

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**LÊ GIA HUY - 52100033
ĐẶNG MINH PHONG - 52100987**

THE LOOP

BÁO CÁO CUỐI KỲ PHÁT TRIỂN TRÒ CHƠI

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN**



**LÊ GIA HUY - 52100033
ĐẶNG MINH PHONG - 52100987**

THE LOOP

BÁO CÁO CUỐI KỲ PHÁT TRIỂN TRÒ CHƠI

Người hướng dẫn
ThS. Vũ Đình Hồng

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn đến thầy Vũ Đình Hồng, người trực tiếp giảng dạy và hướng dẫn em thực hiện bài báo cáo, cũng như đã tận tình giúp đỡ và tạo điều kiện cho chúng em trong quá trình học tập.

Dù đã đầu tư thời gian và nỗ lực để hoàn thành báo cáo, nhưng với những hạn chế về kinh nghiệm cũng như kiến thức thì việc có sai sót trong bài làm là không thể tránh khỏi. Chúng em mong nhận được sự đóng góp ý kiến của thầy để có thể hoàn thiện hơn nội dung đề tài

Chúng em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 23 tháng 4 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Lê Gia Huy

Đặng Minh Phong

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của ThS. Vũ Đình Hồng. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 23 tháng 4 năm 2024

Tác giả

(Ký tên và ghi rõ họ tên)

Lê Gia Huy

Đặng Minh Phong

THE LOOP

TÓM TẮT

Trong báo cáo này bao gồm quá trình lên kế hoạch và hiện thực dự án game The Loop, các cơ sở lý thuyết, kiến trúc và tất cả những gì liên quan đến The Loop đều được đề cập bên dưới

MỤC LỤC

DANH MỤC HÌNH VẼ	viii
DANH MỤC BẢNG BIỂU	xi
CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI THE LOOP.....	1
1.1 Lý do chọn đề tài.....	1
1.2 Mục tiêu thực hiện đề tài.....	1
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	2
2.1 Thể loại Roguelike và Roguelite.....	2
2.1.1 Roguelike	2
2.1.2 Roguelite	3
2.1.3 Sự mờ nhạt trong định nghĩa	4
2.2 Các thể loại phụ.....	5
2.2.1. 2D Platformer	5
2.2.2. Side-Scrolling	5
2.2.3. Hack & Slash	5
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ	6
3.1 Sơ đồ Usecase	6
3.2 Sơ đồ Usecase	6
3.2.1 Bắt đầu game	6
3.2.2 Tiếp tục game.....	7
3.2.3 Cài đặt âm thanh	9
3.2.4 Cài đặt đồ họa	10

3.2.5	Thoát game.....	11
3.2.6	Dừng game.....	12
3.2.7	Trở về menu	13
3.2.8	Điều khiển nhân vật chạy.....	14
3.2.9	Điều khiển nhân vật nhảy	15
3.2.10	Điều khiển nhân vật lướt.....	16
3.2.11	Điều khiển nhân vật đánh thường	17
3.2.12	Điều khiển nhân vật dùng kỹ năng	18
3.2.13	Nâng cấp nhân vật.....	20
3.2.14	Thay đổi vũ khí	21
3.2.15	Chuyển màn chơi	22
3.3	Sơ đồ Tuần tự	23
3.3.1.	Bắt đầu game	23
3.3.2.	Cài đặt	24
3.3.3.	Thoát game.....	24
3.3.4.	Tạm dừng game	25
3.3.5.	Trở về menu	26
3.3.6.	Điều khiển nhân vật	27
3.3.7.	Tương tác môi trường	28
3.3.8.	Nâng cấp nhân vật.....	29
3.4	Sơ đồ Hoạt động.....	30
3.4.1.	Bắt đầu game	30
3.4.2.	Cài đặt	31

3.4.3. Thoát game.....	31
3.4.4. Tạm dừng game	32
3.4.5. Trở về menu	32
3.4.6. Tiếp tục game.....	33
3.4.7. Nhân vật di chuyển	34
3.4.8. Nhân vật nhảy	35
3.4.9. Nhân vật lướt	36
3.4.10. Nhân vật đánh thường.....	37
3.4.11. Nhân vật dùng kỹ năng	38
3.4.12. Đổi vũ khí	39
3.4.13. Nâng cấp nhân vật.....	40
CHƯƠNG 4. HIỆN THỰC HỆ THỐNG.....	41
4.1 Unity2D	41
4.2 Nhân vật chính	41
4.2.1 Các components	41
4.2.2 Animation.....	44
4.2.3 Điều khiển người chơi	46
4.3 Kẻ thù	54
4.3.1 A* Pathfinding Project.....	54
4.3.2 AI dành cho kẻ thù di chuyển dưới đất	56
4.3.3 Phát hiện và tấn công người chơi.....	57
4.4 Camera	59
4.5 Lưu tiến trình chơi.....	61

CHƯƠNG 5. KẾT QUẢ ĐẠT ĐƯỢC.....	64
5.1 Tổng quan.....	64
5.2 Gameplay.....	65
5.3 Cấu trúc	67
5.4 Giao diện người dùng (UI)	68
5.5 Âm thanh và đồ họa.....	69
CHƯƠNG 6. KẾT LUẬN.....	71
6.1 Kết luận	71
6.2 Hướng phát triển	71
TÀI LIỆU THAM KHẢO	73

DANH MỤC HÌNH VẼ

Hình 2.1 Gameplay của Rogue (1980).....	2
Hình 2.2. Game ADOM thuộc thể loại Roguelike.....	3
Hình 3.1. Sơ đồ Usecase	6
Hình 3.2. Sơ đồ tuần tự bắt đầu game.....	23
Hình 3.3. Sơ đồ tuần tự cài đặt.....	24
Hình 3.4. Sơ đồ tuần tự thoát game.....	24
Hình 3.5. Sơ đồ tuần tự tạm dừng game	25
Hình 3.6. Sơ đồ tuần tự trở về menu	26
Hình 3.7. Sơ đồ tuần tự điều khiển nhân vật.....	27
Hình 3.8. Sơ đồ tuần tự tương tác môi trường	28
Hình 3.9. Sơ đồ tuần tự nâng cấp nhân vật	29
Hình 3.10. Sơ đồ hoạt động bắt đầu game	30
Hình 3.11. Sơ đồ hoạt động cài đặt.....	31
Hình 3.12. Sơ đồ hoạt động thoát game.....	31
Hình 3.13. Sơ đồ hoạt động tạm dừng game.....	32
Hình 3.14 Sơ đồ hoạt động trở về menu	32
Hình 3.15 Sơ đồ hoạt động tiếp tục game	33
Hình 3.16. Sơ đồ hoạt động nhân vật di chuyển	34
Hình 3.17. Sơ đồ hoạt động nhân vật nhảy	35
Hình 3.18. Sơ đồ hoạt động nhân vật lướt	36
Hình 3.19. Sơ đồ hoạt động nhân vật đánh thường.....	37
Hình 3.20. Sơ đồ hoạt động nhân vật dùng kỹ năng.....	38

Hình 3.21. Sơ đồ hoạt động đối vũ khí	39
Hình 3.22. Sơ đồ hoạt động nâng cấp nhân vật	40
Hình 4.1. InputSystem cho nhân vật	42
Hình 4.2. Player Input	43
Hình 4.3. Damage Controller	44
Hình 4.4. FSM của nhân vật chính.....	45
Hình 4.5. Script quản lý animation	46
Hình 4.6. Script điều khiển người chơi.....	47
Hình 4.7. One Way Platform.....	48
Hình 4.8. Script OnDown cho phép nhảy xuống platform	48
Hình 4.9. OnDash điều khiển lướt	49
Hình 4.10. Buff kỹ năng.....	50
Hình 4.11. Các kỹ năng của The Loop qua Script	51
Hình 4.12 ApplyBuff	52
Hình 4.13. AttackController.....	53
Hình 4.14. Collider của AttackController	53
Hình 4.15. Áp dụng AttackController.....	54
Hình 4.16. Pathfinder cho FlyingEye (1).....	55
Hình 4.17. Pathfinder cho FlyingEye (2).....	55
Hình 4.18. Pathfinder cho FlyingEye (3).....	56
Hình 4.19. Pathfinder cho quái vật dưới đất (1)	56
Hình 4.20. Pathfinder cho quái vật dưới đất (2)	57
Hình 4.21. Đối tượng con cho cơ chế Detection.....	58

Hình 4.22. DetectionZone	59
Hình 4.23. Cinemachine.....	60
Hình 4.24. Lưu tiến trình chơi (1)	62
Hình 4.25. Lưu tiến trình chơi (2)	62
Hình 4.26. Load tiến trình chơi (1)	63
Hình 4.27. Load tiến trình chơi (2)	63
Hình 5.1. Bối cảnh ngôi làng	64
Hình 5.2. Các loại vũ khí có thể dùng.....	65
Hình 5.3. Ví dụ một màn chơi điển hình.....	66
Hình 5.4. Hệ thống phần thưởng.....	67
Hình 5.5. Boss của game.....	68
Hình 5.6. Lối thiết kế của Dead Cells	70

DANH MỤC BẢNG BIỂU

Bảng 3.1 Usecase bắt đầu game.....	7
Bảng 3.2. Usecase tiếp tục game.....	8
Bảng 3.3. Usecase cài đặt âm thanh.....	10
Bảng 3.4. Usecase cài đặt đồ họa.....	11
Bảng 3.5. Usecase thoát game.....	12
Bảng 3.6. Usecase dừng game	13
Bảng 3.7. Usecase trở về menu	14
Bảng 3.8. Usecase điều khiển nhân vật chạy	15
Bảng 3.9. Usecase điều khiển nhân vật nhảy	16
Bảng 3.10. Usecase điều khiển nhân vật lướt	17
Bảng 3.11. Usecase điều khiển nhân vật đánh thương	18
Bảng 3.12. Usecase điều khiển nhân vật dùng kỹ năng.....	19
Bảng 3.13. Usecase nâng cấp nhân vật	20
Bảng 3.14. Usecase thay đổi vũ khí	21
Bảng 3.15. Usecase chuyển màn chơi.....	22

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI THE LOOP

1.1 Lý do chọn đề tài

Trong thế giới game, mỗi thể loại game đều có cho mình một sức hút đặc biệt khiến game thủ thích thú và gắn bó với nó. Roguelite cũng không ngoại lệ, với yếu tố ngẫu nhiên và nâng cấp qua mỗi vòng chơi, không lần chơi nào là giống nhau, điều này tạo ra sự mới lạ và độc đáo với mỗi lần chơi.

Lý do đề tài The Loop được chọn là vì mong muốn tạo ra một tựa game roguelite 2D đủ thách thức và mới lạ, tạo ra trải nghiệm chơi game độc đáo và thú vị. Bên cạnh đó, việc phát triển một game roguelite cũng đòi hỏi các kỹ năng lập trình và thiết kế game đa dạng, từ việc xử lý các hệ thống ngẫu nhiên đến việc cân nhắc thiết kế màn chơi và cấu trúc dữ liệu phù hợp.

1.2 Mục tiêu thực hiện đề tài

Để tạo ra một tựa game roguelite đủ hấp dẫn, The Loop phải đạt được những mục tiêu sau:

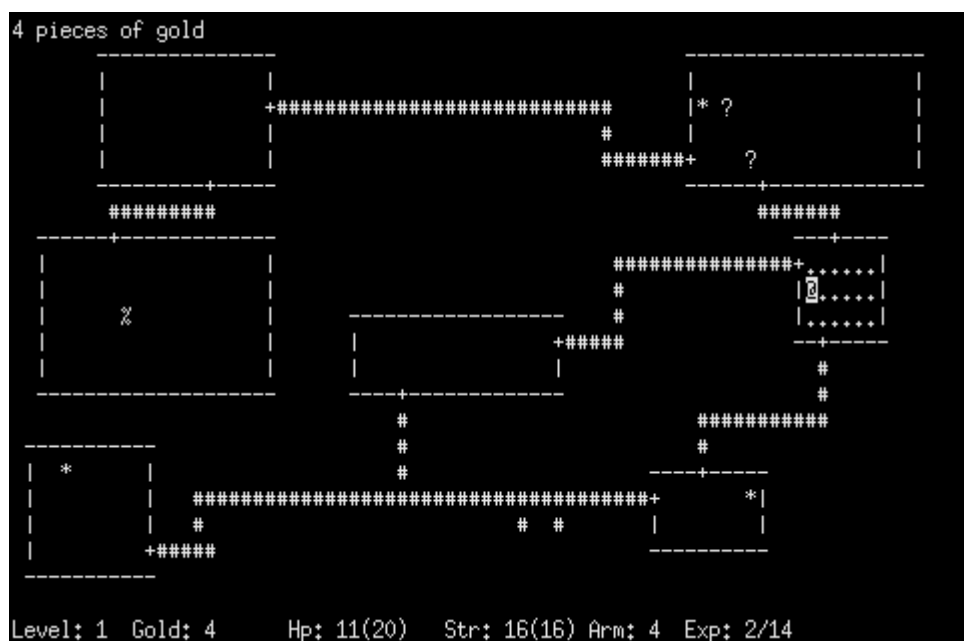
- Dễ tiếp cận nhưng thách thức: The Loop có thể được chơi bằng những thao tác đơn giản, nhưng phải đủ thách thức để người chơi có thể chơi đi chơi lại.
- Gameplay có tính đa dạng: để tạo ra sự ngẫu nhiên và không lặp lại của roguelike, The Loop phải có hệ thống vũ khí, quái vật, kỹ năng đủ lớn và sáng tạo.
- Trải nghiệm người chơi: các động tác di chuyển, đánh, hệ thống bản đồ phải có độ đơn giản để người chơi bắt kịp gameplay, nhưng cũng đủ độ phức tạp để người chơi tập làm quen trong vài lần chơi đầu
- Hoàn thiện: The Loop phải đảm bảo chất lượng từ hình ảnh đến âm thanh, đặc biệt là phải ổn định khi chơi

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Thể loại Roguelike và Roguelite

2.1.1 Roguelike

Trước khi nói về roguelite, hãy nói về roguelike trước, roguelike (giống như “Rogue”) là một thể loại game xuất phát từ game Rogue năm 1980. Rogue là một game kiểu mê cung đường hầm, trong đó người chơi phải vượt qua các màn chơi được bố trí như đường hầm, thu thập vật phẩm và đánh bại kẻ thù. Điểm đặc biệt của Rogue là game được thiết kế mỗi lần chơi mê cung sẽ được thiết kế khác nhau, khiến người chơi không thể hoàn thành mê cung bằng cách nhớ lại đường đi. Những tựa game về sau lấy cảm hứng từ Rogue để thiết kế lối chơi tương tự được gọi là Roguelike.



Hình 2.1 Gameplay của Rogue (1980)

Dựa trên Rogue, một số tựa game mang định nghĩa roguelike đã ra đời như: Vultures, Tales of Maj'Eyal, Dungeonmans, ADOM,... tất cả các game đều có những đặc điểm như sau:

- Các levels trong game được tạo ngẫu nhiên trong mỗi lần chơi

- Tính chơi lại cao
- Gameplay theo lượt
- Di chuyển dạng lưới (Grid)
- Chết vĩnh viễn (Permadeath), tức là sau khi chết người chơi sẽ chơi lại từ đầu và mất hết tất cả vật phẩm



Hình 2.2. Game ADOM thuộc thể loại Roguelike

2.1.2 Roguelite

Roguelite về cơ bản là một biến thể của Roguelike, nó vẫn mang những giá trị cốt lõi của Roguelike nhưng được thay đổi đi để mang lại sự mới mẻ. So với Roguelike, Roguelite được đánh giá là dễ tiếp cận hơn với người chơi, cụ thể là một số tựa game cho phép người chơi khi chết được phép giữ lại một số vật phẩm, điều này đồng nghĩa càng chơi thì nhân vật của người chơi sẽ càng mạnh. Một số tựa game còn kết hợp với những thể loại khác thay vì bó buộc vào thể loại theo lượt, ví dụ như The Binding of Isaac với thể loại Shoot 'em up, Gunfire Reborn với thể loại FPS,

điều khiến các tựa game này được gọi là “Roguelite” vì nó vẫn có một số đặc điểm như sự ngẫu nhiên trong mỗi lần chơi, khi chết vẫn chơi lại từ đầu,...

Một số tựa game Roguelite nổi bật, đồng thời cũng là cảm hứng, tiêu chuẩn của The Loop:

- Hades: Action Roguelite kết hợp với Hack & Slash, các dungeon của game và những vật phẩm buff được ngẫu nhiên trong mỗi lượt chơi
- Blade Assault: Roguelike platformer với đồ họa pixel và gameplay Hack & Slash
- Dead Cells: tương tự Blade Assault, nhưng Dead Cells được thiết kế với mỗi levels là một bản đồ rộng lớn để người chơi khám phá
- Spirifall: cũng là một roguelite platformer nhưng mỗi level của game là một map nhỏ kiểu sandbox, bó buộc người chơi ở không gian nhỏ và tiêu diệt quái theo từng lượt

2.1.3 Sự mờ nhạt trong định nghĩa

Theo thời gian, nhiều người hiện nay không còn phân biệt rạch ròi hai khái niệm “roguelike” và “roguelite” nữa. Khi ai đó nói về “roguelike”, có thể họ không nghĩ rằng nó là game chui hầm kiếm vật phẩm giống như Rogue thời xưa mà chỉ là một game có môi trường ngẫu nhiên và permadeath, hiện nay khái niệm phổ biến nhất dành cho roguelite là tương tự game roguelike nhưng khi nhân vật chết sẽ được giữ một vài nâng cấp.

Nhiều người không đồng ý với bộ tiêu chuẩn về roguelike vì không thể bắt các game ngày nay duy trì lối chơi như Rogue. Điều này cũng tương tự như sau khi game Doom ra mắt năm 1993, các game theo phong cách giống với nó từng được gọi là “Doom clone” nhưng ngày nay đã được gọi là game bắn súng góc nhìn thứ nhất và không thể bắt tất cả chúng đều có cách chơi giống hệt như Doom.

2.2 Các thể loại phụ

2.2.1. 2D Platformer

Platformer là thể loại game mà trong đó mục tiêu chính là di chuyển nhân vật của người chơi giữa các điểm được gọi là platform trong môi trường. Platformer được đặc trưng bởi các cấp độ có địa hình không đồng đều và yêu cầu người chơi phải nhảy hoặc leo để vượt qua. Một số tựa game nổi tiếng với lối thiết kế này là Mario, Megaman, Prince of Persia,...

2.2.2. Side-Scrolling

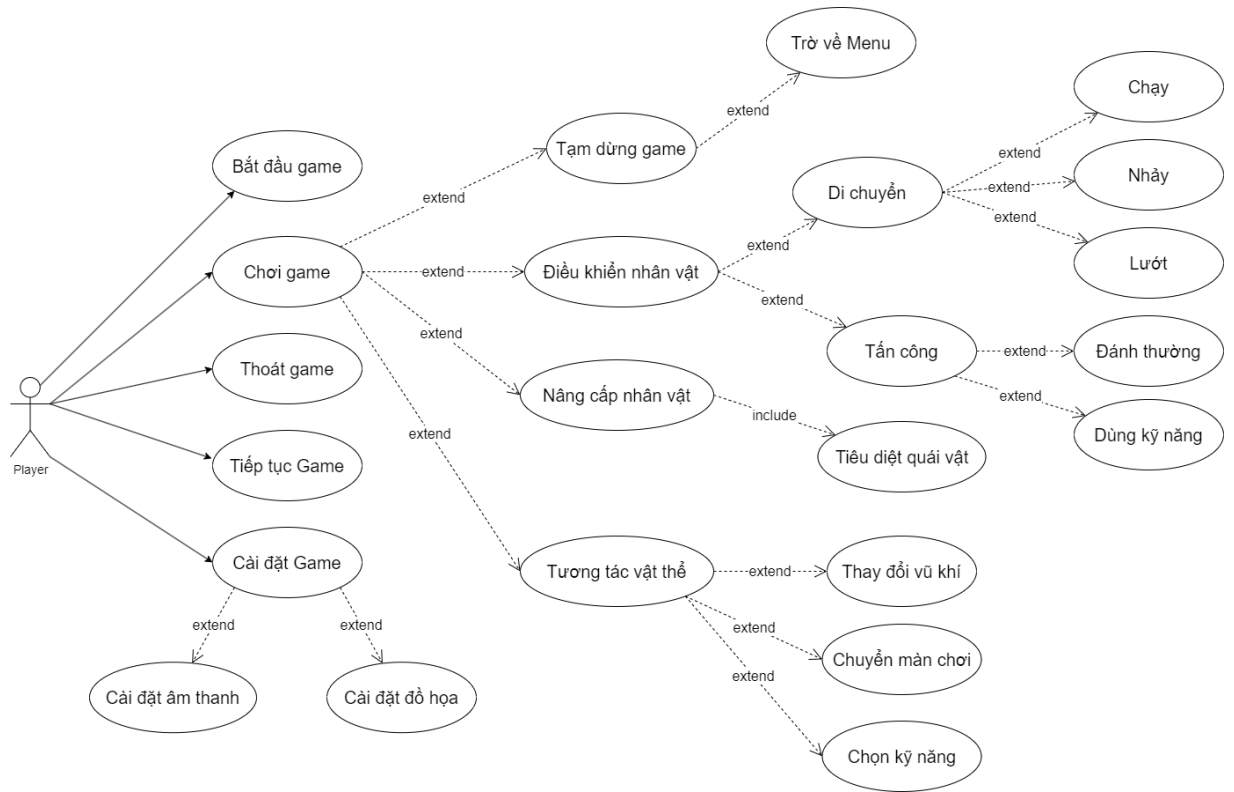
Side-Scrolling là một thể loại mà trong đó camera sẽ được đặt ngang nhân vật và di chuyển theo nhân vật khi di chuyển sang trái hoặc phải. Dù có thể được áp dụng với game 2D và 3D, Side-Scrolling thường được biết nhiều qua game 2D do hầu hết những tựa game ngày xưa đều dùng góc nhìn này.

2.2.3. Hack & Slash

Hack & Slash là một thể loại nhấn mạnh vào gameplay chiến đấu với vũ khí cận chiến, có thể có vũ khí tầm xa như một vũ khí phụ, thể loại này thường đi liền với thể loại hành động. Một số tựa game nổi tiếng với thể loại này có thể kể đến Devil May Cry, Nier:Automata,...

CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ

3.1 Sơ đồ Usecase



Hình 3.1. Sơ đồ Usecase

3.2 Sơ đồ Usecase

3.2.1 Bắt đầu game

Mã use case	UC01
Tên Use Case	Bắt đầu game
Mô tả	Người chơi đã khởi động game và muốn bắt đầu quá trình chơi game
Tác nhân	Người chơi

Mã use case	UC01	
Sự kiện kích hoạt	Bắt đầu vào game	
Điều kiện tiên quyết	Người chơi đã khởi động game	
Kết quả	Game chuyển cảnh sang khu vực bắt đầu	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn vào nút “Start” ở menu	1.1 Hệ thống kiểm tra tiến trình chơi game 1.2 Game bắt đầu chuyển cảnh sang khu vực bắt đầu
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.1 Usecase bắt đầu game

3.2.2 Tiếp tục game

Mã use case	UC02
Tên Use Case	Tiếp tục game
Mô tả	Người chơi đã khởi động game và muốn tiếp tục quá trình chơi game

Mã use case	UC02	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Bắt đầu vào game	
Điều kiện tiên quyết	Người chơi đã khởi động game và đã có tiến trình chơi game được lưu trước đó	
Kết quả	Game chuyển cảnh sang khu vực đã được lưu tiến trình	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn vào nút “Start” ở menu	1.1 Hệ thống kiểm tra tiến trình chơi game 1.2 Game bắt đầu chuyển cảnh sang khu vực người chơi đã lưu tiến trình trước đó
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - File save bị lỗi 	

Bảng 3.2. Usecase tiếp tục game

3.2.3 Cài đặt âm thanh

Mã use case	UC03	
Tên Use Case	Cài đặt âm thanh	
Mô tả	Người chơi muốn thay đổi âm lượng của game	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi thay đổi âm thanh trong phần cài đặt	
Điều kiện tiên quyết	Người chơi đã vào phần cài đặt của game	
Kết quả	Âm lượng game thay đổi	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn vào nút “Settings” ở menu	1.1 Hệ thống chuyển sang màn hình Settings
	2. Người chơi thay đổi âm lượng bằng cách kéo thả thanh âm lượng	2.1 Hệ thống thay đổi âm lượng dựa trên thanh âm lượng hiện tại
	3. Người chơi nhấn “Save”	3.1 Hệ thống lưu thay đổi và chuyển trở lại màn hình Menu

Mã use case	UC03
Ngoại lệ	- Game bị lỗi, bug không xác định

Bảng 3.3. Usecase cài đặt âm thanh

3.2.4 Cài đặt đồ họa

Mã use case	UC04	
Tên Use Case	Cài đặt đồ họa	
Mô tả	Người chơi muốn thay đổi đồ họa của game	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi thay đổi đồ họa trong phần cài đặt	
Điều kiện tiên quyết	Người chơi đã vào phần cài đặt của game	
Kết quả	Đồ họa game thay đổi	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn vào nút “Settings” ở menu	1.2 Hệ thống chuyển sang màn hình Settings

Mã use case	UC04	
	2. Người chơi thay đổi đồ họa bằng cách lựa chọn đồ họa	2.2 Hệ thống thay đổi đồ họa trò chơi
	3. Người chơi nhấn “Save”	3.1 Hệ thống lưu thay đổi và chuyển trở lại Menu
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.4. Usecase cài đặt đồ họa

3.2.5 Thoát game

Mã use case	UC05
Tên Use Case	Thoát game
Mô tả	Người chơi có nhu cầu thoát game và dừng quá trình chơi
Tác nhân	Người chơi
Sự kiện kích hoạt	Thoát khỏi trò chơi
Điều kiện tiên quyết	Người chơi đã khởi động game
Kết quả	Trò chơi kết thúc tiến trình

Mã use case	UC05	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn vào nút “Exit” ở menu	1.3 Trò chơi tắt và kết thúc tiến trình
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.5. Usecase thoát game

3.2.6 Dừng game

Mã use case	UC06	
Tên Use Case	Dừng game	
Mô tả	Người chơi muốn tạm dừng game	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Tạm dừng tiến trình chơi game	
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi	
Kết quả	Game được tạm dừng	
Luồng sự kiện	Tác nhân	Hệ thống

Mã use case	UC06	
	1. Người chơi nhấn vào nút “Esc” trên bàn phím	1.1 Hệ thống tạm dừng game và chuyển sang menu tạm dừng
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.6. Usecase dừng game

3.2.7 Trở về menu

Mã use case	UC07	
Tên Use Case	Trở về Menu	
Mô tả	Người chơi muốn trở về menu chính của game	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Game trở về màn hình menu chính	
Điều kiện tiên quyết	Người chơi đã tạm dừng game	
Kết quả	Game trở về màn hình menu chính	
Luồng sự kiện	Tác nhân	Hệ thống

Mã use case	UC07	
	1. Người chơi nhấn vào nút “Menu” ở menu tạm dừng	1.1 Hệ thống lưu tiến trình hiện tại 1.2 Trò chơi trở về màn hình chính
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.7. Usecase trở về menu

3.2.8 Điều khiển nhân vật chạy

Mã use case	UC08	
Tên Use Case	Điều khiển nhân vật chạy	
Mô tả	Người chơi điều khiển cho nhân vật chạy	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi ấn nút điều khiển nhân vật chạy	
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi	
Kết quả	Nhân vật di chuyển	
Luồng sự kiện	Tác nhân	Hệ thống

Mã use case	UC08	
	1. Người chơi nhấn nút “A” hoặc “D” trên bàn phím	1.1 Hệ thống nhận input 1.2 Hệ thống cho nhân vật thực hiện động tác chạy
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đang thực hiện động tác lướt - Nhân vật đang thực hiện động tác đánh thường và dùng kỹ năng - Nhân vật đang bị tấn công 	

Bảng 3.8. Usecase điều khiển nhân vật chạy

3.2.9 Điều khiển nhân vật nhảy

Mã use case	UC09
Tên Use Case	Điều khiển nhân vật nhảy
Mô tả	Người chơi điều khiển cho nhân vật nhảy
Tác nhân	Người chơi
Sự kiện kích hoạt	Người chơi ấn nút điều khiển nhân vật nhảy
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi
Kết quả	Nhân vật nhảy

Mã use case	UC09	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn nút “K” trên bàn phím	1.1 Hệ thống nhận input 1.2 Hệ thống cho nhân vật thực hiện động tác nhảy
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đang ở trên không - Nhân vật đang thực hiện động tác đánh thường và dùng kỹ năng - Nhân vật đang bị tấn công 	

Bảng 3.9. Usecase điều khiển nhân vật nhảy

3.2.10 Điều khiển nhân vật lướt

Mã use case	UC10
Tên Use Case	Điều khiển nhân vật lướt
Mô tả	Người chơi điều khiển cho nhân vật lướt
Tác nhân	Người chơi
Sự kiện kích hoạt	Người chơi ấn nút điều khiển nhân vật lướt
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi

Mã use case	UC10	
Kết quả	Nhân vật lướt về phía trước	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn nút “L” trên bàn phím	1.1 Hệ thống nhận input 1.2 Hệ thống cho nhân vật thực hiện động tác lướt
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đang thực hiện động tác lướt - Nhân vật đang thực hiện động tác đánh thương và dùng kỹ năng - Nhân vật đang bị tấn công - Động tác lướt đang trong thời gian hồi 	

Bảng 3.10. Usecase điều khiển nhân vật lướt

3.2.11 Điều khiển nhân vật đánh thương

Mã use case	UC11
Tên Use Case	Điều khiển nhân vật đánh thương
Mô tả	Người chơi điều khiển cho nhân vật đánh thương
Tác nhân	Người chơi

Mã use case	UC11	
Sự kiện kích hoạt	Người chơi ấn nút điều khiển nhân vật đánh thường	
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi	
Kết quả	Nhân vật đánh thường	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn nút “J” trên bàn phím	1.1 Hệ thống nhận input 1.2 Hệ thống cho nhân vật thực hiện động tác đánh thường
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đang thực hiện động tác lướt - Nhân vật đang thực hiện động tác dùng kỹ năng - Nhân vật đang bị tấn công 	

Bảng 3.11. Usecase điều khiển nhân vật đánh thường

3.2.12 Điều khiển nhân vật dùng kỹ năng

Mã use case	UC12
Tên Use Case	Điều khiển nhân vật dùng kỹ năng
Mô tả	Người chơi điều khiển cho nhân vật dùng kỹ năng

Mã use case	UC12	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi ấn nút điều khiển nhân vật dùng kỹ năng	
Điều kiện tiên quyết	Người chơi đang trong quá trình chơi	
Kết quả	Nhân vật dùng kỹ năng	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn nút “I” trên bàn phím	1.1 Hệ thống nhận input 1.2 Hệ thống cho nhân vật dùng kỹ năng
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đang thực hiện động tác đánh thương và dùng kỹ năng - Nhân vật đang bị tấn công - Kỹ năng của nhân vật đang trong thời gian hồi - Nhân vật không có kỹ năng 	

Bảng 3.12. Usecase điều khiển nhân vật dùng kỹ năng

3.2.13 Nâng cấp nhân vật

Mã use case	UC13	
Tên Use Case	Nâng cấp nhân vật	
Mô tả	Người chơi nâng cấp sức mạnh nhân vật	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi lựa chọn nâng cấp cho nhân vật	
Điều kiện tiên quyết	Người chơi đã hoàn thành một level	
Kết quả	Nhân vật được nâng cấp	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi chọn nâng cấp trên màn hình nâng cấp	1.1 Hệ thống nhận lựa chọn của người chơi 1.2 Hệ thống cập nhật nâng cấp cho nhân vật
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.13. Usecase nâng cấp nhân vật

3.2.14 Thay đổi vũ khí

Mã use case	UC14	
Tên Use Case	Thay đổi vũ khí	
Mô tả	Người chơi cho nhân vật thay đổi vũ khí	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi ấn nút thay đổi vũ khí	
Điều kiện tiên quyết	Người chơi đang đứng ở vị trí của vũ khí muốn đổi	
Kết quả	Nhân vật đổi vũ khí	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi điều khiển nhân vật đứng ở vũ khí muốn đổi và nhấn “F”	1.2 Hệ thống nhận input 1.2 Hệ thống cập nhật vũ khí tương ứng cho nhân vật
Ngoại lệ	<ul style="list-style-type: none"> - Game bị lỗi, bug không xác định - Nhân vật đã sở hữu vũ khí muốn đổi 	

Bảng 3.14. Usecase thay đổi vũ khí

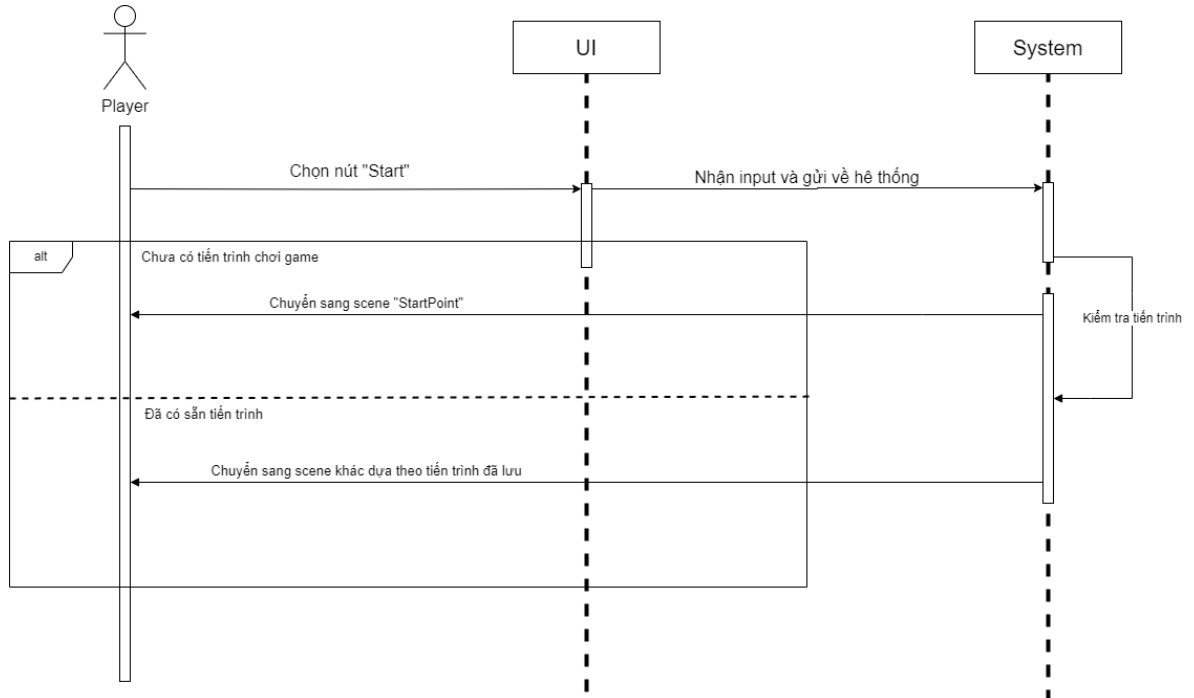
3.2.15 Chuyển màn chơi

Mã use case	UC15	
Tên Use Case	Chuyển màn chơi	
Mô tả	Người chơi chuyển sang màn chơi khác	
Tác nhân	Người chơi	
Sự kiện kích hoạt	Người chơi ấn nút chuyển màn chơi	
Điều kiện tiên quyết	Người chơi đã hoàn thành màn chơi và nâng cấp nhân vật	
Kết quả	Chuyển sang màn chơi khác	
Luồng sự kiện	Tác nhân	Hệ thống
	1. Người chơi nhấn nút “F” trên bàn phím khi đứng ở vật thể chuyển màn chơi	1.3 Hệ thống nhận input 1.2 Hệ thống chuyển sang màn chơi khác
Ngoại lệ	- Game bị lỗi, bug không xác định	

