

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

=====***=====



ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KỸ THUẬT PHẦN MỀM

ĐỀ TÀI:
TÌM HIỂU VỀ UNITY VÀ XÂY DỰNG TRÒ CHƠI LASER
DEFENDER

CBHD:	Th.S Đỗ Ngọc Sơn
Sinh viên:	Phạm Văn Đồng
Mã số sinh viên:	2020607343

Hà Nội, năm 2024

LỜI CẢM ƠN

Em xin trân trọng cảm ơn Th.S Đỗ Ngọc Sơn đã tận tình dẫn dắt, định hướng từ những ngày đầu tiên thực hiện Đồ án tốt nghiệp. Xuất phát từ những ý tưởng ban đầu, vạch ra kế hoạch và thực hiện đều được sự tư vấn, hướng dẫn chi tiết của thầy để em có thể đưa ra bản Đồ án tốt nghiệp hoàn thiện cuối cùng.

Em xin trân trọng cảm ơn các thầy, cô giáo trong khoa Công nghệ thông tin, trường Đại học Công Nghiệp Hà Nội đã đưa ra những góp ý, phản biện hữu ích và sâu sắc giúp cho Đồ án tốt nghiệp và sản phẩm của em được hoàn thiện và đạt giá trị ở mức cao hơn.

Em xin chân thành cảm ơn trường Đại học Công Nghiệp Hà Nội nói chung và khoa Công nghệ thông tin nói riêng, đã luôn tạo điều kiện tốt nhất về cả cơ sở vật chất và giảng dạy, hỗ trợ sinh viên một cách tối đa. Từ đó, sinh viên có cơ hội học tập, phấn đấu, trau dồi kiến thức trên trường lớp lẫn thực hành trong thực tiễn nhằm tạo ra những con người có ích cho đất nước, xã hội trong tương lai.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Phạm Văn Đồng

LÝ DO CHỌN ĐỀ TÀI

Trò chơi điện tử, bắt đầu xuất hiện từ khoảng giữa những năm 1950 và 1960. Đã trở nên càng ngày phổ biến trong đời sống hiện đại và phát triển thành một ngành công nghiệp mạnh trong lĩnh vực giải trí của con người. Vậy trò chơi điện tử là gì?

Trò chơi điện tử, hay còn gọi là Video Game, là một loại hình giải trí tương tác trong đó người chơi tương tác với một hệ thống máy tính hoặc các thiết bị điện tử thông minh để điều khiển và tương tác với hình ảnh video hoặc đồ họa động được hiển thị trên màn hình. Trò chơi điện tử có thể chia thành nhiều thể loại, bao gồm đối kháng, phiêu lưu, chiến thuật, thể thao, và nhiều thể loại khác.

Ngoài ra trong trò chơi điện tử, người chơi thường điều khiển các nhân vật hoặc đối tượng trong môi trường ảo và tham gia vào các nhiệm vụ, thách thức, hoặc các tình huống tương tác khác. Trò chơi điện tử có thể được thiết kế cho nhiều nền tảng, bao gồm máy tính cá nhân, Console Game (như PlayStation, Xbox), điện thoại di động, máy tính bảng, và nhiều hệ thống khác.

Ngày nay, trong thời đại công nghệ 4.0. Ngành công nghiệp Game càng ngày càng phát triển mở rộng một cách lớn mạnh và ảnh hưởng một cách sâu sắc đến đời sống giải trí của con người hiện đại. Thế nhưng, cũng trong thời đại này, quyền lực và tài nguyên của ngành công nghiệp càng ngày tập trung vào tay của các nhà đội ngũ phát triển game lớn. Họ thường kiểm soát trong việc những dự án nào được phát triển và phát hành. Điều này dẫn đến việc các ý tưởng mới, phá cách và đôi khi là những trải nghiệm nghệ thuật độc đáo bị lạc lõng hoặc bị từ chối. Sự tập trung vào lợi nhuận và các mô hình kinh doanh truyền thống khiến cho các tựa game được ra mắt khó có sự đổi mới và thường đi theo một lối mòn gây nhàm chán.

Trong tình cảnh đó, đã xuất hiện một thuật ngữ mới mang tên “Indie Game”, hay còn gọi là trò chơi điện tử độc lập, là những sản phẩm được phát triển bởi các đội ngũ phát triển game nhỏ, độc lập và thường là không thuộc sở

hữu của các hãng phát triển lớn. Sự xuất hiện của Indie Game là một phản kháng rõ ràng từ cộng đồng phát triển game đối với sự độc quyền và quyền lực của các nhà sản xuất game lớn, đồng thời cũng là phản ứng của cộng đồng người chơi, những người đang tìm kiếm những trải nghiệm mới, không giới hạn bởi các chuẩn mực thị trường.

Indie Game mở ra một không gian cho các nhóm phát triển nhỏ và độc lập, không bị ràng buộc bởi áp lực từ các nhà xuất bản lớn. Các nhà phát triển Indie có tự do để thử nghiệm những ý tưởng mới, thậm chí là những dự án có nguy cơ cao mà các đội lớn có thể sẽ tránh xa. Điều này tạo nên sự đa dạng không chỉ trong cách chơi mà còn trong cách người chơi tương tác với nghệ thuật và câu chuyện. Thế nên các tựa game độc lập thường nhận được sự ủng hộ mạnh mẽ từ người hâm mộ và cộng đồng, giúp chúng phát triển và tiếp tục tồn tại mặc dù không có sự hỗ trợ lớn từ các nhà sản xuất hàng đầu.

Tóm lại, Indie Game là một xu hướng mạnh mẽ và sẽ càng ngày phát triển lớn mạnh trong tương lai. Với sự giúp đỡ của các Game Engine mới phù hợp cho các nhà phát triển độc lập như Unity, Godot,.. Nó ảnh hưởng sâu sắc đến ngành công nghiệp game, không chỉ ở các nước có lịch sử phát triển game lâu năm, mà còn ở các nước đang phát triển, trong đó có Việt Nam.

Nhận thức được sức ảnh hưởng của Indie Game và mong muốn có thể đóng góp một phần sức lực của bản thân trong sự phát triển của Indie Game tại Việt Nam. Em xin được áp dụng những kiến thức đã được học và tìm hiểu để xây dựng đề tài: **“Nghiên cứu Unity Engine, xây dựng ứng dụng Game 2D Laser Defender”**.

MỤC LỤC

LỜI CẢM ƠN.....	2
LÝ DO CHỌN ĐỀ TÀI.....	3
Chương 1. TỔNG QUAN VỀ DÒNG GAME VÀ MÔI TRƯỜNG LẬP TRÌNH.....	10
1.1 Giới thiệu đề tài	10
1.2 Giới thiệu về Indie Game	11
1.2.1 Game Indie là gì?	11
1.2.2 Lịch sử phát triển của game Indie	11
1.2.3 Ưu điểm.....	12
1.2.4 Nhược điểm.....	13
1.3 Giới thiệu phần mềm Unity.....	13
1.3.1 Ưu nhược điểm của phần mềm	14
1.3.2 Hướng dẫn tải và cài đặt (môi trường windows)	16
1.3.3 Giao diện phần mềm Unity	18
Chương 2. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG GAME LASER DEFENDER.....	25
2.1. Giới thiệu tổng quan.....	25
2.1.1. Thông tin game.....	25
2.1.2. Thể loại game và yếu tố game.....	25
2.1.3. Đối tượng chơi	25
2.1.4. Nền tảng	26
2.2. Kịch bản game.....	26
2.2.1. Mô tả	26

2.2.2.	Luật chơi	26
2.2.3.	Nhân vật (Player)	27
2.2.4.	Kẻ địch (enemies)	28
2.2.5.	Item	29
2.2.6.	Chướng ngại vật	29
2.2.7.	Tương tác và điều khiển game	30
2.3.	Storyboard	31
2.3.1.	Sơ đồ các màn hình	31
2.4.	Tài nguyên	31
2.4.1.	Hình ảnh	31
2.4.2.	Màu sắc	32
2.4.3.	Âm thanh	32
2.4.4.	Font chữ	33
2.4.5.	Hiệu ứng và animation	33
Chương 3.	XÂY DỰNG GAME LASER DEFENDER.....	34
3.1.	Các kỹ thuật thực hiện.....	34
3.1.1.	Tạo giao diện cho game	34
3.1.2.	Singleton Pattern	34
3.1.3.	Sprite	35
3.1.4.	Collision Detection	35
3.1.5.	UI và HUD	36
3.1.6.	Code chức năng chính của game.....	37
3.2.	Sản phẩm màn hình	44
3.2.1.	Màn hình bắt đầu.....	44

3.2.2.	Màn hình chọn nhân vật.....	44
3.2.3.	Màn hình chơi game.....	45
3.2.4.	Màn hình thua cuộc.....	46
3.2.5.	Màn hình cài đặt.....	46
3.2.6.	Màn tạo phòng/ vào phòng	47
3.2.7.	Màn hình nhập tên.....	47
3.2.8.	Màn hình nhiều người chơi.....	48
KẾT LUẬN.....		49
TÀI LIỆU THAM KHẢO.....		51

DANH MỤC HÌNH ẢNH

Hình 2.1: Hình ảnh minh họa Indie Game	11
Hình 1-1: Các hệ điều hành Unity nhắm tới.	14
Hình 1-2: Đăng nhập Unity Hub	16
Hình 1-3: Kích hoạt Unity Hub	17
Hình 1-4: Chọn phiên bản để cài đặt công cụ Unity	17
Hình 1-5: Giao diện chính của phần mềm Unity	18
Hình 1-6: Giao diện project	18
Hình 1-7: Giao diện màn hình Hierarchy	19
Hình 1-8: Giao diện màn hình Inspector.	20
Hình 1-9: Giao diện màn hình Scence.	21
Hình 1 -10: Giao diện màn hình Game.	22
Hình 2.3.1: Storyboard của game.	31
Hình 2.4.1: Hình ảnh các đối tượng trong game	31
Hình 2.4.2: Tương phản màu sắc thân thiện với mắt	32
Hình 3.1.1: Thiết kế mà hình gameplay trong photoshop.	34
Hình 3.1.2: Singleton Pattern quản lý các đối tượng trong game.	34
Hình 3.1.3: Hàm di chuyển player.	37
Hình 3.1.4: Hàm quản lý làn sóng enemy	38
Hình 3.1.5: Hàm Spawn enemy.	39
Hình 3.1.6: Hàm tìm đường đi của enemy.	40
Hình 3.1.7: Hàm tấn công.	41
Hình 3.1.8: Hàm kết nối với server	42
Hình 3.1.9: Hàm kết nối với server	43
Hình 3.2.1: Màn hình bắt đầu.	44

