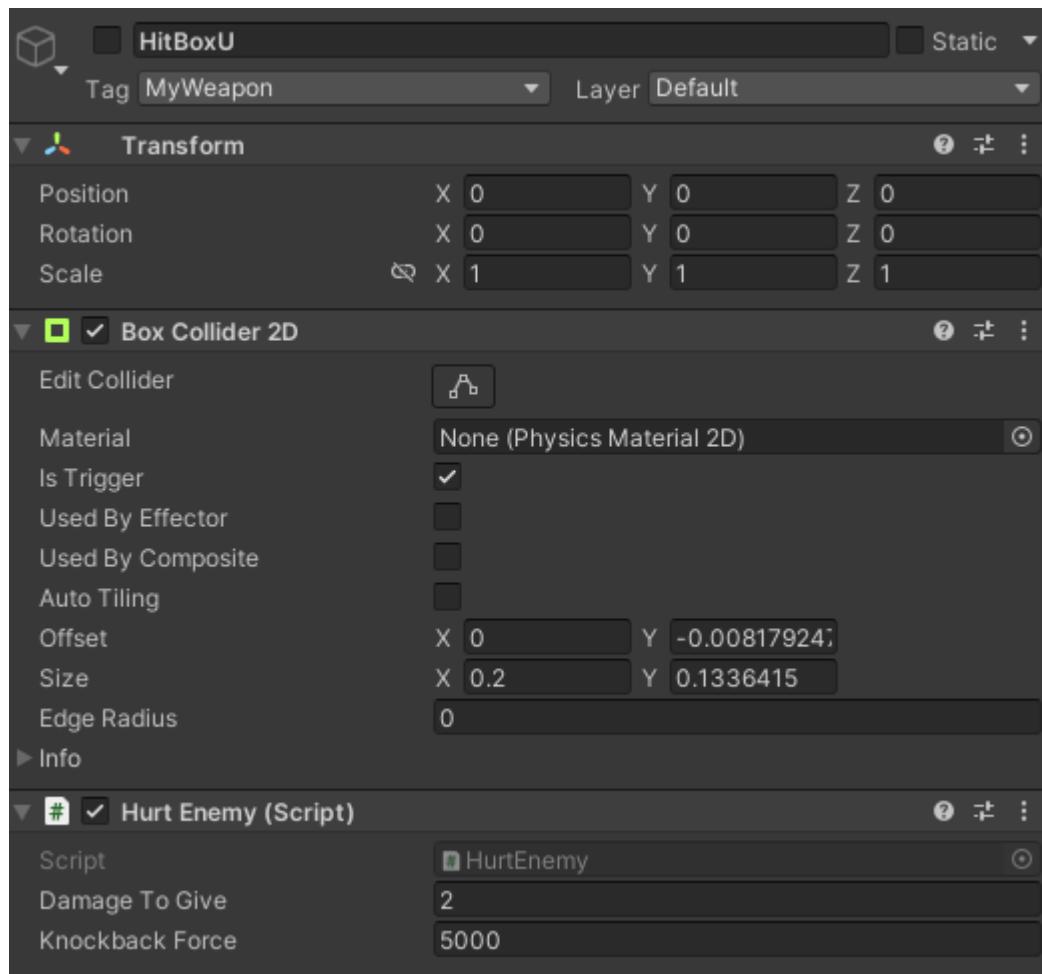
#### 4.1.4. Kỹ thuật xử lý va chạm vật lý (Collision)

