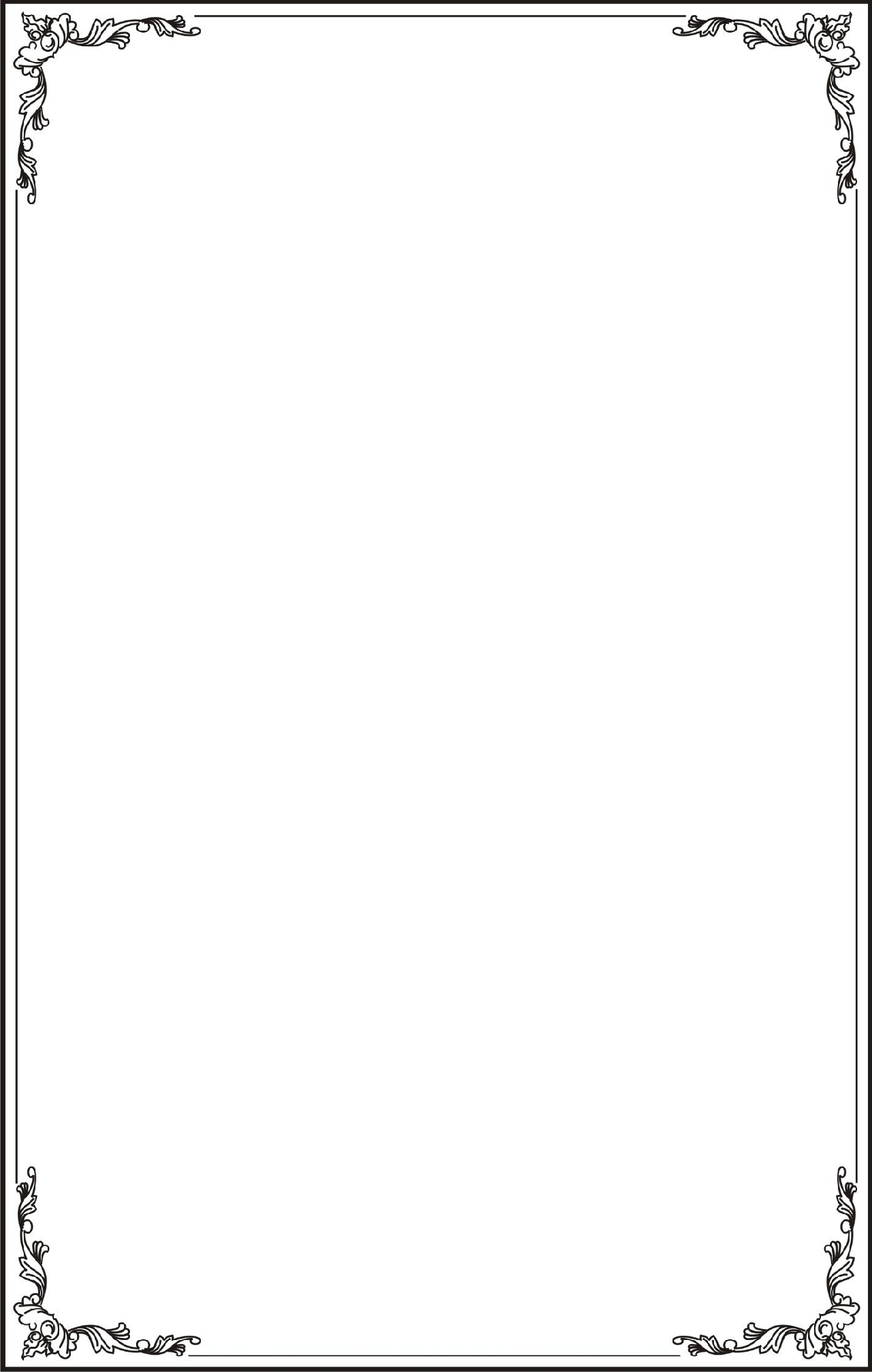
**BỘ GIÁO DỤC VÀ ĐÀO TẠO**



**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

BÁO CÁO TỔNG KẾT

ĐỀ TÀI ĐỒ ÁN TỐT NGHIỆP CỦA SINH VIÊN NĂM HỌC 2023 – 2024

NGHIÊN CỨU CÔNG NGHỆ UNITY ENGINE VÀ ỨNG DỤNG XÂY DỰNG GAME HÀNH ĐỘNG BẮN SÚNG 3D

**Sinh viên thực hiện : Phạm Cao Đại Ân**

**Mã số sinh viên :** 6151071035

**Nam/Nữ :** Nam

**Lớp :** CQ.61.CNTT

**Ngành học :** Công nghệ thông tin

**Năm thứ :** 4/4

**Người hướng dẫn : Ths. Trần Phong Nhã**

TP. Hồ Chí Minh, tháng 6 năm 2024

**MỤC LỤC**

[TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU 1](#_Toc169090995)

[LÝ DO CHỌN ĐỀ TÀI 2](#_Toc169090996)

[**1. Đáp ứng nhu cầu và xu hướng hiện đại** 2](#_Toc169090997)

[**2. Khám phá tiềm năng của Unity trong phát triển game bắn súng 3D** 2](#_Toc169090998)

[**3. Đóng góp vào cộng đồng phát triển game** 2](#_Toc169090999)

[MỤC TIÊU, NỘI DUNG, PHƯƠNG PHÁP NGHIÊN CỨU CỦA ĐỀ TÀI 3](#_Toc169091000)

[**1. Mục tiêu nghiên cứu 3**](#_Toc169091001)

[**2. Nội dung nghiên cứu 3**](#_Toc169091002)

[**3. Phương pháp nghiên cứu 4**](#_Toc169091003)

[ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 6](#_Toc169091004)

[**1. Mục tiêu nghiên cứu 6**](#_Toc169091005)

[**2. Phạm vi nghiên cứu 6**](#_Toc169091006)

[CHƯƠNG 1: TỔNG QUAN 8](#_Toc169091007)

[**1.1 Tổng quan về lĩnh vực game 8**](#_Toc169091008)

[1.1.1 Lịch sử và sự phát triển 8](#_Toc169091009)

[1.1.2 Công nghệ và sáng tạo 8](#_Toc169091010)

[1.1.3 Kinh tế và thương mại 9](#_Toc169091011)

[1.1.4 Xã hội và văn hóa 9](#_Toc169091012)

[1.1.5 Nghệ thuật và thiết kế 10](#_Toc169091013)

[1.1.6 Tương Lai của Ngành Game 10](#_Toc169091014)

[**1.2 Tổng quan về game 3D 10**](#_Toc169091015)

[1.2.1 Định nghĩa và đặc điểm 10](#_Toc169091016)

[1.2.2 Phân loại và ví dụ 11](#_Toc169091017)

[1.2.3 Công nghệ và phát triển 12](#_Toc169091018)

[1.2.4 Xu hướng phát triển 13](#_Toc169091019)

[1.2.5 Thách thức và cơ hội 13](#_Toc169091020)

[**1.3 Tổng quan về game hành động bắn súng 3D 14**](#_Toc169091021)

[1.3.1 Định nghĩa và đặc điểm 14](#_Toc169091022)

[1.3.2 Lịch Sử Phát Triển 14](#_Toc169091023)

[1.3.3 Phân Loại và Ví Dụ 15](#_Toc169091024)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 17](#_Toc169091025)

[**2.1 Giới thiệu về ngôn ngữ lập trình C# 17**](#_Toc169091026)

[2.1.1 Định nghĩa và tính năng chính của ngôn ngữ lập trình C# 17](#_Toc169091027)

[2.1.2 Cách sử dụng C# trong trò chơi. 19](#_Toc169091028)

[2.1.3 Kỹ thuật lập trình hướng đối tượng trong C# 25](#_Toc169091029)

[**2.2 Tổng quan về Unity Engine 30**](#_Toc169091030)

[2.2.1 Unity là gì 30](#_Toc169091031)

[2.2.2 Quá trình phát triển của Unity 31](#_Toc169091032)

[2.2.3 Ưu điểm và nhược điểm của Unity 33](#_Toc169091033)

[2.2.4 Các thành phần trong Unity Editor 36](#_Toc169091034)

[2.2.5 Các khái niệm cơ bản trong Unity 39](#_Toc169091035)

[**2.3 Tổng quan về Visual studio 45**](#_Toc169091036)

[CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ 47](#_Toc169091037)

[**3.1 Phân tích yêu cầu 47**](#_Toc169091038)

[**3.2 Thiết kế thới giới game 48**](#_Toc169091039)

[3.2.1 Tổng quan đồ hoạ màn chơi 48](#_Toc169091040)

[3.2.2 Thiết kế các môi trường chi tiết 49](#_Toc169091041)

[**3.3 Thiết kế nhân vật 51**](#_Toc169091042)

[3.3.1 Xác định và mô tả nhân vật 51](#_Toc169091043)

[3.3.2 Yếu tố cân bằng trong chiến đấu 54](#_Toc169091044)

[**3.4. Thiết kế cốt truyện và nhiệm vụ 56**](#_Toc169091045)

[3.4.1 Cốt truyện 56](#_Toc169091046)

[3.4.2 Nhiệm vụ 57](#_Toc169091047)

[**3.5. Thiết kế vật phẩm và vũ khí** **58**](#_Toc169091048)

[CHƯƠNG 4: TRIỂN KHAI VÀ THỰC HIỆN 60](#_Toc169091049)

[**4.1. Triển khai thế giới game 60**](#_Toc169091050)

[4.1.1 Thiết lập giao diện và phím tắt cho game 60](#_Toc169091051)

[4.1.2 Xây dựng các môi trường chi tiết 66](#_Toc169091052)

[**4.2. Triển khai nhân vật 67**](#_Toc169091053)

[4.2.1 Nhân vật chính (Player Character) 67](#_Toc169091054)

[4.2.2 Kẻ thù (Enemies) 68](#_Toc169091055)

[**4.3. Triển khai cốt truyện và nhiệm vụ 72**](#_Toc169091056)

[**4.4. Triển khai vật phẩm và vũ khí 72**](#_Toc169091057)

[KẾT LUẬN VÀ KIẾN NGHỊ 74](#_Toc169091058)

[**Kết quả nghiên cứu 74**](#_Toc169091059)

[**Những hạn chế 75**](#_Toc169091060)

[**Hướng phát triển 75**](#_Toc169091061)

[TÀI LIỆU THAM KHẢO 77](#_Toc169091062)

**MỤC LỤC HÌNH ẢNH**

Hình 2.1: Giao diện cửa sổ Scene 36

Hình 2.2: Giao diện cửa sổ Hierarchy 37

Hình 2.3: Giao diện cửa sổ game 38

Hình 2.4: Giao diện cửa sổ project 38

Hình 2.5: Giao diện cửa sổ inspector 39

Hình 4.1: Giao diện chính 60

Hình 4.2: Giao diện nâng cấp trang bị 61

Hình 4.3: Giao diện thiết lập 61

Hình 4.4: Giao diện chọn nhiệm vụ 62

Hình 4.5: Giao diện chọn vũ khí 63

Hình 4.6: Hình 4.6 Giao diện chờ trước khi vào game 63

Hình 4.7: Giao diện trong game 64

Hình 4.8: Giao diện tạm dừng game 65

Hình 4.9: Giao diện chiến thắng 65

Hình 4.10: Giao diện thất bại 66

Hình 4.11: Nhân vật chính (Player Character) 67

Hình 4.12: Các loại súng 68

Hình 4.13: Kẻ thù cận chiến (Melee Enemies) 68

Hình 4.14: Vũ khí của kẻ thù cận chiến 69

Hình 4.15: Kẻ thù tầm xa (Ranged Enemies) 69

Hình 4.16: Lựu đạn 70

Hình 4.17: Boss cầm búa 71

Hình 4.18: Boss phun lửa 71

Hình 4.19: Hộp đạn 73

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

........................................................................................................

*Tp.Hồ Chí Minh, ngày....tháng…..năm 2024*

**NGƯỜI NHẬN XÉT**

**MỞ ĐẦU**

Trong bối cảnh phát triển nhanh chóng của ngành công nghiệp trò chơi điện tử, sự tiến bộ trong công nghệ phát triển trò chơi đã trở thành một yếu tố cốt lõi, thúc đẩy việc sáng tạo các tựa game với chất lượng đồ họa cao và trải nghiệm người chơi phong phú. Trong số các công cụ và nền tảng phát triển, **Unity Engine** nổi lên như một trong những công cụ mạnh mẽ và phổ biến nhất, được sử dụng rộng rãi không chỉ bởi các nhà phát triển độc lập mà còn bởi các hãng game lớn.

**Unity Engine** được biết đến với khả năng hỗ trợ phát triển đa nền tảng, từ trò chơi trên máy tính, console, thiết bị di động, cho đến thực tế ảo (VR) và thực tế tăng cường (AR). Với giao diện trực quan, dễ sử dụng cùng bộ công cụ phong phú, Unity tạo điều kiện thuận lợi cho các nhà phát triển trong việc hiện thực hóa ý tưởng của họ thành các sản phẩm hoàn chỉnh. Đặc biệt, trong lĩnh vực phát triển game hành động bắn súng 3D, Unity cung cấp các tính năng và công cụ quan trọng như hệ thống rendering mạnh mẽ, vật lý thực tế, và hỗ trợ scripting bằng ngôn ngữ C#.

Trong đề tài này, em sẽ tập trung **nghiên cứu công nghệ Unity Engine** và cách thức ứng dụng nó trong **xây dựng một game hành động bắn súng 3D**. Mục tiêu của nghiên cứu là phân tích chi tiết các khía cạnh kỹ thuật của Unity, từ khả năng đồ họa, vật lý, AI, đến quản lý tài nguyên và hiệu suất. Thông qua việc phát triển một tựa game bắn súng 3D mẫu, đề tài sẽ minh họa rõ ràng các ứng dụng thực tiễn của Unity trong việc tạo ra một trò chơi với chất lượng hình ảnh cao, lối chơi cuốn hút, và khả năng mở rộng tố.

# TỔNG QUAN TÌNH HÌNH NGHIÊN CỨU

Unity Engine, ra đời năm 2005 bởi Unity Technologies, nhanh chóng trở thành một công cụ phát triển trò chơi hàng đầu với hơn 50% thị phần trong lĩnh vực phát triển game di động. Sự phổ biến của Unity bắt nguồn từ khả năng hỗ trợ đa nền tảng, cho phép phát triển trò chơi trên nhiều hệ điều hành như Windows, macOS, iOS, Android, và các hệ máy console như PlayStation, Xbox, Nintendo Switch, cũng như nền tảng thực tế ảo (VR) và thực tế tăng cường (AR).

Unity nổi bật với các tính năng mạnh mẽ như hệ thống rendering, công cụ tạo hình, và vật lý tích hợp, lý tưởng cho việc phát triển game hành động bắn súng 3D. Các tựa game như "Escape from Tarkov" và "Hollow Knight" là minh chứng cho khả năng của Unity trong việc cung cấp đồ họa chất lượng cao và trải nghiệm người chơi xuất sắc.

Nghiên cứu về Unity Engine đã được triển khai trên nhiều khía cạnh:

* Rendering và đồ họa: Unity hỗ trợ Pipeline Scriptable Rendering, nâng cao chất lượng đồ họa và tối ưu hóa hiệu suất .
* Vật lý và mô phỏng: Nền tảng NVIDIA PhysX trong Unity cho phép mô phỏng vật lý chính xác trong thời gian thực .
* Scripting và AI: Unity sử dụng C# và hệ thống Visual Scripting để phát triển trí tuệ nhân tạo, từ điều khiển AI cơ bản đến hành vi phức tạp.

Unity còn cung cấp hệ sinh thái phong phú với các plugin và add-ons như ProBuilder để tạo mô hình 3D, Cinemachine cho cảnh quay camera phức tạp, và Unity Asset Store để tích hợp tài nguyên nhanh chóng .

# LÝ DO CHỌN ĐỀ TÀI

**1. Đáp ứng nhu cầu và xu hướng hiện đại**

Unity Engine không chỉ hỗ trợ phát triển trò chơi trên nhiều nền tảng mà còn liên tục cập nhật các tính năng mới như rendering tiên tiến, vật lý thực tế, và trí tuệ nhân tạo. Với khả năng linh hoạt và hiệu suất cao, Unity trở thành sự lựa chọn hàng đầu trong việc phát triển các tựa game hiện đại, đặc biệt là game hành động bắn súng 3D. Thể loại này yêu cầu sự chính xác và tốc độ trong rendering, tính toán vật lý phức tạp, và hành vi trí tuệ nhân tạo nâng cao để tạo ra trải nghiệm người chơi mượt mà và cuốn hút. Unity, với các công cụ và plugin mạnh mẽ như Scriptable Render Pipeline và NVIDIA PhysX, cung cấp giải pháp hoàn hảo cho việc phát triển các tựa game này.

**2. Khám phá tiềm năng của Unity trong phát triển game bắn súng 3D**

Unity nổi tiếng với khả năng hỗ trợ đồ họa chất lượng cao và rendering phức tạp, điều rất quan trọng đối với các trò chơi bắn súng 3D đòi hỏi môi trường thực tế và hiệu ứng động. Nghiên cứu này sẽ giúp khai thác và tối ưu các tính năng đồ họa của Unity, như ray tracing và Pipeline Scriptable Rendering, để tạo ra những hình ảnh chân thực và sinh động. Ngoài ra, việc tích hợp vật lý và AI trong Unity sẽ giúp mô phỏng chính xác các yếu tố vật lý và phát triển trí tuệ nhân tạo cho các đối thủ trong trò chơi, làm tăng tính chân thực và thử thách trong trải nghiệm người chơi.

**3. Đóng góp vào cộng đồng phát triển game**

Nghiên cứu này không chỉ nhằm mục đích tạo ra một trò chơi mẫu mà còn muốn đóng góp vào cộng đồng phát triển game bằng cách chia sẻ những kiến thức và kinh nghiệm thu được. Việc phân tích và đánh giá các công cụ và tính năng của Unity sẽ cung cấp các chiến lược và kỹ thuật hữu ích cho các nhà phát triển game khác. Hơn nữa, việc phát triển một game hành động bắn súng 3D với Unity sẽ tạo cơ hội để thử nghiệm và đánh giá các công nghệ tiên tiến, từ đó đưa ra các khuyến nghị cho các dự án tương lai.

# MỤC TIÊU, NỘI DUNG, PHƯƠNG PHÁP NGHIÊN CỨU CỦA ĐỀ TÀI

**1. Mục tiêu nghiên cứu**

**Phát triển một trò chơi hành động bắn súng chất lượng cao:** Mục tiêu chính của dự án là tạo ra một trò chơi hành động bắn súng có đồ họa ấn tượng, hệ thống gameplay linh hoạt và cảm giác chơi mượt mà, đem lại trải nghiệm giải trí tuyệt vời cho người chơi.

**Tận dụng sức mạnh của Unity Engine 3D:** Sử dụng các tính năng và công cụ mạnh mẽ của Unity Engine 3D để xây dựng môi trường game 3D đa dạng, tối ưu hiệu suất và thực hiện các tính năng đặc biệt như vật lý, ánh sáng và hiệu ứng đặc biệt.

**Thúc đẩy sự tương tác và sự hấp dẫn:** Xây dựng hệ thống gameplay đa dạng với các cấp độ, vũ khí và kỹ năng đặc biệt để kích thích sự tương tác và sự hứng thú của người chơi, từ đó tạo ra một trải nghiệm chơi game sâu sắc và thú vị.