Hình 3.2.2: Màn hình Shop.	44
Hình 3.2.3: Màn hình chọn nhân vật.	45
Hình 3.2.4: Màn hình chơi game.	45
Hình 3.2.5: Màn hình thua cuộc.	46
Hình 3.2.6: Màn hình cài đặt	46
Hình 3.2.7: Màn hình tạo/ vào phòng	47
Hình 3.2.8: Màn hình chọn tên	47
Hình 3.2.9: Màn hình nhiều người chơi	48

Chương 1. TỔNG QUAN VỀ DÒNG GAME VÀ MÔI TRƯỜNG LẬP TRÌNH

1.1 Giới thiệu đề tài

Một trò chơi hay video game là một trải nghiệm điện tử tương tác, đòi hỏi người chơi tương tác với giao diện người dùng hoặc thiết bị đầu vào. Các phản hồi trực quan được tạo ra thông qua các thiết bị như bàn điều khiển, bàn phím, hay thiết bị cảm biến chuyển động. Những phản hồi này thường xuất hiện trên màn hình hiển thị video, chẳng hạn như màn hình máy tính, TV, hoặc thiết bị tai nghe thực tế ảo. Tiếng ồn từ trò chơi thường được truyền tải qua loa hoặc tai nghe, và đôi khi cảm nhận haptic được tích hợp qua công nghệ xúc giác.

Trò chơi video được xác định dựa trên nền tảng của chúng, bao gồm các thể loại như trò chơi arcade, trò chơi trên máy console và trò chơi trên máy tính cá nhân (PC). Gần đây, lĩnh vực này đã mở rộng sang trò chơi di động thông qua điện thoại thông minh và máy tính bảng, cũng như kết hợp với hệ thống thực tế ảo và thực tế tăng cường. Trò chơi Laser Defender là một tác phẩm độc đáo với lối chơi phong phú và đa dạng, trò chơi hứa hẹn mang đến cho người chơi những trải nghiệm đáng nhớ về cả mặt hình ảnh và âm thanh.

Laser Defender là một trải nghiệm hành động phiêu lưu đầy kịch tính, đưa người chơi vào vai anh hùng điều khiển một chiến hạm chiến đấu với các thế lực ngoài hành tinh. Trò chơi này thực sự tạo nên một trải nghiệm sống động, với đồ họa tuyệt vời và âm thanh sống động. Chúng ta sẽ cùng nhau khám phá và thảo luận về sự thú vị của game Laser Defender này và cách nó mang đến cho chúng ta những phút giây giải trí và thách thức đáng nhớ.

1.2 Giới thiệu về Indie Game



Hình 2.1: Hình ảnh minh họa Indie Game

1.2.1 Game Indie là gì?

Indie Game, hay còn gọi là trò chơi độc lập, là một dòng thể loại game tập trung vào sự độc lập sáng tạo của các đội ngũ phát triển nhỏ, thường là các nhóm độc lập hoặc cá nhân, nằm ngoài tầm kiểm soát của các đại gia trong ngành công nghiệp game. Điều quan trọng không chỉ là sản phẩm cuối cùng mà còn là quá trình phát triển, nơi sự tự do và độ sáng tạo không bị giới hạn.

Indie Game không tuân theo các tiêu chí hay yêu cầu thị trường chung, mà thường xuyên mang lại những trải nghiệm độc đáo, dựa trên cái nhìn cá nhân của người sáng tạo. Các đội ngũ này thường tự chủ trong việc thiết kế, sản xuất và phân phối sản phẩm của mình, tạo ra một không gian mà không bị áp lực bởi thị trường hay quy định từ các công ty lớn.

1.2.2 Lịch sử phát triển của game Indie

Indie game không chỉ là một hiện tượng mới, mà là một tiến triển đáng chú ý trong thế giới game. Thập kỷ 2000 chứng kiến một bước ngoặt quan trọng khi các công cụ phát triển game trở nên dễ dàng tiếp cận hơn và internet mở ra cơ hội cho việc quảng bá không giới hạn. Các nhà phát triển có thể chia sẻ và

bán các sản phẩm của họ trực tiếp với cộng đồng trực tuyến, giảm đi sự phụ thuộc vào các đối tác lớn.

Những tựa game như “Braid”(2008) của Jonathan Blow hay “Limbo”(2010) của PlayDead là những điểm mốc quan trọng, khiến người chơi và ngành công nghiệp phải chú ý đến khả năng sáng tạo của indie game. Càng về sau, các tựa game như “Undertale”(2015) của Toby Fox hay “Hollow Knight”(2017) của Team Cherry đã chứng minh rằng indie game không chỉ là những thử nghiệm nhỏ mà còn là những tác phẩm nghệ thuật có tầm ảnh hưởng lớn đối với cả ngành công nghiệp game toàn cầu.

Indie Game tại Việt Nam cũng đã góp phần vào sự sôi nổi của ngành công nghiệp game toàn cầu. Trong những năm gần đây, cộng đồng game độc lập Việt Nam đã trở thành một phần quan trọng trong hành trình phát triển của indie game trên thế giới. Một ví dụ điển hình là “Hoa”(2020) của Skrollcat Studio, một tựa game đẹp mắt với đồ họa tinh tế và âm nhạc giao hưởng độc đáo. Ngoài ra, “Thần Trùng”(2022) của DUT Studio là một ví dụ khác về sự đa dạng trong indie game ở Việt Nam. Trò chơi này sử dụng nguyên tắc của kinh dị tâm lý và đồ họa 3D để tạo ra một trải nghiệm mới mẻ và kịch tính.

1.2.3 Ưu điểm

- Sáng tạo và độ tự do: Indie game thường mang đến cho người chơi những trải nghiệm độc đáo do sự sáng tạo không giới hạn và độ tự do của các nhà phát triển.
- Khám phá ý tưởng mới: Các đội ngũ nhỏ thường dám thử nghiệm với các ý tưởng mới và đôi khi mang đến cho người chơi những trò chơi mà các công ty lớn không dám thử nghiệm.
- Phong cách nghệ thuật đa dạng: Indie game thường chú trọng đến nghệ thuật và thiết kế, mang đến cho người chơi những trải nghiệm đồ họa và âm nhạc đa dạng và sáng tạo.
- Mang lại giải trí và giảm căng thẳng: Indie game có thể giúp người chơi giải trí và giảm căng thẳng sau một ngày làm việc mệt mỏi.

1.2.4 Nhược điểm

- Ngân sách hạn chế: Do hạn chế về nguồn lực tài chính, các đội ngũ indie thường phải làm việc với ngân sách hạn chế, có thể ảnh hưởng đến chất lượng sản phẩm.
- Quảng bá khó khăn: Thiếu nguồn lực tiếp thị và quảng bá, nhiều tựa game indie có thể gặp khó khăn khi cạnh tranh với các sản phẩm từ các công ty lớn.
- Không ổn định về doanh thu: Do không có sự hỗ trợ lâu dài từ các đối tác lớn, các đội ngũ indie thường phải đối mặt với rủi ro tài chính và không ổn định về doanh thu.

1.3 Giới thiệu phần mềm Unity

Unity là một game engine đa nền tảng được phát triển bởi Unity Technologies, mà chủ yếu để phát triển video game cho máy tính, consoles và điện thoại. Lần đầu tiên nó được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, đến nay đã mở rộng 27 nền tảng.

Unity hỗ trợ đồ họa 2D và 3D, các chức năng được viết chủ yếu qua ngôn ngữ C#. Trong 2D games, Unity cho phép nhập sprites và một renderer thế giới Công nghệ đa phương tiện 8 2D tiên tiến. Đối với 3D games, Unity cho phép thiết lập các đặc điểm kỹ thuật của các kết cấu và độ phân giải mà công cụ trò chơi hỗ trợ, cung cấp các hỗ trợ cho bump mapping, reflection mapping, parallax mapping, cảnh không gian ambient occlusion (SSAO), hiệu ứng bóng đổ bằng cách sử dụng shadow maps, render thiết lập toàn cảnh đến hiệu ứng.

Unity cung cấp các dịch vụ cho nhà phát triển, bao gồm: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity API, Unity Multiplayer, Unity Performance Reporting and Unity Collaborate.

Unity 3D Engine là một môi trường phát triển tích hợp, mạnh mẽ, hỗ trợ thao tác kéo thả, tùy biến giao diện nhanh chóng, trực quan. Cung cấp các công cụ xử lý đồ họa, tích hợp sẵn thư viện vật lý, tính toán va chạm...

Unity 3D Engine hỗ trợ phát triển cả game 2D và 3D, hỗ trợ nhiều nền tảng thông dụng như OSX, Linux, Window, Web, iOS, Window Phone 8, Android, PS3... với cộng đồng người dùng, hỗ trợ rộng lớn.



Hình 1-1: Các hệ điều hành Unity nhắm tới.

Unity 3D Engine có phiên bản miễn phí và trả phí, hỗ trợ chạy trên hệ điều hành Window và MacOS. Unity Engine hướng tới người sử dụng chuyên nghiệp và cả nghiệp dư, nên khá dễ để sử dụng. Với ngôn ngữ lập trình bằng C. Ngày nay rất nhiều nhà phát triển game lựa chọn Unity 3D Engine để phát triển bởi khả năng hỗ trợ đa nền tảng và sự mạnh mẽ tiện dụng của Unity 3D Engine. Đến với Unity, các bạn sẽ không cần phải băn khoăn về các vấn đề xử lý, các khái niệm đồ họa phức tạp... tất cả đều trở nên dễ dàng và nhanh chóng với Unity..

