ĐẠI HỌC BÁCH KHOA HÀ NỘI

KHOA TOÁN - TIN

-----📖-----

LẬP TRÌNH GAME VỚI UNITY ENGINE VÀ NGÔN NGỮ C#

ĐỒ ÁN II

Chuyên ngành: TOÁN TIN

Giảng viên hướng dẫn: **TS. Đoàn Duy Trung**

Sinh viên thực hiện: **Nguyễn Như Cường**

Mã sinh viên: **20200076**

HÀ NỘI – 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

1. **Mục tiêu và nội dụng của đồ án:**

2. **Kết quả đạt được:**

3. **Ý thức làm việc của sinh viên:**

Hà Nội, ngày tháng năm 2024

Giảng viên hướng dẫn

(*Ký và ghi rõ họ tên*)

MỤC LỤC

Danh sách hình vẽ

MỞ ĐẦU

Trong thời đại kỹ thuật số và sự phát triển công nghệ, ngành công nghiệp game đang trở thành một trong những lĩnh vực phát triển nhanh nhất trên toàn cầu. Người chơi ngày càng đòi hỏi trải nghiệm game tốt hơn, sự sáng tạo và tính giải trí cao. Trong số hàng ngàn trò chơi đứng đầu về doanh thu trên các nền tảng phân phối game, souls-like là một thể loại phổ biến và thu hút sự quan tâm của đông đảo người chơi.

Thể loại "soul-like" trong game là một thể loại được lấy cảm hứng từ trò chơi "Demon's Souls" và "Dark Souls" của FromSoftware. Những trò chơi thuộc thể loại này chủ yếu tập trung vào những trải nghiệm khó khăn, hành động chặt chẽ và hệ thống chiến đấu đòi hỏi kỹ năng.

Do đó em đã chọn đề tài "Lập trình game với Unity engine và ngôn ngữ C#" để đáp ứng nhu cầu của thị trường và người chơi hiện nay. Đề tài này sẽ giúp em chuẩn bị và phát triển các kỹ năng cần thiết, áp dụng kiến thức đã học vào quy trình phát triển một trò chơi dòng souls-like.

Game có tên là "**Da Sun Kids**", hứa hẹn cung cấp một trải nghiệm thú vị, dễ tiếp cận và không mấy phần thư giãn cho người chơi đúng như phong cách của dòng game souls-like.

Tuy nhiên, với những kiến thức, kỹ năng còn nhiều hạn chế nên không tránh được việc đề tài còn nhiều thiếu sót. Em rất mong nhận được ý kiến đóng góp và nhận xét của thầy để em được bổ sung, nâng cao kiến thức của mình.

Em xin chân thành cảm ơn !

Chương 1. Tổng quan

1. Đặt vấn đề

1.1. Tình trạng hiện tại của ngành công nghiệp game

Kể từ khi phát hành Pong và máy chơi game gia đình dành cho người tiêu dùng vào đầu những năm 1970, ngành công nghiệp trò chơi điện tử đã đi được một chặng đường dài.

Với các trò chơi và nền tảng trò chơi hiện đại cung cấp đồ họa chân thực, mô phỏng thực tế và khả năng kết nối trực tuyến với hàng triệu người chơi khác, chơi trò chơi điện tử không chỉ là sở thích thời thơ ấu của nhiều người mà còn phát triển thành một phong cách sống của mọi người, mọi lứa tuổi.

Thị trường toàn cầu cho trò chơi điện tử rất lớn. Vào năm 2021, toàn bộ ngành công nghiệp trò chơi điện tử đã mở rộng với tổng doanh thu là 180,3 tỷ USD, tăng 1,4% so với năm trước khi ngành này trải qua sự tăng trưởng vượt bậc do đại dịch toàn cầu gây ra. Trò chơi điện tử có thể đi đến đâu từ đây? Nếu ngành này tuân theo mô hình tăng trưởng lũy tiến nhất quán, một số chuyên gia dự đoán doanh thu thị trường toàn cầu sẽ đạt 510 tỷ USD vào năm 2031.[1]

Do sự tăng trưởng tự nhiên của thị trường, khả năng tiếp cận Internet ngày càng tăng, sự phát triển của công nghệ trò chơi điện tử tiên tiến và tỷ lệ tương tác cao hơn do đại dịch COVID-19 gây ra vào năm 2020, ngành công nghiệp trò chơi điện tử hiện được coi là một trong những ngành lớn nhất ở Hoa Kỳ. quy mô thị trường trò chơi điện tử ở Mỹ vượt 85,86 tỷ USD, vượt mức cao nhất mọi thời đại trước đó vào năm 2020 là 76,15 tỷ USD.[1]

Về doanh thu hàng năm, thị trường trò chơi điện tử ở Mỹ đứng thứ hai vào năm 2020 (chỉ sau Trung Quốc), tạo ra 36,92 tỷ USD. Và mức chi tiêu của người tiêu dùng tiếp tục duy trì ở mức cao. Chỉ riêng từ tháng 4 đến tháng 6 (quý 2 năm 2021), tổng chi tiêu chung của người tiêu dùng cho trò chơi điện tử ở Hoa Kỳ đạt tổng cộng 14 tỷ USD, tăng 2% so với quý 2 năm 2020.[1]

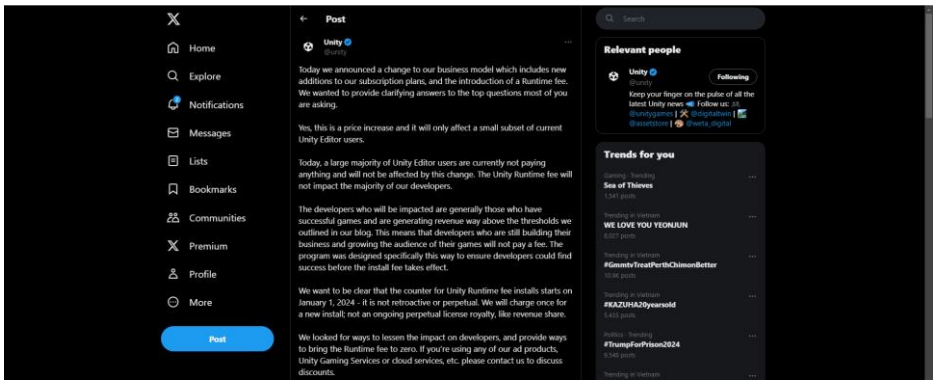
Cũng giống như nhân khẩu học của trò chơi trên thiết bị di động rất đa dạng, toàn bộ ngành công nghiệp trò chơi đang chứng kiến sự thay đổi về những gì tạo nên game thủ thời hiện đại. Vào năm 2021, phụ nữ chiếm 45% số game thủ ở Hoa Kỳ, tăng từ 41% số game thủ Hoa Kỳ được xác định là phụ nữ trong năm trước.

Khi thế hệ Millennials lớn lên với trò chơi điện tử như một phần bình thường của cuộc sống, độ tuổi của người chơi game trung bình cũng tăng lên. Dưới đây là bảng phân tích nhân khẩu học theo độ tuổi từ cuộc khảo sát năm 2021:[1]

- Dưới 18 tuổi: 20%
- 18 đến 34 tuổi: 38%
- 35-44 tuổi: 14%
- 45 đến 54 tuổi: 12%
- 55 đến 64 tuổi: 9%
- 65 tuổi trở lên: 7%

1.2. Những thách thức trong việc phát triển game trên nền tảng Unity

Một trong những thách thức mà các nhà phát triển game trên nền tảng Unity đang gặp phải là việc nhà phát hành đang có ý định áp dụng những chính sách thu phí mới cho nền tảng của họ.



Hình 1. Unity thay đổi luật vào tháng 9

Vào tháng 9 năm nay, đã có một số thách thức và vấn đề mà các nhà phát triển trò chơi sử dụng nền tảng Unity phải đối mặt. Những thách thức này xoay quanh

những thay đổi trong mô hình kinh doanh của Unity và những lo ngại về sự hỗ trợ và tin cậy. Dưới đây là những thách thức chính đã được nhấn mạnh:

- **Kế hoạch kiếm tiền mới:** Unity đã công bố kế hoạch kiếm tiền mới bao gồm Phí thời gian chạy [3]. Khoản phí này dựa trên số lần cài đặt trò chơi được xây dựng bằng công cụ Unity. Các nhà phát triển sử dụng gói chi phí thấp hơn sẽ phải trả phí khi họ đạt đến ngưỡng doanh thu và cài đặt nhất định, trong khi những nhà phát triển ở gói cấp cao hơn có các ngưỡng khác nhau [3]. Sự thay đổi này đã gây ra sự thất vọng và bối rối giữa các nhà phát triển, vì họ nhận ra rằng khoản phí mà họ phải chịu có thể rất đáng kể, đặc biệt là vì kế hoạch sẽ có hiệu lực hồi tố [3].
- **Tin cậy và Hỗ trợ:** Việc Unity xử lý kế hoạch kiếm tiền mới và các vấn đề khác đã làm dấy lên lo ngại về sự tin cậy và hỗ trợ từ nền tảng [2]. Các nhà phát triển cảm thấy rằng quyết định của Unity triển khai Phí thời gian chạy là một khoản thuế hồi tố đối với trò chơi và làm tổn hại đến niềm tin mà họ đã xây dựng trong nhiều năm [2]. , đã có những phàn nàn về việc thiếu sự hỗ trợ và liên lạc từ Unity khi các nhà phát triển gặp phải sự cố hoặc cần hỗ trợ thêm [2].
- **Tác động đến việc phát triển trò chơi:** Kế hoạch kiếm tiền mới có những tác động tiềm tàng đối với việc phát triển trò chơi. Các nhà phát triển không chắc chắn làm cách nào Unity có được dữ liệu cần thiết để tính phí và có những lo ngại về tác động đối với các gói từ thiện, bản demo và trò chơi trong hợp đồng với các dịch vụ phân phối [3]. Cũng có những lo lắng về việc những kẻ vi phạm bản quyền và có đức tin xấu có thể lạm dụng các bản cài đặt để tác động đến tài chính của các nhà phát triển [3].

Phản ứng của cộng đồng đối với chính sách định giá này là tiêu cực, nhiều nhà phát triển chỉ trích nó là không công bằng và mang tính lợi dụng, đặc biệt là đối với các hãng phim độc lập nhỏ. Một số mối lo ngại cụ thể được các nhà phát triển đưa ra bao gồm khả năng người dùng độc hại cài đặt và gỡ cài đặt trò chơi liên

tục để thu phí cho nhà phát triển, tác động đến các gói từ thiện và dịch vụ đăng ký, cũng như việc Unity đơn phương thay đổi chính sách giá của mình mà không thông báo trước cho nhà phát triển.

Cuộc tranh cãi xung quanh chính sách giá của Unity đã làm tổn hại đến danh tiếng của Unity với nhiều nhà phát triển và có khả năng một số nhà phát triển có thể chuyển sang các game engine khác như Unreal Engine hoặc Godot.[2]

2. Mục tiêu nghiên cứu

2.1. Xác định mục tiêu của dự án

Như đã nói ngay ở phần mở đầu của bài báo cáo, thể loại mà em lựa chọn để xây dựng tựa game của em là dòng souls-like. Và một tựa game có thể nói đã trở thành tấm gương, hay nói chính xác hơn là huyền thoại với thể loại này là Hollow Knight.

Hollow Knight là một tựa game hành động phiêu lưu được phát triển và xuất bản bởi Team Cherry. Game được phát hành vào năm 2017 và đã thu hút sự công nhận rộng rãi từ cộng đồng game thủ cũng như giới chuyên môn. Điều đặc biệt về Hollow Knight chính là cách nó kết hợp thành công các yếu tố như hành động, khám phá, và thử thách trong một thế giới tối tăm, đầy bí ẩn..

Hình 2. Game Hollow Knight

Trong Hollow Knight, người chơi sẽ nhập vai vào một con kiến võ sĩ và khám phá một hệ thống hang động rộng lớn, đầy nguy hiểm và đa dạng. Trò chơi tập trung vào việc đánh bại các quái vật khác nhau, thu thập sức mạnh và khám phá câu chuyện phức tạp của thế giới ngầm này. Đồ họa độc đáo và tinh tế, âm nhạc sâu sắc cùng với gameplay đòi hỏi kỹ năng đã làm nên sức hấp dẫn đặc biệt của Hollow Knight.

Tựa game này đã trở thành một nguồn cảm hứng lớn cho nhiều nhà phát triển game độc lập và đã góp phần định hình thể loại souls-like. Cách mà Hollow Knight tạo ra một thế giới đầy bí ẩn, đồng thời tạo ra thách thức khó nhằn mà người chơi phải vượt qua đã trở thành một tiêu chuẩn đáng ngưỡng mộ cho các tựa game souls-like khác.

Mục tiêu chính của dự án "**Da Sun Kids**" là xây dựng một tựa game dòng souls-like độc đáo và hấp dẫn, mang lại trải nghiệm khó nhằn, bí ẩn và thử thách cho người chơi. Chúng ta sẽ tạo ra một thế giới tối tăm, đầy nguy hiểm và đầy bí ẩn, nơi người chơi được khám phá và chiến đấu để vượt qua các thử thách đầy cam go.

Các mục tiêu cụ thể của dự án:

- **Thiết kế gameplay chất lượng cao:** Chúng tôi sẽ tạo ra một hệ thống gameplay hành động chặt chẽ, tập trung vào việc đánh bại quái vật và trải nghiệm chiến đấu thú vị. Sự phối hợp giữa các yếu tố như phản xạ, kỹ năng và chiến thuật sẽ là yếu tố quan trọng để thành công trong game.
- **Xây dựng một thế giới độc đáo:** Chúng tôi sẽ tạo ra một thế giới game đầy bí ẩn, đa dạng và sâu sắc. Các môi trường, nhân vật và câu chuyện sẽ được thiết kế kỹ lưỡng để tạo ra sự hấp dẫn và lôi cuốn cho người chơi.
- **Cung cấp thử thách và cảm giác đáng đồng cảm:** Chúng tôi sẽ tạo ra những thử thách khó nhằn và đòi hỏi sự tập trung, kiên nhẫn và kỹ năng từ người chơi. Đồng thời, chúng tôi cũng sẽ xây dựng một câu chuyện sâu sắc và nhân vật có sự phát triển để người chơi có thể tương tác và đồng cảm.

- **Tạo ra một trải nghiệm tương tác tốt:** Chúng tôi sẽ tạo ra một cộng đồng game thủ tích cực và hỗ trợ, trong đó người chơi có thể trao đổi kinh nghiệm, thách đấu với nhau và chia sẻ thành tựu. Chúng tôi cũng sẽ cung cấp cập nhật và hỗ trợ liên tục để đảm bảo rằng tựa game luôn có sự phát triển và nâng cao trải nghiệm của người chơi.

Tuy nhiên do thời gian có hạn nên ở trong phạm vi của đồ án 2 này, em chỉ xây dựng AI cho con BOSS chính của game. Cấu hình để cho con boss biết cách xử lý cho từng tình huống gặp phải trong tựa game.

2.2. Mô tả về bối cảnh trong tựa game

"**Da Sun Kids**" là một tựa game dòng souls-like độc đáo, nằm trong một thế giới tưởng tượng đầy huyền bí và sự nguy hiểm. Bối cảnh của trò chơi được đặt trong một vương quốc thần thoại bị ám ảnh bởi sự xâm lăng của quỷ dữ và thế lực tà ác.

Cốt truyện xoay quanh một tổ chức tối mật gọi là "**DSK – Da Sun Kids**", hay còn được biết đến với cái tên "**Những đứa con của mặt trời**", những chiến binh dũng cảm và kiên nhẫn đã cam kết chiến đấu chống lại sự thống trị của quỷ dữ. Nhân vật chính, người chơi, sẽ nhập vai vào một trong những thành viên của tổ chức này, có nhiệm vụ tiêu diệt và đánh bại quỷ dữ để bảo vệ con người và tái lập sự cân bằng trong vương quốc.

Bối cảnh của trò chơi " **Da Sun Kids**" bao gồm những địa điểm ma quái đa dạng như hang động tối tăm, rừng rậm đầy nguy hiểm, và thành trì hoang tàn. Người chơi sẽ khám phá và chiến đấu trong các môi trường này, đối mặt với những quái vật quỷ dữ khát máu và các thử thách đầy cam go.

