**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

NGUYỄN THÀNH TÍN MSSV: N17DCCN153 CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM 2017-2022 Lớp: D17CQCP02-N

**Gáy**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**ĐỒ ÁN**

**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài*: “Xây dựng Game 2D Soldier's Avenger”**

**Người hướng dẫn: Ths. Huỳnh Trung Trụ**

**Sinh viên thực hiện: Nguyễn Thành Tín**

**Mã số sinh viên: N17DCCN153**

**Lớp: D17CQCP02-N**

**Hệ** **: Chính Quy**

TP. HCM

2021

**TP. HỒ CHÍ MINH, NĂM 2021**

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

------------------------------



**ĐỒ ÁN**

**TỐT NGHIỆP ĐẠI HỌC**

***Đề tài*:**

**“Xây dựng Game 2D Soldier's Avenger”**

**Người hướng dẫn: Ths. Phan Thanh Hy**

**Sinh viên thực hiện: Lê Gia Bảo Phú**

**Mã số sinh viên: N17DCCN131**

**Lớp: D17CQCP02-N**

**Hệ: Chính Quy**

**TP. HỒ CHÍ MINH, NĂM 2021**

BM.HV-KT.01/06

|  |  |
| --- | --- |
| HỌC VIỆN | **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM** |
| CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG | **Độc lập – Tự do – Hạnh phúc** |
| CƠ SỞ TẠI TP. HỒ CHÍ MINH |  |
| **Khoa: Công nghệ thông tin 2** | *TP. Hồ Chí Minh, ngày 21 tháng 09 năm 2021* |

ĐỀ TÀI ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

Căn cứ Quyết định số: 402/QĐ-HVCS, ngày 21 tháng 09 năm 2021 của Phó Giám đốc Học viện – Phụ trách Cơ sở tại TP. Hồ Chí Minh về việc “phê duyệt danh sách giáo viên hướng dẫn và giao đề tài đồ án tốt nghiệp cho sinh viên Đại học chính quy Khóa 2017-2022 Ngành Công nghệ thông tin, An toàn thông tin và Công nghệ đa phương tiện”;

Khoa Công nghệ thông tin 2 giao nhiệm vụ thực hiện Đồ án tốt nghiệp cho sinh viên:

1. **Họ và tên sv : Nguyễn Thành Tín Mã SV : N17DCCN153**

**Lớp : D17CQCP02-N Khóa : 2017-2022**

**Ngành đào tạo : Công nghệ thông tin Hệ đào tạo : Đại học Chính quy**

1. **Tên đề tài tốt nghiệp: Xây dựng Game 2D Soldier's Avenger**
2. **Nội dung chính của Đồ án:**

**Lý thuyết:**

Tìm hiểu nghiệp vụ, ngôn ngữ và công nghệ sử dụng:

-Tìm hiểu và xây dựng game trên Engine Unity3D trên thiết bị Mobile và PC

-Tìm hiểu ngôn ngữ C# , JavaScript

-Tìm hiểu NodeJS, WebSocket

-Database: MongoDb

**Thực hành:**

-Thiết kế Cơ sở dữ liệu hệ thống.

-Thiết kế và xây dựng hệ thống UI : Home, Inventory, Daily Reward, List Mission, Leader Board, GameSetting , GamePlay,.....

- Xử lí đăng nhập : login , register.

- Xây dựng hệ thống cốt truyện chính.

- Xây dựng và thiết kế hệ thống Player.

- Xây dựng và thiết kế hệ thống Enemy.

- Xây dựng và thiết kế Map, Trap.

- Xây dựng và thiết kế Bonus Items.

- Xây dựng hệ thống effect chiến đấu, nhạc nền, âm thanh.

- Xây dựng GamePlay :

+ Map : xây dựng Map cho mỗi màn chơi, hệ thống bẫy tương tác với Player.

+ Player : chiến đấu với Enemy, nhận vàng, kinh nghiệm nâng cấp sức mạnh, kĩ năng, vũ khí,... qua từng màn chơi.

+ Enemy: chiến đấu với Player , xử lí AI né tránh, giữ khoảng cách, tấn công Player.

- Đưa game lên CHPlay.

1. **Cơ sở dữ liệu ban đầu:**
2. **Giáo viên hướng dẫn: ThS. Huỳnh Trung Trụ**
3. **Ngày giao đề tài: 27/09/2021**
4. **Ngày nộp quyển: 07/12/2021**

**TRƯỞNG KHOA CNTT**

**Nơi nhận:**

* *Sinh viên có tên tại khoản 1;*
* *Lưu: VP Khoa.*

**TS. Nguyễn Hồng Sơn**

# MỞ ĐẦU

Hiện nay, công nghệ thông tin đã có những bước phát triển mạnh mẽ theo cả chiều rộng và sâu.Máy tính điện tử không còn là một thứ phương tiện quý hiếm mà đang ngày càng trở thành một công cụ làm việc và giải trí thông dụng của con người, không chỉ ở công sở mà còn ngay cả trong gia đình.

Đi cùng với sự tiến hóa công nghệ, nhu cầu giải trí của con người cũng được nâng tầm phát triển, trong đó Game điện tử là một khái niệm được sinh ra và phát triển theo quá trình phát triển mạnh mẽ của máy móc điện tử và Internet.

Game điện tử trên thế giới hiện nay đã đến những bước tiến lớn nhờ công nghệ hiện đại, hướng con người đến trải nghiệm tốt nhất về tất cả các mặt: âm thanh, đồ họa, gameplay, cốt truyện…. trên nhiều nền tảng như mobile, PC, gamepad, playstation,….Trên thế giới, game điên tử đã trở nên phổ biến toàn cầu, trở thành một nền công nghiệp phát triển mạnh mẽ là nơi để đầu tư chất xám hiệu quả. Mỗi giai đoạn phát triển và mỗi nền tảng đều có những game điện tử để lại dấu mốc quan trọng:Mario, HaftLife, GTA, …

Ở Việt Nam cũng có rất nhiều doanh nghiệp cá nhân đã và đang phát triển mạnh về lĩnh vực Game điện tử, nhiều studio cũng như cá nhân đạt nhiều thành tựu nổi tiếng toàn cầu : Emobi Game với game 7554 Chiến dịch Điện Biên Phủ gây tiếng vang lớn trên cộng đồng game thế giới về cốt truyện , hoặc gần đây với cú nổ mang tính lịch sử đối với thể loại game đơn giản và vô tận mang tên Flappy Bird của lập trình viên Nguyễn Hà Đông vang danh Việt Nam trên thị trường game quốc tế. Những phát triển trên đã tạo đà và cả động lực, ý tưởng cho thế hệ mới phát triển mạnh mẽ hơn, tiếp cận với công nghệ tiên tiến trên thế giới.

Để tiếp cận và góp phần đẩy mạnh sự phổ biến của game điên tử ở Việt Nam, em đã quyết định thực hiện đề tài “**Xây dựng Game 2D Soldier's Avenger**”. Việc xây dựng game 2D trên sẽ tạo bước đà đủ lớn để tiếp cận với game điện tử về mọi mặt, phát triển kĩ năng hoàn thiện để tiến tới các mục tiêu lớn hơn trong tương lai.Đồng thời, tạo ra một game 2D thân thiện với người chơi, đơn giản cuốn hút và đặc biệt là tính giải trí cho mọi người sau những khoảng thời gian làm việc và học tập mệt mỏi.

# LỜI CẢM ƠN

Đồ án tốt nghiệp đóng một vai trò rất quan trọng và mang lại nhiều hiệu quả tích cực đối với quá trình học tập và nghiên cứu của sinh viên khi còn ngồi trên ghế giảng đường Đại Học. Đồ án tốt nghiệp giúp cho sinh viên hệ thống hóa lại kiến thức mà sinh viên đã được học, tìm hiểu và nắm bắt được những kiến thức mới cũng như đi sâu vào những vấn đề và lĩnh vực mình yêu thích.

Bản báo cáo này trình bày về quá trình xây dựng và hoàn thiện game 2D cơ bản : xậy dựng bản đồ, nhân vật , kẻ thù cũng như gameplay và các chức năng cần thiết của một game điện tử trên nền tảng PC và mobile.

Trong quá trình thực hiện không thể tránh những thiếu sót, kính mong thầy và các bạn thông cảm.

Em xin chân thành cảm ơn sự giúp đỡ nhiệt tình của thầy **Huỳnh Trung Trụ** trong quá trình em thực hiện và hoàn thành đồ án này. Đồng thời, em xin gửi lời cảm ơn chân thành đến các thầy cô trong **Khoa Công nghệ Thông tin 2** đã tạo mọi điều kiện thuận lợi để em có thể thực hiện được bài báo cáo tốt nghiệp này

**TP.Hồ Chí Minh, ngày 7 tháng 10 năm 2021**

**Sinh viên**

**Nguyễn Thành Tín**

**MỤC LỤC**

[MỞ ĐẦU 4](#_Toc89297892)

[LỜI CẢM ƠN 5](#_Toc89297893)

[DANH SÁCH CÁC BẢNG, SƠ ĐỒ, HÌNH 8](#_Toc89297894)

[KÝ HIỆU CÁC CỤM TỪ VIẾT TẮT 9](#_Toc89297895)

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 10](#_Toc89297896)

[1.1 Lý do chọn đề tài: 10](#_Toc89297897)

[1.2 Mục đích, ý nghĩa chọn đề tài: 10](#_Toc89297898)

[1.3 Phương pháp nghiên cứu: 10](#_Toc89297899)

[1.4 Phạm vi, đối tượng đề tài: 11](#_Toc89297900)

[1.5 Ý nghĩa thực tiễn đề tài 11](#_Toc89297901)

[1.6 Cấu trúc báo cáo 11](#_Toc89297902)

[CHƯƠNG 2: CƠ SỞ LÝ LUẬN VÀ NỀN TẢNG XÂY DỰNG HỆ THỐNG 13](#_Toc89297903)

[2.1. Tổng quan về Game điện tử 13](#_Toc89297904)

[2.1.1. Khái niệm Game điện tử 13](#_Toc89297905)

[2.1.2. Lợi ích của Game điện tử 13](#_Toc89297906)

[2.1.3. Các yêu cầu trong một game điện tử hay 13](#_Toc89297907)

[2.2. Tìm hiểu công cụ thực hiện đề tài. 16](#_Toc89297908)

[2.2.1 Giới thiệu Unity3D 16](#_Toc89297909)

[2.2.2 Giới thiệu Visual Studio 2019 25](#_Toc89297910)

[2.2.3. Giới thiệu về MongoDB 27](#_Toc89297911)

[2.3. Tìm hiểu về .NET 28](#_Toc89297912)

[2.3.1 Ngôn ngữ C# 28](#_Toc89297913)

[2.3.2 Tìm hiểu về .NET Framework 29](#_Toc89297914)

[2.4. Tổng quan về RestFul API 29](#_Toc89297915)

[CHƯƠNG 3: Xây dựng game 2D Soldier's Avenger 31](#_Toc89297916)

[3.1 Khảo sát hệ thống Game 2D Soldier's Avenger 31](#_Toc89297917)

[3.2 Mô hình Usecase 31](#_Toc89297918)

[3.2.1 Sơ đồ Usecase cho người chơi 31](#_Toc89297919)

[3.2.2 Các nhóm đối tượng : 31](#_Toc89297920)

[3.2.3 Sơ đồ Component : 33](#_Toc89297921)

[Hình : sơ đồ Component tổng quát 33](#_Toc89297922)

[3.3 Xây dựng các hệ thống 34](#_Toc89297923)

[3.3.1 Xây dựng Hệ thống quản lí 34](#_Toc89297924)

[3.3.2 Xây dựng Hệ thống điều khiển 34](#_Toc89297925)

[3.3.3 Xây dựng Hệ thống Bẫy 35](#_Toc89297926)

[3.3.4 Xây dựng Core và State Machine áp dụng Computer Science 36](#_Toc89297927)

[3.3.5 Xây dựng hệ thống Player dựa trên Core và State Machine 42](#_Toc89297928)

[3.3.6 Xây dựng hệ thống Enemy dựa trên Core và AI State Machine 50](#_Toc89297929)

[3.6 Thiết lập mô hình thực thể 56](#_Toc89297930)

[3.6.1 Xác định các thực thể 56](#_Toc89297931)

[3.6.2 Xác định mối quan hệ giữa các thực thể 56](#_Toc89297932)

[3.6.3 Sơ đồ ERD (Entity Relationship Diagram) 56](#_Toc89297933)

[3.6.4 Mô hình dữ liệu quan hệ 56](#_Toc89297934)

[3.7 Thiết kế dữ liệu 56](#_Toc89297935)

[3.7.1 Từ điển dữ liệu 56](#_Toc89297936)

[3.7.2 Mô hình diagram 56](#_Toc89297937)

[3.8 Xây dựng dự án 56](#_Toc89297938)

[3.8.1 Xây dựng BackEnd 56](#_Toc89297939)

[CHƯƠNG 4: DEMO GAME 57](#_Toc89297940)

[CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 58](#_Toc89297941)

[4.1 Kết quả 58](#_Toc89297942)

[TÀI LIỆU THAM KHẢO 59](#_Toc89297943)

# DANH SÁCH CÁC BẢNG, SƠ ĐỒ, HÌNH

**Hình 1 Phần mềm Unity3D**

**Hình 2 Phần mềm Visual Studio 2019**

**Hình 3 Database MongoDB**

**Hình 4 Lịch sử phát triển C#**

**Hình 5 Đặc trưng ngôn ngữ C#**

**Hình 6 Tổng quan ResFul API**

**Hình 7: Core**

**Hình 8: Animation Player**

**Hình 9: Ledge Climb State**

**Hình 10A: Blend Tree Player**

**Hình 10B: Pramester BlendTree Player**

**Hình 11: AbilityState và Movement**

**Hình 12: Chuyển đổi giữa In Air State và Grounded State**

**Hình 13: In Air State sang Touching Wall State**

**Hình 14: Ledge Climb State**

**Hình 15: Animation thực thể player dùng kiếm**

**Hình 16: Animation Weapon kiếm**

**Hình17: Animation thực thể Player dùng cung**

**Hình 18: Animation cung tên**

**Hình 19: Animation thực thể Player dùng dao**

**Hình 20: Animation dao**

**Hình 21: Animation thực thể Player dùng khiên**

**Hình 22: Animation khiên**

**Hình 23: Sơ đồ hoạt động Enemy 1 Pork**

**Hình 24 : Animation Enemy 1 Pork**

**Hình 25: Sơ đồ hoạt động Enemy 2 Archers**

**Hình 26: Animation Enemy 2 Archers**

**Hình 27: Sơ đồ hoạt động Enemy 3 Wizard**

**Hình 28: Animation Enemy 3 Wizard**

**Hình 29: Sơ đồ hoạt động Enemy 4 Goblin**

**Hình 30: Animation Enemy 4 Goblin**

**Hình 31: Sơ đồ hoạt động Enemy 5 Skeleton**

**Hình 32: Animation Enemy 5 Skeleton**

# KÝ HIỆU CÁC CỤM TỪ VIẾT TẮT

|  |  |
| --- | --- |
|  | |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
| NSD = User = Account | Quyền làm việc được cấp cho NGƯỜI SỬ DỤNG trên hệ thống App, quyền này thông thường được đảm bảo bởi tên đăng nhập (username) và mật khẩu (password); cũng có thể gọi là “tài khoản người dùng” [“tài khoản”] |
|  |  |

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài:

Hiện nay, Game điện tử đối với nhân loại là một món ăn tinh thần không thể thiếu với mọi lứa tuổi, mọi giới tính,…Game điện tử trở thành nơi giải trí hiệu quả sau những giờ học tập và làm việc căng thẳng, nơi để mọi người giao lưu, kết bạn, giải trí, rèn luyện và làm mọi việc mà trên thực tế chúng ta chưa đủ kinh tế hoặc thời gian dài để thực hiện.Đối với lập trình viên, đây là mảnh đất màu mỡ để đầu tư chất xám hiệu quả nhất, nơi ta có thể xây dựng mọi thứ bằng trí tưởng tượng của chúng ta và các dòng code.

Việc chọn đề tài trên cũng là bước đầu tiên trên con đường tiếp cận các công nghệ làm game cũng như hướng tới việc xây dựng thế giới ảo trong Game.Việc xây dựng một game là xây dựng một thế giới ảo với những quy luật mà do chính chúng ta, game’s master đề ra, nói đơn giản, chúng ta xây dựng game chính là xây dựng 1 thế giới mơ ước của chính chúng ta.

## 1.2 Mục đích, ý nghĩa chọn đề tài:

* Xây dựng được một thế giới game 2D với những quy luật, nơi mà chúng ta vào vai một người lính tiến vào công cuộc trả thù những con quái vật và cả những kẻ thù ác.
* Xây dựng một gameplay thân thiện, người chơi sẽ được giải trí và giải đố, phát triển sự nhanh nhạy trong việc xử lí thông tin, phát triển IQ trong quá trình chơi game..
* Là bước đà để tiếp cận công nghệ thế giới.
* Giúp làm quen với việc xây dựng game hướng tới xây dựng một thế giới game rộng lớn.

## 1.3 Phương pháp nghiên cứu:

Tìm hiểu “**Xây dựng Game 2D Soldier's Avenger**”

* Khảo sát hệ thống

- Khảo sát các game có nội dung hoặc gameplay tương tự với phong cách phiêu lưu chiến đấu và cả giải đố ví dụ: Mario,…

* Phân tích chức năng hệ thống

- Từ quá trình khảo sát, phân tích và đưa ra bảng các chức năng chính của hệ thống và phân tích chi tiết từng chức năng

* Phân tích thiết kế với UML

- Vẽ và phân tích biểu đồ User Case

- Biểu đồ tuần tự

- Biểu đồ lớp

- Biểu đồ hoạt động

* Phân tích thiết kế CSDL
* Phân tích thiết kế chương trình

**a. Lý Thuyết:**

Tìm hiểu nghiệp vụ, ngôn ngữ và công nghệ sử dụng:

* Tìm hiểu và xây dựng game trên Engine Unity3D trên thiết bị Mobile và PC
* Tìm hiểu ngôn ngữ C# , JavaScript
* Tìm hiểu NodeJS, WebSocket
* Database: MongoDb

**b. Thực Hành:**

* Thiết kế Cơ sở dữ liệu hệ thống.
* Thiết kế và xây dựng hệ thống UI : Home, Inventory, Daily Reward, List Mission, Leader Board, GameSetting , GamePlay,.....
* Xử lí đăng nhập : login , register.
* Xây dựng hệ thống cốt truyện chính.
* Xây dựng và thiết kế hệ thống Player.
* Xây dựng và thiết kế hệ thống Enemy.
* Xây dựng và thiết kế Map, Trap.
* Xây dựng và thiết kế Bonus Items.
* Xây dựng hệ thống effect chiến đấu, nhạc nền, âm thanh.
* Xây dựng GamePlay :
* Map : xây dựng Map cho mỗi màn chơi, hệ thống bẫy tương tác với Player.
* Player : chiến đấu với Enemy, nhận vàng, kinh nghiệm nâng cấp sức mạnh, kĩ năng, vũ khí,... qua từng màn chơi.
* Enemy: chiến đấu với Player , xử lí AI né tránh, giữ khoảng cách, tấn công Player.
* Đưa game lên CHPlay.

**c. Công cụ hỗ trợ:**

* Visual Studio Code
* Unity3D

## 1.4 Phạm vi, đối tượng đề tài:

* Độ tuổi: 10+ (trên 10 tuổi, đủ nhận thức về trò chơi và giải đố trong trò chơi)
* Giới tính: phù hợp với tất cả giới tính.
* Độ phủ sóng: game sẽ được đưa lên các cửa hàng điện tử có thể tiếp cận dễ dàng với người chơi thông qua internet.
* Thiết bị: game đơn giản phù hợp với mọi loại thiết bị trên nhiều nền tảng.
* Nền tảng : PC và Mobile : điều khiển gamepad keyboard và mouse trên PC, và UI Button trên Mobile.
* Thể loại: game Phiêu lưu , giải đố phù hợp với nhu cầu giải trí.

## 1.5 Ý nghĩa thực tiễn đề tài

* Game giúp người chơi hóa thân vào nhân vật mô phỏng, tham gia các trận chiến và giải đố, chiến đấu giúp người chơi giải trí cũng như tăng cường khả năng xử lí tình huống, phát triển IQ.
* Việc xây dựng game sẽ giúp ta tiếp cận và học nhiều kĩ năng trong việc xây dựng thế giới game trong tương lai.

## 1.6 Cấu trúc báo cáo

Phần 1: Mở đầu

Phần 2: Nội dung

- Chương I: Tổng quan đề tài

- Chương II: Cơ sở lý luận và nền tảng xây dựng hệ thống

- Chương III: Xây dựng GamePlay

- Chương IV: Demo Game

- Chương V: Đánh giá kết quả và hướng phát triển đề tài

Phần 3: Tài liệu tham khảo

# CHƯƠNG 2: CƠ SỞ LÝ LUẬN VÀ NỀN TẢNG XÂY DỰNG HỆ THỐNG

## 2.1. Tổng quan về Game điện tử

### 2.1.1. Khái niệm Game điện tử

Trò chơi điện tử là trò chơi sử dụng thiết bị điện tử để tạo ra một hệ thống tương tác với người chơi nhằm mục đích chính là giải trí.

Game điện tử phải được hiểu đầy đủ là bao gồm: Các trò chơi dùng điện báo đánh chữ (Teletype games), Máy chơi game cầm tay (Electronic handhelds), Máy bắn bi và các thiết bị tương tự (Pinball machines and similar devices), Trò chơi đổi thưởng (Redemption games), Máy đánh bạc (Slot machines), Trò chơi âm thanh (Audio games) và Trò chơi video (Video games).

