

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**⋅⋅⋅⋅⋅**🙣🕮🙡**⋅⋅⋅⋅**

**BÁO CÁO THỰC NGHIỆM THUỘC HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG GAME**

**XÂY DỰNG GAME 2D “BẮN MÁY BAY”**

**GVHD : Ths. Vũ Đức Huy**

**Nhóm : 09**

**Sinh Viên : Nguyễn Văn Quý**

**Phùng Văn Trường**

**Trần Hoàng Minh Quang**

**Nguyễn Đức Doanh**

**Lớp: 20241IT6028003 Khóa: K16**

*Hà Nội - 12/2024*

**MỤC LỤC**

MỞ ĐẦU 5

1. Tên đề tài 5

2. Nội dung nghiên cứu 5

2.1. Kiến thức 5

2.2. Kỹ năng 6

CHƯƠNG 1. THIẾT KẾ Ý TƯỞNG GAME 8

1.1. Giới thiệu 8

1.2. Thể loại game 8

1.3. Tóm tắt game 8

1.4. Khách hàng mục tiêu 9

1.5. Điểm mạnh của game 9

1.6. Phong cách nghệ thuật game 9

1.7. Thiết bị trải nghiệm game 10

CHƯƠNG 2. THIẾT KẾ PHÁT TRIỂN GAME 12

2.1. Thiết kế kịch bản game 12

2.1.1. Cách chơi chính 12

2.1.2. Cốt truyện của game 12

2.1.3. Các phần tử của game 13

2.1.4. Các cơ chế của game 15

2.2. Thiết kế giao diện 16

2.2.1. Biểu đồ - Flowchart 16

2.2.1.1. Bắt đầu game 16

2.2.1.2. Tạm dừng game 17

2.2.1.3. Chọn chế độ 18

2.2.1.4. Điều khiển nhân vật 19

2.2.2. Mô tả 19

2.2.3. Giao diện các màn hình 22

2.2.3.1. Màn hình thiết kế GamePlay và GamePlayEasy 22

2.2.3.2. Màn hình thiết kế Game Over Panel 23

2.2.3.3. Màn hình thiết kế MainMenu 24

2.2.3.4. Màn hình thiết kế Mode Panel 24

2.3. Thiết kế âm thanh 25

2.3.1. Âm thanh music\_background 25

2.3.2. Âm thanh S2157 và S364 25

2.3.3. Âm thanh explosion\_enemy 25

2.3.4. Âm thanh explosion\_player 25

2.3.5. Âm thanh explosion\_asteroid 25

2.3.6. Âm thanh weapon\_enemy 25

2.3.7. Âm thanh weapon\_player 25

CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ 26

3.1. Cài đặt chương trình 26

3.1.1. Máy bay người chơi 26

3.1.2. Xử lý di chuyển 26

3.1.3. Xử lý va chạm với các vật thể khác 27

3.1.4. Máy bay kẻ thù 28

3.1.5. Máy bay lao vào 32

3.1.6. Máy bay boss 34

3.1.7. Viên đạn máy bay người chơi 38

3.1.8. Viên đạn máy bay địch 40

3.1.9. Hình nền 43

3.1.10. Xử lý âm thanh 44

3.2. Một số giao diện trò chơi 48

3.2.1. Màn hình chính 48

3.2.2. Giao diện chế độ dễ 48

3.2.3. Giao diện chế độ khó 49

3.2.4. Giao diện tạm dừng game 50

3.2.5. Giao diện kết thúc game 50

KẾT LUẬN 52

1. Kết quả đạt được 52

2. Hướng phát triển 52

TÀI LIỆU THAM KHẢO 53

**DANH MỤC HÌNH ẢNH**

# MỞ ĐẦU

## 1. Tên đề tài

Xây dựng game 2D “Bắn máy bay”

## 2. Nội dung nghiên cứu

### 2.1. Kiến thức

* Lập trình trên Unity:
* Sử dụng ngôn ngữ C# để viết các kịch bản (scripts): C# là ngôn ngữ lập trình chính được sử dụng trong Unity. Việc viết các kịch bản bằng C# giúp điều khiển hành vi của máy bay, kẻ thù, và các đối tượng khác trong game. Các kịch bản này bao gồm việc xử lý chuyển động, va chạm, và các hành động khác của các đối tượng trong game.
* Tìm hiểu cách sử dụng Unity Physics 2D: Unity cung cấp hệ thống vật lý 2D mạnh mẽ để xử lý va chạm và các hiệu ứng vật lý khác. Việc sử dụng Unity Physics 2D giúp tạo ra các tương tác chân thực giữa các đối tượng trong game, chẳng hạn như đạn bắn trúng máy bay địch hoặc máy bay va chạm với chướng ngại vật.
* Ứng dụng Unity Input System: Unity Input System cho phép lập trình viên dễ dàng điều khiển các đối tượng trong game thông qua các thiết bị đầu vào như bàn phím, chuột, hoặc gamepad. Việc sử dụng Unity Input System giúp người chơi có thể điều khiển máy bay một cách mượt mà và chính xác.
* Quản lý tài nguyên game:
* Xử lý và tích hợp các tài nguyên 2D: Các tài nguyên 2D như sprite máy bay, đạn, kẻ thù và hiệu ứng nổ (explosions) cần được xử lý và tích hợp vào game. Việc này bao gồm việc cắt và xử lý sprite, tạo các hoạt ảnh (animation) và tích hợp chúng vào các kịch bản game.
* Thiết kế giao diện người dùng (UI): Giao diện người dùng là một phần quan trọng của game, giúp người chơi tương tác với game một cách dễ dàng. Việc thiết kế UI bao gồm việc tạo bảng điểm, thanh máu, màn hình kết thúc game và các yếu tố UI khác để cung cấp thông tin cho người chơi và cải thiện trải nghiệm chơi game.
* Logic game:
* Cấu trúc game theo các trạng thái chính: Game cần được cấu trúc theo các trạng thái chính như bắt đầu, chơi, và kết thúc. Việc này giúp quản lý luồng chơi game một cách rõ ràng và dễ dàng điều chỉnh.
* Tính toán và hiển thị điểm số: Điểm số của người chơi được tính toán dựa trên số lượng kẻ thù bị tiêu diệt và các yếu tố khác. Việc hiển thị điểm số giúp người chơi theo dõi tiến trình của mình và tạo động lực để cải thiện kỹ năng chơi game.
* Xây dựng hệ thống sinh kẻ thù (enemy spawning): Hệ thống sinh kẻ thù giúp tạo ra các kẻ thù mới trong game theo thời gian, tăng dần độ khó để thử thách người chơi. Việc này bao gồm việc xác định vị trí, thời gian và loại kẻ thù xuất hiện.

