NGUYỄN XUÂN ĐẠT

KỸ THUẬT PHẦN MỀM

BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC

NGÀNH KỸ THUẬT PHẦN MỀM

PHÁT TRIỂN ỨNG DỤNG GAME "MY'S HEART" VỚI UNITY ENGINE

CBHD: TS. Lê Xuân Hùng

Sinh viên: Nguyễn Xuân Đạt

Mã số sinh viên: 2021602969

Hà Nội – Năm 2024

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC CÁC TỪ VIẾT TẮT

Ký hiệu chữ viết tắt	Chữ viết đầy đủ
AI	Artificial Intelligence (Trí tuệ nhân tạo)
UI	User Interface (Giao diện người dùng)
2D	Two-Dimensional
3D	Three-Dimensional

LỜI CẢM ƠN

Lời đầu tiên em rất chân thành cảm ơn thầy **TS. Lê Xuân Hùng** cùng toàn thể giảng viên trường Đại học Công Nghiệp Hà Nội đã hỗ trợ và giúp đỡ em trong suốt thời gian học tập tại đây. Sau quãng thời gian học tập và nghiên cứu tại trường, em đã không chỉ nhận được kiến thức, kinh nghiệm từ các thầy, cô, các anh chị khóa trước mà còn giúp em hoàn thiện hơn về bản thân, định hình được về công việc mơ ước của mình trong tương lai sắp tới.

Cùng với đó, em muốn gửi lời cảm ơn đặc biệt đến thầy **TS. Lê Xuân Hùng**; Người đã hỗ trợ, chỉ dẫn và đưa ra lời khuyên cho em trong suốt quá trình thực hiện đồ án tốt nghiệp của mình. Cùng với việc đưa ra các chỉ dẫn với kiến thức chuyên sâu, thầy còn có rất nhiều lời khuyên bổ ích không chỉ cho quá trình làm đồ án, mà còn là những lời khuyên về con đường sự nghiệp trong tương lai. Qua đó em cũng đó có thêm góc nhìn khác về lĩnh vực công nghệ thông tin và các chuyên ngành của công nghệ thông tin.

Em rất mong nhận được ý kiến nhận xét và đóng góp từ thầy cô, để từ đó có thể hoàn thiện hơn nữa dự án của mình. Em hy vọng dự án của mình không chỉ dừng lại ở mức đồ án tốt nghiệp. Một lần nữa em xin chân thành cảm ơn thầy cô trường Đại học Công Nghiệp Hà Nội, chúc thầy cô cùng nhà trường phát triển vững mạnh, là một trong những nơi trọng điểm đào tạo nhân tài cho quốc gia.

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

1. Lý do chọn đề tài

Những năm gần đây, ngành công nghiệp game đang dần phát triển lớn mạnh ở cả trong và ngoài nước. Gần đây nhất, bom tấn "Black Myth Wukong" đã tạo nên một làn sóng đặc biệt trong ngành phát triển game; việc đưa các yếu tố về văn hoá vào game đã giúp "Black Myth Wukong" đã góp phần tạo nên sự đặc sắc và thú vị cho tựa game này. Bên cạnh đó việc các thư viện game và game engine ngày càng trở nên mạnh mẽ đã góp phần thúc đẩy sự bùng nổ của ngành công nghiệp game. Một trong những game engine phổ biến nhất là "Unity Engine", công cụ vô cùng mạnh mẽ không chỉ trong việc xây dựng game mà còn là các ứng dụng 3D, thực tế ảo, … Các tiện ích mà Unity cung cấp sẵn cho các nhà phát triển cùng với việc hỗ trợ đa nền tảng, dễ dàng trong quá trình học tập và phát triển đã khiến Unity trở nên phổ biến như hiện nay.

Thế nhưng, cùng với sự phát triển của xã hội nhịp sống con người ngày càng hối hả, chúng ta dần bị cuốn vào vòng xoáy "cơm - áo - gạo - tiền", thời gian để thư giãn có vẻ như ngày càng ngắn lại, việc dành một khoảng thời gian lớn để chơi game đối với nhiều người ngày càng khó khăn. Tựa game "My's Heart" sinh ra để giúp giải tỏa đi phần nào căng thẳng sau ngày dài làm việc. Là một dạng biến thể của thể loại game Tower defense, My's Heart sẽ mang tới một lối chơi đơn giản, tiết tấu vừa phải để có thể trải nghiệm nó ngay cả trong quá trình di chuyển trên các phương tiện công cộng.

2. Mục đích đề tài

- Nghiên cứu về Game Engine nói chung và Unity Engine nói riêng. Ứng dụng Unity Engine vào phát triển game 2D.
- Xây dựng lên trò chơi My's Heart với phong cách đồ họa pixel, cùng với lối chơi và phong cách chơi độc lạ, nhưng vẫn giữ được sự đơn giản của trò chơi.
- Ứng dụng các thành phần về xử lý va chạm, hệ thống vật lý, … được cung cấp với Unity vào việc phát triển game.

- Tối ưu hóa các tài nguyên đồ họa, âm thanh của trò chơi, áp dụng design pattern vào quá trình viết kịch bản cho game từ đó giúp game hoạt động mượt mà trên nền tảng di động.

3. Đối tượng nghiên cứu

- Nghiên cứu về Unity và Game Engine.
- Ứng dụng Unity để phát triển game 2D.
- Tìm hiểu về hệ sinh thái của Unity.
- Phân tích và thiết kế các tính năng của trò chơi, bao gồm quản lý tài nguyên, tương tác người chơi, và môi trường game.

4. Phạm vi nghiên cứu

- Nghiên cứu tổng quan về Unity Engine và các công cụ do unity cung cấp.
- Nghiên cứu các thuật toán va chạm và cách chúng được áp dụng trong môi trường game.
- Nghiên cứu cách thức quản lý tài nguyên và tối ưu hóa đồ họa của game trên nhiều nền tảng.
- Phát triển tựa game My’s Heart trên nền tảng Android cùng với khả năng mở rộng sang IOS trong tương lai.

5. Ý nghĩa khoa học và thực tiễn của đề tài

- Khoa học: Đề tài tập trung vào việc tìm hiểu game engine, quy trình phát triển game, cách viết kịch bản, tối ưu và quản lý tài nguyên của trò chơi. Cùng với đó là vận dụng được các thành phần cơ bản (Xử lý va chạm, hệ thống vật lý, …) do Unity cung cấp vào phát triển game.
- Thực tiễn: My’s Heart không chỉ là một trò chơi, mà còn là một công cụ học tập thực tế cho sinh viên, giúp làm quen với toàn bộ quy trình phát triển game từ thiết kế, lập trình đến thử nghiệm. Trò chơi này còn có tiềm năng mở rộng sang nhiều nền tảng khác nhau, mang lại trải nghiệm giải trí đơn giản nhưng đầy cuốn hút.

6. Kết quả đạt được

- Hoàn thiện trò chơi My’s Heart trên nền tảng Unity Engine.
- Hiểu được về quy trình phát triển game, cách các game engine hỗ trợ các nhà phát triển.

- Trò chơi có thể chạy ổn định trên nền tảng Android với khả năng mở rộng sang các nền tảng khác trong tương lai.

7. Bố cục báo cáo

Chương 1: Tổng quan về Unity và Game Engine.

Chương 2: Thiết kế và xây dựng trò chơi My’s Heart bao gồm hệ thống điều khiển, quản lý vật phẩm, xây dựng các màn chơi và tương tác giữa trò chơi và người chơi.

Chương 3: Thử nghiệm và đánh giá trò chơi, bao gồm quá trình tối ưu hóa hiệu suất và các tính năng được triển khai trong game, ưu và nhược điểm và hướng phát triển của trò chơi.

CHƯƠNG 1. TỔNG QUAN VỀ GAME ENGINE

Trò chơi video là một hình thức giải trí kỹ thuật số phổ biến, nơi người chơi tương tác với các thiết bị như cần điều khiển, bàn phím, chuột, và nhận phản hồi trực quan trên màn hình. Các trò chơi này thường đi kèm với âm thanh sống động, tạo ra một trải nghiệm chân thực và hấp dẫn. Một số trò chơi còn tích hợp phản hồi xúc giác, giúp người chơi cảm nhận được các rung động và tác động trong trò chơi, tăng thêm phần thú vị và thực tế.

Trò chơi video có thể được phân loại theo nhiều nền tảng khác nhau như arcade, console, PC, và di động. Mỗi nền tảng đều có những đặc điểm riêng biệt và mang lại những trải nghiệm khác nhau cho người chơi. Ngành công nghiệp trò chơi video đã phát triển mạnh mẽ trong những năm gần đây, nhờ vào sự tiến bộ của các công nghệ mới như thực tế ảo (Virtual Reality) và điện toán đám mây. Những công nghệ này không chỉ nâng cao chất lượng đồ họa và âm thanh mà còn mang đến những trải nghiệm đa dạng và phong phú hơn cho người chơi.

Các thể loại trò chơi video rất đa dạng, bao gồm hành động, phiêu lưu, chiến thuật, nhập vai, thể thao, và đối kháng. Mỗi thể loại đều có những đặc trưng và lối chơi riêng, thu hút người chơi với những sở thích khác nhau. Ví dụ, Defense Grid: The Awakening là một trò chơi thuộc thể loại tower defense trên PC. Trò chơi này nổi bật với đồ họa đẹp mắt và âm thanh sống động, mang lại trải nghiệm hấp dẫn và đầy thử thách cho người chơi. Bên cạnh đó, còn có rất nhiều trò chơi khác thuộc các thể loại khác nhau, từ những trò chơi hành động nhanh nhẹn đến những trò chơi chiến thuật đòi hỏi sự tính toán và lập kế hoạch kỹ lưỡng.

Ngoài ra, ngành công nghiệp trò chơi video còn chứng kiến sự ra đời của nhiều xu hướng mới, như trò chơi trực tuyến nhiều người chơi và trò chơi di động.

Những xu hướng này đã mở rộng phạm vi và tầm ảnh hưởng của trò chơi video, thu hút một lượng lớn người chơi từ khắp nơi trên thế giới. Với sự phát triển không ngừng của công nghệ, tương lai của ngành công nghiệp trò chơi video hứa hẹn sẽ còn nhiều điều thú vị và bất ngờ.

Một yếu tố quan trọng góp phần vào sự phát triển này là các game engine. Game engine là phần mềm cung cấp các công cụ và tính năng cần thiết để phát triển trò chơi điện tử. Nó bao gồm các thành phần như đồ họa, âm thanh, vật lý, và trí tuệ nhân tạo. Một số game engine phổ biến hiện nay bao gồm Unity, Unreal Engine, và Godot. Các game engine này giúp các nhà phát triển tiết kiệm thời gian và công sức trong việc tạo ra các trò chơi chất lượng cao.

1.1 Game Engine là gì?

Công nghệ game engine, hay còn được nhiều người gọi là Phần mềm game. Đây là một phần mềm chuyên dụng dùng để xây dựng và thiết kế game được nhiều người lựa chọn. Bạn có thể hình dung một cách đơn giản, công nghệ game này cung cấp cho các nhà lập trình game những nguyên liệu cơ bản để họ sẽ dễ dàng phát triển game của mình.

Một game engine thường sẽ bao gồm có kết xuất đồ họa cho các hình ảnh thiết kế 2D hay 3D, các công cụ vật lý, hình ảnh hoạt hình, trí tuệ nhân tạo cho các nhân vật, phân luồng, tạo các dòng dữ liệu xử lý, quản lý bộ nhớ game, dựng các ảnh đồ thị,… Từ 1 phần mềm game engine các bạn có thể phát triển ra rất nhiều các tựa game khác nhau. Đây là một giải pháp giúp các bạn tiết kiệm nhiều thời gian và chi phí cho nhà sản xuất.

Hình 1.1 Công nghệ game engine là phần mềm game

1.2 Công nghệ game engine ra đời nhằm mục đích gì?

Công nghệ game engine ra đời nhằm cung cấp cho những nhà phát triển game một bộ công cụ nền để dễ dàng phát triển và có thể tái sử dụng chúng từng phần. Từ một phiên bản game engine, người ta có thể sẽ phát triển thêm vô số các phiên bản game khác nhau. Nó đã giúp các nhà lập trình game giải quyết bài toán phát triển khó nhằn và tốn kém nhiều thời gian, công sức cũng như chi phí.

Đặc biệt trong bối cảnh ngành công nghiệp game hiện nay – một ngành được đánh giá là vô cùng cạnh tranh, và có tốc độ phát triển được tính theo cấp số nhân, thì những sản phẩm mang trí tuệ cao lại mang tính sống còn. Game engine ra đời với mục tiêu giúp các công ty giải quyết các yêu cầu cấp thiết về vấn đề giảm thiểu chi phí và sáng tạo game, để từ đó giúp giảm giá thành và tiếp tục duy trì sức hút của game đối với người tiêu dùng.

