

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM**



**BÁO CÁO ĐỒ ÁN 2
PHÁT TRIỂN VÀ TỐI ƯU GAME 2D PLATFORMER
TRÊN THIẾT BỊ DI ĐỘNG BẰNG UNITY ENGINE**

Giảng viên hướng dẫn : TS. Huỳnh Minh Đức

Sinh viên thực hiện 1 : Nguyễn Tiến Đạt - 21521942

Lớp : SE122.P21.PMCL

TP HCM, tháng 1 năm 2025

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

KHOA CÔNG NGHỆ PHẦN MỀM



BÁO CÁO ĐỒ ÁN 2

**PHÁT TRIỂN VÀ TỐI ƯU GAME 2D PLATFORMER
TRÊN THIẾT BỊ DI ĐỘNG BẰNG UNITY ENGINE**

Giảng viên hướng dẫn : TS. Huỳnh Minh Đức

Sinh viên thực hiện 1 : Nguyễn Tiến Đạt - 21521942

Lớp : SE122.P21.PMCL

TP HCM, tháng 1 năm 2025



ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỀ TÀI:

PHÁT TRIỂN VÀ TỐI UU GAME 2D PLATFORMER TRÊN THIẾT BỊ DI ĐỘNG
BẰNG UNITY ENGINE

Tên đề tài tiếng Anh:

DEVELOP AND OPTIMIZE 2D PLATFORMER GAME ON MOBILE DEVICES USING
UNITY ENGINE

Cán bộ hướng dẫn: TS. Huỳnh Minh Đức

Thời gian thực hiện: Từ ngày 11/9/2024 đến ngày 31/12/2024

Sinh viên thực hiện:

Nguyễn Tiến Đạt - 21521942

Nội dung đề tài:

1. Giới thiệu:

- Ngày nay, với sự phát triển mạnh mẽ của điện thoại thông minh, các trò chơi trên thiết bị di động chứng kiến một sự bùng nổ mạnh mẽ. Theo báo cáo của trang Udonis, thị trường game toàn cầu tạo ra doanh thu 187,7 tỷ USD, trong đó game mobile chiếm gần 93 tỷ USD tính đến năm 2023. Giữa những dòng game “hot” hiện nay thì tất nhiên không thể không kể đến thể loại 2D Platformer (hay còn gọi là game đi cảnh). Tuy vậy, để phát triển một game 2D Platformer vừa hấp dẫn về mặt nội dung, vừa tối ưu về hiệu năng trên nền tảng di động là một thách thức không nhỏ. Với sự hỗ trợ mạnh mẽ từ Unity Engine, một công cụ phát triển game đa nền tảng phổ biến, đồ án này tập trung vào quá trình xây dựng và trình bày các phương pháp để tối ưu một tựa game 2D Platformer trên thiết bị di động.
- Đề tài sẽ giải quyết các vấn đề về thiết kế gameplay, đồ họa, quản lý tài nguyên và tối ưu hóa hiệu năng để đảm bảo trải nghiệm mượt mà cho người chơi trên các thiết bị có cấu hình khác nhau. Đồng thời, đề tài cũng sẽ nghiên cứu và áp dụng các kỹ thuật tối ưu, từ việc tối ưu hóa mã nguồn, quản lý bộ nhớ, đến cải thiện tốc độ khung hình (FPS) trên Unity Engine.

2. Mục tiêu:

- **Xây dựng được một tựa game mobile đáp ứng các tiêu chí:**
 - UI/UX thân thiện, dễ thao tác, phù hợp với lối chơi trên màn hình cảm ứng.
 - Được tối ưu dung lượng, đảm bảo game chạy mượt mà trên mọi thiết bị di động.
 - Gameplay dễ dàng làm quen, có sự thử thách và mang tính chơi lại nhiều lần.
 - Có game-mechanics dễ học hỏi nhưng khó thành thạo.
 - Có hệ thống vật phẩm, nhờ vào các item thu thập được mà người chơi có thể đổi chúng để cường hóa nhân vật.
 - Có boss-fight đòi hỏi người chơi vận dụng kỹ năng, điều khiển nhân vật thành thạo để vượt qua.

3. Phạm vi:

- **Phạm vi môi trường:**
 - Triển khai sản phẩm trên môi trường di động.
- **Phạm vi chức năng:**
 - Thiết kế nhân vật chính với các khả năng di chuyển cơ bản như: chạy, nhảy, bám tường, lướt, leo trèo.
 - Thiết kế hệ thống kẻ thù hoạt động theo các hành vi được lập trình sẵn.
 - Thiết kế hệ thống chướng ngại vật gây trở ngại cho người chơi.
 - Thiết kế hệ thống phần thưởng trong quá trình chơi.
 - Thiết kế hệ thống vật phẩm đặc biệt hỗ trợ cho người chơi mà chỉ có thể đổi lấy bằng phần thưởng tích lũy trong quá trình chơi.
 - Cơ chế điều khiển dễ làm quen, mượt mà, phản hồi nhanh trên các thiết bị cảm ứng.
 - Xây dựng nhiều cấp độ khác nhau với độ khó tăng dần, có thời gian hoàn thành riêng, mở khóa cấp độ dựa trên thành tích.
 - Khả năng lưu tiến trình chơi và tiếp tục chơi từ vị trí đã lưu.

4. Đối tượng:

- Người chơi game muốn giải trí, khám phá game trên điện thoại.

5. Phương pháp thực hiện:

- Trải nghiệm các game platformer nổi tiếng, từ đó phân tích, xác định các tính năng cần triển khai.
- Lập kế hoạch, phân chia công việc theo các mốc thời gian.
- Tìm hiểu công nghệ sẽ sử dụng.
- Kiểm thử, phát hiện và sửa lỗi (nếu có).
- Xây dựng level.
- Báo cáo đồ án.

6. Công nghệ:

- Engine: Unity.
- Ngôn ngữ: C#.
- Management: Github, Notion.
- Tools: Aseprite, Firebase Analytics.
- IDE: VS2022.

7. Kết quả mong đợi

- Tạo được một game mobile với giao diện thân thiện, gameplay dễ làm quen, có sự thử thách và đánh khả năng của người chơi.
- Giúp người chơi có trải nghiệm mượt mà, ổn định.
- Bổ sung thêm một số game-mechanic hay.
- Quá trình thực hiện đạt đúng tiến độ của môn học.
- Ghi nhận được phản hồi tích cực thông qua Firebase Analytics.

8. Hướng phát triển của đề tài

- Mở rộng hệ thống level, map, nâng cấp thể loại game thành Metroidvania.
- Thêm các loại quái, vật cản với cơ chế mới, thêm nhiều Boss-fight trong game.
- Mở rộng hệ thống phần thưởng và vật phẩm.
- Thêm một vài game-mechanic mới.

Thời gian	Công việc thực hiện
Từ 11/09/2024 đến 17/09/2024	- Chuyển thể phiên bản PC của game sang phiên bản mobile.
Từ 18/09/2024 đến 18/10/2024	- Sửa lỗi trong quá trình chuyển đổi phiên bản, thêm các điều khiển trên thiết bị di động.
Từ 19/10/2024 đến 31/10/2024	- Xây dựng lại hệ thống level, hệ thống vật phẩm, phần thưởng, thêm hệ thống Shop.
Từ 11/11/2024 đến 28/11/2024	- Sửa các lỗi phát sinh qua các lần test trong game, xây dựng hệ thống lưu trữ dữ liệu người chơi qua các phiên chơi.
Từ 5/12/2024 đến 20/12/2024	- Xây dựng thêm 3 levels mới, tiếp tục sửa các lỗi phát sinh trong quá trình test.
Từ 31/12/2024 đến 02/01/2025	- Xây dựng boss-fight, refactor code, config lại các chỉ số của quái.
Từ 04/01/2025 đến 06/01/2025	- Hoàn thiện game, viết báo cáo và chuẩn bị các tài liệu liên quan trong báo cáo.

ĐỀ CƯƠNG CHI TIẾT	3
LỜI CẢM ƠN	10
NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN	11
Chương 1. GIỚI THIỆU ĐỀ TÀI VÀ KHÁI NIỆM	12
1.1 Thế nào là game Platformer ?	12
1.2 Thế nào là thiết kế game ?	13
1.3 Lý do chọn đề tài	13
1.4 Phạm vi đề tài	14
1.5 Chức năng	14
1.6 Công nghệ sử dụng	14
1.7 Thông tin người thực hiện đề tài	15
Chương 2. CƠ SỞ LÝ THUYẾT	16
2.1 Unity Engine	16
2.2 C#	18
2.3 Github	19
2.4 Notion	20
2.5 Aseprite	21
2.6 Firebase Analytics	22
2.7 Visual Studio 2022	23
3. PHÂN TÍCH VÀ THIẾT KẾ	24
3.1 Gameflow	24
3.2 Sơ đồ lớp	25
3.3 Sơ đồ trạng thái (State)	26
Chương 4. THIẾT KẾ GAME	28
4.1 Thế nào là thiết kế Game-mechanic(Cơ chế game)	28
4.2 Thế nào là Game Element(Thành phần game)	28
4.3 Level Design(Thiết kế level game)	29
4.4 Một vài thiết kế đã áp dụng trong game	30
Chương 5. THIẾT KẾ GIAO DIỆN	34
5.1 Danh sách màn hình	34
5.2 Chi tiết màn hình	35
5.2.1 Màn hình chính	35
5.2.2 Màn hình “Chọn màn chơi”	36
5.2.3 Màn hình “Chi tiết màn chơi”	37
5.2.4 Màn hình Shop	38
5.2.5 Màn hình “Chi tiết vật phẩm”	39
5.2.6 Màn hình “Âm thanh”	39
5.2.7 Màn hình “Thông tin thêm”	40

5.2.8 Màn hình “Màn chơi bất kỳ”	41
5.2.9 Màn hình “Kết quả”	42
5.2.10 Màn hình “Boss fight”	43
Chương 6. MỘT SỐ KỸ THUẬT TỐI ƯU GAME 2D TRÊN UNITY ENGINE	44
6.1 Đặt vấn đề	44
6.2 Các phương pháp tối ưu	44
6.2.1 Kỹ thuật Object Pooling.	44
6.2.2 Kỹ thuật chia sẻ Data với Scriptable Object.	45
6.2.3 Sử dụng Sprite Atlas để giảm draw calls.	46
6.2.4 Sử dụng Frame Debugger để giảm draw calls.	46
6.2.5 Hạn chế sử dụng các thành phần sau	47
6.2.6 Tối ưu Tilemap Collider kết hợp Collider2D	47
Chương 7. DESIGN PATTERN SỬ DỤNG	49
7.1 Thế nào là Design Pattern ?	49
7.2 Các Design Pattern sử dụng trong game và kinh nghiệm	49
Chương 8. KẾT LUẬN	52
8.1 Ưu điểm	52
8.2 Nhược điểm	52
8.3 Hướng phát triển	52
Chương 9. TÀI LIỆU THAM KHẢO VÀ MÃ NGUỒN	53
1. Tham Khảo	53
1.1. Lý thuyết về Unity 2D.	53
1.2. Lý thuyết về các pattern trong lập trình game.	53
1.3. Cộng đồng người dùng Unity Việt Nam.	53
1.4. Chia sẻ về kỹ thuật trong lập trình game bởi người Việt	53
1.5. Lý thuyết State Pattern trong lập trình game	53
1.6. Diễn đàn Unity	53
1.7. Diễn đàn Game Developer	53
1.8. Framework Game 2D thầy Dũng Đinh (nền tảng lập trình game)	53
2. Mã Nguồn	53

LỜI CẢM ƠN

Đồ án này là đồ án đầu tiên em thực hiện trong môn Đồ Án của trường, dựa trên kinh nghiệm lập trình game được tích lũy từ trước. Qua quá trình cố gắng tìm tòi, học hỏi kiến thức khi làm đồ án, em đã học hỏi được rất nhiều điều về thiết kế game, học được cách làm việc hiệu quả. Em xin gửi lời cảm ơn chân thành đến:

- ❖ Các quý thầy cô trường Đại học Công nghệ Thông tin nói chung và thầy Huỳnh Minh Đức nói riêng, đã tận tình giúp đỡ, truyền đạt những kiến thức bổ ích cho em, đồng thời luôn lắng nghe ý kiến và góp ý cho em ngày một phát triển hơn.
- ❖ Cuối cùng, xin cảm ơn gia đình, anh chị em, bạn bè là hậu phương vững chắc cho em yên tâm tập trung hoàn thành đồ án.

Thành phố Hồ Chí Minh, ngày 04 tháng 1 năm 2025

Sinh viên thực hiện

Nguyễn Tiên Đạt

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Tp.HCM, ngày ... tháng ... năm 20....

GVHD

TS. Huỳnh Minh Đức

Chương 1. GIỚI THIỆU ĐỀ TÀI VÀ KHÁI NIỆM

1.1 Thế nào là game Platformer ?

Bắt đầu từ những năm 80 của thế kỷ trước, trò chơi điện tử nổi lên như một hiện tượng toàn cầu, đặc biệt với sự ra đời của các máy chơi game như Atari 2600, Nintendo Entertainment System (NES). Trong số những trò chơi thời ấy, có thể kể đến những tượng đài trong dòng game 2D Platformer nổi tiếng như Super Mario Bros., Megaman, Donkey Kong, ... Dưới đây là một vài đặc điểm của dòng này:

- Thế giới trong game chịu ảnh hưởng của các định luật vật lý cơ bản ngoài đời thực như: ma sát, trọng lực, va chạm, lực đẩy, quán tính, ...
- Nhân vật chính sẽ có các kỹ năng chạy, nhảy, ... và dùng chúng để vượt qua các chướng địa hình (platform) hay chướng ngại vật hoặc kẻ địch.
- Có nhiều level (màn chơi), từng màn chơi là từng bối cảnh khác và độ khó cũng tăng dần, có cốt truyện dẫn dắt người chơi, có những phần thưởng cho người chơi trong quá trình chơi game.



Tựa game Super Mario Bros. 3.



Tựa game Megaman X4 (hay còn gọi là Rockman).

