

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



ĐỒ ÁN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG GAME BLACKOUT OPS
BẰNG THƯ VIỆN SDL TRONG C++

GVHD: TS. Nguyễn Mạnh Cường
Sinh viên: Nguyễn Mạnh Thắng
Mã sinh viên: 2021607883
Lớp: KHMT02 – K16

Hà Nội – 2025

MỤC LỤC

MỤC LỤC	1
DANH MỤC HÌNH ẢNH	4
DANH MỤC CÁC TỪ VIẾT TẮT	6
LỜI CẢM ƠN	7
LỜI NÓI ĐẦU	8
CHƯƠNG 1. TỔNG QUAN VỀ TRÒ CHƠI ĐIỆN TỬ.....	10
1.1 Giới thiệu tổng quan về game, phân loại game	10
1.1.1 Tổng quan về game	10
1.1.2 Phân loại game	10
1.2 Các công cụ hỗ trợ làm game.....	14
1.2.1 Game Engines	14
1.2.2 Công cụ lập trình (Programming Tools)	15
1.2.3 Công cụ thiết kế đồ họa 2D (Graphics and Animation Tools).....	16
1.2.4 Công cụ âm thanh (Audio Tools).....	16
1.3 Tổng quan về thư viện SDL trong C++	16
1.3.1 Tổng quan về thư viện SDL.....	16
1.3.2 Ưu nhược điểm của phần mềm.....	18
CHƯƠNG 2. THIẾT KẾ GAME	27
2.1 Giới thiệu tổng quan.....	27
2.1.1 Thông tin game	27
2.1.2 Thể loại game và yếu tố game	28
2.1.3 Đối tượng chơi	29
2.1.4 Nền tảng	29
2.2 Thiết kế kịch bản game	30
2.2.1 Mô tả	30
2.2.2 Thiết kế luật chơi	31

2.2.3 Thiết kế các màn của game	32
2.3 Thiết kế các vật phẩm thu thập	32
2.4 Thiết kế cách tính điểm	33
2.5 Thiết kế tương tác và điều khiển game	34
2.6 Storyboard	35
2.6.1 Sơ đồ các màn hình	35
2.6.2 Màn hình bắt đầu	35
2.6.3 Màn hình chơi game	36
2.6.4 Màn hình hướng dẫn chơi game	37
2.6.5 Màn hình chọn bản đồ và chọn nhân vật	37
2.6.6 Màn hình khi thua	39
CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ.....	40
3.1 Xây dựng giao diện game	40
3.1.1 Singleton Pattern	41
3.1.2 Factory Pattern	42
3.1.3 Observer	44
3.1.4 Thuật toán trong game	45
3.2 Code chức năng chính của game.....	47
3.2.1 Các phương thức cần thiết để xây dựng logic	47
3.2.2 Phần code tạo bản đồ và tải bản đồ.....	49
3.2.3 Phương thức xây dựng đạn	51
3.2.4 Phương thức xây dựng di chuyển và tấn công của Boss	53
3.2.5 Phương thức xây dựng sức mạnh nhân vật.....	55
3.2.6 Phương thức xử lý va chạm	57
3.2.7 Phương thức xử lý vụ nổ trong game	59
3.3 Kết quả giao diện game.....	61
3.3.1 Màn hình chính	61
3.3.2 Màn hình hướng dẫn chơi game	62
3.3.3 Màn hình chọn bản đồ.....	63

3.3.4 Màn hình chọn nhân vật.....	64
3.3.5 Màn hình chờ vào game.....	65
3.3.6 Màn hình chơi game	66
3.3.7 Màn hình thua game.....	67
3.3.8 Màn hình thắng game.....	68
3.4 Đánh giá	68
3.4.1 Ưu điểm.....	68
3.4.2 Nhược điểm.....	69
3.4.3 Hướng khắc phục	69
KẾT LUẬN	71
TÀI LIỆU THAM KHẢO	73

DANH MỤC HÌNH ẢNH

Hình 1.1: Hình ảnh máy chơi game Arcade	11
Hình 1.2: Hình ảnh thiết bị chơi game Console.....	12
Hình 1.3: Hình ảnh thiết bị chơi game PC	13
Hình 1.4: Hình ảnh thiết bị chơi game Mobile	14
Hình 1.5: Logo thư viện SDL.....	16
Hình 1.6: Tựa game SuperTuxKart.....	23
Hình 1.7: Tựa game Hedgewars.....	23
Hình 1.8: Tựa game Battle for Wesnoth.....	24
Hình 1.9: Tựa game Aquaria	24
Hình 1.10: Tựa game Neverball.....	25
Hình 1.11: Tựa game Secret Maryo Chronicles	25
Hình 1.12: Tựa game Lugaru	26
Hình 2.1: Giao diện game Blackout Ops (lose game)	32
Hình 2.2: Hình đồng tiền vàng.....	33
Hình 2.3: Hình đồng tiền trong game.....	33
Hình 2.4: Hình kim cương trong game	33
Hình 2.5: Sơ đồ các màn hình.....	35
Hình 2.6: Giao diện trang chủ của game.....	36
Hình 2.7: Vị trí các nút trong trận đấu game.....	36
Hình 2.8: Màn hình hướng dẫn chơi game	37
Hình 2.10: Giao diện chọn nhân vật.....	38
Hình 2.11: Màn hình khi thua	39
Hình 3.1: Xây dựng BackgroundMap.....	40
Hình 3.2: Xây dựng các block.....	41
Hình 3.3: Singleton Pattern	41

Hình 3.4: Các phương thức cần thiết tạo ra game.....	47
Hình 3.5: Chương trình tạo bản đồ	49
Hình 3.6: Phương thức xây dựng đạn	51
Hình 3.7: Phương thức xây dựng di chuyển và tấn công của quái vật	53
Hình 3.8: Phương thức xây dựng sức mạnh của nhân vật	55
Hình 3.9: Phương pháp xử lý va chạm.....	57
Hình 3.10: Phương thức xử lý vụ nổ.....	59
Hình 3.11: Giao diện chính của game.....	61
Hình 3.12: Màn hình hướng dẫn chơi game	62
Hình 3.13: Màn hình chọn bản đồ.....	63
Hình 3.14: Màn hình chọn nhân vật.....	64
Hình 3.15: Màn hình chờ	65
Hình 3.16: Giao diện trong game.....	66
Hình 3.17: Giao diện khi thua game	67
Hình 3.18: Giao diện chiến thắng	68

DANH MỤC CÁC TỪ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ viết đầy đủ
1	CSDL	Database
2	CMS	Content Management System System
3	DH	University
4	SDL	Simple DirectMedia Layer
5	IT	Information Technology

LỜI CẢM ƠN

Trước tiên với tình cảm sâu sắc và chân thành nhất, cho phép em được bày tỏ lòng biết ơn đến các thầy cô của trường Đại học Công Nghiệp Hà Nội. Nhờ có sự nhiệt tình, tận tâm và kiến thức uyên thâm của các thầy cô, em đã có cơ hội học hỏi và tiếp thu nhiều bài học quý báu và bổ ích. Những kiến thức và kỹ năng mà em đã nhận được không chỉ là hành trang cho đồ án này mà còn là nền tảng vững chắc cho con đường sự nghiệp của em sau này.

Em xin gửi lời cảm ơn đặc biệt đến thầy Nguyễn Mạnh Cường, người đã dành nhiều thời gian và công sức để hướng dẫn em trong suốt quá trình thực hiện đề tài. Thầy không chỉ cung cấp những kiến thức chuyên môn cần thiết mà còn truyền đạt những kinh nghiệm thực tiễn quý báu, giúp em tự tin và chủ động hơn trong công việc nghiên cứu. Mỗi buổi thảo luận, hướng dẫn của thầy đều là những dịp quý báu để em hiểu sâu hơn về các lĩnh vực liên quan đến đề tài, mở rộng tầm nhìn và phát triển kỹ năng tư duy phản biện.

Trong quá trình làm đồ án, khó tránh khỏi sai sót. Em rất mong nhận được ý kiến đóng góp từ thầy cô. Em xin chúc thầy cô luôn dồi dào sức khỏe, luôn vui vẻ và thành công trong cuộc sống.

Em xin trân trọng cảm ơn!

Sinh viên

Nguyễn Mạnh Thắng

LỜI NÓI ĐẦU

Trong những năm gần đây, ngành công nghiệp game không ngừng phát triển mạnh mẽ, đặc biệt là với sự gia tăng của các nền tảng phát triển trò chơi mã nguồn mở và thư viện hỗ trợ phát triển game đa nền tảng. Công nghệ thông tin, học máy, và trí tuệ nhân tạo đang ngày càng đóng vai trò quan trọng trong việc cải thiện chất lượng và trải nghiệm của các trò chơi. Với sự hỗ trợ của các thư viện như SDL, game phát triển trở nên dễ dàng hơn trong việc tối ưu hóa hiệu năng và nâng cao tính tương tác của người chơi. Blackout Ops, trò chơi thuộc thể loại platformer đi cảnh, tiêu diệt quái vật, được chọn làm đề tài vì đây là một thể loại game đơn giản nhưng lại mang lại nhiều thử thách cho người chơi, giúp làm quen với các kỹ thuật lập trình game cơ bản cũng như các thuật toán trong game như va chạm và AI (trí tuệ nhân tạo) trong điều khiển hành vi của quái vật. Đề tài này không chỉ mang tính ứng dụng cao mà còn là cơ hội để sinh viên tìm hiểu sâu hơn về lập trình đồ họa và tối ưu hóa hiệu suất game.

Đề tài này nghiên cứu và ứng dụng thư viện SDL để phát triển trò chơi 2D, tạo ra một trò chơi Blackout Ops với đồ họa đơn giản nhưng lối chơi hấp dẫn. Ngoài việc áp dụng các thuật toán quản lý va chạm và AI, đề tài còn tập trung vào tối ưu hóa đồ họa và âm thanh để trò chơi hoạt động mượt mà trên nhiều nền tảng khác nhau. Thông qua đề tài này, nhóm phát triển hy vọng có thể không chỉ tạo ra một trò chơi thú vị mà còn giúp sinh viên hiểu rõ quy trình phát triển game từ thiết kế đến thử nghiệm.

Sự hỗ trợ và sự chỉ dẫn tận tâm từ các giảng viên không chỉ là nguồn động viên mà còn là chìa khóa mở ra cánh cửa cho sự thành công trong dự án của em. Hy vọng rằng, thông qua sự kết hợp giữa sự đam mê cá nhân và sự hướng dẫn chuyên sâu, trò chơi của em sẽ không chỉ là một sản phẩm độc đáo mà còn là một trải nghiệm giải trí thú vị và chất lượng cho cộng đồng người chơi. Em tin tưởng rằng, với sự hỗ trợ không ngừng và khích lệ từ các giáo viên, dự án của em sẽ ngày càng hoàn thiện và có thể đạt được thành công toàn diện khi xuất hiện trên thị trường rộng lớn. Chúc mừng và tri ân những giáo viên, những người đã đồng hành và hỗ trợ em trong hành trình sáng tạo này.

Báo cáo này của em được chia làm 3 chương chính:

- **Chương 1: Tổng quan về trò chơi điện tử:** Nội dung của chương này sẽ đưa ra khái niệm về trò chơi điện tử cũng như ngành công nghiệp trò chơi điện tử hiện nay trong đời sống
- **Chương 2: Thiết kế game:** Đưa ra bài toán, kịch bản, gameplay cũng như thiết kế những chi tiết, hình ảnh cần thiết.
- **Chương 3: Cài đặt chương trình và kết quả:** Chương này sẽ đề trình bày những loại code kết hợp với nhau để tạo ra 1 tựa game 2D hoàn chỉnh, trình bày kết quả của game, gồm giao diện chính và gameplay cuối cùng.

Qua đề tài này, em hy vọng có thể đóng góp một cách tiếp cận hiệu quả trong việc phát triển trò chơi 2D sử dụng thư viện SDL, từ đó hỗ trợ sinh viên và nhà phát triển mới làm quen với lập trình game đa nền tảng. Kết quả nghiên cứu không chỉ mang lại giá trị học thuật trong việc ứng dụng các thuật toán xử lý va chạm và AI, mà còn có ý nghĩa thực tiễn trong việc xây dựng các sản phẩm giải trí có tính tương tác cao, tối ưu hiệu năng và dễ triển khai. Đề tài cũng mở ra tiềm năng cho việc nghiên cứu sâu hơn về tối ưu hóa đồ họa, xử lý âm thanh và mở rộng ứng dụng game trong giáo dục hoặc mô phỏng kỹ thuật.

CHƯƠNG 1. TỔNG QUAN VỀ TRÒ CHƠI ĐIỆN TỬ

1.1 Giới thiệu tổng quan về game, phân loại game

1.1.1 Tổng quan về game

Trò chơi video hay còn gọi là video game là một loại hình giải trí kỹ thuật số, trong đó người chơi tương tác thông qua giao diện người dùng hoặc thiết bị đầu vào như cần điều khiển, tay cầm, bàn phím, chuột hoặc các thiết bị cảm biến chuyển động để điều khiển các nhân vật hoặc đối tượng trong game và nhận phản hồi trực quan. Phản hồi này thường xuất hiện trên các thiết bị hiển thị video như TV, màn hình máy tính, màn hình cảm ứng, hoặc các thiết bị thực tế ảo, mang đến cho người chơi những trải nghiệm hình ảnh sinh động. Ngoài ra, hầu hết các trò chơi còn được bổ sung hiệu ứng âm thanh qua loa ngoài hoặc tai nghe, giúp tăng tính sống động. Đặc biệt, một số trò chơi còn kết hợp công nghệ phản hồi xúc giác, cho phép người chơi cảm nhận rung động hoặc tác động từ trò chơi, làm tăng cường cảm giác chân thực và hấp dẫn.

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

1.1.2 Phân loại game

a. Game Arcade

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

lòng về cả thị giác và thính giác, đồng thời thúc đẩy người chơi chinh phục những đỉnh cao mới trong hành trình của mình.



Hình 1.1: Hình ảnh máy chơi game Arcade

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

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

b. Game trên máy Console

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



Hình 1.2: Hình ảnh thiết bị chơi game Console

Các hệ máy chơi trò chơi điện tử (video game console), được phát triển bởi nhiều hãng sản xuất trên toàn cầu, thường được phân chia thành từng “thế hệ” dựa trên những điểm tương đồng về công nghệ phần cứng và tính năng phần mềm. Mỗi thế hệ thường giữ vai trò chủ đạo trên thị trường trong khoảng thời gian từ 5 đến 7 năm, trước khi được thay thế bởi thế hệ kế tiếp với những cải tiến kỹ thuật vượt trội hơn. Ngoài ra, các ứng dụng phát triển trên nền tảng này có thể được tăng cường hiệu năng đồ họa thông qua sự tích hợp với các thư viện chuyên dụng như OpenGL, Vulkan hoặc DirectX.

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

Một trò chơi máy tính cá nhân (PC game) là một trò chơi video được chơi trên một máy tính cá nhân chứ không phải trên một giao diện điều khiển. Trò chơi được điều khiển bằng các thiết bị PC đầu vào như bàn phím, chuột, phím

điều khiển, v.v. Game PC có thể chơi có hoặc không có kết nối Internet, và đã được đưa ra từ sự ra đời của máy tính cá nhân. Một số lượng lớn các trò chơi có sẵn cho nền tảng PC.



Hình 1.3: Hình ảnh thiết bị chơi game PC

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

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

d. Game trên nền tảng di động (Mobile Game)

Một báo cáo của Statista cho thấy Với hơn 5 tỷ người dùng điện thoại thông minh trên khắp thế giới, không có gì bất ngờ khi ngành phát triển game trên điện thoại di động đang phát triển mạnh. Việc sử dụng ứng dụng và sự thâm nhập của điện thoại thông minh vẫn đang tăng với tốc độ ổn định, không có bất kỳ dấu hiệu chậm lại nào trong tương lai gần.