Hình 1.2 Công nghệ game ra đời cung cấp bộ công cụ nền

1.3 Giới thiệu về Unity Engine

- **Giới thiệu**

Unity là game engine đa nền tảng được phát triển bởi Unity Technologies. Lần đầu tiên được công bố chạy trên hệ điều hành OS X, tại Apple's Worldwide Developers Conference vào năm 2005, giờ đã mở rộng 27 nền tảng. Từ trò chơi đến phát triển công nghiệp, Game Unity có tất cả các công cụ cần để hiện thực hóa thành công tất cả các dự án 3D thời gian thực của bạn.

- **Ứng dụng**
    - Lĩnh vực giải trí
        - Tạo trò chơi: Unity mang đến cho chúng ta sự tự do để tạo ra các trò chơi với thể loại và phong cách vô cùng đa dạng, từ nghệ thuật pixel 2D đến đồ họa 3D cao cấp. Tùy theo cách thức và ý tưởng phát triển.

- Đa nền tảng khi sử dụng Game Unity: Unity có thể giúp trò chơi của bạn hoạt động trên máy tính PC. PlayStation, Xbox, Meta Quest, VisionOS, iOS, Android, Nintendo, Switch™, Web, v.v. Game Unity thúc đẩy sự thành công của trò chơi dành cho thiết bị di động bằng các công cụ tạo mẫu nhanh, tối ưu hóa hiệu suất và tăng doanh thu, nâng cao mức độ tương tác của người chơi trong suốt vòng đời của trò chơi. Nhanh chóng linh hoạt cho các lập trình viên. Cho dù bạn bắt đầu từ con số không đến khi bạn là một nhà phát triển chuyên nghiệp.
- Multiplayer với Game Unity: Chạy hoặc lưu trữ các trò chơi multiplayer chơi quy mô lớn bằng các công cụ và dịch vụ của Unity, ngay cả bên ngoài Unity Engine và việc áp dụng các sever linh hoạt: Photon Engine, Unity Networking.

- Lĩnh vực khác
  - Game Unity được sử dụng trong Ô tô vận tải: Vận hành linh hoạt cho các hệ thống trên ô tô vận tải. Đánh giá thiết kế, giảm lỗi cho kỹ sư dịch vụ hiện trường hay tạo ra trải nghiệm hấp dẫn trên xe thông qua giao diện người-máy (HMI), hãy giúp quy trình làm việc trên ô tô của bạn hiệu quả hơn bằng công nghệ 3D thời gian thực.
  - Game Unity được sử dụng trong kiến trúc, xây dựng kỹ thuật: Với công nghệ 3D thời gian thực.Với việc liên kết dữ liệu qua nhiều nền tảng mục tiêu, giảm bớt sự phức tạp trong hoạt động và cải thiện sự hợp tác của các bên liên quan để đưa ra quyết định sáng suốt hơn.

- Game Unity được sử dụng trong chế tạo máy, năng lượng: Nâng cao cách bạn thiết kế, xây dựng và tiếp thị các bộ phận cũng như sản phẩm phức tạp bằng nền tảng hỗ trợ 3D thời gian thực trong ngành công nghiệp 4.0.

- **Ưu và nhược điểm**
  - Ưu điểm
    - Unity có một cộng đồng rất lớn về asset và plugin – trong đó có rất nhiều resources free và có nhiều thứ rất đáng bỏ tiền
    - Unity có bộ công cụ rất trực quan và editor có thể mở rộng bằng plugins
    - Unity hỗ trợ rất nhiều định dạng asset khác nhau và có thể tự động chuyển đổi đến định dạng phù hợp nhất với nền tảng thích hợp
    - Unity hỗ trợ nhiều nền tảng: di động, desktop, web và console
    - Việc triển khai đến các nền tảng khác nhau cũng khá dễ quản lý
    - Bạn có thể dễ dàng xây dựng một game 3D mà không cần cấu hình quá phức tạp
    - Unity bản free có hầu hết những tính năng quan trọng nhất
    - Unity bản trả phí phù hợp với các developer chuyên nghiệp
  - Nhược điểm
    - Việc hợp tác rất khó khăn. Unity sử dụng một server asset rất hiệu quả để hỗ trợ các đội phát triển phần mềm hợp tác với nhau. Tuy nhiên nếu bạn không sử dụng nó thì việc chia sẻ code và asset giữa các thành viên trong team có thể gây ra những vấn đề

nghiêm trọng. Lựa chọn tốt nhất là sử dụng một số công cụ quản lý resource bên ngoài nhưng có một vài binary file không thể merge được với nhau và việc cập nhật asset có thể gây nên một số vấn đề trong scenes, mất kết nối đến script và các đối tượng khác

- Hiệu năng chưa thật sự ấn tượng cho đến khi Unity 5 ra mắt. Unity 5 đã chạy hầu hết trên một luồng duy nhất và hầu như không sử dụng thêm 1 nhân phụ nào trên các thiết bị di động. Bộ biên dịch chưa được tối ưu tốt cho các bộ xử lý ARM trên hầu hết các thiết bị di động. Để giải quyết vấn đề này thì Unity đã quyết định transpile (source-to-source compiler) sang C++ và sử dụng LLVM để tối ưu được nhiều hơn thay vì giải quyết vấn đề này trực tiếp trên các phiên bản sau này
- Mã nguồn của engine không được công bố kể cả cho những người dùng chấp nhận trả tiền. Điều đó có nghĩa là nếu bạn gặp một bug với engine bạn phải chờ Unity fix chúng trong các bản tiếp theo. Điều này có thể gây nên những vấn đề nghiêm trọng với project của bạn

- **Các tựa game được phát triển bằng Unity**
  - Subway Surfers

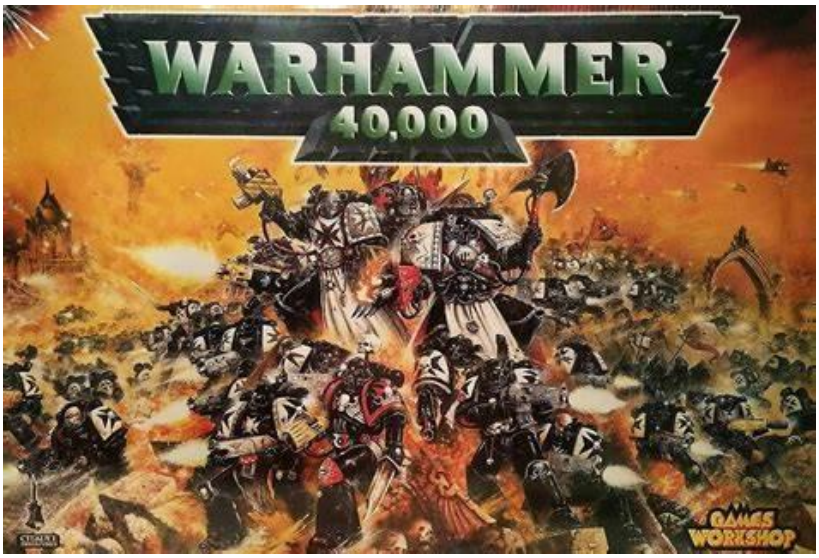
Đây là một trò chơi chạy vô tận, trong đó bạn sẽ vào vai một nhân vật chạy qua các chướng ngại vật khác nhau với nhiều thử thách. Được phát triển và tồn tại hơn 10 năm đến nay, SYBO đã sử dụng Game Unity để xây dựng Subway Surfers từ giấc mơ của hai sinh viên trở thành một trong những trò chơi được tải xuống nhiều nhất trong lịch sử. Đồng thời,

SYBO đã giúp Unity tinh chỉnh công cụ của mình bằng dữ liệu trải nghiệm người dùng phong phú và kỹ thuật tiên tiến nhất.

Hình 1.3 Tựa game Subway Surfers

- Warhammer 40000

Đây là tựa game do Games Workshop sản xuất và sử dụng Engine Game Unity. là trò chơi chiến tranh thu nhỏ phổ biến nhất trên thế giới. Nó phổ biến các quốc gia như Anh, Mỹ, Nhật bản và một số nước châu u. Trong đó mô tả thiên niên kỷ thứ 41, khi các phe phái tham chiến từ các nền văn minh cổ đại và các đế chế mới nổi chiến đấu với những trận chiến bất tận. Nhân loại đứng một mình, bị bao vây tứ phía bởi kẻ dị giáo, người đột biến và người ngoài hành tinh với các vũ khí hiện đại. Khi đó mọi kẻ thù không có lòng thương xót và chiến đấu liên tục không có thời gian nghỉ ngơi.



Hình 1.4 Tựa game Warhammer 40000

- Greak: Memories of Azur

Là trò chơi một người chơi cuộn màn hình bên với hình ảnh vẽ tay chân thực. Bạn sẽ vào vai anh chị em: Greak, Adara và Raydel để hướng dẫn họ đi qua vùng đất Azur. Bằng cách sử dụng điều khiển luân phiên giữa chúng và sử dụng khả năng độc đáo của chúng để thoát khỏi cuộc xâm lược của Urlag. Nhà phát triển sử dụng công cụ Game Unity rất linh hoạt.

Hình 1.5 Tựa game Greak: Memories of Azur

- Among Us

Một trò chơi giống Mafia lấy cảm hứng từ The Thing (1982), trò chơi lấy bối cảnh bên ngoài không gian vũ trụ, ở đó người chơi đóng một trong hai vai trò: đa số là các thành viên phi hành đoàn (Crewmates) và một số ít là những kẻ giả mạo (Impostors).Phi hành đoàn có mục tiêu tìm ra và loại bỏ tất cả kẻ giả mạo cũng như hoàn thành các nhiệm vụ xung quanh bản đồ, còn phía giả mạo có mục tiêu giết các thành viên phi hành đoàn mà không bị ai phát hiện. Trò chơi được InnerSloth phát triển và được phát hành vào ngày 15 tháng 6 năm 2018 sử dụng công cụ Game Unity.

Hình 1.6 Tựa game Among Us

1.4 Những điểm nổi bật trong Unity 6

Phiên bản mới nhất của Unity là phiên bản Unity 6 (6000.0.23f1) LTS {Tính đến ngày 24/10/2024}

Việc lấy tên “Unity 6” là để đánh dấu một bước tiến lớn trong sự phát triển của công cụ này. Unity Technologies đã quyết định chuyển sang Unity 6 để phản ánh những cải tiến đáng kể và các tính năng mới mà phiên bản này mang lại. Đây

là phiên bản ổn định và hiệu suất cao nhất từ trước đến nay, với nhiều tháng thử nghiệm và tối ưu hóa trước khi phát hành chính thức.

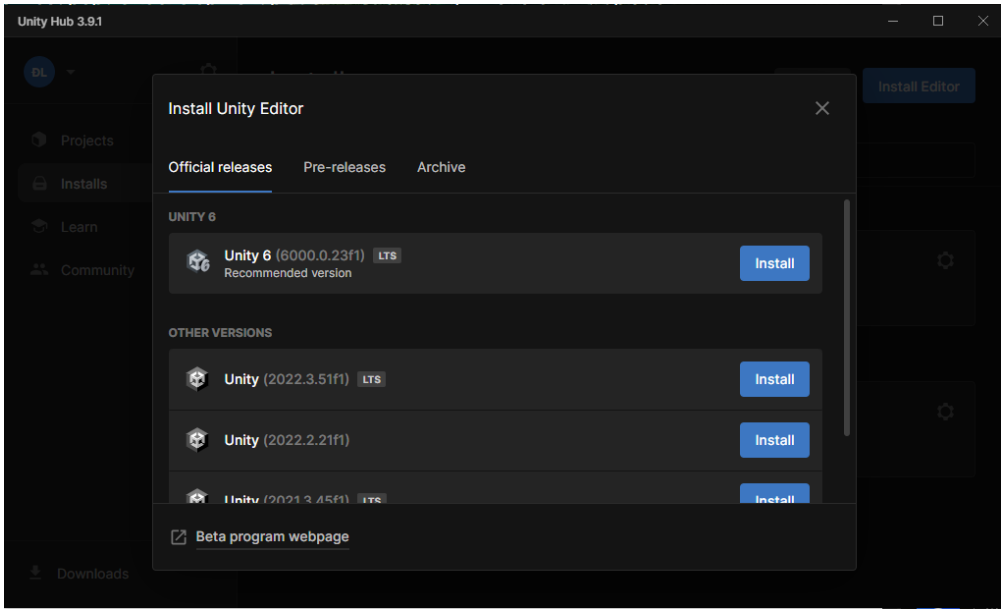
Unity 6 bao gồm tất cả các tính năng và chức năng từ các phiên bản trước như Unity 2023.1 và Unity 2023.2, cùng với các cải tiến mới trong đồ họa, hiệu suất, công cụ AI, và hỗ trợ cho các nền tảng mới như visionOS 21. Việc đặt tên này cũng giúp người dùng dễ dàng nhận biết và phân biệt với các phiên bản trước đó, đồng thời nhấn mạnh sự đổi mới và tiến bộ của Unity trong ngành công nghiệp phát triển trò chơi.

