**TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

------------

**ĐỒ ÁN TỐT NGHIỆP KỸ SƯ**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**MÃ NGÀNH: 7480201**

**KHÓA: 2020 – 2025**

***ĐỀ TÀI:***

**PHÁT TRIỂN TRÒ CHƠI UNITY2D PLATFORMER**

|  |  |
| --- | --- |
| **SINH VIÊN THỰC HIỆN:** | **LÊ BẢO NGUYÊN** |
| **LỚP:** | **20CN1** |

**HÀ NỘI 2-2025**

**TRƯỜNG ĐẠI HỌC KIẾN TRÚC HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

------------

**ĐỒ ÁN TỐT NGHIỆP KỸ SƯ**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**MÃ NGÀNH: 7480201**

**KHÓA: 2020 – 2025**

***ĐỀ TÀI:***

**PHÁT TRIỂN TRÒ CHƠI UNITY2D PLATFORMER**

|  |  |  |
| --- | --- | --- |
| **GIÁO VIÊN HƯỚNG DẪN** | **:** | **TS. TRẦN TRUNG** |
| **SINH VIÊN THỰC HIỆN** | **:** | **LÊ BẢO NGUYÊN** |
| **LỚP** | **:** | **20CN1** |

**HÀ NỘI 2-2025**

# **LỜI CẢM ƠN**

Lời đầu tiên, em xin gửi lời cảm ơn sâu sắc nhất từ tận đáy lòng đến tất cả những người đã đồng hành, hỗ trợ và tạo điều kiện cho em trong suốt hành trình học tập và thực hiện đồ án tốt nghiệp kỹ sư với đề tài: **Phát triển trò chơi Unity2D Platformer**.

Em xin bày tỏ lòng biết ơn chân thành đến Ban lãnh đạo **Trường Đại học Kiến Trúc Hà Nội**. Nhờ tầm nhìn, sự quan tâm và tạo dựng môi trường giáo dục lý tưởng của Quý Nhà Trường, em và các bạn sinh viên đã có cơ hội được học tập và phát triển trong một không gian tràn đầy tri thức và cảm hứng.

Tiếp theo, xin gửi lời cảm ơn trân trọng đến **Khoa Công Nghệ Thông Tin**, nơi đã trang bị cho em không chỉ kiến thức chuyên môn mà còn là những bài học quý giá về tư duy và cách tiếp cận thực tế. Những nỗ lực của các Thầy Cô trong Khoa đã luôn là động lực lớn giúp em vượt qua những thử thách trong hành trình học tập.

Đặc biệt, em xin gửi lời tri ân sâu sắc đến thầy **TS. Trần Trung**, giảng viên hướng dẫn đã đồng hành cùng em trong từng bước thực hiện đồ án. Những hướng dẫn tận tình, lời khuyên sâu sắc của thầy không chỉ giúp em hoàn thành tốt đề tài mà còn mở ra cho em nhiều góc nhìn mới mẻ trong lĩnh vực Công nghệ Thông tin.

Bên cạnh đó, em xin gửi lời cảm ơn đặc biệt đến cô **ThS.** **Đinh Kim Phượng**, cố vấn học tập lớp **20CN1**, người đã luôn ân cần theo sát, động viên và định hướng cho chúng em trong cả học tập lẫn cuộc sống sinh viên đầy ý nghĩa.

Là một sinh viên còn nhiều hạn chế về kinh nghiệm, em hiểu rằng đồ án này vẫn có thể tồn tại những thiếu sót. Em rất mong nhận được sự đóng góp, chỉ dẫn từ các Thầy Cô và mọi người để em có cơ hội hoàn thiện bản thân, không ngừng học hỏi và phát triển trong tương lai.

Em xin chân thành cảm ơn!

**LỜI CAM ĐOAN**

Em xin cam đoan rằng đây là công trình nghiên cứu của riêng em và được thực hiện dưới sự hướng dẫn khoa học tận tình của **TS. Trần Trung**. Các nội dung nghiên cứu, kết quả trong đề tài tốt nghiệp **“Phát triển trò chơi Unity2D Platformer”** là hoàn toàn trung thực và chưa từng được công bố dưới bất kỳ hình thức nào trước đây. Những số liệu, dữ kiện được sử dụng trong các bảng biểu để phục vụ cho việc phân tích, nhận xét và đánh giá đều được trích dẫn từ các nguồn đáng tin cậy, đồng thời được ghi rõ trong phần tài liệu tham khảo.

Trong quá trình thực hiện, em cũng sử dụng một số nhận xét, đánh giá cũng như số liệu từ các tác giả khác, cơ quan hoặc tổ chức liên quan. Tất cả các nguồn thông tin này đều đã được trích dẫn và chú thích đầy đủ, rõ ràng.

Nếu có bất kỳ sai sót hoặc hành vi gian lận nào xảy ra, em xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của mình. **Trường Đại học Kiến Trúc Hà Nội** không liên quan đến bất kỳ vi phạm tác quyền hay bản quyền nào phát sinh từ công trình này (nếu có).

|  |
| --- |
| *Hà Nội, ngày ...... tháng ...... năm 202…* |
| Sinh viên |
|  |
| Lê Bảo Nguyên |

# **LỜI NÓI ĐẦU**

**Lời nói đầu**

Với sự phát triển không ngừng của công nghệ, việc sáng tạo và triển khai các ứng dụng giải trí đã trở thành một phần quan trọng trong cuộc sống hiện đại. Dự án "Fun\_Trap" được xây dựng nhằm mang lại một không gian trải nghiệm sáng tạo, nơi người dùng không chỉ được giải trí mà còn được khám phá những giá trị mới từ công nghệ.

**Đặt vấn đề**

Trong bối cảnh công nghệ số ngày càng phát triển, các ứng dụng và trò chơi số đã trở thành một phần không thể thiếu trong đời sống hàng ngày. Dự án "Fun\_Trap" ra đời nhằm mục đích mang đến một trải nghiệm sáng tạo và giải trí cho người dùng, đồng thời khuyến khích việc học hỏi và ứng dụng công nghệ hiện đại trong phát triển trò chơi.

**Mục đích dự án**

Dự án "Fun\_Trap" được phát triển với các mục đích sau:

* Xây dựng một trò chơi độc đáo, hấp dẫn và dễ tiếp cận.
* Khai thác các tính năng tiên tiến của Unity để tạo ra sản phẩm chất lượng cao.
* Tạo môi trường học tập và thực hành cho các nhà phát triển phần mềm, đặc biệt là trong lĩnh vực phát triển trò chơi.

**Nội dung dự án**

Dự án tập trung vào các nội dung chính sau:

* Phát triển các cảnh trò chơi và giao diện trực quan, sinh động.
* Tích hợp các tính năng vật lý và hoạt họa để nâng cao trải nghiệm người chơi.
* Tối ưu hóa mã nguồn và tổ chức thư mục dự án để đảm bảo khả năng mở rộng và bảo trì.

**Ý nghĩa dự án**

Dự án "Fun\_Trap" không chỉ là một sản phẩm giải trí mà còn là một minh chứng cho sự sáng tạo và ứng dụng công nghệ hiện đại. Dự án có tiềm năng truyền cảm hứng cho các nhà phát triển trẻ, đồng thời góp phần nâng cao giá trị của ngành công nghiệp trò chơi tại địa phương.

**Công nghệ Sử dụng**

Dựa trên cấu trúc thư mục, dự án có vẻ được phát triển bằng Unity, với các công nghệ và công cụ sau:

* **Unity**: Công cụ phát triển trò chơi và ứng dụng đa nền tảng.
* **TextMesh Pro**: Plugin cho việc xử lý văn bản.
* **Visual Scripting**: Công cụ kéo-thả để tạo logic trò chơi mà không cần mã hóa.

**Hướng Dẫn Cài Đặt và Chạy Dự Án**

**Yêu cầu hệ thống**:

* + Unity Editor (phiên bản khuyến nghị: [phiên bản Unity được sử dụng])
  + Hệ điều hành: Windows, macOS, hoặc Linux.

**Cài đặt**:

* + Tải mã nguồn từ [GitHub](https://github.com/nguyenlam1962c5/Fun_Trap).
  + Giải nén nếu cần thiết.
  + Mở thư mục dự án trong Unity Editor.

**Chạy dự án**:

* + Trong Unity Editor, mở một scene chính từ thư mục Scenes.
  + Nhấn nút "Play" để kiểm tra.

**Kết luận**

Dự án "Fun\_Trap" là một nền tảng tiềm năng cho [mô tả ngắn về mục tiêu hoặc ứng dụng thực tế của dự án]. Việc tổ chức rõ ràng các thư mục và sử dụng công cụ mạnh mẽ như Unity mang lại khả năng mở rộng và phát triển lâu dài.

# **MỤC LỤC**

[LỜI CẢM ƠN i](#_Toc190502548)

[LỜI CAM ĐOAN ii](#_Toc190502549)

[LỜI NÓI ĐẦU iii](#_Toc190502550)

[MỤC LỤC v](#_Toc190502551)

[DANH MỤC BẢNG viii](#_Toc190502552)

[DANH MỤC HÌNH ix](#_Toc190502553)

[TRANG KÝ HIỆU VÀ CHỮ VIẾT TẮT, THUẬT NGỮ x](#_Toc190502554)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 1](#_Toc190502555)

[1.1. Giới thiệu về trò chơi 2D Platformer 1](#_Toc190502556)

[1.1.1. Định nghĩa và đặc điểm của trò chơi platform 1](#_Toc190502557)

[1.1.2. Vai trò và ý nghĩa của thể loại trò chơi này 1](#_Toc190502558)

[1.1.3. Ví dụ nổi bật về trò chơi platformer trên thị trường 2](#_Toc190502559)

[1.2. Công cụ phát triển trò chơi Unity 2D 2](#_Toc190502560)

[1.2.1. Tổng quan về Unity 2](#_Toc190502561)

[1.2.2. Các tính năng của Unity 2D hỗ trợ phát triển platformer 2](#_Toc190502562)

[1.2.3. So sánh Unity với các công cụ phát triển khác 4](#_Toc190502563)

[1.3. Các khái niệm cốt lõi trong phát triển trò chơi 5](#_Toc190502564)

[1.3.1. Vật lý trong game (Game physics) 5](#_Toc190502565)

[1.3.2. Lập trình hướng sự kiện (event-driven programming) 6](#_Toc190502566)

[1.3.3. Hệ thống hoạt ảnh (animation system) 7](#_Toc190502567)

[1.4. Tổng quan về lập trình C# trong Unity 8](#_Toc190502568)

[1.4.1. C# cơ bản và ứng dụng trong Unity 8](#_Toc190502569)

[1.4.2. Script và cách hoạt động trong Unity 9](#_Toc190502570)

[1.4.3. Các thư viện hỗ trợ hữu ích 11](#_Toc190502571)

[CHƯƠNG 2: TỔNG QUAN ĐỀ TÀI 13](#_Toc190502572)

[2.1. Mục tiêu và phạm vi của đề tài 13](#_Toc190502573)

[2.1.1. Mục tiêu chính của trò chơi 13](#_Toc190502574)

[2.1.2. Phạm vi phát triển (thời gian, công cụ, nguồn lực) 13](#_Toc190502575)

[2.2. Phân tích các trò chơi platformer hiện có 16](#_Toc190502576)

[2.2.1. Phân tích các trò chơi điển hình (ví dụ: Mario, Hollow Knight) 16](#_Toc190502577)

[2.2.2. Điểm mạnh và yếu của các trò chơi này 17](#_Toc190502578)

[2.2.3. Bài học kinh nghiệm cho đề tài 17](#_Toc190502579)

[2.3. Ý tưởng phát triển trò chơi 18](#_Toc190502580)

[2.3.1. Cốt truyện và nội dung 18](#_Toc190502581)

[2.3.2. Tính năng chính và cơ chế gameplay (game mechanics) 20](#_Toc190502582)

[2.3.3. Điểm khác biệt và sáng tạo của trò chơi 21](#_Toc190502583)

[2.4. Các công cụ và công nghệ áp dụng 21](#_Toc190502584)

[2.4.1. Unity và plugin cần thiết 21](#_Toc190502585)

[2.4.2. Công cụ thiết kế đồ họa và âm thanh 23](#_Toc190502586)

[CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG 24](#_Toc190502587)

[3.1. Thiết kế và phát triển 24](#_Toc190502588)

[3.1.1. Quy trình thiết kế (game design document, wireframe) 24](#_Toc190502589)

[3.1.2. Phân chia các giai đoạn phát triển 25](#_Toc190502590)

[3.1.3. Kiểm thử và đánh giá các phiên bản 26](#_Toc190502591)

[3.2. Cấu trúc và chức năng của trò chơi 27](#_Toc190502592)

[3.2.1. Cấu trúc tổng quan (menu chính, màn chơi, giao diện kết thúc) 27](#_Toc190502593)

[3.2.2. Các chức năng chính (điều khiển, chiến đấu, vượt chướng ngại vật) 29](#_Toc190502594)

[3.3. Tích hợp và tối ưu hóa 32](#_Toc190502595)

[3.3.1. Tích hợp âm thanh và đồ họa 32](#_Toc190502596)

[3.3.2. Tối ưu hóa mã nguồn 32](#_Toc190502597)

[3.3.3. Đóng gói và triển khai (WebGL, Windows và Android) 32](#_Toc190502598)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 33](#_Toc190502599)

[4.1. Đánh giá kết quả 33](#_Toc190502600)

[4.1.1. Thành tựu đạt được 33](#_Toc190502601)

[4.1.2. Hạn chế và khó khăn 33](#_Toc190502602)

[4.2. Hướng phát triển trong tương lai 33](#_Toc190502603)

[4.2.1. Nâng cấp tính năng và đồ họa 33](#_Toc190502604)

[4.2.2. Mở rộng nội dung (thêm màn chơi, thêm chế độ chơi) 33](#_Toc190502605)

[4.2.3. Ứng dụng AI hoặc tính năng đa người chơi 33](#_Toc190502606)

[TÀI LIỆU THAM KHẢO 34](#_Toc190502607)

[PHỤ LỤC 36](#_Toc190502608)

# **DANH MỤC BẢNG**

# **DANH MỤC HÌNH**

# **TRANG KÝ HIỆU VÀ CHỮ VIẾT TẮT, THUẬT NGỮ**

# **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

## **Giới thiệu về trò chơi 2D Platformer**

### ***Định nghĩa và đặc điểm của trò chơi platform***

Game Platform còn gọi là game đi cảnh, là một nhánh thuộc thể loại game hành động. Điểm nổi bật của thể loại này đó là người chơi sẽ điều khiển nhân vật vượt qua các thử thách như địa hình hiểm trở, chướng ngại vật, hoặc kẻ thù. Các thao tác như nhảy, leo trèo, hoặc đu dây là yếu tố cốt lõi, yêu cầu kỹ năng điều khiển chính xác từ người chơi. Qua mỗi cấp độ, độ khó sẽ tăng dần với thiết kế môi trường đa dạng hơn, sự xuất hiện của những thử thách mới, tạo nên sức hấp dẫn lâu dài.

