

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI  
KHOA CÔNG NGHỆ THÔNG TIN



# ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI

## THIẾT KẾ, LẬP TRÌNH TRÒ CHƠI ALIENS WAR 3D

Giảng viên hướng dẫn : ThS. Nguyễn Việt Hưng

Sinh viên thực hiện : Chu Xuân Hùng

Lớp : CNTT2 – K61

Mã sinh viên : 201200146

Hà Nội – 2024

## LỜI CẢM ƠN

Trong báo cáo này, em xin chân thành cảm Khoa Công Nghệ Thông Tin, Trường Đại học Giao Thông Vận Tải đã tạo điều kiện tốt nhất cho em được thực hiện đề tài này.

Em muôn gửi lời biết ơn sâu sắc tới giảng viên hướng dẫn ThS. Nguyễn Việt Hưng, là người đã tận tình hướng dẫn và đồng hành cùng em trong suốt thời gian thực hiện đề tài. Em cũng xin cảm ơn Giảng viên ThS. Nguyễn Việt Hưng đã có những trao đổi, những chỉ dẫn giúp em giải quyết các vấn đề và hoàn thành tốt đề án của mình.

Không thể không nhắc đến sự hỗ trợ và khích lệ từ phía gia đình, bạn bè và người thân. Sự ủng hộ từ các bạn đã là nguồn động viên lớn lao, giúp em vượt qua những khó khăn và hoàn thành dự án một cách thành công.

Cuối cùng, em muôn gửi lời biết ơn đến bản thân mình, vì sự kiên nhẫn, nỗ lực và sự cống hiến trong suốt thời gian thực tập. Đây là một trải nghiệm quý báu mà em sẽ luôn nhớ và mang theo trong sự nghiệp phát triển game của mình.

Xin chân thành cảm ơn!

Trân trọng,

Sinh viên thực hiện

**Chu Xuân Hùng**

# MỤC LỤC

LỜI CẢM ƠN.....	1
MỤC LỤC .....	2
DANH MỤC HÌNH ẢNH.....	3
LỜI MỞ ĐẦU .....	5
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	6
1.1. Tổng quan Unity Engine.....	6
1.2. Giới thiệu chung về Unity Engine. ....	8
1.3. Các đặc điểm và tính năng của Unity. ....	9
1.4. Các thành phần trong Unity. ....	11
1.5. Giao diện của Unity. ....	16
1.6. Kiến thức về thiết kế trò chơi 3D. ....	20
CHƯƠNG 2: TỔNG QUAN TRÒ CHƠI .....	23
2.1. Mục tiêu và ý nghĩa.....	23
2.2. Phạm vi đề tài.....	23
2.3. Phương pháp nghiên cứu.....	24
2.4. Bối cảnh trò chơi và mạch trò chơi. ....	25
2.5. Giới thiệu nhân vật và môi trường trong trò chơi. ....	26
2.6. Tổng quan về luật chơi và cách chơi.....	28
CHƯƠNG 3: THIẾT KẾ, LẬP TRÌNH TRÒ CHƠI ALIENS WAR 3D.....	30
3.1. Xây dựng trò chơi. ....	30
3.2. Điều khiển I/O khi chơi trò chơi. ....	65
3.3. Giao diện GUI.....	66
3.4. Các âm thanh trong trò chơi. ....	69
3.5. Kiểm thử và đánh giá. ....	69
KẾT LUẬN VÀ KIẾN NGHỊ.....	71
DANH MỤC TÀI LIỆU THAM KHẢO .....	72

# DANH MỤC HÌNH ẢNH

Hình 1. 1: Hình ảnh minh họa đa nền tảng.....	6
Hình 1. 2: Logo của Unity Engine .....	8
Hình 1. 3: Asset trong Unity.....	11
Hình 1. 4: Các scene trong Unity.....	11
Hình 1. 5: Kéo tài nguyên vào scene để sử dụng .....	12
Hình 1. 6:Các thành phần trong đối tượng camera.....	13
Hình 1. 7: Cách tạo file Script mới. ....	14
Hình 1. 8: Một file Script đang gắn vào đối tượng. ....	15
Hình 1. 9: Một số đối tượng trong Prefabs.....	15
Hình 1. 10: Giao diện của Unity .....	16
Hình 1. 11:Các nút chức năng cho cửa Scene.....	17
Hình 1. 12: Cửa sổ Hierarchy.....	17
Hình 1. 13: Cửa sổ Inspector.....	18
Hình 1. 14: Cửa sổ Project .....	19
Hình 1. 15: Các loại hình ảnh trong cửa sổ game. ....	20
Hình 3. 1: Đồ họa trò chơi Aliens War 3D. ....	30
Hình 3. 2: Nhân vật chính Alex Hunter. ....	30
Hình 3. 3: Animation Aim Idle. ....	31
Hình 3. 4: Animation Aim Walk Backward.....	31
Hình 3. 5: Animation Aim Walking.....	31
Hình 3. 6: Animation Death .....	32
Hình 3. 7: Animation Jump backward. ....	32
Hình 3. 8: Animation Jump Forward. ....	32
Hình 3. 9: Animation Normal Backwards Run.....	33
Hình 3. 10: Animation Normal Backwards Walk.....	33
Hình 3. 11: Animation Normal Idle. ....	33
Hình 3. 12: Animation Normal Walk Forward. ....	34
Hình 3. 13: Animation Run Backward Left. ....	34
Hình 3. 14: Animation Run Backward Right.....	34
Hình 3. 15: Animation Run Backwards. ....	35
Hình 3. 16: Animation Run Forward. ....	35
Hình 3. 17: Animation Run Forward Left.....	35
Hình 3. 18: Animation Run Forward Right.....	36
Hình 3. 19: Animation Run Left. ....	36
Hình 3. 20: Animation Run Right. ....	36
Hình 3. 21: Animation Shoot. ....	37
Hình 3. 22: Animation Strafe Left. ....	37
Hình 3. 23: Animation Strafe Right. ....	37
Hình 3. 24: Animation Walk Backward Left. ....	38
Hình 3. 25: Animation Walk Backward Right. ....	38
Hình 3. 26: Animation Walk Forward Left.....	38
Hình 3. 27: Animation Walk Forward Right.....	39
Hình 3. 28: Code di chuyển nhân vật. ....	39
Hình 3. 29: Code xoay nhân vật.....	39
Hình 3. 30: Code bắn súng.....	40
Hình 3. 31: Code đổi súng.....	40
Hình 3. 32: Code đổi trạng thái nhân vật. ....	41
Hình 3. 33. Thanh trạng thái nhân vật.....	41
Hình 3. 34: Quái vật Wolf.....	42
Hình 3. 35: Quái vật Catfish.....	42
Hình 3. 36: Quái vật Rake.....	43
Hình 3. 37: Animation Attack. ....	43

Hình 3. 38: Animation Die.....	43
Hình 3. 39: Animation Hit.....	44
Hình 3. 40: Animation Idle.....	44
Hình 3. 41: Animation Run.....	44
Hình 3. 42: Animation Walk.....	45
Hình 3. 43: Animation Attack.....	45
Hình 3. 44: Animation Die.....	45
Hình 3. 45: Animation Hit.....	46
Hình 3. 46: Animation Idle.....	46
Hình 3. 47: Animation Run.....	46
Hình 3. 48: Animation Walk.....	47
Hình 3. 49: Animation Attack.....	47
Hình 3. 50: Animation Die.....	47
Hình 3. 51: Animation Hit.....	48
Hình 3. 52: Animation Idle.....	48
Hình 3. 53: Animation Run.....	48
Hình 3. 54: Animation Walk.....	49
Hình 3. 55: Code tìm đường đi bằng AI NavMeshAgent .....	49
Hình 3. 56: Code va chạm.....	50
Hình 3. 57: Code thay đổi trạng thái quái vật.....	50
Hình 3. 58: Code quái vật tấn công.....	51
Hình 3. 59: Code quái vật bị tiêu diệt.....	51
Hình 3. 60: Môi trường rừng trong trò chơi.....	51
Hình 3. 61: Map trong trò chơi.....	52
Hình 3. 62: Ngôi nhà chứa cửa hàng.....	52
Hình 3. 63: Ngôi nhà bình thường .....	53
Hình 3. 64: Các loại cây trong trò chơi .....	53
Hình 3. 65: Camera góc nhìn thứ 3 .....	54
Hình 3. 66: Camera lúc ngắm bắn.....	55
Hình 3. 67: Code camera.....	56
Hình 3. 68: Súng lục.....	57
Hình 3. 69: Súng Shotgun .....	57
Hình 3. 70: Súng máy hạng nhẹ .....	57
Hình 3. 71: Súng trường.....	58
Hình 3. 72: Súng bắn tỉa.....	58
Hình 3. 73: Viên đạn .....	58
Hình 3. 74: Code hàm tạo.....	58
Hình 3. 75: Code bắn súng chung .....	59
Hình 3. 76: Code bắn súng Shotgun .....	59
Hình 3. 77: Code viên đạn .....	60
Hình 3. 78: Trụ hồi máu .....	60
Hình 3. 79: Cỗng tạo quái vật .....	61
Hình 3. 80: Code Gate.....	61
Hình 3. 81: Code menu chính.....	62
Hình 3. 82: Code điều chỉnh nhạc nền .....	63
Hình 3. 83: Code điều chỉnh nhạc nền và âm thanh.....	63
Hình 3. 84: Code menu trong game .....	64
Hình 3. 85: Code mua súng và mua đạn .....	65
Hình 3. 86: Nhân vật nhảm bắn .....	66
Hình 3. 87: Giao diện menu chính .....	66
Hình 3. 88: Giao diện chức năng cài đặt trong menu chính .....	67
Hình 3. 89: Giao diện menu tạm dừng .....	67
Hình 3. 90: Giao diện chức năng cài đặt trong menu tạm dừng.....	68
Hình 3. 91: Giao diện cửa hàng .....	68
Hình 3. 92: Giao diện GameOver.....	69

## LỜI MỞ ĐẦU

Thị trường game tại Việt Nam đang phát triển và có tiềm năng lớn. Trong thời đại công nghệ thông tin hiện nay, sản phẩm công nghệ ngày càng chịu sự đánh giá khắt khe hơn từ phía những người dùng, đặc biệt là về sản phẩm Game được nhận rất nhiều sự đánh giá từ phía Game thủ, hay chỉ là những người chơi bình thường. Ngành công nghiệp Game hiện nay có thể nói là bùng nổ, với tốc độ phát triển đến chóng mặt, rất nhiều những Game hay và hấp dẫn đã được ra đời trong thời gian qua. Phía sau những Game phát triển và nổi tiếng như vậy đều có một Game Engine. Game Engine là một công cụ hỗ trợ, một Middleware giúp người phát triển viết game một cách nhanh chóng và đơn giản, đồng thời cung cấp khả năng tái sử dụng các tài nguyên và mã nguồn cao do có thể phát triển nhiều game từ một Game Engine.

Từ xu hướng phát triển và những bất cập trên, đồ án này sẽ khảo sát và thiết kế, lập trình trò chơi Aliens War 3D trên Unity Engine rất phổ biến và không kém mạnh mẽ hiện nay nhằm thực nghiệm việc phát triển một trò chơi về hành động bắn súng sinh tồn 3D. Để chuẩn bị kiến thức và kỹ năng cho định hướng phát triển game sau này của mình, góp phần vào sự phát triển của ngành công nghiệp game tại Việt Nam.

Tại các chương đầu tiên của đồ án sẽ được trình bày lần lượt về các khái niệm chung về Unity Engine và kiến thức thiết kế trò chơi 3D. Chương tiếp theo sẽ giới thiệu tổng quan về trò chơi Aliens War 3D bao gồm cơ chế, phạm vi, bối cảnh và mạch trò chơi. Giới thiệu nhân vật, môi trường và hướng dẫn luật chơi và cách chơi. Sau những nội dung về thiết kế, em sẽ trình bày về nội dung nghiên cứu lập trình trên Unity về ngôn ngữ lập trình cùng các lớp, các hàm trong thư viện có sẵn của Unity thông qua các ví dụ thực tế khi phát triển thành một game Aliens War 3D. Cuối cùng là trình bày về kết quả nghiên cứu đạt được và những vấn đề chưa được giải quyết khi phát triển trò chơi Aliens War 3D.

Vì thế em chọn đề tài “**Thiết kế, lập trình trò chơi Aliens War 3D**” để đưa đến cho người chơi những giây phút giải trí và thư giãn sau những giờ làm việc căng thẳng. Qua việc phát triển trò chơi này, em mong muốn không chỉ học hỏi và nâng cao kỹ năng lập trình và thiết kế game, mà còn mang đến cho cộng đồng game một sản phẩm giải trí chất lượng, góp phần làm phong phú thêm thế giới trò chơi điện tử.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

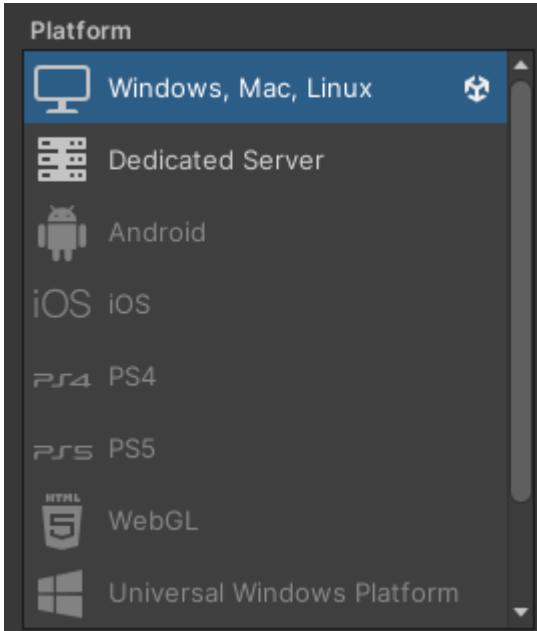
## 1.1. Tổng quan Unity Engine.

### 1.1.1. Unity là gì?

Đã qua rồi thời kỳ làm game trên nền Flash cẩn bản và buồn chán với những chuyển động thật cứng nhắc. Unity mang lại sức mạnh kỳ diệu cho nhân vật mà chúng ta muốn thể hiện sống động hơn trong không gian 3 chiều đầy huyền ảo. Công nghệ cao này tạo ra một bước đột phá mới về sự khác biệt trong công nghệ làm game hiện nay, mang đến cho người chơi 1 cảm giác rất khác lạ và hào hứng trong từng chuyển động, tương lai công nghệ này được áp dụng vào game Việt Nam sẽ mở ra một trang mới trong thế giới game 2D, 3D huyền ảo.

Unity được dùng để làm video game, hoặc những nội dung có tính tương tác như thể hiện kiến trúc, hoạt hình 2D, 3D thời gian thực. Unity tương tự với Director, Blender game engine, Virtools hay Torque Game Builder trong khía cạnh dùng môi trường đồ họa tích hợp ở quá trình phát triển game là chính.

Unity là một trong những engine được giới làm game không chuyên cực kỳ ưa chuộng bởi khả năng tuyệt vời của nó là phát triển trò chơi đa nền. Trình biên tập có thể chạy trên Windows và Mac OS, và có thể xuất ra game cho Windows, Mac, Wii, iOS, Android. Game cũng có thể chơi trên trình duyệt web thông qua plugin Unity Web Player. Unity mới bổ sung khả năng xuất ra game trên widget cho Mac, và cả Xbox 360, PlayStation 3.



Hình 1. 1: Hình ảnh minh họa đa nền tảng.

Chỉ với khoản tiền bỏ ra khá khiêm tốn (1.500 USD) là phiên bản pro đã nằm trong tay của chúng ta, dĩ nhiên tại Việt Nam số tiền này vẫn là quá lớn nhưng thật may là đã có phiên bản Unity Free. Tuy nhiên, nhiều tính năng quan trọng (Network) bị cắt giảm nhưng đó không phải là vấn đề quá lớn nếu muốn phát triển một tựa game tầm trung.