##### a) Box Collider 2D



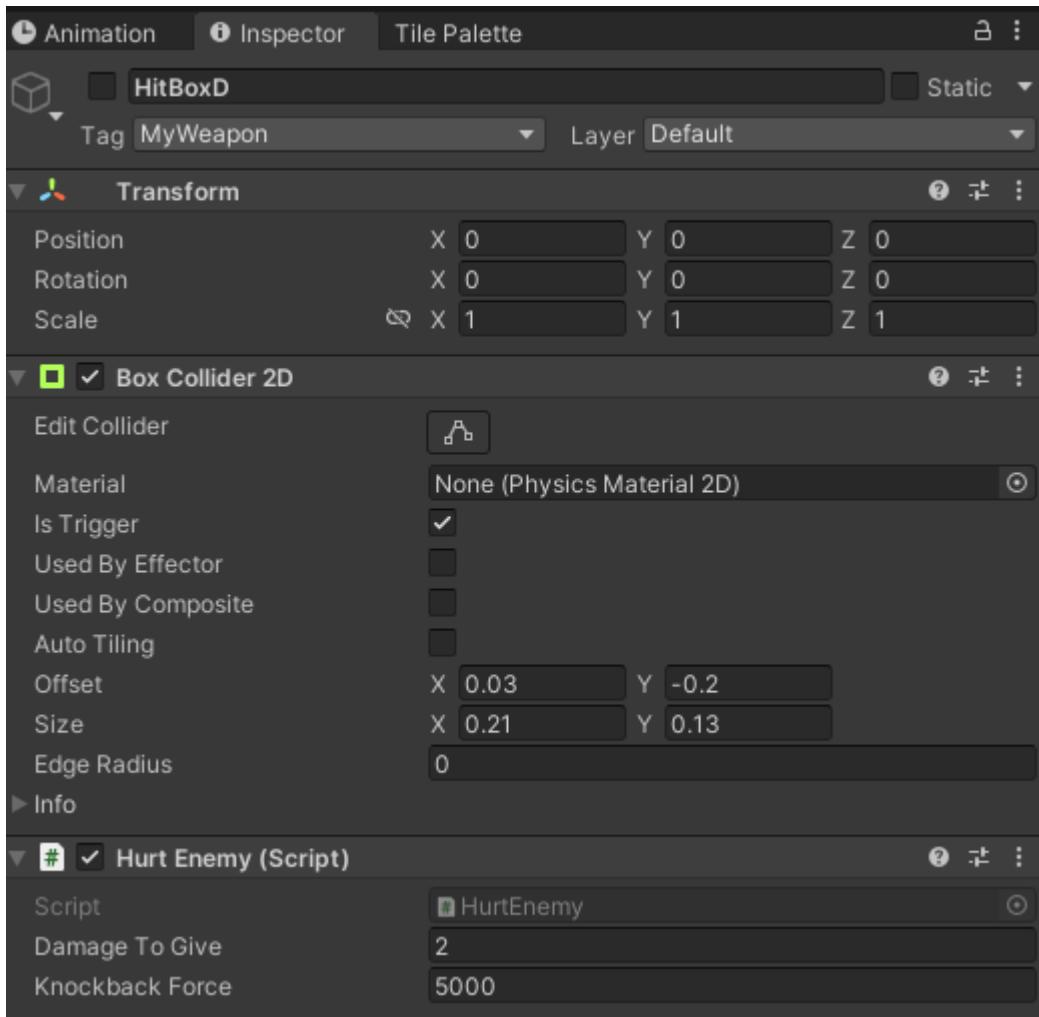
Hình 50: Thông số thuộc tính tấn công theo hai hướng trái phải.

Thể hiện một khối va chạm giữa thanh kiếm và enemy theo hai hướng trái phải.



*Hình 51: Thông số thuộc tính tấn công phía trên.*

Thể hiện một khối va chạm giữa thanh kiếm và enemy theo hướng phía trên.