1.3.1 Ưu nhược điểm của phần mềm

Ưu điểm

- ✓ Đa nền tảng: Unity hỗ trợ phát triển ứng dụng trên nhiều nền tảng, bao gồm máy tính, điện thoại di động, máy tính bảng, console chơi game và thực tế ảo/ thực tế tăng cường.

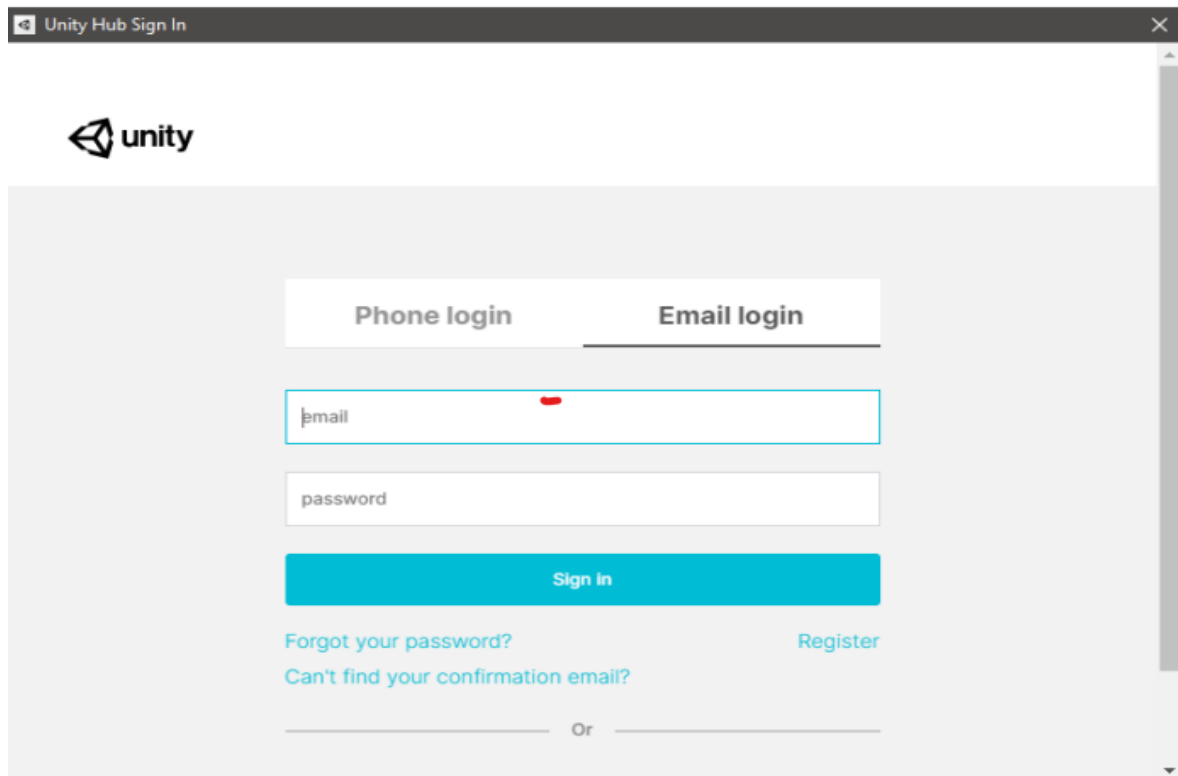
- ✓ Hỗ trợ đa ngôn ngữ và đa công nghệ: Unity hỗ trợ nhiều ngôn ngữ lập trình như C#, JavaScript và Boo. Ngoài ra, Unity cũng hỗ trợ các công nghệ và framework phổ biến như .NET, Mono, và Xamarin.
- ✓ Cộng đồng và tài liệu phong phú: Unity có một cộng đồng lớn và nhiệt tình, cung cấp nhiều tài liệu, hướng dẫn và nguồn tài nguyên. Bạn có thể tìm thấy các diễn đàn, trang web, video hướng dẫn, và các tài liệu chi tiết để giúp bạn trong quá trình phát triển.
- ✓ Công cụ thiết kế đồ họa và trực quan: Unity cung cấp một giao diện trực quan và các công cụ thiết kế đồ họa giúp bạn tạo và chỉnh sửa các tài nguyên như hình ảnh, âm thanh và đồ họa vector
- ✓ Hỗ trợ phát triển nhanh chóng: Unity cung cấp các khung thức và thư viện mạnh mẽ giúp bạn phát triển nhanh chóng các tính năng như vật lý, xử lý sự kiện, đa luồng, quản lý tài nguyên và trực quan hóa. Bạn cũng có thể tận dụng các giao diện người dùng và các tính năng giúp tăng tốc quá trình phát triển.
- ✓ Hỗ trợ kỹ thuật cao: Unity cung cấp hỗ trợ kỹ thuật chuyên nghiệp thông qua các kênh hỗ trợ trực tuyến, bao gồm tư vấn kỹ thuật, bản vá lỗi và cập nhật thường xuyên. Điều này giúp bạn giải quyết các vấn đề phát triển và vận hành ứng dụng một cách hiệu quả.

Nhược điểm

- ✗ Unity không phù hợp với các dự án lớn: Không thể đào sâu quá vừa là điểm mạnh vừa là điểm yếu của Unity. Một mặt, nó cho phép quy trình nhanh chóng, thích ứng tốt với người mới bắt đầu, mặt khác, điều đó có nghĩa là Unity có thể không phải là thứ bạn đang tìm kiếm nếu bạn đang hy vọng tạo ra bất cứ thứ gì thật đặc biệt hoặc trên quy mô lớn.
- ✗ Unity thúc đẩy các phương pháp mã xấu: Vấn đề nằm ở chỗ Unity không nhất thiết phải trở thành một công cụ trò chơi. Ban đầu nó được dùng để phát triển web và JavaScript

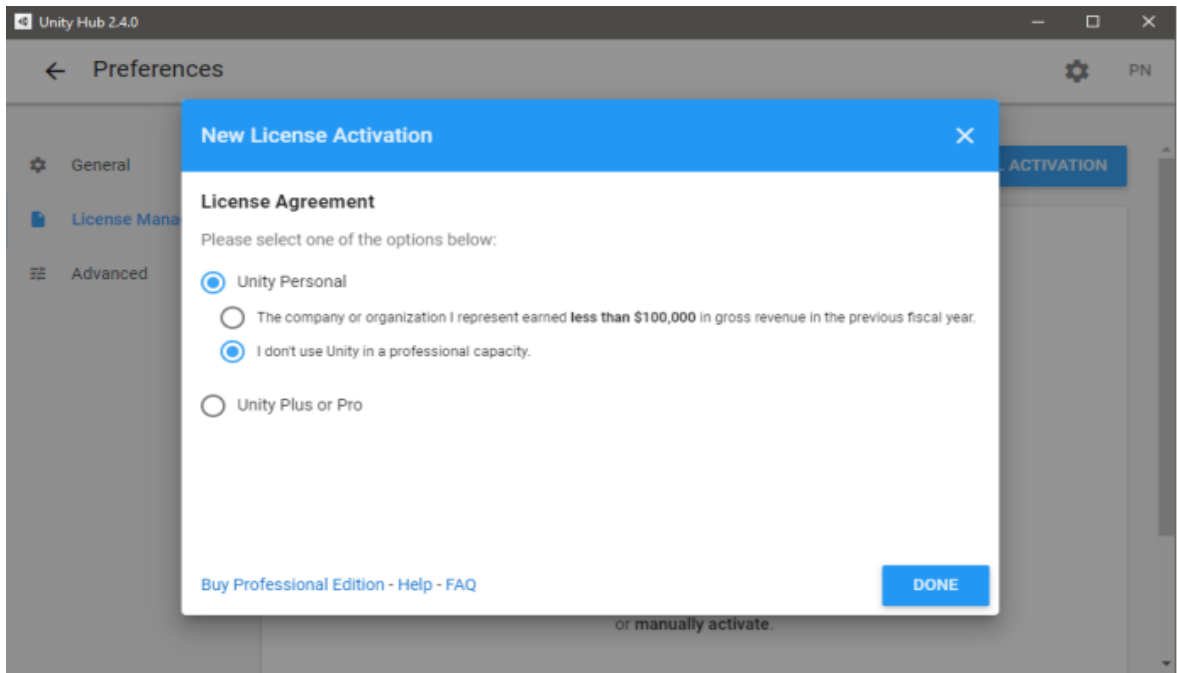
1.3.2 Hướng dẫn tải và cài đặt (môi trường windows)

Vào trang <https://unity.com/> chọn Download UnityHub, sau khi tải xuống hoàn tất double click vào file đã tải. Việc cài đặt UnityHub diễn ra bình thường và tương tự như cài đặt các chương trình khác. Sau khi cài đặt, khởi động UnityHub rồi đăng nhập vào tài khoản Unity, tài khoản Unity có thể được tạo miễn phí tại trang chủ.