Khi còn ở thời kỳ đỉnh cao, game đi cảnh chiếm thị phần lớn trên các hệ máy console, từng là biểu tượng cho thế hệ game thủ thời kỳ đầu. Tuy nhiên, với sự phát triển mạnh mẽ của thể loại game bắn súng và các thể loại mới, mức độ phổ biến của game đi cảnh đã giame giần theo thời gian. Dù vậy, nhờ có cộng đồng người chơi trung thành, dòng game này vẫn giữ được chỗ đứng vững chắc và tiếp tục được duy trì, phát triển trong ngành công nghiệp game hiện đại.

Điểm cơ bản nhất của game platform là lối chơi điều khiển nhân vật trong môi trường với địa hình khó khăn. Đồng thời, người chơi còn phải vượt qua rất nhiều đối thủ trong game và tránh né tất cả các chướng ngại vật mới có thể chiến thắng. Lối chơi theo phong cách năng động, thách thức khả năng phản xạ và sự khéo léo của người chơi.

Các chuyển động phổ biến trong game đi cảnh thường là đi bộ, chạy, nhảy, leo trèo và tấn công. Mỗi game sẽ có những sáng tạo khác nhau nhưng nhìn chung tất cả game đi cảnh đều có môi trường game yêu cầu các nhân vật phải nhảy qua. Quỹ đạo của bước nhảy sẽ không bị thay đổi dù bất kỳ tình huống nào nhưng người chơi cần lưu ý đến thời gian trong game và những yếu tố có thể gây mất mạng khi chơi game.

Hầu hết các game platform đều có mức độ khó tăng dần theo cấp độ. Ở những màn chơi khó nhất sẽ gặp trùm game và người chơi phải vận dụng và kết hợp tất cả các kỹ năng đã học được để có thể tiêu diệt được tên trùm. [1]

### ***Vai trò và ý nghĩa của thể loại trò chơi này***

Thể loại Game Platform đóng vai trò tối quan trọng trong quá trình hình thành và phát triển ngành công nghiệp game. Với lối chơi phụ thuộc phần lớn vào kỹ năng cá nhân của người chơi, Game Platform giúp phát triển kỹ năng của người chơi, rèn luyện phản xạ, lối tư duy logic và khả năng điều khiển chính xác nhằm giải quyết thử thách khó khăn.

Cùng với những đóng góp to lớn trong ngành công nghiệp game, đặt nền móng cho các thiết kế môi trường, cơ chế mới và đồng thời là cảm hứng cho những thể loại game bắn súng, hành động sau này. Từ đó, Game Platform đã gắn liền với tuổi thơ của nhiều thế hệ, trở thành biểu tượng trong lịch sử phát triển game, từ đó khai sinh ra các tựa game huyền thoại một thời như Mario, Contra, Sonic,…

### ***Ví dụ nổi bật về trò chơi platformer trên thị trường***

Một số tựa game platformer kinh điển đã góp phần định hình thể loại này bao gồm Super Mario Bros, đại diện tiêu biểu với thiết kế màn chơi sáng tạo và cách chơi hấp dẫn. Sonic the Hedgehog của Sega lại nổi bật với tốc độ nhanh và khả năng điều khiển mượt mà, mang lại cảm giác phấn khích cho người chơi.

Trong thời kỳ hiện đại, Celeste là một ví dụ thành công, không chỉ nhờ lối chơi thử thách mà còn bởi cốt truyện sâu sắc và thông điệp về sức mạnh vượt qua nghịch cảnh. Hollow Knight cũng là một đại diện đáng chú ý, kết hợp yếu tố platforming với phiêu lưu và thế giới mở, tạo nên một trải nghiệm phong phú và đầy cảm xúc. Những tựa game này đều mang tính biểu tượng, thể hiện sự phát triển và sáng tạo không ngừng của thể loại game đi cảnh.

## **Công cụ phát triển trò chơi Unity 2D**

### ***Tổng quan về Unity***

Unity là một công cụ phát triển trò chơi mạnh mẽ, đa nền tảng, cho phép các nhà phát triển tạo ra các trò chơi với chất lượng cao. Với giao diện thân thiện và kho tài liệu phong phú, Unity cung cấp mọi thứ từ các công cụ lập trình, thiết kế đồ họa, cho tới khả năng xuất bản trên nhiều nền tảng khác nhau.

### ***Các tính năng của Unity 2D hỗ trợ phát triển platformer***

Hệ thống Unity Tilemap giúp dễ dàng tạo và lặp lại các chu kỳ thiết kế cấp độ và cho phép các nghệ sĩ và nhà thiết kế nhanh chóng tạo nguyên mẫu khi xây dựng thế giới trò chơi 2D. Trình chỉnh sửa Tilemap được cài đặt thông qua Trình quản lý gói. Physics Engine: Cung cấp các công cụ vật lý giúp tạo ra các hiệu ứng chuyển động mượt mà và chân thực. Một Tilemap bao gồm một lớp phủ lưới và một số thành phần khác làm việc cùng nhau. Toàn bộ hệ thống cho phép bạn vẽ các mức bằng cách sử dụng các công cụ gạch và cọ và xác định các quy tắc cho cách ứng xử của gạch. Với Tilemaps, bạn có thể tạo các nền tảng với các cạnh động, gạch hoạt hình, các vị trí gạch ngẫu nhiên và hơn thế nữa. Hệ thống Unity Tilemap hoàn hảo cho các dự án 2D có chứa mức độ chơi trò chơi vì nó cho phép người dùng đạt mức nguyên mẫu có thể được kiểm tra ngay lập tức trong công cụ trò chơi.

Sprite Animations là các clip hoạt hình được tạo cho tài sản 2D. Có nhiều cách khác nhau để tạo Sprite Animations. Một cách là tạo chúng từ Sprite Sheet, một bộ sưu tập Sprites được sắp xếp trong một lưới. Các Sprites sau đó được biên dịch thành một Clip hoạt hình sẽ phát từng Sprite để tạo hoạt hình, giống như một cuốn sách lật. Sprite Animations cũng có thể được tạo bằng cách sử dụng hoạt hình keyframe trong Unity, thông qua cửa sổ Animation.

Editor Scripting là một tính năng rất mạnh mẽ. Nó cho phép các nhà phát triển tạo ra các công cụ và mở rộng Unity Editor để thực hiện các tác vụ tùy chỉnh có thể hỗ trợ phát triển dự án của bạn. Điều này bao gồm menu “cheat ” tùy chỉnh, công cụ hoạt hình và hơn thế nữa. Unity cung cấp cho bạn sức mạnh để mở rộng chức năng và sử dụng của nó, làm cho nó trở thành một công cụ rất thích ứng có thể phù hợp với nhu cầu chính xác của bạn. Bạn có thể xác định cách các thuộc tính được hiển thị trong Thanh tra với Ngăn kéo thuộc tính tùy chỉnh. Bạn có thể tạo bất kỳ số lượng cửa sổ tùy chỉnh nào trong dự án của mình, chúng hoạt động giống như Thanh tra, Cảnh hoặc bất kỳ cửa sổ Unity tích hợp nào khác. Đây là một cách tuyệt vời để thêm giao diện người dùng vào hệ thống phụ cho trò chơi của bạn.

Colliders and Triggers: Trong hầu hết các dự án Unity, sẽ cần phải tạo ra các tương tác vật lý. Các tình huống trong đó vật lý là cần thiết sẽ bao gồm bất kỳ việc sử dụng trọng lực, hoặc khi các vật thể va chạm và phản ứng với nhau. Máy va chạm được áp dụng cho GameObjects để thể hiện hình dạng vật lý của chúng trong mô phỏng vật lý. Theo mặc định, GameObjects với thành phần Rigidbody được áp dụng sẽ bị Collider chặn (va chạm). Mặc dù Colliders có thể chặn Rigidbody, chúng cũng có thể được sử dụng làm Triggers. Khi một máy va chạm được đặt dưới dạng Trình kích hoạt, nó sẽ phát hiện các tương tác bên ngoài từ các đối tượng trò chơi khác và thực thi mã nằm trong chức năng OnTriggerEnter, OnTriggerExit hoặc OnTriggerStay trong tập lệnh..

### ***So sánh Unity với các công cụ phát triển khác***

Unreal Engine – Đồ họa đỉnh cao nhưng đầy thách thức: được phát triển bởi Epic Games, nổi bật với khả năng tạo ra đồ họa ấn tượng, phù hợp cho các dự án game AAA có yêu cầu cao về hình ảnh. Được hỗ trợ bởi công nghệ tiên tiến, Unreal Engine cung cấp các công cụ mạnh mẽ như ánh sáng động và hiệu ứng vật lý chi tiết. Điều này làm cho Unreal Engine trở thành lựa chọn hàng đầu của các nhà phát triển game chuyên nghiệp. Tuy nhiên, để tận dụng tối đa sức mạnh của Unreal Engine, người dùng cần có kiến thức sâu rộng về lập trình, đặc biệt là C++, cùng với một cấu hình máy tính đủ mạnh. Hệ thống Blueprint của Unreal giúp giảm bớt phần nào khó khăn cho những ai không rành về mã hóa, nhưng vẫn đòi hỏi thời gian để làm quen. Ngoài ra, hiệu suất và chất lượng đồ họa cao của Unreal có thể dẫn đến thời gian phát triển lâu hơn và chi phí cao hơn. Mặc dù vậy, Unreal Engine vẫn là lựa chọn lý tưởng cho các dự án cần tập trung vào chất lượng đồ họa và trải nghiệm người chơi đỉnh cao.

Godot – Công cụ mã nguồn mở linh hoạt và miễn phí: Godot là một công cụ phát triển game mã nguồn mở được cộng đồng đánh giá cao nhờ tính linh hoạt và khả năng tùy chỉnh mạnh mẽ. Đây là lựa chọn tuyệt vời cho cả các dự án 2D và 3D, đặc biệt với những ai muốn tìm kiếm một công cụ miễn phí nhưng không kém phần chuyên nghiệp. Godot sử dụng ngôn ngữ lập trình GDScript, tương tự Python, giúp việc học tập và triển khai trở nên dễ dàng hơn, đặc biệt với người mới bắt đầu. Công cụ này còn hỗ trợ các ngôn ngữ khác như C# và C++, tăng tính đa dạng trong lập trình. Tuy nhiên, khả năng xử lý đồ họa 3D của Godot vẫn chưa thể so sánh với Unity hay Unreal Engine, khiến nó phù hợp hơn với các dự án nhỏ hoặc vừa. Một điểm mạnh khác của Godot là giao diện người dùng trực quan và khả năng xuất bản trên nhiều nền tảng mà không yêu cầu phí bản quyền. Mặc dù vậy, cộng đồng người dùng của Godot vẫn đang phát triển, và tài nguyên hỗ trợ không phong phú bằng các công cụ lớn như Unity hoặc Unreal. Dù vậy, với tính linh hoạt và chi phí bằng 0, Godot là lựa chọn lý tưởng cho những nhà phát triển muốn thử nghiệm sáng tạo mà không bị ràng buộc bởi chi phí.

GameMaker Studio – Lựa chọn đơn giản cho game 2D: GameMaker Studio là một công cụ phát triển game tập trung chủ yếu vào 2D, được biết đến với giao diện kéo thả thân thiện, dễ tiếp cận cho cả những người không có nền tảng lập trình. Ngôn ngữ lập trình riêng của công cụ này, GML (GameMaker Language), được thiết kế đơn giản, dễ học, giúp người dùng nhanh chóng làm quen và triển khai ý tưởng của mình. GameMaker Studio phù hợp cho các dự án nhỏ hoặc trung bình, đặc biệt là trong lĩnh vực game di động hoặc indie. Tuy nhiên, điểm yếu lớn nhất của công cụ này là khả năng hỗ trợ đồ họa 3D còn hạn chế, khiến nó không phải là lựa chọn tốt cho các dự án yêu cầu hình ảnh phức tạp. Một nhược điểm khác là người dùng phải mua giấy phép riêng nếu muốn xuất bản trò chơi trên các nền tảng khác nhau, điều này có thể làm tăng chi phí phát triển. Dù vậy, với khả năng tập trung mạnh mẽ vào 2D và sự đơn giản trong sử dụng, GameMaker Studio là một giải pháp phù hợp cho những ai muốn nhanh chóng tạo ra một sản phẩm hoàn chỉnh mà không cần đầu tư quá nhiều thời gian hoặc nguồn lực.