Bảng 3.15. Usecase chuyển màn chơi

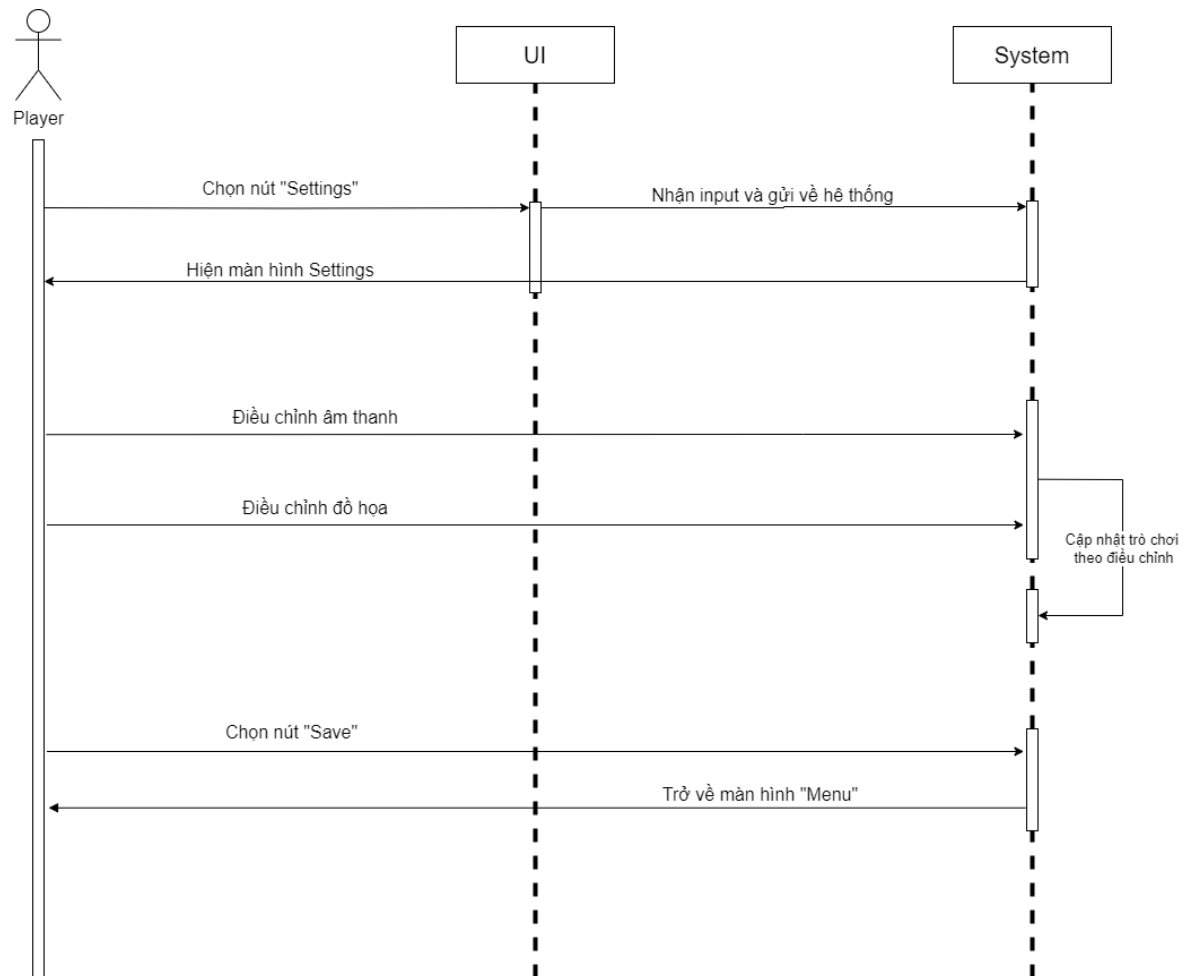
3.3 Sơ đồ Tuần tự

3.3.1. Bắt đầu game



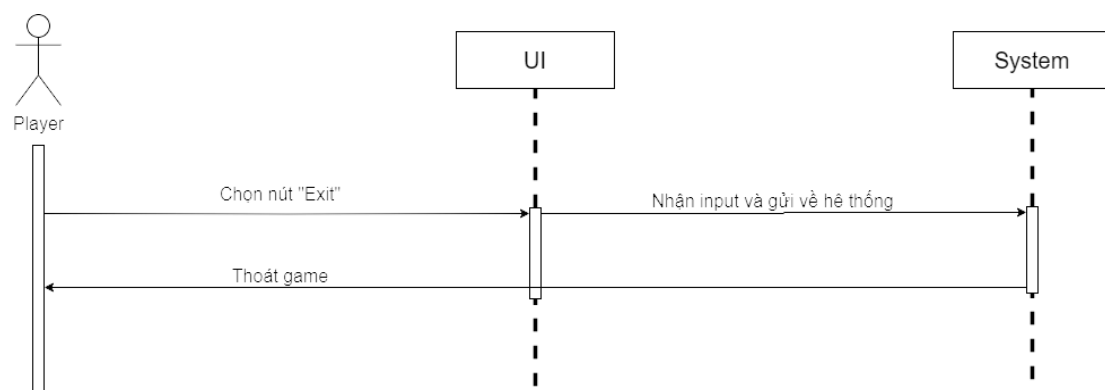
Hình 3.2. Sơ đồ tuần tự bắt đầu game

3.3.2. Cài đặt



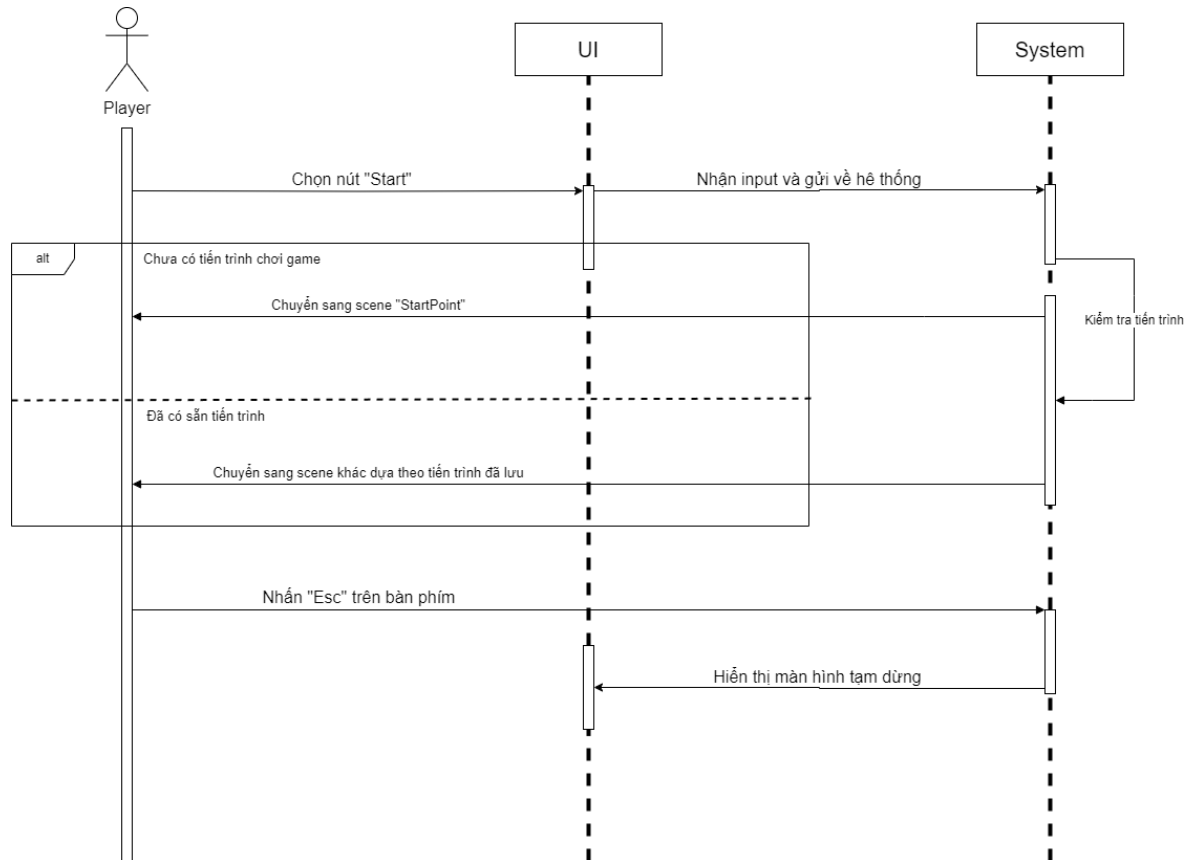
Hình 3.3. Sơ đồ tuần tự cài đặt

3.3.3. Thoát game



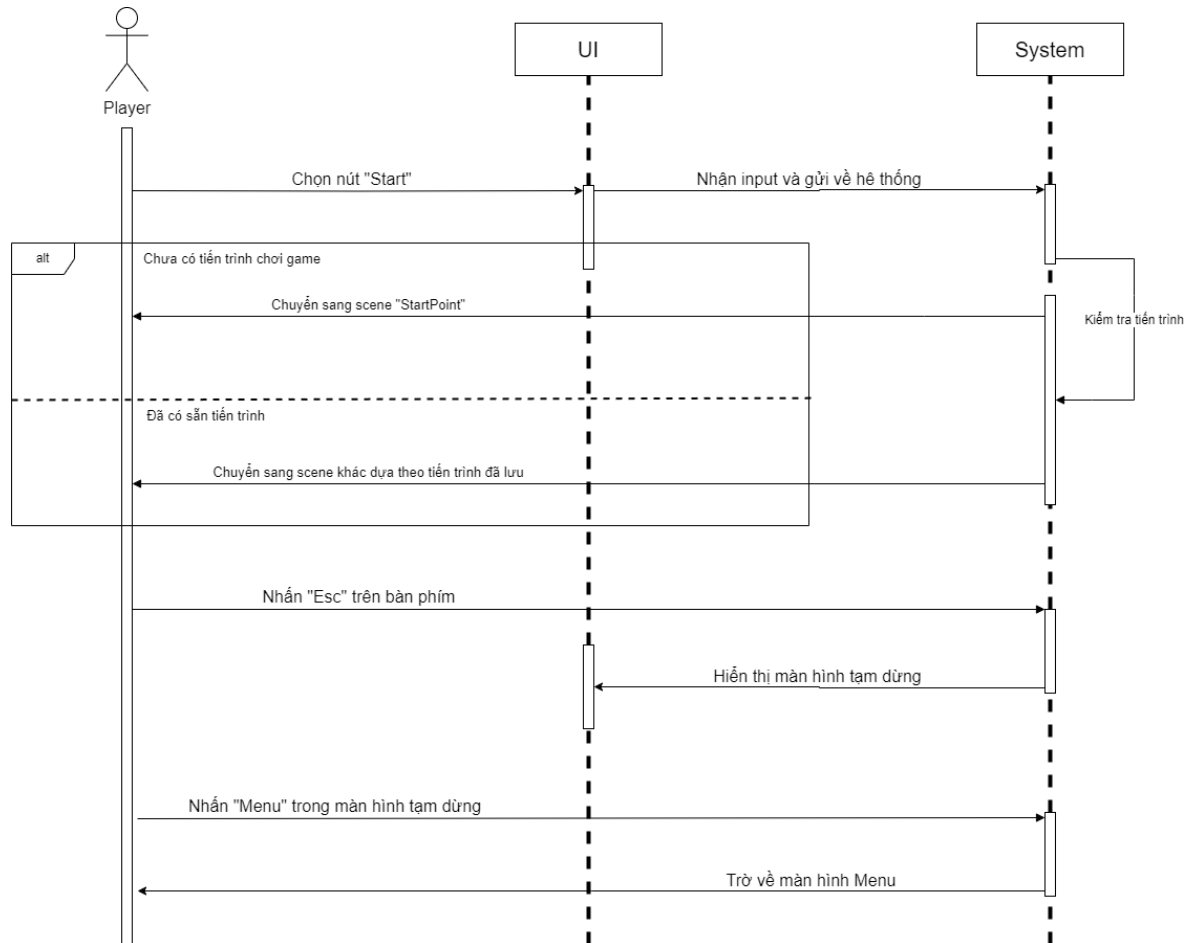
Hình 3.4. Sơ đồ tuần tự thoát game

3.3.4. Tạm dừng game



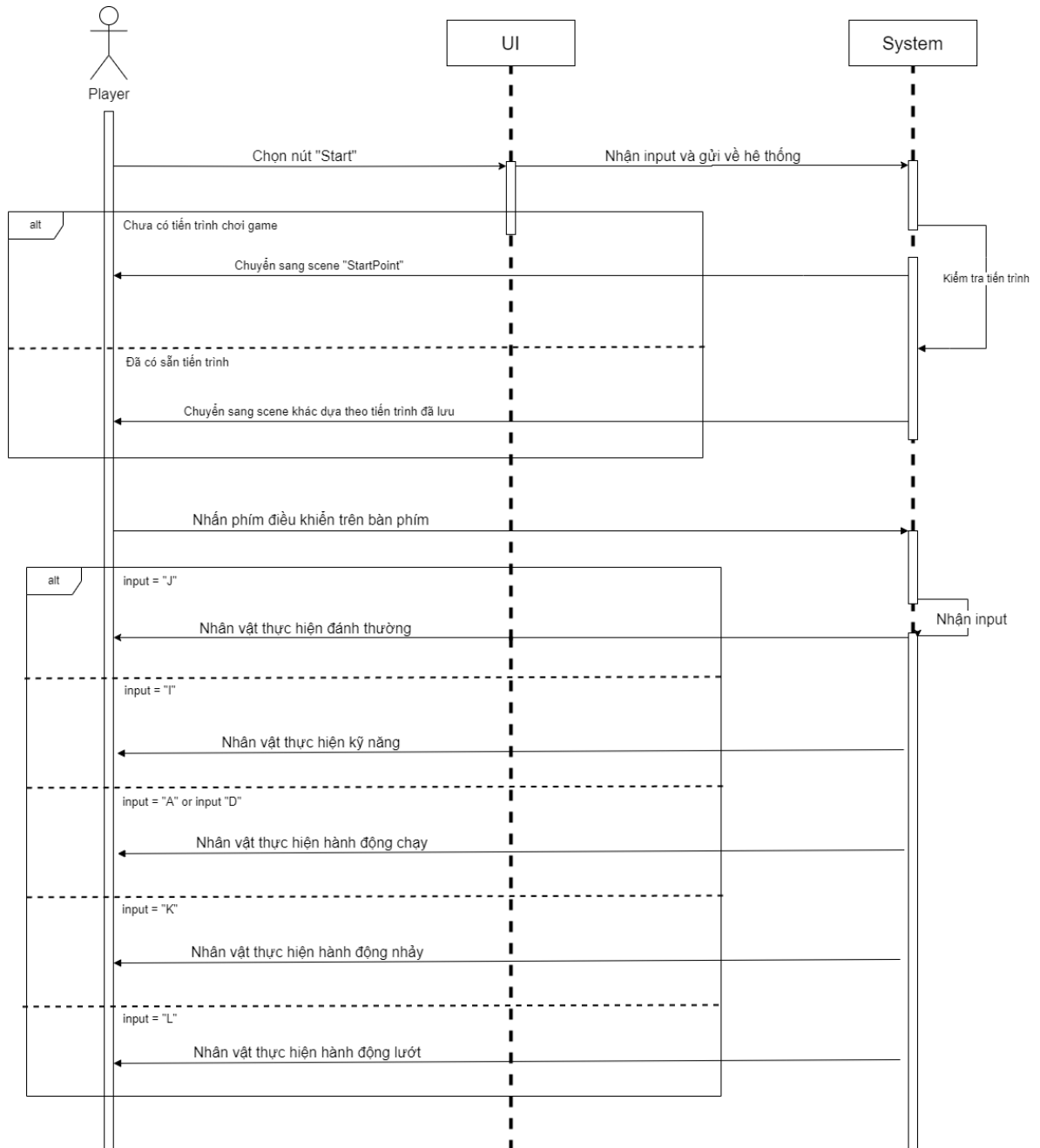
Hình 3.5. Sơ đồ tuần tự tạm dừng game

3.3.5. Trở về menu



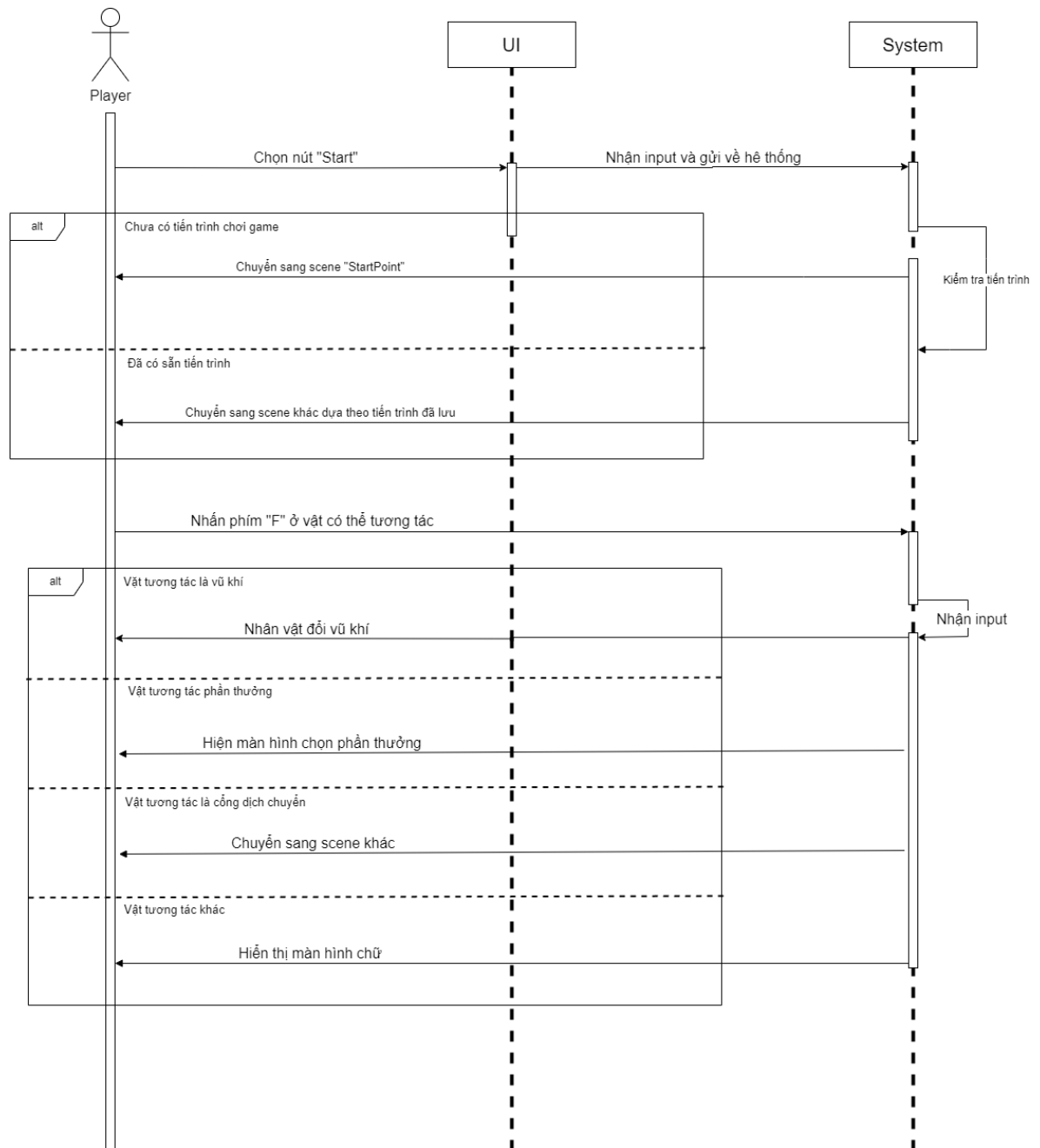
Hình 3.6. Sơ đồ tuần tự trở về menu

3.3.6. Điều khiển nhân vật



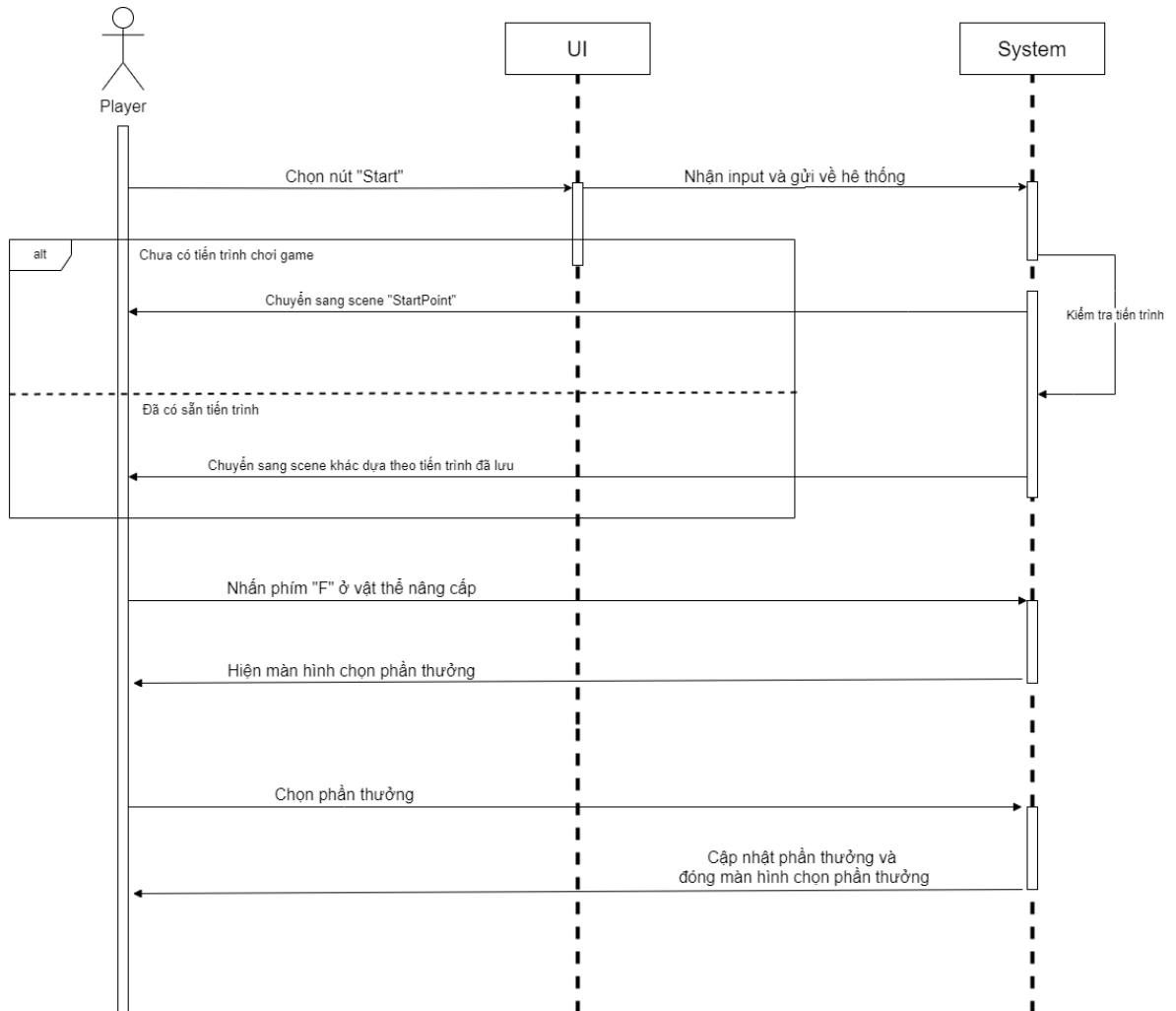
Hình 3.7. Sơ đồ tuần tự điều khiển nhân vật