Vào năm 2009, Unity nằm trong top 5 game engine tốt nhất cho việc sản xuất game với chỉ sau 4 năm phát triển. Unity đứng thứ 4, xếp sau Unreal Engine 3, Gamebryo Engine (được VTC Studio mua về phát triển SQUAD) và Cry Engine 2. Lượng tài liệu hướng dẫn Unity rất phong phú. Hơn thế nữa nó còn có sẵn một cộng đồng cực lớn với diễn đàn riêng. Bất cứ điều gì không hiểu chúng ta đều có thể thoải mái hỏi và nhận được câu trả lời nhanh chóng, tận tâm.

Quá trình tạo địa hình cũng như truy xuất từ các phần mềm 3DSMax, Maya, Cinema4D... rất nhanh chóng. Sức mạnh và sự tiện lợi của Unity là vô cùng lớn.

Sức mạnh: Unity có thể tạo ra được nhiều loại game 2D, 3D đa dạng, dễ sử dụng với người làm game chưa chuyên nghiệp, chất lượng cao, chạy hầu hết trên các hệ điều hành.

Sự tiện lợi: nếu chúng ta là một người chuyên dùng 3Dmax, hay Maya hoặc phần mềm mã nguồn mở Blender thì quả là thật tuyệt, chúng ta sẽ có một lợi thế lớn khi viết game trên Unity này, bởi công việc tạo các mô hình 2D, 3D sẽ trở lên dễ dàng hơn rất nhiều, việc kết hợp giữa người lập trình và người thiết kế các mô hình sẽ nhanh và hiệu quả hơn. Trong Unity chúng ta có thể import trực tiếp các file mô hình đang thiết kế và sẽ thiết kế hoàn thiện tiếp nếu chưa xong trong khi đó công việc import chỉ diễn ra một lần. Không như việc phải dùng các công cụ khác để thực hiện viết game chúng ta sẽ phải xuất chúng ra một dạng nào đó và mỗi lần sửa lại phần mô hình chúng ta lại phải import lại, và như thế là quá mất thời gian trong việc tạo và chỉnh sửa các mô hình theo ý muốn. Ngoài ra Unity còn cho chúng ta trực tiếp tạo các mô hình nếu muốn. Việc đặt các thuộc tính vật lý trong Unity cũng cực kỳ dễ dàng và hỗ trợ sẵn nhiều chức năng.

### 1.1.2. Sơ lược lịch sử hình thành và phát triển của Unity.

Hình thành: Phần lõi của Unity ban đầu được viết bởi Joachim Ante vào năm 2001. Sau đó công ty được hình thành vào năm 2005 và bắt đầu với phiên bản 1.0. Đến năm 2007, Unity được nâng lên phiên bản 2.0. Unity bắt đầu hỗ trợ iPhone vào năm 2008. Vào tháng 6/2010, Unity chính thức hỗ trợ Android và cho ra đời phiên bản 3.0 có hỗ trợ Android vào tháng 9/2010 và bây giờ là phiên bản Unity 5. Có thể thấy tốc độ phát triển của Unity khá nhanh.

Giải thưởng: Unity đã đoạt được nhiều giải lớn với những giải chính sau:

- Năm 2006, Unity đạt "Best Use of Mac OS X Graphics" tại Apple's WWDC. Đây là lần đầu tiên một công cụ phát triển game đạt được chất lượng do giải thưởng uy tín này đưa ra.
- Năm 2009, Unity Technologies có tên trong "Top 5 công ty game của năm" do Gamasutra tổ chức.
- Năm 2010, Unity đoạt giải Best Engine Finalist do Develop Magazine bình chọn, giải Technology Innovation Award của Wall Street Journal ở thể loại phần mềm.

Khách hàng: Unity được trên 250.000 người đăng ký sử dụng gồm Bigpoint, Cartoon Network, Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Ubisoft, Warner Bros, các hãng phim lớn nhỏ, các chuyên gia độc lập, sinh viên và những người đam mê.

### **1.1.3. Tính năng của Engine Unity.**

Môi trường phát triển được tích hợp với tính năng kế thừa, khả năng chỉnh sửa đồ họa, chức năng kiểm tra chi tiết, và đặc biệt tính năng xem trước game ngay trong lúc xây dựng (live game preview).

Triển khai được trên nhiều nền tảng:

- Chương trình độc lập trên Windows và Mac OS
- Trên web, thông qua Unity Web Player plugin cho Internet Explorer, Firefox, Safari, Opera, Chrome, cho cả Windows và Mac OS.
- Trên Mac OS Dashboard widget. Cho Nintendo Wii (cần mua license thêm.)
- Cho iPhone, iPad application (cần mua license thêm.)
- Cho Google Android (cần mua license thêm.)
- Cho Microsoft Xbox 360 (cần mua license thêm.)
- Cho Sony PlayStation 3 (cần mua license thêm.)

Tài nguyên (model, âm thanh, hình ảnh, ...) được tải vào trong Unity và tự động cập nhật nếu tài nguyên có sự thay đổi. Unity hỗ trợ các kiểu định dạng từ 3DS Max, Maya, Blender, Cinema 4D và Cheetah3D.

Graphics engine sử dụng Direct3D (Windows), OpenGL (Mac, Windows), OpenGL ES (iPhone OS), và các API khác trên Wii.

Hỗ trợ bump mapping, reflection mapping, parallax mapping, Screen Space Ambient Occlusion v...V...

Unity Asset Server: Đây là một tính năng khá mới của Unity, theo đó Unity sẽ cung cấp một hệ thống quản lý theo dạng phiên bản cho tất cả asset và cả script. Đây là một kho chứa các tài nguyên cần thiết cho việc làm game. Khi import cũng như sửa chữa, trạng thái của asset ngay lập tức được cập nhật. Server chạy trên database opensource PostgreSQL và có thể truy cập trên cả Mac lẫn Windows, Linux. Asset Server đòi hỏi một khoản phí phụ trội là \$499 cho mỗi bản copy Unity, và một license Unity Pro.

## **1.2. Giới thiệu chung về Unity Engine.**



Hình 1. 2: Logo của Unity Engine

- Nhà phát triển: Unity Technologies
- Phiên bản mới nhất : 5 (phát hành vào ngày 11/4/2012)
- Được viết bởi ngôn ngữ : C++, C#
- Phát triển Game cho các hệ điều hành : Windows, Mac OS X (tạo và đóng gói), Wii, iPhone/iPad, Xbox 360, Android, PS3 (chỉ đóng gói ; cần giấy phép bổ sung cho từng nền tảng)

- Giấy phép: Độc quyền
- Website: <https://Unity.com/>

Unity là một 3D Game Engine, là một công cụ thiết kế Game dành cho PC, Mac và nhiều hệ máy di động khác.

Unity được sự hỗ trợ của Just-In-Time Compilation (JIT), sử dụng thư viện mã nguồn mở C++ Mono. Bằng việc sử dụng JIT, những Engine như Unity có thể tận dụng lợi thế của tốc độ biên dịch. Những đoạn code do chúng ta viết sẽ được Unity biên dịch ra Mono trước khi nó được thực thi. Điều này rất quan trọng cho Game để thực thi code vào những thời điểm cần thiết trong suốt thời gian chạy (Runtime).

Ngoài thư viện Mono, Unity cũng tận dụng chức năng của những thư viện phần mềm khác vào chức năng của nó, như Engine mô phỏng vật lý PhysicX của Nvidia, OpenGL và DirectX cho kết xuất hình ảnh 3D, OpenAL cho âm thanh. Tất cả các thư viện này được xây dựng thành những tính năng tự động hoặc công cụ trực quan vào Unity, vì thế chúng ta không cần phải lo lắng về việc phải học thế nào để sử dụng chúng một cách riêng lẻ.

Unity có một cộng đồng người dùng rất mạnh (rất lớn) luôn chia sẻ những Plugins, công cụ của họ dưới hình thức gói phần mềm bổ sung.

Có thể sản xuất các trò chơi theo tiêu chuẩn chuyên nghiệp, xuất bản 3D cho cả Mac và PC cũng như sở hữu riêng một Web Player của riêng mình, Unity là một trong những Game Engine có tốc độ phát triển nhanh nhất. Unity cũng có phiên bản phát triển Game cho hệ máy Wii của Nintendo và Iphone của Apple, có nghĩa là một khi chúng ta làm chủ được những vấn đề cơ bản, không chỉ phát triển Game cho máy tính cá nhân mà chúng ta còn có thể phát triển Game cho các hệ máy console và thiết bị di động.

### **1.3. Các đặc điểm và tính năng của Unity.**

#### **1.3.1. Rendering (Kết xuất hình ảnh).**

Giống như tất cả các Engine hoàn chỉnh khác, Unity hỗ trợ đầy đủ khả năng kết xuất hình ảnh (Redering) cùng nhiều hỗ trợ cho phép áp dụng các công nghệ phổ biến trong lĩnh vực đồ họa 3D nhằm cải thiện chất lượng hình ảnh. Các phiên bản gần đây nhất của Unity được xây dựng lại thuật toán nhằm cải thiện hiệu suất kết xuất hình ảnh đồng thời tăng cường chất lượng hình ảnh sau khi kết xuất.

Một số hỗ trợ:

- Unity cung cấp sẵn 100 Shaders với đầy đủ các loại phổ biến nhất.
- Hỗ trợ Surface Shaders, Occlusion Culling, GLSL Optimizer.
- Hỗ trợ LOD.

#### **1.3.2. Lighting (Ánh sáng).**

Ánh sáng là một điều thiết yếu giúp môi trường trở nên đẹp và thực tế hơn. Unity cũng cung cấp nhiều giải pháp đa dạng cho phép chúng ta áp dụng ánh sáng một cách tốt nhất vào môi trường trong trò chơi với nhiều loại nguồn sáng như ánh sáng có hướng (Directional Light), ánh sáng điểm (Point Light), ... Một số công nghệ và kỹ thuật về ánh sáng được Unity hỗ trợ: Lightingmapping, Realtime Shadows, hiệu ứng Sunshafts và Lens Flares.

### **1.3.3. Terrains (Địa hình).**

Terrains còn gọi chung là địa hình bao gồm phần đất nền của môi trường trong trò chơi cùng các đối tượng gắn liền như cây, cỏ, ...

Unity cung cấp một công cụ hỗ trợ rất tốt khả năng này với tên gọi là Terrains Tools cho phép chúng ta thiết kế địa hình với các công cụ vẽ dưới dạng Brush có nhiều thông số tùy chỉnh để tạo hình và lát Texture cho địa hình. Cùng với Terrain Tools là Tree Creator, một công cụ mạnh mẽ cho phép chúng ta tạo ra cây cối với hình dạng, kích thước và kiểu cách đa dạng.

### **1.3.4. Substances (Texture thông minh).**

Substances có thể hiểu đơn giản là một dạng tùy biến Textures nhằm làm đa dạng chúng trong nhiều điều kiện môi trường khác nhau. Unity cung cấp khả năng này thông qua các API dựng sẵn trong thư viện, hỗ trợ lập trình viên lập trình để tùy biến hình ảnh được kết xuất của Texture.

### **1.3.5. Physics (Vật lý).**

PhysX là một Engine mô phỏng và xử lí vật lý cực kỳ mạnh mẽ được phát triển bởi nhà sản xuất card đồ họa hàng đầu thế giới NVIDIA. Unity đã tích hợp Engine này vào để đảm nhận mọi vấn đề vật lý. Một số vấn đề vật lý được hỗ trợ bởi Unity như: Soft Bodies, Rigitbodies, Ragdolls, Joints, Cars,...

### **1.3.6. Pathfinding (Tìm Đường).**

Đây là một tính năng rất mới mẻ đến từ phiên bản Unity 3.5. Với các phiên bản trước, để phát triển khả năng tìm đường cho trí thông minh nhân tạo (AI), nhà phát triển phải hoàn toàn tự xây dựng cho mình một hệ thống tìm đường riêng biệt. Tuy nhiên ở phiên bản 3.5 đến nay, Unity hỗ trợ cho chúng ta tính năng Pathfinding cho phép tạo ra khả năng tìm đường cho AI nhờ vào khái niệm lưới định hướng (NavMesh).

### **1.3.7. Audio (âm thanh).**

Về âm thanh, Unity tích hợp FMOD – công cụ âm thanh thuộc hàng mạnh nhất hiện nay. Qua đó Unity hỗ trợ chúng ta nhập và sử dụng nhiều định dạng tập tin âm thanh khác nhau.

### **1.3.8. Programming (lập trình).**

Lập trình là một trong những yếu tố quan trọng nhất trong phát triển Game. Lập trình cho phép nhà phát triển tạo nên khả năng tương tác, trí thông minh và yếu tố Gameplay cho trò chơi.

Unity cho phép chúng ta lập trình bằng nhiều ngôn ngữ mạnh mẽ và phổ biến với các lập trình viên như: C#, Java Scrip và Boo.

### 1.3.9. Networking.

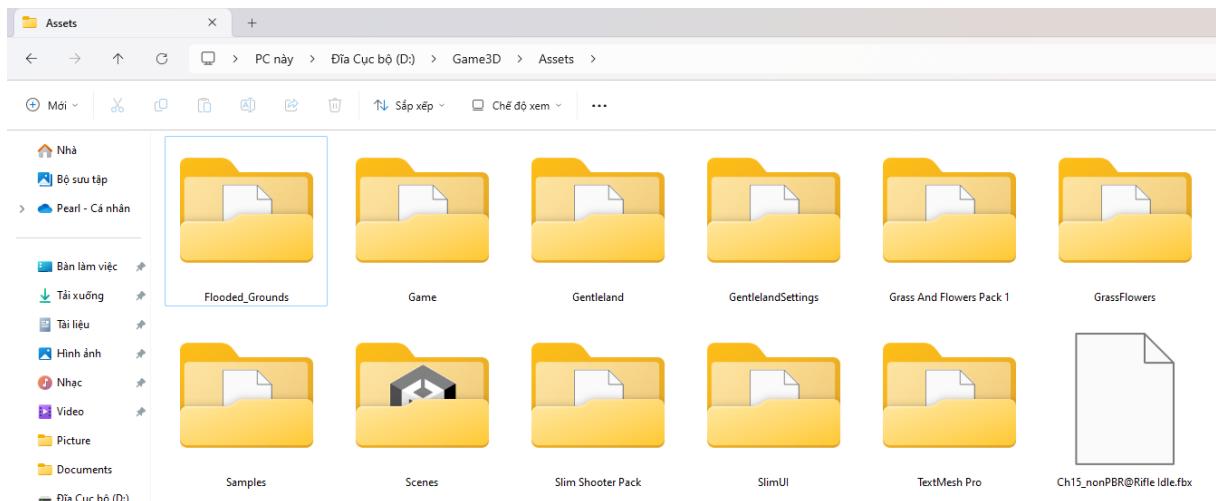
Networking cho phép chúng ta tạo ra các trò chơi trực tuyến (online) – một trong những thể loại trò chơi thu hút được nhiều người chơi nhất. Tính năng này sẽ hỗ trợ đầy đủ để chúng ta tạo nên các khía cạnh phổ biến trong Game online như hệ thống điểm kinh nghiệm, chat và tương tác thời gian thực, ...

Một số tính năng cung cấp bởi Networking như: State Synchronization, Realtime Networking, Remote Procedure Calls, Backend Connectivity, Web Browser Integration, Web Connectivity.