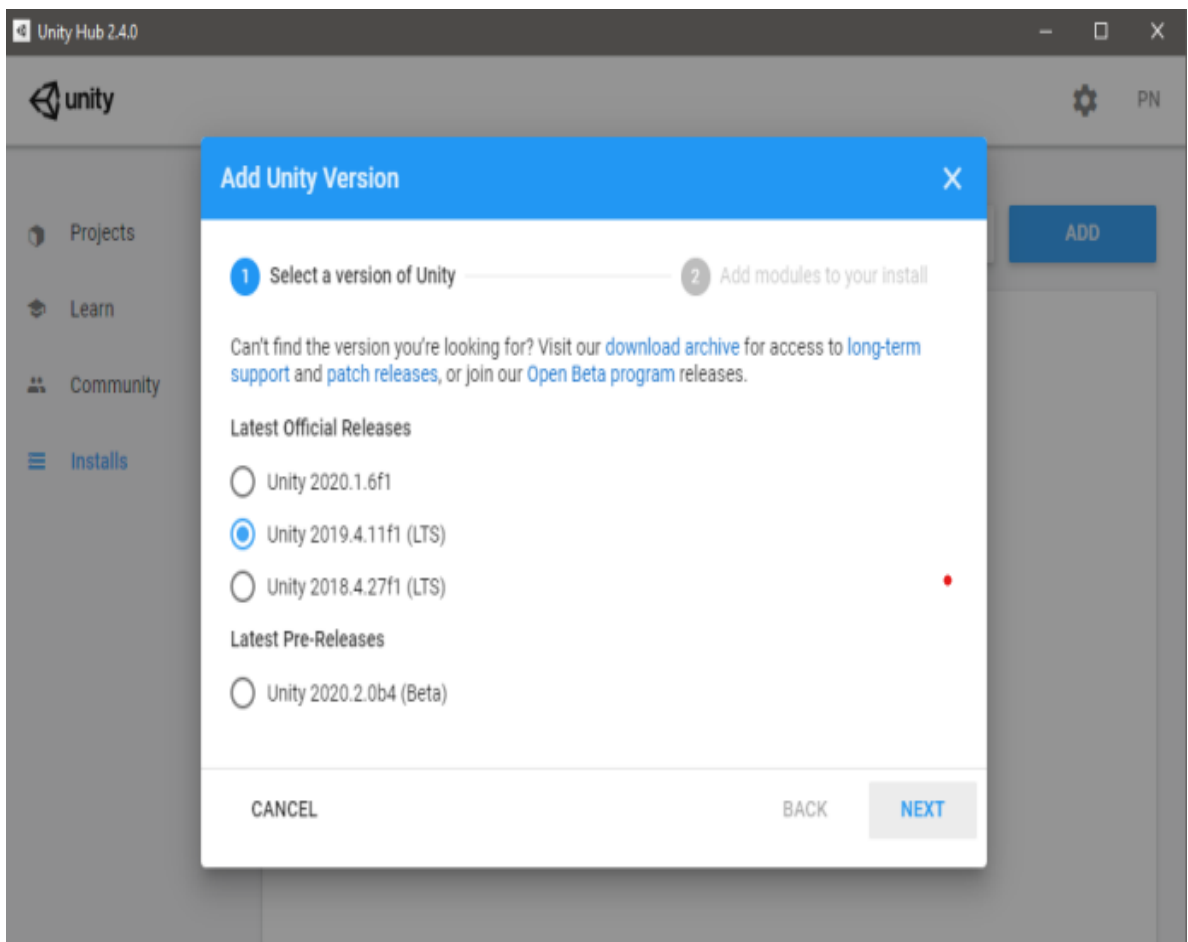
Hình 1-2: Đăng nhập Unity Hub

Sau khi đăng nhập trong trường hợp chưa có License thì phải tạo 1 License, vào Preferences bên cạnh profile góc trên phải và chọn License Management và chọn như hình để kích hoạt 1 License miễn phí.



Hình 1-3: Kích hoạt Unity Hub

Quay ra màn hình chính, chọn Installs->ADD để cài đặt Unity.



Hình 1-4: Chọn phiên bản để cài đặt công cụ Unity

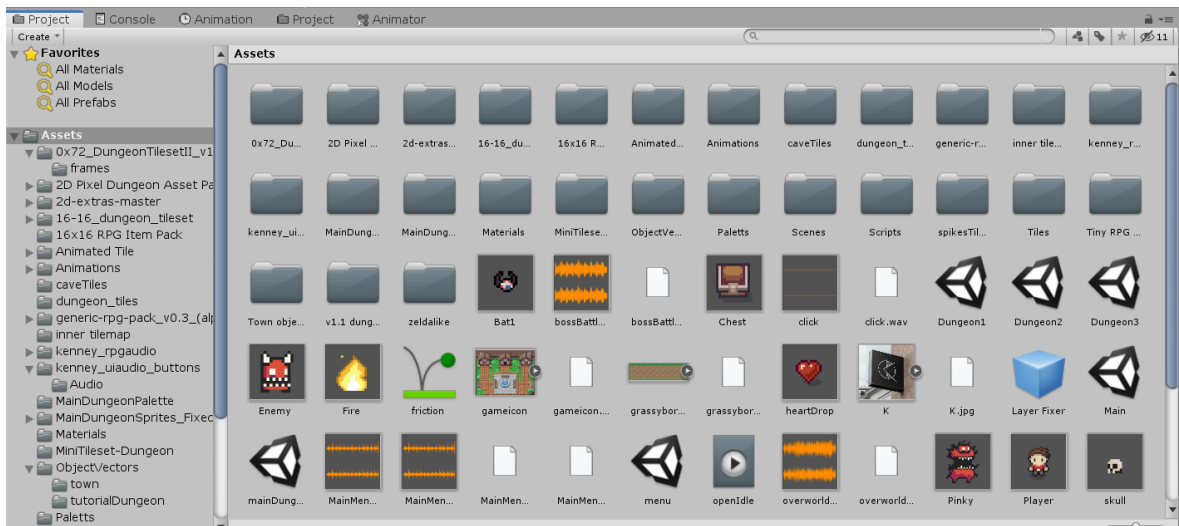
1.3.3 Giao diện phần mềm Unity

1.3.3.1 Giao diện chính



Hình 1-5: Giao diện chính của phần mềm Unity

1.3.3.2 Giao diện project



Hình 1-6: Giao diện project

- Là nơi hiển thị tài sản và tổ chức của dự án: Tập, script, kết cấu, mô hình,...

- Asset là các phần tử tồn tại dưới dạng tệp trong thư mục Assets: Kết cấu, mesh, tệp âm thanh, script,...
- Game Object là đối tượng một phần của cảnh (level).
- Có thể tạo Asset từ Game Object và có thể tạo Game Object từ Asset.
- Di chuyển Asset
 - Unity duy trì liên kết giữa các tài sản khác nhau liên quan đến các dự án.
 - Di chuyển hoặc xóa các phần tử bên ngoài Unity có thể gây ra sự cố, nên thực hiện việc quản lý tài sản bên trong Unity.
 - Khi nhấn vào một thư mục trong Project view, nội dung của thư mục được hiển thị trong phần Assets ở bên phải.
- Tổ chức Project
 - Các loại asset như cảnh, script, kết cấu,... nên có thư mục riêng.
 - Các nút Favorites cho phép chọn nhanh tất cả các asset thuộc một loại nhất định.
 - Tìm kiếm với thanh tìm kiếm sẽ thu hẹp kết quả giữa Assets và Asset Store.
 - Asset Store duyệt qua các tài sản phù hợp với tiêu chí tìm kiếm từ Unity Asset Store.
 - Có thể thu hẹp thêm kết quả theo nội dung miễn phí và trả phí

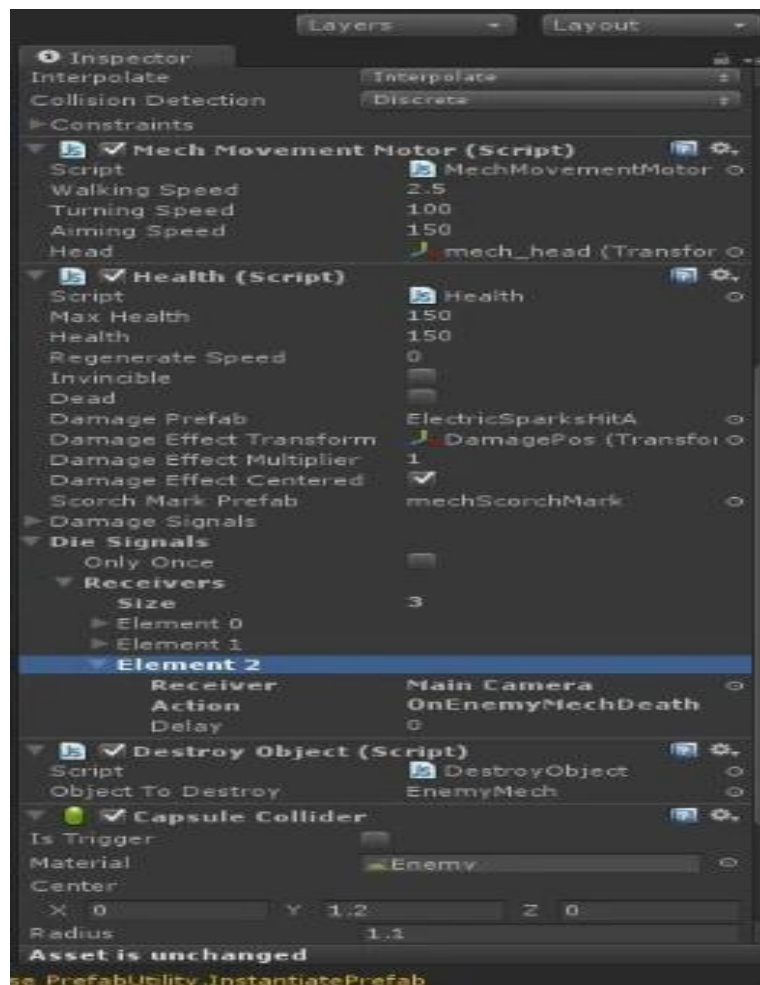
1.3.3.3 Hierarchy view



Hình 1-7: Giao diện màn hình Hierarchy

- Là nơi hiển thị tất cả các phần tử trong cảnh hiện tại thay vì toàn bộ dự án.
- Tạo dự án lần đầu tiên sẽ nhận được cảnh mặc định chỉ có hai phần tử là Main Camera và Directional Light.
- Khi thêm các phần tử vào cảnh, chúng sẽ xuất hiện trong Hierarchy View.