### 2.2. Kỹ năng

* Kỹ năng lập trình:
* Viết mã sạch (clean code): Viết mã sạch giúp đảm bảo game hoạt động mượt mà, dễ bảo trì và mở rộng. Điều này bao gồm việc sử dụng các biến và hàm có tên rõ ràng, tổ chức mã theo cấu trúc logic và tuân thủ các nguyên tắc lập trình tốt.
* Sử dụng các mẫu thiết kế (design patterns): Các mẫu thiết kế như Singleton hoặc State Machine giúp quản lý các đối tượng và trạng thái trong game một cách hiệu quả. Việc sử dụng các mẫu thiết kế giúp cải thiện cấu trúc mã và dễ dàng mở rộng tính năng của game.
* Kỹ năng đồ họa:
* Sử dụng Unity Sprite Editor: Unity Sprite Editor là công cụ mạnh mẽ để cắt và xử lý sprite. Việc sử dụng Unity Sprite Editor giúp tạo ra các sprite chất lượng cao và tối ưu hóa chúng cho game.
* Thiết kế hoạt ảnh đơn giản (animation): Hoạt ảnh giúp tạo ra các chuyển động mượt mà và sống động cho các đối tượng trong game. Việc thiết kế hoạt ảnh bao gồm việc tạo các khung hình (frames) và sắp xếp chúng theo thứ tự để tạo ra các chuyển động liên tục.
* Kỹ năng tối ưu hóa:
* Sử dụng các công cụ của Unity như Profiler: Unity Profiler là công cụ hữu ích để kiểm tra hiệu suất game. Việc sử dụng Unity Profiler giúp phát hiện các vấn đề về hiệu suất và tối ưu hóa mã để cải thiện trải nghiệm chơi game.
* Tối ưu hóa việc quản lý tài nguyên: Quản lý tài nguyên hiệu quả giúp giảm thiểu số lượng đối tượng hoạt động trên màn hình cùng lúc, từ đó cải thiện hiệu suất game. Việc này bao gồm việc sử dụng các kỹ thuật như pooling để quản lý các đối tượng tái sử dụng.
* Kỹ năng làm việc nhóm:
* Sử dụng Unity Collaborate hoặc các công cụ quản lý mã nguồn như Git: Unity Collaborate và Git là các công cụ hữu ích để phối hợp với nhóm phát triển. Việc sử dụng các công cụ này giúp quản lý mã nguồn, theo dõi các thay đổi và làm việc cùng nhau một cách hiệu quả.
* Phân chia công việc giữa các thành viên: Làm việc nhóm hiệu quả đòi hỏi sự phân chia công việc rõ ràng giữa các thành viên. Điều này bao gồm việc xác định vai trò và trách nhiệm của từng thành viên, như thiết kế đồ họa, lập trình, kiểm thử và hoàn thiện sản phẩm.

# CHƯƠNG 1. THIẾT KẾ Ý TƯỞNG GAME

## 1.1. Giới thiệu

Game "Bắn máy bay 2D" là một trò chơi giải trí đơn giản nhưng đầy thách thức, mang đến trải nghiệm sống động cho người chơi. Trò chơi được thiết kế với mục tiêu tạo ra một sản phẩm dễ tiếp cận nhưng vẫn giữ được tính hấp dẫn và kịch tính. Với Unity, game tận dụng công cụ mạnh mẽ để xây dựng giao diện trực quan, cơ chế điều khiển mượt mà và hiệu ứng đồ họa sinh động.

Unity là một nền tảng phát triển game mạnh mẽ và phổ biến, cung cấp môi trường lý tưởng để tạo ra các trò chơi 2D và 3D. Với khả năng hỗ trợ đa nền tảng và thư viện công cụ phong phú, Unity giúp lập trình viên dễ dàng phát triển các trò chơi chất lượng cao. Dựa trên nền tảng này, game “Bắn máy bay” được thiết kế với mục tiêu mang đến trải nghiệm giải trí hấp dẫn, đồng thời áp dụng các kiến thức lập trình và đồ họa vào thực tiễn.

Trò chơi lấy bối cảnh một cuộc chiến ngoài vũ trụ, nơi người chơi điều khiển máy bay để tiêu diệt kẻ thù và né tránh các đợt tấn công của máy bay địch. Game được xây dựng với phong cách đồ họa đơn giản nhưng bắt mắt, cùng lối chơi dễ tiếp cận nhưng không kém phần thử thách. Người chơi sẽ trải qua 2 chế độ chơi là dễ và khó với các thử thách của từng chế độ và phải sử dụng kỹ năng điều khiển máy bay một cách khéo léo để sống sót và đạt điểm cao.

## 1.2. Thể loại game

“Bắn máy bay 2D”thuộc thể loại game hành động (action) kết hợp yếu tố bắn súng arcade cổ điển. Người chơi điều khiển một chiếc máy bay chiến đấu, tiêu diệt kẻ thù và né tránh các chướng ngại vật để giành điểm cao. Thể loại này mang đến sự kịch tính và hấp dẫn, thu hút người chơi bởi lối chơi nhanh gọn và đầy thử thách.

## 1.3. Tóm tắt game

Game có 2 chế độ dễ và khó. Người chơi bắt đầu game với một chiếc máy bay được trang bị súng. Kẻ thù xuất hiện theo từng đợt và tấn công người chơi. Điểm số sẽ được tính dựa trên số lượng kẻ thù bị tiêu diệt. Game kết thúc khi máy bay của người chơi bị trúng đạn hoặc va chạm với máy bay địch quá nhiều.

Trong quá trình chơi, người chơi sẽ phải điều khiển máy bay một cách khéo léo để né tránh các đợt tấn công của kẻ thù và tiêu diệt chúng. Kẻ thù sẽ xuất hiện từ trên xuống theo vị trí ngẫu nhiên, tạo sự đa dạng và thách thức cho người chơi.

## 1.4. Khách hàng mục tiêu

Độ tuổi: Từ 10 tuổi trở lên, bao gồm cả trẻ em, thanh thiếu niên và người lớn yêu thích thể loại game arcade.

Mục tiêu: Phù hợp với người chơi muốn tìm kiếm trải nghiệm giải trí ngắn hạn, đơn giản nhưng đầy kịch tính.

Địa lý: Hướng đến thị trường toàn cầu nhờ tính đơn giản và phổ biến của game 2D.

## 1.5. Điểm mạnh của game

Game “Bắn máy bay 2D” có nhiều điểm mạnh giúp thu hút người chơi và giữ chân họ lâu dài:

- Cốt truyện đơn giản: Không yêu cầu người chơi phải đầu tư thời gian để tìm hiểu, dễ dàng chơi ngay.

- Đồ họa: Sử dụng phong các 2D tối giản nhưng sinh động, với hiệu ứng bắn súng, nổ và chuyển động mượt mà.

- Cách chơi: Lối chơi nhanh gọn hấp dẫn, phù hợp với cả người chơi mới và người chơi có kinh nghiệm.

- Tính thử thách: Độ khó thay đổi qua hai chế độ khó và dễ, giúp duy trì hứng thú và thúc đẩy người chơi quay lại để cải thiện thành tích.

## 1.6. Phong cách nghệ thuật game

Game "Bắn máy bay 2D" áp dụng phong cách nghệ thuật khoa học viễn tưởng với bối cảnh không gian rộng lớn và bí ẩn. Thiết kế tập trung vào việc tạo ra một môi trường sống động và gây ấn tượng mạnh về thị giác, đồng thời vẫn giữ được sự tối giản để duy trì hiệu suất mượt mà.

* Màu sắc và bối cảnh:
* Nền game mang màu sắc tối (đen, xanh đậm) để tái hiện không gian vũ trụ sâu thẳm, làm nổi bật các vật thể như máy bay, đạn và kẻ thù.
* Hiệu ứng ánh sáng (light effects) được sử dụng để tạo cảm giác năng động, chẳng hạn như ánh sáng từ vụ nổ, tia bắn đạn, và phản chiếu từ các đối tượng đang di chuyển.
* Thiết kế đối tượng:
* Máy bay của người chơi: Mang hình dạng hiện đại với các chi tiết gọn gàng, dễ nhận biết, và có màu sắc sáng như xanh lam để nổi bật trên nền tối.
* Kẻ thù: Các máy bay địch được thiết kế với vẻ ngoài hung dữ và màu sắc tương phản (ví dụ: đỏ, xám) để thể hiện sự nguy hiểm.
* Hiệu ứng đặc biệt:
* Hiệu ứng nổ: Sử dụng các hoạt ảnh sống động và nhiều tia sáng để tạo cảm giác chân thực khi máy bay hoặc đạn va chạm.
* Hiệu ứng bắn: Mỗi viên đạn phát ra ánh sáng mờ nhạt khi di chuyển, tạo ấn tượng rõ nét về đường bay.
* Chuyển động của nền: Các ngôi sao hoặc ánh sáng trong không gian di chuyển chậm, tạo cảm giác chiều sâu và tăng tính chân thực cho bối cảnh không gian.
* Âm thanh và giao diện:
* Kết hợp với âm nhạc và hiệu ứng âm thanh đậm chất viễn tưởng, bao gồm âm thanh của đạn bắn, vụ nổ và các cảnh báo trong game.
* Giao diện người dùng (UI) được thiết kế đơn giản nhưng trực quan, sử dụng tông màu xanh neon để phù hợp với phong cách tương lai.