1.2 Thế nào là thiết kế game ?

Trong quá trình phát triển game, thiết kế game là một điều không thể thiếu, nếu coi như việc xây dựng mã nguồn, vẽ bối cảnh, nhân vật là khung xương, da thịt thì việc thiết kế game có thể coi như là thổi hồn vào cho nó. Về cơ bản, thiết kế game là quá trình tạo ra cấu trúc, cơ chế và trải nghiệm của một trò chơi. Nó yêu cầu sự kết hợp giữa nghệ thuật, công nghệ và kỹ năng giải quyết vấn đề. Thiết kế game bao gồm việc lập kế hoạch cho cách chơi (gameplay), mục tiêu mà người chơi cần đạt được, bối cảnh game và cơ chế điều khiển để tương tác với thế giới trong game. Ở [Chương 4](#), chúng ta sẽ đi sâu hơn về phần này, còn dưới đây là các thành phần chính:

- Cơ chế chơi (Game Mechanics): Là các quy tắc và hệ thống xác định cách trò chơi hoạt động. Cơ chế chơi thường bao gồm các hành động mà người chơi có thể thực hiện, như nhảy, tấn công, giải đố, hoặc tương tác với môi trường.
- Thiết kế cấp độ (Level Design): Là quá trình xây dựng các không gian hoặc môi trường mà người chơi sẽ tương tác trong suốt quá trình chơi. Thiết kế cấp độ yêu cầu tạo ra các thử thách hợp lý, sự tiến bộ qua từng cấp, và tạo cảm giác thú vị cho người chơi.
- Cốt truyện và bối cảnh (Storytelling): Đôi với nhiều game, cốt truyện đóng vai trò làm nền tảng để hướng dẫn và kết nối các sự kiện, hành động trong trò chơi. Bối cảnh cũng đóng vai trò quan trọng trong việc tạo ra môi trường, thế giới mà người chơi sẽ khám phá.
- Thiết kế giao diện, trải nghiệm (UI/UX): Tập trung vào việc làm sao để người chơi cảm thấy thoải mái khi tương tác với game bao gồm bố trí các thành phần trên giao diện, âm thanh sống động và tính dễ tiếp cận.

1.3 Lý do chọn đề tài

Với sự phát triển như vũ bão của công nghệ hiện nay, thị trường game đặc biệt là game di động được hưởng thành quả lớn. Thống kê từ trang [Udonis](#) cho thấy

có khoảng 2,5 tỷ người chơi game di động ở toàn cầu và lượng doanh thu của nó chiếm gần một nửa so với tổng doanh thu game nói chung. Chính vì thế, em chọn việc phát triển một tựa game 2D trên thiết bị di động và cũng là bàn đạp để theo đuổi sự nghiệp phát triển game sau này.

1.4 Phạm vi đề tài

- Triển khai ứng dụng trên môi trường di động (hiện tại là thiết bị có hệ điều hành Android).

1.5 Chức năng

- Thiết kế nhân vật chính với các khả năng di chuyển cơ bản như: chạy, nhảy, bám tường, lướt, leo trèo.
- Thiết kế hệ thống kẻ thù hoạt động theo các hành vi được lập trình sẵn.
- Thiết kế hệ thống chướng ngại vật gây trở ngại cho người chơi.
- Thiết kế hệ thống phần thưởng trong quá trình chơi.
- Thiết kế hệ thống vật phẩm đặc biệt hỗ trợ cho người chơi mà chỉ có thể đổi lấy bằng phần thưởng tích lũy trong quá trình chơi.
- Cơ chế điều khiển dễ làm quen, mượt mà, phản hồi nhanh trên các thiết bị cảm ứng.
- Xây dựng nhiều cấp độ khác nhau với độ khó tăng dần, có thời gian hoàn thành riêng, mở khóa cấp độ dựa trên thành tích.
- Khả năng lưu tiến trình chơi và tiếp tục chơi từ vị trí đã lưu.

1.6 Công nghệ sử dụng

- Unity Engine: Hỗ trợ phát triển game cross - platform (đa nền tảng).
- C#: Ngôn ngữ lập trình chính mà Unity hỗ trợ.
- Github: Quản lý mã nguồn.
- Notion: Tổ chức, quản lý và sắp xếp kế hoạch cho dự án.
- Aseprite: Công cụ custom, chỉnh sửa, vẽ pixel art.
- Firebase Analytics: Dùng để thống kê, phân tích hành vi người dùng, qua đó giúp cải thiện game.
- VS2022: IDE (môi trường phát triển tích hợp) dùng để viết, chỉnh sửa, gỡ lỗi mã nguồn.

1.7 Thông tin người thực hiện đề tài

STT	Tên thành viên	Mã sinh viên	Email
1	Nguyễn Tiên Đạt	21521942	21521942@gm.uit.edu.vn

Thông tin thành viên nhóm

Chương 2. CƠ SỞ LÝ THUYẾT

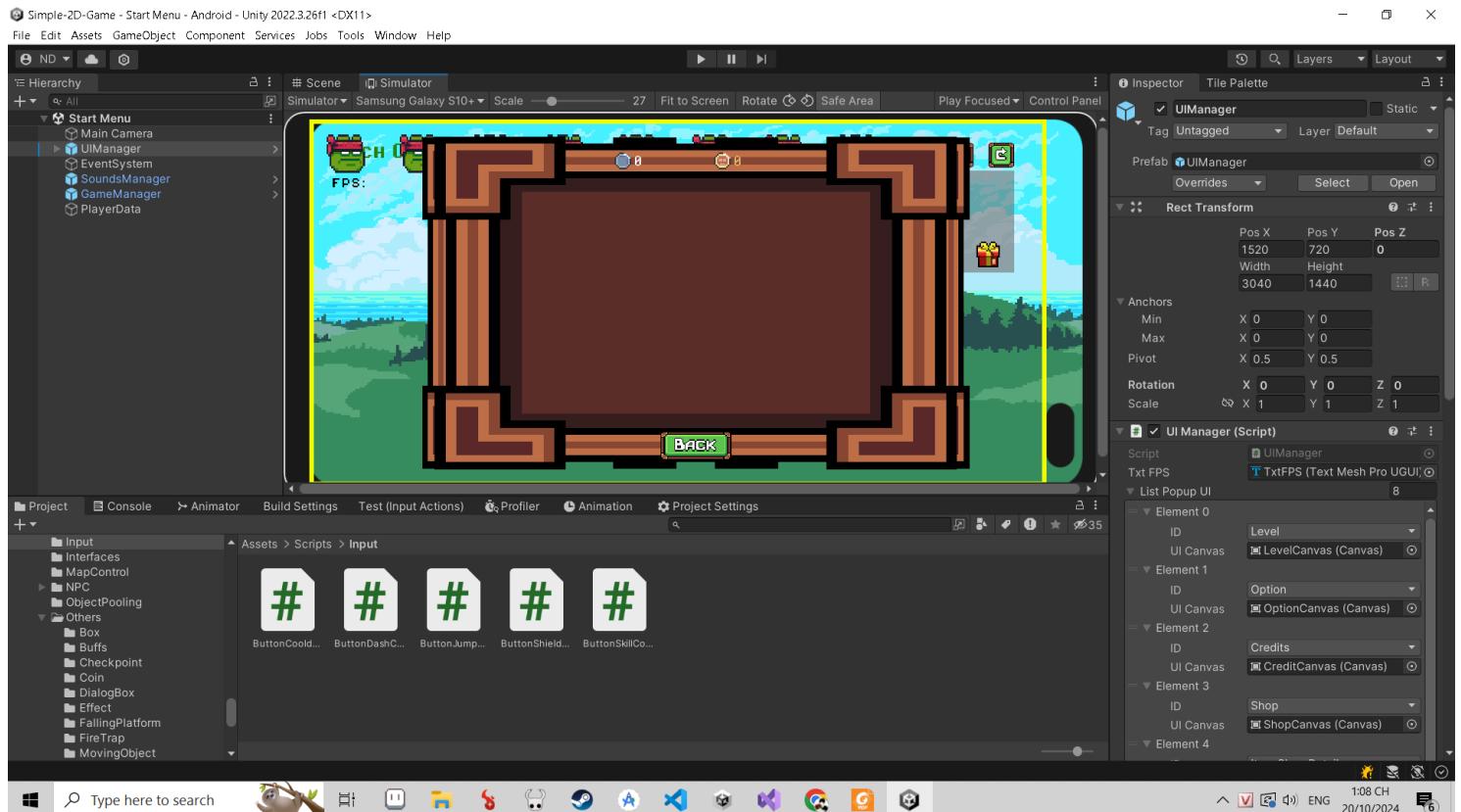
2.1 Unity Engine

Unity Engine là một game engine đa nền tảng được phát triển bởi công ty Unity Technologies dùng để hỗ trợ tối đa cho việc phát triển game 2D, 3D cho lập trình viên cũng như các nhà thiết kế game.

Giao diện và không gian làm việc dễ làm quen kết hợp với cộng đồng người sử dụng rất nhiều và nguồn tài liệu phong phú làm cho game engine này trở thành một trong những game engine hàng đầu trong việc phát triển game.

Với tính năng cross-platform giúp cho lập trình viên chỉ cần lập trình một lần nhưng có thể build cho nhiều hệ điều hành, platform khác nhau như:

- PC: Windows, Mac, Linux,...
- Mobile: Android, iOS,...



Giao diện làm việc của Unity Engine

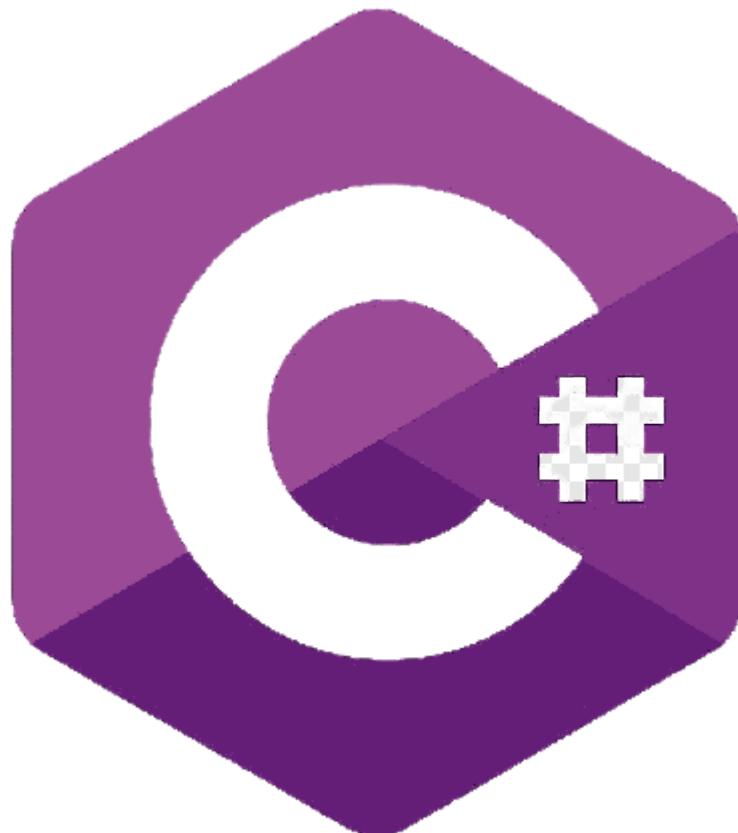
Các khái niệm cơ bản trong một project Unity bao gồm:

- Assets: Là những tài nguyên được sử dụng trong dự án, nó có thể là hình ảnh 2D, models 3D, âm thanh, hiệu ứng, mã nguồn, ... Unity có thư viện Asset Store - nơi chứa rất nhiều assets miễn phí hoặc trả phí, phù hợp cho những người mới học và tìm hiểu cũng như hỗ trợ asset cho các team, studio làm game.
- GameObject: Là một đối tượng trong Unity. Đối tượng này có thể là nhân vật chính, kẻ thù, bẫy, hiệu ứng, phần thưởng, ... Điểm chung của chúng là đều có component Transform đi kèm.
- Components: Là các thuộc tính của GameObject (có thể được thêm vào hoặc loại bỏ đi) như Sprite (hiển thị hình ảnh 2D), âm thanh, Rigidbody (Để Unity thêm GameObject này vào hệ thống vật lý của nó), ...
- Scripts: Là phần mã để điều khiển, tương tác cho người chơi cũng như cho cả hệ thống game hoạt động. Unity hỗ trợ viết script bằng ngôn ngữ thuận hướng đối tượng là C#.
- Scene: Là một cảnh trong game. Đây là nơi lập trình viên sẽ sắp xếp, chỉnh sửa các GameObjects cũng như xây dựng UI/UX.
- Prefabs: Là một GameObject hoàn chỉnh được lưu trữ để tái sử dụng nhiều lần.
- Camera: Dùng để thể hiện khung hình, là góc nhìn mà người chơi thấy được trong game.
- Hierarchy: Là cửa sổ chứa tất cả các GameObject hiện có trong Scene, cũng là nơi để tạo các GameObject mới.
- Simulator: Là cửa sổ cho phép lập trình viên chạy game trên nền tảng họ muốn (PC, Mobile) mà không cần phải build game ra - đây là một trong những điểm mạnh của Unity Engine.
- Project: Là cửa sổ chứa toàn bộ thư mục của dự án, trong từng thư mục sẽ chứa các thành phần cần thiết làm nên dự án.
- Console: Là cửa sổ hỗ trợ lập trình viên debug trong quá trình phát triển game.
- ...

2.2 C#

Là một ngôn ngữ lập trình đa năng, được phát triển bởi Microsoft vào đầu những năm 2000 như một phần của nền tảng .NET. Với cú pháp rõ ràng và dễ hiểu, C# nhanh chóng trở thành một trong những ngôn ngữ lập trình phổ biến nhất cho việc phát triển ứng dụng trên nhiều nền tảng, bao gồm desktop, web, và đặc biệt là game. C# được thiết kế để hỗ trợ lập trình hướng đối tượng, cho phép lập trình viên xây dựng các ứng dụng phức tạp với các thành phần có thể tái sử dụng và mở rộng.

Ngôn ngữ này nổi bật với khả năng tích hợp mạnh mẽ vào các công cụ phát triển của Microsoft, chẳng hạn như Visual Studio, mang lại cho lập trình viên một môi trường lập trình mạnh mẽ và thân thiện. Bên cạnh đó, C# còn hỗ trợ nhiều tính năng tiên tiến như LINQ (Language Integrated Query), giúp truy vấn dữ liệu một cách dễ dàng và hiệu quả, cùng với tính năng async/await cho lập trình bất đồng bộ, giúp cải thiện hiệu suất ứng dụng.



Logo của C#

2.3 Github

Là một nền tảng web nổi tiếng, được ra mắt vào năm 2008, dành cho việc lưu trữ và quản lý mã nguồn, sử dụng hệ thống kiểm soát phiên bản Git. Nó đã nhanh chóng trở thành một công cụ thiết yếu cho lập trình viên và các nhóm phát triển phần mềm, giúp họ hợp tác và làm việc hiệu quả hơn.

