

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**KHOA: CÔNG NGHỆ THÔNG TIN**

-----



**BÁO CÁO BÀI TẬP LỚN**  
**HỌC PHẦN: ĐỒ ÁN CHUYÊN NGÀNH**  
**ĐỀ TÀI: XÂY DỰNG VÀ THIẾT KẾ GAME 2D PLATFORM**  
**“TINY FIGHT”**

**Giảng viên hướng dẫn:** ThS. Nguyễn Thái Cường

**Sinh viên thực hiện:** Hoàng Nguyễn Tùng Huy - 2022601138

Nguyễn Trung Hiếu - 2022600419

Nguyễn Mạnh Hoàn - 2022600679

Đình Duy Thái - 2021602866

Nguyễn Khải Hưng - 2022600876

**Nhóm:** 5

**Lớp – Khóa:** 20241IT6071003 – K17

**Hà Nội, năm 2024**

## LỜI CẢM ƠN

Trước tiên, em xin bày tỏ lòng biết ơn sâu sắc đến quý thầy cô trong khoa đã tận tình giảng dạy, truyền đạt những kiến thức quý báu và luôn tạo điều kiện thuận lợi để nhóm em có thể học tập và rèn luyện trong suốt quá trình học vừa qua. Những kiến thức và kinh nghiệm mà thầy cô chia sẻ chính là nền tảng vững chắc giúp nhóm em hoàn thành tốt bài báo cáo này.

Đặc biệt, nhóm em xin chân thành cảm ơn quý thầy Nguyễn Thái Cường là giảng viên môn Đồ án chuyên ngành đã tận tình giảng dạy và hướng dẫn em trong suốt quá trình học tập và thực hiện bài báo cáo này. Chúng em xin cảm ơn các lời nhận xét và ý kiến bổ sung của thầy đã hỗ trợ, đóng góp ý kiến để bài làm được hoàn thiện tốt hơn.

**Nhóm sinh viên thực hiện**

Hoàng Nguyễn Tùng Huy

Nguyễn Trung Hiếu

Nguyễn Mạnh Hoàn

Đinh Duy Thái

Nguyễn Khải Hưng

## MỤC LỤC

LỜI CẢM ƠN .....	i
MỤC LỤC .....	ii
DANH MỤC HÌNH ẢNH .....	iv
DANH MỤC BẢNG BIỂU .....	vii
DANH MỤC KÝ HIỆU, THUẬT NGỮ, TỪ VIẾT TẮT .....	viii
MỞ ĐẦU .....	1
CHƯƠNG 1: TỔNG QUAN VỀ TRÒ CHƠI ĐIỆN TỬ .....	3
1.1. Tổng quan về trò chơi điện tử .....	3
1.2. Các công cụ phá triển game .....	4
1.2.1. Tổng quan về Engine Unity .....	4
1.2.2. Tổng quan về Microsoft Visual Studio .....	15
1.3. Giới thiệu về đề tài game “Tiny fight” .....	17
CHƯƠNG 2: THIẾT KẾ GAME .....	19
2.1. Giới thiệu tổng quan .....	19
2.1.1. Thể loại game và yếu tố game .....	19
2.1.2. Đối tượng chơi .....	20
2.1.3. Nền tảng .....	20
2.2. Thiết kế kịch bản game .....	20
2.2.1. Cốt truyện game .....	20
2.2.2. Luật chơi .....	22
2.2.3. Tương tác và điều khiển game .....	22
2.3. Thiết kế các phần tử của game .....	23
2.3.1. Người chơi (Player) .....	23
2.3.2. Quái thường (enemy) và quái khủng (boss) .....	25
2.3.3. Đối tượng có thể tương tác (Interactables) .....	27

2.4. Thiết kế màn chơi.....	28
2.5. Thiết kế giao diện.....	29
2.5.1. Menu khởi đầu .....	29
2.5.2. UI trong game .....	29
2.5.3. Màn hình tăng giảm âm lượng.....	30
2.5.4. Màn hình nâng cấp nhân vật.....	30
2.5.5. Màn hình dừng game .....	31
2.5.6. Màn hình chiến thắng.....	31
2.5.7. Màn hình thua trận.....	32
2.5.8. Story board.....	32
2.6. Tài nguyên.....	33
CHƯƠNG 3: CÁC KỸ THUẬT XÂY DỰNG GAME VÀ KẾT QUẢ THỰC NGHIỆM.....	34
3.1. Các kỹ thuật xây dựng game.....	34
3.1.1. Các kỹ thuật đối với Player.....	34
3.1.2. Các kỹ thuật đối với quái nhỏ .....	39
3.1.3. Các kỹ thuật đối với quái khủng .....	41
3.1.4. Kỹ thuật thiết kế map.....	45
3.2. Kết quả thực nghiệm .....	46
3.2.1. Main menu .....	46
3.2.2. Màn hình game .....	48
3.2.3. Màn hình chiến thắng và thua game .....	49
KIỂM THỬ GAME .....	50
KẾT LUẬN .....	59
TÀI LIỆU THAM KHẢO.....	60
PHỤ LỤC .....	61

## DANH MỤC HÌNH ẢNH

Hình 1.1. Logo Unity .....	5
Hình 1.2. Main menu scene.....	9
Hình 2.1. Nhân vật chính Pink Nhon .....	23
Hình 2.2. Nhân vật di chuyển.....	23
Hình 2.3. Nhân vật nhảy .....	24
Hình 2.4. Nhân vật chạy.....	24
Hình 2.5. Nhân vật lướt.....	24
Hình 2.6. Nhân vật bắn đạn.....	24
Hình 2.7. Nhân vật nhận sát thương .....	24
Hình 2.8. Nhân vật chết.....	24
Hình 2.9. Bảng mô tả hệ thống quái thường .....	25
Hình 2.10. Elite Wolf .....	25
Hình 2.11. Newt Slinger.....	25
Hình 2.12. Bat .....	25
Hình 2.13. Elite Wolf .....	26
Hình 2.14. Necromancer .....	26
Hình 2.15. Undead .....	27
Hình 2.16. Rương báu .....	27
Hình 2.17. Kinh nghiệm.....	27
Hình 2.18. Bình máu .....	27
Hình 2.19. Bình năng lượng.....	28
Hình 2.20. Sao trời .....	28
Hình 2.21. Bẫy gai .....	28
Hình 2.22. Thang.....	28
Hình 2.23. Bom .....	28
Hình 2.24. Hình dung màn hình menu chính.....	29
Hình 2.25. Hình dung màn hình giao diện trong game.....	29

Hình 2.26. Hình dung màn hình tăng giảm âm lượng .....	30
Hình 2.27. Hình dung màn hình nâng cấp nhân vật.....	30
Hình 2.28. Hình dung màn hình dừng game.....	31
Hình 2.29. Hình dung màn hình chiến thắng .....	31
Hình 2.30. Hình dung màn hình thua trận.....	32
Hình 2.31. Story board của game.....	32
Hình 3.1. Di chuyển nhân vật .....	34
Hình 3.2. Tốc độ di chuyển thay đổi theo trạng thái.....	34
Hình 3.3. Nhảy với giới hạn số lần .....	35
Hình 3.4. Đặt lại số lần nhảy khi chạm đất.....	35
Hình 3.5. Kỹ thuật lướt với thời gian hồi chiêu .....	35
Hình 3.6. Phát hiện và xử lý trò thang .....	36
Hình 3.7. Trạng thái khi trò thang .....	36
Hình 3.8. Tấn công tiêu tốn mana.....	37
Hình 3.9. Kỹ thuật tăng tốc trong thời gian ngắn .....	37
Hình 3.10. Phát hiện đạn va chạm với người chơi.....	38
Hình 3.11. Nhận sát thương và bị đẩy lùi .....	38
Hình 3.12. Đặt nhân vật tại điểm spawn.....	39
Hình 3.13. Kiểm tra thể trạng khi máu xuống dưới 0.....	39
Hình 3.14. Di chuyển và đổi hướng của quái nhỏ .....	39
Hình 3.15. Phát hiện người chơi trong tầm thông qua khoảng cách.....	40
Hình 3.16. Thực hiện tấn công của quái đánh xa.....	40
Hình 3.17. Thực hiện tấn công của quái đánh gần.....	40
Hình 3.18. Chọn ngẫu nhiên vị trí mục tiêu.....	41
Hình 3.19. Boss di chuyển ngẫu nhiên trong vùng giới hạn.....	41
Hình 3.20. Phát hiện người chơi trong tầm truy đuổi .....	41
Hình 3.21. Boss đuổi theo người chơi khi người chơi ở trong phạm vi .....	42
Hình 3.22. Boss thực hiện tấn công .....	42
Hình 3.23. Boss phát hiện và né tránh đạn.....	43

Hình 3.24. Boss chết và rơi vật phẩm .....	43
Hình 3.25. FSM chuyển đổi giữa các trạng thái dựa trên các điều kiện.....	44
Hình 3.26. Tile map.....	45
Hình 3.27. Map chơi 1 .....	45
Hình 3.28. Map chơi 2 .....	46
Hình 3.29. Map chơi 3 .....	46
Hình 3.30. Main menu .....	46
Hình 3.31. Tutorial .....	47
Hình 3.32. Chọn màn chơi .....	47
Hình 3.33. Màn hình trong game .....	48
Hình 3.34. Màn hình dừng game .....	48
Hình 3.35. Màn hình chiến thắng.....	49
Hình 3.36. Màn hình thua trận .....	49

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1. Bảng mô tả nhân vật chính .....	23
Bảng 2.2. Bảng mô tả hệ thống quái khủng .....	26
Bảng 2.3. Bảng mô tả các đối tượng có thể tương tác .....	27
Bảng 3.1. Bảng kiểm thử các chức năng của nhân vật .....	52
Bảng 3.2. Bảng kiểm thử các chức năng của quái .....	54
Bảng 3.3. Bảng kiểm thử tương tác với vật thể môi trường .....	55
Bảng 3.4. Bảng kiểm thử chức năng hệ thống và giao diện .....	56
Bảng 3.5. Bảng tổng hợp kết quả kiểm thử.....	58



## DANH MỤC KÝ HIỆU, THUẬT NGỮ, TỪ VIẾT TẮT

STT	Ký hiệu / Thuật ngữ	Tiếng Anh	Giải nghĩa
1	RPG	Role-Playing Game	Game nhập vai
2	Retro Style	Retro Style	Phong cách đồ họa mô phỏng các game cổ điển
3	IDE	Integrated Development Environment	Môi trường phát triển tích hợp
4	FPS	Frame Per Second	Số khung hình hiển thị mỗi giây
5	GameObject	Game Object	Đối tượng trong game có thể nhìn thấy hoặc tương tác
6	Component	Component	Thành phần mở rộng chức năng cho GameObject
7	Prefab	Prefabricated Object	Mẫu đối tượng có thể tái sử dụng
8	Asset	Asset	Tài nguyên game
9	Script	Script	Mã lệnh lập trình
10	Rigidbody	Rigidbody	Thành phần vật lý
11	Collider	Collider	Thành phần xử lý va chạm
12	UI	User Interface	Giao diện người dùng
13	HP	Health Points	Điểm máu của nhân vật

14	MP / Mana	Mana Point	Năng lượng sử dụng để thi triển kỹ năng
15	FSM	Finite State Machine	Máy trạng thái hữu hạn
16	Animator	Animator	Công cụ điều khiển hoạt ảnh trong Unity
17	Avatar	Avatar	Đối tượng mô phỏng nhân vật trong
18	Scene	Scene	Màn chơi hoặc không gian trong game
19	Inspector	Inspector Panel	Bảng thuộc tính để điều chỉnh các GameObject
20	Hierarchy	Hierarchy Panel	Cây phân cấp các GameObject trong một Scene
21	HUD	Heads-Up Display	Thành phần giao diện hiển thị khi chơi game (máu, năng lượng...)
22	Pixel Art	Pixel Art	Phong cách đồ họa 2D
23	Trigger	Trigger Collider	Va chạm kích hoạt sự kiện
24	NPC	Non-Player Character	Nhân vật không do người chơi điều khiển
25	Spawn Point	Spawn Point	Điểm sinh ra của nhân vật/quái
26	Visual Scripting	Visual Scripting	Lập trình kéo-thả

27	Animator Controller	Animator Controller	Bộ điều phối hoạt ảnh
28	HP Potion	Health Potion	Vật phẩm giúp hồi máu
29	Energy Potion	Mana Potion	Vật phẩm giúp hồi năng lượng
30	AoE	Area of Effect	Kỹ năng hoặc hiệu ứng diện rộng
31	BGM	Background Music	Nhạc nền trong game
32	SFX	Sound Effects	Hiệu ứng âm thanh trong game
33	DOTS	Data-Oriented Technology Stack	Công nghệ xử lý hiệu suất cao của Unity
34	SRP	Scriptable Render Pipeline	Hệ thống render tùy biến theo yêu cầu của Unity
35	2D/3D	2 dimensional/ 3 dimensional	Hai chiều / Ba chiều
36	Cross-platform	Cross-platform	Đa nền tảng
37	AI	Artificial Intelligent	Trí tuệ nhân tạo
38	Level design	Level design	Thiết kế màn chơi
39	Gimmick	Gimmick	Cơ chế đặc biệt hoặc bất ngờ
40	Platformer	Platformer	Thể loại game

## MỞ ĐẦU

Trong bối cảnh ngành công nghệ thông tin không ngừng phát triển, game điện tử đang dần trở thành một ngành công nghiệp lớn thu hút đông đảo giới trẻ. Cũng như phát triển phần mềm hay một website, phát triển game không chỉ yêu cầu sự đầu tư nghiên cứu, tư duy lập trình mà còn cần một niềm đam mê đối với ngành công nghiệp giải trí này. Do đó, nhóm chúng em đã quyết định triển khai đề tài game của mình: “Tiny Fight”.

Đề tài kể trên là một tựa game 2D platform đơn giản và rất phổ biến hiện nay. Bằng các kiến thức đã trang bị từ các học phần trước cùng những nghiên cứu bằng cả quan sát lẫn trải nghiệm thực tế, nhóm có thể ứng dụng để xây dựng, thiết kế game và lập trình game 2D có phong cách đồ họa pixel cũng như mang đậm dấu ấn retro để tạo được nét riêng so với những game có cùng đề tài trên thị trường.

Báo cáo này được chia thành các chương với nội dung cụ thể như sau:

- **Chương 1: Tổng quan về trò chơi điện tử:** Chương này trình bày bối cảnh phát triển của game điện tử hiện đại, giới thiệu tổng quan về ý tưởng cũng như mục tiêu của đề tài, đồng thời nêu ra các yêu cầu chức năng cơ bản mà game cần đáp ứng.
- **Chương 2: Thiết kế game:** Chương này đi sâu vào khía cạnh thiết kế, bao gồm kịch bản, bố cục màn chơi, thiết kế nhân vật, vũ khí, giao diện và các yếu tố đồ họa, âm thanh. Mục tiêu là giúp người đọc có cái nhìn trực quan về quá trình hình thành ý tưởng và triển khai sản phẩm.
- **Chương 3: Các kỹ thuật xây dựng game và kết quả thực nghiệm:** Chương này dành để trình bày quá trình cài đặt, thử nghiệm và nhận xét về hiệu năng của game, kèm theo các hình ảnh minh họa, bảng số liệu và phân tích kết quả nhằm đánh giá mức độ hoàn thiện của sản phẩm.

Qua đề tài kể trên, nhóm chúng em sẽ được trang bị những kinh nghiệm về quy trình phát triển ứng dụng game thực tế cũng như nâng cao trình độ thiết

kế và phát triển game của các thành viên trong nhóm và mở rộng khả năng ứng dụng công cụ Unity và ngôn ngữ lập trình C# vào thực tiễn. Ngoài các kỹ năng về kiến thức chuyên ngành, đề tài lần này cũng là một dự án mà nhóm có thể phát triển được kỹ năng giao tiếp cũng như làm việc nhóm một cách hiệu quả.

## CHƯƠNG 1: TỔNG QUAN VỀ TRÒ CHƠI ĐIỆN TỬ

### 1.1. Tổng quan về trò chơi điện tử

Trong thời đại số hóa hiện nay, thị trường trò chơi điện tử được gọi là một “Ngành công nghiệp sáng tạo” tạo ra nhiều giá trị lớn cũng như cung cấp nhiều việc làm. Với sự phát triển của máy tính cá nhân và Internet tốc độ cao, trò chơi điện tử trở nên phổ biến toàn cầu và tạo ra doanh thu hơn 150 tỷ USD mỗi năm (Newzoo, 2019).. Sự thu hút này đã thu hút sự quan tâm lớn của các nhà đầu tư và nhà khoa học[1].

