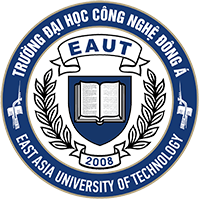
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**HỌC PHẦN: ĐỒ ÁN CHUYÊN NGÀNH CÔNG NGHỆ PHẦN MỀM**

**MÃ ĐỀ THI: ĐỀ 143**

**CHỦ ĐỀ 3: XÂY DỰNG ỨNG DỤNG TRÊN THIẾT BỊ DI ĐỘNG**

**TÊN ĐỀ TÀI: XÂY DỰNG GAME 2D FLATFORMER SỬ DỤNG UNITY**

**LỚP TÍN CHỈ: ĐACNPM.03.K13.01.LH.C04.1\_LT**

**Giảng viên hướng dẫn: ThS. Đỗ Thị Huyền**

**Danh sách sinh viên thực hiện: Nhóm 10**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20222109 | Ngô Anh Nguyên | DCCNTT13.10.11 |
| **3** | 20220136 | Nguyễn Thái Sơn | DCCNTT13.10.1 |

**Bắc Ninh - 2025**

**Mục lục**

[DANH MỤC HÌNH ẢNH 3](#_Toc17753)

[DANH MỤC BẢNG BIỂU VÀ SƠ ĐỒ 4](#_Toc4805)

[MỞ ĐẦU 5](#_Toc18844)

[CHƯƠNG 1: Giới thiệu chung 6](#_Toc17590)

[1.1. Mục tiêu và phạm vi 6](#_Toc15798)

[1.2. Bố cục báo cáo 7](#_Toc30779)

[Chương 2: KIẾN THỨC NỀN TẢNG 8](#_Toc27341)

[2.1. 2D Platformer 8](#_Toc13477)

[2.2. Unity Engine 8](#_Toc6824)

[2.2.1. Tổng quan về Unity Engine 8](#_Toc25964)

[2.2.2. Lý do lựa chọn và Ưu điểm của Unity 8](#_Toc30821)

[2.3. Firebase 9](#_Toc26782)

*[2.2.1. Tổng quan về Firebase](#_Toc21431)* [9](#_Toc21431)

[2.3.2. Firebase Authentication 9](#_Toc31121)

[2.3.3. Firebase Realtime Database 10](#_Toc17221)

[2.3.4. Lý do lựa chọn và Ưu điểm của Firebase 10](#_Toc8784)

[2.4. Kiến trúc code trong phát triển game 10](#_Toc18153)

[2.5. Kỹ thuật tối ưu đồ họa trong game 11](#_Toc29028)

[CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG 13](#_Toc20689)

[3.1. Biểu đồ phân cấp chức năng hệ thống 13](#_Toc24289)

[3.2. Biểu đồ usecase tổng quát 14](#_Toc5338)

[3.3. Biểu đồ Use case chi tiết 15](#_Toc25313)

[3.3.1. Sơ đồ Use case chức năng 15](#_Toc14241)

[3.3.2. Xác định Use case 17](#_Toc4309)

[3.4. Biểu đồ lớp 19](#_Toc783)

[3.5. Biểu đồ trình tự 22](#_Toc26190)

[3.6. Sơ đồ hoạt động 24](#_Toc13433)

[3.7. Phác thảo và mô tả giao diện 25](#_Toc12594)

[3.8. Cấu trúc cơ sở dữ liệu Firebase Realtime Database 25](#_Toc22664)

[CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH 27](#_Toc20601)

[4.1. Môi trường phát triển 27](#_Toc2940)

[4.2. Cấu trúc mã nguồn 27](#_Toc27617)

[4.3. Triển khai Gameplay 29](#_Toc26772)

[4.4. Triển khai Firebase Authentication 30](#_Toc25153)

[CHƯƠNG 5: KẾT LUẬN 32](#_Toc2460)

[5.1. Kết quả đạt được 32](#_Toc27480)

[5.2. Hạn chế 32](#_Toc16362)

[5.3. Hướng phát triển 33](#_Toc4439)

[TÀI LIỆU THAM KHẢO 35](#_Toc14933)

# DANH MỤC HÌNH ẢNH

[Hình 3. 1: Sơ đồ phân cấp chức năng 13](#_Toc9535)

[Hình 3. 2: Sơ đồ Use case tổng quát 14](#_Toc23254)

[Hình 3. 3: Sơ đồ Use case chức năng cốt lõi 15](#_Toc894)

[Hình 3. 4: Sơ đồ Use case Bắt đầu trận đấu trận đấu 16](#_Toc1195)

[Hình 3. 5: Sơ đồ Use case Tính năng xã hội 17](#_Toc18795)

[Hình 3. 6: Sơ đồ lớp Kiến trúc thiết kế sự kiện 19](#_Toc13633)

[Hình 3. 7: Sơ đồ lớp Hệ thống quản lý 19](#_Toc19111)

[Hình 3. 8: Sơ đồ lớp Hệ thống máu 20](#_Toc10798)

[Hình 3. 9: Biểu đồ trình tự Hoạt động chính của người chơi 21](#_Toc27936)

[Hình 3. 10: Biểu đồ trình tự Trận đấu 22](#_Toc32166)

[Hình 3. 12: Biểu đồ hoạt động Xác thực tài khoản 23](#_Toc1586)

[Hình 3. 13: Biểu đồ hoạt động Người dùng nhận input 23](#_Toc11174)

[Hình 3. 14: Biểu đồ hoạt động Người chơi bắt đầu game 24](#_Toc30124)

[Hình 3. 15: Biểu đồ hoạt động Ngời chơi va chạm kẻ địch và nhận sát thương 24](#_Toc16742)

[Hình 3. 16: Biểu đổ hoạt động Đạn của người chơi va chạm kẻ địch 25](#_Toc5639)

[Hình 3. 17: Biểu đồ hoạt động Người chơi hoàn thành màn chơi 25](#_Toc4396)

# DANH MỤC BẢNG BIỂU VÀ SƠ ĐỒ

[Bảng 3. 1 : Chi tiết Use case 17](#_Toc1910)

# MỞ ĐẦU

Trong kỷ nguyên số hóa, ngành công nghiệp game đang phát triển mạnh mẽ và trở thành một trong những lĩnh vực giải trí được ưa chuộng nhất. Game 2D Platformer, với lối chơi cổ điển nhưng vẫn hấp dẫn, luôn có một vị trí vững chắc trong lòng người chơi. Đồ án này tập trung vào việc phát triển một game 2D Platformer hoàn chỉnh sử dụng Unity Engine, một công cụ phát triển game mạnh mẽ và linh hoạt. Đặc biệt, đồ án còn tích hợp các dịch vụ backend như Firebase (Authentication và Realtime Database) để quản lý dữ liệu người dùng và bảng xếp hạng. Ngoài ra, việc áp dụng các kiến trúc code tối ưu và kỹ thuật đồ họa tiên tiến sẽ đảm bảo hiệu suất và khả năng mở rộng của game.