Hình 1.7 Màn hình khởi động Unity 6

- Cập nhật mới:
    - Cải tiến đồ họa và hiệu suất: Phiên bản này cung cấp các tùy chọn ánh sáng mới, nâng cao khả năng kết xuất và hiệu suất tổng thể.

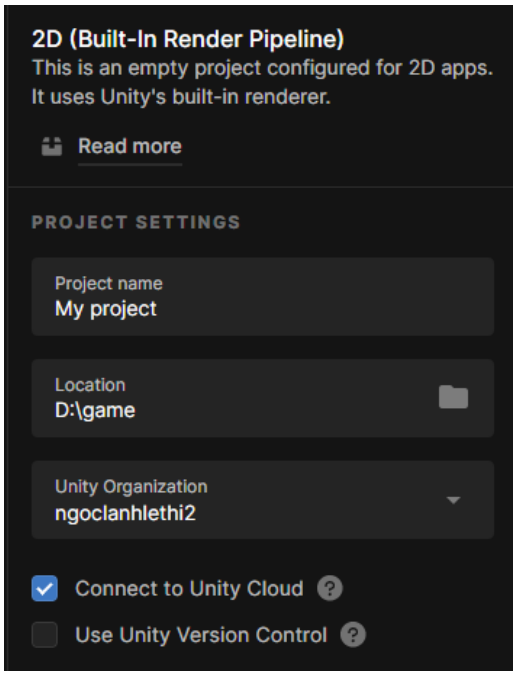
- Công cụ tạo game đa người chơi: Các công cụ tạo game đa người chơi được nâng cấp, giúp việc phát triển các trò chơi trực tuyến trở nên dễ dàng và hiệu quả hơn.
- Hỗ trợ mở rộng cho web và XR (Extended Reality): Phiên bản này mở rộng các tính năng cho web và thực tế mở rộng (XR), bao gồm thực tế ảo (VR - Virtual Reality), thực tế tăng cường (AR) và thực tế hỗn hợp (MR - Mixed Reality).
- Công cụ AI: Các công cụ trí tuệ nhân tạo (AI) được cải tiến, hỗ trợ phát triển các tính năng AI phức tạp trong trò chơi.
- Hỗ trợ visionOS 2: Unity 6000.0.23f1 bao gồm các tính năng PolySpatial mới như nhiều khối lượng, blendshapes và mục tiêu kết xuất stereo, cùng với hỗ trợ cho các tính năng của visionOS 2.

- Cài đặt và tạo dự án với Unity 6
  - Trước khi bắt đầu tải Unity 6, cần phải cài đặt Unity Hub từ trang chủ của Unity (Download the Unity Hub | Unity)



Hình 1.8 Giao diện cài đặt Unity Editor của Unity Hub

- Lựa chọn phiên bản Unity 6000.0.22f1 (hoặc 6000.0.23f1 – Phiên bản mới nhất)
  - Unity đã cung cấp rất nhiều Platforms cho việc xây dựng các mẫu Game và ứng dụng khác nhau tùy thuộc vào mục tiêu của dự án.
- Bên cạnh các template để tạo nhanh các dự án, Unity 6 còn hỗ trợ Unity Version Control – Qua đó giúp các nhà phát triển có thể quản lý các phiên bản và thay đổi của dự án một các khoa học và dễ dàng.



Hình 1.9 Version Control trên Unity 6

CHƯƠNG 2. GIỚI THIỆU GAME MY'S HEART

2.1 Giới thiệu tổng quan

2.1.1 Thông tin chung

"Ma trơi"; Những ngọn lửa lập lòe trong đêm tối, rượt đuổi theo những ai đang cố gắng chạy trốn khỏi chúng. Hình ảnh ấy đã tạo nên nguồn cảm hứng chính để xây dựng nên tựa game My's Heart. Trong trò chơi này, bạn sẽ sử dụng các trụ đá kỳ lạ nằm xung quanh, điều khiển ngọn lửa bí ẩn kia để từ đó tiêu diệt các kè t5huf đang rình mò, nhăm nhe phá hủy bức tượng nữ thần, nơi đang chứa đựng trái tim của "Dị giới" này. Bên cạnh việc bảo vệ nữ thần, bạn sẽ thu thập các đồng tiền đặc biệt thứ giúp bạn tăng cường sức mạnh thắp sáng lại những trụ đá phòng thủ; Thứ sẽ khiến bạn cảm thấy dễ thở hơn trong trận chiện vô tận này. Hãy nhớ, "Không chỉ có mỗi bạn ngày càng mạnh hơn, những kẻ gớm ghiếc kia cũng vậy."

Trong trận chiến, sức mạnh của bạn và cả lũ quái vật sẽ ngày càng tăng lên, chung không chỉ mạnh hơn, khó bị đánh bại hơn mà còn đông hơn và hung hăng hơn nữa. Hãy cố gắng tăng cường sức mạnh của bạn thật nhanh nếu không muốn thất bại 1 cách nhanh chóng. Bên cạnh đó, trò chơi sẽ cung cấp cho bạn các vật phẩm hỗ trợ từ hồi phục, phòng thủ, cùng với các kỹ năng đặc biệt. Những tòa tháp phòng thủ bằng chính nguồn năng lượng từ địa linh xung quanh cũng phần nào giúp bạn đỡ cảm thấy bị thất thế.

Mục tiêu của trò chơi là việc bạn cần phòng thủ lâu nhất có thể. Không đơn giản chỉ là việc di chuyển hỏa cầu, quái vật sẽ liên tục tấn công từ mọi hướng, hãy chú ý quan sát và điều khiển hỏa cầu 1 cách khéo léo để có thể tiêu diệt lũ quái vật trước khi chúng chạm tay được vào nữ thần. Nếu như thanh máu của nữ thần tụt về không, trò chơi sẽ phải tạm dừng, hãy đảm bảo rằng sức mạnh và trang bị

của bạn đã đầy đủ để khởi động 1 cuộc chiến mới. Thế nhưng đừng vội chủ quan, lũ quái vật lần này đã mạnh mẽ hơn trước hãy hết sức cẩn trọng.

Là một dạng biến thể của dòng game Tower Defense, tựa game mang đến cho người chơi 1 phong cách chơi không quá phức tạp, qua đó giúp người chơi có thể giải trí trong những quãng thời gian ngắn khi đang di chuyển trên phương tiện công cộng, giờ giải lao, … Hy vọng rằng bạn sẽ cảm thấy thoải mái hơn phần nào sau khi trải nghiệm tựa game, lấy lại năng lượng tích cực để tiếp tục hành trình học tập và làm việc của mình.

2.1.1 Thể loại và yếu tố game

My's Heart là 1 tựa game tập trung vào việc xây dựng phòng thủ và tiêu diệt quái vật xâm lăng. Có thể xem My's Heart là một thể loại biến thể từ dòng game "*Tower defense",* bởi lẽ thay vì tập trung vào việc phòng thủ bằng các công trình tĩnh, tựa game này kết hợp giữa các công trình phòng thủ tĩnh và động. Người chơi sẽ được tự tay điều khiển đường đi của hỏa cầu tấn công kẻ địch.

Các yếu tố mà game mang lại:

- ✓ Khả năng quan sát và ứng biến: Quát vật, chúng ở khắp nơi tấn công liên tục và vây kín xung quanh bạn. Phải nhớ chú ý quan sát, nhanh tay nhanh mắt điều khiển hỏa cầu tiêu diệt chúng càng nhanh càng tốt. Việc ứng biến nhanh nhạy cũng là một phần quan trong, trong lúc bạn đang phân vân xem nên điều hướng cho hỏa cầu hay sử dụng kỹ năng hỗ trợ thì lũ quái vật đã tiến gần hơn tới tượng nữ thần. Hãy chắc rằng bạn đưa ra các thao tác sáng suốt.
- ✓ Độ tập trung, độ khó: Tương tự như tựa game Temple Run, chỉ cần một vài giây lơ là có thể khiến bạn mất mạng, đôi lúc chỉ vài giây mất tập

trung bạn sẽ phải bắt đầu 1 cuộc chiến mới. Truy nhiên với việc có thanh máu lớn cùng các vật phẩm bổ trợ bạn sẽ không mất đi quá nhiều trong lúc bạn quay đi để kiểm tra điểm dừng xe buýt hay tiếng chuông giờ vào học. Với độ khó ngày càng tăng, bạn sẽ cần bình tĩnh để xử lý với rất nhiều quái vật đang vây quanh mình. Hoảng loạn và sợ hãi chỉ làm cho độ chính xác của bạn bị giảm xuống.

✓ Kỹ năng điều khiển: Với việc điều khiển hỏa cầu thông qua các trụ đá ở các vị trí cố định, bạn cần phải có thao tác tay đủ nhanh nhạy để chuyển hướng hỏa cầu 1 cách hợp lý. Việc nắm bắt được đường đi của hỏa cầu giúp bạn tối ưu được thời gian di chuyển của mình.

2.1.2 Đối tướng chơi

Game My's Heart tập trung chính vào thao tác tay để di chuyển quả cầu lửa 1 cách khéo léo, mong rằng tựa game này có thể hướng tới các người chơi từ 12 tuổi trở lên.

Tựa game sẽ mang lại những phút giây thư giãn cho cả trẻ em và người lớn. Bên cạnh đó game còn giúp người chơi phần nào luyện tạo khả năng phản xạ, khả năng bao quát vấn đề khi những con quái vật sẽ liên tục xuất hiện và tấn công 360^0 xung quanh bạn và còn cả khả năng ước lượng khoảng cách từ đó đưa ra các phương án điều khiển 1 cách tối ưu hơn. Kết hợp cùng các yếu tố khác, hy vong tựa game sẽ góp phần nào tăng khả năng tập trung, tư duy, ước lượng khoảng cách cho các bạn nhỏ.

Tuy vậy mục tiêu chính của tựa game vẫn là lứa tuổi thanh thiếu niên, người làm việc văn phòng. Những người có nhiều bộn bề trong cuộc sống, người cần thứ gì đó giải trí trong lúc chờ và di chuyển trên các phương tiện công cộng, giờ nghỉ giải lao ngắn tại văn phòng, ….

2.1.3 Nền tảng

- Nền tảng phát triển:

  My’s Heart được phát triển trên nền tảng Unity Engine (công cụ phát triển game phổ biến nhất hiện nay với ngôn ngữ C#), Unity cho phép phát triển ứng dụng trên đa nền tảng. Unity cung cấp rất nhiều công cụ mạnh mẽ về đồ họa, hệ thống vật lý, âm thanh, tương tác người dùng, …. Unity cũng là 1 công cụ tối ưu cho các ứng dụng 3D không chỉ với game mà còn là các hệ thống thực tế ảo, và thực tế ảo tăng cường.

  Cùng với sự phát triển, các phiên bản về sau của Unity không những ngày càng tối ưu cho hiệu suất xử lý mà còn cung cấp thêm nhiều công cụ phát triển khác.

- Nền tảng thiết bị:

  Hiện tại, My’s Heart đang phát triển để hướng tới nền tảng ‘mobile’, với mục tiêu ngắn hạn là hệ điều hành Android. Việc này giúp người chơi có thể trải nghiệm game ở bất cứ nơi nào. Với việc Unity tối ưu về hiệu suất xử lý đồ họa sẽ giúp tựa game hướng tới số lượng thiết bị di động nhiều nhất có thể. Qua đó cũng tạo đà để có thể đưa tựa game lên nền tảng IOS, nơi có những yêu cầu về hiệu năng khắt khe hơn.

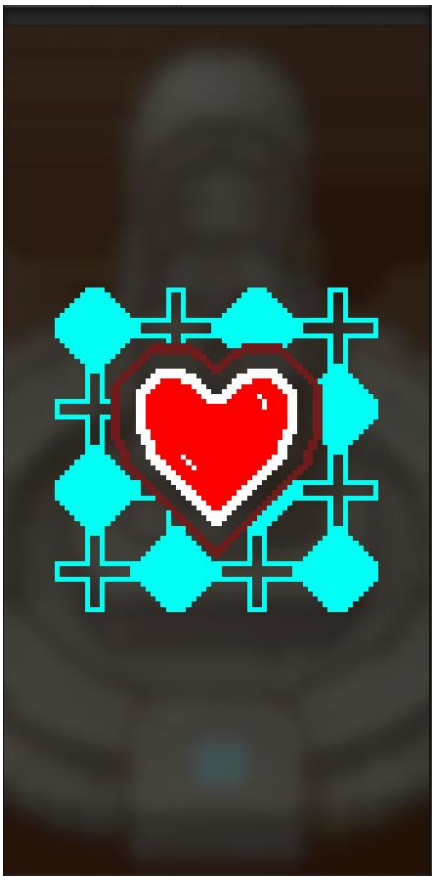
2.2 Kịch bản game

2.2.1 Mô tả

My’s Heart là một tựa game thuộc thể loại Tower Defense, tuy rằng tựa game vẫn tập trung vào mục tiêu phòng thủ, tiêu diệt quái vật, song việc kết hợp giữa các công trình phòng thủ cố định và người chơi chủ động di chuyển hỏa cầu để tấn công kẻ địch sẽ mang tới luồng gió khác cho tựa game. Khi bước vào trò chơi, người chơi sẽ trở thành những người canh gác; Bảo vệ trái tim và thân thể của Bức tượng nữ thần là nhiệm vụ hàng đầu. Hãy đánh bại những linh hồn ác quỷ đang miệt mài tấn công, với chúng sự hủy diệt mới là mục tiêu cao nhất.