GitHub cho phép người dùng tạo và quản lý các repository (kho lưu trữ) cho mã nguồn, từ đó dễ dàng theo dõi các thay đổi, xem lịch sử và quay lại các phiên bản trước đó. Nền tảng này cũng hỗ trợ việc hợp tác nhóm qua tính năng pull requests, cho phép nhiều lập trình viên cùng làm việc trên một dự án, đề xuất thay đổi và thảo luận mã trước khi hợp nhất vào nhánh chính.

Ngoài ra, GitHub còn cung cấp hệ thống theo dõi vấn đề (issues) để ghi chú và quản lý các lỗi, yêu cầu tính năng hoặc nhiệm vụ trong dự án, cùng với khả năng tạo tài liệu chi tiết thông qua file README.md, giúp người khác dễ dàng hiểu và sử dụng mã nguồn.



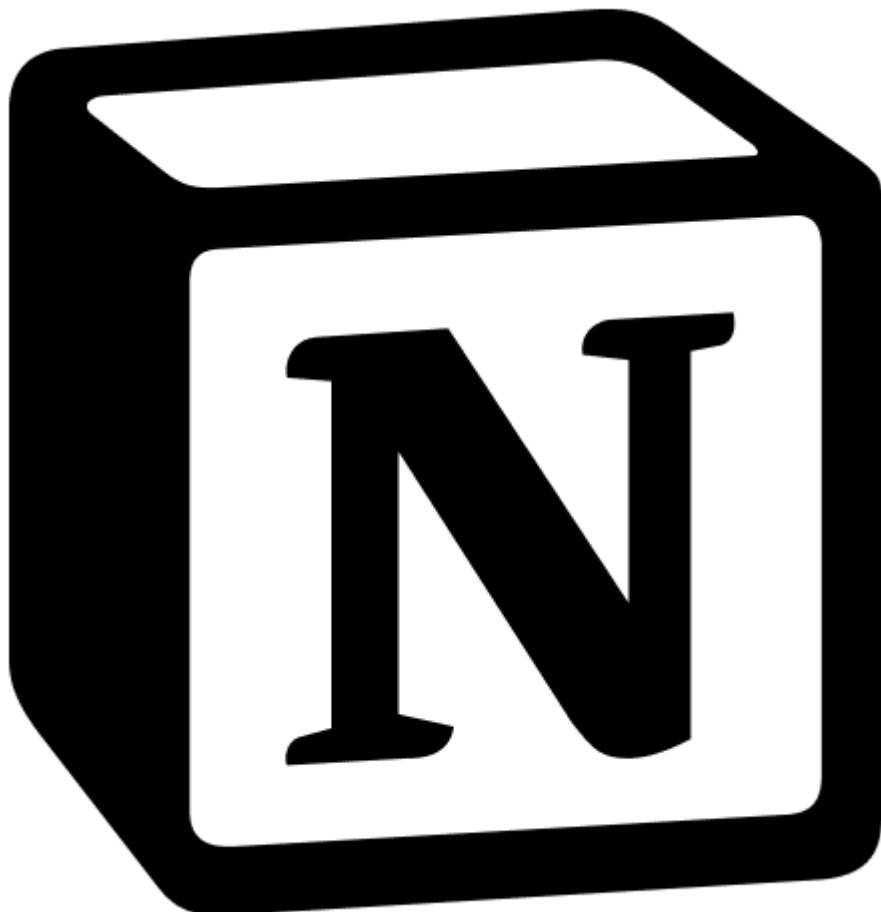
Logo của GitHub.

2.4 Notion

Là một công cụ quản lý và tổ chức thông tin đa năng, kết hợp nhiều tính năng như ghi chú, quản lý dự án, lập kế hoạch, và tạo cơ sở dữ liệu. Được phát hành lần đầu vào năm 2016, Notion nhanh chóng thu hút sự quan tâm của người dùng nhờ vào giao diện thân thiện và tính linh hoạt cao, cho phép người dùng tùy chỉnh không gian làm việc của mình theo cách mà họ mong muốn.

Một trong những điểm mạnh của Notion là khả năng kết hợp nhiều loại nội dung trong cùng một trang, từ văn bản, hình ảnh đến bảng biểu và danh sách kiểm tra. Người dùng có thể tạo các trang con, sử dụng các mẫu sẵn có, hoặc xây dựng từ đầu để phù hợp với nhu cầu cá nhân hoặc nhóm. Ngoài ra, Notion cho phép người dùng chia sẻ và hợp tác với nhau, giúp việc làm việc nhóm trở nên dễ dàng và hiệu quả hơn.

Notion cũng tích hợp tính năng quản lý dự án thông qua việc tạo bảng Kanban, lịch trình và theo dõi tiến độ công việc, giúp người dùng có cái nhìn tổng quan về các nhiệm vụ và thời hạn. Khả năng tạo cơ sở dữ liệu mạnh mẽ cho phép người dùng lưu trữ, tổ chức và tìm kiếm thông tin một cách dễ dàng.



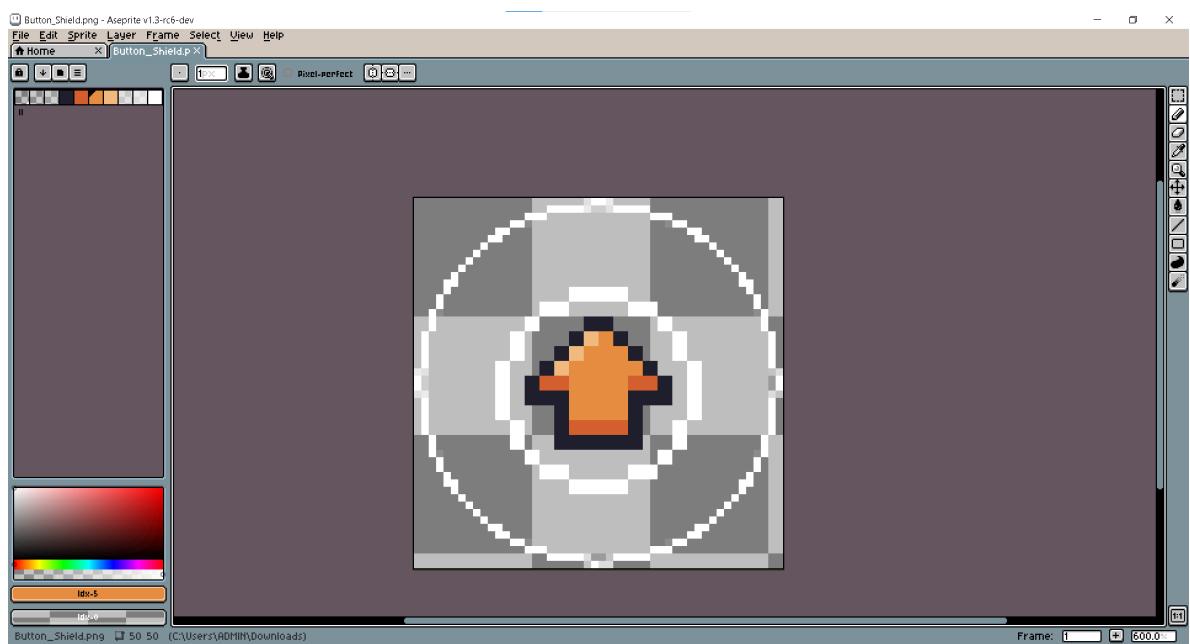
Logo của Notion.

2.5 Aseprite

Là một phần mềm đồ họa pixel art chuyên dụng, được thiết kế để giúp người dùng tạo ra các hình ảnh và hoạt ảnh theo phong cách pixel, phổ biến trong game 2D. Được phát triển bởi David Capello, Aseprite đã trở thành một công cụ yêu thích trong cộng đồng game developer, nghệ sĩ và nhà thiết kế đồ họa nhờ vào giao diện thân thiện và các tính năng mạnh mẽ.

Phần mềm này cho phép người dùng dễ dàng tạo, chỉnh sửa và xuất hình ảnh pixel với độ chính xác cao. Aseprite hỗ trợ nhiều công cụ vẽ, từ cọ vẽ, tô màu đến các công cụ chọn lựa, giúp người dùng thao tác nhanh chóng và hiệu quả. Một trong những tính năng nổi bật của Aseprite là khả năng tạo hoạt ảnh, cho phép người dùng tạo và quản lý các khung hình (frames) trong một dự án hoạt ảnh, từ đó tạo ra những chuyển động mượt mà và sống động.

Aseprite cũng hỗ trợ nhiều định dạng tệp khác nhau, bao gồm GIF, PNG và các định dạng ảnh raster khác, giúp người dùng dễ dàng xuất các tác phẩm của mình để sử dụng trong game hoặc chia sẻ với cộng đồng. Ngoài ra, phần mềm còn tích hợp các tính năng như palette quản lý màu sắc, layer để tổ chức các yếu tố đồ họa, và tính năng onion skinning, giúp người dùng theo dõi các khung hình trước và sau trong quá trình tạo hoạt ảnh.



Màn hình làm việc của Aseprite.

2.6 Firebase Analytics

Là một dịch vụ phân tích ứng dụng di động miễn phí do Google phát triển, nhằm giúp các nhà phát triển hiểu rõ hơn về hành vi của người dùng trong ứng dụng của họ. Nằm trong hệ sinh thái Firebase, một nền tảng phát triển ứng dụng đa chức năng, Firebase Analytics cung cấp các công cụ mạnh mẽ để thu thập, phân tích và báo cáo dữ liệu người dùng.

Một trong những điểm nổi bật của Firebase Analytics là khả năng theo dõi các sự kiện trong ứng dụng một cách tự động và tùy chỉnh. Nhà phát triển có thể định nghĩa các sự kiện như lượt xem màn hình, tương tác với nút, hoặc hoàn thành các nhiệm vụ cụ thể. Điều này cho phép họ theo dõi cách người dùng tương tác với ứng dụng và xác định các điểm mạnh, điểm yếu trong trải nghiệm người dùng.



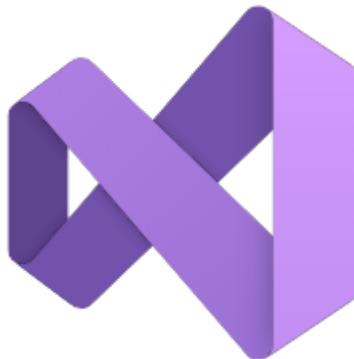
Logo của Firebase

2.7 Visual Studio 2022

Là phiên bản mới nhất của môi trường phát triển tích hợp (IDE) mạnh mẽ do Microsoft phát triển, phục vụ cho lập trình viên trong việc xây dựng ứng dụng cho nhiều nền tảng khác nhau, bao gồm ứng dụng desktop, web, di động và game. Được ra mắt vào tháng 8 năm 2021, Visual Studio 2022 đã mang lại nhiều cải tiến và tính năng mới, giúp tối ưu hóa quy trình phát triển và tăng cường hiệu suất làm việc của lập trình viên.

Một trong những điểm nổi bật của Visual Studio 2022 là việc chuyển sang kiến trúc 64-bit, cho phép IDE xử lý các dự án lớn hơn mà không gặp phải giới hạn về bộ nhớ. Điều này mang lại trải nghiệm mượt mà hơn, đặc biệt khi làm việc với các dự án phức tạp hoặc khi sử dụng nhiều tính năng cùng lúc. Giao diện người dùng cũng đã được cải tiến với thiết kế hiện đại, giúp người dùng dễ dàng tìm kiếm và truy cập các công cụ và tính năng.

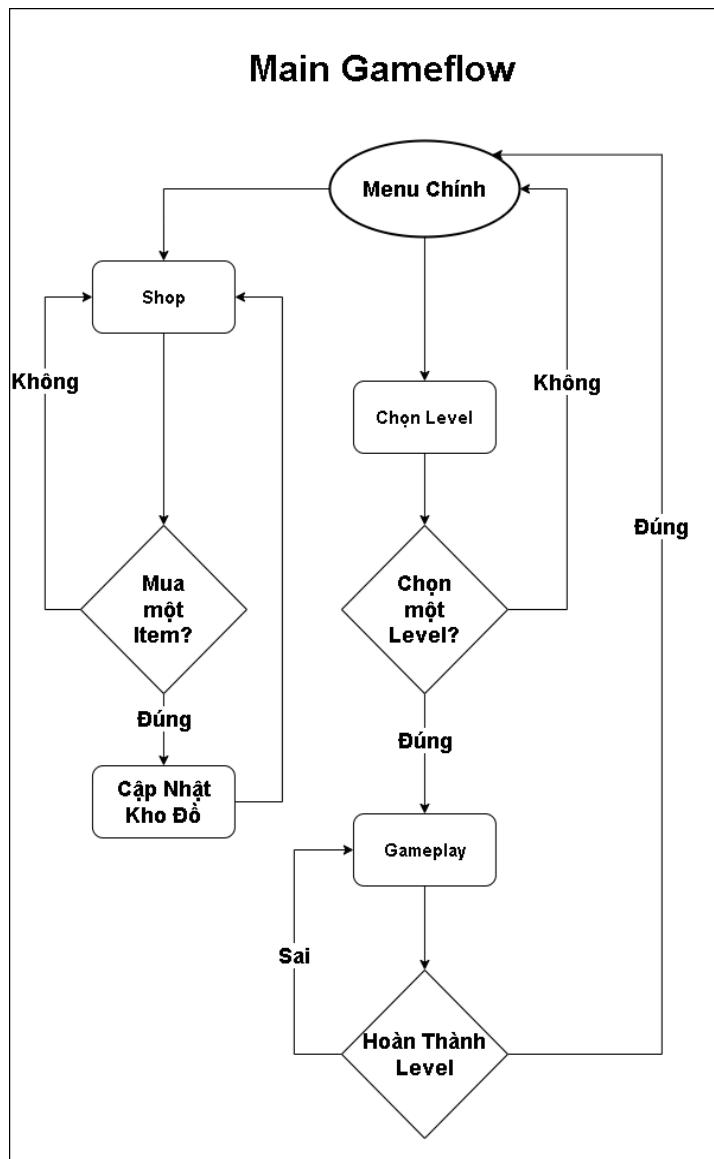
Visual Studio 2022 hỗ trợ nhiều ngôn ngữ lập trình, bao gồm C#, C++, Python, JavaScript và TypeScript, cho phép lập trình viên làm việc trong môi trường quen thuộc. Các tính năng như IntelliCode cung cấp gợi ý thông minh cho mã lập trình, giúp tăng tốc quá trình viết mã và giảm thiểu lỗi. Hệ thống quản lý lỗi và gỡ lỗi được cải tiến, cho phép lập trình viên dễ dàng xác định và khắc phục các vấn đề trong mã của mình.