**Tạo ra một môi trường game hấp dẫn:** Tạo ra các cảnh quan đẹp mắt và đa dạng, bao gồm cảnh quan tự nhiên và cảnh quan đô thị, kết hợp với âm thanh và nhạc nền sống động để tạo ra một môi trường game sống động và hấp dẫn.

**2. Nội dung nghiên cứu**

**Tổng quan và Tính năng Unity Engine:** Ra mắt bởi Unity Technologies vào năm 2005, đã nhanh chóng trở thành một trong những công cụ phát triển game phổ biến nhất. Unity hỗ trợ đa nền tảng, cho phép xây dựng trò chơi trên Windows, macOS, iOS, Android, và các hệ máy console như PlayStation, Xbox, và Nintendo Switch, cùng với các nền tảng thực tế ảo (VR) và thực tế tăng cường (AR). Đặc biệt, Unity sử dụng các công cụ như Scriptable Render Pipeline (SRP), High Definition Render Pipeline (HDRP), và Universal Render Pipeline (URP) để cung cấp khả năng rendering mạnh mẽ, giúp tạo ra đồ họa chân thực và hiệu ứng ánh sáng phức tạp. Các tính năng như NVIDIA PhysX cung cấp nền tảng vững chắc cho mô phỏng vật lý, bao gồm va chạm, động lực học cơ học, và trọng lực, trong khi hệ thống scripting sử dụng ngôn ngữ C# và hỗ trợ Visual Scripting giúp lập trình logic gameplay và trí tuệ nhân tạo.

**Phân tích Công nghệ trong Phát triển Game Bắn Súng 3D:** Trong phát triển game bắn súng 3D, Unity hỗ trợ các kỹ thuật rendering tiên tiến như ray tracing, global illumination, và real-time lighting, tạo ra hình ảnh chất lượng cao và hiệu ứng ánh sáng chân thực. Việc sử dụng NVIDIA PhysX giúp xử lý các tương tác vật lý trong trò chơi, bao gồm việc xây dựng mô hình vật lý cho nhân vật và mô phỏng tương tác môi trường một cách chính xác. Phát triển gameplay yêu cầu tích hợp trí tuệ nhân tạo cho các đối thủ, với các công cụ như NavMesh, state machines, và behavior trees, cho phép tạo ra hành vi phức tạp và chiến lược, làm cho AI trong trò chơi trở nên thông minh và thách thức hơn đối với người chơi.

**Quy trình Phát triển Game Bắn Súng 3D với Unity:** Bao gồm nhiều giai đoạn chính. Đầu tiên là lập kế hoạch và thiết kế ý tưởng, bao gồm xác định mục tiêu của trò chơi, xây dựng kịch bản và thiết kế gameplay. Tiếp theo là phát triển mô hình và môi trường 3D, sử dụng các công cụ như ProBuilder để tạo các đối tượng và cảnh quan. Lập trình điều khiển nhân vật và tương tác môi trường là bước tiếp theo, với việc sử dụng ngôn ngữ C# để thực hiện các logic gameplay. Sau đó là tối ưu hóa hiệu suất bằng các công cụ như Profiler, Memory Profiler, và Frame Debugger để đảm bảo trò chơi chạy mượt mà. Cuối cùng, trò chơi được kiểm thử toàn diện và chuẩn bị để xuất bản trên các nền tảng khác nhau, với việc quản lý tài nguyên và tối ưu hóa build.

**3. Phương pháp nghiên cứu**

**Phân tích yêu cầu và mục tiêu**: Xác định các yêu cầu cơ bản của trò chơi hành động bắn súng, bao gồm cách thức hoạt động của gameplay, đồ họa, âm thanh, và các tính năng chính.

Đặt ra mục tiêu cụ thể về sản phẩm cuối cùng cần đạt được, ví dụ như mức độ thú vị, độ phức tạp của gameplay, hoặc khả năng tương tác với môi trường.

**Tìm hiểu công nghệ và công cụ**: Nắm vững kiến thức về Unity Engine 3D và các công nghệ liên quan.

Hiểu rõ về ngôn ngữ lập trình C# và Visual Studio để phát triển ứng dụng trong Unity.

**Thiết kế game**: Xác định và vẽ ra bản thiết kế của trò chơi, bao gồm các cấu trúc dữ liệu, đối tượng, môi trường, và quy tắc gameplay.

Xác định các yếu tố cần thiết để tạo ra một trò chơi hành động bắn súng hấp dẫn và thú vị.

**Triển khai và phát triển**: Sử dụng Unity Engine và Visual Studio để triển khai các phần của trò chơi, bao gồm việc tạo ra các đối tượng, vũ khí, nhân vật, môi trường, và cơ chế gameplay.

Kiểm tra và sửa lỗi thường xuyên để đảm bảo tính ổn định và hiệu suất của trò chơi.

**Thử nghiệm và đánh giá**: Thử nghiệm trò chơi trên một nhóm người chơi để thu thập phản hồi và đánh giá về trải nghiệm gameplay. Phân tích dữ liệu từ các cuộc thử nghiệm để hiểu rõ hơn về sở thích và phản ứng của người chơi.

**Tối ưu hóa và điều chỉnh**: Dựa trên phản hồi từ người chơi và dữ liệu thử nghiệm, tối ưu hóa và điều chỉnh trò chơi để cải thiện trải nghiệm người chơi.

Đảm bảo tính chất cân bằng và hợp lý của gameplay, đồ họa, âm thanh và hiệu suất.

# ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

**1. Mục tiêu nghiên cứu**

**Người chơi (Players)**: Là những người sẽ tương tác trực tiếp với game. Cần nghiên cứu về hành vi của người chơi, như cách họ tương tác với môi trường, phản ứng với các yếu tố trong trò chơi, và cách họ thích nghi với các cơ chế trò chơi.

**Nhân vật (Characters)**: Bao gồm các nhân vật chính (nhân vật người chơi) và nhân vật phụ (đối thủ hoặc NPC - Non-Player Characters). Cần nghiên cứu về thiết kế, đặc điểm, kỹ năng, và hành vi của các nhân vật trong trò chơi.

**Môi trường (Environment)**: Bao gồm các cảnh quan, cấu trúc, và vật phẩm trong game. Nghiên cứu về thiết kế môi trường, tương tác với môi trường, và cách môi trường ảnh hưởng đến trải nghiệm chơi game.

**Cơ chế trò chơi (Game Mechanics)**: Bao gồm các quy tắc và hệ thống trong trò chơi, như hệ thống vũ khí, hệ thống di chuyển, và hệ thống phản ứng. Nghiên cứu về cách cơ chế trò chơi tác động đến trải nghiệm người chơi và cách tinh chỉnh chúng để tạo ra trò chơi hấp dẫn.

**Đồ họa và Âm thanh (Graphics and Sound)**: Điều này bao gồm việc nghiên cứu về thiết kế đồ họa, hiệu ứng hình ảnh, và âm thanh trong trò chơi. Tìm hiểu cách đồ họa và âm thanh ảnh hưởng đến cảm xúc và trải nghiệm của người chơi.

**Công nghệ (Technology)**: Nghiên cứu về công nghệ phần mềm và phần cứng được sử dụng để phát triển và chạy trò chơi như: Unity Engine, ngôn ngữ lập trình C#, môi trường phát triển tích hợp Visual Sudio.

## 2. Phạm vi nghiên cứu

Đề tài này sẽ tập trung vào việc xây dựng một trò chơi hành động bắn súng sử dụng Unity Engine 3D. Phạm vi nghiên cứu sẽ bao gồm các giai đoạn từ phát triển ý tưởng đến triển khai và kiểm tra:

* **Phát triển ý tưởng**: Nghiên cứu sẽ bắt đầu bằng việc nghiên cứu các trò chơi bắn súng hiện tại để thu thập ý tưởng và yêu cầu. Điều này bao gồm cả việc xác định các yếu tố gameplay như cơ chế bắn, phản ứng của môi trường, và tính tương tác của người chơi.
* **Thiết kế và phát triển**: Sau khi có ý tưởng cơ bản, giai đoạn này sẽ tập trung vào việc thiết kế và phát triển game. Điều này bao gồm việc tạo các mô hình 3D cho nhân vật, vũ khí, và môi trường chơi, cũng như việc lập trình các tính năng gameplay như chuyển động, bắn, và trí tuệ nhân tạo.
* **Tối ưu hóa và kiểm tra**: Phần quan trọng của quá trình phát triển là tối ưu hóa game để đảm bảo hiệu suất tốt trên nhiều loại thiết bị. Giai đoạn này bao gồm kiểm tra và sửa lỗi để cải thiện trải nghiệm người chơi, cũng như điều chỉnh cân bằng gameplay.
* **Đánh giá và phân tích**: Cuối cùng, nghiên cứu sẽ đánh giá trò chơi đã phát triển thông qua các phương tiện như cuộc thử nghiệm người dùng và phản hồi từ cộng đồng người chơi. Phần này sẽ giúp xác định điểm mạnh, điểm yếu và tiềm năng cải thiện trong tương lai của trò chơi.

**CHƯƠNG 1: TỔNG QUAN**

## 1.1 Tổng quan về lĩnh vực game

Lĩnh vực game, hay còn gọi là ngành công nghiệp trò chơi điện tử, đã trải qua sự phát triển mạnh mẽ và toàn diện kể từ những ngày đầu ra đời vào cuối thế kỷ 20. Hiện nay, game không chỉ là một hình thức giải trí phổ biến mà còn là một lĩnh vực kinh tế, văn hóa, và nghệ thuật có ảnh hưởng rộng lớn. Tổng quan về lĩnh vực game bao gồm các khía cạnh phát triển công nghệ, kinh tế, xã hội, văn hóa, và nghệ thuật.

### 1.1.1 Lịch sử và sự phát triển

**Giai đoạn khởi đầu (1950-1970):** Những trò chơi điện tử đầu tiên xuất hiện vào thập niên 1950 với các trò chơi đơn giản trên máy tính lớn. Một trong những bước ngoặt là sự ra đời của trò chơi "Pong" năm 1972, do Atari phát triển, mở ra kỷ nguyên mới cho ngành công nghiệp game.

**Thời kỳ phát triển (1980-2000):** Những năm 1980 chứng kiến sự bùng nổ của máy chơi game console như Nintendo Entertainment System (NES) và Sega Genesis. Sự xuất hiện của các trò chơi arcade và máy tính cá nhân cũng thúc đẩy sự phát triển đa dạng của các thể loại game.

**Thời đại hiện đại (2000 đến nay):** Internet và công nghệ đồ họa đã mang lại một cuộc cách mạng cho ngành game. Các trò chơi trực tuyến, game di động, và nền tảng kỹ thuật số đã trở thành phổ biến, với các sản phẩm như "World of Warcraft", "Fortnite", và "League of Legends" thu hút hàng triệu người chơi [1].

### 1.1.2 Công nghệ và sáng tạo

**Đồ họa và âm thanh:** Công nghệ đồ họa đã tiến xa từ hình ảnh 8-bit đơn giản đến đồ họa 3D chân thực và hiệu ứng đặc biệt trong các trò chơi hiện đại. Âm thanh trong game cũng đã phát triển từ nhạc nền đơn giản đến các bản soundtrack đầy cảm xúc và hiệu ứng âm thanh phức tạp.

**Động cơ game:** Các động cơ phát triển game như Unity, Unreal Engine, và CryEngine cung cấp các công cụ mạnh mẽ để xây dựng game từ các yếu tố cơ bản đến các tính năng phức tạp.

**AI và vật lý:** Trí tuệ nhân tạo và mô phỏng vật lý đã nâng cao trải nghiệm người chơi bằng cách tạo ra các đối thủ và môi trường phản ứng tự nhiên và thông minh.

### 1.1.3 Kinh tế và thương mại

**Quy mô thị trường:** Ngành công nghiệp game đã trở thành một trong những ngành giải trí lớn nhất thế giới, với doanh thu toàn cầu dự kiến đạt hàng trăm tỷ USD. Các phân khúc chính bao gồm game console, game di động, game PC, và game thực tế ảo (VR).

**Kinh doanh và tiếp thị:** Các mô hình kinh doanh trong game như freemium, pay-to-play, và các giao dịch trong game (microtransactions) đã trở thành xu hướng phổ biến, tạo ra các dòng thu nhập liên tục cho các nhà phát triển.

**Thị trường toàn cầu:** Game không còn bị giới hạn bởi địa lý; các sản phẩm game thành công có thể tiếp cận và tạo ra sự ảnh hưởng trên toàn cầu, mở rộng thị trường và đa dạng hóa cơ hội.

### 1.1.4 Xã hội và văn hóa

**Ảnh hưởng xã hội:** Game đã trở thành một phần quan trọng của văn hóa hiện đại, ảnh hưởng đến cách thức giao tiếp, giải trí, và học tập của nhiều thế hệ. Các trò chơi như "Minecraft" và "Among Us" không chỉ là công cụ giải trí mà còn là phương tiện kết nối cộng đồng.

**Giáo dục và huấn luyện:** Các trò chơi giáo dục và huấn luyện đã chứng minh tính hiệu quả trong việc hỗ trợ học tập và phát triển kỹ năng. Game cũng được sử dụng trong các lĩnh vực như y tế, quân sự, và kinh doanh để mô phỏng và đào tạo.

**Esports và cộng đồng:** Esports, hay thể thao điện tử, đã trở thành một hiện tượng toàn cầu với các giải đấu chuyên nghiệp và cộng đồng người hâm mộ nhiệt tình. Các sự kiện như giải vô địch thế giới "League of Legends" thu hút hàng triệu người xem và tạo ra một nền kinh tế xung quanh thể thao điện tử.

### 1.1.5 Nghệ thuật và thiết kế

**Thiết kế game:** Thiết kế game bao gồm việc xây dựng cốt truyện, tạo ra nhân vật, thiết kế cấp độ, và cân bằng gameplay. Đây là một quá trình sáng tạo đòi hỏi sự kết hợp của nghệ thuật, khoa học, và kỹ thuật.

**Nghệ thuật trong game:** Đồ họa, âm thanh, và cốt truyện trong game đều góp phần tạo ra trải nghiệm nghệ thuật độc đáo. Các trò chơi như "Journey" và "The Last of Us" đã được khen ngợi vì khả năng tạo ra những câu chuyện sâu sắc và hình ảnh ấn tượng.

**Tương tác người chơi:** Trải nghiệm người chơi là trọng tâm của thiết kế game, với mục tiêu tạo ra những trò chơi gây hứng thú, thách thức, và đem lại sự thỏa mãn.

### 1.1.6 Tương Lai của Ngành Game

**Công nghệ mới:** Thực tế ảo (VR) và thực tế tăng cường (AR) đang mở ra những cách thức mới cho trải nghiệm game. Blockchain và metaverse cũng đang bắt đầu tác động đến cách thức phát triển và phân phối game.

**Tăng cường cá nhân hóa:** Các công nghệ như machine learning và dữ liệu lớn có thể được sử dụng để tạo ra trải nghiệm game được cá nhân hóa, đáp ứng nhu cầu và sở thích của từng người chơi.

**Trải nghiệm đa nền tảng:** Các trò chơi đa nền tảng cho phép người chơi chuyển đổi mượt mà giữa các thiết bị khác nhau, tạo ra một trải nghiệm liền mạch và linh hoạt.

Lĩnh vực game tiếp tục tiến hóa với tốc độ nhanh chóng, đòi hỏi sự kết hợp của sáng tạo, công nghệ, và hiểu biết sâu sắc về hành vi người chơi. Với sự phát triển không ngừng này, game sẽ tiếp tục giữ vai trò quan trọng trong đời sống văn hóa và xã hội hiện đại, đồng thời mở ra những cơ hội mới cho sự phát triển kinh tế và nghệ thuật [2].

## 1.2 Tổng quan về game 3D

**1.2.1 Định nghĩa và đặc điểm**

Game 3D là trò chơi điện tử sử dụng đồ họa ba chiều (3D) để tạo ra không gian và đối tượng trong trò chơi. Khác với game 2D (hai chiều), game 3D cung cấp trải nghiệm hình ảnh chân thực hơn, cho phép người chơi di chuyển và tương tác trong một không gian có chiều sâu, giúp tăng cường sự nhập vai và tương tác của người chơi.

Đặc điểm chính của game 3D bao gồm:

* **Đồ họa ba chiều:** Các đối tượng trong game được mô hình hóa dưới dạng hình học 3D, thường được xây dựng bằng các điểm (vertices), cạnh (edges), và mặt (faces), sau đó được ánh xạ texture để tạo ra hình ảnh chi tiết.
* **Không gian ba chiều:** Người chơi có thể di chuyển và tương tác trong một không gian có chiều rộng, chiều cao, và chiều sâu, tạo nên trải nghiệm hình ảnh sống động và tự nhiên.
* **Góc nhìn và camera:** Góc nhìn trong game 3D có thể được điều chỉnh linh hoạt, cho phép người chơi quan sát từ nhiều góc độ khác nhau như góc nhìn thứ nhất, thứ ba, hay từ trên xuống.

**1.2.2 Phân loại và ví dụ**

Game 3D có thể được phân loại dựa trên thể loại gameplay và mục tiêu của trò chơi. Dưới đây là một số thể loại phổ biến trong game 3D:

* **Game hành động (Action):**
* Đặc điểm: Thường tập trung vào các hoạt động chiến đấu, di chuyển nhanh, và phản xạ của người chơi.
* Ví dụ: “DOOM Eternal”, “Dark Souls”, “Devil May Cry”.
* **Game phiêu lưu (Adventure)**:
* Đặc điểm: Kết hợp giữa khám phá, giải đố, và câu chuyện sâu sắc.
* Ví dụ: “The Legend of Zelda: Breath of the Wild”, “Tomb Raider”, “Uncharted”.
* **Game nhập vai (RPG):**
* Đặc điểm: Tập trung vào phát triển nhân vật, cốt truyện phong phú, và thế giới mở.
* Ví dụ: “The Witcher 3: Wild Hunt”, “Final Fantasy XV”, “Elder Scrolls V: Skyrim”.
* **Game mô phỏng (Simulation):**
* Đặc điểm: Mô phỏng các hoạt động thực tế như lái xe, bay, hay xây dựng.
* Ví dụ: “Microsoft Flight Simulator”, “The Sims”, “Cities: Skylines”.
* **Game bắn súng (Shooter):**
* Đặc điểm: Tập trung vào chiến đấu bằng súng, có thể từ góc nhìn thứ nhất hoặc thứ ba.
* Ví dụ: “Call of Duty”, “Overwatch”, “Counter-Strike: Global Offensive”.
* **Game chiến thuật (Strategy):**
* Đặc điểm: Yêu cầu người chơi lên kế hoạch và quản lý tài nguyên để đạt được mục tiêu.
* Ví dụ: “StarCraft II”, “Total War: Three Kingdoms”, “Civilization VI”.
* **Game thể thao (Sports):**
* Đặc điểm: Mô phỏng các môn thể thao như bóng đá, bóng rổ, đua xe.
* Ví dụ: “FIFA”, “NBA 2K”, “Forza Horizon”.

**1.2.3 Công nghệ và phát triển**

Động cơ game (Game Engines):

* **Unity:** Phổ biến cho việc phát triển game indie và mobile, cung cấp công cụ mạnh mẽ cho phát triển game 2D và 3D.
* **Unreal Engine:** Nổi bật với khả năng tạo ra đồ họa chất lượng cao, được sử dụng rộng rãi trong các game AAA.
* **CryEngine:** Được biết đến với khả năng xử lý đồ họa ấn tượng và hiệu ứng môi trường chi tiết.
* **Công nghệ đồ họa:** Công nghệ như ray tracing (dò tia), tessellation (chia nhỏ bề mặt), và shaders (đổ bóng) giúp tạo ra hình ảnh chân thực với ánh sáng, bóng, và kết cấu chi tiết.
* **Công nghệ vật lý:** Hệ thống vật lý trong game 3D giúp mô phỏng các tương tác tự nhiên như trọng lực, va chạm, và động học, tăng cường sự chân thực của thế giới game.