Đề tài trên đi vào xây dựng video game, thuộc Game điện tử cũng chính là xu hướng hiện đại phát triển nhất của Game điện tử.

### 2.1.2. Lợi ích của Game điện tử

**2.1.2.1. Ngành dịch vụ đang hot nhất trên thị trường**

Game điện tử nhất là video game hiện tại đang phát triển mạnh mẽ do sự phát triển của công nghệ cùng với nhu cầu giải trí càng cao của con người. Do đó đây luôn là môi trường màu mỡ cho việc đầu tư chất xám cũng như lợi nhuận đáng kể do ngành dịch vụ này mang lại do có thị trường người dùng trải dài trên mọi lứa tuổi, mọi giới tính, mọi nơi…

Người chơi còn có thể đầu tư vào game, buôn bán vật phẩm giá trị trong game để tạo thu nhập, từ đó đem lại lợi nhuận cho cả những người chơi giỏi.

**2.1.2.2. Tính giải trí**

Điều tiên quyết trong một game điện tử là tính giải trí. Một game điện tử phát triển được phải đảm bảo được nhu cầu giải trí của con người. Trong thời đại ngày nay, nhu cầu giải trí ngày càng cao cũng như chất lượng phải đảm bảo,do đó các game điện tử phải đảm bảo tính giải trí phù hợp với thời đại để tránh bị lạc hậu, lãng quên.

**2.1.2.3. Phát triển bản thân người chơi**

Game điện tử không chỉ đơn giản để giải trí, mà còn đảm bảo tính tò mò, thích thú và thử thách người chơi, và để hoàn thành game, người chơi phải hoàn thiện khả năng phản xạ, tính logic và khả năng giải đố tùy thuộc vào thể loại game. Do đó game nâng cao khả năng của người chơi, tăng cường IQ, luyện phản xạ cho tay, mắt…

Một số game ra đời nhằm giúp đỡ trong cả ngành y học, trong vật lí trị liệu giúp đỡ bệnh nhân đi đứng, lấy lại cảm giác từ các giác quan qua các game thực tế ảo hoặc các game phản xạ cơ bản.

**2.1.2.4. Tính xã hội**

Ngày nay với sự phát triển của internet đã tạo nên một cộng đồng game lớn mạnh trên toàn thế giới, các hội nhóm game hoặc các tổ chức ngày càng lón được tạo ra, xây dựng các giải đấu dành cho những game thủ chuyên nghiệp. Game điện tử trở thành môn thể thao điện tử không thể thiếu trong các kì vận hội trong tương lai.

Game online giúp các cá nhân tương tác với nhau như một xã hội thu nhỏ, giúp kết bạn mới hoặc kết hợp chơi với nhau tạo nên tính đoàn kết trong cộng đồng.

### 2.1.3. Các yêu cầu trong một game điện tử hay

Mỗi trò chơi đều có những đặc tính riêng tạo nên sự khác biệt so với những trò chơi khác. Nhưng đặc biệt một số game lại sở hữu các “quyền năng” vô hình nào đó khiến game thủ không thể nào dứt ra khỏi mà cứ chơi mãi ngày này qua ngày khác, trong khi có những game lại bị bỏ xó không thương tiếc, chờ ngày gỡ cài đặt.

Dưới đây là các yêu cầu của 1 game điện tử hay :

**2.1.3.1.Các nấc thang trình độ**

Đây có lẽ là khía cạnh quan trọng nhất làm nên sự hấp dẫn nhất của một trò chơi. Điểm cốt lõi của eSports và competitive games chính là sự cải thiện liên tục, thay đổi liên tục, đòi hỏi người chơi phải luôn luôn nâng cao kỹ năng của bản thân, tạo nên những thách thức để người chơi vượt qua, từ đó làm nên tính cạnh tranh và lôi cuốn của trò chơi.Dù vẫn có những tựa game cực kỳ khó như Mega man Classic hoặc game chiến thuật nhóm tuyệt hay như MOBA, nhưng tựu chung thì cách phổ biến nhất để lôi cuốn người chơi đó là cung cấp các cách thức rõ ràng giúp họ nâng cao trình độ. Nếu trong game không có tính năng đấu PvP (Player versus Player - người chơi đấu với người chơi) thì hãy sử dụng leaderboard (bảng xếp hạng), giúp người chơi có thể dễ dàng so sánh mình với những người chơi khác. Thêm vào đó, hãy nhớ bao gồm cả tính năng thử thách hàng ngày (daily challenges) để tạo thành thói quen cho người chơi.

**2.1.3.2.** **Cốt truyện hấp dẫn**

Khi được yêu cầu đưa ra ví dụ về một game có cốt truyện hấp dẫn, nhiều người sẽ nghĩ ngay đến những tựa game nặng tính kể chuyện như game JRPG (Japanese Role Playing Game - thể loại game nhập vai Nhật Bản) hay những game phương Tây có chế độ một người chơi (single player). Tuy nhiên, hiếm có khi nào mà một trò chơi lại bị “dở đi” bởi các yếu tố kể chuyện.Các yếu tố thuộc về kiến thức cũng khá cần thiết, thậm chí những game eSports như Liên minh huyền thoại(LMHT) cũng đã giải quyết được vấn đề này. Mỗi lượt chơi có thể không liên quan nhiều với nhau nhưng bằng việc cùng nhau giải quyết các vấn đề trong thế giới giả tưởng của game, LMHT đã tạo được sự liên kết và cộng hưởng giữa nhiều game thủ trong game. Xây dựng thế giới trong game và tạo nên tình huống truyện hấp dẫn không phải là những cách duy nhất để thỏa mãn yêu cầu của người chơi về mặt cốt truyện. Nhiều tình tiết trong game MMO(Massively Multiplayer Online) hiện nay còn hấp dẫn hơn cả câu chuyện gốc do những nhà văn chuyên nghiệp viết cho các nhà phát triển game. Các yếu tố trên được đưa vào trong mỗi trò chơi thông qua những hình ảnh trực quan, khơi gợi cảm xúc từ phía người chơi và trở thành câu chuyện riêng của mỗi người.

**2.1.3.3. Khả năng thành thục**

Hầu hết thời điểm chúng ta vui vẻ nhất khi chơi một trò chơi đó là khi chúng ta đã tương đối thành thạo nó và có thể chơi tự nhiên mà không cần phải suy nghĩ nhiều. Không cần phải “cày” cật lực để kiếm XP hay khám phá những hang động trong Spelunky đến cả nghìn lần, đôi khi niềm vui đơn giản đến từ những lúc CHỈ chơi game mà thôi.Điều này có thể áp dụng như nhau ở cả dòng game phổ thông và core game. Đôi khi sự mệt mỏi của những chuyến bay dài có thể bị đánh tan bởi những đường chạy gần như vô tận (endless run) của Banana kong hay Temple run. Những game này vẫn đòi hỏi người chơi có kỹ năng nhưng chúng gần như đem lại trạng thái thư giãn tuyệt đối, giúp chúng ta quên đi quãng thời gian dài dằng dặc và sự mệt mỏi của những chuyến bay.

**2.1.3.4. Sự bí ẩn**

Mong muốn tìm được câu trả lời cho những câu hỏi kéo dài không có lời giải đáp hoặc biết cho bằng được kết thúc của một chuyện gì đó hẳn là tác nhân mạnh mẽ nhất khiến một người kiên nhẫn theo dõi câu chuyện cho đến cuối cùng. Đôi khi chúng ta cứ nấn ná xem cho bằng hết một bộ phim chỉ vì muốn biết kết thúc của nó ra sao dù cho phim dở tệ. Game cũng vậy, dù cho rất hay chăng nữa nhưng nếu trong game không có những tình tiết bí mật nhằm kích thích trí tò mò của người chơi thì cũng khó mà giữ chân họ được lâu.

**2.1.3.5. Giảm bớt/Tối thiểu hóa hình phạt khi người chơi mất mạng trong game**

Trong nhiều game, sau khi bị “chết”, người chơi đã quen với việc có những hình đồ họa sặc sỡ màu mè thể hiện những câu kiểu như “Hết mạng. Đi lối này!” như thể không có chúng thì chúng ta không biết đường để... thoát game vậy. Thực ra không nhất thiết phải buộc người chơi rời khỏi game mỗi khi mất mạng như thế. Thay vì như vậy các nhà phát triển game nên áp dụng những cách thức nhẹ nhàng và tế nhị hơn để giảm thiểu hình phạt cho việc “chết” trong game, có thể chỉ đơn giản là một menu được thiết kế hợp lý, hoặc đến cách thức phức tạp hơn như là hệ thống tính năng autosave. Game Buff knight của studio Hàn Quốc Buff đã thực hiện điều này rất tốt khi dẫn dắt người chơi đến cửa hàng nhằm upgrade nhân vật của mình sau khi vừa bị mất mạng.

**2.1.3.6. Tính mới lạ**

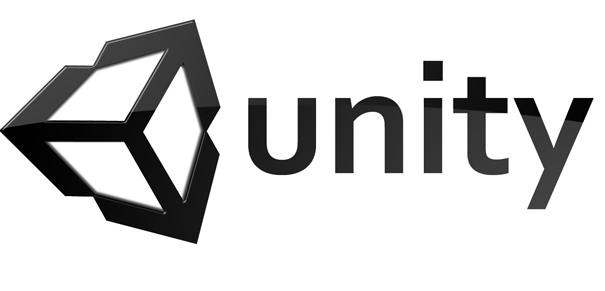
Chắc chắn là người chơi nào cũng sẽ có cảm giác thích thú phấn khích khi được chơi một trò chơi "không giống ai". Khi Flappy bird mới ra lò, rất nhiều người chơi nhanh chóng trở nên hào hứng khi chưa bao giờ gặp trò chơi nào khó như vậy!Đồ họa đẹp mắt, thế giới trong game sinh động mới lạ, nhân vật kì quặc hay cốt truyện bất ngờ v.v. hãy cố gắng tạo ra bất cứ thứ gì có thể khiến cho trò chơi của bạn khác lạ so với những game đã xuất hiện nhan nhản trên thị trường, chỉ có thế thì mới khiến người chơi có hứng thú được. Và tất nhiên sự mới mẻ này cũng chỉ tồn tại được trong một thời gian ngắn mà thôi, vì chỉ cần có game nào trở nên thành công một tẹo thì sẽ nhanh chóng có một lô một lốc những game ăn theo nó ngay, đơn cử như Flappy bird!

**2.1.3.7. Thử thách trí tuệ**

Người chơi luôn thích được thử thách, có thể chỉ đơn giản là một câu đố, hoặc một thử thách phức tạp cần phải “động não” nhiều hơn. Các thể loại thử thách trí tuệ trải dài từ chọn cánh cửa nào để mở trong các survival horror game cho đến việc hoàn thành một cuộc bố ráp như trong các game MMO. Cảm giác giải mã được câu đố hay tìm ra được những bước thích hợp để hoàn thành nhiệm vụ quả thật khiến người chơi thấy thích thú và thỏa mãn. Những nhà phát triển game Nhật Bản từ lâu đã khai thác yếu tố này một cách rất hiệu quả, với series Level 5’s Professor Layton và nhiều game khác trong dòng game survival horror.

## 2.2. Tìm hiểu công cụ thực hiện đề tài.

### 2.2.1 Giới thiệu Unity3D

****

***Hình 1 Phần mềm Unity3D***

(Nguồn: *https://teky.edu.vn/blog/unity-3d-la-gi*/)

Hiện nay, với 27 nền tảng phát triển liên tục 15 năm (từ năm 2005), Unity đang được đánh giá là một trong những phần mềm đa nền tảng mạnh mẽ nhất cho những lập trình viên game.

Unity hỗ trợ được hầu hết trên các nền tảng từ IOS, Android, hay Windowa, MacOs,…, sử dụng các ngôn ngữ lập trình C# và Javascript. Điều đó đã tạo nên thế mạnh để Unity trở thành chìa khóa để phát triển phần mềm game tốt nhất hiện nay.

Ngoài ra, việc cho phép lập trình game theo thời gian thực của Unity đã cho phép các Designer có thêm nhiều lợi thế:

* Không cần Render.
* Thiết kế được Map, Character, Interface Graphic,..từ những phần mềm khác (như Modo, Lightwave,…)
* Tối ưu hóa định dạng file theo dạng “.FBX” hay “.exe”. Vì vậy, thiết kế có thể dễ dàng chia sẻ trên các thiết bị khác.

Bên cạnh đó, Unity có khả năng cung cấp các tính năng tích hợp quan trọng. Lập trình với Unity ngày càng trở nên đơn giản với các công cụ:

* Cho phép lập trình viên có thể chia sẻ các bản thiết kế.
* Cho phép sử dụng các tài nguyên hiệu ứng có sẵn trên Unity. Hoặc bạn có thể mua hoặc bán sản phẩm cho story của Unity.
* Cung cấp hệ thống toàn diện, dễ sử dụng cho lập trình viên.

Được coi là chìa khóa để phát triển phần mềm game, Unity 3D sở hữu nhiều ưu điểm.

* Về quy mô sử dụng phần mềm.

Theo một số thống kê, hiện nay có tới 47% số các nhà phát triển game sử dụng Unity 3D. Nó đang được coi là nền tảng thiết kế game phổ biến nhất trên toàn thế giới.

Cộng đồng Unity rất lớn mạnh. Với quy mô trên toàn thế giới, cộng đồng designer được thành lập ở nhiều nơi. Trong đó chia sẻ các kiến thức và kinh nghiệm về Unity 3D. Nhờ đó, kho sưu tập tài nguyên hiệu ứng có thể dử dụng trong Unity 3D ngày càng trở nên phong phú, đa dạng,…

* Chi phí sử dụng phần mềm.

Chính sách về chi phí của Unity được nới lỏng đến mức tối đa. Phần mềm được sử dụng miễn phí cho các cá nhân và doanh nghiệp có doanh thu thấp. Nếu muốn nâng cấp và sử dụng bản Pro, chi phí phải bỏ ra cũng rất nhỏ. Chỉ từ 1.500 USD cho một năm sử dụng Unity 3D.

* Cách sử dụng phần mềm đơn giản.

Là một phần mềm lập trình đa nền tảng, có thể dử dụng trên nhiều thiết bị khác nhau. Unity 3D cung cấp một hệ thống toàn diện cho cả nhiệm vụ soạn thảo và sửa lỗi. Điều này giúp phần mềm lập trình này có thể tối ưu hóa mọi công việc cho lập trình viên. Vì vậy, Unity 3D đảm bảo dễ sử dụng cho cả newbie.

* Nền tảng được phát triển liên tục, không ngừng nâng cấp phần mềm.

Phát triển Unity trong vòng 15 năm (từ năm 2015 đến nay). Unity Technologies đã liên tục cho ra đời các phiên bản khác nhau của Unity. Các phiên bản không ngừng được chú trọng phát triển tạo được những tính năng ấn tượng.

Sau khi khởi động Unity cửa sổ Editor xuất hiện: Cửa sổ Editor của Unity gồm nhiều Tab gọi là các view, và ta có thể tùy ý kéo thả sắp xếp các view theo ý thích.

#### **2.2.1.1 Cửa sổ Project**

* Mỗi Project của Unity đều chứa một thư mục Assets.Nội dung của thư mục này được hiển thị trong Project View. Đây là nơi chứa tất cả các assets để tạo Game của bạn như Scenes, Script, 3D models, Textures, Audio, Prefabs. Chúng ta không nên di chuyển asseets bằng cách sử dụng Window Explorer (hay Finder trong MAC), vì nó sẽ phá vỡ cấu trúc siêu dữ liệ của assets với nhau, và luôn sử dụng Project View để cấu trúc assets của mình.
* Để thêm assets vào project của mình, chúng ta có thể kéo thả bất kỳ file nào vào trong project view hoặc vào công cụ Assets – Import new Assets( Click chuột phải vào Project View). Scenes cũng được lưu trữ trong Project view, và đây là một level độc lập mang tính cá nhân. Chúng ta dễ dàng tạo một assets game trong Unity bằng cách chọn hình tam giác nhỏ nằm bên phải Create trong cửa sổ Project hoặc click chuột phải trong Project View, sau đó chọn assets tương ứng.
* Ngoài ra ta cũng có thể thêm các assets và đổi tên các assets một cách dễ dàng.

#### **2.2.1.2 Hierarchy**

* **Hierarchy** là hệ thống phân cấp.
* Trong Hierarchy chứa các GameObject hiện thời, một số có thể trỏ trực tiếp tới những file assets như 3D models, một số khác đại diện cho Prefabs – những đối tượng đã được tùy biến, dùng làm các công việc khác nhau sau này trong Game của bạn. Bạn có thể chọn và parenting Object trong Hierarchy . Một Object có thể được thêm vào hay loại bỏ trong scene và có thể thấy nó mất đi hay xuất hiện trong Hierarchy.

#### **2.2.1.3 Parenting**

* **Parenting** là thư mục chứa hay thư mục gốc, bất kỳ một game object nào muốn là đối tượng con(child) thì ta chỉ việc kéo thả đối tượng đó vào trong đối tượng dự tính làm Parenting trong Hierarchy và nó sẽ kế thừa chuyển động và quay của parenting

#### **2.2.1.4 Toolbar**

* **Toolbar** chứa 5 loại điều khiển cơ bản, mỗi loại giữ một vai trò quan trọng trong Editor.

A: Transform Tool: được dung với Scene view, như quay trái , phải, lên trên, xuống dưới, phóng to thu nhỏ đối tượng.

B: Transform Gizmo Toggles: dung cho việc thể hiện Scene view.

C: Play/Pause/Step Buttons: dung cho view game, chỵ game ngay trong Editor để kiểm tra.

D: Layer Drop-down kiểm soát đối tượng nào đang được thục hiện trong Scene view

E: Layout Drop-down kiểm soát sự sắp xếp cảu các Views.

#### **2.2.1.5 Scene View**

* **Scene View** là nơi bạn Design Game của bạn, đối tượng Maneuvering và Importanting trong Scene view (chuyển động và điều khiển) là hai trong số các chức năng quan trọng của Unity, ở góc bên phải của Scene là Scene Gizmo, nó thể hiện hướng nhìn trong không gian của camera trong Scene View hiện thời, cho phép thay đổi góc nhìn trực quan và nhanh chóng.
* Click lên các nhánh hình nón để chuyển qua các góc nhìn khác nhau có thể xem ở chế độ Isometric Model(tức ở dạng mặt cắt hai chiều), để chuyển qua chế độ 3D bạn Click vào hình vuông ở giữa hay giữ phím Shift + Click để chuyển đổi chế độ nhìn.
* Khi xây dựng một Game, bạn sẽ đặt rất nhiều đối tượng vào trong Game của bạn. Bạn có thể dử dụng các công cụ Transform Tools ở trong Toolbar để di chuyển, xoay, phóng to tu nhỏ từng đối tượng.Khi bạn chọn một đối tượng trong Scene View, xung quanh đối tượng được chọn sẽ có những thay đổi tương ứng với từng chế độ trong Transform Tools.Sau đó bạn thay đổi đối tượng tùy ý, nếu muốn chính xác bạn có thể chỉnh chi tiết ở bảng Inspector.
* Scene View Control Bar: Thanh điều khiển Scene view cho phép bạn xem screen Game với nhiều chế độ như đã phủ vật liệu (material), khung lưới(wireframe), RGB, Overdraw, …Bạn cũng có thể xem và cả nghe những hiệu ứng ánh sáng, âm thanh, những thành phần động trong Game… ngay tại Scene.

#### **2.2.1.6 Game View**

**Game View** được rendered từ những Camera trong Game. Đó là những gì được nhìn thấy khi hoàn tất, khi Game được xuất bản. Bạn sẽ cần ít nhất là một hoặc nhiều hơn số lượng các Camera để quết định những gì mà người chơi sẽ nhìn thấy khi họ chơi Game.

#### **2.2.1.7 Play Mode**

Sử dụng những nút trên Toolbar để điều khiển Editor Play Mode, và xem trước Game của bạn sẽ như thế nào khi chơi. Trong chế Play, mọi giá trị thay đổi sẽ được lưu tạm, và bị xóa khi thoát khỏi chế độ play.

#### **2.2.1.8 Inspector**

Games trong Unity được tạo ra bởi tập hợp rất nhiều GameObject, trong đó bao gồm meshes, scripts, âm thanh, hay những đối tượng Graphic như nguồn sáng v..v... Inspector sẽ hiển thị mọi thông tin về đối tượng đang làm việc một cách chi tiết, kể cả những Components được đính kèm và những thuộc tính của nó. Tại đây bạn có thể điều chỉnh, thiết lập mọi thông số chức năng của những mối liên kết GameObject-Component.

Mọi thuộc tính thể hiện trong Inspector đều có thể được tùy biến một cách trực tiếp. Ngay cả với những biến trong script cũng có thể được hiệu chỉnh mà không cần xem mã. Trong script, nếu bạn định nghĩa một giá trị là public cho một kiểu đối tượng (như GameObject hay Transform), bạn có thể drag-drop một GameObject hay một Prefab vào trong Inspector để gán giá trị cho nó.

Chúng ta có thể click lên icon hình bánh răng nhỏ bên phải hay click chuột phải lên tên cảu Component để xuất hện context menu dành cho những thiết lập của Component.

Inspector cũng sẽ thể hiện mọi thông số Import Setting của assets đang làm việc.