## **Các khái niệm cốt lõi trong phát triển trò chơi**

### ***Vật lý trong game (Game physics)***

Hệ thống vật lý trong Unity là một thành phần quan trọng, giúp mô phỏng các hiện tượng vật lý thực tế trong môi trường game. Bằng cách sử dụng các thành phần như Rigidbody và Collider, Unity cho phép các đối tượng trong game tương tác với nhau thông qua các lực như trọng lực, va chạm và ma sát. Rigidbody cung cấp phương thức điều khiển chuyển động và vị trí của GameObject dựa trên các nguyên tắc vật lý, trong khi Collider xác định hình dạng của đối tượng để xử lý va chạm. Sự kết hợp giữa Rigidbody và Collider giúp tạo ra các tương tác vật lý chân thực, nâng cao trải nghiệm người chơi. [2]

Ngoài các thành phần cơ bản, Unity còn cung cấp các công cụ như Joints và Physics Materials để tinh chỉnh hành vi vật lý của các đối tượng. Joints cho phép kết nối hai Rigidbody với nhau hoặc với một điểm cố định trong không gian, tạo ra các chuyển động phức tạp như bản lề hoặc lò xo. Physics Materials cho phép điều chỉnh các thuộc tính như độ ma sát và độ đàn hồi, ảnh hưởng đến cách các đối tượng trượt hoặc nảy khi va chạm. Việc sử dụng các công cụ này giúp nhà phát triển tạo ra các hiệu ứng vật lý đa dạng và phù hợp với yêu cầu cụ thể của trò chơi. [3]

Để đảm bảo hiệu suất và tính chính xác trong mô phỏng vật lý, Unity sử dụng hệ thống FixedUpdate() để cập nhật các tính toán vật lý ở các khoảng thời gian cố định. Điều này giúp đồng bộ hóa các tính toán vật lý với tốc độ khung hình, tránh các vấn đề liên quan đến hiệu suất và đảm bảo tính nhất quán trong hành vi của các đối tượng. Ngoài ra, Unity còn cung cấp các công cụ như Physics Debugger để kiểm tra và tối ưu hóa các tương tác vật lý trong game, giúp nhà phát triển dễ dàng phát hiện và khắc phục các vấn đề liên quan đến vật lý. [3]

### ***Lập trình hướng sự kiện (event-driven programming)***

Lập trình hướng sự kiện (event-driven programming) là một mô hình trong đó luồng thực thi của chương trình được xác định bởi các sự kiện, chẳng hạn như đầu vào từ người dùng, thông báo từ các chương trình khác hoặc thay đổi trạng thái hệ thống. Trong phát triển game với Unity, mô hình này cho phép các đối tượng trong game phản ứng linh hoạt với các sự kiện xảy ra, giúp quản lý tương tác và hành vi của chúng một cách hiệu quả. Thay vì kiểm tra liên tục trạng thái trong hàm Update, các đối tượng có thể đăng ký lắng nghe các sự kiện cụ thể và chỉ phản hồi khi sự kiện đó xảy ra, giúp giảm tải tài nguyên và tăng hiệu suất cho game. [4]

Việc áp dụng lập trình hướng sự kiện trong Unity thường được thực hiện thông qua việc sử dụng các delegate và event trong C#. Delegate là một kiểu tham chiếu đến phương thức, cho phép gán các phương thức như là tham số, trong khi event là một delegate được gắn với một class để quản lý các sự kiện. Bằng cách sử dụng delegate và event, các thành phần trong game có thể giao tiếp với nhau một cách lỏng lẻo, giảm sự phụ thuộc chặt chẽ giữa các module. Chẳng hạn, khi một viên đạn va chạm với người chơi, sự kiện "PlayerHit" có thể được kích hoạt, và các hệ thống liên quan như cập nhật thanh máu, phát âm thanh bị thương, hoặc hiển thị hiệu ứng hình ảnh sẽ lắng nghe và phản hồi tương ứng. [5]

Việc triển khai hệ thống sự kiện trong Unity có thể được thực hiện thông qua việc xây dựng một EventManager để quản lý các sự kiện toàn cục. EventManager này sẽ cho phép các đối tượng đăng ký lắng nghe hoặc hủy đăng ký các sự kiện, cũng như kích hoạt các sự kiện khi cần thiết. Điều này tạo ra một kênh giao tiếp trung gian giữa các thành phần trong game, giúp tách biệt rõ ràng trách nhiệm của từng module và tăng tính linh hoạt trong việc mở rộng hoặc bảo trì mã nguồn. Ngoài ra, Unity còn cung cấp các lớp như UnityAction và UnityEvent để hỗ trợ việc triển khai các sự kiện một cách thuận tiện và hiệu quả. [6]

### ***Hệ thống hoạt ảnh (animation system)***

Hệ thống hoạt hình của Unity, thường được gọi là 'Mecanim', cung cấp một quy trình làm việc mạnh mẽ và tinh vi cho việc tạo và quản lý hoạt hình trong các dự án game. Nó hỗ trợ cả các đoạn hoạt hình nhập khẩu từ các công cụ bên ngoài như Autodesk 3ds Max hoặc Maya, cũng như các hoạt hình được tạo trực tiếp trong Unity. Mecanim cho phép áp dụng hoạt hình từ một mô hình nhân vật này sang mô hình khác thông qua tính năng retargeting, giúp tiết kiệm thời gian và công sức trong việc tạo hoạt hình cho nhiều nhân vật khác nhau. Ngoài ra, hệ thống còn cung cấp các công cụ để căn chỉnh các đoạn hoạt hình một cách đơn giản, cho phép xem trước các đoạn hoạt hình, các chuyển tiếp và tương tác giữa chúng, giúp các nhà hoạt hình có thể làm việc độc lập hơn với lập trình viên và tạo nguyên mẫu cũng như xem trước hoạt hình trước khi tích hợp vào mã gameplay.

Quy trình làm việc với hệ thống hoạt hình của Unity dựa trên khái niệm về các đoạn hoạt hình (Animation Clips), chứa thông tin về cách các đối tượng thay đổi vị trí, xoay hoặc các thuộc tính khác theo thời gian. Các đoạn này được tổ chức thành một hệ thống dạng lưu đồ gọi là Animator Controller, hoạt động như một máy trạng thái để theo dõi đoạn hoạt hình nào đang được phát và khi nào nên chuyển đổi hoặc pha trộn giữa các đoạn. Ví dụ, một Animator Controller đơn giản có thể chỉ chứa một hoặc hai đoạn để điều khiển một vật phẩm quay và nảy, hoặc để hoạt hình một cánh cửa mở và đóng vào thời điểm thích hợp. Trong khi đó, một Animator Controller phức tạp hơn có thể chứa hàng tá hoạt hình cho các hành động của nhân vật chính, và có thể pha trộn giữa nhiều đoạn cùng lúc để tạo ra chuyển động mượt mà khi người chơi di chuyển trong cảnh.

Hệ thống Mecanim cũng cung cấp các tính năng đặc biệt để xử lý các nhân vật hình người, bao gồm khả năng retargeting hoạt hình từ bất kỳ nguồn nào (như dữ liệu chuyển động, thư viện hoạt hình bên thứ ba) sang mô hình nhân vật của bạn, cũng như điều chỉnh các định nghĩa cơ bắp để đảm bảo nhân vật biến dạng chính xác. Các tính năng này được kích hoạt thông qua hệ thống Avatar của Unity, nơi các nhân vật hình người được ánh xạ tới một định dạng nội bộ chung. Mỗi thành phần - đoạn hoạt hình, Animator Controller và Avatar - được kết hợp trên một GameObject thông qua thành phần Animator, tham chiếu đến Animator Controller và, nếu cần, Avatar cho mô hình đó. Animator Controller, đến lượt nó, chứa các tham chiếu đến các đoạn hoạt hình mà nó sử dụng, tạo nên một hệ thống hoạt hình linh hoạt và hiệu quả cho các dự án game. [7]

## **Tổng quan về lập trình C# trong Unity**

### ***C# cơ bản và ứng dụng trong Unity***

C Sharp (C#) là ngôn ngữ lập trình thuần hướng đối tượng được Microsoft phát triển năm 2000 chủ yếu được thiết kế Anders Hejlsberg. Được xây dựng dựa trên 2 ngôn ngữ trước đó là C++ và Java nên C# có được sự cân bằng giữa Visual Basic, C++, Delphi và Java. Với các đặc điểm nổi bật như: khả năng hướng đổi tượng hỗ trợ đóng gói, đa hình, kế thừa; hỗ trợ lập trình bất đồng bộ giúp các tác vụ chạy song song như Java; quảng lý bộ nhớ tự đồng nhờ vào Garbage Collection giúp gaimr thiểu lỗi rò rỉ; cùng với đó là bộ thư viện phong phú và mạnh mẽ của .NET Framework giúp phát triển ứng dụng từ desktop, web đến game.

Từ những ưu điểm kể trên, C# được dùng làm ngôn ngữ chính để lập trình trong Unity với các ứng dụng cụ thể như sau:

* **Viết logic cho trò chơi**
* Quản lý các hành động của nhân vật: di chuyển, nhảy, tấn công, hoặc tương tác với môi trường.
* Xử lý sự kiện: phản hồi khi người chơi nhấn nút, thu thập vật phẩm, hoặc gặp kẻ thù.
* Điều khiển AI cho NPC (Non-Player Character).
* **Tương tác với Unity API**

Unity cung cấp các API tích hợp mạnh mẽ qua C# để quản lý các thành phần trong trò chơi, như:

* **MonoBehaviour:** Lớp cơ sở cho tất cả các script, hỗ trợ các phương thức như Start() và Update() để xử lý logic thời gian thực.
* **Transform:** Quản lý vị trí, xoay và tỷ lệ của các đối tượng trong game.
* **Physics:** Sử dụng các chức năng xử lý va chạm, trọng lực, và động lực học.
* **Phát triển công cụ trong Unity Editor**

Tạo các công cụ tùy chỉnh để hỗ trợ thiết kế game, chẳng hạn như hệ thống quản lý tài nguyên hoặc chỉnh sửa đối tượng trong môi trường làm việc.

* **Xử lý bất đồng bộ và đa luồng**

Sử dụng các tính năng như async/await trong C# để xử lý các tác vụ như tải dữ liệu, chơi nhạc, hoặc kết nối mạng mà không làm chậm trò chơi.

* **Tích hợp công nghệ hiện đại**
* Phát triển game đa người chơi (Multiplayer) với API mạng.
* Tích hợp các plugin và thư viện từ bên thứ ba để mở rộng tính năng.

### ***Script và cách hoạt động trong Unity***

Script trong Unity có nhiệm vụ chạy các chắc năng bằng mã thay vì xử lý bằng **Unity Editor UI.** Điều này cho phép bạn vượt ra khỏi những giới hạn rằng buộc bởi **Editor UI**. Ngoài ra, việc công khai tương tác trực tiếp với các API Unity cho phép chúng có quyền kiểm soát và tinh chỉnh ở mức độ cao hơn.

Trong Unity, **Script** giúp định nghĩa các logic và hành vi của các đối tượng trong trò chơi. Unity sử dụng **C# scripts** để viết mã, cho phép tương tác với Unity Engine thông qua các API mạnh mẽ nh:.

* **Tạo và Gắn Script:**
* Tạo script trong Unity thông qua Project Window, Unity sẽ tự động tạo tệp với cấu trúc cơ bản kế thừa từ lớp **MonoBehaviour**.
* Gắn script vào một **GameObject** trong Scene, script sẽ trở thành một **Component** của GameObject.
* **Cấu Trúc Cơ Bản Của Script:**
* Start(): Được gọi một lần khi GameObject được kích hoạt trong Scene, dùng để khởi tạo.
* Update(): Được gọi mỗi khung hình (frame), dùng để xử lý các thay đổi liên tục.
* **Tương Tác Với Unity API:**
* Transform: Điều khiển vị trí, xoay, và tỷ lệ của GameObject.
* Rigidbody: Thêm các tính năng vật lý như trọng lực hoặc lực.
* Collider: Phát hiện và xử lý va chạm.
* Input: Xử lý đầu vào từ bàn phím, chuột, hoặc tay cầm điều khiển.
* **Các Vòng Đời Quan Trọng Của MonoBehaviour:**
* Awake(): Gọi trước Start(), thường dùng để khởi tạo các thành phần hoặc tài nguyên.
* OnEnable()/OnDisable(): Gọi khi GameObject được kích hoạt hoặc tắt.
* FixedUpdate(): Chạy ở mỗi khung vật lý, thường dùng cho logic vật lý.
* LateUpdate(): Gọi sau Update(), dùng cho các tác vụ liên quan đến camera.
* **Tích Hợp Sự Kiện Và Tương Tác:**
* Xử lý va chạm thông qua các sự kiện như OnCollisionEnter.
* Xử lý trigger với OnTriggerEnter.
* **Quản Lý Bất Đồng Bộ (Asynchronous):**
* Unity hỗ trợ **Coroutine** để xử lý các tác vụ diễn ra theo thời gian, ví dụ: chờ hoặc lặp.
* Sử dụng các phương thức như yield return để điều khiển luồng thực thi.
* **Kết Nối Giữa Các Script:**
* Tham chiếu trực tiếp thông qua GetComponent.
* Tham chiếu công khai bằng cách kéo thả trong Unity Editor.

### ***Các thư viện hỗ trợ hữu ích***

Unity đã cung cấp rất nhiều bộ thư viện mạnh mẽ để hỗ trợ các lập trình viên và tự động quản lý các thành phần của trò chơi. Dưới đây có 6 thư viện quan trọng và được sử dụng phổ biến nhất:

Thư viện UnityEngine: Là nền tảng cốt lõi của Unity, cung cấp các lớp và chức năng cần thiết để phát triển trò chơi và ứng dụng tương tác. Nó bao gồm các thành phần quan trọng như GameObject, Transform, Vector3, và nhiều đối tượng khác, cho phép nhà phát triển tạo và quản lý các yếu tố trong thế giới ảo. Ví dụ, GameObject là đơn vị cơ bản của mọi đối tượng trong Unity, trong khi Transform xác định vị trí, xoay và tỉ lệ của đối tượng đó trong không gian. Sự kết hợp của các lớp này giúp xây dựng cấu trúc và hành vi cho các thành phần trong trò chơi một cách hiệu quả.

Ngoài các lớp cơ bản, UnityEngine còn cung cấp các công cụ để xử lý vật lý, âm thanh, hoạt hình và giao diện người dùng. Các lớp như Rigidbody và Collider hỗ trợ mô phỏng vật lý thực tế, cho phép các đối tượng tương tác thông qua va chạm và lực. Thành phần AudioSource và AudioClip giúp tích hợp âm thanh vào trò chơi, tạo ra trải nghiệm sống động cho người chơi. Hệ thống hoạt hình được quản lý thông qua các lớp như Animator và AnimationClip, cho phép tạo ra các chuyển động mượt mà và phức tạp cho nhân vật và đối tượng. Bên cạnh đó, các lớp như Canvas và UI hỗ trợ xây dựng giao diện người dùng, giúp tạo ra các menu, nút bấm và các thành phần tương tác khác.

Việc sử dụng thư viện UnityEngine đòi hỏi kiến thức về lập trình C# và hiểu biết về cấu trúc của Unity. Nhà phát triển cần nắm vững cách sử dụng các lớp và phương thức trong thư viện để tạo ra các chức năng mong muốn. Tài liệu chính thức của Unity cung cấp thông tin chi tiết về từng thành phần trong UnityEngine, bao gồm mô tả, cú pháp và ví dụ minh họa, giúp nhà phát triển dễ dàng tra cứu và áp dụng trong dự án của mình. Ngoài ra, cộng đồng Unity cũng chia sẻ nhiều tài nguyên và hướng dẫn hữu ích, hỗ trợ việc học tập và giải quyết các vấn đề trong quá trình phát triển. [4]

**UnityEngine.UI:** cũng thư viện hỗ trợ việc xây dựng giao diện người dùng (UI) cho trò chơi. Nó cung cấp các thành phần như Button, Text, Canvas, giúp bạn dễ dàng tạo các nút, hiển thị văn bản hoặc quản lý bố cục giao diện. Ngoài ra, UnityEngine.UI còn hỗ trợ tùy chỉnh hành vi giao diện, chẳng hạn như xử lý sự kiện khi người chơi nhấn nút, kéo thanh trượt hoặc điều chỉnh giao diện theo kích thước màn hình.