Chúng kiến những thay đổi và nhận thấy những xu hướng này, rõ ràng là xu hướng game di động sẽ không sớm chậm lại trong tương lai gần và đến một lúc nào đó sẽ bỏ xa các nền tảng còn lại vì sự tiện ích cũng như tính giải trí cao.



Hình 1.4: Hình ảnh thiết bị chơi game Mobile

1.2 Các công cụ hỗ trợ làm game

1.2.1 Game Engines

– Unity:

+ Nền tảng: Đa nền tảng (Windows, macOS, Linux).

+ Đặc điểm: Dễ học, hỗ trợ 2D/3D, tích hợp sẵn hệ thống vật lý và script bằng C#.

- + Ứng dụng: Game di động, VR/AR, và game indie.
- Unreal Engine:
 - + Nền tảng: Đa nền tảng (Windows, macOS, Linux).
 - + Đặc điểm: Khả năng đồ họa đỉnh cao, dùng ngôn ngữ lập trình C++
 - + Ứng dụng: Game AAA, mô phỏng VR/AR, và phim hoạt hình.
- Godot Engine:
 - + Nền tảng: Đa nền tảng, mã nguồn mở.
 - + Đặc điểm: Gọn nhẹ, hỗ trợ 2D/3D, tích hợp GDScript (tương tự Python).
 - + Ứng dụng: Game indie và dự án giáo dục.
- CryEngine:
 - + Nền tảng: Windows.
 - + Đặc điểm: Đồ họa mạnh mẽ, phù hợp cho game AAA.
 - + Ứng dụng: Các game hành động hoặc FPS.

1.2.2 Công cụ lập trình (Programming Tools)

- Visual Studio/Visual Studio Code: IDE phổ biến cho các ngôn ngữ lập trình như C#, C++, Python. Tích hợp tốt với Unity và Unreal Engine.
- JetBrains Rider: IDE mạnh mẽ dành cho C#, hỗ trợ tốt khi làm việc với Unity.
- Git/GitHub/GitLab/Bitbucket: Công cụ quản lý mã nguồn và làm việc nhóm.
- PyCharm: IDE cho Python, thường được dùng với Godot hoặc các dự án indie.

1.2.3 Công cụ thiết kế đồ họa 2D (Graphics and Animation Tools)

- Photoshop/Illustrator: Dùng để thiết kế texture, nhân vật, và giao diện game.
- Spine: Dùng để tạo hoạt ảnh cho nhân vật, giao diện...

1.2.4 Công cụ âm thanh (Audio Tools)

- Audacity: Phần mềm mã nguồn mở để chỉnh sửa và tạo hiệu ứng âm thanh.
- FMOD: Công cụ tích hợp âm thanh tương tác trong game.
- Wwise: Một hệ thống âm thanh chuyên nghiệp, hỗ trợ âm thanh động trong game.

1.3 Tổng quan về thư viện SDL trong C++

1.3.1 Tổng quan về thư viện SDL

SDL là một thư viện lập trình đa nền tảng mã nguồn mở, được thiết kế để hỗ trợ phát triển các ứng dụng đa phương tiện như trò chơi video, mô phỏng đồ họa và các ứng dụng tương tác. Thư viện này đóng vai trò như một cầu nối giữa phần cứng và phần mềm, giúp nhà phát triển dễ dàng làm việc với các hệ thống âm thanh, đồ họa, đầu vào từ thiết bị ngoại vi, cũng như kết nối mạng mà không cần phải lo lắng về các chi tiết kỹ thuật phức tạp của từng nền tảng khác nhau.



Hình 1.5: Logo thư viện SDL

SDL được phát triển bởi Sam Lantinga và đội ngũ phát triển của ông tại Blizzard Entertainment. Thư viện này đã được sử dụng trong nhiều trò chơi nổi tiếng như Civilization IV, Half-Life, và Minecraft. SDL hỗ trợ nhiều nền tảng khác nhau như Windows, macOS, Linux, Android và iOS. Thư viện này cũng được viết bằng nhiều ngôn ngữ lập trình như C, C++, Python, Rust và Java. Một số tính năng của SDL bao gồm hỗ trợ đa cửa sổ, vẽ hình và trò chơi 2D/3D, hỗ trợ âm thanh và âm nhạc, hỗ trợ đầu vào từ bàn phím, chuột, cảm ứng và thiết bị điều khiển, hỗ trợ mạng và đa luồng, cũng như đa ngôn ngữ.

Ngoài ra, SDL cũng có thể tích hợp với các thư viện khác như OpenGL, Vulkan, và DirectX để cung cấp các chức năng đồ họa mạnh mẽ hơn cho các ứng dụng của bạn.

Với sự đa dạng và tính năng mạnh mẽ của mình, SDL là một lựa chọn tuyệt vời cho các nhà phát triển muốn phát triển các ứng dụng đa phương tiện trên nhiều nền tảng khác nhau.

SDL là một thư viện mã nguồn mở được thiết kế để cung cấp các chức năng đa phương tiện cho các ứng dụng trên nhiều nền tảng khác nhau. Các chức năng chính của SDL bao gồm:

- Đồ họa: SDL hỗ trợ vẽ hình và trò chơi 2D/3D thông qua việc sử dụng OpenGL, Vulkan hoặc DirectX. Nó cũng hỗ trợ đa cửa sổ, cho phép bạn tạo nhiều cửa sổ trong cùng một ứng dụng.
- Âm thanh: SDL hỗ trợ phát lại âm thanh và âm nhạc qua nhiều định dạng khác nhau, bao gồm WAV, MP3 và Ogg Vorbis. Nó cũng hỗ trợ đa kênh và âm lượng điều chỉnh.

- Đầu vào: SDL hỗ trợ đầu vào từ bàn phím, chuột, cảm ứng và thiết bị điều khiển. Nó cũng hỗ trợ nhận diện cử chỉ và âm thanh qua microphone.
- Mạng: SDL hỗ trợ kết nối mạng TCP/IP và UDP để cho phép các ứng dụng trò chơi trực tuyến.
- Đa luồng: SDL hỗ trợ đa luồng để giúp các nhà phát triển xử lý các tác vụ phức tạp một cách hiệu quả hơn.
- Đa ngôn ngữ: SDL hỗ trợ nhiều ngôn ngữ lập trình như C, C++, Python, Rust và Java.

Với những tính năng mạnh mẽ này, SDL đã được sử dụng trong nhiều ứng dụng đa phương tiện và trò chơi nổi tiếng trên nhiều nền tảng khác nhau. Nó còn được sử dụng rộng rãi trong giáo dục và nghiên cứu để giúp các nhà nghiên cứu phát triển các ứng dụng đa phương tiện và trò chơi.

1.3.2 Ưu nhược điểm của phần mềm

Ưu điểm:

SDL mang lại nhiều lợi ích đáng kể cho các nhà phát triển ứng dụng, đặc biệt là trong lĩnh vực phát triển trò chơi và các ứng dụng đa phương tiện. Một số ưu điểm chính của SDL bao gồm:

- ✓ Đa nền tảng: SDL hỗ trợ nhiều nền tảng khác nhau, từ các hệ điều hành máy tính như Windows, macOS, và Linux cho đến các nền tảng di động như Android và iOS. Khả năng đa nền tảng này giúp các nhà phát triển có thể tạo ra ứng dụng chạy mượt mà trên nhiều thiết bị mà không cần phải viết mã riêng biệt cho từng nền tảng, từ đó tiết kiệm thời gian và chi phí phát triển.

- ✓ Dễ sử dụng: SDL cung cấp các API đơn giản, trực quan, dễ học và dễ sử dụng, ngay cả với những người mới bắt đầu. Thư viện này cho phép nhà phát triển tập trung vào phần cốt lõi của ứng dụng mà không cần quan tâm quá nhiều đến việc xử lý chi tiết phần cứng hoặc các giao diện hệ thống phức tạp.
- ✓ Tích hợp dễ dàng: Một trong những lợi thế quan trọng của SDL là khả năng tích hợp mạnh mẽ với các thư viện khác như OpenGL, Vulkan, và DirectX. Điều này cho phép các nhà phát triển dễ dàng kết hợp SDL với các công nghệ đồ họa cao cấp để xây dựng các trò chơi và ứng dụng đồ họa phức tạp với hiệu suất cao.
- ✓ Hỗ trợ đa ngôn ngữ: SDL được viết bằng ngôn ngữ lập trình C, nhưng nó cũng có thể được sử dụng trong nhiều ngôn ngữ lập trình khác như C++, Python, Java, Rust, và nhiều ngôn ngữ khác. Điều này giúp các nhà phát triển có thể sử dụng ngôn ngữ mà họ quen thuộc nhất, tạo điều kiện thuận lợi cho việc phát triển ứng dụng.
- ✓ Tính linh hoạt: SDL cung cấp các chức năng linh hoạt và có thể tùy chỉnh, cho phép các nhà phát triển điều chỉnh và mở rộng các chức năng theo nhu cầu cụ thể của từng ứng dụng. Điều này giúp tăng cường khả năng sáng tạo trong quá trình phát triển và đảm bảo rằng ứng dụng có thể đáp ứng đầy đủ các yêu cầu kỹ thuật.

Ngoài những lợi ích cơ bản kể trên, SDL còn có một số tính năng và ứng dụng đặc biệt, giúp nó trở thành một công cụ hữu ích trong nhiều lĩnh vực:

1. SDL là một trong những thư viện được sử dụng rộng rãi trong ngành công nghiệp trò chơi để phát triển các trò chơi đa nền tảng. Với các chức năng đồ họa 2D/3D, xử lý âm thanh, điều khiển thiết bị đầu vào (như bàn phím,

chuột, gamepad) và hỗ trợ kết nối mạng, SDL giúp nhà phát triển dễ dàng tạo ra các trò chơi với hiệu suất cao trên nhiều nền tảng khác nhau. Hỗ trợ phát triển ứng dụng giáo dục: SDL cũng được sử dụng trong giáo dục để phát triển các ứng dụng đa phương tiện. Ví dụ, các ứng dụng giáo dục có thể sử dụng SDL để thể hiện các khái niệm hình ảnh và âm thanh một cách trực quan và sinh động.

2. SDL cũng đóng vai trò quan trọng trong lĩnh vực giáo dục, nơi nó được sử dụng để phát triển các ứng dụng đa phương tiện giúp truyền đạt kiến thức một cách trực quan và sinh động. Nhờ các khả năng xử lý đồ họa và âm thanh mạnh mẽ, các ứng dụng giáo dục có thể mô phỏng các khái niệm phức tạp bằng hình ảnh và âm thanh, giúp học sinh dễ dàng hiểu và tiếp thu kiến thức hơn.
3. Tính di động: SDL hỗ trợ phát triển các ứng dụng trên thiết bị di động, như điện thoại thông minh và máy tính bảng. Điều này cho phép các nhà phát triển dễ dàng mở rộng ứng dụng của mình sang các nền tảng di động mà không cần phải viết mã hoàn toàn mới. Với sự phổ biến của các thiết bị di động ngày nay, SDL trở thành một công cụ hữu ích cho việc phát triển các ứng dụng linh hoạt và đa năng.
4. Cộng đồng phát triển: SDL có một cộng đồng phát triển lớn và tích cực. Các thành viên trong cộng đồng thường xuyên chia sẻ tài liệu, ví dụ mã nguồn, và hỗ trợ cho các nhà phát triển khác. Sự hỗ trợ này giúp việc học và sử dụng SDL trở nên dễ dàng hơn, đặc biệt đối với những người mới bắt đầu.

SDL là một thư viện đa phương tiện mạnh mẽ và linh hoạt, được sử dụng rộng rãi trong nhiều lĩnh vực như phát triển trò chơi, giáo dục, nghiên cứu khoa học, và ứng dụng di động. Với khả năng hỗ trợ đa nền tảng, tính dễ sử dụng và tích hợp mạnh mẽ, SDL giúp nhà phát triển tiết kiệm thời gian và công sức trong

quá trình phát triển ứng dụng. Thư viện này không chỉ phù hợp cho các dự án nhỏ mà còn là lựa chọn tuyệt vời cho các sản phẩm lớn với yêu cầu kỹ thuật cao. SDL tiếp tục là một trong những công cụ quan trọng cho những ai muốn xây dựng các ứng dụng đa phương tiện chuyên nghiệp và hiệu quả.

Nhược điểm:

Mặc dù SDL là một thư viện đa phương tiện mạnh mẽ và đa năng, nhưng như bất kỳ công nghệ nào khác, nó cũng tồn tại một số nhược điểm cần xem xét khi lựa chọn sử dụng trong phát triển ứng dụng hoặc trò chơi. Một số hạn chế của SDL bao gồm:

- **Thiếu tính năng đồ họa cao cấp:** Mặc dù SDL hỗ trợ tốt đồ họa 2D, nhưng khi nói đến đồ họa 3D và các hiệu ứng đặc biệt, SDL không thể so sánh với các thư viện chuyên dụng hơn như OpenGL, Vulkan, hoặc các công cụ mạnh mẽ như Unity hay Unreal Engine. SDL chủ yếu hướng đến các trò chơi và ứng dụng sử dụng đồ họa 2D cơ bản, vì vậy các nhà phát triển cần kết hợp SDL với các thư viện đồ họa khác để có thể đạt được hiệu suất và tính năng cao hơn trong đồ họa 3D.
- **Không hỗ trợ đa luồng đầy đủ:** Một nhược điểm khác của SDL là khả năng hỗ trợ đa luồng (multithreading) không đầy đủ. Điều này có thể làm giảm hiệu suất của ứng dụng khi có nhiều sự kiện hoặc quá trình xử lý xảy ra đồng thời. Các ứng dụng yêu cầu sự phân tán công việc ra nhiều luồng xử lý có thể gặp phải giới hạn khi sử dụng SDL. Mặc dù SDL cung cấp một số cơ chế cơ bản cho việc tạo và quản lý luồng, nhưng khả năng này không thực sự mạnh mẽ hoặc linh hoạt như trong các thư viện khác.
- **Khó khăn trong việc xử lý âm thanh:** SDL cung cấp các chức năng xử lý âm thanh cơ bản, bao gồm phát nhạc và hiệu ứng âm thanh. Tuy nhiên, nếu ứng dụng cần xử lý các tính năng âm thanh phức tạp hơn như xử lý kỹ thuật số, khuếch đại, hoặc âm thanh vòm, SDL có thể không đáp ứng đầy đủ. Để

đạt được các yêu cầu âm thanh phức tạp, các nhà phát triển thường phải sử dụng thêm các thư viện xử lý âm thanh chuyên nghiệp như FMOD, OpenAL, hoặc PortAudio để mở rộng chức năng.

- Không hỗ trợ tốt trên các nền tảng di động: Mặc dù SDL có khả năng hỗ trợ phát triển ứng dụng trên Android và iOS, nhưng nó không thực sự tối ưu cho các nền tảng di động như các công cụ phát triển chuyên dụng khác như Unity hoặc Unreal Engine. Những công cụ này cung cấp nhiều tính năng vượt trội dành riêng cho di động, từ việc tối ưu hóa hiệu suất cho đến hỗ trợ giao diện người dùng và tích hợp sẵn các tính năng cảm biến của thiết bị di động.
- Khó khăn trong việc tìm hiểu và sử dụng: Đối với các nhà phát triển mới bắt đầu, SDL có thể trở nên phức tạp và khó hiểu do cú pháp và các đặc tính khác nhau của nó. Trong khi nhiều thư viện hiện đại hướng đến sự dễ sử dụng và tài liệu phong phú, SDL có thể yêu cầu một chút kiên nhẫn và hiểu biết về lập trình cấp thấp. Những người mới làm quen với phát triển đa phương tiện hoặc trò chơi có thể gặp khó khăn trong việc tiếp cận thư viện này và phải dành nhiều thời gian để làm quen.