Hãy điều khiển ngọn lửa, thứ được tạo nên từ những vật chất kỳ lạ được phủ kín xuống mảnh đất này. Nhưng cũng đừng quên kích hoạt lại những bệ đá phòng thủ, nó sẽ giúp đỡ phần nào trong cuộc chiến gần như vô tận này.

Sau khi hoàn thành các màn chơi, game sẽ gửi tặng bạn những phần quà nho nhỏ, và bạn có thể nghỉ ngơi hoặc quay lại với công việc của mình, hoặc sẽ chiến đấu tiếp nếu như bạn vẫn chưa muốn dừng lại. Sự kết hợp giữa việc sử dụng các công trình phòng thủ tĩnh và hỏa cầu tấn công động sẽ hứa hẹn mang lại phong cách chơi mới lạ mà cũng không quá khó để tiếp cận. Không những vậy My’s Heart còn giúp bạn luyện tập khả năng kiểm soát, chi tiêu một cách hợp lý (Có vẻ sẽ có rất nhiều thứ cần tài nguyên để nâng cấp), hãy tính toán 1 cách hợp lý để có thể dễ dàng ngăn cản lũ quái vật.



Hình 2.1 Màn hình khởi động game My's Heart

2.2.2 Các quy tắc trò chơi

Trong Game sẽ có 3 map chơi chính, với số lượng bệ đá phòng thủ, bệ đá điều khiển, cổng xuất hiện quái vật khác nhau. Các map chơi được sắp xếp theo 1 thứ tự nhất định và sẽ yêu cầu cần đạt được cấp độ nhất định của màn chơi trước đó để có thể mở khóa. Việc này góp phần giảm căng thẳng khi bước vào các map có số lượng cổng quái vật nhiều hơn. Một khi xuất hiện, những con quái vật sẽ lao nhanh đến phá hủy mục tiêu, không những thế chúng còn có các khả năng khác nhau như tăng tốc, tàng hình, … Hãy xem xét và tính toán bước đi của mình thật cẩn thận giữ chúng thật xa 'Nữ thần'. Ở các màn chơi sẽ có chỉ số về lượng quái vật mà bạn cần tiêu diệt để hoàn thành, sau khi đạt mục tiêu phần thưởng sẽ hiện ra, cùng với đó là các lựa chọn để tạm dừng trò chơi, tiếp tục chơi hoặc vào kho đồ để trang bị thêm những vật phẩm phụ trợ.

Người chơi sẽ tập trung vào việc điều khiển hỏa cầu lướt qua quái vật nhằm gây sát thương và tiêu diệt chúng, những bệ đá phòng thủ sau khi được kích hoạt sẽ tấn công sau một quãng thời gian nhất định. Sau các màn chơi sức mạnh của quái vật sẽ được gia tăng, ở các mốc màn chơi cố định số lượng quái vật sinh ra cũng sẽ tăng lên. Cùng với đó sau khi tiêu diệt quái vật người chơi sẽ nhận được tiền thưởng và kinh nghiệm, qua đó giúp người chơi nâng cấp sức mạnh, mua thêm các item hỗ trợ.

Điều kiện qua màn chơi:

- Người chơi tiêu diệt đủ số lượng quái vật yêu cầu.

Điều kiện thất bại:

- Lượng HP của người chơi tụt về 0.

  Nếu như thất bại người chơi sẽ cần trở về màn hình chính, sử dụng số tiền vừa kiếm được nâng cấp chỉ số, mua thêm các vật phẩm hỗ trợ để có thể quay lại trận chiến.

Hình 2.2 Giao diện game My's Heart (Dead UI)

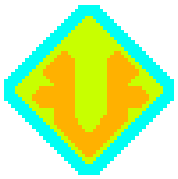
2.2.3 Thiết kế các màn của game

My’s Heart được thiết kế với 3 map chơi chính, với mỗi map sẽ có các màn chơi riêng, với mỗi màn chơi người chơi sẽ cần tiêu diệt 1 số lượng quái vật nhất định. Sau mỗi màn chơi, người chơi sẽ nhận được lượng vàng, cùng với các vật phẩm phụ trợ ngẫu nhiên sau các mốc màn chơi cố định. Việc phân chia map và màn chơi như vậy giúp người chơi có thể tạm dừng game bất cứ lúc nào để tiếp tục cho công việc và quay lại vào những quãng thời gian khác. Các map chơi sẽ có độ khó nhất định và yêu cầu về cấp độ ở các màn chơi trước đó để mở khóa.

2.3 Vật phẩm thu thập

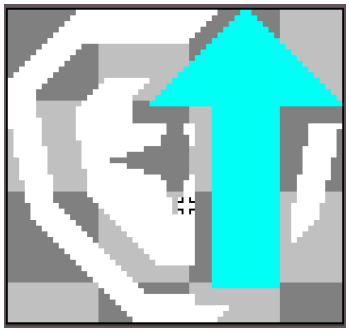
Số lượng quái vật ở các màn chơi và map là khác nhau, việc tiêu diệt càng nhiều quái vật sẽ thu thập được càng nhiều tiền:

- Đồng tiền:
    - Sẽ rơi ra sau khi tiêu diệt quái vật.
    - Là thứ thiết yếu để mở khóa kỹ năng, mua sắm vật phẩm và nâng cấp công trình phòng thủ.



Hình 2.3 Hình đồng tiền

- Các item hỗ trợ:
    - Sẽ nhận được ngẫu nhiên sau khi hoàn thành các mốc màn chơi cố định.
    - Mỗi item sẽ có một khả năng khác nhau (Hồi phục HP, tăng giáp).



Hình 2.4 Hình icon vật phẩm phụ trợ

2.4 Cơ chế tính điểm

Tựa game My's Heart tính điểm dựa trên lượng vàng bạn kiếm được từ các màn chơi và sẽ ghi nhận lại thời gian bạn đã dành ra cho map chơi đó.

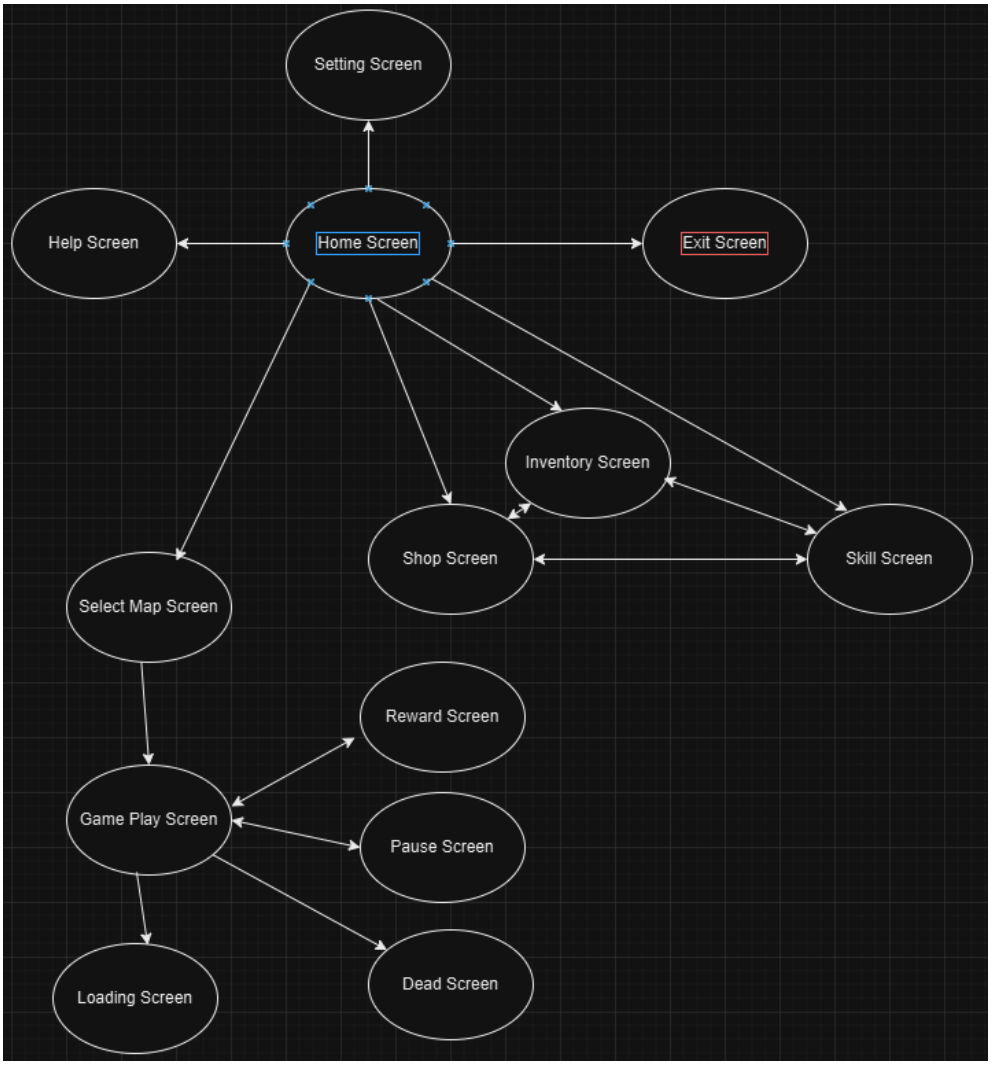
Việc tiêu diệt các quái vật có cấp độ cao sẽ mang lại cho bạn nhiều điểm hơn.

2.5 Tương tác và điều khiển

My's Heart là một tựa game trên nền tảng mobile nên người chơi sẽ thao tác qua màn hình cảm ứng. Việc điều khiển nhân vật và tấn công sẽ được thực hiện thông qua việc ấn vào các bệ đá điều khiển, điều này mang lại một cách chơi mới thay vì di chuyển bằng joystick hoặc các nút di chuyển như thông thường. Kết với với việc game được thiết kế để chơi ở màn hình dọc, trên lý thuyết người chơi có thể chơi game bằng một tay trên các thiết bị có màn hình vừa phải.

2.6 Storyboard

2.6.1 Sơ đồ màn hình



Hình 2.5 Sơ đồ các màn chơi

2.6.2 Màn hình bắt đầu

Hình 2.6 Màn hình khởi đầu

- Nút 'PLAY' để bắt đầu game.

- Nút shop dùng để mở giao diện Cửa hàng, Kho vật phẩm và Quản lý kỹ năng nhân vật.

- Nút setting để mở giao diện cài đặt cho game.

- Nút help dùng để xem các thông tin về game và hướng dẫn cơ bản.

- Nút thoát dùng để thoát khỏi trò chơi. 

2.6.3 Màn hình lựa chọn map, thông tin map

- Màn hình hiển thị danh sách các map chơi.
- Nút ‘More’ để xem thêm thông tin chi tiết của map chơi.
- Nút ‘Home’ dùng để quay trở về màn hình khởi đầu.
- Nút ‘Play’ dùng để bắt đầu màn chơi.
- Nút ‘Back’ để quay lại màn hình lựa chọn map.

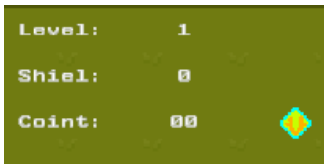
Hình 2.8 Giao diện chọn màn chơi

Hình 2.7 Giao diện thông tin chi tiết màn chơi

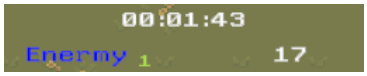
2.6.4 Màn hình chơi game

- Nút Menu dùng để tạm dừng trò chơi.

- Các thông số cơ bản của người chơi.



- Thời gian chơi và số lượng quái vật cần tiêu diệt để vượt qua màn chơi.



- Khu vực hiển thị các kỹ năng cơ bản của người chơi.

- Khu vực hiển thị các vật phẩm phụ trợ. (Nếu có)

Hình 2.9 Giao diện chơi game chính

2.6.5 Màn hình tạm dừng

- Nút ‘Play’ để tiếp tục màn chơi.
- Nút ‘Skill’ để mở bảng quản lý kỹ năng người chơi.
- Nút ‘Inv’ dùng để mở kho vật phẩm.
- Nút ‘Home’ dùng để quay trở về màn hình khởi đầu.