Trong cuộc hành trình của mình, người chơi sẽ phải rèn luyện kỹ năng chiến đấu, thu thập trang bị mạnh mẽ, và khám phá bí ẩn về nguồn gốc của quỷ dữ. Cốt truyện sẽ tiết lộ những tình tiết ly kỳ và những nhân vật phụ đáng nhớ, tạo nên một trải nghiệm phiêu lưu sâu sắc và gây cấn trong thế giới " **Da Sun Kids**".

Với " **Da Sun Kids**", bạn sẽ được trải nghiệm một cuộc phiêu lưu đậm chất souls-like, khám phá một thế giới đầy mạo hiểm và đối đầu với những quỷ dữ tàn ác.

Chương 2. Tổng quan kỹ thuật

1. Nền tảng phần mềm và ngôn ngữ lập trình

1.1. Tổng quan về Unity Engine

1.1.1. Unity là gì?

Unity là một công cụ phát triển game đa nền tảng, được tạo ra bởi Unity Technologies và giới thiệu lần đầu tiên vào năm 2005 tại sự kiện Apple's WorldWide Developer Conference. Được biết đến với tư cách là một "cross-platform engine," Unity cho phép nhà phát triển tạo ra các trò chơi và ứng dụng chất lượng cao có thể chạy trên nhiều nền tảng khác nhau như PC, consoles, thiết bị di động (smartphone và tablet), và trên các trình duyệt web.

Khả năng của Unity không chỉ giới hạn trong việc tạo ra đồ họa 2D và 3D mà còn bao gồm hiệu ứng đẹp mắt và âm thanh sống động. Sự linh hoạt của công cụ này là một điểm mạnh lớn, cho phép nhà phát triển thực hiện các ý tưởng sáng tạo và đa dạng trong việc xây dựng trải nghiệm trò chơi.

Với Unity, nhà phát triển có thể xây dựng trò chơi và ứng dụng trực quan, đa nền tảng, từ máy tính để bàn, điện thoại di động, máy chơi game, cho đến các thiết bị AR/VR như Oculus Rift, HTC Vive, Microsoft HoloLens và nhiều nền tảng khác. Unity cũng hỗ trợ việc xuất bản sản phẩm lên các nền tảng như Windows, macOS, iOS, Android và nền tảng điện tử tiêu dùng khác.

Đặc biệt, Unity đã thu hút sự chú ý của cả cộng đồng game thủ và những người phát triển, nhờ khả năng tạo ra trò chơi chất lượng cao với đồ họa và âm thanh nổi bật. Sự phát triển liên tục đã đưa Unity qua nhiều phiên bản, và phiên bản mới nhất, 2023.2.3f1, tiếp tục mang đến nhiều tính năng nổi trội, giúp nâng cao trải nghiệm phát triển game và ứng dụng. Điều này đã củng cố vị thế của Unity là một trong những công cụ phát triển game phổ biến và mạnh mẽ nhất trên thị trường ngày nay.

1.1.2. Ưu điểm của Unity

Unity3D cung cấp một loạt chức năng cốt lõi đa dạng để hỗ trợ quá trình phát triển game. Tính năng này bao gồm công cụ dựng hình (rendering) cho cả hình ảnh 2D và 3D, tính toán và phát hiện va chạm, quản lý âm thanh, xử lý mã nguồn, tạo hiệu ứng hình ảnh động, và thậm chí cả trí tuệ nhân tạo và phân luồng. Unity3D cung cấp mọi công cụ mà nhà phát triển cần để xây dựng trò chơi của mình.

Một điểm độc đáo của Unity3D là khả năng hỗ trợ đa nền tảng. Tính năng này cho phép nó hỗ trợ hầu hết các nền tảng hiện tại, từ PlayStation, Xbox, iOS, Android, Windows, Linux, OS X đến trình duyệt web. Điều này giúp tối ưu hóa quá trình phát triển và chuyển đổi giữa các nền tảng một cách linh hoạt, làm cho Unity3D trở thành một giải pháp lý tưởng cho việc phát triển trò chơi đa nền tảng, nơi người chơi có thể tham gia trên nhiều hệ điều hành và thiết bị khác nhau như Web, PC, Mobile, Tablet,...

Đặc biệt, Unity3D được thiết kế để dễ sử dụng, với môi trường phát triển tích hợp cung cấp hệ thống toàn diện cho lập trình viên. Từ soạn thảo mã nguồn đến xây dựng công cụ tự động hóa và trình sửa lỗi, Unity hỗ trợ các lập trình viên mỗi bước trong quá trình phát triển. Điều này làm cho nó trở nên hấp dẫn cho cả lập trình viên mới và các studio chuyên nghiệp. Bên cạnh đó, với cộng đồng lớn trên các diễn đàn công nghệ, người dùng có thể dễ dàng tìm kiếm sự hỗ trợ và chia sẻ kinh nghiệm.

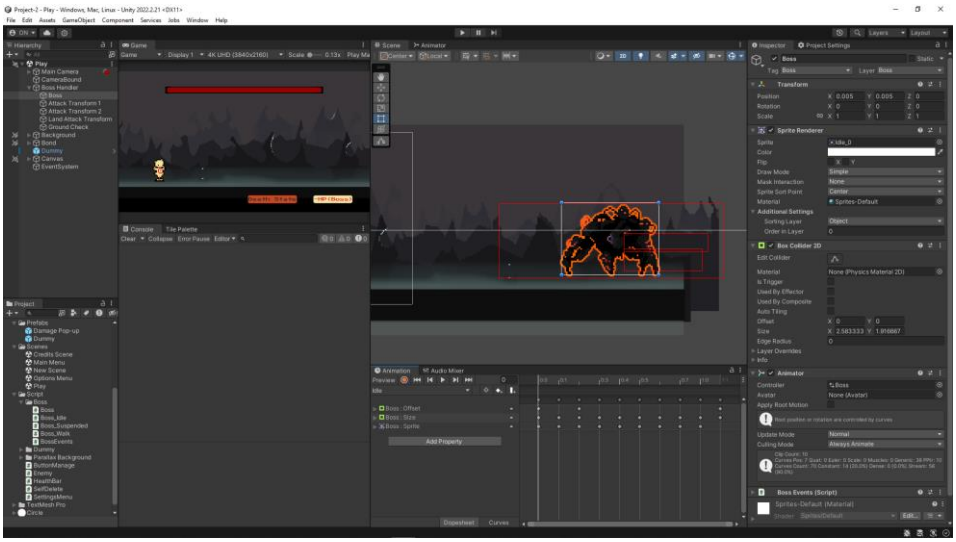
Tính kinh tế cao cũng là một ưu điểm của Unity3D. Unity Technologies cung cấp bản miễn phí cho người dùng cá nhân và doanh nghiệp có doanh thu dưới 100,000 USD/năm. Điều này tạo điều kiện thuận lợi cho các nhà phát triển nhỏ và startup bắt đầu phát triển game mà không gặp khó khăn về tài chính. Bản Pro với chi phí 1,500 USD/năm cũng được coi là một sự đầu tư hợp lý cho những tiện ích mà nó mang lại.

Ngoài ra, Unity còn cung cấp cho các lập trình viên kho asset lớn và đa dạng. Gavin Price, Giám đốc Studio Playtonic, đã ca ngợi cửa hàng asset của Unity với

việc cung cấp một kho tài nguyên bổ sung đáng tin cậy. Cửa hàng này có một dự trữ tốt về các công cụ bổ sung, được ghi chú đầy đủ và được hỗ trợ bởi một cộng đồng nhà phát triển tuyệt vời. Điều này giúp nhà phát triển tìm kiếm và sử dụng các tài nguyên phù hợp để làm việc trên dự án của mình.

Theo số liệu, khoảng 1,5 triệu nhà phát triển truy cập vào Cửa hàng nội dung của Unity mỗi tháng, với hơn 56.000 gói asset có sẵn. Điều này đại diện cho hơn một triệu nội dung cá nhân có sẵn để người sáng tạo sử dụng. Điều này cho thấy cộng đồng Unity đang đóng góp rất nhiều tài nguyên để hỗ trợ nhà phát triển trong quá trình tạo ra trò chơi và ứng dụng.

1.1.3. Các thành phần của Unity Editor



Hình 3. Tổng quan một project trên Uniy Editor

❖ **Cửa sổ scene**

Cửa sổ Scenes (Cảnh) là nơi hiển thị và quản lý các cảnh trong dự án của bạn. Mỗi cảnh đại diện cho một phần của trò chơi hoặc một màn chơi cụ thể.

Cửa sổ Scenes cung cấp các chức năng và tác vụ sau:

- **Hiển thị cảnh:** Cửa sổ Scenes hiển thị nội dung của cảnh hiện tại. Bạn có thể xem các đối tượng, sắp xếp chúng và thao tác với chúng.

- **Tạo cảnh mới:** Bằng cách nhấp chuột phải vào cửa sổ Scenes và chọn "Create" hoặc "Create Empty", bạn có thể tạo ra một cảnh mới cho trò chơi của mình.
- **Thay đổi cảnh hiện tại:** Bạn có thể chuyển đổi giữa các cảnh trong dự án bằng cách nhấp chuột vào tên cảnh trong cửa sổ Scenes. Điều này cho phép bạn xem và chỉnh sửa các cảnh khác nhau trong dự án.
- **Xóa và sắp xếp cảnh:** Bạn có thể xóa cảnh khỏi dự án bằng cách nhấp chuột phải vào tên cảnh và chọn "Delete". Bạn cũng có thể sắp xếp thứ tự các cảnh bằng cách kéo và thả chúng trong cửa sổ Scenes.
- **Tìm kiếm cảnh:** Cửa sổ Scenes cung cấp khung tìm kiếm để bạn tìm kiếm các cảnh trong dự án dựa trên tên hoặc các thuộc tính khác.
- **Thiết lập cảnh mặc định:** Bạn có thể đặt một cảnh làm cảnh mặc định, tức là cảnh sẽ được mở khi bạn chạy trò chơi. Điều này được thực hiện bằng cách nhấp chuột phải vào cảnh và chọn "Set as Scene View".

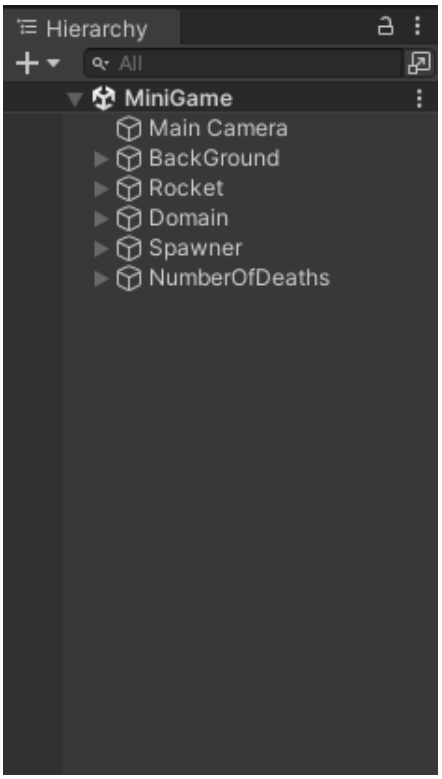
Cửa sổ Scenes giúp bạn quản lý và làm việc với các cảnh trong dự án Unity của bạn. Bạn có thể tạo, xóa, sắp xếp và chuyển đổi giữa các cảnh, cho phép bạn tổ chức và chỉnh sửa nội dung của trò chơi một cách linh hoạt.

❖ **Cửa sổ Hierarchy**

Cửa sổ Hierarchy hiển thị các đối tượng theo cấu trúc cây, cho thấy mối quan hệ cha-con giữa chúng. Mỗi đối tượng trong cảnh sẽ có một mục hiển thị trong cửa sổ Hierarchy, và bạn có thể xem và chỉnh sửa thuộc tính của từng đối tượng qua cửa sổ Inspector khi chọn chúng trong Hierarchy.

Các thao tác thực hiện với cửa sổ Hierarchy:

- **Thêm đối tượng mới:** Bằng cách nhấp chuột phải vào cửa sổ Hierarchy và chọn "Create Empty" hoặc các loại đối tượng khác để thêm đối tượng mới vào cảnh.
- **Xóa và sắp xếp đối tượng:** Bạn có thể xóa đối tượng bằng cách nhấp chuột phải vào đối tượng và chọn "Delete" hoặc nhấn phím Delete trên bàn phím. Bạn cũng có thể sắp xếp thứ tự các đối tượng bằng cách kéo và thả chúng trong cửa sổ Hierarchy.
- **Tìm kiếm đối tượng:** Cửa sổ Hierarchy cung cấp khung tìm kiếm để bạn tìm kiếm các đối tượng trong cảnh dựa trên tên hoặc các thuộc tính khác.
- **Đặt lớp và tag:** Bạn có thể đặt lớp (Layer) và tag cho các đối tượng trong Hierarchy, giúp phân loại và xử lý chúng trong quá trình lập trình.



Hình 4. Cửa sổ Hierarchy

Cửa sổ Hierarchy là một công cụ hữu ích để quản lý cấu trúc và hiển thị các đối tượng trong cảnh, giúp bạn dễ dàng điều chỉnh và làm việc với các thành phần khác nhau của trò chơi.

❖ **Cửa sổ Game**

Trong Unity Editor, cửa sổ Game (Game View) là cửa sổ hiển thị trực tiếp kết quả của trò chơi trong quá trình phát triển. Nó cho phép bạn xem và tương tác với trò chơi trong thời gian thực.

Cửa sổ Game có các chức năng và tính năng sau:

- Hiển thị trực tiếp trò chơi: Cửa sổ Game hiển thị trò chơi của bạn như nó sẽ xuất hiện trong quá trình chạy. Bạn có thể thấy các đối tượng, hiệu ứng, cảnh quan, và tất cả các yếu tố khác của trò chơi trong thời gian thực.
- Kiểm tra hiệu năng: Bằng cách chạy trò chơi trong cửa sổ Game, bạn có thể kiểm tra hiệu năng của trò chơi, như tốc độ khung hình (FPS), tài nguyên sử dụng, và các chỉ số hiệu suất khác. Điều này giúp bạn đánh giá và tối ưu hóa trò chơi của mình.
- Tương tác với trò chơi: Bạn có thể tương tác với trò chơi trong cửa sổ Game, giống như người chơi thực tế sẽ làm. Bạn có thể di chuyển, nhảy, bắn, hoặc thực hiện các hành động khác để kiểm tra tính năng và trải nghiệm của trò chơi.
- Điều chỉnh kích thước cửa sổ: Bạn có thể điều chỉnh kích thước cửa sổ Game để phù hợp với kích thước màn hình hoặc thiết bị mà bạn muốn đích đến. Điều này giúp bạn kiểm tra và đảm bảo rằng trò chơi của bạn hoạt động và trông tốt trên các nền tảng khác nhau.
- Chế độ xem khác nhau: Cửa sổ Game cung cấp nhiều chế độ xem khác nhau, bao gồm chế độ xem duy nhất, chế độ xem phân tách (Split View) để so sánh hai cảnh khác nhau, và chế độ xem VR để kiểm tra trò chơi ảo thực.

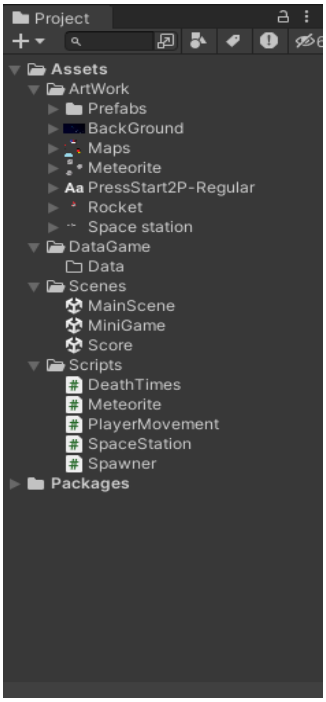
Cửa sổ Game là công cụ quan trọng để kiểm tra và phát triển trò chơi trong quá trình làm việc với Unity. Nó cho phép bạn xem trực tiếp trò chơi, tương tác với nó và kiểm tra hiệu suất, giúp bạn tạo ra một trải nghiệm trò chơi tốt nhất cho người chơi.