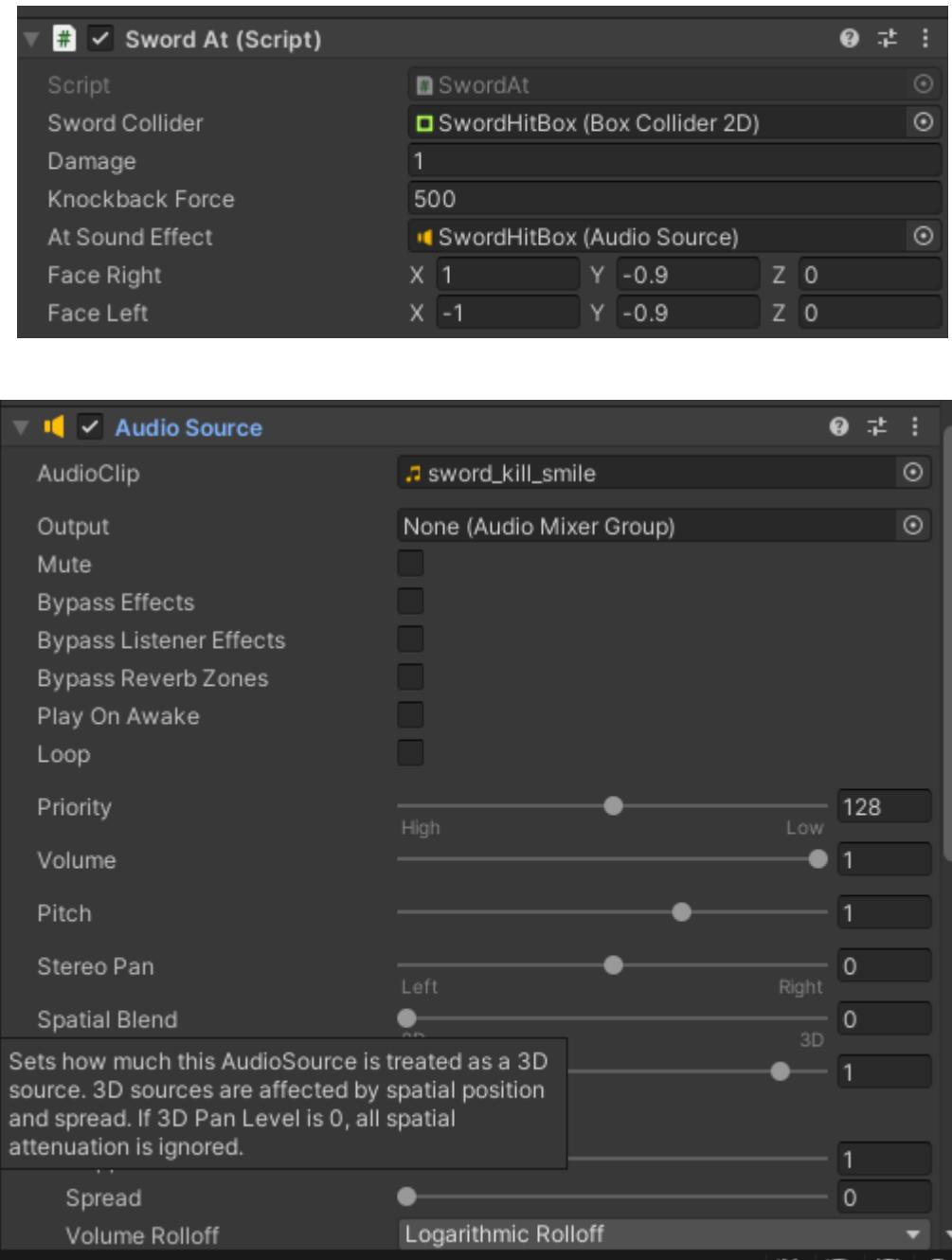
Hình 52: Thông số thuộc tính tấn công phía dưới.

Thể hiện một khối va chạm giữa thanh kiếm và enemy theo hướng phía dưới.

Box Collider 2D là một thành phần trong game engine Unity được sử dụng để phát hiện va chạm giữa các đối tượng 2D. Nó được sử dụng cùng với thành phần Rigidbody 2D để cung cấp các thuộc tính vật lý cho đối tượng như khối lượng và trọng lực.

Box Collider 2D tạo ra một hình chữ nhật bao quanh hình dạng hoặc sprite của đối tượng. Kích thước của collider có thể được điều chỉnh để phù hợp với hình dạng cụ thể của đối tượng. Khi hai đối tượng có thành phần Box Collider 2D va chạm, Unity phát hiện va chạm và kích hoạt các sự kiện hoặc