Hình 2.10 Màn hình tạm dừng trò chơi

2.6.6 Màn hình khi người chơi chết

Hình 2.11 Giao diện khi người chơi bị giết

- Khi người chơi chết sẽ phải trở về màn hình khởi đầu, nâng cấp và trang bị thêm vật phẩm để quay lại trận chiến.

2.6.7 Màn hình thông báo nhận thưởng

Hình 2.12 Giao diện nhận thưởng

Người chơi sẽ thấy được phần thưởng và có thể lựa chọn chơi tiếp bằng cách ấn vào ‘(Tap to Close)’, mở kho vật phẩm hoặc trở về màn hình khởi đầu.

2.6.8 Màn hình Shop, Inventory và Skill

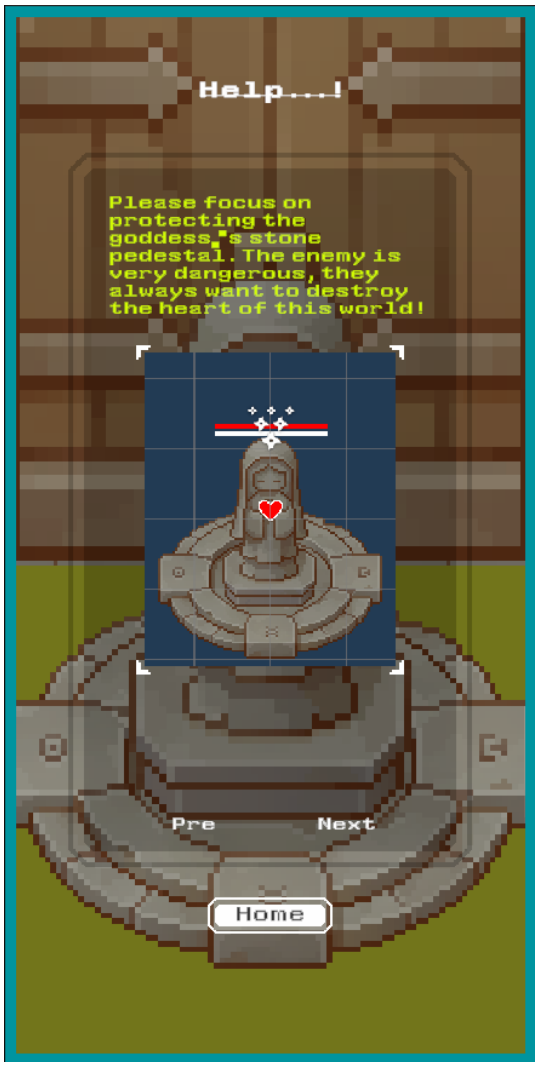
Hình 2.15 Giao diện cửa hàng (Shop)

Hình 2.13 Giao diện kho vật phẩm (Inventory)

Hình 2.14 Giao diện thông tin kỹ năng (Skill)

- **Shop:** Là nơi người dùng có thể sử dụng tiền để mua thêm các vật phẩm phụ trợ cho trận chiến.
- **Inventory:** Nơi chứa các vật phẩm phụ trợ, người dùng có thể thêm nó vào các ô trang bị để sử dụng trong trận chiến.
- **Skills:** Người chơi có thể xem thông tin kỹ năng và nâng cấp các kỹ năng khi đủ lượng tiền yêu cầu.

2.6.9 Màn hình trợ giúp



Hình 2.16 Giao diện trợ giúp trong game

- Người chơi có thể xem thêm các thông tin cơ bản về game ở đây.

CHƯƠNG 3. THỬ NGHIỆM SẢN PHẨM VÀ ĐÁNH GIÁ

3.1 Công cụ và kỹ thuật

3.1.1 Các công cụ chính

Các công cụ Unity cung cấp rất đa dạng và mạnh mẽ, bên cạnh đó người phát triển còn có thể sử dụng các gói Package khác để hỗ trợ việc xây dựng và phát triển game:

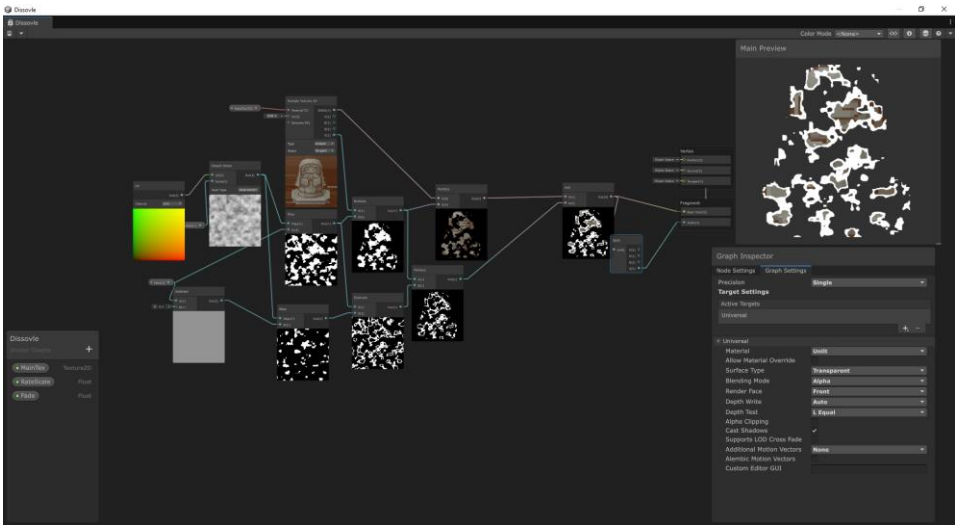
- **Particle System:** Particle System là một hệ thống cho phép bạn tạo và điều khiển các hạt nhỏ (particles) để mô phỏng các hiện tượng tự nhiên hoặc các hiệu ứng đặc biệt. Mỗi particle có thể có các thuộc tính như vị trí, tốc độ, màu sắc, kích thước, và thời gian sống.
    - **Cấu trúc của Particle System**
      Particle System trong Unity bao gồm nhiều thành phần khác nhau, mỗi thành phần có thể được cấu hình để tạo ra các hiệu ứng mong muốn:
      Emitter: Xác định cách các hạt được phát ra, bao gồm vị trí, tốc độ, và hướng phát ra.
      Shape: Xác định hình dạng của vùng phát ra hạt, chẳng hạn như hình cầu, hình nón, hoặc hình hộp.
      Renderer: Xác định cách các hạt được hiển thị, bao gồm texture, màu sắc, và kích thước.
      Modules: Các module khác nhau cho phép bạn điều chỉnh các thuộc tính của hạt, chẳng hạn như tốc độ, trọng lực, và va chạm.
    - **Các thành phần chính của Particle System**
      Main Module: Điều chỉnh các thuộc tính cơ bản của Particle System như thời gian sống, tốc độ, kích thước, và màu sắc của hạt.
      Emission Module: Điều chỉnh tốc độ phát ra hạt và số lượng hạt được phát ra.
      Shape Module: Xác định hình dạng của vùng phát ra hạt.
      Velocity over Lifetime Module: Điều chỉnh tốc độ và hướng di chuyển của hạt theo thời gian.

Color over Lifetime Module: Điều chỉnh màu sắc của hạt theo thời gian.

Size over Lifetime Module: Điều chỉnh kích thước của hạt theo thời gian.

Collision Module: Cho phép các hạt va chạm với các đối tượng khác trong cảnh.

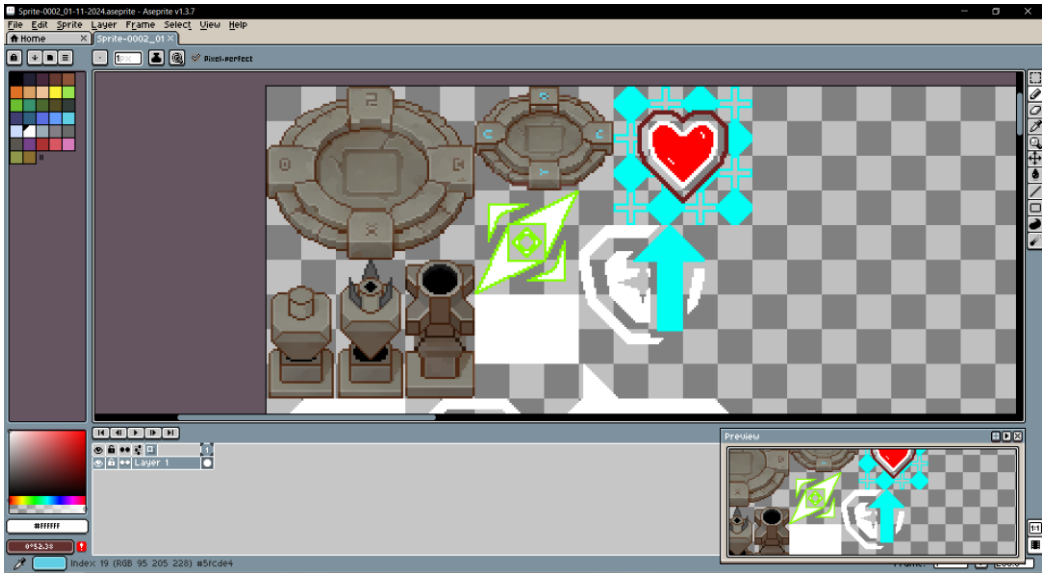
- **Shader Graph:** Shader Graph là một công cụ mạnh mẽ trong Unity, cho phép bạn tạo và chỉnh sửa các shader một cách trực quan mà không cần viết mã. Đây là một phần của Unity's Scriptable Render Pipeline (SRP), giúp bạn dễ dàng tạo ra các hiệu ứng đồ họa phức tạp và tùy chỉnh.



Hình 3.1 Giao diện chỉnh sửa Shader graph trong Unity

Các thành phần đồ họa được thiết kế trên phần mềm Aseprite với phong cách pixel. Aseprite là một phần mềm chỉnh sửa hình ảnh và tạo hoạt hình pixel art, được thiết kế chủ yếu cho việc vẽ và tạo hoạt hình pixel art. Phần mềm này chạy trên các hệ điều hành Windows, macOS và Linux, và cung cấp nhiều công cụ khác nhau để chỉnh sửa hình ảnh và hoạt hình như các lớp (layers), khung hình (frames), hỗ trợ tilemap, giao diện dòng lệnh (CLI), và scripting bằng Lua.

Ngoài là công cụ vẽ và chỉnh sửa hình ảnh, Aseprite còn hỗ trợ khả năng xây dựng animation, hiệu ứng 2d, xây dựng sprite cho tilemap thông qua Tiled Mode và có thể chuyển đổi các bức ảnh có độ phân giải cao về dạng pixel art.



Hình 3.2 Giao diện làm việc của phần mềm Aseprite

3.1.2 Design Pattern

Design pattern là các giải pháp tổng thể đã được tối ưu hóa, được tái sử dụng cho các vấn đề phổ biến trong thiết kế phần mềm mà chúng ta thường gặp phải hàng ngày. Việc sử dụng các design pattern sẽ giúp chúng ta giảm được thời gian và công sức suy nghĩ ra các cách giải quyết cho những vấn đề đã có lời giải. Lợi ích của việc sử dụng các mô hình Design Pattern vào phần mềm đó chính là giúp chương trình chạy uyển chuyển hơn, dễ dàng quản lý tiến trình hoạt động, dễ nâng cấp bảo trì, …

Hiện nay, có tổng cộng 23 mẫu thiết kế (Design Patterns) phổ biến được chia thành ba nhóm chính: Creational Patterns, Structural Patterns, và Behavioral Patterns.

➢ Các công cụ Unity thường dùng:

Sprite Editor

Sprite Editor trong Unity là một công cụ mạnh mẽ giúp bạn chuẩn bị và chỉnh sửa các sprite cho dự án của mình. Công cụ này cho phép bạn cắt các hình ảnh lớn hoặc sprite sheets thành các sprite riêng lẻ, giúp tối ưu hóa và quản lý các sprite một cách hiệu quả.

Chức năng chính:

- Slicing: Tự động cắt các sprite từ hình ảnh lớn hoặc sprite sheets.
- Custom Outline: Chỉnh sửa hình dạng của mesh mà Unity render sprite lên.
- Custom Physics Shape: Chỉnh sửa hình dạng vật lý của sprite.
- Secondary Textures: Liên kết các texture bổ sung với sprite đã chỉnh sửa.

Animation

Animation trong Unity là một hệ thống mạnh mẽ cho phép bạn tạo và điều khiển các hoạt hình cho các đối tượng trong game. Hệ thống này bao gồm nhiều công cụ và tính năng để tạo ra các hoạt hình phức tạp và tùy chỉnh.

Chức năng chính:

- Retargeting: Áp dụng các hoạt hình được tạo cho một mô hình sang mô hình khác.
- Animation Events: Thêm dữ liệu vào clip nhập khẩu để xác định khi nào các hành động cụ thể xảy ra đồng bộ với hoạt hình.
- Animation Weights: Điều khiển trọng số của hoạt hình tại runtime.