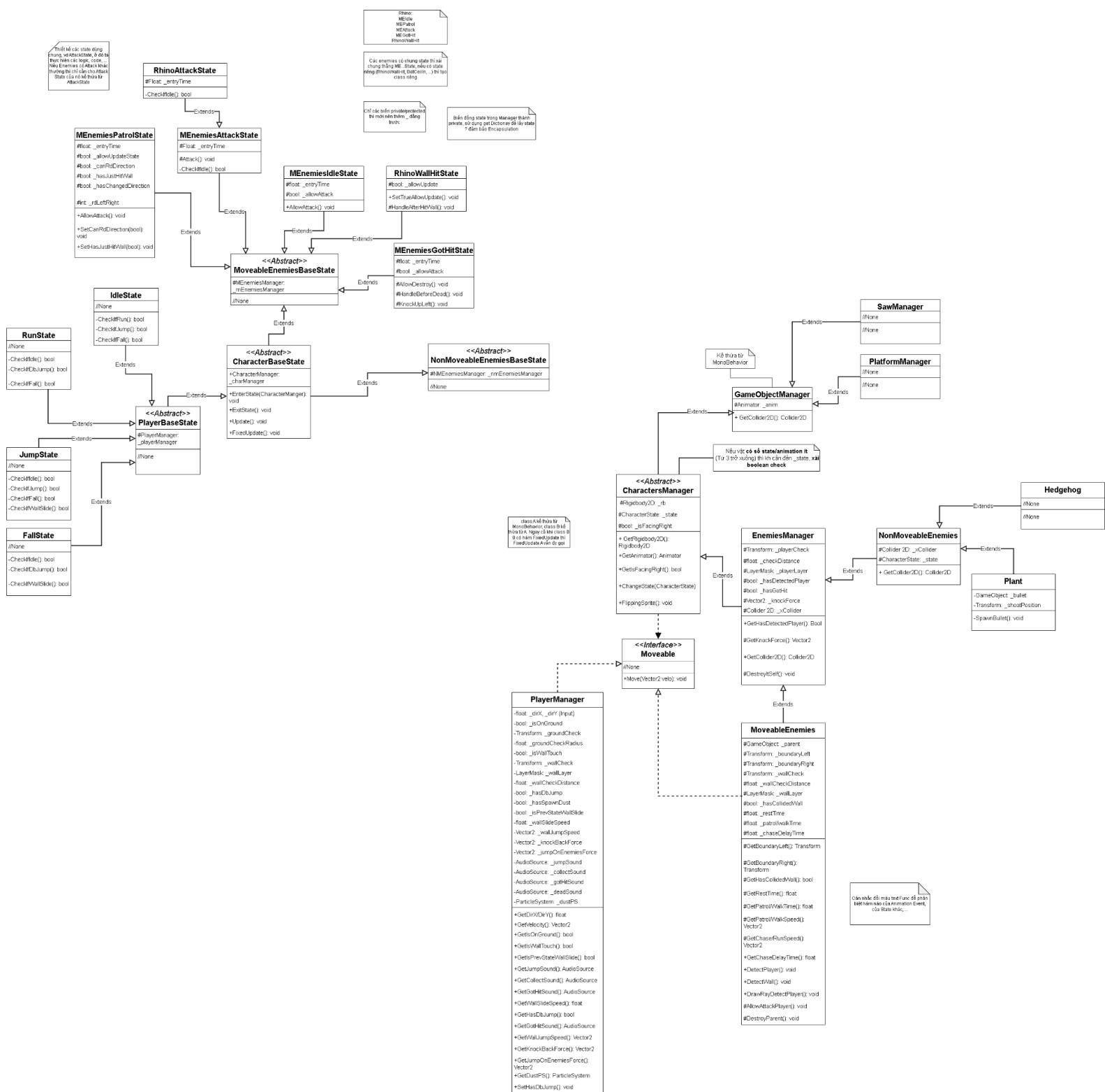
Logo của Visual Studio 2022

3. PHÂN TÍCH VÀ THIẾT KẾ

3.1 Gameflow



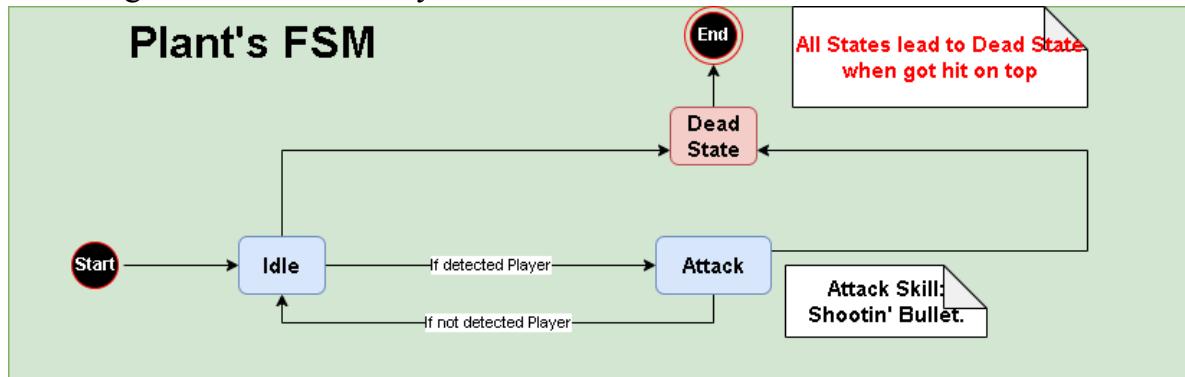
3.2 Sơ đồ lớp



Sơ đồ lớp tổng quát.

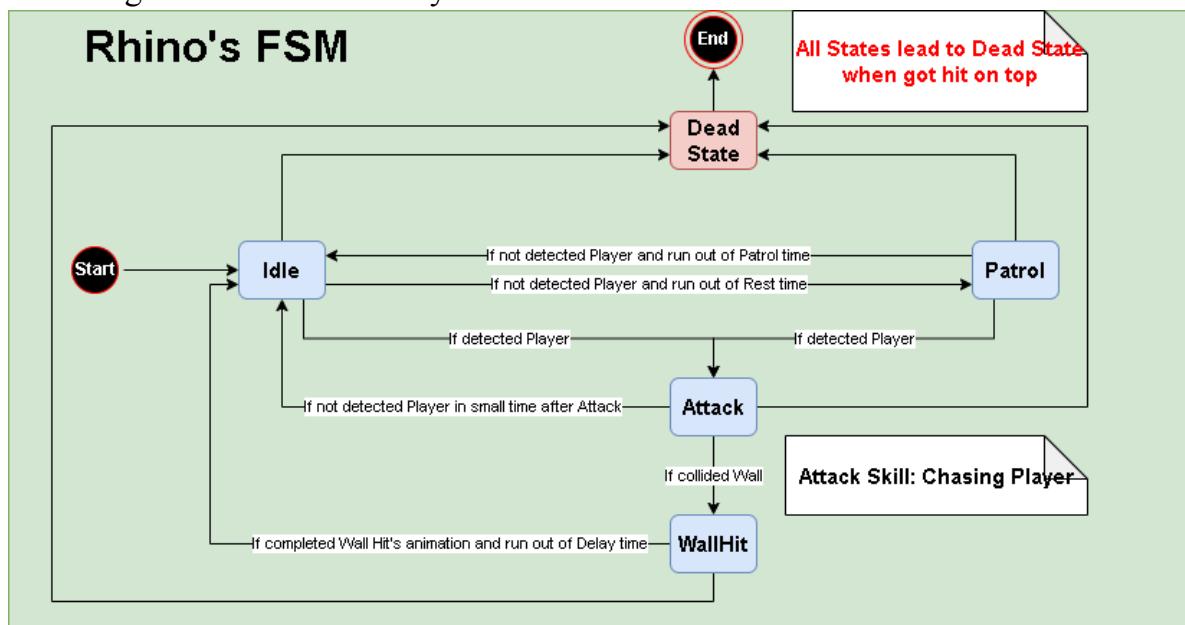
3.3 Sơ đồ trạng thái (State)

- State diagram của Plant enemy



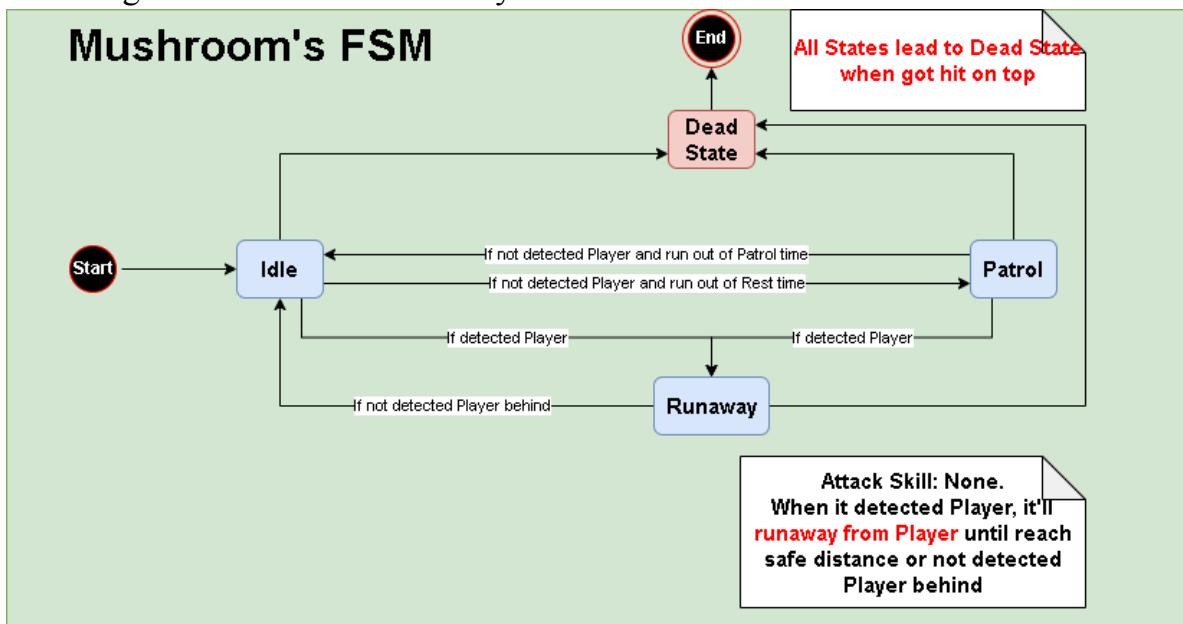
State diagram Plant.

- State diagram của Rhino enemy



State diagram Rhino.

- State diagram của Mushroom enemy



State diagram Mushroom.

Chương 4. THIẾT KẾ GAME

4.1 Thế nào là thiết kế Game-mechanic(Cơ chế game)

Game Mechanic(Cơ chế chơi) là các phương thức kết hợp các yếu tố trong game (Game Element) để định nghĩa và hỗ trợ cho Gameplay. Game Mechanic xác định trò chơi sẽ diễn ra như thế nào. Nó bao gồm các hành động mà người chơi có thể thực hiện và cách chúng ảnh hưởng đến thế giới trong game. Game mechanic có những đặc điểm sau:

- Mô tả cách chơi: Các cơ chế định nghĩa cách người chơi điều khiển nhân vật hoặc tác động đến môi trường trong game.
- Tương tác: Liên quan đến các hành động cụ thể như nhảy, bắn, chạy, giải đố, hoặc tương tác với đối tượng.
- Tính năng: Mỗi game mechanic mang lại trải nghiệm độc đáo, như chiến đấu, crafting, hoặc stealth.

Người
gây nguy

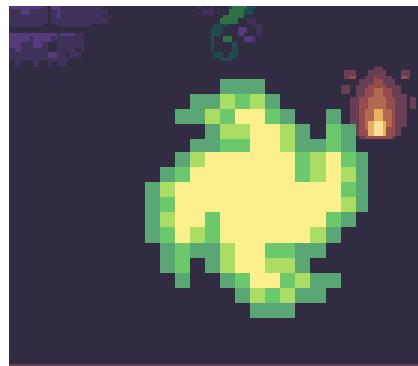
Hình dưới là minh họa một game mechanic là cơ chế Shield(khiên) trong game của em. Người chơi bấm sử dụng kỹ năng này sẽ giúp họ tránh được mọi đòn đánh hoặc một vật cản hiểm trong thời gian ngắn sau khi kích hoạt.



Khiên giúp bảo vệ người chơi.

4.2 Thế nào là Game Element(Thành phần game)

Game Element có thể hiểu một cách đơn giản là các thành phần tạo nên game của bạn. Trong game, các Game Element chính có thể kể đến như: nhân vật, trang bị, các yếu tố môi trường, UI,... Một Game Element có thể được tác động bởi nhiều Game Mechanic khác nhau. Giống như việc cùng một loại nguyên liệu, người đầu bếp có thể chế biến nhiều món ăn khác nhau (bò xào, bò hàm,...). Tương tự thế, một Game Mechanic có thể tác động lên nhiều Game Element. Giống như việc cùng một cách chế biến, đầu bếp có thể sử dụng trên nhiều nguyên liệu giống nhau (rau xào, thịt xào, mì xào,...). Hình dưới là chiếc cổng dẫn đến level khác trong game - là một trong những Game Element.



Chiếc cổng trong game.