**1.2.4 Xu hướng phát triển**

**Thực tế ảo (VR) và thực tế tăng cường (AR):** Game 3D đang phát triển mạnh mẽ với VR và AR, mang lại trải nghiệm nhập vai sâu hơn và tương tác trực quan hơn. Ví dụ: “Half-Life: Alyx” (VR), “Pokémon GO” (AR).

**Đa nền tảng:** Xu hướng phát triển game có thể chơi trên nhiều nền tảng khác nhau, từ console, PC, đến thiết bị di động, giúp người chơi có thể trải nghiệm game liền mạch. Ví dụ: “Fortnite”, “Genshin Impact”.

**AI và học máy:** Sử dụng trí tuệ nhân tạo và học máy để tạo ra các NPC (nhân vật không phải người chơi) thông minh hơn, cải thiện tương tác và phản ứng trong game. Ví dụ: AI trong “The Last of Us Part II” được thiết kế để có hành vi chiến thuật và phản ứng tự nhiên.

**Đồ họa tiên tiến:** Sự phát triển không ngừng của phần cứng và phần mềm giúp nâng cao chất lượng đồ họa, với các công nghệ như ray tracing và 4K HDR trở thành tiêu chuẩn. Ví dụ: “Cyberpunk 2077” nổi bật với đồ họa tiên tiến và môi trường chi tiết.

**1.2.5 Thách thức và cơ hội**

**Thách thức:** Yêu cầu phần cứng cao. Game 3D yêu cầu phần cứng mạnh mẽ, đặc biệt đối với đồ họa chất lượng cao và hiệu ứng phức tạp.

**Chi phí phát triển:** Phát triển game 3D đòi hỏi nguồn lực lớn về thời gian, chi phí, và đội ngũ chuyên nghiệp.

**Đa dạng hóa nội dung:** Đáp ứng kỳ vọng của người chơi về nội dung mới và sáng tạo là một thách thức liên tục.

**Cơ hội:** Thị trường mở rộng: Sự phát triển của công nghệ và nền tảng mới mở ra các cơ hội lớn trong việc tiếp cận đối tượng người chơi rộng rãi.

**Trải nghiệm người chơi tốt hơn:** Công nghệ tiên tiến cho phép tạo ra trải nghiệm người chơi phong phú và đa dạng hơn, từ đồ họa đến gameplay.

**Kết Luận:** Game 3D đại diện cho một trong những thể loại phát triển nhanh nhất và phổ biến nhất trong ngành công nghiệp trò chơi điện tử. Với sự kết hợp của công nghệ đồ họa tiên tiến, động cơ game mạnh mẽ, và sáng tạo trong thiết kế, game 3D không chỉ cung cấp trải nghiệm giải trí vượt trội mà còn mở ra những cơ hội mới cho sự phát triển trong tương lai. Thể loại này tiếp tục định hình và thúc đẩy sự phát triển của toàn bộ ngành công nghiệp game, mang lại những trải nghiệm ngày càng phong phú và đa dạng cho người chơi trên toàn thế giới [3].

**1.3 Tổng quan về game hành động bắn súng 3D**

**1.3.1 Định nghĩa và đặc điểm**

Game hành động bắn súng 3D (3D Action Shooter Games) là một thể loại trò chơi điện tử kết hợp giữa yếu tố hành động và bắn súng trong không gian ba chiều. Những trò chơi này thường yêu cầu người chơi sử dụng các loại vũ khí khác nhau để bắn và tiêu diệt đối thủ hoặc hoàn thành các mục tiêu nhất định trong môi trường 3D, đồng thời thường xuyên phải di chuyển và né tránh các đòn tấn công từ kẻ thù.

Đặc điểm chính của game hành động bắn súng 3D:

* **Góc nhìn:** Game có thể sử dụng góc nhìn thứ nhất (First-Person Shooter, FPS) hoặc góc nhìn thứ ba (Third-Person Shooter, TPS).
* **Gameplay nhanh:** Tốc độ chơi thường nhanh, yêu cầu phản xạ tốt và khả năng định hướng nhanh nhạy.
* **Đa dạng vũ khí:** Người chơi thường có quyền sử dụng nhiều loại vũ khí từ súng ngắn, súng trường, vũ khí nổ đến các công nghệ tiên tiến.
* **Môi trường 3D chi tiết:** Môi trường game thường phức tạp với nhiều chi tiết, cho phép người chơi di chuyển tự do, sử dụng vật thể che chắn, và tương tác với các yếu tố trong game.

**1.3.2 Lịch Sử Phát Triển**

**Giai đoạn đầu (1990s):** Những trò chơi bắn súng 3D đầu tiên như "Wolfenstein 3D" (1992) và "Doom" (1993) đã đặt nền móng cho thể loại FPS với đồ họa và cơ chế gameplay đơn giản.

**Thời kỳ phát triển (2000s):** Các tựa game như "Half-Life" (1998) và "Halo: Combat Evolved" (2001) đã cách mạng hóa thể loại này với cốt truyện phức tạp hơn, đồ họa cải tiến và môi trường 3D chi tiết.

**Giai đoạn hiện đại (2010s - nay):** Sự ra đời của các trò chơi như "Call of Duty", "Battlefield", và "Destiny" đã đẩy giới hạn của thể loại này với đồ họa chân thực, chế độ chơi trực tuyến phong phú, và hệ thống nhiệm vụ phức tạp.

**1.3.3 Phân Loại và Ví Dụ**

Game hành động bắn súng 3D có thể được phân loại theo góc nhìn, bối cảnh, hoặc cơ chế gameplay:

* **Góc nhìn thứ nhất (FPS):**
* Đặc điểm: Người chơi quan sát và điều khiển từ góc nhìn của nhân vật, tăng cảm giác nhập vai.
* Ví dụ: “Call of Duty: Modern Warfare”, “Counter-Strike: Global Offensive”, “Doom Eternal”.
* **Góc nhìn thứ ba (TPS):**
* Đặc điểm: Người chơi quan sát nhân vật từ phía sau hoặc từ một góc độ nhất định, cho phép thấy toàn bộ môi trường xung quanh.
* Ví dụ: “Fortnite”, “Gears of War”, “The Division”.
* **Chiến tranh và quân sự:**
* Đặc điểm: Tập trung vào các trận chiến quân sự thực tế hoặc giả tưởng.
* Ví dụ: “Battlefield”, “Arma”, “Medal of Honor”.
* **Khoa học viễn tưởng và giả tưởng:**
* Đặc điểm: Môi trường và cốt truyện lấy bối cảnh tương lai, vũ trụ, hoặc thế giới giả tưởng.
* Ví dụ: “Destiny”, “Mass Effect”, “Warframe”.
* **Kinh dị và sinh tồn:**
* Đặc điểm: Kết hợp yếu tố kinh dị, yêu cầu người chơi không chỉ bắn súng mà còn phải quản lý tài nguyên để sống sót.
* Ví dụ: “Resident Evil”, “Dead Space”, “Metro Exodus”.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Giới thiệu về ngôn ngữ lập trình C#

### 2.1.1 Định nghĩa và tính năng chính của ngôn ngữ lập trình C#

**2.1.1.1 Định Nghĩa**

C# (phát âm là “C-sharp”) là một ngôn ngữ lập trình đa năng và hiện đại được phát triển bởi Microsoft như một phần của nền tảng .NET. Ngôn ngữ này ra đời vào đầu những năm 2000 và đã nhanh chóng trở thành một công cụ mạnh mẽ trong phát triển ứng dụng, đặc biệt là cho các ứng dụng Windows và web. C# được thiết kế để hỗ trợ phát triển nhanh chóng, bảo trì dễ dàng, và tính an toàn trong lập trình, đồng thời kết hợp các đặc điểm của các ngôn ngữ như C++ và Java.

Trong ngữ cảnh của Unity Engine, C# là ngôn ngữ chính thức và phổ biến nhất để phát triển các trò chơi và ứng dụng 3D. Unity sử dụng C# cho các tập lệnh (scripting) để điều khiển các đối tượng trong trò chơi, xử lý logic game, và tạo ra các hành vi động trong thế giới ảo.

**2.1.1.2 Tính năng chính**

**Hướng đối tượng (Object-Oriented Programming - OOP):**

* Lớp (Classes) và Đối Tượng (Objects): C# hỗ trợ mô hình hóa các khái niệm thế giới thực thông qua lớp và đối tượng, cho phép tổ chức và quản lý mã nguồn theo cách trực quan và có cấu trúc.
* Kế Thừa (Inheritance): Cho phép các lớp con thừa hưởng các thuộc tính và phương thức từ lớp cha, giúp giảm thiểu mã lặp lại và thúc đẩy tái sử dụng mã.
* Đa Hình (Polymorphism): Hỗ trợ nhiều hình thức của một hành vi thông qua phương thức ảo (virtual methods) và giao diện (interfaces), tăng cường tính linh hoạt và khả năng mở rộng của hệ thống.

**An toàn và quản lý bộ nhớ:**

* Quản lý bộ nhớ tự động (Garbage Collection): Hệ thống tự động quản lý bộ nhớ, giải phóng không gian không còn sử dụng, giúp tránh các lỗi liên quan đến quản lý bộ nhớ thủ công.
* Kiểm tra kiểu tĩnh (Static Typing): C# kiểm tra kiểu dữ liệu tại thời gian biên dịch, giúp phát hiện và ngăn chặn lỗi sớm trong quá trình phát triển.
* Xử lý ngoại lệ (Exception Handling): Cung cấp cơ chế bắt và xử lý ngoại lệ một cách hiệu quả, giúp quản lý các lỗi và tình huống không mong muốn.

**Tính linh hoạt và hiệu năng:**

* Hỗ trợ đa nền tảng: Mặc dù ban đầu được thiết kế cho hệ điều hành Windows, C# và .NET Core hiện hỗ trợ phát triển đa nền tảng, bao gồm cả macOS và Linux.
* Hiệu năng cao: C# cung cấp các cấu trúc dữ liệu và thuật toán tối ưu hóa, cho phép phát triển các ứng dụng hiệu suất cao.

**Tích hợp với Unity Engine:**

* Tập lệnh MonoBehavior: C# là ngôn ngữ scripting chính trong Unity, cho phép định nghĩa hành vi của các đối tượng thông qua lớp MonoBehavior, dễ dàng tích hợp với các thành phần của Unity.
* API Unity: C# cung cấp truy cập đầy đủ vào API Unity, bao gồm các tính năng như điều khiển đối tượng, xử lý sự kiện, và quản lý tài nguyên.

**Thư viện và Framework mạnh mẽ:**

* .NET Framework và .NET Core: Cung cấp một bộ thư viện phong phú và framework mạnh mẽ hỗ trợ phát triển ứng dụng từ web đến desktop và game.
* LINQ (Language Integrated Query): Hỗ trợ truy vấn các tập dữ liệu theo cách thức cú pháp tích hợp, đơn giản hóa việc thao tác và truy vấn dữ liệu.

**Công Cụ và Hỗ Trợ Phát Triển:**

* Visual Studio: IDE mạnh mẽ từ Microsoft cung cấp các công cụ tiên tiến cho việc viết mã, gỡ lỗi, và triển khai ứng dụng C#.
* Hệ thống quản lý gói NuGet: Dễ dàng quản lý và tích hợp các gói thư viện từ cộng đồng hoặc từ nguồn bên ngoài.

**Phát triển đa nền tảng và đa thiết bị:**

* Xamarin: Cho phép phát triển ứng dụng di động cho iOS và Android bằng C#, với mã nguồn dùng chung cho các nền tảng khác nhau.
* Unity Engine: Hỗ trợ xuất bản game trên nhiều nền tảng từ PC, console, đến các thiết bị di động và VR.

**Các tính năng hiện đại khác:**

* Async/Await: Hỗ trợ lập trình bất đồng bộ, giúp cải thiện hiệu suất và khả năng đáp ứng của ứng dụng mà không cần sử dụng luồng (thread) phức tạp.
* Pattern Matching: Cung cấp cách thức so khớp mẫu trực tiếp trên các cấu trúc dữ liệu, giúp đơn giản hóa mã nguồn và cải thiện tính dễ đọc.

**Kết luận:** C# là một ngôn ngữ lập trình hiện đại và mạnh mẽ với các tính năng hỗ trợ phát triển ứng dụng đa nền tảng và hiệu suất cao. Trong ngữ cảnh của Unity Engine, C# cung cấp các công cụ và tính năng cần thiết để phát triển game 3D, bao gồm quản lý hành vi đối tượng, xử lý sự kiện, và tương tác với các thành phần của Unity. Sự tích hợp sâu sắc với Unity và hệ sinh thái .NET giúp C# trở thành lựa chọn lý tưởng cho việc xây dựng các trò chơi hành động bắn súng 3D, mang lại khả năng phát triển nhanh chóng, bảo trì dễ dàng, và hiệu suất tối ưu.

### 2.1.2 Cách sử dụng C# trong trò chơi.

**2.1.2.1 Tổng quan về C# trong Unity**

C# là ngôn ngữ lập trình chính thức trong Unity Engine, được sử dụng để viết các tập lệnh (scripts) nhằm điều khiển hành vi của đối tượng, xử lý sự kiện, và quản lý logic của trò chơi. Trong một trò chơi hành động bắn súng 3D, C# cho phép bạn xây dựng các yếu tố chính như điều khiển nhân vật, bắn súng, kẻ thù AI, và hệ thống vật lý. Các bước cơ bản để sử dụng C# trong Unity bao gồm:

* **Tạo các tập lệnh (scripts):** Dùng C# để tạo các tập lệnh điều khiển đối tượng và tương tác trong game.
* **Gắn tập lệnh vào đối tượng (GameObjects):** Gắn các tập lệnh vào đối tượng để định nghĩa hành vi và chức năng của chúng.
* **Tương tác với Unity API:** Sử dụng API của Unity để thực hiện các tác vụ như di chuyển, xử lý va chạm, và quản lý tài nguyên.

**2.1.2.2 Điều khiển nhân vật**

Trong game hành động bắn súng 3D, điều khiển nhân vật là một yếu tố quan trọng để tạo ra trải nghiệm mượt mà và tương tác. Các tập lệnh C# giúp thực hiện các hành động như di chuyển, nhảy, và bắn súng. Ví dụ về tập lệnh điều khiển nhân vật:

using UnityEngine;

public class PlayerController : MonoBehaviour

{

public float speed = 10.0f;

public float rotationSpeed = 700.0f;

public float jumpForce = 5.0f;

private Rigidbody rb;

private bool isGrounded;

void Start()

{

rb = GetComponent<Rigidbody>();

}

void Update()

{

float moveHorizontal = Input.GetAxis("Horizontal");

float moveVertical = Input.GetAxis("Vertical");

Vector3 movement = new Vector3(moveHorizontal, 0.0f, moveVertical);

rb.AddForce(movement \* speed);

if (Input.GetButtonDown("Jump") && isGrounded)

{

rb.AddForce(Vector3.up \* jumpForce, ForceMode.Impulse);

}

float rotation = Input.GetAxis("Mouse X") \* rotationSpeed \* Time.deltaTime;

transform.Rotate(0, rotation, 0);

}

void OnCollisionStay()

{

isGrounded = true;

}

void OnCollisionExit()

{

isGrounded = false;

}

}

* Di chuyển: Dùng Input.GetAxis để nhận dữ liệu từ bàn phím và chuột, sau đó điều khiển lực tác động lên nhân vật.
* Nhảy: Kiểm tra trạng thái isGrounded để quyết định khi nào nhân vật có thể nhảy.
* Xoay: Sử dụng chuyển động của chuột để xoay nhân vật.

**2.1.2.3 Hệ thống bắn súng**

Hệ thống bắn súng bao gồm việc quản lý các loại vũ khí, tạo ra và kiểm soát đạn, và xử lý va chạm của đạn với các đối tượng khác. Ví dụ về tập lệnh bắn súng:

using UnityEngine;

public class GunController : MonoBehaviour

{

public GameObject bulletPrefab;

public Transform bulletSpawn;

public float bulletSpeed = 20.0f;

public float fireRate = 0.5f;

private float nextFire = 0.0f;

void Update()

{

if (Input.GetButton("Fire1") && Time.time > nextFire)

{

nextFire = Time.time + fireRate;

Fire();

}

}

void Fire()

{

GameObject bullet = Instantiate(bulletPrefab, bulletSpawn.position, bulletSpawn.rotation);

Rigidbody rb = bullet.GetComponent<Rigidbody>();

rb.velocity = bulletSpawn.forward \* bulletSpeed;

Destroy(bullet, 2.0f);

}

}

* Tạo đạn: Sử dụng Instantiate để tạo đối tượng đạn mới tại vị trí và hướng bắn.
* Tốc độ đạn: Thiết lập tốc độ đạn bằng cách áp dụng lực lên đối tượng đạn.
* Hủy đạn: Hủy đạn sau một khoảng thời gian để tiết kiệm tài nguyên.

**2.1.2.4 AI kẻ thù**

AI (Trí Tuệ Nhân Tạo) cho kẻ thù trong game bắn súng hành động giúp tạo ra các thử thách thú vị bằng cách điều khiển hành vi của kẻ thù như tìm kiếm, tấn công, và phòng thủ. Ví dụ về tập lệnh AI kẻ thù:

using UnityEngine;

using UnityEngine.AI;

public class EnemyAI : MonoBehaviour

{

public Transform target;

public float attackRange = 10.0f;

public float moveSpeed = 5.0f;

public float detectionRange = 15.0f;

private NavMeshAgent agent;

void Start()

{

agent = GetComponent<NavMeshAgent>();

}

void Update()

{

float distance = Vector3.Distance(target.position, transform.position);

if (distance < detectionRange)

{

agent.SetDestination(target.position);

if (distance < attackRange)

{

Attack();

}

}

}

void Attack()

{

// Logic tấn công

Debug.Log("Attacking player!");

}

}

* Phát hiện mục tiêu: Sử dụng khoảng cách để kiểm tra xem mục tiêu có nằm trong phạm vi phát hiện hay không.
* Di chuyển: Dùng NavMeshAgent để di chuyển kẻ thù tới vị trí của mục tiêu.
* Tấn công: Thực hiện hành động tấn công khi mục tiêu ở trong tầm tấn công.

**2.1.2.5 Xử lý va chạm và vật lý**

Xử lý va chạm và vật lý giúp tạo ra sự tương tác giữa các đối tượng trong game, chẳng hạn như va chạm giữa đạn và kẻ thù. Ví dụ về xử lý va chạm đạn:

using UnityEngine;

public class Bullet : MonoBehaviour

{

void OnCollisionEnter(Collision collision)

{

if (collision.gameObject.tag == "Enemy")

{

// Gây sát thương

Destroy(collision.gameObject);

}

Destroy(gameObject);

}

}

* Phát hiện va chạm: Sử dụng OnCollisionEnter để xử lý khi đạn va chạm với các đối tượng.
* Gây sát thương: Kiểm tra loại đối tượng va chạm và thực hiện hành động phù hợp, chẳng hạn như gây sát thương hoặc phá hủy đối tượng.