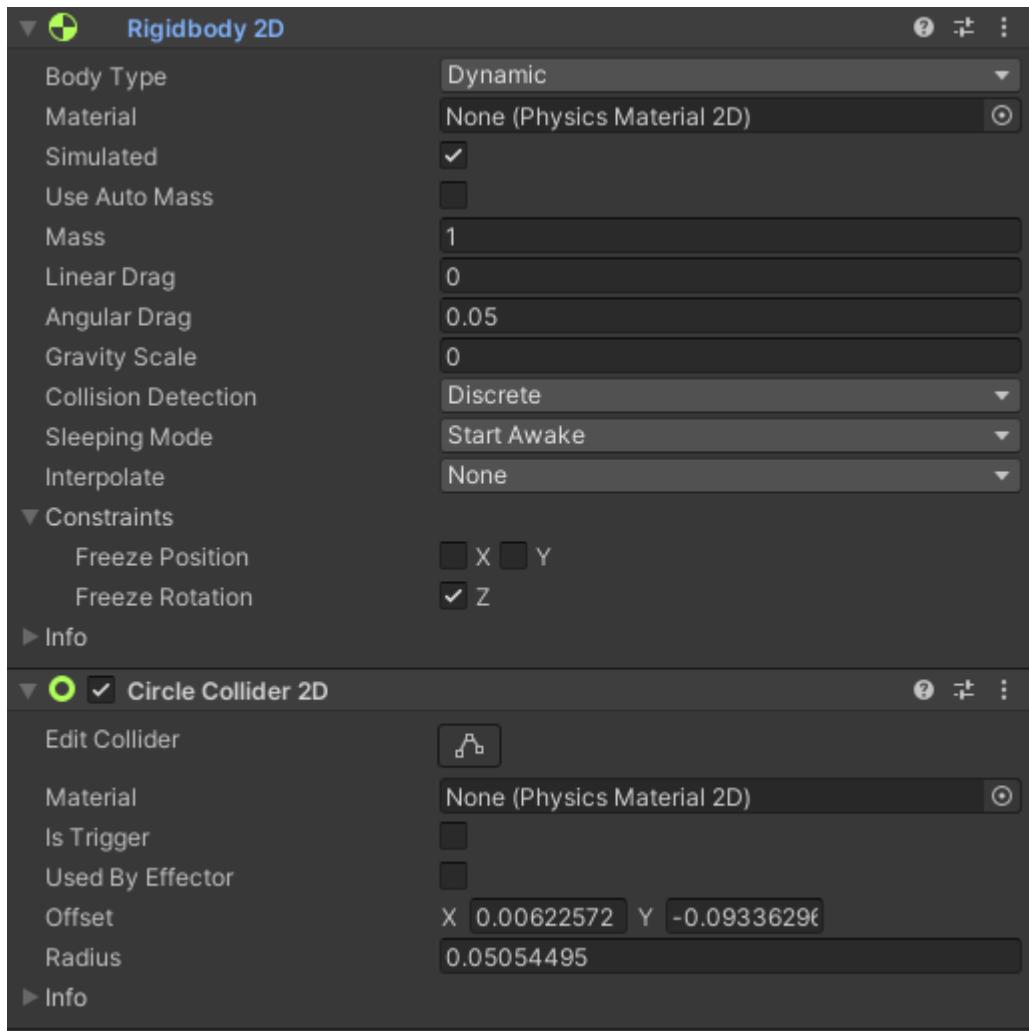
hành vi thích hợp, chẳng hạn như phát một hiệu ứng âm thanh hoặc áp dụng một lực vào đối tượng.



Hình 53: Thông số âm thanh ứng với các HitBox.

Thể hiện các âm thanh được dùng trong các HitBox sử dụng kiểm tấn công kẻ thù.

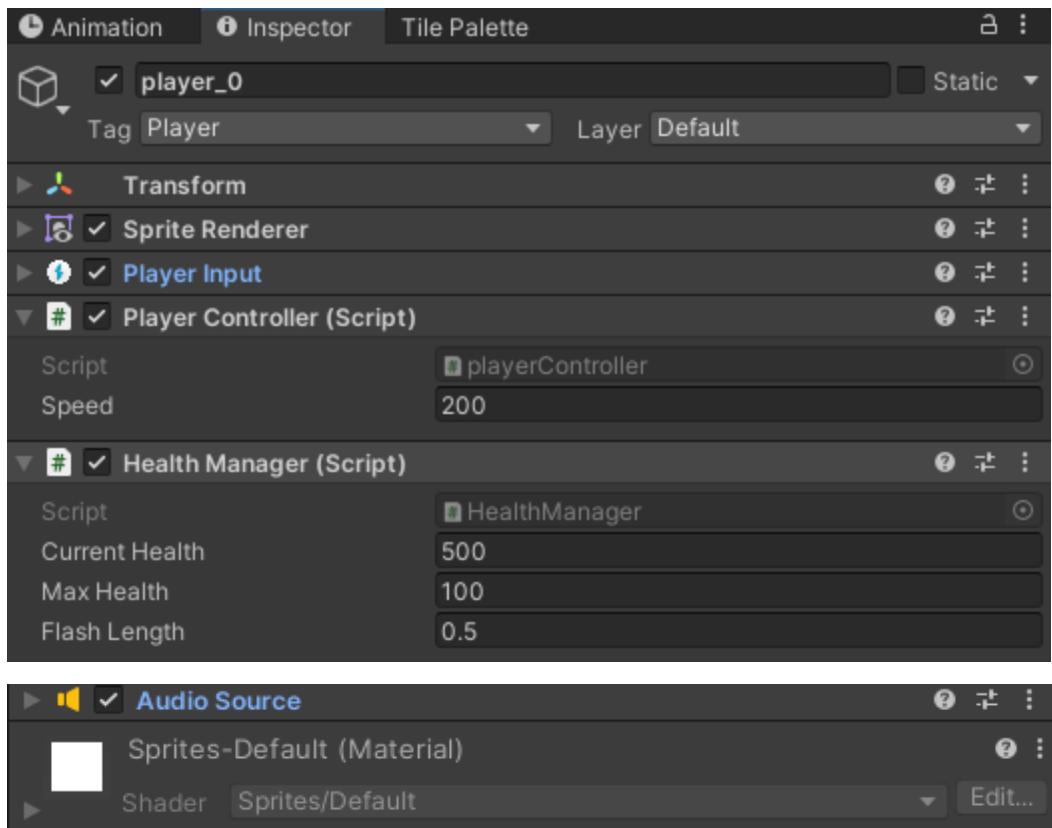
### b) Các đối tượng vật lý của nhân vật