## 1.7. Thiết bị trải nghiệm game

Game được thiết kế để chạy trên các thiết bị sau:

- Máy tính: Windows, macOS (sử dụng bàn phím để điều khiển)

- Điện thoại di dộng: Android và iOS (sử dụng cảm ứng để điều khiển)

# CHƯƠNG 2. THIẾT KẾ PHÁT TRIỂN GAME

## 2.1. Thiết kế kịch bản game

### 2.1.1. Cách chơi chính

Người chơi sẽ sử dụng chuột và bàn phím để điều khiển trò chơi. Khi vào giao diện chính của trò chơi, người chơi sẽ chọn chế độ trong hai chế độ dễ và khó để chơi. Người chơi sẽ sử dụng các phím “A W D S” hoặc các phím lên xuống trái phải từ bàn phím để di chuyển nhân vật và sử dụng phím space để bắn. Trong khi chơi người chơi có thể chọn bật tắt âm thanh hoặc bấm tạm dừng chò chơi. Khi tạm dừng người chơi có thể chọn tiếp tục hoặc bắt đầu lại từ đầu trò chơi.

### 2.1.2. Cốt truyện của game

Vào năm 2157, Trái Đất đã đạt đến đỉnh cao của khoa học và công nghệ. Tuy nhiên, nguồn tài nguyên thiên nhiên đã cạn kiệt, buộc loài người phải tìm kiếm sự sống và tài nguyên ngoài không gian. Trong quá trình khám phá các hành tinh mới, con người vô tình kích hoạt một tín hiệu cổ đại, đánh thức một nền văn minh ngoài hành tinh bị lãng quên – Đội quân Shadow Fleet.

Shadow Fleet, vốn là những chiến binh máy móc được lập trình để bảo vệ kho báu và nguồn tài nguyên quý giá của dải ngân hà, nay coi loài người là mối đe dọa. Chúng phát động một cuộc tấn công toàn diện vào các con tàu thăm dò và căn cứ của loài người.

Bạn vào vai Captain Nova, một phi công xuất sắc của Lực lượng Phòng thủ Không gian Trái Đất (Earth Defense Force - EDF). Được giao nhiệm vụ lái chiếc máy bay chiến đấu tiên tiến nhất – **Starblade X** để tiêu diệt máy bay địch, bạn phải bảo vệ Trái Đất khỏi sự xâm lược của Shadow Fleet.

Nếu bạn chiến thắng bạn sẽ trở thành anh hùng của Trái Đất, nhưng cũng phát hiện ra một tín hiệu bí ẩn từ dải ngân hà xa xôi, ám chỉ rằng nguy hiểm vẫn còn tiềm tàng. Còn nếu bạn thua, Shadow Fleet sẽ tiến hành cuộc tấn công bùng nổ vào Trái Đất dẫn đến sự diệt vong của nền văn minh nhân loại.

### 2.1.3. Các phần tử của game

Game bắn máy bay 2D được xây dựng với các đối tượng như máy bay người chơi, máy bay kẻ thù, các máy bay lao vào, viên đạn từ máy bay người chơi và máy bay kẻ thù. Các đối tượng này có các thuộc tính và hành vi riêng biệt:

* Máy bay người chơi: Di chuyển lên xuống, qua lại trên màn hình. Người chơi điều khiển máy bay này bằng các phím mũi tên và bắn đạn bằng phím spacebar.
* Máy bay kẻ thù: Xuất hiện ngẫu nhiên ở phía trên màn hình và di chuyển xuống dưới với tốc độ khác nhau tùy thuộc vào điểm số của người chơi. Tốc độ và tần suất xuất hiện của chúng tăng lên khi điểm số của người chơi cao hơn.
* Máy bay lao vào: Cũng di chuyển từ trên xuống, nhưng không bắn đạn mà lao thẳng máy bay vào người chơi.
* Viên đạn máy bay người chơi: Sẽ được bắn ra từ máy bay người chơi với tốc độ cố định và số lần bắn ra từ máy bay sẽ được điều khiển bởi người chơi qua phím spacebar.
* Viên đạn máy bay kẻ thù: Được bắn ra từ máy bay kẻ thù với số lần và tốc độ tùy thuộc vào điểm số của người chơi.
* Âm thanh: Gồm có các tiếng bắn đạn của máy bay, máy bay nổ, nhạc nền màn hình menu, nhạc nền màn hình trò chơi.

Mô tả chi tiết các phần tử của game:

* Máy bay người chơi:

+ Tính chất: Là nhân vật chính do người chơi điều khiển.

+ Hoạt động: Di chuyển theo các hướng và bắn đạn.

+ Ảnh hưởng: Bị phá hủy khi va chạm với máy bay địch, đạn của địch hoặc máy bay lao vào.

+ Hình ảnh đồ họa: Hình ảnh máy bay được thiết kế bằng phần mềm đồ họa Photoshop.

* Máy bay kẻ thù:

+ Tính chất: Là kẻ địch của người chơi.

+ Hoạt động: Di chuyển từ trên xuống, bắn đạn.

+ Ảnh hưởng: Bị phá hủy khi trúng đạn của người chơi.

+ Hình ảnh đồ họa: Hình ảnh máy bay với thiết kế khác với máy bay người chơi.

* Viên đạn:

+ Tính chất: Là vũ khí tấn công của máy bay người chơi và kẻ thù.

+ Hoạt động: Di chuyển theo hướng bắn.

+ Ảnh hưởng: Gây sát thương hoặc phá hủy đối tượng va chạm.

+ Hình ảnh đồ họa: Hình ảnh viên đạn laser ánh sáng ngắn hình chữ nhật.

* Máy bay boss:

+ Tính chất: Là một loại máy bay địch đặc biệt, mạnh hơn và to hơn các máy bay địch thông thường, xuất hiện khi người chơi đạt đến một số điểm nhất định.

+ Hoạt động: Di chuyển theo một quỹ đạo riêng và bắn ra nhiều đạn.

+ Ảnh hưởng: Nếu người chơi va chạm với máy bay boss hoặc bị đạn của nó bắn trúng sẽ bị chết.

+ Hình ảnh đồ họa: Hình ảnh máy bay với thiết kế lớn hơn và phức tạp hơn các máy bay khác.

* Âm thanh:

+ Tính chất: Gồm các tiếng bắn đạn, máy bay nổ, nhạc nền.

+ Hoạt động: Phát khi có sự kiện tương ứng xảy ra.

+ Ảnh hưởng: Tạo hiệu ứng âm thanh, tăng trải nghiệm cho người chơi.

* Các yếu tố khác:

+ Màn chơi: Bối cảnh trong game có thể thay đổi tùy theo chế độ chơi hoặc màn chơi.

+ Điểm số: Thể hiện thành tích của người chơi.

+ Giao diện: Menu, màn hình game over.

+ Nền game: Hình ảnh nền di chuyển tạo cảm giác chuyển động bay cho máy bay.