3.3.7. Tương tác môi trường



Hình 3.8. Sơ đồ tuần tự tương tác môi trường

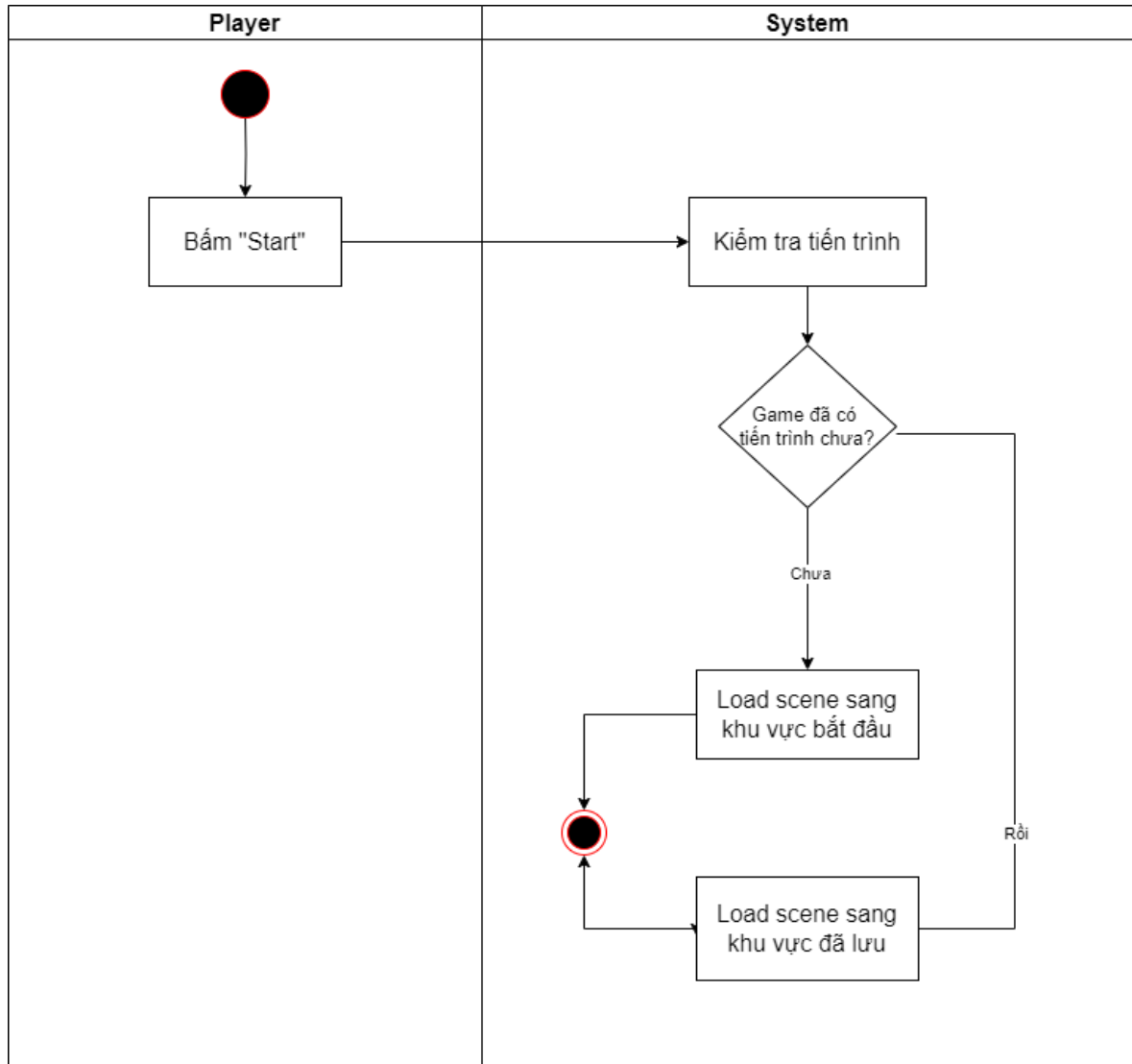
3.3.8. Nâng cấp nhân vật



Hình 3.9. Sơ đồ tuần tự nâng cấp nhân vật

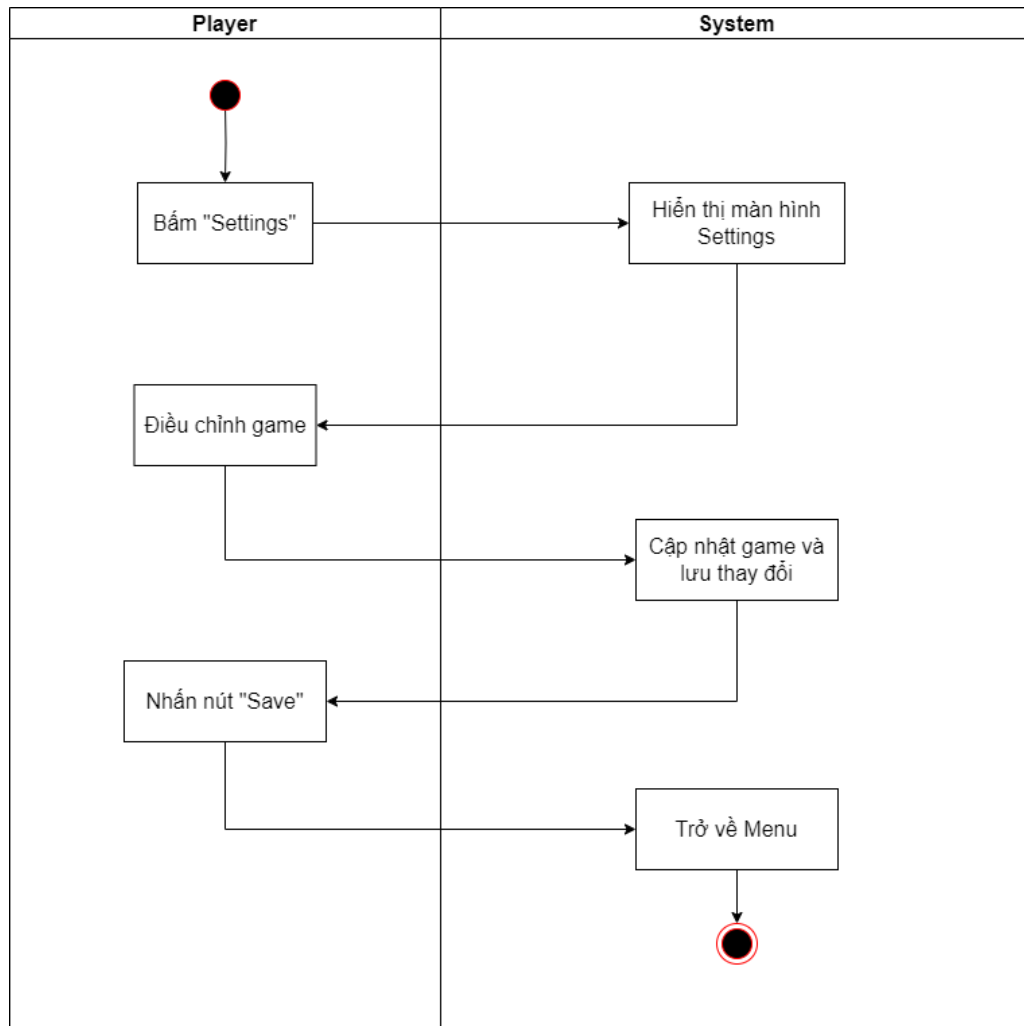
3.4 Sơ đồ Hoạt động

3.4.1. Bắt đầu game



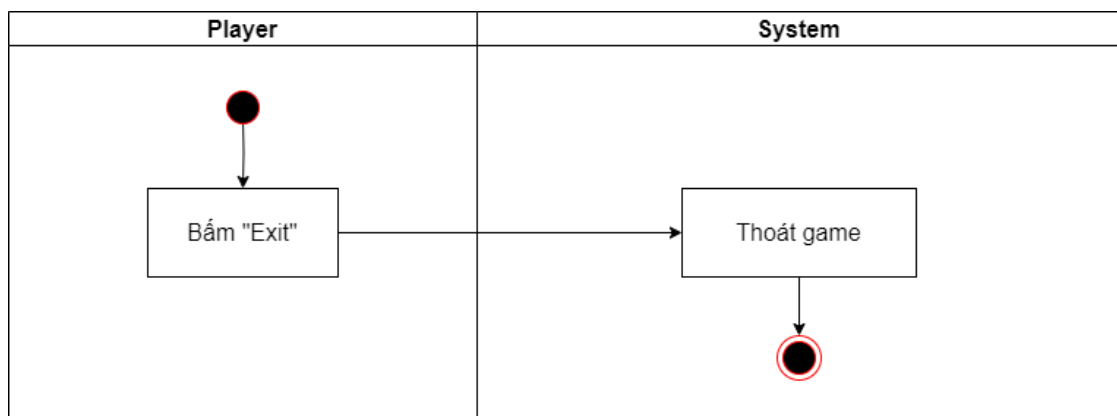
Hình 3.10. Sơ đồ hoạt động bắt đầu game

3.4.2. Cài đặt



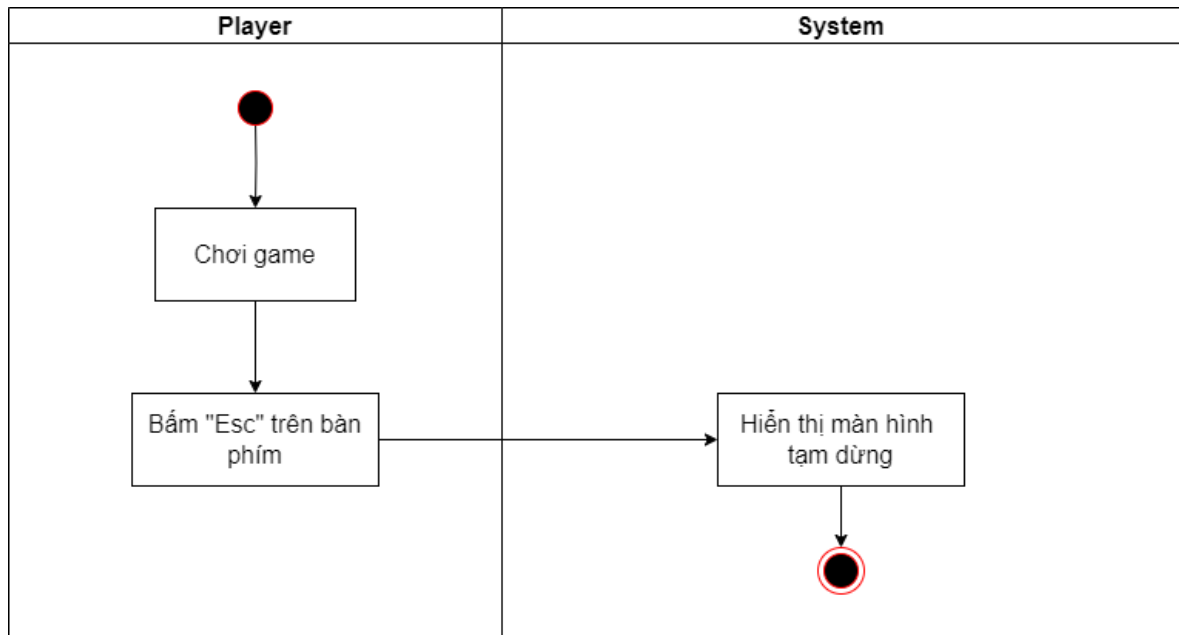
Hình 3.11. Sơ đồ hoạt động cài đặt

3.4.3. Thoát game



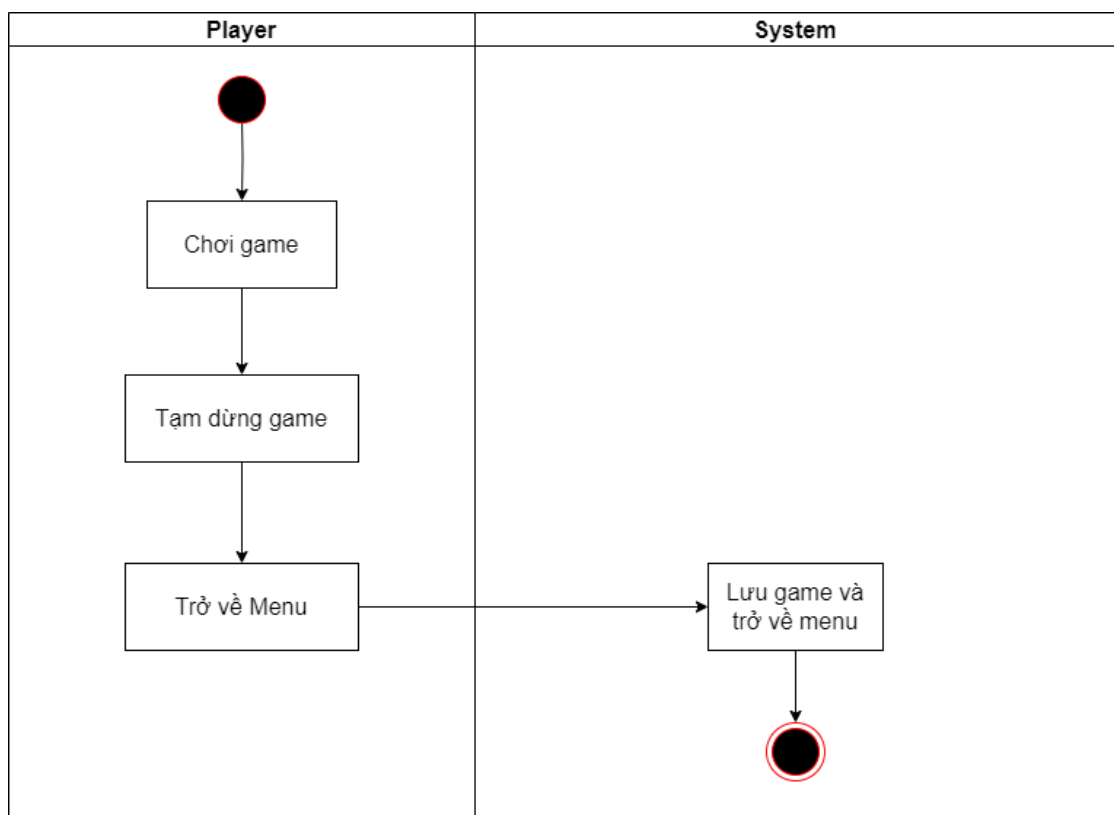
Hình 3.12. Sơ đồ hoạt động thoát game

3.4.4. Tạm dừng game



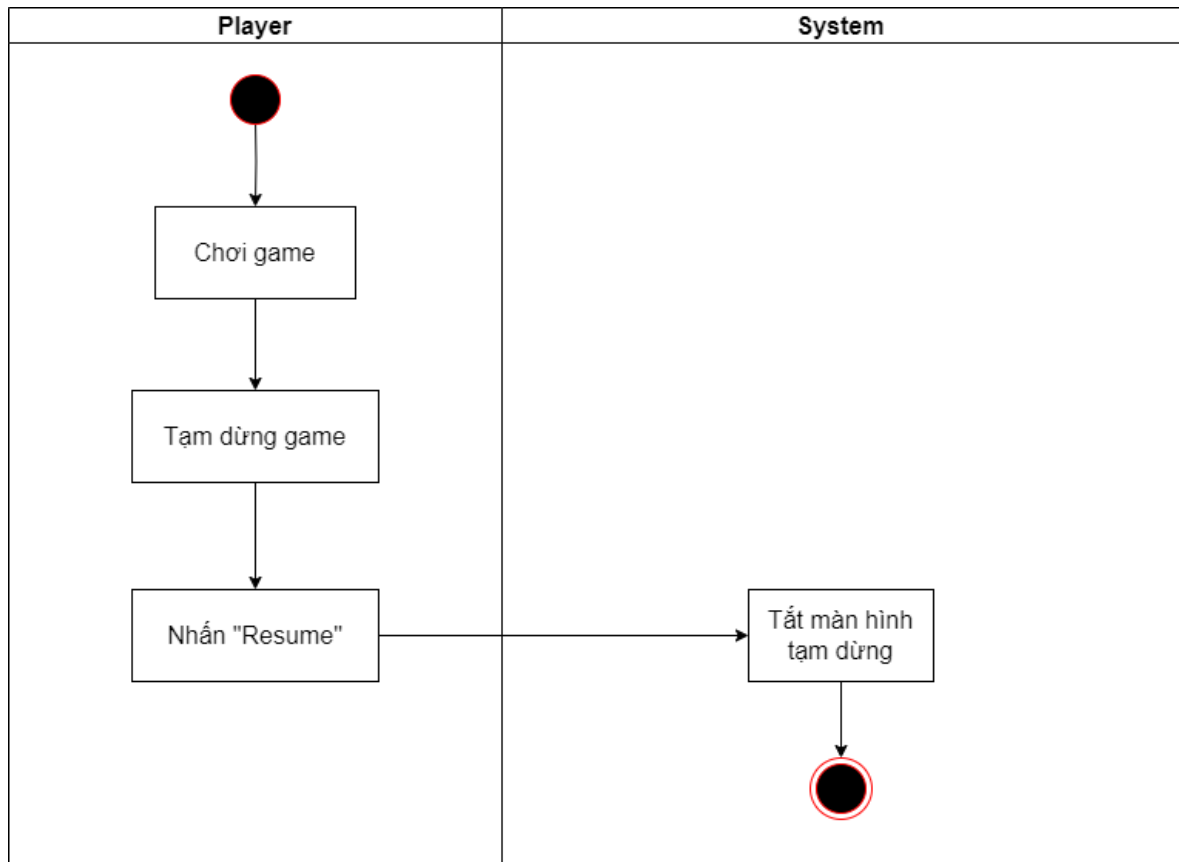
Hình 3.13. Sơ đồ hoạt động tạm dừng game

3.4.5. Trở về menu



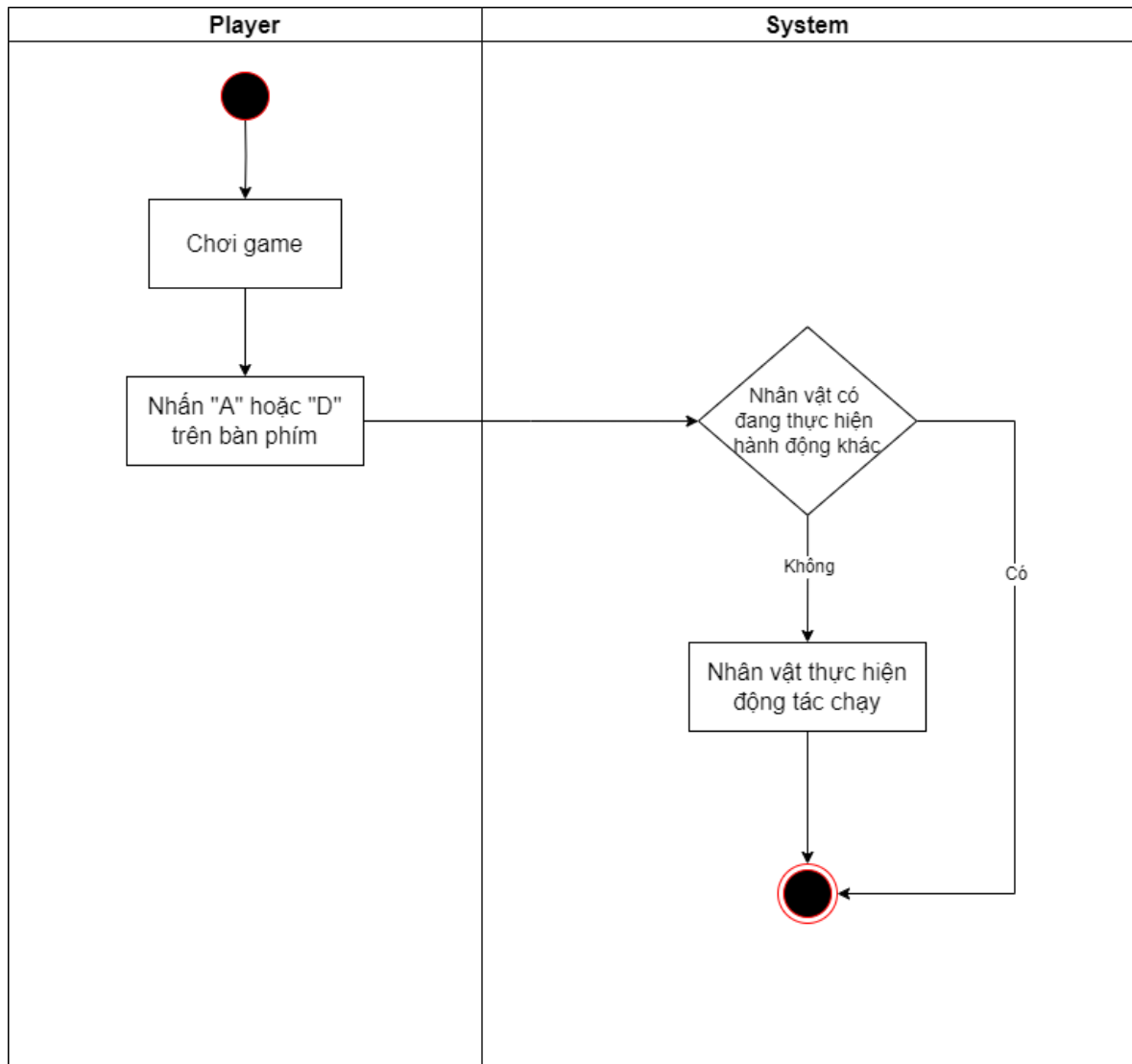
Hình 3.14 Sơ đồ hoạt động trở về menu

3.4.6. Tiếp tục game



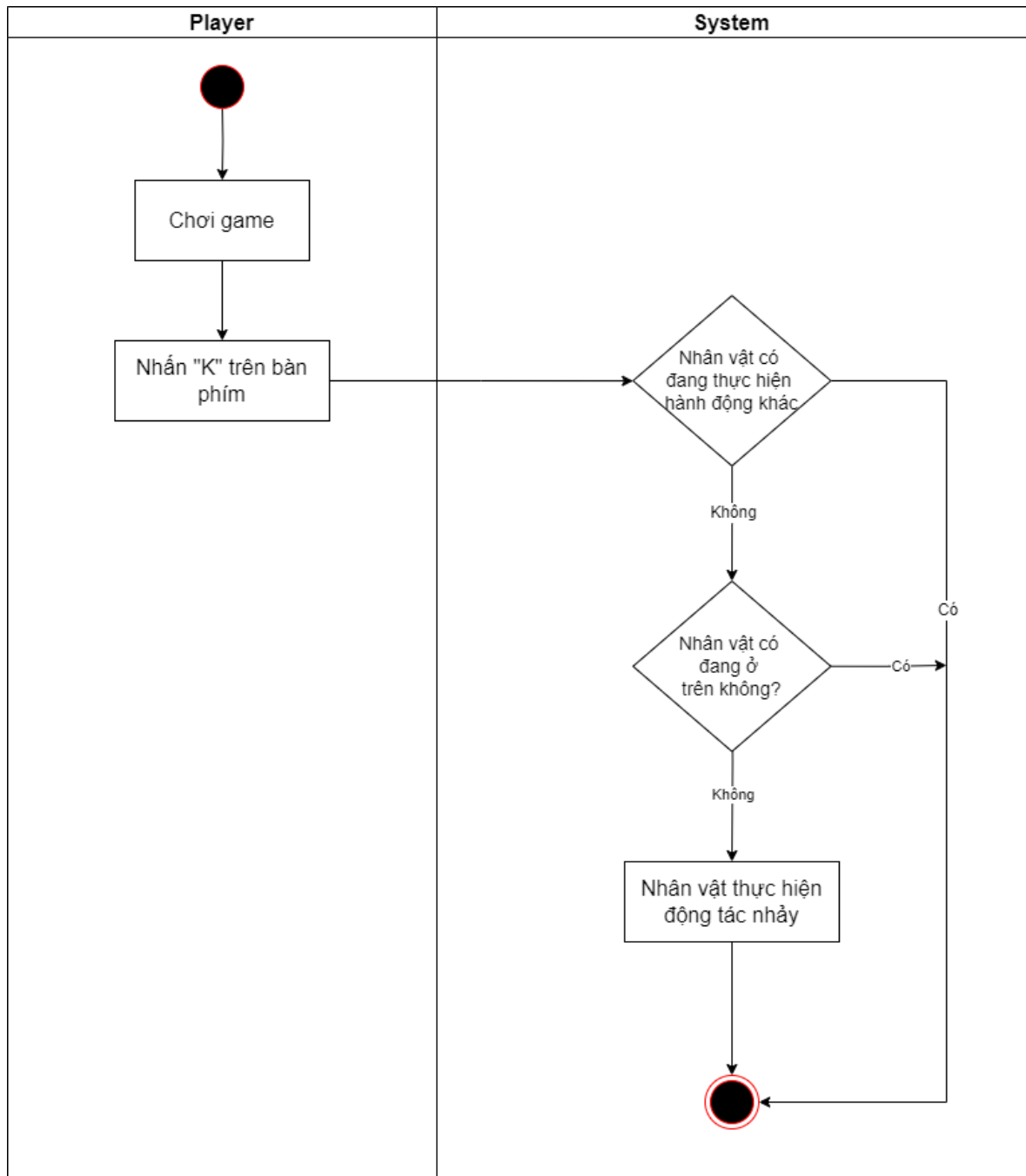
Hình 3.15 Sơ đồ hoạt động tiếp tục game

3.4.7. Nhân vật di chuyển



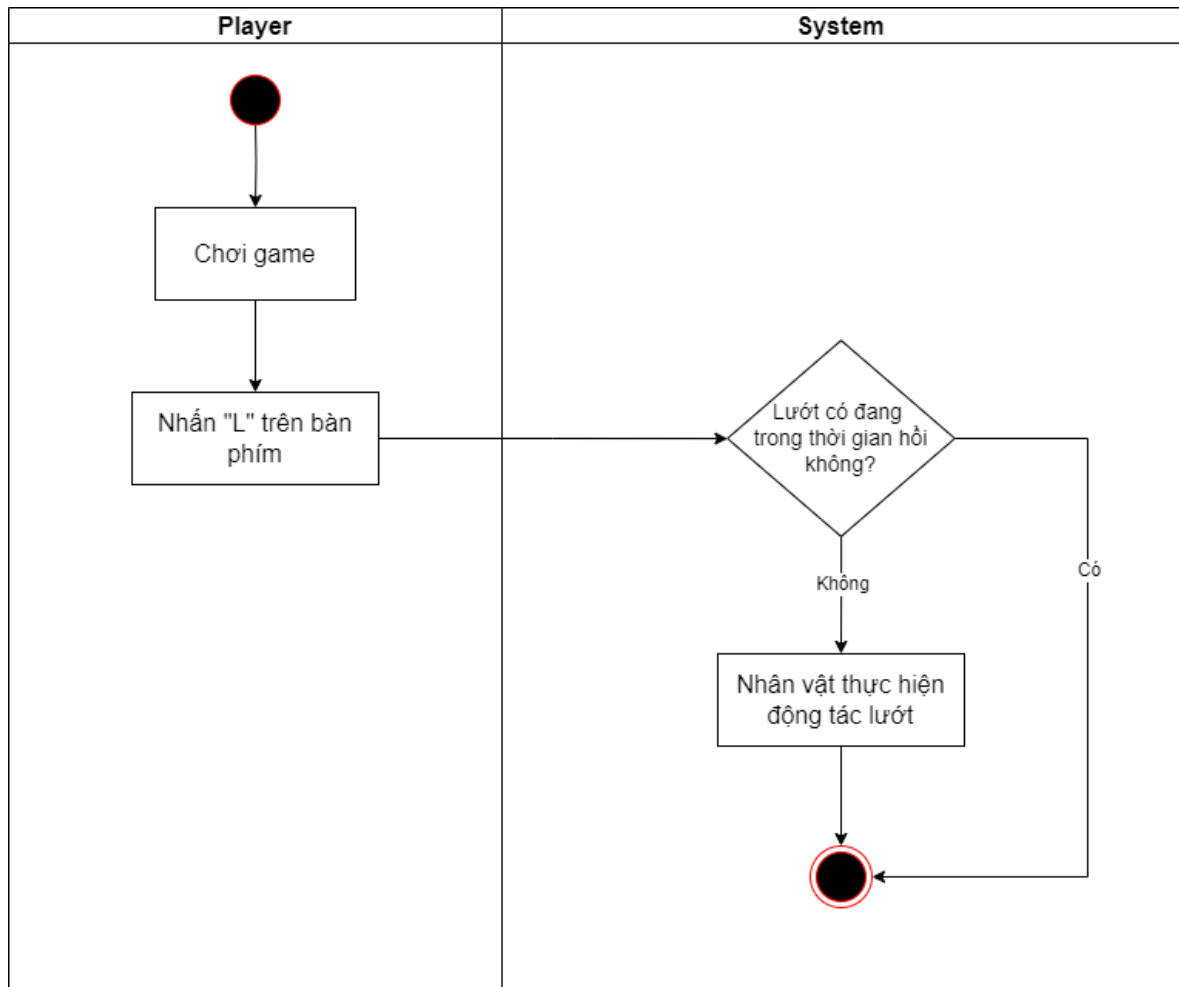
Hình 3.16. Sơ đồ hoạt động nhân vật di chuyển