## 1.4. Các thành phần trong Unity.

### 1.4.1. Assets.

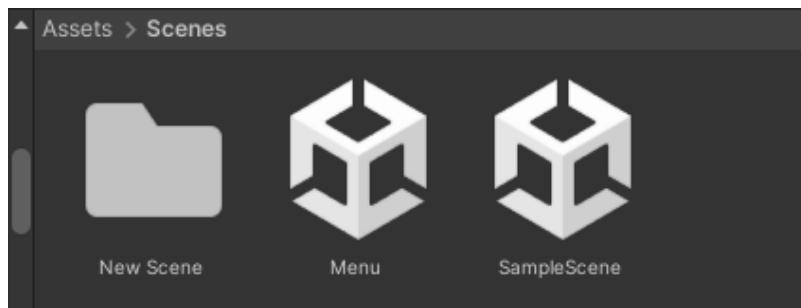
Assets là những tài nguyên xây dựng nên một dự án Unity. Từ những tập tin hình ảnh, mô hình 3D đến các tập tin âm thanh. Unity gọi các tập tin mà chúng ta dùng để tạo nên trò chơi là tài sản (Assets). Điều này lý giải tại sao tất cả các tập tin, thư mục của các dự án Unity đều được lưu trữ trong một thư mục có tên là "Assets".



Hình 1. 3: Asset trong Unity.

### 1.4.2. Scenes.

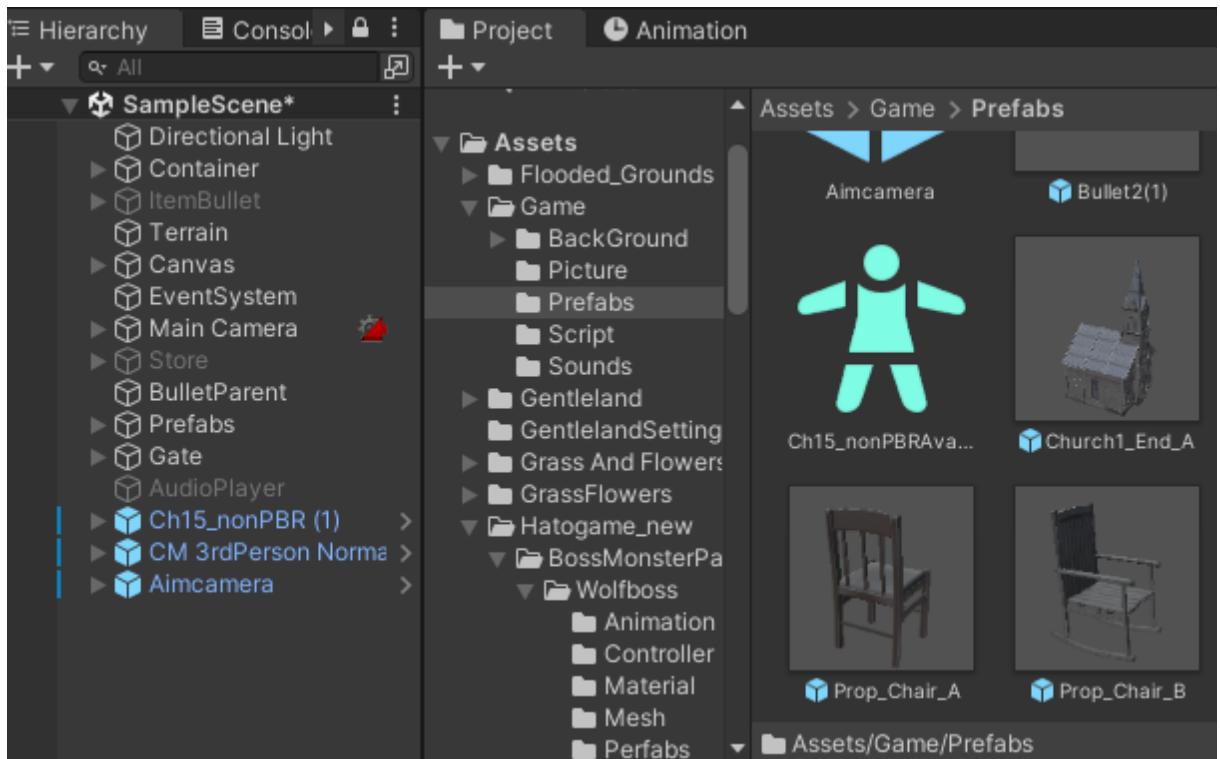
Trong Unity, chúng ta cần hiểu một cảnh (hay một phân đoạn) nghĩa là một màn chơi riêng biệt hoặc một khu vực hay thành phần có trong nội dung của trò chơi (ví dụ như Game menu). Bằng cách tạo nên nhiều Scenes cho trò chơi, chúng ta có thể phân phối thời gian tải hoặc kiểm tra các phần khác nhau của trò chơi một cách riêng lẻ.



Hình 1. 4: Các scene trong Unity.

### 1.4.3. Game Object.

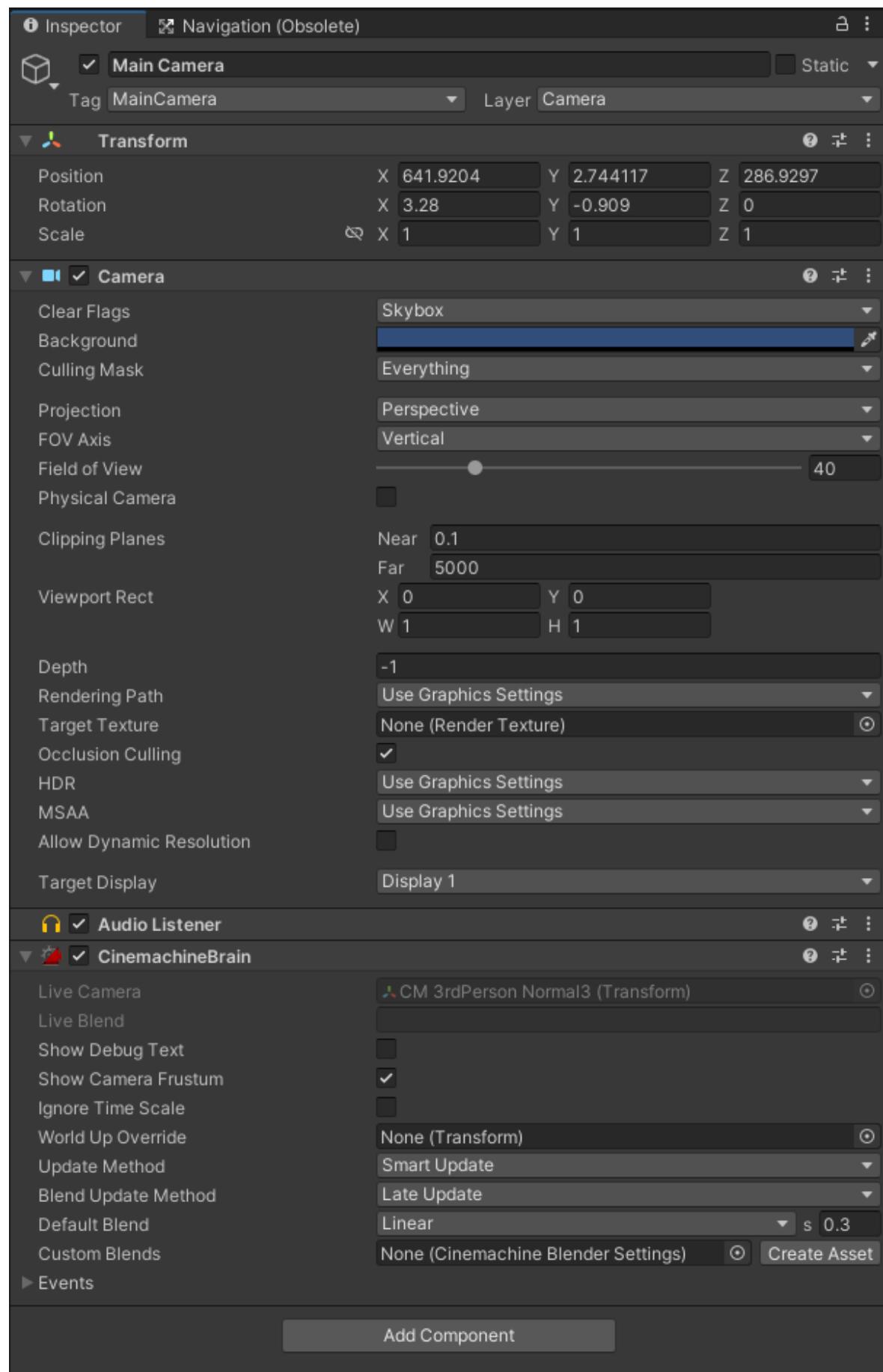
Khi Assets được sử dụng trong Scene, chúng trở thành Game Object – một thuật ngữ được sử dụng trong Unity (đặc biệt là trong mảng lập trình). Tất cả các Game Object đều chứa ít nhất một thành phần là Transform. Transform là thông tin về vị trí, góc xoay và tỉ lệ của đối tượng, tất cả được mô tả bởi bộ 3 số X, Y, Z trong hệ trục tọa độ. Thành phần này có thể được tùy biến lại trong quá trình lập trình nhằm thay đổi vị trí, góc quay và tỉ lệ của đối tượng qua các đoạn mã. Từ các thành phần cơ bản này, chúng ta sẽ tạo ra Game Object với các thành phần khác, bổ sung chức năng cần thiết để xây dựng nên bất kỳ một thành phần nào trong kịch bản Game mà chúng ta đã tưởng tượng.



Hình 1. 5: Kéo tài nguyên vào scene để sử dụng.

### 1.4.4. Components

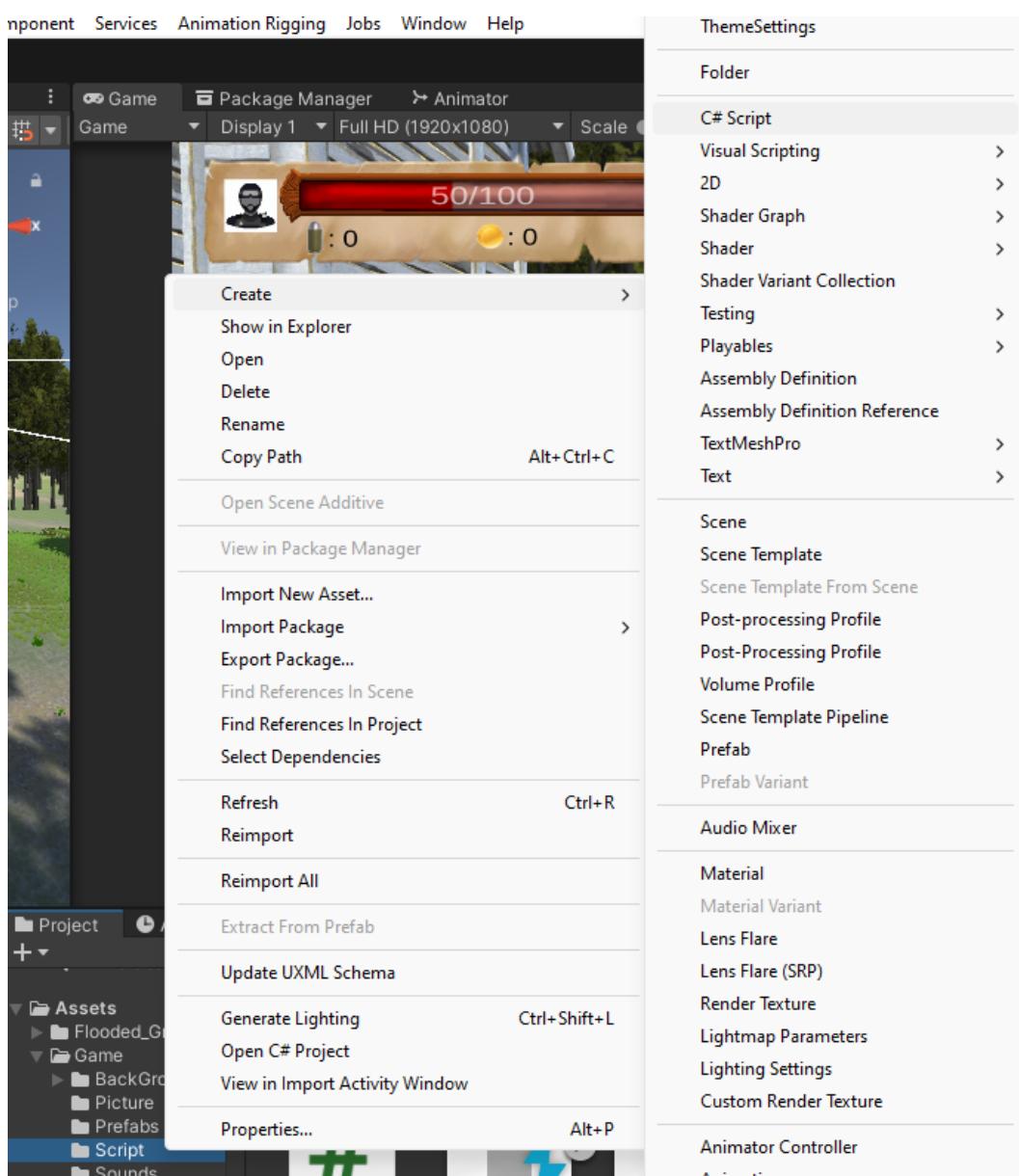
Components có nhiều hình thức khác nhau. Chúng có thể xác định hành vi, cách xuất hiện,... hay ảnh hưởng đến các khía cạnh khác trong chức năng của Game Object trong trò chơi. Bằng cách “gắn” chúng vào trong Game Object, chúng ta ngay lập tức có thể áp dụng tác động của chúng lên đối tượng. Những Components phổ biến trong quá trình phát triển trò chơi đều được Unity hỗ trợ sẵn. Ví dụ như thành phần Rigidbody đã được đề cập hay các yếu tố đơn giản khác như ánh sáng, Camera và nhiều thành phần khác. Để tạo nên các yếu tố tương tác trong trò chơi, chúng ta sẽ sử dụng Script (mã kịch bản), chúng cũng được xem như là một Components trong Unity.



Hình 1. 6:Các thành phần trong đối tượng camera.

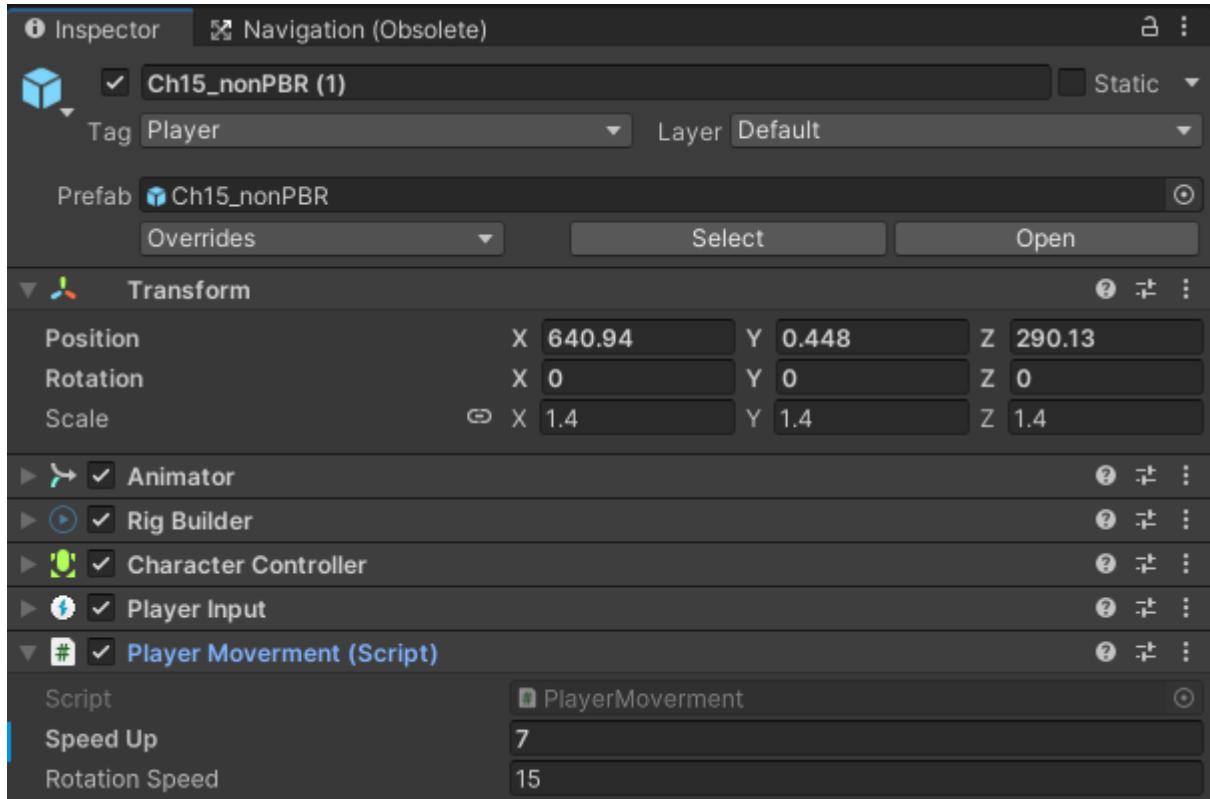
### 1.4.5. Scripts.