Hình 54: Thông shuộc tính vật lý của nhân vật

Thành phần Rigidbody 2D cho phép đối tượng di chuyển và phản ứng với các lực vật lý như lực ma sát, lực đẩy, lực kéo, lực va chạm và trọng lực. Nó cũng cho phép đối tượng bị ảnh hưởng bởi các lực ngoài, chẳng hạn như lực của người chơi hoặc đối tượng khác trong trò chơi.

Thành phần Circle Collider 2D ở đây cũng có chức năng tương tự Box Collider 2D dùng để tạo va chạm của nhân vật với Tile Map được cố định là solidObjects và va chạm với các Enemy xuất hiện trong map của game.



Hình 55: Các thuộc tính đi kèm khác

⇒ Kết quả cho các xử lý va chạm:

- Tạo được sự tương tác giữa các vật 2D.
- Chiến đấu với các kẻ thù.

## 4.2. Cài đặt chương trình và kết quả

### 4.2.1. Cài đặt Unity Hub

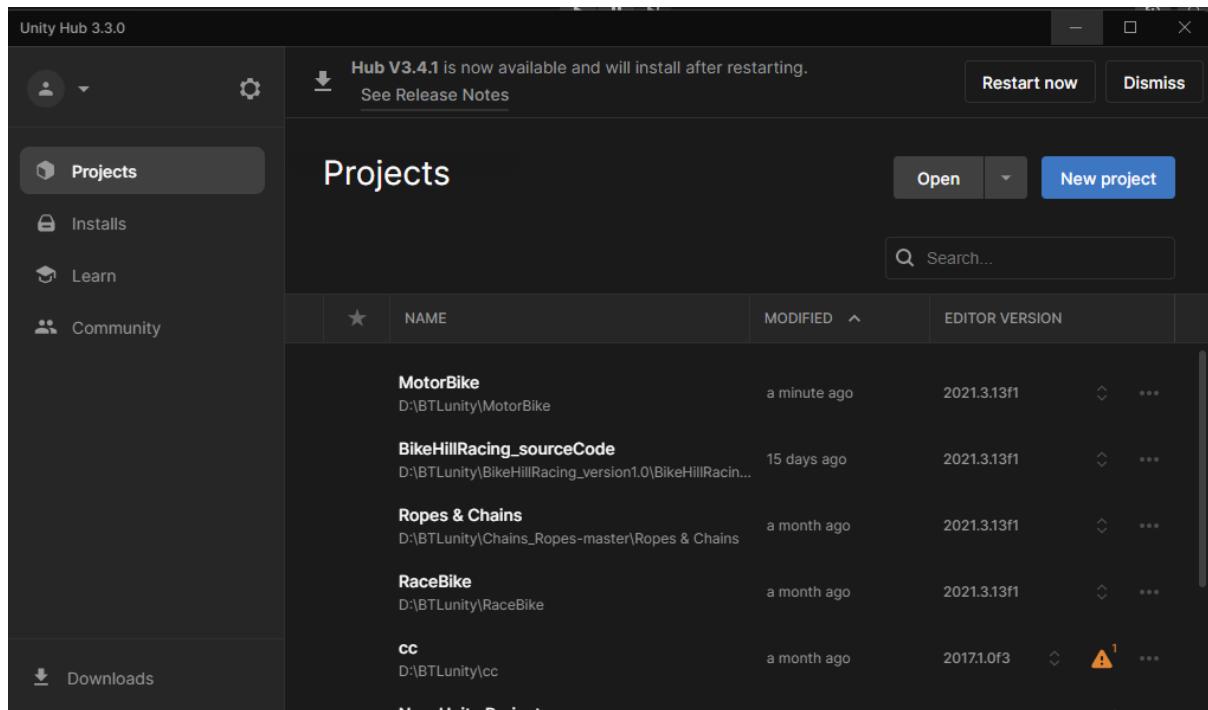
- Unity Hub giúp dễ dàng quản lý các dự án Unity của mình và nếu lập trình viên Unity cần cài đặt nhiều phiên bản Unity (điều đó xảy ra) thì Unity Hub cũng có thể quản lý điều đó.
- Có thể cài đặt Unity Hub tại <https://store.unity.com/download-nu>.