3.4.8. Nhân vật nhảy



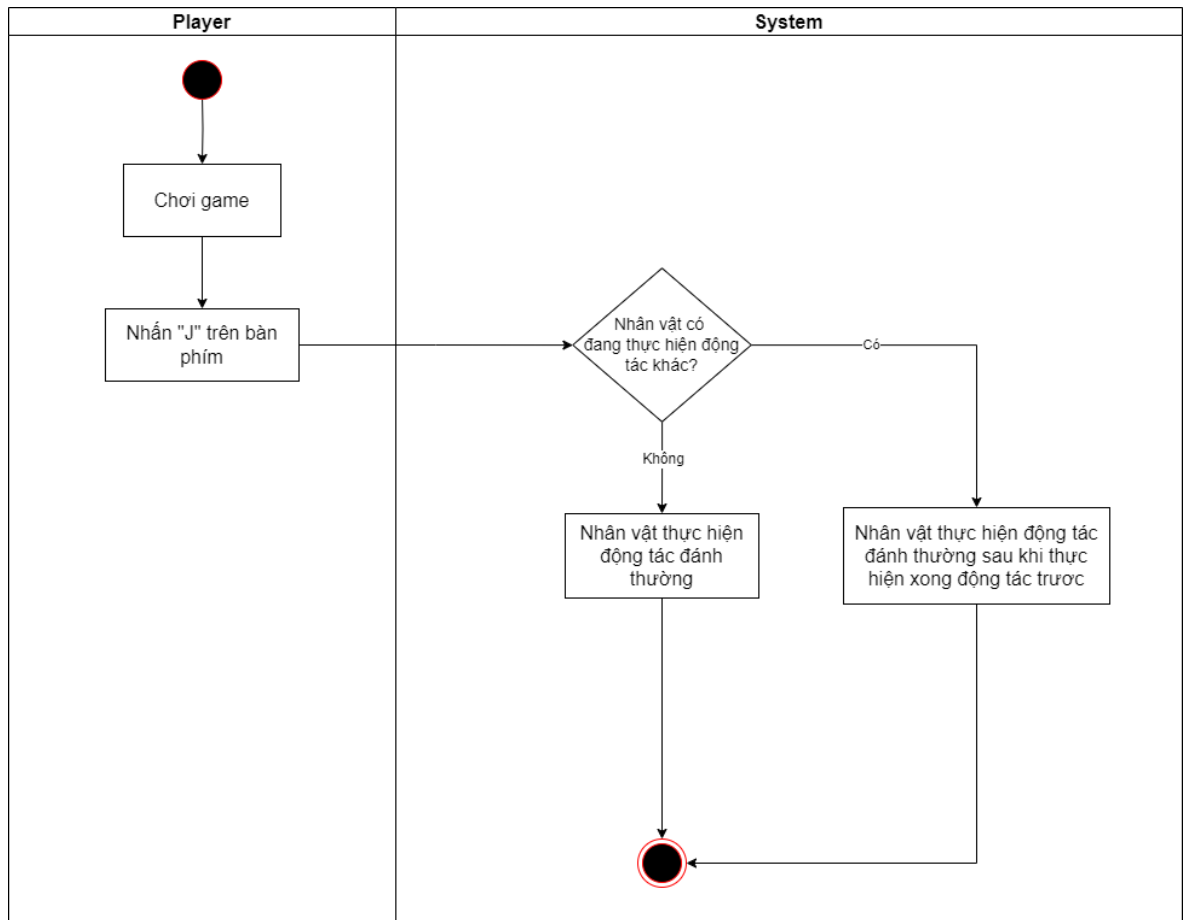
Hình 3.17. Sơ đồ hoạt động nhân vật nhảy

3.4.9. Nhân vật lượt



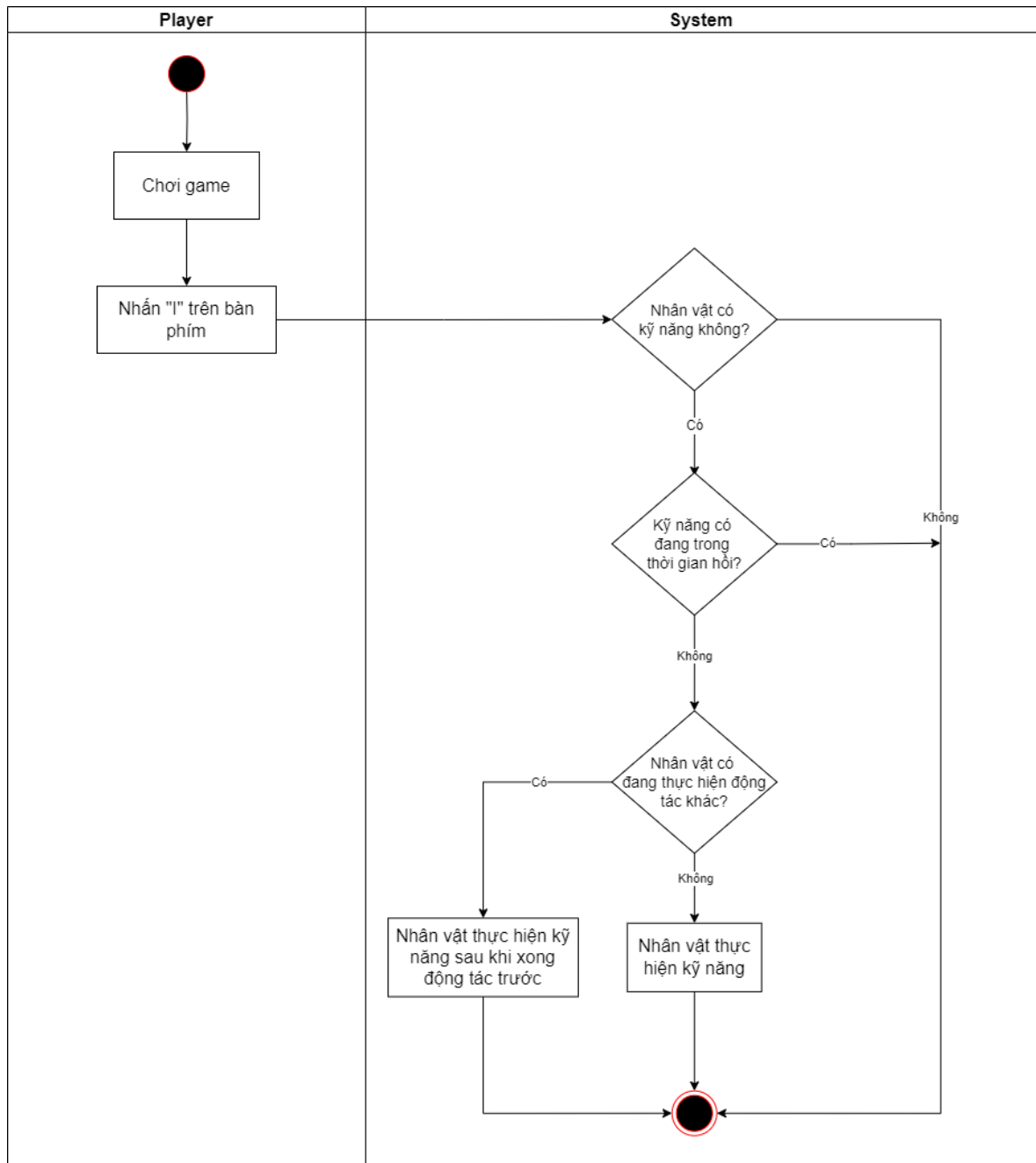
Hình 3.18. Sơ đồ hoạt động nhân vật lượt

3.4.10. Nhân vật đánh thưởng



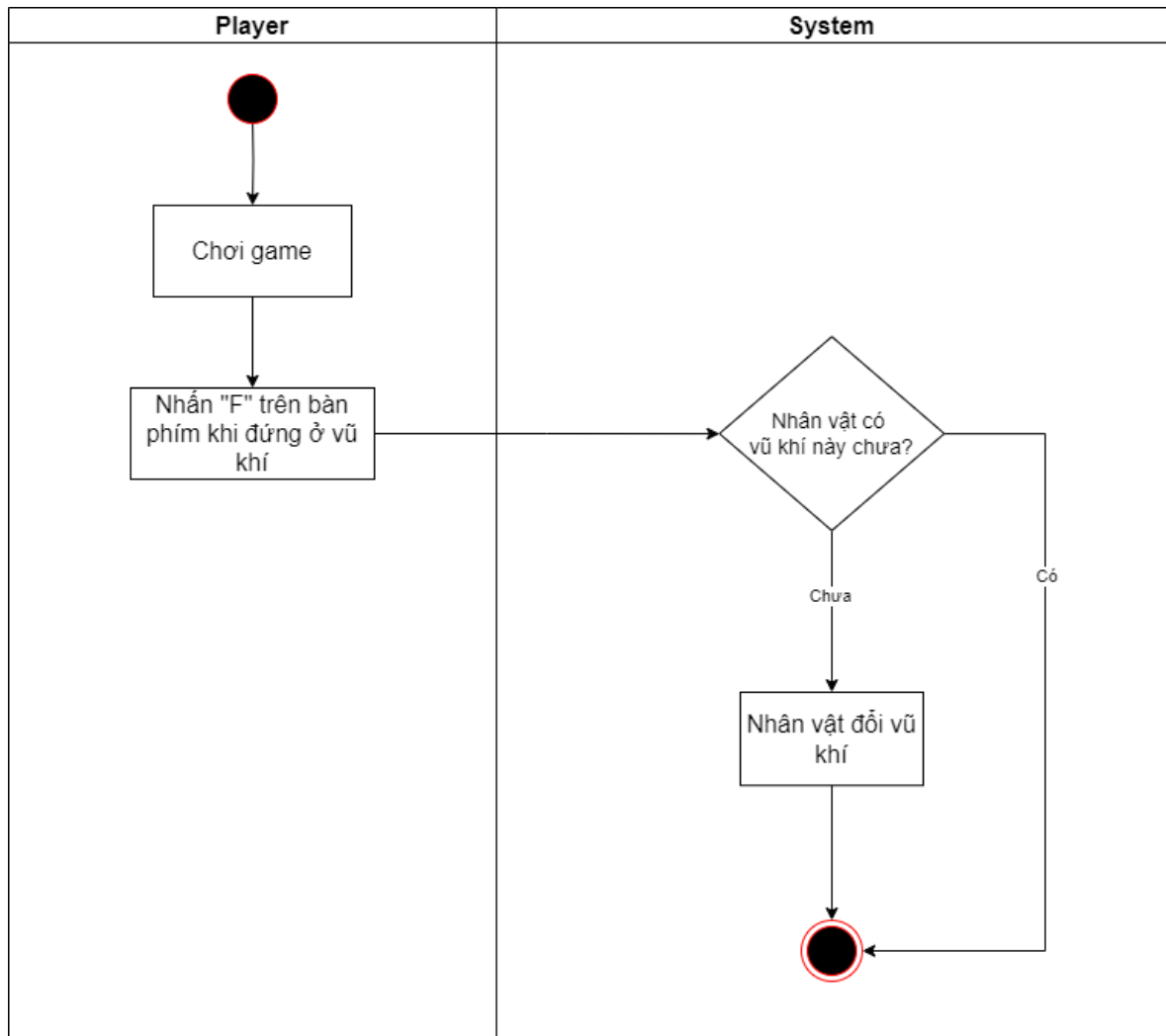
Hình 3.19. Sơ đồ hoạt động nhân vật đánh thưởng

3.4.11. Nhân vật dùng kỹ năng



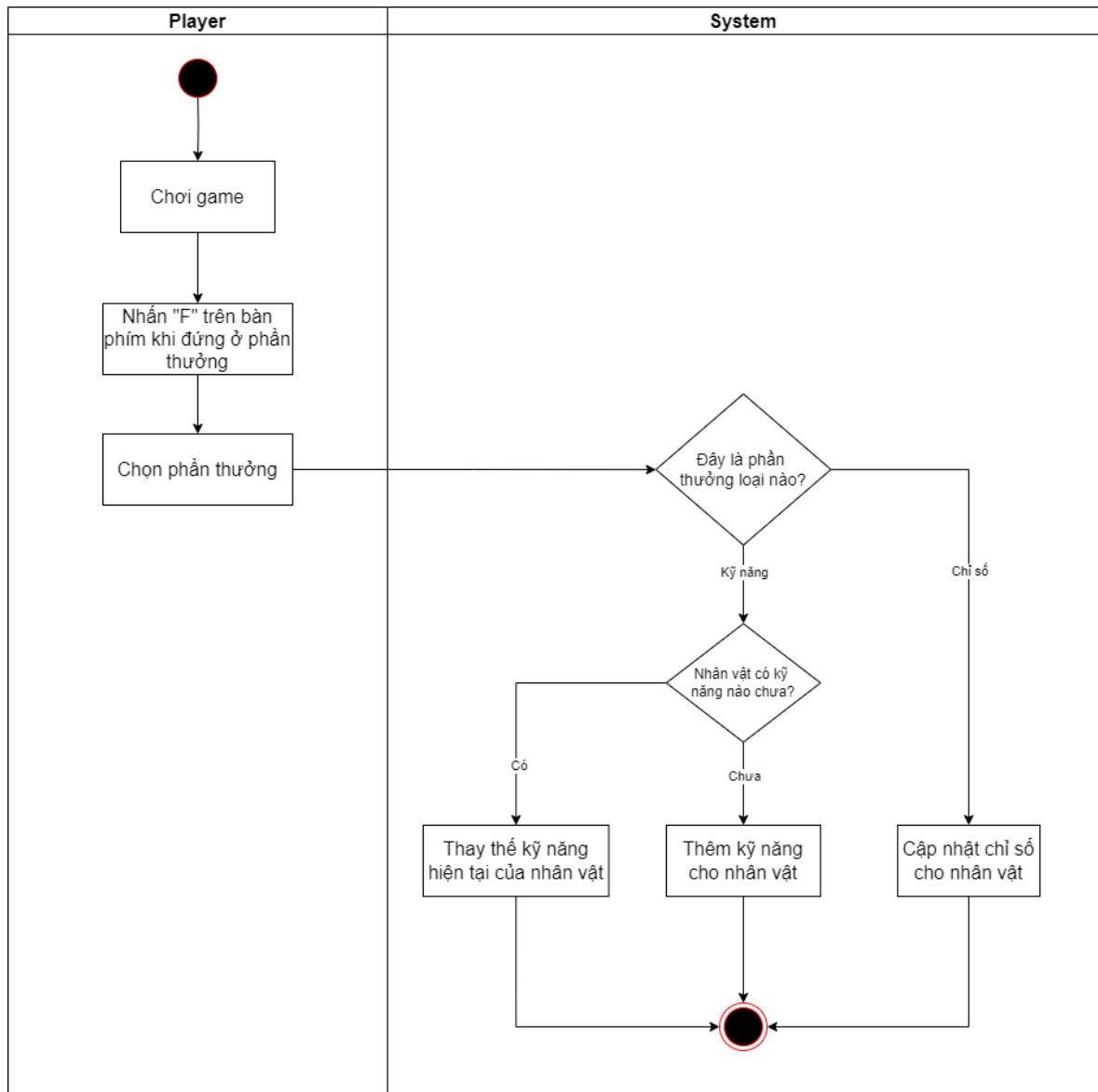
Hình 3.20. Sơ đồ hoạt động nhân vật dùng kỹ năng

3.4.12. Đổi vũ khí



Hình 3.21. Sơ đồ hoạt động đổi vũ khí

3.4.13. Nâng cấp nhân vật



Hình 3.22. Sơ đồ hoạt động nâng cấp nhân vật

CHƯƠNG 4. HIỆN THỰC HỆ THỐNG

4.1 Unity2D

Trong Unity2D gồm 2 loại dự án đó chính là Unity2D và Unity2D URP (Universal Render Pipeline). Khác với một dự án game 2D thông thường, dự án URP cung cấp cho người dùng hiệu suất và linh hoạt hơn trong việc render đồ họa, đặc biệt là trên nền tảng di động và máy tính cá nhân. URP cung cấp các tính năng và hiệu suất tốt hơn cho việc render đồ họa 2D, bao gồm hỗ trợ đa nền tảng, cải thiện về hiệu suất, và các hiệu ứng đồ họa nâng cao.

Tuy nhiên, việc sử dụng Unity 2D URP không đảm bảo rằng mọi dự án game sẽ chạy tốt trên mọi thiết bị yếu. Cho nên dự án chúng em sẽ tiếp cận tới Unity2D thông thường nhằm đảm bảo hiệu suất trong quá trình làm việc cũng như sử dụng trên máy tính cá nhân.

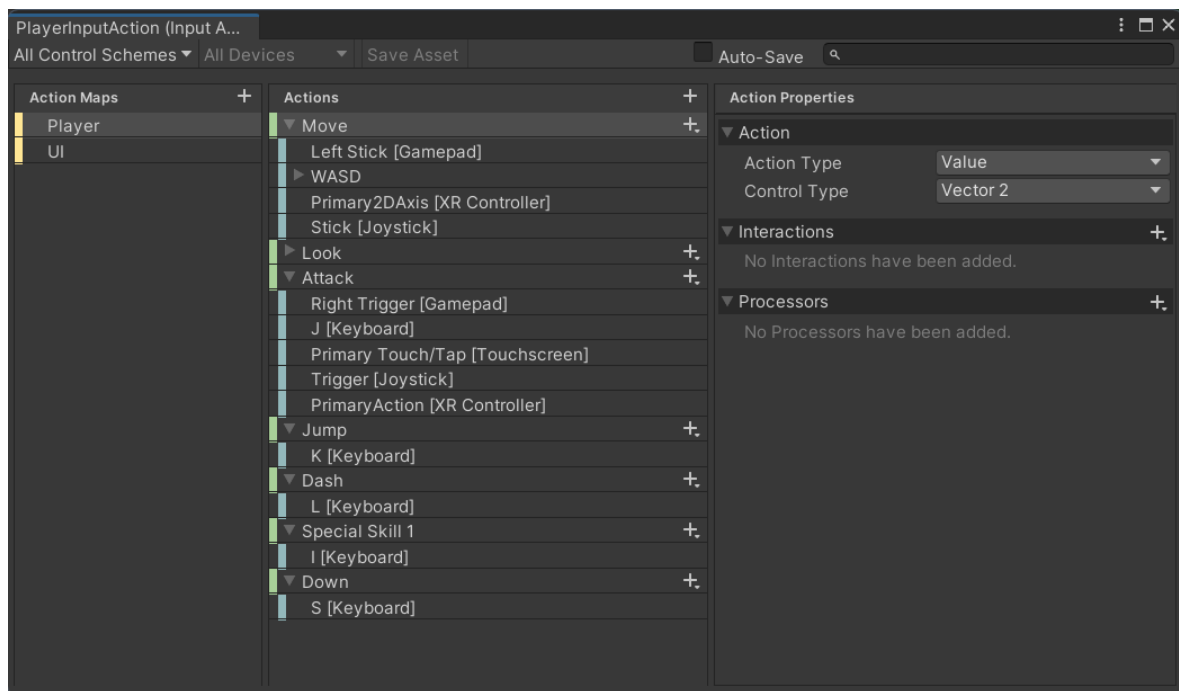
4.2 Nhân vật chính

4.2.1 Các components

Nhân vật chính được lấy ý tưởng dựa trên sprite có sẵn và được chỉnh sửa lại sao cho phù hợp với game hơn. Hình ảnh của người chơi được tạo bởi thành phần `SpriteRenderer` cho phép gắn hình ảnh vào đối tượng.

Để tạo nên tương tác vật lý và các môi trường xung quanh, thì hai thành phần `Rigidbody2D` và `Collider2D` đóng vai trò quan trọng trong việc thiết lập tương tác vật lý 2D với môi trường xung quanh.

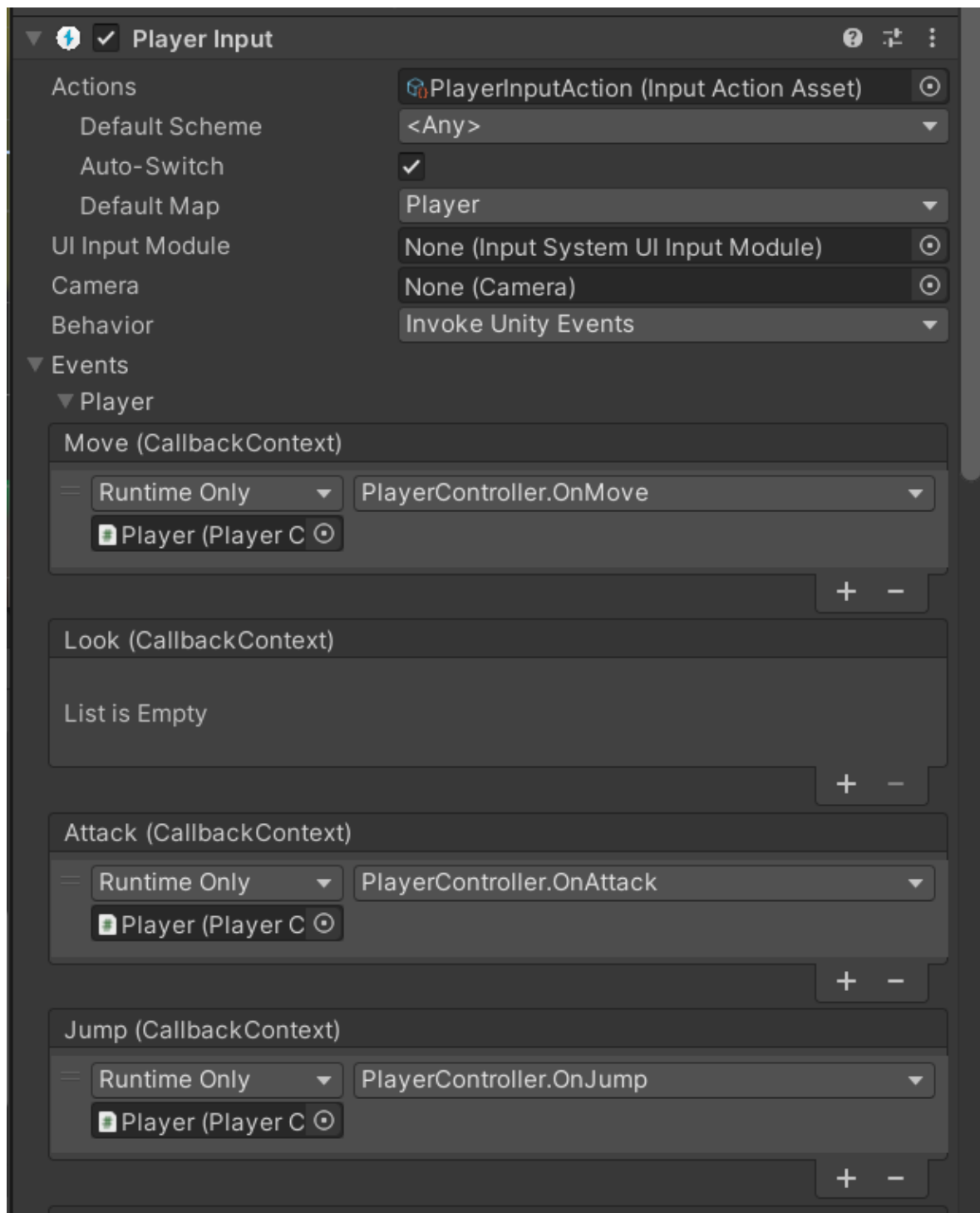
Về di chuyển, dự án sẽ sử dụng `InputSystem` cho phép đối tượng bắt sự kiện Input từ bàn phím và từ các sự kiện đó sẽ gọi những hàm nào.