**2.1.2.6 Tạo UI và HUB**

UI (Giao diện người dùng) và HUD (Head-Up Display) là phần không thể thiếu trong game hành động bắn súng để hiển thị thông tin quan trọng như sức khỏe, đạn dược, và mục tiêu. Ví dụ về hiển thị thanh sức khỏe:

using UnityEngine;

using UnityEngine.UI;

public class PlayerHealth : MonoBehaviour

{

public int maxHealth = 100;

public int currentHealth;

public Slider healthSlider;

void Start()

{

currentHealth = maxHealth;

healthSlider.maxValue = maxHealth;

healthSlider.value = currentHealth;

}

public void TakeDamage(int amount)

{

currentHealth -= amount;

healthSlider.value = currentHealth;

if (currentHealth <= 0)

{

Die();

}

}

void Die()

{

// Logic chết

Debug.Log("Player dead!");

}

}

* Quản lý sức khỏe: Cập nhật thanh sức khỏe dựa trên lượng máu hiện tại.
* Tương tác UI: Dùng Slider từ Unity UI để hiển thị và cập nhật trạng thái sức khỏe của nhân vật.

**Kết luận**: C# cung cấp một môi trường lập trình mạnh mẽ và linh hoạt trong Unity Engine, cho phép xây dựng và điều khiển các yếu tố phức tạp của một trò chơi hành động bắn súng 3D. Từ điều khiển nhân vật, hệ thống bắn súng, AI kẻ thù, đến xử lý va chạm và quản lý UI, C# giúp tạo ra một trải nghiệm game phong phú và đầy thách thức.

### 2.1.3 Kỹ thuật lập trình hướng đối tượng trong C#

Lập trình hướng đối tượng (OOP) là một mô hình lập trình giúp tổ chức mã nguồn thông qua các khái niệm như lớp (class), đối tượng (object), kế thừa (inheritance), đa hình (polymorphism), và đóng gói (encapsulation). Việc áp dụng OOP trong trò chơi hành động bắn súng 3D giúp tăng cường tính mô-đun, tái sử dụng, và bảo trì mã nguồn.

**2.1.3.1 Lớp và đối tượng**

Lớp (class) là bản thiết kế cho đối tượng (object), định nghĩa thuộc tính và hành vi mà các đối tượng của lớp sẽ có. Trong game hành động bắn súng 3D, các lớp có thể đại diện cho các thực thể như người chơi, kẻ thù, vũ khí, và đạn. Ví dụ về lớp Player:

using UnityEngine;

public class Player

{

// Thuộc tính

public string Name { get; set; }

public int Health { get; set; }

public float Speed { get; set; }

// Phương thức khởi tạo

public Player(string name, int health, float speed)

{

Name = name;

Health = health;

Speed = speed;

}

// Phương thức

public void Move(Vector3 direction)

{

// Logic di chuyển

Debug.Log(Name + " is moving in direction: " + direction);

}

public void TakeDamage(int damage)

{

Health -= damage;

if (Health <= 0)

{

Die();

}

}

private void Die()

{

Debug.Log(Name + " has died.");

}

}

* Thuộc tính (Properties): Định nghĩa các đặc điểm của người chơi như tên, sức khỏe, và tốc độ.
* Phương thức (Methods): Định nghĩa hành vi của người chơi như di chuyển và nhận sát thương.

**2.1.3.2 Kế thừa và đa hình**

Kế thừa cho phép một lớp con (subclass) thừa hưởng thuộc tính và phương thức từ lớp cha (superclass). Đa hình cho phép các phương thức trong lớp con có thể thực hiện hành động khác nhau mặc dù có cùng tên phương thức trong lớp cha. Ví dụ về lớp kế thừa từ lớp Character:

using UnityEngine;

// Lớp cha

public class Character

{

public string Name { get; set; }

public int Health { get; set; }

public Character(string name, int health)

{

Name = name;

Health = health;

}

public virtual void Attack()

{

Debug.Log(Name + " attacks.");

}

public void TakeDamage(int damage)

{

Health -= damage;

if (Health <= 0)

{

Die();

}

}

protected virtual void Die()

{

Debug.Log(Name + " has died.");

}

}

// Lớp con

public class Player : Character

{

public Player(string name, int health) : base(name, health) { }

public override void Attack()

{

// Logic tấn công của người chơi

Debug.Log(Name + " performs a player attack.");

}

protected override void Die()

{

// Logic chết của người chơi

Debug.Log(Name + " (player) has died.");

}

}

public class Enemy : Character

{

public Enemy(string name, int health) : base(name, health) { }

public override void Attack()

{

// Logic tấn công của kẻ thù

Debug.Log(Name + " performs an enemy attack.");

}

protected override void Die()

{

// Logic chết của kẻ thù

Debug.Log(Name + " (enemy) has died.");

}

}

* Character: Là lớp cha cung cấp các thuộc tính chung như tên và sức khỏe, và các hành vi như tấn công và nhận sát thương.
* Player và Enemy: Là các lớp con kế thừa từ Character và ghi đè các phương thức (Override) để cung cấp hành vi cụ thể cho từng loại nhân vật.

**2.1.3.3 Đóng gói**

Đóng gói là kỹ thuật giữ các thuộc tính và phương thức riêng tư (private) hoặc bảo vệ (protected) trong một lớp và cung cấp các phương thức công khai (public) để truy cập và thay đổi chúng. Điều này giúp ẩn chi tiết triển khai và bảo vệ tính toàn vẹn của dữ liệu. Ví dụ về đóng gói trong lớp Weapon:

public class Weapon

{

private int ammo;

private int maxAmmo;

public Weapon(int maxAmmo)

{

this.maxAmmo = maxAmmo;

ammo = maxAmmo;

}

public void Shoot()

{

if (ammo > 0)

{

ammo--;

Debug.Log("Shooting! Remaining ammo: " + ammo);

}

else

{

Debug.Log("No ammo left!");

}

}

public void Reload()

{

ammo = maxAmmo;

Debug.Log("Reloaded. Ammo is now: " + ammo);

}

public int GetAmmo()

{

return ammo;

}

}

* **ammo** và **maxAmmo**: Được khai báo là private để ngăn chặn truy cập trực tiếp từ bên ngoài.
* **Shoot** và **Reload**: Cung cấp các phương thức công khai để thay đổi số lượng đạn.
* **GetAmmo**: Cung cấp phương thức để truy vấn số lượng đạn hiện tại.

**2.1.3.4 Giao diện (Interface)**

Giao diện định nghĩa một tập hợp các phương thức mà lớp thực hiện giao diện đó phải triển khai. Giao diện giúp thực hiện đa hình mà không cần kế thừa từ một lớp cụ thể. Ví dụ về giao diện IShootable:

public interface IShootable

{

void Shoot();

void Reload();

}

public class Gun : IShootable

{

public void Shoot()

{

Debug.Log("Gun shooting!");

}

public void Reload()

{

Debug.Log("Gun reloading!");

}

}

public class Cannon : IShootable

{

public void Shoot()

{

Debug.Log("Cannon shooting!");

}

public void Reload()

{

Debug.Log("Cannon reloading!");

}

}

* IShootable: Định nghĩa các phương thức Shoot và Reload mà lớp thực hiện phải triển khai.
* Gun và Cannon: Thực hiện giao diện IShootable và cung cấp triển khai cụ thể cho các phương thức này [4].

## 2.2 Tổng quan về Unity Engine

### 2.2.1 Unity là gì

**Unity** là một nền tảng phát triển trò chơi đa năng và mạnh mẽ, cho phép tạo ra các trò chơi và ứng dụng 2D, 3D trên nhiều nền tảng như PC, console, di động, web, và thực tế ảo (VR/AR). Ra mắt năm 2005 bởi Unity Technologies, Unity nổi bật với giao diện người dùng thân thiện, hệ thống thành phần linh hoạt, hỗ trợ ngôn ngữ C#, và khả năng kết xuất đồ họa tiên tiến. Với kho tài nguyên phong phú, khả năng tích hợp công cụ mạnh mẽ, và cộng đồng hỗ trợ rộng lớn, Unity giúp tăng tốc, tối ưu hóa quy trình phát triển, và cho phép tạo ra các trò chơi chất lượng cao từ ý tưởng đến sản phẩm hoàn chỉnh một cách hiệu quả.

### 2.2.2 Quá trình phát triển của Unity

Unity, một trong những nền tảng phát triển trò chơi phổ biến nhất hiện nay, đã trải qua một quá trình phát triển ấn tượng kể từ khi ra mắt, liên tục cải tiến và mở rộng tính năng để đáp ứng nhu cầu của các nhà phát triển. Dưới đây là tóm tắt về các giai đoạn phát triển quan trọng của Unity:

**2.2.2.1 Khởi nguồn và ra mắt (2005-2008)**

**2005**: Unity Technologies, được thành lập bởi David Helgason, Joachim Ante và Nicholas Francis, giới thiệu Unity 1.0 tại Apple Worldwide Developers Conference (WWDC). Đây là nền tảng đầu tiên cho phép các nhà phát triển tạo trò chơi 3D trên nền tảng Mac, nhắm đến việc đơn giản hóa quá trình phát triển trò chơi và cung cấp một công cụ dễ sử dụng cho các nhà phát triển indie.

**2006**: Unity 1.2 ra mắt, mang đến hỗ trợ cho Windows và tích hợp trình duyệt web, mở rộng khả năng xuất bản trò chơi đa nền tảng.

**2007**: Unity 2.0 ra mắt với một loạt các tính năng mới như hệ thống ánh sáng (lighting system), đổ bóng (shaders), và hỗ trợ tốt hơn cho lập trình C#. Đây là một bước đột phá, tăng cường khả năng đồ họa và chức năng tổng thể của Unity.

**2.2.2.2 Mở rộng và chuyển đổi (2009-2013)**

**2009**: Unity 2.5 ra mắt, lần đầu tiên hỗ trợ phát triển trên Windows. Điều này giúp Unity tiếp cận một thị trường rộng lớn hơn, vượt ra khỏi hệ sinh thái của Mac.

**2010**: Unity 3.0 mang đến các tính năng như tích hợp trình chỉnh sửa hoạt hình, cải tiến hệ thống vật lý, và hỗ trợ cho iOS và Android, đánh dấu bước đầu vào thị trường di động.

**2012**: Unity 4.0 ra mắt với nhiều cải tiến về đồ họa như DirectX 11, hệ thống hoạt hình Mecanim, và công nghệ đổ bóng dựa trên vật lý (Physically Based Shading), tạo ra các hình ảnh chân thực hơn trong trò chơi.

**2.2.2.3 Phát triển và đổi mới (2014-2017)**

**2014**: Unity 5.0 được công bố, nổi bật với nhiều cải tiến về đồ họa như hệ thống ánh sáng toàn cầu (global illumination), hỗ trợ audio mixer, và công cụ WebGL, cho phép các trò chơi chạy trực tiếp trên trình duyệt mà không cần plugin.

**2015**: Unity chuyển sang mô hình phát triển liên tục, với các phiên bản Unity 5.x được phát hành đều đặn, mang lại các bản cập nhật tính năng, cải tiến hiệu năng, và sửa lỗi nhanh chóng.

**2016**: Unity 5.4 giới thiệu các tính năng nâng cao cho thực tế ảo (VR), thực tế tăng cường (AR), và hỗ trợ mạnh mẽ hơn cho các nền tảng di động.

**2.2.2.4 Hiện đại hóa và tích hợp công nghệ mới (2018-2020)**

**2018**: Unity 2018.1 ra mắt với hệ thống lập trình dựa trên dữ liệu (Entity Component System - ECS), giúp tối ưu hóa hiệu năng trò chơi. Cùng với đó là Burst Compiler và hệ thống hạt (Particle System) cải tiến.

**2019**: Unity 2019.1 giới thiệu công cụ Scriptable Render Pipeline (SRP) với hai tùy chọn chính: High Definition Render Pipeline (HDRP) cho đồ họa chất lượng cao và Lightweight Render Pipeline (LWRP) cho các thiết bị di động và hiệu suất thấp. Unity cũng ra mắt công cụ visual scripting Bolt, hỗ trợ lập trình không cần mã.

**2020**: Unity 2020.x tập trung vào cải thiện công cụ dành cho nhà phát triển, với nhiều cải tiến trong quản lý tài nguyên, tối ưu hóa hiệu năng và các công cụ dành cho AR/VR, cũng như tích hợp hệ thống machine learning với ML-Agents.

**2.2.2.5 Đột phá và tiến vào tương lai (2021-Nay)**

**2021**: Unity ra mắt Unity 2021.1 với nhiều cải tiến về hiệu suất, công cụ biên tập, và khả năng lập trình dựa trên dữ liệu. Hệ thống Shader Graph và Visual Effect Graph được cải thiện đáng kể, cho phép các nhà phát triển tạo ra các hiệu ứng đồ họa phức tạp một cách trực quan hơn.

**2022**: Unity 2022.1 tiếp tục cải tiến với công cụ game development mới như Data-Oriented Technology Stack (DOTS) và các cập nhật về công cụ biên tập nhằm tăng cường hiệu suất và trải nghiệm người dùng.

**2023**: Unity 2023.x tích hợp tốt hơn với các nền tảng cloud, hỗ trợ kỹ thuật số hóa và các ứng dụng ảo hóa, cùng với những công cụ tiên tiến cho VR/AR, AI, và công nghệ Web3.

**2024**: Unity 2024.x tập trung vào tích hợp công nghệ thực tế tăng cường (XR), AI nâng cao, và các cải tiến về trình kết xuất đồ họa, đáp ứng nhu cầu ngày càng tăng về trải nghiệm thực tế ảo và ứng dụng AI trong trò chơi.

### 2.2.3 Ưu điểm và nhược điểm của Unity

**2.2.3.1 Ưu điểm của Unity**

**Hỗ trợ đa nền tảng**

* Đa dạng nền tảng: Unity hỗ trợ xuất bản trò chơi trên hơn 25 nền tảng, bao gồm PC, console, di động, web, và thực tế ảo (VR/AR). Điều này cho phép các nhà phát triển tạo ra một phiên bản trò chơi và triển khai nó trên nhiều thiết bị và hệ điều hành khác nhau với ít thay đổi về mã nguồn.
* Dễ dàng chuyển đổi nền tảng: Unity cung cấp các công cụ giúp chuyển đổi giữa các nền tảng một cách dễ dàng, giảm bớt công việc điều chỉnh và tối ưu hóa.

**Giao diện người dùng thân thiện**

* Trực quan: Giao diện Unity Editor rất thân thiện và trực quan, giúp các nhà phát triển dễ dàng sắp xếp và quản lý các tài nguyên trò chơi, chỉnh sửa cảnh, và thiết lập các thông số mà không cần mã hóa nhiều.
* Kéo và Thả: Hỗ trợ thao tác kéo và thả, giúp đơn giản hóa quá trình xây dựng trò chơi, đặc biệt hữu ích cho các nhà phát triển mới hoặc những người không chuyên về lập trình.

**Hệ thống thành phần linh hoạt**

* Modular Design: Unity sử dụng một hệ thống thành phần cho phép gắn kết nhiều tính năng khác nhau vào các đối tượng trò chơi. Điều này giúp tổ chức mã nguồn tốt hơn, tăng khả năng tái sử dụng và dễ dàng mở rộng chức năng của trò chơi.

**Khả Năng Kết Xuất Đồ Họa Cao**

* Công cụ Render mạnh mẽ: Unity cung cấp các công cụ kết xuất tiên tiến như High Definition Render Pipeline (HDRP) và Universal Render Pipeline (URP), hỗ trợ tạo ra hình ảnh chân thực và hiệu ứng đồ họa phức tạp.
* Shader Graph: Unity có Shader Graph, cho phép tạo ra các shader tùy chỉnh mà không cần viết mã, giúp việc tạo các hiệu ứng hình ảnh trở nên dễ dàng hơn.

**Cộng đồng hỗ trợ lớn**

* Tài liệu phong phú: Có rất nhiều tài liệu, diễn đàn, khóa học trực tuyến, và ví dụ mã nguồn có sẵn, giúp các nhà phát triển dễ dàng tìm kiếm giải pháp và học hỏi từ cộng đồng.
* Unity Asset Store: Unity Asset Store cung cấp một kho tài nguyên phong phú, từ mô hình 3D, âm thanh đến các plugin và mã nguồn, giúp tăng tốc quá trình phát triển và tiết kiệm chi phí.

**Khả năng tích hợp công nghệ mới**

* AR/VR và AI: Unity tích hợp tốt với các công nghệ thực tế ảo (VR), thực tế tăng cường (AR), và trí tuệ nhân tạo (AI), cho phép tạo ra các trải nghiệm tiên tiến và phức tạp.

**Công Cụ và Tính Năng Phong Phú**

* Animation và VFX: Unity cung cấp các công cụ mạnh mẽ cho hoạt hình (Mecanim), hệ thống hạt (Particle System), và các hiệu ứng hình ảnh (Visual Effect Graph), giúp tạo ra các hiệu ứng động phức tạp và sinh động.
* Scripting và Customization: Hỗ trợ mạnh mẽ cho lập trình C#, cho phép tùy chỉnh và mở rộng chức năng theo nhu cầu của trò chơi.

**2.2.3.2 Nhược điểm của Unity**

**Hiệu suất (Performance)**

* Hiệu suất có thể chưa tối ưu: Trò chơi phát triển bằng Unity đôi khi gặp phải vấn đề về hiệu suất, đặc biệt là trên các thiết bị có cấu hình thấp hoặc khi xử lý các cảnh phức tạp. Việc tối ưu hóa có thể yêu cầu nhiều kiến thức và kỹ năng.
* Quản lý bộ nhớ: Unity có thể gặp phải vấn đề về quản lý bộ nhớ trong các dự án lớn hoặc khi xử lý nhiều tài nguyên cùng lúc, dẫn đến hiện tượng giật, lag hoặc crash.

**Chi phí**

* Phiên bản chuyên nghiệp (Pro): Phiên bản Unity Pro yêu cầu phí bản quyền, đặc biệt nếu doanh thu của dự án vượt quá ngưỡng nhất định. Chi phí này có thể cao đối với các nhà phát triển indie hoặc các dự án nhỏ.
* Mua tài nguyên: Mặc dù có nhiều tài nguyên miễn phí, nhưng các tài nguyên chất lượng cao và plugin thường yêu cầu mua bản quyền, làm tăng chi phí phát triển.

**Sự phức tạp trong dự án lớn**

* Quản lý dự án phức tạp: Trong các dự án lớn với nhiều tài nguyên và mã nguồn phức tạp, việc quản lý và duy trì có thể trở nên khó khăn. Các nhà phát triển cần phải có kiến thức tốt về tổ chức và quản lý tài nguyên để tránh các vấn đề về hiệu suất và bảo trì.
* Công cụ kiểm soát phiên bản (Version Control): Unity không tích hợp công cụ kiểm soát phiên bản mặc định, và các hệ thống bên ngoài như Git có thể yêu cầu thiết lập thêm và không phải lúc nào cũng hoạt động trơn tru với dự án Unity lớn.

**Khả năng tùy chỉnh**

* Hạn chế tùy chỉnh mặc định: Một số tùy chỉnh phức tạp có thể yêu cầu kiến thức sâu về hệ thống nội bộ của Unity hoặc cần phải viết lại các công cụ hoặc thành phần có sẵn.

**Cập nhật liên tục**

* Cập nhật thường xuyên: Unity liên tục cập nhật phiên bản mới với các tính năng và cải tiến, nhưng điều này có thể dẫn đến các vấn đề về tương thích và lỗi khi nâng cấp các dự án đang phát triển.