# CHƯƠNG 1: Giới thiệu chung

1. **Mục tiêu và phạm vi**

**Mục tiêu:**

* Xây dựng một game 2D Platformer với gameplay hấp dẫn và cơ chế điều khiển mượt mà.
* Triển khai hệ thống xác thực người dùng (login, register) bằng Firebase Authentication.
* Lưu trữ và quản lý dữ liệu game (ví dụ: điểm số, thành tích) theo thời gian thực bằng Firebase Realtime Database.
* Áp dụng các kiến trúc code tốt để nâng cao tính bảo trì, khả năng mở rộng và hiệu suất của mã nguồn.
* Sử dụng các kỹ thuật tối ưu hóa đồ họa để đảm bảo

**Phạm vi:**

* Game bao gồm một số cấp độ cơ bản với các yếu tố platformer như nhảy, di chuyển, thu thập vật phẩm và tránh chướng ngại vật.
* Hệ thống đăng nhập/đăng ký người dùng và lưu trữ dữ liệu cá nhân đơn giản.
* Tính năng multiplayer cơ bản: tạo phòng, tham gia phòng, đồng bộ hóa vị trí nhân vật, hoạt ảnh và một số tương tác đơn giản giữa người chơi.
* Tập trung vào tối ưu hóa hiệu suất trên nền tảng PC và di động (Android cơ bản).
* Không bao gồm các tính năng phức tạp như chat trong game, hệ thống bạn bè, giao dịch vật phẩm ảo, hay AI nâng cao cho kẻ thù.

1. **Bố cục báo cáo**

Báo cáo này được cấu trúc thành 6 chương chính:

**Chương 1: Giới thiệu** - Trình bày tổng quan về đồ án, mục tiêu, phạm vi và bố cục báo cáo.

**Chương 2: Cơ sở lý thuyết và Lựa chọn công cụ** - Cung cấp nền tảng kiến thức về các công nghệ và khái niệm được sử dụng trong đồ án, cùng với lý do lựa chọn và ưu điểm của từng công cụ.

**Chương 3: Phân tích và Thiết kế** - Chi tiết hóa các yêu cầu, thiết kế gameplay, kiến trúc hệ thống và kiến trúc code.

**Chương 4: Triển khai** - Mô tả quá trình hiện thực hóa các chức năng, tích hợp dịch vụ và áp dụng các kỹ thuật tối ưu.

**Chương 5: Kiểm thử và Đánh giá** - Trình bày chiến lược kiểm thử, kết quả và đánh giá hiệu suất của sản phẩm.

**Chương 6: Kết luận và Hướng phát triển** - Tổng kết những thành quả đạt được, chỉ ra các hạn chế và đề xuất hướng phát triển trong tương lai.

# Chương 2: KIẾN THỨC NỀN TẢNG

1. **2D Platformer**

Game 2D Platformer là một thể loại game mà người chơi điều khiển một nhân vật di chuyển qua các nền tảng (platforms), nhảy qua các khoảng trống, leo trèo và thường xuyên phải đối mặt với các chướng ngại vật hoặc kẻ thù. Đặc điểm chính của thể loại này là gameplay tập trung vào kỹ năng di chuyển, sự phối hợp tay mắt và tính toán thời gian chính xác. Một số yếu tố phổ biến bao gồm: thu thập vật phẩm, giải đố, chiến đấu và khám phá môi trường.

1. **Unity Engine**
2. **Tổng quan về Unity Engine**

Unity là một công cụ phát triển game và ứng dụng 2D/3D đa nền tảng mạnh mẽ, được phát triển bởi Unity Technologies. Với môi trường phát triển tích hợp (IDE) trực quan, Unity cho phép các nhà phát triển tạo ra các trải nghiệm tương tác với mã nguồn được viết bằng C#. Các tính năng nổi bật của Unity bao gồm:

* **Editor trực quan**: Kéo và thả đối tượng, xem trước cảnh game theo thời gian thực.
* **Hỗ trợ đa nền tảng**: Xuất bản game lên Windows, macOS, Linux, Android, iOS, WebGL, console, v.v.
* **Công cụ 2D và 3D mạnh mẽ**: Hỗ trợ Sprite Editor, Tilemap Editor, hệ thống vật lý 2D/3D (Physics2D/Physics), Animator cho hoạt ảnh, hệ thống UI Canvas.
* **Asset Store**: Nơi cung cấp tài nguyên, công cụ và tiện ích mở rộng.
* **Scripting API**: Sử dụng C# để viết logic game.

1. **Lý do lựa chọn và Ưu điểm của Unity**

Chúng tôi lựa chọn Unity Engine để phát triển đồ án này vì những lý do và ưu điểm nổi bật sau:

* **Phù hợp với Game 2D Platformer**: Unity cung cấp bộ công cụ 2D mạnh mẽ như Sprite Editor, Tilemap Editor, và hệ thống vật lý 2D chuyên biệt (Physics2D), giúp việc phát triển game platformer trở nên hiệu quả và chính xác. Khả năng xử lý collision detection và input handling linh hoạt là yếu tố then chốt cho gameplay mượt mà của thể loại này.
* **Môi trường phát triển tích hợp (IDE) thân thiện**: Unity Editor trực quan, cho phép kéo thả đối tượng, xem trước cảnh game theo thời gian thực, và điều chỉnh các thuộc tính dễ dàng thông qua Inspector. Điều này giúp đẩy nhanh tốc độ phát triển và thử nghiệm.
* **Kiến trúc Component-based**: Unity sử dụng kiến trúc Component-based, cho phép tách biệt các logic khác nhau thành các component độc lập (ví dụ: PlayerController, HealthSystem, EnemyAI). Điều này giúp mã nguồn dễ bảo trì, dễ mở rộng và tăng khả năng tái sử dụng.
* **Hỗ trợ đa nền tảng**: Unity cho phép phát triển game một lần và triển khai trên nhiều nền tảng khác nhau như PC (Windows, macOS, Linux) và di động (Android, iOS), tiết kiệm thời gian và công sức phát triển.
* **Cộng đồng lớn và tài liệu phong phú**: Unity có một cộng đồng nhà phát triển khổng lồ và nguồn tài liệu, hướng dẫn, khóa học dồi dào. Điều này rất hữu ích cho việc tìm kiếm giải pháp, học hỏi kinh nghiệm và khắc phục sự cố trong quá trình phát triển.
* **Tích hợp dễ dàng các công cụ và dịch vụ bên thứ ba**: Unity Asset Store cung cấp vô số tài nguyên và plugin, cho phép tích hợp dễ dàng các thư viện như Firebase SDK, giúp đơn giản hóa việc thêm các tính năng phức tạp như backend.