#### **2.2.1.9 Tùy biến không gian làm việc**

Bạn có thể tùy ý thay đổi cấu trúc của các Views bằng cách click-dragging những tab tới những nơi bạn muốn. Đặt cạnh một View nào đó, nó sẽ tự phân chia windows ra, đặt ra ngoài nó sẽ tự động tạo thành một windows riêng. (dịch thoáng).

Các tab cũng có thể kéo ra ngoài cửa sổ chính và được sắp xếp theo ý thích người dùng. Nếu cần thiết, bạn cũng có thể cho nó nằm ngang hàng với các Views khác trong cửa sổ chính.

Bạn cũng có thể lưu lại những thiết lập về cấu trúc của các Views, bằng cách click vào menu Dropdown Layout ở thanh Toolbar, chọn Save Layout... Đặt tên cho nó và sau đó bạn dễ dàng gọi nó ra bằng cách chọn nó trong Dropdown Menu Layout.

Bất kỳ lúc nào bạn cũng có thể right-click lên một tab của bất kỳ view nào để xem những tính năng khác như Maximize hay thêm vào một tab mới trong cùng một cửa sổ.

#### **2.2.1.10 Quản lí Asset**

* Workflow là “tiến trình công việc” hay “quy trình làm việc”, tức là khi bạn import hoặc tạo ra asset, bạn không cùng một lúc tạo ra toàn bộ, mà là theo trình tự, tùy theo nhu cầu, theo kịch bản, bạn import hoặc tạo ra asset theo thứ tự.
* Unity hỗ trợ import hầu như tất cả các định dạng 3D model hiện nay. Khi một đối tượng hay vật thể 3D được import hoặc tạo ra trong Unity, nó đều được gọi là asset.
* Khi lưu trữ asset, nên lưu trong thư mục Assets đặt trong thư mục Project của bạn. Lúc mở project, Unity sẽ tự động dò tìm và import tất cả vào trong Project.
* Khi bạn chọn một asset trong Project View, những thông số import cho asset sẽ hiển thị ở Inspector. Mỗi loại asset sẽ có mỗi kiểu thông số khác nhau.
* Chỉ cần click-drag các khung lưới được vẽ trong 3D (mesh) từ Project View vào cửa sổ Hierarchy hoặc Scene View để thêm nó vào trong Scene. Khi thêm một mesh vào trong scene, tức là bạn đã tạo ra một GameObject có Mesh Renderer Component. Tương tự khi thêm vật liệu, hoặc âm thanh, bạn cũng phải gán nó vào một GameObject đã có sẵn trong Scene hay Project.
* Sử dụng các kiểu Assets khác nhau
* Mối liên hệ giữa một số kiểu assets thông dụng:
* Một Texture (kết cấu) được áp cho một Material(vật liệu).
* Một Material được áp cho một GameObject (với một Mesh Renderer Component)
* Một Animation được áp cho một GameObject (với một Animation Component)
* Một file âm thanh được áp cho một GameObject (với một Audio Source Component)
* Prefabs là một tính năng hữu ích trong Unity, nó là một bộ sưu tập của nhiều GameObject và Components mà có thể được dùng lại nhiều lần trong scene. Nhiều đối tượng riêng biệt có thể được tạo ra chỉ bằng một Prefab, được gọi là Intance.
* Lấy ví dụ những cái cây trong game. Ta tạo ra một Prefab cho cây, sau đó có thể thêm nó vào trong game để tạo thành một vườn cây. Bởi vì tất cả cây đều cùng một Prefab, nên sau này, nếu bạn cần thay đổi màu lá, màu vỏ cây v..v... cho toàn bộ vườn cây, bạn chỉ cần đơn giản hiệu chỉnh thông số trong Prefab, sau đó áp nó cho toàn bộ đối tượng bằng cách chọn GameObject->Apply Changes to Prefab từ menu chính.
* Thật dễ dàng và nhanh chóng, Prefab sẽ giúp bạn tiết kiệm rất nhiều thời gian trong việc quản lý và hiệu chỉnh asset.
* Tưởng tượng Prefab như là một khung sườn, một khuôn mẫu. Nghĩa là khi Prefab thay đổi, tất cả những bản sao sẽ được cập nhật theo. Có nhiều cách để thay đổi, như bạn có thể thay đổi chính từ Prefab gốc, hoặc từ những bản sao của nó rồi áp ngược lại cho toàn bộ Prefab.
* Để tạo ra một Prefab từ một GameObject trong scene, đầu tiên, khởi tạo trước một Prefab mới trong Project View, đặt tên cho Prefab đó theo ý bạn thích, sau đó, click và drag đối tượng GameObject mà bạn muốn nó trở thành một Prefab vào trong Prefab, khi đó tên của GameObject sẽ chuyển sang xanh dương.
* Bạn đã làm quen với việc import, thay đổi, và liên kết asset trong một Prefab. Giờ bạn muốn có một chút thay đổi với asset? Chỉ cần double-click nó trong Project View. Chương trình chỉnh sửa tương ứng sẽ chạy và bạn có thể chỉnh sửa theo ý thích, khi hoàn tất, chỉ cần save nó lại, Unity sẽ tự động nhận biết sự thay đổi và import lại asset đó một cách nhanh chóng. Những asset liên kết trong Prefab cũng sẽ thay đổi theo.
* Ví dụ bạn thêm một bảng hiệu trong game bằng file Photoshop. Double-click lên nó, Photoshop sẽ chạy, bạn chỉnh sửa bảng hiệu trong Photoshop, save lại. Quay lại Unity, bạn sẽ thấy nó đã thay đổi, thật tiện lợi phải không nào?
* Thêm labels cho những asset quả không là một ý tưởng tồi nếu bạn muốn giữ nó ngăn nắp, nhằm làm cho việc tìm kiếm và quản lý, sử dụng nó dễ dàng hơn.
* Các bước đính thêm label cho asset:
* Chọn asset mà bạn muốn thêm label. (Bằng project view).
* Trong cửa sổ Inspector click lên biểu tượng "Add Label" nếu bạn chưa có label nào trước đó.
* Nếu asset đã có label rồi, click vào đó.
* Gõ label vào đó.
* Ghi chú:
* Bạn có thể đính nhiều label cho một asset.
* Để ngăn cách, hay tạo mới label khác, chỉ cần dùng khoảng trắng space bar hay enter và viết tiếp label khác.

#### **2.2.1.10 Khởi tạo Scene**

* Scene chứa tất cả các đối tượng trong Game, nó thường được dùng để tạp màn hình chính trong game, những level độc lập, và mọi thứ khác trong game. Tức là mỗi Scene là một tầng độc lập, và trong mỗi Scenen bạn có thể lập môi trường, vật thể, chi tiết, kể cả design và xây dựng game từ những thứ nhỏ nhất.
* Khi bạn đã có một prefabs bạn có thể copy nó ra thành nhiều bản một cách dễ dàng, đó gọi là Instancing prefabs. Để tạo Instancing của bất kỳ một Prefabs nào, chỉ cần drag nó từ trong Project view qua Hierarchy hay trực tiếp trong Scene view.
* Khi bạn đã có một Prefabs hoặc game Object đang làm việt bạn có thể thêm các tính năng cho nó bằng cách sử dụng Component (Script cũng là một Component). Để thêm một Component chỉ cần chọn đối tượng rồi chọn Component tương ứng trong menu Component, bạn sẽ thấy xuất hiện các thông số trên cửa sổ Inspector. Script cũng được mặc định trong menu Component/
* Trong trường hợp việc thêm Componet làm gãy liên kết giữa những GameObject với prefab của nó thì bạn có thể dùng Gameobject -> Apply Changes to Prefab từ menu để thiết lập lại liên kết đó.
* Khi đã có một GameObject trong Scene bạn có thể dùng công cụ Transform Tools để di chuyển, hoặc thay đổi giá trị Transform trong Inspector để có vị trid chnhs xác hơn.
* Camera chính là mắt trong Game, mọi thứ người chơi có thể nhìn thấy đều thông qua Camera, nó cũng được hiển thị và thay đổi, di chuyển như bất kỳ một GameObject nào, và nó thực chất cũng là một GameObject và nó có đầy đủ thuộc tính của GameObject ngoài ra nó còn có các thuộc tính riêng cho Camera. Unity đã cung cấp một số Script sẵn kèm theo gói cài đặt mẫu Asset khi bạn khởi tạo mới một project.
* Hầu như chúng ta phải thêm nguồn sáng cho các bối cảnh của Game.
* Có ba kiểu nguồng sáng cơ bản, mỗi nguồn sáng khác nhau sẽ thay đổi sắc thái của game rất nhiều.
* Dirictional Light: Có thể hiểu là với nguồn sáng này ta có thể định hướng chiếu cho nó, các tia chiếu là song song
* Point Light: là điểm sáng, thường dùng để tạo các điểm sáng nhỏ trên các Character, hay các cảnh trong Game theo đúng ý tưởng của chủ nhân.
* Spotlight: Đèn tại chỗ chỉ chiếu theo một chiều hình nón, thường dùng là hiệu ứng chiếu cho đèn các loại xe, đèn pin, ánh sáng hắt vào object trong các cảnh…
* Camera :
* Như Camera được sử dụng trong các bộ phim , Camera trong Unity cũng được dùng để hiển thị game trên thế giới cho người chơi. Nó cũng được coi là một Game Object trong Unity. Bạn có thể xoay, di chuyển…. tùy chỉnh nó theo ý tưởng của bạn
* Camera được sử dụng để hiển thị cảnh trong game, chúng ta có thể làm cho game của mình trở nên độc đáo hơn nhờ tùy chỉnh Camera. Tron g một cảnh, chúng ta có thể có một hoặc rất nhiều Camera.
* Đặc điểm Camera:
* Clear Flags: Xác định các bộ phận mà màn hình sẽ bị xóa. Thuận tiện khi sử dụng nhiều máy ảnh và để vẽ nhiều đổi tượng khác nhau. Không xóa nó sẽ hiển thị màu đen xì.
* Background: Màu nền cho phần màn hình còn lại.
* Culling Mask: Chỉ định các lớp đối tượng của bạn trong Inspector. Cho phép hoặc bỏ qua các đối tượng được hiển thị trong Camera.
* Projection:
* Perspective: Camera hiển thị các đối tượng theo phối cảnh trọn vẹn.
* Orthographic: Hiển thị các đối tượng như một thể thống nhất, không có theo nghĩa của phối cảnh(Perspective).
* Size: Kích thước quan sát của Camera khi chọn phép chiếu là: Orthographic.
* Field of view: Chiều rộng của góc nhìn Camera. Được đo bằng độ dọc theo trục Local Y.
* Clipping plane:
* Near: Khoảng cách gần nhất hiển thị trong Camera.
* Far: Khoảng cách xa nhất hiển thị được trong tầm nhìn của Camera.
* Normalized View Port Rect:
* X: Bắt đầu từ vị trí ngang mà Camera hiển thị.
* Y: Bắt đầu từ vị trí thẳng đứng mà Camera hiển thị.
* Depth: Vị trí. Camera có một giá trị lớn hơn sẽ được hiển thị lên đầu. Tức là cái nào có giá trị lớn hơn thì nó sẽ được ưu tiên được hiển thị lên màn hình.
* Rendering path: Tùy chọn cho việc xác định phương pháp vẽ những gì sẽ được hiển thị lên Camera.
* Use Player Setting: Sử dụng cài đặt Player.
* Vertex Lit: tất cả các đối tượng được đưa ra bởi máy ảnh sẽ được trả lại như đối tượng Vertex-Lit.
* Forward: tất cả các đối tượng sẽ được trả về như một tài liệu. Giống như tiêu chuẩn Unity 2.x.
* Deferred Lighting (Unity Pro only): Mất phí, không chơi. Tất cả các đối tượng sẽ được rút ra mà không có ánh sáng, sau đó tất cả ánh sáng được trả lại trong hàng đợi Render.
* Target Texture: Phiên bản free 4.0 đã có. Tham chiếu đến một texture. Cho phép dựng hình High Dynamic Range cho máy ảnh này.
* ĐỊA HÌNH
* Sử dụng: Trên Menu chính, chọn Create Terrain
* Terrain Script: Tại đây chúng ta có thể vẽ địa hình, nâng/hạ địa hình, vẽ cây, cỏ, cài đặt cho Terrain.
* Brushes: Gồm một số dạng bút lông phục vụ việc vẽ địa hình.
* MẮT LƯỚI(MESHES)
* Khi một đối tượng mô hình 3D được Import. Unity sẽ gán cho nó một đại diện như một bộ lưới. Lưới phải được đính kèm với một Game Object bằng cách sử dụng một thành phần Filter Meshes.
* Định dạng 3D
* Export file định dạng 3D như: \*.FBX, \*.OBJ.
* Ưu điểm:
* Chỉ xuất ra dữ liệu bạn cần.
* Kiểm tra lại dữ liệu.(Import lại)
* Nhìn chung tập tin rất nhẹ.
* Khuyến khích phương pháp Module.
* Hỗ trợ các gói 3D.
* Nhược điểm:
* Có thể gây chậm trong việc tạo mẫu và sử dụng lại.
* Khó có thể theo dõi
* Sở hưu các file ứng dụng 3D như: .Max, .Blend.
* Ưu điểm:
* Nhanh chóng lặp đi lặp lại các quá trình.
* Đơn giản việc sử dụng ban đầu.
* Nhược điêm:
* Công cụ hay bản sao của định dạng đó phải được cài đặt trên máy.
* Tập tin có thể cồng kềnh với các file dữ liệu không cần thiết.
* Các tập tin lớn có thể làm chậm cập nhật của Unity.
* Chưa xác nhận khó khăn để khắc phục sự cố các vấn đề.
* Animation View Guide (Legacy)
* Ctrl + 6(Windows): mở Animation View.
* Để sử dụng Animation cơ bản, trước tiên chọn Create new clip. Animation sẽ hiển thị các tùy chọn trên Animation View hoặc trên Inspector. Để áp dụng animation cho một đối tượng nào đó bạn phải chọn vào đối tượng đó à thuộc tính animation tương ứng sẽ hiển thị trên Animation View tương ứng.
* Trên Animation view có thể chọn một thuộc tính hoặc một vào thuộc tính để tùy biến hoạt hình theo các thuộc tính của đối tượng. Chính sửa Curves bằng cách tại các frame đã được tạo, kéo đường kẻ theo ý bạn muốn. Nó sẽ hiển thị ngay trên Scene View hoặc trên Inspector.
* Tại các điểm Frame, có thể xóa frame, chỉnh/chọn tiếp tuyến, ….
* Có thể sử dụng button nhỏ (+) để thêm Frame hoặc Event cho Animation hoặc click-right lên thanh frame.
* Animation Scripting (Legacy): hệ thống Animation của Unity cho phép bạn tạo đẹp nhân vật của mình bằng các pha trộn các vật liệu, chất phụ gia, đồng bộ hóa thời gian, chu kì, các lớp ảnh động, kiểm soát tất cả các các khía cạnh phát lại của hình ảnh động(thời gian, tốc độ, sự pha trộn – trọng lượng)
* Một nhân vật hoạt hình luôn có hai vấn đề chính đó là sự di chuyển của nó trong thế giới và tạo hiệu ứng cho phù hợp.
* Animation Blending:
* Trong các trò chơi ngày nay, hình ảnh động hòa trộn làm cho nhân vật trở nên trơn chu hơn. Tại bất cứ thời điểm nào trong game, bạn đều có thể chuyển đổi qua lại giữa hình ảnh động và hình ảnh tĩnh(hình ảnh nhàn rỗi).
* Đây là nơi các hình ảnh động được Blending. Tất cả các hình ảnh động được thêm, pha trộn vào nhau để được hình ảnh động cuối cùng.
* Animation Layers: là một khái niệm hữu ích giúp bạn nhóm các hình ảnh động và ưu tiên cho chúng.
* Hệ thống hoạt hình của Unity có thể pha trộn giữa nhiều Clip hoạt hình theo ý bạn muốn. Có thể gán lượng pha trộn bằng tay hay sử dụng animation.CrossFade()(Cái này sẽ sinh liều lượng pha trộn tự động).
* Lượng Blen luôn được bình thường hóa trước khi được áp dụng. Tưởng tượng như bạn có một chu kỳ Walk và một chu kỳ chạy, cả hai đều có lượng là 1(100%). Khi bình thường hóa thì mỗi chu kỳ chiếm 50% lượng. Tuy nhiên nếu bạn muốn cho bên nào đó có lượng lớn hơn thì bạn có thể Blen bằng tay.
* Animation Mixing: Animation Mixing cho phép cắt giảm số Animation mà bạn cần trong game bởi chỉ cần một số Animation được áp dụng trên thân đối tượng. Điều đó có nghĩa là đối tượng hoạt hình có thể được sử dụng chung với các đối tượng hoạt hình khác.
* Sound
* Audio Listener: Giống như một thiết bị Microphone. Nó nhận đầu vào từ bất cứ một nguồn âm thanh nào trong Scenes và các âm thanh thông qua máy tính. Đối với hầu hết các ứng dụng, nó là ý nghĩa nhất để gắn tai nghe lên Main Camera.
* Ranh giới/độ ảnh hưởng âm thanh nghe của Reverb Zone được áp dụng cho toàn Scene thì có thể nghe thấy âm thanh trên bất cứ địa điểm nào của Scene.
* Properties: Phải thêm âm thanh vào Scene và chỉnh sửa thuộc tính của âm thanh bên Inspector View.
* Audio Clip: Chọn file âm thanh cho Scene.
* Mute: bật/tắt âm thanh.
* Bypass Effects: Lọc nhanh hiệu ứng “by-pass” to audio source. Một cách dễ dàng nhất để bật/tắt hiệu ứng(effect).
* Play on Wake: Nếu enable, âm thanh sẽ được chạy ngay khi ra mắt Scene. Nếu để Disable, khi cần chạy âm thanh chúng ta phải gọi phương thức/chức năng Play() từ Script.
* Priority: Xác định độ ưu tiên của Audio Source trong số tất cả các Audio Source có trong Scene.
* Priority = 0: Rất quan trọng. Sử dụng ở mức 0 cho bài nhạc để tránh bị thường xuyên trao đổi.
* Priority =256: Độ quan trọng thấp nhất(Độ ưu tiên ở mức thấp nhất).
* Mặc định(default) = 128.
* Volumn: Làm thế nào để âm thanh lớn trên một bộ phận của thế giới Scene từ Audio Source.
* Pitch: Số dùng thay đổi tốc độ của âm thanh. Tốc độ bằng 1 là ở mức chạy bình thường.
* 3D Source Setting: Cài đặt, cái mà được áp dụng cho Audio source nếu Audio Clip là một file âm thanh 3D.
* Pan Lever: Cài đặt, làm thế nào để máy 3D có được hiệu ứng từ Audio source.
* Spread: Cài đặt góc ảnh hưởng tới âm thanh 3D Stereo nếu Audio Clip là một âm thanh 3D.
* Doppler Lever: Xác định bao nhiêu hiệu ứng âm thanh Dopper sẽ được áp dụng cho Audio Source(Nếu cài đặt là 0 thì sẽ không có hiệu ứng nào được áp dụng).
* Min Distance: Với Min Distance, âm thanh sẽ đạt mức to nhất có thể. Ngoài Min Distance, nó sẽ bắt đầu suy yếu đi. Tăng MinDistance của âm thanh để được âm thanh lớn hơn trong thế giới 3D và giảm MinDistance để cho âm thanh nhỏ hơn.
* Max Distance: Khoảng cách mà âm thanh dừng suy giảm. Sau khoảng cách này nó sẽ tạm ngừng không cho MinDistance tăng nữa. Tức là đây là giá trị Max mà MinDistance sẽ đạt tới.
* Rolloff Mode: Làm thế nào nhanh chóng mất dần âm thanh. Giá trị cao hơn, gần gũi hơn Listener có được trước khi nghe âm thanh.(Điều này được xác định bởi một Graph).
* Logarithmic Rolloff: Âm thanh là lớn khi bạn ở gần nguồn âm thanh. Nhưng khi chúng ta nhận được từ các đối tượng nó sẽ giảm đi nhanh chóng một cách đáng kể.
* Linear Rolloff: Càng xa nguồn âm thanh, bạn nghe thấy càng ít.
* Custom Rolloff:
* 2D Sound Setting: Cài đặt áp dụng cho nguồn âm thanh, nếu Audio Clip là âm thanh 2D.
* Pan 2D: Cài đặt hiệu ứng trên nguồn âm thanh.
* Type Rolloff
* Distance Function:
* Volume: Amplitude(0.0 - 1.0) over distance.
* Pan: Left(-1.0) to Right(1.0) over distance.
* Spread: Angle (degrees 0.0 - 360.0) over distance.
* Low-Pass (only if LowPassFilter is attached to the AudioSource): Cutoff Frequency (22000.0-10.0) over distance.
* Tạo một nguồn âm thanh:
* Import file âm thanh của bạn vào project. Nó là những Audio Clip.
* Tới Game Object -> Create Empty từ menubar.
* Trên file âm thanh đang chọn trên Game Object View, chọn Component à Audio à Audio Source.
* Thuộc tính của nguồn âm thanh sẽ được hiển thị trên Inspector View

### 2.2.2 Giới thiệu Visual Studio 2019



**Hình 2 Phần mềm Visual Studio 2019**

(Nguồn: *https://lapcamerahanoi.com/key-visual-studio-2019/)*

* Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft.
* Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu.
* Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C, C++ và C++/CLI , VB.NET , C# và F#. Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng rẽ. Nó cũng hỗ trợ XML/XSLT, HTML/XHTML, JavaScript và CSS
* Một số tính năng Visual Studio Code:
* Cải thiện trải nghiệm tìm kiếm nâng cao

Một trong những tính năng hữu ích trong Visual Studio 2019 là hộp tìm kiếm được cập nhật. Trong một phiên bản trước, nó được gọi là Khởi động nhanh. Trải nghiệm tìm kiếm mới trong Visual Studio 2019 nhanh hơn và hoạt động hiệu quả để có kết quả tốt nhất.