### 2.1.4. Các cơ chế của game

* Cơ chế điều khiển: Người chơi sử dụng các phím mũi tên để di chuyển máy bay và phím spacebar để bắn. Các phím mũi tên di chuyển máy bay theo chiều ngang và chiều dọc.
* Cơ chế tấn công: Máy bay người chơi bắn ra đạn để tiêu diệt máy bay địch. Máy bay địch cũng bắn đạn vào máy bay người chơi.
* Cơ chế sinh địch: Máy bay địch xuất hiện ngẫu nhiên ở phía trên màn hình. Tần suất xuất hiện và tốc độ di chuyển của chúng tăng lên khi điểm số của người chơi cao hơn.
* Cơ chế va chạm: Khi máy bay người chơi va chạm với máy bay địch, đạn của địch hoặc máy bay lao vào, trò chơi sẽ kết thúc.
* Cơ chế tính điểm: Người chơi được cộng điểm khi bắn trúng máy bay địch hoặc ăn được vàng. Điểm số cao hơn sẽ làm tăng độ khó của trò chơi.
* Cơ chế âm thanh: Âm thanh được phát ra khí bắn đạn, máy bay nổ, hoặc khi có sự kiện khác xảy ra.
* Ví dụ về cơ chế phức tạp:

+ Khi người chơi đạt đến một số điểm nhất định (vd: 50 điểm), sẽ có một loại máy bay địch đặc biệt (vd: Máy bay boss) xuất hiện. Máy bay boss này có thể bắn ra nhiều đạn hơn, di chuyển nhanh hơn và có lượng máu lớn hơn so với các loại máy bay địch thông thường.

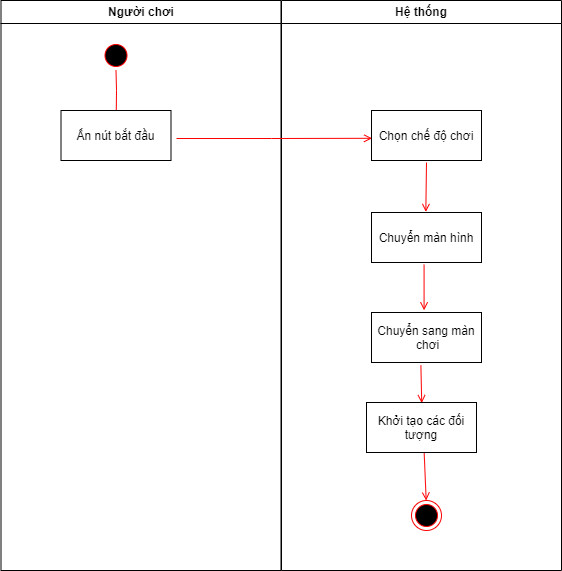
+ Khi máy bay boss xuất hiện, nhạc nền có thể thay đổi để tăng thêm sự căng thẳng và kịch tính. Nếu người chơi không tiêu diệt được máy bay boss trong một khoảng thời gian nhất định thì máy bay boss sẽ trở nên mạnh hơn, bắn nhiều đạn hơn và tốc độ nhanh hơn, khiến cho người chơi khó khăn hơn trong việc tiêu diệt.

+ Nếu người chơi tiêu diệt được máy bay boss sẽ nhận được một số điểm thưởng lớn và trò chơi có thể chuyển sang một giai đoạn mới với độ khó cao hơn.

## 2.2. Thiết kế giao diện

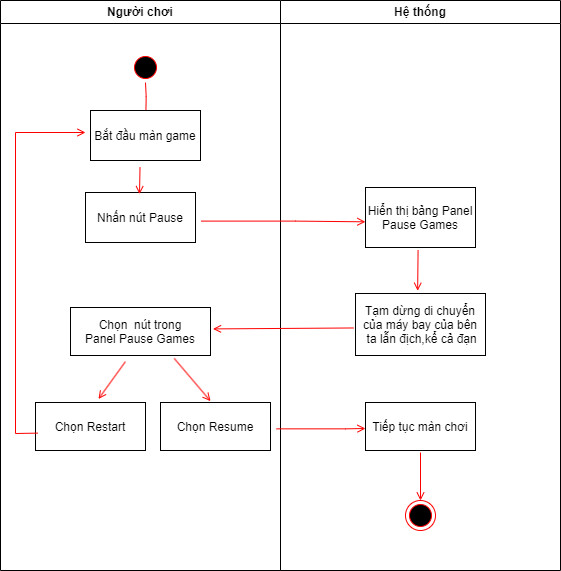
### 2.2.1. Biểu đồ - Flowchart

2.2.1.1. Bắt đầu game

****

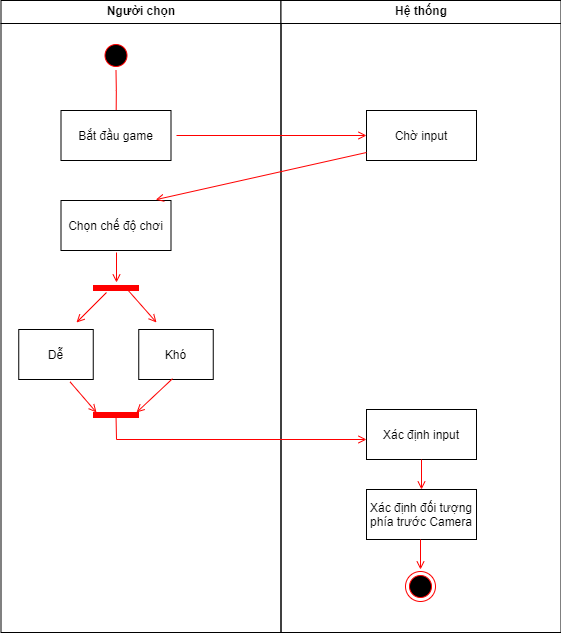
*Hình 1. Biểu đồ bắt đầu game*

2.2.1.2. Tạm dừng game

****

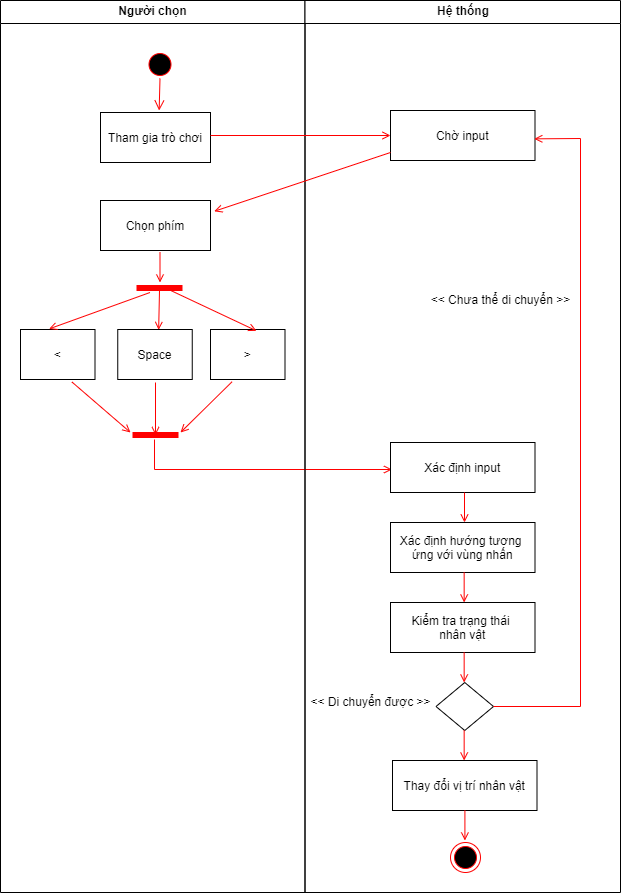
*Hình 2. Biểu đồ tạm dừng game*

2.2.1.3. Chọn chế độ



*Hình 3. Biểu đồ chọn chế độ*

2.2.1.4. Điều khiển nhân vật



*Hình 4. Biểu đồ điều khiển nhân vật*

### 2.2.2. Mô tả

* Bắt đầu game:

+ Khi người chơi ấn nút icon của game thì hệ thống sẽ hiển thị giao diện chính của game.