SDL là một thư viện đa phương tiện mạnh mẽ và linh hoạt, tuy nhiên, nó cũng có một số nhược điểm cần xem xét. Thư viện này không phù hợp với các ứng dụng đòi hỏi khả năng xử lý đồ họa 3D cao cấp hoặc đa luồng phức tạp. Hơn nữa, việc xử lý âm thanh nâng cao và hỗ trợ tối ưu cho các nền tảng di động cũng chưa thực sự mạnh mẽ so với các công cụ hiện đại khác. Ngoài ra, SDL có thể gây khó khăn trong quá trình học và sử dụng đối với những nhà phát triển mới.

Tuy nhiên, đối với các dự án tập trung vào đồ họa 2D, yêu cầu khả năng đa nền tảng, và không quá phức tạp về đồ họa hay âm thanh, SDL vẫn là một lựa chọn tốt. Điều này đặc biệt đúng với các dự án nhỏ hoặc vừa, nơi tính dễ tiếp cận và hiệu suất được ưu tiên. SDL cũng có cộng đồng hỗ trợ mạnh mẽ, giúp cung cấp tài liệu và trợ giúp khi cần thiết.

Dưới đây là một số trò chơi phổ biến được phát triển bằng SDL:

1. SuperTuxKart: Một trò chơi đua xe kart miễn phí và mã nguồn mở với nhiều nhân vật và đường đua.



Hình 1.6: Tựa game SuperTuxKart

2. Hedgewars: Một trò chơi chiến thuật theo lượt miễn phí và mã nguồn mở với những chú nhím làm nhân vật chính.



Hình 1.7: Tựa game Hedgewars

3. Battle for Wesnoth: Một trò chơi chiến thuật theo lượt miễn phí và mã nguồn mở với nội dung liên quan đến thể loại phong cách huyền ảo.



Hình 1.8: Tựa game Battle for Wesnoth

4. Aquaria: Một trò chơi phiêu lưu hành động thương mại với nhân vật giống như tiên cá khám phá một thế giới dưới nước.



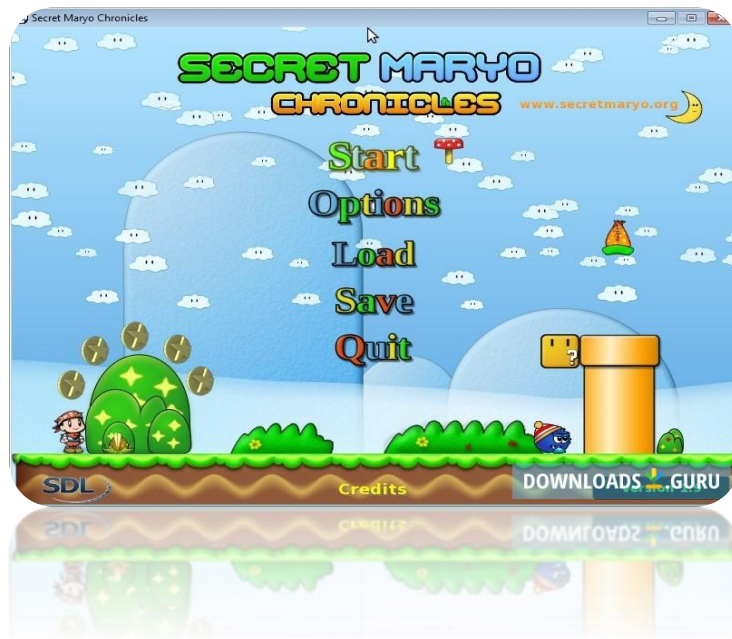
Hình 1.9: Tựa game Aquaria

5. Neverball: Một trò chơi đổ miễn phí và mã nguồn mở với một quả bóng lăn qua các chướng ngại vật khác nhau.



Hình 1.10: Tựa game Neverball

6. Secret Maryo Chronicles: Một trò chơi nền tảng miễn phí và mã nguồn mở với nhân vật giống như Mario điều khiển qua các cấp độ khác nhau.



Hình 1.11: Tựa game Secret Maryo Chronicles

7. Lugaru: Một trò chơi hành động thương mại với nhân vật giống như thỏ tham gia vào các trận chiến tay đôi.



Hình 1.12: Tựa game Lugaru

Các trò chơi này cho thấy tính linh hoạt của SDL và khả năng hỗ trợ nhiều thể loại và phong cách trò chơi khác nhau.

CHƯƠNG 2. THIẾT KẾ GAME

2.1 Giới thiệu tổng quan

2.1.1 Thông tin game

Blackout Ops là một trò chơi sinh tồn đầy hấp dẫn trong thế giới 2D, lấy cảm hứng từ tựa game huyền thoại Contra. Người chơi sẽ hóa thân vào vai một nhân vật trong thế giới Fantasy, bắt đầu hành trình khám phá vùng đất mới với những thử thách khắc nghiệt. Trên con đường phiêu lưu này, bạn sẽ đối mặt với vô số quái vật nguy hiểm, mỗi con quái mang theo những đặc tính riêng biệt và trở thành trở ngại chính trên hành trình của bạn. Nhiệm vụ của bạn không chỉ là tiêu diệt tất cả kẻ thù mà còn thu thập những vật phẩm quý giá, giúp nâng cấp vũ khí và trang bị của mình để tăng cường sức mạnh, khả năng chiến đấu và vượt qua các thử thách khó khăn hơn.

Trong Blackout Ops, độ khó của trò chơi không ngừng tăng lên theo thời gian. Ban đầu, người chơi có thể dễ dàng tiêu diệt quái vật, nhưng càng về sau, chúng sẽ trở nên mạnh mẽ hơn, tấn công nhanh và khó lường hơn. Ngoài việc tăng tốc độ ra đòn, quái vật còn được trang bị nhiều máu hơn, đòi hỏi người chơi phải nhanh nhẹn và có chiến thuật khéo léo để vượt qua. Điều này tạo ra một trải nghiệm căng thẳng và đầy kịch tính khi người chơi phải liên tục điều chỉnh lối chơi của mình để thích ứng với các thách thức ngày càng lớn.

Blackout Ops cũng thách thức khả năng quản lý thời gian của người chơi. Bạn sẽ có 300 giây để hoàn thành mỗi màn chơi. Nếu thời gian kết thúc mà bạn chưa vượt qua được thử thách, bạn sẽ bị tính là thua cuộc, điều này tạo thêm áp lực và yêu cầu người chơi phải nhanh chóng đưa ra quyết định trong mỗi hành động. Tuy nhiên, đừng lo lắng, sau mỗi màn chơi đầy thử thách và căng thẳng, người chơi sẽ nhận được những phần thưởng xứng đáng. Những phần thưởng này có thể là tiền, vũ khí mới, hoặc các vật phẩm hỗ trợ, giúp người chơi chuẩn bị tốt hơn cho những màn chơi tiếp theo.

Với lối chơi đầy cuốn hút và sự kết hợp giữa yếu tố hành động và chiến thuật, Blackout Ops hứa hẹn mang đến những phút giây giải trí thú vị, đầy thách thức và không kém phần hồi hộp. Đây chắc chắn là một tựa game không thể bỏ qua đối

với những người yêu thích thể loại sinh tồn trong thế giới 2D cổ điển nhưng vẫn đòi hỏi kỹ năng và tính chiến thuật cao.

2.1.2 Thể loại game và yếu tố game

Đây là thể loại game đi ả (platformer) với lối chơi tập trung vào việc tiêu diệt quái vật và thu thập các vật phẩm để tích điểm. Người chơi sẽ phiêu lưu qua nhiều cấp độ khác nhau, đối đầu với các thử thách liên tục gia tăng về độ khó và yêu cầu người chơi không chỉ kỹ năng chiến đấu mà còn cả sự nhạy bén trong việc né tránh và phòng thủ.

Game mang lại nhiều yếu tố như:

- ✓ Tăng khả năng quan sát và sự nhanh nhẹn: Trong mỗi màn chơi, người chơi cần rèn luyện khả năng quan sát nhanh nhạy để nhận diện đòn tấn công từ quái vật. Việc nhanh chóng né tránh đạn và các chướng ngại vật sẽ là yếu tố quan trọng để tồn tại và tiếp tục cuộc hành trình. Độ khó cao, yêu cầu người chơi tập trung tỉ mỉ vào từng hành động
- ✓ Độ khó cao, yêu cầu tập trung: Mức độ khó của game được thiết kế tăng dần, đòi hỏi người chơi phải có sự tập trung cao độ vào từng hành động nhỏ nhất. Những pha hành động căng thẳng với các đòn tấn công nhanh và khó lường từ quái vật sẽ buộc người chơi phải tính toán từng bước di chuyển một cách cẩn thận.
- ✓ Kỹ năng điều khiển nhân vật: Một trong những yếu tố quyết định thành công trong Blackout Ops chính là khả năng điều khiển nhân vật của người chơi. Bạn sẽ phải khéo léo nhảy qua các chướng ngại vật, tránh đòn của đối thủ, và di chuyển một cách mượt mà trong khi tìm cách tiêu diệt kẻ thù. Đặc biệt, mỗi cấp độ sẽ kết thúc bằng một trận đấu với boss – nhân vật mạnh mẽ nhất trong màn chơi, đòi hỏi bạn phải sử dụng mọi kỹ năng điều khiển để giành chiến thắng.
- ✓ Sự kiên nhẫn và bền chí: Với mức độ thử thách ngày càng gia tăng, người chơi phải có sự kiên nhẫn để vượt qua những màn chơi khó khăn và không bỏ

cuộc khi gặp phải những tình huống phức tạp. Blackout Ops không chỉ là một trò chơi về hành động mà còn là một bài kiểm tra về sự bền chí và lòng quyết tâm.

2.1.3 Đối tượng chơi

Với lối chơi game đơn giản nhưng yêu cầu sự tập trung cao độ và khả năng điều khiển khéo léo game hiện tại đang phát triển tới đối tượng:

- ✓ Trẻ em từ 10 tuổi trở lên: Trò chơi không chỉ mang tính giải trí mà còn giúp trẻ rèn luyện sự kiên trì và tập trung. Qua mỗi màn chơi, trẻ em sẽ học cách đối phó với khó khăn, thử thách sự nhạy bén và tăng cường khả năng điều khiển, xử lý tình huống. Với độ khó tăng dần và các phần thưởng xứng đáng khi vượt qua các chương ngại, trò chơi giúp trẻ rèn luyện tính kiên nhẫn và tư duy logic một cách tự nhiên.
- ✓ Người lớn muốn xả stress hoặc chơi cùng con: Đối với người lớn, Blackout Ops là một công cụ hữu ích để thư giãn sau những giờ làm việc căng thẳng. Lối chơi nhẹ nhàng, không quá phức tạp nhưng đòi hỏi sự khéo léo và tập trung, giúp họ quên đi áp lực hàng ngày và giải tỏa stress. Trò chơi cũng là một hoạt động thú vị để cha mẹ và con cái có thể cùng nhau trải nghiệm, gắn kết thông qua những thử thách chung, tạo ra những khoảnh khắc vui vẻ và ý nghĩa trong gia đình.

2.1.4 Nền tảng

Game được xây dựng bằng SDL (thư viện của ngôn ngữ C++), cho phép phát triển các ứng dụng đa phương tiện, đặc biệt là trò chơi, với hiệu suất cao. SDL cung cấp các công cụ cần thiết để xử lý đồ họa, âm thanh, và tương tác người dùng, giúp game có khả năng chạy mượt mà và ổn định trên các thiết bị.

Nhờ sử dụng SDL, Blackout Ops có thể phát triển chủ yếu trên nền tảng Windows, bao gồm Laptop và PC, nơi mà SDL hoạt động hiệu quả nhất. SDL

cung cấp hỗ trợ mạnh mẽ cho các hệ thống sử dụng Windows, từ đó đảm bảo rằng game có thể khai thác tối đa sức mạnh của phần cứng, bao gồm xử lý đồ họa và âm thanh. Điều này làm cho trò chơi có thể tận dụng khả năng của máy tính để tạo ra trải nghiệm chơi game với hiệu suất cao, hình ảnh đẹp mắt và âm thanh sống động.

Ngoài ra, với nền tảng SDL, game cũng có tiềm năng mở rộng sang các nền tảng khác như Linux hoặc macOS nếu cần thiết, nhờ tính đa nền tảng của thư viện này. Tuy nhiên, trong giai đoạn hiện tại, trọng tâm phát triển của Blackout Ops là nhắm đến người dùng Windows, nơi phần lớn người chơi sử dụng các thiết bị Laptop và PC để trải nghiệm game.

SDL không chỉ giúp game chạy mượt mà mà còn giảm thiểu phức tạp trong quá trình phát triển, cho phép nhà phát triển tập trung vào cải thiện lối chơi và chất lượng trải nghiệm của người chơi.

2.2 Thiết kế kịch bản game

2.2.1 Mô tả

Là một trò chơi thể loại đi ải, platformer, với lối chơi tập trung vào việc vượt qua các thử thách, tiêu diệt quái vật và thu thập vật phẩm. Nhân vật chính của trò chơi được xây dựng trong bối cảnh fantasy – thế giới huyền ảo đầy màu sắc và kỳ bí, nơi người chơi nhập vai để phiêu lưu, chiến đấu, và khám phá. Điểm đặc biệt của trò chơi nằm ở sự thiết kế quái vật với hình ảnh hoạt hình gần gũi và dễ thương, tạo sự thân thiện và kích thích hứng thú cho người chơi, đặc biệt là những đối tượng trẻ tuổi.

Mục tiêu chính của trò chơi là tiêu diệt toàn bộ quái vật xuất hiện trên đường đi và thu thập các vật phẩm có giá trị, giúp người chơi nâng cấp vũ khí, trang bị để chuẩn bị cho trận chiến cuối cùng với boss – kẻ thù mạnh nhất trong màn chơi. Sau khi vượt qua các con quái và hoàn thành nhiệm vụ, người chơi phải đối mặt với boss để đạt được chiến thắng cuối cùng.

Một yếu tố hấp dẫn khác của Blackout Ops là góc nhìn trong trò chơi. Thay vì góc nhìn từ trên xuống hoặc góc nhìn thứ ba như trong nhiều trò chơi platformer truyền thống, Blackout Ops được chơi ở góc nhìn thứ nhất. Điều này tạo ra trải nghiệm trực quan và đắm chìm hơn, khi người chơi cảm nhận như chính mình

đang phiêu lưu trong thế giới ảo 3D. Góc nhìn này không chỉ tạo ra sự hấp dẫn mà còn giúp kích thích khả năng quan sát của người chơi, khi họ cần chú ý đến mọi chi tiết xung quanh để né tránh các đòn tấn công và tìm cách tiêu diệt quái vật một cách hiệu quả.

Sự kết hợp giữa đồ họa 3D và góc nhìn thứ nhất mang lại cảm giác chân thực, tạo ra trải nghiệm sống động, giúp người chơi dễ dàng hòa mình vào thế giới trong game và đối mặt với các thử thách đầy hứng thú. Blackout Ops không chỉ đơn thuần là một trò chơi hành động, mà còn đòi hỏi sự khéo léo, tập trung, và chiến thuật từ người chơi để hoàn thành nhiệm vụ và giành chiến thắng.