Bây giờ kết quả tìm kiếm sẽ hiển thị các đề xuất khi bạn nhập vào hộp tìm kiếm và nó sẽ tự động trình bày kết quả. Bạn cũng có thể sử dụng các phím tắt trong khi tìm kiếm và điều này sẽ giúp bạn nhớ các truy vấn tìm kiếm để sử dụng trong tương lai và nhớ lại bất cứ khi nào bạn xây dựng các ứng dụng web mới.

* Tái cấu trúc được cải thiện

Tái cấu trúc C # mới giúp tổ chức mã của bạn dễ dàng hơn. Chỉ cần gọi các phép tái cấu trúc bằng cách nhấn Ctrl +. và chọn hành động bạn muốn thực hiện.

* IntelliCode tốt hơn

Visual Studio IntelliCode là một phần mở rộng giúp tăng cường nỗ lực phát triển phần mềm của bạn bằng cách sử dụng trí tuệ nhân tạo (AI).

Nó cung cấp các đề xuất và các đối tượng mã hóa trong quá trình phát triển. Intellicode sẽ đạt được sức mạnh khi sử dụng 2000 dự án nguồn mở khác Quy trình làm việc trên nền tảng đám mây

Cửa sổ bắt đầu trình bày cho bạn một số tùy chọn để giúp bạn mã hóa nhanh chóng. Trước tiên, chúng tôi đã đặt tùy chọn để sao chép hoặc kiểm tra mã từ repo.

* Chia sẻ trực tiếp

Visual Studio Live Share là một dịch vụ dành cho nhà phát triển cho phép bạn chia sẻ một cơ sở mã và bối cảnh của nó với đồng đội và nhận được sự cộng tác hai chiều ngay lập tức từ bên trong Visual Studio. Với Live Share, đồng đội có thể đọc, điều hướng, chỉnh sửa và gỡ lỗi một dự án mà bạn đã chia sẻ với họ và thực hiện một cách liền mạch và an toàn.

* Đánh giá mã tích hợp

Chúng tôi đang giới thiệu một tiện ích mở rộng mới mà bạn có thể tải xuống để sử dụng với Visual Studio 2019. Với tiện ích mở rộng mới này, bạn có thể xem lại, chạy và thậm chí gỡ lỗi các yêu cầu kéo từ nhóm của mình mà không cần rời khỏi Visual Studio. Chúng tôi hỗ trợ mã trong cả kho GitHub và Azure DevOps.

* Tính năng tìm kiếm trong khi gỡ lỗi

Bạn đã biết rằng gỡ lỗi là quá trình xác định lỗi và loại bỏ các lỗi đó trong phần mềm. Tìm kiếm các đối tượng và các giá trị phù hợp trong quá trình gỡ lỗi rất hữu ích cho các nhà phát triển.

Trong Visual Studio 2019 tìm kiếm trong khi gỡ lỗi được thêm Địa phương, Ô tô và cửa sổ Xem được sử dụng để tìm các giá trị và đối tượng. Dưới đây hoạt hình sẽ giải thích quá trình thực tế gỡ lỗi cửa sổ tìm kiếm trong VS2019.

* Hiệu suất tốt hơn

Cho dù bạn đang mã hóa trong C ++ hay .NET Core, các điểm dừng dữ liệu có thể là một lựa chọn tốt để chỉ đặt các điểm dừng thông thường. Điểm dừng dữ liệu cũng tuyệt vời cho các tình huống như tìm kiếm nơi một đối tượng toàn cầu đang được sửa đổi hoặc được thêm hoặc xóa khỏi danh sách.

* Dọn mã trong một lần nhấp

Lệnh dọn mã mới trong Visual Studio 2019 là phát hiện các cảnh báo và đề xuất bằng nút một lần bấm. Lệnh này sẽ giúp định dạng mã và thực hiện các thay đổi thành định dạng mã hóa được đề xuất bởi các tệp và cài đặt .editorconfig. Nó cũng cho phép lưu các bộ sưu tập các bản sửa lỗi làm hồ sơ mặc định

* Hỗ trợ Xamarin

Visual Studio 2019 hỗ trợ Xamarin.Forms 4.0 cùng với các tính năng mới bao gồm các mẫu dự án Shell mới và các điều khiển CollectionView mới. Các mẫu Shell giúp người dùng thiết kế các ứng dụng dành cho thiết bị di động với giao diện được cải tiến và được liên kết chặt chẽ với Material Design. Xamarin CollectionView là một giải pháp thay thế tốt hơn và linh hoạt hơn cho các điều khiển ListView.

### 2.2.3. Giới thiệu về MongoDB



**Hình 3 Database MongoDB**

(Nguồn*: https://viblo.asia/p/mongodb-la-gi-co-so-du-lieu-phi-quan-he-bJzKmgoPl9N.html*)

* MongoDB là một hệ quản trị cơ sở dữ liệu mã nguồn mở, là CSDL thuộc NoSql và được hàng triệu người sử dụng.
* MongoDB là một database hướng tài liệu (document), các dữ liệu được lưu trữ trong document kiểu JSON thay vì dạng bảng như CSDL quan hệ nên truy vấn sẽ rất nhanh.
* Với CSDL quan hệ chúng ta có khái niệm bảng, các cơ sở dữ liệu quan hệ (như MySQL hay SQL Server...) sử dụng các bảng để lưu dữ liệu thì với MongoDB chúng ta sẽ dùng khái niệm là collection thay vì bảng
* So với RDBMS thì trong MongoDB collection ứng với table, còn document sẽ ứng với row , MongoDB sẽ dùng các document thay cho row trong RDBMS.
* Các collection trong MongoDB được cấu trúc rất linh hoạt, cho phép các dữ liệu lưu trữ không cần tuân theo một cấu trúc nhất định.
* Thông tin liên quan được lưu trữ cùng nhau để truy cập truy vấn nhanh thông qua ngôn ngữ truy vấn MongoDB

**2.2.3.1. Ưu điểm của MongoDB**

* Do MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ có các kích cỡ và các document khác nhau, linh hoạt trong việc lưu trữ dữ liệu, nên bạn muốn gì thì cứ insert vào thoải mái.
* Dữ liệu trong MongoDB không có sự ràng buộc lẫn nhau, không có join như trong RDBMS nên khi insert, xóa hay update nó không cần phải mất thời gian kiểm tra xem có thỏa mãn các ràng buộc dữ liệu như trong RDBMS.
* MongoDB rất dễ mở rộng (Horizontal Scalability). Trong MongoDB có một khái niệm cluster là cụm các node chứa dữ liệu giao tiếp với nhau, khi muốn mở rộng hệ thống ta chỉ cần thêm một node với vào cluster:
* Trường dữ liệu “\_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
* Khi có một truy vấn dữ liệu, bản ghi được cached lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.
* Hiệu năng cao: Tốc độ truy vấn (find, update, insert, delete) của MongoDB nhanh hơn hẳn so với các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Với một lượng dữ liệu đủ lớn thì thử nghiệm cho thấy tốc độ insert của MongoDB có thể nhanh tới gấp 100 lần so với MySQL.

**2.2.3.2. Nhược điểm của PostgreSQL:**

* Một ưu điểm của MongoDB cũng chính là nhược điểm của nó. MongoDB không có các tính chất ràng buộc như trong RDBMS nên khi thao tác với mongoDB thì phải hết sức cẩn thận.
* Tốn bộ nhớ do dữ liệu lưu dưới dạng key-value, các collection chỉ khác về value do đó key sẽ bị lặp lại. Không hỗ trợ join nên dễ bị dữ thừa dữ liệu.
* Khi insert/update/remove bản ghi, MongoDB sẽ chưa cập nhật ngay xuống ổ cứng, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng điêù này sẽ là nhược điểm vì sẽ có nguy cơ bị mất dữ liệu khi xảy ra các tình huống như mất điện..

## 2.3. Tìm hiểu về .NET

### 2.3.1 Ngôn ngữ C#

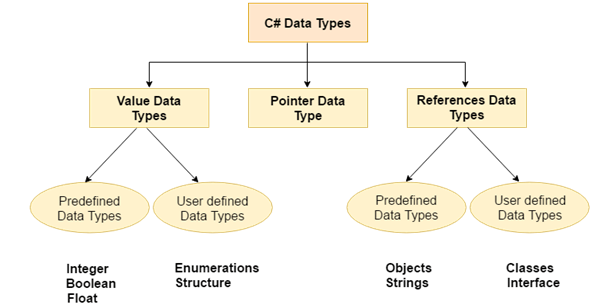
**2.3.1.1 Lịch sử phát triển của C#**



**Hình 4 Lịch sử phát triển C#**

(Nguồn: [https://kcntt.duytan.edu.vn/Home/ArticleDetail/vn/168/3633/lich-su-phat-trien-cua-c)](https://vi.wikipedia.org/wiki/Java_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh))

**2.3.1.2 Những đặc trưng của ngôn ngữ C#**



**Hình 5 Đặc trưng ngôn ngữ C#**

- Ngôn ngữ lập trình C# có những đặc trưng cơ bản sau:

* Đơn giản(Simple)
* Ngôn ngữ lập trình hiện đại(Modern programming language)
* Lập trình hướng đối tượng(Object oriented)
* Kiểu an toàn(Type safe)
* Khả năng tương tác(Interoperability)
* Có thể mở rộng và cập nhật(Scalable and Updateable)
* Hướng thành phần(Component oriented)
* Ngôn ngữ lập trình có cấu trúc(Structured programming language)
* Nhiều thư viện(Rich Library)
* Tốc độ nhanh(Fast speed)

### 2.3.2 Tìm hiểu về .NET Framework

**2.3.2.1 .NET Framework**

* .NET Framework được Microsoft đưa ra chính thức từ năm 2002. .NET Framework chỉ hoạt động trên Windows. Những nền tảng ứng dụng như WPF, Winforms, ASP.NET(1-4) hoạt động dựa trên .NET Framework.
* Mono là phiên bản cộng đồng nhằm mang .NET đến những nền tảng ngoài Windows. Mono được phát triển chủ yếu nhằm xây dựng những ứng dụng với giao diện người dùng và được sử dụng rất rộng rãi: Unity Game, Xamarin…
* Cho đến năm 2013, Microsoft định hướng đi đa nền tảng và phát triển .NET core. .NET core hiện được sử dụng trong các ứng dụng Universal Windows platform và ASP.NET Core. Từ đây, C# có thể được sử dụng để phát triển các loại ứng dụng đa nền tảng trên các hệ điều hành khác nhau (Windows, Linux, MacOS,…)

## 2.4. Tổng quan về RestFul API



**Hình 6 Tổng quan ResFul API**

(Nguồn: <https://movan.vn/rest-api-gioi-thieu-rest-api/>)

- REST (REpresentational State Transfer) được đưa ra vào năm 2000, trong luận văn tiến sĩ của Roy Thomas Fielding (đồng sáng lập giao thức HTTP).Nó là một dạng chuyển đổi cấu trúc dữ liệu, là một phong cách kiến ​​trúc cho việc thiết kế các ứng dụng có kết nối. Nó sử dụng HTTP đơn giản để tạo cho giao tiếp giữa các máy. Vì vậy, thay vì sử dụng một URL cho việc xử lý một số thông tin người dùng, REST gửi một yêu cầu HTTP như GET, POST, DELETE, vv đến một URL để xử lý dữ liệu.

- API (Application Programming Interface) là giao diện lập trình ứng dụng giúp tạo ra các phương thức kết nối với các thư viện và ứng dụng khác nhau.

- REST API là một ứng dụng chuyển đổi cấu trúc dữ liệu có các phương thức để kết nối với các thư viện và ứng dụng khác. REST API không được xem là một công nghệ, nó là một giải pháp để tạo ra các ứng dụng web services thay thế cho các kiểu khác như SOAP, WSDL (Web Service Definition Language),…

- Các ưu điểm của REST:

* Giúp cho ứng dụng trở nên rõ ràng hơn.
* REST URL đại diện cho resource chứ không phải là hành động.
* Dữ liệu được trả về với nhiều định dạng khác nhau như: xml, html, rss, json …
* Code đơn giản và ngắn gọn.
* REST chú trọng vào tài nguyên hệ thống.

# CHƯƠNG 3: Xây dựng game 2D Soldier's Avenger

## 3.1 Khảo sát hệ thống Game 2D Soldier's Avenger

Hiện nay, game điện tử rất phát triển và phủ rộng toàn cầu. Ngoài các game riêng biệt trên các nền tảng với từng độ tuổi phù hợp với nội dung game thì dạng game phiêu lưu 2D phù hợp với mọi lứa tuổi, mọi thiết bị là điểm đến phù hợp nhất để tiếp cận người chơi.

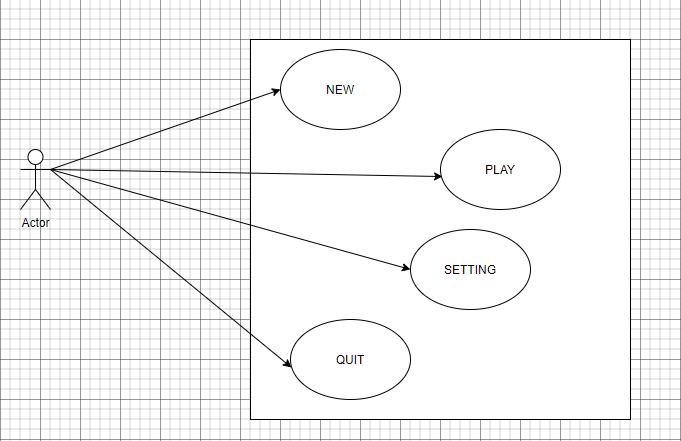
Qua quá trình tìm hiều, xác định xây dựng 1 game cần:

* Xây dựng Core cho tất cả đối tượng
* Xây dựng hệ thống Player dựa trên Core
* Xây dựng hệ thống Enemy dựa trên Core
* Xây dựng hệ thống Tile Map
* Xây dựng hệ thống bẫy.
* Xây dựng hiệu ứng chiến đấu.
* Xây dựng hệ thống âm thanh nền và âm thanh chiến đấu.
* Xây dựng UI game.

## 3.2 Mô hình Usecase

### 3.2.1 Sơ đồ Usecase cho người chơi

Giao diện người dùng chính có các chức năng :

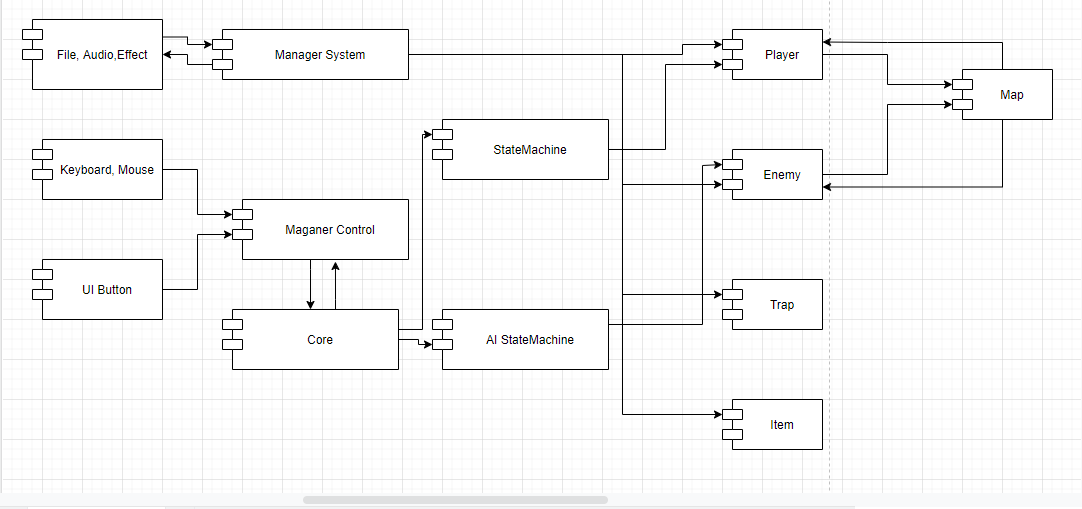
* New : chơi mới.
* Play: tiếp tục chơi.
* Setting: cài đặt trong game: graphic, screen, volume.
* Quit: thoát game.

Hình :

### 3.2.2 Các nhóm đối tượng :

* Hệ thống quản lí :
* quản lí Audio, Image các đối tượng.
* Lưu và load dữ liệu người chơi bao gồm level, coin và currentExp.
* Hệ thống điều khiển:
* hệ thống điều khiển Keyboard và Mouse trên nền tảng PC.
* hệ thống điều kiển UI Button trên màn hình ở nền tảng Android.
* Hệ thống UI:
* Quản lí Quest, màn chơi
* Quản lí Setting cho game: âm thanh, màn hình, đồ họa.
* Hệ thống Map :
* Flatform: đất, tường, mỏm đá tác động trực tiếp với player và enemy.
* Background: nền sau, tạo không khí địa điểm chiến đấu.
* Decorate front: nền trang trí ở phía trước layer Player và Enemy: đèn, cây, cỏ, nấm…
* Decorate back: nền trang trí ở phía sau layer Player và Enemy: đèn, cây, cỏ, nấm, bụi cây…
* Hệ thống Bẫy:
* Thiết kế Animation: mỗi loại bẫy có cách hoạt động và tác động khác nhau.
* Tác động Player và gây sát thương.
* Xây dựng hệ thống item:
* Xây dựng item: item thường như coin, hạt exp và item vật phẩm như poision.
* Xây dựng Inventory cho Player chứa các item nhặt được
* Xây dựng hệ thống hiệu ứng chiến đấu :
* HitParticle : hiệu ứng bị chém
* Hiệu ứng máu, thịt khi chết.
* Hiệu ứng lướt của Player: Image tạo bóng lướt.
* Xây dựng Core cho các thực thể trong game:
* Hệ thống di chuyển: đi, đứng.
* Hệ thống tương tác: Tương tác với vật phẩm và đối tượng khác.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm đất, chạm tường, chạm trần.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.
* Xây dựng hệ thống Player :
* Kế thừa Core
* Animation cho các trạng thái.
* StateMachine cho Player chuyển đổi trạng thái.
* Xây dựng hệ thống Enemy :
* Kế thừa Core
* Animation cho các trạng thái.
* StateMachine cho Enemy chuyển đổi trạng thái.
* Sau khi chết sẽ làm rơi item coin hoặc hạt exp.

### 3.2.3 Sơ đồ Component :



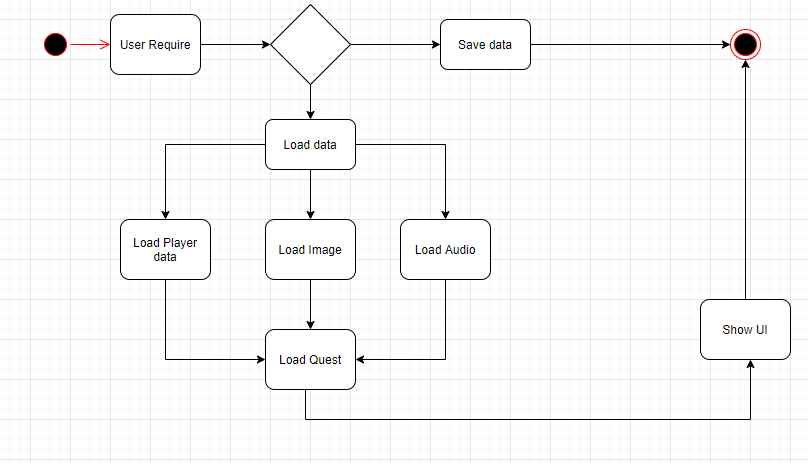
### Hình : sơ đồ Component tổng quát

## 3.3 Xây dựng các hệ thống

### 3.3.1 Xây dựng Hệ thống quản lí

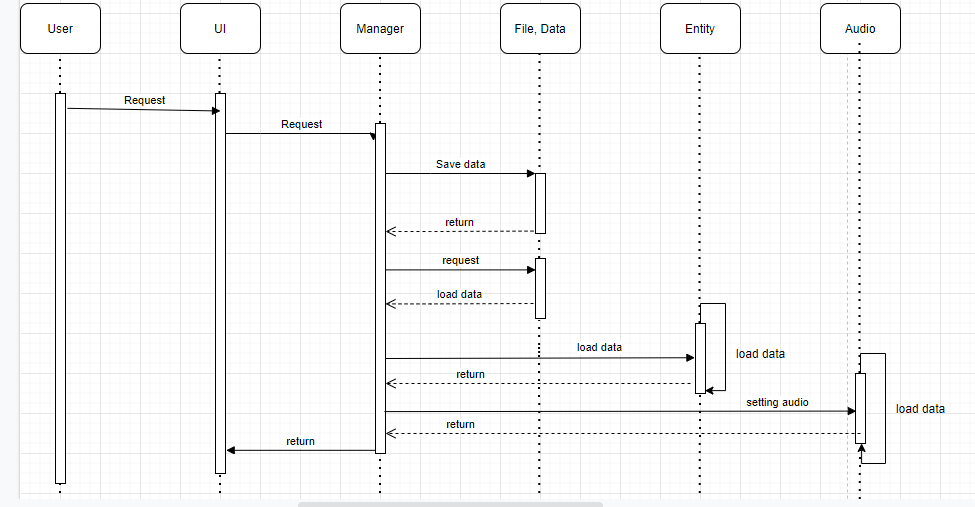
Chức năng:

* quản lí Audio, Image các đối tượng.
* Lưu và load dữ liệu người chơi bao gồm level, coin và currentExp lên file.