**Tiếp đến là UnityEngine.SceneManagement** có nhiệm vụ quản lý các Scene trong trò chơi là nhiệm vụ chính của thư viện này. Bạn có thể sử dụng LoadScene() để chuyển đổi giữa các màn chơi hoặc UnloadScene() để giải phóng bộ nhớ của Scene không còn cần thiết. Thư viện còn hỗ trợ các thao tác tải Scene theo cách bất đồng bộ, giúp giảm thời gian chờ và tối ưu hóa hiệu suất của trò chơi.

Theo đó là: **UnityEngine.Physics** là bộthư viện vật lý giúp mô phỏng các hiện tượng thực tế như trọng lực, va chạm, hoặc lực tác động giữa các đối tượng. Rigidbody cho phép các vật thể tương tác theo quy luật vật lý, trong khi Collider và Raycast hỗ trợ phát hiện va chạm hoặc kiểm tra các vật thể trong không gian 3D. Sử dụng UnityEngine.Physics, bạn có thể tạo các hiệu ứng vật lý sống động và chân thực hơn.

Trong đó, **UnityEngine.AI** lại dành riêng cho phát triển trí tuệ nhân tạo (AI), thư viện này giúp các đối tượng như NPC (Non-Player Character) tự tìm đường và di chuyển thông minh. NavMesh tạo bản đồ đường đi, NavMeshAgent kiểm soát di chuyển của nhân vật AI, còn NavMeshObstacle xử lý các chướng ngại vật trong môi trường. Với thư viện này, bạn có thể tạo ra AI phản ứng linh hoạt và phù hợp với môi trường trò chơi.

Cuối cùng là **TextMeshPro.** Đây là thư viện mạnh mẽ để xử lý văn bản trong trò chơi, cho phép hiển thị chữ với độ phân giải cao và hiệu suất tốt hơn so với các công cụ văn bản mặc định. Nó cung cấp các tùy chọn như đổi màu chữ, hiệu ứng đổ bóng, và tạo kiểu chữ 3D, giúp cải thiện chất lượng hiển thị văn bản. TextMeshPro là lựa chọn lý tưởng cho các trò chơi cần giao diện đẹp và rõ ràng.

# **CHƯƠNG 2: TỔNG QUAN ĐỀ TÀI**

1. **Mục tiêu và phạm vi của đề tài**

### ***Mục tiêu chính của trò chơi***

Thử thách kỹ năng xử lý tình huống của người chơi: Trò chơi "Fun\_Trap" được thiết kế để kiểm tra khả năng tư duy nhanh và xử lý tình huống của người chơi. Trong môi trường đầy cạm bẫy, người chơi phải liên tục đưa ra quyết định trong thời gian ngắn để vượt qua các chướng ngại vật. Điều này không chỉ yêu cầu sự nhanh nhẹn mà còn đòi hỏi khả năng dự đoán và lập kế hoạch. Mỗi cấp độ trong trò chơi đều có sự gia tăng về độ khó, tạo động lực để người chơi không ngừng cải thiện kỹ năng.

Khuyến khích tính kiên nhẫn và sự bền bỉ: Trò chơi tạo ra một loạt thử thách khiến người chơi có thể thất bại nhiều lần trước khi hoàn thành mục tiêu. Tuy nhiên, chính những lần thất bại này sẽ rèn luyện tính kiên nhẫn và ý chí không bỏ cuộc. "Fun\_Trap" không chỉ là một trò chơi giải trí mà còn là một bài học về sự kiên trì trong cuộc sống.

Tăng cường khả năng ghi nhớ và phản xạ: Một phần quan trọng của trò chơi là việc người chơi phải ghi nhớ vị trí của các bẫy hoặc nguy hiểm đã gặp phải. Kết hợp với khả năng phản xạ nhanh, "Fun\_Trap" giúp người chơi rèn luyện trí nhớ và cải thiện khả năng phản ứng với những thay đổi bất ngờ.

Mang lại niềm vui và giải trí: Bên cạnh những yếu tố thử thách, "Fun\_Trap" vẫn giữ mục tiêu chính là tạo ra niềm vui cho người chơi. Thiết kế đồ họa, âm thanh và các yếu tố bất ngờ trong trò chơi mang lại trải nghiệm thú vị, khiến người chơi cảm thấy hứng thú trong mỗi lần chinh phục.

### ***Phạm vi phát triển (thời gian, công cụ, nguồn lực)***

Kế hoạch phát triển trò chơi "Fun\_Trap" bắt đầu từ ngày 21/10/2024 và kéo dài 15 tuần, chia thành 7 giai đoạn cụ thể. Mỗi giai đoạn đều được làm đứng tiến độ dưới sự theo dõi và hướng dẫn của giảng viên hướng dẫn ThS. Trần Trung.

Giai đoạn đầu tiên, kéo dài 1 tuần, là thời điểm phát triển bước vào thế giới của Unity – công cụ sẽ định hình toàn bộ trò chơi "Fun\_Trap". Đây không chỉ là quá trình làm quen với giao diện và các tính năng cơ bản của Unity mà còn là cơ hội để khám phá những tiềm năng sáng tạo mà công cụ này mang lại. Trong tuần này, cần tập trung nghiên cứu cách Unity hỗ trợ phát triển trò chơi 2D, từ quản lý tài nguyên, tạo hình ảnh động đến tích hợp âm thanh. Đồng thời, đây cũng là lúc ý tưởng trò chơi được mài giũa, chuyển từ những phác thảo sơ khai thành một phong cách cụ thể, độc đáo. Bằng cách kết hợp sự hiểu biết về công cụ với những ý tưởng sáng tạo, nhóm đặt nền móng vững chắc cho toàn bộ dự án, đảm bảo rằng trò chơi không chỉ khả thi về mặt kỹ thuật mà còn hấp dẫn về mặt trải nghiệm.

Tiếp nối giai đoạn đầu, cần dành 2 tuần để thu thập và nghiên cứu tài liệu, một bước đi quan trọng nhằm xây dựng nền tảng lý thuyết vững chắc cho trò chơi "Fun\_Trap". Đây là thời điểm tập trung tìm hiểu về cơ chế trò chơi, thiết kế giao diện người dùng, và cách tạo ra những yếu tố thử thách hấp dẫn. Các tài liệu tham khảo bao gồm hướng dẫn sử dụng Unity, tài liệu về thiết kế trò chơi, và nghiên cứu các trò chơi tương tự để học hỏi ý tưởng. Đồng thời, nhóm sẽ phân tích và lập kế hoạch cho các yếu tố chính như cách di chuyển của nhân vật, cách hoạt động của cạm bẫy, và phần thưởng trong trò chơi. Kết quả của giai đoạn này không chỉ là một bản thiết kế sơ bộ mà còn là sự thống nhất về tầm nhìn sáng tạo, đảm bảo có thể hiểu rõ mục tiêu và phương hướng phát triển của dự án. Đây là bước đệm để biến ý tưởng thành hiện thực một cách hiệu quả và có tổ chức.

Trong 4 tuần tiếp theo, trọng tâm của sẽ là phát triển các thành phần cốt lõi của trò chơi "Fun\_Trap". Hai tuần đầu tiên được dành riêng để xây dựng nhân vật chính, bao gồm thiết kế hình dáng, chuyển động và các chức năng cơ bản như di chuyển, nhảy hoặc tương tác với môi trường. Đây là bước quan trọng nhằm đảm bảo nhân vật hoạt động mượt mà, mang lại cảm giác điều khiển tự nhiên và thú vị. Sau khi hoàn thiện nhân vật chính, tôi sẽ dành 2 tuần tiếp theo để phát triển các yếu tố thử thách như cạm bẫy và phần thưởng. Các cạm bẫy sẽ được thiết kế với mức độ khó tăng dần, tạo động lực cho người chơi cải thiện kỹ năng. Đồng thời, phần thưởng và các chức năng hỗ trợ sẽ được tích hợp để khuyến khích và giữ chân người chơi. Việc cân bằng giữa thử thách và phần thưởng sẽ giúp trò chơi trở nên hấp dẫn và đầy tính giải trí. Đây là giai đoạn quan trọng, đặt nền tảng cho tính chất đặc trưng và lối chơi độc đáo của "Fun\_Trap".

Trong 2 tuần tiếp theo, tôi sẽ tập trung thiết kế và lập trình các kẻ thù trong trò chơi "Fun\_Trap" để tạo thêm thử thách và chiều sâu cho trải nghiệm người chơi. Quá trình này bắt đầu bằng việc xây dựng hình dáng và hành vi của các kẻ thù, từ những đối thủ di chuyển đơn giản đến những kẻ có khả năng tương tác phức tạp hơn với môi trường và nhân vật chính. Để đảm bảo sự đa dạng, tôi sẽ thiết kế nhiều loại kẻ thù với đặc điểm riêng, như tốc độ di chuyển khác nhau, khả năng tấn công đặc biệt, hoặc cách xuất hiện bất ngờ.

Ngoài ra, yếu tố cân bằng sẽ được đặt lên hàng đầu. Tôi sẽ điều chỉnh mức độ khó của các kẻ thù sao cho phù hợp với tiến độ của người chơi, tạo nên những thử thách vừa đủ mà không gây cảm giác quá áp lực. Việc tích hợp các kẻ thù với cạm bẫy và phần thưởng đã thiết kế trước đó cũng là một phần quan trọng, đảm bảo chúng hoạt động đồng bộ và mang lại trải nghiệm chơi game mượt mà.

Đây là giai đoạn đòi hỏi sự sáng tạo và tư duy logic, bởi tôi cần đảm bảo rằng mỗi kẻ thù không chỉ là thử thách mà còn góp phần làm nổi bật nét đặc trưng của trò chơi. Kết quả mong đợi là một hệ thống kẻ thù vừa hấp dẫn vừa thúc đẩy người chơi khám phá và chinh phục "Fun\_Trap".

Trong 4 tuần tiếp theo, tôi sẽ tập trung vào giai đoạn kiểm thử, một bước quan trọng nhằm đảm bảo trò chơi "Fun\_Trap" hoạt động ổn định và đạt chất lượng cao. Quá trình kiểm thử sẽ bắt đầu với việc kiểm tra các chức năng chính, như chuyển động của nhân vật, hoạt động của cạm bẫy, và hành vi của kẻ thù. Tôi sẽ sử dụng phương pháp kiểm thử thủ công kết hợp với công cụ tự động để phát hiện lỗi logic, lỗi đồ họa, và các vấn đề về hiệu suất. Đặc biệt, tôi sẽ chú ý đến trải nghiệm người chơi bằng cách thử nghiệm trên các cấu hình máy tính khác nhau để đảm bảo tính tương thích.

Bên cạnh việc xử lý lỗi, tôi sẽ xây dựng khung báo cáo chức năng, trong đó mô tả chi tiết cách các thành phần của trò chơi hoạt động và tương tác với nhau. Báo cáo này không chỉ là tài liệu kỹ thuật mà còn giúp tôi hệ thống hóa toàn bộ quá trình phát triển, từ ý tưởng đến thực thi.

Cuối cùng, 2 tuần cuối cùng sẽ được dành riêng để hoàn thiện báo cáo đồ án. Tôi sẽ tổng hợp tất cả các kết quả, bao gồm phân tích tiến trình, thành tựu đạt được, và những khó khăn đã vượt qua. Đây cũng là lúc tôi đánh giá tổng thể dự án, đưa ra những bài học kinh nghiệm và đề xuất hướng phát triển trong tương lai. Việc hoàn thiện báo cáo không chỉ giúp tôi khép lại dự án một cách chuyên nghiệp mà còn tạo tiền đề cho những dự án tiếp theo

1. **Phân tích các trò chơi platformer hiện có**

### ***Phân tích các trò chơi điển hình (ví dụ: Mario, Hollow Knight)***

Super Mario là một trong những trò chơi điển hình của thể loại platformer, nổi bật với lối chơi đơn giản nhưng cực kỳ cuốn hút. Trong Mario, người chơi nhập vai vào nhân vật chính để vượt qua các chướng ngại vật, đánh bại kẻ thù và hoàn thành các màn chơi. Cơ chế nhảy được thiết kế rất linh hoạt, đóng vai trò trung tâm trong toàn bộ trải nghiệm. Bên cạnh đó, các yếu tố power-ups như nấm tăng kích thước hay hoa lửa cung cấp đa dạng chiến thuật, tạo thêm động lực khám phá màn chơi. Mỗi cấp độ trong Mario đều có thiết kế tuyến tính nhưng không hề nhàm chán nhờ sự kết hợp khéo léo giữa thử thách và phần thưởng. Người chơi dễ dàng bị thu hút bởi cách Mario tích hợp những bí mật nhỏ như khu vực ẩn hay đồng xu thưởng. Với đồ họa 2D đơn giản nhưng đầy màu sắc, kết hợp âm nhạc vui nhộn, Mario mang đến trải nghiệm vừa giải trí, vừa khuyến khích người chơi rèn luyện kỹ năng.

Hollow Knight là đại diện tiêu biểu của thể loại Metroidvania, mang đến trải nghiệm khám phá và chiến đấu đầy thử thách. Trò chơi xây dựng một thế giới phi tuyến tính, nơi người chơi cần mở khóa các kỹ năng đặc biệt để truy cập vào những khu vực mới. Không giống như Mario, Hollow Knight có bản đồ phức tạp, nhiều đường rẽ và sự kết nối ngầm, tạo nên cảm giác bí ẩn. Lối chơi tập trung vào chiến đấu cận chiến đòi hỏi sự chính xác cao, kết hợp cùng các cơ chế khám phá giúp nâng cao chiều sâu trải nghiệm. Thẩm mỹ của trò chơi mang phong cách gothic, với đồ họa 2D chi tiết và tông màu u tối, tạo nên bầu không khí huyền bí. Câu chuyện được kể chủ yếu qua môi trường, các đoạn hội thoại ngắn và các vật phẩm thu thập được, khuyến khích người chơi tự tìm hiểu. Độ khó cao nhưng công bằng của Hollow Knight giúp người chơi cảm nhận rõ ràng thành tựu khi vượt qua thử thách.