1. **Firebase**
2. ***Tổng quan về Firebase***

Firebase là một nền tảng phát triển ứng dụng di động và web toàn diện được Google cung cấp. Nó bao gồm một bộ các dịch vụ BaaS (Backend-as-a-Service) giúp các nhà phát triển xây dựng ứng dụng nhanh chóng mà không cần quản lý cơ sở hạ tầng backend phức tạp. Trong đồ án này, chúng ta sử dụng hai dịch vụ chính của Firebase: Firebase Authentication và Firebase Realtime Database.

1. **Firebase Authentication**

Firebase Authentication cung cấp các dịch vụ backend, SDK dễ sử dụng và thư viện UI có sẵn để xác thực người dùng vào ứng dụng của bạn. Nó hỗ trợ nhiều phương thức xác thực khác nhau như email/mật khẩu, tài khoản Google, Facebook, Twitter, v.v. Firebase Authentication giúp đơn giản hóa quá trình quản lý tài khoản người dùng, bảo mật thông tin và tích hợp liền mạch vào ứng dụng.

1. **Firebase Realtime Database**

Firebase Realtime Database là một cơ sở dữ liệu NoSQL được lưu trữ trên đám mây. Dữ liệu được lưu dưới dạng cây JSON và được đồng bộ hóa theo thời gian thực đến mọi client được kết nối. Điều này có nghĩa là khi dữ liệu thay đổi, tất cả các client đang lắng nghe dữ liệu đó sẽ nhận được cập nhật gần như ngay lập tức. Tính năng này rất hữu ích cho các ứng dụng yêu cầu đồng bộ hóa dữ liệu nhanh chóng như game đa người chơi, ứng dụng chat, hoặc bảng xếp hạng trực tiếp.

1. **Lý do lựa chọn và Ưu điểm của Firebase**

Firebase được chọn làm nền tảng backend cho đồ án này nhờ các ưu điểm sau:

* **Dễ dàng tích hợp và sử dụng**: Firebase cung cấp SDK Unity chuyên dụng, giúp việc tích hợp các dịch vụ Firebase vào dự án Unity trở nên cực kỳ đơn giản. Các API thân thiện với nhà phát triển, giúp tiết kiệm thời gian đáng kể so với việc tự xây dựng backend.
* **Đồng bộ hóa dữ liệu thời gian thực**: Firebase Realtime Database là giải pháp lý tưởng cho các tính năng yêu cầu cập nhật dữ liệu liên tục như bảng xếp hạng trực tiếp, trạng thái phòng chơi và có thể mở rộng cho tính năng chat trong game. Dữ liệu được đồng bộ hóa tức thì giữa tất cả các client được kết nối.
* **Xác thực người dùng mạnh mẽ**: Firebase Authentication cung cấp một giải pháp xác thực an toàn, đa dạng phương thức (email/mật khẩu, Google, v.v.) và dễ dàng quản lý. Điều này giúp đồ án tập trung vào logic game thay vì lo lắng về vấn đề bảo mật và quản lý tài khoản.
* **Khả năng mở rộng (Scalability)**: Firebase được thiết kế để mở rộng một cách tự động, có thể xử lý một lượng lớn người dùng và dữ liệu mà không yêu cầu cấu hình phức tạp từ phía nhà phát triển.
* **Chi phí hiệu quả (Cost-Effective)**: Firebase có gói miễn phí hào phóng và các gói trả phí linh hoạt, phù hợp với các dự án cá nhân hoặc nhóm nhỏ.
* **Tích hợp tốt với các dịch vụ Google Cloud**: Dễ dàng mở rộng sang các dịch vụ khác của Google Cloud nếu cần trong tương lai.

1. **Kiến trúc code trong phát triển game**

Việc áp dụng các kiến trúc code phù hợp là rất quan trọng để xây dựng một game có tính bảo trì cao, dễ mở rộng và dễ kiểm thử. Một số kiến trúc và mẫu thiết kế phổ biến bao gồm:

* **Singleton Pattern**: Đảm bảo chỉ có một thể hiện của một lớp và cung cấp một điểm truy cập toàn cục cho nó. Thường dùng cho các Manager (GameManager, AudioManager).
* **Observer Pattern / Event-driven Architecture**: Cho phép các đối tượng đăng ký lắng nghe các sự kiện và được thông báo khi sự kiện đó xảy ra, giúp giảm sự phụ thuộc giữa các thành phần.
* **State Machine Pattern**: Quản lý các trạng thái khác nhau của một đối tượng (ví dụ: nhân vật có các trạng thái Idle, Running, Jumping, Attacking) và các chuyển đổi giữa chúng.
* **Scriptable Objects**: Các asset dữ liệu có thể được tạo trong Editor và chia sẻ giữa nhiều GameObject hoặc kịch bản, giúp tách dữ liệu khỏi logic và dễ dàng chỉnh sửa.
* **Model-View-Controller (MVC) / Model-View-Presenter (MVP)**: Phân chia ứng dụng thành các lớp riêng biệt để quản lý dữ liệu (Model), hiển thị (View) và xử lý logic (Controller/Presenter). Mặc dù thường thấy trong phát triển ứng dụng UI, các nguyên tắc này có thể áp dụng trong game.

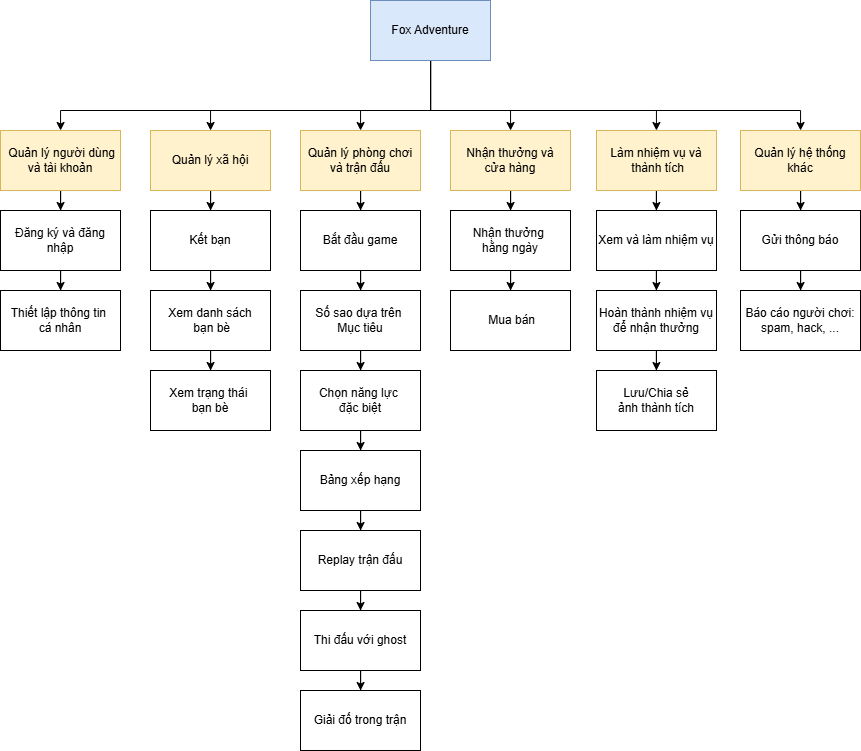
1. **Kỹ thuật tối ưu đồ họa trong game**