❖ **Cửa sổ Project**

Cửa sổ Project (Project Window) trong Unity Editor là nơi bạn quản lý tất cả các tài nguyên và tệp tin trong dự án của mình. Nó cung cấp một cái nhìn toàn diện về cấu trúc và nội dung của dự án, cho phép bạn tổ chức, tạo, xóa, di chuyển và quản lý các tệp tin và thư mục.

Dưới đây là các thành phần và tính năng chính của cửa sổ Project:

- **Hiển thị tệp tin và thư mục:** Cửa sổ Project hiển thị tất cả các tệp tin và thư mục có trong dự án của bạn. Bạn có thể tạo mới, xóa, di chuyển và sắp xếp các tệp tin và thư mục theo ý muốn.
- **Tìm kiếm và lọc:** Bạn có thể sử dụng chức năng tìm kiếm để nhanh chóng tìm kiếm các tệp tin và thư mục theo tên hoặc loại. Ngoài ra, bạn có thể áp dụng bộ lọc để hiển thị chỉ các tệp tin cụ thể hoặc loại tệp tin.
- **Xem trước tệp tin:** Khi bạn chọn một tệp tin trong cửa sổ Project, bạn có thể xem trước nhanh nội dung của nó trong cửa sổ xem trước. Điều này rất hữu ích khi bạn muốn xem trước hình ảnh, âm thanh, video hoặc bất kỳ tệp tin nào khác trong dự án của mình.



Hình 5. Cửa sổ Project

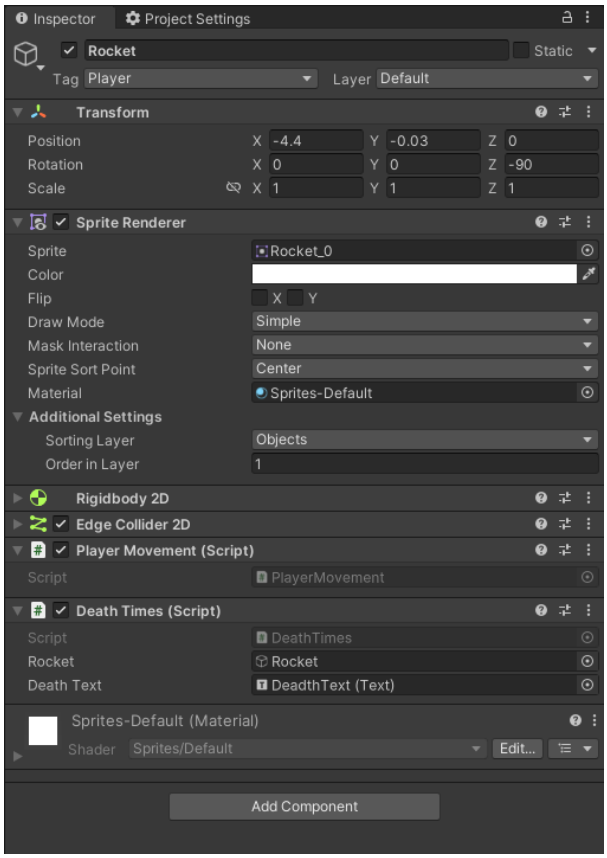
- **Quản lý tài nguyên:** Cửa sổ Project cho phép bạn quản lý tất cả các tài nguyên của dự án, bao gồm hình ảnh, âm thanh, video, tệp tin mã nguồn và các tài nguyên khác. Bạn có thể kéo và thả các tài nguyên vào các đối tượng trong cảnh hoặc sử dụng chúng trong các kịch bản và mã nguồn.
- **Giao tiếp với các cửa sổ khác:** Cửa sổ Project liên kết với các cửa sổ khác trong Unity, cho phép bạn kéo và thả tệp tin và thư mục vào cửa sổ Scene để tạo đối tượng, hoặc kéo và thả vào cửa sổ Inspector để điều chỉnh thuộc tính. Nó cung cấp một giao diện linh hoạt để làm việc với các thành phần khác trong Unity.

Cửa sổ Project cũng cho phép thêm các tài nguyên vào Favorites để truy cập nhanh vào những tài nguyên thường sử dụng. Việc tổ chức tài nguyên trong cửa

sổ Project giúp game developer dễ dàng tìm kiếm và quản lý tài liệu và tập tin trong quá trình phát triển trò chơi.

❖ Cửa sổ Inspector

Inspector trong Unity là một trong những cửa sổ quan trọng nhất trong quá trình phát triển trò chơi. Nó cung cấp thông tin và kiểm soát chi tiết về các đối tượng và thành phần trong cảnh hiện tại.



Hình 6. Cửa sổ Inspector

Dưới đây là các đặc điểm và chức năng chính của cửa sổ Inspector:

- **Hiển thị thông tin chi tiết:** Khi bạn chọn một đối tượng hoặc thành phần trong cảnh, cửa sổ Inspector hiển thị các thuộc tính và thông tin chi tiết về nó. Bạn có thể thấy và chỉnh sửa các thuộc tính như tên, vị trí, quy mô, hình dạng, vật liệu, ánh sáng, âm thanh và các thông số khác liên quan.

- **Thay đổi giá trị thuộc tính:** Bằng cách tương tác với cửa sổ Inspector, bạn có thể thay đổi giá trị của các thuộc tính của đối tượng hoặc thành phần. Bạn có thể điều chỉnh các giá trị số, chọn từ các danh sách thả xuống, hoặc kéo và thả các giá trị trong các thanh trượt.
- **Điều khiển và tương tác:** Cửa sổ Inspector cung cấp các công cụ và giao diện để điều khiển và tương tác với các đối tượng trong cảnh. Bạn có thể kích hoạt và vô hiệu hóa các thành phần, thiết lập các sự kiện, áp dụng các hiệu ứng và thay đổi các thông số để tùy chỉnh hành vi và ngoại hình của đối tượng.
- **Thực hiện tùy chỉnh mã nguồn:** Nếu bạn là một lập trình viên, cửa sổ Inspector cho phép bạn truy cập và chỉnh sửa mã nguồn của các lớp và kịch bản liên quan đến đối tượng hoặc thành phần. Bạn có thể mở tập tin mã nguồn, chỉnh sửa mã, gỡ lỗi và áp dụng các thay đổi mà không cần rời khỏi cửa sổ Inspector.

Cửa sổ Inspector là nơi bạn tìm thấy thông tin chi tiết về các đối tượng và thành phần trong dự án Unity của bạn. Nó cho phép bạn điều chỉnh và tùy chỉnh các thuộc tính, điều khiển hành vi và tương tác của các đối tượng trong trò chơi.

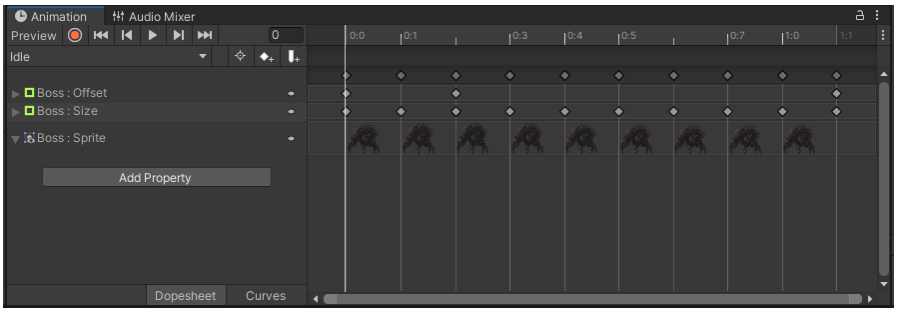
❖ **Cửa số Animation**

Một số điểm quan trọng về cửa sổ Animation:

- **Hiển thị các hoạt cảnh:** Cửa sổ Animation hiển thị các hoạt cảnh (animation clips) có sẵn trong dự án của bạn. Bạn có thể chọn hoạt cảnh để xem và chỉnh sửa.
- **Tạo keyframe và chỉnh sửa động cơ:** Bằng cách sử dụng cửa sổ Animation, bạn có thể tạo keyframe để ghi lại các giá trị thuộc tính của một đối tượng trong một khung thời gian cụ thể. Bạn có thể điều chỉnh các giá trị thuộc tính để tạo chuyển động và hiệu ứng mong muốn cho đối tượng.
- **Chỉnh sửa đường dẫn chuyển động:** Cửa sổ Animation cho phép bạn chỉnh sửa đường dẫn chuyển động của một đối tượng trong quá trình hoạt

cảnh. Bạn có thể kéo và thả các keyframe, điều chỉnh độ cong và tốc độ của đường dẫn chuyển động để tạo hiệu ứng chuyển động phù hợp.

- **Tạo blend và transition:** Bạn có thể tạo các trạng thái chuyển tiếp (transition) giữa các hoạt cảnh khác nhau và điều chỉnh sự trộn (blend) giữa chúng. Điều này cho phép bạn tạo ra các chuyển động mượt mà và tự nhiên khi các hoạt cảnh được kích hoạt hoặc chuyển đổi.
- **Gắn kết và xử lý sự kiện:** Cửa sổ Animation cho phép bạn gắn kết các sự kiện (events) với các keyframe, cho phép bạn kích hoạt các hành động hoặc chức năng khác trong trò chơi tại các điểm thời gian cụ thể.



Hình 7. Cửa sổ Animation

Cửa sổ Animation là một công cụ mạnh mẽ để tạo và chỉnh sửa các hoạt cảnh trong Unity. Nó cho phép bạn tạo ra các chuyển động, hiệu ứng và trạng thái chuyển tiếp cho các đối tượng trong trò chơi của bạn, đóng vai trò quan trọng trong việc tạo ra trải nghiệm trò chơi đa dạng và hấp dẫn.

1.1.4. Các khái niệm cơ bản trong Unity

❖ ***GameObject (Đối tượng trò chơi)***

Game Object là thành phần cơ bản và quan trọng nhất trong Unity. Nó đại diện cho mọi đối tượng trong trò chơi và có chức năng là một khung chứa để gắn kết các thành phần khác như Renderer, Collider, Script và nhiều thành phần khác. Game Object có cấu trúc phân cấp linh hoạt và có thể chứa Game Object con. Mỗi Game Object có thuộc tính và thông số riêng, và bạn có thể tương tác và điều khiển chúng thông qua việc gắn kết script và các thành phần khác. Game Object

là nền tảng để xây dựng trò chơi trong Unity và đóng vai trò quan trọng trong việc tạo ra các đối tượng và tương tác trong trò chơi.

❖ ***Component (Thành phần)***

Component (Thành phần) là một phần tử cơ bản và quan trọng trong Unity. Nó được gắn kết vào Game Object để xác định và điều khiển hành vi, tính năng và hiệu ứng của đối tượng trong trò chơi. Có một loạt các thành phần sẵn có trong Unity, bao gồm Renderer (bộ xử lý đồ họa), Collider (bộ xử lý va chạm), Rigidbody (điều khiển vật lý), Script (kịch bản tự viết), Audio Source (nguồn âm thanh), Particle System (hệ thống hạt), và nhiều thành phần khác.

Mỗi thành phần có chức năng riêng và đóng vai trò quan trọng trong việc tạo ra các đối tượng và tương tác trong trò chơi. Các thành phần có thể tương tác với nhau, kết hợp và mở rộng để tạo ra các trò chơi đa dạng và phong phú. Bạn cũng có thể tạo ra các thành phần tùy chỉnh bằng cách viết Script của riêng mình, mở rộng khả năng và chức năng của Unity để đáp ứng nhu cầu cụ thể của trò chơi. Component là một yếu tố quan trọng để xây dựng trò chơi trong Unity và mang đến sự linh hoạt và đa dạng trong việc quản lý và điều khiển các thành phần của đối tượng trong trò chơi.

❖ ***Sprite***

Sprite trong Unity là một hình ảnh 2D được sử dụng để hiển thị đối tượng trong trò chơi. Bạn có thể tạo và sử dụng sprite để biểu diễn nhân vật, vật phẩm và các yếu tố khác trong môi trường trò chơi 2D. Unity cung cấp các công cụ để nhập, chỉnh sửa và hiển thị sprite trong trò chơi của bạn.

❖ ***Prefabs (Mẫu)***

Prefab (Mẫu) trong Unity là một đối tượng có thể được tạo ra và sử dụng lại trong nhiều cảnh khác nhau. Nó giúp tiết kiệm thời gian và duy trì tính nhất quán trong phát triển trò chơi. Bằng cách tạo Prefab, bạn có thể tái sử dụng các đối

tượng và áp dụng các thay đổi cho tất cả các phiên bản của chúng. Prefab là một công cụ hữu ích để tạo ra các đối tượng và quản lý chúng trong trò chơi của bạn.

❖ ***Script (Kịch bản)***

Script là một thành phần quan trọng trong Unity, cho phép bạn viết mã lệnh để điều khiển hành vi và tương tác của đối tượng trong trò chơi. Bằng cách viết script, bạn có thể thực hiện các chức năng, xử lý sự kiện, và tạo ra các hành động đáp ứng trong trò chơi.

Script được viết bằng ngôn ngữ lập trình như C# hoặc JavaScript, và nó được gắn kết vào các Game Object trong Unity. Mỗi script có thể được gắn kết với một đối tượng cụ thể và điều khiển hành vi của nó, như di chuyển, tương tác, hoặc xử lý logic trò chơi. Bằng cách sử dụng các hàm và biến trong script, bạn có thể thực hiện các tác vụ phức tạp như kiểm tra va chạm, tạo đối tượng, điều khiển hoạt động của đối tượng, và nhiều hơn nữa.

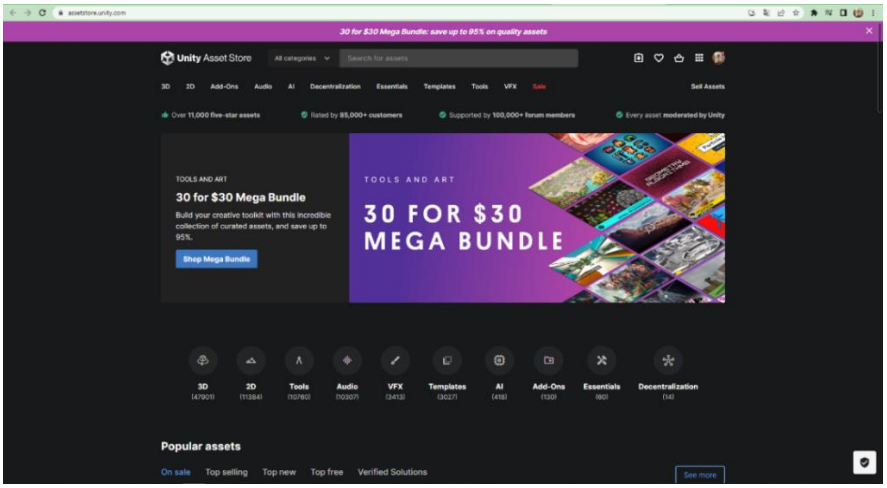
❖ ***Scenes (Cảnh)***

Scene (Cảnh) trong Unity là một màn chơi hoặc phần trong trò chơi, gồm các đối tượng và tài nguyên tương ứng. Cảnh giúp tổ chức và quản lý các đối tượng trong trò chơi, cho phép bạn thêm, xóa và chỉnh sửa chúng. Bằng cách chuyển đổi giữa các cảnh, bạn có thể tạo ra các trạng thái khác nhau và quản lý luồng chạy của trò chơi một cách dễ dàng. Quản lý cảnh là một phần quan trọng trong phát triển trò chơi Unity.

❖ ***Assets (Tài nguyên)***

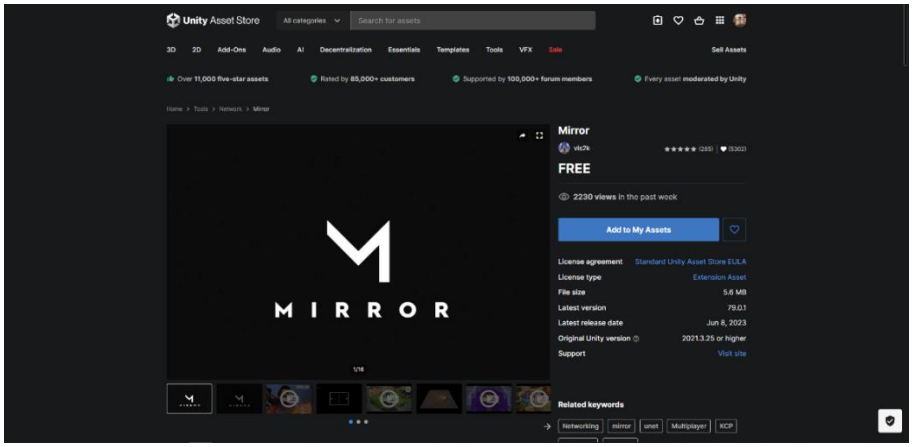
Trong Unity, thành phần Assets đóng vai trò quan trọng trong việc tạo, quản lý và sử dụng các tài nguyên (resources) trong dự án. Thành phần Assets bao gồm các file và thư mục chứa thông tin và dữ liệu được sử dụng trong trò chơi, ví dụ như hình ảnh, âm thanh, mô hình 3D, cấu hình, kịch bản (script), v.v.