2.2.2 Thiết kế luật chơi

Mỗi màn chơi gồm tập hợp các quái vật, quái vật gồm có hai loại chính là quái vật cận chiến và quái vật đánh xa. Quái vật cận chiến sẽ tấn công người chơi khi nhân vật đứng ngay bên cạnh chúng. Chúng thường có xu hướng áp sát nhanh và thực hiện các đòn tấn công mạnh mẽ ở cự ly gần, buộc người chơi phải di chuyển linh hoạt và phản ứng nhanh để tránh bị tấn công. Quái vật đánh xa có khả năng tấn công từ khoảng cách xa hơn, thường là trong phạm vi 640 pixel so với vị trí của nhân vật. Chúng sẽ bắn ra các loại đạn tấn công, tạo ra một thách thức lớn hơn khi người chơi phải né tránh không chỉ đòn tấn công của chúng mà còn phải tìm cách tiếp cận và tiêu diệt chúng từ khoảng cách an toàn. Mỗi màn chơi sẽ có một boss xuất hiện ở cuối bản đồ, thường mạnh hơn rất nhiều so với các quái vật thông thường. Boss có thể kết hợp cả khả năng tấn công cận chiến và tấn công tầm xa, tạo nên trận đấu kịch tính và đầy thử thách cho người chơi.

Người chơi có thể tiêu diệt quái vật bằng cách bắn đạn vào chúng, và quái vật cũng có thể tấn công lại. Quái vật đánh gần thì sẽ tấn công khi nhân vật đứng ngay bên cạnh nó. Còn quái vật đánh xa sẽ bắn ra đạn trong khoảng cách 640 pixel.

Mỗi màn chơi sẽ có 3 mạng và có giới hạn thời gian là 300 giây để người chơi hoàn thành màn.

Điều kiện thắng:

- ✓ Người chơi phải tiêu diệt được boss ở cuối map sẽ được tính là chiến thắng.

Điều kiện thua:

Nếu người chơi vi phạm một trong hai điều kiện dưới đây sẽ được tính là thua:

- ✓ Bị hết tất cả mạng chơi trước khi tiêu diệt được boss
- ✓ Hết thời gian quy định mà vẫn chưa tiêu diệt được boss

Mỗi màn chơi khi thua cuộc sẽ có hai lựa chọn là thoát ra khỏi màn hình(bấm nút ESC) hoặc chơi lại màn chơi đó(bấm nút nút ESC) hoặc chơi lại màn chơi đó(bấm nút SPACE)



Hình 2.1: Giao diện game Blackout Ops (lose game)

2.2.3 Thiết kế các màn của game

Blackout Ops chỉ có một màn chơi duy nhất nhưng được chia thành 3 bản đồ (map) khác nhau, mang lại sự đa dạng và thách thức liên tục cho người chơi. Mỗi bản đồ đều có thiết kế độc đáo với sự sắp xếp khác nhau về vị trí của quái vật và vật phẩm, đồng thời độ khó cũng sẽ tăng dần từ map đầu tiên đến map thứ hai và map thứ ba để tạo thêm thử thách.

2.3 Thiết kế các vật phẩm thu thập

Mỗi map sẽ có những vật phẩm khác nhau map càng khó vật phẩm nhận được sẽ càng giá trị. Các vật phẩm bao gồm

- Đồng tiền vàng:

- ✓ Thu thập bằng cách nhặt trên đường đi ở map thứ nhất
- ✓ Có giá trị tương đương với 1 đồng



Hình 2.2: Hình đồng tiền vàng

- Đồng tiền vàng bản nâng cấp:

- ✓ Thu thập bằng cách nhặt trên đường đi ở map thứ nhất



Hình 2.3: Hình đồng tiền trong game

- ✓ Có giá trị tương đương với 2 đồng

- Kim cương:

- ✓ Thu thập bằng cách nhặt trên đường đi ở map thứ nhất
- ✓ Có giá trị tương đương với 3 đồng



Hình 2.4: Hình kim cương trong game

2.4 Thiết kế cách tính điểm

Blackout Ops cung cấp điểm số cho người chơi để tăng cảm giác đạt được thành tựu qua mỗi màn, với công thức tính điểm như sau:

- ✓ **Mỗi lần tiêu diệt được quái vật nhỏ:** (bao gồm quái vật đánh cận chiến và quái vật đánh xa) sẽ được 1 điểm.

✓ **Điểm cho thời gian thừa:** Mỗi 10 giây người chơi còn thừa người chơi sẽ nhận được 2 điểm cộng.

✓ **Khi tiêu diệt được boss:** sẽ được 50 điểm cộng

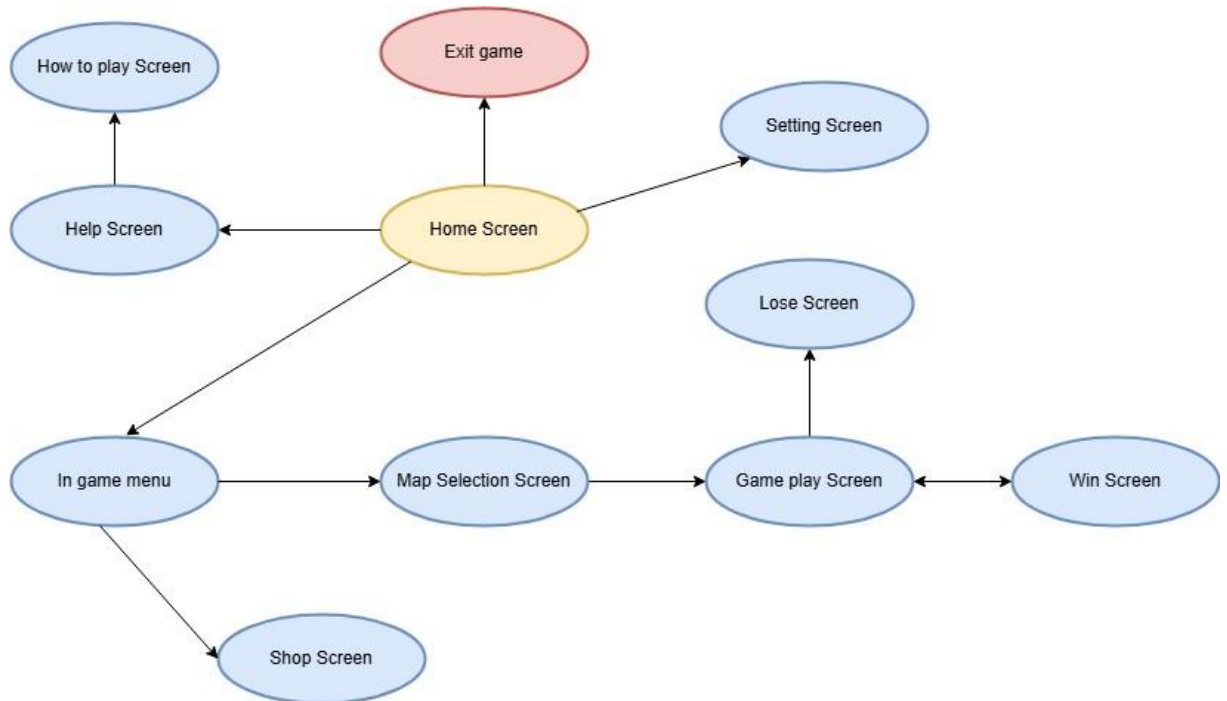
Tổng điểm = số lần tiêu diệt quái vật + điểm cho thời gian còn lại + điểm tiêu diệt được boss

2.5 Thiết kế tương tác và điều khiển game

Trong Blackout Ops, tất cả các thao tác điều khiển trò chơi đều được thực hiện thông qua bàn phím và chuột trên laptop hoặc PC, giúp mang lại trải nghiệm điều khiển trực quan và đơn giản cho người chơi. Cụ thể, người chơi có thể sử dụng nút A để di chuyển sang trái và nút D để di chuyển sang phải, tạo sự linh hoạt trong việc điều hướng nhân vật. Bên cạnh đó, thao tác tấn công được thực hiện bằng cách nhấp chuột trái, cho phép người chơi bắn đạn để tiêu diệt quái vật trên đường đi. Sự kết hợp giữa các phím di chuyển và chuột giúp người chơi vừa có thể điều khiển nhân vật linh hoạt, vừa có thể tấn công kẻ thù một cách nhanh chóng và chính xác. Thiết kế điều khiển đơn giản này không chỉ giúp người chơi dễ dàng làm quen mà còn tăng cường trải nghiệm nhập vai trong trò chơi.

2.6 Storyboard

2.6.1 Sơ đồ các màn hình



Hình 2.5: Sơ đồ các màn hình

2.6.2 Màn hình bắt đầu






- ✓ Bên trái là nút HELP đây là hướng dẫn chơi game
- ✓ Nút PLAY để bắt đầu chơi game

- ✓ Nút ESC để thoát game



Hình 2.6: Giao diện trang chủ của game

2.6.3 Màn hình chơi game

- ✓ Số mạng 
- ✓ Số tiền thu thập được 
- ✓ Số điểm đang có 
- ✓ Thời gian còn lại 
- ✓ Tạm dừng game 
- ✓ Thoát ra ngoài

Hình 2.7: Vị trí các nút trong trận đấu game



2.6.4 Màn hình hướng dẫn chơi game

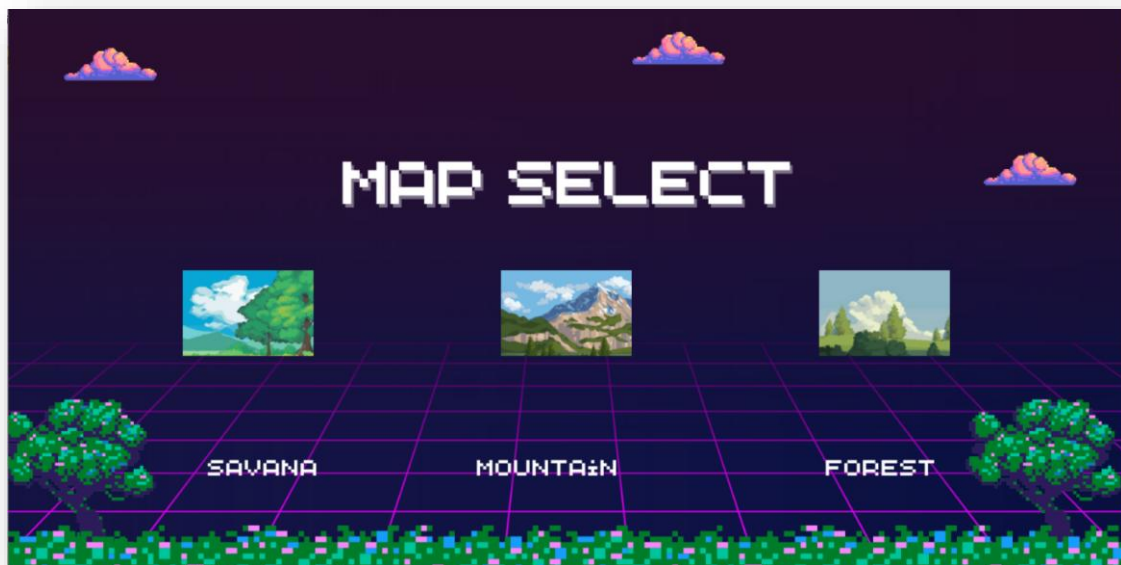
Màn hình giới thiệu về trò chơi và hướng dẫn di chuyển cho người chơi.



Hình 2.8: Màn hình hướng dẫn chơi game

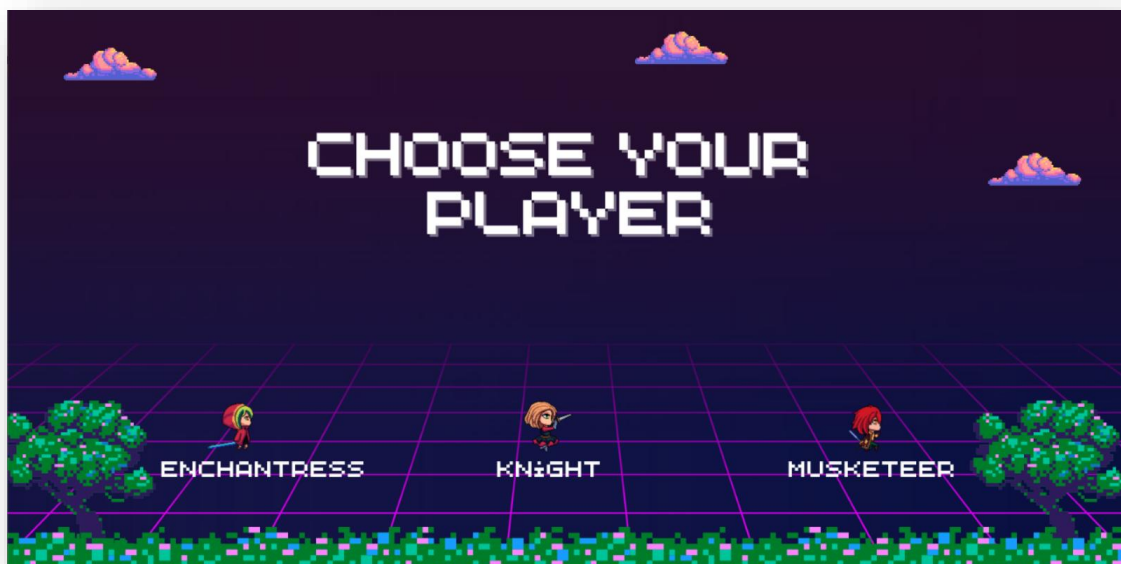
2.6.5 Màn hình chọn bản đồ và chọn nhân vật

Có 3 bản đồ(map) cho người chơi lựa chọn đó là SAVANA, MOUNTAIN và FOREST, 3 map này có độ khó và phần thưởng tăng dần từ dễ đến trung bình và khó và map FOREST là được hoàn thiện nhất trong 3 map yêu cầu kĩ năng của người chơi cao nhất.



Hình 2.9. Giao diện chọn bản đồ

Có 3 nhân vật cho người chơi lựa chọn đó là ENCHANTRESS, KNIGHT và MUSKETEER, 3 nhân vật này có sát thương và độ khó tương đương nhau, người chơi có thể trải nghiệm cả 3 nhân vật để tăng sự hứng thú khi chơi game.



Hình 2.10: Giao diện chọn nhân vật

2.6.6 Màn hình khi thua

Khi thua có 2 lựa chọn đó là SPACE để chơi lại hoặc ESC để thoát ra ngoài màn hình chính.