Super Mario và Hollow Knight, mặc dù cùng thuộc dòng game 2D, lại mang đến hai phong cách trải nghiệm hoàn toàn khác biệt. Mario hướng đến lối chơi nhanh, trực quan và tập trung vào thử thách vượt chướng ngại vật. Ngược lại, Hollow Knight mang lại cảm giác khám phá sâu sắc và nhấn mạnh yếu tố chiến đấu phức tạp. Trong khi Mario có cấp độ thiết kế tuyến tính, dễ tiếp cận với mọi người chơi, Hollow Knight lại xây dựng thế giới mở rộng, phi tuyến tính và chứa đựng nhiều bí mật. Về mặt thẩm mỹ, Mario sử dụng đồ họa sáng sủa, vui tươi, phù hợp với mọi lứa tuổi, còn Hollow Knight lại chọn phong cách gothic u tối, phù hợp với những người yêu thích chiều sâu và sự huyền bí. Cả hai đều rất thành công trong việc mang đến trải nghiệm độc đáo nhờ vào thiết kế thông minh và khả năng thu hút người chơi ở từng chi tiết.

### ***Điểm mạnh và yếu của các trò chơi này***

Super Mario và Hollow Knight đều là những tựa game điển hình, sở hữu nhiều điểm mạnh vượt trội, nhưng đồng thời cũng tồn tại một số hạn chế. Super Mario nổi bật với lối chơi dễ tiếp cận, thích hợp cho mọi lứa tuổi nhờ thiết kế đơn giản nhưng hiệu quả. Cơ chế nhảy linh hoạt cùng sự xuất hiện của các power-ups như nấm hoặc sao giúp duy trì cảm giác vui nhộn và động lực chơi game. Bên cạnh đó, thiết kế tuyến tính của từng màn chơi giúp người chơi dễ dàng tiếp cận và hiểu rõ mục tiêu. Tuy nhiên, chính yếu tố tuyến tính này đôi khi khiến trò chơi thiếu sự thử thách đối với những người chơi lâu năm. Mario cũng không tập trung nhiều vào cốt truyện, dẫn đến việc thiếu chiều sâu trong trải nghiệm đối với một số đối tượng thích khám phá.

Hollow Knight lại có điểm mạnh ở thiết kế thế giới phức tạp, giàu chiều sâu và khuyến khích khám phá. Người chơi luôn bị hấp dẫn bởi những khu vực ẩn và câu chuyện được kể thông qua môi trường, mang lại cảm giác gắn kết. Cơ chế chiến đấu đòi hỏi sự chính xác và kỹ năng cao tạo ra thử thách đáng giá. Tuy nhiên, nhược điểm của Hollow Knight là độ khó cao, có thể khiến người chơi mới cảm thấy nản lòng. Việc không có hướng dẫn rõ ràng trong giai đoạn đầu trò chơi cũng gây khó khăn cho những ai không quen với thể loại Metroidvania. Ngoài ra, phong cách thẩm mỹ u tối và nhịp độ chậm rãi có thể không phù hợp với những người chơi yêu thích lối chơi nhanh và giải trí đơn giản.

### ***Bài học kinh nghiệm cho đề tài***

Từ việc phân tích Super Mario và Hollow Knight, chúng ta có thể rút ra nhiều bài học quan trọng để áp dụng vào quá trình thiết kế và phát triển trò chơi. Đầu tiên, tính dễ tiếp cận và sự cân bằng giữa thử thách và phần thưởng là yếu tố quan trọng giúp giữ chân người chơi. Super Mario đã chứng minh rằng việc thiết kế cơ chế chơi đơn giản nhưng chặt chẽ, kết hợp cùng các phần thưởng như power-ups hoặc đồng xu, có thể tạo ra động lực mạnh mẽ. Vì vậy, trong quá trình phát triển trò chơi, cần đảm bảo rằng người chơi mới có thể dễ dàng tiếp cận, nhưng vẫn có những thử thách hấp dẫn để họ tiếp tục khám phá.

Thứ hai, Hollow Knight dạy chúng ta về tầm quan trọng của chiều sâu trong thiết kế thế giới và cốt truyện. Một thế giới phi tuyến tính với các khu vực liên kết chặt chẽ, bí mật ẩn giấu, và những yếu tố câu chuyện được hé lộ qua môi trường sẽ tăng cường trải nghiệm người chơi. Điều này cho thấy rằng việc đầu tư vào yếu tố khám phá và xây dựng nội dung đa tầng là rất cần thiết, đặc biệt nếu muốn tạo ra trò chơi mang lại cảm giác dài lâu và ý nghĩa.

Bài học khác từ cả hai trò chơi là việc thiết kế phong cách thẩm mỹ và âm thanh đồng bộ với lối chơi. Super Mario dùng đồ họa sáng sủa và nhạc nền vui nhộn để nhấn mạnh tính giải trí, trong khi Hollow Knight lựa chọn phong cách gothic với tông màu tối để phù hợp với bầu không khí bí ẩn và thử thách. Điều này cho thấy sự hài hòa giữa yếu tố nghệ thuật và cơ chế trò chơi có vai trò quan trọng trong việc xây dựng thương hiệu và ấn tượng của trò chơi.

Cuối cùng, cần chú ý đến mức độ khó phù hợp với đối tượng mục tiêu. Super Mario thành công nhờ khả năng mở rộng đối tượng từ trẻ em đến người lớn, trong khi Hollow Knight tạo dựng tên tuổi bằng việc tập trung vào những người chơi yêu thích thử thách. Vì vậy, khi phát triển trò chơi, cần xác định rõ đối tượng người chơi và xây dựng mức độ khó phù hợp để giữ chân họ, đồng thời cung cấp trải nghiệm thú vị và đáng nhớ.

1. **Ý tưởng phát triển trò chơi**

### ***Cốt truyện và nội dung***

Câu chuyện của trò chơi này bắt đầu từ những bí mật bị chôn vùi trong quá khứ, nơi vương quốc một thời hùng mạnh đã sụp đổ dưới sức mạnh của những phù thủy tăm tối. Hàng thế kỷ đã trôi qua, và những di vật cổ xưa cùng những bẫy chết người của họ dần bị lãng quên. Nhưng khi một thanh niên trẻ khám phá ra bản đồ cổ và những lời nguyền vẫn còn tồn tại, anh ta đã đặt chân vào một cuộc phiêu lưu đầy nguy hiểm để tìm ra sự thật. Liệu anh ta có thể vượt qua những cạm bẫy chết người, giải mã được những bí ẩn mà thế giới này giấu kín, và thay đổi số phận của vương quốc? Cuộc hành trình không chỉ là về việc sống sót, mà còn là sự đấu tranh với quá khứ để cứu lấy tương lai.

* Quá khứ:

Trước đây, thế giới của trò chơi là một vương quốc yên bình, nơi người dân sống hòa bình dưới sự cai trị của một vị vua sáng suốt. Vị vua này nổi tiếng với trí tuệ và lòng nhân từ, đã đưa ra những quyết định khôn ngoan để bảo vệ vương quốc. Tuy nhiên, trong bóng tối của sự thịnh vượng này, một nhóm phù thủy mưu mô đã âm mưu lật đổ vua. Họ đã tạo ra một mạng lưới bẫy và thử thách nguy hiểm trong các khu vực của vương quốc, hy vọng sẽ khiến những ai dám xâm nhập vào khu vực này bị tiêu diệt. Các bẫy này đã bị chôn vùi qua thời gian, bị quên lãng khi các thế hệ mới lớn lên và không còn ai nhớ đến chúng.

* Hiện tại:

Vương quốc đã suy tàn, và vị vua cũng đã qua đời. Thế nhưng, một ngày nọ, một thanh niên trẻ (nhân vật chính của trò chơi) phát hiện ra một bí mật từ di sản của gia đình mình. Người này phát hiện ra một chiếc bản đồ cổ xưa dẫn đến những khu vực bị lãng quên, nơi chứa đựng những bẫy chết người mà các phù thủy đã tạo ra. Thay vì hoảng sợ, thanh niên này quyết định khám phá những khu vực đó để tìm kiếm những di vật quý giá, cũng như những câu chuyện bị chôn vùi trong quá khứ. Tuy nhiên, hắn nhanh chóng nhận ra rằng đây không phải là một nhiệm vụ dễ dàng. Các bẫy được thiết kế một cách tinh vi và cực kỳ nguy hiểm, thách thức khả năng sinh tồn của hắn.

Trong quá trình phiêu lưu, nhân vật chính phải đối mặt với những thử thách khó khăn, vượt qua các cạm bẫy, cũng như những sinh vật cổ xưa do phù thủy tạo ra để bảo vệ kho báu. Qua đó, hắn dần phát hiện ra một âm mưu đen tối đang diễn ra, và cuộc hành trình không chỉ là về việc khám phá những bí mật, mà còn là chiến đấu chống lại những thế lực xấu xa muốn phá hủy thế giới.

* Tương lai:

Khi nhân vật chính vượt qua tất cả những thử thách và phá vỡ lời nguyền của các phù thủy, hắn sẽ đối mặt với một thế lực lớn hơn, một tổ chức mới nổi dậy từ những mảnh vỡ của quá khứ. Tổ chức này mong muốn tái lập quyền lực của các phù thủy xưa và lôi kéo thế giới vào một cuộc chiến tranh đẫm máu. Nhân vật chính sẽ phải huy động các đồng minh từ những người sống sót và những nhân vật trong quá khứ để ngăn chặn âm mưu này.

Tương lai của vương quốc phụ thuộc vào hành động của nhân vật chính. Nếu hắn thất bại, thế giới sẽ chìm vào bóng tối và nỗi sợ hãi. Nhưng nếu thành công, hắn sẽ giúp khôi phục lại hòa bình, nhưng không phải là sự yên bình như trước kia, mà là một vương quốc nơi mọi người phải luôn cảnh giác trước những bóng ma của quá khứ và những thử thách không bao giờ chấm dứt.

### ***Tính năng chính và cơ chế gameplay (game mechanics)***

Gameplay được thiết kế xoay quanh lối chơi đơn giản nhưng đồng thời mang tính thách thức cao. Dòng game có thể nhấn mạnh vào yếu tố phản xạ nhanh nhạy và điều khiển nhân vật để vượt qua các chướng ngại vật hoặc thu thập vật phẩm. Thứ tự game dự kiến sẽ mang phong cách vui nhộn, đơn giản nhưng cũng đầy tính cuốn hút, nhấn mạnh vào sự thư giãn và đáp ứng nhanh của người chơi. Gameplay nhấn mạnh đến việc cân bằng giữa thời gian ngắn và mục độ thách thức, giúp người chơi luôn tìm được động lực quay lại chấp nhận các thử thách tiếp theo.

Hệ thống nhân vật và mục tiêu: Cơ chế nhân vật trong gameplay có thể bao gồm các yếu tố như chuyển động tự do, các hành động chính (nhảy, né tránh, hoặc tương tác). Nếu gameplay mang yếu tố giải đố, các tài sản trong game có thể yêu cầu người chơi kết hợp logic và kỹ năng quan sát để đạt được mục tiêu. Đặc biệt, nếu game hỗ trợ hệ thống nhân vật có thể nâng cấp hoặc thay đổi ngoại hình, người chơi sẽ được trải nghiệm tính cá nhân hóa, tăng tính hấp dẫn.

Tính năng môi trường và tương tác: Một yếu tố quan trọng trong gameplay có thể là cách môi trường thay đổi hoặc tương tác với hành động của người chơi. Các hình ảnh từ thư mục gợi ý về một môi trường có thể mang tính động (như thay đổi ánh sáng, địa hình, hoặc các yếu tố bất ngờ). Môi trường này sẽ tạo cảm giác sống động, thách thức người chơi vượt qua bằng cách tận dụng cơ chế nhảy, né, hoặc phản xạ nhanh trước các sự kiện bất ngờ.

Cơ chế phần thưởng và thách thức: Hệ thống phần thưởng có thể được xây dựng dựa trên thành tích của người chơi, như hoàn thành thử thách trong thời gian giới hạn hoặc thu thập vật phẩm đặc biệt. Ngược lại, các thách thức như chướng ngại vật, câu đố phức tạp, hoặc sự giới hạn tài nguyên sẽ tăng độ khó và kích thích người chơi quay lại để cải thiện kết quả. Điều này giúp gameplay giữ được sự hấp dẫn trong thời gian dài.

Yếu tố mở rộng và tính năng bổ sung: Cuối cùng, để tăng giá trị chơi lại, game có thể tích hợp các tính năng bổ sung như chế độ chơi mới, bảng xếp hạng trực tuyến, hoặc nội dung mở rộng. Điều này không chỉ tăng cường sự gắn bó của người chơi mà còn mở ra tiềm năng cho các bản cập nhật trong tương lai. Sự kết hợp giữa gameplay sáng tạo và nội dung mở rộng sẽ làm nên sức hút lâu dài cho trò chơi.

### ***Điểm khác biệt và sáng tạo của trò chơi***

Trò chơi được thiết kế với những đặc điểm đồng thời gây bất ngờ và thu hút người chơi. Trước hết, các mảnh ván chơi luôn thay đổi để duy trì tính tươi mới. Cơ chế gameplay được tích hợp giữa những thách thức đòi hỏi phản xạ nhanh nhạy và các khoảnh khắc vui nhộn nhẹ nhàng. Hệ thống thắng thua được thiết kế có tính động cao, đem lại cảm giác thắng đầy thành tích những không quá khó để gây nản lòng.

Một trong những yếu tố sáng tạo đồng thời làm nên sự khác biệt chính là hệ thống vật phẩm. Các vật phẩm không chỉ mang tính trang trí mà còn yêu cầu người chơi sử dụng một cách thông minh để vượt qua những khó khăn. Đặc biệt, các cấp độ thách thức thường tích hợp nhiều yếu tố độc đáo như môi trường thay đổi liên tục, hoặc chông gai vàng bất ngờ. Tính linh hoạt trong thiết kế khiến người chơi luôn phải điều chỉnh chiến lược.

1. **Các công cụ và công nghệ áp dụng**

### ***Unity và plugin cần thiết***

Gameplay của dự án có thể được thiết kế xoay quanh lối chơi đơn giản nhưng đồng thời mang tính thách thức cao. Gameplay nhấn mạnh vào yếu tố phản xạ nhanh nhạy và kỹ năng điều khiển nhân vật để vượt qua các chướng ngại vật hoặc thu thập vật phẩm. Trò chơi hướng đến phong cách vui nhộn và dễ tiếp cận, nhưng vẫn đầy tính cuốn hút nhờ cân bằng khéo léo giữa thời gian chơi ngắn và mức độ thách thức. Điều này tạo động lực để người chơi quay lại, nhất là những ai tìm kiếm sự giải trí nhẹ nhàng mà vẫn kết hợp tính thống trí.

TextMeshPro (3.0.6): TextMeshPro là một plugin hỗ trợ hiển thị văn bản chất lượng cao trong Unity. Plugin này cung cấp nhiều tùy chọn để tăng tính thẩm mỹ và hiển thị văn bản rõ ràng dù trên các thiết bị có độ phân giải khác nhau. Tính năng nổi bật bao gồm đổ bóng, viền văn bản, đổ màu gradient, và hiệu ứng animation. TextMeshPro không chỉ tăng tính chuyên nghiệp mà còn tối ưu hóa hiệu suất, giúp duy trì tốc độ xử lý của trò chơi.