Được Unity xem như một Components, Script là một thành phần thiết yếu trong quá trình phát triển trò chơi và đáng được đề cập đến như một khái niệm “chìa khóa”. Unity cung cấp cho chúng ta khả năng viết Script bằng cả 3 loại ngôn ngữ là: JavaScript, C# và Boo (một dẫn xuất của ngôn ngữ Python). Unity không đòi hỏi chúng ta phải học làm thế nào để lập trình trong Unity, nhưng hầu như chúng ta phải sử dụng Script tại mỗi thành phần trong kịch bản mà chúng ta phát triển. Unity đã xây dựng sẵn một tập hợp đa dạng các lớp, hàm mà chúng ta hoàn toàn có thể ứng dụng trong quá trình lập trình cho trò chơi của mình. Để viết script, chúng ta sẽ làm việc với một trình biên tập Script độc lập của Unity, hoặc với chương trình Mono Developer được tích hợp và đồng bộ với Unity trong những phiên bản mới nhất hiện nay. Mono developer là một IDE khá tốt để lập trình khi cung cấp nhiều chức năng tương tự như Visual studio. Mã nguồn viết trên Mono Developer sẽ được cập nhật và lưu trữ trong dự án Unity.



Hình 1. 7: Cách tạo file Script mới.

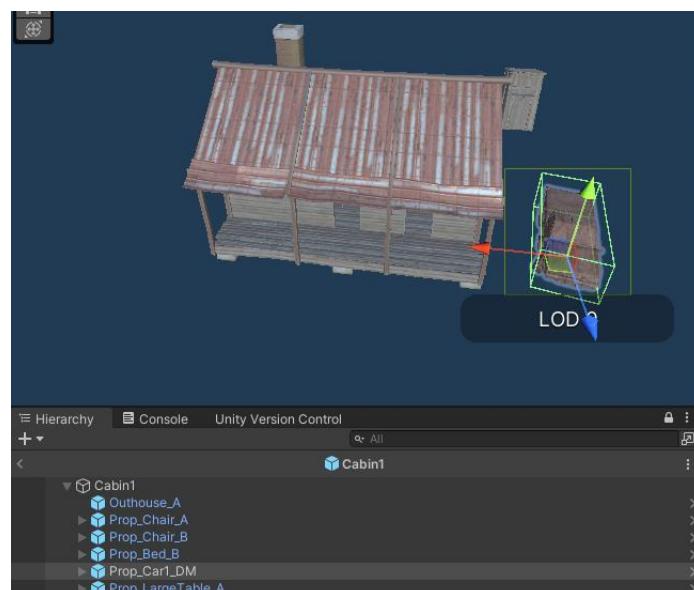
Một đoạn Script muốn thực thi được thì nó phải được gắn vào một đối tượng



Hình 1. 8: Một file Script đang gắn vào đối tượng.

#### 1.4.6. Prefabs.

Prefabs cho phép chúng ta lưu trữ các đối tượng với những Components và những thiết đặt hoàn chỉnh. Có thể so sánh với khái niệm cơ bản là MovieClip trong Adobe Flash, Prefabs chỉ đơn giản là một Container (một đối tượng chứa) rỗng mà chúng ta có thể đưa bất kì một đối tượng hay dữ liệu mẫu nào mà chúng ta muốn tái sử dụng về sau.

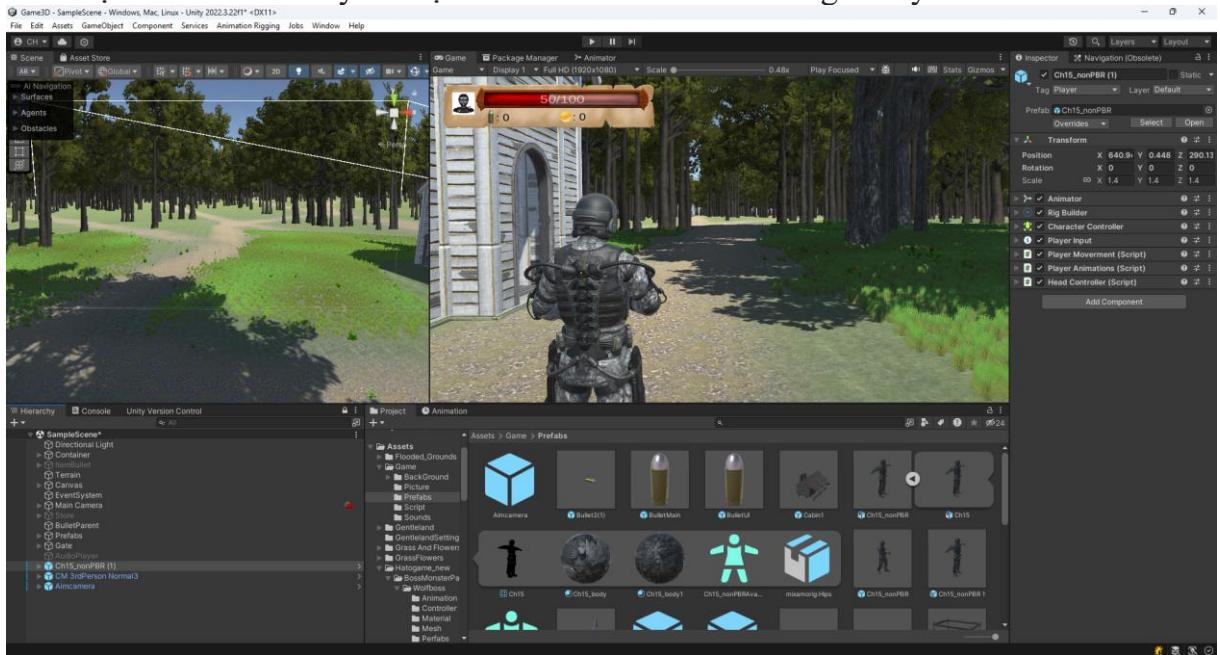


Hình 1. 9: Một số đối tượng trong Prefabs.

## 1.5. Giao diện của Unity.

### 1.5.1. Giao diện.

Giao diện của Unity có khả năng tùy chỉnh bố trí tương tự như nhiều môi trường làm việc khác. Dưới đây là một kiểu bố trí điển hình trong Unity:



Hình 1. 10: Giao diện của Unity.

#### Chú thích:

- **Scene:** Nơi mà trò chơi sẽ được xây dựng.
- **Hierarchy:** Danh sách các Game Object trong scene.
- **Inspector:** Những thiết lập, thành phần, thuộc tính của đối tượng (hoặc Asset) đang được chọn.
- **Game:** Cửa sổ xem trước, nó chỉ hoạt động trong chế độ “Play” (Preview – xem trước).
- **Project:** Danh sách các Assets của dự án, được ví như thư viện của dự án.

### 1.5.2. Cửa sổ Scene và Hierarchy.

Cửa sổ scene là nơi mà chúng ta sẽ xây dựng các thực thể, đối tượng của dự án vào đó. Cửa sổ cung cấp góc nhìn phối cảnh (Perspective (góc nhìn 3D), chúng ta có thể chuyển qua các góc nhìn khác như từ trên xuống hoặc từ dưới lên (Top Down), từ trái sang phải hoặc phải sang trái (Side On), từ trước ra sau hoặc sau đến trước (Front On). Cửa sổ này sẽ kết hợp xuất đầy đủ những hình ảnh trong thế giới của trò chơi mà chúng ta tạo ra dưới dạng một vùng biên tập mà chúng ta có thể biên tập, chỉnh sửa trực tiếp thế giới đó.

Khi kéo thả Asset vào cửa sổ Scene, Assets sẽ trở thành Game Object. Cửa sổ Scene được ràng buộc cùng với cửa sổ Hierarchy, cửa sổ Hierarchy liệt kê danh sách các Game Object có trong Scene và được sắp xếp theo thứ tự chữ cái từ A-Z.



Hình 1. 11:Các nút chức năng cho cửa Scene.

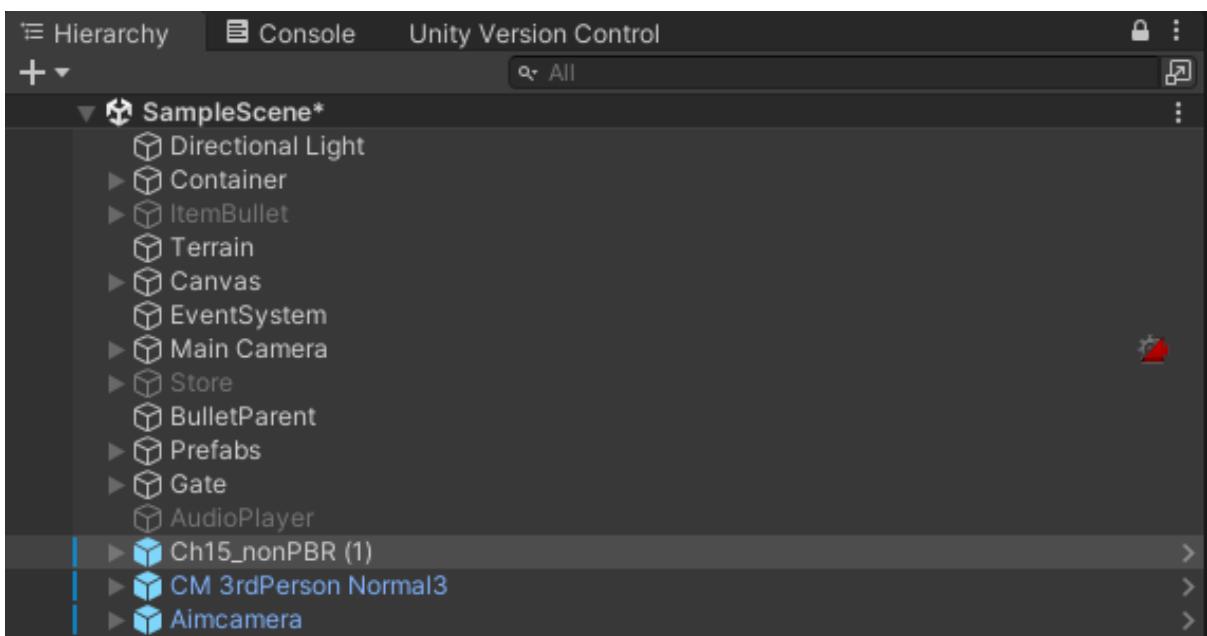
Cửa sổ Scene còn đi kèm với 4 bốn nút chức năng hữu ích được hiển thị dưới dạng hình ảnh như trên. Chúng có thể được lựa chọn thông qua các phím tắt Q, W, E và R. Những nút này có các chức năng như sau:

**Công cụ bàn tay (Q):** Công cụ này cho phép chúng ta di chuyển đến một khu vực nào đó trong Scene bằng thao tác kéo thả thuộc trái.

**Công cụ di chuyển (W):** Công cụ này cho phép chúng ta chọn một đối tượng trong cảnh và thực hiện thao tác di chuyển, thay đổi vị trí của đối tượng đó. Khi chọn, tại vị trí của đối tượng sẽ hiển thị các trục và mặt phẳng gắn liền với đối tượng cho phép chúng ta di chuyển đối tượng trượt theo các trục, mặt phẳng hoặc di chuyển một cách tùy ý.

**Công cụ xoay (E):** Công cụ này có đặc điểm và cách sử dụng giống với công cụ di chuyển, tuy nhiên thay vì để di chuyển vị trí của đối tượng thì công cụ này giúp chúng ta xoay đối tượng xoay quanh trục hay tâm của đối tượng.

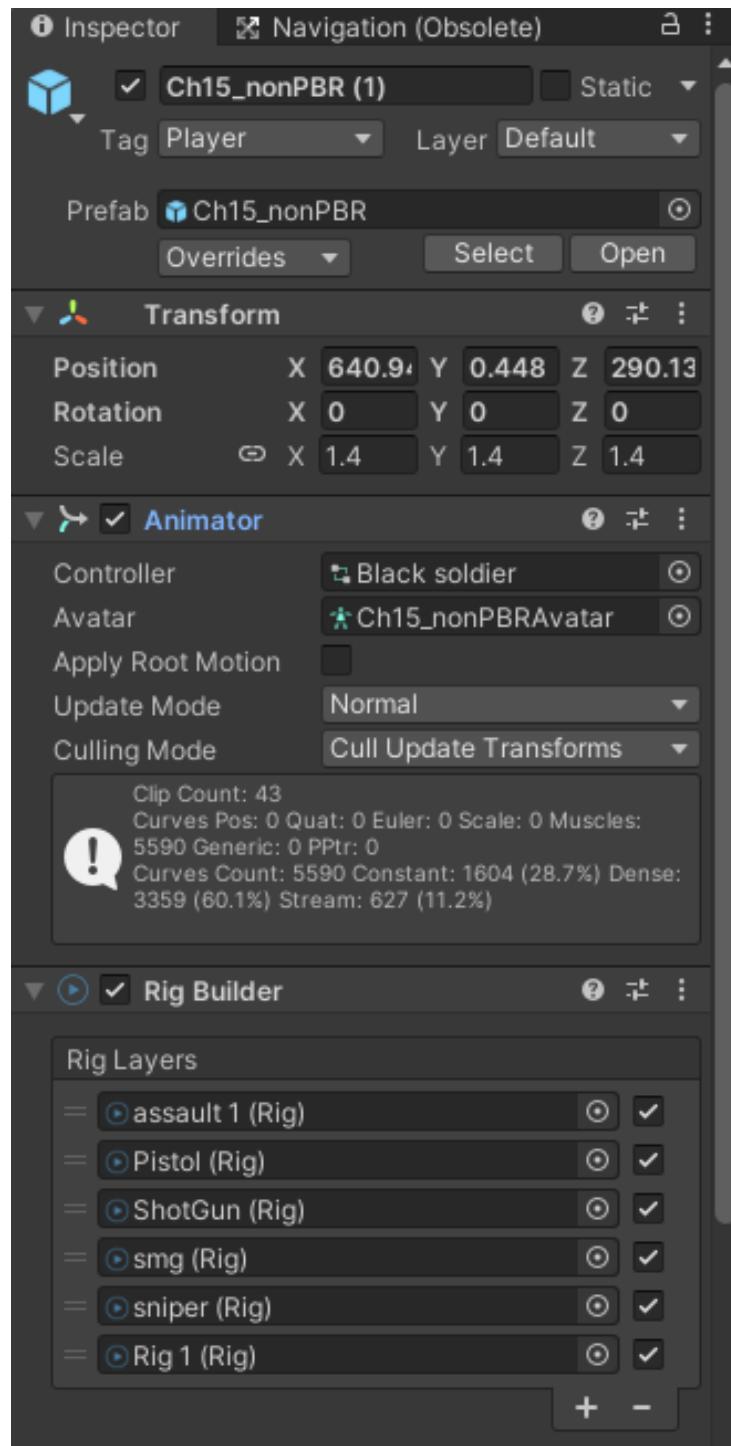
**Công cụ điều chỉnh tỉ lệ (R):** Cũng tương tự như công cụ di chuyển và xoay, công cụ này cho phép chúng ta tùy chỉnh kích thước, tỉ lệ của đối tượng một cách tùy ý.