Unity hỗ trợ nhiều định dạng tài nguyên khác nhau, bao gồm PNG, JPEG, WAV, MP3, FBX, OBJ và nhiều hơn nữa. Bạn có thể nhập khẩu tài nguyên từ các công cụ ngoại vi, tạo tài nguyên mới trực tiếp trong Unity, hoặc tải xuống từ Unity Asset Store theo đường link: https://assetstore.unity.com/



Hình 8. Unity Asset Store

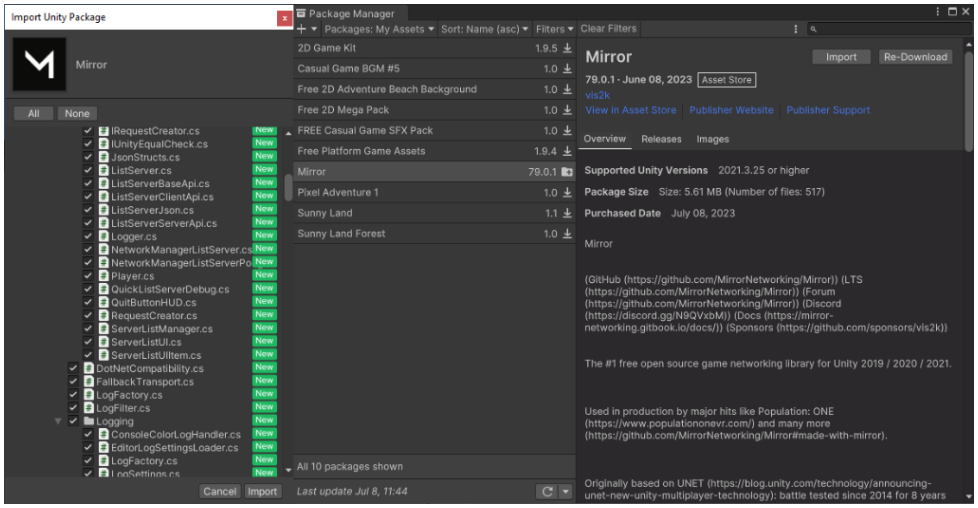
Để trực quan hơn thì hãy cùng nhau tải một asset có sẵn trên Unity Asset Store. Ví dụ ở đây ta muốn thêm assets "MIRROR" vào unity:



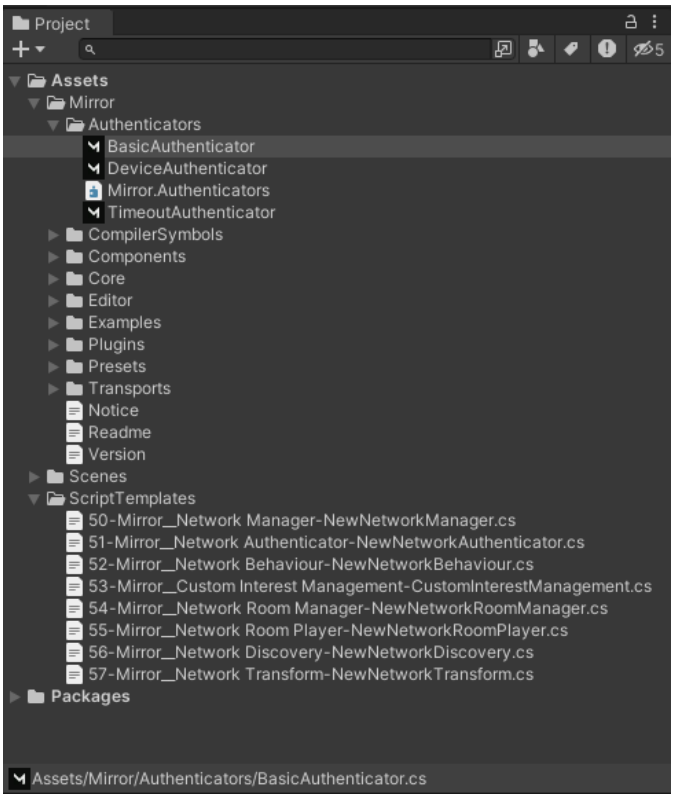
Hình 9. Một asset bất kỳ trên Unity Asset Store

Đầu tiên chúng ta nhấn vào "Add to My Assets" để tải assets trên về Unity. Ở ví dụ đây là một assets free tuy nhiên có rất nhiều những assets bắt buộc phải trả một khoản phí để được sử dụng.

Sau khi đã tải hoàn thành chúng ta mở Unity lên vào phần Package Manager, ở mục Package chọn “My Assets”, lúc này sẽ hiện lên danh sách các assets mà bạn tải về từ Unity Asset Store. Bước tiếp theo chỉ cần nhấn “Download” và sau đó import vào unity là xong.



Hình 10. Install assets từ Unity Asset Store



Hình 11. Import assets từ Unity Asset Store

❖ ***Transform (Biến đổi)***

Transform (Biến đổi) là một thành phần cơ bản trong Unity được gắn kết với mỗi đối tượng (GameObject). Nó đại diện cho vị trí, quy mô và đóng vai trò quan trọng trong việc định vị, xoay và tỷ lệ các đối tượng trong không gian 3D hoặc 2D.

Thông qua thành phần Transform, bạn có thể thay đổi vị trí của đối tượng trong không gian bằng cách điều chỉnh các giá trị x, y, z của nó. Quy mô cho phép bạn thay đổi kích thước của đối tượng, còn xoay cho phép bạn xoay đối tượng theo các trục x, y, z.

Ngoài ra, Transform cũng cho phép bạn truy cập và thay đổi các thông số khác như vị trí toàn cục (global position), vị trí cục bộ (local position), quy mô toàn cục (global scale), quy mô cục bộ (local scale), góc xoay toàn cục (global rotation), và góc xoay cục bộ (local rotation).

Để trực quan về dễ hiểu hơn, xét một ví dụ về công thức toán cho việc chuyển đổi giữa vị trí toàn cầu (global position) và vị trí cục bộ (local position) trong Unity có thể được mô tả như sau:

- **Từ vị trí cục bộ (local position) đến vị trí toàn cầu (global position):**
    + Để chuyển từ vị trí cục bộ sang vị trí toàn cầu, ta cần xác định vị trí toàn cầu của đối tượng trong không gian.
    + Ta có công thức: ***global position = parent's global position + local position.***
    + Điều này có nghĩa là vị trí toàn cầu của đối tượng bằng vị trí toàn cầu của đối tượng cha cộng với vị trí cục bộ của đối tượng hiện tại.
- **Từ vị trí toàn cầu (global position) đến vị trí cục bộ (local position):**
    + Để chuyển từ vị trí toàn cầu sang vị trí cục bộ, ta cần xác định vị trí cục bộ của đối tượng liên quan đến đối tượng cha của nó.

+ Ta có công thức: ***local position = global position - parent's global position.***
+ Điều này có nghĩa là vị trí cục bộ của đối tượng bằng vị trí toàn cầu của đối tượng hiện tại trừ đi vị trí toàn cầu của đối tượng cha.

Lưu ý rằng công thức trên chỉ áp dụng khi đối tượng có một đối tượng cha duy nhất. Trong trường hợp đối tượng có nhiều đối tượng cha, công thức này cần được áp dụng lần lượt từ đối tượng cha gốc cho đến đối tượng cha trực tiếp của đối tượng hiện tại.

1.2. Ngôn ngữ lập trình sử dụng trong phát triển game trên Unity

Ngôn ngữ lập trình chủ yếu được sử dụng trong phát triển game trên Unity là C#. Unity hỗ trợ nhiều ngôn ngữ khác nhau như JavaScript và Boo, nhưng trong thời gian gần đây, C# đã trở thành ngôn ngữ phổ biến và chủ đạo cho phát triển game trên nền tảng này.

Ngôn ngữ C# (C-Sharp) là một ngôn ngữ lập trình hiện đại và mạnh mẽ, được phát triển bởi Microsoft. C# được ra đời vào năm 2000 và từ đó đã trở thành một trong những ngôn ngữ phổ biến nhất trong lĩnh vực phát triển ứng dụng và phần mềm.

C# được thiết kế dựa trên nền tảng của ngôn ngữ C++, nhưng nó đã thừa hưởng nhiều tính năng mới và cải tiến, giúp tăng tính hiệu quả và dễ sử dụng. Với C#, nhà phát triển có thể xây dựng các ứng dụng đa nền tảng cho Windows, macOS và Linux, cũng như ứng dụng di động cho iOS và Android.

Một trong những đặc điểm nổi bật của C# là tính hướng đối tượng mạnh mẽ, cho phép mô hình hóa các đối tượng trong thế giới thực vào code. C# cung cấp một cú pháp linh hoạt và dễ hiểu, giúp tăng khả năng tái sử dụng mã và tạo ra các ứng dụng có cấu trúc rõ ràng.

Ngoài ra, C# cũng hỗ trợ các tính năng tiên tiến như thuộc tính, sự kiện, đa luồng, xử lý ngoại lệ và LINQ (Language-Integrated Query), cho phép truy vấn

và xử lý dữ liệu dễ dàng hơn. C# cũng có hệ thống quản lý bộ nhớ tự động thông qua garbage collector, giúp giảm tải công việc của nhà phát triển trong việc quản lý bộ nhớ.

C# được tích hợp chặt chẽ với Framework .NET của Microsoft, cung cấp một tập hợp các thư viện mạnh mẽ và công cụ phát triển, giúp tăng tốc quá trình phát triển ứng dụng. Người dùng C# cũng có thể sử dụng Xamarin hoặc .NET Core để phát triển ứng dụng di động hoặc đa nền tảng.

Với sự phát triển không ngừng và sự hỗ trợ từ cộng đồng lập trình viên rộng lớn, C# đã trở thành một công cụ quan trọng cho việc phát triển các ứng dụng và hệ thống phức tạp trên nhiều nền tảng. Sự kết hợp giữa tính năng mạnh mẽ, dễ sử dụng và khả năng tích hợp với Framework .NET đã làm cho C# trở thành một ngôn ngữ lập trình không thể thiếu đối với các nhà phát triển phần mềm và ứng dụng.

2. Công cụ và tài nguyên

2.1. Các công cụ hỗ trợ phát triển game trên Unity

2.1.1. Unity Editor

Như đã nói ở trên thì Unity Editor là một môi trường làm việc chính trong quá trình phát triển game trên nền tảng Unity. Đây là nơi lập trình viên và nhà phát triển tạo, chỉnh sửa, và quản lý dự án game của họ.

Unity Editor là một môi trường làm việc đa nhiệm và linh hoạt, mang đến nhiều chức năng quan trọng cho quá trình phát triển game trên nền tảng Unity. Scene View và Game View cho phép tạo, chỉnh sửa và kiểm tra đối tượng và trải nghiệm người chơi. Hierarchy Window và Project Window giúp quản lý cấu trúc dự án và tài nguyên. Inspector Window cung cấp thông tin chi tiết và điều chỉnh thuộc tính của đối tượng, trong khi Console Window hỗ trợ debug và giải quyết lỗi. Animation Window cho phép tạo và chỉnh sửa animation một cách thuận tiện. Asset Store Tab liên kết trực tiếp đến Unity Asset Store, nơi lập trình viên có thể

tải về tài nguyên sẵn có. Build Settings giúp cấu hình và xây dựng game cho nhiều nền tảng. Profiler hỗ trợ đánh giá hiệu suất, trong khi Services Tab kết nối với các dịch vụ như quảng cáo và đám mây. Unity Editor đóng vai trò quan trọng trong việc tối ưu hóa quá trình phát triển và tạo ra các sản phẩm game chất lượng.

2.1.2. Visual Studio

Visual Studio của Microsoft là một môi trường phát triển tích hợp (IDE) được sử dụng rộng rãi cho việc viết mã nguồn trong nhiều ngôn ngữ lập trình, bao gồm C#. Trong lĩnh vực phát triển game trên Unity, Visual Studio được ưa chuộng và mạnh mẽ, với sự tối ưu hóa để tích hợp tốt với Unity, một công cụ phổ biến cho phát triển game.

Hình 12. Visual Studio 2022

Dưới đây là một số chức năng và tính năng quan trọng của Visual Studio khi sử dụng cho phát triển game trên Unity:

- **Tự động Hoàn Thành Mã và Kiểm Tra Lỗi:** Visual Studio hỗ trợ tính năng tự động hoàn thành mã, giúp lập trình viên tiết kiệm thời gian và giảm lỗi cú pháp. Hệ thống hoàn thành mã tự động sẽ gợi ý các đoạn mã dựa trên ngữ cảnh, từ đó giúp bạn tìm kiếm và sử dụng các phương pháp và biến liên quan một cách nhanh chóng. Ngoài ra, Visual Studio cũng cung cấp

hệ thống kiểm tra lỗi realtime, giúp phát hiện và sửa lỗi ngay khi bạn viết mã. Điều này tăng hiệu suất và chất lượng mã nguồn, giúp bạn phát triển game một cách nhanh chóng và ổn định.

- **Debugging và Profiling cho Mã nguồn Unity:** Visual Studio tích hợp chặt chẽ với Unity, cung cấp công cụ debugging mạnh mẽ để theo dõi và giải quyết lỗi trong mã nguồn. Bạn có thể dễ dàng tạo các điểm dừng (breakpoints), theo dõi giá trị biến, và xem thông tin gỡ lỗi chi tiết. Ngoài ra, Visual Studio cũng hỗ trợ tính năng profiling, giúp đo lường hiệu suất ứng dụng. Bằng cách phân tích các phần của mã nguồn có thể gây ra hiệu năng kém, bạn có thể xác định các điểm bottleneck và tối ưu hóa mã nguồn để đảm bảo trải nghiệm chơi game mượt mà.
- **Hỗ Trợ Mở Rộng và Tích Hợp Dễ Dàng:** Visual Studio hỗ trợ mở rộng thông qua các Extension, cho phép tích hợp các plugin và công cụ bổ sung theo nhu cầu của lập trình viên. Bằng cách sử dụng các Extension, bạn có thể tăng cường khả năng phát triển của môi trường lập trình và tùy chỉnh nó cho phù hợp với nhu cầu riêng của bạn. Ngoài ra, Visual Studio cũng tích hợp dễ dàng với Unity thông qua các plugin đặc biệt. Điều này giúp quá trình phát triển được liên kết chặt chẽ với Unity Editor, mang lại sự thuận tiện và hiệu quả trong việc phát triển game trên Unity.
- **IntelliSense:** IntelliSense là một tính năng quan trọng trong Visual Studio, cung cấp gợi ý và thông tin về mã nguồn trong quá trình nhập liệu. Khi bạn bắt đầu gõ mã, IntelliSense sẽ tự động gợi ý các từ khóa, phương pháp, biến và các thành phần khác của mã nguồn. Điều này giúp lập trình viên viết mã nhanh chóng và chính xác, giảm thiểu lỗi cú pháp và nhanh chóng tìm kiếm thông tin cần thiết.
- **Xử Lý Nâng Cao và Tích Hợp Source Control:** Visual Studio hỗ trợ các tính năng xử lý nâng cao như refactoring, giúp tối ưu hóa và tái cấu trúc mã nguồn. Bằng cVisual Studio cung cấp một số chức năng và tính năng quan trọng khi sử dụng cho phát triển game trên Unity.

- **Remote Development:** Visual Studio hỗ trợ tính năng phát triển từ xa, cho phép lập trình viên phát triển trực tiếp trên máy chủ từ xa hoặc thiết bị khác, tăng tính linh hoạt trong quá trình phát triển.