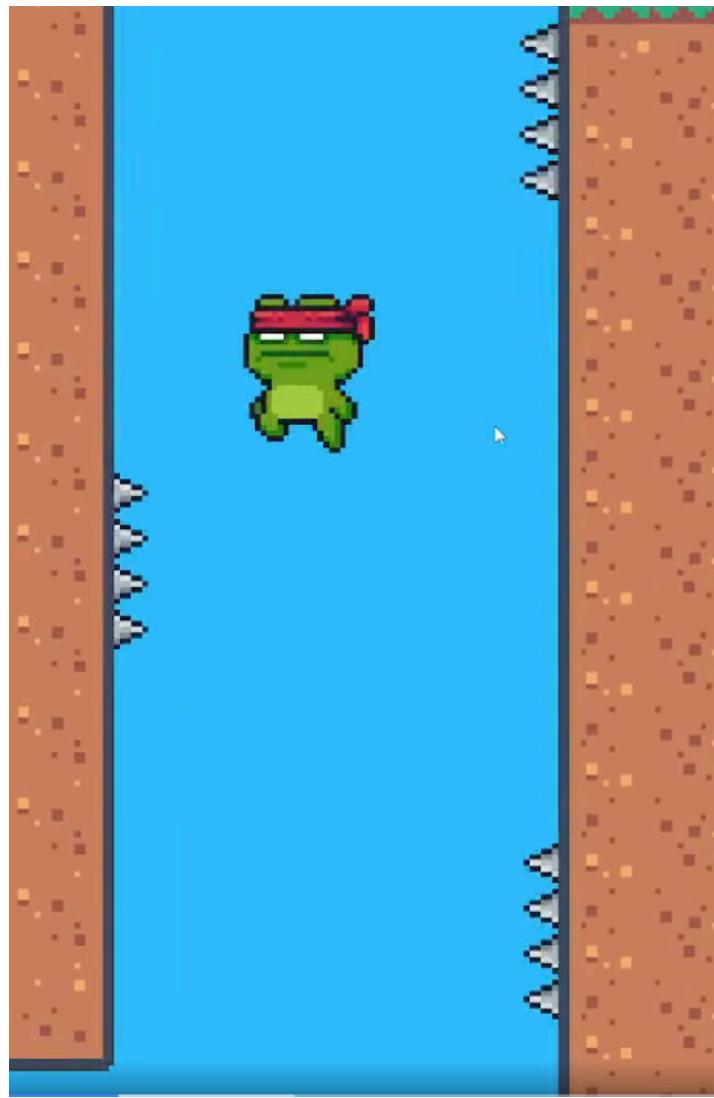
4.3 Level Design(Thiết kế level game)

Level Design là quá trình tạo ra và bố trí các cấp độ hoặc màn chơi trong trò chơi. Mục tiêu của thiết kế cấp độ là tạo ra các môi trường chơi đa dạng, hấp dẫn và thách thức người chơi, đồng thời dẫn dắt họ qua các giai đoạn khác nhau của trò chơi một cách mạch lạc và thú vị. Các yếu tố chính của Level Design bao gồm:

- Sơ Đồ Cấp Độ (Level Layout): Bố trí không gian chơi, đường đi, và vị trí các chướng ngại vật, kẻ thù.
- Độ Khó (Difficulty): Tăng dần độ khó từ cấp độ này sang cấp độ khác để giữ sự hứng thú và thách thức cho người chơi.
- Thủ Thách và Nhiệm Vụ (Challenges and Objectives): Các nhiệm vụ cụ thể mà người chơi phải hoàn thành trong mỗi cấp độ, như vượt qua một chướng ngại vật khó khăn hoặc tiêu diệt kẻ thù mạnh.
- Vật Phẩm và Power-ups: Vị trí đặt các vật phẩm để người chơi thu thập, giúp họ trong hành trình.
- Điểm Đánh Dấu (Checkpoints): Các điểm mà người chơi có thể quay lại nếu thất bại, giúp giảm cảm giác khó chịu khi phải lặp lại toàn bộ cấp độ.
- Giao Tiếp Thị giác (Visual Cues): Sử dụng màu sắc, ánh sáng và các yếu tố hình ảnh khác để hướng dẫn người chơi và tạo điểm nhấn.

thể
những
level
những
“nhảy”

Tất nhiên, đây là yếu tố chiếm nhiều thời gian và công sức để tạo được một level gọi là “có chơi được” và “hoàn chỉnh”. Việc thử thách người chơi qua từng level cũng là một trong cách giữ chân họ, tăng tính chơi lại của một con game. Hình dưới là một cảnh trong một mà em thiết kế, rõ ràng như trong hình sẽ thấy để vượt qua chướng ngại vật này và đến vùng mới thì bắt buộc người chơi sẽ phải kết hợp thành thực kỹ năng “bám tường” và “cóc”.



Một phân cảnh trong game yêu cầu kỹ năng người chơi.

4.4 Một vài thiết kế đã áp dụng trong game

Dưới đây là một vài thiết kế nổi bật mà em đã “tham khảo” và áp dụng trong tựa game này:

- Coyote Time & Variable Jump Height

Coyote Time (thường gọi là "thời gian coyote") là cơ chế cho phép người chơi nhảy ngay cả khi họ đã rời khỏi nền (platform) trong một khoảng thời gian rất ngắn. Tên gọi này xuất phát từ nhân vật hoạt hình Wile E. Coyote, thường bị rơi xuống vực sau khi đứng “lơ lửng” một chút trên không trung. Mục đích của việc này là để tăng sự mượt mà và tính dễ chịu trong điều khiển nhân vật. Một vài tựa game mang tính platform nổi tiếng có sử dụng cơ chế này như: Celeste, Dead Cells, ... Cách hoạt động:

- ❖ Khi nhân vật rời khỏi nền, một bộ đếm thời gian bắt đầu.
- ❖ Trong khoảng thời gian này, nếu người chơi nhấn nút nhảy, trò chơi vẫn cho phép thực hiện hành động nhảy.



Minh họa cơ chế “Coyote-time”.

người chơi
của việc này
nhảy một

Variable Jump Height (nhảy biến đổi độ cao) là cơ chế cho phép người chơi điều chỉnh độ cao của cú nhảy tùy thuộc vào thời gian họ giữ nút nhảy. Thời gian giữ nút nhảy càng lâu thì nhân vật sẽ nhảy càng cao và ngược lại. Mục đích là để tăng tính kiểm soát nhân vật cũng như tạo cảm giác tự nhiên thay vì luôn độ cao cố định.

- Cơ chế Key-Lock cho level design

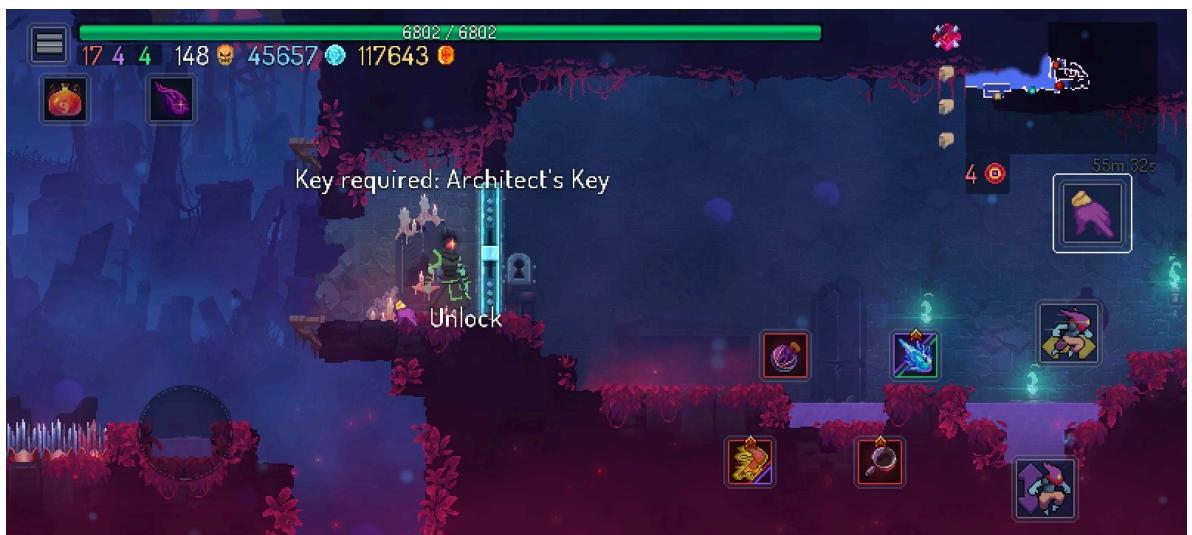
Key-Lock là một cơ chế phổ biến trong thiết kế cấp độ (level design) của trò chơi, đặc biệt là trong các game platformer, phiêu lưu, và giải đố. Đây là một cách để giới hạn sự tiến triển của người chơi và khuyến khích họ khám phá, giải quyết thử thách hoặc đạt được mục tiêu cụ thể trước khi tiến xa hơn, nó bao gồm:

- ❖ “Key” - Chìa khóa: Đại diện cho một điều kiện hoặc đối tượng cần thu thập, giải quyết, hoặc thực hiện (ví dụ: chìa khóa, mã số, công tắc, vật phẩm đặc biệt).

- ❖ “Lock” - Ô khoá: Là rào cản hoặc trạng thái ngăn người chơi tiến xa hơn (ví dụ: cửa bị khóa, chướng ngại vật, cánh cổng, khu vực tạm không thể với tới).

Cơ chế này yêu cầu người chơi phải tìm hoặc đáp ứng điều kiện (key) để vượt qua cản (lock) để đạt được những mục tiêu mới trong trò chơi.

rào



Một số vùng bị giới hạn trong map game Dead Cells.

Nếu
phải
khám phá và đặt

Ví dụ cụ thể về thiết kế “Key - Lock” có trong map game Dead Cells ở phía trên. người chơi không có key - ở đây là “Architect’s Key” thì sẽ không thể đến được những vùng khác trong map - lock. Việc thiết kế này sẽ khuyến khích người chơi hoàn thành những mục tiêu nào đó để không ngừng nâng cấp nhân vật để chân đến những vùng đất mới.

- Hệ thống phần thưởng và mở khóa vật phẩm

Mở khóa là quá trình cho phép người chơi truy cập vào nội dung mới sau khi đáp ứng điều kiện nhất định. Điều này giúp trò chơi duy trì sự hấp dẫn bằng cách đưa ra các mục tiêu dài hạn và ngắn hạn. Dưới đây là thiết kế mở khóa vật phẩm bằng những phần thưởng đặc biệt, từ đó người chơi mới có thể mua vật phẩm đó.



Vật phẩm trong Shop.

- Boss-fight (đánh trùm)

Boss Fight là một phần quan trọng trong thiết kế game, đặc biệt là các thể loại như platformer, RPG, hay hành động. Đây là một thử thách lớn mà người chơi phải vượt qua, thường là ở cuối một màn chơi hoặc chương quan trọng, để tiến xa hơn trong trò chơi. Việc thiết kế một màn boss-fight hay sẽ khiến người chơi họ cảm thấy hứng thú, đáng bồi thời gian để đánh bại nó. Nhưng việc này cũng cần phải đảm bảo mechanic và độ khó của nó sẽ không quá khó hoặc quá dễ đối với người chơi. Phía dưới là hình ảnh một con boss (trùm) trong project này.



Hình ảnh tên boss (trùm).

- **Onboarding**

Onboarding là cách mà một trò chơi giới thiệu cơ chế, luật chơi và kỹ năng cơ bản cho người chơi một cách tự nhiên, không cần thông qua hướng dẫn trực tiếp hoặc dài dòng.

Ví dụ như trong tựa game Super Mario Bros., các level đầu tiên, đặc biệt là Level 1-1, được coi là một ví dụ kinh điển của Implicit Tutorial. Nó sử dụng thiết kế level để dạy người chơi các cơ chế cơ bản, như nhảy, né tránh, và cách tương tác với môi trường, mà không cần bất kỳ đoạn văn bản nào.



Mô tả Onboarding.

Chương 5. THIẾT KẾ GIAO DIỆN

5.1 Danh sách màn hình

STT	Tên màn hình	Ý nghĩa / Ghi chú
1	Màn hình chính	Màn hình chính khi mở game, đi kèm các tùy chọn.
2	Màn hình chọn màn chơi	Hiển thị các level hiện có và những level đã mở khoá.
3	Màn hình chi tiết màn chơi	Hiển thị thông tin chi tiết của màn chơi đó.
4	Màn hình Shop	Hiển thị các vật phẩm có trong Shop đi kèm số vàng, bạc hiện có.
5	Màn hình chi tiết vật phẩm	Hiển thị thông tin chi tiết vật phẩm được chọn.
6	Màn hình âm thanh	Hiển thị âm lượng các loại âm thanh trong game, cho phép người chơi cấu hình và lưu.
7	Màn hình thông tin thêm	Hiển thị thông tin thêm về dự án.
8	Màn hình màn chơi bất kỳ	Hiển thị cảnh một màn chơi bất kỳ.
9	Màn hình kết quả	Hiển thị kết quả chi tiết sau khi kết thúc màn chơi.
10	Màn hình Boss-fight	Hiển thị khung cảnh đánh Boss trong game.

Danh sách màn hình.

5.2 Chi tiết màn hình

5.2.1 Màn hình chính



Hình ảnh màn hình chính

Màn hình hiển thị khi mở game, giới thiệu tên game và hiển thị tất cả các lựa chọn trong game cho người chơi như:

- ❖ Start: Mở màn hình chọn level lên cho người chơi.
- ❖ Shop: Mở màn hình Shop lên cho người chơi.
- ❖ Option: Mở màn hình âm thanh lên cho người chơi.
- ❖ Credits: Mở màn hình thông tin thêm về dự án cho người chơi.
- ❖ Quit: Thoát game.

5.2.2 Màn hình “Chọn màn chơi”



Hình ảnh màn hình “Chọn màn chơi”

Màn hình này hiển thị tất cả màn chơi (những nút bấm nhỏ) có trong game và các màn chơi đã vượt qua/mở khoá (có số). Người chơi có thể chọn bất kỳ màn chơi nào và nó sẽ dẫn đến màn hình “[Chi tiết màn chơi](#)”.