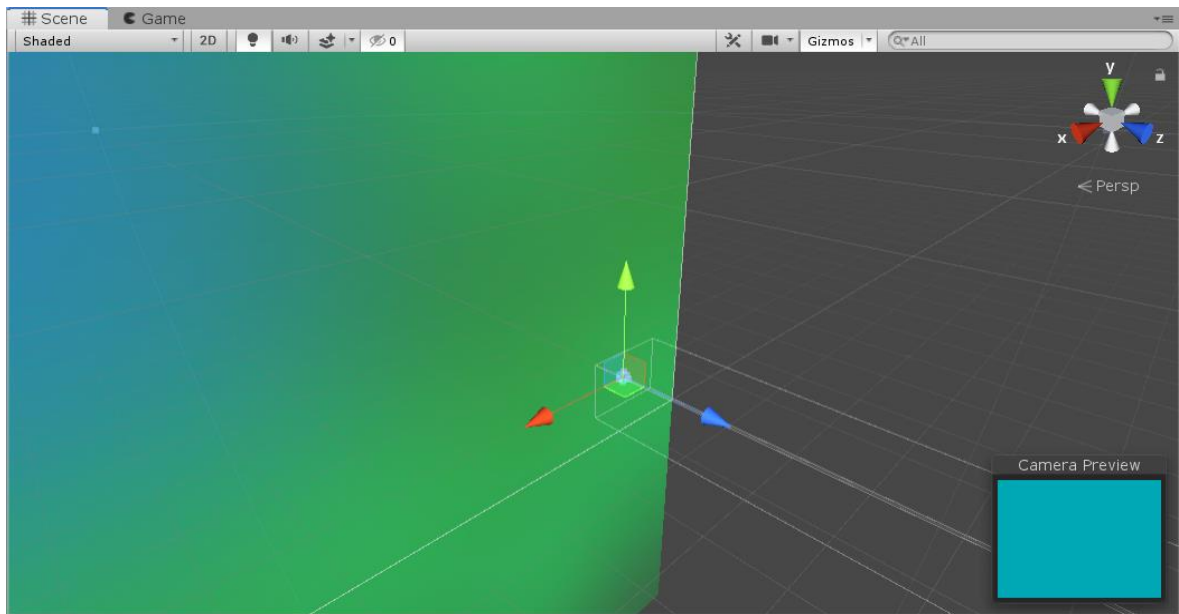
1.3.3.4 Inspector view



Hình 1-8: Giao diện màn hình Inspector.

- Cho phép xem tất cả các thuộc tính của một phần tử hiện đang được chọn.
- Nhấn vào đối tượng trong Project view hoặc Hierarchy view, Inspector sẽ hiển thị thông tin của đối tượng đó.
- Bỏ chọn hộp kiểm bên cạnh tên của đối tượng, nó sẽ bị vô hiệu hóa và không xuất hiện trong dự án.

1.3.3.5 Scene view



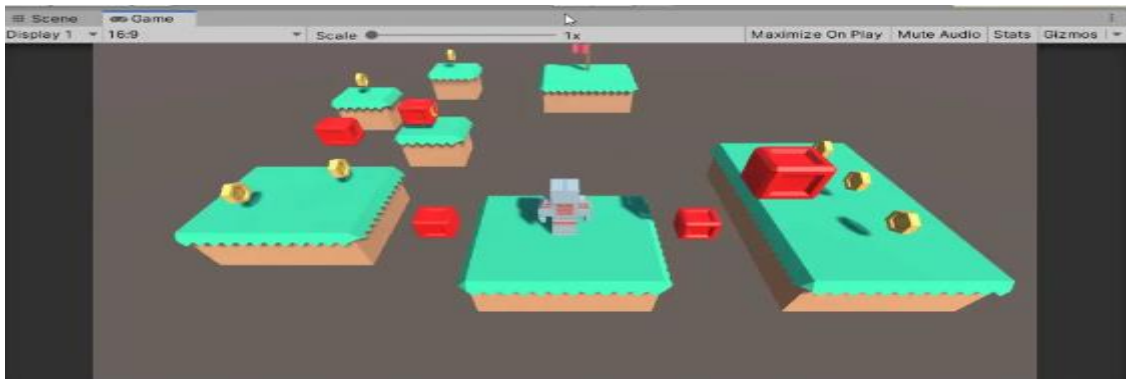
Hình 1-9: Giao diện màn hình Scence.

- Cho phép xem trò chơi một cách trực quan khi nó đang xây dựng.
 - Sử dụng chuột, phím để di chuyển trong cảnh và thiết lập vị trí cho các đối tượng.
 - Các điều khiển trong Scene view
 - Draw mode
 - Kiểm soát cách cảnh được vẽ.
 - Mặc định là Shaded, các đối tượng sẽ được vẽ với kết cấu màu sắc đầy đủ.
 - 2D/3D view
 - Thay đổi từ chế độ xem 3D sang chế độ xem 2D.
 - Ở chế độ xem 2D, scene gizmo không được hiển thị.
 - Gizmo selector
 - Cho phép chọn gizmos, nghĩa là các chỉ báo giúp đỡ lỗi trực quan hoặc hỗ trợ thiết lập xuất hiện trong scene view.
-

-
- Xác định xem lưới vị trí có hiển thị hay không.
 - Scene gizmo
 - Điều khiển này hiển thị hướng hiện đang đối diện và căn chỉnh scene view với một trục.
 - Có các chỉ báo X, Y và Z phù hợp với ba trục giúp dễ dàng nhận biết chính xác đang nhìn theo hướng nào trong cảnh
 - Nhấn vào một trong các trục của gizmo, scene view ngay lập tức bám vào trục đó và theo hướng đó.
 - Nhấn vào hộp ở giữa gizmo để chuyển đổi giữa chế độ Iso (Isometric) và Persp (Perspective)

1.3.3.6 Game view

- Cho phép “chơi” trò chơi bên trong trình editor bằng cách cung cấp mô phỏng đầy đủ về cảnh.
- Giao diện



Hình 1 -10: Giao diện màn hình Game.

- Nếu game view bị ẩn sau scene view hoặc không thấy tab của nó thì nhấn Play thì tab game view sẽ xuất hiện.
 - Các nút điều khiển
 - Play
 - Cho phép phát cảnh hiện tại.
-

-
- Tất cả các điều khiển, hoạt ảnh, âm thanh và hiệu ứng đều hiện diện và hoạt động.
 - Để dừng trò chơi đang chạy, nhấn lại vào nút Play.
 - Pause
 - Tạm dừng việc thực hiện game view hiện đang chạy.
 - Trò chơi duy trì trạng thái và tiếp tục chính xác vị trí của nó sau khi tạm dừng.
 - Nhấn vào nút Pause một lần nữa để trò chơi tiếp tục chạy.
 - Step
 - Hoạt động trong khi Game view bị tạm dừng và khiến trò chơi thực thi một khung hình duy nhất của trò chơi.
 - Cho phép “bước” qua trò chơi và gỡ lỗi gặp phải.
 - Nhấn vào nút Step trong khi trò chơi đang chạy khiến trò chơi tạm dừng.
 - Aspect drop-down
 - Chọn tỷ lệ khung hình cho cửa sổ game view trong khi chạy.
 - Mặc định là Free Aspect, có thể thay đổi để phù hợp với tỷ lệ khung hình của nền tảng đang phát triển.
 - Maximize on play
 - Xác định xem Game view có chiếm toàn bộ trình editor khi chạy hay không.
 - Mặc định, tính năng này bị tắt và trò chơi đang chạy chỉ có kích thước của tab Game view.
 - Mute Audio
 - Tắt âm thanh khi chơi trò chơi.
 - Điều này rất hữu ích khi cảm thấy khó chịu vì phải nghe bài test game lặp đi lặp lại.
-

-
- Stats
 - Xác định số liệu thống kê kết xuất có được hiển thị trên màn hình trong khi trò chơi đang chạy hay không.
 - Những số liệu thống kê này hữu ích để đo lường hiệu suất của cảnh.
 - Mặc định, các số liệu thông kê được tắt.
 - Gizmos
 - Đây vừa là nút vừa là menu thả xuống.
 - Xác định xem gizmos có được hiển thị trong khi trò chơi đang chạy hay không.
 - Mặc định, gizmo Game view không được hiển thị.
 - Menu thả xuống trên nút này xác định gizmo nào xuất hiện nếu gizmos được bật. Kéo camera xung quanh cảnh
-

Chương 2. PHÂN TÍCH VÀ THIẾT KẾ ỨNG DỤNG GAME LASER DEFENDER

2.1. Giới thiệu tổng quan

2.1.1. Thông tin game

Laser Defender là một trò chơi thuộc thể loại bắn súng không gian 2D, nơi người chơi sẽ điều khiển một tàu vũ trụ và chiến đấu chống lại đợt tấn công không ngừng từ đối thủ. Mục tiêu của trò chơi này là tạo ra một trải nghiệm giải trí hấp dẫn, kết hợp giữa tốc độ, kỹ năng và phản xạ của người chơi trong việc né tránh và tấn công kẻ địch.

Xây dựng game Laser Defender không chỉ là việc tạo ra một trò chơi giải trí mà còn là cơ hội để người chơi trau dồi kỹ năng và tận hưởng một trải nghiệm thú vị từ những pha hành động nhanh nhẹn và chiến đấu không khoan nhượng.

2.1.2. Thể loại game và yếu tố game

Thể loại: trò chơi 2d, hard core.

Game mang lại nhiều yếu tố như :

- Tăng khả năng quan sát và sự nhanh nhẹn
- Giúp tăng tư duy logic và trí nhớ
- Mang tính giải trí
- Độ khó vừa phải, tính thử thách cao

2.1.3. Đối tượng chơi

Với lối chơi game đơn giản bằng các thao tác dễ dàng và luật chơi dễ hiểu, game hiện tại đang phát triển tới đối tượng :

- Trẻ em từ 10 tuổi trở lên muốn rèn luyện sự nhanh nhẹn, quan sát.
 - Người lớn muốn xả stress hoặc chơi cùng con.
-

2.1.4. Nền tảng

Game được xây dựng bằng Unity nên có khả năng phát triển tốt trên các nền tảng lớn. Nhưng hiện tại game tập trung chủ yếu vào nền tảng Window(Laptop và PC) và Smart phone Android.

2.2. Kịch bản game

2.2.1. Mô tả

Đây là một trò chơi thể loại bắn súng 2D, lấy hình ảnh những chiến hạm vũ trụ để tăng cảm hứng cho người chơi. Mục đích của trò chơi là điều khiển chiến hạm của mình để chiến đấu chống lại kẻ địch để bảo vệ trái đất. Trò chơi được chơi ở góc nhìn thứ ba trong môi trường 2d. Điều này tạo cảm giác trực quan gây kích thích khả năng quan sát của người chơi.