Timeline (1.6.5): Timeline là một công cụ trong Unity dùng để chỉnh sửa và quản lý hoạt ảnh theo dòng thời gian. Plugin này giúp nhà phát triển dễ dàng tạo ra các cảnh quay phim, sự kiện, hoặc hiệu ứng đồng. Bằng giao diện kéo thả và thao tác trực quan, người dùng có thể dựng Timeline để đồng bộ nhiều yếu tố như animation, âm thanh, và đổi tượng trong game.

Unity UI (1.0.0): Unity UI cung cấp các công cụ và thư viện để xây dựng giao diện người dùng (UI). Plugin này cho phép nhà phát triển tạo ra các thanh điều hướng, nút bấm, khung văn bản, và các thành phần UI tương tác khác. Unity UI có tính tùy biến cao và tự động tích hợp với nhiều thiết bị khác nhau, giúp trò chơi duy trì trải nghiệm tốt trên các nền tảng.

Visual Scripting (1.9.4): Visual Scripting giúp nhà phát triển tạo logic gameplay mà không cần viết mã. Thay vì dòng lệnh, plugin cung cấp giao diện kéo thả và kết nối các node logic. Visual Scripting đặc biệt phù hợp với những người mới làm quen Unity, nhất là trong việc xây dựng các prototype nhanh.

Feature 2D (2.0.1): Feature 2D cung cấp các công cụ chuyên biệt cho việc phát triển game 2D. Tính năng nổi bật bao gồm hệ thống tilemap, sprite động, và animation frame-by-frame. Plugin này giúp giảm thời gian thiết kế và tăng độ chính xác trong việc quản lý tài nguyên 2D.

IDE Plugins: Dự án bao gồm các plugin IDE như Rider, Visual Studio, và VSCode, cho phép nhà phát triển tích hợp trực tiếp các IDE và Unity. Các plugin này cung cấp tính năng gợ lỗi, gợi ý code, và quản lý dự án, tăng độ hiệu quả trong lắp trình.

Input System (1.7.0): Input System quản lý việc tương tác người chơi, hỗ trợ nhiều thiết bị như bàn phím, chuột, tay cầm, và cả màn hình cảm ứng. Hệ thống linh hoạt này giúp nhà phát triển tích hợp dễ dàng và tối đa tính tương thích trên nhiều nền tảng.

Android Logcat (1.4.4): Android Logcat cung cấp công cụ gỡ lỗi cho Android trực tiếp từ Unity. Plugin này giúp hiển thị các log runtime như các vấn đề hiệu suất, đọc logcat, và xử lý sự cố trong khi test game trên thiết bị thật.

### ***Công cụ thiết kế đồ họa và âm thanh***

# **CHƯƠNG 3: XÂY DỰNG ỨNG DỤNG**

1. **Thiết kế và phát triển**
2. ***Quy trình thiết kế (game design document, wireframe)***

Quy trình thiết kế của dự án "Fun\_Trap" bắt đầu với việc xây dựng Game Design Document (GDD), trong đó xác định rõ ý tưởng, lối chơi, nhân vật, cơ chế hoạt động và các yêu cầu kỹ thuật. GDD giúp đảm bảo rằng toàn bộ nhóm phát triển có cùng tầm nhìn về sản phẩm, đồng thời đóng vai trò như một tài liệu tham khảo trong suốt quá trình phát triển. Trong giai đoạn đầu, nhóm phát triển tập trung vào việc xác định cốt truyện, phong cách đồ họa, cơ chế điều khiển và các tính năng chính của trò chơi. Các tài liệu này không chỉ giúp định hình sản phẩm mà còn giúp giảm thiểu rủi ro khi phát triển, tránh việc thay đổi ý tưởng đột ngột làm ảnh hưởng đến tiến độ.

Sau khi hoàn thiện GDD, nhóm phát triển tiến hành thiết kế wireframe để minh họa giao diện và bố cục tổng thể của trò chơi. Wireframe là một bước quan trọng trong việc hình dung cách người chơi tương tác với trò chơi, từ màn hình chính, menu, đến các giao diện điều khiển nhân vật. Trong dự án "Fun\_Trap", wireframe được tạo bằng các công cụ như Figma hoặc Adobe XD, giúp nhóm phát triển có cái nhìn trực quan về cách sắp xếp các yếu tố trên màn hình. Quá trình này giúp đảm bảo tính logic, tiện dụng và tối ưu hóa trải nghiệm người dùng trước khi chuyển sang giai đoạn lập trình.

Giai đoạn tiếp theo là phát triển nguyên mẫu (prototype), trong đó nhóm phát triển tạo ra một phiên bản thử nghiệm của trò chơi với các tính năng cơ bản. Mục tiêu của nguyên mẫu là kiểm tra xem các cơ chế chơi có hoạt động đúng như thiết kế hay không, đồng thời thu thập phản hồi sớm để có thể điều chỉnh trước khi hoàn thiện. Trong "Fun\_Trap", nguyên mẫu bao gồm các yếu tố quan trọng như cơ chế di chuyển, vật lý trong game, và các hành vi của đối tượng trong môi trường ảo. Nhờ đó, nhóm phát triển có thể phát hiện lỗi, điều chỉnh hiệu suất và cải thiện trải nghiệm chơi một cách hiệu quả.

Sau khi nguyên mẫu được thử nghiệm và điều chỉnh, nhóm phát triển tiến hành thiết kế chi tiết các yếu tố trong trò chơi. Bao gồm việc tạo mô hình 3D hoặc sprite 2D, thiết kế hiệu ứng âm thanh, hình ảnh động và xây dựng các cấp độ (level design). Việc này đòi hỏi sự phối hợp chặt chẽ giữa các nhà thiết kế đồ họa, lập trình viên và chuyên gia âm thanh để đảm bảo mọi yếu tố được tích hợp hài hòa. Trong "Fun\_Trap", các yếu tố như giao diện người dùng, hệ thống tương tác và cơ chế vật lý được hoàn thiện ở giai đoạn này để mang lại trải nghiệm tốt nhất cho người chơi.

Bước cuối cùng trong quy trình thiết kế là kiểm thử và tinh chỉnh trò chơi. Trò chơi được chạy thử trên nhiều thiết bị khác nhau để kiểm tra hiệu suất, tìm lỗi và cải thiện gameplay. Nhóm phát triển thu thập phản hồi từ người chơi thử nghiệm, sau đó điều chỉnh lại các yếu tố chưa hợp lý. Trong "Fun\_Trap", việc kiểm thử không chỉ tập trung vào hiệu suất mà còn đảm bảo rằng các thử thách trong trò chơi đủ thú vị và không gây khó chịu cho người chơi. Quy trình này tiếp tục diễn ra cho đến khi sản phẩm đạt được chất lượng tốt nhất trước khi phát hành.

1. ***Phân chia các giai đoạn phát triển***

Quá trình phát triển dự án "Fun\_Trap" được chia thành nhiều giai đoạn để đảm bảo tính khoa học và hiệu quả. Mỗi giai đoạn có mục tiêu cụ thể, giúp nhóm phát triển kiểm soát tiến độ và tối ưu hóa tài nguyên. Việc phân chia giai đoạn không chỉ giúp dễ dàng quản lý mà còn đảm bảo chất lượng của sản phẩm.

Giai đoạn 1: Lên ý tưởng và lập kế hoạch

Giai đoạn đầu tiên tập trung vào việc xác định mục tiêu và ý tưởng chính của trò chơi. Nhóm phát triển tiến hành nghiên cứu thị trường, xác định đối tượng người chơi và phân tích các sản phẩm tương tự. Từ đó, các yếu tố cốt lõi như phong cách đồ họa, lối chơi, cơ chế game và công nghệ sử dụng được xác định. Sau đó, nhóm phát triển lập kế hoạch chi tiết, bao gồm lịch trình phát triển, phân công nhiệm vụ và tài nguyên cần thiết.

Giai đoạn 2: Thiết kế nguyên mẫu và thử nghiệm cơ bản

Sau khi hoàn thành kế hoạch, nhóm phát triển tạo ra nguyên mẫu (prototype) để kiểm tra các ý tưởng ban đầu. Giai đoạn này bao gồm việc phát triển bản demo với các cơ chế chơi chính, giúp đánh giá tính khả thi của dự án. Các phản hồi từ nhóm thử nghiệm nội bộ được thu thập để điều chỉnh thiết kế trước khi bước vào giai đoạn phát triển chính thức.

Giai đoạn 3: Phát triển và xây dựng hệ thống

Ở giai đoạn này, nhóm lập trình bắt đầu xây dựng hệ thống trò chơi dựa trên các thiết kế đã được phê duyệt. Các tính năng quan trọng như điều khiển nhân vật, hệ thống vật lý, trí tuệ nhân tạo (AI) và giao diện người dùng được phát triển song song. Nhóm thiết kế đồ họa cũng tạo ra mô hình, hiệu ứng và môi trường trong game để đảm bảo tính thẩm mỹ và trải nghiệm hấp dẫn.

Giai đoạn 4: Kiểm thử và tối ưu hóa

Khi các tính năng chính đã được triển khai, trò chơi bước vào giai đoạn kiểm thử. Các lỗi kỹ thuật, vấn đề hiệu suất và trải nghiệm người chơi được đánh giá chi tiết. Nhóm phát triển thực hiện các cải tiến để tối ưu hóa tốc độ xử lý, giảm thiểu lỗi và nâng cao chất lượng tổng thể. Quá trình này có thể kéo dài do yêu cầu kiểm tra kỹ lưỡng trước khi phát hành.

Giai đoạn 5: Ra mắt thử nghiệm và thu thập phản hồi

Trước khi phát hành chính thức, trò chơi được ra mắt thử nghiệm (beta test) với một nhóm người chơi giới hạn. Phản hồi từ người chơi thực tế giúp nhóm phát triển xác định những điểm cần cải thiện, từ lối chơi đến hiệu suất và giao diện. Các điều chỉnh cuối cùng được thực hiện để đảm bảo trò chơi đạt chất lượng cao nhất trước khi phát hành rộng rãi.

Giai đoạn 6: Phát hành và bảo trì

Sau khi hoàn tất các giai đoạn trên, trò chơi chính thức được phát hành. Nhóm phát triển tiếp tục theo dõi hiệu suất, thu thập phản hồi từ người dùng và phát hành các bản cập nhật để sửa lỗi hoặc bổ sung nội dung mới. Việc bảo trì và cập nhật liên tục giúp trò chơi duy trì sức hút và đáp ứng nhu cầu của người chơi trong thời gian dài.

1. ***Kiểm thử và đánh giá các phiên bản***

Giai đoạn 1: Kiểm thử nội bộ ban đầu:

Do nguồn nhân lực hạn chế, quá trình kiểm thử ban đầu được thực hiện bởi chính người phát triển và một số bạn bè thân thiết. Mục tiêu chính là kiểm tra các lỗi cơ bản trong cơ chế trò chơi, điều khiển nhân vật và hiệu suất hoạt động trên các thiết bị khác nhau. Các phản hồi từ nhóm kiểm thử nhỏ này được ghi nhận để điều chỉnh và sửa lỗi kịp thời.

Giai đoạn 2: Kiểm thử mở rộng trong nhóm nhỏ:

Sau khi sửa các lỗi chính từ lần kiểm thử đầu tiên, nhóm phát triển tiếp tục thử nghiệm trò chơi trên các thiết bị khác nhau trong nhóm kiểm thử nhỏ. Các yếu tố như độ mượt của chuyển động, tính ổn định của trò chơi và cảm giác khi trải nghiệm được đánh giá kỹ càng. Những vấn đề được ghi nhận sẽ giúp tối ưu hóa hiệu suất và điều chỉnh cơ chế chơi phù hợp hơn.

Giai đoạn 3: Thử nghiệm trên nhóm người chơi hạn chế:

Với sự hỗ trợ từ một số bạn bè, trò chơi được kiểm thử bởi nhóm người chơi có trải nghiệm đa dạng. Mỗi người chơi sẽ được yêu cầu thử nghiệm một khoảng thời gian nhất định và ghi nhận lại những cảm nhận, khó khăn gặp phải cũng như các đề xuất cải thiện. Dữ liệu từ thử nghiệm này rất quan trọng vì giúp xác định những vấn đề phát sinh trong môi trường thực tế.

Giai đoạn 4: Hoàn thiện và kiểm thử lần cuối:

Sau khi có phản hồi từ nhóm kiểm thử nhỏ, nhóm phát triển tiến hành sửa lỗi và tinh chỉnh trải nghiệm tổng thể. Những vấn đề như lỗi đồ họa, mất cân bằng trong lối chơi hoặc hiệu suất kém sẽ được khắc phục. Quá trình kiểm thử lần cuối này giúp đảm bảo trò chơi đạt chất lượng tốt nhất trong điều kiện nguồn lực kiểm thử hạn chế.

Giai đoạn 5: Phát hành thử nghiệm và thu thập phản hồi:

Do không có đội ngũ kiểm thử chuyên nghiệp, trò chơi sẽ được phát hành thử nghiệm (soft launch) đến một số lượng người chơi nhỏ. Người chơi sẽ được khuyến khích phản hồi về những trải nghiệm của họ thông qua khảo sát hoặc trò chuyện trực tiếp. Việc thu thập phản hồi từ người dùng thực tế giúp tinh chỉnh các tính năng quan trọng trước khi ra mắt chính thức.

Giai đoạn 6: Phát hành chính thức và cập nhật liên tục:

Sau khi hoàn tất kiểm thử và sửa lỗi, trò chơi sẽ được phát hành rộng rãi. Do không có đội ngũ hỗ trợ sau phát hành, việc bảo trì và cập nhật sẽ được thực hiện dựa trên phản hồi từ cộng đồng người chơi. Các bản cập nhật định kỳ sẽ giúp trò chơi duy trì chất lượng, sửa lỗi phát sinh và có thể bổ sung thêm nội dung mới nhằm giữ chân người chơi.

1. **Cấu trúc và chức năng của trò chơi**
2. ***Cấu trúc tổng quan (menu chính, màn chơi, giao diện kết thúc)***

* Tổng quan về cấu trúc trò chơi:

Trò chơi "Fun\_Trap" được thiết kế với một cấu trúc rõ ràng và trực quan, bao gồm ba phần chính: Menu chính, Màn chơi và Giao diện kết thúc. Mỗi phần được xây dựng nhằm mang lại trải nghiệm tốt nhất cho người chơi, đảm bảo tính mạch lạc và dễ sử dụng. Menu chính cung cấp các tùy chọn điều hướng quan trọng, màn chơi là trung tâm của trải nghiệm tương tác, và giao diện kết thúc giúp tổng kết kết quả chơi. Các thành phần này được thiết kế thống nhất để tạo nên một trải nghiệm liền mạch và dễ hiểu.