Visual Studio không chỉ là một công cụ phát triển mã nguồn mạnh mẽ mà còn là một phần quan trọng của quy trình làm việc tích hợp cho lập trình game trên nền tảng Unity, đảm bảo hiệu suất và chất lượng sản phẩm cuối cùng.

2.1.3. Pixilart

Pixilart là một ứng dụng trực tuyến mạnh mẽ dành cho việc tạo và chia sẻ nghệ thuật pixel. Với Pixilart, người dùng có thể thỏa sức sáng tạo và tạo ra các tác phẩm nghệ thuật pixel hoàn hảo, từ hình ảnh đơn giản đến các bức tranh phức tạp.

Với giao diện trực quan và dễ sử dụng, Pixilart cung cấp một bảng vẽ mạnh mẽ với công cụ và tính năng đa dạng để tạo và chỉnh sửa nghệ thuật pixel. Người dùng có thể chọn từ một loạt màu sắc, kích thước cọ và các công cụ vẽ khác để tạo ra các đường nét chính xác và chi tiết.

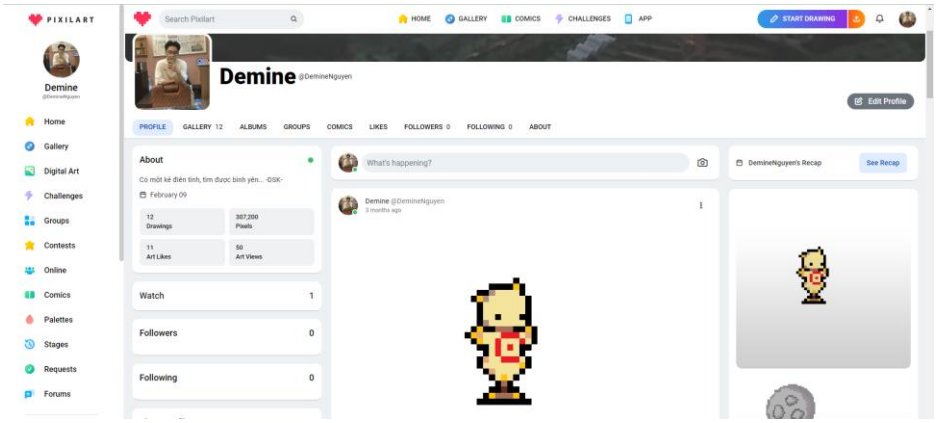
Hình 13. Công cụ vẽ Pixel Pixilart

Ngoài ra, Pixilart cũng cung cấp các tính năng tiên tiến như lớp, lịch sử chỉnh sửa và khả năng xem trước, giúp người dùng dễ dàng thao tác và tổ chức công việc của mình. Điều này cho phép người dùng tạo ra các tác phẩm pixel nghệ thuật đa lớp và phức tạp mà không bị giới hạn.

Một trong những điểm nổi bật của Pixilart là khả năng chia sẻ và tương tác với cộng đồng người dùng. Người dùng có thể tải lên và chia sẻ tác phẩm của mình trên Pixilart Gallery, nơi mọi người có thể tương tác, đánh giá và bình luận về các tác phẩm. Điều này tạo ra một môi trường sáng tạo và cộng đồng nghệ thuật năng động.

Với Pixilart, bất kỳ ai, từ người mới bắt đầu đến các nghệ sĩ pixel kỳ cựu, đều có thể tạo ra những tác phẩm nghệ thuật pixel độc đáo và thú vị. Với sự kết hợp giữa tính linh hoạt, sức mạnh và khả năng tương tác, Pixilart trở thành một công cụ quan trọng trong cộng đồng nghệ thuật pixel và cung cấp một nền tảng tuyệt vời để thể hiện sự sáng tạo của mọi người.

Trong tựa game của em thì em đã sử dụng Pixilart để vẽ model cho hình nhân mà em sử dụng trong tựa game này.



Hình 14. Profile Pixilart của em

2.1.4. Profiler

Profiler là một công cụ tích hợp mạnh mẽ trong Unity, thiết kế để giúp nhà phát triển theo dõi, đánh giá, và tối ưu hiệu suất của ứng dụng. Đặc biệt quan trọng trong phát triển game, Profiler cung cấp cái nhìn sâu sắc vào việc sử dụng tài nguyên hệ thống và hiệu suất của mã nguồn, từ đó giúp lập trình viên và nhóm phát triển xác định và giải quyết các vấn đề liên quan đến hiệu suất.

Profiler trong Unity không chỉ là một công cụ theo dõi hiệu suất mà còn mang lại nhiều chức năng quan trọng giúp nhà phát triển hiểu rõ hơn về cách ứng dụng của họ hoạt động và làm thế nào có thể tối ưu hóa chúng:

- **Đánh Giá Tài Nguyên Hệ Thống và Hiệu Suất Code:** Profiler không chỉ giúp xem xét việc sử dụng tài nguyên hệ thống mà còn phân tích hiệu suất của mã nguồn. Bằng cách theo dõi CPU, GPU, và bộ nhớ, nó giúp xác định rõ phần nào của ứng dụng tạo áp lực lớn lên hệ thống.
- **Xác Định và Giải Quyết Vấn Đề về Hiệu Suất:** Profiler cung cấp báo cáo chi tiết về các vấn đề liên quan đến hiệu suất, như thời gian xử lý quá lâu, chậm GPU, hoặc sử dụng bộ nhớ không hiệu quả. Nhờ đó, nhà phát triển có thể nhanh chóng xác định và giải quyết lỗi hiệu suất.
- **Phân Tích Đồ Thị và Biểu Đồ Thời Gian Thực:** Profiler hiển thị biểu đồ thời gian thực về CPU, GPU, và bộ nhớ, giúp nhìn rõ sự thay đổi của hiệu suất theo thời gian. Điều này giúp nhà phát triển phân tích các biểu đồ để xác định các điểm đặc biệt như spike hoặc giảm hiệu suất đột ngột.
- **Tối Ưu Hóa Mã Nguồn và Tài Nguyên:** Profiler không chỉ là công cụ chẩn đoán, mà còn là một phương tiện tối ưu hóa. Bằng cách cung cấp thông tin chi tiết về hiệu suất, nó giúp lập trình viên tinh chỉnh mã nguồn để giảm tải hệ thống, sử dụng bộ nhớ hiệu quả hơn, và tối ưu hóa các phần quan trọng của ứng dụng.
- **Hỗ Trợ Trong Quá Trình Phát Triển:** Profiler không chỉ là công cụ dành cho giai đoạn kiểm thử mà còn có sẵn trong Unity Editor, giúp nhà phát triển theo dõi hiệu suất ngay cả khi đang phát triển và thử nghiệm. Việc này giúp phát hiện vấn đề ngay từ giai đoạn phát triển sớm nhất.
- **Kết Hợp Với Công Cụ Khác:** Profiler tích hợp chặt chẽ với các công cụ khác trong Unity như Animator, Physics Debugger, tạo ra một trải nghiệm toàn diện giúp nhà phát triển hiểu rõ hơn về hiệu suất và tương tác giữa các phần của game. Điều này giúp đưa ra quyết định chiến lược để cải thiện trải nghiệm người chơi.

2.1.5. Unity Version Control (VCS)

Trong Unity, quản lý phiên bản (version control) vẫn là một phần quan trọng trong quy trình phát triển game để theo dõi sự thay đổi trong mã nguồn, tài nguyên và cấu trúc dự án. Unity vẫn hỗ trợ nhiều hệ thống quản lý phiên bản phổ biến, chủ yếu là Git và Unity Collaborate.

❖ **GIT**

- **Khởi tạo Repository:** Unity tích hợp chặt chẽ với Git, cho phép bạn khởi tạo repository trực tiếp từ Unity Editor hoặc sử dụng Git command line. Bằng cách tạo repository, bạn có thể lưu trữ và theo dõi mã nguồn, tài nguyên và cấu trúc dự án của bạn.
- **Theo dõi File và Thay đổi:** Git theo dõi sự thay đổi trong các file, bao gồm script, tài nguyên hình ảnh, âm thanh và các tệp khác. Bạn có thể sử dụng Git để commit và lưu trữ các phiên bản khác nhau của dự án.
- **Nhánh (Branching) và Merge:** Unity hỗ trợ tạo và quản lý các nhánh (branches) trong Git. Các nhánh cho phép các thành viên trong nhóm phát triển làm việc độc lập trên các tính năng riêng của dự án mà không làm ảnh hưởng đến nhau. Sau đó, bạn có thể merge (hợp nhất) các thay đổi từ các nhánh khác nhau để kết hợp các tính năng đã phát triển.
- **Unity YAML Files:** Unity sử dụng các file YAML để lưu trữ thông tin về cấu trúc dự án, thiết lập và các tài nguyên. Điều này giúp Git hiểu và theo dõi sự thay đổi trong cấu trúc của dự án Unity. Các file YAML cung cấp thông tin về các tệp và thư mục, các thiết lập của dự án và các kết nối giữa các tài nguyên.

❖ **Unity Collaborate**

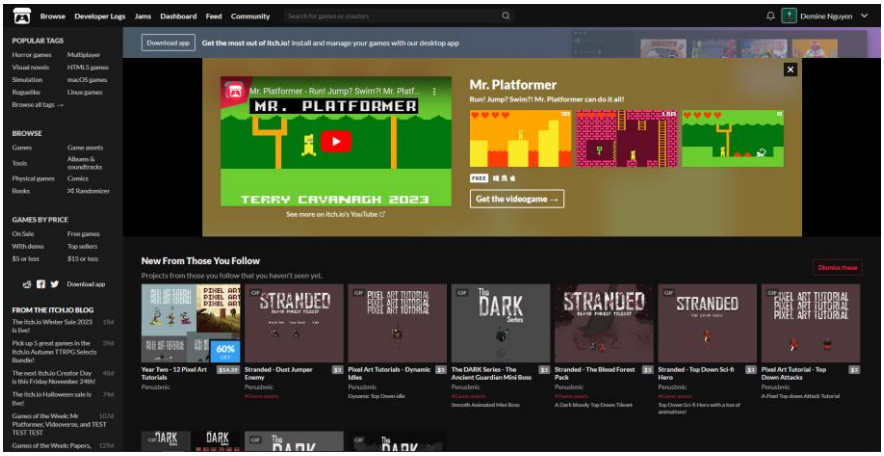
- **Tích hợp Tự động:** Unity Collaborate là một dịch vụ quản lý phiên bản tích hợp sẵn trong Unity. Nó cung cấp tính năng theo dõi thay đổi, commit và nhánh một cách tự động mà không yêu cầu cấu hình phức tạp. Collaborate tự động xác định các tệp và tài nguyên cần commit và cung cấp giao diện trực quan để thực hiện các thao tác quản lý phiên bản.

- **Tích hợp Dễ dàng:** Collaborate được tích hợp sẵn trong Unity Editor, giúp đơn giản hóa quá trình quản lý phiên bản và làm việc nhóm. Bạn có thể truy cập các tính năng Collaborate trực tiếp từ giao diện Unity Editor, mà không cần phải sử dụng các công cụ bên ngoài hoặc kiến thức sâu về Git.
- **Quay lại Phiên bản Cụ thể:** Collaborate cho phép nhóm quay lại các phiên bản cụ thể của dự án. Điều này rất hữu ích khi cần khắc phục lỗi, xử lý các vấn đề liên quan đến sự thay đổi hoặc so sánh các phiên bản khác nhau của dự án.
- **Bảo mật và Quyền truy cập:** Collaborate cung cấp tính năng quản lý quyền truy cập và bảo mật trong dự án Unity. Bạn có thể kiểm soát người dùng nào được phép thực hiện thay đổi và commit trong dự án, giúp đảm bảo an ninh và kiểm soát quyền truy cập.

Lựa chọn giữa Git và Unity Collaborate phụ thuộc vào yêu cầu cụ thể của dự án và sự thoải mái của nhóm phát triển với các công cụ này. Trong mọi trường hợp, quản lý phiên bản là một phần quan trọng để đảm bảo sự ổn định và theo dõi sự phát triển của dự án Unity.

2.2. Tài nguyên có sẵn và cộng đồng hỗ trợ

2.2.1. Itch.io



Hình 15. Trang web itch.io

Itch.io là một cộng đồng trực tuyến cho các nhà phát triển game độc lập và người chơi, nơi họ có thể tạo, chia sẻ, và khám phá trò chơi sáng tạo. Nền tảng này không chỉ hỗ trợ trò chơi mà còn cung cấp không gian cho các tác phẩm nghệ thuật số khác như ứng dụng, sách, và âm nhạc.

Ở trong tựa game của em, ngoài model của hình nhân (Dummy), thì hầu hết các model nhân vật khác được em lấy từ trên đây.

❖ **Đặc điểm và ưu điểm**

- **Đăng Tải và Chia Sẻ:** Nhà phát triển có thể đăng tải trò chơi của họ một cách dễ dàng trên itch.io, kèm theo hình ảnh, video, và mô tả để người chơi có cái nhìn tổng quan về sản phẩm.
- **Tính Tự Do và Độc Lập**: itch.io nổi tiếng với tính tự do và sự độc lập. Nhà phát triển có quyền quyết định giá cả, cũng như cách họ muốn phân phối và bán trò chơi của mình.
- **Cộng Đồng Chia Sẻ:** Cộng đồng trên itch.io là một phần quan trọng, nơi mọi người có thể tương tác, đánh giá, và đăng bình luận về trò chơi. Điều này tạo ra môi trường tích cực để phát triển và cải thiện sản phẩm.
- **Giao Diện Người Dùng Thân Thiện**: Giao diện itch.io được thiết kế để dễ sử dụng và trải nghiệm người dùng tốt, giúp cả nhà phát triển và người chơi dễ dàng tương tác với nền tảng.
- **Đa Dạng Nội Dung:** Nền tảng này không chỉ hỗ trợ trò chơi mà còn nơi để người sáng tạo chia sẻ các tác phẩm nghệ thuật số khác như ứng dụng, sách, và âm nhạc.
- **Hỗ Trợ Cho Game Jam:** itch.io thường được sử dụng làm nền tảng chính cho các sự kiện game jam, giúp những dự án ngắn hạn và sáng tạo có cơ hội được người chơi khám phá.
- **Hỗ Trợ Thương Mại và Phi Thương Mại:** Nhà phát triển có thể chọn giữa mô hình kinh doanh thương mại hoặc phi thương mại cho sản phẩm của mình, tạo ra sự linh hoạt trong quá trình phân phối.

- **Hệ Thống Thanh Toán An Toàn:** itch.io cung cấp các phương thức thanh toán an toàn, bảo vệ thông tin người dùng và nhà phát triển.

2.2.2. Unity Documentation

Unity Documentation là tài liệu chính thức từ Unity không chỉ là nguồn thông tin chính xác và chi tiết, mà còn là nguồn hỗ trợ quan trọng cho nhà phát triển game sử dụng Unity. Bao gồm API Reference, hướng dẫn, và ví dụ mã nguồn, tài liệu này đảm bảo rằng người phát triển có đầy đủ thông tin để tận dụng tối đa các tính năng mạnh mẽ của Unity.

❖ **Đặc điểm và ưu điểm**

- **Hướng Dẫn Chi Tiết:** Unity Documentation cung cấp hướng dẫn chi tiết về cách sử dụng mọi khía cạnh của công cụ, từ cài đặt dự án mới đến xử lý logic phức tạp trong game. Thông tin được tổ chức rõ ràng, giúp người đọc dễ dàng theo dõi và hiểu rõ về các tính năng và quy trình làm việc của Unity.
- **Hỗ Trợ Giải Quyết Vấn Đề:** Unity Documentation là một nguồn lực quan trọng cho việc giải quyết vấn đề. Bạn có thể tìm thấy hướng dẫn về cách xử lý các lỗi phổ biến, vấn đề hiệu suất, và các thách thức khác trong quá trình phát triển game. Cung cấp các mẹo và chiến lược giúp người phát triển vượt qua những thách thức phổ biến.
- **Cập Nhật Liên Tục:** Unity Documentation được cập nhật liên tục theo các phiên bản mới của Unity. Điều này giúp đảm bảo rằng người phát triển luôn có truy cập vào thông tin mới nhất và tương thích với phiên bản Unity đang sử dụng.
- **Tổ Chức và Tìm Kiếm Hiệu Quả:** Hệ thống tìm kiếm trong tài liệu Unity rất mạnh mẽ, giúp người phát triển nhanh chóng tìm thấy thông tin cụ thể mà họ đang cần. Tài liệu được tổ chức theo chủ đề, giúp người đọc dễ dàng theo dõi quá trình học và nắm bắt thông tin theo hướng thích hợp.