Việt Nam nằm trong số 5 quốc gia hàng đầu thế giới về lĩnh vực sản xuất game mobile tính theo lượt tải xuống trong nửa đầu năm 2023. Sự trỗi dậy của ngành công nghiệp trò chơi điện tử không phải là điều quá ngạc nhiên đối với quốc gia có tỷ lệ sử dụng smartphone thuộc hàng cao nhất châu Á và có khoảng một nửa dân số dưới 30 tuổi. Năm 2013, Việt Nam lần đầu tiên thu hút sự chú ý của các game thủ toàn sau cơn sốt Flappy Bird. Đây là một trò chơi đơn giản nhưng gây nghiện, được phát triển bởi Nguyễn Hà Đông. Game này nổi tiếng đến mức trở thành hiện tượng toàn cầu. Sự thành công bất ngờ từ một tựa game di động đơn giản đã “kích thích” các nhà sản xuất game trong nước. Và một thập kỷ sau, Việt Nam đang trên con đường trở thành một cường quốc game thực sự[2].

Ngày nay, chúng ta sống trong một thế giới với đồ họa 3D siêu thực được dựng bởi những cỗ máy đủ mạnh để có thể “nổi dậy” và thống trị loài người. Một số trò chơi còn thách thức người chơi so sánh hình ảnh trong game với thực tế để phân biệt đâu là game, đâu là đời thực. Những tiến bộ công nghệ mới thậm chí cho phép quét toàn bộ cơ thể diễn viên rồi áp dụng như một texture lên mô hình 3D. Mô hình đó cũng có thể được xây dựng từ dữ liệu quét 3D và đặt vào môi trường game thời gian thực. Kết quả là một nhân vật 3D, khi được chiếu sáng phù hợp, có thể vượt qua “thung lũng kỳ lạ”[3] (Ám chỉ khi một

nhân vật 3D đạt đến mức độ chân thực đủ cao để não người không phân biệt được với thực tế).

Tuy nhiên, nhiều game thủ lại khao khát một thứ gì đó mang hơi hướng cổ điển hơn. Một số muốn trải nghiệm game đơn giản, không cần đến sự phức tạp của không gian ba chiều. Nhiều người chỉ nhớ về một thời kỳ giản dị, khi cảnh vật chỉ di chuyển trên hai chiều thay vì ba. Chính vì nhóm game thủ cực kỳ đông đảo này, nghệ thuật làm game 2D đã được hồi sinh.

Ngoài ra, sự bùng nổ của thiết bị di động và máy tính bảng trong năm năm qua cũng góp phần vào sự trỗi dậy của game 2D do giới hạn phần cứng của các thiết bị này. Tuy nhiên, sự hồi sinh này không đi kèm với công nghệ lỗi thời từng được dùng để tạo ra những tựa game 2D kinh điển và tiến hóa thành công cụ làm game hiện đại ngày nay. Thay vào đó, công nghệ game 2D hiện tại đã tận dụng sức mạnh tạo nên những tựa game đỉnh đám ngày nay và kết hợp với những ưu điểm thiết kế giúp các trò chơi điện tử đầu tiên trở nên khả thi.

Trong bối cảnh đó, đề tài “Tiny Fight” hướng tới việc khai thác những ưu điểm của thể loại 2D platformer: từ cách điều khiển nhân vật, cơ chế bắn hạ quái vật cho đến việc thiết kế màn chơi với đầy đủ các yếu tố thử thách như chướng ngại vật, kẻ địch và vật phẩm nâng cấp. Mục tiêu của đề tài không chỉ là tạo ra một trò chơi giải trí hấp dẫn mà còn giúp nhóm nghiên cứu vận dụng kiến thức về công nghệ và nghệ thuật đồ họa pixel để phát triển sản phẩm game thực tế, góp phần nâng cao kỹ năng lập trình và thiết kế của các thành viên trong nhóm.

## **1.2. Các công cụ phá triển game**

### **1.2.1. Tổng quan về Engine Unity**

#### **1.2.1.1. Giới thiệu**

Unity được dùng để làm video game, hoặc những nội dung có tính tương tác như thể hiện kiến trúc, hoạt hình 2D, 3D thời gian thực. Unity hao hao với

Director, Blender game engine, Virtools hay Torque Game Builder trong khía cạnh dùng môi trường đồ họa tích hợp ở quá trình phát triển game là chính.

Unity là một trong những engine được giới làm game không chuyên cực kỳ ưa chuộng bởi khả năng tuyệt vời của nó là phát triển trò chơi đa nền. Trình biên tập có thể chạy trên Windows và Mac OS, và có thể xuất ra game cho Windows, Mac, Wii, iOS, Android. Game cũng có thể chơi trên trình duyệt web thông qua plugin Unity Web Player. Unity mới bổ sung khả năng xuất ra game trên widget cho Mac, và cả Xbox 360, PlayStation 3.



*Hình 1.1. Logo Unity*

Tuy nhiên, để hiểu rõ hơn lý do tại sao Unity lại phổ biến trong thế giới phát triển trò chơi, chúng ta hãy cùng tìm hiểu một số yếu tố chính tạo nên sự nổi tiếng của Unity.

#### – Khả năng hỗ trợ đa nền tảng

Unity được thiết kế như một công cụ phát triển game đa nền tảng (cross-platform). Điều này có nghĩa là nhà phát triển chỉ cần xây dựng một lần, có thể triển khai trò chơi trên nhiều thiết bị và hệ điều hành khác nhau mà không tốn thêm nhiều công đoạn porting. Khả năng đa nền tảng rộng lớn giúp tối đa hóa đối tượng người chơi tiềm năng và tiết kiệm thời gian, công sức phát triển, một trong những lý do chính giúp Unity trở nên phổ biến trong ngành game hiện nay

#### – Hỗ trợ phát triển 2D, 3D và VR/AR

Unity cho phép tạo cả game 2 chiều và 3 chiều trong cùng một môi trường. Công cụ này cung cấp các hệ thống xử lý sprite, vật lý 2D và trình diễn thế giới



2D, đồng thời hỗ trợ đầy đủ cho đồ họa 3D (texture, ánh sáng, đổ bóng, v.v.). Ngoài ra, Unity cũng có bộ công cụ mạnh mẽ dành cho phát triển thực tế ảo (VR) và thực tế tăng cường (AR). Khả năng bao quát cả 2D, 3D và công nghệ VR/AR khiến Unity trở thành lựa chọn ưu tiên cho cả game truyền thống lẫn các ứng dụng tương tác mới, vì nhà phát triển không phải học nhiều công cụ khác nhau cho từng loại game.

#### – **Giá trị kinh tế và bản quyền**

Unity có lợi thế về mặt kinh tế nhờ mô hình bản quyền linh hoạt. Phiên bản Unity Personal dành cho cá nhân hoặc nhóm nhỏ có doanh thu hạn chế (dưới 100.000 – 200.000 USD/năm) được sử dụng miễn, giúp bất kỳ ai cũng có thể tiếp cận và bắt đầu phát triển game. Các gói đăng ký trả phí (Unity Plus, Pro, Enterprise) bổ sung thêm tính năng cao cấp phục vụ nhóm chuyên nghiệp. Cách phân phối này cho phép nhà phát triển quy mô nhỏ tiết kiệm chi phí khởi đầu, đồng thời cung cấp lộ trình nâng cấp cho dự án lớn hơn. Sự kết hợp giữa phiên bản miễn phí (chi phí thấp) và tính năng chuyên nghiệp của bản trả phí đóng góp vào sức hấp dẫn chung của Unity trong cộng đồng phát triển game.

#### – **Đồ họa và hiệu ứng hình ảnh chất lượng cao**

Unity được trang bị bộ công cụ đồ họa hiện đại, đáp ứng các yêu cầu hình ảnh ngày càng cao của game hiện nay. Các tính năng như Data-Oriented Technology Stack (DOTS) và Scriptable Render Pipeline (SRP) cho phép tận dụng tối đa tài nguyên phần cứng và tối ưu hóa hiệu suất đồ họa, giúp game có thể chạy mượt mà ngay cả trên thiết bị cấu hình thấp. Nhờ SRP, nhà phát triển có thể tinh chỉnh quy trình dựng hình để đạt được hiệu ứng phức tạp và độ chân thực cao. Tất cả các công nghệ này làm tăng năng lực thị giác của Unity, khiến các tựa game phát triển bằng Unity có thể đạt chất lượng đồ họa cao, từ đó được nhiều studio lớn lẫn nhỏ lựa chọn.

### – Tính năng Chế độ chơi (Play mode)

Một trong những tính năng nổi bật của Unity là Chế độ chơi (Play Mode) trong Unity Editor. Đây là cơ chế cho phép nhà phát triển chạy thử trò chơi ngay trong môi trường phát triển, chỉ với một cú nhấn nút “Play”. Theo tài liệu chính thức, “khả năng kiểm thử ứng dụng bằng cách chuyển từ chế độ Edit sang Play là một trong các tính năng cốt lõi của Unity”. Chế độ này cho phép xem trước chính xác cách trò chơi hoạt động như khi build lên thiết bị thật, từ đó phát hiện lỗi và điều chỉnh một cách nhanh chóng.

### – Giao diện người dùng thân thiện

Ngoài lợi thế quan trọng xoay quanh kiến thức mã hóa tối thiểu mà việc tạo trò chơi bằng Unity yêu cầu, công cụ trò chơi vô song này còn tự hào với giao diện thân thiện và đơn giản. Về mặt này, người dùng thấy giao diện thân thiện với nhà phát triển, có thể lập trình hoàn toàn của Unity dễ sử dụng và trực quan, cho phép họ dễ dàng biến ý tưởng sáng tạo của mình thành hiện thực.

### – Cửa hàng tài nguyên phong phú

Unity Asset Store là kho tài nguyên tích hợp sẵn, cung cấp vô số nội dung đã được tạo sẵn giúp tăng tốc phát triển game. Tại đây có đa dạng asset như mô hình 3D, sprite 2D, hoạt ảnh, hiệu ứng âm thanh, kịch bản mẫu... Một nguồn mô tả Asset Store như “một kho tàng đồ sộ các tài nguyên đã có sẵn” từ 3D model đến hiệu ứng âm thanh. Khi cần, nhà phát triển chỉ cần mua hoặc tải xuống các asset này thay vì tự tạo, giúp giảm đáng kể thời gian và chi phí sản xuất nội dung.

### – Phát triển ít mã hóa (Low-code)

Unity hỗ trợ mạnh mẽ phát triển trò chơi mà không cần viết nhiều mã thủ công. Ví dụ, engine này có hệ thống visual scripting (như Bolt) cho phép thiết kế logic trò chơi bằng cách kéo-thả các khối node thay vì viết code C# truyền thống. Theo một nguồn, visual scripting của Unity cho phép người dùng “tạo

logic và kịch bản bằng cách kết hợp các nút, loại bỏ nhu cầu viết mã thủ công”. Nhờ vậy, các nhà thiết kế và họa sĩ không chuyên lập trình cũng có thể nhanh chóng tạo ra tính năng mới mà không phải lo lắng về cú pháp ngôn ngữ.

#### – Cộng đồng phát triển và hỗ trợ mạnh mẽ

Unity sở hữu cộng đồng phát triển rộng lớn và tích cực. Cộng đồng này được xem như một tài sản quan trọng, cung cấp hàng loạt diễn đàn, blog, kho tutorial và mã nguồn mẫu hỗ trợ từ cơ bản đến nâng cao. Khi gặp khó khăn kỹ thuật, nhà phát triển có thể dễ dàng tìm câu trả lời qua diễn đàn Unity, Stack Overflow hay các kênh mạng xã hội. Ngoài ra, Unity thường xuyên tổ chức sự kiện, hội thảo và cập nhật công nghệ mới, tạo sân chơi để các lập trình viên chia sẻ kiến thức và kinh nghiệm. Môi trường cộng tác và hỗ trợ lẫn nhau này không chỉ giúp giải quyết vấn đề nhanh chóng mà còn kích thích sáng tạo, tạo nên hệ sinh thái bền vững khiến Unity luôn giữ vững vị thế hàng đầu[4].

#### 1.2.1.2. Một số thành phần và các khái niệm quan trọng trong Unity

Unity được xây dựng dựa trên một kiến trúc theo thành phần, trong đó hầu hết các đối tượng trong game đều là các GameObject được mở rộng chức năng thông qua các Component. Các thành phần và khái niệm cơ bản dưới đây giúp người phát triển hiểu rõ cách hoạt động của engine:

##### **Asset**

Là tập hợp các tài nguyên được sử dụng trong dự án, bao gồm hình ảnh, âm thanh, video, script, mô hình 3D và các tệp dữ liệu khác. Tất cả các thành phần này được lưu trữ trong thư mục Assets của dự án, giúp quản lý và truy xuất dễ dàng trong quá trình phát triển.

##### **Scene**

Scene là không gian làm việc chính của Unity, nơi bạn bố trí các Game Object để tạo nên môi trường game. Mỗi Scene thường đại diện cho một màn chơi hoặc một phần của game. Quá trình chuyển đổi giữa các Scene giúp tạo ra trải nghiệm liền mạch cho người chơi.



*Hình 1.2. Main menu scene*

### **Game Object**

Đây là đối tượng cơ bản nhất trong Unity. Mọi đối tượng hiển thị hoặc tương tác trong game, từ nhân vật, vật phẩm cho đến các hiệu ứng đặc biệt, đều được tạo ra dưới dạng GameObject. Mỗi GameObject có thể chứa một hoặc nhiều Component để định nghĩa hành vi và thuộc tính của nó.

### **Component**

Components là các thành phần trong game, bổ sung tính năng cho các Game Object. Mỗi Component có chức năng riêng biệt. Đa phần các Component phụ thuộc vào Transform, vì nó lưu trữ các thông số cơ bản của Game Object. Bản chất của Game Object là không có gì cả, các đặc tính và khả năng của Game Object nằm hoàn toàn trong các Component. Do đó chúng ta có thể xây dựng nên bất kỳ Game Object nào trong game mà chúng ta có thể tưởng tượng được.

### **Prefab**

Prefab là mẫu đối tượng (GameObject) được lưu trữ để tái sử dụng nhiều lần. Khi một Prefab được tạo, mọi thay đổi về sau có thể được áp dụng cho tất

cả các bản sao, giúp tiết kiệm thời gian và duy trì tính nhất quán cho các đối tượng lặp đi lặp lại trong game.

### **Scripts**

Scripts được Unity xem như một Component. Đây là thành phần thiết yếu trong quá trình phát triển game. Bất kỳ một game nào, dù đơn giản nhất đều cần đến Scripts để tương tác với các thao tác của người chơi, hoặc quản lý các sự kiện để thay đổi chiều hướng của game tương ứng với kịch bản game.

Unity cung cấp cho lập trình viên khả năng viết Script bằng các ngôn ngữ: JavaScript, C#. Unity không đòi hỏi lập trình viên phải học cách lập trình trong Unity, nhưng trong nhiều tình huống, chúng ta cần sử dụng Script trong mỗi phần của kịch bản game.

Để viết Script, chúng ta có thể làm việc với một trình biên tập Script độc lập của Unity, hoặc làm việc trên Mono Developer được tích hợp vào Unity trong những phiên bản gần đây. Mono Developer là một IDE khá tốt, cung cấp nhiều chức năng tương tự Visual Studio chúng ta cũng có thể dùng Visual Studio để viết file C# như bình thường. Mã nguồn viết trên Mono Developer sẽ được cập nhật và lưu trữ trong dự án trên Unity.

### **Hierarchy và Inspector**

- **Hierarchy:** Là cửa sổ hiển thị cấu trúc phân cấp của tất cả các GameObject trong Scene, cho phép người phát triển quản lý và sắp xếp các đối tượng một cách trực quan.
- **Inspector:** Cung cấp giao diện để chỉnh sửa thuộc tính của GameObject và các Component được gắn kèm, cho phép điều chỉnh chi tiết mà không cần viết mã.

### **Collider**

Hệ thống xử lý va chạm bao gồm 2D và 3D. Được chia ra các va chạm của các hình cơ bản:

- **Is Trigger:** dạng true/false, cho phép va chạm có đi xuyên qua không?

- Material: tham chiếu đến physic material.
- Center: điều chỉnh thông số tâm của va chạm theo các trục tương ứng.
- Size: điều chỉnh kích thước theo các trục tương ứng.
- Capsule Colider: Va chạm dành cho các khối hình trụ.
- Is Trigger: dạng true/false, cho phép va chạm có đi xuyên qua không?
- Material: tham chiếu đến physic material.
- Center: điều chỉnh thông số tâm của va chạm theo các trục tương ứng.
- Radius: điều chỉnh kích thước bán kính.
- Height: độ cao
- Direction: xoay hình trụ theo các trục tương ứng.

**Box collider:** Va chạm cho các vật hình hộp chữ nhật đối với 3D và hình chữ nhật đối với 2D.

**Mesh Colilider:** Va chạm dạng lưới dành các vật thể không xác định hình thể.

**Sphere Colider:** va chạm áp dụng cho các vật hình cầu.

**Wheel collider:** dành cho vật thể hình bánh xe. Nó sẽ mô phỏng hệ thống va chạm giống với các bánh xe.

**Terrain Collider:** dành cho các vật thể địa hình và terrain collider sẽ dựa theo địa hình đó.

### **Rigidbody**

Hệ thống mô phỏng vật lý trong game. Cũng chia ra 2D

- Mass: khối lượng (đơn vị tùy ý).
- Drag: Sức cản không khí ảnh hưởng như thế nào với đối tượng khi di chuyển. 0 có nghĩa là không có sức cản không khí, và vô cùng làm cho các đối tượng di chuyển ngay lập tức dừng lại.
- Angular Drag: Sức cản không khí ảnh hưởng đến các đối tượng khi quay từ mô-men xoắn. 0 có nghĩa là không có sức cản không khí.