[Hình ảnh 1: Sơ đồ cấu trúc tổng quan của trò chơi, thể hiện ba phần chính và cách chúng liên kết với nhau]

* Menu chính:

Menu chính là điểm bắt đầu của trò chơi, nơi người chơi có thể chọn các tùy chọn như Bắt đầu trò chơi, Tùy chỉnh, Hướng dẫn và Thoát. Thiết kế menu chú trọng vào tính đơn giản nhưng vẫn đảm bảo tính thẩm mỹ, sử dụng các nút bấm rõ ràng và dễ nhận biết. Ngoài ra, phần nền động hoặc hiệu ứng nhỏ được thêm vào để tăng tính sinh động.

Người chơi có thể truy cập phần Hướng dẫn để hiểu cơ chế trò chơi trước khi bắt đầu. Nếu muốn tùy chỉnh các thiết lập như âm thanh, độ khó hoặc điều khiển, họ có thể vào mục Tùy chỉnh. Các nút điều hướng được bố trí hợp lý nhằm giúp trải nghiệm sử dụng menu trở nên mượt mà và trực quan.

[Hình ảnh 2: Giao diện Menu chính, hiển thị các nút tùy chọn và thiết kế tổng thể]

* Cấu trúc màn chơi:

Màn chơi là phần quan trọng nhất của trò chơi, nơi người chơi tham gia vào thử thách chính. Trong Fun\_Trap, màn chơi được thiết kế với cơ chế di chuyển nhân vật, tránh bẫy và thu thập vật phẩm. Mỗi màn có độ khó khác nhau, yêu cầu người chơi phải phản xạ nhanh và có chiến lược phù hợp.

Các yếu tố trong màn chơi bao gồm nhân vật chính, vật phẩm, bẫy và môi trường xung quanh. Thiết kế giao diện tối ưu nhằm giúp người chơi dễ dàng quan sát và phản ứng kịp thời. Ngoài ra, hệ thống HUD (Head-Up Display) cung cấp các thông tin như thời gian, điểm số và trạng thái nhân vật.

[Hình ảnh 3: Minh họa giao diện trong một màn chơi, bao gồm nhân vật chính, vật phẩm và bẫy]

* Hệ thống HUD trong màn chơi:

HUD là một phần quan trọng trong giao diện màn chơi, giúp người chơi theo dõi các thông số cần thiết mà không làm gián đoạn trải nghiệm. Hệ thống này hiển thị thanh máu, số điểm, vật phẩm thu thập và thời gian còn lại. Các thành phần này được sắp xếp hợp lý trên màn hình để không che khuất tầm nhìn quan trọng.

Bên cạnh đó, các hiệu ứng hình ảnh và âm thanh cũng được tích hợp để tăng cường trải nghiệm. Ví dụ, khi nhân vật bị trúng bẫy, thanh máu sẽ giảm và có hiệu ứng nhấp nháy. Nếu người chơi thu thập vật phẩm đặc biệt, một hiệu ứng ánh sáng nhỏ sẽ xuất hiện để thông báo.

[Hình ảnh 4: Minh họa giao diện HUD trong màn chơi, hiển thị thanh máu, điểm số và thời gian]

* Giao diện kết thúc:

Sau khi hoàn thành một màn chơi hoặc khi nhân vật thất bại, giao diện kết thúc sẽ xuất hiện để hiển thị kết quả. Giao diện này bao gồm điểm số cuối cùng, xếp hạng, và các tùy chọn tiếp theo như chơi lại, trở về menu chính hoặc thoát.

Để tạo sự hấp dẫn, hiệu ứng chuyển cảnh mượt mà được áp dụng khi chuyển từ màn chơi sang giao diện kết thúc. Nếu người chơi đạt thành tích cao, giao diện sẽ hiển thị hiệu ứng đặc biệt nhằm khuyến khích họ tiếp tục thử thách.

[Hình ảnh 5: Giao diện kết thúc màn chơi, hiển thị điểm số và các tùy chọn]

* Liên kết giữa các thành phần:

Các phần của trò chơi được liên kết mạch lạc để tạo nên trải nghiệm thống nhất. Sau khi chọn Bắt đầu trò chơi từ menu chính, người chơi sẽ chuyển sang màn chơi. Khi màn chơi kết thúc, giao diện điểm số sẽ xuất hiện, cho phép người chơi quyết định tiếp tục hoặc quay về menu chính.

Việc liên kết mượt mà giữa các giao diện giúp người chơi cảm thấy thoải mái và không bị gián đoạn trong trải nghiệm. Các hiệu ứng chuyển đổi cũng đóng vai trò quan trọng trong việc duy trì cảm giác liền mạch.

[Hình ảnh 6: Biểu đồ mô tả sự liên kết giữa Menu chính, Màn chơi và Giao diện kết thúc]

1. ***Các chức năng chính (điều khiển, chiến đấu, vượt chướng ngại vật)***

* Tổng quan về các chức năng chính:

Trò chơi "Fun\_Trap" được xây dựng với nhiều chức năng quan trọng, giúp tạo nên một trải nghiệm phong phú và hấp dẫn cho người chơi. Ba chức năng chính bao gồm hệ thống điều khiển nhân vật, cơ chế cảm biến cạm bẫy, và vượt chướng ngại vật. Mỗi chức năng này đều đóng vai trò quan trọng trong việc định hình phong cách chơi và thách thức mà người chơi cần vượt qua. Việc thiết kế hệ thống điều khiển tối ưu giúp người chơi dễ dàng thao tác, trong khi cơ chế chiến đấu và vượt chướng ngại vật mang đến thử thách đầy kịch tính.

[Hình ảnh 1: Sơ đồ tổng quan về ba chức năng chính trong trò chơi]

* Hệ thống điều khiển nhân vật:

Hệ thống điều khiển trong "Fun\_Trap" được thiết kế để đơn giản nhưng vẫn mang lại độ chính xác cao. Người chơi có thể sử dụng các phím di chuyển để điều khiển nhân vật tiến lên, lùi lại, nhảy hoặc cúi thấp để né tránh chướng ngại vật. Nếu chơi trên nền tảng di động, trò chơi sẽ hỗ trợ các phím ảo trên màn hình để tối ưu hóa thao tác.

Ngoài ra, hệ thống điều khiển cũng phản hồi nhanh chóng để đảm bảo nhân vật di chuyển mượt mà và chính xác. Điều này đặc biệt quan trọng trong các màn chơi yêu cầu tốc độ cao hoặc phản xạ nhanh. Các hiệu ứng hình ảnh như rung nhẹ hoặc thay đổi màu sắc nhân vật sẽ được tích hợp để thông báo cho người chơi khi họ thực hiện hành động thành công.

[Hình ảnh 2: Giao diện điều khiển nhân vật, bao gồm phím di chuyển và các thao tác khác]

* Cơ chế cảm biến bẫy

Cảm biến bẫy là một yếu tố quan trọng trong trò chơi, giúp người chơi có thể phát hiện các nguy hiểm tiềm ẩn trên đường đi. Hệ thống này hoạt động dựa trên các tín hiệu trực quan như ánh sáng mờ, tín hiệu âm thanh, hoặc hiệu ứng môi trường. Người chơi cần quan sát kỹ các dấu hiệu bất thường để tránh bẫy hoặc tìm cách vô hiệu hóa chúng.

Thiết kế cơ chế cảm biến bẫy không chỉ giúp tăng cường độ khó mà còn tạo chiều sâu cho lối chơi. Một số bẫy có thể được phát hiện từ xa, trong khi những bẫy khác chỉ xuất hiện khi nhân vật đến gần. Người chơi có thể sử dụng các công cụ hỗ trợ như đèn pin, cảm biến xung điện, hoặc quét môi trường để xác định vị trí của bẫy.

[Hình ảnh 4: Mô tả một phân đoạn nhân vật sử dụng cảm biến để phát hiện bẫy]Cơ chế vượt chướng ngại vật:

Vượt chướng ngại vật là một trong những yếu tố quan trọng tạo nên thử thách của trò chơi. Các chướng ngại vật có thể bao gồm bẫy gai, vực sâu, cửa sập, hoặc vật thể di chuyển. Người chơi cần sử dụng khả năng nhảy, trượt hoặc né để vượt qua các thử thách này một cách an toàn.

Thiết kế của cơ chế vượt chướng ngại vật đòi hỏi người chơi phải quan sát kỹ lưỡng và phản xạ nhanh. Một số màn chơi yêu cầu kết hợp nhiều thao tác cùng lúc để tránh bẫy và tiến lên phía trước. Các hiệu ứng âm thanh và hình ảnh sẽ giúp cảnh báo người chơi về những nguy hiểm sắp tới, tăng cường tính hồi hộp trong trò chơi.

[Hình ảnh 4: Mô tả một phân đoạn nhân vật vượt chướng ngại vật, thể hiện các loại bẫy khác nhau]

* Tương tác giữa ba chức năng chính:

Ba chức năng chính trong trò chơi không hoạt động riêng lẻ mà có sự kết nối chặt chẽ với nhau. Ví dụ, trong quá trình di chuyển, người chơi cần linh hoạt để né tránh các cạm bãy, chạy và nhảy qua chướng ngại vật trong khi tấn công. Điều này tạo ra một lối chơi năng động và không ngừng thay đổi.

Ngoài ra, một số màn chơi sẽ được thiết kế để kiểm tra khả năng kết hợp giữa các chức năng này. Người chơi có thể gặp tình huống phải chiến đấu với kẻ thù trong khi né tránh bẫy.

[Hình ảnh 5: Minh họa một tình huống thực tế trong game, nơi người chơi vừa chiến đấu vừa né tránh chướng ngại vật]

* Tối ưu hóa trải nghiệm người chơi:

Trong quá trình phát triển "Fun\_Trap", việc tối ưu hóa trải nghiệm người chơi là một yếu tố quan trọng. Hệ thống điều khiển được tinh chỉnh để đảm bảo phản hồi nhanh và chính xác. Cơ chế chiến đấu được thiết kế sao cho vừa đơn giản, dễ tiếp cận nhưng vẫn giữ được sự thử thách. Các chướng ngại vật cũng được bố trí hợp lý để tăng dần độ khó mà không gây cảm giác quá sức.

Bên cạnh đó, việc thiết kế hiệu ứng hình ảnh, âm thanh và phản hồi trực quan giúp người chơi dễ dàng nắm bắt cơ chế và có trải nghiệm nhập vai tốt hơn. Các tính năng hỗ trợ như hướng dẫn ban đầu hoặc chế độ luyện tập cũng có thể được tích hợp để giúp người chơi làm quen với trò chơi.

[Hình ảnh 6: Giao diện hướng dẫn người chơi về các chức năng chính]

1. **Tích hợp và tối ưu hóa**
2. ***Tích hợp âm thanh***
3. ***Tối ưu hóa mã nguồn***

* Tối ưu hóa cấu trúc thừa kế

Cấu trúc thừa kế trong hệ thống bẫy được tổ chức hiệu quả thông qua lớp cơ sở TrapBase. Lớp này cung cấp các thuộc tính và phương thức cơ bản như isActivated, duration, và isPermanent, cho phép các lớp con kế thừa và mở rộng chức năng. Việc sử dụng từ khóa virtual và override cho phép các lớp con tùy chỉnh hành vi theo nhu cầu cụ thể. Đặc biệt, phương thức ResetTrap() trong TrapBase đã được cải tiến để lưu trữ và khôi phục trạng thái ban đầu của các thành phần như Transform, SpriteRenderer và Rigidbody2D, giúp việc reset bẫy trở nên đồng nhất và dễ dàng hơn.

[Hình 3.3.1: Cấu trúc thừa kế của hệ thống bẫy]

* Tối ưu hóa quản lý tài nguyên

Các script đã được tối ưu để quản lý tài nguyên hiệu quả thông qua việc sử dụng các phương thức vòng đời của Unity như Awake(), Start(), và OnDisable(). Trong TrapControl, việc reference đến targetTrap được thay đổi từ tìm kiếm theo tag sang tham chiếu trực tiếp qua Inspector, giảm thiểu việc tìm kiếm GameObject trong runtime. Các component như SpriteRenderer và Rigidbody2D được cache trong biến private để tránh việc gọi GetComponent() liên tục, đặc biệt trong các phương thức Update().

[Hình 3.3.2: So sánh hiệu suất trước và sau khi tối ưu]

* Tối ưu hóa xử lý animation

Hệ thống animation được tối ưu thông qua việc sử dụng Coroutine thay vì Update() để xử lý các hiệu ứng như scale và di chuyển. Trong Trap03 và Trap04, các animation di chuyển và scale được thực hiện qua IEnumerator, cho phép kiểm soát chính xác thời gian và giảm tải cho Update loop. DeathAnimation sử dụng kỹ thuật tương tự để xử lý animation khi player chết, với việc tắt physics và các component không cần thiết để tối ưu hiệu suất.

[Hình 3.3.3: Flowchart quy trình xử lý animation]

* Tối ưu hóa logic điều khiển

Logic điều khiển trong TrapControl đã được cải tiến để xử lý hiệu quả hơn việc kích hoạt bẫy. Thay vì sử dụng tag để tìm kiếm trap, script now sử dụng tham chiếu trực tiếp thông qua Inspector, giảm thiểu overhead khi runtime. Việc kiểm tra va chạm với player được tối ưu bằng cách sử dụng phương pháp AABB (Axis-Aligned Bounding Box) thay vì Physics2D.OverlapArea, giúp giảm chi phí tính toán physics.

[Hình 3.3.4: Diagram logic điều khiển bẫy]

* Tối ưu hóa xử lý physics

Hệ thống physics được tối ưu thông qua việc sử dụng hợp lý Rigidbody2D và các constraint. Trong Trap03, việc sử dụng RigidbodyConstraints2D.FreezeRotation và đặt gravityScale = 0 giúp kiểm soát chính xác chuyển động của bẫy. Player class tối ưu việc xử lý va chạm bằng cách tắt các collider không cần thiết khi player chết, giảm thiểu việc tính toán physics không cần thiết.