Hình 2.11: Màn hình khi thua

CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ

3.1 Xây dựng giao diện game

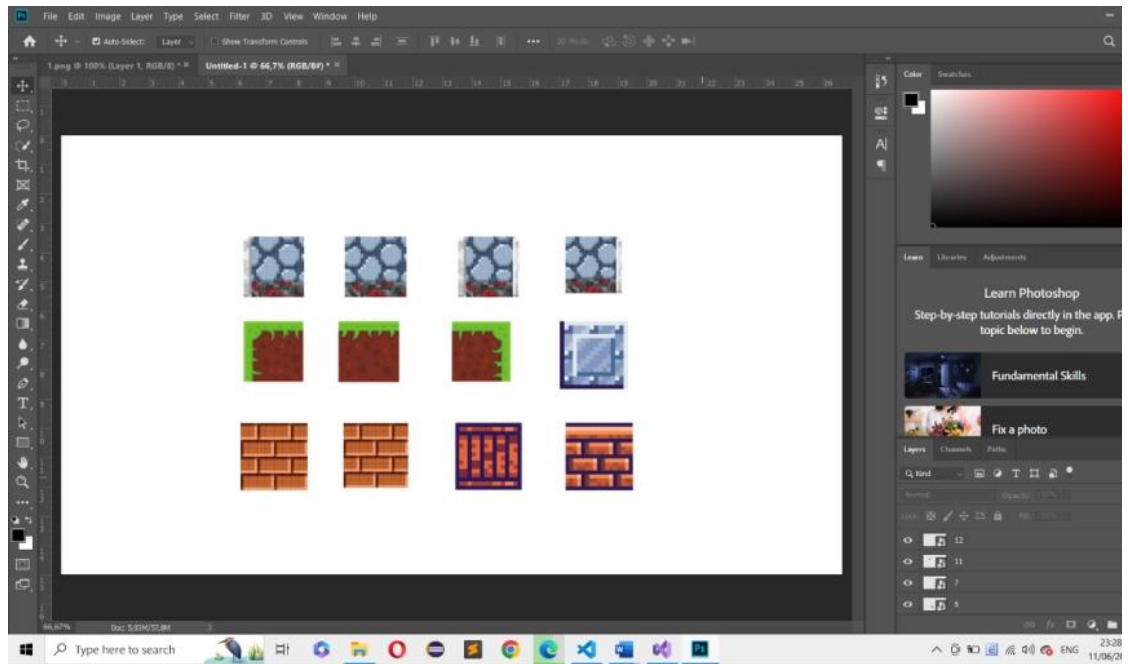
Các kỹ thuật sử dụng để tạo lập giao diện cho game Blackout Ops

- Thực hiện thiết kế giao diện với phần mềm Adobe Photoshop sử dụng tài nguyên hình ảnh ở trên:



Hình 3.1: Xây dựng BackgroundMap

- Xây dựng BackgroundMap
- Xây dựng các block



Hình 3.2: Xây dựng các block

3.1.1 Singleton Pattern

Singleton	
-	<u>singleton : Singleton</u>
-	Singleton()
+	<u>getInstance() : Singleton</u>

Hình 3.3: Singleton Pattern

Singleton Pattern là một mẫu thiết kế phần mềm trong đó chỉ có một thể hiện duy nhất của một lớp được tạo ra trong suốt vòng đời của ứng dụng. Điều này rất quan trọng đối với các hệ thống quản lý phức tạp, đặc biệt trong phát triển game, nơi cần sự điều phối đồng bộ và nhất quán giữa nhiều thành phần khác nhau. Việc

sử dụng Singleton Pattern giúp kiểm soát tốt hơn các đối tượng cần được quản lý chặt chẽ, đảm bảo rằng chúng luôn tồn tại và hoạt động một cách có trật tự.

Trong phát triển game, Singleton Pattern thường được ứng dụng vào các đối tượng quản lý toàn bộ trạng thái trò chơi như GameManager. GameManager là một đối tượng chịu trách nhiệm điều phối toàn bộ hệ thống, từ việc điều khiển cấp độ, quản lý logic trò chơi đến xử lý các sự kiện trong suốt quá trình chơi. Với Singleton Pattern, việc chỉ có duy nhất một GameManager giúp đảm bảo tất cả các thành phần trong game được điều phối một cách nhất quán, tránh việc xung đột hoặc quản lý không đồng bộ.

AudioManager cũng là một ví dụ phổ biến của việc áp dụng Singleton Pattern. AudioManager quản lý toàn bộ hệ thống âm thanh của trò chơi, bao gồm nhạc nền, hiệu ứng âm thanh và điều chỉnh âm lượng. Khi sử dụng Singleton Pattern, sẽ chỉ có một thể hiện của AudioManager trong toàn bộ game, điều này giúp kiểm soát âm thanh hiệu quả hơn, tránh việc nhiều luồng âm thanh chạy cùng lúc gây ra lỗi đồng bộ hay xung đột âm thanh. Nhờ vậy, trải nghiệm âm thanh trong trò chơi sẽ trở nên mượt mà và dễ quản lý hơn.

DataManager, với nhiệm vụ quản lý việc lưu trữ và đọc dữ liệu của người chơi, cũng là một đối tượng thích hợp để sử dụng Singleton Pattern. Bằng cách giới hạn chỉ có một DataManager, việc lưu trữ và truy xuất dữ liệu được thực hiện một cách đồng nhất và an toàn. Điều này đặc biệt quan trọng khi xử lý các tệp lưu game, bởi nếu có nhiều đối tượng cùng quản lý dữ liệu, dễ dẫn đến lỗi ghi đè không mong muốn hoặc dữ liệu không nhất quán.

Singleton Pattern mang lại nhiều lợi ích trong việc đảm bảo tính duy nhất của các đối tượng quan trọng trong game. Với mỗi đối tượng như GameManager, AudioManager hay DataManager, việc giới hạn số lượng thể hiện sẽ giúp việc quản lý và điều phối các tài nguyên hiệu quả hơn. Người phát triển có thể dễ dàng truy cập các đối tượng này từ bất kỳ đâu trong trò chơi mà không cần phải khởi tạo lại, giúp tiết kiệm tài nguyên và tăng cường khả năng quản lý tổng thể.

3.1.2 Factory Pattern

Khi phát triển các trò chơi, việc tạo ra các popup như popup win game, popup lose game, hay popup shop là một phần quan trọng để tương tác với người chơi.

Tuy nhiên, nếu bạn khởi tạo sẵn những đối tượng này ngay từ đầu game và để chúng tồn tại trong suốt thời gian chơi, điều này có thể dẫn đến việc lãng phí tài nguyên RAM của người dùng, đặc biệt là nếu các popup này không được sử dụng thường xuyên. Đây là một ví dụ điển hình về việc không tối ưu hóa tài nguyên trong quá trình phát triển game.

Thực tế, trong hầu hết các trò chơi, bạn không thể biết trước chính xác khi nào người chơi sẽ cần sử dụng các popup này. Ví dụ, popup win game hoặc popup lose game chỉ xuất hiện khi người chơi hoàn thành hoặc thất bại một màn chơi, trong khi popup shop chỉ xuất hiện khi người chơi muốn mua vật phẩm. Chính vì vậy, việc khởi tạo những đối tượng này ngay từ đầu có thể khiến tài nguyên máy tính bị chiếm dụng một cách không cần thiết, đặc biệt nếu những popup này không được sử dụng trong suốt quá trình chơi.

Một giải pháp hiệu quả để giải quyết vấn đề này là khởi tạo đối tượng popup khi thực sự cần thiết trong runtime. Điều này có nghĩa là thay vì tạo ra các đối tượng popup ngay từ khi bắt đầu game, bạn sẽ chỉ tạo và hiển thị chúng khi có sự kiện xảy ra, chẳng hạn như khi người chơi chiến thắng hoặc thua cuộc, hoặc khi họ truy cập vào cửa hàng. Bằng cách này, bạn có thể tiết kiệm bộ nhớ và tối ưu hóa hiệu suất của trò chơi, giúp giảm tải cho máy tính của người chơi và cải thiện trải nghiệm tổng thể.

Cách tiếp cận này cũng cho phép bạn dễ dàng quản lý các đối tượng popup, khi chúng chỉ xuất hiện và chiếm dụng tài nguyên trong thời gian ngắn mà chúng thực sự cần thiết. Sau khi popup đã hoàn thành vai trò của nó (ví dụ, người chơi đã đóng popup shop sau khi mua sắm), đối tượng này có thể được giải phóng khỏi bộ nhớ, giúp trò chơi tiếp tục chạy một cách mượt mà mà không bị gánh nặng bởi các đối tượng không sử dụng.

Với cách khởi tạo popup trong runtime, bạn có thể linh hoạt và chủ động hơn trong việc quản lý tài nguyên, đồng thời mang đến một trò chơi tối ưu hơn cho người dùng. Điều này đặc biệt quan trọng trong các trò chơi có quy mô lớn hoặc trên các nền tảng di động, nơi bộ nhớ và hiệu suất luôn là yếu tố cần được tối ưu cẩn thận.

3.1.3 Observer

Observer Pattern là một mẫu thiết kế thường được sử dụng trong việc quản lý các sự kiện liên quan đến nhiều đối tượng mà không cần tạo ra sự phụ thuộc chặt chẽ giữa chúng. Ứng dụng của Observer Pattern trong phát triển game rất phổ biến, đặc biệt khi một sự kiện trong trò chơi có thể dẫn đến hàng loạt hành động liên quan khác nhau. Mẫu này cho phép các đối tượng giao tiếp và phản ứng với nhau một cách liên kết lỏng lẻo (loosely coupled), giúp cải thiện tính linh hoạt và dễ bảo trì của trò chơi.

Ví dụ điển hình là sự kiện thắng game. Khi người chơi thắng một màn chơi, sự kiện này không chỉ đơn giản là hiển thị popup thắng, mà còn có thể kích hoạt các hành động khác như cộng thêm vàng và level, cập nhật điểm số, và cập nhật giao diện thanh menu để phản ánh những thay đổi này. Thay vì mỗi đối tượng như popup thắng, hệ thống điểm, hay thanh menu phải phụ thuộc vào sự kiện thắng game và gọi trực tiếp đến nhau, bạn có thể sử dụng Observer Pattern để các đối tượng này tự động lắng nghe sự kiện và hành động tương ứng mà không cần trực tiếp tham chiếu đến nhau.

Một ví dụ khác là khi người chơi mua trợ giúp. Khi người chơi mua một vật phẩm hỗ trợ trong trò chơi, sẽ có nhiều hành động khác xảy ra đồng thời: trừ vàng của người chơi, tăng số lượng trợ giúp, và cập nhật giao diện để hiển thị số lượng trợ giúp hiện tại. Với Observer Pattern, sự kiện mua trợ giúp sẽ kích hoạt một loạt các hành động liên quan mà không cần các đối tượng phụ thuộc trực tiếp vào sự kiện mua.

Observer Pattern hoạt động dựa trên nguyên tắc “1-nhiều”: một đối tượng (thường được gọi là subject) có thể có nhiều đối tượng khác (được gọi là observers) theo dõi nó. Khi trạng thái của subject thay đổi, tất cả các observers liên quan sẽ tự động được thông báo và thực hiện hành động phù hợp. Ví dụ, khi sự kiện thắng game xảy ra, tất cả các đối tượng phụ thuộc như popup thắng, hệ thống vàng, giao diện điểm số đều nhận được thông báo và cập nhật trạng thái của mình mà không cần phải gọi trực tiếp đến sự kiện thắng game.

Việc sử dụng Observer Pattern giúp giảm bớt sự phụ thuộc không cần thiết giữa các đối tượng trong trò chơi. Các đối tượng có thể hoạt động một cách độc

lập hơn và dễ dàng tương tác thông qua các sự kiện. Điều này không chỉ giúp hệ thống trở nên linh hoạt hơn, mà còn giúp dễ bảo trì, mở rộng trò chơi mà không cần thay đổi nhiều mã nguồn. Nếu có thêm các sự kiện hoặc hành động mới trong tương lai, chỉ cần thêm các observer mới mà không ảnh hưởng đến cấu trúc hiện tại.

Tóm lại, Observer Pattern là một giải pháp tối ưu để xử lý các sự kiện phức tạp và liên kết nhiều đối tượng trong game mà vẫn giữ được tính độc lập giữa chúng. Điều này giúp đảm bảo rằng trò chơi có thể mở rộng và phát triển một cách dễ dàng mà không gặp phải vấn đề về quản lý phụ thuộc giữa các đối tượng.

3.1.4 Thuật toán trong game

Thuật toán trong game (Algorithm Game) đóng vai trò quan trọng trong việc xác định cách các đối tượng trong trò chơi tương tác với nhau và với môi trường xung quanh. Đối với Blackout Ops, các thuật toán này không chỉ giúp xây dựng lối chơi mượt mà mà còn nâng cao trải nghiệm cho người chơi bằng cách tạo ra những tình huống bất ngờ và thách thức trong game.

Trong Blackout Ops, thuật toán quan trọng đầu tiên là thuật toán xác định va chạm. Đây là một yếu tố cốt lõi để xác định khi nào các đối tượng trong game va chạm với nhau, chẳng hạn như khi nhân vật của người chơi bị quái vật tấn công. Thuật toán này dựa trên các phép toán hình học, tính toán khoảng cách giữa các đối tượng để xác định khi nào chúng tiếp xúc với nhau. Cụ thể, khi nhân vật bị quái vật cận chiến tấn công hoặc bị quái vật đánh xa bắn trúng, nhân vật sẽ bị mất mạng. Điều này tạo ra một môi trường thử thách và yêu cầu người chơi phải thận trọng trong việc di chuyển và né tránh các đòn tấn công từ kẻ thù.

Bên cạnh thuật toán va chạm, thuật toán trí tuệ nhân tạo (AI) cũng đóng vai trò quan trọng trong việc điều khiển hành vi của các kẻ địch và các đối tượng không phải do người chơi điều khiển. AI giúp tạo ra những hành vi thông minh và đa dạng cho quái vật trong game. Chẳng hạn, khi nhân vật bị quái vật đánh xa bắn trúng, AI của quái vật sẽ ngừng tấn công lần nữa, đảm bảo rằng người chơi không bị tiêu diệt bởi một quái vật duy nhất nhiều lần. Điều này không chỉ giảm sự khó chịu mà còn giúp duy trì sự hứng thú cho người chơi, tránh cảm giác lặp lại và chán nản.

Mặt khác, thuật toán AI cũng được áp dụng để tăng độ khó của game trong một số tình huống. Ví dụ, khi người chơi cố tình né tránh quái vật bằng cách đi vòng qua chúng, quái vật sẽ tự động quay lại và tấn công từ phía sau. Hành động này tạo ra một thử thách mới, buộc người chơi phải cẩn trọng hơn và đưa ra các chiến thuật hợp lý để vượt qua những tình huống khó khăn. Điều này khiến trò chơi không chỉ là việc tránh né đơn giản mà còn yêu cầu sự nhanh nhạy và linh hoạt trong các tình huống hành động.

Nhờ việc sử dụng các thuật toán xác định va chạm và AI một cách hiệu quả, Blackout Ops không chỉ tạo ra những màn chơi đầy kịch tính mà còn mang lại trải nghiệm game phong phú và đa dạng. Người chơi được thách thức liên tục, nhưng đồng thời cũng không cảm thấy bị áp đảo quá mức, điều này khuyến khích họ tiếp tục chơi và cố gắng giành chiến thắng.

3.2 Code chức năng chính của game

3.2.1 Các phương thức cần thiết để xây dựng logic

```
BaseObject g_background;  
BaseObject g_Menu;  
BaseObject g_lose_menu;  
  
TTF_Font* font_time = NULL;  
  
Button PlayButton(HELP_BUTTON_POSX, HELP_BUTTON_POSY);  
Button HelpButton(PLAY_BUTTON_POSX, PLAY_BUTTON_POSY);  
Button ExitButton(EXIT_BUTTON_POSX, EXIT_BUTTON_POSY);  
Button BackButton(BACK_BUTTON_POSX, BACK_BUTTON_POSY);  
Button PauseButton(PAUSE_BUTTON_POSX, PAUSE_BUTTON_POSY);  
Button ContinueButton(PAUSE_BUTTON_POSX, PAUSE_BUTTON_POSY);  
Button Map1Button(MAP1_BUTTON_POSX, MAP1_BUTTON_POSY);  
Button Map2Button(MAP2_BUTTON_POSX, MAP2_BUTTON_POSY);  
Button Map3Button(MAP3_BUTTON_POSX, MAP3_BUTTON_POSY);  
Button ExitToMenuButton(QUIT_PAUSE_BUTTON_POSX, QUIT_PAUSE_BUTTON_POSY);  
Button Player1Button(PAYER1_BUTTON_POSX, PAYER1_BUTTON_POSY);  
Button Player2Button(PAYER2_BUTTON_POSX, PAYER2_BUTTON_POSY);  
Button Player3Button(PAYER3_BUTTON_POSX, PAYER3_BUTTON_POSY);  
  
SDL_Rect gPlayButton[BUTTON_TOTAL];  
SDL_Rect gHelpButton[BUTTON_TOTAL];  
SDL_Rect gExitButton[BUTTON_TOTAL];  
SDL_Rect gBackButton[BUTTON_TOTAL];  
SDL_Rect gPauseButton[BUTTON_TOTAL];  
SDL_Rect gContinueButton[BUTTON_MOUSE_OVER];  
SDL_Rect gPlayAgainButton[BUTTON_TOTAL];  
SDL_Rect gMap1Button[BUTTON_MOUSE_OVER];  
SDL_Rect gMap2Button[BUTTON_MOUSE_OVER];  
SDL_Rect gMap3Button[BUTTON_MOUSE_OVER];  
SDL_Rect gExitToMenuButton[BUTTON_TOTAL];  
SDL_Rect gPlayer1Button[BUTTON_TOTAL];  
SDL_Rect gPlayer2Button[BUTTON_TOTAL];  
SDL_Rect gPlayer3Button[BUTTON_TOTAL];  
  
⊕ bool InitData() { ... }  
  
⊕ bool LoadBackground(std::string map_background) { ... }  
  
⊕ void LoadButton() { ... }  
  
⊕ void close() { ... }  
  
⊕ std::vector<ThreatsObject*> MakeThreadList() { ... }
```

Hình 3.4: Các phương thức cần thiết tạo ra game

Hình ảnh trên mô tả một phần của mã nguồn trong dự án phát triển game, đặc biệt là phần khai báo các đối tượng và phương thức liên quan đến giao diện điều khiển của trò chơi. Các đối tượng Button được định nghĩa để quản lý các nút như Play, Help, Exit, Back, Map, Pause, và Continue. Các nút này được thiết lập

vị trí cụ thể bằng các hằng số POSX và POSY tương ứng với tọa độ trên giao diện. Bên cạnh đó, các biến kiểu SDL_Rect được sử dụng để lưu trữ vị trí và kích thước của các nút tương ứng cho các sự kiện, chẳng hạn như di chuyển chuột qua nút (BUTTON_MOUSE_OVER).