Không thể làm cho vật dừng quay hẳn chỉ bằng cách thiết lập Angular Drag của nó đến vô cùng.

- Use Gravity: Nếu được kích hoạt, các đối tượng bị ảnh hưởng bởi lực hấp dẫn.
- Is Kinematic: Nếu được kích hoạt, các đối tượng sẽ không được thúc đẩy bởi động cơ vật lý.
- Interpolate: giảm xóc.
- Collision detection: Được sử dụng để ngăn chặn các đối tượng chuyển động nhanh qua các đối tượng khác mà không phát hiện va chạm.
- Constraints: Ràng buộc Những hạn chế về chuyển động của Rigidbody.

### **Sprite**

Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó. Unity cho phép tùy chỉnh màu sắc, kích thước, độ phân giải của một hình ảnh 2D.

### **Animator**

Trong 2D thì animation là tập một hình ảnh động dựa trên sự thay đổi liên tục của nhiều sprite khác nhau. Trong 3D là một tập hợp các sự thay đổi theo thời gian của đối tượng trong không gian. Mỗi thay đổi là một key frame. Key Frame hay Frame là một trạng thái của một animation. Animator gồm các thành phần:

- Controller: Bộ điều khiển animation gắn liền với nhân vật này. Nó sẽ quản lý các animation clip, các thông số tốc độ của animation, thứ tự các clip...
- Avatar: thành phần tạo hình ảnh cho object.
- Apply Root Motion: dạng true-false, cho phép thiết lập animation có di chuyển theo không gian đã được tạo khi cấu hình animation.
- Animate Physics: dạng true-false, khi được chọn, các hình ảnh trong animation sẽ có thể tương tác vật lý với nhau.

- Culling mode: chọn chế độ cho hình ảnh động.

### **Audio Source**

Âm thanh trong game. Gồm cả âm thanh 2D và 3D.

- Audio clip: tham chiếu đến file âm thanh.
- Mute: chơi ở chế độ như tắt tiếng.
- Bypass effects: bộ lọc hiệu ứng áp dụng cho các nguồn âm thanh.
- Bypass listener effects: Điều này là để nhanh chóng chuyển tất cả các hiệu ứng Listener on / off.
- Pass Reverb Zones: Điều này là để nhanh chóng chuyển tất cả các khu Reverb on / off.
- Play on awake: Nếu được kích hoạt, âm thanh sẽ bắt đầu chơi lúc cảnh ra mắt. Nếu vô hiệu hóa, cần phải bắt đầu nó bằng cách sử dụng lệnh Play () từ kịch bản script.
- Loop: Kích hoạt tính năng này để làm cho Clip âm thanh lặp lại.
- Pitch: Xác định ưu tiên của nguồn âm thanh này trong số tất cả những nguồn âm cùng tồn tại trong bối cảnh đó. (Ưu tiên: 0 = quan trọng nhất, 256 = ít quan trọng nhất Mặc định = 128).
- 3D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh nếu Audio Clip là một âm thanh 3D.
- 2D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh nếu Audio Clip là một âm thanh 2D.

### **Camera**

Là một game object đặc biệt trong scene, dùng để xác định tầm nhìn, quansát các đối tượng khác trong game. Bao gồm các thuộc tính:

- Clear flags: Xác định phần nào của màn hình sẽ bị xóa. Đây là tiện dụng khi sử dụng nhiều máy ảnh để vẽ các yếu tố trò chơi khác nhau.



- Background: Màu áp dụng cho các màn hình còn lại sau khi tắt cả các yếu tố trong quan điểm đã được rút ra và không có skybox.
- Culling Mask: Bao gồm hoặc bỏ qua lớp của các đối tượng được đưa ra bởi các Camera.
- Projection: khả năng của máy ảnh để mô phỏng góc nhìn.
- Field of view (thuộc tính chỉ xuất hiện khi chọn Perspective trong mục Projection): Chiều rộng của góc nhìn của Camera, đo bằng độ dọc theo trục Y.
- HDR: Cho phép High Dynamic Range dựng hình cho camera này.

### **Transform**

- Transform: quản lý object trong không gian ba chiều, theo ba thông số:
- Position: quản lý vị trí hiện tại của object.
- Rotation: quản lý các thông số quay của object theo các trục x, y, z.
- Scale: quản lý các thông số phóng to, thu nhỏ theo các trục x, y, z.

### **Renderer**

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

- Sprite: Các đối tượng Sprite để render. Đối tượng Sprite có thể được tạo ra từ textures bằng cách sử dụng các thiết lập Sprite.
- Color: Vertex màu.
- Material: Chất liệu được sử dụng để làm sprite.
- Sorting Layer: Các layer được sử dụng để xác định các ưu tiên của sprite này trong khi hiển thị.
- Order in Layer: Các ưu tiên của sprite trong layer của nó. Con số thấp hơn được kết xuất đầu tiên và con số tiếp theo phủ bên dưới.

Những thành phần và khái niệm cơ bản trên là nền tảng để người phát triển hiểu cách Unity xây dựng và quản lý các đối tượng trong game. Việc nắm vững

kiến trúc này giúp quá trình phát triển, gỡ lỗi và mở rộng chức năng game trở nên dễ dàng và hiệu quả hơn.[5]

## **1.2.2. Tổng quan về Microsoft Visual Studio**

### **1.2.2.1. Giới thiệu**

Visual Studio là môi trường phát triển tích hợp (IDE) hàng đầu của Microsoft, được thiết kế để hỗ trợ lập trình viên xây dựng ứng dụng đa nền tảng một cách mạnh mẽ và hiệu quả. Với khả năng tương thích nhiều ngôn ngữ lập trình như C#, C++, Python, JavaScript và TypeScript, Visual Studio cung cấp bộ công cụ toàn diện bao gồm IntelliSense (gợi ý code thông minh), trình gỡ lỗi (debugger) chuyên sâu, thiết kế giao diện kéo thả, và tích hợp liền mạch với các nền tảng điện toán đám mây như Azure. Phiên bản Visual Studio 2022 tiếp tục khẳng định vị thế với hiệu suất 64-bit vượt trội, hỗ trợ .NET 6, và tối ưu hóa cho Windows 11.

### **1.2.2.2. Tính năng**

Tính đến nay, Visual Studio vẫn được coi là phần mềm lập trình hệ thống hàng đầu, chưa có phần mềm nào có thể thay thế được nó. Được đánh giá cao như vậy bởi Visual Studio sở hữu nhiều tính năng cực kỳ hấp dẫn. Cụ thể:

- Đa nền tảng: Phần mềm lập trình Visual Studio của Microsoft hỗ trợ sử dụng trên nhiều nền tảng khác nhau. Không giống như các trình viết code khác, Visual Studio sử dụng được trên cả Windows, Linux và Mac Systems. Điều này cực kỳ tiện lợi cho lập trình viên trong quá trình ứng dụng.
- Đa ngôn ngữ lập trình: Không chỉ hỗ trợ đa nền tảng, Visual Studio cũng cho phép sử dụng nhiều ngôn ngữ lập trình khác nhau từ C#, F#, C/C++, HTML, CSS, Visual Basic, JavaScript,... Bởi vậy, Visual Studio có thể dễ dàng phát hiện và thông báo cho bạn khi các chương trình có lỗi.

- Hỗ trợ website: Visual Studio code cũng hỗ trợ website, đặc biệt trong công việc soạn thảo và thiết kế web.
- Kho tiện ích mở rộng phong phú: Mặc dù Visual Studio có hệ thống các ngôn ngữ hỗ trợ lập trình khá đa dạng. Nhưng nếu lập trình viên muốn sử dụng một ngôn ngữ khác, bạn có thể dễ dàng tải xuống các tiện ích mở rộng. Tính năng hấp dẫn này được hoạt động như một phần chương trình độc lập nên không lo làm giảm hiệu năng của phần mềm.
- Lưu trữ phân cấp: Phần lớn các tệp dữ liệu đoạn mã của Visual Studio đều được đặt trong các thư mục tương tự nhau. Đồng thời, Visual Studio cũng cung cấp một số thư mục cho các tệp đặc biệt để bạn lưu trữ an toàn, dễ tìm, dễ sử dụng hơn.
- Kho lưu trữ an toàn: Với Visual Studio, bạn có thể hoàn toàn yên tâm về tính lưu trữ, bởi phần mềm đã được kết nối GIT và một số kho lưu trữ an toàn được sử dụng phổ biến hiện nay.
- Màn hình đa nhiệm: Visual Studio sở hữu tính năng màn hình đa nhiệm, cho phép người dùng mở cùng lúc nhiều tập tin, thư mục dù chúng có thể không liên quan tới nhau.
- Hỗ trợ viết code: Khi sử dụng code vào trong lập trình, với Visual Studio, công cụ này có thể đề xuất tới các lập trình viên một số tùy chọn thay thế nhằm điều chỉnh đôi chút để đoạn code áp dụng thuận tiện hơn cho người dùng.
- Hỗ trợ thiết bị đầu cuối: Phần mềm Visual Studio cũng tích hợp các loại thiết bị đầu cuối, giúp người dùng không cần chuyển đổi giữa hai màn hình hay trở về thư mục gốc khi thực hiện một thao tác cần thiết nào đó.
- Hỗ trợ Git: Do kết nối với GitHub nên Visual Studio cho phép hỗ trợ sao chép, kéo thả trực tiếp. Các mã code này sau đó cũng có thể thay đổi và lưu lại trên phần mềm.

- Intellisense: Tính năng nhắc Intellisense được sử dụng hầu hết trong các phần mềm lập trình, bao gồm cả Visual Studio. Tuy nhiên, so với các trình viết mã, Visual Studio vẫn được đánh giá cao về tính chuyên nghiệp. Đặc biệt, tính năng này còn có thể phát hiện tất cả các đoạn mã không đầy đủ, nhắc lập trình viên, gợi ý sửa đổi, khai báo biến tự động trong trường hợp lập trình viên quên, giúp bổ sung cú pháp còn thiếu,...
- Tính năng comment: Một tính năng cũng khá hay ho, hỗ trợ cho người lập trình trong trường hợp “nhớ nhớ quên quên” đó là tính năng bình luận. Tính năng này cho phép lập trình viên để lại nhận xét, giúp dễ dàng ghi nhớ công việc cần hoàn thành, không bỏ sót công đoạn nào.

### 1.3. Giới thiệu về đề tài game “Tiny fight”

Đề tài game “Tiny Fight” là một dự án nghiên cứu và phát triển game hành động 2D mà trong đó tập trung vào lối chơi vượt ải kết hợp yếu tố bắn súng và né tránh. Mục tiêu của đề tài là xây dựng một sản phẩm game mang đậm phong cách retro, với đồ họa pixel-art đơn giản nhưng sinh động, đồng thời tích hợp các cơ chế gameplay đa dạng nhằm tạo ra trải nghiệm chơi game hấp dẫn cho người dùng.

Game “Tiny Fight” được thiết kế dựa trên nguyên tắc cân bằng giữa tính giải trí và thử thách. Người chơi sẽ điều khiển nhân vật chính vượt qua các màn chơi được xây dựng với cấu trúc đa dạng, từ những màn dễ làm quen cho người mới bắt đầu cho tới các màn chơi có độ khó tăng dần với sự xuất hiện của nhiều loại quái vật, bẫy nguy hiểm và các chướng ngại vật độc đáo. Trong mỗi màn chơi, người chơi không chỉ phải khéo léo sử dụng các chiêu thức di chuyển và kỹ năng tấn công mà còn cần thu thập vật phẩm để nâng cấp sức mạnh, từ đó có thể đối đầu hiệu quả với các con boss mạnh mẽ ở cuối màn.

Đề tài nhằm khai thác những ưu điểm của công nghệ phát triển game hiện đại, đặc biệt là sự linh hoạt và mạnh mẽ của Unity kết hợp với ngôn ngữ lập

trình C#. Qua đó, dự án không chỉ hướng đến việc tạo ra một sản phẩm giải trí chất lượng mà còn là cơ hội để nhóm nghiên cứu vận dụng và rèn luyện kỹ năng lập trình, thiết kế giao diện người dùng, xử lý hiệu ứng âm thanh cũng như tối ưu hóa trải nghiệm chơi game trên nhiều thiết bị khác nhau.

Thông qua việc thực hiện đề tài “Tiny Fight”, nhóm nghiên cứu hy vọng sẽ đóng góp một sản phẩm game có giá trị, thể hiện được sự sáng tạo trong thiết kế và phát triển game điện tử. Đồng thời, đề tài cũng mở ra hướng đi mới cho việc ứng dụng các công nghệ tiên tiến trong lĩnh vực phát triển game, góp phần thúc đẩy sự phát triển chung của ngành công nghiệp trò chơi điện tử trong thời đại số hóa hiện nay.

## CHƯƠNG 2: THIẾT KẾ GAME

### 2.1. Giới thiệu tổng quan

#### 2.1.1. Thể loại game và yếu tố game

##### Khái niệm game nhập vai

- Game nhập vai (Role-playing games), viết tắt là RPG là tựa game mà khi người chơi hay game thủ tham gia tựa game này, người chơi sẽ được hóa thân thành một nhân vật trong Game có khả năng điều khiển và kiểm soát nhân vật và có thể sáng tạo cho mình từ ngoại hình, màu da hay trang bị.
- Với Game nhập vai (RPG), tựa tựa game này sẽ biến người chơi thành một nhân vật trong game với vai trò nhất định trong một thế giới được thiết lập sẵn. Trong thế giới đó, bạn toàn quyền có khả năng điều khiển nhân vật của mình và sử dụng các tài nguyên game cho để phát triển câu chuyện của mình và chiến thắng trò chơi[6].

##### Khái niệm game phiêu lưu

Trò chơi phiêu lưu hay trò chơi mạo hiểm là một thể loại video game mà trong đó giả định người chơi là nhân vật chính trong một câu chuyện có tính tương tác tiến triển theo hướng khám phá và vượt qua thử thách. Game thuộc thể loại này thường xây dựng cốt truyện đủ lôi cuốn để giữ chân người chơi, cho phép nó được diễn dịch sang một phạm vi rộng lớn loại hình truyền thông mang tính tường thuật như sách vở hay điện ảnh, và bao hàm hàng loạt thể loại văn chương. Gần như toàn bộ trò chơi phiêu lưu (dưới hình thức văn bản hay đồ họa) được thiết kế cho một người chơi, bởi vì chúng nhấn mạnh vào kịch bản và nhân vật, khiến cho nhiều người chơi sẽ trở nên khó khăn[7].

“Tiny Fight” cũng là một tựa game phiêu lưu 2D, nơi mà người chơi sẽ được hóa thân thành nhân vật chính tên “Pink Nhon”, một quái vật nhỏ màu hồng. Bằng các yếu tố hành động, chiến thuật và khám phá, người chơi sẽ được

đắm chìm trong một thế giới giả tưởng đầy bí ẩn và nguy hiểm, từ đó trò chơi tạo ra một trải nghiệm chơi game thú vị và cuốn hút.

### **2.1.2. Đối tượng chơi**

Trò chơi được phát triển cho những độ tuổi trẻ để giải trí, thư giãn sau những phút học tập, làm việc mệt mỏi. Với lối chơi game đơn giản bằng các thao tác bấm phím/chuột và luật chơi dễ hiểu. Độ tuổi tối thiểu để có thể thao tác game phù hợp là từ trên 12 tuổi.

### **2.1.3. Nền tảng**

Game được xây dựng bằng Unity nên có khả năng phát triển tốt trên các nền tảng lớn. Nhưng với thời gian hạn chế cũng như quy mô hiện tại của học phần, nhóm chúng em quyết định game sẽ tập trung chủ yếu vào nền tảng Windows và rất mong có cơ hội mở rộng ra các nền tảng trên mobile như IOS hay Android ở các dự án tiếp theo.

## **2.2. Thiết kế kịch bản game**

### **2.2.1. Cốt truyện game**

Pink Nhon, một sinh vật nhỏ bé màu hồng, sống ở vùng đất huyền bí Kiboria, nơi ánh sáng và màu sắc hòa quyện tạo nên một khung cảnh rực rỡ đầy mê hoặc. Kiboria vốn là chốn bình yên, nơi mọi sinh vật đều tận hưởng vẻ đẹp của thiên nhiên. Nhưng một ngày nọ, trong lúc khám phá một hang động sâu thẳm, Pink Nhon vô tình chạm vào một viên đá phát sáng. Ngay khoảnh khắc ấy, nó bị cuốn vào một thế giới song song, nơi Kiboria trở nên tối tăm và méo mó. Cây cối ở đây khô héo, bầu trời bị bao phủ bởi bóng tối, còn những người bạn từng tràn đầy sức sống giờ hóa thành tượng đá lạnh lẽo. Dù bối rối trước sự thay đổi đột ngột, Pink Nhon cảm nhận được một tia quyết tâm bùng lên trong lòng.