+ Giao diện chính có nền lấy giao diện vũ trụ, đây là giao diện đầu tiên mà người chơi nhìn thấy khi mở game.

+ Giao diện có nút “Play” để bắt đầu trò chơi và nút “Quit” để thoát game.

+ Khi người chơi click vào nút “Play” trên màn hình thì hệ thống sẽ chuyển sang màn hình giao diện chọn chế độ chơi.

+ Có 2 chế độ chơi cho người chơi lựa chọn, nếu người chơi click vào button “Easy” hoặc “Hard” thì hệ thống sẽ chuyển sang màn hình giao diện trò chơi bắt đầu chiến đấu.

+ Nếu người chơi không chọn chế độ, người chơi có thể click button “Close” để trở lại màn hình giao diện chính của game.

+ Khi người chơi click vào nút “Quit” trên màn hình giao diện chính hay màn hình giao diện chọn chế độ chơi thì game sẽ đóng lại.

* Giao diện tạm dừng game:

+ Người chơi có thể nhấn nút “Pause” để tạm dừng game.

+ Khi đó hệ thống sẽ hiển thì Panel Pause Games màn hình sẽ xuất hiện các nút ‘Resume” và “Restart” và sẽ tạm dừng di chuyển của máy bay người chơi và kẻ địch, kể cả đạn, khi đó người chơi có thể chọn “Resume” để tiếp tục tiến trình của trò chơi hoặc chọn “Restart” để bắt đầu lại trò chơi.

* Giao diện chế độ dễ:

+ Trong chế độ này, người chơi có 3 mạng (được thể hiện bằng thanh máu).

+ Tốc độ di chuyển của máy bay địch và tốc bắn chậm, giúp người chơi dễ dàng điều khiển và đạt được điểm cao.

+ Ở chế độ này sẽ chỉ có các máy bay địch bình thường, sẽ không xuất hiện máy bay boss.

* Giao diện chế độ khó:

+ Người chơi chỉ có 1 mạng duy nhất trong chế độ này.

+ Máy bay địch sẽ di chuyển và bắn nhanh dần theo điểm số, có cả máy bay lao vào.

+ Khi đạt số điểm nhất định sẽ có máy bay boss với nhiều máu hơn và bắn nhiều đạn hơn, đòi hỏi người chơi phải có kĩ năng cao.

* Giao diện kết thúc game:

+ Giao diện này xuất hiện khi máy bay người chơi bị phá hủy.

+ Khi đó màn hình sẽ hiển thì button “Restart” và button “Exit”, nếu người chơi muốn tiếp tục chơi game thì có thể click vào button “Restart” game sẽ được bắt đầu lại từ đầu, còn nếu người chơi không muốn tiếp tục chơi thì có thể click vào button “Exit” game sẽ dừng lại và chuyển sang màn hình giao diện chính của trò chơi.

* Các yếu tố giao diện chung:

+ Chuyển động nhân vật: Người chơi điều khiển máy bay bằng các phím mũi tên di chuyển hoặc các phím “A D W S” để di chuyển máy bay lên xuống và bắn đạn bằng phím spacebar.

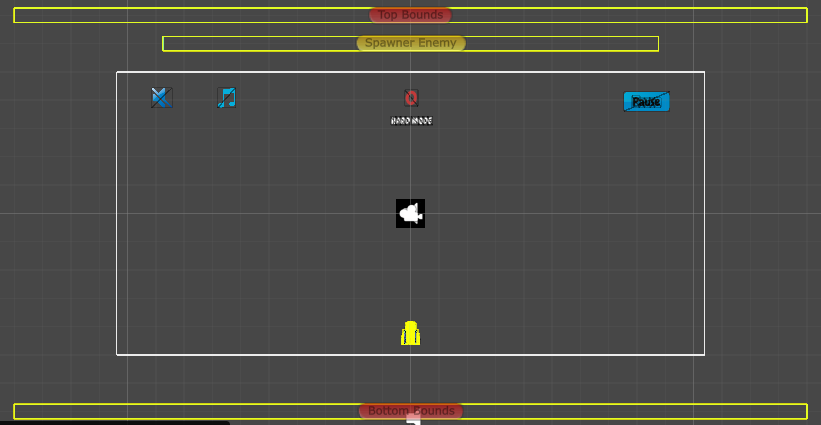
+ Đối tượng trong game: Bao gồm máy bay người chơi, máy bay kẻ thù, máy bay lao vào, viên đạn từ máy bay người chơi và máy bay kẻ thù.

+ Âm thanh: Có các hiệu ứng âm thanh như tiếng bắn, tiếng nổ, và nhạc nền.

+ Điểm số: Điểm số được hiển thị trên màn hình trong quá trình chơi, người chơi sẽ được cộng điểm khi tiêu diệt máy bay kẻ thù, đối với máy bay địch thông thường khi tiêu diệt 1 máy bay sẽ được cộng 3 điểm, đối với máy bay lao vào khi tiêu diệt 1 máy bay sẽ được cộng 1 điểm và đối với máy bay boss khi tiêu diệt sẽ được cộng 10 điểm. Game kết thúc khi máy bay người chơi bị tiêu diệt và điểm số sẽ dừng lại, nếu người chơi click vào button “Restart” thì điểm số sẽ trở về 0 điểm.

### 2.2.3. Giao diện các màn hình

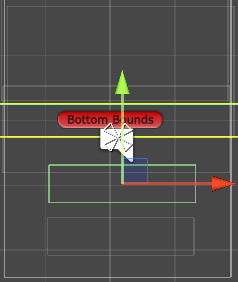
2.2.3.1. Màn hình thiết kế GamePlay và GamePlayEasy



*Hình 5. Màn hình thiết kế GamePlay và GamePlayEasy*

* Top Bounds và Bottom Bounds: Chốt chặn trên dưới để phát hiện khi có vật thể tiến vào khu vực này và thực hiện xóa vật thể đó khỏi Main View.
* Spawner Enemy: Vùng máy bay định được sinh ra.
* Plane Player: Người chơi.
* Mute: Bật/tắt âm thanh hiệu ứng game.
* Music: Bật/tắt nhạc nền.
* Pause: Tạm dừng.

2.2.3.2. Màn hình thiết kế Game Over Panel



*Hình 6. Màn hình thiết kế Game Over Panel*

* Restart Button: Thực hiện lại trò chơi.
* Exit Button: Thoát trò chơi và quay trở về MainMenu.
* End Score: Thông báo số điểm đạt được.

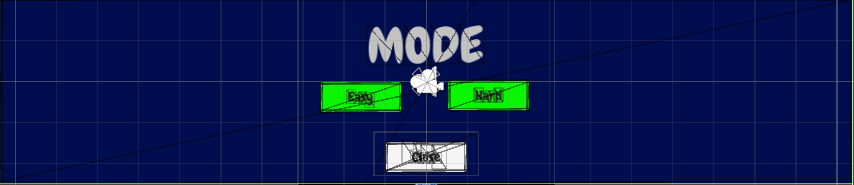
2.2.3.3. Màn hình thiết kế MainMenu



*Hình 7. Màn hình thiết kế MainMenu*

* Play Button: Nút bấm có chức năng bắt đầu trò chơi.
* Quit Button: Nút bấm có chức năng thoát trò chơi.