2.2.3. 2D - Unity Forum

2D-Unity Forum là một cộng đồng trực tuyến chuyên về phát triển game 2D bằng Unity. Nơi đây cho phép những người sáng tạo và nhà phát triển giao lưu, hỏi đáp và chia sẻ kiến thức về việc tạo ra trò chơi 2D sử dụng Unity Engine. Diễn đàn này đóng vai trò quan trọng trong việc tạo ra một môi trường hỗ trợ cho những người làm game 2D.

❖ **Đặc điểm và ưu điểm**

- **Hướng dẫn và bài viết:** Thành viên có thể viết và chia sẻ các hướng dẫn, bài viết về các kỹ thuật, chiến lược và tiện ích hữu ích cho phát triển game 2D trong Unity. Bài viết có thể tập trung vào các chủ đề như animation, quản lý sprite hoặc cách tối ưu hóa hiệu suất.
- **Giới thiệu dự án và demo:** Người dùng có thể giới thiệu và chia sẻ dự án game 2D của họ, kèm theo demo để nhận phản hồi từ cộng đồng. Đây là cơ hội để những dự án nhỏ hoặc thử nghiệm nhận được sự chú ý và hỗ trợ.
- **Thảo luận và giao lưu:** Diễn đàn tạo ra không gian thảo luận và giao lưu giữa các thành viên, từ chia sẻ ý tưởng game cho đến trao đổi kinh nghiệm và thảo luận về các xu hướng mới trong làm game 2D.
- **Hỗ trợ qua cộng đồng:** Cộng đồng này không chỉ giúp đỡ kỹ thuật mà còn mang lại sự hỗ trợ tinh thần và động viên cho những người mới bắt đầu hoặc gặp khó khăn trong hành trình phát triển game.
- **Thông báo và sự kiện:** Diễn đàn thông báo về các sự kiện, cuộc thi hoặc các buổi workshop 2D Unity để cộng đồng được cập nhật với các cơ hội mới.

Chương 3. Giải quyết vấn đề

1. Thiết kế game

1.1. Tổng quan trò chơi

- **Tên trò chơi:** Da Sun Kids
- **Thể loại game:** Souls-like, 2D platform
- **Bản sắc trò chơi:** Mang đúng bản chất của dòng souls-like, với độ khó cao, yêu cầu người chơi phải tập trung cao độ để cố gắng vượt qua từng thử thách.
- **Nghê thuật và thiết kế (Art style):** Sử dụng đồ họa 2D pixel, có xu hướng hoạt hình.

1.2. Gameplay

Như đã nói ở trên thì ở trong phạm vi của đồ án này, do thời gian không đủ nên em chỉ mới kịp xây dựng AI cho con Boss chính của game. Dựa vào bối cảnh của game cũng như tạo hình của model mà em đã chọn em đã đặt tên cho con Boss là "Delectric Spider".

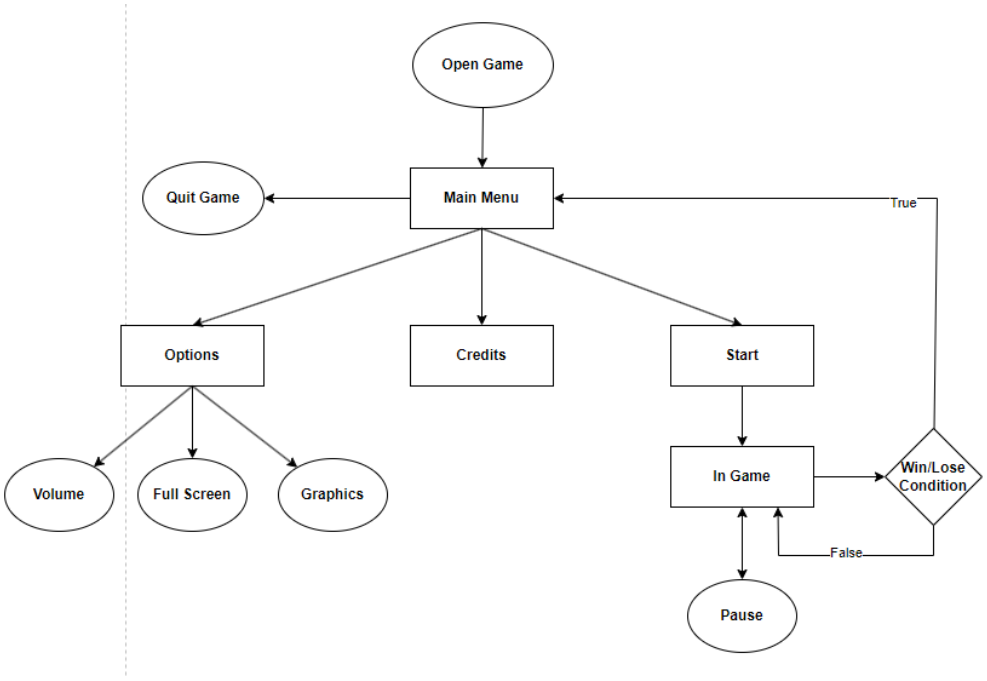
Và để thể hiện được những gì mà con Boss có thể làm, em sử dụng 1 hình nhân (Dummy) với mục đích chính là để đánh giá được mức độ sát thương từ Boss cũng như dễ dàng hơn cho người xem có thể hiểu được cơ chế hoạt động của Boss.

- **Mục tiêu**

Như đa số các game thuộc thể loại souls-like, mục tiêu chính của trò chơi là đánh bại hết tất cả các quái ở trong game. Rất đơn giản và dễ hiểu, tuy nhiên điều làm nên sự khác biệt cho các tựa game thuộc thể loại này đó là nó rất khó, cực kỳ khó. Và chính việc khó khăn để đánh bại quái là điều làm nên sự hấp dẫn cho thể loại này.

- **Tiến trình game**

Sơ đồ bên dưới mô tả tổng quát tiến trình game, cách hoạt động của các scene trong game.



Hình 16. Sơ đồ tiến trình game

1.3. Model của nhân vật trong game

Trong tựa game ở thời điểm hiện tại sẽ gồm 2 nhân vật chính là hình nhân – Dummy và Delectric Spider.

Hình 17. Model của Boss

Hình 18. Model của hình nhân

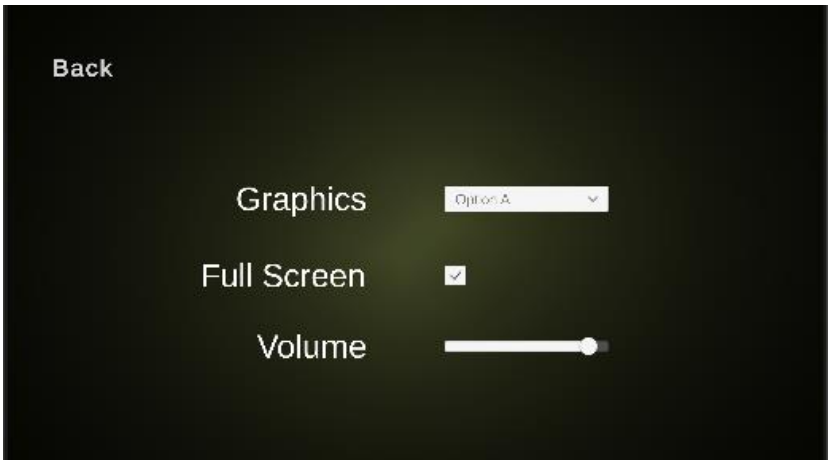
1.4. Điều khiển trong trò chơi

Trong version 1 của game hiện nay, người chơi sẽ điều khiển Dummy di chuyển theo chiều ngang bằng 2 phím điều hướng trái và phải. Do mục đích chính của đồ án lần này là cấu hình AI cho Boss nên điều khiển ở trong game khá đơn giản.

1.5. Giao diện người dùng

Giao diện người dùng (UI) là điểm tương tác và giao tiếp giữa người và máy tính trong một thiết bị. Nó bao gồm các thành phần như màn hình hiển thị, bàn phím, chuột và các phần khác của máy tính để bàn. Giao diện người dùng cũng áp dụng cho các ứng dụng và trang web, là cách mà người dùng tương tác với chúng [4].

Hình 19. Main menu



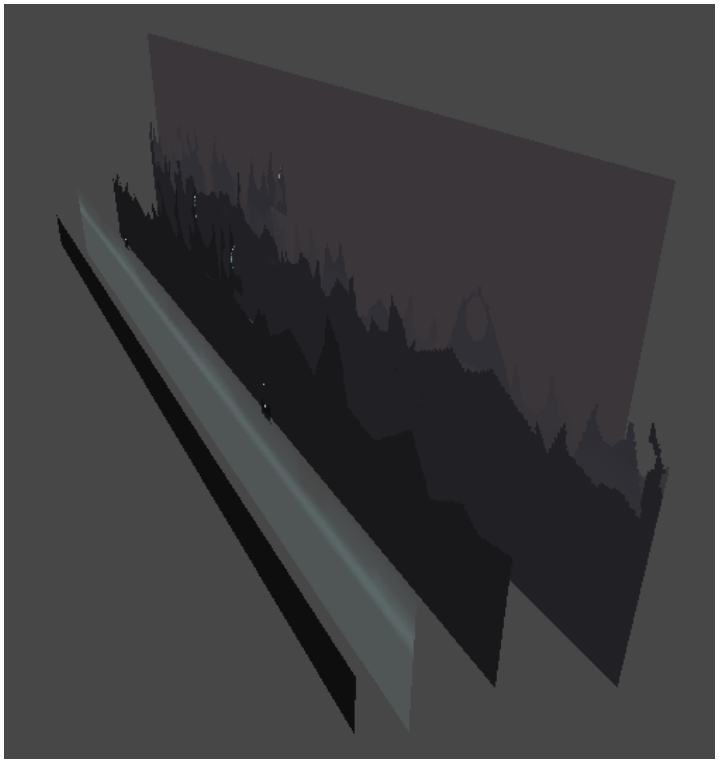
Hình 20. Options menu

Hình 21. Cửa sổ khi Pause game

2. Phát triển game trên Unity

2.1. Xây dựng Background

Ý tưởng của Backgroud là một thành phố hoang tàn sau cuộc tàn phá của Delectric Spider. Background của game được xây dựng từ các layer khác nhau, bao gồm cả hiệu ứng của một số object ở trong game.



Hình 22. Các layer backgroud

Điểm đặc biệt của background này là nó được áp dụng hiệu ứng "Parallax". Hiệu ứng parallax là một hiện tượng được sử dụng trong thiết kế đồ họa và trang web để tạo ra một cảm giác sâu, sống động và một chiều sâu giả tưởng cho người xem. Hiệu ứng này xảy ra khi các vật thể trong tầm nhìn di chuyển với tốc độ khác nhau, tạo ra một hiệu ứng sự chuyển động hoặc sự thay đổi vị trí tương đối giữa chúng.[5]

Để thiết lập được hiệu ứng như vậy chúng ta sẽ cần phải kết hợp giữa việc code và xử lý camera ở trên game. Camera sẽ di chuyển theo nhân vật mà mình điều khiển.

```
public class Parallax : MonoBehaviour
{
    [SerializeField] private float parallaxFactor;

    public void Move(float delta)
    {
        Vector3 newPos = transform.localPosition;
        newPos.x -= delta * parallaxFactor;

        transform.localPosition = newPos;
    }
}
```

Hình 23. Parallax.cs

```
public class ParallaxCamera : MonoBehaviour
{
    public delegate void ParallaxCameraDelegate(float deltaMovement);
    public ParallaxCameraDelegate onCameraTranslate;

    private float oldPosition;

    void Start()
    {
        oldPosition = transform.position.x;
    }

    void Update()
    {
        if (transform.position.x != oldPosition)
        {
            if (onCameraTranslate != null)
            {
                float delta = oldPosition - transform.position.x;
                onCameraTranslate(delta);
            }

            oldPosition = transform.position.x;
        }
    }
}
```

Hình 24. ParallaxCamera.cs

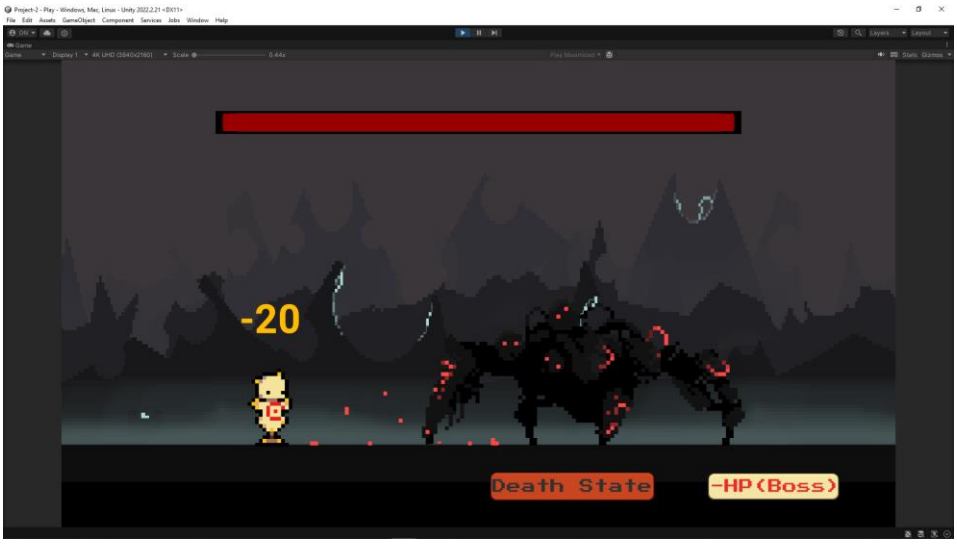
```
public class ParallaxController : MonoBehaviour
{
    public ParallaxCamera parallaxCamera;
    public List<Parallax> parallaxLayers = new List<Parallax>();

    void Start()
    {
        if (parallaxCamera == null)
            parallaxCamera = Camera.main.GetComponent<ParallaxCamera>();

        if (parallaxCamera != null)
            parallaxCamera.onCameraTranslate += Move;

        SetLayers();
    }

    void SetLayers()
    {
        parallaxLayers.Clear();

        for (int i = 0; i < transform.childCount; i++)
        {
            Parallax layer = transform.GetChild(i).GetComponent<Parallax>();

            if (layer != null)
            {
                layer.name = "Layer-" + i;
                parallaxLayers.Add(layer);
            }
        }
    }

    void Move(float delta)
    {
        foreach (Parallax layer in parallaxLayers)
        {
            layer.Move(delta);
        }
    }
}
```

Hình 25. ParallaxController.cs

2.2. Xây dựng hình nhân (Dummy) trong game

Nhân vật Dummy được xây dựng với mục đích để thể hiện được những gì mà con Boss có thể làm trong game. Cho nên việc xây dựng lên nó cũng khá đơn giản, chỉ việc add model của Dummy vào game và code cơ chế điều khiển cho nó.

Và để thể hiện được rõ những sát thương mà nó nhận được thì em đã code để khi Boss gây sát thương lên nó, thì sẽ hiển thị sát thương lên màn hình của game.