Không để nỗi sợ chế ngự, Pink Nhon sớm phát hiện ra một khả năng đặc biệt: nó có thể hấp thụ những Mảnh Năng Lượng rải rác, tàn dư của trái tim Kiboria, Nguồn Năng Lượng Thuần Khiết. Với ý chí khôi phục ánh sáng cho

vùng đất bị lãng quên này, Pink Nhon bắt đầu cuộc hành trình đầy hiểm nguy qua ba khu vực khắc nghiệt. Mỗi nơi đều được canh giữ bởi một kẻ thù đáng gờm: Tử thần, kẻ chủ mưu đứng sau sự suy tàn của Kiboria, triệu hồi. Cái Chết đã đánh cắp Nguồn Năng Lượng Thuần Khiết, âm mưu hợp nhất hai thế giới thành một vương quốc bóng tối vĩnh cửu.

Cuộc hành trình dẫn Pink Nhon đến Rừng Tĩnh Lặng, nơi không một âm thanh nào vang lên, ngay cả tiếng lá xào xạc cũng biến mất, và bóng tối len lỏi qua từng tán cây như một thực thể sống. Tại đây, nó đối mặt với Con Dơi, một sinh vật nhanh nhẹn và xảo quyệt ẩn mình trong bóng tối, bất ngờ tung ra những đòn tấn công. Sau một trận chiến căng thẳng, Pink Nhon vượt qua những cú đánh úp của Con Dơi và hạ gục nó, giành lấy Mạnh Năng Lượng đầu tiên. Chiến thắng này mở khóa một sức mạnh mới: khả năng bắn ra những luồng năng lượng, giúp Pink Nhon thêm tự tin đối diện với thử thách phía trước.

Rời khỏi sự im lặng ngột ngạt của khu rừng, Pink Nhon tiến vào Hang Đá Rỉ Sét, một mê cung ngầm với những đường hầm quanh co, nơi không khí nồng nặc mùi rỉ sét và tiếng vọng của cỗ máy xưa cũ vang lên từ xa. Sâu trong hang động, Pháp Sư Tử Thi đang chờ đợi, kẻ điều khiển một đội quân hồn ma đáng sợ. Không chút nao núng, Pink Nhon dũng cảm tiến lên, dùng sức mạnh ngày càng lớn để xua tan đám tay sai của Pháp Sư và cuối cùng tiêu diệt tên phù thủy bóng tối. Phần thưởng là Mạnh Năng Lượng thứ hai, ban cho nó khả năng phá vỡ những chướng ngại cản đường.

Điểm đến cuối cùng là Tháp Ánh Sáng Phản Chiếu, một công trình bí ẩn nơi gương soi bề cong thực tại, và những ảo ảnh nhảy múa trong ánh sáng lập lòe. Pink Nhon khéo léo điều khiển ánh sáng để mở ra những lối đi ẩn, từng bước tiến gần hơn đến trận chiến cuối cùng. Trên đỉnh tháp, nó đối đầu với cái Chết, một thực thể sở hữu sức mạnh bóng tối áp đảo. Cuộc chiến diễn ra khốc liệt, thử thách mọi dũng khí và kỹ năng của Pink Nhon. Nhưng bằng sự kiên trì, Pink Nhon chiến thắng, khôi phục Nguồn Năng Lượng Thuần Khiết và xua tan bóng tối khỏi Kiboria.



Khi vùng đất được tái sinh trong ánh sáng và màu sắc, Pink Nhon được ca ngợi như một người hùng. Tuy nhiên, khi tiếng reo hò dần lắng xuống, một tiếng thì thầm yếu ớt vang lên từ sâu trong bóng tối, gợi ý về những mối đe dọa mới đang rình rập. Dù Kiboria đã an toàn, cuộc hành trình của Pink Nhon vẫn chưa thực sự khép lại.

### 2.2.2. Luật chơi

Game cho phép người chơi tự do di chuyển theo không gian game quy ước.

Khi tấn công hoặc sử dụng kỹ năng, người chơi sẽ mất năng lượng.

Người chơi có thể gặp tương tác có lợi hoặc bất lợi của các vật phẩm trong suốt quá trình khám phá như rương báu hay bẫy gai...

#### **Điều kiện thắng:**

- Người chơi thu thập đủ số “sao trời” mỗi game yêu cầu thông qua việc nhặt rương và đánh quái để qua màn chơi tiếp theo.
- Người chơi hoàn thiện toàn bộ 3 màn chơi sẽ hoàn thành game.

#### **Điều kiện thua:**

- Số lượng máu của người chơi bằng 0.

### 2.2.3. Tương tác và điều khiển game

“Tiny Fight” được xây dựng với một hệ thống điều khiển quen thuộc với những người chơi đã quen với thể loại phiêu lưu và cũng dễ tiếp cận ngay cả với những người mới:

- Di chuyển: Sử dụng các phím mũi tên hoặc phím A (trái), D (phải), W (lên) và S (xuống) để di chuyển nhân vật tương ứng sang trái, sang phải, leo lên thang và leo xuống thang. Sử dụng phím Space để nhảy (tối đa 02 nhịp một lần) để di chuyển nhân qua các môi trường.
- Tấn công: Sử dụng chuột trái để tấn công bằng súng.

- Kỹ năng: Ấn phím Shift trái sẽ giúp nhân vật của bạn lướt đi một khoảng ngắn. Lướt có thể giúp bạn thoát khỏi các tính huống nguy hiểm nhanh chóng như né đạn hay qua phần địa hình nguy hiểm.

## 2.3. Thiết kế các phần tử của game

### 2.3.1. Người chơi (Player)

Player: Nhân vật Pink Nhon mà người chơi cần điều khiển



- Máu tối đa: 100
- Năng lượng tối đa: 100
- Tốc độ hồi năng lượng: 10 năng lượng/s









Hình 2.1. Nhân vật chính Pink Nhon

Hành động của nhân vật:

Bảng 2.1. Bảng mô tả nhân vật chính




Hành động	Mô tả	Điều kiện	Hình ảnh
Di chuyển	Di chuyển với tốc độ 5 đơn vị/s	Ấn phím mũi tên hoặc A, D	 <p>Hình 2.2. Nhân vật di chuyển</p>
Nhảy	Nhảy 10 đơn vị (tối đa 2 lần) theo chiều dọc	Ấn phím Space	

			<i>Hình 2.3. Nhân vật nhảy</i>
Leo trèo	Leo thang với tốc độ 3 đơn vị/s	Ấn phím W (lên) hoặc S (xuống)	 <i>Hình 2.4. Nhân vật chạy</i>
Lướt	Lướt một khoảng 15 đơn vị/0.2s, tiêu tốn 10 năng lượng và có 1s hồi chiêu	Ấn phím Shift trái	 <i>Hình 2.5. Nhân vật lướt</i>
Bắn	Bắn một viên đạn bay tối đa 30 đơn vị với tốc độ 10 đơn vị/s, tiêu tốn 10 năng lượng, gây 10 sát thương và gây đẩy lùi đối với quái nhỏ	Sử dụng chuột trái	 <i>Hình 2.6. Nhân vật bắn đạn</i>
Nhận sát thương	Mất đi 10 máu và không thể bị nhận sát thương trong 0.25 giây kế tiếp	Bị đánh sát thương từ quái hoặc bẫy	 <i>Hình 2.7. Nhân vật nhận sát thương</i>
Chết	Ngã xuống và tan biến, màn chơi kết thúc	Lượng máu chạm mức 0	  <i>Hình 2.8. Nhân vật chết</i>

### 2.3.2. Quái thường (enemy) và quái khủng (boss)


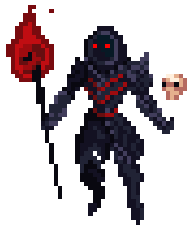
Enemy: Quái thường có lượng máu không quá lớn và có 2 loại là quái đánh gần và quái đánh xa, bị tiêu diệt khi người chơi đánh bại


*Hình 2.9. Bảng mô tả hệ thống quái thường*

Đối tượng	Đặc điểm	Mô tả	Hình ảnh
Elite Wolf	Quái di chuyển tầm trung, chỉ có thể di chuyển trái phải, đánh cận chiến tốc độ tầm trung	Máu: 100 Tốc độ: 3 đơn vị/s Sát thương: 10	 <i>Hình 2.10. Elite Wolf</i>
Newt Slinger	Quái di chuyển chậm, chỉ có thể di chuyển trái phải, có thể phát hiện người chơi trong tầm, đánh xa tốc độ tầm trung	Máu: 150 Tốc độ: 1.5 đơn vị/s Sát thương: 10	 <i>Hình 2.11. Newt Slinger</i>
Bat	Quái bay nhanh, di chuyển theo quỹ đạo, có thể tấn công người chơi từ xa	Máu: 175 Tốc độ: 2 đơn vị/s Sát thương: 10	 <i>Hình 2.12. Bat</i>

Boss: Quái có lượng máu và kích cỡ khủng, là đối thủ lớn nhất của người chơi ở cuối mỗi màn chơi

*Bảng 2.2. Bảng mô tả hệ thống quái khủng*


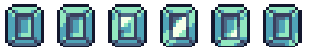

Đối tượng	Đặc điểm	Mô tả	Hình ảnh
Monster Bat	Quái khủng ở cuối ván chơi 1, có thể bay, né đạn hoặc đi chuyển theo người chơi. Bắn đạn theo chùm đường thẳng hoặc hình vòng tròn	Máu: 100 Tốc độ bay: 2 đơn vị/s Tốc độ đạn: 5 đơn vị/s Số lượng đạn/lần: 5 đối với đường thẳng và 12 đối với vòng tròn Sát thương: 10/mỗi viên	 <p><i>Hình 2.13. Elite Wolf</i></p>
Necromancer	Quái khủng ở cuối ván chơi 2, có thể bay, né đạn và di chuyển theo người chơi, bắn đạn chùm đường thẳng hoặc vòng tròn	Máu: 300 Tốc độ bay: 2 đơn vị/s Tốc độ đạn: 5 đơn vị/s	 <p><i>Hình 2.14. Necromancer</i></p>

Undead	Quái khủng ở cuối ván chơi 2, có thể bay, né đạn và di chuyển theo người chơi, bắn đạn chùm đường thẳng hoặc vòng tròn	Máu: 400 Tốc độ bay: 2 đơn vị/s Tốc độ đạn: 5 đơn vị/s	 <p><i>Hình 2.15. Undead</i></p>
--------	--	--	---

### 2.3.3. Đối tượng có thể tương tác (Interactables)

Interactables là các đối tượng trong môi trường game mà người chơi có thể tương tác trực tiếp để đạt được một mục đích cụ thể.

*Bảng 2.3. Bảng mô tả các đối tượng có thể tương tác*

Đối tượng	Đặc điểm	Mô tả	Hình ảnh
Rương báu	Chiếc rương chứa vật phẩm được đặt trên đường khám phá	Roi ra vật phẩm ngẫu nhiên có lợi cho người chơi	 <p><i>Hình 2.16. Rương báu</i></p>
Kinh nghiệm	Điểm kinh nghiệm nhận được khi mở rương báu	Vật phẩm giúp người chơi nâng cấp cho nhận vật của mình	 <p><i>Hình 2.17. Kinh nghiệm</i></p>
Bình máu	Bình nhận được ngẫu nhiên khi mở rương báu	Vật phẩm giúp hồi cho người chơi 10 HP	 <p><i>Hình 2.18. Bình máu</i></p>

Bình năng lượng	Bình nhật được ngẫu nhiên khi mở rương báu	Vật phẩm giúp tăng tốc độ hồi năng lượng và tốc độ chạy trong một khoảng thời gian	 <p><i>Hình 2.19. Bình năng lượng</i></p>
Sao trời	Vật phẩm hình dạng ngôi sao	Thu thập khi nhật rương hoặc tiêu diệt quái, thu thập đủ để đạt điều kiện qua màn	 <p><i>Hình 2.20. Sao trời</i></p>
Bẫy gai	Cạm bẫy yêu cầu người chơi phải né tránh	Trừ 10 máu của người chơi nếu đạp trúng	 <p><i>Hình 2.21. Bẫy gai</i></p>
Thang	Có hình dạng cái thang	Người chơi có thể leo trèo để vượt qua địa hình	 <p><i>Hình 2.22. Thang</i></p>
Bom	Phát nổ khi người chơi chạm vào	Người chơi sẽ mất 10 máu nếu bị bom nổ	 <p><i>Hình 2.23. Bom</i></p>

#### 2.4. Thiết kế màn chơi

Game sẽ có tổng cộng 3 màn chơi, mỗi màn chơi sau sẽ khó hơn các màn chơi trước và yêu cầu phải hoàn thành hết các màn chơi trước thì mới có thể tiếp tục. Độ khó của từng màn chơi như sau:

- Màn chơi 1: Yêu cầu người chơi học cách di chuyển, leo trèo, vượt bẫy và đánh quái tầm gần. Các vật phẩm hỗ trợ xuất hiện nhiều
- Màn chơi 2: Thêm quái bắn xa, yêu cầu người chơi học cách né tránh.
- Màn chơi 3: Xuất hiện quái bay và bắn xa yêu cầu người chơi phải biết di chuyển kết hợp né đạn và tấn công.