- **Thế giới không gian:** Trò chơi diễn ra trong không gian với hình ảnh nền là không gian vũ trụ.
- **Player (người chơi):** Người chơi điều khiển một tàu vũ trụ có thể di chuyển ngang dọc và bắn lasers.
- **Enemies (kẻ địch):** Các kẻ địch xuất hiện từ các phía khác nhau của màn hình và tấn công player bằng cách bắn lasers.
- **Gameplay:** Người chơi cần né tránh các đợt tấn công của enemies bằng cách di chuyển và bắn hạ chúng trước khi bị tiêu diệt.

2.2.2. Luật chơi

➤ Người chơi sử dụng các phím mũi tên hoặc các nút điều hướng trên bàn điều khiển để di chuyển tàu vũ trụ né tránh kẻ địch và những đợt tấn công của chúng.

- A/ mũi tên trái: di chuyển sang trái
 - D/ mũi tên phải: di chuyển sang phải
-

-
- W/ mũi tên lên: di chuyển lên trên
 - S/ mũi tên xuống: di chuyển xuống dưới
 - Người chơi click chuột/ space để bắn laser tiêu diệt kẻ địch.
 - Đối với chế độ nhiều người chơi, người chơi có thể điều hướng bắn đạn bằng chuột.
 - Người chơi có thể tăng sức mạnh bằng cách nhặt các vật phẩm hỗ trợ.
 - Người chơi kiểm điểm số bằng cách tiêu diệt kẻ địch.

Điều kiện thắng:




Chế độ chơi đơn: Người chơi tiêu diệt quái vật cuối cùng (boss).


Điều kiện thua:

Sinh lực của người chơi về mức 0.

2.2.3. Nhân vật (Player)




- Người chơi điều khiển các chiến hạm của mình để chiến đấu với kẻ địch.



Chiến hạm	Sinh lực	Sát thương	Điểm
 Alpha	300	25	0
 Beta	300	30	0
 Metal	400	20	0

 Apolo	350	25	0
--	-----	----	---

2.2.4. Kẻ địch (enemies)

- Kẻ địch xuất hiện từ các phía của màn hình và xả đạn tấn công người chơi.
- Khi kẻ địch va chạm với người chơi thì sẽ gây sát thương lên người chơi.
- Người chơi có thể tấn công kẻ địch để nhận điểm.

Kẻ địch	Sinh lực	Sát thương	Điểm
 O-Sk	50	25	25
 B-Sk	75	20	25
 G-Sk	100	30	25

 F-Sk	75	35	25
 LordMT	1000	50	10000

2.2.5. Item

- Trong quá trình chơi người chơi có thể thu thập những vật phẩm có lợi cho mình:




pill-green: Hồi 50 sinh lực cho player.










pill-blue: Hồi 25 sinh lực cho player.

2.2.6. Chương ngại vật

- Trong quá trình chơi người chơi sẽ gặp phải một số khó khăn do va phải các thiên thạch xuất hiện ngẫu nhiên gây mất sinh lực player.
- Người chơi có thể tấn công để phá hủy và nhận điểm.

Thiên thạch	Sinh lực	Sát thương	Điểm
	100	25	50

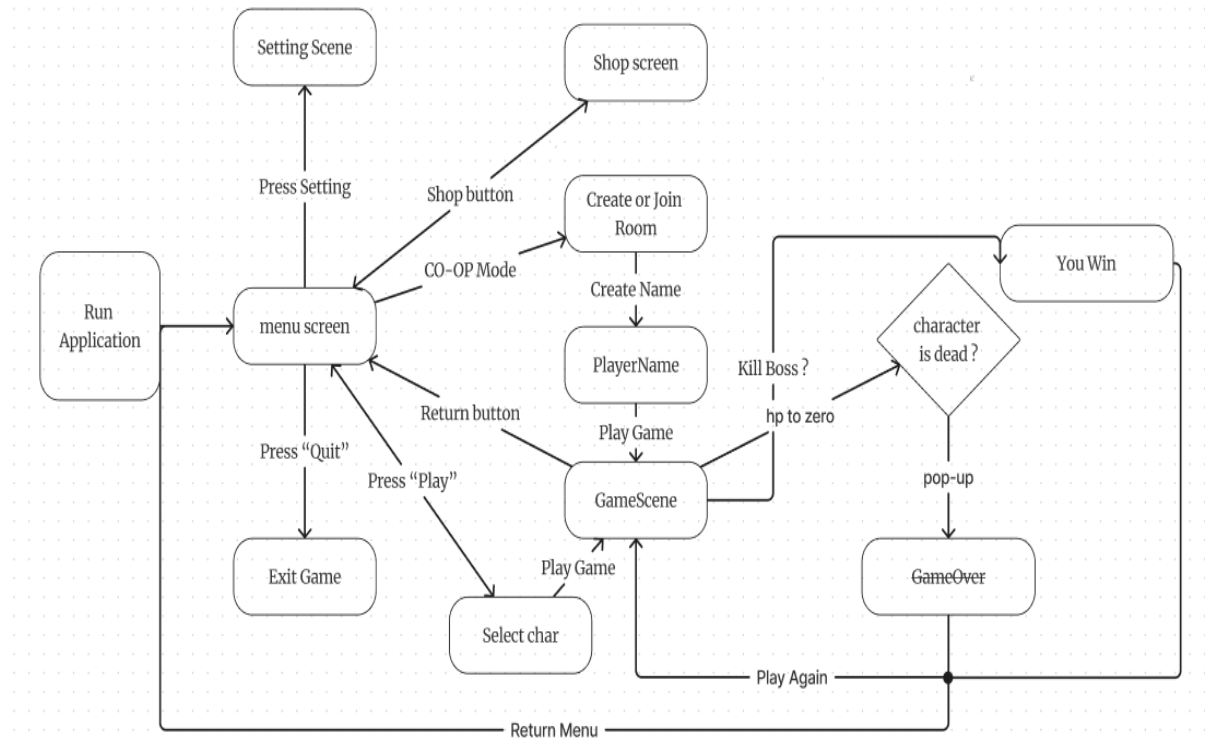
	125	75	80
	140	50	50
	125	60	75
	80	50	50
	130	70	130
	150	40	100
	50	25	25

2.2.7. Tương tác và điều khiển game

Tất cả các thao tác điều khiển trò chơi đều được thực hiện bởi cảm ứng (Smartphone), chuột (PC). Người chơi sử dụng chuột trái click hoặc nút space để tấn công, các nút điều hướng (AWSD/ các phím mũi tên) để di chuyển.

2.3. Storyboard

2.3.1. Sơ đồ các màn hình



Hình 2.3.1: Storyboard của game.

2.4. Tài nguyên

2.4.1. Hình ảnh

- Tàu vũ trụ và vật phẩm : <https://www.kenney.nl/assets/space-shooter-redux>



Hình 2.4.1: Hình ảnh các đối tượng trong game

2.4.2 Màu sắc

Các hình ảnh có màu sắc sặc sỡ trên nền background màu tối giúp làm nổi bật vật thể và kích thích người chơi.

Đa dạng màu sắc nhưng loại bỏ các cặp màu gây chói như: đỏ - xanh, trắng - đỏ, ... gây khó chịu cho mắt.



Hình 2.4.2: Tương phản màu sắc thân thiện với mắt

2.4.3. Âm thanh

- Sử dụng các âm thanh nhạc nền , click , âm thanh kèm theo màn thắng ,màn thua giúp game sinh động hơn.
- Âm thanh phát mỗi combo được tăng cao độ dựa trên số combo giúp nhịp chơi nhanh hơn, người chơi cảm thấy kịch tính hơn.

STT	Tên	Tác dụng	Nguồn
1	Juhani Junkala [Retro Game Music Pack]	Nhạc nền cho game	https://opengameart.org/content/5-chiptunes-action
2	sfx_laser2	Âm thanh bắn laser	https://www.kenney.nl/assets/space-shooter-redux
3	explosionCrunch_000	Âm thanh nhặt item	https://www.kenney.nl/assets/space-shooter-redux
4	sfx_shieldDown	Âm thanh trúng đạn	https://www.kenney.nl/assets/space-shooter-redux

2.4.4. Font chữ

Font: +) kenvector_future

+) kenvector_future_thin

Kích thước: tùy chỉnh

Màu chữ: tùy chỉnh

Nguồn: <https://www.kenney.nl/assets/space-shooter-redux>

2.4.5. Hiệu ứng và animation

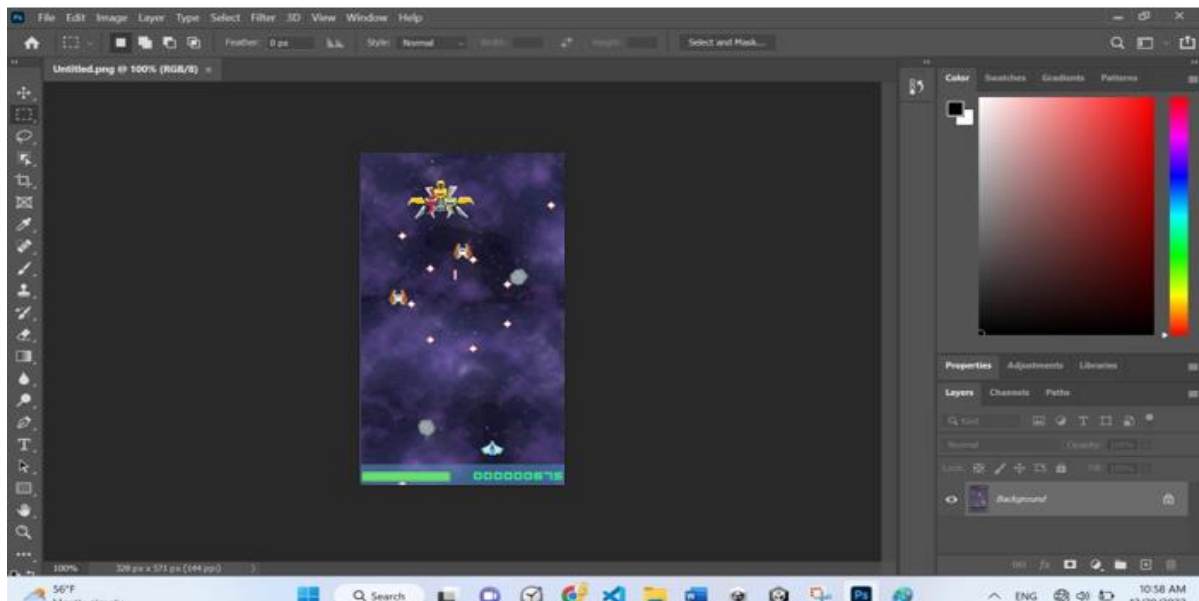
- ✓ Hover nên button sẽ đổi màu, tạo hiệu ứng được chọn.
- ✓ Hiệu ứng nổ khi trúng đạn.
- ✓ Hoạt ảnh xoay của thiên thạch, item.
- ✓ Hiệu ứng Scroll của backgroud.