Tối ưu hóa đồ họa là quá trình giảm thiểu tài nguyên GPU và CPU cần thiết để render cảnh game mà vẫn duy trì chất lượng hình ảnh chấp nhận được. Các kỹ thuật phổ biến bao gồm:

* **Sprite Packing and Atlas**: Gộp nhiều sprite nhỏ vào một texture lớn (atlas) để giảm số lượng lệnh vẽ (draw calls), cải thiện hiệu suất rendering.
* **Texture Compression**: Nén texture để giảm kích thước bộ nhớ và băng thông GPU cần thiết.
* **Object Pooling**: Tái sử dụng các đối tượng game (đạn, hiệu ứng, kẻ thù) thay vì liên tục tạo và hủy chúng, giúp giảm gánh nặng garbage collection.
* **Occlusion Culling**: Không render các đối tượng bị che khuất bởi các đối tượng khác, giảm số lượng đa giác và lệnh vẽ.
* **Frustum Culling**: Không render các đối tượng nằm ngoài tầm nhìn của camera.
* **Static/Dynamic Batching**: Unity tự động nhóm các đối tượng có cùng vật liệu để render trong một draw call duy nhất. Static batching áp dụng cho các đối tượng tĩnh, dynamic batching cho các đối tượng di chuyển nhỏ.
* **Layer and Sorting Layers**: Quản lý thứ tự render của các đối tượng 2D và kiểm soát tương tác vật lý.
* **Shader Optimization**: Sử dụng shader đơn giản, hiệu quả hơn.
* **Level of Detail (LOD)**: Sử dụng các phiên bản đối tượng có độ chi tiết khác nhau tùy thuộc vào khoảng cách từ camera.

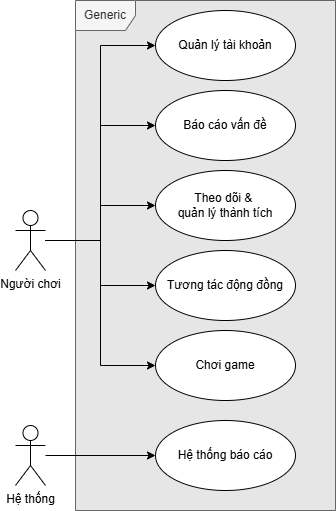
# CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