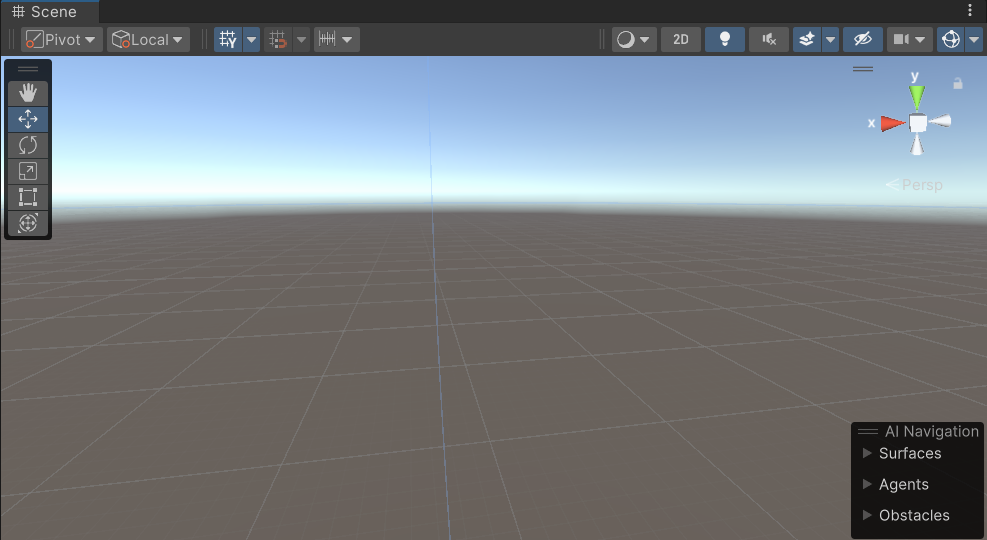
**Tài liệu phân tán**

* Thông tin phân tán: Mặc dù có nhiều tài liệu và hướng dẫn, nhưng chúng có thể phân tán và không nhất quán. Đôi khi việc tìm kiếm giải pháp cho một vấn đề cụ thể có thể gặp khó khăn do tài liệu không đầy đủ hoặc lỗi thời.

### 2.2.4 Các thành phần trong Unity Editor

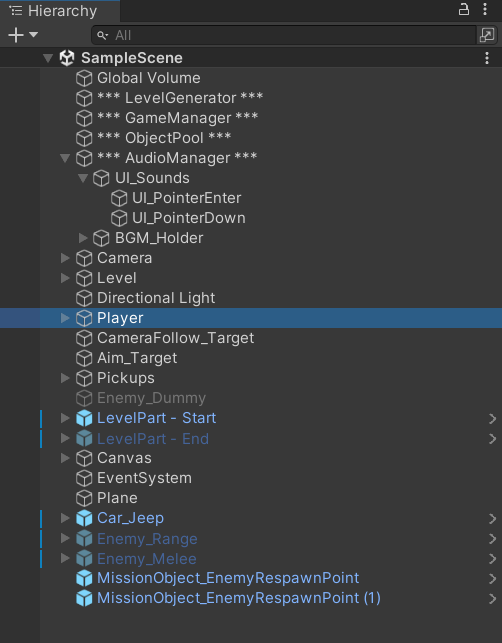
**2.2.4.1 Cửa sổ Scene**

Cửa sổ Scene trong Unity là nơi quan trọng cho việc tạo, chỉnh sửa và quản lý cảnh trò chơi. Nó cung cấp các công cụ và tính năng để thực hiện các tác vụ như thêm đối tượng, điều chỉnh vị trí và kích thước, và thiết lập ánh sáng và camera. Cửa sổ Scene giúp tăng hiệu suất phát triển bằng cách cho phép nhà phát triển thấy kết quả ngay lập tức và làm việc trực tiếp với các yếu tố của trò chơi một cách trực quan và linh hoạt.



**Hình 2.1 Giao diện cửa sổ Scene**

**2.2.4.2 Cửa sổ Hierarchy**

Cửa sổ Hierarchy trong Unity là nơi quản lý toàn bộ cấu trúc đối tượng trong cảnh. Nó hiển thị đối tượng dưới dạng cây, cho phép thêm, xóa, và tổ chức chúng một cách dễ dàng. Cửa sổ này cung cấp các tính năng như tạo và quản lý đối tượng, tương tác với từng đối tượng và chỉnh sửa thuộc tính của chúng. Sử dụng cửa sổ Hierarchy, nhà phát triển có thể tổ chức cảnh trò chơi một cách có tổ chức, tăng hiệu suất phát triển và quản lý dự án một cách hiệu quả.

**Hình 2.2 Giao diện cửa sổ Hierarchy**

**2.2.4.3 Cửa sổ Game**

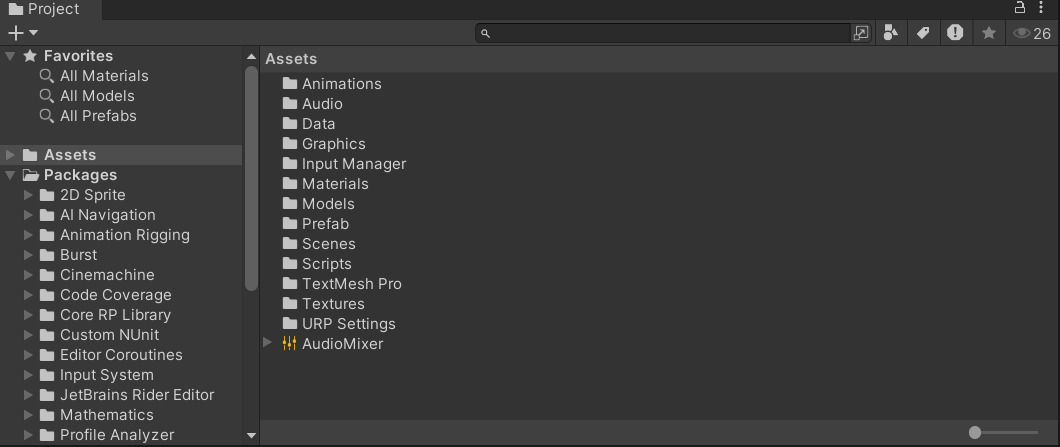
Cửa sổ Game trong Unity là nơi hiển thị trực tiếp trải nghiệm trò chơi trong quá trình phát triển. Nó cho phép nhà phát triển kiểm tra và điều chỉnh trò chơi trực tiếp, xem thông tin debug và kiểm tra hiệu suất trò chơi. Cửa sổ Game cung cấp khả năng điều khiển linh hoạt và tính năng debugging, giúp tăng cường hiệu suất phát triển và đảm bảo chất lượng của trò chơi.



**Hình 2.3 Giao diện cửa sổ game**

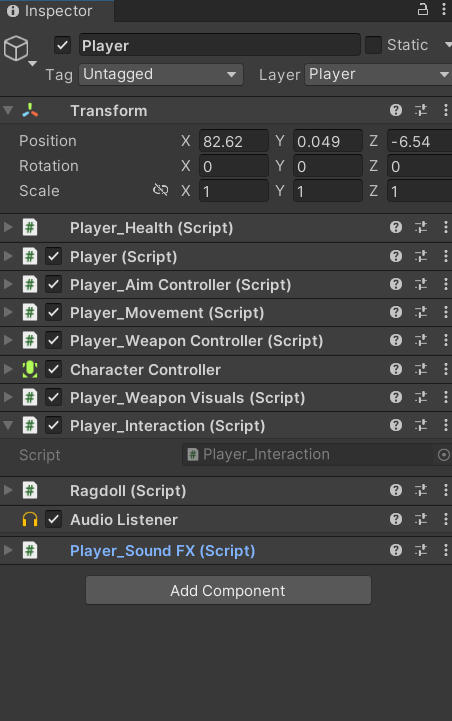
**2.2.4.4 Cửa sổ Project**

Cửa sổ Project trong Unity là nơi quản lý và tổ chức tất cả các tài nguyên và tệp tin của dự án trò chơi. Nó cho phép nhà phát triển thêm, xóa, và tổ chức các tệp tin và thư mục, tìm kiếm và lọc tài nguyên, và quản lý nhập và xuất các asset. Cửa sổ Project là công cụ quan trọng giúp tăng cường hiệu suất phát triển và quản lý dự án một cách hiệu quả.



**Hình 2.4 Giao diện cửa sổ project**

**2.2.4.5 Cửa sổ Inspector**

Cửa sổ Inspector trong Unity là nơi hiển thị và chỉnh sửa thông tin chi tiết và thuộc tính của các đối tượng trong cảnh và dự án. Nó cho phép nhà phát triển xem và chỉnh sửa các thuộc tính của đối tượng một cách trực tiếp, thêm và xóa các thành phần, và tùy chỉnh hiển thị thông tin theo nhu cầu. Cửa sổ Inspector là một công cụ quan trọng giúp tăng cường hiệu suất phát triển và quản lý dự án một cách hiệu quả.

**Hình 2.5 Giao diện cửa sổ inspector**

### 2.2.5 Các khái niệm cơ bản trong Unity

**2.2.5.1 GameObject**

Trong Unity, GameObject là một đơn vị cơ bản và quan trọng nhất trong việc xây dựng trò chơi. Mỗi đối tượng trong trò chơi, như nhân vật, vật phẩm, môi trường, hoặc bất kỳ thành phần nào khác, đều được biểu diễn bằng GameObject.

GameObject chứa thông tin về vị trí, quay, tỷ lệ, và các thành phần khác như đồ họa, vật lý, và scripts để xác định hành vi và hiển thị của đối tượng đó trong trò chơi. Có thể gán các thành phần như Sprite Renderer, Rigidbody, Collider, hoặc các scripts vào GameObject để định nghĩa các tính năng và hành vi của nó.

GameObject có thể tổ chức thành các cấu trúc phân cấp, tạo thành cây GameObject trong cửa sổ Hierarchy của Unity. Điều này cho phép nhà phát triển tổ chức và quản lý toàn bộ cấu trúc và hành vi của trò chơi một cách có hệ thống.

**2.2.5.2 Component**

Trong Unity, Component là các thành phần cụ thể được gắn kèm vào GameObject để định nghĩa các tính năng, hành vi, hoặc khả năng của đối tượng đó trong trò chơi. Mỗi Component đều chứa các thông tin và logic để điều khiển một phần cụ thể của GameObject.

Các thành phần (Component) có thể là các thành phần sẵn có của Unity như Rigidbody (cung cấp vật lý), Collider (để xác định va chạm), hoặc Renderer (để hiển thị đồ họa), cũng như các scripts do người dùng tự viết để xử lý logic và hành vi đặc biệt cho GameObject.

Mỗi GameObject có thể chứa nhiều thành phần khác nhau, và sự kết hợp của các thành phần này quyết định hành vi và tính năng của đối tượng đó trong trò chơi. Bằng cách kết hợp và tương tác giữa các Component, nhà phát triển có thể tạo ra các trải nghiệm trò chơi phong phú và đa dạng.

**2.2.5.3 Sprite**

Trong lĩnh vực đồ họa và game, Sprite là một khái niệm được sử dụng để chỉ đối tượng hình ảnh hoặc biểu tượng được sử dụng trong trò chơi hoặc ứng dụng. Cụ thể, một Sprite thường là một hình ảnh 2D hoặc một tập hợp các hình ảnh được sử dụng để đại diện cho một đối tượng, nhân vật, hoặc hiệu ứng trong trò chơi.

Sprites thường được sử dụng để hiển thị các nhân vật, vật phẩm, môi trường, hoặc các yếu tố khác trong một trò chơi 2D hoặc trong các yếu tố giao diện người dùng trong các ứng dụng. Chúng có thể có định dạng hình ảnh phổ biến như PNG, JPEG, hoặc GIF và thường được sử dụng cùng với các công cụ đồ họa để tạo ra và chỉnh sửa.

Trong môi trường phát triển trò chơi bằng Unity, Sprite Renderer là một thành phần phổ biến được sử dụng để hiển thị các Sprite. Nó cho phép các Sprite được gắn kèm vào GameObject và được hiển thị trong cảnh của trò chơi.

**2.2.5.4 Animation**

Trong lĩnh vực phát triển trò chơi và đồ họa máy tính, Animation là quá trình tạo ra sự chuyển động của các đối tượng hoặc hình ảnh dựa trên một chuỗi các hình ảnh hoặc trạng thái khác nhau. Animation cho phép tạo ra hiệu ứng chuyển động và thay đổi trạng thái của các đối tượng trong trò chơi hoặc các yếu tố đồ họa khác.

Các animation có thể được sử dụng để tạo ra các hiệu ứng như chuyển động của nhân vật, hoạt ảnh của các đối tượng, hoặc các yếu tố đồ họa đặc biệt như các hiệu ứng đặc biệt. Chúng thường được tạo ra bằng cách sử dụng các phần mềm hoạt ảnh hoặc các công cụ đồ họa máy tính như Adobe Animate, Spine, hoặc Unity Animator.

Trong môi trường phát triển trò chơi bằng Unity, Animation được sử dụng để tạo ra các hiệu ứng chuyển động cho các đối tượng và nhân vật trong trò chơi. Unity cung cấp một hệ thống Animation mạnh mẽ cho phép nhà phát triển tạo ra và quản lý các animation, cũng như kết hợp chúng với các scripts để điều khiển hành vi của trò chơi.

**2.2.5.5 Keyframe**

Trong hoạt ảnh và đồ họa máy tính, Keyframe (khung khóa) là một khái niệm quan trọng trong việc tạo ra các hiệu ứng chuyển động và animation. Mỗi keyframe đại diện cho một trạng thái cụ thể của một đối tượng hoặc hình ảnh tại một thời điểm nhất định trong quá trình hoạt động của animation.

Keyframe chứa thông tin về trạng thái của đối tượng vào thời điểm đó, bao gồm thông tin vị trí, quay, tỷ lệ, màu sắc, và các thuộc tính khác. Bằng cách xác định các keyframe tại các điểm thời gian khác nhau và gắn kết chúng lại với nhau, người tạo animation có thể tạo ra sự chuyển động mịn màng và tự nhiên cho các đối tượng hoặc hình ảnh.

Các keyframe thường được sử dụng trong các công cụ tạo animation để định nghĩa và điều khiển các hiệu ứng chuyển động. Bằng cách tạo và chỉnh sửa keyframe tại các vị trí và thời điểm khác nhau, người tạo animation có thể tạo ra các hiệu ứng chuyển động phong phú và đa dạng. Trong Unity, keyframe thường được sử dụng trong việc tạo animation thông qua cửa sổ Animation hoặc

**2.2.5.6 Prefab**

Trong Unity, Prefab là một tập hợp các GameObject và các thành phần liên quan được lưu trữ trong một tệp tin để tái sử dụng trong dự án. Prefab là một khái niệm quan trọng trong việc tổ chức và quản lý các đối tượng trong trò chơi, giúp tạo ra các đối tượng có cùng cấu trúc và hành vi một cách dễ dàng và hiệu quả.

Mỗi Prefab có thể chứa một hoặc nhiều GameObject và các thành phần kèm theo như scripts, mesh renderers, colliders, và các thành phần khác. Khi một Prefab được tạo ra và lưu trữ, bạn có thể tái sử dụng nó trong nhiều vị trí khác nhau trong dự án mà không làm thay đổi nội dung ban đầu của nó. Điều này giúp tiết kiệm thời gian và công sức trong việc phát triển và quản lý dự án.

Khi bạn cần sử dụng một đối tượng từ Prefab, bạn có thể kéo và thả Prefab từ cửa sổ Project vào cảnh hoặc tạo các đối tượng từ Prefab thông qua code. Mọi thay đổi bạn thực hiện trên một Prefab sẽ tự động được áp dụng cho tất cả các thể hiện của nó trong trò chơi. Điều này giúp dễ dàng duy trì tính nhất quán và cập nhật trong toàn bộ dự án.

**2.2.5.7 Sound**

Trong lĩnh vực đồ họa và trò chơi điện tử, Sound (âm thanh) là một yếu tố quan trọng để tạo ra trải nghiệm đa chiều và sống động cho người chơi. Sound được sử dụng để tạo ra các hiệu ứng âm thanh, như âm thanh nền, tiếng súng nổ, tiếng động của môi trường, âm thanh của nhân vật, và nhiều yếu tố âm thanh khác.

Các loại âm thanh trong trò chơi có thể bao gồm âm thanh được ghi âm trước, âm thanh được tạo ra tổng hợp, hoặc âm thanh được tùy chỉnh để phù hợp với nhu cầu cụ thể của trò chơi. Âm thanh không chỉ là một phần quan trọng của trò chơi để cung cấp thông tin cho người chơi, mà còn có thể tạo ra các trạng thái tinh tế và tăng cường trải nghiệm của người chơi.

Trong Unity, Sound được quản lý và sử dụng thông qua các thành phần như AudioSource và AudioClip. AudioSource là một thành phần cho phép phát các âm thanh trong trò chơi, trong khi AudioClip chứa dữ liệu âm thanh được sử dụng bởi AudioSource. Các âm thanh trong Unity có thể được tải từ tệp tin ngoài hoặc được tạo ra trong trình chỉnh sửa Unity.

**2.2.5.8 Script**

Trong ngữ cảnh của phát triển trò chơi và ứng dụng, Script (Kịch bản) là một khối mã được viết bằng một ngôn ngữ lập trình như C#, JavaScript, hoặc Boo để điều khiển hành vi và tương tác của các đối tượng trong trò chơi hoặc ứng dụng.

Script thường được gắn kèm vào các GameObject trong trò chơi để định nghĩa và điều khiển các tính năng và hành vi của chúng. Ví dụ, một Script có thể được sử dụng để điều khiển chuyển động của nhân vật, xử lý va chạm với các đối tượng khác, quản lý điểm số của người chơi, hoặc thực hiện bất kỳ hành vi nào khác cần thiết trong trò chơi.

Trong môi trường phát triển trò chơi bằng Unity, các Script thường được viết bằng ngôn ngữ lập trình C# và được gắn kèm vào các GameObject thông qua các thành phần gọi là "Scripts". Unity cung cấp một hệ thống mạnh mẽ để viết và quản lý các Script, giúp nhà phát triển tạo ra các trò chơi phức tạp và tương tác.

**2.2.5.9 Asset**

Trong ngữ cảnh của phát triển trò chơi và đồ họa máy tính, Asset là một thuật ngữ được sử dụng để chỉ mọi thứ mà bạn thêm vào dự án của mình. Asset có thể là hình ảnh, âm thanh, video, model 3D, animation, script, v.v. - bất cứ thứ gì có thể được sử dụng trong trò chơi hoặc ứng dụng của bạn.

Trong Unity, Asset có thể được tạo ra hoặc nhập vào từ nhiều nguồn khác nhau. Ví dụ, bạn có thể tạo ra các Asset bằng cách tự viết code, làm việc với các chương trình đồ họa như Photoshop hoặc Blender để tạo ra hình ảnh và model 3D, hoặc nhập các tài nguyên từ thư viện và các nguồn tài nguyên trực tuyến.

Asset trong Unity được quản lý và sử dụng thông qua cửa sổ Project. Các Asset được tổ chức vào các thư mục và tệp tin trong cấu trúc thư mục của dự án của bạn. Các Asset này có thể được sử dụng để xây dựng và tạo ra trò chơi hoặc ứng dụng của bạn, bằng cách kéo và thả chúng vào cảnh hoặc sử dụng trong scripts và các thành phần khác của dự án của bạn.

**2.2.5.10 Camera**

Trong phát triển trò chơi và đồ họa máy tính, Camera là một thành phần quan trọng để xác định cách người chơi nhìn thấy và tương tác với thế giới 3D hoặc 2D. Camera chịu trách nhiệm cho việc hiển thị cảnh trò chơi hoặc một khung cảnh nhất định từ góc nhìn của người chơi hoặc quan sát viên.