Animator

Animator là một thành phần trong Unity giúp bạn gán hoạt hình cho một GameObject. Thành phần này yêu cầu một tham chiếu đến một Animator Controller, định nghĩa các clip hoạt hình nào sẽ được sử dụng và kiểm soát khi nào và làm thế nào để chuyển đổi giữa chúng.

Chức năng chính:

- Animator Controller: Điều khiển hoạt hình thông qua các Animation Layers với Animation State Machines và Animation Blend Trees.
- Avatar: Giao diện để retargeting hoạt hình từ một rig sang rig khác.
- Root Motion: Kiểm soát vị trí và xoay của nhân vật từ hoạt hình hoặc từ script.

Profiler

Profiler trong Unity là một công cụ giúp bạn thu thập và hiển thị thông tin hiệu suất của ứng dụng. Bạn có thể kết nối nó với các thiết bị trên mạng của mình hoặc các thiết bị kết nối với máy tính để kiểm tra cách ứng dụng chạy trên nền tảng phát hành dự định.

Hình 3.3 Công cụ Profiler

Chức năng chính:

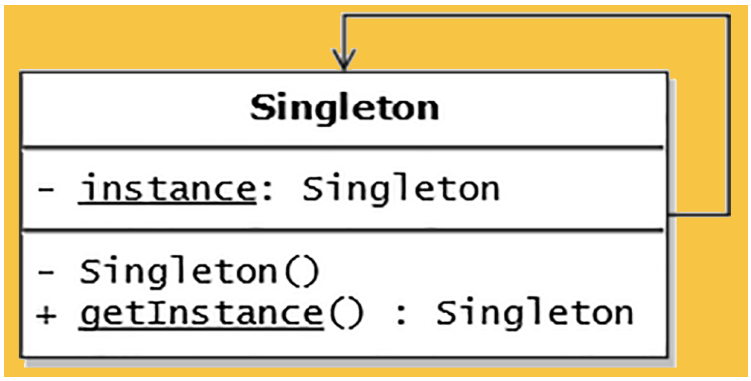
- Performance Data: Thu thập và hiển thị dữ liệu hiệu suất dưới dạng biểu đồ.
- Memory Profiler: Cung cấp phân tích chi tiết về hiệu suất bộ nhớ.
- Frame Debugger: Cho phép bạn đóng băng playback của game trên một khung hình cụ thể và xem các draw calls riêng lẻ.

Ngoài ra Unity còn cung cấp sẵn cho các nhà phát triển rất nhiều công cụ khác hỗ trợ làm việc với môi trường 2D, 3D và thực tế ảo. Các hệ thống vật lý và xử lý va chạm trong Unity góp phần giảm thiểu các công việc lặp đi lặp lại, từ đó tăng hiệu suất làm việc cho nhà phát triển.

➢ Các Design Pattern phổ biến:

Singleton Design Pattern

Singleton Pattern là một mẫu thiết kế thuộc nhóm Creational Patterns, đảm bảo rằng một lớp chỉ có một thể hiện duy nhất và cung cấp một điểm truy cập toàn cục đến thể hiện đó. Điều này rất hữu ích trong các trường hợp cần quản lý tài nguyên hoặc các đối tượng toàn cục, chẳng hạn như bộ điều khiển trò chơi, bộ quản lý âm thanh, hoặc các cấu hình hệ thống.



Hình 3.4 Singleton Design Pattern

Singleton Design Pattern có 2 đặc điểm chính:

- **Đảm bảo duy nhất**: Chỉ có một thể hiện duy nhất của lớp được tạo ra trong suốt vòng đời của ứng dụng.
- **Điểm truy cập toàn cục**: Cung cấp một điểm truy cập toàn cục đến thể hiện duy nhất này, giúp dễ dàng quản lý và sử dụng.

  Chính vì vậy việc sử dụng **Singleton Design Pattern** cho các lớp như UIManager (Dùng để quản lý các thành phần giao diện cần tương tác trong game) sẽ giúp việc truy cập và cập nhật giao diện một cách đơn giản hơn, hạn chế việc tham chiếu quá nhiều đến cùng 1 thành phần UI dẫn đến mã nguồn phức tạp và khó bảo trì.

Về mặt ưu điểm, **Singleton Design Pattern** mang lại:

- Tiết kiệm tài nguyên: Đảm bảo rằng chỉ có một thể hiện duy nhất của lớp, giúp tiết kiệm tài nguyên hệ thống.
- Dễ dàng quản lý: Cung cấp một điểm truy cập toàn cục, giúp dễ dàng quản lý và sử dụng thể hiện duy nhất này.
- Kiểm soát truy cập: Giúp kiểm soát truy cập đến thể hiện duy nhất, ngăn chặn việc tạo ra nhiều thể hiện không cần thiết.

Nhược điểm:

- Khó kiểm tra: Singleton Pattern có thể làm cho việc kiểm tra đơn vị (unit testing) trở nên khó khăn hơn do sự phụ thuộc vào thể hiện duy nhất.
- Khó mở rộng: Việc mở rộng lớp Singleton có thể gặp khó khăn do cấu trúc cố định của nó.
- Tiềm ẩn vấn đề về đồng bộ hóa: Trong môi trường đa luồng, việc đảm bảo rằng chỉ có một thể hiện duy nhất có thể gặp vấn đề về đồng bộ hóa.

Observer Pattern

Observer Pattern là một mẫu thiết kế thuộc nhóm Behavioral Patterns, cho phép một đối tượng (được gọi là subject) thông báo cho các đối tượng khác (được gọi là observers) về các thay đổi trạng thái của nó. Điều này rất hữu ích trong các hệ thống sự kiện hoặc thông báo, nơi mà nhiều đối tượng cần được cập nhật khi trạng thái của một đối tượng thay đổi.

Đặc điểm chính của Observer Pattern:

- Một - nhiều: Định nghĩa một sự phụ thuộc một - nhiều giữa các đối tượng, giúp khi một đối tượng thay đổi trạng thái, tất cả các đối tượng phụ thuộc đều được thông báo và cập nhật tự động.
- Tách biệt: Giúp tách biệt logic của subject và observers, làm cho mã nguồn dễ bảo trì và mở rộng.

Lợi ích của Observer Pattern:

- Tách biệt logic: Giúp tách biệt logic của subject và observers, làm cho mã nguồn dễ bảo trì và mở rộng.
- Dễ dàng mở rộng: Dễ dàng thêm hoặc loại bỏ observers mà không ảnh hưởng đến subject.
- Tự động cập nhật: Các observers tự động được thông báo và cập nhật khi trạng thái của subject thay đổi.

Nhược điểm của Observer Pattern:

- Hiệu suất: Có thể ảnh hưởng đến hiệu suất nếu có quá nhiều observers hoặc nếu việc thông báo và cập nhật tốn nhiều tài nguyên.
- Phụ thuộc: Tạo ra sự phụ thuộc giữa subject và observers, có thể làm phức tạp hóa mã nguồn.

Object Pooling Design Pattern

Object Pooling là một mẫu thiết kế thuộc nhóm Creational Patterns, được sử dụng để quản lý và tái sử dụng các đối tượng thay vì tạo mới và hủy chúng liên tục. Điều này giúp cải thiện hiệu suất và giảm thiểu việc sử dụng tài nguyên, đặc biệt là trong các ứng dụng yêu cầu tạo và hủy nhiều đối tượng trong thời gian ngắn, chẳng hạn như game.

Đặc điểm chính của Object Pooling:

- Tái sử dụng đối tượng: Thay vì tạo mới và hủy đối tượng liên tục, Object Pooling cho phép tái sử dụng các đối tượng đã được tạo ra.
- Quản lý tài nguyên hiệu quả: Giảm thiểu việc sử dụng tài nguyên hệ thống bằng cách quản lý số lượng đối tượng được tạo ra và hủy bỏ.
- Cải thiện hiệu suất: Giảm thiểu chi phí tạo và hủy đối tượng, giúp cải thiện hiệu suất của ứng dụng.

Lợi ích của Object Pooling:

- Tiết kiệm tài nguyên: Giảm thiểu việc tạo và hủy đối tượng liên tục, giúp tiết kiệm tài nguyên hệ thống.
- Cải thiện hiệu suất: Giảm thiểu chi phí tạo và hủy đối tượng, giúp cải thiện hiệu suất của ứng dụng.
- Quản lý đối tượng dễ dàng: Dễ dàng quản lý và theo dõi các đối tượng trong pool.

Nhược điểm của Object Pooling:

- Phức tạp hơn: Cần phải quản lý pool và đảm bảo rằng các đối tượng được tái sử dụng đúng cách.
- Bộ nhớ: Cần phải duy trì một số lượng đối tượng trong pool, có thể tốn bộ nhớ nếu không được quản lý tốt.

Việc sử dụng Object pooling để spawn quái vật hoặc bắn đạn giúp tránh việc phải Instantiate và Destroy GameObject nhiều lần trong game từ đó giúp cải thiện hiệu suất của game và các tài nguyên khác.

3.2 Các chức năng chính của Game

Việc Unity tạo sẵn các component về xử lý vật lý như Rigidbody và Rigidbody2D hay xử lý với va chạm với các component về Collider giúp phần nào rút ngắn thời gian phát triển. Cùng với sự tùy biến cho các component, các nhà phát triển có thể kết hợp các component lại với nhau theo nhiều cách từ đó tạo nên vô vàn các chức năng khác nhau.

3.2.1 Code điều khiển và xử lý va chạm

```
void Update()
{
    if (PlayerControl.instance._fireBall != null && PlayerControl.instance._targetPos != null)
    {
        if (Vector3.Distance(PlayerControl.instance._fireBall.transform.position, PlayerControl.instance._targetPos.position) > 0.05f)
        {
            direction = PlayerControl.instance._targetPos.position - PlayerControl.instance._fireBall.transform.position;
            direction.Normalize();
            Vector3 subdirection = PlayerControl.instance._fireBall.transform.position - PlayerControl.instance._targetPos.position;
            float angle = Vector3.Angle(Vector3.up, subdirection);
            angle *= direction.x > 0 ? 1 : -1;
            PlayerControl.instance._fireBall.transform.position += direction * _speed * Time.deltaTime;
            transform.GetChild(0).transform.rotation = Quaternion.Euler(new Vector3( 0, 0, angle));
        }
        else
        {
            Quaternion rotation = Quaternion.LookRotation(Vector3.back);
            transform.GetChild(0).transform.rotation = rotation;
        }
    }
}
```

Hình 3.5 Code di chuyên cho hỏa cầu

Với việc lấy dữ liệu về điểm đích mà người chơi muốn di chuyển đến thông qua lớp PlayerControl (là một Singleton) sẽ xác định được hướng di chuyển. Trong trường hợp khoảng cách giữa hỏa cầu và điểm đích lớn hơn một khoảng cách nhất định sẽ di chuyển hỏa cầu dần về phía đích thông qua việc liên tục tính toán và cập nhật vị trí mới. Việc sử dụng hàm Normalize() nhằm đảm bảo rằng vector hướng di chuyển sẽ luôn có độ dài là 1 đơn vị từ đó duy trì vận tốc cho du khoảng cách tới đích có thay đổi.

```
private void OnTriggerEnter2D(Collider2D collision)
{
    if(collision != null)
    {
        if(collision.gameObject.tag == "Enermy")
        {
            skeletonsScript sck = collision.gameObject.GetComponent<skeletonsScript>();
            if(sck != null)
            {
                sck.takeDamage(_damage);
            }
        }
    }
}
```

Hình 3.6 Code sử dụng Collider để phát hiện khi va chạm với quái vật

Việc sử dụng OnTrigger của Collider nhằm mục đích phát hiện va chạm nhưng không gây phản ứng vật lý từ đó tạo nên khả năng lướt qua vật thể của hỏa cầu và gây sát thương lên những con quái vật giống như cách những hồn ma làm.

```
void Update()
{
    if(_isDead)
    {
        _ani.SetTrigger("Dead");
    }
    else if(timeCount > 0)
    {
        timeCount -= Time.deltaTime;
        _ani.SetTrigger("Pause");
    }
    else if(_target != null)
    {
        Vector3 dir = _target - transform.position;
        dir.Normalise();
        _ani.SetFloat("Horizontal", dir.x);
        _ani.SetFloat("Vertical", dir.y);
        if ( Vector3.Distance(transform.position, _target) > 1f)
        {
            transform.position += dir * (_speed * Time.deltaTime);
            _ani.SetTrigger("Run");
        }
        else
        {
            _ani.SetTrigger("Attack");
        }
    }
}
```

Hình 3.7 Code di chuyển và tấn công của quái vật

Tương tự với hỏa cầu, quái vật cũng có mục tiêu tấn công chúng sẽ tiến lên cho tới khi mục tiêu ở vào tầm tấn công, chúng sẽ dừng lại và bắt đầu phá hủy. Thông qua component Animator (_ani) do Unity cung cấp có thể dễ dàng quản lý các trạng thái animation cũng như phát các animation clip theo mong muốn. Sử dụng các hàm như SetFloat(), SetTrigger() lập trình viên có thể thay đổi các giá trị điều kiện từ đó phát các đoạn animation mong muốn. Trong đoạn code này

SetFloat với tham số “Horizontal” và “Vertical” nhằm xác định hướng của quái vật từ đó đưa ra các animation có hướng gần đúng với hướng di chuyển của nhân vật. Vì game được phát triển ở dạng 2D nên các hướng di chuyển của nhân vật chỉ tương đối nhằm giảm thiểu số lượng animation clip cần có. Với SetTrigger, dùng để kích hoạt các trạng thái di chuyển hoặc tấn công của quái vật.

```
public override void Dead()
{
    PlayerControl.instance.enermySpawning = Mathf.Clamp(PlayerControl.instance.enermySpawning -1, 0, PlayerControl.instance.enermySpawning);
    userPointManager.UserPoint.towerLV[PlayerControl.instance.currentTower].timeSpawn += 1;
    _isDead = true;
    transform.GetChild(0).GetComponent<AudioSource>().Play();
    Invoke("inActive", 2.0f);
}
```

Hình 3.8 Code khi quái vật bị tiêu diệt

Để tránh việc sinh ra và phá hủy GameObject liên tục, việc tái sử dụng các Object đó giúp tối ưu tài nguyên và hiệu suất cho game. Thay vì sử dụng Destroy() với quái vật bị tiêu diệt, ta sẽ inactive quái vật đi sau khi chúng chết, sau đó các cổng spawn quái vật có thể reset các chỉ số cho quái vật và active chúng lại từ đó một đợt tấn công mới sẽ bắt đầu.