Hình 1. 12: Cửa sổ Hierarchy.

### 1.5.3. Cửa sổ Inspector.

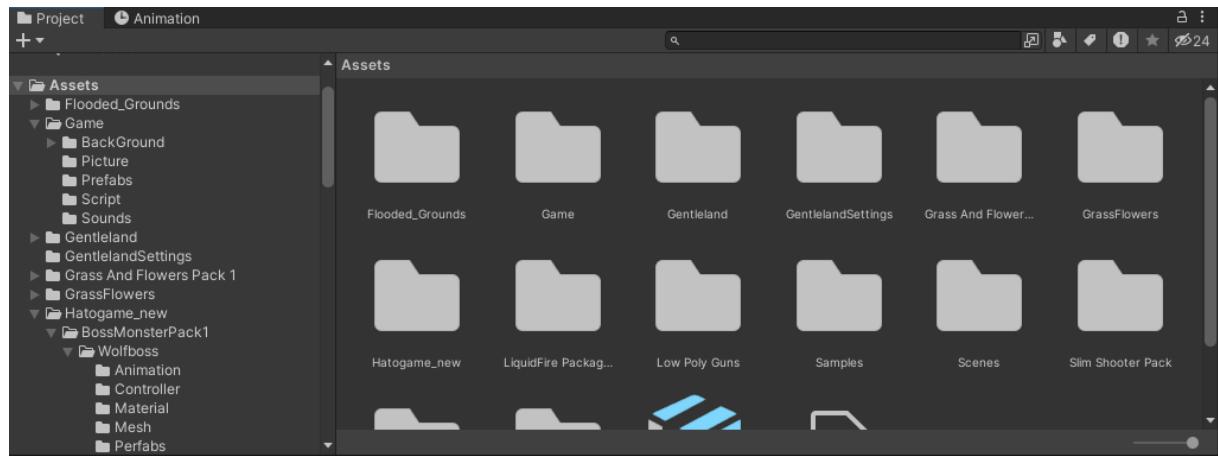
Cửa sổ Inspector có thể được xem như một công cụ cho phép chúng ta tùy chỉnh các thiết đặt, các thành phần của Game Object hoặc Assets đang được chọn. Cửa sổ này sẽ hiển thị đầy đủ các Components của đối tượng mà chúng ta chọn. Nó cho phép chúng ta điều chỉnh các biến của Components dưới các hình thức như: Textbox, Slider, Button, Drop-down Menu... Ngoài việc hiển thị các Component của đối tượng được chọn, cửa sổ Inspector còn hiển thị các thiết đặt chung của hệ thống hay của trò chơi khi ta chọn chúng từ menu Edit.



Hình 1. 13: Cửa sổ Inspector.

Trong hình trên, chúng ta thấy cửa sổ Inspector đang hiển thị một vài thuộc tính, Components của một đối tượng đang được chọn. Trong đó, bao gồm 2 Components là Transform và Animation. Cửa sổ Inspector sẽ cho phép chúng ta thay đổi các thiết đặt trên. Các Components này còn có thể được tạm thời vô hiệu hóa vào bất kỳ lúc nào chúng ta muốn bằng cách bỏ chọn Checkbox ở góc trên bên trái của mỗi Component, việc này sẽ rất hữu ích cho chúng ta khi muốn kiểm tra hay thử nghiệm các Components này. Ngoài ra, cửa Inspector còn cho phép chúng ta vô hiệu hóa toàn bộ một đối tượng đang được chọn bằng cách bỏ chọn Checkbox ở trên cùng góc trái của cửa sổ Inspector.

#### 1.5.4. Cửa sổ Project.



Hình 1. 14: Cửa sổ Project.

Cửa sổ Project là cửa sổ cho phép chúng ta nhìn thấy trực tiếp nội dung của thư mục Assets của dự án. Mỗi dự án Unity đều được chứa trong một thư mục cha. Trong đó có 3 thư mục con là Assets, Library và Temp (chỉ có khi Unity đang chạy). Đặt tất cả các Assets vào thư mục Assets có nghĩa là ngay lập tức chúng ta sẽ thấy chúng xuất hiện trong cửa sổ Project. Ngoài ra, khi thay đổi vị trí của Assets trong thư mục Assets hay lưu tập tin lại từ một chương trình ứng dụng thứ 3 nào khác (ví dụ như Photoshop), sẽ làm cho Unity nhập lại (Re-Import) Assets, phản ánh sự thay đổi này ngay lập tức trong cửa sổ Project và Scene có sử dụng Assets vừa được thay đổi.

Cửa sổ Project được tích hợp nút Create, nút này cho phép chúng ta tạo mới bất kì một Assets nào mới nào, ví dụ như Script, Prefabs, Materials, ...

#### 1.5.5. Cửa sổ Game.

Cửa sổ Game sẽ được gọi khi chúng ta nhấn vào nút Play (là một hành động thực hiện test trò chơi). Cửa sổ này cho phép chúng ta tùy chọn về thiết đặt tỉ lệ màn hình, nó phản ánh phạm vi trong Scene mà người chơi có thể thấy được với mỗi tỉ lệ màn hình tương ứng, ví dụ như với mỗi tỉ lệ màn hình 4:3, 16:9 thì người chơi sẽ có một phạm vi nhìn thấy khác nhau.

Sau khi nhấn vào nút Play, chúng ta sẽ ở chế độ Testing, lúc này mọi thay đổi về các thuộc tính, Components, ... của đối tượng sẽ chỉ là tạm thời. Tức là chúng sẽ trở về như ban đầu (trước khi nhấn nút Play) sau khi kết thúc chế độ Testing.



Hình 1. 15: Các loại hình ảnh trong cửa sổ game.

## 1.6. Kiến thức về thiết kế trò chơi 3D.

Thiết kế trò chơi 3D là một lĩnh vực phức tạp và bao gồm nhiều khía cạnh kỹ thuật cũng như nghệ thuật. Dưới đây là các kiến thức riêng cần thiết để thiết kế trò chơi 3D:

### 1.6.1. Mô hình hóa 3D (3D Modeling).

Mô hình hóa 3D là quá trình tạo ra các đối tượng ba chiều trong không gian ảo. Các công cụ phổ biến như Blender, Maya, 3Ds Max, và ZBrush được sử dụng để tạo ra các mô hình 3D.

**Lưới đa giác (Polygon Mesh):** Các mô hình 3D được xây dựng từ các lưới đa giác. Việc tối ưu hóa số lượng đa giác là rất quan trọng để đảm bảo chi tiết mô hình mà không làm giảm hiệu suất trò chơi.

**Kết cấu (Texturing):** Áp dụng các hình ảnh hoặc màu sắc lên bề mặt của mô hình 3D để tạo ra vẻ ngoài chân thực. Các phần mềm như Substance Painter và Photoshop thường được sử dụng cho quá trình này.

**Rigging và Animation:** Rigging là việc tạo khung xương cho mô hình để chuẩn bị cho quá trình hoạt hình. Animation là việc tạo các chuyển động và hành vi cho mô hình, sử dụng các công cụ như Maya và Blender.

### 1.6.2. Ánh sáng và Hiệu ứng (Lighting and Effects).

Ánh sáng và hiệu ứng đóng vai trò quan trọng trong việc tạo ra không gian và cảm giác chân thực cho trò chơi.

**Ánh sáng (Lighting):** Unity cung cấp các loại ánh sáng như ánh sáng điểm (point light), ánh sáng khu vực (area light), và ánh sáng định hướng (directional light). Ánh sáng động và tinh giáp giúp tối ưu hóa hiệu suất trong khi vẫn duy trì chất lượng hình ảnh.

**Hiệu ứng hậu kỳ (Post-Processing Effects):** Các hiệu ứng như đổ bóng, làm mờ (blur), độ sâu trường ảnh (depth of field), và hiệu ứng bloom giúp cải thiện chất lượng hình ảnh và tạo nên cảm giác điện ảnh.

### 1.6.3. Cơ chế vật lý (Physics).

Cơ chế vật lý là yếu tố quan trọng để tạo ra các chuyển động và tương tác chân thực trong trò chơi.

**Động lực học (Dynamics):** Unity sử dụng các engine vật lý như PhysX để mô phỏng các hiện tượng vật lý như trọng lực, va chạm, và lực.

**Va chạm (Collision Detection):** Đảm bảo rằng các đối tượng trong trò chơi có thể tương tác và va chạm với nhau một cách hợp lý. Unity cung cấp nhiều loại collider (hộp, cầu, lưỡi) để xử lý va chạm.

### 1.6.4. Trí tuệ nhân tạo (AI).

Trí tuệ nhân tạo (AI) tạo ra các nhân vật không phải người chơi (NPC) có thể tương tác và thách thức người chơi.

**Hành vi NPC (NPC Behavior):** Sử dụng các thuật toán AI để lập trình các hành vi như tuần tra, tấn công, phòng thủ và tìm đường (pathfinding). Unity cung cấp công cụ NavMesh để hỗ trợ tìm đường.

### 1.6.5. Gameplay và Cơ chế trò chơi (Gameplay and Mechanics).

Gameplay là cốt lõi của bất kỳ trò chơi nào, bao gồm các quy tắc, mục tiêu và cách thức tương tác của người chơi với trò chơi.

**Thiết kế cấp độ (Level Design):** Xây dựng các cấp độ và môi trường trong trò chơi sao cho thú vị và thách thức. Level design bao gồm việc tạo ra địa hình, bố trí kẻ thù, và đặt các vật phẩm.

**Cơ chế trò chơi (Game Mechanics):** Xác định các quy tắc và hệ thống chi phối hành động và tương tác trong trò chơi. Điều này bao gồm hệ thống chiến đấu, hệ thống giải đố, và các cơ chế khác như nhảy, chạy, và né tránh.

### 1.6.6. Âm thanh và Nhạc nền (Audio and Music).

Âm thanh và nhạc nền đóng vai trò quan trọng trong việc tạo nên không khí và cảm xúc cho trò chơi.

**Hiệu ứng âm thanh (Sound Effects):** Các hiệu ứng âm thanh như tiếng bước chân, tiếng súng, và âm thanh môi trường giúp tăng cường tính chân thực và hấp dẫn.

**Nhạc nền (Background Music):** Nhạc nền phải phù hợp với từng giai đoạn và tình huống trong trò chơi, từ những bản nhạc hào hứng trong các trận đánh đến những giai điệu nhẹ nhàng trong các cảnh yên bình.

### 1.6.7. Giao diện người dùng (UI).

Giao diện người dùng (UI) bao gồm các menu, biểu tượng, và các yếu tố điều khiển khác mà người chơi tương tác.

**Thiết kế UI:** UI phải trực quan, dễ sử dụng và phù hợp với phong cách tổng thể của trò chơi. Unity cung cấp nhiều công cụ để thiết kế và triển khai UI, bao gồm Unity UI và TextMeshPro.

**Trải nghiệm người dùng (UX):** Đảm bảo rằng người chơi có thể dễ dàng điều hướng và tương tác với trò chơi. Điều này bao gồm việc kiểm tra và tinh chỉnh các yếu tố UI để đảm bảo chúng hoạt động một cách mượt mà và không gây khó khăn cho người chơi.

Tóm lại, kiến thức về thiết kế trò chơi 3D bao gồm hiểu biết về mô hình hóa 3D, ánh sáng, cơ chế vật lý, AI, gameplay, âm thanh và UI. Các kỹ năng và kiến thức này là cần thiết để tạo ra một trò chơi 3D hấp dẫn và chất lượng cao.

## **CHƯƠNG 2: TỔNG QUAN TRÒ CHƠI**

### **2.1. Mục tiêu và ý nghĩa.**

Trong chương này em sẽ bắt đầu với việc nhấn mạnh vào mục tiêu và ý nghĩa của dự án thiết kế và lập trình trò chơi Aliens War 3D. Dự án này không chỉ là một nhiệm vụ đơn thuần mà còn là cơ hội để thể hiện sự sáng tạo và kiến thức mà em đã tích lũy trong quá trình học tập.

Mục tiêu hàng đầu của em là tạo ra một trò chơi hấp dẫn và độc đáo. Bằng cách tạo ra một trải nghiệm chơi game tốt nhất có thể, em hy vọng sẽ mang lại niềm vui và thử thách cho người chơi, đồng thời đem lại cho em cơ hội để phát triển kỹ năng và kiến thức trong lĩnh vực công nghệ thông tin.

Ngoài ra, dự án cũng mang ý nghĩa lớn đối với sự phát triển cá nhân của em. Qua việc tham gia vào quá trình phát triển trò chơi này, em sẽ có cơ hội thực hành và nâng cao kỹ năng lập trình, thiết kế và quản lý dự án.

Bằng cách tạo ra một sản phẩm chất lượng và mang tính sáng tạo, em hy vọng sẽ góp phần vào việc phát triển ngành công nghiệp game cũng như cộng đồng game thủ. Điều này sẽ là một minh chứng rõ ràng cho sự cam kết và nỗ lực của em trong lĩnh vực này, đồng thời mang lại niềm tự hào và động lực cho sự phát triển cá nhân của em trong tương lai.

Do đó, việc hiểu rõ và đặt ra mục tiêu rõ ràng cho dự án Aliens War 3D là bước quan trọng để em có thể tiến xa hơn trong quá trình thực hiện dự án và đạt được kết quả mong đợi.

### **2.2. Phạm vi đề tài.**

Trong phần này, em sẽ mô tả các thành phần chính của trò chơi Aliens War 3D, bao gồm đồ họa, âm nhạc, cốt truyện và gameplay, để hiểu rõ hơn về tổng thể của dự án. Chúng ta sẽ xác định phạm vi dự án và những yếu tố quyết định về mức độ hoàn thiện mong muốn, nhằm đảm bảo rằng trò chơi đáp ứng được mong đợi của người chơi.

#### **2.2.1. Đồ họa.**

Aliens War 3D sử dụng đồ họa 3D chất lượng cao để tái hiện một thế giới hậu tận thế chân thực và sống động.

Các mô hình nhân vật và quái vật được thiết kế cẩn thận, mang lại trải nghiệm hấp dẫn cho người chơi.

#### **2.2.2. Âm nhạc.**

Âm nhạc và hiệu ứng âm thanh được tích hợp một cách chuyên nghiệp, tạo ra không khí căng thẳng và phấn khích trong trò chơi.

Sự lựa chọn âm nhạc và âm thanh phản ánh chính xác tình hình và cảm xúc của từng tình huống trong trò chơi.

#### **2.2.3. Cốt truyện.**

Cốt truyện của Aliens War 3D xoay quanh cuộc chiến đấu giữa con người và người ngoài hành tinh Xeno Horde.

Người chơi sẽ vào vai một lính đánh thuê, nhiệm vụ của họ là tiêu diệt toàn bộ lực lượng xâm lược và giải cứu thế giới khỏi sự thống trị của chúng.

#### **2.2.4. Gameplay.**

Aliens War 3D mang lại một trải nghiệm gameplay đa dạng và phong phú, kết hợp các yếu tố của thể loại hành động, bắn súng, nhập vai và sinh tồn.