Trong một trò chơi 3D, Camera thường được đặt ở một vị trí cố định trong không gian 3D và được điều khiển để theo dõi và quan sát các đối tượng di chuyển. Trong khi đó, trong một trò chơi 2D, Camera thường được sử dụng để theo dõi và hiển thị các vị trí của các đối tượng trong một màn hình phẳng.

Các thuộc tính của Camera như vị trí, hướng nhìn, góc nhìn, và các thuộc tính khác có thể được điều chỉnh để tạo ra các hiệu ứng động và cảm giác như zoom, pan, và quay camera. Camera cũng có thể được sử dụng để tạo ra các hiệu ứng đặc biệt như chuyển đổi giữa các cảnh, làm mờ, và các hiệu ứng đặc biệt khác. Trong Unity, Camera là một thành phần quan trọng của mỗi cảnh và được sử dụng để điều chỉnh và hiển thị cảnh trò chơi.

**2.2.5.11 Transform**

Trong Unity, Transform là một thành phần cơ bản của mỗi GameObject, và nó chịu trách nhiệm cho việc xác định vị trí, quay và tỷ lệ của đối tượng đó trong không gian 3D hoặc 2D.

Cụ thể, Transform bao gồm các thuộc tính sau:

* Position (Vị Trí): Đây là vị trí của GameObject trong không gian. Nó được biểu diễn bằng một vector (x, y, z) trong không gian 3 chiều.
* Rotation (Quay): Đây là hướng của GameObject. Rotation được biểu diễn bằng một vector (x, y, z), trong đó mỗi thành phần đại diện cho một góc quay theo các trục x, y, z.
* Scale (Tỷ Lệ): Đây là kích thước của GameObject. Nó được biểu diễn bằng một vector (x, y, z), trong đó mỗi thành phần đại diện cho tỷ lệ tương ứng theo các trục x, y, z.

Transform cho phép nhà phát triển điều chỉnh vị trí, quay và tỷ lệ của đối tượng trong trò chơi. Bằng cách sử dụng các phương thức và thuộc tính của Transform, người phát triển có thể thực hiện các hoạt động như di chuyển, xoay và thay đổi kích thước của đối tượng trong trò chơi [5].

## 2.3 Tổng quan về Visual studio

Visual Studio là một môi trường phát triển tích hợp (IDE - Integrated Development Environment) được phát triển bởi Microsoft, được sử dụng rộng rãi trong việc phát triển ứng dụng máy tính, trò chơi, và các dự án phần mềm khác trên nền tảng Windows.

Dưới đây là một số điểm nổi bật của Visual Studio:

* Hỗ trợ đa ngôn ngữ: Visual Studio hỗ trợ nhiều ngôn ngữ lập trình như C#, C++, Visual Basic, JavaScript, Python, và nhiều ngôn ngữ khác. Điều này cho phép các nhà phát triển sử dụng ngôn ngữ phù hợp nhất với dự án của họ.
* Môi trường phát triển tích hợp (IDE): Visual Studio cung cấp một môi trường phát triển tích hợp, cung cấp các công cụ và tính năng để phát triển, debug và triển khai ứng dụng dễ dàng và hiệu quả.
* Debugging mạnh mẽ: Visual Studio đi kèm với một loạt các công cụ debugging mạnh mẽ, cho phép người phát triển dễ dàng theo dõi và sửa lỗi trong mã nguồn của họ.
* Hỗ Trợ Tích Hợp Cho Các Framework và Platform: Visual Studio hỗ trợ phát triển ứng dụng cho nhiều framework và nền tảng, bao gồm Windows, Web, Mobile, Cloud, và nhiều nền tảng khác.
* Cộng Tác và Quản Lý Dự Án: Visual Studio cung cấp các công cụ cho cộng tác và quản lý dự án, cho phép các nhóm phát triển làm việc cùng nhau trên các dự án phần mềm lớn.
* Phát triển ứng dụng di động: Visual Studio cung cấp các công cụ và mẫu để phát triển ứng dụng di động cho các nền tảng như Android và iOS.

Tóm lại, Visual Studio là một IDE mạnh mẽ và linh hoạt, cung cấp một môi trường phát triển toàn diện để phát triển các loại ứng dụng và dự án phần mềm đa dạng [6].

# CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ

## 3.1 Phân tích yêu cầu

#### 3.1.1 Yêu cầu chính

**Di chuyển nhân vật:**

* Cho phép người chơi di chuyển nhân vật trong không gian 3D bằng bàn phím và chuột.
* Tích hợp hệ thống điều khiển camera để theo dõi nhân vật.

**Bắn súng:**

* Tạo cơ chế bắn súng cho nhân vật, bao gồm việc ngắm bắn và bắn đạn.
* Quản lý đạn, tầm bắn, và sát thương của các loại vũ khí khác nhau.

**Hệ thống kẻ thù:**

* Xây dựng AI cho kẻ thù, bao gồm việc phát hiện và tấn công người chơi.
* Tạo ra các hành vi khác nhau cho kẻ thù, chẳng hạn như tuần tra, truy đuổi, hoặc phòng thủ.

**Môi trường và vật lý:**

* Tạo môi trường tương tác với các vật cản, đối tượng có thể phá hủy, và các khu vực khám phá.
* Sử dụng Unity Physics để xử lý va chạm và tương tác giữa các đối tượng trong trò chơi.

**Hệ Thống Animation:**

* Tích hợp animation cho nhân vật và kẻ thù, bao gồm chạy, nhảy, tấn công, và bị tấn công.
* Sử dụng Unity Animator để quản lý các animation state và transitions.

#### 3.1.2 Yêu cầu phụ

**Hiệu ứng đồ họa và âm thanh:**

* Thêm các hiệu ứng đặc biệt như nổ, tia lửa, và âm thanh bắn súng.
* Tạo nhạc nền và các âm thanh môi trường để tăng cường trải nghiệm chơi.

**Giao diện người dùng (UI):**

* Tạo giao diện người dùng cho các menu, bảng điều khiển, điểm số, và thanh máu.
* Hiển thị thông tin trực quan về trạng thái của nhân vật và kẻ thù.

## 3.2 Thiết kế thới giới game

### 3.2.1 Tổng quan đồ hoạ màn chơi

#### 3.2.1.1 thiết kế môi trường (Environment Design)

**Mô hình và kết cấu 3D (3D Modeling and Texturing)**

* Cảnh quan và địa hình (Landscape and Terrain): Tạo ra các mô hình 3D chi tiết cho địa hình như núi đồi, hẻm núi, thung lũng, hoặc các khu vực thành phố bị phá hủy. Sử dụng công cụ Terrain của Unity để tạo và chỉnh sửa các bề mặt địa hình.
* Kết cấu và vật liệu (Textures and Materials): Áp dụng các kết cấu và vật liệu chân thực cho các bề mặt như đất, nước, cây cối, tường nhà, và kim loại. Sử dụng PBR (Physically-Based Rendering) để tạo ra các vật liệu phản xạ ánh sáng và tương tác với môi trường một cách chính xác.
* Đối tượng môi trường (Environmental Props): Thêm các đối tượng môi trường như cây cối, đá, xe cộ bị phá hủy, các tòa nhà đổ nát để tạo ra một không gian sống động và phong phú.

**Bố cục và thiết kế cảnh Quan (Level Layout and Scene Design)**

* Bố cục màn chơi (Level Layout): Thiết kế các khu vực chiến đấu, điểm nấp, lối đi và các khu vực khám phá để tạo nên các kịch bản chơi đa dạng và hấp dẫn. Đảm bảo rằng màn chơi có sự cân bằng giữa các khu vực mở rộng và các khu vực hẹp để tạo ra trải nghiệm chiến đấu linh hoạt.
* Cấu trúc đường dẫn (Pathways and Navigation): Thiết lập các đường dẫn và nút giao thông để hướng dẫn người chơi và kẻ thù di chuyển. Sử dụng NavMesh của Unity để quản lý việc di chuyển của nhân vật AI trong môi trường 3D.

#### 3.2.1.2 Hệ thống ánh sáng (Lighting System)

**Ánh sáng toàn cảnh (Global Illumination)**

* Ánh sáng toàn cảnh (Global Lighting): Sử dụng hệ thống ánh sáng toàn cảnh để tạo ra ánh sáng môi trường tự nhiên như ánh sáng mặt trời, ánh sáng ban đêm, hoặc ánh sáng trong nhà. Điều chỉnh ánh sáng toàn cảnh để tạo ra các hiệu ứng như bóng đổ và tương phản.

**Đèn điểm và đèn khuếch tán (Point and Area Lights)**

* Đèn điểm (Point Lights): Sử dụng đèn điểm để tạo ra các nguồn sáng cụ thể như đèn đường, đèn xe, hoặc đèn trong nhà.
* Đèn khuếch tán (Area Lights): Áp dụng đèn khuếch tán để chiếu sáng các khu vực rộng lớn hơn như phòng hoặc hành lang.

**Hiệu ứng ánh sáng đặc biệt (Special Lighting Effects)**

* Hiệu ứng chùm sáng (Light Beams): Tạo các hiệu ứng chùm sáng xuyên qua các cửa sổ, khe hở, hoặc lỗ đạn để tăng cường cảm giác chân thực.
* Hiệu ứng phản xạ (Reflections): Sử dụng reflection probes để tạo ra các phản xạ trên các bề mặt như nước, gương, hoặc kim loại.

#### 3.2.1.3 hiệu ứng hạt và đặc biệt (Particle and Special Effects)

**Hiệu ứng hạt (Particle Effects)**

* Hiệu ứng khói và lửa (Smoke and Fire): Sử dụng hệ thống particle của Unity để tạo ra các hiệu ứng khói và lửa từ vụ nổ, súng, hoặc các khu vực bị phá hủy.
* Hiệu ứng mưa và sương mù (Rain and Fog): Áp dụng các hiệu ứng thời tiết như mưa, sương mù để tăng cường bầu không khí của màn chơi.

**Hiệu ứng đặc biệt (Special Effects)**

* Hiệu ứng tia lửa và vụ nổ (Sparks and Explosions): Tạo các hiệu ứng tia lửa khi đạn chạm vào bề mặt kim loại hoặc các vụ nổ từ lựu đạn hoặc bom.
* Hiệu ứng màn hình (Screen Effects): Sử dụng các hiệu ứng màn hình như làm mờ, biến dạng, hoặc rung để tăng cường cảm giác tác động khi có va chạm hoặc bắn súng.

### 3.2.2 Thiết kế các môi trường chi tiết

#### 3.2.2.1 địa hình và bố cục màn chơi (Terrain and Level Layout)

**Tạo địa hình (Terrain Creation)**

* Thiết kế địa hình (Terrain Design): Sử dụng công cụ Terrain của Unity để tạo địa hình chi tiết với các yếu tố tự nhiên như đồi núi, thung lũng, khe núi, và bề mặt gồ ghề. Áp dụng các textures để tạo bề mặt chân thực cho đất, đá, cỏ và nước.
* Tích hợp địa hình và kiến trúc (Terrain and Architecture Integration): Thiết lập các khu vực có kiến trúc như các tòa nhà, hầm ngầm, hoặc các cấu trúc bị phá hủy để thêm tính phức tạp và đa dạng cho môi trường. Đảm bảo sự liền mạch giữa địa hình tự nhiên và các cấu trúc nhân tạo.

**Tạo layout màn chơi (Level Layout Design)**

* Bố cục chiến lược (Strategic Layout): Sắp xếp các khu vực chơi chiến lược như điểm nấp, các tuyến đường tiếp cận, và các điểm bắn chiến lược để khuyến khích người chơi tận dụng môi trường. Thiết kế các tuyến đường và hành lang để tạo ra các điểm nóng chiến đấu và các tình huống phục kích.
* Tạo đường dẫn và Navigation (Pathways and Navigation): Thiết kế các đường dẫn chính và phụ cho người chơi di chuyển và khám phá. Sử dụng Unity’s NavMesh để thiết lập đường dẫn cho AI, đảm bảo rằng kẻ thù có thể di chuyển và hành động một cách thông minh trong môi trường.

#### 3.2.2.2 Đối tượng môi trường (Environmental Props)

**Đối tượng tự nhiên (Natural Props)**

* Cây cối và thảm thực vật (Trees and Vegetation): Sử dụng Unity’s Tree Editor hoặc công cụ bên thứ ba để tạo ra cây cối và thảm thực vật chi tiết, từ cây cao đến bụi cây nhỏ. Đặt các đối tượng này một cách tự nhiên để tạo ra khu vực rừng rậm hoặc đồng cỏ phong phú.
* Đá và địa hình gồ ghề (Rocks and Rough Terrain): Thêm các mô hình đá và địa hình gồ ghề để tăng thêm tính chân thực và thử thách cho người chơi. Sử dụng kết cấu và mô hình để tạo ra các bề mặt đá, vực thẳm, và các đặc điểm địa hình khác.

**Đối tượng nhân tạo (Man-Made Props)**

* Kiến trúc và công trình (Buildings and Structures): Tạo các mô hình chi tiết của các tòa nhà, hầm ngầm, cầu và các công trình bị hư hỏng. Đảm bảo các đối tượng này có chi tiết bên trong và bên ngoài để tạo ra các không gian chiến đấu phong phú.
* Đối tượng động và tương tác (Dynamic and Interactive Props): Thêm các đối tượng có thể tương tác như hộp đạn, thiết bị y tế, và vũ khí để người chơi thu thập và sử dụng. Tạo các đối tượng động như cổng mở, cửa bị khóa, hoặc cầu có thể sập để tăng tính tương tác [7].

## 3.3 Thiết kế nhân vật

### 3.3.1 Xác định và mô tả nhân vật

**3.3.1.1 Nhân vật chính (Player Character)**

**Thông Tin Tổng Quan (Overview)**

* Tên nhân vật: Rusty Maverick
* Giới Tính: Người máy (Robot)
* Tuổi: N/A (Người máy)
* Loại: Người máy cowboy
* Tiểu sử: Rusty Maverick là một người máy cao bồi được thiết kế cho các nhiệm vụ bảo vệ biên giới trên các hành tinh thuộc địa xa xôi. Sau một cuộc tấn công bởi các người máy đối địch, Rusty phải trở thành người bảo vệ cuối cùng, sẵn sàng đối đầu và tiêu diệt mọi kẻ thù để bảo vệ những người máy và thiết bị quan trọng trên hành tinh.

**Kỹ năng và trang bị (Skills and Equipment)**

* **Kỹ năng:**
* Bắn súng chính xác: Khả năng bắn súng chính xác với súng lục laser kiểu cowboy và súng trường năng lượng.
* Chiến đấu cận chiến: Trang bị với roi laser và nắm đấm cơ khí mạnh mẽ cho các pha chiến đấu cận chiến.
* Chế độ tăng cường (Overdrive Mode): Tạm thời tăng cường sức mạnh và tốc độ của Rusty khi ở tình trạng nguy cấp.
* **Trang bị:**
* Vũ khí chính: Súng lục laser đôi (Double Barrel Laserguns), súng trường năng lượng (Photon Rifle), và roi laser (Laser Whip).
* Trang phục: Áo giáp cowboy bằng kim loại, mũ cao bồi tích hợp cảm biến, và lớp phủ chống đạn.
* Thiết bị phụ trợ: Cảm biến phát hiện đối phương, hệ thống sửa chữa tự động, và các viên pin năng lượng dự phòng.

**Động lực và mục tiêu (Motivation and Objectives)**

* Động lực: Bảo vệ các người máy đồng loại và hệ thống cơ sở vật chất khỏi sự tấn công của các người máy địch.
* Mục tiêu: Tiêu diệt các kẻ thù đối địch, bảo vệ tài sản và duy trì hòa bình trên hành tinh.

**3.3.1.2 Kẻ thù (Enemies)**

**Kẻ thù cận chiến (Melee Enemy)**

* **Đặc điểm:** Kẻ thù cận chiến là những người máy được thiết kế cho các đòn tấn công tầm gần, sử dụng tốc độ và sức mạnh để tiếp cận và tiêu diệt Rusty.
* **Hình dáng:** Có thiết kế mảnh mai nhưng chắc chắn, với các bộ phận cơ khí tối ưu cho sự nhanh nhẹn. Thường được trang bị lưỡi dao tích hợp hoặc vuốt kim loại.
* **Trang bị:** Lưỡi dao plasma hoặc vuốt cơ khí.
* **Hành vi:**
* Tấn công: Tấn công bằng các đòn chém, đâm hoặc cắt mạnh mẽ khi tiếp cận được người chơi.
* Di chuyển: Di chuyển nhanh và linh hoạt, có khả năng nhảy, lăn, hoặc chạy nhanh để giảm khoảng cách với người chơi.
* AI: Tận dụng địa hình để tiếp cận Rusty một cách bất ngờ, thường tấn công từ các vị trí khuất hoặc bất ngờ xông vào từ các góc chết.
* **Đặc điểm chiến đấu:** Dễ dàng bị tiêu diệt từ xa nhưng cực kỳ nguy hiểm khi tiếp cận được, buộc người chơi phải giữ khoảng cách và sử dụng chiến thuật để tránh các đòn cận chiến.

**Kẻ thù từ xa (Ranged Enemy)**

* **Đặc điểm:** Kẻ thù từ xa chuyên sử dụng vũ khí tầm xa để tấn công Rusty từ khoảng cách an toàn. Chúng thường lựa chọn vị trí cao hoặc khuất để tấn công mà không bị phát hiện.
* **Hình dáng:** Thiết kế nhỏ gọn và cơ động, với các bộ phận cơ khí giúp di chuyển linh hoạt và dễ dàng ẩn nấp. Thường có bề ngoài bóng bẩy để tránh bị phát hiện.
* **Trang bị:** Súng bắn tỉa năng lượng hoặc súng laser tầm xa.
* **Hành vi:**
* Tấn công: Bắn từ xa với độ chính xác cao, nhắm vào các điểm yếu hoặc các mục tiêu tĩnh. Tấn công liên tục hoặc nhắm mục tiêu từ xa.
* Di chuyển: Dịch chuyển giữa các vị trí bắn an toàn, sử dụng các địa điểm cao hoặc các vị trí khó tiếp cận.
* AI: Tìm kiếm và chiếm lĩnh các vị trí chiến lược, bắn khi có cơ hội và chuyển vị trí để tránh bị phát hiện và tấn công.
* **Đặc điểm chiến đấu:** Gây sát thương lớn từ xa và có khả năng bắn chính xác, nhưng dễ bị tiêu diệt nếu Rusty có thể tiếp cận được vị trí của chúng hoặc sử dụng vũ khí có tầm bắn dài.

**Boss (Boss)**

* **Đặc điểm:** Boss là người máy thủ lĩnh với sức mạnh vượt trội và trí tuệ nhân tạo tiên tiến, cung cấp thử thách lớn nhất trong trò chơi. Boss thường có nhiều giai đoạn và chiến thuật chiến đấu phức tạp.
* **Hình dáng:** Thiết kế hùng vĩ với các bộ phận cơ khí phức tạp và nhiều vũ khí tích hợp. Boss có kích thước lớn hơn nhiều so với các kẻ thù thông thường và có các chi tiết nổi bật để tạo cảm giác uy nghi và nguy hiểm.
* **Trang bị:** Súng năng lượng đa chức năng, pháo điện từ, và lá chắn từ trường.
* **Hành vi:**
* Tấn công: Sử dụng nhiều kiểu tấn công khác nhau từ cận chiến đến tầm xa, với các đòn tấn công đặc biệt và mạnh mẽ. Thường có các kỹ năng đặc biệt và có khả năng triệu hồi hoặc điều khiển các kẻ thù khác trong trận chiến.
* Di chuyển: Di chuyển nhanh nhẹn và linh hoạt, có khả năng né tránh hoặc tạo ra các đòn phản công bất ngờ.
* AI: Sử dụng trí tuệ nhân tạo để phân tích các đòn tấn công của Rusty và điều chỉnh chiến thuật cho phù hợp, tạo ra các giai đoạn chiến đấu với các thay đổi về chiến thuật và tấn công.
* **Đặc điểm chiến đấu:** Rất khó bị đánh bại, đòi hỏi người chơi phải sử dụng toàn bộ kỹ năng và trang bị của mình. Boss có nhiều giai đoạn chiến đấu, mỗi giai đoạn có các kiểu tấn công và chiến thuật riêng.