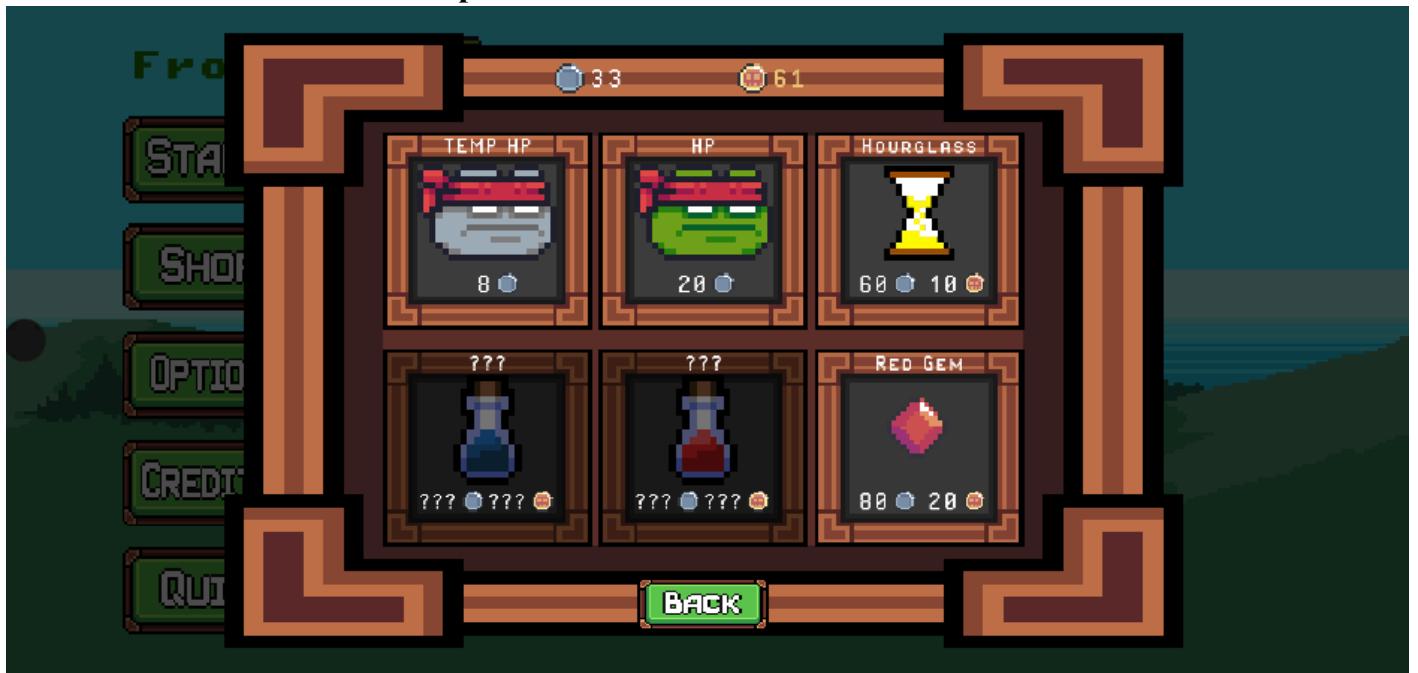
5.2.3 Màn hình “Chi tiết màn chơi”



Hình ảnh màn hình “Chi tiết màn chơi”

Đây là phần hiển thị chi tiết của màn chơi mà người chơi chọn, bao gồm đoạn ngắn giới thiệu nội dung sẽ có trong màn chơi đi kèm thời gian hoàn thành/thời gian cho phép và 2 nút “Play” - vào màn chơi và “Back” - quay lại màn hình chính.

5.2.4 Màn hình Shop



Hình ảnh màn hình Shop

Trong Shop sẽ hiển thị các vật phẩm trong game đi kèm với tên và giá bán của chúng (nếu đã được mở khoá) và lượng vàng bạc của người chơi. Từng vật phẩm trong Shop đều có công dụng đặc biệt, qua từng màn chơi, người chơi có thể dùng phần thưởng nhận được để nâng cấp nhân vật

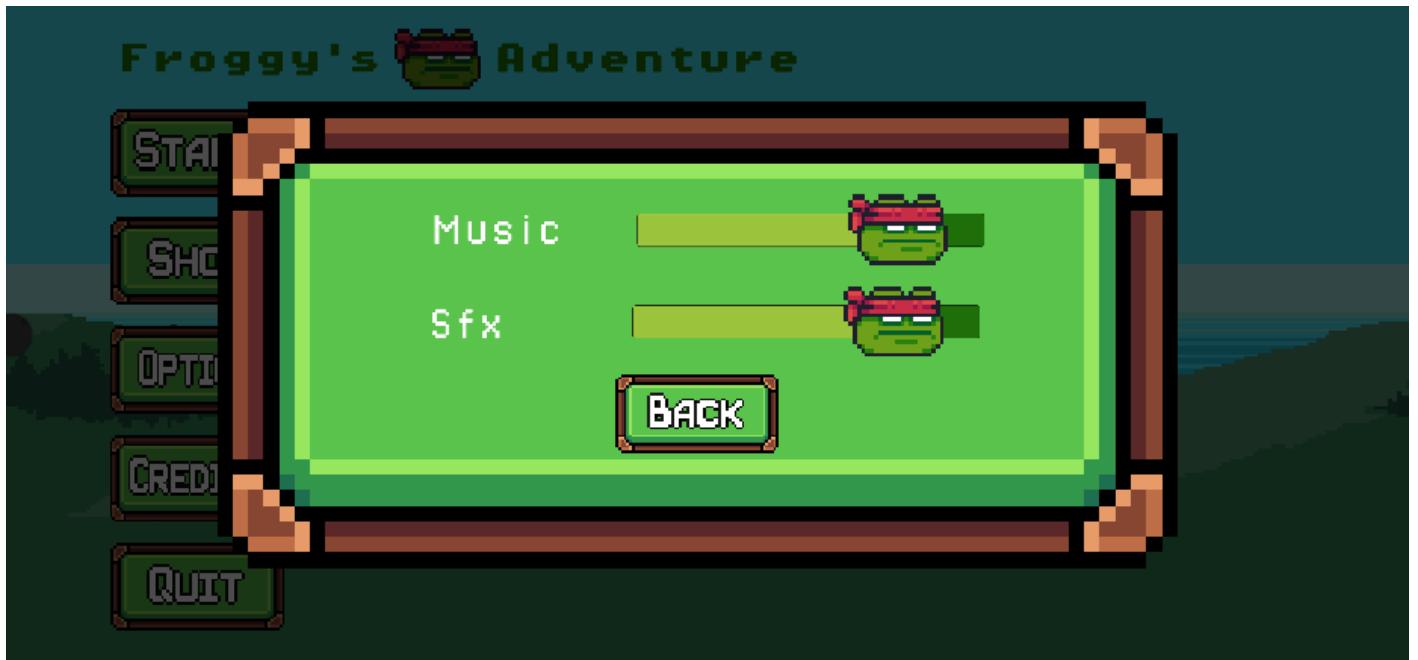
5.2.5 Màn hình “Chi tiết vật phẩm”



Hình ảnh màn hình “Chi tiết vật phẩm”

Phần trên hiển thị thông tin chi tiết của vật phẩm đi kèm với mô tả sau khi mua nó.

5.2.6 Màn hình “Âm thanh”



Hình ảnh màn hình “Âm thanh”

Màn hình này để người chơi có thể tự config(cấu hình) 2 loại âm thanh trong game đó là Sound Effect (SFX) và Nhạc nền (Music).

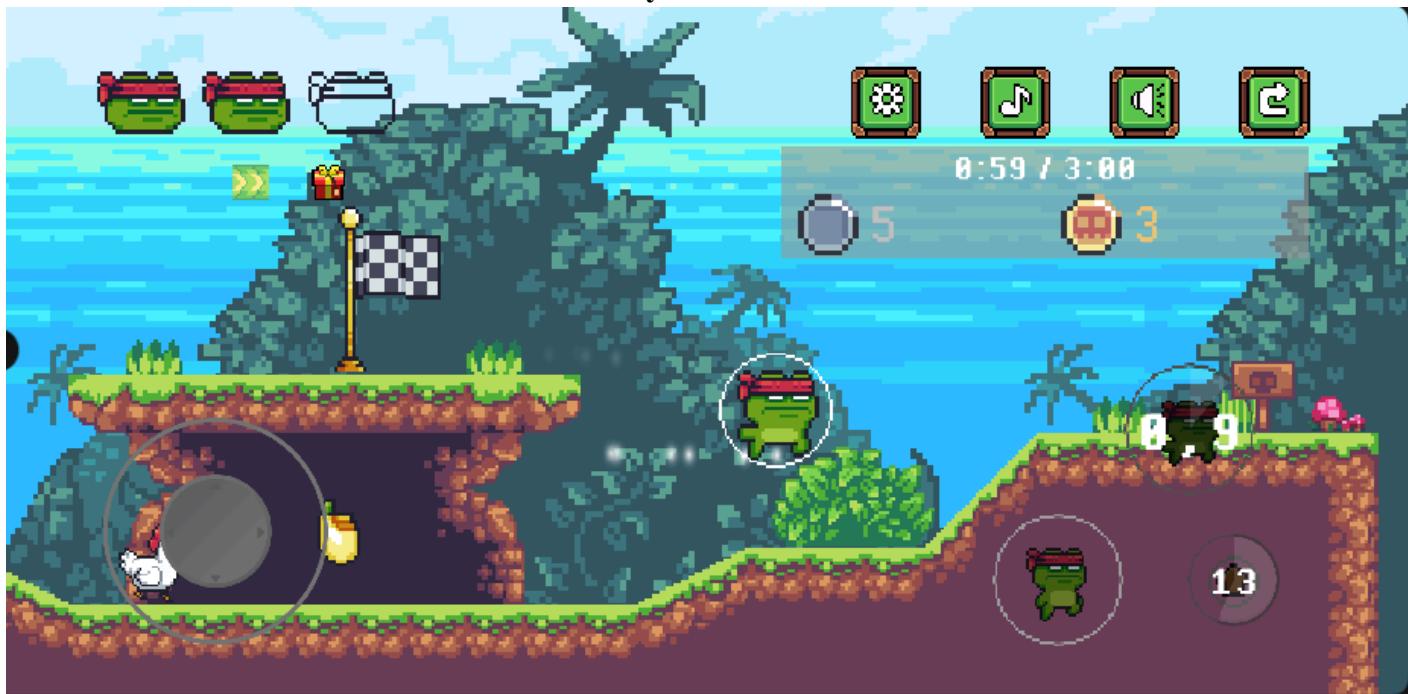
5.2.7 Màn hình “Thông tin thêm”



Hình ảnh màn hình “Thông tin thêm”

Hiển thị thông tin chi tiết của dự án này, bao gồm các nguồn asset, âm thanh, nhạc, ...

5.2.8 Màn hình “Màn chơi bất kỳ”



Hình ảnh màn hình “Màn chơi bất kỳ”

Màn hình preview một màn chơi bất kỳ trong game. Trong gameplay, thanh máu cùng các hiệu ứng buff từ vật phẩm sẽ hiển thị ở góc trái trên màn hình. Góc trái dưới sẽ là thanh Joystick để điều khiển nhân vật. Ở phía góc phải trên sẽ hiển thị các nút nhỏ hỗ trợ và thanh mini HUD hiển thị thời gian còn lại cũng như số vàng, bạc lượm được trong màn chơi. Và cuối cùng ở góc phải dưới sẽ là các nút kỹ năng của người chơi, chỉ những kỹ năng đã được mở khoá bằng vật phẩm thì mới hiển thị trên giao diện.

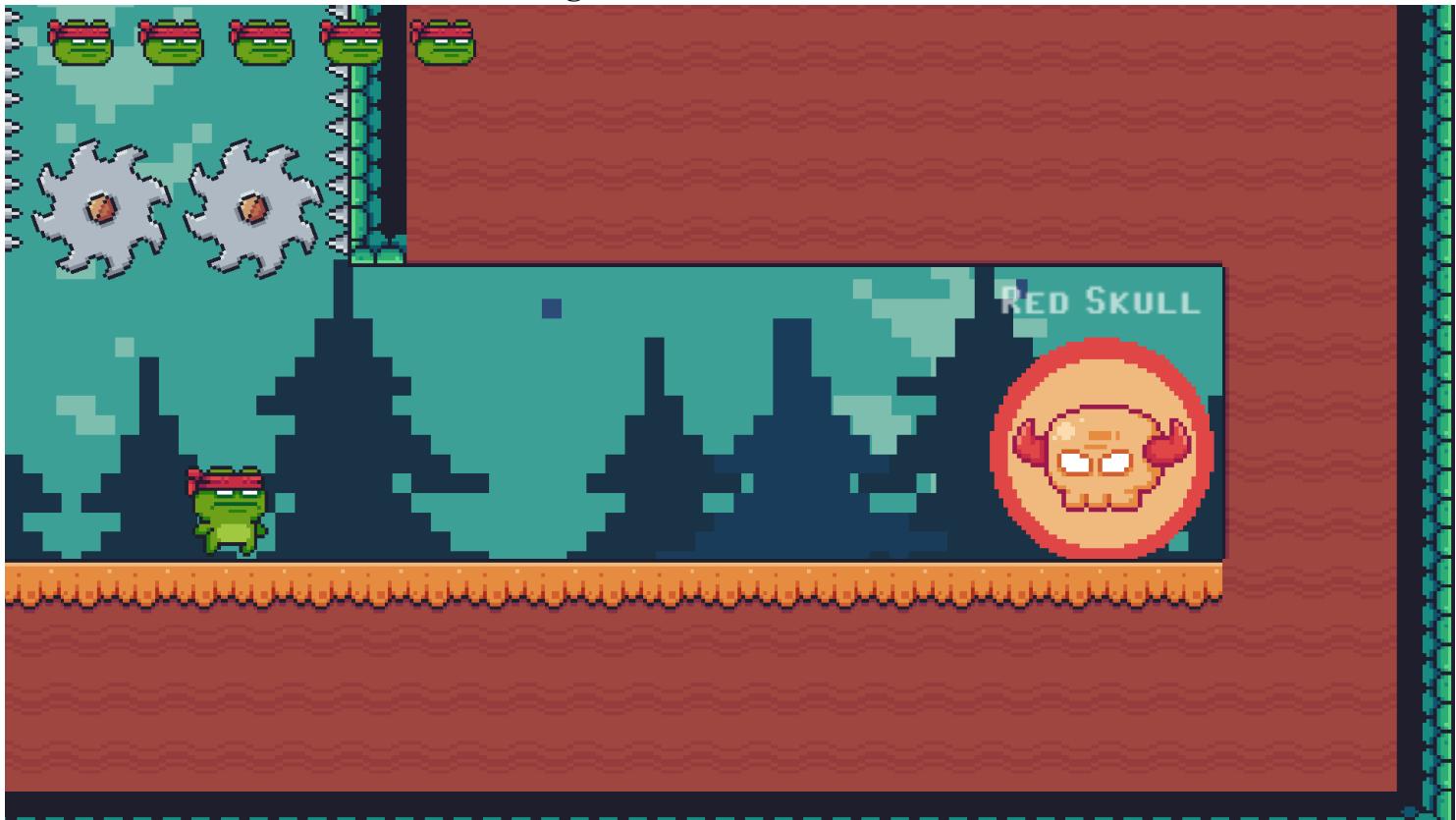
5.2.9 Màn hình “Kết quả”



Hình ảnh màn hình “Kết quả”

Màn hình này cung cấp thông tin như trạng thái màn chơi(thắng, thua), số vàng bạc lượm được, số item trái cây lượm được, thời gian hoàn thành và các mini button hỗ trợ cho người chơi như chuyển màn kế, chơi lại, về màn hình chính, mở Shop, mở chọn level.

5.2.10 Màn hình “Boss fight”



Hình ảnh màn hình Boss-fight

Màn hình này preview một phân cảnh đánh trùm (Boss-fight) trong game. Người chơi sẽ bị giới hạn trong một khu vực và sẽ phải hạ tên trùm bằng cách vận dụng các kỹ năng đã học được xuyên suốt các màn chơi trước.