Chương 3. XÂY DỰNG GAME LASER DEFENDER

3.1. Các kỹ thuật thực hiện

3.1.1. Tạo giao diện cho game

- Các kỹ thuật sử dụng để tạo lập giao diện cho game Laser Defender:
- ✓ Thực hiện thiết kế giao diện với phần mềm Adobe Photoshop sử dụng các tài nguyên hình ảnh ở trên.



Hình 3.1.1: Thiết kế màn hình gameplay trong photoshop.

3.1.2. Singleton Pattern

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

Hình 3.1.2: Singleton Pattern quản lý các đối tượng trong game.

Ứng dụng vào các đối tượng quản lý như GameManager(quản lý toàn bộ trạng thái game), AudioManager(quản lý trình phát âm thanh), DataManager(quản lý việc lưu và xuất toàn bộ dữ liệu của người chơi), ...Pattern này thực sự hữu ích

khi bạn cần có chính xác một đối tượng quản lý, điều phối trên toàn bộ scene một cách nhanh chóng.

3.1.3. Sprite

Sprites là các đối tượng 2D đơn giản có hình ảnh đồ họa (được gọi là kết cấu) trên chúng. Unity sử dụng các họa tiết theo mặc định khi động cơ ở chế độ 2D. Khi được xem trong không gian 3D, các họa tiết sẽ có vẻ mỏng như tờ giấy, vì chúng không có chiều rộng Z.

Sprites luôn đối mặt với máy ảnh ở góc vuông góc trừ khi xoay trong không gian 3D.

Bất cứ khi nào Unity tạo ra một sprite mới, nó sử dụng một kết cấu. Kết cấu này sau đó được áp dụng trên GameObject mới và thành phần Sprite Renderer được đính kèm. Điều này làm cho trò chơi của chúng tôi hiển thị với kết cấu của chúng tôi, cũng như cung cấp cho nó các thuộc tính liên quan đến giao diện trên màn hình.

3.1.4. Collision Detection

Collision Detection (Phát hiện va chạm) là một khía cạnh quan trọng trong phát triển game, đặc biệt là trong game 2D và 3D. Nó cho phép bạn xác định khi nào các đối tượng trong game va chạm với nhau để thực hiện các hành động phản ứng như phản đòn, đánh thắng, hay xử lý các tương tác khác

Discrete Collision Detection: Trong phát hiện va chạm rời rạc, va chạm được kiểm tra ở một số điểm cố định trong một khung hình cố định. Nó thường nhanh và phù hợp cho hầu hết các trò chơi 2D.

Continuous Collision Detection: Trong phát hiện va chạm liên tục, va chạm được kiểm tra liên tục trong suốt quá trình di chuyển của đối tượng. Điều này đảm bảo rằng không có "xuyên qua" (tunnelling) trong trường hợp di chuyển nhanh. Tuy nhiên, nó có thể yêu cầu tính toán tốn kém hơn.

Rigidbody2D : là các thành phần cho phép xử lý vật lý, bao gồm cả va chạm. Khi bạn sử dụng Rigidbody, Unity sẽ tự động xử lý va chạm và tương tác vật lý với các collider.

BoxCollider2D là một thành phần trong Unity được sử dụng để xác định hình dạng và vùng va chạm của một GameObject trong không gian 2D. Nó là một loại collider phổ biến trong phát triển game 2D và thường được sử dụng cho các đối tượng có hình dạng hộp chữ nhật. Bạn có thể xử lý va chạm bằng cách sử dụng các hàm trình quản lý va chạm và callbacks trong code của bạn. Điều này cho phép bạn thực hiện các hành động cụ thể khi va chạm xảy ra.

3.1.5. UI và HUD

UI (User Interface) và HUD (Heads-Up Display) là hai phần quan trọng trong game để tạo ra giao diện người dùng và hiển thị thông tin quan trọng cho người chơi.

UI elements là các thành phần hoặc đối tượng dùng để hiển thị thông tin và tương tác với người chơi. Các UI elements thường bao gồm nút, hình ảnh, văn bản, thanh trượt, hộp thoại, và nhiều thành phần khác.

HUD elements là các thành phần của giao diện người dùng được hiển thị trực tiếp trên màn hình chơi game để cung cấp thông tin quan trọng cho người chơi. Chúng bao gồm thông tin như thanh máu, điểm số, vật phẩm hiện tại, bản đồ, thời gian còn lại, và nhiều thông tin khác.

HUD elements thường cố định ở một vị trí trên màn hình và không di chuyển theo camera hoặc nhân vật trong game. Điều này giúp người chơi có thể theo dõi thông tin quan trọng mà không cần chú ý đến việc di chuyển camera.

HUD elements thường được thiết kế để truyền tải thông tin một cách trực quan và dễ hiểu. Ví dụ, thanh máu có thể được hiển thị bằng hoạt ảnh trái tim, và số điểm có thể hiển thị bằng văn bản số.

3.1.6. Code chức năng chính của game

Các đoạn code chức năng chính của game Laser Defender:

➤ Hàm di chuyển nhân vật:

```

8  {
9      [SerializeField] float moveSpeed = 5f;
10     Vector2 rawInput;
11
12     [SerializeField] float paddingLeft;
13     [SerializeField] float paddingRight;
14     [SerializeField] float paddingTop;
15     [SerializeField] float paddingBottom;
16
17     Vector2 mindBounds;
18     Vector2 maxBounds;
19
20     Shooter shooter;
21
22     Unity Message | 0 references
23     void Awake() {}
24
25     Unity Message | 0 references
26     private void Start() {}
27
28     Unity Message | 0 references
29     void Update() {}
30
31     1 reference
32     void InitBounds() {}
33
34     1 reference
35     void Move()
36     {
37         Vector2 delta = rawInput * moveSpeed * Time.deltaTime;
38         Vector2 newPos = new Vector2();
39         newPos.x = Mathf.Clamp(transform.position.x + delta.x, mindBounds.x + paddingLeft, maxBounds.x - paddingRight);
40         newPos.y = Mathf.Clamp(transform.position.y + delta.y, mindBounds.y + paddingBottom, maxBounds.y - paddingTop);
41         transform.position = newPos;
42     }
43
44     0 references
45     void OnMove(InputValue value)
46     {
47         rawInput = value.Get<Vector2>();
48     }

```

Hình 3.1.3: Hàm di chuyển player.

➤ Hàm quản lý làn sóng kẻ địch

```

4
5 [CreateAssetMenu(menuName = "Wave Config", fileName = "New Wave Config")]
6 public class WaveConfigSO : ScriptableObject
7 {
8     [SerializeField] List<GameObject> enemyPrefabs;
9     [SerializeField] Transform pathPrefab;
10    [SerializeField] float moveSpeed = 5f;
11
12    [SerializeField] float timeBetweenEnemySpawns = 1f;
13    [SerializeField] float spawnTimeVariance = 0f;
14    [SerializeField] float miniumSpawnTime = 0.2f;
15
16
17    1 reference
18    public int GetEnemyCount()
19    {
20        return enemyPrefabs.Count;
21    }
22
23    1 reference
24    public GameObject GetEnemyPrefab(int index)
25    {
26        return enemyPrefabs[index];
27    }
28
29    1 reference
30    public Transform GetStartingWaypoint()
31    {
32        return pathPrefab.GetChild(0);
33    }
34
35    1 reference
36    public List<Transform> GetWaypoints()
37    {
38        List<Transform> waypoints = new List<Transform>();
39        foreach(Transform child in pathPrefab)
40        {
41            waypoints.Add(child);
42        }
43        return waypoints;
44    }
45
46    1 reference
47    public float GetMoveSpeed()
48    {
49        return moveSpeed;
50    }
51
52    1 reference
53    public float GetRandomSpawnTime()
54    {
55        float spawnTime = Random.Range(timeBetweenEnemySpawns - spawnTimeVariance,
56                                         timeBetweenEnemySpawns + spawnTimeVariance);
57        return Mathf.Clamp(spawnTime, miniumSpawnTime, float.MaxValue);
58    }
59 }

```

Hình 3.1.4: Hàm quản lý làn sóng enemy

➤ Hàm sinh kẻ địch

```

7 [SerializeField] List<WaveConfigSO> waveConfigs;
8 [SerializeField] float timeBetweenWaves;
9 WaveConfigSO currentWave;
10
11 [SerializeField] bool isLooping;
12 /// <summary>
13 ///
14 /// </summary>
15 [SerializeField] GameObject bossPref;
16 [SerializeField] bool isSpawn = true;
17 [SerializeField] int n_score = 500;
18 Scorekeeper scorekeeper;
19
20 Unity Message | 0 references
21 private void Awake()
22 {
23     scorekeeper = FindObjectOfType<Scorekeeper>();
24 }
25
26 Unity Message | 0 references
27 void Start()
28 {
29     StartCoroutine(SpawnEnemyWave());
30 }
31
32 Unity Message | 0 references
33 private void Update()...
34
35 1 reference
36 public WaveConfigSO GetCurrentWave()
37 {
38     return currentWave;
39 }
40
41 1 reference
42 IEnumerator SpawnEnemyWave()
43 {
44     do
45     {
46         foreach (WaveConfigSO wave in waveConfigs)
47         {
48             currentWave = wave;
49             for (int i = 0; i < currentWave.GetEnemyCount(); i++)
50             {
51                 Instantiate(currentWave.GetEnemyPrefab(i),
52                             currentWave.GetStartingWaypoint().position,
53                             Quaternion.Euler(0, 0, 180),
54                             transform);
55                 yield return new WaitForSeconds(currentWave.GetRandomSpawnTime());
56             }
57             yield return new WaitForSeconds(timeBetweenWaves);
58         }
59     }
60     while (isLooping);
61 }

```

Hình 3.1.5: Hàm Spawn enemy.