Sơ đồ activity:

Hình : Sơ đồ Activity Hệ thống quản lí

Lược đồ sequence:

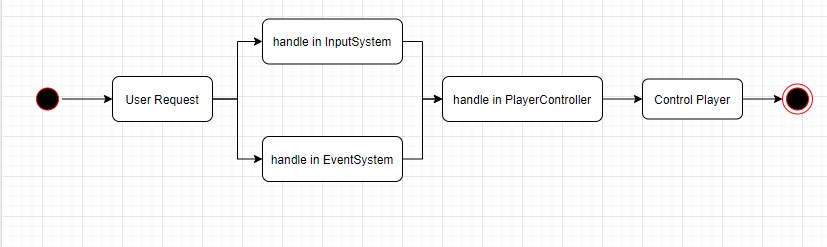
hình : Lượt đồ Sequence Hệ thống quản lí

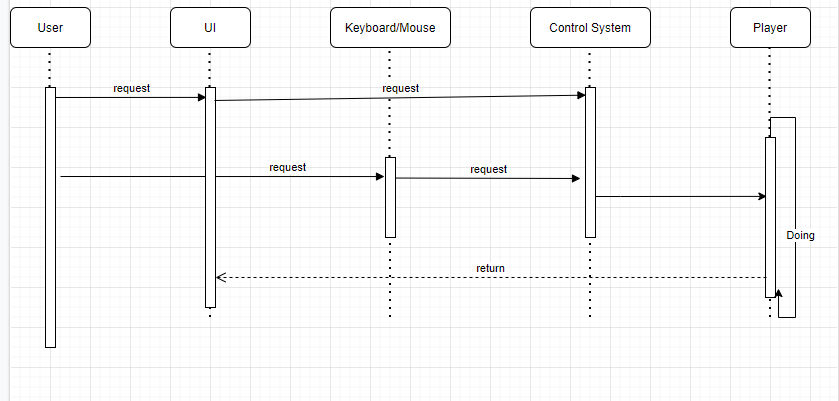
### 3.3.2 Xây dựng Hệ thống điều khiển

Chức năng: Nhận yêu cầu từ người dùng tương tác đến nhân vật trong game bao gồm :

* Keyboard và Mouse cho nền tảng PC, gamepad cho hệ tay cầm.
* UI Button cho nền tảng Andoid.

Lượt đồ Activity:

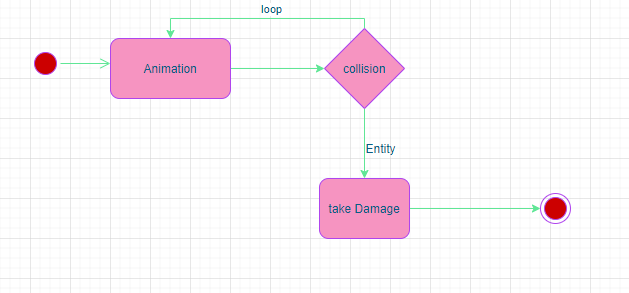
Hình : Lượt đồ Activity hệ thống điều khiển

Lượt đồ sequence: 

Hình: Lượt đồ sequence hệ thống điều khiển

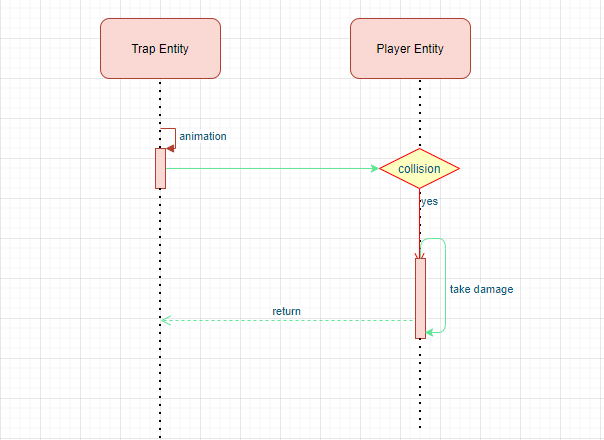
### 3.3.3 Xây dựng Hệ thống Bẫy

Chức năng: phối hợp với địa hình map tạo thành các chướng ngại vật tạo độ khó cho Player.

Lượt đồ Activity:

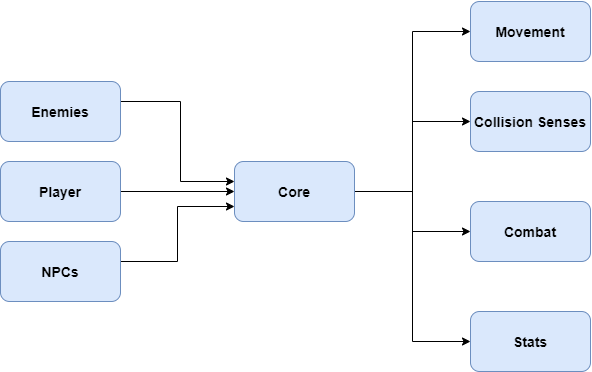
Hình : Lượt đồ Activity hệ thống Bẫy

Lượt đồ Sequence:

Hình : Lượt đồ Sequence hệ thống bẫy

### 3.3.4 Xây dựng Core và State Machine áp dụng Computer Science

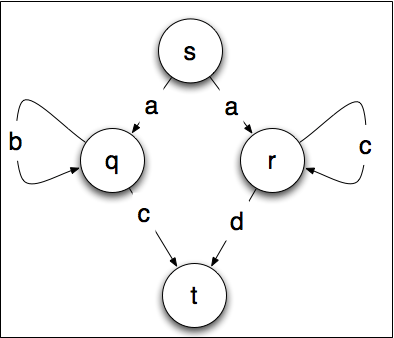
#### **3.3.4.1 Core**

* 

**Hình 7: Core**

* Hệ thống di chuyển: đi, đứng.
* Hệ thống tương tác: Tương tác với vật phẩm và đối tượng khác.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm đất, chạm tường, chạm trần.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

#### **3.3.4.2 State Machine áp dụng Computer Science**



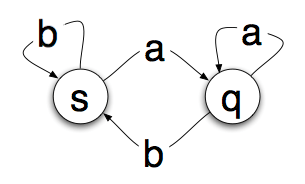
Hình : State Machine

##### **3.3.4.2.1 Một số khái niệm về Computer Sciences**

* Khoa học máy tính cho phép chúng ta lập trình, nhưng có thể làm rất nhiều việc lập trình mà không hiểu các khái niệm khoa học máy tính cơ bản.
* Điều này không phải lúc nào cũng tệ. Khi lập trình, ta làm việc ở mức độ trừu tượng cao hơn.
* Ví dụ khi lái xe, ta chỉ quan tâm đến hai hoặc ba bàn đạp, một chiếc gearshift, và một cái bánh lái. Bạn có thể vận hành một cách an toàn ô tô mà không có ý tưởng rõ ràng nào về cách hoạt động của nó.
* Tuy nhiên, nếu bạn muốn vận hành một chiếc xe ở các giới hạn của khả năng của nó, bạn cần biết nhiều về ô tô hơn là chỉ ba bàn đạp, gearshift và bánh lái lái.
* Điều đó cũng đúng về lập trình. Nhiều công việc hàng ngày có thể được hoàn thành với ít hoặc không hiểu về **Computer Sciences**. Bạn không cần hiểu lý thuyết tính toán để xây dựng một form liên lạc trên website.
* Tuy nhiên, nếu bạn định viết mã yêu cầu tính toán nghiêm trọng, bạn sẽ cần phải hiểu thêm về cách tính toán hoạt động.
* Ta sẽ xem xét logic đằng sau một trong những thiết bị tính toán trừu tượng đơn giản nhất là máy trạng thái hữu hạn: **finite state machine** và áp dụng vào xây dựng State Machine cho Player và AI State Machine cho Enemy.

##### **3.3.4.2.2 Tìm hiểu Finite State Machine: máy trạng thái hữu hạn**

* Một máy trạng thái hữu hạn là một sự trừu tượng toán học được dùng để thiết kế các thuật toán.
* Nói một cách đơn giản hơn, khi đọc một đầu vào, máy sẽ chuyển từ trạng thái này sang trạng thái khác. Mỗi trạng thái chỉ định trạng thái chuyển sang do một số đầu vào nhất định. Nghe có vẻ phức tạp nhưng rất đơn giản.
* Tưởng tượng một thiết bị đọc một mảnh giấy dài. Đối với mỗi inch giấy ở đó có một lá thư được in trên đó chữ' a' hoặc chữ' b'.
* Khi bộ máy trạng thái đọc mỗi chữ cái, nó thay đổi trạng thái. Đây là một cỗ máy trạng thái rất đơn giản:



Hình : State machine đọc chữ cái

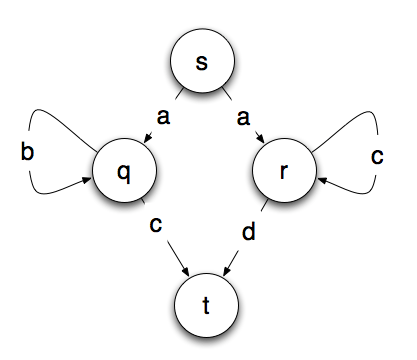
* Vòng tròn là một trạng thái của bộ máy.Mũi tên là chuyển vị .Vậy, nếu bạn đang ở trạng thái s và đọc chữ "a" bạn sẽ chuyển sang trạng thái q .Nếu bạn đọc chữ "b”, bạn sẽ ở lại trạng thái s .
* Vậy nếu chúng ta bắt đầu từ trạng thái s và đọc từ trái sang phải, chúng ta sẽ đọc “a” và chuyển đến trạng thái q.
* Sau đó, chúng ta sẽ đọc 'b' và chuyển về nhà nước s .
* Chữ "b" sẽ giữ chúng ta lại tại trạng thái s hoặc chữ “a” đưa chúng ta trở lại trạng thái q. Đơn giản thôi, nhưng vấn đề là gì?
* Hóa ra bạn có thể chạy một cuộn giấy qua máy trạng thái và nói điều gì đó về chuỗi chữ cái, bằng cách kiểm tra trạng thái mà bạn kết thúc trên.
* Trong bộ máy trạng thái đơn giản trên, nếu chúng ta kết thúc tại trạng thái s thì cuộn giấy phải kết thúc bằng chữ' b'. Nếu chúng ta kết thúc trong trạng thái q cuộn giấy sẽ kết thúc bằng chữ 'a'.
* Nghe có vẻ vô nghĩa, nhưng những vấn đề rất tệ có thể được giải quyết bằng cách tiếp cận này. Một ví dụ rất đơn giản là xác định xem một trang html chứa các thẻ này trong thứ tự này hay không: <html> <head> </head> <body> </body> </html>
* Máy trạng thái có thể di chuyển đến trạng thái cho thấy nó đã đọc thẻ html, vòng lặp cho đến khi nó đọc thẻ đóng html, trong vòng lặp đó nó sẽ thẻ head tương tự và sẽ lặp cho đến khi đọc được thẻ đóng head tương ứng, vân vân.
* Nếu nó tạo thành công cho trạng thái cuối cùng, thì bạn có những thẻ cụ thể đó trong đúng thứ tự.
* Các máy trạng thái hữu hạn cũng có thể được sử dụng để biểu diễn nhiều hệ thống khác như cơ chế của một máy đo đỗ xe, máy móc pop, máy bơm khí tự động, và tất cả các loại khác.

##### **3.3.4.2.3 Deterministic Finite State Machines: máy trạng thái hữu hạn xác định.**

* Những máy trạng trên là máy trạng thái hữu hạn xác định. Từ bất kỳ trạng thái nào, chỉ có một chuyển đổi cho bất kỳ đầu vào nhập nào. Nói cách khác, không thể có hai đường dẫn ra khỏi trạng thái khi bạn đọc chữ 'a'. Đầu tiên, điều này nghe thật ngớ ngẩn khi tạo ra sự phân biệt này.
* Một tập các quyết định như thế nào nếu cùng đầu vào có thể dẫn đến việc di chuyển đến nhiều trạng thái hơn? Bạn không thể nói với máy tính, nếu x bằng true rồi thực hiện việc A hoặc thực hiện việc B được.
* Ta có thể làm điều đó với máy trạng thái.
* Nếu đầu ra của máy trạng thái là một trạng thái. Nó sẽ đi qua tất cả các quá trình, sau đó là trạng thái cuối cùng được đọc, và sau đó một hành động được thực hiện. Một máy trạng thái thì không làm đi bất cứ thứ gì khi di chuyển từ trạng thái này sang trạng thái khác.
* Nó xử lý, và khi đến cuối, trạng thái được đọc và một thứ gì đó bên ngoài kích hoạt hành động mong muốn (ví dụ, pha chế một lon nước ngọt). Đây là một khái niệm quan trọng khi nói đến máy trạng thái hữu hạn xác định.

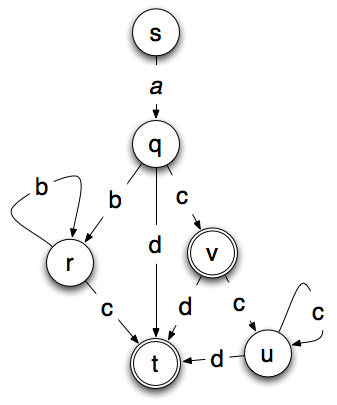
##### **3.3.4.2.3 None-Deterministic Finite State Machines: máy trạng thái hữu hạn không xác định.**

* Máy trạng thái hữu hạn không xác định là máy trạng thái hữu hạn trong đó đầu vào nhất định từ một trạng thái cụ thể có thể dẫn đến nhiều hơn một trạng thái khác nhau.
* Ví dụ: giả sử chúng ta muốn xây dựng một máy trạng thái hữu hạn có thể nhận dạng các chuỗi ký tự:
* Bắt đầu bằng chữ cái ‘a’
* Và sau đó là không hoặc nhiều lần xuất hiện của chữ ‘b’
* Hoặc, không hoặc nhiều lần xuất hiện của chữ cái ‘c’
* Được kết thúc bằng chữ cái tiếp theo của bảng chữ cái.
* Các chuỗi hợp lệ sẽ là:
* abbbbbbbbbc
* abbbc
* acccd
* acccccd
* ac (không có lần xuất hiện nào của b)
* ad (không có lần xuất hiện nào của c)
* Vì vậy, nó sẽ nhận ra chữ cái ‘a’ theo sau là không hoặc nhiều hơn cùng một chữ cái ‘b’ hoặc ‘c’, tiếp theo là chữ cái tiếp theo của bảng chữ cái.
* Một cách rất đơn giản để biểu diễn điều này là với một máy trạng thái trông giống như bên dưới, trong đó trạng thái cuối cùng của t có nghĩa là chuỗi đã được chấp nhận và khớp với mẫu.



Hình : Non-Deterministic Finite State Machines

* Bạn có thấy vấn đề? Từ điểm xuất phát s, chúng ta không biết phải đi con đường nào. Nếu chúng ta đọc ký tự ‘a’, chúng ta không biết nên chuyển sang trạng thái q hay r.
* Có một số cách để giải quyết vấn đề này. Một là bằng cách bẻ khóa lại. Bạn chỉ đơn giản là đi tất cả các con đường có thể, và bỏ qua hoặc lùi lại những con đường mà bạn gặp khó khăn.
* Về cơ bản đây là cách hầu hết các máy tính chơi cờ vua hoạt động. Họ xem xét tất cả các khả năng - và tất cả các khả năng của những khả năng đó - và chọn con đường mang lại cho họ nhiều lợi thế nhất so với đối thủ.
* Tùy chọn khác là chuyển đổi máy không xác định thành máy xác định.
* Một trong những thuộc tính thú vị của máy không xác định là tồn tại một thuật toán để biến bất kỳ máy không xác định nào thành máy xác định. Tuy nhiên, nó thường phức tạp hơn nhiều.
* May mắn thay cho chúng ta, ví dụ trên chỉ phức tạp hơn một chút. Trên thực tế, cái này đủ đơn giản để chúng ta có thể biến nó thành một cỗ máy xác định trong đầu mà không cần đến sự hỗ trợ của một thuật toán chính thức.
* Máy bên dưới là phiên bản xác định của máy không xác định ở trên. Trong máy bên dưới, trạng thái cuối cùng của t hoặc v đạt được bởi bất kỳ chuỗi nào được máy chấp nhận.



Hình : Deterministic Finite State Machines after change

* Mô hình không xác định có bốn trạng thái và sáu quá trình chuyển đổi. Mô hình xác định có sáu trạng thái, mười trạng thái chuyển tiếp và hai trạng thái cuối cùng có thể có.
* Đó có thể không nhiều hơn nữa, nhưng độ phức tạp thường tăng lên theo cấp số nhân. Một máy không xác định có kích thước vừa phải có thể tạo ra một máy xác định hoàn toàn khổng lồ.

##### **3.3.4.2.4 Kết luận và hướng áp dụng vào xây dựng State Machine trong đồ án**

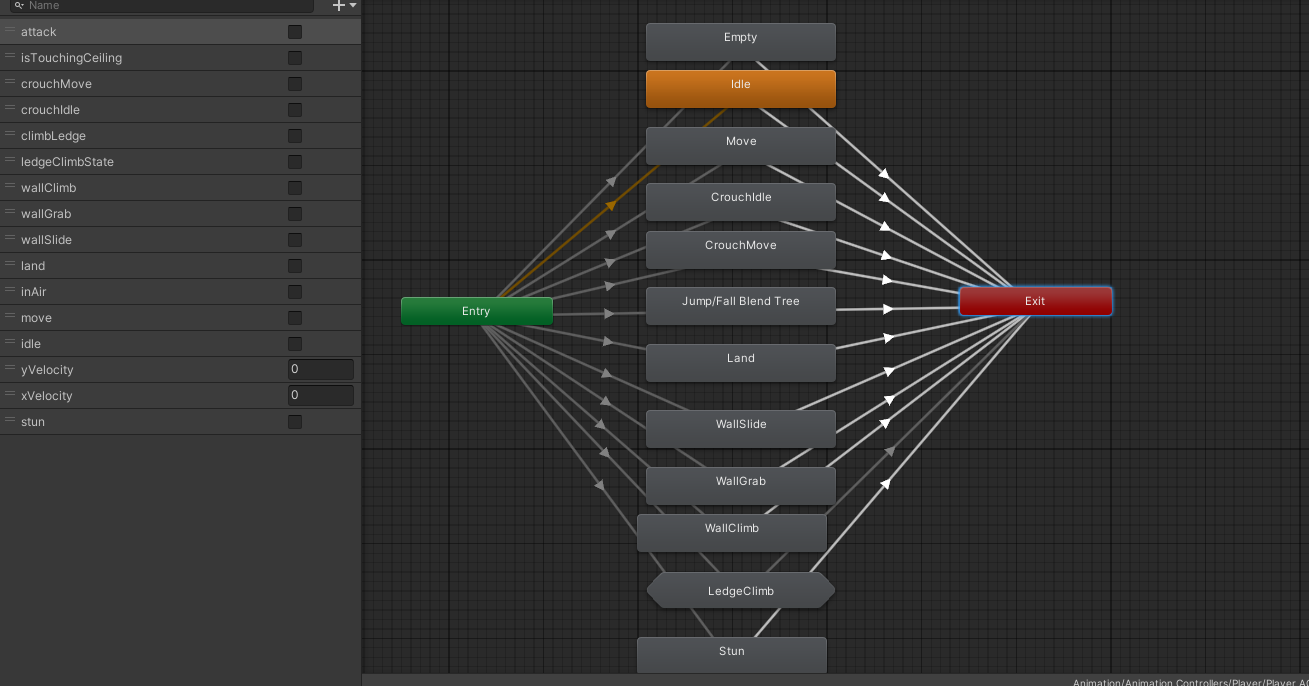
* Kết Luận:
* Bất kể những hạn chế của chúng, máy trạng thái là một khái niệm rất quan trọng đối với tính toán. Đặc biệt, điều quan trọng là đối với bất kỳ máy trạng thái không xác định nào mà bạn có thể thiết kế, tồn tại một máy trạng thái xác định thực hiện điều tương tự.
* Đây là một điểm mấu chốt, bởi vì nó có nghĩa là bạn có thể thiết kế thuật toán của mình theo bất kỳ cách nào dễ nghĩ nhất. Khi bạn có một thuật toán hoạt động, bạn có thể chuyển đổi nó thành bất kỳ dạng nào hiệu quả nhất.
* Việc hiểu rằng các máy trạng thái hữu hạn và các biểu thức chính quy là tương đương về mặt chức năng sẽ mở ra một số cách sử dụng cực kỳ thú vị cho các công cụ biểu thức chính quy - đặc biệt khi nói đến việc tạo các quy tắc nghiệp vụ có thể thay đổi mà không cần biên dịch lại hệ thống.
* Nền tảng về Khoa học Máy tính cho phép bạn giải một vấn đề mà bạn không biết cách giải và lập luận: “Tôi không biết cách giải X, nhưng tôi biết cách giải Y. Và tôi biết cách chuyển đổi một giải pháp. cho Y thành một giải pháp cho X. Vì vậy, bây giờ tôi biết làm thế nào để giải quyết X. ”
* Hướng áp dụng vào đề tài:

Từ những định nghĩa và ví dụ trên, ta sẽ xây dựng State Machine cho Player và AI State Machine cho các Enemy: Deterministic Finite State Machines

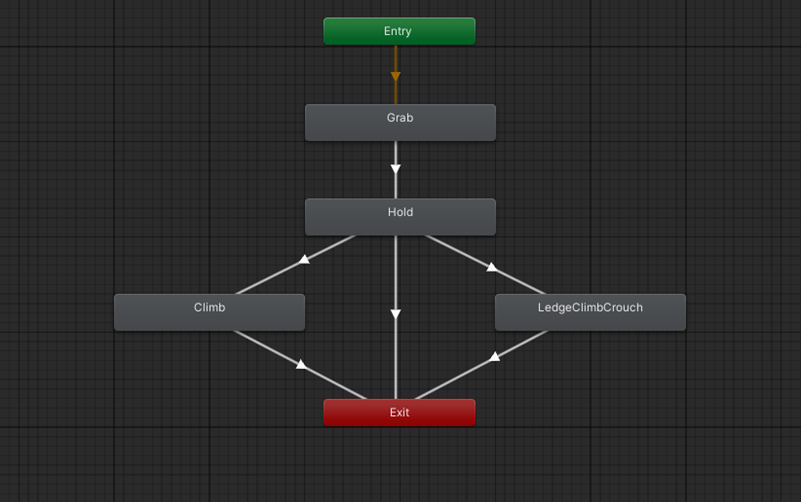
* Xây dựng các trạng thái cho Player và Enemy.
* Xác định trạng thái ban đầu.
* Xác định điều kiện để chuyển đổi trạng thái: Input từ người dùng và tương tác của môi trường flatform trong game sẽ quyết định trạng thái của Player và Enemy.
* Nếu không thông qua một điều kiện bất kì để chuyển đổi trạng thái thì Enemy sẽ trở về trạng di chuyển ban đầu, tạo thành một hệ thống State Machine khép kín mà ta gọi là AI State Machine.