Bên cạnh việc định nghĩa các đối tượng điều khiển, các phương thức như InitData(), LoadBackground(), LoadButton(), và close() cũng được khai báo để phục vụ cho quá trình khởi tạo và quản lý tài nguyên của trò chơi. InitData() chịu trách nhiệm khởi tạo các dữ liệu cần thiết cho game, từ việc thiết lập các biến đến nạp tài nguyên cơ bản. LoadBackground() và LoadButton() được sử dụng để tải các hình nền và các nút điều khiển, đảm bảo các thành phần đồ họa được hiển thị chính xác. Phương thức close() đảm bảo tài nguyên của game như hình ảnh và âm thanh sẽ được giải phóng khi không còn cần thiết, giúp tiết kiệm bộ nhớ và tránh rò rỉ tài nguyên.

Ngoài ra, một phương thức quan trọng khác là MakeThreadList(), chịu trách nhiệm tạo ra danh sách các đối tượng phục vụ cho xử lý đa luồng. Trong trò chơi, việc xử lý đa luồng giúp tối ưu hóa hiệu suất, khi các hành động khác nhau như đồ họa, âm thanh và logic trò chơi có thể được xử lý đồng thời. Điều này mang lại trải nghiệm mượt mà hơn cho người chơi, đặc biệt khi game đòi hỏi xử lý nhiều tác vụ phức tạp cùng lúc.

Các thành phần như nút bấm, hình nền và hệ thống đa luồng đóng vai trò quan trọng trong việc tạo nên một giao diện trò chơi thân thiện và trực quan. Sự kết hợp giữa đồ họa và các yếu tố điều khiển giúp trò chơi trở nên dễ dàng tiếp cận hơn với người chơi, đồng thời đảm bảo hiệu suất hoạt động của game luôn ổn định nhờ các phương thức quản lý tài nguyên hiệu quả.

3.2.2 Phần code tạo bản đồ và tải bản đồ

```
void GameMap::LoadMap(std::string name)
{
    std::ifstream in; // ngan hon dung fopen_s
    in.open(name);
    game_map_.max_x_ = MAX_MAP_X * TILE_SIZE;
    game_map_.max_y_ = MAX_MAP_Y * TILE_SIZE;

    for (int i = 0; i < MAX_MAP_Y; i++) { ... }

    game_map_.start_x_ = 0;
    game_map_.start_y_ = 0;

    game_map_.file_name_ = name;

    in.close();
}

void GameMap::LoadTiles(SDL_Renderer* screen, std::string path)
{
    for (int i = 0; i < MAX_TILES; i++)
    {
        std::string file_img = path;
        std::string num = std::to_string(i);
        file_img.insert(11, num);

        tile_map[i].LoadImg(file_img, screen);
    }
}

void GameMap::DrawMap(SDL_Renderer* screen)
{
    int x1 = 0;
    int x2 = 0;
    int y1 = 0;
    int y2 = 0;
    int map_x = 0;
    int map_y = 0;
    map_x = game_map_.start_x_ / TILE_SIZE;
    x1 = (game_map_.start_x_ % TILE_SIZE) * -1;
    x2 = x1 + SCREEN_WIDTH + (x1 == 0 ? 0 : TILE_SIZE);
    map_y = game_map_.start_y_ / TILE_SIZE;
    y1 = (game_map_.start_y_ % TILE_SIZE) * -1;
    y2 = y1 + SCREEN_HEIGHT + (y1 == 0 ? 0 : TILE_SIZE);
    for (int i = y1; i < y2; i += TILE_SIZE)
    {
        map_x = game_map_.start_x_ / TILE_SIZE;
        for (int j = x1; j < x2; j += TILE_SIZE)
        {
            int val = game_map_.tile[map_y][map_x];
            if (val > 0)
            {
                tile_map[val].SetRect(j, i);
                tile_map[val].Render(screen);
            }
            map_x++;
        }
        map_y++;
    }
}
```

Hình 3.5: Chương trình tạo bản đồ

Hình ảnh này mô tả các phương thức được sử dụng để tải và vẽ bản đồ (map) trong trò chơi thông qua thư viện SDL. Cụ thể, hình ảnh này bao gồm ba phương thức chính:

1. `LoadMap(std::string name)`: Phương thức này chịu trách nhiệm tải file bản đồ từ một đường dẫn hoặc tên file được chỉ định. Nó mở file, thiết lập kích thước bản đồ dựa trên các hằng số như `MAX_MAP_X` và `MAX_MAP_Y`, và gán các giá trị tọa độ khởi đầu cho bản đồ. Sau khi hoàn tất quá trình đọc dữ liệu, file sẽ được đóng lại.
2. `LoadTiles(SDL_Renderer screen, std::string path)`: Phương thức này dùng để tải các tile (ô vuông nhỏ tạo nên bản đồ) từ file hình ảnh. Nó duyệt qua từng tile và sử dụng hàm `loadImg()` để tải hình ảnh tương ứng và lưu trữ trong `tile_map`. Mỗi tile được gán một số chỉ mục để có thể sử dụng khi vẽ bản đồ.
3. `DrawMap(SDL_Renderer screen)`: Phương thức này vẽ bản đồ lên màn hình. Nó tính toán các tọa độ bắt đầu và kết thúc của bản đồ cần được hiển thị, dựa trên tọa độ khởi đầu và kích thước của màn hình. Sau đó, nó duyệt qua từng tile trong bản đồ và chỉ vẽ các tile có giá trị lớn hơn 0. Mỗi tile được thiết lập vị trí và sau đó được render lên màn hình thông qua hàm `Render(screen)`.

3.2.3 Phương thức xây dựng đạn

```
bool BulletObject::LoadImgBullet(SDL_Renderer* des)
{
    bool ret = false;
    if (bullet_type_ == LASER_BULLET)
    {
        ret = LoadImg("img//laser_bullet.png", des);
    }
    else
    {
        ret = LoadImg("img//sphere_bullet.png", des);
    }
    return ret;
}

void BulletObject::HandleMove(const int& x_border, const int& y_border)
{
    if (bullet_dir_ == RIGHT)
    {
        rect_.x += x_val_;
        if (rect_.x > x_border)
        {
            is_move_ = false;
        }
    }
    else if (bullet_dir_ == LEFT)
    {
        rect_.x -= x_val_;
        if (rect_.x < 0)
        {
            is_move_ = false;
        }
    }
}
```

Hình 3.6: Phương thức xây dựng đạn

Hình ảnh trên minh họa hai phương thức trong lớp BulletObject liên quan đến việc xử lý các viên đạn trong game.

1. LoadImgBullet(SDL_Renderer des):

- Phương thức này chịu trách nhiệm tải hình ảnh của các loại đạn khác nhau.
- Nó sử dụng biến `bullet_type_` để xác định loại đạn cần được tải, chẳng hạn như `LASER_BULLET` hay `SPHERE_BULLET`.

- Nếu là LASER_BULLET, phương thức sẽ gọi hàm LoadImg để tải hình ảnh "laser_bullet.png". Ngược lại, nếu không phải loại đạn laser, nó sẽ tải hình ảnh "sphere_bullet.png".
- Kết quả của quá trình tải hình ảnh sẽ được lưu vào biến ret, và phương thức sẽ trả về giá trị này để xác nhận việc tải ảnh có thành công hay không.

2. HandleMove(const int& x_border, const int& y_border):

- Phương thức này xử lý chuyển động của đạn, dựa trên hướng của đạn (được lưu trong biến bullet_dir_).
- Nếu hướng đạn là RIGHT, đạn sẽ di chuyển về phía bên phải bằng cách tăng giá trị x của nó. Nếu đạn vượt quá biên giới x_border, việc di chuyển sẽ dừng lại bằng cách gán is_move_ = false.
- Tương tự, nếu hướng đạn là LEFT, đạn sẽ di chuyển về phía bên trái và giá trị x sẽ giảm. Nếu đạn vượt qua biên giới trái (vị trí $x < 0$), đạn cũng sẽ dừng di chuyển.

3.2.4 Phương thức xây dựng di chuyển và tấn công của Boss

```
void BossObject::CheckToMap(Map& g_map)
{
    int x1 = 0;
    int x2 = 0;
    int y1 = 0;
    int y2 = 0;
    on_ground_ = 0;
    int height_min = height_frame_;
    x1 = (x_pos_ + x_val_) / TILE_SIZE;
    x2 = (x_pos_ + x_val_ + width_frame_ - 1) / TILE_SIZE;

    y1 = (y_pos_) / TILE_SIZE;
    y2 = (y_pos_ + height_min - 1) / TILE_SIZE;

    if (x1 >= 0 && x2 < MAX_MAP_X && y1 >= 0 && y2 < MAX_MAP_Y)
    {
        if (x_val_ > 0)
        {
            if ((g_map.tile[y1][x2] != BLANK_TILE) || (g_map.tile[y2][x2] != BLANK_TILE))
            {
                x_pos_ = x2 * TILE_SIZE;
                x_pos_ -= width_frame_ + 1;
                x_val_ = 0;
            }
        }
        else if (x_val_ < 0)
        {
            if ((g_map.tile[y1][x1] != BLANK_TILE) || (g_map.tile[y2][x1] != BLANK_TILE))
            {
                x_pos_ = (x1 + 1) * TILE_SIZE;
                x_val_ = 0;
            }
        }
    }

    int width_min = width_frame_;
    x1 = (x_pos_) / TILE_SIZE;
    x2 = (x_pos_ + width_min) / TILE_SIZE;

    y1 = (y_pos_ + y_val_) / TILE_SIZE;
    y2 = (y_pos_ + y_val_ + height_frame_) / TILE_SIZE;

    if (x1 >= 0 && x2 < MAX_MAP_X && y1 >= 0 && y2 < MAX_MAP_Y)
    {
        if (y_val_ > 0)
        {
            if ((g_map.tile[y2][x1] != BLANK_TILE) || (g_map.tile[y2][x2] != BLANK_TILE))
            {
                y_pos_ = y2 * TILE_SIZE;
                y_pos_ -= height_frame_;
                y_val_ = 0;
                on_ground_ = 1;
            }
        }
        else if (y_val_ < 0)
        {
            if ((g_map.tile[y1][x1] != BLANK_TILE) || (g_map.tile[y1][x2] != BLANK_TILE))
            {
                y_pos_ = (y1 + 1) * TILE_SIZE;
                y_val_ = 0;
            }
        }
    }

    x_pos_ += x_val_;
    y_pos_ += y_val_;
}
```

Hình 3.7: Phương thức xây dựng di chuyển và tấn công của quái vật

Hình ảnh này hiển thị mã nguồn của phương thức CheckToMap() trong lớp BossObject. Đây là phương thức kiểm tra va chạm giữa đối tượng boss và các đối tượng trên bản đồ trò chơi, nhằm xử lý việc di chuyển của boss trong môi trường game. Các phép toán trong phương thức chủ yếu dựa trên việc kiểm tra các tile (ô vuông) trong bản đồ để xem liệu boss có va chạm với bất kỳ đối tượng nào, ví dụ như tường hoặc chướng ngại vật.

Mô tả chi tiết:

- `x1, y1, x2, y2`: Đây là các biến lưu trữ vị trí của các ô tile mà boss hiện đang nằm trên. Tọa độ của các ô này được tính toán bằng cách chia tọa độ của boss theo `TILE_SIZE` để xác định vị trí tile trong hệ tọa độ của bản đồ.
- `on_ground`: Biến này được sử dụng để xác định liệu boss có đang đứng trên mặt đất hay không.
- `height_min, width_min`: Các biến này lưu trữ chiều cao và chiều rộng của boss, được sử dụng để tính toán phạm vi của boss trong bản đồ.

Kiểm tra va chạm theo chiều ngang:

- Phương thức bắt đầu bằng việc kiểm tra va chạm theo chiều ngang (x). Tọa độ của boss được chia theo `TILE_SIZE` để xác định các tile tương ứng. Sau đó, phương thức kiểm tra xem tile đó có phải là một `BLANK_TILE` (tile trống) hay không. Nếu không phải, điều này có nghĩa là boss đã va chạm với một đối tượng khác (ví dụ: tường), do đó, giá trị `x_pos` và `x_val` sẽ được điều chỉnh để boss không đi xuyên qua vật thể.

Kiểm tra va chạm theo chiều dọc:

- Sau khi kiểm tra theo chiều ngang, phương thức tiếp tục kiểm tra va chạm theo chiều dọc (y). Cơ chế kiểm tra tương tự như chiều ngang, kiểm tra xem boss có va chạm với một tile không trống. Nếu boss đang di chuyển xuống và va chạm với mặt đất, biến `on_ground` sẽ được gán giá trị 1 để biểu thị rằng boss đã chạm đất.

3.2.5 Phương thức xây dựng sức mạnh nhân vật

```
1
2   #include "PlayerPower.h"
3
4   PlayerPower::PlayerPower()
5   {
6       number_ = 0;
7   }
8
9   PlayerPower::~~PlayerPower()
10  {
11  }
12
13
14  void PlayerPower::AddPos(const int& xp)
15  {
16      pos_list_.push_back(xp);
17  }
18
19  void PlayerPower::Init(SDL_Renderer* screen)
20  {
21      LoadImg("img//player_pw.png", screen);
22      number_ = 3;
23      if (pos_list_.size() > 0)
24      {
25          pos_list_.clear();
26      }
27      AddPos(20);
28      AddPos(60);
29      AddPos(100);
30  }
31
32  void PlayerPower::Show(SDL_Renderer* screen)
33  {
34      for (int i = 0; i < pos_list_.size(); i++)
35      {
36          rect_.x = pos_list_.at(i);
37          rect_.y = 0;
38          Render(screen);
39      }
40  }
```

Hình 3.8: Phương thức xây dựng sức mạnh của nhân vật

Hình ảnh này hiển thị mã nguồn của lớp PlayerPower trong trò chơi, liên quan đến việc quản lý năng lượng hoặc "power" của người chơi.

Mô tả chi tiết:

1. Constructor và Destructor:

- `PlayerPower::PlayerPower()`: Hàm khởi tạo đặt giá trị ban đầu cho `number_` là 0. Điều này có thể hiểu là năng lượng ban đầu của người chơi được thiết lập thành 0.
- `PlayerPower::~~PlayerPower()`: Hàm hủy (destructor) không làm gì đặc biệt nhưng có thể dùng để giải phóng tài nguyên hoặc xử lý các bước dọn dẹp nếu cần thiết.