## Biểu đồ phân cấp chức năng hệ thống



Hình 3. 1: Sơ đồ phân cấp chức năng

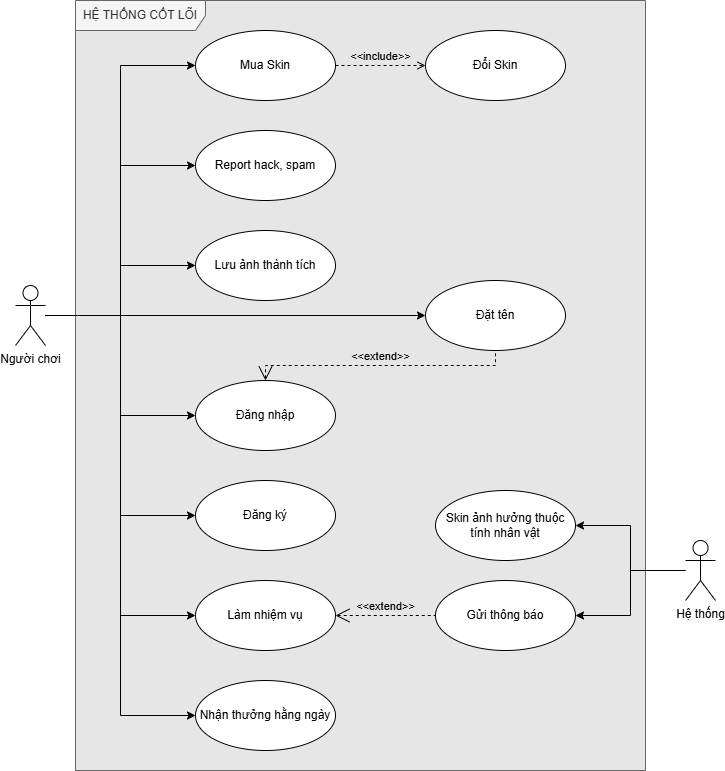
## Biểu đồ usecase tổng quát



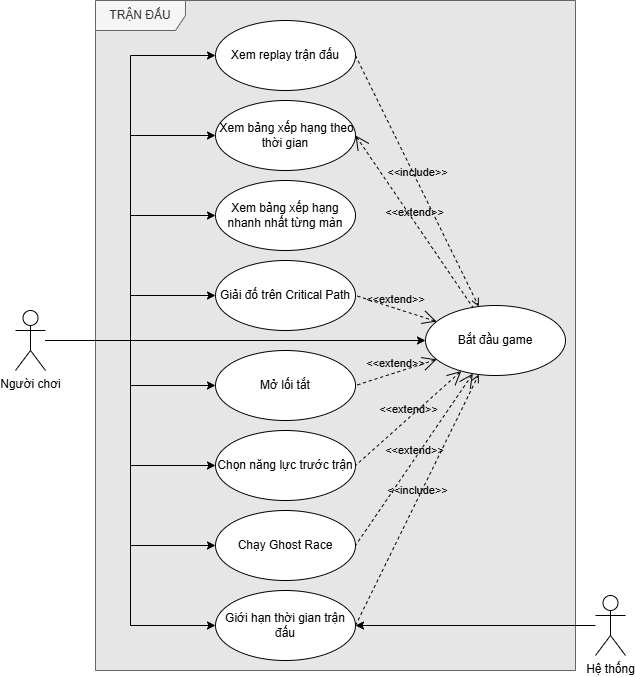
Hình 3. 2: Sơ đồ Use case tổng quát

## Biểu đồ Use case chi tiết

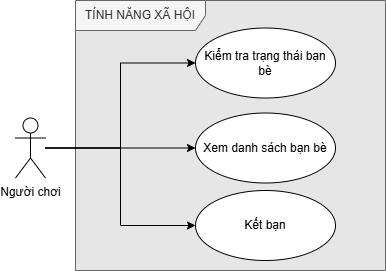
### Sơ đồ Use case chức năng



Hình 3. 3: Sơ đồ Use case chức năng cốt lõi



Hình 3. 4: Sơ đồ Use case Bắt đầu trận đấu trận đấu



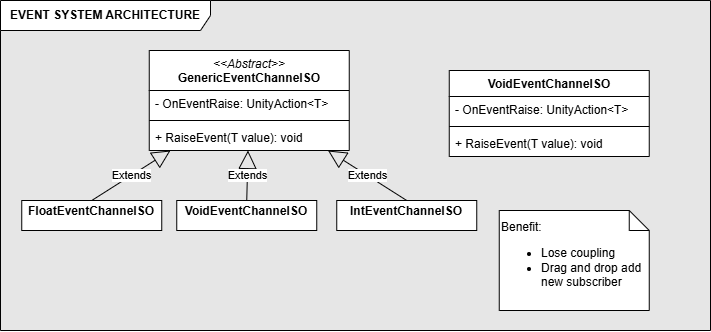
Hình 3. 5: Sơ đồ Use case Tính năng xã hội

### Xác định Use case

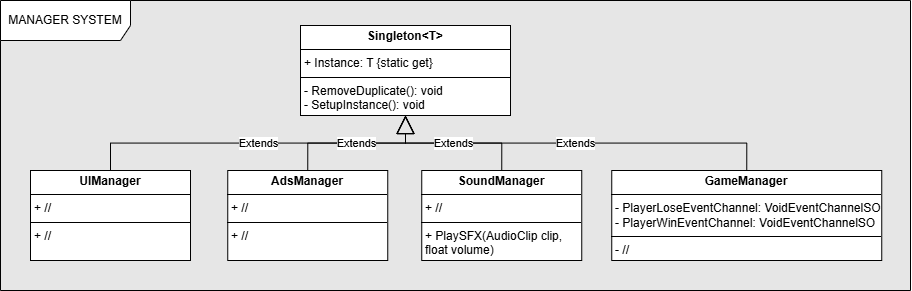
Bảng 3. 1 : Chi tiết Use case

|  |  |  |
| --- | --- | --- |
| **Actor Chính** | **Use Case** | **Mô tả** |
| Người chơi | Đăng ký | Người chơi tạo một tài khoản mới để truy cập hệ thống game. |
| Đăng nhập | Người chơi xác thực để truy cập tài khoản và các tính năng game. |
| Đổi tên | Người chơi thay đổi tên hiển thị của mình trong game. |
| Nhận thưởng hằng ngày | Người chơi nhận phần thưởng tiền ảo (Coin) một lần mỗi ngày. |
| Mua Skin | Người chơi sử dụng tiền tệ trong game để mua các Skin mới cho nhân vật. |
| Đổi Skin | Người chơi thay đổi Skin đang sử dụng cho nhân vật của mình. |
| Làm nhiệm vụ | Người chơi hoàn thành nhiệm vụ yêu cầu trong một trận đấu. |
| Report Spam Chat | Người chơi báo cáo các tin nhắn hoặc hành vi spam trong kênh chat, các trường hợp nghi ngờ gian lận liên quan đến tiền tệ. |
| Lưu và chia sẻ ảnh thành tích | Người chơi lưu lại ảnh màn hình thành tích hoặc chia sẻ khoảnh khắc đáng nhớ trong game. |
| Bắt đầu game | Người chơi (thường là chủ phòng) khởi động trận đấu sau khi đủ người hoặc điều kiện. |
| Xem Replay trận đấu | Người chơi xem lại một trận đấu đã diễn ra trước đó. |
| Xem Bảng xếp hạng theo thời gian | Người chơi xem bảng xếp hạng tổng quát dựa trên thời gian hoàn thành các màn chơi. |
| Xem Bảng xếp hạng nhanh nhất từng màn | Người chơi xem bảng xếp hạng chi tiết về thời gian hoàn thành nhanh nhất cho từng màn chơi cụ thể. |
| Chọn năng lực trước trận | Người chơi lựa chọn một năng lực đặc biệt để sử dụng trong trận đấu sắp tới. |
| Chạy Ghost Race | Người chơi tham gia một trận đấu mà đối thủ là "bóng ma" (ghi lại từ lượt chơi tốt nhất trước đó). |
| Giải đố trên Critical Path | Người chơi giải các câu đố xuất hiện trên đường đi chính của màn chơi để tiến xa hơn. |
| Mở lối tắt | Người chơi phát hiện và mở khóa các lối đi tắt trong màn chơi để rút ngắn thời gian. |
| Kết bạn | Người chơi gửi hoặc chấp nhận lời mời kết bạn với người chơi khác. |
| Xem danh sách bạn bè | Người chơi xem danh sách các bạn bè của mình trong game. |
| Kiểm tra trạng thái bạn bè | Người chơi xem trạng thái online/offline, trong trận/phòng chờ của bạn bè. |
| Hệ thống | Gửi thông báo | Hệ thống tự động gửi các thông báo đến người chơi (ví dụ: hoàn thành nhiệm vụ, nhận thưởng, sự kiện mới). |
| Skin ảnh hưởng thuộc tính nhân vật | Hệ thống áp dụng các thuộc tính đặc biệt lên nhân vật khi một Skin cụ thể được trang bị. |
| Giới hạn thời gian trận đấu | Hệ thống thiết lập và quản lý thời gian tối đa cho mỗi trận đấu. |