### 3.3.5 Xây dựng hệ thống Player dựa trên Core và State Machine

* Các Trạng thái của Player :
* AbilityState bao gồm :
* Attack State: trạng thái tấn công.
* Dash State: trạng thái lướt.
* Jump State: trạng thái nhảy thường.
* Wall Jump State: trạng thái nhảy tường.
* DodgeRoll: trạng thái cuộn tròn.
* In Air State: trạng thái trên không trung không chạm đất, tường.
* Touching Wall State: trạng thái khi chạm tường bao gồm :
* Wall Grab State: trạng thái bám tường.
* Wall Slide State: trạng thái trượt xuống tường.
* Wall Climb State: trạng thái trèo tường.
* Grounded State : trạng thái khi chạm đất bao gồm :
* Land State: trạng thái chạm đất khi vận tốc y bằng 0 khi đang trên không trung.
* Idle State: trạng thái đứng yên.
* Move State: trạng thái di chuyển.
* Crouch Move State: trạng thái di chuyển khom lưng.
* Crouch Idle State: trạng thái khom lưng.
* Ledge Climb: trạng thái trèo lên mỏm đá.
* Animation của Player :

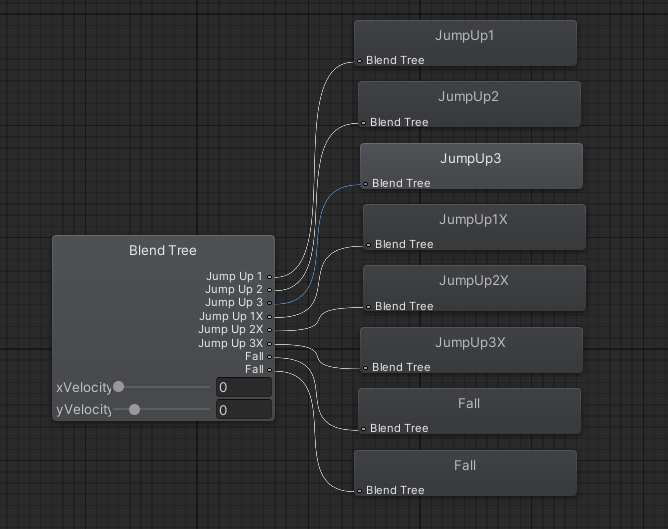
**Hình 8: Animation Player**

* Ledge Climb State: Trạng thái leo mỏm đá, đu mỏm đá.



**Hình 9: Ledge Climb State**

* BlendTree Ledge Climb State : Animation cho sự kiện nhảy lên và rơi xuống khi không tác động vector lực (vector x =0) và khi tác động lên vật thể 1 vector lực(vector x khác 0)



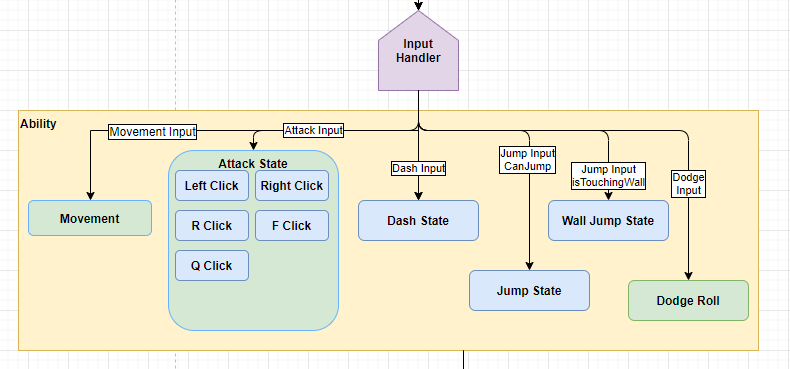
**Hình 10A: Blend Tree Player**

* Pramester BlendTree Player: các giá trị vector lực x, y để xác định animation của Player.

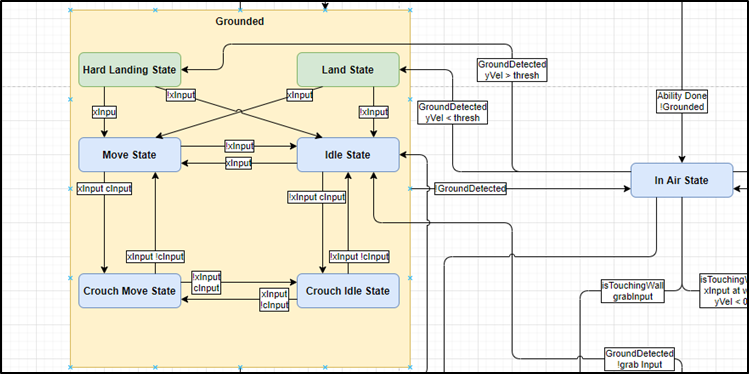


**Hình 10B: Pramester BlendTree Player**

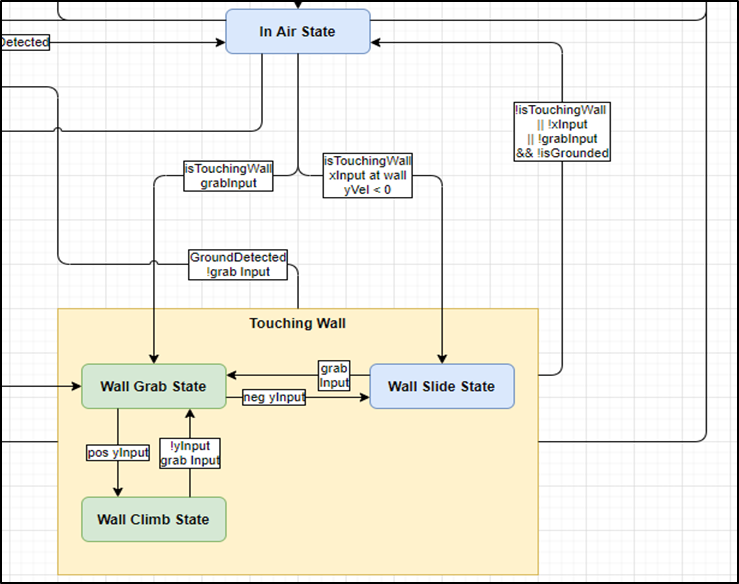
* Hệ thống Input chuyển đổi các trạng thái của AbilityState: di chuyển, tấn công, lướt, nhảy, nhảy tường.

 **Hình 11: Input Handler, AbilityState và Movement**

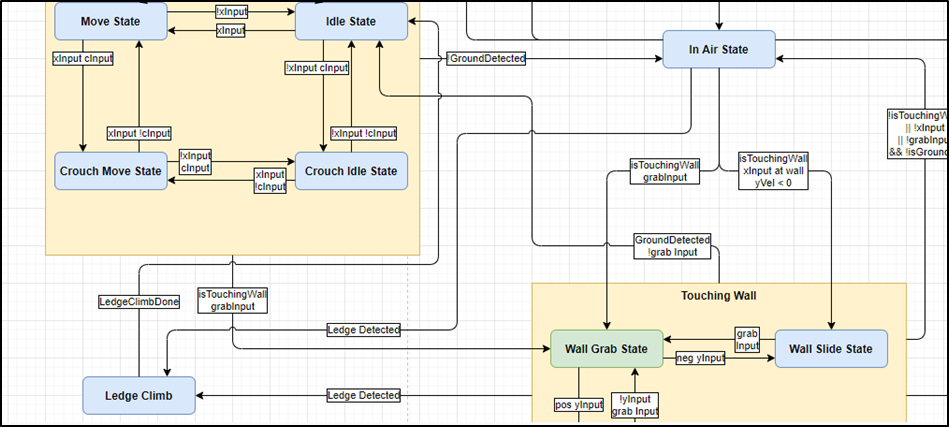
* Chuyển đổi các trạng thái Player :
* In Air State và Grounded State

**Hình 12: Chuyển đổi giữa In Air State và Grounded State**

* In Air State và Touching Wall State

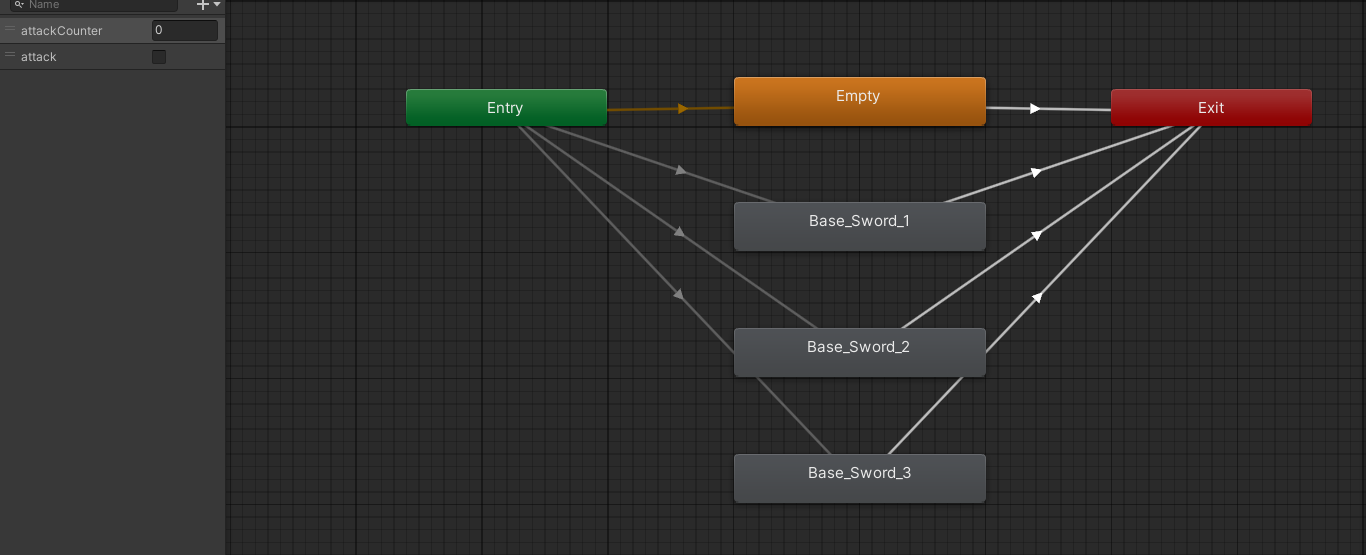


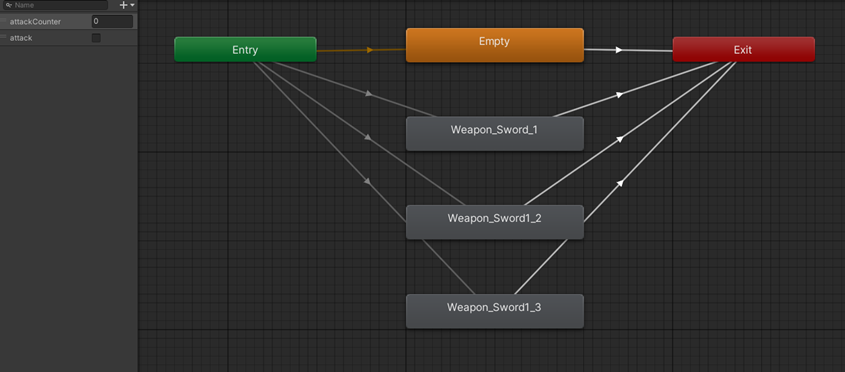
**Hình 13: In Air State sang Touching Wall State**

* Chuyển đổi giữa Ledge Climb State với các trạng thái khác:

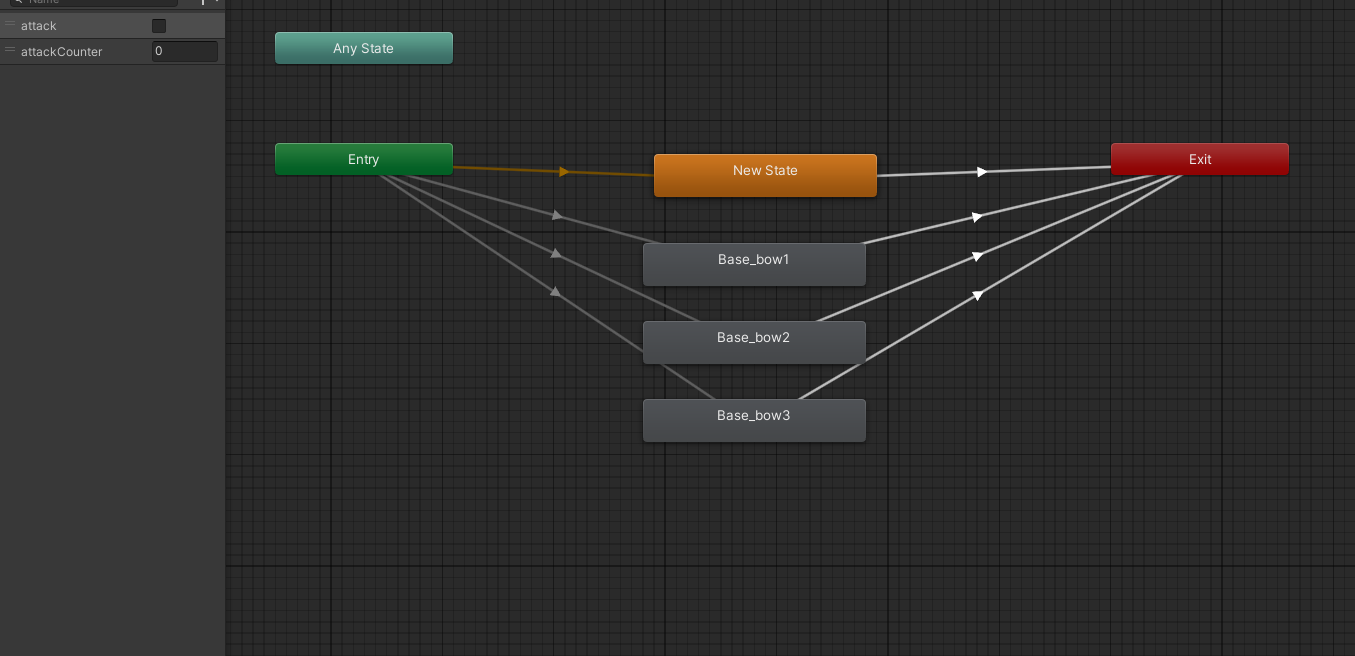
**Hình 14: Ledge Climb State**

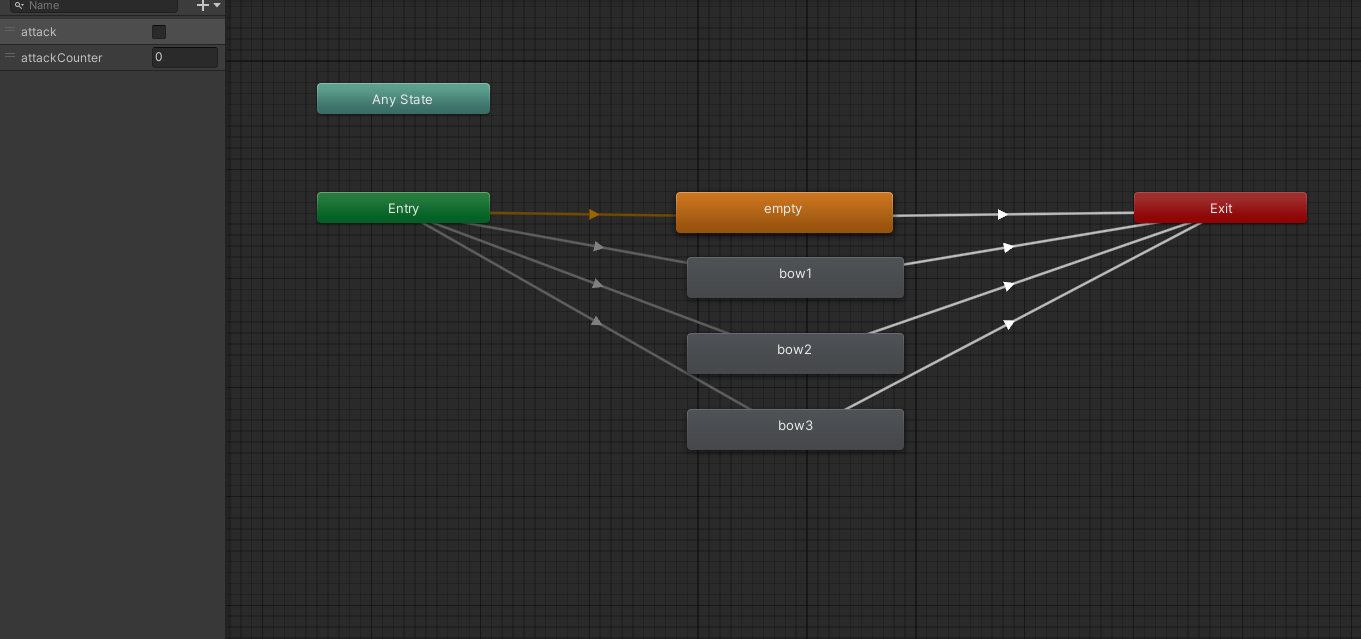
* Hệ thống tương tác: Chiến đấu với quái vật, kẻ thù, nhặt vật phẩm, sử dụng vật phẩm.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra Player chạm đất, chạm tường, chạm trần.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại, vàng, kinh nghiệm, cấp độ.
* Hệ thống vũ khí: kiếm, dao, khiên, cung tên.
* Kiếm: thiết lập animation đồng bộ giữa Player và kiếm.