3.2.2 Lưu trữ dữ liệu

```
[Serializable]
public class items
{
    public string name;
    public string title;
    public int expense;
    public string des;

    public items(string name, string title, int expense, string des)
    {
        this.name = name;
        this.title = title;
        this.expense = expense;
        this.des = des;
    }
}
```

Hình 3.9 Class lữu trữ thông tin vật phẩm với Serializable Object

Serializable Object trong Unity là các đối tượng có thể được chuyển đổi thành một định dạng có thể lưu trữ hoặc truyền tải, và sau đó có thể được tái tạo lại từ định dạng đó. Điều này rất hữu ích khi bạn cần lưu trữ trạng thái của đối tượng, truyền dữ liệu giữa các hệ thống, hoặc lưu trữ dữ liệu trong các tệp.

Tại sao cần Serializable Object?

- **Lưu trữ dữ liệu:** Bạn có thể lưu trữ trạng thái của đối tượng vào tệp và sau đó tái tạo lại đối tượng từ tệp đó.
- **Truyền dữ liệu:** Serializable Object cho phép bạn truyền dữ liệu giữa các hệ thống hoặc qua mạng.
- **Lưu trữ trong Unity:** Unity sử dụng serialization để lưu trữ dữ liệu trong các tệp dự án, chẳng hạn như các tệp cảnh (scene) và tài sản (asset).

Ưu Điểm

- Dễ Dàng Lưu Trữ và Tải Dữ Liệu: Lưu Trữ Dữ Liệu: Serializable Object cho phép bạn dễ dàng lưu trữ trạng thái của đối tượng vào các tệp, giúp việc lưu trữ và khôi phục dữ liệu trở nên đơn giản.
- Tải Dữ Liệu: Bạn có thể dễ dàng tải lại dữ liệu từ các tệp đã lưu, giúp khôi phục trạng thái của đối tượng một cách nhanh chóng.
- Truyền Dữ Liệu: Truyền Qua Mạng: Serializable Object cho phép bạn truyền dữ liệu giữa các hệ thống hoặc qua mạng một cách hiệu quả.
- Chia Sẻ Dữ Liệu: Dữ liệu có thể được chia sẻ giữa các thành phần khác nhau của ứng dụng hoặc giữa các ứng dụng khác nhau.
- Tích Hợp Tốt Với Unity: Lưu Trữ Trong Unity: Unity sử dụng serialization để lưu trữ dữ liệu trong các tệp dự án, chẳng hạn như các tệp cảnh (scene) và tài sản (asset).
- Hỗ Trợ Inspector: Các đối tượng serializable có thể được hiển thị và chỉnh sửa trong Unity Inspector, giúp việc quản lý dữ liệu trở nên dễ dàng hơn.
- Hỗ Trợ Đa Dạng Định Dạng:
    - JSON: Bạn có thể sử dụng JsonUtility để serialize và deserialize đối tượng thành định dạng JSON và ngược lại.
    - Binary: Bạn cũng có thể sử dụng các phương pháp khác để serialize đối tượng thành định dạng nhị phân.

Nhược Điểm

- Giới Hạn Về Loại Dữ Liệu: Không Hỗ Trợ Tất Cả Các Loại Dữ Liệu: Một số loại dữ liệu không thể được serialize trực tiếp, chẳng hạn như các đối tượng không có thuộc tính [System.Serializable] hoặc các đối tượng không thể serialize như Transform.
- Hiệu Suất: Chi Phí Hiệu Suất: Quá trình serialization và deserialization có thể tốn tài nguyên và ảnh hưởng đến hiệu suất của ứng dụng, đặc biệt là khi làm việc với các đối tượng lớn hoặc phức tạp.
- Bảo Mật: Rủi Ro Bảo Mật: Dữ liệu serialized có thể dễ dàng bị đọc hoặc chỉnh sửa nếu không được bảo vệ đúng cách, dẫn đến các rủi ro bảo mật.
- Khó Khăn Trong Quản Lý Phiên Bản: Quản Lý Phiên Bản: Khi cấu trúc của đối tượng thay đổi, việc quản lý phiên bản của dữ liệu serialized có thể trở nên phức tạp và dễ gây lỗi.

```
private const string dataName = "GameData";

private void Awake()
{
    //PlayerPrefs.DeleteAll();
    //PlayerPrefs.DeleteKey(dataName);
    if (_home != null) _home.SetActive(PlayerPrefs.GetString("SaveSetting") != "");
    if (_playUi != null) _playUi.SetActive(false);
    if (_shopUi != null) _shopUi.SetActive(false);
    if (_settingUi != null) _settingUi.SetActive(false);
    if (_helpUi != null) _helpUi.SetActive(PlayerPrefs.GetString("SaveSetting") == "");
    if (_quitUi != null) _quitUi.SetActive(false);
    if (PlayerPrefs.GetString("SaveSetting") == "")
    {
        PlayerPrefs.SetString("SaveSetting", settingManager.settingJson.toJson());
    }
    else
    {
        settingManager.settingJson = SettingJson.fromJson(PlayerPrefs.GetString("SaveSetting"));
    }

    string gameData = PlayerPrefs.GetString(dataName);
    if (gameData == "")
    {
        PlayerPrefs.SetString(dataName, userPointManager.toJsonCode(userPointManager.UserPoint));
    }
    else
    {
        userPointManager.UserPoint = userPointManager.getJsonCode(gameData);
    }
}
```

Hình 3.10 Code tải dữ liệu khi bắt đầu game

Khi các thành phần của màn chơi khởi động sẽ tải các dữ liệu đã được lưu trữ trước đó. Trong trường hợp không tìm thấy dữ liệu bắt buộc sẽ phải khởi tạo lại các dữ liệu mặc định và lưu trữ. Với PlayerPrefs là một lớp trong Unity được

sử dụng để lưu trữ các giá trị đơn giản như chuỗi (string), số nguyên (int) và số thực (float) giữa các phiên chơi game. Điều này rất hữu ích khi bạn cần lưu trữ các thiết lập của người chơi, điểm số cao, hoặc các trạng thái khác mà bạn muốn duy trì giữa các lần chơi. Kết hợp với việc chuyển đổi các dữ liệu về dạng Json giúp dễ dàng lưu trữ và tải dữ liệu lên bất cứ lúc nào. Trong trường hợp cần bảo mật dữ liệu có thể mã hóa Json bằng các thuật toán mã hóa trước khi lưu trữ.

3.2.3 Canvas và các thành phần giao diện

Canvas là một thành phần quan trọng trong Unity, được sử dụng để tạo và quản lý giao diện người dùng (UI). Canvas hoạt động như một vùng chứa cho tất cả các thành phần UI như nút, văn bản, hình ảnh, và các yếu tố tương tác khác.

Unity cung cấp sẵn các thành phần UI cơ bản từ Image, Button, Slider …. Bên cạnh đó các nhà phát triển cũng có thể tạo mới hoặc tùy chỉnh các thành phần giao diện có sẵn trong qua các Script bằng cách sử dụng CustomEditor.

```
[CustomEditor(typeof(SliderSub))] public class CustomSliderEditor : Editor

    public override void OnInspectorGUI()
    {
        // Cập nhật các thay đổi trong các thuộc tính
        serializedObject.Update();

        // Vẽ các thuộc tính cơ bản của Slider
        EditorGUILayout.PropertyField(minValue);
        EditorGUILayout.PropertyField(maxValue);
        //EditorGUILayout.PropertyField(value);
        if(wholeNumbers.boolValue)
            value.floatValue = (int) EditorGUILayout.Slider((float) value.floatValue, minValue.floatValue, maxValue.floatValue);
        else
            value.floatValue = EditorGUILayout.Slider(value.floatValue, minValue.floatValue, maxValue.floatValue);
        EditorGUILayout.PropertyField(wholeNumbers);

        // Vẽ lại thuộc tính CustomValue
        SliderSub slider = (SliderSub)target;
        slider.fillIndex = EditorGUILayout.FloatField("Fill Index", slider.fillIndex);
        slider.audioPlay = (AudioSource)EditorGUILayout.ObjectField("Audio Play", slider.audioPlay, typeof(AudioSource), true);
        slider.subText = EditorGUILayout.TextField("Sub Text", slider.subText);

        // Vẽ các thuộc tính của Slider mà bạn muốn hiển thị
        EditorGUILayout.PropertyField(transition);

        // Kiểm tra loại Transition để vẽ các thuộc tính tương ứng
        if (transition.enumValueIndex == (int)Selectable.Transition.ColorTint)
        {
            EditorGUILayout.PropertyField(colors);  // Hiển thị Color Tint
        }
        else if (transition.enumValueIndex == (int)Selectable.Transition.SpriteSwap)
        {
            EditorGUILayout.PropertyField(spriteState);  // Hiển thị Sprite Swap
        }
        else if (transition.enumValueIndex == (int)Selectable.Transition.Animation)
        {
            EditorGUILayout.PropertyField(animationTriggers);  // Hiển thị Animation
        }

        // **Hiển thị các thuộc tính liên quan đến Slider mà bạn cần thêm**
        EditorGUILayout.PropertyField(handleRect);  // Hiển thị Handle Rect
        EditorGUILayout.PropertyField(fillRect);    // Hiển thị Fill Rect
        EditorGUILayout.PropertyField(onValueChanged);  // Hiển thị On Value Changed
        EditorGUILayout.PropertyField(direction);   // Hiển thị Direction

        // Cập nhật đối tượng sau khi chỉnh sửa
        serializedObject.ApplyModifiedProperties();
    }
}
#endif
```

Hình 3.11 Script Custom Slider component trong unity

3.3 Hình ảnh sản phẩm

3.3.1 Màn hình khởi đầu

Hình 3.12 Thiết kế màn hình chính

3.3.2 Màn hình lựa chọn màn chơi

Hình 3.13 Thiết kế màn hình lựa chọn màn chơi

3.3.3 Màn hình

Hình 3.14 Thiết kế màn hình game play

3.3.4 Màn hình tạm dừng

Hình 3.15 Thiết kế màn hình tạm dừng

3.3.5 Màn hình chết

Hình 3.16 Thiết kế màn hình chết

3.3.6 Màn hình nhận thưởng

Hình 3.17 Thiết kế màn hình nhận thưởng

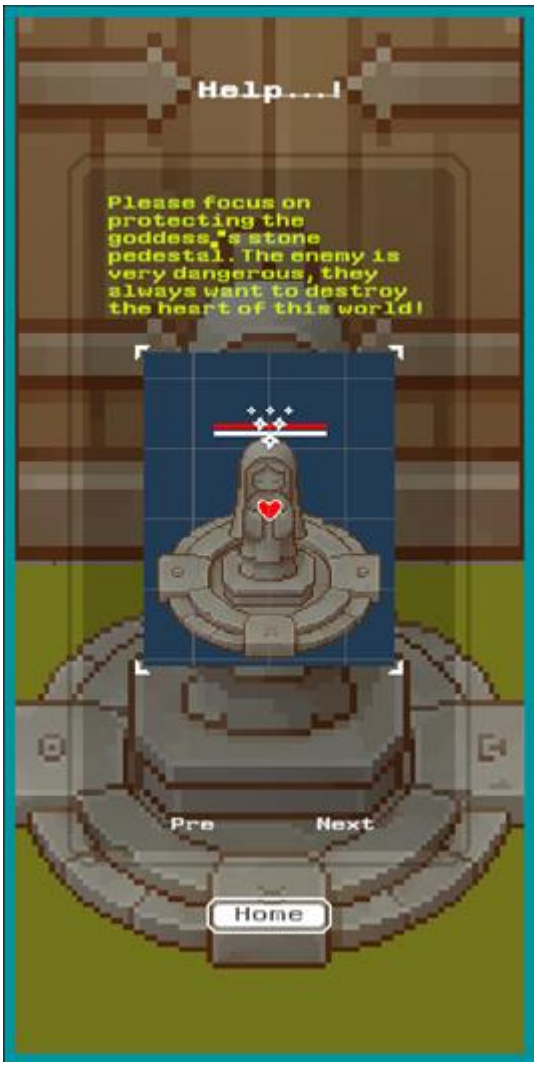
3.3.7 Màn hình Shop, Inventory và Skills