Hình 26. Hiển thị lượng sát thương mà hình nhân nhận phải

Code của hình nhân sẽ bao gồm:

```
public class CameraBound : MonoBehaviour
{
    public Transform dummy;
    public Camera cam;

    [SerializeField] private float horizontalMargin = 0.3f;
    [SerializeField] private float verticalMargin = 0.4f;
    [SerializeField] private float depth = -10;
    [SerializeField] private float smoothTime = 0.25f;

    Vector3 target;
    Vector3 lastPosition;
    Vector3 currentVelocity;

    private void LateUpdate()
    {
        SetTarget();
        MoveCamera();
    }

    void SetTarget()
    {
        Vector3 movementDelta = dummy.position - lastPosition;
        Vector3 screenPos = cam.WorldToScreenPoint(dummy.position);
        Vector3 bottomLeft = cam.ViewportToScreenPoint(new Vector3(horizontalMargin, verticalMargin, 0));
        Vector3 topRight = cam.ViewportToScreenPoint(new Vector3(1 - horizontalMargin, 1 - verticalMargin, 0));

        if (screenPos.x < bottomLeft.x || screenPos.x > topRight.x)
        {
            target.x += movementDelta.x;
        }

        if (screenPos.y < bottomLeft.y || screenPos.y > topRight.y)
        {
            target.y += movementDelta.y;
        }

        target.z = depth;
        lastPosition = dummy.position;
    }

    void MoveCamera()
    {
        transform.position = Vector3.SmoothDamp(transform.position, target, ref currentVelocity, smoothTime);
    }
}
```

Hình 27. CameraBond.cs

```
public class DummyController : MonoBehaviour
{
    public static DummyController Instance;

    public GameObject popupDamagePrefab;

    private Rigidbody2D rb;
    public SpriteRenderer sr;

    protected float x = 0f;

    public TMP_Text popUpText;


    [SerializeField] private float speed = 20f;


    private void Awake()
    {
        if (Instance == null)
            Instance = this;
    }

    private void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        sr = GetComponent<SpriteRenderer>();
    }

    private void Update()
    {
        x = Input.GetAxis("Horizontal");

        Flip();
    }

    private void FixedUpdate()
    {
        rb.velocity = new Vector2(x * speed, rb.velocity.y);
    }

    private void Flip()
    {
        if (x < 0f)
        {
            transform.eulerAngles = new Vector2(0, 180);
        }
        else if (x > 0f)
        {
            transform.eulerAngles = new Vector2(0, 0);
        }
    }

    public void TakeDamage(float damage)
    {
        popUpText.text = "-" + damage.ToString();

        Vector3 headPosition = transform.position + new Vector3(0f, 2f, 0f);

        Instantiate(popupDamagePrefab, headPosition, Quaternion.identity);
    }
}
```

Hình 28. DummyController.cs

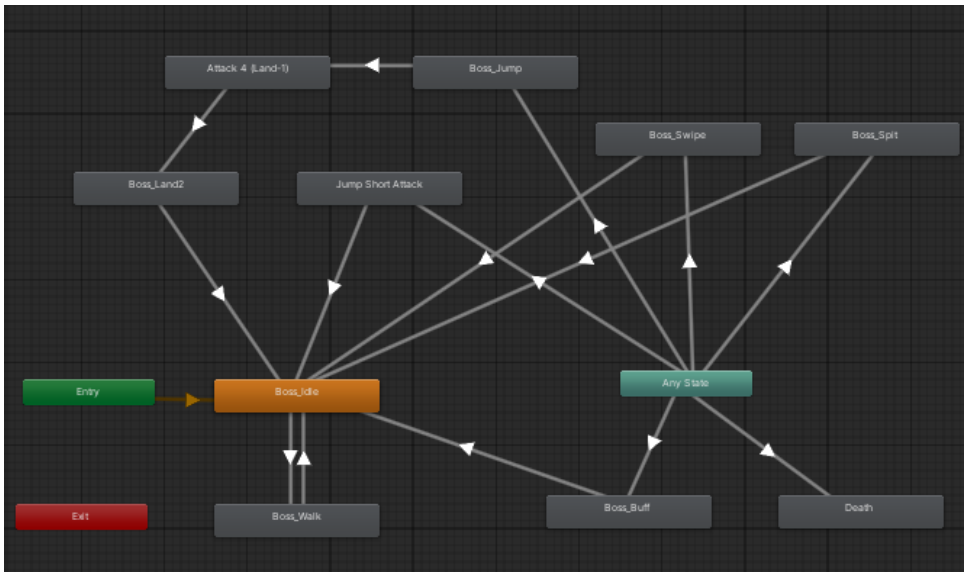
2.3. Xây dựng nhân vật Deletric Spider

Đồ án này tập trung để xây dựng AI cho Boss cho nên đây là phần được đầu tư nhiều thời gian vào nhất. Delectric Spider là một con nhện máy được tạo ra bởi một nhà bác học điên, nhưng trong quá trình phát triển đã xảy ra lỗi làm cho nó bị mất kiểm soát và tàn phá hết tất cả mọi thứ xung quanh nó.

2.3.1. Animation của Boss

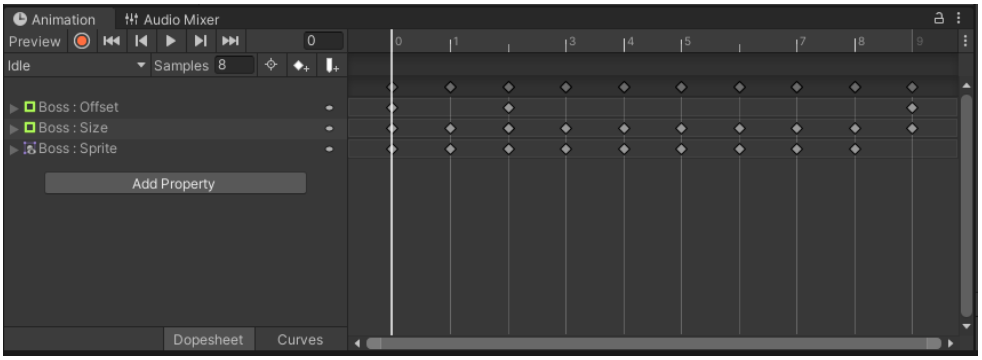
Delectric Spider bao gồm 8 hoạt cảnh gồm: *Swipe, Spit, Jump, Land, Buff, Death, Idle, Walk.* Sau khi tính toán thì em đã xây dựng cho con Boss có 4 cách thức tấn công bao gồm: Swipe, Spit, Jump short, Jump high. Sự khác biệt giữa 2 kiểu Junp là độ cao và phạm vi của nó cũng như tính sát thương mà nó gây ra.

Sơ đồ Animator của Boss như sau:



Hình 29. Cửa sổ Animator của Boss

Sử dụng cửa sổ Animation để căn chỉnh lại tất cả animation của Boss trước khi thêm nó vào trong game.



Hình 30. Cửa sổ Animation của Boss

2.3.2. Hitbox của Boss

Hitbox trong game là một thuật ngữ được sử dụng rộng rãi trong các game đối kháng, game bắn súng và nhiều loại game khác. Hitbox đóng vai trò quan trọng trong gameplay bằng cách quyết định xem một đòn đánh hay kỹ năng nào đó có trúng vào đối thủ hay không. Nếu hitbox của nhân vật đang ở trong phạm vi hitbox của đối thủ, thì đòn đánh sẽ gây ra sát thương. Tuy nhiên, nếu hitbox của nhân vật không đúng với hitbox của đối thủ, thì đòn đánh sẽ không gây ra sát thương, mà chỉ làm cho đối thủ phòng tránh tốt hơn và di chuyển tạo khoảng cách [6].

Hình 31. Vùng Hitbox của Boss

Các vùng đỏ tương ứng với từng loại hitbox của từng loại skill mà Boss có thể sử dụng. Khi mà nhân vật bạn điều khiển di chuyển vào các vùng đỏ mà lúc đó nó sử dụng skill tương ứng thì bạn sẽ bị trừ damage theo từng loại skill tương ứng. Và để có thể gây sát thương thì cũng sẽ cần phải code để làm được điều đó.

```
public class BossEvents : MonoBehaviour
{
    void SwipeDamageDummy()
    {
        if (DummyController.Instance.transform.position.x - transform.position.x != 0)
        {
            SwipeHit(Boss.Instance.sideAttackTransform1, Boss.Instance.sideAttackArea1);
        }
    }

    void SpitDamageDummy()
    {
        if (DummyController.Instance.transform.position.x - transform.position.x != 0)
        {
            SpitHit(Boss.Instance.sideAttackTransform2, Boss.Instance.sideAttackArea2);
        }
    }

    void HighJumpDamageDummy()
    {
        if (DummyController.Instance.transform.position.x - transform.position.x != 0)
        {
            HighJumpHit(Boss.Instance.landAttackTransform, Boss.Instance.landAttackArea);
        }
    }

    void ShortJumpDamageDummy()
    {
        if (DummyController.Instance.transform.position.x - transform.position.x != 0)
        {
            ShortJumpHit(Boss.Instance.landAttackTransform, Boss.Instance.landAttackArea);
        }
    }
```

Hình 32. Code để gán vào các event trên animation

Các method trên là để add vào event trên từng animation để gây sát thương lên Dummy.

```
    void SwipeHit(Transform attackTransform, Vector2 attackArea)
    {
        Collider2D objectsToHit = Physics2D.OverlapBox(attackTransform.position, attackArea, 0);

        if (objectsToHit.GetComponent<DummyController>() != null)
        {
            objectsToHit.GetComponent<DummyController>().TakeDamage(Boss.Instance.damageSwipe);
        }
    }

    void SpitHit(Transform attackTransform, Vector2 attackArea)
    {
        Collider2D objectsToHit = Physics2D.OverlapBox(attackTransform.position, attackArea, 0);

        if (objectsToHit.GetComponent<DummyController>() != null)
        {
            objectsToHit.GetComponent<DummyController>().TakeDamage(Boss.Instance.damageSpit);
        }
    }

    void HighJumpHit(Transform attackTransform, Vector2 attackArea)
    {
        Collider2D objectsToHit = Physics2D.OverlapBox(attackTransform.position, attackArea, 0);

        if (objectsToHit.GetComponent<DummyController>() != null)
        {
            objectsToHit.GetComponent<DummyController>().TakeDamage(Boss.Instance.damageHighJump);
        }
    }

    void ShortJumpHit(Transform attackTransform, Vector2 attackArea)
    {
        Collider2D objectsToHit = Physics2D.OverlapBox(attackTransform.position, attackArea, 0);

        if (objectsToHit.GetComponent<DummyController>() != null)
        {
            objectsToHit.GetComponent<DummyController>().TakeDamage(Boss.Instance.damageShortJump);
        }
    }
```

Hình 33. Các method để gây sát thương

Còn với đoạn code này là để xác định xem Dummy có nằm ở trong hitbox hay không.

2.3.3. Quản lý skill của Boss

Như đã nói trên thì con Boss này bao gồm 4 skill, vậy việc quản lý nó sẽ như thế nào. Tất nhiên vẫn phải code để cho nó có thể random các loại kỹ năng khác nhau.

Con Boss này bao gồm 2 trạng thái: Trạng thái 1 trên 2/3 lượng máu, trạng thái 2 dưới 1/3 lượng máu. Và việc lựa chọn skill của nó cũng phụ thuộc nó thuốc trạng thái nào.

❖ **Trạng thái 1 của Boss**

Bên đây là các skill của Boss khi ở trạng thái 1 bao gồm: *Swipe, Spit, High Jump, Short Jump*. Tất nhiên để khó hơn thì ở trạng thái 1 tốc độ đánh của Boss sẽ chậm hơn và tốc độ di chuyển cũng tương tự.

```
#region Boss_State1
IEnumerator SwipeAttack()
{
    attacking = true;
    rb.velocity = Vector2.zero;

    anim.SetTrigger("Swipe");
    yield return new WaitForSeconds(1f);
    anim.ResetTrigger("Swipe");

    ResetAllAttacks();
}

IEnumerator SpitAttack()
{
    attacking = true;
    rb.velocity = Vector2.zero;

    anim.SetTrigger("Spit");
    yield return new WaitForSeconds(1f);
    anim.ResetTrigger("Spit");

    ResetAllAttacks();
}

IEnumerator HighJumpAttack()
{
    attacking = true;

    anim.SetTrigger("Jump");
    yield return new WaitForSeconds(2f);

    anim.SetTrigger("Suspended");
    //yield return new WaitForSeconds(0.5f);

    if (Grounded() == true)
    {
        anim.SetTrigger("Land");
    }

    ResetAllAttacks();
}

IEnumerator ShortJumpAttack()
{
    attacking = true;

    anim.SetTrigger("JumpShort");
    yield return new WaitForSeconds(3f);
    anim.ResetTrigger("JumpShort");

    ResetAllAttacks();
}
#endregion
```

Hình 34. Các skill của Boss ở trạng thái 1

❖ Trạng thái 2 của Boss

Điểm có thể dễ nhận thấy nhất ở đây là nó chỉ còn gồm 3 skill nữa, không còn skill high Jump nữa. Tuy nhiên thì tốc độ đánh sẽ nhanh hơn và mỗi đòn lại có các kiểu riêng. Ví dụ: 3 đòn Swipe 1 lúc, 2 đòn nhảy 1 lúc,…

```
#region Boss State 2
IEnumerator TripleSwipeAttack()
{
    attacking = true;
    rb.velocity = Vector2.zero;

    anim.SetTrigger("Swipe");
    yield return new WaitForSeconds(1.5f);
    anim.ResetTrigger("Swipe");

    anim.SetTrigger("Swipe");
    yield return new WaitForSeconds(1f);
    anim.ResetTrigger("Swipe");

    anim.SetTrigger("Swipe");
    yield return new WaitForSeconds(1f);
    anim.ResetTrigger("Swipe");

    ResetAllAttacks();
}

IEnumerator DoubleSpitAttack()
{
    attacking = true;
    rb.velocity = Vector2.zero;

    anim.SetTrigger("Spit");
    yield return new WaitForSeconds(1.2f);
    anim.ResetTrigger("Spit");

    anim.SetTrigger("Spit");
    anim.ResetTrigger("Spit");

    ResetAllAttacks();
}

IEnumerator DoubleShortJumpAttack()
{
    attacking = true;

    anim.SetTrigger("JumpShort");
    anim.ResetTrigger("JumpShort");

    anim.SetTrigger("JumpShort");
    yield return new WaitForSeconds(1.5f);
    anim.ResetTrigger("JumpShort");

    ResetAllAttacks();
}
#endregion
```

Hình 35. Các Skill của Boss ở trạng thái 2

❖ Kiểm soát các skill

Câu hỏi đặt ra là làm sao để biết Boss sẽ sử dụng skill gì để tấn công, em sẽ sử dụng 1 method là "AttackHandler()" để random ra skill mà Boss lựa chọn sử dụng.

```
public void AttackHandler()
{
    if (currentEnemyState == EnemyStates.Boss_State1)
    {
        float randomValue = Random.value;

        if (randomValue < 0.7f)
        {
            if (Vector2.Distance(DummyController.Instance.transform.position, rb.position) <= attackRange)
                ManageTypeOfAttack1();
        }
        else if (randomValue < 0.95f)
        {
            if (Vector2.Distance(DummyController.Instance.transform.position, rb.position) <= jumpAttackRange)
                StartCoroutine(ShortJumpAttack());
        }
        else
        {
            StartCoroutine(HighJumpAttack());
        }
    }

    if (currentEnemyState == EnemyStates.Boss_State2)
    {
        float randomValue = Random.value;

        if (randomValue < 0.3f)
        {
            if (Vector2.Distance(DummyController.Instance.transform.position, rb.position) <= attackRange)
                ManageTypeOfAttack1();
        }
        else if (randomValue < 0.6f)
        {
            if (Vector2.Distance(DummyController.Instance.transform.position, rb.position) <= attackRange)
                StartCoroutine(TripleSwipeAttack());
        }
        else
        {
            if (Vector2.Distance(DummyController.Instance.transform.position, rb.position) <= jumpAttackRange)
                StartCoroutine(DoubleShortJumpAttack());
        }
    }
}
```

Hình 36. Method AttackHandler()

2.4. Xây dựng các scene trong game

2.4.1. Main menu scene

Khi vừa bắt đầu vào trò chơi, đây sẽ là scene đầu tiên mà chúng ta sẽ gặp. Trong main menu sẽ có 4 lựa chọn: *START*, *OPTIONS*, *CREDIT*, *QUIT*.