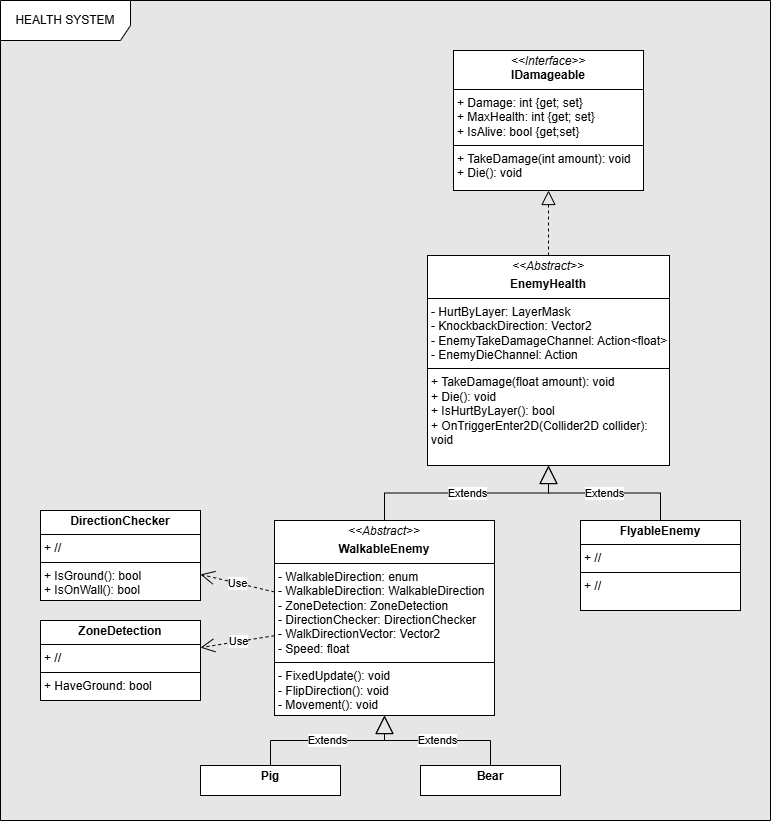
## Biểu đồ lớp



Hình 3. 6: Sơ đồ lớp Kiến trúc thiết kế sự kiện

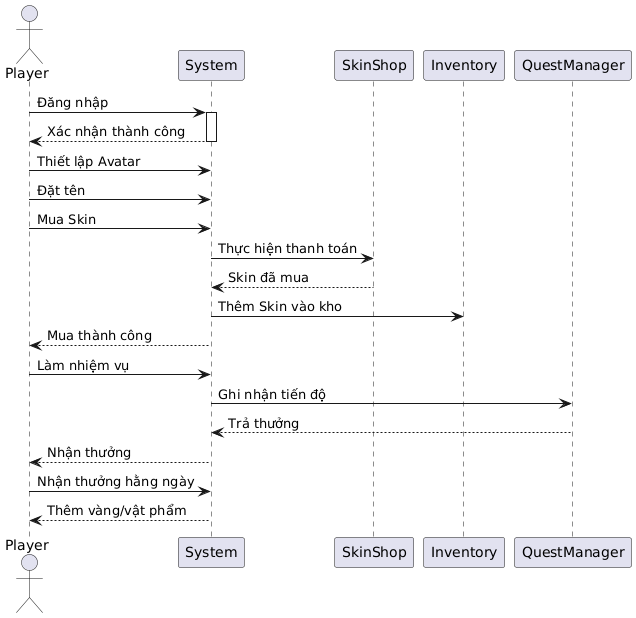


Hình 3. 7: Sơ đồ lớp Hệ thống quản lý

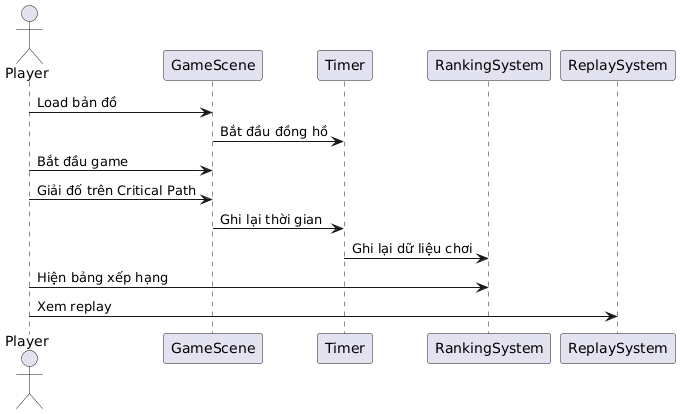


Hình 3. 8: Sơ đồ lớp Hệ thống máu

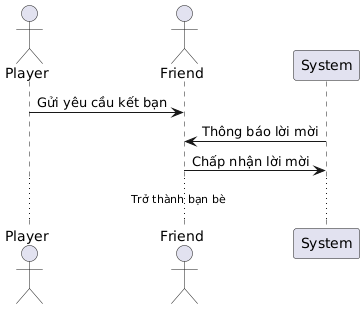
## Biểu đồ trình tự



Hình 3. 9: Biểu đồ trình tự Hoạt động chính của người chơi

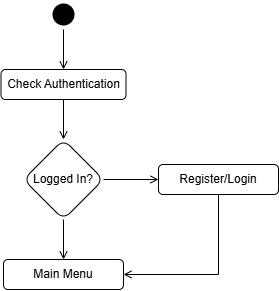


Hình 3. 10: Biểu đồ trình tự Trận đấu

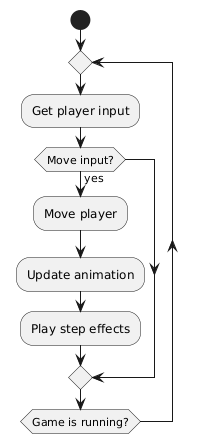


Hình 3. 11: Biểu đồ trình tự Tương tác xã hội

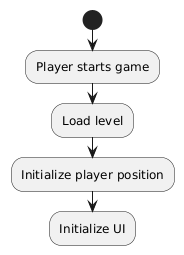
## Sơ đồ hoạt động



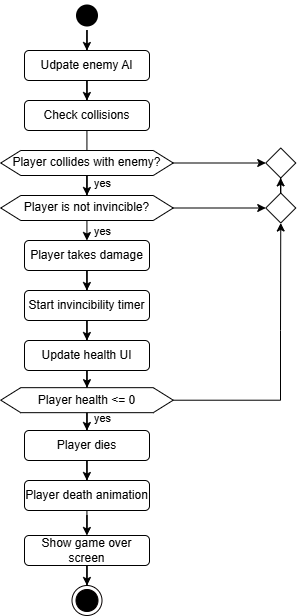
Hình 3. 12: Biểu đồ hoạt động Xác thực tài khoản



Hình 3. 13: Biểu đồ hoạt động Người dùng nhận input



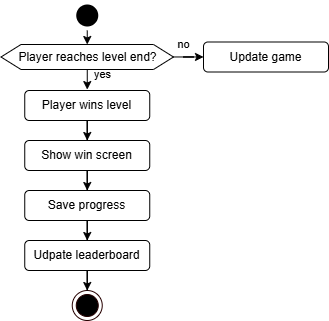
Hình 3. 14: Biểu đồ hoạt động Người chơi bắt đầu game



Hình 3. 15: Biểu đồ hoạt động Ngời chơi va chạm kẻ địch và nhận sát thương

## IMG_256

Hình 3. 16: Biểu đổ hoạt động Đạn của người chơi va chạm kẻ địch



Hình 3. 17: Biểu đồ hoạt động Người chơi hoàn thành màn chơi

## Phác thảo và mô tả giao diện

## Cấu trúc cơ sở dữ liệu Firebase Realtime Database

Dữ liệu sẽ được tổ chức dưới dạng cây JSON. Dưới đây là một ví dụ về cấu trúc dữ liệu chính:

{

  "Leaderboards": {

    "Level 1": {

      "n1VfopPVEdbl6aA8NVeND0Hx7p13": 3.9908547401428223,

      "u8Qt28qTdlb2PSiy6M57tIvyneq1": 1.2167034149169922

    }

  },

  "users": {

    "n1VfopPVEdbl6aA8NVeND0Hx7p13": {

      "active": false,

      "userName": "Admin"

    },

    "u8Qt28qTdlb2PSiy6M57tIvyneq1": {

      "active": false,

      "userName": "Bap"

    }

  }

}

**users**: Chứa thông tin profile của từng người chơi, indexed by userId.

**Leaderboard**: Chứa các bản ghi điểm số cao nhất để tạo bảng xếp hạng. Sử dụng timestamp hoặc điểm số làm key để sắp xếp.