Hình 3.18 Thiết kế màn hình cửa hàng

Hình 3.19 Thiết kế màn hình kho đồ

Hình 3.20 Thiết kế màn hình quản lý kỹ năng

3.3.8 Màn hình hướng dẫn



Hình 3.21 Thiết kế màn hình hướng dẫn trò chơi

3.4 Đánh giá

Về tổng quan, việc sử dụng Unity Engine để phát triển game My's Heart là một lựa chọn sáng suốt, bởi lẽ Unity là một game engine rất mạnh mẽ, không quá khó cho người mới bắt đầu, cộng đồng phát triển lớn mạnh trên khắp thế giới cũng như lượng tài liệu khổng lồ. Từ đó việc tiếp cận Unity trở nên dễ dàng hơn phần nào; Ngoài ra Unity không yêu cầu cấu hình thiết bị phần cứng quá cao như Unreal Engine nên lượng thiết bị có thể tiếp cận Unity là rất lớn.

- Ưu điểm của Unity:

Unity là một công cụ phát triển game mạnh mẽ với nhiều ưu điểm nổi bật. Một trong những điểm mạnh của Unity là khả năng hỗ trợ đa nền tảng, cho phép các nhà phát triển phát hành game trên nhiều hệ điều hành khác nhau như PC, console, di động và VR, giúp tiết kiệm thời gian và công sức. Giao diện trực quan và dễ sử dụng của Unity cũng là một lợi thế lớn, giúp cả những người mới bắt đầu cũng có thể nhanh chóng làm quen và phát triển game. Cộng đồng lớn của Unity cung cấp nhiều tài nguyên, diễn đàn và hướng dẫn, hỗ trợ người dùng trong quá trình phát triển. Unity còn tích hợp nhiều công cụ để phát triển đồ họa, vật lý và âm thanh, giúp nâng cao chất lượng game. Unity Asset Store cung cấp hàng ngàn tài nguyên và plugin mà người dùng có thể dễ dàng tích hợp vào dự án của mình. Đặc biệt, Unity có khả năng phát triển ứng dụng AR và VR mạnh mẽ, đáp ứng nhu cầu ngày càng cao trong ngành công nghiệp game và giải trí. Unity cũng thường xuyên cập nhật và cải tiến công cụ, giúp người dùng tiếp cận với những công nghệ mới nhất.

- Nhược điểm:

Tuy mạnh mẽ và dễ dàng tiếp cận như vậy, song Unity cũng có một số nhược điểm cần cân nhắc. Một số nhà phát triển phản ánh rằng Unity gặp khó khăn trong việc xử lý các dự án game lớn và phức tạp, đặc biệt là về khả năng tối ưu hóa hiệu năng. Chính sách phí mới của Unity dựa trên số lượt cài đặt game đã gây ra phản ứng mạnh mẽ từ phía các nhà phát triển nhỏ và vừa. Hệ thống rendering của Unity cũng không cạnh tranh bằng các engine

khác như Unreal Engine, đặc biệt đối với các dự án có yêu cầu đồ họa cao. Việc hợp tác trong Unity có thể gặp khó khăn nếu không sử dụng các công cụ quản lý tài nguyên bên ngoài, điều này có thể gây ra vấn đề trong việc chia sẻ code và asset giữa các thành viên trong team. Unity cũng có thể ngốn nhiều tài nguyên của máy, gây ra tình trạng giật, lag khi tải game. Cuối cùng, mã nguồn của Unity không được công bố kể cả cho những người dùng chấp nhận trả tiền, điều này có thể hạn chế khả năng tùy chỉnh sâu của các nhà phát triển.

- Giải pháp

Để khắc phục những nhược điểm của Unity, người phát triển có thể áp dụng một số giải pháp hiệu quả. Đầu tiên, về vấn đề hiệu suất không tối ưu trên các dự án lớn, có thể cải thiện bằng cách tối ưu hóa mã nguồn, giảm thiểu các phép tính nặng và sử dụng các thuật toán tối ưu hơn. Ngoài ra, sử dụng kỹ thuật culling và Level of Detail (LOD) để chỉ render những đối tượng gần camera, đồng thời tối ưu hóa bộ nhớ sẽ giúp tiết kiệm tài nguyên và cải thiện hiệu suất. Một phương pháp khác là chia nhỏ dự án thành các scene nhỏ hơn và nạp chúng khi cần thiết, thay vì tải toàn bộ dự án cùng lúc.

Đối với vấn đề xử lý đồ họa 3D hạn chế, người phát triển có thể áp dụng các kỹ thuật tối ưu hóa đồ họa như baking ánh sáng, giảm độ phân giải texture và sử dụng shader đơn giản hơn để giảm tải cho GPU. Việc điều chỉnh đồ họa cho từng nền tảng và sử dụng các tài nguyên bên ngoài từ Asset Store cũng là cách để cải thiện chất lượng hình ảnh mà vẫn duy trì hiệu suất.

Trong trường hợp các công cụ và tính năng nâng cao đắt tiền, một giải pháp hiệu quả là tận dụng các công cụ miễn phí hoặc mã nguồn mở thay thế cho các tính năng của Unity Pro. Nếu có ngân sách, người phát triển cũng nên đầu tư thông minh vào các gói trả phí, chỉ sử dụng những công cụ thật sự cần thiết cho dự án của mình.

Với những game 2D phức tạp, mặc dù Unity đã hỗ trợ khá tốt, nhưng người phát triển có thể sử dụng các plugin hoặc tài nguyên bổ sung từ Asset Store để tăng cường các tính năng. Cùng với đó, áp dụng các kỹ thuật tối ưu

hóa như sử dụng Tilemap, sprite batching và culling trong game 2D sẽ giúp nâng cao hiệu suất và giảm thiểu vấn đề phát sinh khi phát triển game phức tạp. Thường xuyên cập nhật Unity cũng giúp tận dụng những cải tiến mới nhất về công cụ 2D.

Cuối cùng, với yêu cầu về kỹ năng lập trình, người phát triển có thể sử dụng các framework và mẫu code có sẵn từ cộng đồng hoặc Asset Store để giảm bớt khối lượng công việc lập trình. Đồng thời, việc học thêm về lập trình C# qua các khóa học trực tuyến hoặc tài liệu chuyên sâu sẽ giúp người phát triển nắm vững các kỹ thuật cần thiết. Tham gia các cộng đồng hỗ trợ như StackOverflow hay Unity Forum cũng là một cách để học hỏi và giải quyết vấn đề nhanh chóng. Tất cả những giải pháp trên sẽ giúp người phát triển khắc phục các nhược điểm của Unity và tối ưu hóa quá trình phát triển game.

KẾT LUẬN

Trong thời đại hiện nay, song hành cùng việc xã hội ngày càng phát triển, con người dần hướng tới những thứ văn minh cao hơn thì nhu cầu giải trí, thỏa mãn những sở thích, đam mê cũng như sức khỏe tinh thần ngày càng được chú trọng. Việc phát triển game ngày càng được mở rộng và dần được chấp nhận là một nghề ở Việt Nam hiện nay. Cùng với đó sự phát triển của các công cụ phát triển game từ xây dựng, thiết kế đồ họa và âm thanh trong game đến các engine hỗ trợ lập trình game đã và đang thúc đẩy ngành công nghiệp game phát triển mạnh mẽ hơn nữa. Trong những năm vừa qua, dù chúng ta đã và đang bị ảnh hưởng bởi sự suy thoái do đại dịch Covid-19 mang lại, nhưng chúng ta vẫn có sự phát triển một cách mạnh mẽ không chỉ với các tựa game nước ngoài mà còn với các tựa game do các nhà phát triển trong nước làm ra; đã có rất nhiều tựa game Việt gây được tiếng vang lớn không chỉ ở trong nước mà còn ở thị trường quốc tế ví dụ như: Hoa, Thần Trùng, Tai Ương, Cỏ Máu, …. Việc đó không chỉ khẳng định chất lượng và tài năng của các nhà phát triển game Việt Nam, mà đó còn là nguồn động lực to lớn cho những nhà phát triển Việt khác và những sinh viên có đam mê về phát triển game, mong muốn theo đuổi và trở thành một người phát triển game đóng góp vào ngành công nghiệp phát triển game của nước ta.

Cùng với xu thế đó việc phát triển My’s Heart bằng Unity Engine bên cạnh việc luyện tập thêm về kỹ năng lập trình, còn là tiếp cận với các công cụ mà các engine game cung cấp, hiểu rõ hơn phần nào về quá trình phát triển các tựa game. Ngoài ra, thông qua dự án này em còn tiếp cập được thêm với các phần khác trong quy trình phát triển game, từ đó giúp em nhận ra việc phát triển một tựa game là sự tổng hợp của cả đội ngũ với rất nhiều thành phần từ lên ý tưởng, thiết kế đồ họa và âm thanh, tạo dựng chuyển động cho nhân vật, viết các kịch bản game, ….

Trong quá trình phát triển dự án tuy rằng đã học được thêm nhiều điều mới, xây dựng được một tựa game tương đối hoàn chỉnh và có thể chơi trên các thiết bị di động, song vẫn còn tồn tại nhiều điểm hạn chế như ý tưởng game còn nhiều thiếu sót, chủ yếu vẫn đang phát triển dựa trên những gì bản thân muốn, các đoạn

mã của game chưa được tối ưu tốt. Khá nhiều thành phần đồ họa và âm thanh vẫn phải sử dụng các tài nguyên của người khác; chính vì vậy rất nhiều thành phần về mặt đồ họa và âm thanh chưa thực sự đồng nhất với nhau. Các tính năng của game chưa thực sự ổn định, vẫn còn tồn tại khá nhiều lỗ hổng, Game chỉ mới có thể chạy được trên các điện thoại Android. Trong thời gian sắp tới, em sẽ cố gắng học hỏi thêm về những gì liên quan đến phát triển game từ đó phát triển ý tưởng này nhiều hơn nữa như: Việc xây dựng và thiết kế các boss quái vật và đa dạng các loại quái vật khác nhau giúp tăng trải nghiệm cho người chơi, củng cố lại level design giúp game có thể mở rộng và hướng tới nhiều tệp khách hàng hơn nữa. Cùng với đó là hệ thống phần thưởng và nhiệm vụ hằng ngày, và điều em hy vọng mình có thể hoàn thiện nó trong thời gian sắp tới là một cốt truyện nhỏ cho game, cùng với đó là cơ chế thu thập mảnh ghép ký ức và khám phá cốt truyện game; với nguyên tác ý tưởng thì game My’s Heart sẽ là một trong các mắt xích sự kiện của một cốt truyện lớn hơn. Song những điều đó có vẻ đang hơi quá sức đối với bản thân của em.

Còn ở thời điểm hiện tại, em rất chân thành cảm ơn thầy Lê Xuân Hùng, người đã trực tiếp hướng dẫn em trong quá trình làm đồ án tốt nghiệp này. Cùng với đó là lời cảm ơn gửi tới toàn thể giảng viên và nhân viên trường Đại học Công Nghiệp Hà Nội, chúc quý thầy cô cùng nhà trường ngày càng phát triển, luôn là một trong những nơi đào tạo nguồn nhân lực chất lượng cao cho nước ta. Một lần nữa em xin chân thành cảm ơn rất nhiều!

Em xin trân trọng cảm ơn!

Nguyễn Xuân Đạt

TÀI LIỆU THAM KHẢO

[1] Giáo trình phân tích thiết kế hướng đối tượng (Vũ Thị Dương, Phùng Đức Hòa, Nguyễn Thị Hương Lan - Trường Đại học Công nghiệp Hà Nội – NXB Khoa học và Kĩ thuật)

[2] Lập trình Game với Unity (Janine Suvak - Trường đại học FPT – NXB Bách Khoa Hà Nội)

[3] Sdl Game Development (Shaun Mitchell - Packt Publishing)

[4] Một số tài liệu khác trên Internet