➤ Hàm tìm đường của enemy

```

5  public class Pathfinder : MonoBehaviour
6  {
7      EnemySpawn enemySpawner;
8      WaveConfigSO waveConfig;
9      List<Transform> waypoints;
10     int waypointIndex = 0;
11
12     private void Awake()
13     {
14         enemySpawner = FindObjectOfType<EnemySpawn>();
15     }
16     void Start()
17     {
18         waveConfig = enemySpawner.GetCurrentWave();
19         waypoints = waveConfig.GetWaypoints();
20         transform.position = waypoints[waypointIndex].position;
21     }
22
23     void Update()
24     {
25         FollowPath();
26     }
27
28     void FollowPath()
29     {
30         if(waypointIndex < waypoints.Count)
31         {
32             Vector3 targetPosition = waypoints[waypointIndex].position;
33             float delta = waveConfig.GetMoveSpeed() * Time.deltaTime;
34             transform.position = Vector3.MoveTowards(transform.position, targetPosition, delta);
35             if(transform.position == targetPosition)
36             {
37                 waypointIndex++;
38             }
39         }
40         else
41         {
42             Destroy(gameObject);
43         }
44     }

```

Hình 3.1.6: Hàm tìm đường đi của enemy.

➤ Hàm tấn công

```

27 void Start()
28 {
29     if (useAI)
30     {
31         isFiring = true;
32     }
33 }
34
35 @ Unity Message | 0 references
36 void Update()
37 {
38     Fire();
39 }
40
41 1 reference
42 void Fire()
43 {
44     if (isFiring && firingCoroutine == null)
45     {
46         firingCoroutine = StartCoroutine(FireContinuously());
47     }
48     else if (!isFiring && firingCoroutine != null)
49     {
50         StopCoroutine(firingCoroutine);
51         firingCoroutine = null;
52     }
53 }
54
55 1 reference
56 IEnumerator FireContinuously()
57 {
58     while (true)
59     {
60         GameObject instance = Instantiate(projectilePrefabs,
61         transform.position,
62         Quaternion.identity);
63         Rigidbody2D rb = instance.GetComponent<Rigidbody2D>();
64         if(rb != null)
65         {
66             rb.velocity = transform.up * projectileSpeed;
67         }
68         Destroy(instance, projectileLifeTime);
69
70         float timeToNextProjectile = Random.Range(baseFiringRate - firingRateVariance,
71         baseFiringRate + firingRateVariance);
72         timeToNextProjectile = Mathf.Clamp(timeToNextProjectile, minimumFiringRate, float.MaxValue);
73
74         audioPlayer.PlayShootingClip();
75
76         yield return new WaitForSeconds(timeToNextProjectile);
77     }
78 }

```

Hình 3.1.7: Hàm tấn công.

➤ Hàm kết nối server

```

1  using Photon.Pun;
2  using System.Collections;
3  using System.Collections.Generic;
4  using TMPPro;
5  using UnityEngine;
6
7  public class M_CreateAndJoinRoom : MonoBehaviourPunCallbacks
8  {
9      public TMP_InputField createInput;
10     public TMP_InputField joinInput;
11
12     private void Start()
13     {
14         PhotonNetwork.ConnectUsingSettings();
15     }
16
17     public override void OnConnectedToMaster()
18     {
19         base.OnConnectedToMaster();
20         PhotonNetwork.JoinLobby();
21     }
22
23     public override void OnJoinedLobby()
24     {
25         base.OnJoinedLobby();
26         Debug.Log("....");
27     }
28
29     public void CreateRoom()
30     {
31         PhotonNetwork.CreateRoom(createInput.text);
32     }
33
34     public void JoinRoom()
35     {
36         PhotonNetwork.JoinRoom(joinInput.text);
37     }
38
39     public override void OnJoinedRoom()
40     {
41         base.OnJoinedRoom();
42         PhotonNetwork.LoadLevel("M_Game");
43     }
44

```

Hình 3.1.8: Hàm kết nối với server

➤ Hàm Ai di chuyển nhân vật

```

102 void CaculatePath()
103 {
104     FindNearestPlayer();
105     Vector2 target = FindTarget();
106
107     if (seeker.IsDone() && (reachDestination || updateContinuesPath))
108     {
109         seeker.StartPath(transform.position, target, OnPathComplete);
110     }
111 }
112
113 1 reference
114 void OnPathComplete(Path p)
115 {
116     if (p.error) return;
117     path = p;
118     //Move to target
119     MoveToTarget();
120 }
121
122 1 reference
123 void MoveToTarget()
124 {
125     if (moveCoroutine != null) StopCoroutine(moveCoroutine);
126     moveCoroutine = StartCoroutine(MoveToTargetCoroutine());
127 }
128
129 1 reference
130 IEnumerator MoveToTargetCoroutine()
131 {
132     int currentWP = 0;
133     reachDestination = false;
134
135     while (currentWP < path.vectorPath.Count)
136     {
137         Vector2 direction = ((Vector2)path.vectorPath[currentWP] - (Vector2)transform.position).normalized;
138         Vector2 force = direction * moveSpeed * Time.deltaTime;
139         transform.position += (Vector3)force;
140
141         float distance = Vector2.Distance(transform.position, path.vectorPath[currentWP]);
142         if (distance < nextWPDistance)
143             currentWP++;
144
145         yield return null;
146     }
147     reachDestination = true;
148 }

```

Hình 3.1.9: Hàm kết nối với server

3.2. Sản phẩm màn hình

3.2.1. Màn hình bắt đầu



Hình 3.2.1: Màn hình bắt đầu.

3.2.2. Màn hình Shop



Hình 3.2.2: Màn hình Shop.

3.2.3. Màn hình chọn nhân vật



Hình 3.2.3: Màn hình chọn nhân vật.

3.2.4. Màn hình chơi game



Hình 3.2.4: Màn hình chơi game.

3.2.5. Màn hình thua cuộc



Hình 3.2.5: Màn hình thua cuộc.

3.2.6. Màn hình cài đặt



Hình 3.2.6: Màn hình cài đặt

3.2.7. Màn tạo phòng/ vào phòng



Hình 3.2.7: Màn hình tạo/ vào phòng

3.2.8. Màn hình nhập tên



Hình 3.2.8: Màn hình chọn tên

3.2.9. Màn hình nhiều người chơi



Hình 3.2.9: Màn hình nhiều người chơi

KẾT LUẬN

Việc phát triển một trò chơi 2D Indie Game là một công việc phức tạp và đòi hỏi nhiều thời gian và nỗ lực. Mặc dù bước đầu có chút khó khăn trong việc phát triển ứng dụng, tuy nhiên, với sự cố gắng nỗ lực và nghiên cứu đồng thời cùng sự hỗ trợ từ ThS. Đỗ Ngọc Sơn, em đã có thể thành công tạo ra được một trò chơi đầy thú vị và hấp dẫn bằng cách sử dụng Unity Engine.

Qua quá trình thực hiện đề tài, em đã tìm hiểu, tích lũy và học hỏi thêm được nhiều kinh nghiệm cũng như kiến thức công nghệ mới.

***Những kết quả đạt được:**

Về công nghệ:

- Hiểu được quá trình thiết kế một ứng dụng game thực tế đi từ bước cơ bản: khảo sát công cụ, thiết kế ý tưởng, phân tích hệ thống, thiết kế ứng dụng và kiểm thử.
- Nắm được các kiến thức lập trình game, sử dụng thành thạo ngôn ngữ C# để xây dựng hệ thống của game.
- Nắm vững về cách xây dựng ứng dụng game thông qua Unity Engine.
- Nắm được cách tích hợp các sản phẩm bên thứ ba như: Naughty Attributes, XInputDotNet,... để tối ưu ứng dụng game.
- Sử dụng thành thạo các công cụ hỗ trợ lập trình: Visual Studio.

Về ứng dụng sản phẩm:

- Cho phép người chơi trải nghiệm một cốt truyện game sâu sắc và đầy ý nghĩa
 - Tăng tính cuốn hút với người chơi bằng các cơ chế giải đố đa dạng.
 - Giao diện game đơn giản, thân thiện và dễ sử dụng.
-

-
- Lưu trữ tiến trình chơi game của người chơi một cách nhanh chóng.

***Các vấn đề chưa được giải quyết**

- Hình ảnh chưa được tối ưu. Hình ảnh một số đồ vật còn vỡ.
- Chuyển màn chưa được tối ưu, chưa có hiệu ứng chuyển màn.
- Chưa tối ưu hóa ứng dụng game một cách toàn diện. Một số lỗi kỹ thuật còn phát sinh.

***Hướng phát triển**

- Tiếp tục tìm hiểu để hoàn thiện các chức năng và tài nguyên còn thiếu sót.
 - Thêm hệ thống các level và nâng cấp player.
 - Cải thiện đồ họa và âm thanh bằng công nghệ Ray Tracing và âm thanh vòm
 - Nâng cấp ứng dụng để có thể kết nối với nhiều thiết bị chơi game và chạy trên nhiều nền tảng.
 - Tích hợp các thư viện bên thứ ba của các nền tảng trực tuyến như Steam, GOG,... để sử dụng các tính năng hỗ trợ và thương mại ứng dụng trên các nền tảng đó.
-

TÀI LIỆU THAM KHẢO

- [1] <https://assetstore.unity.com/>
 - [2] <https://docs.unity.com/>
 - [3] <https://learn.unity.com/>
 - [4] James Robertson(2022), Unity Game Development: From Zero to Hero, NXB Packt Publishing.
 - [5] Mark Deloura(2021), Unity 2D Game Development: Learn to Create Your Own 2D Games, NXB O'Reilly Media.
 - [6] Ebook Game programming with Unity and C#: A complete beginner's guide - Casey Hardman - Thư viện điện tử Đại học công nghiệp hà nội.
 - [7] Advanced AI in Unity [Tutorial] - Physics, Pathfinding, Editor Adjustments - YouTube.
 - [8] Easy Enemy Health Bars in Unity – YouTube.
-