### 4.2.2. Tạo Unity ID

- Trong quá trình cài đặt, lập trình viên Unity sẽ được yêu cầu tạo một ID Unity. Ngoài việc được yêu cầu sử dụng trình chỉnh sửa, ID Unity là chìa khóa cho nhiều dịch vụ Unity bao gồm khả năng tải các dự án lên đám

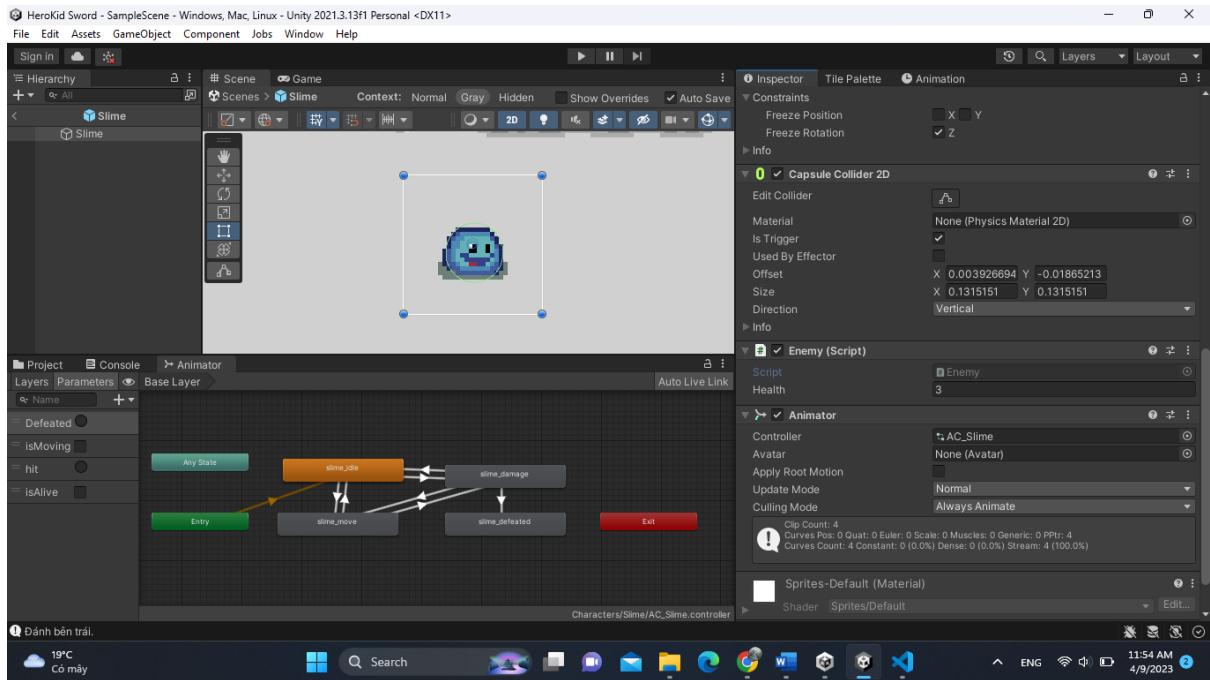
mây, truy cập các tính năng cộng đồng trên Connect và theo dõi việc học trong các khóa học của Unity!