Hình 4.1. InputSystem cho nhân vật

Các nút di chuyển chính của người chơi là A – di chuyển bên trái, D – di chuyển bên phải, S – di chuyển xuống các platform, J – thực hiện tấn công, K – thực hiện nhảy, L – thực hiện lướt và I – dùng để thi triển kỹ năng đặc biệt (nếu có).

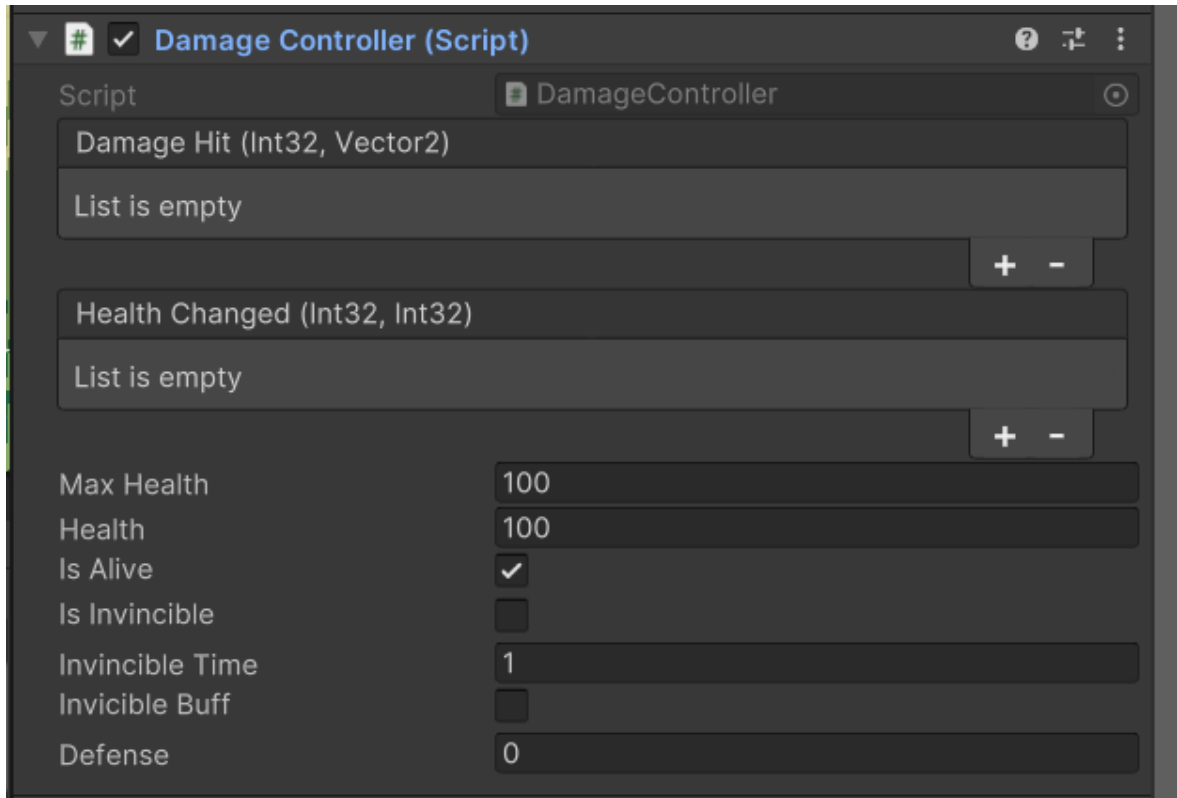
Các sự kiện trên sẽ có trên thành phần Player Input của người chơi và các sự kiện nhấn nút sẽ gọi nhưng hàm nào bên trong Script của người chơi.



Hình 4.2. Player Input

Để tạo ra hoạt ảnh của người chơi thì cần thành phần Animator cho phép tạo các hoạt ảnh của người chơi bằng sprite hình ảnh chuyển động của người chơi.

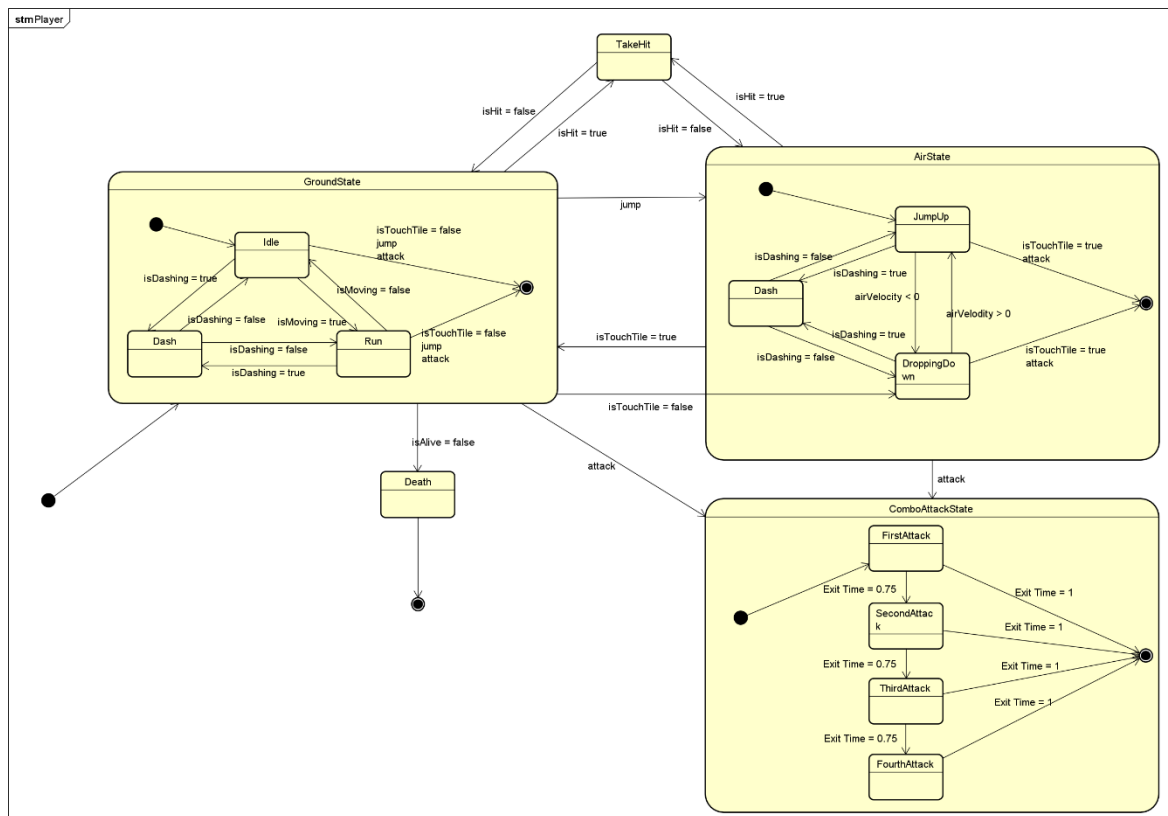
Để quản lý lượng máu của người chơi thì ta cần có một script để quản lý nó, Script DamageController sẽ quản lý lượng máu của người chơi, bật bắt tử cho người chơi, có thể hồi phục và trừ máu thông qua các hàm Hit và Heal. Script này sẽ được sử dụng chung cho các đối tượng có sự sống. Sử dụng các Unity Event để có thể tạo các sự kiện hồi máu, mất máu để UI thanh máu có thể bắt các sự kiện đó.



Hình 4.3. Damage Controller

4.2.2 Animation

Finite State Machine (FSM), một mô hình được sử dụng để quản lý trạng thái của các đối tượng trong trò chơi. FSM là một biểu đồ trạng thái, trong đó mỗi trạng thái đại diện cho một trạng thái cụ thể mà đối tượng có thể ở vào thời điểm nào đó. Các sự kiện và điều kiện có thể chuyển đổi đối tượng từ trạng thái này sang trạng thái khác.



Hình 4.4. FSM của nhân vật chính

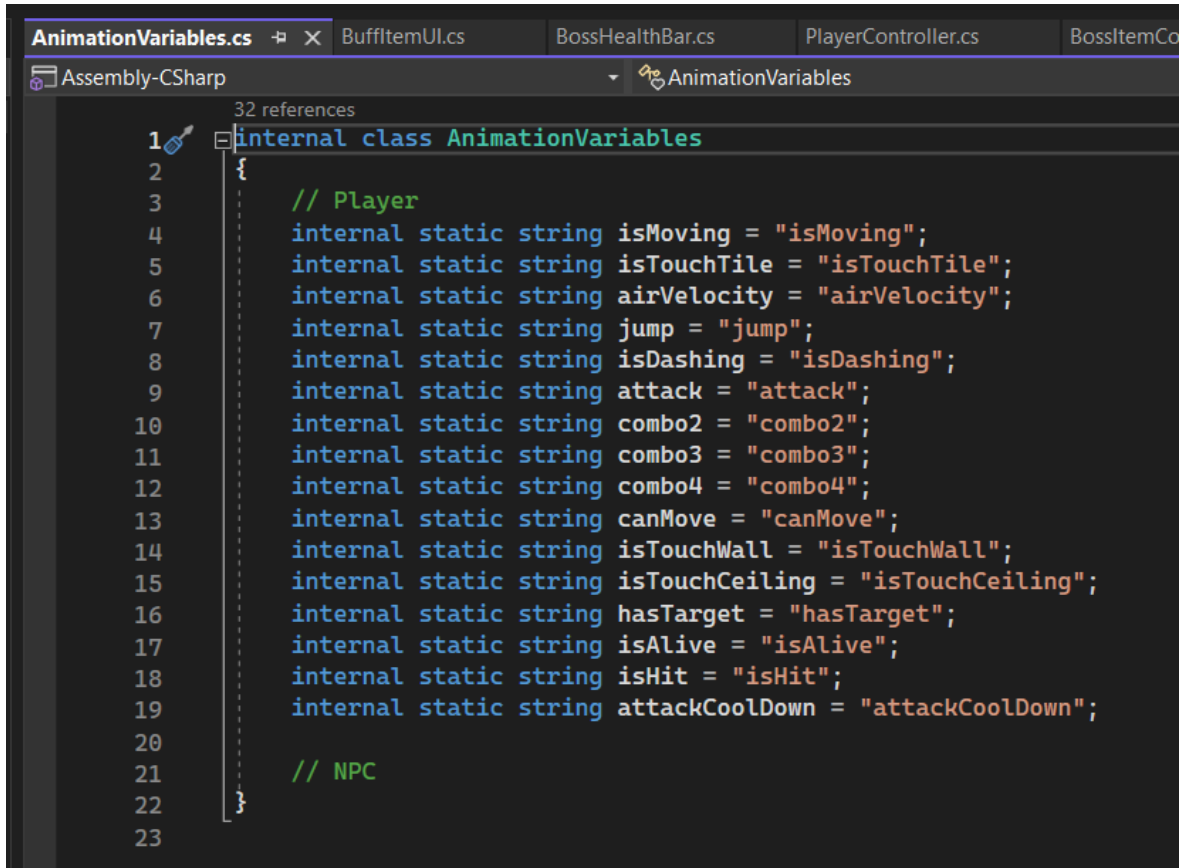
Người chơi có các 3 Animator Controller tương ứng với 3 loại vũ khí là kiếm, thương và rìu, Trong các Animator Controller của người chơi có chung các hoạt ảnh sau tuy nhiên sẽ có một vài hình ảnh tương ứng với vũ khí đó.

- Idle: hoạt ảnh nhân rồi của người chơi khi không di chuyển
- Walk: hoạt ảnh di chuyển
- Dash: hoạt ảnh lướt, người chơi sẽ xoay người dưới đất hoặc trên không
- JumpUp: hoạt ảnh người chơi đang nhảy lên
- DroppingDown: hoạt ảnh người chơi đang rơi xuống
- ComboAttack: gồm 4 hoạt ảnh tấn công (mỗi loại vũ khí đều có 4 hoạt ảnh)
- TakeHit: hoạt ảnh người chơi bị tấn công
- Death: hoạt ảnh người chơi bị hạ gục

Mỗi trạng thái sẽ là mỗi hình ảnh khác nhau, các trạng thái sẽ tương tác thông qua các transition, các transition này hoạt động thông qua các điều kiện của các

parameter như boolean (isMoving, isDashing, ...), float (airVelocity) và các trigger (attack, jump).

Để dễ dàng quản lý các parameter trong quá trình gọi trong các script thì ta cần một script để quản lý các parameter đó thông qua chuỗi.



Hình 4.5. Script quản lý animation

4.2.3 Điều khiển người chơi

Input System là một package vừa mới ra mắt vào năm 2023, package này cho phép dự án game có thể điều khiển trên bất cứ thiết bị nào (dùng bàn phím, chạm màn hình, tay cầm, ...). Tuy nhiên dự án này sẽ chỉ tiếp cận trên máy tính.

Các sự kiện nhấn nút sẽ gọi các hàm bên trong script chính của nhân vật là PlayerController. Ví dụ khi ta nhấn các phím WASD thì thành phần Player Input sẽ bắt sự kiện vào gọi tới hàm OnMove của người chơi.

```

0 references
public void OnMove(InputAction.CallbackContext context)
{
    moveInput = context.ReadValue<Vector2>();

    if (IsAlive)
    {
        IsMoving = moveInput != Vector2.zero;
        soundManager.PlayWalkSound();
        SetFacingDirection(moveInput);
    }
    else
    {
        IsMoving = false;
    }
}

1 reference
private void SetFacingDirection(Vector2 moveInput)
{
    if (moveInput.x > 0 && !IsFacingRight)
    {
        IsFacingRight = true;
    }
    else if (moveInput.x < 0 && IsFacingRight)
    {
        IsFacingRight = false;
    }
}

```

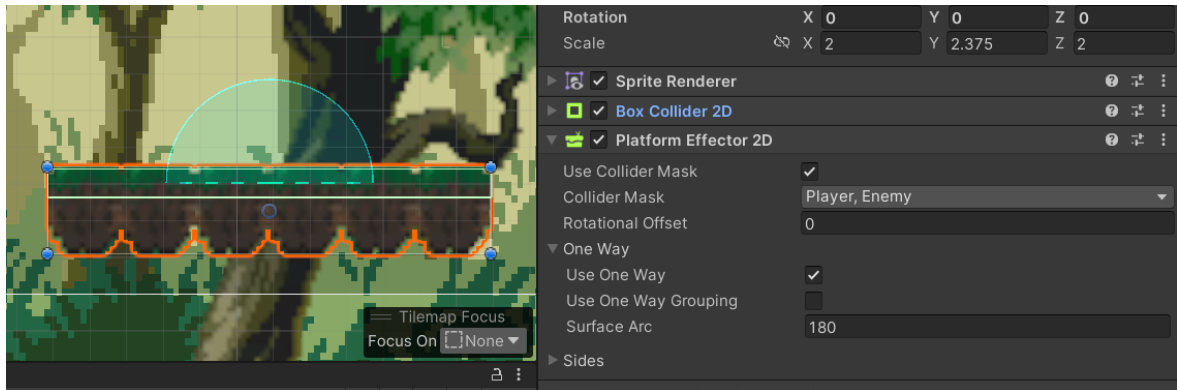
Hình 4.6. Script điều khiển người chơi

Điều này đảm bảo được hiệu suất của dự án khi không cần phải lập trình bên trong hàm Update được gọi sau mỗi frame để bắt sự kiện nhập từ bàn phím.

Hai sự kiện đặc biệt góp phần tăng trải nghiệm chiến đấu của người chơi là OnDown và OnDash.

OnDown là sự kiện mà người chơi sẽ sử dụng RaycastHit2D và ContactFilter2D để kiểm tra dưới chân có va chạm với collider nào có tag là Platform (là các thành phần One way Platform chỉ cho người chơi nhảy xuyên qua và đứng

trên platform đó) sau đó sẽ bỏ qua va chạm với collider đó thông qua hàm IgnoreCollision của Physics2D.



Hình 4.7. One Way Platform

```
0 references
public void OnDown(InputAction.CallbackContext context)
{
    if (context.performed)
    {
        RaycastHit2D[] hits = new RaycastHit2D[1];
        col.Cast(Vector2.down, contactFilter, hits, 0.05f);

        int hitCount = col.Cast(Vector2.down, contactFilter, hits, 0.05f);

        for (int i = 0; i < hitCount; i++)
        {
            Debug.Log("Object hit: " + hits[i].collider.gameObject.name);
        }

        if (hitCount > 0 && hits[0].collider.CompareTag("Platform"))
        {
            Debug.Log("Success!");

            Physics2D.IgnoreCollision(col, hits[0].collider, true);

            StartCoroutine(ResetCollisionAfterDelay(hits[0].collider));
        }
    }
}

1 reference
private IEnumerator ResetCollisionAfterDelay(Collider2D colliderToReset)
{
    yield return new WaitForSeconds(0.5f);

    Physics2D.IgnoreCollision(col, colliderToReset, false);
}
```

Hình 4.8. Script OnDown cho phép nhảy xuống platform

Hàm OnDash sẽ cho phép người chơi lướt với một khoảng cách nhất định, quá trình lướt này sẽ sử dụng Coroutine để xử lý, khi Dash thì người chơi sẽ bắt đầu trong một khoảng thời gian, animation Dash sẽ kích hoạt và vận tốc theo trục x của người chơi được nhân lên với dashPower để người chơi dịch chuyển một khoảng. Cú pháp “yield return new WaitForSeconds()” trong Coroutine sẽ chờ một khoảng thời gian dựa trên số thực đầu vào thì mới thực hiện các dòng code kế tiếp.

```

0 references
public void OnDash(InputAction.CallbackContext context)
{
    if (context.performed && canDash)
    {
        Debug.Log("Dash conditions met, invoking Dash, Player: " + gameObject.activeInHierarchy + gameObject.activeSelf);
        soundManager.PlayDashSound();
        StartCoroutine(DashCoroutine());
    }
}

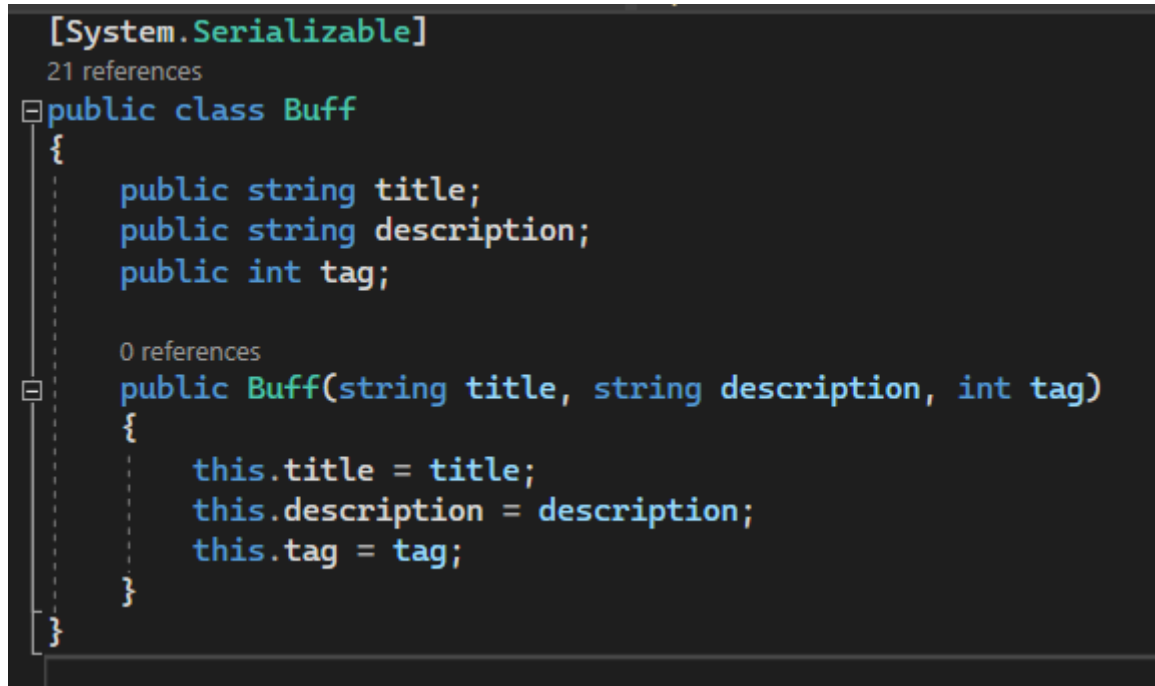
1 reference
private IEnumerator DashCoroutine()
{
    yield return Dash();
    yield return new WaitForSeconds(dashCooldown - 0.05f);
    canDash = true;
    yield return new WaitForSeconds(0.05f);
    canDash = true;
}

1 reference
private IEnumerator Dash()
{
    Debug.Log("Dash!");
    canDash = false;
    isDashing = true;
    damage.isInvincible = true;
    rb.velocity = new Vector2(transform.localScale.x * dashPower, 0);
    animator.SetBool(AnimationVariables.isDashing, isDashing);
    trailRenderer.emitting = true;
    yield return new WaitForSeconds(dashTime);
    trailRenderer.emitting = false;
    isDashing = false;
    damage.isInvincible = false;
    animator.SetBool(AnimationVariables.isDashing, isDashing);
}

```

Hình 4.9. OnDash điều khiển lướt

Các skill đặc biệt của người chơi sẽ được nhận thông qua các Buff kỹ năng. Dựa vào tag của Buff thì Player khi nhận buff sẽ biết được nên kích hoạt kỹ năng nào thông qua các biến boolean.