2. `AddPos(const int& xp)`:

- Phương thức này thêm các giá trị vị trí mới vào danh sách `pos_list_`. `xp` là một giá trị vị trí được truyền vào và được thêm vào danh sách thông qua `push_back()`. Điều này có thể giúp lưu trữ các vị trí cho việc hiển thị năng lượng của người chơi trên màn hình.

3. `Init(SDL_Renderer screen)`:

- Phương thức `Init()` chịu trách nhiệm khởi tạo hình ảnh cho năng lượng của người chơi. Nó sử dụng hàm `LoadImg` để tải hình ảnh từ tệp `"player_pw.png"`.
- Số năng lượng ban đầu của người chơi được thiết lập là 3 (trong biến `number_`).
- Nếu danh sách vị trí `pos_list_` đã có phần tử, nó sẽ xóa danh sách này và khởi tạo lại các vị trí cho năng lượng với các giá trị cố định (20, 60, 100).

4. `Show(SDL_Renderer screen)`:

- Phương thức `Show()` có nhiệm vụ hiển thị năng lượng của người chơi lên màn hình.
- Nó lặp qua danh sách các vị trí đã lưu trong `pos_list_` và thiết lập các tọa độ x cho từng vị trí trong danh sách. Tọa độ y của tất cả các năng lượng được đặt bằng 0.
- Cuối cùng, phương thức `Render(screen)` sẽ được gọi để hiển thị năng lượng tại các vị trí tương ứng trên màn hình.

3.2.6 Phương thức xử lý va chạm

```
bool SDLCommonFunc::CheckCollision(const SDL_Rect& object1, const SDL_Rect& object2)
{
    int left_a = object1.x;
    int right_a = object1.x + object1.w;
    int top_a = object1.y;
    int bottom_a = object1.y + object1.h;

    int left_b = object2.x;
    int right_b = object2.x + object2.w;
    int top_b = object2.y;
    int bottom_b = object2.y + object2.h;

    if (left_a > left_b && left_a < right_b) { ... }

    if (left_a > left_b && left_a < right_b)
    {
        if (bottom_a > top_b && bottom_a < bottom_b)
        {
            return true;
        }
    }

    if (right_a > left_b && right_a < right_b)
    {
        if (top_a > top_b && top_a < bottom_b)
        {
            return true;
        }
    }

    if (right_a > left_b && right_a < right_b) { ... }

    if (left_b > left_a && left_b < right_a) { ... }

    if (left_b > left_a && left_b < right_a) { ... }

    if (right_b > left_a && right_b < right_a) { ... }

    if (right_b > left_a && right_b < right_a) { ... }

    if (top_a == top_b && right_a == right_b && bottom_a == bottom_b) { ... }

    return false;
}
```

Hình 3.9: Phương pháp xử lý va chạm

Hình ảnh này hiển thị mã nguồn của hàm CheckCollision() trong một lớp chứa các hàm tiện ích dùng cho SDL (Simple DirectMedia Layer). Mục đích của hàm này là kiểm tra xem hai đối tượng có va chạm với nhau hay không dựa trên tọa độ của chúng.

Mô tả chi tiết:

Hàm `CheckCollision()` nhận vào hai tham số là hai đối tượng `SDL_Rect` (hình chữ nhật) - `object1` và `object2`. Mỗi đối tượng hình chữ nhật được định nghĩa bởi các tọa độ của các cạnh (trái, phải, trên, dưới) dựa trên vị trí và kích thước của nó.

1. Tính toán biên của các hình chữ nhật:

- `left_a`, `right_a`, `top_a`, `bottom_a`: Biên trái, phải, trên và dưới của đối tượng `object1`.
- `left_b`, `right_b`, `top_b`, `bottom_b`: Biên trái, phải, trên và dưới của đối tượng `object2`.

2. Kiểm tra va chạm:

- Để kiểm tra va chạm giữa hai hình chữ nhật, hàm sẽ kiểm tra xem liệu các biên của chúng có nằm trong vùng của nhau hay không.
- Nếu biên trái và biên phải của `object1` nằm trong khoảng giữa biên trái và biên phải của `object2` và ngược lại, thì sẽ tiến hành kiểm tra xem biên trên và biên dưới có chồng lên nhau hay không.
- Nếu bất kỳ điều kiện nào thỏa mãn, hàm sẽ trả về `true` để biểu thị rằng hai đối tượng đang va chạm.
- Nếu không có va chạm nào, hàm sẽ trả về `false`.

3.2.7 Phương thức xử lý vụ nổ trong game

```
void ExplosionObject::set_clip()
{
    if (frame_width_ > 0 && frame_height_ > 0)
    {
        frame_clip_[0].x = 0;
        frame_clip_[0].y = 0;
        frame_clip_[0].w = frame_width_;
        frame_clip_[0].h = frame_height_;

        frame_clip_[1].x = frame_width_;
        frame_clip_[1].y = 0;
        frame_clip_[1].w = frame_width_;
        frame_clip_[1].h = frame_height_;

        frame_clip_[2].x = frame_width_ * 2;
        frame_clip_[2].y = 0;
        frame_clip_[2].w = frame_width_;
        frame_clip_[2].h = frame_height_;

        frame_clip_[3].x = frame_width_ * 3;
        frame_clip_[3].y = 0;
        frame_clip_[3].w = frame_width_;
        frame_clip_[3].h = frame_height_;

        frame_clip_[4].x = frame_width_ * 4;
        frame_clip_[4].y = 0;
        frame_clip_[4].w = frame_width_;
        frame_clip_[4].h = frame_height_;

        frame_clip_[5].x = frame_width_ * 5;
        frame_clip_[5].y = 0;
        frame_clip_[5].w = frame_width_;
        frame_clip_[5].h = frame_height_;

        frame_clip_[6].x = frame_width_ * 6;
        frame_clip_[6].y = 0;
        frame_clip_[6].w = frame_width_;
        frame_clip_[6].h = frame_height_;

        frame_clip_[7].x = frame_width_ * 7;
        frame_clip_[7].y = 0;
        frame_clip_[7].w = frame_width_;
        frame_clip_[7].h = frame_height_;
    }
}
```

Hình 3.10: Phương thức xử lý vụ nổ

Hình ảnh này hiển thị phương thức `set_clip()` của đối tượng `ExplosionObject` trong trò chơi. Phương thức này chịu trách nhiệm thiết lập các đoạn clip (hay còn gọi là các khung hình) cho hiệu ứng nổ trong trò chơi. Mỗi clip đại diện cho một khung hình khác nhau của hiệu ứng nổ, được cắt từ một ảnh lớn chứa toàn bộ các khung hình của hiệu ứng.

Mô tả chi tiết:

- Đầu tiên, phương thức kiểm tra xem `frame_width_` và `frame_height_` có lớn hơn 0 không. Nếu cả hai giá trị đều lớn hơn 0, quá trình cắt khung hình sẽ được thực hiện.
- Các khung hình của hiệu ứng nổ được lưu trữ trong mảng `frame_clip[]`. Mỗi phần tử trong mảng chứa thông tin về vị trí (tọa độ x, y) và kích thước (chiều rộng w, chiều cao h) của từng khung hình.
- Phương thức này thiết lập 8 khung hình của hiệu ứng nổ, với mỗi khung hình được tính toán dựa trên giá trị `frame_width_` và `frame_height_`.
 - `frame_clip[0]`: Tọa độ x = 0 và y = 0, chiều rộng và chiều cao của khung hình là `frame_width_` và `frame_height_`.
 - `frame_clip[1]` đến `frame_clip[7]`: Tọa độ x của mỗi khung hình được tính bằng bội số của `frame_width_` để cắt các phần tiếp theo trong dải hình ảnh, còn tọa độ y vẫn là 0.

3.3 Kết quả giao diện game

3.3.1 Màn hình chính



Hình 3.11: Giao diện chính của game

Sau khi triển khai và cài đặt game thành công, khi khởi động game sẽ hiển thị giao diện menu chính. Tại đây có các tùy chọn Play, Help và Exit. Nhấn Play để bắt đầu trò chơi, Help để xem hướng dẫn cách chơi, và Exit để thoát khỏi game. Giao diện được thiết kế theo phong cách đồ họa pixel cổ điển, mang lại cảm giác sinh động và gần gũi với người chơi.

3.3.2 Màn hình hướng dẫn chơi game

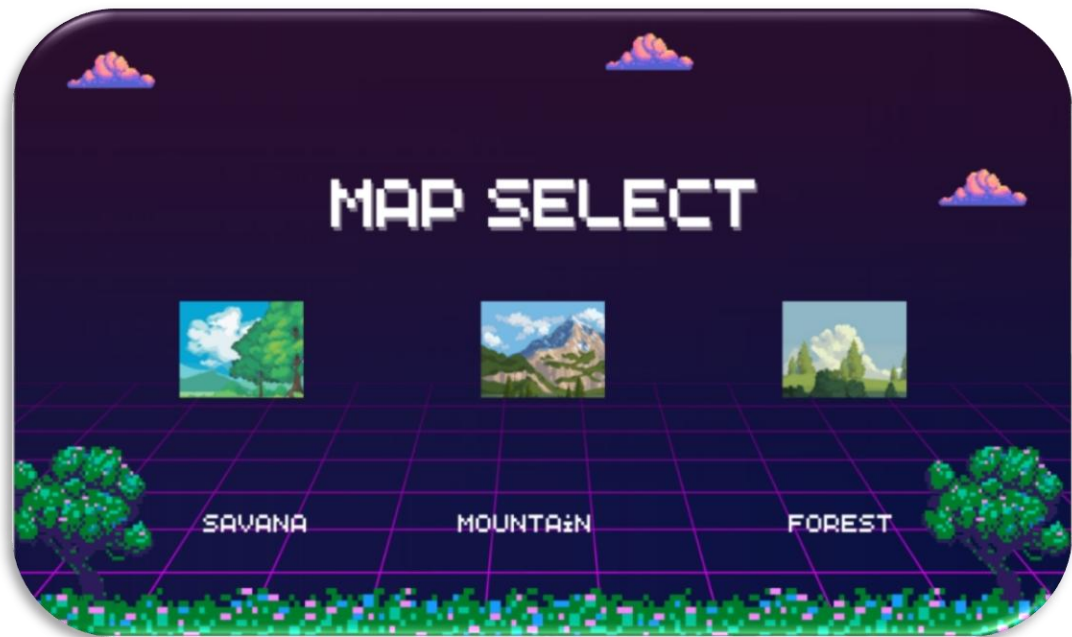


Hình 3.12: Màn hình hướng dẫn chơi game

Khi người chơi chọn mục Help từ giao diện menu chính, màn hình hướng dẫn chơi game sẽ được hiển thị. Tại đây, người chơi sẽ được giới thiệu các thao tác điều khiển cơ bản trong game, bao gồm:

- A: Di chuyển sang trái
- D: Di chuyển sang phải
- Chuột phải: Bắn đạn
- Chuột trái: Nhảy

3.3.3 Màn hình chọn bản đồ

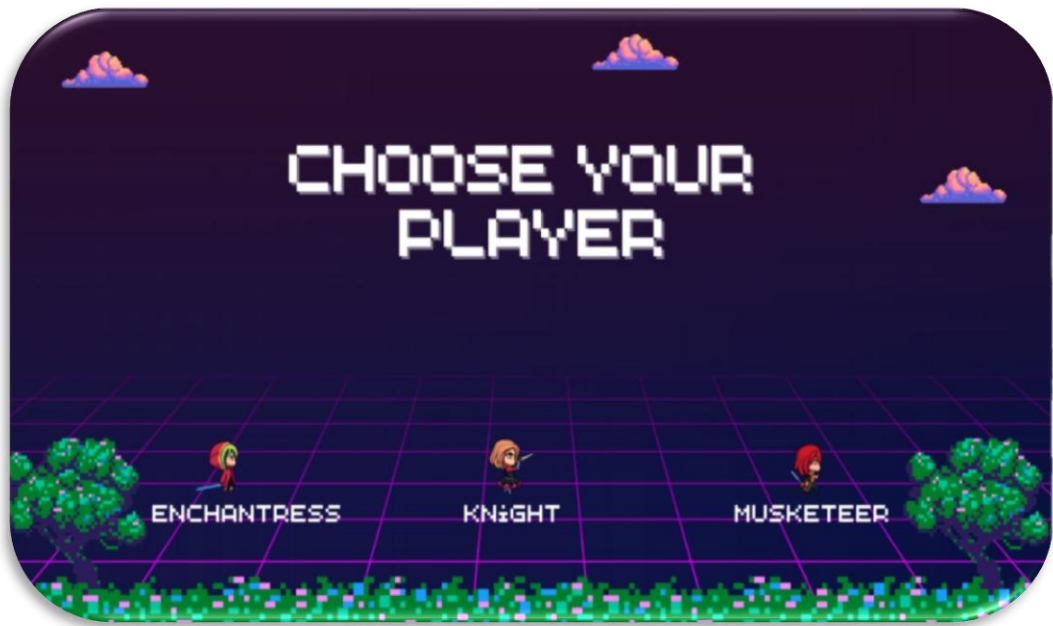


Hình 3.13: Màn hình chọn bản đồ

Sau khi người chơi hoàn thành bước hướng dẫn hoặc truy cập mục chơi chính, hệ thống sẽ hiển thị màn hình Map Select để người chơi lựa chọn bản đồ trước khi bắt đầu game. Giao diện bao gồm ba lựa chọn bản đồ:

- Savana: Đồng cỏ nhiệt đới rộng lớn.
- Mountain: Địa hình đồi núi hiểm trở.
- Forest: Khu rừng rậm rạp, đầy thử thách.

3.3.4 Màn hình chọn nhân vật



Hình 3.14: Màn hình chọn nhân vật

Sau khi người chơi lựa chọn bản đồ, hệ thống sẽ chuyển đến màn hình Choose Your Player, nơi người chơi có thể chọn nhân vật để tham gia trò chơi. Giao diện hiển thị ba nhân vật khác nhau với tên gọi và hình dáng đặc trưng:

- Enchantress: Phù thủy với khả năng phép thuật.
- Knight: Hiệp sĩ với khả năng phòng thủ và cận chiến.
- Musketeer: Xạ thủ với kỹ năng tấn công tầm xa

3.3.5 Màn hình chờ vào game



Hình 3.15: Màn hình chờ

Trước khi bắt đầu trò chơi, hệ thống sẽ hiển thị màn hình Game Start kèm theo thanh tải (loading bar) thể hiện tiến trình khởi tạo tài nguyên game. Màn hình chờ giúp người chơi nhận biết rằng game đang được chuẩn bị để khởi động, đồng thời giữ cho trải nghiệm chơi liền mạch và trực quan.