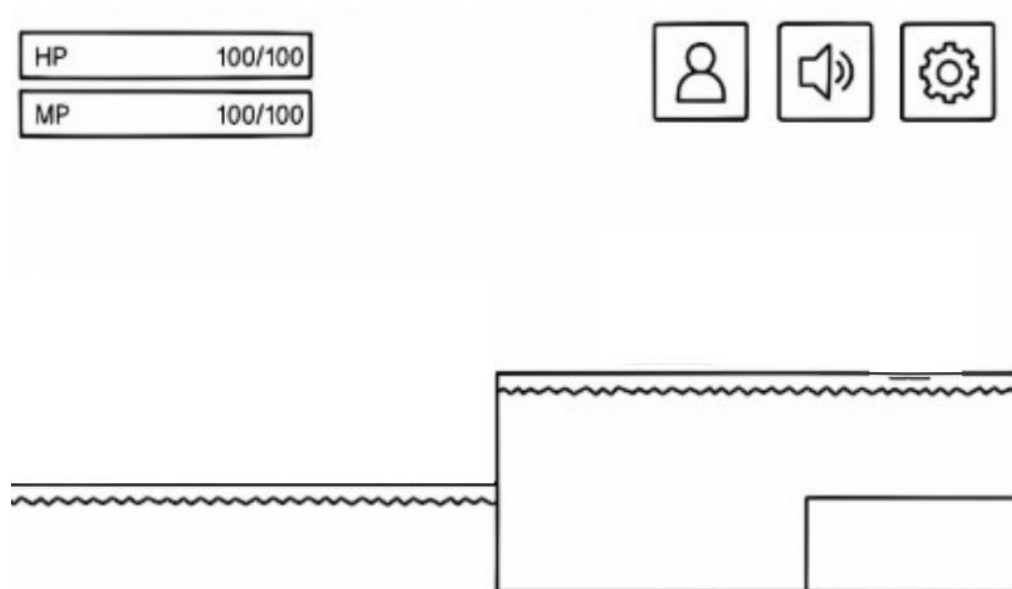
## 2.5. Thiết kế giao diện

### 2.5.1. Menu khởi đầu



Hình 2.24. Hình dung màn hình menu chính

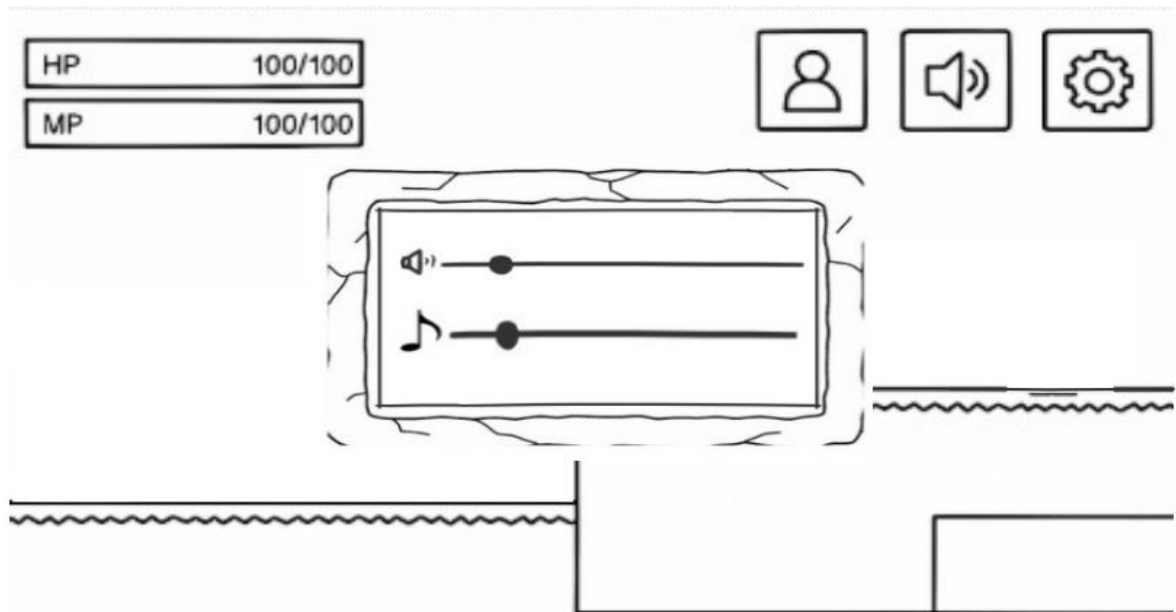
### 2.5.2. UI trong game



Hình 2.25. Hình dung màn hình giao diện trong game

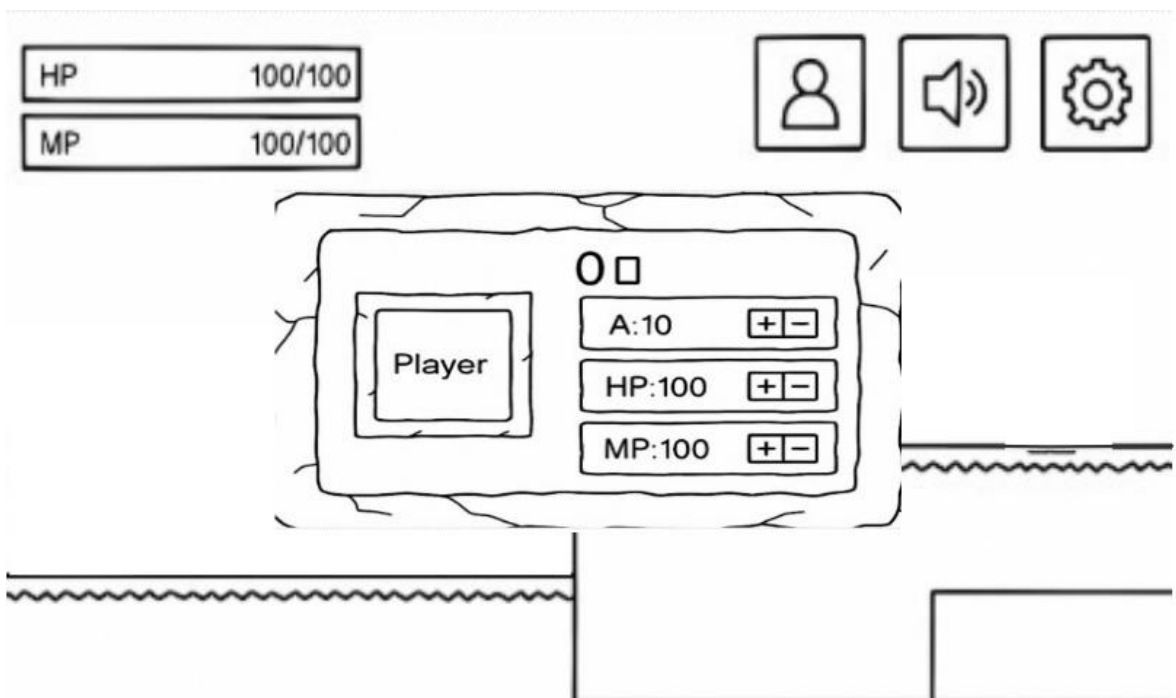


### 2.5.3. Màn hình tăng giảm âm lượng



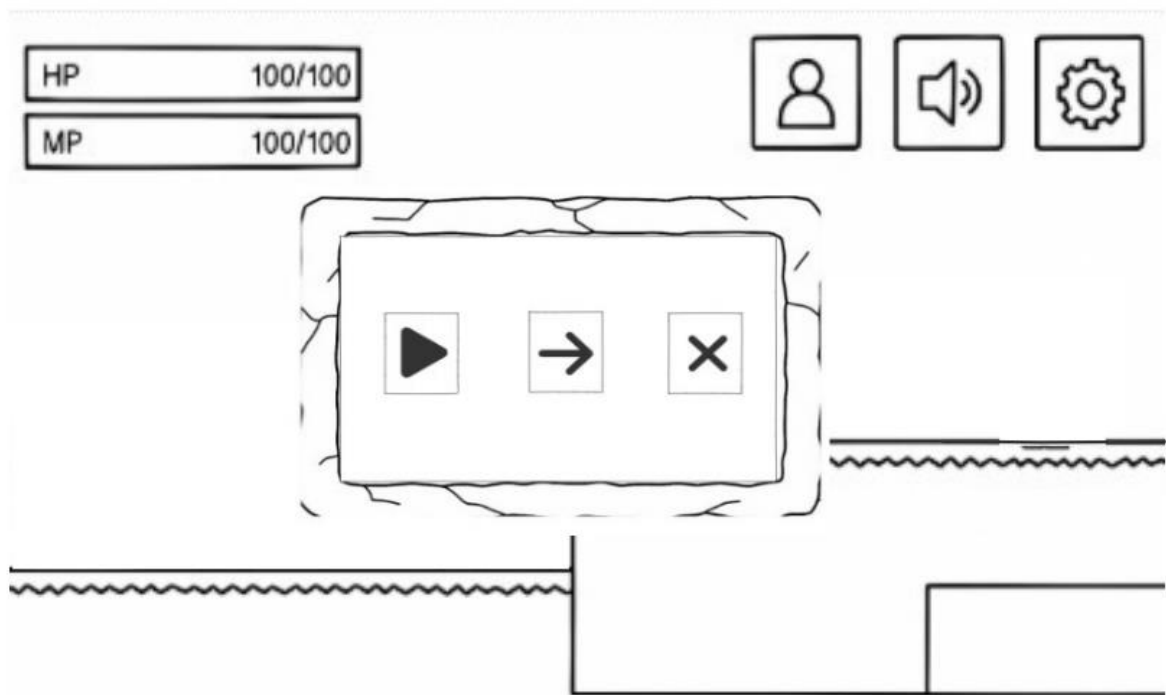
Hình 2.26. Hình dung màn hình tăng giảm âm lượng

### 2.5.4. Màn hình nâng cấp nhân vật



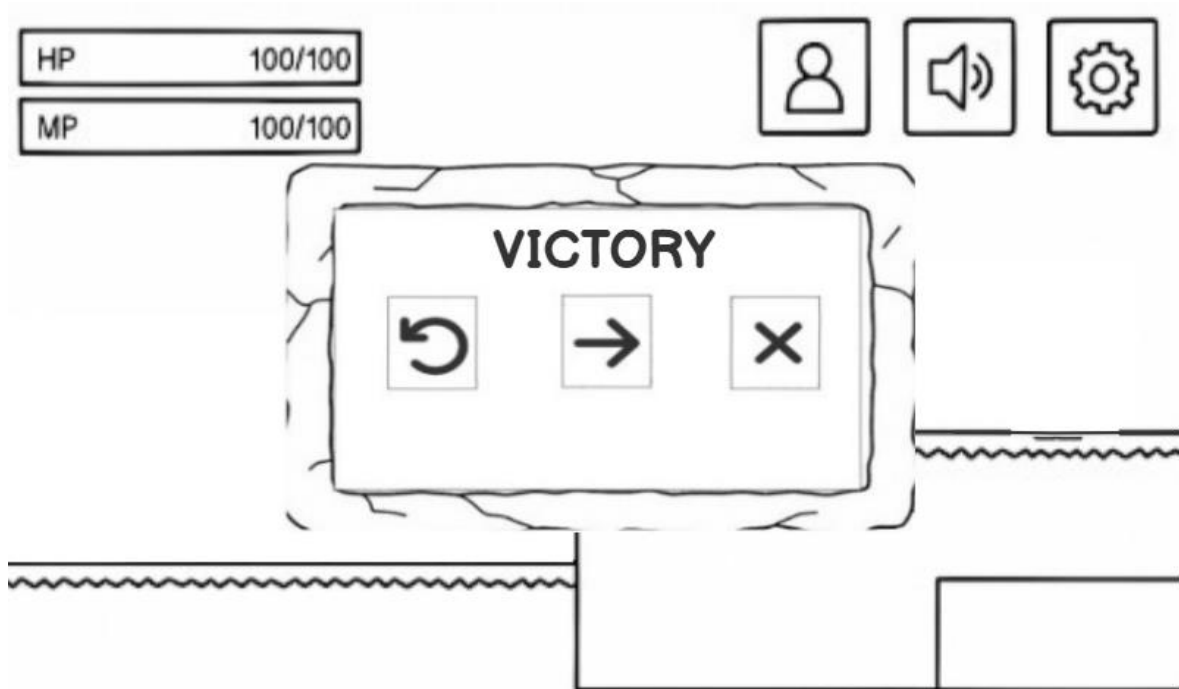
Hình 2.27. Hình dung màn hình nâng cấp nhân vật

### 2.5.5. Màn hình dừng game



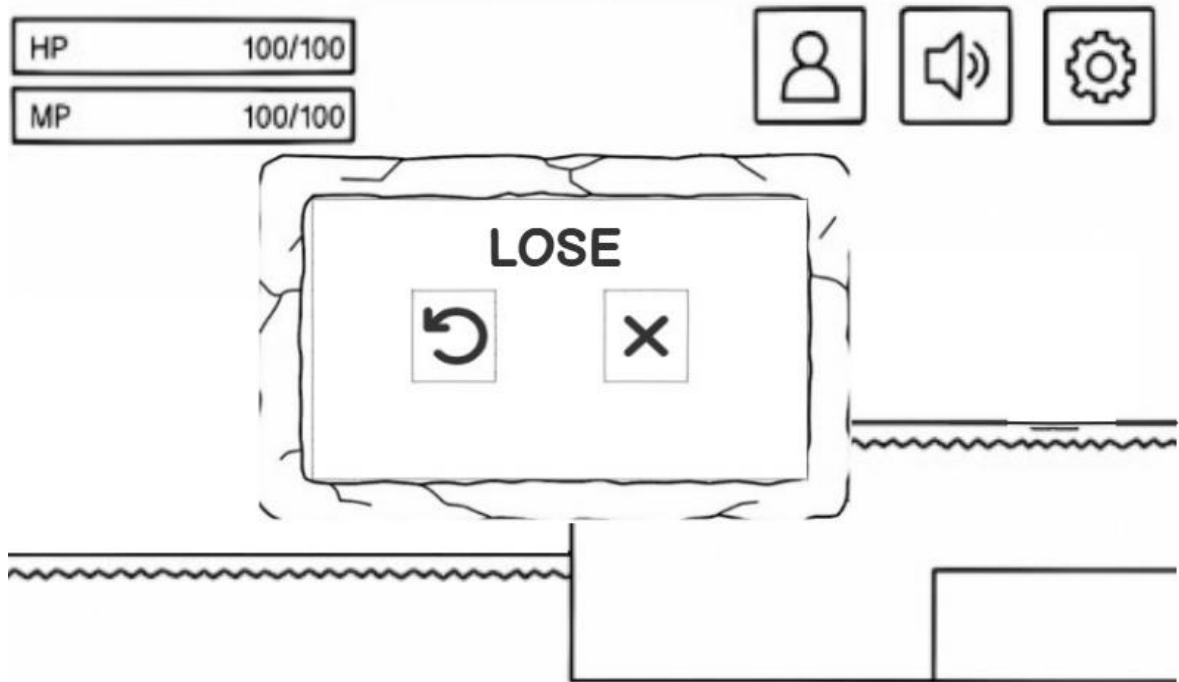
Hình 2.28. Hình dung màn hình dừng game

### 2.5.6. Màn hình chiến thắng



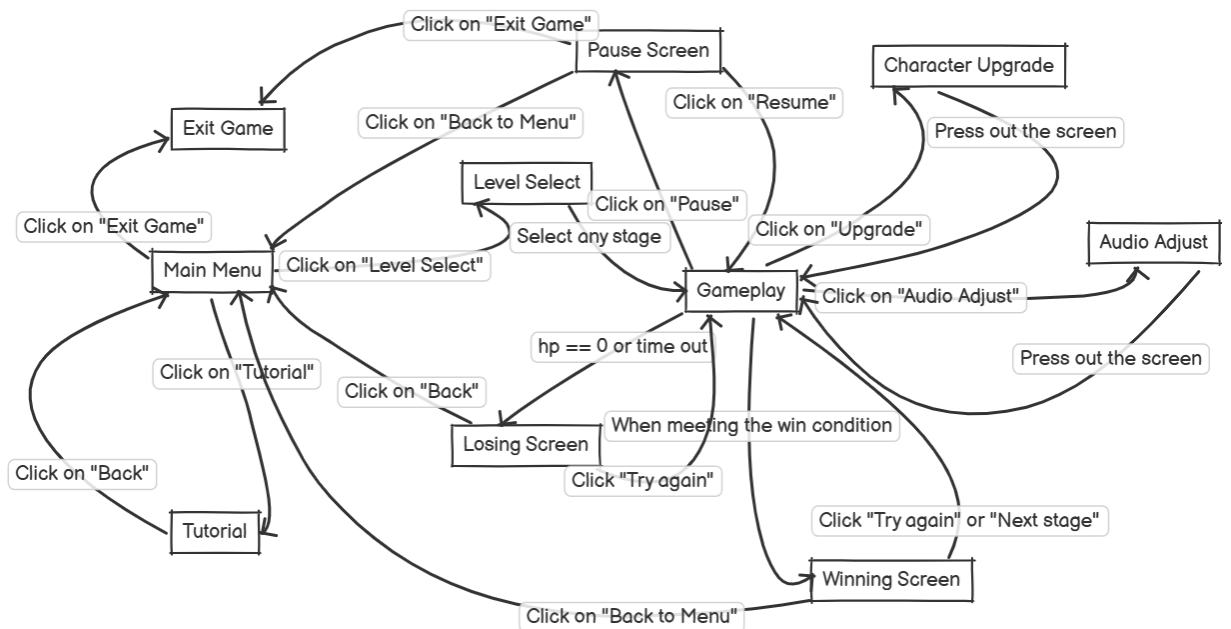
Hình 2.29. Hình dung màn hình chiến thắng

### 2.5.7. Màn hình thua trận



Hình 2.30. Hình dung màn hình thua trận

### 2.5.8. Story board



Hình 2.31. Story board của game

## 2.6. Tài nguyên

- Player : [Free Tiny Hero Sprites Pixel Art by Free Game Assets \(GUI, Sprite, Tilesets\)](#)
- Boss 1: [Necromancer \(Free\) by CreativeKind](#)
- Boss 2: [Boss: Undead Executioner \[FREE\] by Kronovi-](#)
- Boss 3: [Flying Demon 2D Pixel Art by Mattz Art](#)
- Enemy 1: [Critters - Enemy Pack 3 by Foozle](#)
- Enemy 2: [Realm of Critters - Enemy Pack 1 - Pixel Art by Foozle](#)
- Enemy 3: [Medieval Fantasy Character Pack by OcO](#)
- UI: [Stone UI | 2D Icons | Unity Asset Store](#)
- Background: [Free Pixel Cloud and Sky Backgrounds by Free Game Assets \(GUI, Sprite, Tilesets\)](#)
- Âm thanh: [Download Free Game Sound Effects - Royalty-Free Video Game Sounds | Sound Library - Pixabay](#)
- Tile: [Pixel Art Woods Tileset and Background | 2D Environments | Unity Asset Store](#)

## CHƯƠNG 3: CÁC KỸ THUẬT XÂY DỰNG GAME VÀ KẾT QUẢ THỰC NGHIỆM

### 3.1. Các kỹ thuật xây dựng game

#### 3.1.1. Các kỹ thuật đối với Player

##### 3.1.1.1. Điều khiển di chuyển cơ bản

Nhân vật di chuyển trên mặt đất theo trục X dựa trên input từ người chơi và tốc độ di chuyển được xác định bởi trạng thái hiện tại (đang đi bộ, đang nhảy, hoặc đang lướt).

```
rb.velocity = new Vector2(moveInput.x * CurrentMoveSpeed, rb.velocity.y);
```

*Hình 3.1. Di chuyển nhân vật*

```
public float CurrentMoveSpeed
{
    get {
        if (CanMove)
        {
            if (IsMoving && !touchingDirections.IsOnWall)
            {
                if (touchingDirections.IsGrounded)
                {
                    if (isDashing)
                    {
                        return dashSpeed;
                    }
                    else
                    {
                        return walkSpeed;
                    }
                }
                else
                {
                    return airWalkSpeed;
                }
            }
            else
            {
                return 0;
            }
        }
        else
        {
            return 0;
        }
    }
}
```

*Hình 3.2. Tốc độ di chuyển thay đổi theo trạng thái*

### 3.1.1.2. Nhảy

Nhật vật có thể nhảy tối đa maxJumps (hiện để là 2) lần.

```
public void OnJump(InputAction.CallbackContext context)
{
    if (context.started && CanMove && (touchingDirections.IsGrounded || jumpCount < maxJumps))
    {
        AudioManager.instance.PlaySfx(1);
        animator.SetTrigger(AnimationStrings.jumpTrigger);
        rb.velocity = new Vector2(rb.velocity.x, jumpImpulse);
        jumpCount++;
    }
}
```

*Hình 3.3. Nhảy với giới hạn số lần*

```
if (touchingDirections.IsGrounded)
{
    jumpCount = 0;
}
```