**Hình 15: Animation thực thể player dùng kiếm**

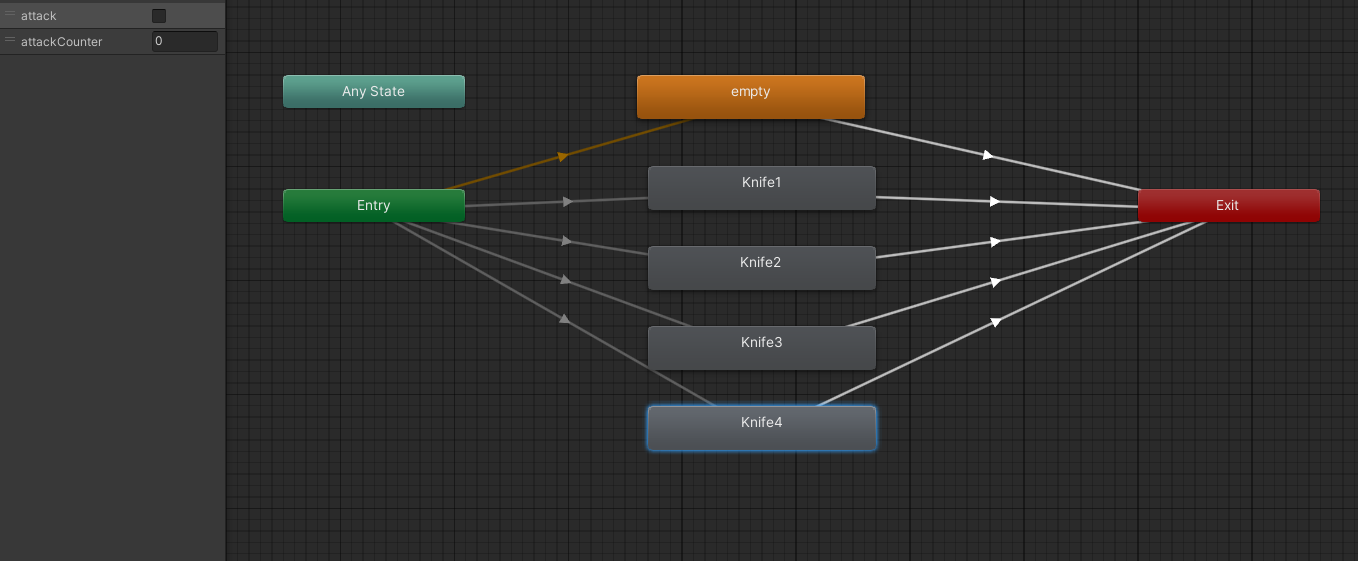
**Hình 16: Animation Weapon kiếm**

* Cung tên : thiết lập animation đồng bộ giữa Player và cung tên

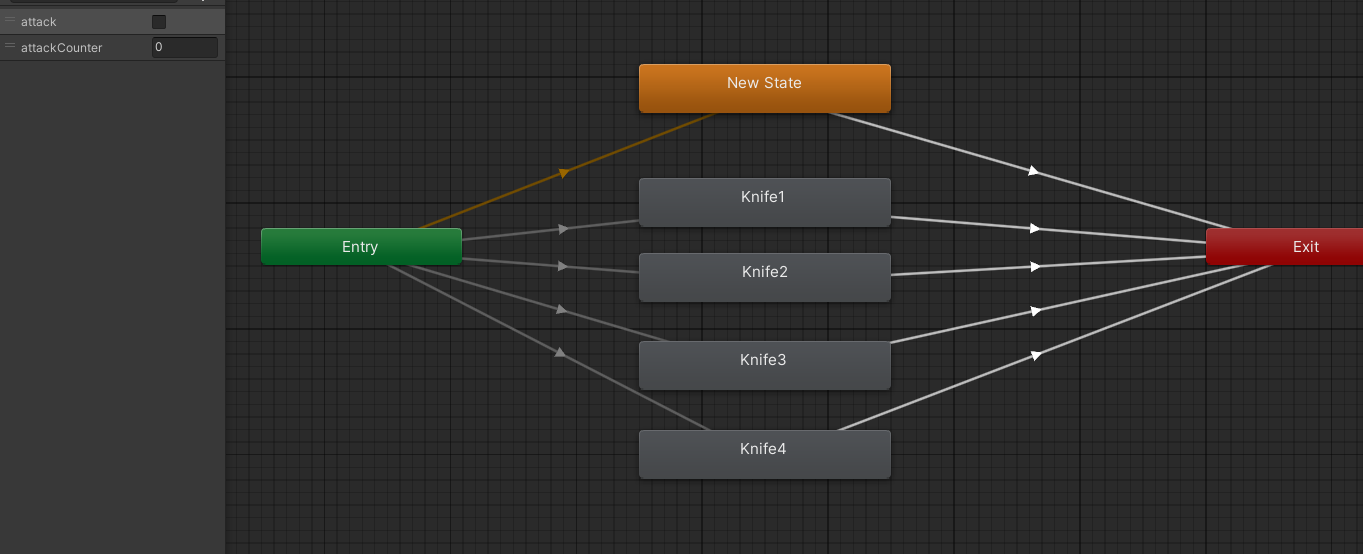
**Hình17: Animation thực thể Player dùng cung**



**Hình 18: Animation cung tên**

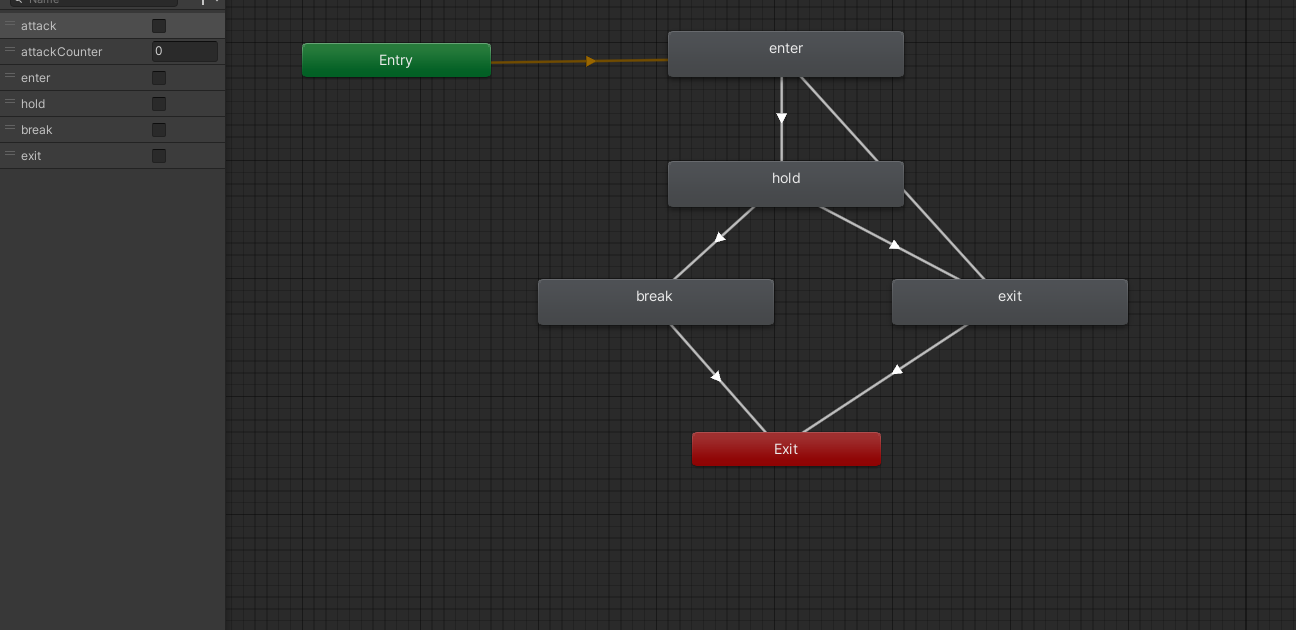
* Dao : thiết lập animation đồng bộ giữa Player và dao

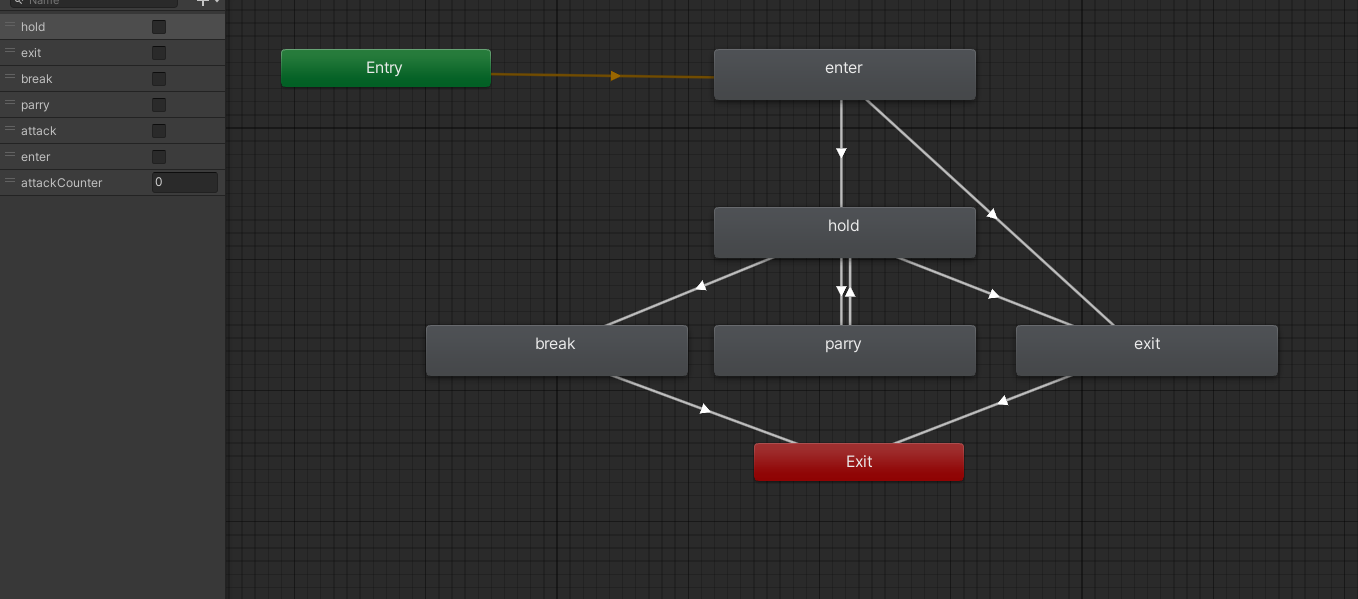
**Hình 19: Animation thực thể Player dùng dao**

****

**Hình 20: Animation dao**

* khiên : thiết lập animation đồng bộ giữa Player và khiên

**Hình 21: Animation thực thể Player dùng khiên**

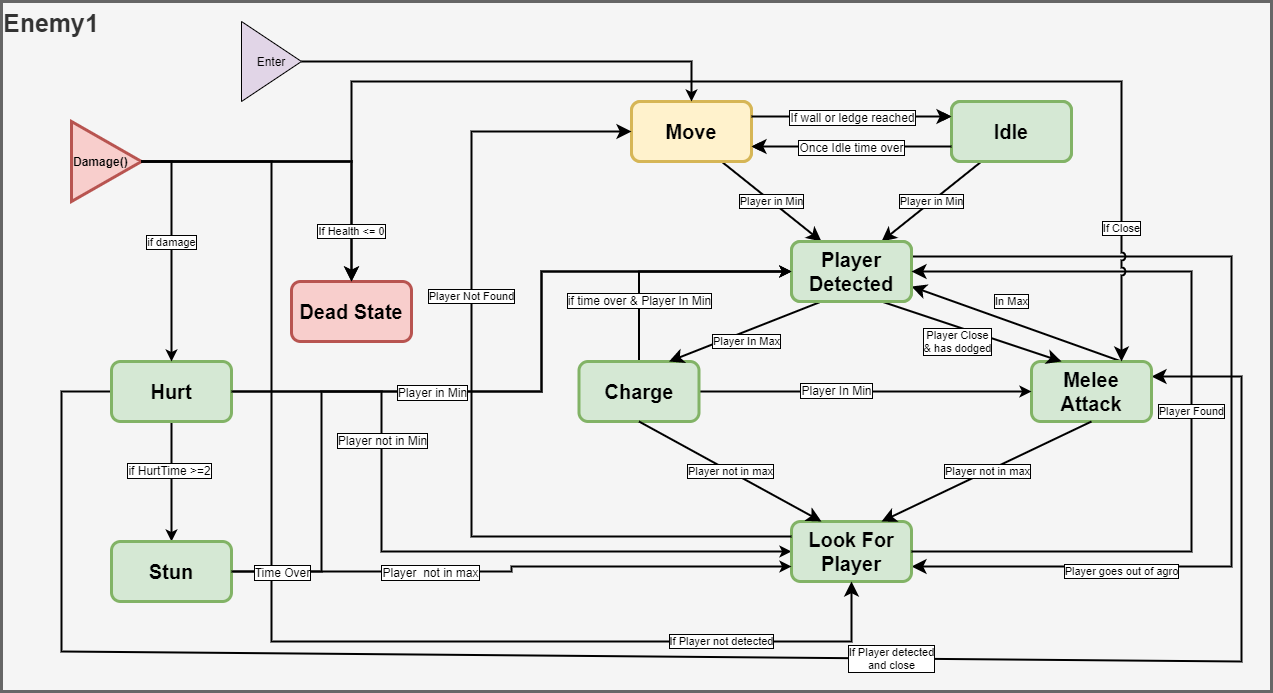
****

**Hình 22: Animation khiên**

### 3.3.6 Xây dựng hệ thống Enemy dựa trên Core và AI State Machine

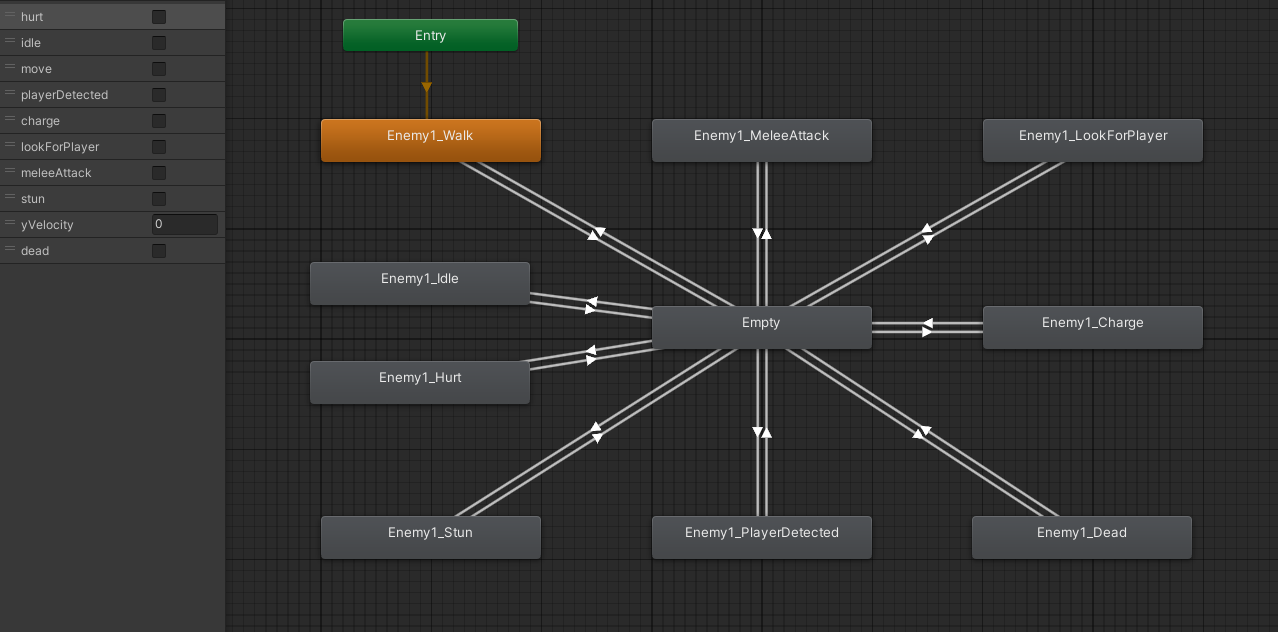
* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

#### **3.3.6.1 Enemy1\_Pork**



**Hình 23: Sơ đồ hoạt động Enemy 1 Pork**

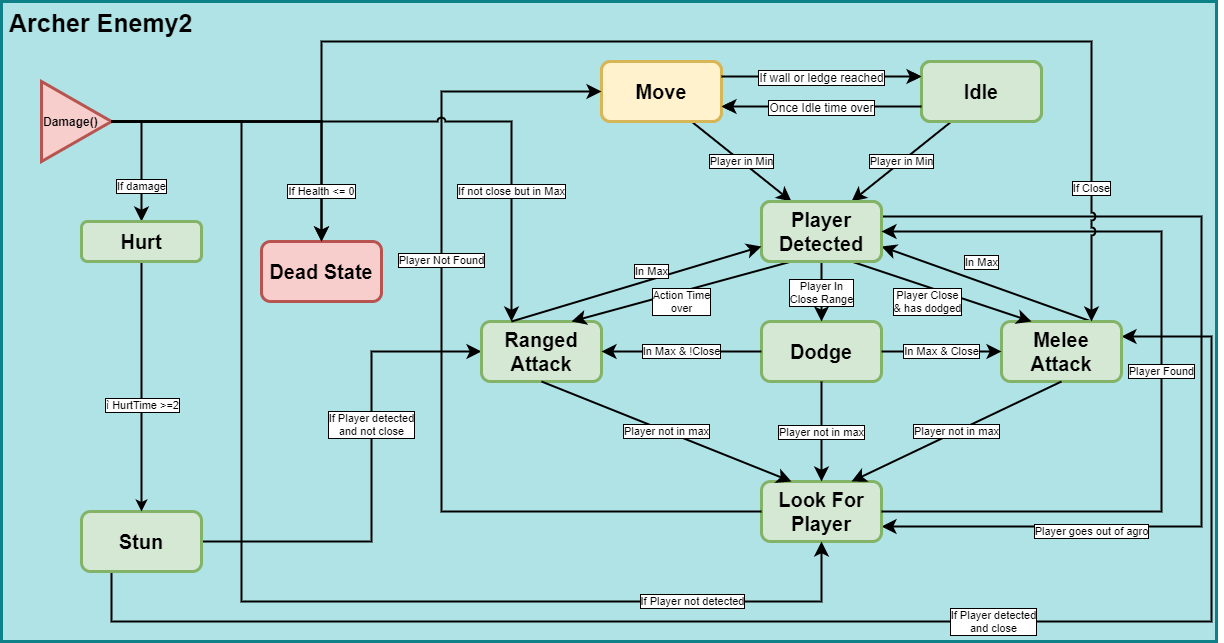
* Các Trạng thái của Enemy1\_Pork:
* Idle State: trạng thái đứng.
* Moving State: trạng thái mặc định di chuyển.
* Player Detected State: trạng thái xác định Player bước vào tầm xử lí.
* Charge Player State: trạng thái chuyển, tăng tốc tiến về phía Player.
* Attack Player State: trạng thái tấn công Player.
* Look For Player State: trạng thái tìm Player khi Player ra khỏi tầm xử lí.
* Stun State: Trạng thái choáng khi máu hiện tại dưới 20%.
* Dead State: Trạng thái chết.

**Hình 24 : Animation Enemy 1 Pork**

* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

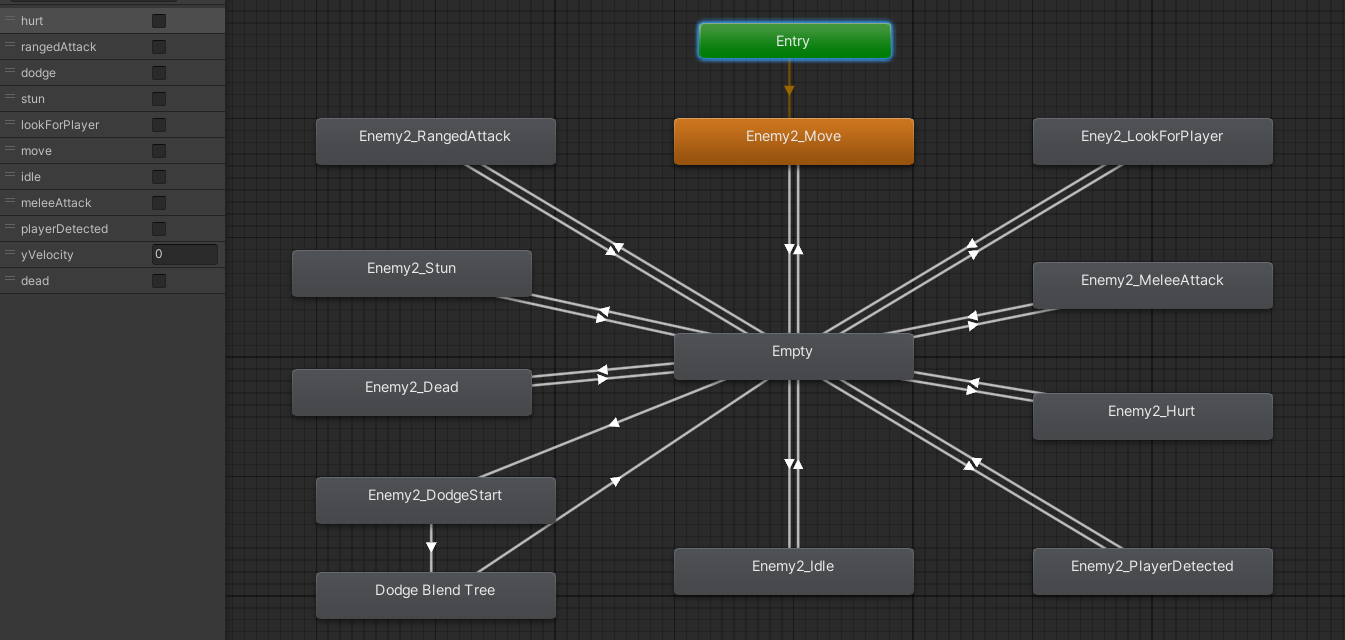
#### **3.3.6.2 Enemy2\_Archers**

* Các Trạng thái của Enemy2\_Archers:



**Hình 25: Sơ đồ hoạt động Enemy 2 Archers**

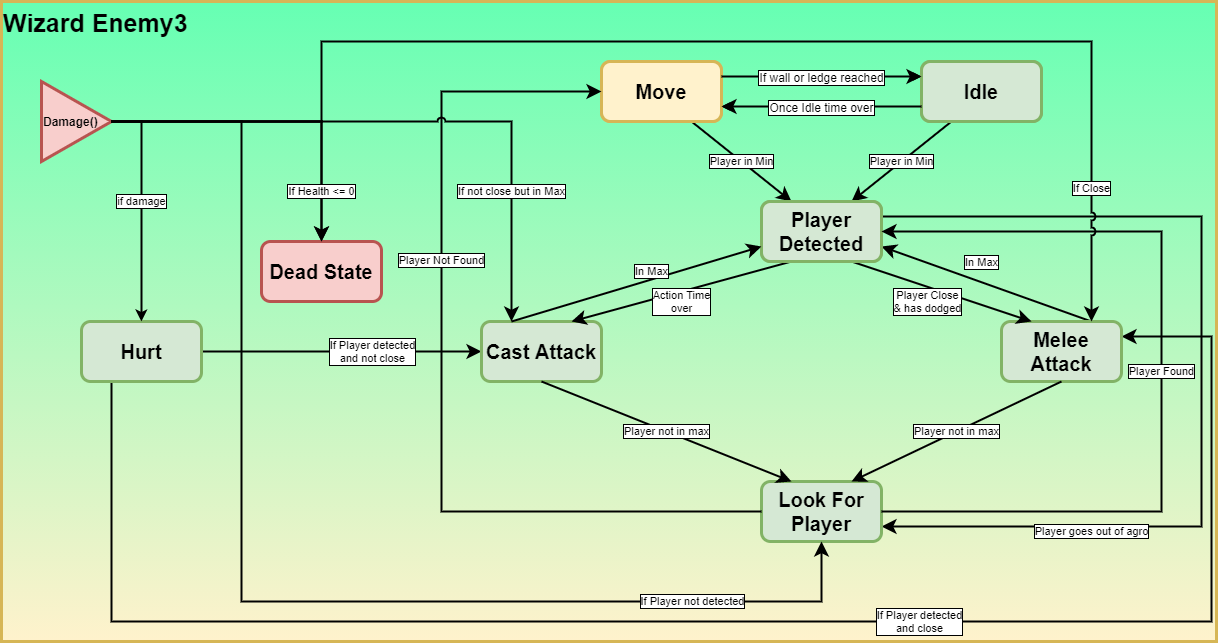
* Idle State: trạng thái đứng.
* Moving State: trạng thái mặc định di chuyển.
* Player Detected State: trạng thái xác định Player bước vào tầm xử lí.
* Dodge State: trạng thái nhảy lùi né tránh sự áp sát của Player.
* Melee Attack Player State: trạng thái tấn công cận chiến khi Player ở trong tầm đánh gần.
* Range Attack Player State: trạng thái bắn cung khi Player ở trong tầm đánh xa.
* Look For Player State: trạng thái tìm Player khi Player ra khỏi tầm xử lí.
* Stun State: Trạng thái choáng khi máu hiện tại dưới 20%.
* Hurt State: Trạng thái tổn thương khi bị đánh.
* Dead State: Trạng thái chết.