### 4.2.3. Import và chạy project



Hình 56: Màn hình khởi chạy UnityHub

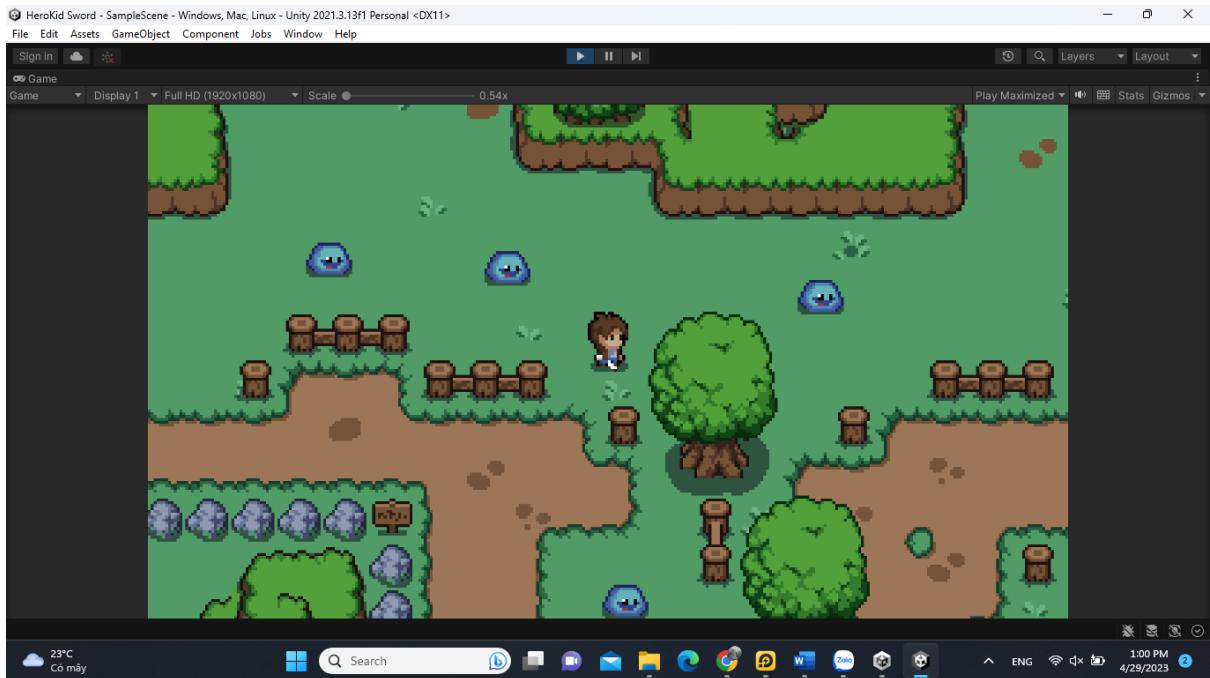
- Để tạo mới Game Project thì bạn cần nhấn NEW vào thực hiện tạo mới.
- Để import Game Project có sẵn thì bạn chỉ cần nhấn ADD và thêm file source project.



Hình 57: Màn hình sau khi import và mở Game project trên Unity

Chính là màn hình thiết kế gồm tất tần tật các cửa sổ để có thể thiết kế ra được game HeroKid Sword.

#### 4.2.4. Kết quả



Hình 58: Khu rừng

Với màn hình khu rừng gồm các kẻ thù như Slime, Nepenthe. Giao diện yên tĩnh trong lành. Dễ dàng cho việc làm quen với trò chơi, thao tác được các đối tượng và sử dụng các kĩ năng không sợ bị đánh bại vì kẻ thù ở đây khá yếu.