*Hình 3.4. Đặt lại số lần nhảy khi chạm đất*

### 3.1.1.3. Lướt

Nhân vật có thể lướt trong một khoảng thời gian ngắn, sau đó phải chờ hồi chiêu.

```
private IEnumerator Dash()
{
    canDash = false;
    isDashing = true;
    animator.SetBool(AnimationStrings.isDashing, true);

    float originalGravity = rb.gravityScale;
    rb.gravityScale = 0;
    rb.velocity = new Vector2(IsFacingRight ? dashSpeed : -dashSpeed, 0);

    yield return new WaitForSeconds(dashTime);

    rb.gravityScale = originalGravity;
    isDashing = false;
    animator.SetBool(AnimationStrings.isDashing, false);

    yield return new WaitForSeconds(dashCooldown);
    canDash = true;
}
```

*Hình 3.5. Kỹ thuật lướt với thời gian hồi chiêu*

### 3.1.1.4. Trèo thang

Nhân vật có thể trèo thang khi chạm vào thang và nhấn phím điều khiển W (lên) hoặc S (xuống). Khi trèo thang, trọng lực bị vô hiệu hóa và nhân vật di chuyển theo trục Y.

```
public void OnClimb(InputAction.CallbackContext context)
{
    if (context.started && currentLadder != null)
    {
        // Căn chỉnh nhân vật vào giữa thang
        transform.position = new Vector2(currentLadder.transform.position.x, transform.position.y);

        SetClimbingState(true);
    }
    else if (context.canceled && isClimbing)
    {
        // Ngừng trèo
        if (!IsOnLadder())
        {
            SetClimbingState(false);
        }
    }
}
```

*Hình 3.6. Phát hiện và xử lý trèo thang*

```
if (isClimbing)
{
    if (moveInput.y != 0)
    {
        rb.velocity = new Vector2(0, moveInput.y * climbSpeed);
    }
    else
    {
        rb.velocity = Vector2.zero;
    }

    rb.gravityScale = 0;

    Physics2D.IgnoreLayerCollision(gameObject.layer, LayerMask.NameToLayer("Ground"), true);

    animator.SetBool(AnimationStrings.isClimbing, true);

    if (currentLadder == null || (!IsOnLadder() && moveInput.y == 0))
    {
        SetClimbingState(false);
    }
}
```

*Hình 3.7. Trạng thái khi trèo thang*

### 3.1.1.5. Tấn công tầm xa

Nhân vật có thể tấn công tầm xa nếu đủ năng lượng (mana).

```
public void OnRangedAttack(InputAction.CallbackContext context)
{
    if (context.started && IsAlive && manaBar != null)
    {
        if (Ui_Main.isGamePaused) return;

        if (EventSystem.current != null && EventSystem.current.IsPointerOverGameObject())
        {
            Debug.Log("Click vào UI nên không bắn");
            return;
        }

        if (manaBar.currentMana >= manaCostPerShot)
        {
            AudioManager.instance.PlaySfx(4);
            manaBar.UseMana(manaCostPerShot);
            animator.SetTrigger(AnimationStrings.rangedAttackTrigger);
        }
        else
        {
            Debug.Log("Không đủ mana để bắn!");
        }
    }
}
```

Hình 3.8. Tấn công tiêu tốn mana

### 3.1.1.6. Tăng tốc tạm thời

Tốc độ di chuyển, tốc độ nhảy và tốc độ lướt tăng lên trong một khoảng thời gian.

```
public void ActivateSpeedBoost(float duration, float multiplier)
{
    StartCoroutine(SpeedBoostCoroutine(duration, multiplier));
}

1 reference
private IEnumerator SpeedBoostCoroutine(float duration, float multiplier)
{
    walkSpeed *= multiplier;
    airWalkSpeed *= multiplier;
    dashSpeed *= multiplier;

    Debug.Log("Speed Buff Activated!");

    yield return new WaitForSeconds(duration);

    walkSpeed /= multiplier;
    airWalkSpeed /= multiplier;
    dashSpeed /= multiplier;

    Debug.Log("Speed Buff Expired.");
}
```

Hình 3.9. Kỹ thuật tăng tốc trong thời gian ngắn



### 3.1.1.7. Xử lý khi nhận sát thương

Phát hiện va chạm với OnTriggerEnter2D để phát hiện va chạm giữa các đối tượng có Collider và Rigidbody. Nhân vật hoặc quái bị đẩy lùi khi nhận sát thương và kiểm tra trạng thái máu.

```
void OnTriggerEnter2D(Collider2D collider)
{
    if (collider.CompareTag("Player")) //
    {
        Damageable playerDamageable = collider.GetComponent<Damageable>();
        if (playerDamageable != null)
        {
            Vector2 knockback = knockbackForce;
            if (GetComponent<Rigidbody2D>().velocity.x < 0)
                knockback = new Vector2(-knockbackForce.x, knockbackForce.y);
            playerDamageable.Hit(damage, knockback);
        }
        Destroy(gameObject);
    }
}
```

Hình 3.10. Phát hiện đạn va chạm với người chơi

```
public void OnHit(int damage, Vector2 knockback)
{
    rb.velocity = new Vector2(knockback.x, rb.velocity.y + knockback.y);

    if (damageable.Health ≤ 0) //
    {
        GameManager.instance.ShowDeathUI();
    }
}
```

Hình 3.11. Nhận sát thương và bị đẩy lùi

### 3.1.1.8. Hệ thống hồi sinh (spawn)

Nhân vật được đặt tại vị trí spawn khi bắt đầu trò chơi.

```
void SetToSpawnPoint()
{
    GameObject spawn = GameObject.Find("PlayerSpawnPoint");
    if (spawn != null)
    {
        transform.position = spawn.transform.position;
    }
    else
    {
        Debug.LogWarning("Không tìm thấy PlayerSpawnPoint trong scene!");
    }
}
```

*Hình 3.12. Đặt nhân vật tại điểm spawn*

### 3.1.1.9. Xử lý sự kiện chết (áp dụng cho cả quái)

Kích hoạt sự kiện chết khi máu xuống 0 (Sử dụng cho tất cả các prefabs nhân vật/quái).

```
private void OnHealthChanged(int currentHealth, int maxHealth)
{
    if (currentHealth ≤ 0 && gameObject != null)
    {
        animator.SetBool(AnimationStrings.isAlive, false);
    }
}
```

*Hình 3.13. Kiểm tra thể trạng khi máu xuống dưới 0*

## 3.1.2. Các kĩ thuật đối với quái nhỏ

### 3.1.2.1. Các hành vi của quái nhỏ

Quái di chuyển theo hướng xác định và đổi hướng khi gặp tường hoặc vách đá. Một số quái có cơ chế phát hiện người chơi trong tầm bằng cách sử dụng khoảng cách hoặc vùng phát hiện.

```
private void FlipDirection()
{
    WalkDirection = (WalkDirection == WalkableDirection.Right) ? WalkableDirection.Left : WalkableDirection.Right;
}
```

*Hình 3.14. Di chuyển và đổi hướng của quái nhỏ*

```
private bool IsPlayerInRange()
{
    if (player == null) return false;

    float horizontalDistance = Mathf.Abs(transform.position.x - player.position.x);

    float verticalDistance = Mathf.Abs(transform.position.y - player.position.y);
    float verticalTolerance = 2f;

    return horizontalDistance ≤ detectionRange && verticalDistance ≤ verticalTolerance;
}
```

Hình 3.15. Phát hiện người chơi trong tầm thông qua khoảng cách

### 3.1.2.2. Tấn công của quái nhỏ

Quái tấn công người chơi khi phát hiện được người chơi trong tầm và thực hiện hoạt ảnh tấn công (bắn đạn hoặc chém).

```
private void TriggerAttack()
{
    animator.SetTrigger("attack");
    StartCoroutine(PerformShoot());
}

1 reference
private IEnumerator PerformShoot()
{
    yield return new WaitForSeconds(0.01f);

    if (projectilePrefab ≠ null && shootPoint ≠ null)
    {
        GameObject bullet = Instantiate(projectilePrefab, shootPoint.position, Quaternion.identity);

        Rigidbody2D bulletRb = bullet.GetComponent<Rigidbody2D>();
        if (bulletRb ≠ null)
        {
            bulletRb.velocity = walkDirectionVector * 10f;
            bulletRb.collisionDetectionMode = CollisionDetectionMode2D.Continuous;
        }
    }
}
```

Hình 3.16. Thực hiện tấn công của quái đánh xa

```
void Update()
{
    HasTarget = attackZone.detectedColliders.Count > 0;

    if(AttackCooldown > 0)
    {
        AttackCooldown -= Time.deltaTime;
    }
}
```

Hình 3.17. Thực hiện tấn công của quái đánh gần

### 3.1.3. Các kỹ thuật đối với quái khủng

#### 3.1.3.1. Di chuyển ngẫu nhiên trong vùng giới hạn

Boss di chuyển trong một vùng giới hạn được xác định bởi một bán kính

```
private void SetRandomTargetPosition()
{
    Vector2 randomOffset = Random.insideUnitCircle * bossZoneRadius;
    randomTargetPosition = initialPosition + randomOffset;
}
```

Hình 3.18. Chọn ngẫu nhiên vị trí mục tiêu

```
private void DynamicMovement()
{
    if (randomMoveTimer ≥ randomMoveInterval)
    {
        SetRandomTargetPosition();
        randomMoveTimer = 0f;
    }

    Vector2 directionToTarget = (randomTargetPosition - (Vector2)transform.position).normalized;
    rb.velocity = directionToTarget * flightSpeed;

    // Nếu đã đến gần vị trí mục tiêu, chọn vị trí mới
    if (Vector2.Distance(transform.position, randomTargetPosition) ≤ waypointReachedDistance)
    {
        SetRandomTargetPosition();
    }
}
```

Hình 3.19. Boss di chuyển ngẫu nhiên trong vùng giới hạn

#### 3.1.3.2. Đuổi theo người chơi trong tầm

Boss đuổi theo người chơi khi người chơi ở trong phạm vi, nếu người chơi quá gần boss chuyển sang di chuyển ngẫu nhiên để tránh.

```
private bool IsPlayerInChaseRange()
{
    if (player == null) return false;
    float distanceToPlayer = Vector2.Distance(transform.position, player.position);
    return distanceToPlayer ≤ chaseDistance;
}
```

Hình 3.20. Phát hiện người chơi trong tầm truy đuổi

```

private void ChasePlayer()
{
    if (player == null) return;

    float distanceToPlayer = Vector2.Distance(transform.position, player.position);
    if (distanceToPlayer > safeDistance)
    {
        Vector2 directionToPlayer = (player.position - transform.position).normalized;
        rb.velocity = directionToPlayer * flightSpeed;
    }
    else
    {
        DynamicMovement();
    }
}

```

Hình 3.21. Boss đuổi theo người chơi khi người chơi ở trong phạm vi

### 3.1.3.3. Thực hiện tấn công

Boss thực hiện tấn công lặp lại theo quy luật

```

private IEnumerator ShootingPattern()
{
    while (true)
    {
        animator.SetTrigger("Attack");

        if (isCircularShooting)
        {
            ShootCircular();
        }
        else
        {
            ShootStraight();
        }

        yield return new WaitForSeconds(shootInterval);
    }
}

```

Hình 3.22. Boss thực hiện tấn công

### 3.1.3.4. Phát hiện người chơi và né đạn

Boss phát hiện đạn gần bằng cách kiểm tra các collider trong phạm vi, nếu phát hiện đạn, boss sẽ tính toán hướng né và di chuyển theo hướng đó

```
private bool IsBulletNearby(out Vector2 evadeDirection)
{
    evadeDirection = Vector2.zero;
    Collider2D[] colliders = Physics2D.OverlapCircleAll(transform.position, evadeDistance);
    foreach (var collider in colliders)
    {
        if (collider.CompareTag("Projectile"))
        {
            Debug.Log("Bullet detected: " + collider.name);
            Rigidbody2D bulletRb = collider.GetComponent<Rigidbody2D>();
            if (bulletRb != null)
            {
                Vector2 bulletDirection = bulletRb.velocity.normalized;
                float randomAngle = Random.Range(-30f, 30f); // Thêm góc ngẫu nhiên từ -30 đến 30 độ
                evadeDirection = Quaternion.Euler(0, 0, randomAngle) * Vector2.Perpendicular(bulletDirection).normalized;
                return true;
            }
        }
    }
    Debug.Log("No bullets detected");
    return false;
}

1 reference
private void EvadeBullet(Vector2 evadeDirection)
{
    rb.velocity = evadeDirection * flightSpeed;
}
```

Hình 3.23. Boss phát hiện và né tránh đạn

### 3.1.3.5. Cơ chế rơi vật phẩm khi chết

```
private void DropItem()
{
    if (itemDropPrefab != null)
    {
        Instantiate(itemDropPrefab, transform.position, Quaternion.identity);
    }
}

1 reference
private IEnumerator DestroyAfterDeath()
{
    DropItem();

    yield return new WaitForSeconds(1f);
    Destroy(gameObject);
}
```

Hình 3.24. Boss chết và rơi vật phẩm

### 3.1.3.6. Kỹ thuật máy hữu hạn trạng thái (FSM)

FSM là một mô hình toán học, trí tuệ nhân tạo đơn giản được sử dụng để quản lý trạng thái của một đối tượng. Trong trường hợp của quái khủng, FSM được sử dụng để quản lý các trạng thái khác nhau của boss đã sử dụng bên trên bao gồm:

- Idle: Boss đứng yên hoặc di chuyển ngẫu nhiên trong tầm.
- Chase: Boss đuổi theo người chơi.
- Attack: Boss thực hiện các đòn tấn công.
- Evade: Boss né tránh đạn của người chơi.
- Death: Boss chết và rơi vật phẩm, biến mất.

```
private void FixedUpdate()
{
    if (damageable.IsAlive)
    {
        if (CanMove)
        {
            evadeTimer += Time.fixedDeltaTime;
            bulletDetectionTimer += Time.fixedDeltaTime;
            randomMoveTimer += Time.fixedDeltaTime;

            if (IsPlayerInBossZone() && IsPlayerInChaseRange() && chaseTimer < chaseDuration)
            {
                ChasePlayer();
                chaseTimer += Time.fixedDeltaTime;
            }
            else if (IsPlayerInBossZone() &&
                evadeTimer ≥ evadeCooldown &&
                bulletDetectionTimer ≥ bulletDetectionCooldown &&
                IsBulletNearby(out Vector2 evadeDirection))
            {
                evadeTimer = 0f;
                bulletDetectionTimer = 0f;
                EvadeBullet(evadeDirection);
            }
            else
            {
                chaseTimer = 0f;
                DynamicMovement();
            }

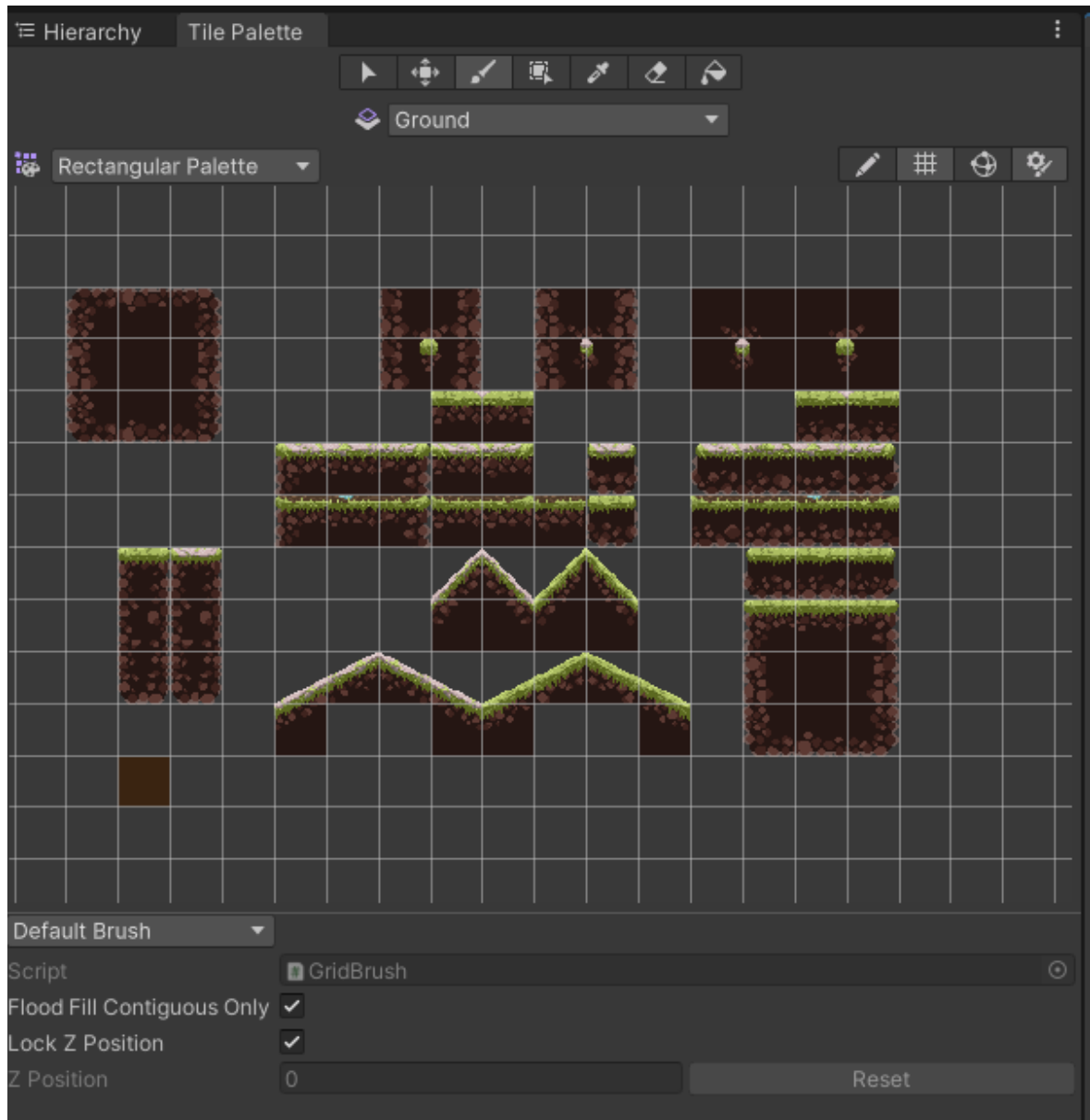
            FlipSpriteBasedOnPlayer();
        }
        else
        {
            rb.velocity = Vector3.zero;
        }
    }
    else
    {
        rb.gravityScale = 2f;
        rb.velocity = new Vector2(0, rb.velocity.y);

        if (!hasStartedDeath)
        {
            hasStartedDeath = true;
            StartCoroutine(DestroyAfterDeath());
        }
    }
}
```