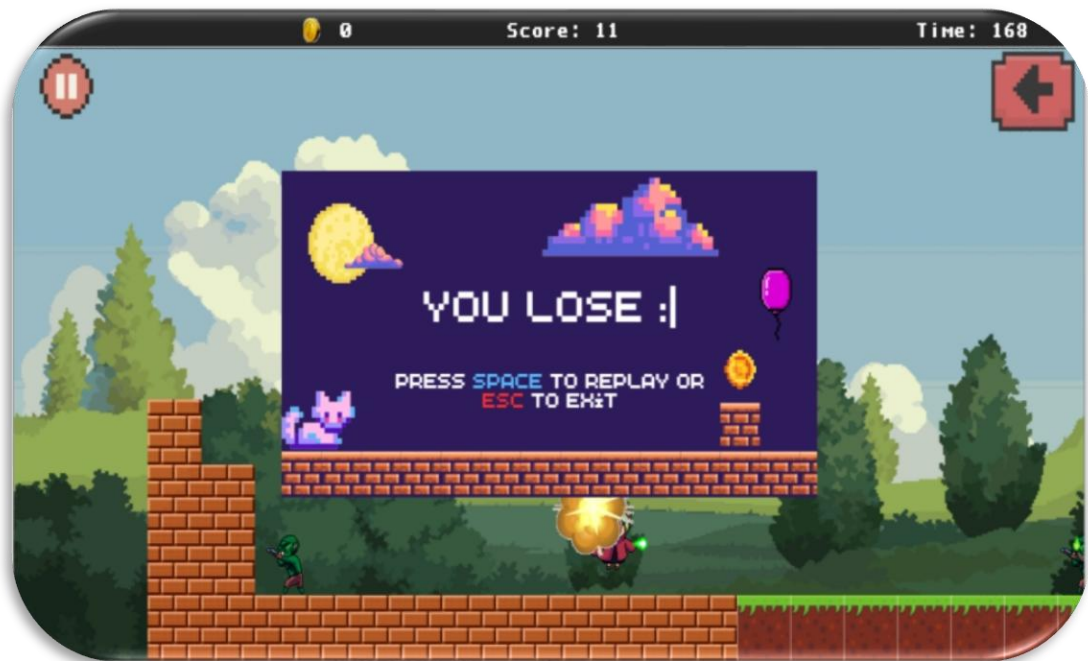
3.3.6 Màn hình chơi game



Hình 3.16: Giao diện trong game

Khi người chơi hoàn tất các bước chuẩn bị như chọn bản đồ và nhân vật, game sẽ chuyển sang màn hình chơi chính. Tại đây, người chơi sẽ điều khiển nhân vật di chuyển, tấn công kẻ địch và vượt qua các chướng ngại vật để hoàn thành màn chơi.

3.3.7 Màn hình thua game

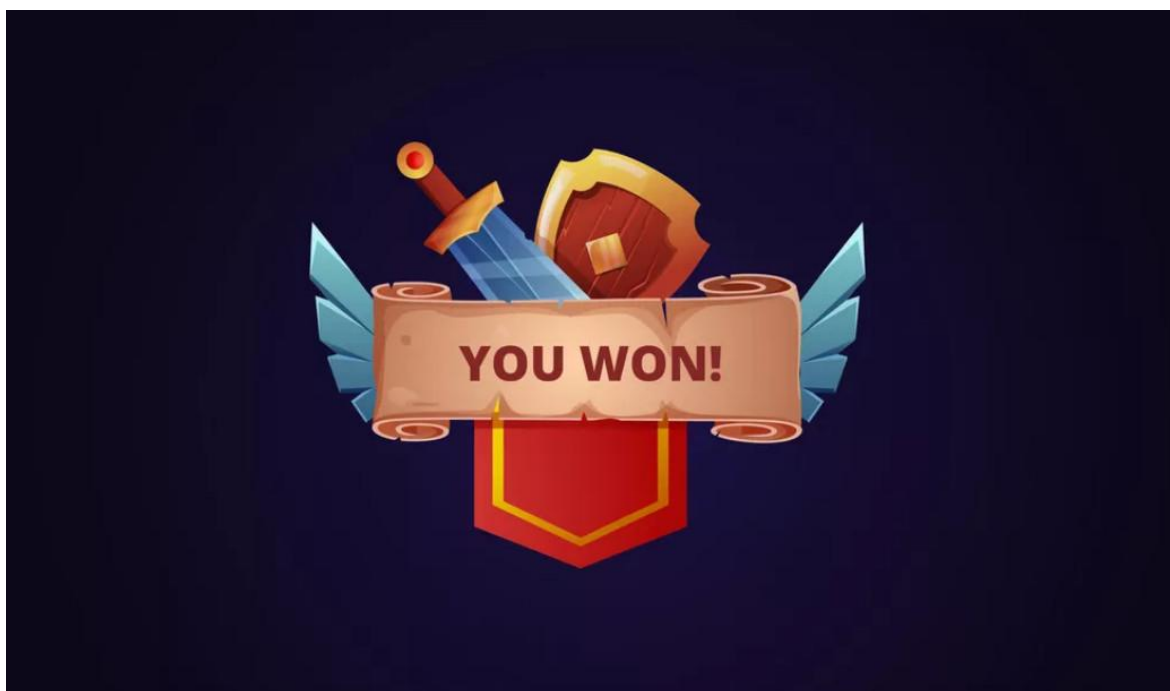


Hình 3.17: Giao diện khi thua game

Khi người chơi để nhân vật mất toàn bộ máu hoặc không hoàn thành mục tiêu trong thời gian quy định, hệ thống sẽ hiển thị màn hình thua cuộc với thông báo "YOU LOSE!" ở vị trí trung tâm. Giao diện hiển thị rõ ràng hai lựa chọn chính:

- Nhấn phím Space để chơi lại từ đầu.
- Nhấn phím ESC để thoát khỏi màn chơi.

3.3.8 Màn hình thắng game



Hình 3.18: Giao diện chiến thắng

Khi người chơi hoàn thành mục tiêu và vượt qua tất cả thử thách trong màn chơi, hệ thống sẽ hiển thị giao diện chiến thắng với thông điệp nổi bật "YOU WON!" ở trung tâm màn hình.

3.4 Đánh giá

3.4.1 Ưu điểm

Việc lựa chọn SDL để phát triển game Blackout Ops là một quyết định táo bạo khi so sánh SDL với những framework làm game mạnh mẽ khác trên thị trường hiện nay. SDL mang đến nhiều ưu điểm đáng chú ý, đặc biệt là khả năng hỗ trợ đa nền tảng. Điều này cho phép Blackout Ops không chỉ giới hạn ở nền tảng Windows, mà có thể dễ dàng mở rộng sang macOS, Linux, và thậm chí cả các hệ điều hành di động như Android và iOS, tạo ra một hệ sinh thái phong phú cho người chơi trên nhiều thiết bị khác nhau. Hơn nữa, SDL giúp trò chơi đạt được hiệu suất tốt nhờ khả năng xử lý đồ họa 2D tối ưu, điều này rất quan trọng đối với thể loại game platformer như Blackout Ops, nơi mà tốc độ khung hình ổn định và mượt mà là yếu tố quyết định trải nghiệm của người chơi.

Không những thế, SDL còn mang đến sự linh hoạt trong việc tích hợp với các thư viện đồ họa mạnh mẽ như OpenGL hay DirectX, mở ra khả năng cải thiện chất lượng hình ảnh trong game. Điều này không chỉ giúp Blackout Ops dễ dàng nâng cấp từ đồ họa 2D cơ bản lên các hiệu ứng 3D cơ bản, mà còn tạo ra tiềm năng vô hạn cho việc phát triển các tính năng đồ họa cao cấp trong tương lai, giúp trò chơi luôn bắt kịp xu hướng của thị trường.

3.4.2 Nhược điểm

SDL vẫn tồn tại một số giới hạn không thể phủ nhận. Mặc dù hỗ trợ tốt cho đồ họa 2D, nhưng SDL lại thiếu sự hỗ trợ cho đồ họa 3D và các hiệu ứng hình ảnh phức tạp, điều này có thể trở thành rào cản lớn nếu muốn mở rộng Blackout Ops theo hướng tăng cường trải nghiệm thị giác. Đặc biệt, với những game hiện đại yêu cầu môi trường 3D phong phú hoặc những hiệu ứng động phức tạp, SDL có thể sẽ trở nên hạn chế. Thêm vào đó, khả năng hỗ trợ đa luồng của SDL còn yếu, khiến việc xử lý nhiều tác vụ cùng lúc trở nên khó khăn. Điều này có thể ảnh hưởng đến hiệu suất game khi Blackout Ops phát sinh nhiều sự kiện đồng thời, ví dụ như trong các cảnh chiến đấu với quái vật liên tục hay xử lý tương tác vật lý trong thời gian thực.

Một điểm nữa cần chú ý là khả năng xử lý âm thanh của SDL còn sơ khai. Với những tựa game có yếu tố âm thanh phức tạp, như việc cần xử lý nhiều luồng âm thanh động, âm nhạc tương tác, hoặc hiệu ứng âm thanh 3D, SDL có thể sẽ không đáp ứng được kỳ vọng của người chơi. Điều này dễ dẫn đến việc âm thanh trong game trở nên đơn điệu và không tạo được ấn tượng mạnh, làm giảm đi cảm xúc của người chơi khi tham gia vào các cuộc phiêu lưu.

3.4.3 Hướng khắc phục

Để vượt qua các nhược điểm này, em cần lên kế hoạch dài hạn để đưa Blackout Ops tiến xa hơn. Việc tích hợp SDL với các thư viện đồ họa tiên tiến hơn như OpenGL, Vulkan hoặc thậm chí DirectX là một giải pháp tất yếu để đưa đồ họa của game lên tầm cao mới. Điều này không chỉ giúp cải thiện hiệu suất xử lý đồ họa mà còn mở ra khả năng phát triển các hiệu ứng thị giác hiện đại, từ đổ bóng đến ánh sáng động, tăng cường tính chân thực và tạo chiều sâu cho môi trường game. Về vấn đề đa luồng, em có thể nghiên cứu sử dụng các công cụ hỗ

trợ đa luồng như SDL Threads hoặc kết hợp với các thư viện bên ngoài như Boost Asio để quản lý và điều phối các sự kiện trong game một cách hiệu quả hơn, đảm bảo game không bị chậm hoặc giật khi xảy ra nhiều hành động cùng lúc.

Trong lĩnh vực âm thanh, một giải pháp hợp lý là tích hợp SDL với các thư viện âm thanh tiên tiến như FMOD hoặc OpenAL, không chỉ giúp nâng cấp trải nghiệm âm thanh, mà còn cho phép Blackout Ops mang đến những bản nhạc nền động, hiệu ứng âm thanh sống động hơn, từ tiếng bước chân, tiếng gió đến các âm thanh chiến đấu. Điều này sẽ giúp game không chỉ hấp dẫn về mặt hình ảnh mà còn có chiều sâu về âm thanh, tăng tính tương tác và cảm xúc của người chơi.

Tóm lại, mặc dù SDL mang đến nhiều ưu thế cho việc phát triển game Blackout Ops, đặc biệt là sự linh hoạt và hỗ trợ đa nền tảng, nhưng những nhược điểm về khả năng xử lý đồ họa 3D, đa luồng và âm thanh cần được giải quyết sớm. Bằng cách kết hợp SDL với các công nghệ hiện đại, em có thể không chỉ khắc phục được những hạn chế này mà còn đưa Blackout Ops trở thành một tựa game có sức hút lớn và khả năng mở rộng mạnh mẽ trong tương lai.

KẾT LUẬN

Trong bối cảnh thị trường game ngày càng phát triển, các thể loại game phiêu lưu hành động ngày càng thu hút sự quan tâm của đông đảo người chơi. Việc phát triển game Blackout Ops nhằm mang đến trải nghiệm mới mẻ và thú vị cho người chơi thông qua không gian 2D kết hợp giữa hành động và phiêu lưu. Em đã chọn SDL làm nền tảng phát triển cho game vì SDL là một thư viện mã nguồn mở mạnh mẽ, cho phép phát triển game đa nền tảng một cách dễ dàng. Với khả năng hỗ trợ Windows, macOS, Linux, và các nền tảng di động, việc chọn SDL giúp em tiếp cận được nhiều đối tượng người chơi khác nhau. Trò chơi tập trung vào yếu tố thử thách với độ khó tăng dần, yêu cầu người chơi phải nhanh nhẹn, tập trung và phát triển kỹ năng chiến đấu để tiêu diệt quái vật và boss trong mỗi màn chơi. Điều này đáp ứng nhu cầu của nhiều người chơi, đặc biệt là những người yêu thích các tựa game hành động và phiêu lưu đậm chất retro.

Việc phát triển Blackout Ops không chỉ giúp em rèn luyện và nâng cao kỹ năng lập trình mà còn là cơ hội để em thử nghiệm các công nghệ và phương pháp phát triển game hiện đại. Trong quá trình phát triển, em đã xây dựng một hệ thống điều khiển mượt mà và linh hoạt, cho phép người chơi dễ dàng tương tác với nhân vật thông qua bàn phím và chuột. Bên cạnh đó, hệ thống quái vật và vật phẩm trong game được thiết kế với mức độ phức tạp tăng dần, mang đến trải nghiệm thử thách cho người chơi ở mỗi màn chơi. Giao diện của game được thiết kế đơn giản, trực quan nhưng vẫn đảm bảo tính thẩm mỹ và dễ sử dụng, giúp người chơi nhanh chóng làm quen và đắm mình vào thế giới của Blackout Ops. SDL hỗ trợ đồ họa 2D tốt, nhưng nhược điểm là chưa hỗ trợ đầy đủ cho đồ họa 3D và các hiệu ứng phức tạp, tuy nhiên đối với thể loại platformer như Blackout Ops, các tính năng của SDL vẫn đáp ứng rất tốt nhu cầu hiện tại.

Mặc dù đã đạt được những kết quả nhất định, nhưng em nhận thấy vẫn còn một số hạn chế và thiếu sót trong quá trình phát triển game. Một trong những hạn chế lớn là mở rộng tính năng multiplayer để người chơi có thể chia sẻ và cạnh tranh điểm số với nhau. Ngoài ra, hệ thống bảo mật và tối ưu hóa giao diện người dùng vẫn cần được cải thiện để nâng cao tính an toàn và tiện lợi cho người chơi. Hơn nữa, đồ họa game hiện tại mới chỉ dừng lại ở mức độ 2D cơ bản và chưa thể

phát triển các hiệu ứng hình ảnh phức tạp hơn. Giao diện vẫn cần được tối ưu hóa để thân thiện hơn với người dùng và nâng cao tính thẩm mỹ của trò chơi.

Trong tương lai, em sẽ tập trung vào việc khắc phục những hạn chế này bằng cách nâng cấp game lên một phiên bản tốt hơn. Một trong những mục tiêu sắp tới là tích hợp AI (trí tuệ nhân tạo) để cải thiện hành vi của quái vật, làm cho chúng trở nên thông minh và khó đoán hơn, từ đó tăng độ hấp dẫn và thử thách của trò chơi. Hệ thống thống kê dữ liệu và quản lý người chơi cũng cần được phát triển thêm để dễ dàng theo dõi và quản lý các hoạt động trong game. Việc tăng cường bảo mật thông tin người chơi và chống gian lận cũng sẽ được chú trọng hơn trong các phiên bản tiếp theo để đảm bảo tính công bằng và an toàn cho người chơi. Với sự nỗ lực không ngừng trong việc học hỏi và phát triển, em tin rằng Blackout Ops sẽ ngày càng hoàn thiện và đáp ứng tốt hơn nhu cầu giải trí của cộng đồng game thủ.

Mặc dù đồ án tốt nghiệp vẫn còn một số hạn chế, nhưng nó đã đóng góp vào quá trình học tập và phát triển kỹ năng của em. Em sẽ cố gắng cải thiện sản phẩm hoàn thiện để phù hợp hơn với nhu cầu của người dùng và thị trường. Em tin rằng, với sự nỗ lực không ngừng, game mà em đã phát triển sẽ ngày càng hoàn thiện và mang lại nhiều giá trị cho người dùng.

TÀI LIỆU THAM KHẢO

1. Tác phẩm sách

- [1] Vũ Thị Dương, Phùng Đức Hòa, Nguyễn Thị Hương Lan. Giáo trình phân tích thiết kế hướng đối tượng, 2015.
- [2] Shaun Mitchell. Sdl Game Development, 2020.
- [3] Janine Suvak. Lập trình Game với Unity, 2015.

2. Tài liệu học thuật trực tuyến

- [1] Học cơ bản về lập trình game cơ bản. URL: <https://lazyfoo.net/tutorials/SDL/>. Lần truy cập gần nhất ngày: 07/05/2025.
- [2] Hướng dẫn về SDL. URL: <https://wiki.libsdl.org/SDL2/Tutorials>. Lần truy cập gần nhất ngày: 18/05/2025.