Góc nhìn thứ 3 mang lại cái nhìn tổng thể và chi tiết cho người chơi trong các tình huống đa dạng.

#### **2.2.5. Phạm vi dự án và mức độ hoàn thiện.**

Phạm vi dự án bao gồm việc phát triển và hoàn thiện các thành phần đã đề cập, đảm bảo rằng trò chơi đáp ứng được mong đợi của người chơi.

Yếu tố quyết định về mức độ hoàn thiện bao gồm thời gian, nguồn lực và mục tiêu của dự án, nhằm đảm bảo rằng trò chơi được phát triển một cách hiệu quả và có chất lượng cao.

### **2.3. Phương pháp nghiên cứu.**

Để tiến hành nghiên cứu và phát triển trò chơi Aliens War 3D, em sẽ sử dụng các công nghệ và phương pháp sau đây, với sự tập trung chủ yếu vào sự tích hợp của tính logic và sáng tạo trong thiết kế cơ chế gameplay:

#### **2.3.1. Sử dụng Unity Game Engine.**

Unity là một nền tảng phát triển trò chơi mạnh mẽ với khả năng hỗ trợ đa nền tảng. Em sẽ tận dụng sức mạnh của Unity để xây dựng và triển khai trò chơi Aliens War 3D trên nhiều nền tảng khác nhau, bao gồm cả di động và desktop.

#### **2.3.2. Ngôn Ngữ Lập Trình C#.**

Unity hỗ trợ việc lập trình bằng ngôn ngữ C#, một ngôn ngữ mạnh mẽ và phổ biến trong việc phát triển trò chơi. Em sẽ sử dụng C# để viết mã cho các tính năng gameplay, xử lý logic và tương tác giữa các phần tử trong trò chơi.

#### **2.3.3. Thiết Kế Gameplay.**

Tính Logic: Em sẽ áp dụng các nguyên lý lập trình và thiết kế logic để xây dựng các cơ chế gameplay như hệ thống di chuyển, phản ứng của quái vật và hệ thống vũ khí.

Sự Sáng Tạo: Đồng thời, sự sáng tạo sẽ được áp dụng trong việc tạo ra các yếu tố gameplay độc đáo và hấp dẫn, bao gồm cốt truyện, môi trường, và các yếu tố sinh động.

#### **2.3.4. Xây Dựng Mô Hình và Animation.**

Em sẽ sử dụng các công cụ tích hợp trong Unity như Blender để xây dựng mô hình 3D cho nhân vật, quái vật và môi trường. Sự sáng tạo sẽ được thể hiện trong việc tạo ra các thiết kế độc đáo và phong phú.

### **2.3.5. Tạo Âm Nhạc và Âm Thanh.**

Sử dụng các công cụ tích hợp âm nhạc và âm thanh trong Unity, em sẽ tạo ra các hiệu ứng âm thanh độc đáo và phù hợp với cảnh quan và tình huống trong trò chơi.

### **2.3.6. Kiểm Thủ và Đánh Giá.**

Em sẽ sử dụng phương pháp kiểm thử thực tế để đảm bảo tính chất linh hoạt và hiệu suất của trò chơi. Phản hồi từ người chơi sẽ được thu thập để đánh giá và điều chỉnh trò chơi sao cho phù hợp và hấp dẫn nhất.

## **2.4. Bối cảnh trò chơi và mạch trò chơi.**

### **2.4.1. Bối cảnh câu chuyện.**

Trong tương lai không xa, Trái Đất bị xâm lược bởi một đạo quân người ngoài hành tinh tàn bạo, được biết đến với tên gọi Xeno Horde. Với vũ khí tiên tiến và công nghệ chiến đấu vượt trội, Xeno Horde tàn phá mọi thứ trên hành tinh mà không hề tỏ lòng thương xót.

Người chơi đảm nhận vai trò của Alex Hunter, một lính đánh thuê hàng đầu được giao nhiệm vụ duy nhất là ngăn chặn cuộc xâm lược của Xeno Horde và bảo vệ Trái Đất. Trong hành trình của mình, Alex phải chiến đấu qua những khu vực đầy nguy hiểm, từ thành phố bị tàn phá đến các cơ sở quân sự bị tấn công, đôi mặt với hàng tá quái vật ngoài hành tinh và thủ lĩnh của chúng.

Trong suốt cuộc hành trình, người chơi sẽ gặp phải nhiều thử thách khốc liệt, nhưng cũng có cơ hội thu thập đạn dược và nâng cấp kỹ năng để trở nên mạnh mẽ hơn trong cuộc chiến chống lại Xeno Horde. Cuộc hành trình sẽ thử thách sức mạnh và quyết tâm của Alex, đặc biệt là trong cuộc chiến cuối cùng với lực lượng lớn mạnh nhất của Xeno Horde để giải cứu Trái Đất khỏi sự tàn phá.

### **2.4.2. Bối cảnh nhân vật.**

Nhân vật chính của chúng ta là Alex Hunter, một lính đánh thuê chuyên nghiệp với quá khứ đầy bí ẩn và kỹ năng chiến đấu vô cùng tài năng. Trước khi trở thành lính đánh thuê, Alex từng là một cựu binh nhưng đã rời khỏi quân đội sau một vụ vi phạm không rõ nguyên nhân. Sau đó, anh gia nhập một tổ chức tư nhân chuyên thuê lính đánh thuê cho các nhiệm vụ nguy hiểm và bí mật trên khắp thế giới.

Câu chuyện bắt đầu khi Alex nhận được một cuộc gọi bí ẩn từ Dr. Emily Brooks, một nhà khoa học tài năng và nổi tiếng. Dr. Brooks đã biết về cuộc xâm lược người ngoài hành tinh đang diễn ra trên Trái Đất và cần sự giúp đỡ của Alex. Cô giao cho Alex một nhiệm vụ khẩn cấp: tiêu diệt toàn bộ lực lượng người ngoài hành tinh và ngăn chung xâm lược Trái Đất.

Alex chấp nhận nhiệm vụ mà không hề biết về những nguy hiểm đang chờ đợi mình. Anh bắt đầu cuộc hành trình ở khu rừng nơi quái vật xuất hiện, nơi anh phải đối mặt với những cuộc chiến căng thẳng và nguy hiểm. Trong quá trình điều tra và chiến đấu, Alex bắt đầu khám phá ra những bí mật sâu kín liên quan đến cuộc xâm lược, bao gồm một âm mưu lớn hơn và sự thật đen tối đang được giấu kín từ lâu.

Với sự giúp đỡ của Dr. Brooks và sự sẵn lòng hy sinh của mình, Alex sẽ phải vượt qua mọi thử thách để tiêu diệt lực lượng người ngoài hành tinh và bảo vệ Trái

Đất khỏi nguy cơ đe dọa. Cuộc hành trình của Alex sẽ đưa anh qua nhiều địa điểm khác nhau trên toàn cầu, từ căn cứ quân sự bí mật đến các thành phố lớn và thậm chí là không gian ngoài lề của hành tinh.

### 2.4.3. Mạch trò chơi.

Mạch trò chơi Aliens War 3D được thiết kế để mang đến cho người chơi một trải nghiệm hành động kịch tính và liên tục:

**Giới thiệu và hướng dẫn:** Người chơi bắt đầu bằng việc nhận nhiệm vụ từ Dr. Emily Brooks, làm quen với cách điều khiển và cơ chế cơ bản của trò chơi.

**Cuộc chiến tại khu rừng:** Alex bắt đầu cuộc hành trình của mình tại một khu rừng bị tàn phá bởi sự xuất hiện của quái vật ngoài hành tinh. Đây là nơi người chơi làm quen với hệ thống chiến đấu và các thử thách đầu tiên.

**Thành phố bị tàn phá:** Sau khi vượt qua khu rừng, người chơi tiến vào một thành phố bị tàn phá, nơi quái vật và lực lượng Xeno Horde chiếm đóng. Tại đây, người chơi sẽ phải đối mặt với những kẻ thù mạnh hơn và thu thập tài nguyên để nâng cấp vũ khí và kỹ năng.

**Cơ sở quân sự bị tấn công:** Alex tiến vào các cơ sở quân sự để tìm kiếm thông tin và giải cứu các nhân vật quan trọng. Đây là những màn chơi có độ khó cao hơn, đòi hỏi sự kết hợp chiến thuật và kỹ năng chiến đấu tốt hơn.

**Tiết lộ âm mưu và cuộc chiến cuối cùng:** Khi tiến gần hơn đến trung tâm của cuộc xâm lược, Alex phát hiện ra âm mưu lớn hơn và sự thật đen tối về cuộc xâm lược này. Trận chiến cuối cùng với lực lượng mạnh nhất của Xeno Horde diễn ra tại căn cứ chính của chúng, quyết định số phận của Trái Đất.

**Giải cứu Trái Đất:** Sau khi đánh bại lực lượng Xeno Horde, Alex hoàn thành nhiệm vụ của mình và giải cứu Trái Đất. Trò chơi kết thúc với cảnh báo về những nguy cơ tương lai và sự sẵn sàng của Alex để bảo vệ Trái Đất trong những thử thách tiếp theo.

Mạch trò chơi Aliens War 3D mang đến cho người chơi một hành trình hấp dẫn và đầy thử thách, kết hợp giữa hành động kịch tính và cốt truyện sâu sắc.

## 2.5. Giới thiệu nhân vật và môi trường trong trò chơi.

### 2.5.1. Nhân vật chính.

**Nhân vật chính:** Alex Hunter.

- **Vai trò:** Nhân vật chính của trò chơi, người chơi sẽ nhập vai vào Alex Hunter.
- **Tiểu sử:** Alex Hunter là một lính đánh thuê chuyên nghiệp với quá khứ đầy bí ẩn. Anh từng là một cựu binh, sau đó tham gia vào một tổ chức tư nhân chuyên thuê lính đánh thuê cho các nhiệm vụ nguy hiểm và bí mật trên khắp thế giới.
- **Kỹ năng:** Alex có khả năng sử dụng nhiều loại vũ khí và kỹ năng chiến đấu đa dạng. Anh ta thông minh, nhanh nhẹn và có khả năng chiến đấu tuyệt vời, điều này giúp anh trở thành một chiến binh đáng gờm trong cuộc chiến chống lại Xeno Horde.

## 2.5.2. Nhân vật phụ.

### Quái vật 1: Wolf.

- **Đặc điểm:** Wolf là những quái vật ngoài hành tinh có vóc dáng to lớn và sức tấn công mạnh mẽ. Chúng di chuyển nhanh và tấn công với lực sát thương cao.
- **Vai trò:** Wolf thường xuất hiện ở các khu vực nguy hiểm, tạo ra những cuộc tấn công bất ngờ và đe dọa tính mạng của Alex. Người chơi cần sử dụng chiến thuật và kỹ năng chiến đấu để đánh bại chúng.

### Quái vật 2: Catfish.

- **Đặc điểm:** Catfish là những kẻ thù có hình dáng giống cá trê với khả năng bơi lội và tấn công từ dưới nước. Chúng có thể phóng ra những đợt sóng xung kích gây tổn thương nặng nề cho kẻ địch.
- **Vai trò:** Catfish thường xuất hiện ở các khu vực gần nước, buộc người chơi phải cảnh giác và chuẩn bị cho các cuộc tấn công dưới nước. Đánh bại Catfish đòi hỏi người chơi phải linh hoạt và có chiến lược hợp lý.

### Quái vật 3: Rake.

- **Đặc điểm:** Rake là những sinh vật ngoài hành tinh có ngoại hình gầy gò nhưng sở hữu móng vuốt sắc bén và khả năng di chuyển nhanh nhẹn. Chúng thường tấn công theo đàn, gây áp lực lớn lên người chơi.
- **Vai trò:** Rake xuất hiện ở nhiều môi trường khác nhau, từ khu rừng rậm rạp đến các cơ sở bị tàn phá. Chúng là kẻ thù phổ biến và nguy hiểm, đòi hỏi người chơi phải luôn sẵn sàng cho các cuộc tấn công đồng loạt.

## 2.5.3. Môi trường trong trò chơi.

### Môi trường trong trò chơi: Khu rừng.

- **Mô tả:** Khu rừng là nơi mà quái vật ngoài hành tinh xuất hiện đầu tiên. Đây là khu vực đầy nguy hiểm với nhiều quái vật rình rập. Khu rừng có những tòa nhà nơi người chơi có thể mua đồ và các cột hồi máu giúp hồi phục sức khỏe.
- **Yếu tố đặc biệt:** Khu rừng cung cấp các địa điểm chiến lược cho người chơi để thu thập tài nguyên, vũ khí và đạn dược. Các cột hồi máu được đặt tại những điểm quan trọng, giúp người chơi hồi phục máu trong những cuộc chiến cam go.

## 2.5.4. Trang bị và vật phẩm hỗ trợ.

### Trang bị:

- **Vũ khí 1:** Súng lục là loại vũ khí cơ bản với tốc độ bắn trung bình và dễ sử dụng.
- **Vũ khí 2:** Shotgun là loại súng mạnh mẽ, hiệu quả trong cự ly gần.
- **Vũ khí 3:** Súng máy hạng nhẹ là loại súng tự động với sức mạnh tấn công liên tục.
- **Vũ khí 4:** Súng trường là loại vũ khí tiên tiến với các tính năng ưu việt và đặc biệt, được thiết kế để cung cấp sức mạnh và hiệu suất vượt trội.
- **Vũ khí 5:** Súng bắn tỉa là loại súng bắn tỉa với tầm bắn xa và độ chính xác cao, lý tưởng để tiêu diệt kẻ thù từ xa.
- **Đạn:** là loại đạn thông thường có thể gây sát thương lên quái vật.

**Item hỗ trợ:** Trụ hồi máu được đặt tại những vị trí chiến lược. Khi người chơi chạm vào các trụ này, máu sẽ được hồi đầy, giúp họ tiếp tục cuộc chiến mà không phải lo lắng về sức khỏe.

Môi trường và nhân vật trong Aliens War 3D được thiết kế tỉ mỉ để mang đến cho người chơi một trải nghiệm phong phú và đầy thử thách. Các khu vực khác nhau, từ khu rừng nguy hiểm đến các địa điểm chiến lược với trang bị và vật phẩm hỗ trợ, tất cả đều góp phần tạo nên một cuộc phiêu lưu hấp dẫn và kịch tính.

## 2.6. Tổng quan về luật chơi và cách chơi.

### 2.6.1. Luật chơi.

Trong Aliens War 3D, người chơi sẽ điều khiển nhân vật chính Alex Hunter để tiêu diệt kẻ thù ngoài hành tinh và hoàn thành các nhiệm vụ trong từng cấp độ. Dưới đây là các quy tắc cơ bản của trò chơi:

**Mục tiêu nhiệm vụ:** Mỗi màn chơi sẽ có một số mục tiêu cụ thể. Các mục tiêu này có thể bao gồm tiêu diệt một số lượng kẻ thù, giải cứu các nhân vật quan trọng, hoặc thu thập các vật phẩm cần thiết.

**Sức khỏe nhân vật:** Alex có một thanh máu hiển thị sức khỏe hiện tại. Nếu máu của Alex hết trước khi hoàn thành nhiệm vụ, trò chơi sẽ kết thúc và người chơi phải bắt đầu lại màn đó.