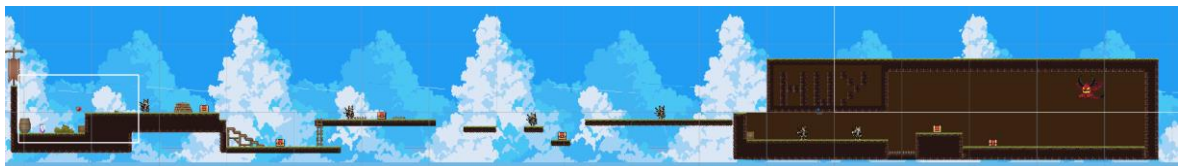
Hình 3.25. FSM chuyển đổi giữa các trạng thái dựa trên các điều kiện

### 3.1.4. Kỹ thuật thiết kế map

Sử dụng tile map



Hình 3.26. Tile map

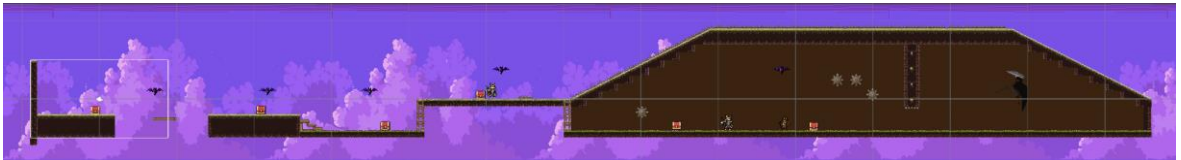


Hình 3.27. Map chơi 1





Hình 3.28. Map chơi 2



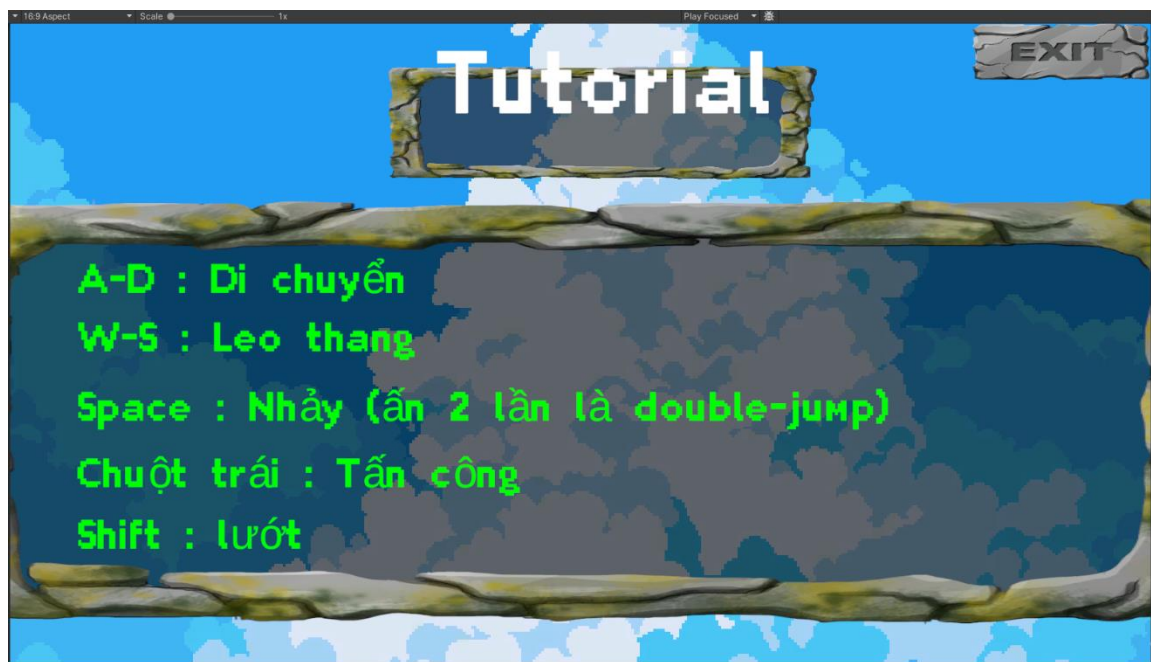
Hình 3.29. Map chơi 3

## 3.2. Kết quả thực nghiệm

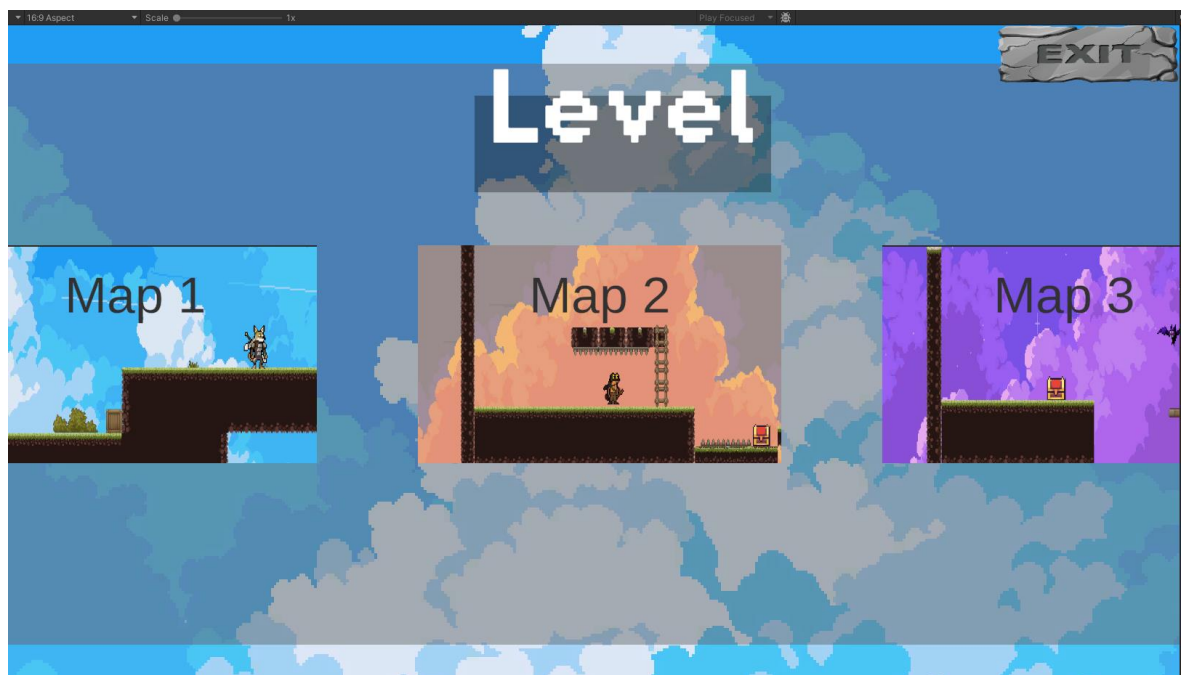
### 3.2.1. Main menu



Hình 3.30. Main menu

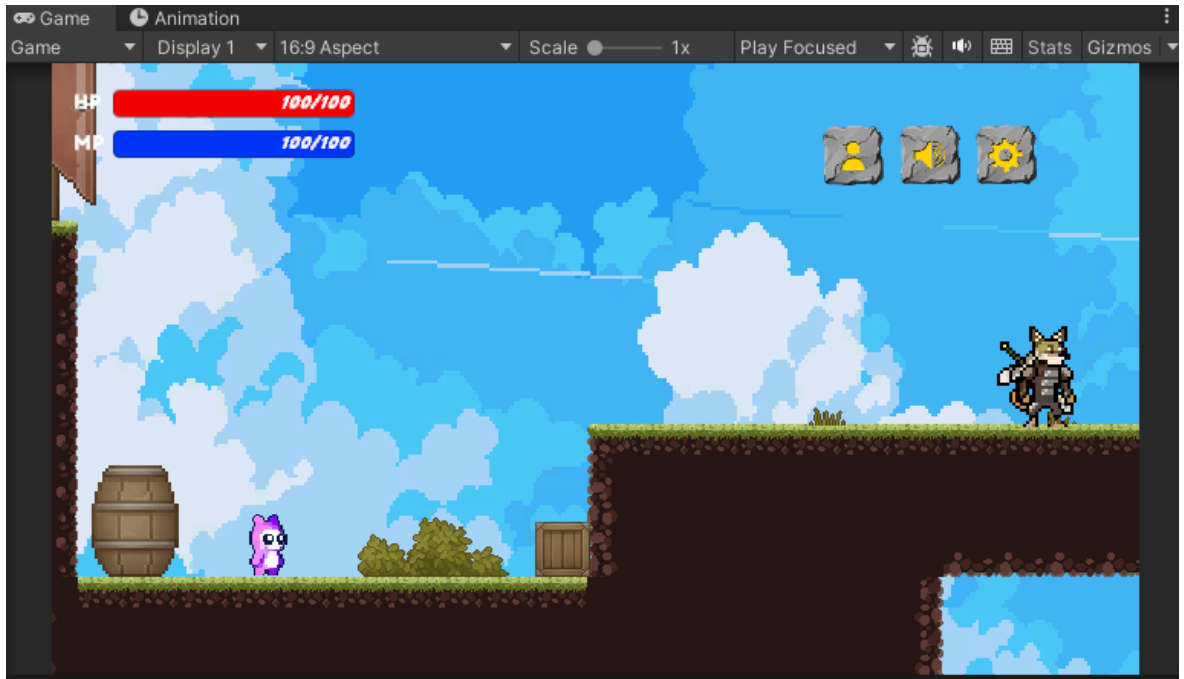


Hình 3.31. Tutorial

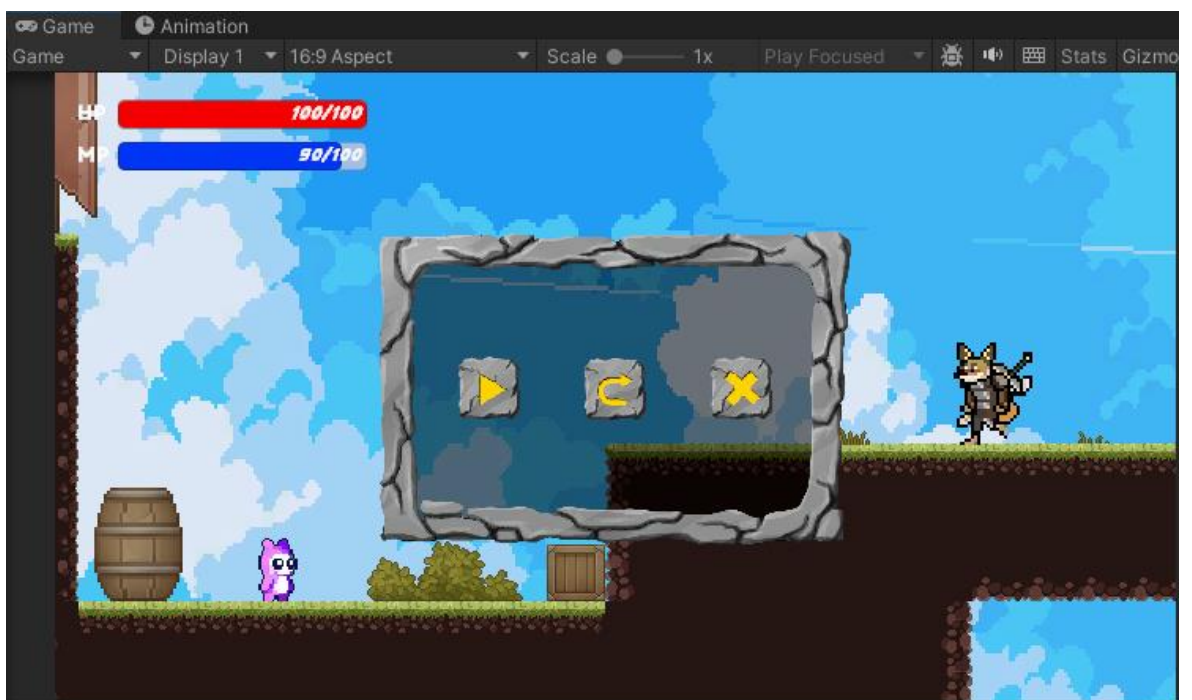


Hình 3.32. Chọn màn chơi

### 3.2.2. Màn hình game

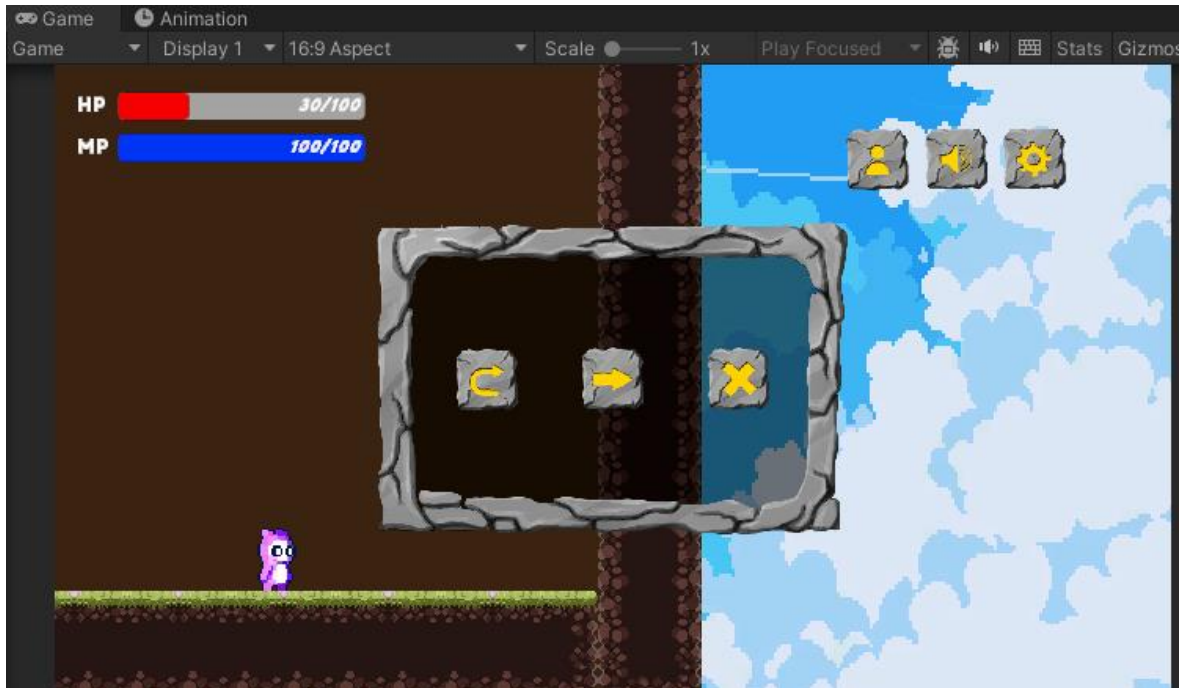


Hình 3.33. Màn hình trong game

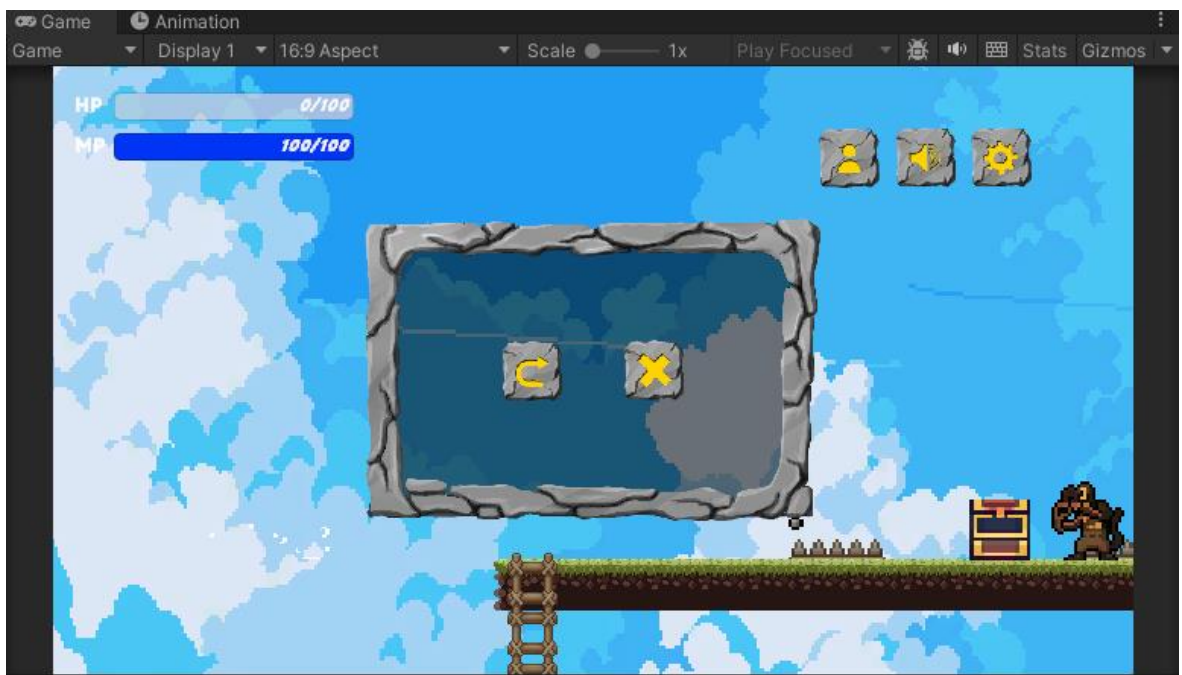


Hình 3.34. Màn hình dừng game

### 3.2.3. Màn hình chiến thắng và thua game



Hình 3.35. Màn hình chiến thắng



Hình 3.36. Màn hình thua trận

## KIỂM THỬ GAME

### 1. Phân tích yêu cầu kiểm thử

#### 1.1. Xác định phạm vi

**Chức năng chính:** di chuyển, nhảy, leo trèo, lướt, bắn súng, nhận sát thương, chết; AI quái thường và boss; tương tác với rương báu, bẫy gai, thang, bom.

**Phi chức năng:** Độ ổn định (không thoát đột ngột khi load scene), tính khả dụng của UI.

#### 1.2. Nguồn yêu cầu

Tài liệu thiết kế game (Chương 2) và định nghĩa luật chơi, thông số nhân vật, quái, môi trường.

#### 1.3. Mục tiêu kiểm thử

Đảm bảo mọi yêu cầu chức năng và phi chức năng đều được thỏa mãn trước khi build.

### 2. Lập kế hoạch kiểm thử

#### 2.1. Mục tiêu

Xác nhận đầy đủ các yêu cầu chức năng, không có lỗi nghiêm trọng (critical/blocker) còn tồn đọng.

#### 2.2. Phạm vi & Ngoại phạm vi

**Trong phạm vi:** tất cả test case chức năng đã thiết kế.

**Ngoại phạm vi:** kiểm thử mạng (multiplayer), localization, các tính năng chưa hoàn thiện (chọn nhiều nhân vật, map bổ sung).

#### 2.3. Tài nguyên

- Nhân lực: 2 thành viên kiểm thử thủ công (Hưng, Thái) và 1 thành viên hỗ trợ debug (Hiếu)
- Kiểm thử được thực hiện ngay sau khi một chức năng được hoàn thành



- Môi trường: Windows 10, Unity Editor 2022.3.33f1, bản build nội bộ.

### **3. Chiến lược kiểm thử**

#### **3.1. Các cấp độ**

- Kiểm thử đơn vị sử dụng Unity Test Framework (edit-mode).
- Kiểm thử tích hợp (tương tác giữa Player – Enemy – UI).
- Play-mode tests (tự động hóa kịch bản gameplay chính).
- Manual exploratory cho trải nghiệm, level design, cảm nhận game.

#### **3.2. Phương pháp**

- Functional Testing (Hộp đen): Kiểm thử các chức năng theo yêu cầu đặt ra (menu, điều kiện thắng, tương tác vật thể...).
- Exploratory Testing: Tester tự do tương tác để tìm lỗi không đoán trước được (ví dụ: kẹt vật lý, nhảy ra khỏi map).
- Regression Testing: Sau mỗi lần sửa lỗi hoặc thêm tính năng, thực hiện lại các test case cũ để kiểm tra có lỗi phát sinh.

#### **3.3. Kỹ thuật**

- Phân vùng tương đương: Phân chia các đầu vào thành nhóm hợp lệ/không hợp lệ như: loại buff nhận từ rương (HP, tốc độ, hồi năng lượng, sao trời).
- Kiểm tra giá trị biên: Áp dụng với giá trị như máu = 0 (chết), năng lượng = 0 (không thể bắn và sử dụng kỹ năng), số lượng vật phẩm sao trời đủ điều kiện qua màn.
- Kiểm thử chuyển đổi trạng thái: Kiểm tra các trạng thái chuyển trong game: chơi, pause, thắng/thua, qua màn, chơi lại, menu.
- Đoán lỗi: Thực hiện các tổ hợp phím một cách liên tục để tìm các bug khó đoán
- Kiểm thử giao diện: Kiểm tra các nút bấm, phản hồi khi bấm.

#### **3.4. Công cụ**

Unity Test Runner, play-mode của Unity, Excel để theo dõi bug.

#### 4. Tiêu chí dừng (Exit criteria)

##### 4.1. Test case

- Tất cả test case chức năng đã thiết kế (30 TC) phải được thực thi.
- Tỷ lệ Pass  $\geq 90\%$ .

##### 4.2. Thời gian & Tài nguyên

- Không vượt quá ngân sách thời gian kiểm thử (2 ngày execution/tuần).
- Không còn tài nguyên tester/developer sẵn sàng để tiếp tục fix & test.

#### 5. Kết quả kiểm thử

*Bảng 3.1. Bảng kiểm thử các chức năng của nhân vật*

ID	Test case decription		Test case procedure	Expected result	Test result
	I	II			
Kiểm tra di chuyển nhân vật					
PL001	Chức năng di chuyển trái, phải		Cho nhân vật di chuyển trái, phải (nhấn vào A/D)	Nhân vật có thể di chuyển trái phải	Pass
	Chức năng di chuyển bị chặn bởi vật thể trong game		Cho nhân vật di chuyển lại các bức tường	Nhân vật bị chặn lại bởi tường	Pass
Kiểm tra nhảy					
PL002	Chức năng nhảy	Nhảy 1 lần	Cho nhân vật nhảy lên (Nhấn nút space)	Nhân vật có thể nhảy	Pass
		Nhảy 2 lần	Cho nhân vật nhảy lên hai lần	Nhân vật nhảy một lần	Pass

			(Nhấn 2 lần nút space)	và nhảy 2 nó sẽ lộn nhào trên không một lần	
Kiểm tra leo trèo					
PL003	Chức năng leo trèo	Trèo thang lên	Cho nhân vật lại gần thang và trèo trên (Nhấn vào W)	Nhân vật có thể trèo thang và đi lên	Pass
		Trèo thang xuống	Cho nhân vật lại gần thang và trèo xuống (Nhấn vào S)	Nhân vật có thể trèo thang và đi xuống	Pass
Kiểm tra bắn đạn					
PL004			Cho nhân vật bắn đạn (Nhấn chuột trái)	Nhân vật có thể bắn đạn	Pass
Kiểm tra chém					
PL005	Chức năng tấn công của nhân (chém)		Cho nhân vật chém (Nhấn chuột phải)	Nhân vật có thể chém	Failed
Kiểm tra lướt					
PL006	Chức năng lướt		Cho nhân vật lướt (Nhấn shift trái)	Nhân vật có thể lướt	Pass
Kiểm tra nhận sát thương					



PL007	Chức năng nhận sát thương của nhân vật	Cho nhân vật di chuyển lại gần quái	Nhân vật bị quái chém hoặc bắn và bị trừ HP	Pass
-------	--	-------------------------------------	---	------

*Bảng 3.2. Bảng kiểm thử các chức năng của quái*

ID	Test case decription		Test case procedure	Expected result	Test result
	I	II			
Kiểm tra di chuyển quái					
EN001	Di chuyển của quái	Quái đi bộ	Di chuyển đến người chơi trong phạm vi gần	Quái di chuyển hướng về người chơi	Fail
		Quái bay	Di chuyển đến người chơi trong phạm vi gần	Quái bay tự động hướng về người chơi	Pass
EN002	Di chuyển sau khi dính đạn	Quái đi bộ	Người chơi bắn vào quái đi bộ	Quái vẫn di chuyển	Pass