2.2.3.4. Màn hình thiết kế Mode Panel



*Hình 8. Màn hình thiết kế Mode Panel*

* Easy Button và Hard Button: Nút bấm có chức năng chọn chế độ (dễ/khó) cho trò chơi do người chơi tự chọn. Đây là điều kiện bắt buộc trước khi tiến đến màn GamePlay hoặc GamePlayEasy.
* Close Button: Nút bấm có chức năng quay trở về màn MainMenu.

## 2.3. Thiết kế âm thanh

### 2.3.1. Âm thanh music\_background

* Sự kiện: Nhạc nền cho MainMenu và Mode Panel
* Tài nguyên: Chỉnh sửa với Adobe Audition

### 2.3.2. Âm thanh S2157 và S364

* Sự kiện: Nhạc nền cho màn hình GamePlay và GamePlayEasy
* Tài nguyên: Chỉnh sửa với Adobe Audition

### 2.3.3. Âm thanh explosion\_enemy

* Sự kiện: Tiếng đạn của kẻ địch
* Tình huống: Khi máy bay địch bị hạ gục, âm thanh sẽ được sử dụng
* Tài nguyên: https://www.myinstants.com/en/index/us/

### 2.3.4. Âm thanh explosion\_player

* Sự kiện: Tiếng đạn của máy bay người chơi điều khiển
* Tình huống: Khi người bị hạ gục, âm thanh sẽ được sử dụng
* Tài nguyên: https://www.myinstants.com/en/index/us/

### 2.3.5. Âm thanh explosion\_asteroid

* Sự kiện: Tiếng va chạm của các vật thể trong game
* Tình huống: Xuất hiện khi có các va chạm trong game giữa đạn của người chơi với đạn của địch
* Tài nguyên: https://www.myinstants.com/en/index/us/

### 2.3.6. Âm thanh weapon\_enemy

* Sự kiện: Tiếng va chạm của các vật thể trong game
* Tình huống: Xuất hiện khi địch bị tiêu diệt
* Tài nguyên: https://www.myinstants.com/en/index/us/

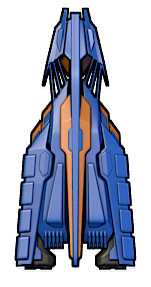
### 2.3.7. Âm thanh weapon\_player

* Sự kiện: Tiếng va chạm của các vật thể trong game
* Tình huống: Xuất hiện khi người chơi bị tiêu diệt
* Tài nguyên: https://www.myinstants.com/en/index/us/

# CHƯƠNG 3. CÀI ĐẶT CHƯƠNG TRÌNH VÀ KẾT QUẢ

## 3.1. Cài đặt chương trình

### 3.1.1. Máy bay người chơi

****

*Hình 9. Máy bay người chơi*

**Xử lý bắn đạn cho máy bay**

IEnumerator Shoot(){

canShoot = false;

yield return new WaitForSeconds (0.3f);

Vector3 temp = transform.position;

temp.y += 0.5f;

Instantiate (\_bullet, temp, Quaternion.identity);

AudioSource.PlayClipAtPoint (\_weaponClip, temp);

//StartCoroutine (Shoot ());

canShoot = true;

}

### 3.1.2. Xử lý di chuyển

void FixedUpdate () {

float h = CnInputManager.GetAxis ("Horizontal") \* speed;

float v = CnInputManager.GetAxis ("Vertical") \* speed;

\_myBody.velocity = new Vector2 (h,v);

}

//

podSate = PodState.ROTATION;

transform.position = \_Origin;

}

break;

}

### 3.1.3. Xử lý va chạm với các vật thể khác

void OnTriggerEnter2D(Collider2D target){

if (target.tag == "Enemy" || target.tag == "Rock" || target.tag == "RedBullet") {

Destroy (gameObject);

Destroy (target.gameObject);

\_explosionPlayer = (GameObject)Instantiate (\_explosionPlayer, target.transform.position, Quaternion.identity);

Destroy (\_explosionPlayer,1);

if (GamePlayController.instance != null) {

GamePlayController.instance.\_PlayerDied ();

}

AudioSource.PlayClipAtPoint (\_explosionPlayerClip, transform.position);

}

}

### 3.1.4. Máy bay kẻ thù

****

*Hình 10. Máy bay kẻ địch thứ 1*

**Xử lý tốc độ bay theo điểm**

void Start(){

if ( GamePlayController.instance.playerScore < 10) {

\_speed = Random.Range (-2, -4);

};

if ( 10 <= GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 20) {

\_speed = Random.Range (-3, -5);

};

if ( 20 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <=30) {

\_speed = Random.Range (-4, -6);

};

if ( 30 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 40) {

\_speed = Random.Range (-5, -7);

};

if ( 40 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 50) {

\_speed = Random.Range (-6, -8);

};

if ( 50 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 60) {

\_speed = Random.Range (-7, -9);

};

if ( 60 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 70) {

\_speed = Random.Range (-8, -10);

};

if ( 70 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 80) {

\_speed = Random.Range (-9, -11);

};

if ( 80 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 90) {

\_speed = Random.Range (-10, -12);

};

if ( 90 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 100) {

\_speed = Random.Range (-11, -13);

};

if ( GamePlayController.instance.playerScore > 100) {

\_speed = Random.Range (-12, -15);

};

StartCoroutine (Shoot ());

}

// Update is called once per frame

void FixedUpdate () {

\_myBody.velocity = new Vector2 (\_myBody.velocity.x, \_speed);

}

void OnTriggerEnter2D(Collider2D target){

if (target.tag == "Bounds") {

Destroy (gameObject);

}

}

**Xử lý tốc độ bắn theo điểm**

IEnumerator Shoot(){

if ( GamePlayController.instance.playerScore <= 30) {

yield return new WaitForSeconds (Random.Range (0.7f, 1f)) ;

};

if ( 30 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 60) {

yield return new WaitForSeconds (Random.Range (0.4f, 0.7f)) ;

};

if (GamePlayController.instance.playerScore > 60) {

yield return new WaitForSeconds (Random.Range (0.2f, 0.5f)) ;

};

Vector3 temp = transform.position;

if (GamePlayController.instance.playerScore <= 20) {

temp.y -= 0.4f;

};

if ( 20 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 40) {

temp.y -= 0.35f;

};

if ( 40 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 60) {

temp.y -= 0.3f;

};

if ( 60 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 80) {

temp.y -= 0.25f;

};

if (GamePlayController.instance.playerScore > 80) {

temp.y -= 0.3f;

};

Instantiate (\_bullet, temp, Quaternion.identity);

AudioSource.PlayClipAtPoint (\_weaponEnemyClip, temp);

StartCoroutine (Shoot ());

}

### 3.1.5. Máy bay lao vào

****

*Hình 11. Máy bay kẻ địch thứ 2*

**Xử lý tốc độ bay theo cấp**

void Start(){

if ( GamePlayController.instance.playerScore <= 10) {

\_speed = Random.Range (-2, -4);

};

if ( 10 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 20) {

\_speed = Random.Range (-3, -5);

};

if ( 20 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 30) {

\_speed = Random.Range (-4, -6);

};

if ( 30 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 40) {

\_speed = Random.Range (-5, -7);

};

if ( 40 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 50) {

\_speed = Random.Range (-6, -8);

};

if ( 50 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 60) {

\_speed = Random.Range (-7, -9);

};

if ( 60 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 70) {

\_speed = Random.Range (-8, -10);

};

if ( 70 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 80) {

\_speed = Random.Range (-9, -11);

};

if ( 80 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 90) {

\_speed = Random.Range (-10, -12);