Hình 37. Giao diện khi vừa vào game

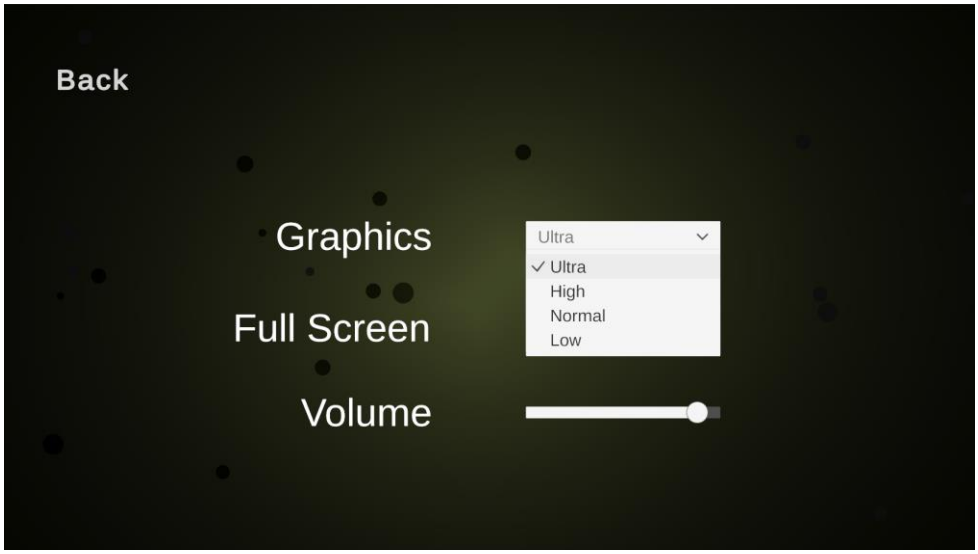
Trên cùng của menu là tên game được em tự thiết kế, để cho bớt nhàm chán em đã thêm animation cho logo này.

Hình 38. Logo tên game

- START: Khi click vào start thì sẽ bắt đầu vào trò chơi .
- OPTIONS: Đây là nơi để bạn có thể chỉnh các option của game là âm thanh, fullscreen và đồ họa của game.
- CREDIT: Hiển thị người tạo ra game.
- QUIT: Thoát game

2.4.2. Options scene

Như đã nói ở trên Options menu là nơi để chỉnh sửa các thông số của game.

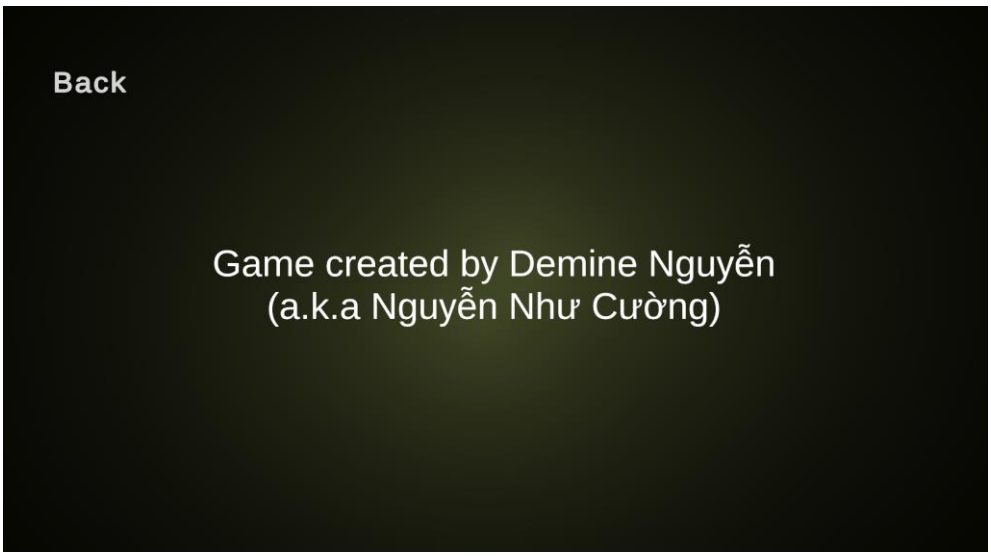


Hình 39. Giao diện menu options

Có 4 mức đồ họa có thể chọn theo thứ tự chất lượng đồ họa giảm dần: *Ultra, High, Normal, Low.* Nếu muốn chọn toàn màn hình thì chúng ta sẽ tích vào Fullscreen và ngược lại. Volume sẽ được cấu hình bằng một thanh slider, chắc cũng không cần phải giải thích vì hầu như hiện nay bất kể một ứng dụng nào có điều chỉnh âm thanh thì chúng ta cũng đã quá quen với kiểu như này.

2.4.3. Credit scene

Hiển thị người làm ra game ở đây là em - Nguyễn Như Cường.



Hình 40. Credit của Game

2.4.4. Play scene

Đây là màn để chúng ta thực hiện chơi tựa game, tuy nhiên do chưa có cơ chế điều khiển nhân vật đánh trả, nên để thể hiện được các cơ chế của Boss như Buff và Death em sẽ tạo thêm các button với nhiệm vụ là trừ máu của Boss theo từng click hoặc là chết luôn.

Hình 41. Cửa sổ khi chơi game

Bên cạnh đó còn hiển thị thanh máu của Boss. Và để có thể trừ máu hoặc là chuyển sang trạng thái chết "Death state". Em cũng phải kết hợp với code để thực hiện trừ máu cho Boss. Mỗi lần click vào nút "-HP(Boss)" sẽ trừ 100 máu của Boss.

```
public class ButtonManage : MonoBehaviour
{
    public void ReduceHPClick()
    {
        Boss.Instance.SetHealth(100f);

        Debug.Log(Boss.Instance.health);
    }

    public void DeathStateClick()
    {
        Boss.Instance.SetHealth(Boss.Instance.health - 5);

        Debug.Log("Boss is death!");
    }
```

Hình 42. Quản lý việc nhấp vào các Button

Ngoài ra khi nhấn phím escape sẽ hiển thị màn hình Pause của game, khác với việc chuyển scene thì paused được tạo bằng canvas và nó sẽ hiển thị đè lên màn của game chứ không chuyển sang màn khác.

Hình 43. Pause Game

3. Kiểm thử và sửa lỗi

Sau khi hoàn thiện các tính năng trong game sẽ bắt đầu đi vào việc kiểm thử và sửa lỗi. Trong quá trình kiểm thử em phát hiện ra game có 2 lỗi

- Lỗi 1: Thi thoảng hình ảnh ở trong Game view bị lúc ẩn, lúc hiện.
- Lỗi 2: Hitbox của Boss có vấn đề.

Sau khi tìm hiểu và sửa lại code thì em đã phát hiện ra, lỗi thứ nhất là do setting hình ảnh trong phần project setting của Unity. Lỗi thứ 2 là do các transform của box setting chưa hợp lý.

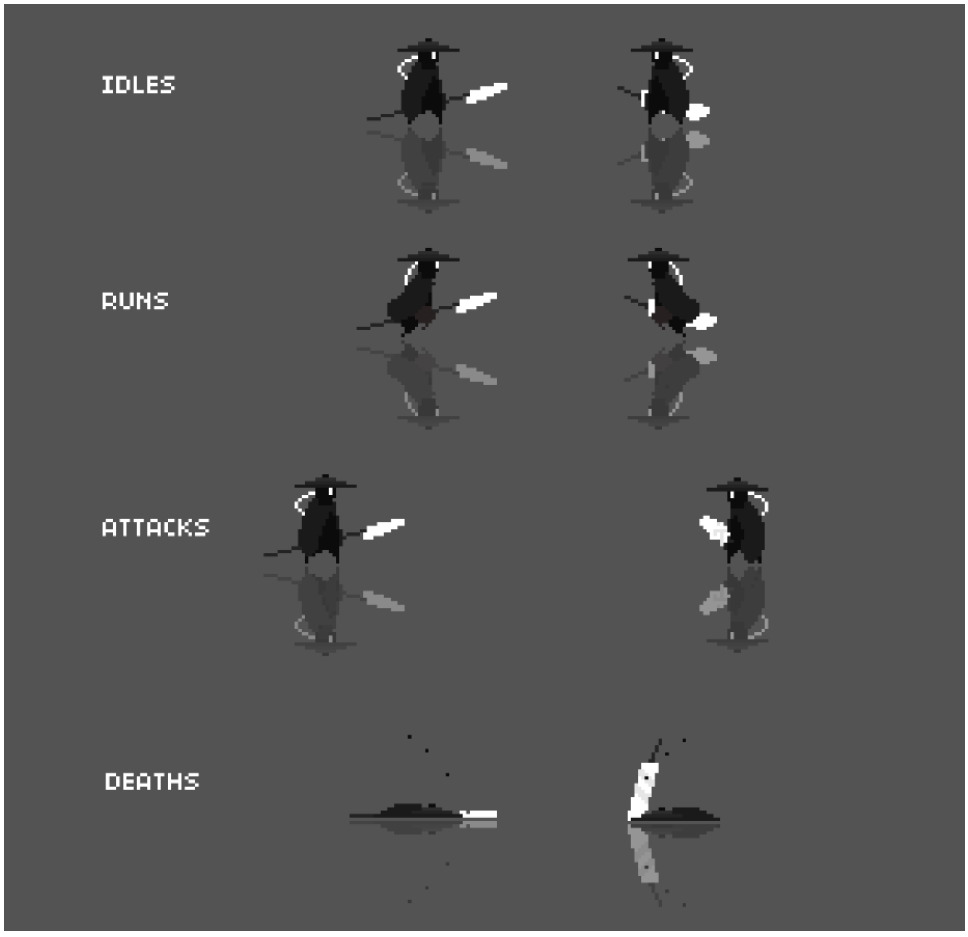
Hình 44. Cấu hình các thành phần của Hitbox

Chương 4. Hướng phát triển

1. Định hướng phát triển trong tương lai

Ở đồ án 2 này mục tiêu chính của em chính là phát triển AI cho Boss cho nên nó chưa thể hiện rõ ràng được những gì mà tựa game có thể đem lại. Tuy nhiên em cũng đã có những tính toán về việc hoàn thiện các màn chơi cũng như thêm nhân vật mà người chơi điều khiển.

Em cũng đã chuẩn bị sẵn model cho nhân vật "Demine", một nhân vật thuộc hội "Những đứa con của mặt trời". Hiện tại đang có 2 mẫu mà em chọn, lúc đấy có thể em sẽ thêm tính năng đổi nhân vật để đa dạng hơn trong quá trình chơi game.



Hình 45. Model nhân vật Demine (trong version tiếp theo của game)

Ngoài ra thì sẽ phát triển thêm hệ thống màn chơi cũng như đa dạng loại quái hơn nữa. Em hy vọng em có thể phát triển hơn tựa game ở đồ án tốt nghiệp của mình.

2. Tương lai của ngành công nghiệp game

Tương lai của ngành công nghiệp game đang hứa hẹn mang đến nhiều cơ hội và thách thức. Dưới đây là những xu hướng và dự đoán phát triển trong tương lai của ngành công nghiệp game trên thế giới và tại Việt Nam, cũng như những cơ hội và thách thức có thể đối mặt.

2.1. Xu hướng và dự đoán phát triển trong tương lai

2.1.1. Trên thế giới

Metaverse: Metaverse là một biên giới tiếp theo trong trò chơi, nơi người chơi có thể tương tác và tham gia vào một thế giới ảo đa dạng và sống động [7].

Web3: Công nghệ Web3 đang thay đổi cách người chơi tương tác với trò chơi, tạo ra môi trường phi tập trung và tăng tính tương tác giữa người chơi và nhà phát triển [7].

Mô hình doanh thu mới: Các mô hình doanh thu mới như giao dịch trong game, bán hàng trong game và quảng cáo trong game đang trở thành xu hướng phát triển trong ngành công nghiệp game [7].

2.1.2. Việt Nam

Tăng trưởng nhanh chóng: Ngành công nghiệp game Việt Nam đang có tốc độ tăng trưởng trung bình 9% mỗi năm, cao hơn trung bình của khu vực Đông Nam Á [8].

eSports: eSports, hay thể thao điện tử, đang trở thành một lĩnh vực mới trong ngành công nghiệp game, với tốc độ tăng trưởng hàng năm là 8,1% và dự kiến có 640 triệu người theo dõi môn này vào năm 2025 [8].

Hệ sinh thái game: Việt Nam cần phát triển một hệ sinh thái game mạnh mẽ để tận dụng tối đa tiềm năng của ngành công nghiệp game, bao gồm sự hỗ trợ từ

chính phủ, sự hợp tác của các doanh nghiệp và sự sáng tạo của cộng đồng lập trình game [8].

2.2. Những cơ hội và thách thức có thể đối mặt

❖ **Cơ hội:**

Tiềm năng phát triển mạnh mẽ: Ngành công nghiệp game đang có tiềm năng phát triển rất lớn, đặc biệt là trong lĩnh vực eSports và mô hình doanh thu mới [9].

Cơ hội kinh doanh và đầu tư lớn: Ngành công nghiệp game mang lại cơ hội kinh doanh và đầu tư lớn, đặc biệt là khi người chơi và người tiêu dùng ngày càng tăng và có xu hướng chi tiêu nhiều hơn cho trò chơi điện tử [9].

❖ **Thách thức:**

Xây dựng hệ sinh thái game mạnh mẽ: Một trong những thách thức lớn của ngành công nghiệp game là xây dựng một hệ sinh thái game mạnh mẽ, bao gồm việc phát triển cộng đồng game, hỗ trợ cho các nhà phát triển game độc lập, và tạo ra môi trường thuận lợi để các công ty game phát triển [9].

Thu hút nhân tài: Ngành công nghiệp game cần thu hút và giữ chân nhân tài, bao gồm các nhà phát triển game, nhà thiết kế đồ họa, nhà sản xuất âm thanh và các chuyên gia khác. Để làm được điều này, cần có chính sách hỗ trợ và môi trường làm việc thuận lợi [9].

Cạnh tranh trên thị trường quốc tế: Để thành công trên thị trường quốc tế, ngành công nghiệp game cần tạo ra những sản phẩm chất lượng cao, cạnh tranh với các công ty game lớn trên thế giới. Điều này đòi hỏi đầu tư vào nghiên cứu và phát triển, đồng thời tạo ra những trò chơi độc đáo và hấp dẫn [9].

KẾT LUẬN

Đồ án cơ bản đã đạt được những mục tiêu đề ra:

- ❖ Game chạy đúng với những gì mà em đặt ra từ đầu.
- ❖ Tuân thủ các nguyên tắc thiết kế, giúp game dễ dàng hơn trong việc bảo trì và cập nhật.
- ❖ Mô hình MVC tách rời logic xử lý, dữ liệu và giao diện cũng giúp game dễ dàng bảo trì và cập nhật hơn.

Tuy nhiên, do thời gian tìm hiểu có hạn và kinh nghiệm, kiến thức còn hạn chế nên đôi khi vẫn xảy ra một số lỗi không mong muốn. Và do không có chuyên môn về mặt đồ họa, âm thanh nên game vẫn còn nhiều thiếu sót về mặt giao diện, hiệu ứng, ...

Tổng kết lại, qua quá trình phát triển game, em đã tích lũy được nhiều kiến thức và kỹ năng quan trọng trong lĩnh vực này. Em tin rằng đồ án này đã đóng góp đáng kể vào việc hiểu và áp dụng các khái niệm, kỹ thuật và quy trình phát triển game. Hy vọng rằng ý tưởng của trò chơi sẽ là đủ tính hấp dẫn cũng như tạo được sự tò mò cho thầy cô để chờ đợi các bản update tiếp theo của game.

Cuối cùng, em xin chân thành cảm ơn sự hỗ trợ và đồng hành của Ts. Đoàn Duy Trung trong quá trình thực hiện đồ án lần này.

TÀI LIỆU THAM KHẢO

[1] https://www.liquidweb.com/insights/video-game-statistics/

[2] https://www.pocketgamer.biz/comment-and-opinion/82975/unity-im-done/

[3] https://www.gamespot.com/articles/game-developers-are-frustrated-with-unitys-new-predatory-business-model/1100-6517689/

[4] https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_ng%C6%B0%E1%BB%9Di_d%C3%B9ng

[5] https://colorme.vn/blog/parallax-la-gi-nhung-website-doc-dao-voi-hieu-ung-parallax

[6] https://fptshop.com.vn/tin-tuc/giai-tri/hitbox-155940

[7] https://vneconomy.vn/techconnect//hai-xu-huong-cong-nghe-dinh-hinh-tuong-lai-cua-nganh-cong-nghiep-tro-choi.htm

[8] https://www.tinnhanhchungkhoan.vn/co-hoi-vuon-minh-tro-thanh-nganh-cong-nghiep-game-ty-usd-post332803.html

[9] https://dangcongsan.vn/khoa-hoc/nganh-cong-nghiep-game-viet-nam-con-nhieu-tiem-nang-phat-trien-644633.html