# CHƯƠNG 4. CÀI ĐẶT CHƯƠNG TRÌNH

## Môi trường phát triển

**Unity Editor:** Phiên bản 6000.0.23f LTS

**IDE:** Rider, Visual Studio Code

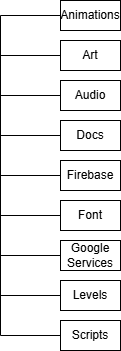
**Ngôn ngữ lập trình:** C#

**Packages:**

* 2D Core, 2D Tilemap Editor.
* Firebase Unity SDK (Authentication, Realtime Database).
* TextMeshPro
* Google Admob

## Cấu trúc mã nguồn

Cấu trúc mã nguồn được tổ chức một cách rõ ràng và khoa học để dễ dàng quản lý, bảo trì và mở rộng. Các thư mục chính được sắp xếp theo chức năng, đảm bảo tính nhất quán và dễ tìm kiếm.



* **Scripts**: Chứa tất cả các script C# của dự án.
* **Manager**: Các script quản lý game chính (GameManager, AudioManager, UIManager).
* **Player**: Script điều khiển nhân vật người chơi (PlayerController, PlayerHealth).
* **Enemies**: Script điều khiển AI và logic của kẻ thù.
* **Environment**: Script liên quan đến môi trường và tương tác cảnh (Platform, Collectible).
* **UI**: Script quản lý giao diện người dùng (MainMenu, InGameUI, LoginUI).
* Firebase: Script xử lý các tương tác với Firebase (AuthenticationManager, DatabaseManager).
* **Utilities**: Các script tiện ích chung hoặc helper.
* **Prefabs**: Chứa các GameObjects được cấu hình sẵn và có thể tái sử dụng (nhân vật, kẻ thù, vật phẩm, hiệu ứng).
* **Scenes**: Chứa các cảnh game khác nhau (MainMenu, Level1, Level2, etc.).
* **Sprites**: Chứa tất cả các tài nguyên đồ họa 2D (nhân vật, nền, vật phẩm).
* **Animations**: Chứa các Animation Clips cho nhân vật và đối tượng.
* **Audio**: Chứa các tệp âm thanh và nhạc nền.
* **Materials**: Chứa các Material cho các đối tượng.
* **ScriptableObjects/Data**: Chứa các Scriptable Objects được sử dụng làm dữ liệu cấu hình.
* **ScriptableObjects/Event Channel**: Script định nghĩa và quản lý hệ thống Event-driven (Scriptable Objects Event Channels).

## Triển khai Gameplay

Việc triển khai gameplay tập trung vào việc tạo ra trải nghiệm 2D Platformer mượt mà và hấp dẫn. Các cơ chế chính bao gồm:

* **Điều khiển nhân vật:**
* Sử dụng Rigidbody2D và Collider2D để quản lý vật lý và va chạm.
* Thực hiện các thao tác di chuyển ngang, nhảy, và leo trèo. Kiểm soát tốc độ di chuyển và lực nhảy để đảm bảo cảm giác điều khiển tự nhiên.
* Xử lý hoạt ảnh (animations) cho các trạng thái di chuyển (Idle, Run, Jump, Attack) thông qua Animator component.
* **Thiết kế cấp độ (Level Design):**
* Sử dụng Tilemap để xây dựng môi trường game nhanh chóng và hiệu quả.
* Đặt các chướng ngại vật, kẻ thù và vật phẩm thu thập vào các vị trí chiến lược.
* Tạo các điểm kiểm soát (checkpoints) và khu vực kết thúc cấp độ.
* **Hệ thống va chạm và tương tác:**
* Sử dụng OnTriggerEnter2D và OnCollisionEnter2D để phát hiện tương tác giữa nhân vật với môi trường, vật phẩm, kẻ thù.
* Triển khai logic thu thập vật phẩm (ví dụ: coin, power-up) và xử lý sát thương khi va chạm với kẻ thù hoặc chướng ngại vật.
* **Hệ thống kẻ thù (Enemy AI):**
* Tạo các loại kẻ thù khác nhau với các hành vi đơn giản (ví dụ: di chuyển qua lại trên một nền tảng, tấn công khi người chơi tiếp cận).
* Sử dụng Raycasting hoặc OverlapCircle để phát hiện người chơi.
* Triển khai hệ thống máu (Health System) cho cả người chơi và kẻ thù, cho phép nhận sát thương và chết.
* **Giao diện người dùng (UI):**
* Sử dụng Unity UI Canvas và TextMeshPro để hiển thị thông tin trong game (điểm số, máu, thời gian).
* Thiết kế các menu (Main Menu, Pause Menu, Game Over Screen) với các nút tương tác.

## Triển khai Firebase Authentication

Tích hợp Firebase Authentication vào game để quản lý tài khoản người chơi, bao gồm các chức năng đăng ký, đăng nhập và đăng xuất.