A screenshot of a code editor showing the implementation of a Buff class. The class is decorated with the [System.Serializable] attribute and has 21 references. It contains three public fields: title (string), description (string), and tag (int). There is also a public constructor Buff(string title, string description, int tag) that initializes these fields. The code is written in C# and uses a dark theme.

```
[System.Serializable]
21 references
public class Buff
{
    public string title;
    public string description;
    public int tag;

    0 references
    public Buff(string title, string description, int tag)
    {
        this.title = title;
        this.description = description;
        this.tag = tag;
    }
}
```

Hình 4.10. Buff kỹ năng

```

PlayerSoundManager soundManager;

[Header("Pyro Blade Skill")]
public GameObject skillPrefab;
public bool skillStatusOne = false;

[Header("Lightning Cloud Skill")]
public GameObject lightningSkillPrefab;
public bool skillStatusTwo = false;

[Header("Shield Skill")]
public GameObject shieldIcon;
public float shieldTime = 7f;
public bool skillStatusThree = false;

[Header("Heal Skill")]
public bool skillStatusFour = false;

[Header("Anemo Blade Skill")]
public GameObject anemoPrefab;
public bool skillStatusFive = false;

[Header("Freeze Skill")]
public GameObject freezeEffectPrefab;
public float freezeTime = 5f;
public bool skillStatusSix = false;

public float skillCooldown = 7f;
public float skillTime = 0f;
public bool canUseSkill = true;

```

Hình 4.11. Các kỹ năng của The Loop qua Script

Khi nhận buff thì ta sẽ gọi tới hàm ApplyBuff để kích hoạt kỹ năng đang nhận của người chơi, các buff về chỉ số sẽ được cộng dồn, tuy nhiên kỹ năng chỉ được sử dụng kỹ năng gần nhất mà người chơi nhận. Sự kiện khi nhấn I sẽ gọi hàm OnSpecialSkill, hàm này sẽ kiểm tra kỹ năng nào đang là true thì sẽ xử lý code kỹ năng của kỹ năng đó.


```

2 references
public void ApplyBuff(Buff buff)
{
    switch (buff.tag)
    {
        // TODO: Thêm các case cho các buff khác
        case 1:
            UpdateSkillStatus(true, false, false, false, false, false, 0f); // Pyro Blade
            break;
        case 2:
            UpdateSkillStatus(false, false, false, true, false, false, 0f); // Heal
            break;
        case 3:
            UpdateSkillStatus(false, true, false, false, false, false, 0f); // Lightning
            break;
        case 4:
            UpdateSkillStatus(false, false, false, false, false, true, freezeTime); // Freeze
            break;
        case 5:
            UpdateSkillStatus(false, false, true, false, false, false, shieldTime); // Shield
            break;
        case 6:
            UpdateSkillStatus(false, false, false, false, true, false, 0f); // Anemo Blade
            break;
        case 7:
            HealthBuff(20); // Hydro - Health
            break;
        case 8:
            AttackBuff(5); // Pyro - Attack
            break;
        case 9:
            DashBuff(0.1f); // Electro - Dash
            break;
        case 10:
            DefenseBuff(2); // Geo - Defense
            break;
        case 11:
            SkillCooldownBuff(0.5f); // Cryo - Skill
            break;
        case 12:
            WalkSpeed(1.5f); // Anemo - Speed
            break;
        default:
            Debug.Log("Buff tag not recognized");
            break;
    }
}

```

Hình 4.12 ApplyBuff

Ý tưởng cho các đòn tấn công của người chơi là trong hoạt ảnh tấn công của người chơi, khi tới đòn tấn công sẽ kích hoạt một collider là một trigger. AttackController là script cho phép khi đối tượng bị trigger bởi collider này thì sẽ gọi tới thành phần DamageController của đối tượng để trừ theo lượng sát thương nhất định.

```

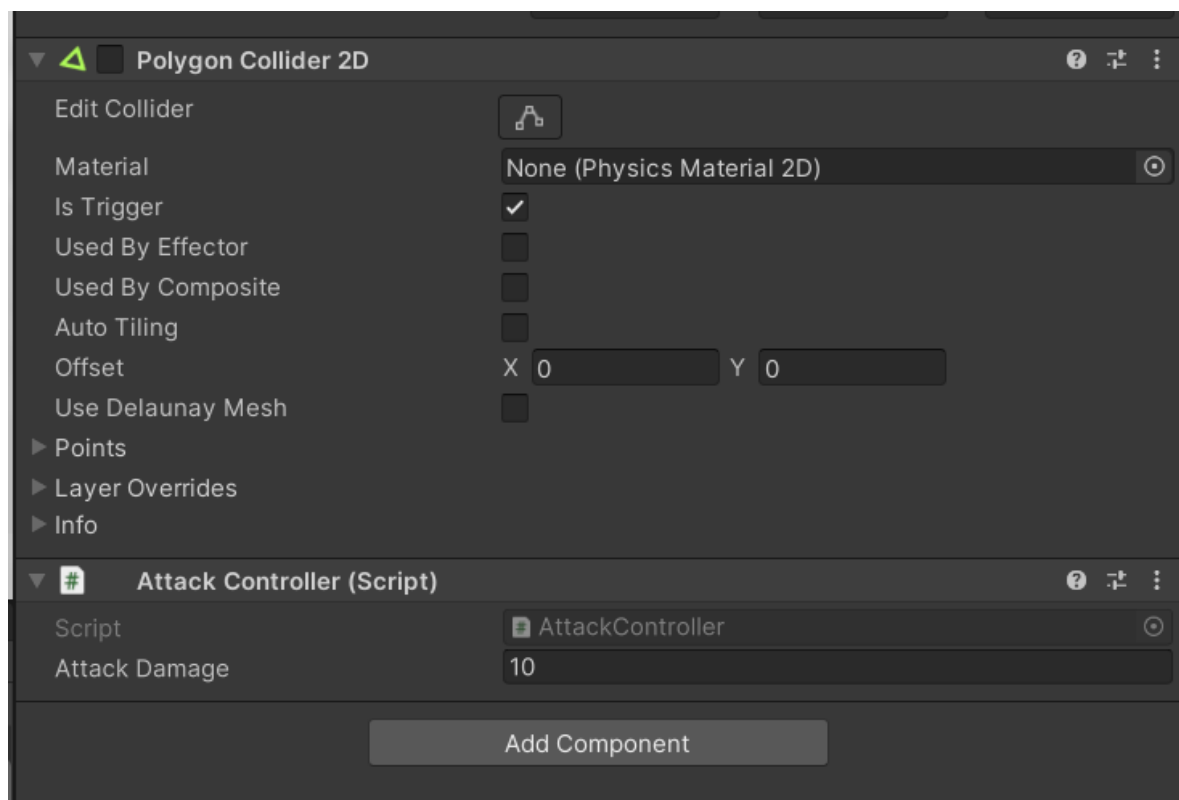
Unity Script (22 asset references) | 3 references
public class AttackController : MonoBehaviour
{
    public int attackDamage = 30;

    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        DamageController damage = collision.GetComponent<DamageController>();

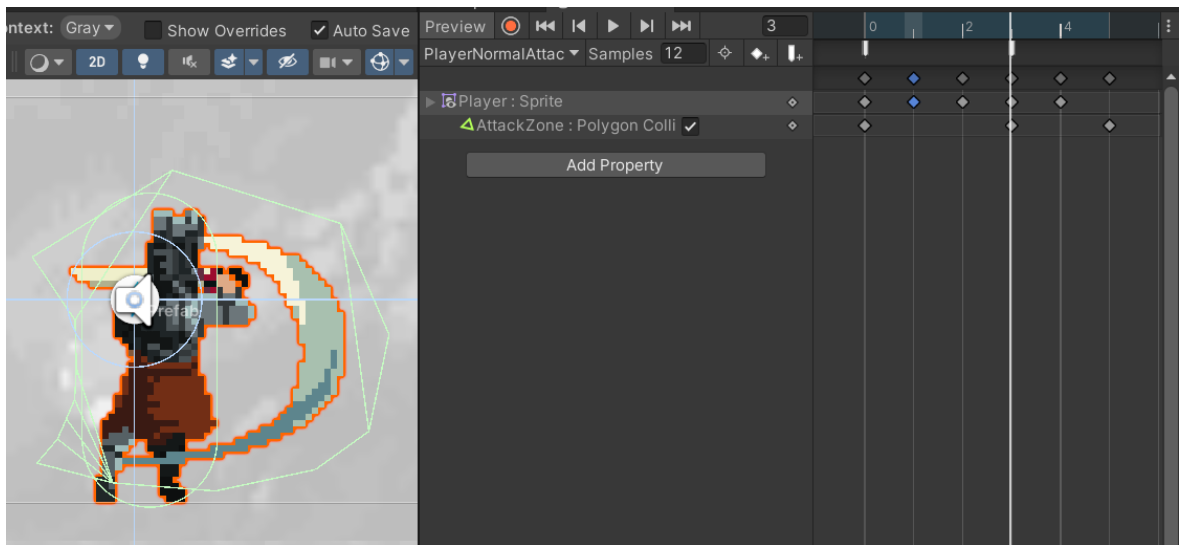
        if (damage != null)
        {
            bool hit = damage.Hit(attackDamage);
            if (hit)
            {
                Debug.Log("Taked hit!");
            }
        }
    }
}

```

Hình 4.13. AttackController



Hình 4.14. Collider của AttackController



Hình 4.15. Áp dụng *AttackController*

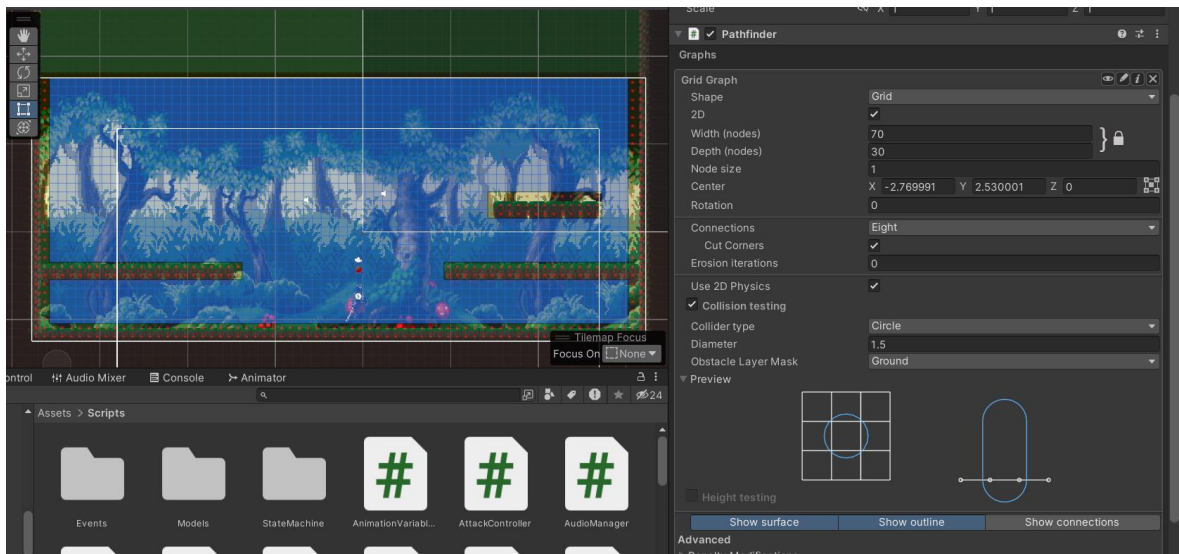
AttackController có thể sử dụng chung cho các Collider gây sát thương.

4.3 Kẻ thù

4.3.1 A* Pathfinding Project

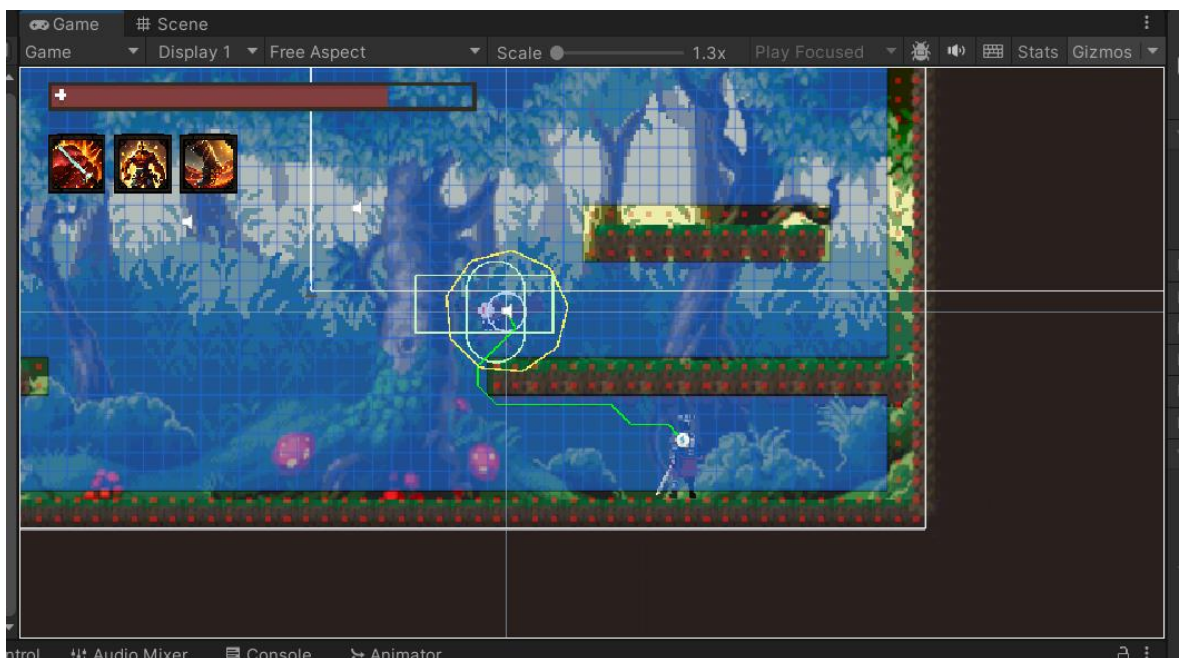
Dự án game sẽ bao gồm 5 loại kẻ thù: Goblin, Mushroom, FireWorm, Flying Eyes và BOSS. Các kẻ thù đều sử dụng FSM đơn giản, AI của các kẻ thù đa số sẽ được lập trình qua Script. Flying Eyes là đối tượng đặc biệt khi có thể bay và tận dụng điều đó thì đối tượng này sẽ sử dụng A* để đuổi theo người chơi.

A* Pathfinding Project là một dự án mã nguồn mở do Arongranberg.com phát triển, cho phép lập trình viên sử dụng cho các dự án game của mình. Script PathFinder được cung cấp có nhiệm vụ quét vùng mà thuật toán A* có thể đi qua. PathFinder sẽ bỏ qua các chướng ngại vật và layer mà người dùng muốn loại bỏ.



Hình 4.16. Pathfinder cho FlyingEye (1)

Hai thành phần tìm đường của FlyingEye dựa trên đường đi mà Pathfinder đã tạo sẵn là Seeker, AIPath và AI Destination Setter. Các thành phần này có nhiệm vụ xác định mục tiêu là Player sau đó sẽ vẽ ra một đường đi ngắn nhất đến người chơi.



Hình 4.17. Pathfinder cho FlyingEye (2)

Ta sẽ cần một script để kẻ thù này có thể di chuyển theo đường đi đã được vẽ. Ta sẽ gọi tới thành phần AIPath của đối tượng để kiểm tra vận tốc tọa độ x mà AIPath quyết định từ đó di chuyển đối tượng theo.

```

// Update is called once per frame
@ Unity Message | 0 references
void Update()
{
    if (aiPath.desiredVelocity.x >= 0.01f)
    {
        transform.localScale = new Vector3(Mathf.Abs(transform.localScale.x), transform.localScale.y, transform.localScale.z);
    }
    else if (aiPath.desiredVelocity.x <= -0.01f)
    {
        transform.localScale = new Vector3(-Mathf.Abs(transform.localScale.x), transform.localScale.y, transform.localScale.z);
    }

    HasTarget = zone.detectedCols.Count > 0;
    if (AttackCoolDown > 0)
    {
        AttackCoolDown -= Time.deltaTime;
    }
}

```

Hình 4.18. Pathfinder cho FlyingEye (3)

4.3.2 AI dành cho kẻ thù di chuyển dưới đất

Di chuyển của Goblin, Mushroom và Fireworm đều giống nhau, các đối tượng sẽ dựa theo vector của người chơi và vị trí của người chơi để tính toán và dẫn người chơi đến vị trí của chơi thông qua hàm Update.

```

// Update is called once per frame
@ Unity Message | 0 references
void Update()
{
    if (player != null && CanMove)
    {
        // Tính toán hướng vector từ Goblin tới người chơi trên trục x
        Vector2 direction = new Vector2(player.position.x - transform.position.x, 0f).normalized;
        // Di chuyển Goblin theo hướng này, giữ nguyên thành phần vận tốc y để cho phép trọng lực tác động
        rb.velocity = new Vector2(direction.x * walkSpeed, rb.velocity.y);

        // Flip hình ảnh nếu cần
        if (direction.x > 0)
        {
            // Đang nhìn về bên phải
            transform.localScale = new Vector3(Mathf.Abs(transform.localScale.x), transform.localScale.y, transform.localScale.z);
        }
        else
        {
            // Đang nhìn về bên trái
            transform.localScale = new Vector3(-Mathf.Abs(transform.localScale.x), transform.localScale.y, transform.localScale.z);
        }
    }
}

```

Hình 4.19. Pathfinder cho quái vật dưới đất (1)

Tuy nhiên, đôi khi người chơi sẽ đứng trên platform cũng như đứng bên dưới platform, việc này sẽ khiến kẻ thù sẽ di chuyển qua lại trên một điểm mà điểm đó bên dưới hoặc bên trên là người chơi đang đứng yên. Để giải quyết vấn đề này, ta cần xác định chiều cao và khoảng cách dựa trên tọa độ y và x để biết khi nào đối tượng nên nhảy lên hoặc đi xuống platform.

Đối tượng nhảy lên chỉ cần thêm một lực vào vận tốc của người chơi trên tọa độ y. Còn khi đi xuống platform thì ta sẽ áp dụng lại mã đi xuống platform của người chơi đó là bỏ qua Collider bằng IgnoreCollision.

```

@ Unity Message | 0 references
private void FixedUpdate()
{
    if (player != null && touchingEvents.IsTouch && Mathf.Abs(player.position.x - transform.position.x) < 2.5f)
    {
        float yDistance = player.position.y - transform.position.y;

        if (yDistance < 0 && Mathf.Abs(yDistance) >= playerDistanceThreshold)
        {
            RaycastHit2D[] hits = new RaycastHit2D[1];
            col.Cast(Vector2.down, contactFilter, hits, 0.05f);

            int hitCount = col.Cast(Vector2.down, contactFilter, hits, 0.05f);

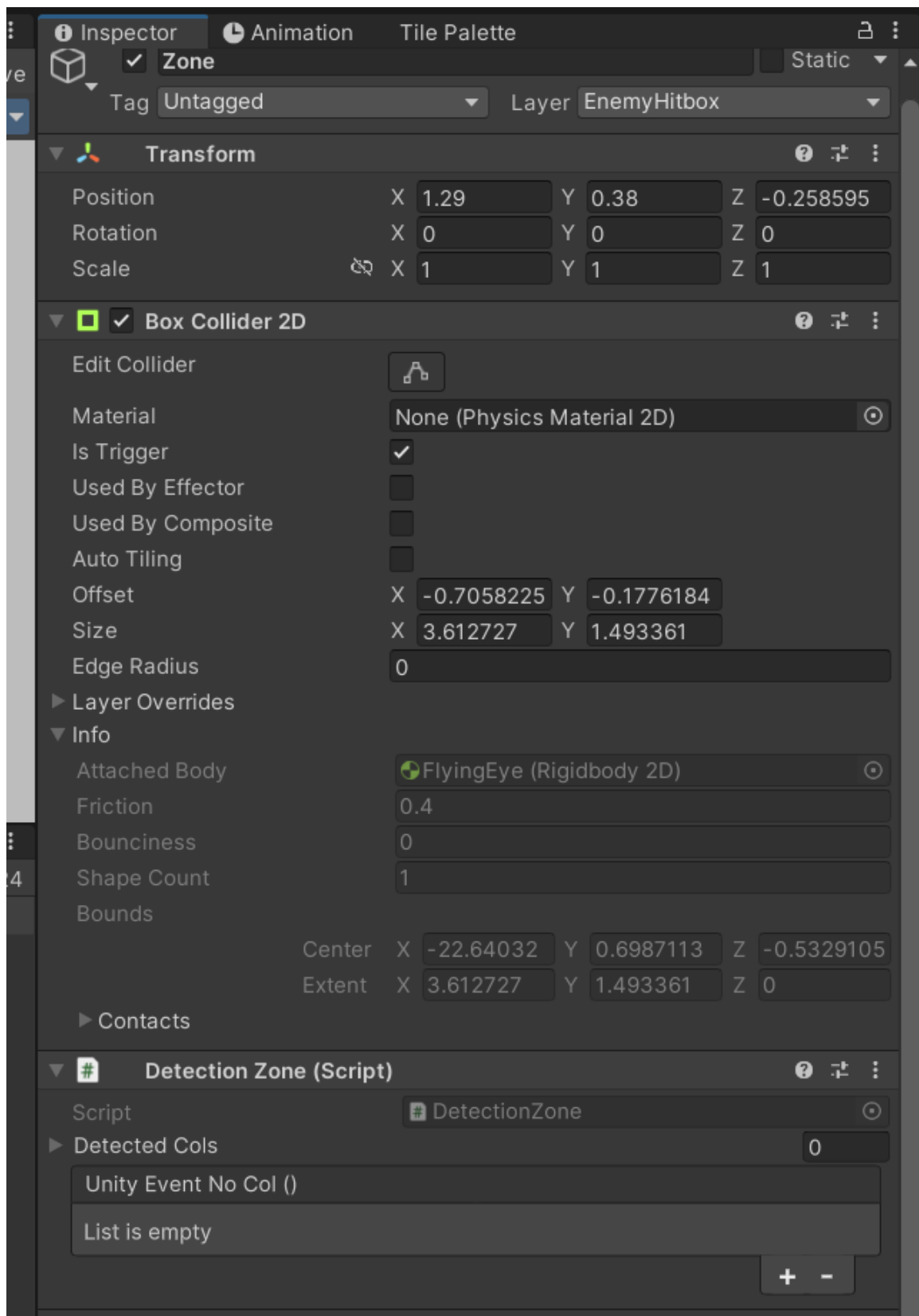
            if (hitCount > 0 && hits[0].collider.CompareTag("Platform"))
            {
                Physics2D.IgnoreCollision(col, hits[0].collider, true);
                StartCoroutine(ResetCollisionAfterDelay(hits[0].collider));
            }
        }
        else if (yDistance > 0 && Mathf.Abs(yDistance) >= playerDistanceThreshold)
        {
            // Kiểm tra nếu enemy chưa đạt đến độ cao tối đa của nhảy
            if (transform.position.y < player.position.y + 6f)
            {
                // Thực hiện hành động nhảy lên
                rb.velocity = new Vector2(rb.velocity.x, jumpForce);
            }
            else
            {
                // Nếu đã đạt đến độ cao tối đa, giảm tốc độ lên hoặc ngừng di chuyển lên
                rb.velocity = new Vector2(rb.velocity.x, 0f);
            }
        }
    }
}
// Start is called before the first frame update

```

Hình 4.20. Pathfinder cho quái vật dưới đất (2)

4.3.3 Phát hiện và tấn công người chơi

Các kẻ thù cần phải có một đối tượng con chứa BoxCollider2D là một Trigger và một Script xử lý sự kiện khi người chơi đi vào vùng phát hiện của kẻ thù.