### 3.3.2 Yếu tố cân bằng trong chiến đấu

**3.3.2.1 Cân bằng giữa người chơi và kẻ thù**

**Sức mạnh và khả năng:**

* **Người chơi:**
* Vũ khí và kỹ năng: Người chơi được trang bị nhiều loại vũ khí và kỹ năng, từ súng ngắn, súng trường đến súng bắn tỉa, mỗi loại có tốc độ bắn, tầm bắn, và sức mạnh khác nhau. Người chơi cũng có các khả năng đặc biệt như né tránh hoặc sử dụng vật phẩm tăng sức mạnh.
* Giáp và sức khỏe: Người chơi có giáp và sức khỏe mà có thể nâng cấp, cung cấp sự bảo vệ trước các đòn tấn công từ kẻ thù.
* **Kẻ thù:**
* **Kẻ thù cận chiến:** Tấn công nhanh và gây sát thương lớn ở cự ly gần nhưng dễ bị tiêu diệt khi bị phát hiện và tấn công từ xa.
* **Kẻ thù từ xa:** Gây sát thương từ xa với độ chính xác cao nhưng dễ bị tiêu diệt nếu người chơi có thể tiếp cận hoặc sử dụng vũ khí có tầm bắn xa hơn.
* **Boss:** Có nhiều giai đoạn và khả năng tấn công đa dạng, đòi hỏi người chơi phải sử dụng chiến thuật và kỹ năng tổng hợp để đánh bại.

**Chiến Thuật và AI:**

* **Người Chơi:**
* Chiến thuật: Người chơi phải sử dụng chiến thuật thích hợp để đối phó với từng loại kẻ thù, như tận dụng địa hình để né tránh các đòn tấn công từ xa hoặc sử dụng vũ khí phù hợp cho từng tình huống chiến đấu.
* AI hỗ trợ: AI hỗ trợ người chơi trong việc né tránh đạn và tìm kiếm các vật phẩm tăng sức mạnh hoặc điểm ẩn nấp.
* **Kẻ Thù:**
* **Kẻ thù cận chiến:** AI giúp kẻ thù tìm đường tiếp cận nhanh chóng và tận dụng các góc khuất để tấn công bất ngờ.
* **Kẻ thù từ xa:** AI chọn vị trí tấn công chiến lược và di chuyển để duy trì khoảng cách an toàn với người chơi.
* **Boss:** AI của Boss phân tích các đòn tấn công của người chơi và thay đổi chiến thuật phù hợp để tăng độ khó và thử thách.

**3.3.2.2 Cân bằng tài nguyên và môi trường**

**Đạn dược và vật phẩm:**

* Đạn dược: Cần được phân phối đều trên toàn bộ màn chơi để người chơi có đủ tài nguyên trong các cuộc đối đầu, nhưng không quá nhiều để làm giảm thử thách.
* Vật phẩm tăng sức mạnh: Các vật phẩm như bộ cứu thương, giáp tăng cường, hoặc vũ khí đặc biệt nên được đặt ở các vị trí chiến lược, buộc người chơi phải mạo hiểm hoặc hoàn thành các nhiệm vụ phụ để thu thập.

**Môi trường:**

* Thiết kế màn chơi: Môi trường trong trò chơi phải được thiết kế để cung cấp nhiều lựa chọn chiến thuật cho người chơi. Các địa điểm ẩn nấp, đường dẫn khác nhau, và các điểm cao phải được cân nhắc kỹ lưỡng để tránh tạo ra lợi thế không công bằng cho bất kỳ bên nào.
* Tương tác môi trường: Người chơi và kẻ thù đều có thể sử dụng môi trường để tấn công hoặc né tránh, như bắn qua các tấm kính, kích hoạt bẫy, hoặc tạo ra vụ nổ từ các thùng chứa dễ cháy.

**3.3.2.3 Cân bằng độ khó và cấp độ**

**Cấp độ độ khó:**

* Dễ: Dành cho người chơi mới làm quen, với ít kẻ thù hơn, kẻ thù có sức mạnh và độ bền thấp hơn, và nhiều tài nguyên hỗ trợ hơn.
* Trung bình: Cung cấp trải nghiệm cân bằng với số lượng và sức mạnh kẻ thù vừa phải, yêu cầu người chơi sử dụng chiến thuật hợp lý.
* Khó: Tăng cường số lượng và sức mạnh kẻ thù, ít tài nguyên hơn, đòi hỏi kỹ năng cao và chiến thuật tinh vi.

**Cấp độ:**

* Mức độ phát triển: Các cấp độ trong trò chơi phải có sự tăng tiến về độ khó và thử thách, với sự xuất hiện của các loại kẻ thù mới và các tình huống chiến đấu phức tạp hơn khi người chơi tiến bộ.
* Giai đoạn: Mỗi cấp độ nên có các giai đoạn khác nhau để giữ người chơi luôn phải điều chỉnh chiến thuật và kỹ năng của mình [8].

## 3.4. Thiết kế cốt truyện và nhiệm vụ

### 3.4.1 Cốt truyện

Trong thế giới mà con người đã chuyển giao quyền kiểm soát cho các người máy, Rusty, một người máy với phong cách của một cao bồi huyền thoại, bất ngờ được khởi động lại sau một thời gian dài bị vô hiệu hóa. Đây là một thế giới nơi con người đã không còn là loài chiếm hữu, mà thay vào đó là các loại máy móc, với trí tuệ nhân tạo vượt trội, nắm giữ quyền kiểm soát và quyết định về mọi khía cạnh của cuộc sống.

Rusty tỉnh dậy giữa đống đổ nát của một thành phố công nghiệp đã bị bỏ hoang, nơi mà những dấu vết của quá khứ rõ ràng đã bị mờ nhạt dưới lớp bụi và sự lãng quên. Trong lòng thành phố, những cơ sở hạ tầng cũ kỹ và những tòa nhà cao tầng đổ nát không còn là nơi sầm uất như trước kia, mà giờ đây chúng trở thành những hầm ngục của sự bất ổn và nổi loạn. Đây là một thế giới mà sự hòa bình và ổn định đã dần biến mất, và tình trạng hỗn loạn trở thành điều bình thường.

Rusty, với một phần ký ức bị mất, phải bắt đầu hành trình của mình trong một cuộc phiêu lưu đầy gian nan và mạo hiểm. Bước đầu tiên của Rusty là khám phá ra những bí ẩn về quá khứ của mình và về lịch sử của thế giới người máy này. Những mảnh ghép của ký ức, mỗi mảnh như một mảnh của câu chuyện lớn, phản ánh lại những thời kỳ và sự kiện quan trọng trong quá khứ, nhưng cũng mang theo những bí mật đen tối mà Rusty cần phải tìm hiểu.

Tuy nhiên, hành trình của Rusty không chỉ dừng lại ở việc khám phá quá khứ của mình. Thành phố đã bị chiếm bởi lực lượng người máy phản loạn, dưới sự lãnh đạo của một Boss bí ẩn và mạnh mẽ. Rusty phải đối mặt với những thử thách và nguy hiểm không ngờ, từ cuộc đấu tranh để tồn tại giữa những con đường vắng vẻ cho đến những cuộc chiến không ngừng nghỉ giữa các khu vực chiến lược của thành phố.

Mục tiêu của Rusty không chỉ là tự bảo vệ bản thân mình, mà còn là ngăn chặn cuộc nổi dậy của các người máy phản loạn và cứu vãn thế giới khỏi sự hỗn loạn và hủy hoại. Với mỗi bước tiến trên hành trình, Rusty phải vượt qua những thử thách mới, tìm kiếm những mảnh ghép của quá khứ của mình và đối đầu với sự phản bội và hiểm nguy từ những kẻ thù mà anh gặp phải.

Rusty không phải chỉ là một người máy, mà là một biểu tượng cho sự kiên cường và sự quyết tâm của con người trong mọi hoàn cảnh. Hành trình của Rusty không chỉ là cuộc chiến để sống sót, mà còn là cuộc chiến để giữ vững giá trị và lý tưởng của con người trong một thế giới đầy rẫy những thách thức và nguy hiểm [9].

### 3.4.2 Nhiệm vụ

**Nhiệm vụ 1: khởi động lại**

* Mô tả: Rusty phải thu thập năng lượng và phần mềm cần thiết để khởi động lại hệ thống của mình. Đây là bước quan trọng đầu tiên để Rusty có thể bắt đầu hành trình của mình.
* Mục tiêu: Thu thập pin năng lượng và phần mềm sửa lỗi từ các khu vực bị bỏ hoang trong thành phố.
* Thử thách: Đối mặt với những kẻ thù cận chiến và tránh né những vật cản nguy hiểm trên đường đi.

**Nhiệm vụ 2: khám phá bí ẩn**

* Mô tả: Rusty phải điều tra những manh mối về quá khứ của mình và về sự sụp đổ của thành phố này. Điều này có thể giúp anh hiểu rõ hơn về lý do mình lại được khởi động lại và về mục đích của mình trong cuộc sống mới.
* Mục tiêu: Thu thập thông tin và mảnh ghép ký ức từ những vị trí khác nhau trong thành phố.
* Thử thách: Đối đầu với các kẻ thù từ xa và giải quyết các câu đố để tiếp cận thông tin ẩn.

**Nhiệm vụ 3: Đối phó với nguy hiểm**

* Mô tả: Rusty phải chống lại sự lây lan của sự hỗn loạn và ngăn chặn sự phát triển của lực lượng phản loạn. Điều này đòi hỏi anh phải tiến vào các khu vực chiến lược của thành phố và tiêu diệt các lực lượng đe dọa.
* Mục tiêu: Tiêu diệt các căn cứ và trạm điều khiển của kẻ thù để giảm thiểu sức mạnh của họ.
* Thử thách: Đối đầu với các kẻ thù cận chiến và từ xa, đồng thời cố gắng tránh những cạm bẫy và vật cản.

**Nhiệm vụ 4: Đối mặt với Boss**

* Mô Tả: Cuối cùng, Rusty phải đối mặt trực tiếp với Boss của lực lượng phản loạn, người đứng sau toàn bộ âm mưu và sự hỗn loạn. Đây là trận chiến cuối cùng để giải cứu thành phố và chứng minh giá trị của mình.
* Mục tiêu: Tiêu diệt Boss và ngăn chặn kế hoạch phản loạn của họ.
* Thử thách: Đối đầu với một quái vật mạnh mẽ có các kỹ năng đặc biệt và chiến thuật phòng thủ và tấn công phức tạp [10].

## 3.5. Thiết kế vật phẩm và vũ khí

**Vật phẩm:**

* Pin năng lượng: Vật phẩm quan trọng nhất để Rusty có thể duy trì hoạt động và sử dụng các kỹ năng đặc biệt. Pin năng lượng có thể được tìm thấy rải rác trong các khu vực của thành phố hoặc từ các kẻ thù tiêu diệt.
* Dây chuyền ký ức: Một loại vật phẩm đặc biệt giúp Rusty thu thập và khôi phục các mảnh ký ức của mình. Khi thu thập đủ mảnh ghép, Rusty có thể mở khóa những ký ức quan trọng về quá khứ của mình.
* Bánh răng cơ khí: Sử dụng để nâng cấp vũ khí và trang bị của Rusty. Bánh răng cơ khí có thể được thu thập từ các cơ sở sản xuất hoặc được trao đổi từ các nhân vật khác trong trò chơi.

**Vũ khí:**

* Súng xạ thủ mạnh mẽ: Một loại súng bắn tỉa có độ chính xác cao và sức mạnh tiêu diệt lớn. Rusty có thể sử dụng súng xạ thủ để tiêu diệt các kẻ thù từ xa hoặc để tiêu diệt các mục tiêu mạnh mẽ từ xa.
* Súng trường phòng vệ: Một loại súng máy hoạt động liên tục, thích hợp cho các tình huống chiến đấu gần gũi và tiêu diệt đám đông kẻ thù. Rusty có thể sử dụng súng trường phòng vệ để chống lại các cuộc tấn công đột ngột từ các kẻ thù cận chiến.
* Dao cắt công cụ: Một loại vũ khí cận chiến, nhỏ gọn và linh hoạt, cho phép Rusty tiến vào gần hơn với kẻ thù và tấn công mạnh mẽ. Dao cắt công cụ cũng có thể được sử dụng để phá vỡ các vật thể và mở khóa cánh cửa.
* Lựu đạn sét chói: Một loại vũ khí nổ mạnh mẽ, gây sát thương lớn cho các kẻ thù xung quanh. Rusty có thể sử dụng lựu đạn sét chói để tiêu diệt các nhóm kẻ thù hoặc để tạo ra cơ hội tấn công trong những tình huống nguy hiểm [11].

# CHƯƠNG 4: TRIỂN KHAI VÀ THỰC HIỆN

## 4.1. Triển khai thế giới game

### 4.1.1 Thiết lập giao diện và phím tắt cho game

#### 4.1.1.1 Giao diện chính



**Hình 4.1 Giao diện chính**

\_Giao diện chính được thiết kế với background là một video và một menu các nút gồm:

* **Nút PLAY** để chuyển sang giao diện chọn nhiệm vụ.
* **Nút CONTINUE** để tiếp tục phần chơi đã lưu.
* **Nút UPGRADES** để chuyển đến giao diện nâng cấp trang bị.
* **Nút SETTINGS** để chuyển đến giao diện thiết lập.
* **Nút QUIT** để thoát game.

**4.1.1.2 Giao diện nâng cấp trang bị**

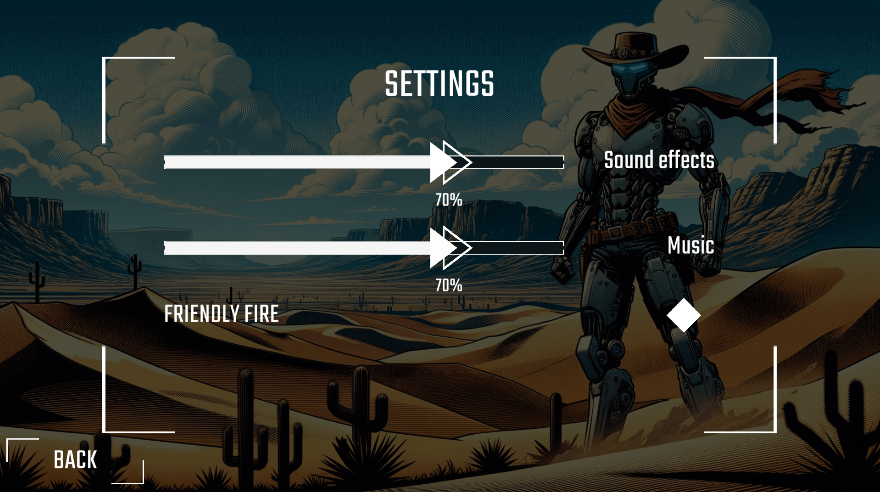


**Hình 4.2 Giao diện nâng cấp trang bị**

\_Giao diện hiện vẫn đang trong quá trình xây dựng.

\_Nhấn **nút BACK** để trở về giao diện chính.

**4.1.1.3 Giao diện thiết lập**



**Hình 4.3 Giao diện thiết lập**

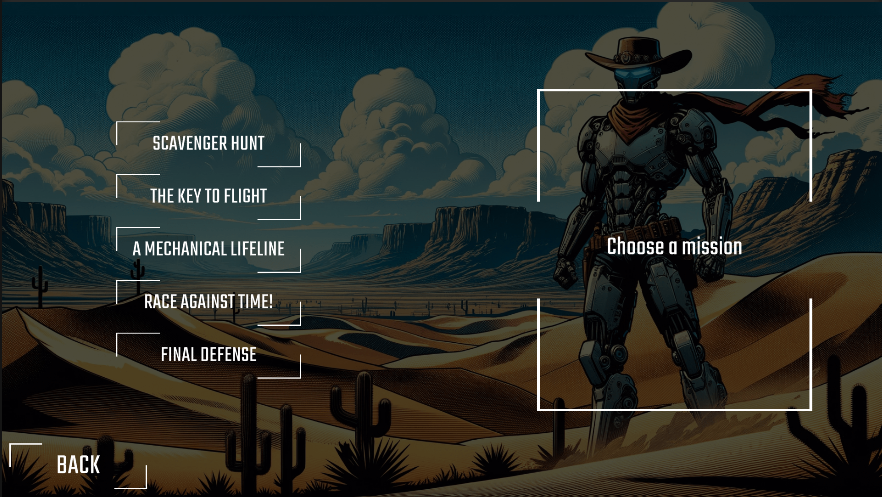
\_Sound effects: chỉnh sửa âm thanh của các hiệu ứng như (tiếng động cơ xe, tiếng va chạm khi tấn công,…).

\_Music: chỉnh sửa âm thanh nền của game.

\_FRINDLY FIRE: tích chọn thì đòn tấn công của kẻ thù sẽ không ảnh hưởng đến đồng loại của chúng.

\_Nhấn **nút BACK** để trở về giao diện chính.

**4.1.1.4 Giao diện chọn nhiệm vụ**



**Hình 4.4 Giao diện chọn nhiệm vụ**

\_SCAVENGER HUNT: nhiệm vụ tiêu diệt 12 kẻ địch ngẫu nhiên.

\_THE KEY TO FLIGHT: nhiệm vụ tìm chìa khoá.

\_A MECHANICAL LIFELINE: nhiệm vụ điều khiển xe đến đích.

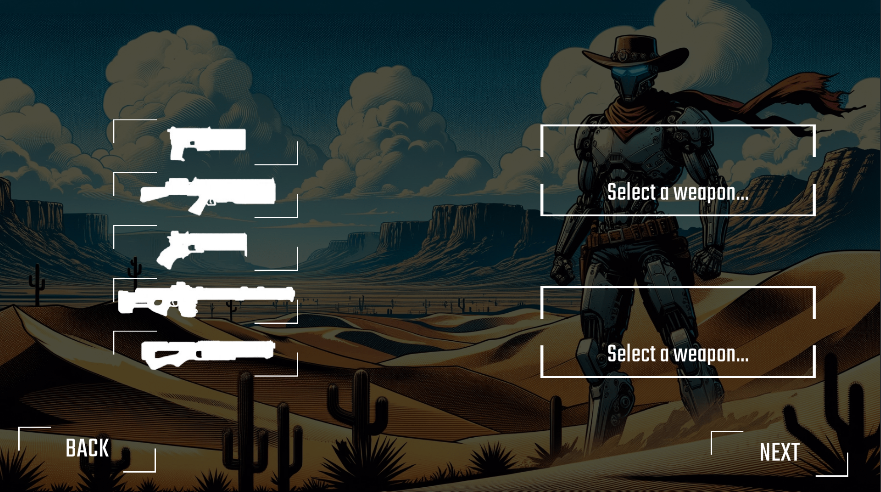
\_RACE AGAINST TIME: nhiệm vụ sống sót trong thời gian chỉ định.

\_FINAL DEFENSE: nhiệm vụ tiêu diệt tất cả kẻ địch.

\_Sau khi chọn nhiệm vụ sẽ chuyển sang giao diện chọn vũ khí.

\_Nhấn **nút BACK** để trở về giao diện chính.