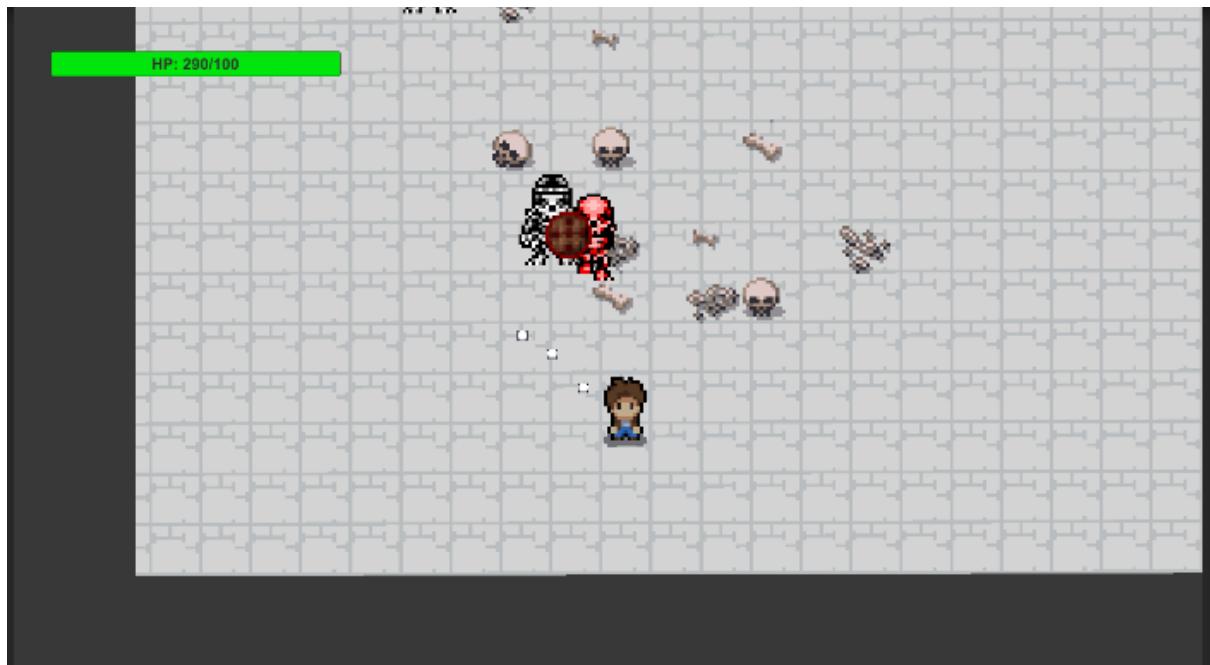


Hình 59: Làng

Là giao diện bắt đầu cuộc hành trình, nơi nhân vật được khởi tạo.

Với màn hình làng thì ở đây khá ít kẻ thù. Bao gồm nhiều đối tượng như cây cối, nhà ở, các vật dụng, NPC bán vật phẩm,...

Âm thanh sử dụng nhẹ nhàng, yên bình.



Hình 60: Hầm ngực

Là giao diện cuối cùng là màn chơi khó khăn nhất, bao gồm các kẻ thù mạnh mẽ như Skeleton White, Skeleton Red và Skeleton Shield.

Đối với người chơi nên cẩn thận và khôn khéo để vượt qua. Có thêm các vật phẩm tăng sức mạnh hoặc kĩ năng thì dễ vượt qua.

## KẾT LUẬN

### Kết quả đạt được

- ✓ Trình bày được tổng quan về công nghệ Unity Engine
- ✓ Hiểu rõ được tác dụng của Animation.
- ✓ Hiểu rõ được cách làm game trên Unity.
- ✓ Xử lý được các lỗi cơ bản trong unity.
- ✓ Xử lý được âm thanh trong game.
- ✓ Xử lý được nhân vật trong game.
- ✓ Xử lý được các quái vật trong game.
- ✓ Xử lý được các vật phẩm trong game.
- ✓ Hoàn thành được game HeroKid Sword.
- ✓ Ứng dụng công nghệ Unity xử lí các bài toán xây dựng lên hệ thống trò chơi hiệu quả. Sử dụng ngôn ngữ C# lập trình lên ứng dụng.
- ✓ Xây dựng được trò chơi có tính giải trí cao và hiệu quả giúp người chơi thoải mái sau những ngày làm việc mệt nhọc.
- ✓ Chức năng đơn giản dễ sử dụng phù hợp với mọi lứa tuổi.

### Hạn chế của đề tài

- Chưa xử lý được tối ưu các ràng buộc, dữ liệu chưa được sắp xếp linh hoạt hợp lý.
- Còn nhiều chức năng chưa được hoàn thiện
- Chưa bắt được hết các lỗi của hệ thống.
- Chưa xử lý được trạng thái hệ thống bị dừng khi đang thao tác và còn một số tồn tại trong việc đặt tên và sử dụng linh hoạt các điều khiển.

### Hướng phát triển

- Tương tác được giữa người chơi thông qua hệ thống.
- Dữ liệu được tối ưu hóa đến mức chi tiết nhất.
- Tối ưu hóa nhân vật hơn giúp nhân vật chuyển được nhiều trạng thái nhân vật.

## TÀI LIỆU THAM KHẢO

1. Janine Suvak - Lập Trình Game Với Unity
2. Unity for Absolute Beginners
3. Unity 2D Game Development
4. Learn Unity for 2D Game Development
5. Learning C# Programming With Unity 2D - Alex Okita
6. Charles Bernardooff - NGUI for Unity – 2014
7. <http://www.unity3dstudent.com/>
8. <http://unity3d.com/learn>
9. <https://opengameart.org/content/skeletons-rework>