Hình 4.21. Đối tượng con cho cơ chế Detection

Detection Zone là script dùng OnTriggerEnter2D để khi người chơi đi vào vùng trigger thì sẽ thêm collider vào một mảng, script chính của đối tượng sẽ kiểm tra mảng này nếu có đối tượng trong mảng sẽ thi triển tấn công.

```

Unity Script (10 asset references) | 7 references
public class DetectionZone : MonoBehaviour
{
    public List<Collider2D> detectedCols = new List<Collider2D> ();
    Collider2D col;
    public UnityEvent unityEventNoCol;

    Unity Message | 0 references
    private void Awake()
    {
        col = GetComponent<Collider2D>();
    }

    Unity Message | 0 references
    private void OnTriggerEnter2D(Collider2D collision)
    {
        detectedCols.Add (collision);
    }

    Unity Message | 0 references
    private void OnTriggerExit2D(Collider2D collision)
    {
        detectedCols.Remove(collision);

        if (detectedCols.Count <= 0)
        {
            unityEventNoCol.Invoke();
        }
    }
}

```

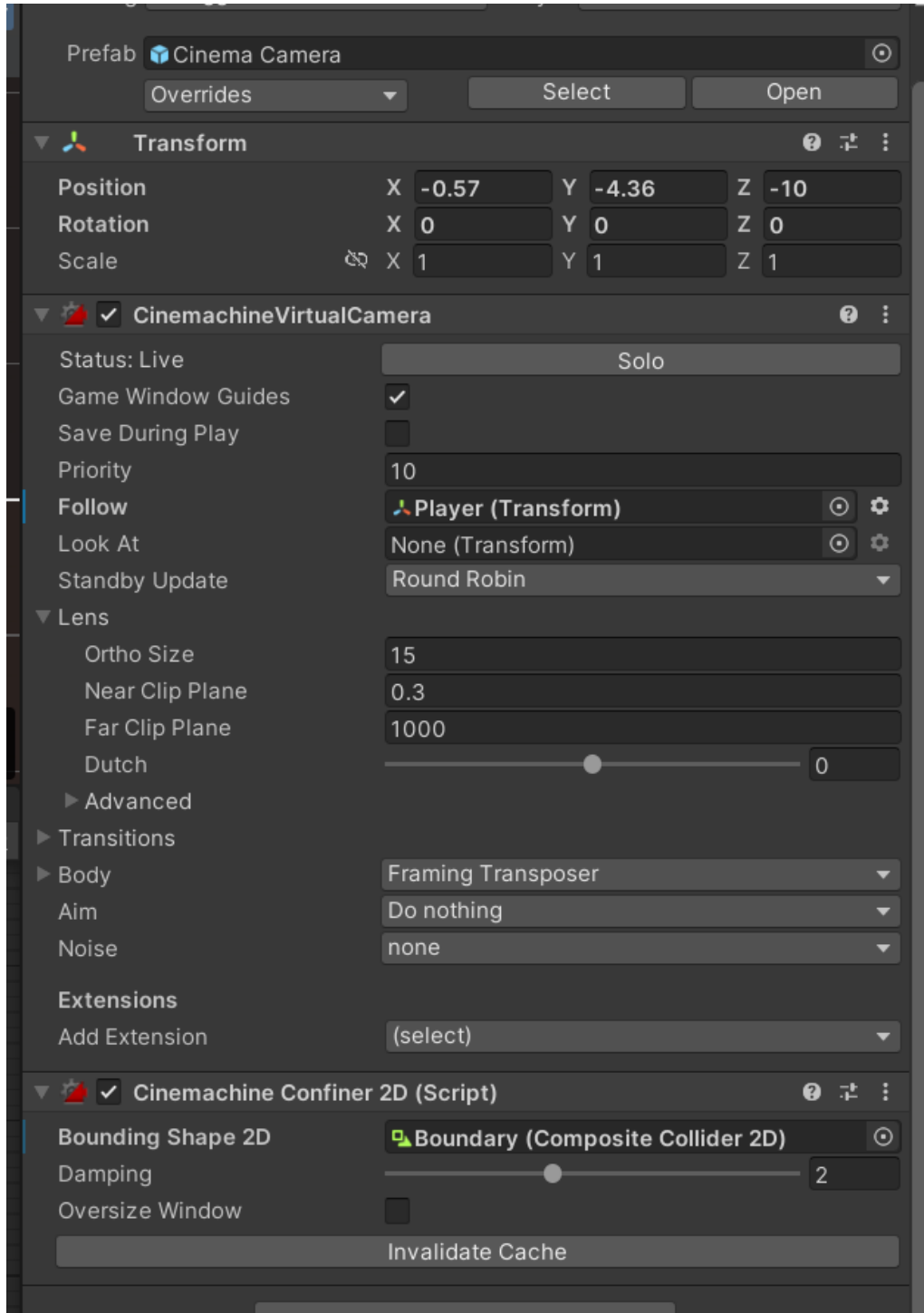
Hình 4.22. DetectionZone

OnTriggerExit2D sẽ xử lý sự kiện nếu người chơi thoát khỏi vùng đó thì sẽ xóa collider của người chơi ra khỏi mảng và kích hoạt sự kiện khi người chơi thoát ra khỏi vùng phát hiện của đối tượng.

4.4 Camera

Cinemachine là một package cung cấp camera với phong cách điện ảnh cho phép camera luôn theo dõi người chơi dưới góc nhìn điện ảnh. Ngoài ra thành phần

Cinemachine Confinder 2D giúp xác định camera sẽ chỉ bên trong một collider (lúc này là Boundary) có Composite Collider. Package giúp cho camera di chuyển và theo dõi người chơi mượt mà hơn giúp tăng mạnh trải nghiệm người dùng.



Hình 4.23. Cinemachine

4.5 Lưu tiến trình chơi

Để lưu trữ tiến trình của người chơi, ta cần một đối tượng không thể bị phá hủy qua các cảnh khác nhau kể từ lúc bắt đầu game cho đến lúc thoát khỏi game. GlobalManager sẽ sử dụng PlayerPrefs để lưu trữ các chỉ số quan trọng của người chơi như khi người chơi thoát game và lưu game:

- Lượng máu hiện tại của người chơi – Health
- Các buff mà người chơi đã nhận (lưu một mảng Buff dưới dạng Json) – Buffs
- Số scene đã đi qua (khi đạt số scene nhất định sẽ có thể qua các scene về boss) – SceneTransitions
- Trạng thái người chơi đã hoàn thành các cảnh trong rừng chưa (PlayerPrefs vẫn chưa lưu được boolean cho nên chỉ lưu giá trị 0 và 1) – FinishNormal
- Vũ khí hiện tại người chơi đang sử dụng (Do PlayerPrefs không lưu đối tượng phức tạp như RuntimeAnimatorController nên ta sẽ chỉ lưu một biến cờ, khi tải dữ liệu lên thì dựa vào cờ sẽ cho người chơi sử dụng animator nào) – CurrentWeaponFlag
- Và index của scene hiện tại – CurrentSceneIndex

```

43 references
public static GlobalManager Instance { get; set; }

[Header("Saved Variable")]
public RuntimeAnimatorController GlobalAnimatorController;
public int health = 100;
public List<Buff> buffs = new List<Buff>();
public int sceneTransitionCount = -1; // Not include Start Point
public bool isFinishNormal = false;
public int sceneIndex;

[Header("Player Animator Controller")]
public RuntimeAnimatorController sword;
public RuntimeAnimatorController spear;
public RuntimeAnimatorController axe;

public int currentWeaponFlag = 1;

```

Hình 4.24. Lưu tiến trình chơi (1)

```

1 reference
public void SavePlayerState()
{
    sceneIndex = SceneManager.GetActiveScene().buildIndex;
    PlayerPrefs.SetInt("Health", health);
    PlayerPrefs.SetString("Buffs", JsonUtility.ToJson(new BuffList { Buffs = buffs }));
    PlayerPrefs.SetInt("SceneTransitions", sceneTransitionCount);
    PlayerPrefs.SetInt("FinishNormal", isFinishNormal ? 1 : 0);
    PlayerPrefs.SetInt("CurrentWeaponFlag", currentWeaponFlag);
    PlayerPrefs.SetInt("CurrentSceneIndex", sceneIndex);
    PlayerPrefs.Save();
}

```

Hình 4.25. Lưu tiến trình chơi (2)

Khi load dữ liệu ta chỉ cần gán các giá trị đã lưu vào các thuộc tính của GlobalManager từ đó các game sẽ sử dụng các thuộc tính từ GlobalManager để tiếp tục trạng thái của người chơi.

```

2 references
public void LoadPlayerState()
{
    sceneIndex = PlayerPrefs.GetInt("CurrentSceneIndex");

    health = PlayerPrefs.GetInt("Health", 100);
    string buffsJson = PlayerPrefs.GetString("Buffs", "{}");
    BuffList loadedBuffs = JsonUtility.FromJson<BuffList>(buffsJson);
    if (loadedBuffs != null && loadedBuffs.Buffs != null)
    {
        buffs = loadedBuffs.Buffs;
    }
    sceneTransitionCount = PlayerPrefs.GetInt("SceneTransitions", 0);
    isFinishNormal = PlayerPrefs.GetInt("FinishNormal", 0) != 0;

    int weaponFlag = PlayerPrefs.GetInt("CurrentWeaponFlag", 1); // Sword is default
    ApplyWeaponController(weaponFlag);
}

```

Hình 4.26. Load tiến trình chơi (1)

```

Unity Message | 0 references
private void Start()
{
    if (FindObjectOfType<GlobalManager>() != null)
    {
        animator.runtimeAnimatorController = GlobalManager.Instance.GlobalAnimatorController;
        damage.Health = GlobalManager.Instance.health;
        if (GlobalManager.Instance.buffs.Count > 0)
        {
            foreach (var buff in GlobalManager.Instance.buffs)
            {
                ApplyBuff(buff);
            }
        }
    }
}

```

Hình 4.27. Load tiến trình chơi (2)

CHƯƠNG 5. KẾT QUẢ ĐẠT ĐƯỢC

5.1 Tổng quan

The Loop sẽ kể về câu chuyện một hiệp sĩ thời trung cổ bị rơi vào một vòng lặp thời gian, khiến anh phải liên tục sống lại cùng một ngày, trong quá trình chơi người chơi sẽ dần tìm hiểu được những bí mật về vòng lặp thời gian và câu chuyện của chàng hiệp sĩ.

Nhân vật chính của câu chuyện sẽ là một hiệp sĩ thời trung cổ, vô tình rơi vào vòng lặp thời gian, anh chàng hiệp sĩ bất đắc dĩ phải vừa chiến đấu bảo vệ ngôi làng của mình, vừa phải tìm cách thoát khỏi vòng lặp.

Bối cảnh trò chơi diễn ra vào thời trung cổ với địa điểm chủ yếu là ở khu rừng, nơi mà ngôi làng của chàng hiệp sĩ tọa lạc, do sự xâm lăng của quái vật nên khu rừng này đã không còn yên bình như trước, hiệp sĩ phải ra tay trước khi nơi đây trở thành vùng đất căn cỗi.



Hình 5.1. Bối cảnh ngôi làng

Mục tiêu của người chơi trong tựa game này là tiêu diệt quái và nâng cấp nhân vật để đánh bại được boss của màn chơi, game sẽ được thiết kế gồm 5 màn chơi được

ngẫu nhiên từ 15 màn chơi có sẵn, sau mỗi màn chơi nhân vật sẽ nhận được các nâng cấp và tiến vào màn đánh boss

5.2 Gameplay

The Loop sẽ được thiết kế theo kiểu platformer và side scrolling với thể loại roguelite làm chủ đạo, người chơi sẽ điều khiển hiệp sĩ tiêu diệt quái vật, nâng cấp sức mạnh và tiến về phía trước cho đến khi đánh bại được boss.

Người chơi sẽ được lựa chọn giữa ba món vũ khí: kiếm, thương, rìu; mỗi vũ khí sẽ có các đặc điểm riêng biệt, mang đến lối chơi khác nhau:

- Kiếm: đây là món vũ khí cân bằng nhất khi có tầm đánh, sát thương đều ở mức trung bình
- Thương: đây là món vũ khí thiên về tấn công nhanh, tuy có sát thương thấp nhưng có combo nhiều đòn giúp lấy số lượng bù chất lượng. Đây cũng là vũ khí có tầm đánh xa nhất
- Rìu: đây là món vũ khí có sức sát thương cao, tuy nhiên đánh đổi về tốc độ và tầm đánh. Đây là vũ khí trái ngược với lối chơi của thương.



Hình 5.2. Các loại vũ khí có thể dùng

Ngoài lựa chọn di chuyển, người chơi còn có thể nhảy để tiếp cận các platform, lướt để né tránh các đòn đánh. Hoặc kết hợp nhảy và lướt để chạm đến những platform cao và xa.

Là một game Roguelite, The Loop không thể thiếu những yếu tố ngẫu nhiên, ở đây là trong các màn chơi, quái vật, và cả các phần thưởng trong game. Tùy theo quái vật và phần thưởng được nhận, trò chơi có thể khó hơn hoặc dễ đi



Hình 5.3. Ví dụ một màn chơi điển hình

Các loại quái vật trong game đều có những đặc trưng riêng biệt, hiện tại trong The Loop có 4 loại quái vật:

- Goblin: loại quái vật bình thường, có khả năng đánh cận chiến với sức tấn công cao và máu thấp
- Mushroom: một loại nấm quỷ có sức tấn công thấp nhưng vô cùng cứng cáp với lượng máu lớn
- Fireworm: sâu lửa với khả năng bắn tầm xa
- Flying Eye: quái vật một mắt có khả năng bay theo người chơi

Sau khi tiêu diệt hết quái ở một level, phần thưởng sẽ xuất hiện dưới dạng ngẫu nhiên, 3 trong số 12 phần thưởng sẽ được xuất hiện cho người chơi lựa chọn, và người chơi chỉ được chọn một trong ba lựa chọn đó



Hình 5.4. Hệ thống phần thưởng

5.3 Cấu trúc

Cấu trúc gameplay của The Loop được cấu tạo từ những map được tạo sẵn, sau đó 5 map trong số đó sẽ được dùng làm thử thách cho người chơi, đến các lượt chơi sau cũng ngẫu nhiên như vậy. Mỗi map được cấu tạo từ nhiều platform khác nhau tạo nên đặc điểm riêng cho từng map

Mỗi map đều có một hệ thống spawn quái ngẫu nhiên, tại các điểm cố định, một loại quái vật ngẫu nhiên sẽ được spawn, gọi là một đợt quái, sau 2 hoặc 3 đợt quái như vậy, người chơi sẽ được xem là hoàn thành level và được nhận phần thưởng

Mỗi lần hoàn thành một level, người chơi đều được nhận một trong ba phần thưởng được ngẫu nhiên, đó là những kỹ năng hoặc chỉ số cộng thêm, và những kỹ năng hay chỉ số này chỉ được dùng cho lần chơi đó, sau đó khi chết hoặc đánh bại boss cuối, người chơi sẽ mất hết tất cả phần thưởng.

Sau 5 level đánh quái vật, người chơi sẽ được đến khu vực nghỉ ngơi, đây là nơi để người chơi hồi máu và chuẩn bị cho màn chơi đánh boss. Khu vực đánh boss và boss sẽ được cố định và không mang yếu tố ngẫu nhiên. Boss cũng sở hữu nhiều đòn đánh khác nhau yêu cầu người chơi phải có chiến thuật cho từng đòn đánh.



Hình 5.5. Boss của game

The Loop cũng sẽ có hệ thống lưu tiến trình hiện tại của người chơi đối với những lần chơi dang dở, người chơi có thể tiếp tục ngay tại thời điểm dừng lại và quá trình chơi sẽ không bị ảnh hưởng.

5.4 Giao diện người dùng (UI)

UI của The Loop được thiết kế theo hướng đơn giản nhưng đầy đủ thông tin cần thiết cho người chơi nắm bắt, tránh gây rối mắt.

Phần Menu sẽ có cái tùy chọn như: “Start”, “Settings”, “Exit”, các tùy chọn này sẽ được thiết kế sao cho diễn nhận diện, dễ đọc.

Trong game cũng sẽ có hướng dẫn cơ bản để người chơi có thể xem và nắm bắt cách điều khiển nhân vật, cách trò chơi vận hành,...

UI cũng thể hiện các thông tin cần thiết của người chơi như: máu, kỹ năng, lượt, sát thương gây ra, lượng máu mất,... Các thông tin khác ảnh hưởng đến người chơi sẽ không được hiển thị cụ thể, nhưng vẫn có thể ước lượng được thông qua gameplay

Khi người chơi đến các vật thể có thể tương tác sẽ xuất hiện hộp thông báo nhỏ để người chơi nhận biết, hộp thông báo sẽ thiết kế phù hợp với bối cảnh của trò chơi sao cho nổi bật nhưng không bị lạc quẻ.

Người chơi có thể tạm dừng game bằng phím tắt, màn hình dừng game sẽ hiển thị kỹ năng hiện có của người chơi, các tùy chọn,...

Người chơi có thể điều chỉnh âm thanh và đồ họa của game thông qua giao diện Settings của game, các điều chỉnh này sẽ được lưu trong suốt quá trình chơi game.

Khi nhân vật hết máu, sẽ có màn hình thông báo Game Over và hiển thị các lựa chọn cho phép chơi lại hoặc thoát game.

5.5 Âm thanh và đồ họa

Nhạc nền của The Loop được tìm kiếm theo ba từ khóa: “medieval”, “classic”, “mystery”. Nhạc trong game cần đảm bảo mang lại cho người chơi cảm giác cổ điển, mang đậm tính trung cổ và bắt tai, tương tự với nhạc folk mà các hiệp sĩ, các gã bộm nhậu hay hát khi muốn tạo bầu không khí.

Trái ngược với nhạc nền khi chưa chiến đấu, nhạc nền khi người chơi gặp quái vật cần mang tính căng thẳng hơn, nhưng không được mất đi yếu tố cổ điển và trung cổ của game

Các hiệu ứng âm thanh khi tấn công, trúng đòn hay tiếng bước chân được thiết kế như âm thanh 8bit như những game pixel thời xưa, nhưng cũng phải đảm bảo đủ chân thật để người chơi có thể nhận biết đó là tiếng của hành động gì

Về đồ họa, The Loop là một tựa game 2D pixel, các Assets đồ họa của game sẽ dựa theo tiêu chí các game cùng thể loại, cụ thể ở đây là Dead Cells



Hình 5.6. Lối thiết kế của Dead Cells

Bối cảnh trong rừng của game được thiết kế để mang lại cảm giác như một khu rừng thật với cây cối và các sinh vật sinh sống trong rừng, các chuyển động của background cũng sẽ được xem xét thực hiện để tăng tính sống động cho màn chơi.

Nhân vật và quái vật cũng được thiết kế với chuyển động mượt mà, mang cảm giác của các nhân vật trung cổ, các loại quái vật sẽ mang hơi hướng của các loài quái vật rừng như nấm, sâu bọ,...

Các hiệu ứng kỹ năng được thiết kế đa dạng tùy theo nguyên tố của kỹ năng, như lửa cần mạnh mẽ, nước thì dịu dàng hơn, khí thì cần nhanh nhẹn, linh hoạt,...

Giao diện người dùng (UI) cũng được thiết kế theo phong cách 2D Pixel đậm chất trung cổ để có thể hài hòa với phong cách của tựa game.

Tất cả những Assets sẽ được tìm kiếm và có thể thiết kế lại để phù hợp hơn với định hướng của tựa game.

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận

Hiện tại The Loop đã hoàn thành khoảng 75% dự định ban đầu, thể hiện được các đặc trưng cơ bản của Roguelite, một số điểm nổi bật của The Loop hiện tại gồm có:

- Hệ thống vũ khí, kỹ năng, chỉ số, level đủ đa dạng và hấp dẫn để chơi lại nhiều lần
- Game đủ khó để thách thức các người chơi
- Một số Assets được lấy từ Itch.io được thiết kế lại để phù hợp với game
- Có tiềm năng sáng tạo thêm, bổ sung thêm các cơ chế khác thú vị hơn
- Đã học hỏi được dù ít hay nhiều từ các tựa game Roguelike đi trước

Tuy vậy, The Loop cũng còn những hạn chế nhất định:

- Hệ thống gameplay tuy đa dạng nhưng vẫn chưa đủ tầm để gọi là một game Roguelite
- Số lượng quái tương đối ít và giống nhau về cách vận hành
- AI còn nhiều điều có thể cải thiện thêm
- Độ khó của game phần ít đến từ sự thiếu mượt mà trong chiến đấu, chưa hẳn là hoàn toàn do thử thách của game

6.2 Hướng phát triển

The Loop hiện tại còn rất mới và có nhiều tiềm năng phát triển, định hướng hiện tại có thể là trau chuốt lại gameplay, cách vận hành của vũ khí và quái vật, đa dạng hóa hơn các phần thưởng. Đặc biệt là mở rộng thêm nội dung, câu chuyện đang còn dở dang của tựa game. Cụ thể là thêm vào các vùng đất mới để khám phá, các boss đủ khó để thử thách người chơi, các câu chuyện được rải rác xuyên suốt hành trình của nhân vật.

Với kinh nghiệm có được từ dự án lần này, từng thành viên trong nhóm cũng sẽ có những định hướng cải thiện bản thân về kỹ năng phát triển và vận hành một tựa game, sáng tạo thêm các ý tưởng mới và phù hợp với thể loại Roguelite nói riêng và

các thể loại khác nói chung. Cuối cùng là phát triển hơn kỹ năng làm việc nhóm và giao tiếp trong học tập, công việc.

TÀI LIỆU THAM KHẢO

Tiếng Việt

aozora (2022), *Game roguelike là gì? Có khác với roguelite hay chỉ là một?*.

<https://bloganchoi.com/game-roguelike-roguelite-la-gi/>

Ngô Tấn Phúc (2021), *Roguelike, Rogue-lite là gì? Cách phân biệt, top 10 game hay nhất*. <https://www.thegioididong.com/game-app/roguelike-la-gi-1274141>

Tiếng Anh

Unity Technologies, *Cinemachine Documentation*.

<https://docs.unity3d.com/Packages/com.unity.cinemachine@2.3/manual/index.html>

Wikipedia, the free encyclopedia, *Rogue (video game)*.

https://en.wikipedia.org/wiki/Rogue_%28video_game%29

Wikipedia, the free encyclopedia, *Roguelike*

<https://en.wikipedia.org/wiki/Roguelike>

A* Pathfinding Project 5.0.9, *Documentation*,

<https://arongranberg.com/astar/docs/>