		Quái bay	Người chơi bắn vào quái bay	Quái vẫn di chuyển	Pass
EN003	Đánh	Quái đi bộ	Người chơi đứng gần trong tầm đánh của quái đi bộ	Quái tấn công bằng đòn cận chiến mỗi 2s	Pass
		Quái bay	Người chơi đứng trong tầm tấn	Quái tấn công bằng đòn bắn từ xa	Pass

			công của quái bay		
EN004	Nhận sát thương	Quái đi bộ	Tấn công quái bằng đạn	Mỗi đòn bắn trúng làm quái mất 10 máu	Pass
		Quái bay			Pass
EN005	Chết khi hết máu	Quái đi bộ	Tấn công quái cho đến khi máu về 0	Quái biến mất và có animation chết	Pass
		Quái bay			Pass
EN006	Cơ chế AI của boss	Boss	Di chuyển quanh khu vực của boss, tấn công người chơi	Quái đổi hướng liên tục để tấn công người chơi	Pass

*Bảng 3.3. Bảng kiểm thử tương tác với vật thể môi trường*

ID	Test case decription	Test case procedure	Expected result	Test result
UI001	Tương tác với rương báu	Cho người chơi đi đến gần rương	Rương mở ra, người chơi nhận được ngẫu nhiên 20 máu hoặc tăng tốc chạy / hồi năng lượng	Pass
		Tương tác bằng cách đi lại gần		
UI002	Nhận buff từ rương báu	Mở rương nhiều lần trong một màn chơi và quan sát loại buff nhận được	Buff ngẫu nhiên giữa 2 loại: +20HP, tăng tốc chạy và tăng hồi năng lượng	Pass

UI003	Va chạm với bẫy gai	Di chuyển người chơi đạp lên bẫy gai	Người chơi bị trừ 10 máu	Pass
UI004	Né bẫy gai thành công	Cho người chơi nhảy qua hoặc né bẫy gai bằng cách di chuyển kỹ năng	Người chơi không mất máu nếu né thành công	Pass
		Không chạm vào hitbox của bẫy gai		
UI005	Sử dụng thang để trèo	Người chơi đứng gần thang	Người chơi có thể leo lên và xuống mượt mà	Pass
		Nhấn phím W/S hoặc phím nhảy		
		Quan sát chuyển động lên và xuống		
UI006	Tương tác với thang bị ngắt giữa chừng	Người chơi nhảy từ giữa thang hoặc rơi từ trên xuống vào giữa thang	Người chơi có thể tiếp tục trèo từ giữa thang hoặc rơi vào đúng giữa vẫn bám được	Failed
		Quan sát xem người chơi có bám vào thang và leo tiếp được không		

*Bảng 3.4. Bảng kiểm thử chức năng hệ thống và giao diện*

ID	Test case decription	Test case procedure	Expected result	Test result
----	-------------------------	---------------------	-----------------	----------------

Trải nghiệm chơi				
NF001	Đảm bảo FPS ổn định trong quá trình chơi	Chạy game trong 3 phút, spawn quái và hiệu ứng liên tục	FPS duy trì ổn định, không giật lag	Pass
Cơ chế chuyển scene của storyboard				
NF002	Cơ chế qua màn	Thu thập đủ 3 vật phẩm sao trời, ấn nút tiếp tục	Bảng chiến thắng hiện lên, sau khi tiếp tục thì màn tiếp theo không bị lỗi dữ liệu	Pass
NF003	Cơ chế lưu tiến trình	Qua màn, trở lại menu, tiếp tục màn chơi trước đó	Có thể chọn tiếp màn chơi đã dừng, không bị chong dữ liệu	Pass
NF004	Cơ chế chơi lại	Thực hiện chết sau đó chọn chơi lại	Quay lại điểm spawn của map đang chơi	Pass
		Thực hiện chơi lại từ menu		
		Thực hiện chiến thắng sau đó chơi lại		
Kiểm tra các nút bấm trong UI				
NF005	Cơ chế hoạt động của các nút bấm	Bấm lần lượt các nút xem nhân vật, âm thanh, bật/tắt âm, cài đặt	Các nút hoạt động đúng như mong muốn	Pass

*Bảng 3.5. Bảng tổng hợp kết quả kiểm thử*

<b>Nhóm chức năng</b>	<b>Tổng số TCs</b>	<b>Passed</b>	<b>Failed</b>
Nhân vật	7	6	1
Quái (Enemy & Boss)	6	6	0
Tương tác với vật thể môi trường	6	5	1
Chức năng hệ thống và giao diện	5	5	0

## KẾT LUẬN

Như đã nêu trong phần giới thiệu tổng quan, nhóm chúng em mong muốn áp dụng các lý thuyết, công cụ đã học để giải quyết đề tài xây dựng game này. Kết hợp việc tự học cũng như nghiên cứu cách xây dựng và thiết kế, lập trình game 2D, nhóm chúng em đã thành thạo hơn trong việc sử dụng công cụ Unity cùng ngôn ngữ C# để tạo ra một sản phẩm game hoàn chỉnh.

Từ đó, sản phẩm game “Tiny Fight” được ra đời, mang một giao diện tổng thể hài hòa, bắt mắt kết hợp âm thanh nhẹ nhàng và quan trọng hơn hết là các nhân vật và quái được thiết kế hoàn chỉnh về mặt logic. Các map chơi được thiết kế lên tới 3 map cùng độ khó của các quái khủng khi áp dụng logic AI là điểm nhấn cuốn hút người chơi so với các tựa game cùng thể loại.

Vì thời gian và nguồn tài nguyên có hạn, chúng em có một vài thứ chưa đạt được mong muốn như: UI còn hạn chế, thiết kế cấp độ màn chơi chưa thực sự tốt về độ khó cũng như số lượng, cơ chế chưa nhiều như mong đợi; vẫn chưa có những kẻ địch có gimmick và cách khắc chế đặc biệt; Platform còn hơi một màu và đơn điệu chưa thỏa yêu cầu đặt ra trong ý tưởng ban đầu.

Trong tương lai, nhóm chúng em đặt ra các mục tiêu tối ưu hóa cũng như nâng cấp trò chơi bằng cách thêm chức năng chọn nhiều nhân vật, các nhân vật có những cơ chế khác biệt, nâng cấp các chỉ số sức mạnh cho người chơi. Bên cạnh đó nhóm cũng hướng tới các map bổ sung với độ khó được thiết kế tốt hơn và kẻ địch, cạm bẫy đa dạng hơn.

## TÀI LIỆU THAM KHẢO

- [1]. Rykała, P. (2020). The growth of the gaming industry in the context of creative industries. Biblioteka Regionalisty. Regional Journal.
- [2]. Con sôt Flappy bird của Việt Nam: <https://danviet.vn/10-nam-sau-con-sot-flappy-bird-viet-nam-tro-thanh-cuong-quoc-ve-game-20230806034411698-d1110518.html>, 3/4/2025).
- [3]. Dave Calabrese (2014). Unity 2D Game Development. Packt Publishing Ltd. Livery Place.
- [4]. Điều gì khiến Unity trở nên phổ biến: <https://www.arnia.com/what-makes-unity-so-popular-in-game-development>, 3/4/2025.
- [5]. Học phát triển game với Unity: [Learn game development w/ Unity | Courses & tutorials in game design, VR, AR, & Real-time 3D | Unity Learn](#), 12/4/2025.
- [6]. Định nghĩa game nhập vai: <https://www.techtarget.com/whatis/definition/role-playing-game-RPG>, 3/4/2025.
- [7]. Định nghĩa game phiêu lưu: [https://vi.wikipedia.org/wiki/Tr%C3%B2\\_ch%C6%A1i\\_phi%C3%AAu\\_l%C6%B0u](https://vi.wikipedia.org/wiki/Tr%C3%B2_ch%C6%A1i_phi%C3%AAu_l%C6%B0u), 3/4/2025.

## **PHỤ LỤC**

PHỤ LỤC 1: PHIẾU HỌC TẬP CÁ NHÂN NHÓM

PHỤ LỤC 2: KẾ HOẠCH THỰC HIỆN BÀI TẬP LỚN

PHỤ LỤC 3: BIÊN BẢN LÀM VIỆC NHÓM