**Phản thưởng:** Tiêu diệt kẻ thù sẽ giúp người chơi kiếm được coin, có thể dùng để nâng cấp vũ khí và trang bị.

### 2.6.2. Hướng dẫn chơi.

Để điều khiển nhân vật và tham gia vào các trận chiến chống lại người ngoài hành tinh, người chơi cần nắm rõ các thao tác cơ bản sau:

- **Di chuyển nhân vật:** Sử dụng các phím WASD trên bàn phím để di chuyển Alex trong môi trường trò chơi.
- **Nhảy:** Sử dụng phím Space trên bàn phím để Alex nhảy lên trong môi trường trò chơi.
- **Chạy nhanh:** Sử dụng phím Shift để tập trung chạy làm cho nhân vật chạy nhanh hơn.
- **Tấn công:** Sử dụng chuột phải để nhắm kẻ thù. Nhấp chuột trái để bắn súng.
- **Chuyển đổi súng:** Sử dụng phím Tab trên bàn phím để đổi các súng mình đang có.
- **Thu thập vật phẩm:** Người chơi có thể thu thập các vật phẩm như gói trang bị đạn và trụ hồi máu các điểm thường trong môi trường trò chơi.
- **Nâng cấp nhân vật:** Sử dụng các coin kiếm được để mua vũ khí cho Alex, giúp anh mạnh mẽ hơn trong các cấp độ sau.

### 2.6.3. Thông kê.

Trò chơi cung cấp các thông kê chi tiết để người chơi có thể theo dõi tiến trình và thành tích của mình:

- **Số đạn:** Hiển thị số lượng đạn mà người chơi còn lại cho các loại vũ khí khác nhau.

- **Số coin:** Hiển thị số lượng coin mà người chơi đã kiếm được từ việc tiêu diệt kẻ thù

Bằng cách nắm vững các luật chơi và cách chơi, người chơi có thể tận hưởng trọn vẹn trải nghiệm đầy kịch tính và hấp dẫn trong Aliens War 3D. Các cơ chế trò chơi được thiết kế để thử thách khả năng chiến đấu, sự khéo léo và chiến lược của người chơi, tạo nên một cuộc phiêu lưu đầy thú vị và không kém phần nguy hiểm.

## CHƯƠNG 3: THIẾT KẾ, LẬP TRÌNH TRÒ CHƠI ALIENS WAR 3D.

### 3.1. Xây dựng trò chơi.

Ý tưởng xây dựng đồ họa trong trò chơi: Xây dựng trò chơi có đồ họa 3D.



Hình 3. 1: Đồ họa trò chơi Aliens War 3D.

#### 3.1.1. Xây dựng nhân vật chính.

Ý tưởng thiết kế nhân vật: Nhân vật chính là một người lính swat, có thể sử dụng nhiều vũ khí linh hoạt trong chiến đấu.

Thiết kế model:



Hình 3. 2: Nhân vật chính Alex Hunter.

Thiết kế animation:

- **Aim Idle:** Animation đứng ngắm tại chỗ của nhân vật.



Hình 3. 3: Animation Aim Idle.

- **Aim Walk Backward:** Animation đi lùi lại lúc cầm súng ngắm.



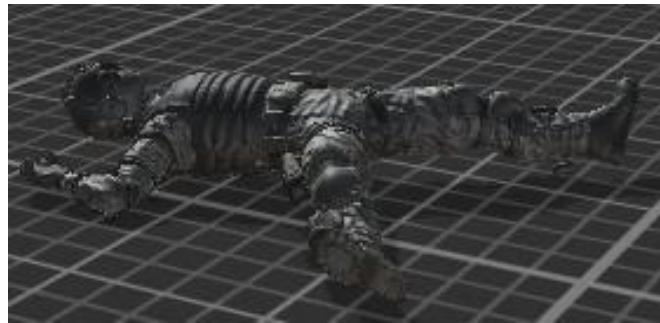
Hình 3. 4: Animation Aim Walk Backward.

- **Aim Walking:** Animation đi bộ của nhân vật.



Hình 3. 5: Animation Aim Walking.

- **Death:** Animation chết của nhân vật.



Hình 3. 6: Animation Death.

- **Jump backward:** Animation nhảy lùi về sau.



Hình 3. 7: Animation Jump backward.

- **Jump Forward:** Animation nhảy về phía trước.



Hình 3. 8: Animation Jump Forward.

- **Normal Backwards Run:** Animation chạy lùi về sau khi hạ súng.



Hình 3. 9: Animation Normal Backwards Run.

- **Normal Backwards Walk:** Animation đi lùi về sau khi hạ súng.



Hình 3. 10: Animation Normal Backwards Walk.

- **Normal Idle:** Animation đứng im khi hạ súng.



Hình 3. 11: Animation Normal Idle.

- **Normal Walk Forward:** Animation đi lên phía trước khi hạ súng.



Hình 3. 12: Animation Normal Walk Forward.

- **Run Backward Left:** Animation chạy lùi về sau hướng sang trái.



Hình 3. 13: Animation Run Backward Left.

- **Run Backward Right:** Animation chạy lùi về sau hướng sang phải.



Hình 3. 14: Animation Run Backward Right.

- **Run Backwards:** Animation chạy lùi về sau.



Hình 3. 15: Animation Run Backwards.

- **Run Forward:** Animation chạy tiến về phía trước.



Hình 3. 16: Animation Run Forward.

- **Run Forward Left:** Animation chạy tiến về phía trước hướng sang trái.



Hình 3. 17: Animation Run Forward Left.

- **Run Forward Right:** Animation chạy tiến về phía trước hướng sang phải.



Hình 3. 18: Animation Run Forward Right.

- **Run Left:** Animation chạy sang bên trái.



Hình 3. 19: Animation Run Left.

- **Run Right:** Animation chạy sang bên phải.



Hình 3. 20: Animation Run Right.

- **Shoot:** Animation bắn.



Hình 3. 21: Animation Shoot.

- **Strafe Left:** Animation đi sang trái



Hình 3. 22: Animation Strafe Left.

- **Strafe Right:** Animation đi sang phải.



Hình 3. 23: Animation Strafe Right.

- **Walk Backward Left:** Animation đi lùi về sau hướng sang trái.



Hình 3. 24: Animation Walk Backward Left.

- **Walk Backward Right:** Animation đi lùi về sau hướng sang phải



Hình 3. 25: Animation Walk Backward Right.

- **Walk Forward Left:** Animation đi tiến về phía trước hướng sang trái.



Hình 3. 26: Animation Walk Forward Left.

- **Walk Forward Right:** Animation đi tiến về phía trước hướng sang phải.



Hình 3. 27: Animation Walk Forward Right.

Lập trình code cho nhân vật:

- Code di chuyển nhân vật.

```
Vector2 input = moveAction.ReadValue<Vector2>();
Vector3 movement = new Vector3(input.x, 0, input.y);
movement = movement.x * mainCamera.right.normalized + movement.z;
movement.y = 0f;
controller.Move(movement * Time.deltaTime * currentSpeed);
```

Hình 3. 28: Code di chuyển nhân vật.

- Code xoay nhân vật.

```
void RotateCharacter()
{
    Quaternion targetRotation = Quaternion.Euler(0, mainCamera.eulerAngles.y, 0);
    transform.rotation = Quaternion.Lerp(transform.rotation, targetRotation, rotationSpeed * Time.deltaTime);
}
```

Hình 3. 29: Code xoay nhân vật.

- Code bắn súng.

```
private void ShootGun()
{
    if (isShooting && aimAction.IsPressed())
    {
        if (shootingCoroutine == null)
        {
            shootingCoroutine = StartCoroutine(ShootContinuously());
        }
    }
    else
    {
        if (shootingCoroutine != null)
        {
            StopCoroutine(shootingCoroutine);
            shootingCoroutine = null;
        }
    }
}
```

Hình 3. 30: Code bắn súng.

- Code đổi súng.

```
private void ChangeGun()
{
    do
    {
        currentGunIndex++;
        if (currentGunIndex >= guns.Length)
        {
            currentGunIndex = 0;
        }
    } while (!buyGuns.IsGunBought(currentGunIndex));
    foreach (var gun in guns)
    {
        gun.SetActive(false);
    }
    foreach (var rig in RigGuns)
    {
        rig.weight = 0f;
    }
    guns[currentGunIndex].SetActive(true);
    RigGuns[currentGunIndex].weight = 1f;
}
```

Hình 3. 31: Code đổi súng.

- Code đổi trạng thái nhân vật.

```

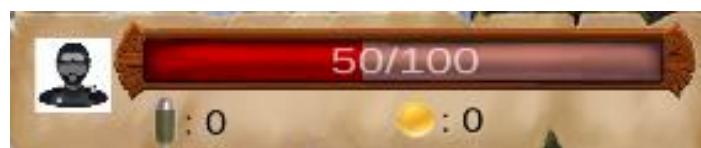
void ChangeAnimation()
{
    if ((upAction.IsPressed() || downAction.IsPressed() || leftAction.IsPressed() ||
        !(upAction.IsPressed() && downAction.IsPressed()) && !(leftAction.IsPres-
    if (upAction.WasReleasedThisFrame())
    {
        wasPressUp = true;
        isPressingUp = false;
    }
    if (downAction.WasReleasedThisFrame())
    {
        wasPressDown = true;
        isPressingDown = false;
    }
    if (leftAction.WasReleasedThisFrame())
    {
        wasPressLeft = true;
        isPressingLeft = false;
    }
    if (rightAction.WasReleasedThisFrame())
    {
        wasPressRight = true;
        isPressingRight = false;
    }

    if ((wasPressUp || wasPressDown) && !(wasPressUp && isPressingDown) && !(was-
        anim.SetFloat(velocityZ) != 0)...];
    if ((wasPressLeft || wasPressRight) && !(wasPressLeft && isPressingRight) && !
        && anim.SetFloat(velocityX) != 0)...];
    if(jumpAction.WasPressedThisFrame() || (jumpAction.WasPressedThisFrame() &&
    else if(jumpAction.WasPressedThisFrame() && downAction.WasPressedThisFrame())
    if (aimAction.IsPressed())...
    else
    {
        toAimValue -= speedAnimation * Time.deltaTime;
        if (toAimValue < 0) toAimValue = 0;
        anim.SetFloat(toAim, Mathf.Lerp(0, 1, toAimValue));
        anim.SetLayerWeight(aimStateLayer, Mathf.Lerp(0, 1, toAimValue));
        anim.SetLayerWeight(normalStateLayer, Mathf.Lerp(1, 0, toAimValue));
        multiAimConstraint.data.offset = new Vector3(8f, 20f, 10f);
    }
}

```

Hình 3. 32: Code đổi trạng thái nhân vật.

Ý tưởng xây dựng thanh trạng thái nhân vật: Có thể hiện lượng máu, số coin hiện có và số đạn theo từng loại.



Hình 3. 33. Thanh trạng thái nhân vật.

### 3.1.2. Xây dựng nhân vật phụ

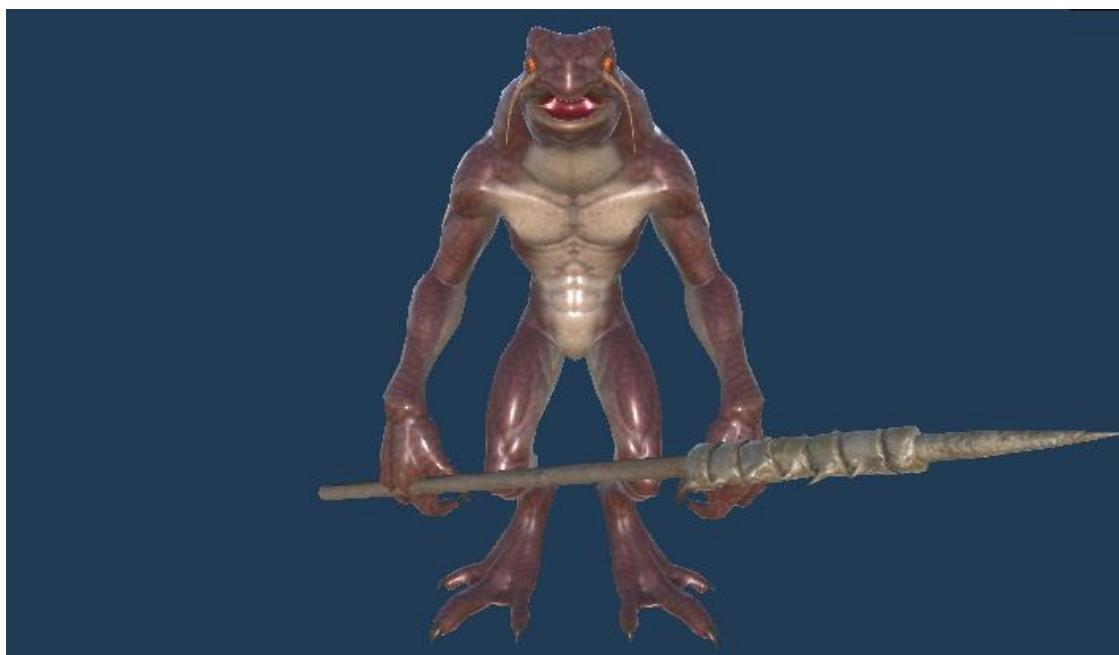
Ý tưởng thiết kế nhân vật: Quái vật Wolf, Catfish và Rake.

- **Wolf:** là một quái vật có ngoại hình giống một con sói, chạy bằng 4 chân và tấn công bằng móng vuốt.
- **Catfish:** là một quái vật có hình dáng giống cá trê, cầm giáo để tấn công.
- **Rake:** là một quái vật có ngoại hình gầy gò, sở hữu móng vuốt sắc bén và khả năng di chuyển.

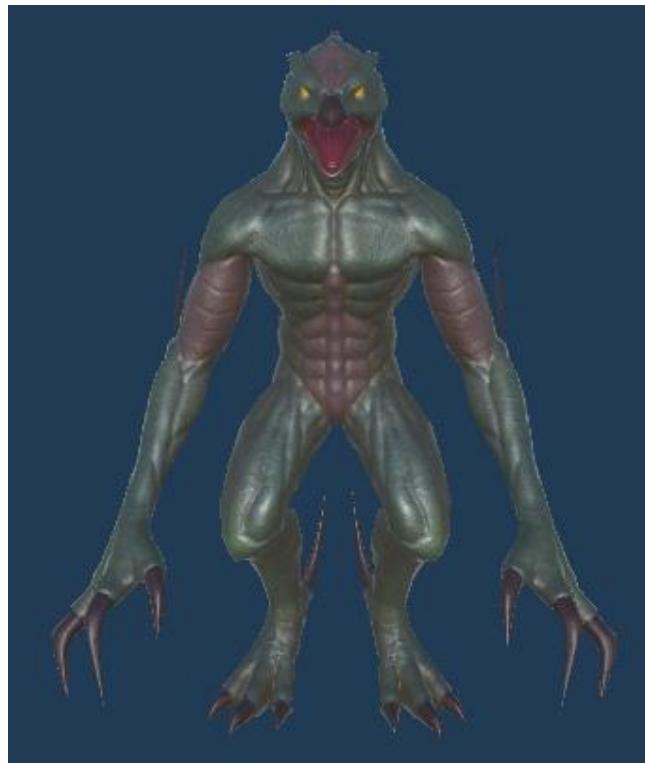
Thiết kế model:



Hình 3. 34: Quái vật Wolf.



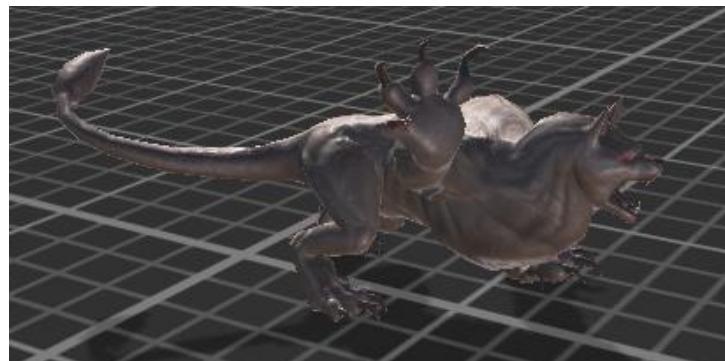
Hình 3. 35: Quái vật Catfish.