};

if ( 90 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 100) {

\_speed = Random.Range (-11, -13);

};

if ( GamePlayController.instance.playerScore > 100) {

\_speed = Random.Range (-12, -15);

};

\_myBody.angularVelocity = Random.Range (0, 400);

### 3.1.6. Máy bay boss



*Hình 12. Máy bay boss*

**Xử lý tốc độ bay**

private void Update()

{

if (enemyType == EnemyType.Individual && moveToFro)

{

if (!exec)

{

transform.position = Vector2.SmoothDamp(transform.position, minW, ref vel1, smoothTime, maxSpeed, Time.deltaTime);

if (transform.position.x - 0.1f <= minW.x)

{

closerToMin = true;

closerToMax = false;

exec = true;

}

}

else

{

if (transform.position.x - 0.1f <= minW.x)

{

closerToMin = true;

closerToMax = false;

}

if (transform.position.x + 0.1f >= maxW.x)

{

closerToMax = true;

closerToMin = false;

}

if (closerToMax)

transform.position = Vector2.SmoothDamp(transform.position, minW, ref vel1, smoothTime, maxSpeed, Time.deltaTime);

else if (closerToMin)

transform.position = Vector2.SmoothDamp(transform.position, maxW, ref vel2, smoothTime, maxSpeed, Time.deltaTime);

}

}

}

void Start()

{

if (projectileToSpawn != null)

InvokeRepeating("Fire", 0.01f, fireRate);

if (enemyType == EnemyType.Individual)

{

minHeight = Screen.height \* 0.6f;

maxHeight = Screen.height \* 0.8f;

Vector2 minH = Camera.main.ScreenToWorldPoint(new Vector2(0f, minHeight));

Vector2 maxH = Camera.main.ScreenToWorldPoint(new Vector2(0f, maxHeight));

float height = Random.Range(minH.y, maxH.y);

minWidth = Screen.width \* 0.2f;

maxWidth = Screen.width \* 0.8f;

minW = Camera.main.ScreenToWorldPoint(new Vector2(minWidth, 0f));

maxW = Camera.main.ScreenToWorldPoint(new Vector2(maxWidth, 0f));

minW.y = height;

maxW.y = height;

float width = Random.Range(minW.x, maxW.x);

transform.position = new Vector2(width, height);

}

}

**Xử lý bắn đạn**

private void Fire()

{

Vector2 targetPos = new Vector2(transform.position.x + projectileSpawnOffset.x, transform.position.y + projectileSpawnOffset.y);

Vector2 targetPos1 = new Vector2(transform.position.x + projectileSpawnOffset1.x, transform.position.y + projectileSpawnOffset1.y);

Vector2 targetPos2 = new Vector2(transform.position.x + projectileSpawnOffset2.x, transform.position.y + projectileSpawnOffset2.y);

GameObject proj = Instantiate(projectileToSpawn, targetPos, projectileToSpawn.transform.rotation) as GameObject;

if (proj.GetComponent<Rigidbody2D>())

proj.GetComponent<Rigidbody2D>().AddForce(projectileSpeed, ForceMode2D.Impulse);

GameObject proj1 = Instantiate(projectileToSpawn, targetPos1, projectileToSpawn.transform.rotation) as GameObject;

if (proj1.GetComponent<Rigidbody2D>())

proj1.GetComponent<Rigidbody2D>().AddForce(projectileSpeed, ForceMode2D.Impulse);

GameObject proj2 = Instantiate(projectileToSpawn, targetPos2, projectileToSpawn.transform.rotation) as GameObject;

if (proj2.GetComponent<Rigidbody2D>())

proj2.GetComponent<Rigidbody2D>().AddForce(projectileSpeed, ForceMode2D.Impulse);

DamageHealthbarEnemy();

}

### 3.1.7. Viên đạn máy bay người chơi

**  **

*Hình 13. Viên đạn máy bay người chơi*

**Xử lý va chạm khi bắn trúng các vật thể**

void OnTriggerEnter2D(Collider2D target){

if (target.tag == "Bounds") {

Destroy (gameObject);

}

if (target.tag == "Enemy") {

Destroy (gameObject);

Destroy (target.gameObject);

\_explosionEnemy = (GameObject)Instantiate (\_explosionEnemy, target.transform.position, Quaternion.identity);

Destroy (\_explosionEnemy,1);

GamePlayController.instance.playerScore++;

AudioSource.PlayClipAtPoint (\_explosionEnemyClip, target.transform.position);

}

if (target.tag == "Rock") {

Destroy (gameObject);

Destroy (target.gameObject);

\_explosionRock = (GameObject)Instantiate (\_explosionRock, target.transform.position, Quaternion.identity);

Destroy (\_explosionRock,1);

GamePlayController.instance.playerScore++;

AudioSource.PlayClipAtPoint (\_explosionRockClip, target.transform.position);

}

}

### 3.1.8. Viên đạn máy bay địch

****

*Hình 14. Viên đạn máy bay địch*

**Xử lý tốc độ đạn bay theo máy bay**

void Start(){

if ( GamePlayController.instance.playerScore <= 10) {

speed = 5;

};

if ( 10 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 20) {

speed = 6;

};

if ( 20 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 30) {

speed = 7;

};

if ( 30 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 40) {

speed = 8;

};

if ( 40 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 50) {

speed = 9;

};

if ( 50 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 60) {

speed = 10;

};

if ( 60 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 70) {

speed = 11;

}

if ( 70 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 80) {

speed = 12;

}

if ( 80 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 90) {

speed = 13;

}

if ( 90 < GamePlayController.instance.playerScore && GamePlayController.instance.playerScore <= 100) {

speed = 14;

}

if ( GamePlayController.instance.playerScore > 100) {

speed = 16;

}

}

**Xử lý va chạm khi trúng máy bay người chơi**

void FixedUpdate () {

\_myBody.velocity = new Vector2 (\_myBody.velocity.x, -speed );

}

void OnTriggerEnter2D(Collider2D target){

if (target.tag == "Bounds") {

Destroy (gameObject);

}

}

### 3.1.9. Hình nền

****

*Hình 15. Hình nền trò chơi*

**Canh hình nền theo màn hình**

void Start () {

var worldHeight = Camera.main.orthographicSize \* 2;//10

var worldWidth = worldHeight \* Screen.width/Screen.height;//10 \* 214 / 356

transform.localScale = new Vector3(worldWidth,worldHeight,0);

}

**Cho hình nền chạy tạo cảm giác bay**

void Start () {

\_mat = GetComponent<Renderer> ().material;

\_offset = \_mat.GetTextureOffset ("\_MainTex");

}

// Update is called once per frame

void Update () {

\_offset.y += speed \* Time.deltaTime;

\_mat.SetTextureOffset ("\_MainTex", \_offset);

}

### 3.1.10. Xử lý âm thanh



*Hình 16. Icon âm thanh*

**Xứ lý tắt mở nhạc**

void Start() {

btnNhac.GetComponent<Button>();

audioSource = gameObject.GetComponent<AudioSource>();

audioSource.Play();

Unmute();

}

**Xử lý chọn nhạc nền**

public void ChangeButtonAudio(){

if (counter == 3)

counter = 0;

else

counter++;

switch (counter)

{

case 0:

btnNhac.image.overrideSprite = MoNhac;

audioSource.clip = NhacNen3;

audioSource.Play();

break;

case 1:

btnNhac.image.overrideSprite = MoNhac;

audioSource.clip = NhacNen2;

audioSource.Play();

break;

case 2:

btnNhac.image.overrideSprite = MoNhac;

audioSource.clip = NhacNen1;

audioSource.Play();

break;

case 3:

btnNhac.image.overrideSprite = TatNhac;

audioSource.Stop();

break;