Chương 6. MỘT SỐ KỸ THUẬT TỐI ƯU GAME 2D TRÊN UNITY ENGINE

6.1 Đặt vấn đề

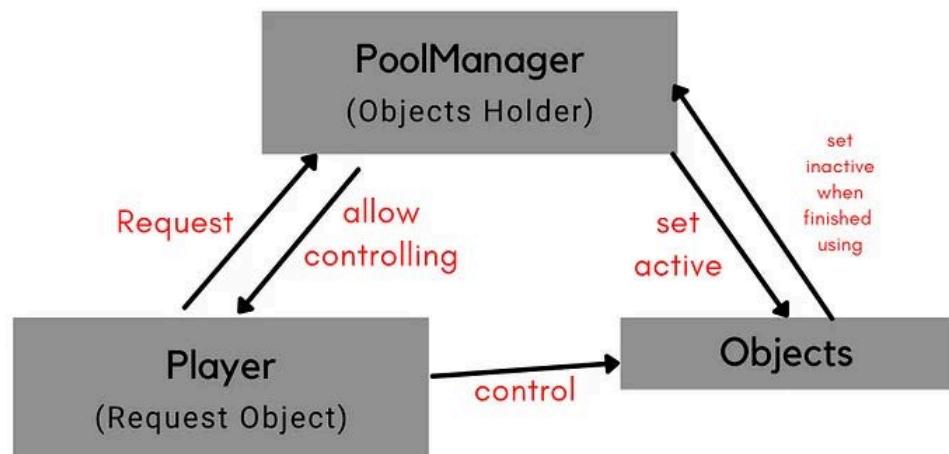
Trong phát triển game trên nền tảng di động, một trong những thách thức lớn là vấn đề tối ưu hóa hiệu suất. Các thiết bị di động có tài nguyên phần cứng giới hạn hơn nhiều so với các hệ máy console hay PC, do đó, việc tối ưu hóa là vô cùng cần thiết để game hoạt động mượt mà, không bị giật, lag tránh làm ảnh hưởng đến trải nghiệm của người chơi. Trong chương này em sẽ trình bày từng kỹ thuật tối ưu một cách chi tiết.

6.2 Các phương pháp tối ưu

6.2.1 Kỹ thuật Object Pooling.

Trong Unity, việc tạo và phá hủy một vật thể bằng các phương thức Instantiate và Destroy liên tục sẽ gây ra vấn đề hiệu năng và có thể gây phân mảnh vùng nhớ (Hiện tượng khi bộ nhớ được chia thành những mảnh rải rác sau một quá trình cấp phát và giải phóng và tổng dung lượng bộ nhớ còn lại có thể đáp ứng yêu cầu nhưng không có một vùng nhớ liên tục đủ lớn để có thể cấp phát). Vì thế, kỹ thuật Object Pooling (tái sử dụng vật thể khi cần đến) được ra đời để giải quyết bài toán này.

Object Pooling Design Pattern

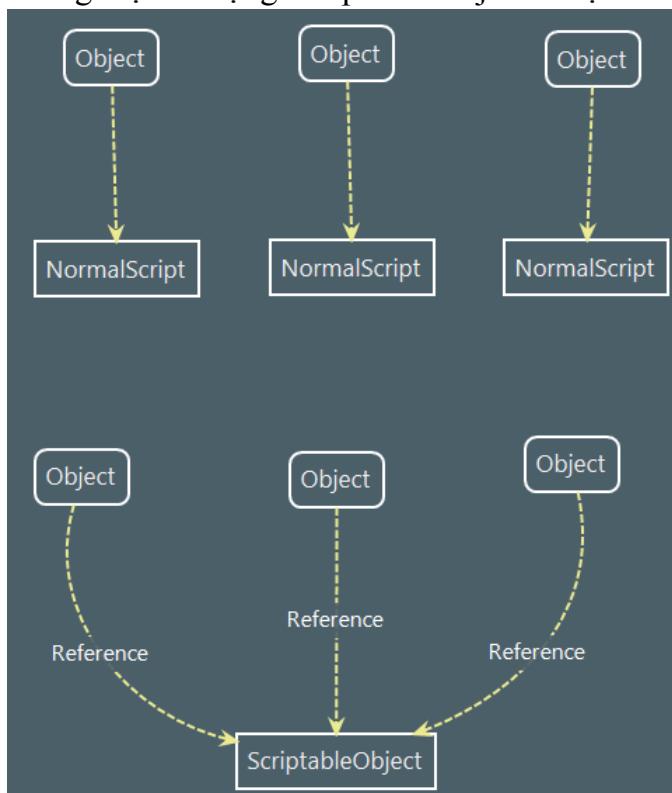


Mô tả pattern Object Pooling.

Về cơ bản, kỹ thuật này sẽ tái sử dụng vật thể đã được tạo ra từ trước. Khi hệ thống yêu cầu một vật thể từ pool (bể chứa) thì PoolManager sẽ kiểm tra có tồn tại một vật thể nào đó từ bể chứa thoả mãn các điều kiện (inactive và là vật thể được yêu cầu) thì sẽ kích hoạt và trả về vật thể đó. Bằng cách tái sử dụng này, chúng ta sẽ tránh được vấn đề liên quan đến việc phân mảnh vùng nhớ do cấp phát và thu hồi liên tục. Chúng ta chỉ cần tạo trước một lượng vật thể vừa đủ dùng và trong quá trình runtime sẽ kích hoạt, ngưng kích hoạt vật thể theo yêu cầu của hệ thống.

6.2.2 Kỹ thuật chia sẻ Data với Scriptable Object.

Trong thế giới game, data (dữ liệu) được chia làm 2 loại chính là Game Data (thông số cấu hình quái, đồ vật,...), chúng bất biến và không thay đổi trong run-time. Ngược lại là Player Data(không cố định, có thể thay đổi trong run-time). Mục này sẽ đề cập đến cách tối ưu Game Data bằng việc sử dụng Scriptable Object - một asset có sẵn trong Unity.

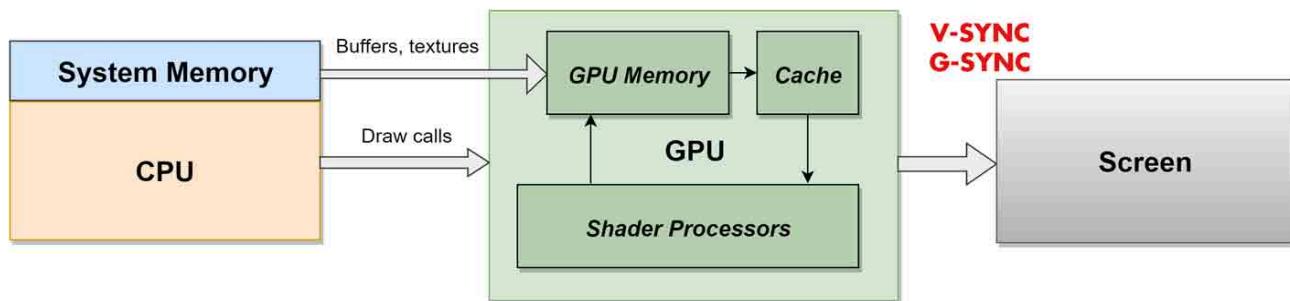


Mô tả cách thức ScriptableObject hoạt động.

Giả dụ một vật thể có 5 trường thuộc tính, trong game có tới 100 vật thể đó thì sau cùng sẽ tồn 5x100 lần bộ nhớ cho 100 vật thể. Bằng cách định nghĩa 1 ScriptableObject có 5 trường thuộc tính đó và giờ đây mỗi vật thể chỉ cần tham chiếu đến cái ScriptableObject đó thì lúc này, dung lượng bộ nhớ cho 100, 1000, 10000, ... vật thể cũng chỉ tồn bộ nhớ cho 5 trường thuộc tính từ ScriptableObject. ScriptableObject hoạt động dựa trên nguyên lý của mẫu thiết kế “Flyweight”. Ngoài ra, ScriptableObject cũng là một asset hữu ích để các Game Designer có thể làm việc chung với Game Developer trên Unity mà không cần phải thông qua code.

6.2.3 Sử dụng Sprite Atlas để giảm draw calls.

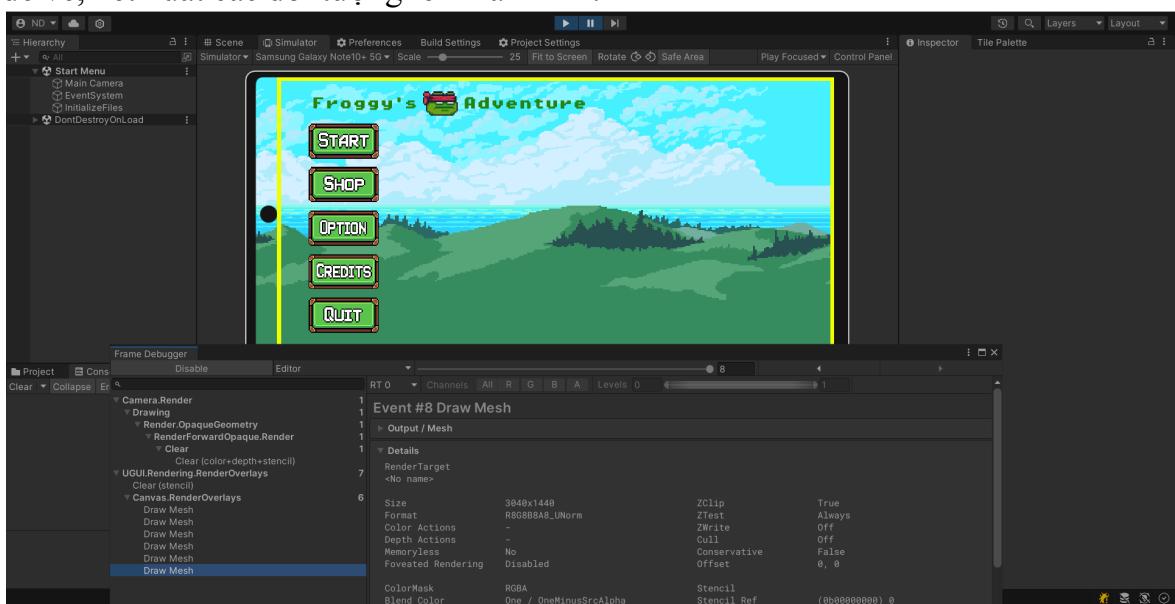
Trong lập trình game, **draw call** là một lệnh mà CPU gửi đến GPU để vẽ, kết xuất một hoặc một nhóm đối tượng lên màn hình. Trong lệnh này sẽ chứa các thông tin về cái thứ mà sẽ được vẽ lên màn hình (kích thước, texture, shader, ...) và rõ ràng, càng nhiều draw call tức là chúng ta sẽ bắt GPU phải làm việc nhiều hơn, dẫn đến hiệu suất của game có thể sẽ bị giảm Unity cũng hỗ trợ việc tối ưu draw call bằng Sprite Atlas - một asset giúp gom nhóm các nhô lè vào 1 texture. Giả dụ bạn có 3 sprite riêng lẻ thì sẽ tốn đến 3 draw call cho 3 sprite việc sử dụng Sprite Atlas để gom các sprite lại sẽ giúp giảm lượng draw call xuống 1.



Minh họa cách 1 draw call hoạt động.

6.2.4 Sử dụng Frame Debugger để giảm draw calls.

Unity Engine có hỗ trợ công cụ “Frame Debugger” dùng chủ yếu để giảm số lượng draw call trong game. Về cách thức hoạt động, chỉ cần chạy game ở Editor và mở tab công cụ này lên, click chuột vào tab này ở thời điểm bất kỳ mà bạn muốn xét, Frame Debugger sẽ dừng game tại chính frame đó và sẽ cho biết ở frame hiện tại có bao nhiêu draw call được thực thi để vẽ, kết xuất các đối tượng lên màn hình.



Cách Frame Debugger hoạt động trong Unity.

Như trên hình, sử dụng công cụ này sẽ cho biết frame hiện tại, có bao nhiêu draw call được thực thi (ở đây là 8) và chúng ta có thể xem lần lượt từng draw call mà các vật thể yêu cầu, từ đó có thể thực hiện gom nhóm các vật thể vào 1 texture hoặc sử dụng Static, Dynamic Batching để tối ưu hóa lượng draw call, tăng hiệu suất cho game.

6.2.5 Hạn chế sử dụng các thành phần sau

- Cache component thay vì gọi phương thức GetComponent<T> nhiều lần nếu có thể. Trong Unity, phương thức GetComponent<T> hoạt động bằng cách nó sẽ quét qua tất cả các thành phần trong Object cho đến khi nó tìm được thành phần “T” đầu tiên.
- Hạn chế sử dụng phương thức Update do bởi phương thức này được thực thi mỗi frame, nếu thực sự không cần thiết thì không cần sử dụng nó.
- Hạn chế sử dụng component Rigidbody2D. Do bởi khi thêm thành phần Rigidbody2D, hệ thống của Unity sẽ thêm vật thể đó vào hệ thống vật lý, gây tăng tính toán và xử lý phức tạp, sau cùng sẽ gây ảnh hưởng đến hiệu năng nếu có quá nhiều Rigidbody2D.
- Giảm va chạm không cần thiết bằng Collision Matrix trong Settings, bằng cách này, Unity sẽ giảm chi phí tính toán cho từng layer được đánh dấu.

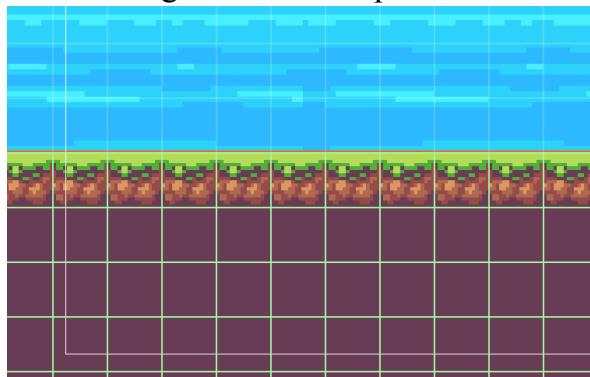
6.2.6 Tối ưu Tilemap Collider kết hợp Collider2D

Trong game platform, Unity cung cấp sẵn hệ thống Tilemap và TilemapCollider để có thể thiết kế map. Tuy vậy nếu việc tối ưu TilemapCollider không kỹ có thể dẫn đến vấn đề hiệu năng, đặc biệt khi bản đồ có kích thước lớn hoặc số lượng collider nhiều.