* **Cài đặt Firebase Unity SDK:**
* Tải và import Firebase Unity SDK vào dự án.
* Cấu hình dự án Firebase trên console và thêm tệp cấu hình google-services.json (cho Android) hoặc GoogleService-Info.plist (cho iOS) vào Unity.
* **Khởi tạo Firebase:**
* Khi ứng dụng khởi động, khởi tạo Firebase App.
* Kiểm tra trạng thái xác thực hiện tại của người dùng.
* **Chức năng Đăng ký (Register):**
* Cho phép người dùng tạo tài khoản mới bằng email và mật khẩu.
* Sử dụng FirebaseAuth.DefaultInstance.CreateUserWithEmailAndPasswordAsync() để gửi yêu cầu đăng ký.
* Xử lý các trường hợp thành công (tạo tài khoản và đăng nhập tự động) và thất bại (email đã tồn tại, mật khẩu yếu).
* **Chức năng Đăng nhập (Login):**
* Cho phép người dùng đăng nhập bằng email và mật khẩu đã đăng ký.
* Sử dụng FirebaseAuth.DefaultInstance.SignInWithEmailAndPasswordAsync() để xác thực.
* Hiển thị thông báo thành công hoặc lỗi (sai email/mật khẩu, tài khoản không tồn tại).
* **Chức năng Đăng xuất (Logout):**
* Người dùng có thể đăng xuất khỏi tài khoản của họ bằng cách gọi FirebaseAuth.DefaultInstance.SignOut().
* Điều hướng người chơi trở lại màn hình đăng nhập hoặc menu chính.
* **Quản lý trạng thái người dùng:**
* Sử dụng OnAuthStateChanged event listener để theo dõi sự thay đổi trạng thái xác thực của người dùng (đăng nhập, đăng xuất).
* Cập nhật UI hoặc chuyển cảnh dựa trên trạng thái xác thực.
* Lưu trữ userId để sử dụng cho Firebase Realtime Database.
* **Xử lý lỗi:**
* Triển khai try-catch để bắt các ngoại lệ từ Firebase API.
* Hiển thị thông báo lỗi rõ ràng cho người dùng (ví dụ: "Email không hợp lệ", "Mật khẩu quá ngắn").
* Việc tích hợp Firebase Authentication giúp đảm bảo tính bảo mật và quản lý người dùng hiệu quả, đồng thời đơn giản hóa quá trình phát triển backend cho game.

# CHƯƠNG 5: KẾT LUẬN

## Kết quả đạt được

Đồ án đã thành công trong việc xây dựng một game 2D Platformer cơ bản với lối chơi hấp dẫn và cơ chế điều khiển mượt mà. Các mục tiêu đề ra ban đầu đã được hoàn thành:

* **Gameplay cốt lõi**: Hệ thống di chuyển, nhảy, thu thập vật phẩm và tương tác với môi trường đã được triển khai hiệu quả, mang lại trải nghiệm chơi game ổn định.
* **Hệ thống xác thực người dùng**: Tích hợp Firebase Authentication thành công, cho phép người chơi đăng ký, đăng nhập và quản lý tài khoản một cách an toàn và dễ dàng.
* **Quản lý dữ liệu thời gian thực**: Sử dụng Firebase Realtime Database để lưu trữ và đồng bộ hóa dữ liệu game như điểm số, thành tích và thông tin người dùng, đảm bảo cập nhật tức thì.
* **Kiến trúc mã nguồn**: Áp dụng các nguyên tắc thiết kế phần mềm tốt như kiến trúc Component-based và Event-driven, giúp mã nguồn dễ bảo trì, mở rộng và tái sử dụng.Tối ưu hóa cơ bản: Thực hiện các kỹ thuật tối ưu hóa đồ họa và hiệu suất cơ bản, giúp game chạy mượt mà trên các nền tảng mục tiêu.

## Hạn chế

Bên cạnh những kết quả đạt được, đồ án vẫn còn một số hạn chế cần được cải thiện:

* **Đa dạng kẻ thù và cấp độ**: Số lượng loại kẻ thù và cấp độ trong game còn hạn chế, chưa đủ phong phú để duy trì sự hấp dẫn lâu dài cho người chơi.
* **Hệ thống UI/UX**: Giao diện người dùng mặc dù hoạt động tốt nhưng còn đơn giản, chưa thực sự tối ưu về mặt trải nghiệm người dùng và tính thẩm mỹ.
* **Tối ưu hóa nâng cao**: Mặc dù đã có tối ưu hóa cơ bản, game vẫn có thể được cải thiện thêm về hiệu suất bằng các kỹ thuật tối ưu hóa đồ họa và logic phức tạp hơn cho các thiết bị di động cấu hình thấp.
* **Thiếu tính năng xã hội mở rộng**: Chưa có các tính năng như hệ thống bạn bè đầy đủ, chia sẻ thành tích trực tiếp lên mạng xã hội.

## Hướng phát triển

Dựa trên những hạn chế và tiềm năng của đồ án, các hướng phát triển trong tương lai có thể bao gồm:

* **Mở rộng tính năng Multiplayer:**
* Phát triển hệ thống phòng chờ nâng cao, cho phép người chơi tạo/tham gia phòng, mời bạn bè.
* Tích hợp tính năng chat trong game.
* Thêm các chế độ chơi multiplayer đa dạng hơn (ví dụ: đua top thời gian, đấu PvP).
* Cải thiện đồng bộ hóa dữ liệu và xử lý trễ (latency) để đảm bảo trải nghiệm multiplayer mượt mà hơn.
* **Bổ sung nội dung game:**
* Thiết kế thêm nhiều cấp độ với độ khó tăng dần và các yếu tố gameplay độc đáo.
* Phát triển các loại kẻ thù mới với AI phức tạp hơn và các thử thách đa dạng.
* Thêm các vật phẩm, power-up và cơ chế tương tác môi trường mới.
* Giới thiệu hệ thống nhiệm vụ hàng ngày/hàng tuần để khuyến khích người chơi.
* Nâng cao trải nghiệm người dùng (UI/UX):
* Thiết kế lại giao diện người dùng theo hướng hiện đại, thân thiện và trực quan hơn.
* Bổ sung hiệu ứng hình ảnh và âm thanh sống động cho UI.
* Cải thiện các phản hồi của hệ thống để người chơi có thể dễ dàng hiểu được trạng thái game.
* **Tối ưu hóa hiệu suất:**
* Áp dụng các kỹ thuật tối ưu hóa chuyên sâu như batching nâng cao, tối ưu hóa shader, và quản lý bộ nhớ tốt hơn để đảm bảo game chạy mượt mà trên nhiều thiết bị.
* Thực hiện profiling thường xuyên để xác định và khắc phục các điểm nghẽn hiệu suất.
* **Phát triển tính năng xã hội:**
* Triển khai hệ thống bạn bè đầy đủ với chức năng gửi/chấp nhận lời mời, xem trạng thái online.
* Cho phép chia sẻ thành tích và replay game lên các nền tảng mạng xã hội.
* Xây dựng hệ thống bảng xếp hạng toàn cầu và theo bạn bè.
* Những hướng phát triển này sẽ giúp game trở nên hoàn thiện, hấp dẫn và có tính cạnh tranh cao hơn trên thị trường game di động.

# TÀI LIỆU THAM KHẢO

[1] Unity Technologies. (N.D.). Unity Documentation. Truy cập từ: [https://docs.unity3d.com/](https://docs.unity3d.com/" \o "null)

[2] Firebase. (N.D.). Firebase Documentation. Truy cập từ: [https://firebase.google.com/docs](https://firebase.google.com/docs" \o "null)

[3] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.