**Hình 26: Animation Enemy 2 Archers**

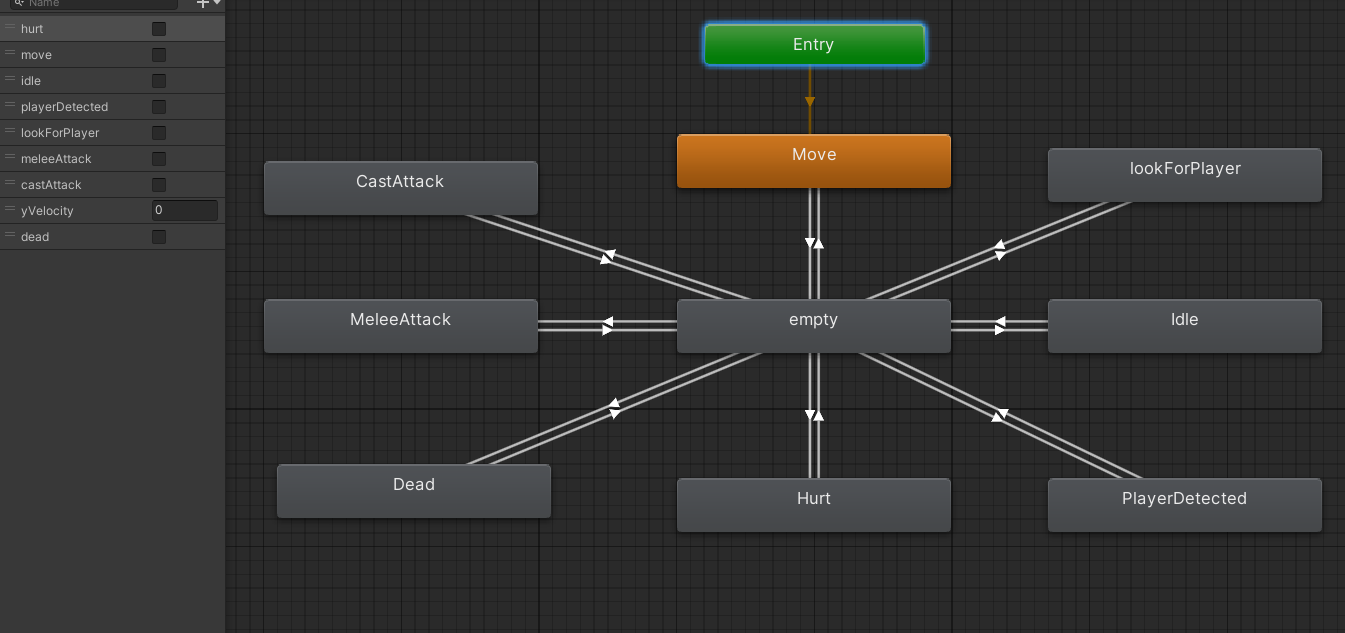
* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

#### **3.3.6.3 Enemy3\_Wizard**

**Hình 27: Sơ đồ hoạt động Enemy 3 Wizard**



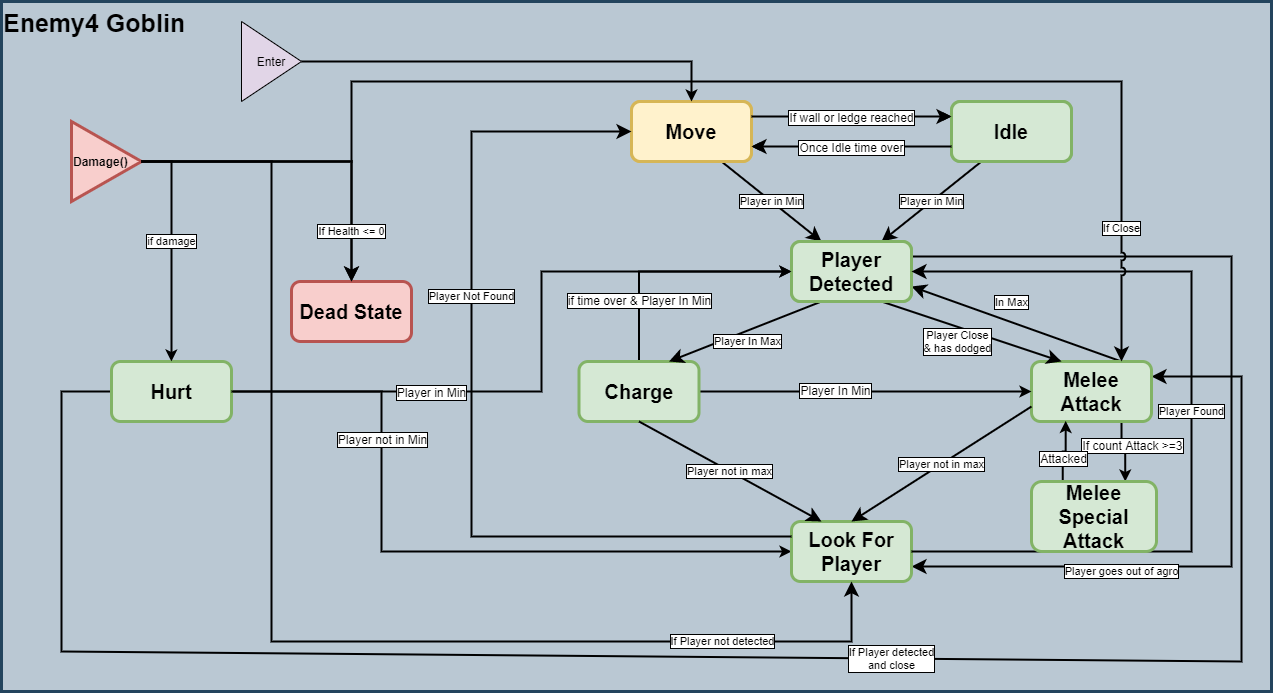
* Các Trạng thái của Enemy3\_Wizard:
* Idle State: trạng thái đứng.
* Moving State: trạng thái mặc định di chuyển.
* Player Detected State: trạng thái xác định Player bước vào tầm xử lí.
* Melee Attack Player State: trạng thái tấn công cận chiến khi Player ở trong tầm đánh gần.
* Cast Attack Player State: trạng thái niệm phép khi Player ở trong tầm đánh xa.
* Look For Player State: trạng thái tìm Player khi Player ra khỏi tầm xử lí.
* Hurt State: Trạng thái tổn thương khi bị đánh.
* Dead State: Trạng thái chết.

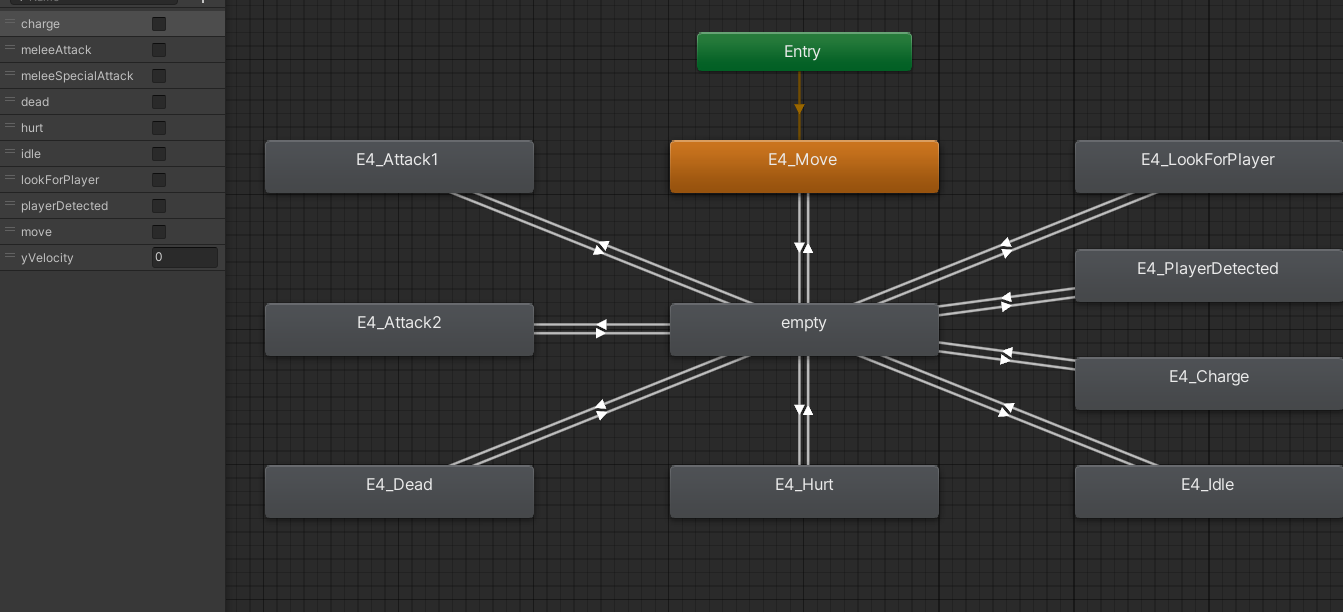
**Hình 28: Animation Enemy 3 Wizard**

* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

#### **3.3.6.4 Enemy4\_Goblin**

**Hình 29: Sơ đồ hoạt động Enemy 4 Goblin**

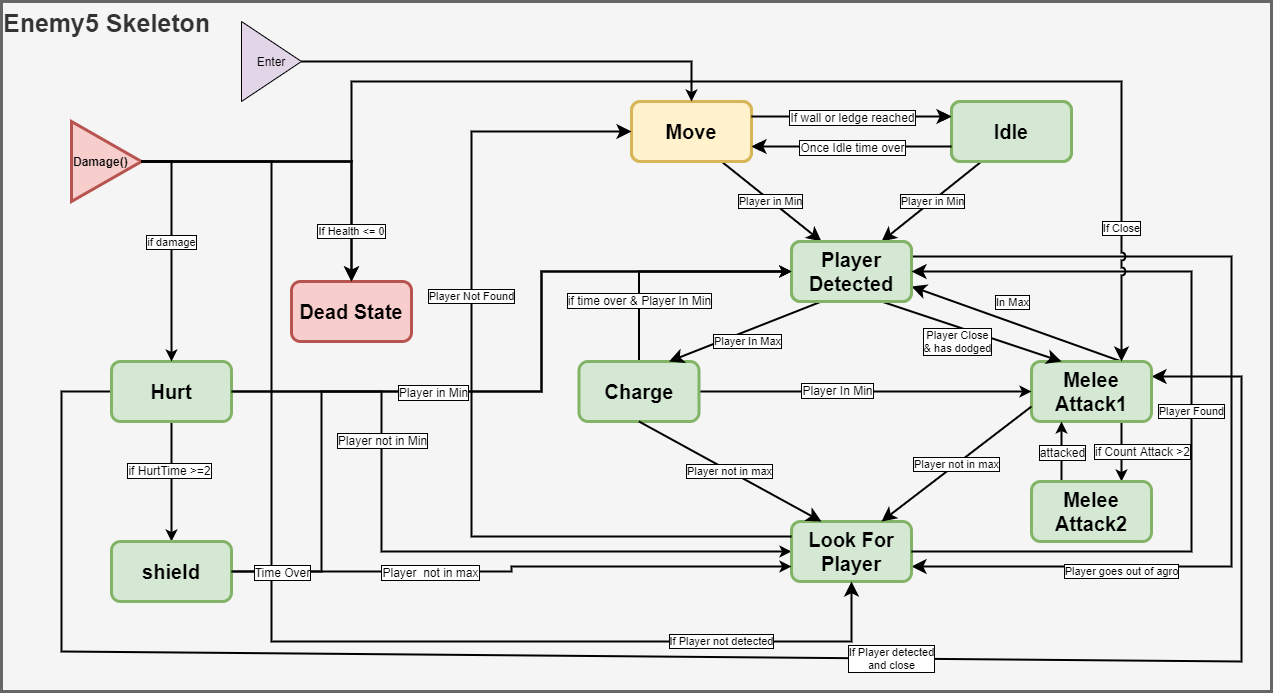


* Các Trạng thái của Enemy4\_Goblin:
* Idle State: trạng thái đứng.
* Moving State: trạng thái mặc định di chuyển.
* Player Detected State: trạng thái xác định Player bước vào tầm xử lí.
* Melee Attack State: trạng thái tấn công cận chiến khi Player ở trong tầm đánh gần.
* Melee Special Attack State: trạng thái tấn công cận chiến khi đã đánh thường hai lần.
* Charge Player State: trạng thái chuyển, tăng tốc tiến về phía Player.
* Look For Player State: trạng thái tìm Player khi Player ra khỏi tầm xử lí.
* Hurt State: Trạng thái tổn thương khi bị đánh.
* Dead State: Trạng thái chết.

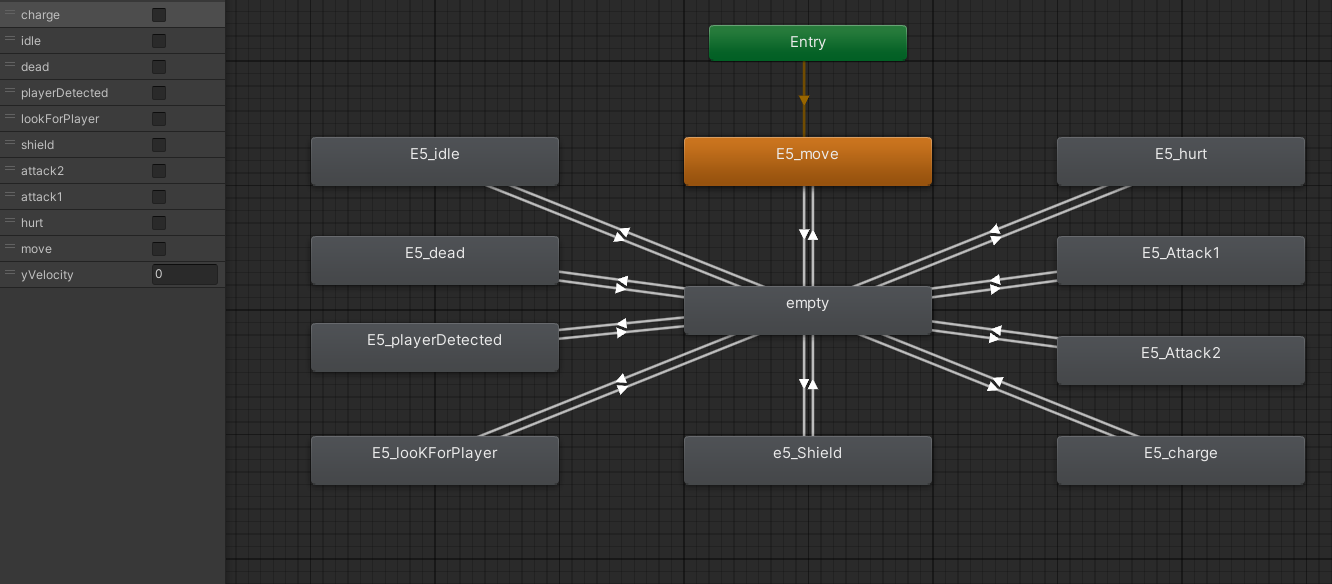
**Hình 30: Animation Enemy 4 Goblin**

* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

#### **3.3.6.5 Enemy5\_Skeleton**



**Hình 31: Sơ đồ hoạt động Enemy 5 Skeleton**

* Các Trạng thái của Enemy5\_Skeleton:
* Idle State: trạng thái đứng.
* Moving State: trạng thái mặc định di chuyển.
* Player Detected State: trạng thái xác định Player bước vào tầm xử lí.
* Melee Attack 1 Player State: trạng thái tấn công cận chiến khi Player ở trong tầm đánh gần.
* Melee Attack 2 State: trạng thái tấn công cận chiến khi đã đánh thường hai lần.
* Charge Player State: trạng thái chuyển, tăng tốc tiến về phía Player.
* Look For Player State: trạng thái tìm Player khi Player ra khỏi tầm xử lí.
* Hurt State: Trạng thái tổn thương khi bị đánh.
* Shield State: Trạng thái mở khiên chắn khi bị tổn thương hai lần.
* Dead State: Trạng thái chết.

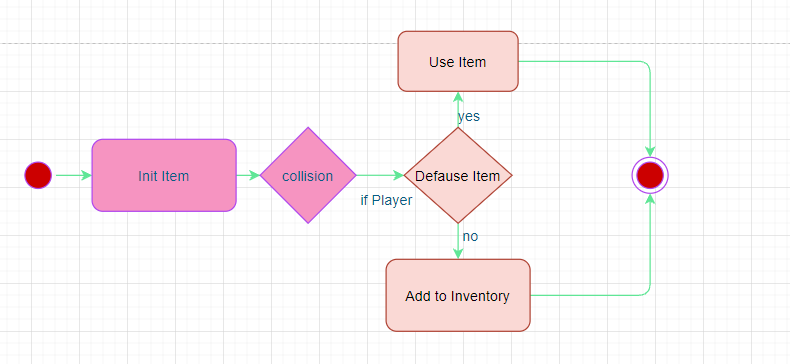
**Hình 32: Animation Enemy 5 Skeleton**

* Hệ thống di chuyển: đi, chạy.
* Hệ thống tương tác: Chiến đấu với Player.
* Hệ thống kiểm tra tương quan môi trường: Kiểm tra chạm tường, vượt ra khỏi địa hình, kiểm tra Player ở trong tầm xử lí.
* Hệ thống trạng thái: Máu tối đa, máu hiện tại.

### 3.3.7 Xây dựng Hệ thống Item và Inventory

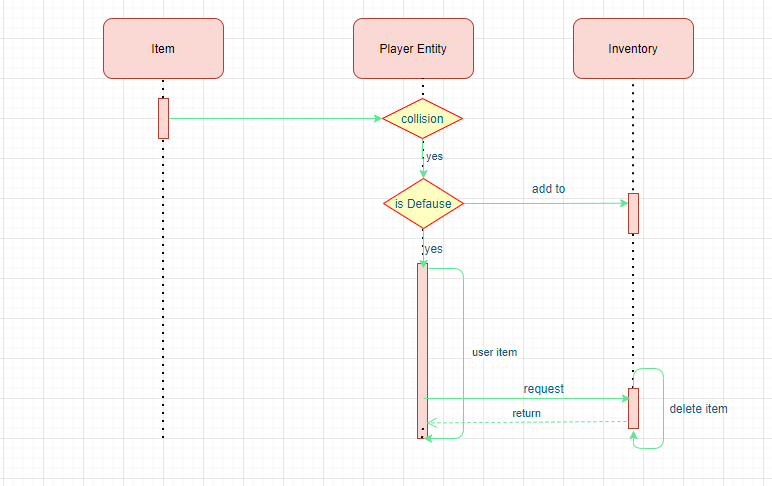
Chức năng: Item và bonus item sẽ giúp đỡ cho Player cho Game giảm bớt độ khó thông qua sử dụng vật phẩm giúp tăng máu, đồng thời tạo tiền đề cho xây dựng Inventory và xây dựng Shop trong tương lai để bán vật phẩm bổ trợ, them nhiều tính năng mới như tăng tốc, bất tử hoặc vật phẩm gây sát thương…

Lượt đồ Activity Item:



Hình : Lượt đồ Activity Item

Lượt đồ Sequence Item và Inventory:



Hình : Lượt đồ Sequence Item và Inventory

## 3.6 Thiết lập mô hình thực thể

### 3.6.1 Xác định các thực thể

### 3.6.2 Xác định mối quan hệ giữa các thực thể

### 3.6.3 Sơ đồ ERD (Entity Relationship Diagram)

### 3.6.4 Mô hình dữ liệu quan hệ

## 3.7 Thiết kế dữ liệu

### 3.7.1 Từ điển dữ liệu

### 3.7.2 Mô hình diagram

## 3.8 Xây dựng dự án

### 3.8.1 Xây dựng BackEnd

# CHƯƠNG 4: DEMO GAME

# CHƯƠNG 5: ĐÁNH GIÁ KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## 4.1 Kết quả

# TÀI LIỆU THAM KHẢO

Danh mục tài liệu tham khảo