Sử dụng Tilemap Collider2D (Tick Used By Composite cho nó) kết hợp với Composite Collider 2D để:

- Giảm số lượng Collider riêng lẻ, giúp hệ thống Unity giảm việc tính toán cho từng collider riêng lẻ.
- Tối ưu hóa xử lý va chạm, bề mặt va chạm mượt mà hơn, giảm tài nguyên CPU tiêu thụ để xử lý va chạm.

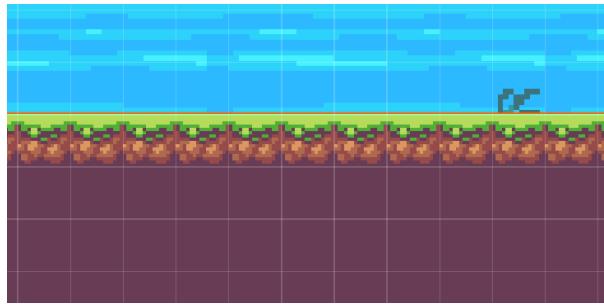
Các hình dưới mô tả việc không tối ưu Tilemap và tối ưu Tilemap.



Tilemap không tối ưu.

Có thể thấy trong hình trên, việc không tối ưu Tilemap dẫn đến bản đồ sẽ hình từng ô vuông nhỏ, từng ô vuông này đều được hệ thống vật lý của Unity xử lý, dẫn đến

thành
lý, dẫn đến



Tilemap đã tối ưu.

những
việc

Còn với cách tối ưu bằng cách trên, giờ đây, bản đồ chỉ còn hiện những collider chỗ cần thiết thay vì từng ô vuông nhỏ như trước, giúp hệ thống vật lý giảm tải cần phải xử lý và nâng cao hiệu suất game.

Chương 7. DESIGN PATTERN SỬ DỤNG

7.1 Thế nào là Design Pattern ?

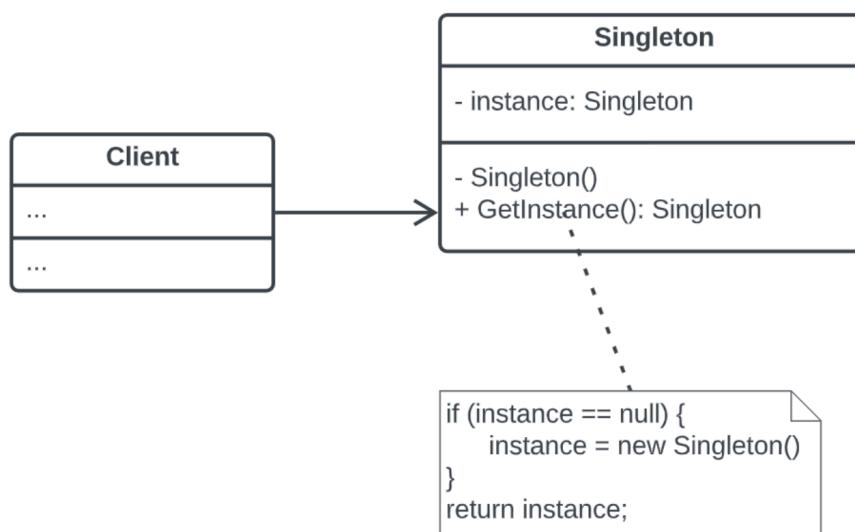
Design Pattern (mẫu thiết kế) là những giải pháp đã được kiểm chứng để giải quyết các vấn đề thường gặp trong thiết kế phần mềm. Chúng cung cấp một cấu trúc hoặc hướng dẫn, giúp lập trình viên dễ dàng tái sử dụng mã, cải thiện khả năng mở rộng và bảo trì của hệ thống. Trong lập trình game, Design Pattern hỗ trợ:

- Tối ưu hóa mã nguồn: Giảm trùng lặp và tạo cấu trúc rõ ràng. Ví dụ, Singleton Pattern thường được dùng để quản lý trạng thái toàn cục như âm thanh, Game Manager.
- Dễ dàng mở rộng và nâng cấp: Khi thêm tính năng mới, sử dụng các pattern như Observer hoặc State giúp giảm ảnh hưởng tới các thành phần khác.
- Tăng tính linh hoạt: Chẳng hạn, Decorator Pattern giúp thêm các hiệu ứng (buff, debuff) mà không cần thay đổi cấu trúc ban đầu của đối tượng.
- Tối ưu: Ví dụ với Flyweight pattern, ta có thể giảm bộ nhớ tiêu thụ đi đáng kể bằng cách chia sẻ dữ liệu dùng chung giữa các đối tượng.

7.2 Các Design Pattern sử dụng trong game và kinh nghiệm

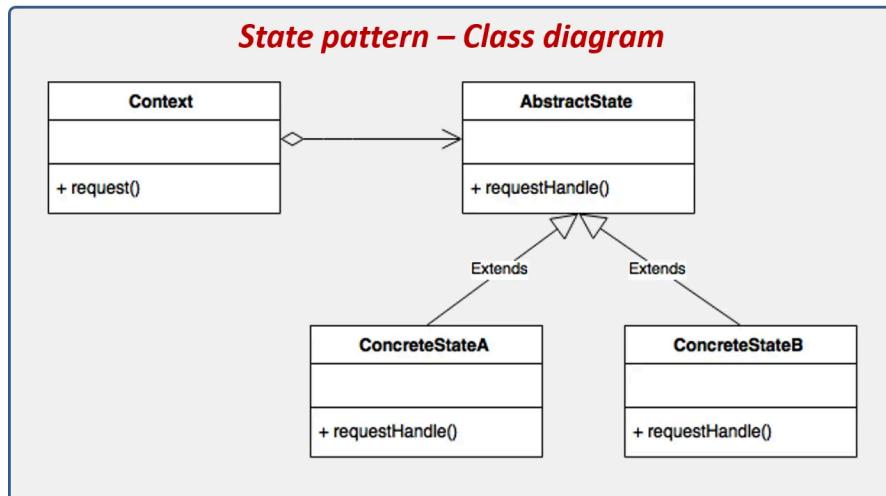
Việc đã có kinh nghiệm trong lập trình game 2D từ môn học trước đã giúp cho em đúc kết được việc sử dụng các Design Pattern và clean code hơn. Dưới đây là các Design Pattern có trong dự án và lý do vì sao nó được sử dụng.

- Singleton: Dùng để quản lý các thành phần toàn cục như Game Manager, SoundManager, UI Manager,... Singleton đảm bảo chỉ có một thể hiện duy nhất của các đối tượng này trong suốt vòng đời của ứng dụng, giúp dễ dàng truy cập và kiểm soát trạng thái toàn cục.



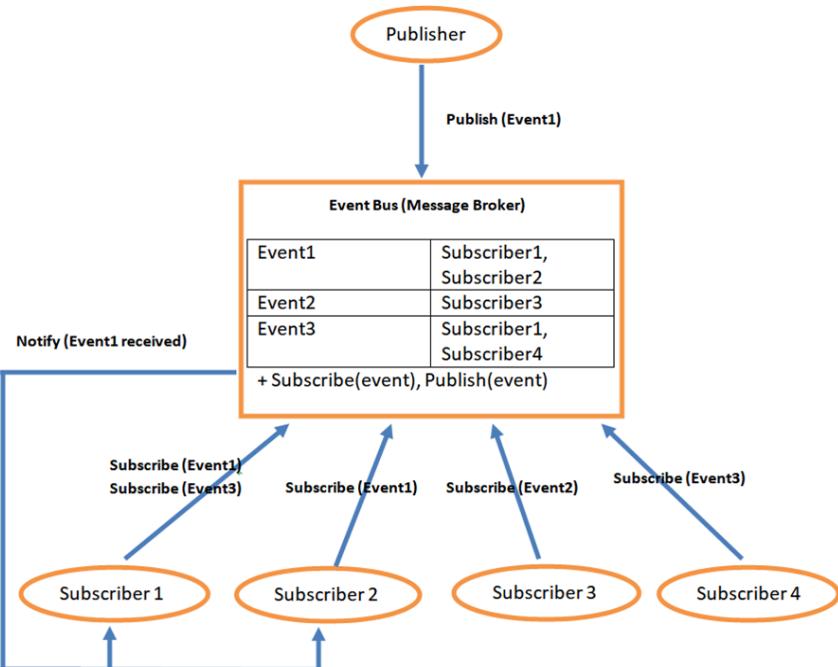
Mô tả Singleton Pattern.

- State: Dùng để quản lý các trạng thái của nhân vật dễ dàng hơn bằng việc chia chúng thành các class để dễ quản lý cũng như bảo trì. Pattern này phát triển từ Finite State Machine (FSM) và cực kỳ hữu ích nếu lượng trạng thái của đối tượng cần quản lý có nhiều hơn 4.



Minh họa State pattern.

- Observer: Sử dụng để quản lý sự kiện và cập nhật UI, ví dụ khi người chơi thu thập tiền xu, thay đổi máu, hoặc hoàn thành mục tiêu. Observer Pattern giúp giảm sự phụ thuộc giữa các đối tượng, đảm bảo các đối tượng liên quan luôn được cập nhật khi có sự thay đổi xảy ra.



Sơ đồ Observer Pattern.

- Flyweight: Tương tự như đã giải thích ở mục “[kỹ thuật chia sẻ Data](#)”, ScriptableObject trong Unity hoạt động tương tự như Flyweight Pattern. Bằng cách sử dụng nó, ta sẽ tiết kiệm được rất nhiều tài nguyên hệ thống.
- Object Pooling: Cũng tương tự như đã giải thích ở mục “[Object Pooling](#)”, sử dụng pattern này giúp ta tiết kiệm được bộ nhớ, tái sử dụng các object trong game và hạn chế việc rò rỉ bộ nhớ đi nhiều lần.

Chương 8. KẾT LUẬN

8.1 Ưu điểm

Trong quá trình phát triển tựa game này, em đã học hỏi được rất nhiều điều trong việc phát triển game nói chung và game di động nói riêng. Dưới đây là một vài ưu điểm của dự án này:

- Hiệu năng được tối ưu: Việc tối ưu hiệu năng đặc biệt là trên thiết bị di động là một trong những yêu cầu tiên quyết trong tựa game này. Bằng cách áp dụng [các kỹ thuật tối ưu](#) đã đề cập ở trên, game hoạt động ổn định, mượt mà và không xuất hiện các lỗi nào gây crash game, đảm bảo mọi thiết bị đều có thể trải nghiệm game một cách ổn định.
- Đảm bảo nội dung game: Việc thiết kế mỗi map đều đảm bảo lượng thời gian chơi một map không quá 4 phút và các map đầu được thiết kế để cho người chơi ở mọi lứa tuổi đều có thể chơi và trải nghiệm game mà không gặp khó khăn. Từng map sau sẽ tăng dần độ khó và yêu cầu người chơi sẽ phải kết hợp các kỹ năng điều khiển nhân vật cũng như mua các vật phẩm hợp lý để qua màn.
- Code-base được thiết kế linh hoạt: Bằng cách áp dụng các [Design Pattern](#), code-base của game cũng đã được thiết kế sao cho dễ mở rộng, dễ hiểu và dễ debug, bảo trì. Mỗi lần thêm một tính năng mới cũng đều không gặp quá nhiều khó khăn trong việc xây dựng mã, đảm bảo không phải sửa, cấu trúc lại mã nguồn nhiều.

8.2 Nhược điểm

Tuy vậy, tựa game vẫn tồn tại những nhược điểm sau do bởi sự thiếu kinh nghiệm trong lĩnh vực phát triển game của bản thân em như là:

- Game design chưa thực sự được tốt: Ở một vài chỗ trong bản đồ, vẫn sẽ có những hụt sạn trong việc thiết kế, điều này là không thể tránh khỏi bởi sự thiếu thốn kinh nghiệm lẫn trải nghiệm trong thiết kế game của bản thân.
- Mã nguồn chưa thực sự “clean” ở một vài chỗ: Trong mã nguồn dự án vẫn tồn tại những phần đoạn mã chưa “clean” khiến cho việc mở rộng hay bảo trì đôi lúc sẽ gặp khó khăn.

8.3 Hướng phát triển

Dự án Phát triển và tối ưu game 2D platformer trên thiết bị di động đã thành công hoàn thành các mục tiêu ban đầu, bao gồm:

- Xây dựng gameplay đơn giản, có thể đáp ứng được một phần tiêu chí “easy to play - hard to master” (dễ chơi nhưng khó thành thạo) và tối ưu hóa về mặt hiệu năng.
- Triển khai thành công game trên nhiều thiết bị, đảm bảo trải nghiệm mượt mà cho người chơi.
- Đánh dấu bước tiến quan trọng trong việc áp dụng các Design Pattern và clean code trong phát triển game của bản thân.

Về hướng phát triển, trước mắt game có thể:

- Bổ sung thêm hệ thống map phù hợp với chủ đề hơn và nhiều map hơn.
- Phát triển thêm nhiều Boss (quái khổng lồ) hơn.
- Bổ sung thêm các trang bị, vật phẩm trong Shop.
- Nghiên cứu thêm các game-mechanic (cơ chế game) hay hơn vào trong trò chơi.
- Phát triển hệ thống Inventory, cho phép người chơi có thể thỏa sức mua, bán, xài vật phẩm một cách chủ động thay vì tự động như hiện tại.

triển

vực

thành

Chương 9. TÀI LIỆU THAM KHẢO VÀ MÃ NGUỒN

1. Tham Khảo

1.1. Lý thuyết về Unity 2D.

<https://docs.unity3d.com/Manual/Unity2D.html>

1.2. Lý thuyết về các pattern trong lập trình game.

<https://gameprogrammingpatterns.com/>

1.3. Cộng đồng người dùng Unity Việt Nam.

<https://www.facebook.com/groups/UnityVietnam>

1.4. Chia sẻ về kỹ thuật trong lập trình game bởi người Việt

<https://thoxaylamcoder.wordpress.com/>

1.5. Lý thuyết State Pattern trong lập trình game

<https://www.youtube.com/watch?v=Vt8aZDPzRjI&t=332s>

1.6. Diễn đàn Unity

<https://discussions.unity.com/c/answers/5>

1.7. Diễn đàn Game Developer

<https://gamedev.stackexchange.com/>

1.8. Framework Game 2D thầy Dũng Đinh (nền tảng lập trình game)

<https://github.com/dungdna2000/gamedev-intro-tutorials>

2. Mã Nguồn

Github: <https://github.com/1609Dzuaa/Froggy-Adventure>