}

}

//Tắt Nhạc

public void Mute(){

btnTatNhac.SetActive(true);

//audioSource.Stop();

AudioListener.volume = 0f;

btnMoNhac.SetActive(false);

// Time.timeScale = 0;

}

//Mở Nhạc

public void Unmute()

{

btnTatNhac.SetActive(false);

audioSource.clip = NhacNen3;

AudioListener.volume = 1f;

audioSource.Play();

btnMoNhac.SetActive(true);

}

//Mở List nhac

public void ListNhac()

{

PanelListNhac.SetActive(true);

btnListNhac.SetActive(false);

}

//Mở nhạc nền 1

public void MoNhacNen1()

{

PanelListNhac.SetActive(false);

btnListNhac.SetActive(true);

audioSource.clip = NhacNen1;

audioSource.Play();

}

//Mở nhạc nền 3

public void MoNhacNen2()

{

PanelListNhac.SetActive(false);

btnListNhac.SetActive(true);

audioSource.clip = NhacNen2;

audioSource.Play();

}

//Mở nhạc nền 3

public void MoNhacNen3()

{

PanelListNhac.SetActive(false);

btnListNhac.SetActive(true);

audioSource.clip = NhacNen3;

audioSource.Play();

}

## 3.2. Một số giao diện trò chơi

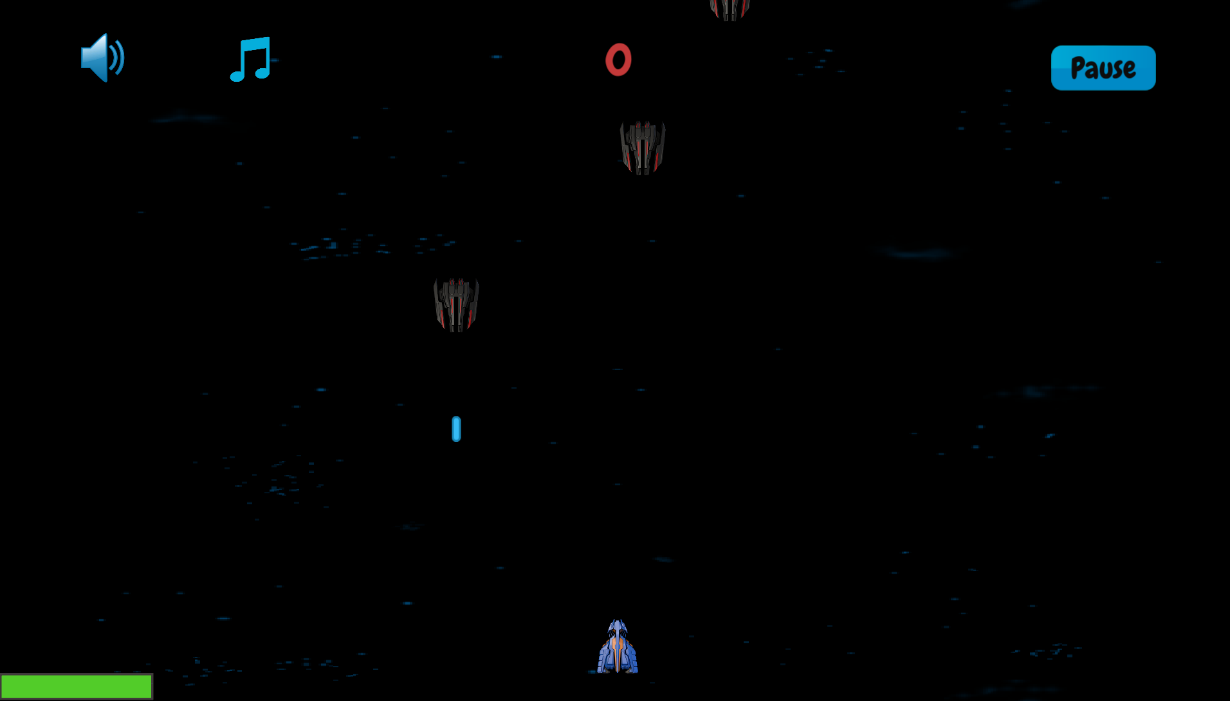
### 3.2.1. Màn hình chính

*Hình 17. Giao diện màn hình chính*

* Người chơi ấn nút “Play” để chọn 2 chế độ chơi dễ và khó
* Nhấn nút “Quit” để thoát khỏi trò chơi

### 3.2.2. Giao diện chế độ dễ

****

*Hình 18. Giao diện chế độ dễ*

* Chế độ dễ người chơi sẽ có 3 mạng (tượng trưng cho thanh máu) và tốc độ di chuyển của máy bay địch và tốc độ bắn của máy bay khá là chậm nên người chơi có thể điều khiển dễ dàng và dễ đạt được số điểm cao.

### 3.2.3. Giao diện chế độ khó

****

*Hình 19. Giao diện chế độ khó*

* Chế độ khó người chơi sẽ có 1 mạng duy nhất, máy địch bao gồm máy bay chiến đấu có tốc độ di chuyển và tốc độ bắn khá nhanh, số điểm của người chơi càng cao đồng nghĩa với tốc độ bay và bắn của địch sẽ càng tăng, ngoài máy bay chiến đấu ra thì có máy bay lao vào như tên lửa người chơi dựa vào kỹ năng của mình để né hay bắn hạ máy bay của địch.
* Khi người chơi đạt số điểm nhất định thì sẽ có máy bay boss với đạn bắn ba tía nhiệm vụ người chơi là tiêu diệt máy bay boss. Do là chế độ khó nên người chơi khó đạt được điểm cao và có duy nhất 1 mạng nếu chết thì người chơi sẽ buộc chơi lại từ đầu.

### 3.2.4. Giao diện tạm dừng game

****

*Hình 20. Giao diện tạm dừng game*

* Người chơi có thể ấn nút “Pause” bên góc phải để dừng cuộc chơi, khi người chơi muốn tiếp tục thì nhấn nút “Resume”, nếu người chơi muốn chơi lại từ đầu thì nhấn nút “Restart”.

### 3.2.5. Giao diện kết thúc game

****

*Hình 21. Giao diện kết thúc game*

* Khi máy bay người chơi bị phá hủy thì đồng nghĩa với việc kết thúc game, người chơi có thể “Restart” để chơi lại hoặc ấn nút “Exit” để thoát ra ngoài màn hình chính.

# KẾT LUẬN

## 1. Kết quả đạt được

* Game có giao diện thân thiện, đẹp mắt, dễ sử dụng.
* Game đã đạt được yêu cầu cơ bản của trò chơi.

## 2. Hướng phát triển

* Tối ưu hóa các xử lý giúp game chạy nhanh hơn.
* Tăng thêm số lượng màn chơi để người chơi thêm nhiều lựa chọn.
* Thêm vũ khí, dạn dược, nâng cấp máy bay và vũ khí
* Cung cấp thêm các công cụ để người chơi có thể chỉnh sửa hoặc tạo ra cho mình riêng các hình ảnh nhân vật, hiệu ứng âm thanh, …

# TÀI LIỆU THAM KHẢO

[1] Ryan Henson Creighton, *Unity 3D Game Development by Example*, Packt Publishing, 2010.

[2] Will Goldstone, *Unity Game Development Essential*, Packt Publishing, 2009.

[3] Efraim Meulenberg, *Game Development with Unity*, VTC, 2010.

[4] Efraim Meulenberg, *Game Development for iPhone/iPad Using Unity iPhone*, VTC, 2010.

[5] Sue Blackman, *Beginning 3D Game Development with Unity 4 All-in-One, Multi-Platform Game Development 2nd Edition* , 2013.