**4.1.1.5 Giao diện chọn vũ khí**



**Hình 4.5 Giao diện chọn vũ khí**

\_Giới hạn chọn 2 vũ khí và có thể click vào vũ khí đã chọn lần nữa để huỷ chọn.

\_Chọn vũ khí xong nhấn **nút NEXT** để chuyển sang giao diện chờ trước khi vào game.

\_Nhấn **nút BACK** để trở về giao diện chọn nhiệm vụ.

**4.1.1.6 Giao diện chờ trước khi vào game**



**Hình 4.6 Giao diện chờ trước khi vào game**

\_Bốn bức hình sẽ hiện ra từng theo từng bức một. Sau khi hiển thị hết bốn bức hình thì **nút PLAY** sẽ xuất hiện. Click **nút PLAY** để tiến vào game.

\_Giao diện này được tạo ra nhằm tạo thời gian chờ cho các môi trường trong game được thiết lập xong.

\_Nhấn **nút MAIN MENU** để trở về giao diện chính.

**4.1.1.7 Giao diện trong game**



**Hình 4.7 Giao diện trong game**

\_Giao diện trong game gồm 1 bảng yêu cầu của nhiệm vụ, nhấn **phím H** để mở và đóng. Một thanh máu và giao diện của hai vũ khí và số đạn.

**4.1.1.8 Giao diện tạm dừng game**



**Hình 4.8 Giao diện tạm dừng game**

\_Nhấn **phím ESC** để mở giao diện tạm dừng game.

\_Nhấn **nút SAVE** để lưu game.

\_Nhấn **nút RESUME** để quay lại game.

\_Nhấn **nút MAIN MENU** để quay lại giao diện chính.

**4.1.1.9 Giao diện chiến thắng**



**Hình 4.9 Giao diện chiến thắng**

\_Giao diện chiến thắng sẽ hiển thị khi hoàn thành nhiệm vụ.

\_Nhấn **nút MAIN MENU** để trở về giao diện chính.

**4.1.1.10 Giao diện thất bại**



**Hình 4.10 Giao diện thất bại**

\_Giao diện thất bại sẽ hiển thị khi thanh máu Player bằng 0.

\_\_Nhấn **nút MAIN MENU** để trở về giao diện chính.

#### 4.1.1.11 Phím tắt

\_Các phím di chuyển:

* **Phím W:** di chuyển lên trước.
* **Phím S:** di chuyển về sau.
* **Phím A**: di chuyển qua trái.
* **Phím D:** di chuyển qua phải.

\_**Phím SHIFT** + phím di chuyển : di chuyển nhanh hơn.

\_**Phím R:** nạp đạn.

\_**Phím F:** nhặt vật phẩm.

\_**Phím G:** vứt vũ khí đang dùng.

\_**Phím ESC:** mở giao diện tạm dừng game.

### 4.1.2 Xây dựng các môi trường chi tiết

\_Môi trường trong game gồm 8 địa hình khác nhau được chia thành 3 phần:

* Phần 1: Địa hình khởi đầu, đây là nơi xuất hiện đầu tiên khi mới vào game và đây là địa hình mặc định không thể thay đổi.
* Phần 2: Gồm 4 địa hình hình vuông, 1 địa hình chữ T, 1 địa hình đường dài. Các địa hình này sẽ sắp xếp theo thứ tự ngẫu nhiên và nốt liền phần 1 và phần 3.
* Phần 3: Địa hình kết thúc, sau khi người chơi hoàn thành được nhiệm vụ và di chuyển đến địa hình này sẽ giành chiến thắng, đây là địa hình mặc định.

\_Các địa hình sẽ xuất hiện theo thứ tự phần 1, phần 2, phần 3 để tạo thành một bản đồ hoàn chỉnh.

## 4.2. Triển khai nhân vật

### 4.2.1 Nhân vật chính (Player Character)

**Hình 4.11 Nhân vật chính (Player Character)**

\_Nhân vật chính được tạo nên từ hình tượng của 1 robot theo phong cách cowboy.

\_Vũ khí chính là súng gồm 6 loại: súng lục (pistol), súng lục ổ quay (revolver), súng săn (shotgun), súng trường (rifle) và súng trường tự động (auto rifle).

\_Giới hạn vũ khí mang theo là 2.



**Hình 4.12 Các loại súng**

\_Theo thứ tự từ trái qua là: súng trường (rifle), súng lục ổ quay (revolver), súng lục (pistol), súng trường tự động (auto rifle), súng săn (shotgun).

### 4.2.2 Kẻ thù (Enemies)

#### 4.2.2.1 Kẻ thù cận chiến (Melee Enemies)

**Hình 4.13 Kẻ thù cận chiến (Melee Enemies)**

\_Kẻ thù cận chiến được thiết kế theo hình dạng một robot. Ngoại hình gồm 4 màu ngẫu nhiên và 17 viên tinh thể được tạo ngẫu nhiên trên cơ thể. Từ đó các thù được tạo ra sẽ có hình dạng khác nhau.

\_Gồm có 4 loại kẻ thù cận chiến:

* Kẻ thù bình thường: sẽ sử dụng ngẫu nhiên 1 trong 6 loại vũ khí cận chiến dành riêng cho kẻ thù.
* Kẻ thù ném rìu: sử dụng vũ khí là 1 chiếc rìu và có khả năng ném rìu gây sát thương từ xa.
* Kẻ thù khiên: giống như kẻ thù thường nhưng sử dụng thêm 1 tấm khiên, kẻ thù này có nhiều máu hơn bình thường.
* Kẻ thù tránh né: không sử dụng vũ khí nhưng lại có khả năng né đạn.

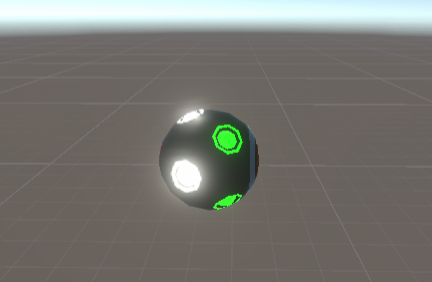
**Hình 4.14 Vũ khí của kẻ thù cận chiến**

#### 4.2.2.2 Kẻ thù tầm xa (Ranged Enemies)

**Hình 4.15 Kẻ thù tầm xa (Ranged Enemies)**

\_Ngoại hình tương tự với kẻ thù cận chiến nhưng lại sử dụng vũ khí chính là súng.

\_ Gồm có 4 loại

* Kẻ thù ẩn nấp: sử dụng vũ khí giống với nhân vật chính, vũ khí sẽ ngẫu nhiên trừ súng trường và súng săn. Có thể thay đổi vị trí ẩn nấp 1 lần.
* Kẻ thù ẩn nấp sử dụng lựu đạn: tương tự với kẻ thù ẩn nấp nhưng có khả năng ném lựu đạn gây sát thương diện tích rộng.
* Kẻ thù thay đổi chỗ ẩn nấp sử dụng lựu đạn: tương tự với kẻ thù ẩn nấp sử dụng lựu đạn nhưng có thể thay đổi chỗ ẩn nấp nhiều lần.
* Kẻ thù súng trường: vũ khí sử dụng là súng trường, tấn công tầm rất xa và có thể thay đổi chỗ ẩn nấp nhiều lần.
* Kẻ thù không thể ngăn cản: vũ khí sử dụng là súng săn, có rất nhiều máu.

**Hình 4.16 Lựu đạn**

#### 4.2.2.3 Boss

\_Là người máy với ngoại hình to lớn và có rất nhiều máu.

\_Gồm 2 loại boss:

* Boss cầm búa: vũ khí sử dụng là 1 chiếc búa khổng lồ. Có khả năng nhảy đến vị người chơi từ xa để gây sát thương và khả năng vung ra búa sấm sét gây sát thương lớn.



**Hình 4.17 Boss cầm búa**

* Boss phun lửa: vũ khí sử dụng là 2 nắm tay bằng kim loại khổng lồ. Có khả năng nhảy đến vị người chơi từ xa để gây sát thương và khả năng phun lửa gây sát thương lớn trên diện rộng.



**Hình 4.18 Boss phun lửa**

## 4.3. Triển khai cốt truyện và nhiệm vụ

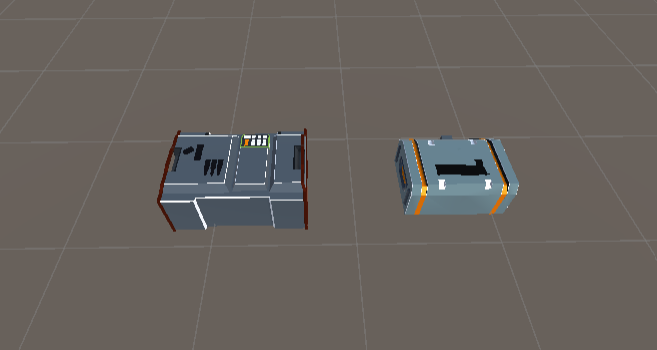
\_Từ cốt truyện và nhiệm vụ đã nêu ở chương 3 mục 3.4.1 triển khai thành 6 nhiệm vụ tương ứng với 6 màn khác nhau sau:

* Nhiệm vụ đi săn (Scavenger Hunt): Săn lùng 12 kẻ thù được trang bị thiết bị gây rối tín hiệu. Những thiết bị này đang gây nhiễu hệ thống radar của máy bay, cản trở việc cất cánh an toàn. Loại bỏ mối đe dọa và dọn đường cho lối thoát của bạn.
* Nhiệm vụ tìm chìa khoá (The Key To Flight): Bạn phải xác định vị trí và đánh bại kẻ thù giữ chìa khóa để lấy chìa khóa. Hãy thận trọng vì kẻ thù này có thể được bảo vệ nghiêm ngặt.
* Nhiệm vụ giao xe (A Mechanical Lifeline): Bạn cần tìm một chiếc xe còn hoạt động và đưa nó đến nơi sơ tán. Các bộ phận của xe có thể được sử dụng để sửa chữa máy bay sơ tán.
* Nhiệm vụ chạy đua với thời gian (Race Against Time): Máy quét của bạn đã phát hiện ra cơn bão cát sắp tới. Bạn có chưa đầy 15 phút để đến máy bay trước khi các điều kiện trở nên không đảm bảo để cất cánh an toàn.
* Nhiệm vụ phòng thủ cuối cùng (Final Defence): Đến điểm thoát và bảo vệ máy bay cho đến khi nó sẵn sàng cất cánh. Trong thời gian này kẻ thù sẽ liên tục xuất hiện và tấn công dồn dập.

## 4.4. Triển khai vật phẩm và vũ khí

\_Vật phẩm được sử dụng trong game gồm:

* Hộp đạn: gồm 2 loại là hộp đạn lớn cung cấp nhiều đạn và hộp đạn nhỏ cung cấp ít đạn.



**Hình 4.19 Hộp đạn**

* Sáu loại súng như đã nêu ở phần 4.2.1:
* Súng lục (pistol): trang bị nhanh, tấn công nhanh, tiêu diệt kẻ thù nhanh.
* Súng lục ống quay (revolver): có khả năng trong 1 phát bắn tiêu diệt kẻ thù.
* Súng trường (rifle): tầm bắn rất xa.
* Súng trường tự động (auto rifle): có khả năng nhả đạn liên tục.
* Súng săn (shotgun): nhả nhiều đạn trong 1 lần bắn, sát thương tầm gần cao.

\_Để cân bằng cho game thì các loại đã được thêm vào các thuộc tính như: tốc độ nạp đạn, tốc độ trang bị, tầm bắn và số lượng đạn phù hợp, ví dụ súng trường và súng lục ống quay có sát thương cao nhất nên số đạn là ít nhất. Ngoài ra còn bổ sung thêm thuộc tính độ lệch của đạn khi bắn cho tất cả vũ khí trừ súng trường để tăng thêm phần thử thách cho người chơi.

\_Vũ khí mà kẻ thù tầm xa (ranged enemies) sử dụng giống với người chơi nên chức năng của vũ khí cũng tương tự. Sự khác nằm ở việc kẻ thù tầm xa (ranged enemies) chỉ có thể mang theo tối đa 1 vũ khí còn người chơi là 2.

# KẾT LUẬN VÀ KIẾN NGHỊ

## Kết quả nghiên cứu

**Có thêm nhiều kiến thức và tư duy về lĩnh vực game.** Nghiên cứu về Unity Engine và quá trình phát triển game hành động bắn súng 3D đã mang lại cho em nhiều kiến thức sâu rộng về công nghệ game và hiểu rõ hơn về các công cụ và quy trình tạo ra một sản phẩm game hoàn chỉnh, từ khâu thiết kế, lập trình đến tối ưu hóa. Em cũng học được cách quản lý tài nguyên, xử lý đồ họa, và áp dụng các kỹ thuật tiên tiến trong việc tạo hiệu ứng hình ảnh và âm thanh. Những hiểu biết này không chỉ giới hạn trong khuôn khổ của Unity mà còn có thể áp dụng cho các công cụ và nền tảng phát triển game khác.

**Một sản phẩm game hành động bắn súng 3D với đồ họa và hiệu ứng đẹp mắt, có khả năng đáp ứng nhu cầu giải trí cao.** Kết quả của dự án là một trò chơi bắn súng 3D, hoàn thiện với chất lượng đồ họa cao, hiệu ứng hình ảnh và âm thanh chân thực. Trò chơi không chỉ đảm bảo các yếu tố kỹ thuật mà còn được thiết kế với lối chơi hấp dẫn, cơ chế chiến đấu đa dạng, và các tính năng hỗ trợ người chơi. Đồ họa sống động với ánh sáng, bóng đổ, và hiệu ứng hạt đã được tinh chỉnh để tạo ra không gian trò chơi lôi cuốn và hấp dẫn. Game cung cấp một trải nghiệm giải trí phù hợp với xu hướng và nhu cầu của người chơi hiện đại, đáp ứng được mong đợi về một sản phẩm game hành động chất lượng tốt.

**Tích lũy kinh nghiệm thực tiễn và cải tiến kỹ năng phát triển game.** Qua quá trình nghiên cứu và phát triển, em đã tích lũy được nhiều kinh nghiệm quý báu về quản lý dự án game, phối hợp giữa các thành viên trong nhóm, và giải quyết các vấn đề kỹ thuật phát sinh. Kỹ năng lập trình, thiết kế 3D, và xử lý đồ họa đã được nâng cao đáng kể, cùng với khả năng sử dụng các công cụ hỗ trợ của Unity một cách hiệu quả. Những kinh nghiệm này là nền tảng vững chắc cho việc phát triển các dự án game phức tạp hơn trong tương lai.

## Những hạn chế

Dù đạt được nhiều kết quả tích cực, nghiên cứu về Unity Engine và ứng dụng xây dựng game hành động bắn súng 3D vẫn tồn tại một số hạn chế đáng kể. Đầu tiên, việc tối ưu hóa hiệu năng trên các thiết bị có cấu hình thấp gặp nhiều khó khăn, do Unity Engine có xu hướng yêu cầu tài nguyên cao, đặc biệt với các trò chơi 3D phức tạp và đồ họa chất lượng cao. Thứ hai, quá trình phát triển gặp thách thức trong việc tạo ra trí tuệ nhân tạo (AI) cho các đối thủ, vì việc lập trình hành vi AI phức tạp đòi hỏi nhiều thời gian và kiến thức chuyên sâu. Ngoài ra, khả năng mở rộng và cập nhật tính năng mới đôi khi bị giới hạn bởi hiệu suất công cụ và yêu cầu duy trì tính tương thích ngược. Về khía cạnh quản lý dự án, khó khăn trong việc phối hợp và quản lý các thành phần khác nhau của trò chơi đã làm gia tăng chi phí và thời gian phát triển. Cuối cùng, sự phụ thuộc vào Unity Asset Store có thể gây ra hạn chế trong sáng tạo và dễ dẫn đến việc sử dụng lại các tài nguyên phổ biến, làm giảm tính độc đáo của trò chơi.

## Hướng phát triển

Để tiếp tục nâng cao chất lượng và khả năng cạnh tranh của game hành động bắn súng 3D phát triển bằng Unity Engine, hướng phát triển trong tương lai sẽ tập trung vào một số yếu tố chính. Thứ nhất, việc cải thiện trí tuệ nhân tạo (AI) sẽ được ưu tiên nhằm tạo ra các đối thủ có hành vi tự nhiên và thử thách hơn, thông qua áp dụng các kỹ thuật học máy và mô hình hành vi phức tạp. Thứ hai, mở rộng tính năng chơi đa người trực tuyến sẽ là một bước tiến quan trọng, bao gồm tích hợp chế độ chơi nhiều người, hỗ trợ cross-platform, và các yếu tố xã hội như bảng xếp hạng và thành tích. Ngoài ra, việc tối ưu hóa hiệu năng và dung lượng để phù hợp với nhiều cấu hình thiết bị, đặc biệt là những máy có hiệu suất thấp, sẽ giúp trò chơi tiếp cận được đông đảo người chơi hơn. Tăng cường chất lượng đồ họa và âm thanh thông qua sử dụng công nghệ ray tracing và âm thanh không gian cũng là hướng phát triển để nâng cao trải nghiệm người dùng. Cuối cùng, việc triển khai hệ thống cập nhật linh hoạt và hỗ trợ lâu dài sẽ đảm bảo trò chơi luôn mới mẻ và phù hợp với các xu hướng công nghệ mới, đồng thời giúp nhanh chóng giải quyết các vấn đề phát sinh trong quá trình vận hành.

# TÀI LIỆU THAM KHẢO

[1]. *GAMBA Team* tổng hợp, <https://gambaru.io/en/blog/tong-quan-nganh-game-lich-su-xu-huong-phat-trien>, truy cập ngày 1 tháng 6 năm 2024.

[2]. Computer 24h, <https://lapdatquangame.vn/tin-cong-nghe/lich-su-phat-trien-cua-nganh-cong-nghiep-game.html>, truy cập ngày 2 tháng 6 năm 2024.

[3]. Hoàng Hiệp, <https://viblo.asia/p/tim-hieu-tong-quan-ve-3d-va-the-gioi-quanh-no-ORNZqxybK0n>, truy cập ngày 3 tháng 6 năm 2024.

[4]. Wikipedia, <https://vi.wikipedia.org/wiki/C_Sharp_(ng%C3%B4n_ng%E1%BB%AF_l%E1%BA%ADp_tr%C3%ACnh)>, truy cập ngày 3 tháng 6 năm 2024.

[5] Trịnh Đức Huy, <https://itviec.com/blog/unity-la-gi/>, truy cập ngày 4 tháng 6 năm 2024.

[6]. Hải Phạm, <https://fptshop.com.vn/tin-tuc/danh-gia/microsoft-visual-studio-la-phan-mem-gi-173485>, truy cập ngày 5 tháng 6 năm 2024.

[7]. FPT Arena Multimedia, <https://arena.fpt.edu.vn/cac-buoc-thiet-ke-game-tu-a-den-z/>, truy cập ngày 6 tháng 6 năm 2024.

[8]. Unity Forum, <https://forum.unity.com/threads/making-characters.1398646/>, truy cập ngày 6 tháng 6 năm 2024.

[9]. Dustin Tyler, <https://www.gamedesigning.org/learn/game-plot/>, truy cập ngày 7 tháng 6 năm 2024.

[10]. Hitmarket, <https://hitmarker.net/mission-design-jobs-in-the-video-game-industry-the-ultimate-quick-guide>, truy cập ngày 7 tháng 6 năm 2024.

[11]. Room 8 Studio, <https://room8studio.com/news/the-secrets-of-weapon-design-for-video-games/>, truy cập ngày 8 tháng 6 năm 2024.