[Hình 3.3.5: So sánh hiệu suất physics trước và sau khi tối ưu]

* Tối ưu hóa quản lý trạng thái

Việc quản lý trạng thái trong các script được tối ưu thông qua việc sử dụng các biến flag và enum thay vì nhiều biến boolean riêng lẻ. Trong Player class, các trạng thái như big, dead, và starpower được quản lý hiệu quả thông qua properties, cho phép kiểm soát truy cập và thay đổi trạng thái. TrapBase sử dụng các biến protected để cho phép các lớp con truy cập và điều chỉnh trạng thái một cách an toàn.

[Hình 3.3.6: State diagram của hệ thống]

1. ***Đóng gói và triển khai (WebGL, Windows và Android)***

* Chuẩn bị project cho việc build

Trước khi tiến hành build game cho các nền tảng, cần thực hiện một số bước chuẩn bị quan trọng. Trong Unity Editor, truy cập Project Settings để cấu hình các thông số cơ bản như Company Name, Product Name và Version. Đối với game platform 2D này, cần đảm bảo các sprite đã được nén với định dạng phù hợp: RGB Compressed ETC2 cho Android, RGB Compressed DXT1/DXT5 cho Windows và WebGL. Các script như PlayerOutOfBounds.cs cần được điều chỉnh để xử lý đúng việc reload scene trên các nền tảng khác nhau thông qua SceneManager.LoadScene().

[Hình 3.3.1: Cấu hình Project Settings cho việc build đa nền tảng]

* Tối ưu hóa cho WebGL

Khi build cho WebGL, cần chú ý đến kích thước file build và khả năng tương thích với các trình duyệt. Trong Player Settings, chọn WebGL template phù hợp và cấu hình Compression Format là Gzip. Các script quản lý Input như PlayerMovement cần được điều chỉnh để hỗ trợ cả keyboard và touch input, đảm bảo game có thể chơi được trên cả desktop và mobile browser. WebGL memory được giới hạn ở 2GB, do đó cần optimizing các texture và audio clip để giảm memory footprint.

[Hình 3.3.2: Cấu hình build settings cho WebGL platform]

* Tối ưu hóa cho Windows

Build game cho Windows yêu cầu cấu hình trong Player Settings để tạo file .exe và các file dependencies. Architecture được set là x86\_64 cho máy tính 64-bit hiện đại. Script system như TrapControl và Player cần được điều chỉnh để xử lý input từ keyboard và mouse một cách mượt mà. File build được đóng gói với các DLL cần thiết và thư mục Data chứa assets. Cấu hình Graphics API để sử dụng DirectX 11 hoặc DirectX 12 cho hiệu suất tốt nhất.

[Hình 3.3.3: Process đóng gói game cho Windows]

* Tối ưu hóa cho Android

Android build yêu cầu cài đặt Android SDK và NDK. Trong Player Settings, cấu hình Minimum API Level là Android 7.0 (API level 24) để đảm bảo tương thích với đa số thiết bị. Script xử lý input như PlayerMovement.cs cần được điều chỉnh để hỗ trợ touch input và các gesture phổ biến trên mobile. Graphics API được set là OpenGL ES 3.0 hoặc Vulkan để tối ưu hiệu suất. Các texture được nén với ETC2 format để giảm kích thước build.

[Hình 3.3.4: Quy trình build và triển khai cho Android]

* Quản lý resource và asset bundle

Việc quản lý resource được tối ưu thông qua sử dụng Asset Bundles cho cả ba nền tảng. Các sprite, audio và prefab được tổ chức thành các bundle riêng biệt, cho phép tải động khi cần thiết. Scripts như TrapBase và các derived class được compile thành managed code assemblies. Asset bundles được nén và cached để giảm thời gian tải và dung lượng lưu trữ. Resource versioning được implement để hỗ trợ việc update content sau này.

[Hình 3.3.5: Diagram quản lý resource và asset bundle]

* Testing và deployment

Quy trình testing và deployment được thiết lập cho từng platform. Với WebGL, game được test trên các browser phổ biến (Chrome, Firefox, Safari) và deployed lên web server hỗ trợ HTTPS. Windows build được test trên các phiên bản Windows 10/11 và distributed thông qua installer package. Android build được test trên các device với API level và screen size khác nhau, sau đó publish lên Google Play Store với proper signing và configuration.

[Hình 3.3.6: Flowchart quy trình testing và deployment]

# **CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

1. **Đánh giá kết quả**
2. ***Thành tựu đạt được***

* Hệ thống bẫy linh hoạt và mở rộng

Một trong những thành tựu quan trọng nhất là việc xây dựng thành công hệ thống bẫy với kiến trúc hướng đối tượng linh hoạt. Thông qua lớp cơ sở TrapBase, đã tạo ra được năm loại bẫy khác nhau với các hành vi độc đáo: hiển thị/ẩn sprite (Trap01, Trap02), di chuyển theo đường thẳng hoặc đường cong (Trap03), thay đổi kích thước (Trap04), và di chuyển theo vòng lặp (Trap05). Hệ thống này cho phép dễ dàng thêm mới các loại bẫy mà không cần sửa đổi code hiện có, tuân thủ nguyên tắc Open-Closed Principle trong thiết kế phần mềm.

[Hình 4.1.1: Tổng quan hệ thống bẫy đã phát triển]

* Hệ thống điều khiển người chơi tối ưu

Phát triển thành công hệ thống điều khiển người chơi với nhiều tính năng phức tạp như thay đổi kích thước (lớn/nhỏ), hiệu ứng sao bất tử, và animation chết. Hệ thống quản lý sprite thông minh cho phép chuyển đổi mượt mà giữa các trạng thái của nhân vật, với các animation được đồng bộ hoàn hảo. Việc tách biệt logic điều khiển (PlayerMovement) và hiển thị (PlayerSpriteRenderer) giúp code dễ bảo trì và mở rộng, đồng thời tối ưu hiệu suất rendering.

[Hình 4.1.2: Demo các trạng thái và animation của người chơi]

* Hệ thống quản lý va chạm thông minh

Đã xây dựng thành công hệ thống xử lý va chạm thông minh giữa người chơi và các bẫy. TrapControl sử dụng phương pháp AABB để kiểm tra va chạm hiệu quả, trong khi Player class xử lý các tình huống va chạm khác nhau một cách linh hoạt. Hệ thống này không chỉ đảm bảo tính chính xác trong việc phát hiện va chạm mà còn tối ưu về mặt hiệu suất, giúp game vận hành mượt mà ngay cả khi có nhiều bẫy hoạt động đồng thời.

[Hình 4.1.3: Biểu đồ hiệu suất hệ thống va chạm]

* Kiến trúc code clear và maintainable

Đạt được một kiến trúc code rõ ràng và dễ bảo trì thông qua việc áp dụng các nguyên tắc SOLID và design patterns phù hợp. Các script được tổ chức theo module rõ ràng: quản lý người chơi, hệ thống bẫy, xử lý animation và điều khiển camera. Việc sử dụng inheritance, interfaces và component-based architecture của Unity một cách hiệu quả giúp giảm thiểu code trùng lặp và tăng khả năng tái sử dụng code.

[Hình 4.1.4: Sơ đồ kiến trúc tổng thể của game]

* Tối ưu hóa hiệu suất trên đa nền tảng

Thành công trong việc tối ưu hóa game để chạy mượt mà trên nhiều nền tảng khác nhau (WebGL, Windows, Android). Các asset được quản lý và nén hiệu quả, physics system được tinh chỉnh để đảm bảo hiệu suất tốt trên mobile. Việc implement hệ thống input đa nền tảng cho phép game hoạt động tốt trên cả PC và thiết bị cảm ứng, trong khi vẫn duy trì được gameplay nhất quán.

[Hình 4.1.5: Biểu đồ so sánh hiệu suất trên các nền tảng]

* Khả năng mở rộng và phát triển

Game đã được xây dựng với khả năng mở rộng cao, cho phép dễ dàng thêm mới các tính năng và content trong tương lai. Hệ thống bẫy có thể được mở rộng với các loại bẫy mới, trong khi hệ thống người chơi có thể được bổ sung thêm các power-up và khả năng mới. Kiến trúc modular và việc sử dụng các design pattern phù hợp đảm bảo rằng việc thêm mới tính năng sẽ không ảnh hưởng đến code hiện có.

[Hình 4.1.6: Roadmap phát triển tương lai của game]

1. ***Hạn chế và khó khăn***

* Hạn chế trong hệ thống bẫy

Hệ thống bẫy hiện tại, mặc dù đã được thiết kế với tính kế thừa linh hoạt thông qua TrapBase, vẫn tồn tại một số hạn chế đáng kể. Việc quản lý trạng thái của bẫy phụ thuộc nhiều vào biến isActivated và isPermanent, làm giới hạn khả năng tạo ra các trạng thái phức tạp hơn. TrapControl sử dụng phương pháp kiểm tra va chạm dựa trên tọa độ đơn giản, có thể gây ra vấn đề với các hình dạng bẫy phức tạp hoặc các góc nhọn. Ngoài ra, việc thiếu cơ chế pooling cho các bẫy có thể gây ra vấn đề về hiệu suất khi số lượng bẫy trong scene tăng lên.

[Hình 4.1.1: Diagram mô tả hạn chế của hệ thống bẫy]

* Khó khăn trong xử lý physics

Việc triển khai physics cho các loại bẫy khác nhau gặp nhiều thách thức đáng kể. Trong Trap03 và Trap05, việc di chuyển các bẫy sử dụng Rigidbody2D đôi khi gây ra hiện tượng giật và không mượt mà, đặc biệt khi tốc độ di chuyển cao hoặc khi có nhiều va chạm xảy ra đồng thời. Hệ thống hiện tại chưa xử lý tốt các trường hợp đặc biệt như va chạm ở góc hoặc va chạm đồng thời với nhiều bẫy, có thể dẫn đến hành vi không mong muốn trong gameplay.

[Hình 4.1.2: Biểu đồ phân tích các vấn đề physics]

* Giới hạn trong hệ thống animation

Hệ thống animation hiện tại, đặc biệt trong PlayerSpriteRenderer và DeathAnimation, còn khá cơ bản và thiếu tính linh hoạt. Việc chuyển đổi giữa các trạng thái animation đôi khi không mượt mà, đặc biệt trong các tình huống chuyển đổi nhanh giữa các trạng thái như nhảy, trượt và chết. Thiếu cơ chế blend giữa các animation và hệ thống particle effect khiến các hiệu ứng visual không đạt được độ sinh động mong muốn.

[Hình 4.1.3: So sánh animation hiện tại và animation mong muốn]

* Hạn chế trong quản lý input

Hệ thống xử lý input hiện tại chưa thực sự tối ưu cho việc hỗ trợ đa nền tảng. PlayerMovement.cs phụ thuộc nhiều vào Input.GetKey và Input.GetKeyDown, làm cho việc chuyển đổi sang input touch trên mobile trở nên phức tạp. Thiếu một lớp trung gian để trừu tượng hóa việc xử lý input khiến việc thêm mới các control scheme hoặc tùy chỉnh keybinding trở nên khó khăn và dễ gây ra lỗi.

[Hình 4.1.4: Flowchart mô tả các vấn đề về input]

* Khó khăn trong optimize hiệu suất

Việc tối ưu hiệu suất trên các nền tảng khác nhau gặp nhiều thách thức. Trên WebGL, việc load và khởi tạo các bẫy có thể gây ra độ trễ đáng kể. Trên Android, việc sử dụng nhiều Rigidbody2D và các effect visual có thể gây ra vấn đề về battery drain và overheating. Hệ thống pooling chưa được triển khai đầy đủ, dẫn đến việc tạo và hủy các GameObject thường xuyên, ảnh hưởng đến hiệu suất của game.

[Hình 4.1.5: Graph hiệu suất trên các nền tảng]

* Giới hạn trong scalability

Kiến trúc hiện tại của game gặp khó khăn trong việc mở rộng quy mô. Việc thêm mới các loại bẫy phức tạp hoặc các mechanic gameplay mới có thể yêu cầu thay đổi đáng kể trong các class cơ sở như TrapBase hoặc Player. Hệ thống event và messaging chưa được triển khai đầy đủ, làm cho việc xử lý communication giữa các component trở nên phức tạp và khó maintain khi game phát triển lớn hơn.

[Hình 4.1.6: Sơ đồ phân tích khả năng mở rộng]

1. **Hướng phát triển trong tương lai**
2. ***Nâng cấp tính năng và đồ họa***
3. ***Mở rộng nội dung (thêm màn chơi, thêm chế độ chơi)***
4. ***Ứng dụng AI hoặc tính năng đa người chơi***

# **TÀI LIỆU THAM KHẢO**

|  |  |
| --- | --- |
| [1] | L. T. T. Mai, “Game đi cảnh (Platform game) là gì? Lịch sử và phân loại,” Công ty Cổ phần Đầu tư Thế Giới Di Động (MWG), 20 05 2024. [Trực tuyến]. Available: https://www.thegioididong.com/game-app/game-di-canh-platform-game-la-gi-lich-su-va-phan-loai-1359174#:~:text=1.-,%C4%90%E1%BB%8Bnh%20ngh%C4%A9a,%2C%20nh%E1%BA%A3y%20xa%2C%20%C4%91u%20d%C3%A2y.. [Đã truy cập 26 12 2024]. |
| [2] | Unity, “Introduction to rigid body physics,” Unity, 14 01 2025. [Trực tuyến]. Available: https://docs.unity3d.com/Manual/RigidbodiesOverview.html. [Đã truy cập 14 01 2025]. |
| [3] | G. Coppens, “Introduction to Physics in Unity,” Medium, 20 05 2021. [Trực tuyến]. Available: https://medium.com/nerd-for-tech/introduction-to-physics-in-unity-8b3f9f467167. [Đã truy cập 28 11 2024]. |
| [4] | Unity, “Unity Scripting API/Library,” Unity, 11 05 2021. [Trực tuyến]. Available: https://discussions.unity.com/t/unity-scripting-api-library/840354. [Đã truy cập 29 11 2024]. |
| [5] | B. Follington, “Event-Driven Game Development in Unity,” 10 09 2019. [Trực tuyến]. Available: https://dev.to/vivavolt/event-driven-game-development-in-unity-5hi2. [Đã truy cập 17 12 2024]. |
| [6] | T. Lực, “Xây dựng hệ thống Event trong Unity,” CTY TNHH TM & DV GINI, 26 06 2024. [Trực tuyến]. Available: https://giniwebseo.vn/xay-dung-he-thong-event-trong-unity/. [Đã truy cập 30 12 2024]. |
| [7] | Unity, “Animation System Overview,” 25 04 2018. [Trực tuyến]. Available: https://docs.unity3d.com/2020.1/Documentation/Manual/AnimationOverview.html. [Đã truy cập 23 11 2024]. |