Hình 3. 36: Quái vật Rake.

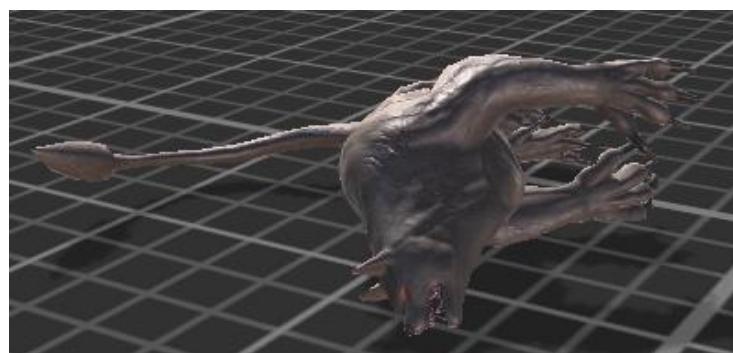
Thiết kế animation:

- Wolf:
  - + Attack: Animation tấn công.



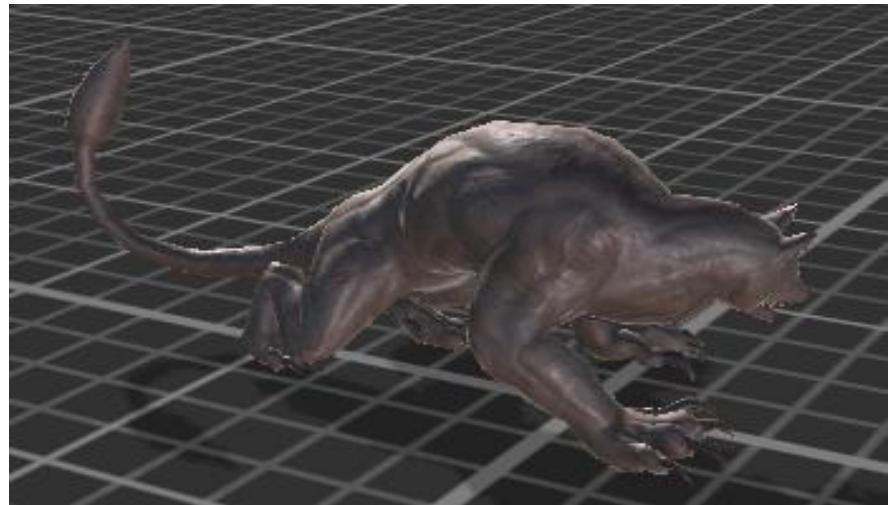
Hình 3. 37: Animation Attack.

- + Die: Animation quái vật chết.



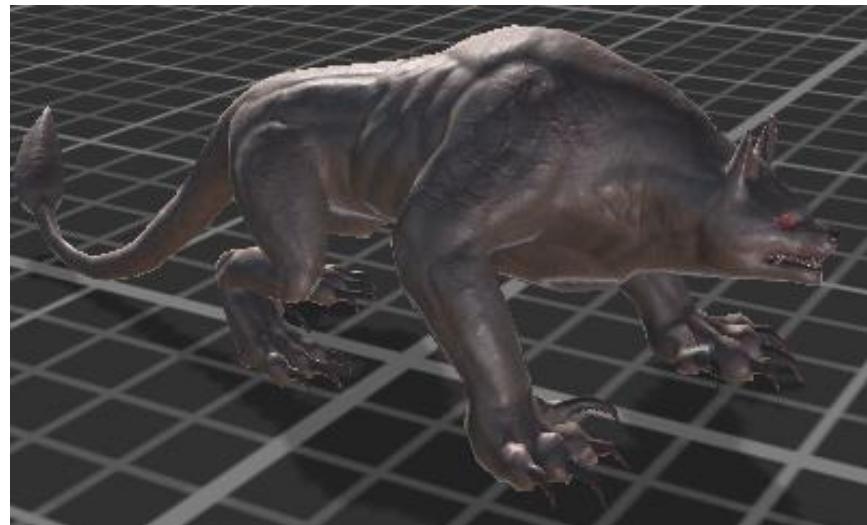
Hình 3. 38: Animation Die.

- + Hit: Animation quái vật bị tấn công.



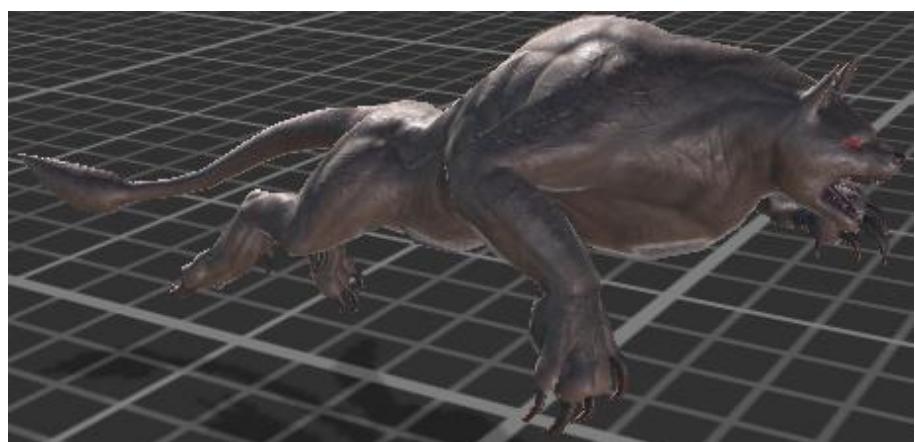
Hình 3. 39: Animation Hit.

- + Idle: Animation đứng im.



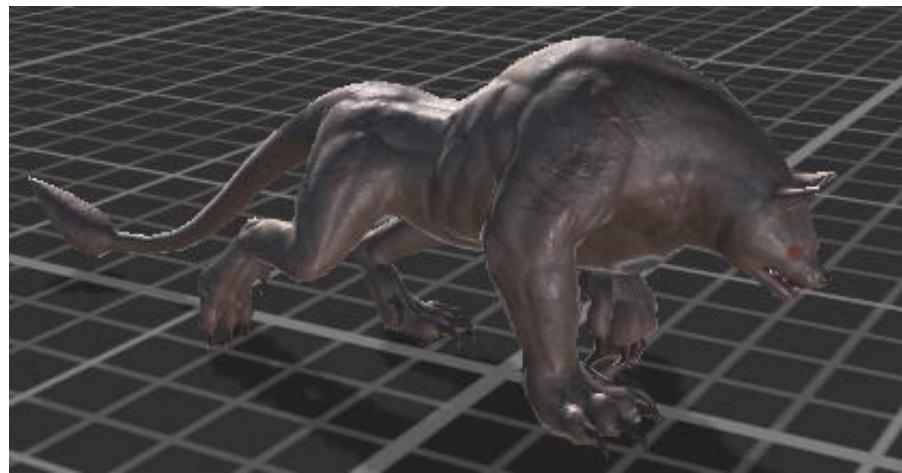
Hình 3. 40: Animation Idle.

- + Run: Animation chạy.



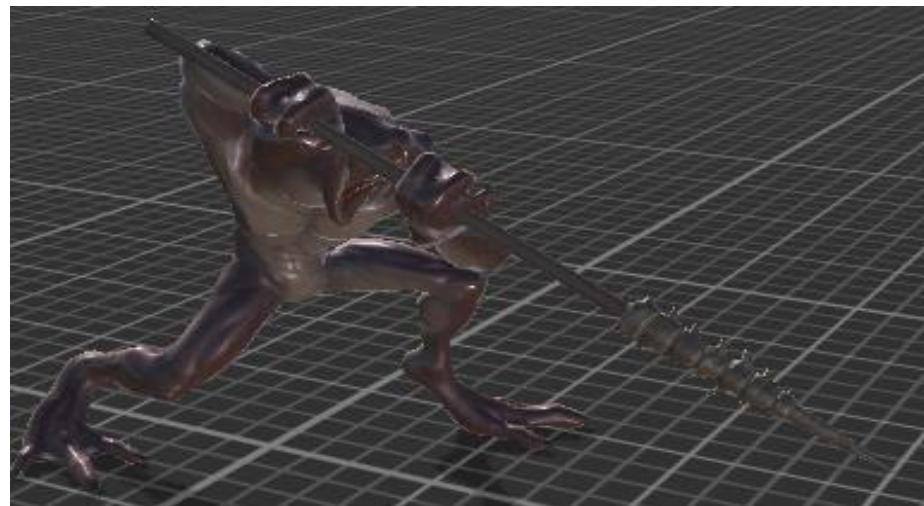
Hình 3. 41: Animation Run.

- + Walk: Animation đi bộ.



Hình 3. 42: Animation Walk.

- Catfish:
  - + Attack: Animation tấn công.



Hình 3. 43: Animation Attack.

- + Die: Animation quái vật chết.



Hình 3. 44: Animation Die.

- + Hit: Animation quái vật bị tấn công.



Hình 3. 45: Animation Hit.

- + Idle: Animation đứng im.



Hình 3. 46: Animation Idle.

- + Run: Animation chạy.



Hình 3. 47: Animation Run.

- + Walk: Animation đi bộ.



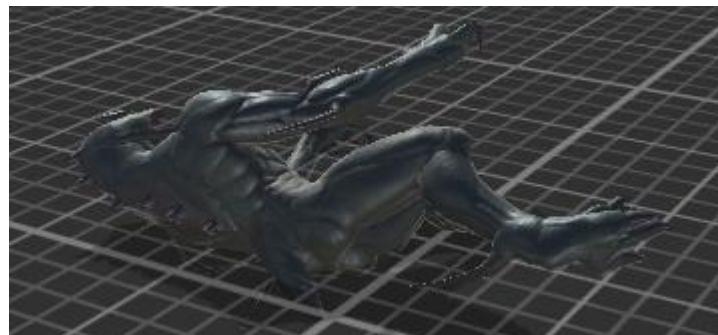
Hình 3. 48: Animation Walk.

- Rake:
  - + Attack: Animation tấn công.



Hình 3. 49: Animation Attack.

- + Die: Animation quái vật chết.



Hình 3. 50: Animation Die.

- + Hit: Animation quái vật bị tấn công.



Hình 3. 51: Animation Hit.

- + Idle: Animation đứng im.



Hình 3. 52: Animation Idle.

- + Run: Animation chạy.



Hình 3. 53: Animation Run.

- + Walk: Animation đi bộ.



Hình 3. 54: Animation Walk.

Lập trình code cho nhân vật:

- Code tìm và di chuyển đến nhân vật chính bằng NavMeshAgent

```
public virtual void MoveToPlayer()
{
    distance = Vector3.Distance(target.transform.position, transform.position);
    if (distance < 100 && health != 0 && !isWaiting)
    {
        MoveToPosition(target.transform.position);
    }
}

1 reference
public virtual void MoveToPosition(Vector3 position)
{
    if (gethit)
    {
        StartCoroutine(WaitAndMove(0.8f));
    }
    if (distance > KC)
    {
        movetoplayer = true;
        attack = false;
        agent.isStopped = false;
        agent.SetDestination(position);
    }
    else if (distance < KC)
    {
        agent.isStopped = true;
        agent.velocity = Vector3.zero;
        movetoplayer = false;
        attack = true;
        StartCoroutine(WaitAndMove(1.6f));
        Vector3 directionToTarget = target.transform.position - transform.position;
        Quaternion targetRotation = Quaternion.LookRotation(directionToTarget, Vector3.up);
        transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation, rotationSpeed);
    }
}
```

Hình 3. 55: Code tìm đường đi bằng AI NavMeshAgent.

- Code va chạm với nhân vật và đạn.

```
public virtual void OnChildTriggerEnter(Collider other, GameObject child)
{
    if (other.CompareTag("bullet"))
    {
        BulletController bullet = other.GetComponent<BulletController>();
        if (bullet != null)
        {
            PlayerAttack(bullet.damage);
            Destroy(other.gameObject);
        }
    }
    else if (other.CompareTag("Player"))
    {
        HeadController player = other.GetComponent<HeadController>();
        if (player != null && attack)
        {
            Attack(player);
        }
    }
}
```

Hình 3. 56: Code va chạm.

- Code thay đổi trạng thái quái vật

```
public virtual void AnimationState()
{
    MovementState state = MovementState.idle;
    if (movetoplayer)
    {
        state = MovementState.run;
    }
    else if (attack)
    {
        state = MovementState.attack1;
    }
    if (gethit)
    {
        state = MovementState.gethit;
    }
    animator.SetInteger("State", (int)state);
}
```

Hình 3. 57: Code thay đổi trạng thái quái vật.

- Code quái vật tấn công

```
public virtual void Attack(HeadController player)
{
    player.Headdown(damage);
}
```

Hình 3. 58: Code quái vật tấn công.

- Code quái vật bị tiêu diệt

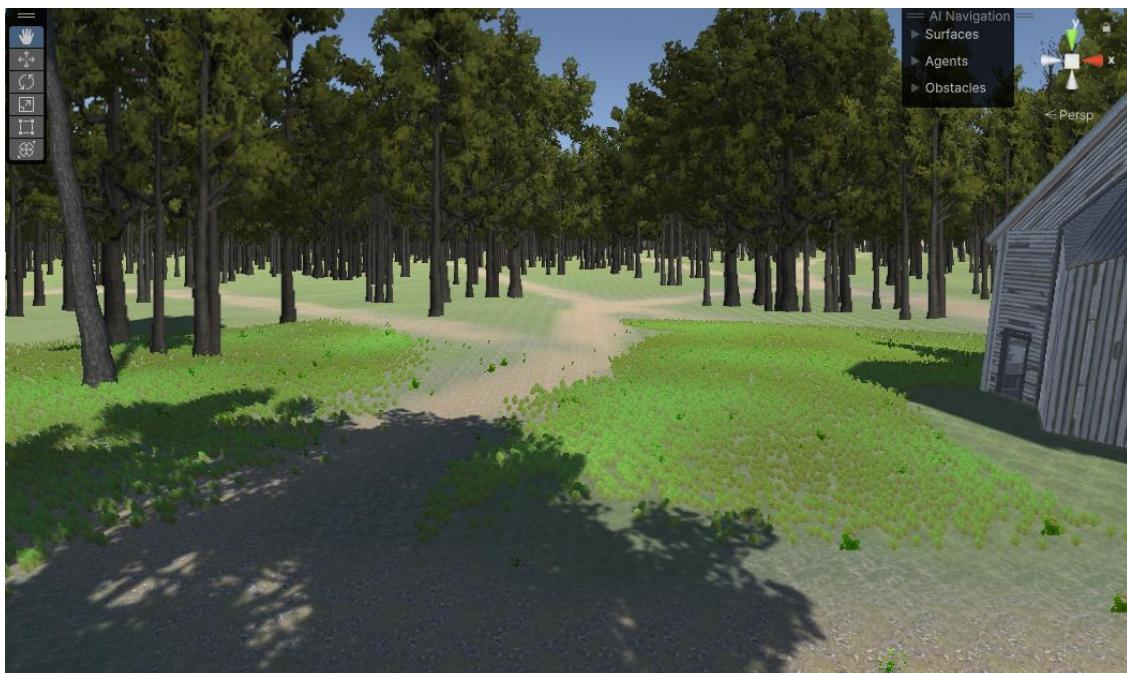
```
public virtual void MonsterDead()
{
    if (health <= 0)
    {
        animator.SetTrigger("death");
        agent.velocity = Vector3.zero;
        StartCoroutine(DestroyAfterAnimation());
        headController.Killcoin += coin;
    }
}
```

Hình 3. 59: Code quái vật bị tiêu diệt.

### 3.1.3. Xây dựng Maps trò chơi.

Ý tưởng xây dựng Maps trò chơi: Thiết kế khu rừng có nhiều cây cối và những lối mòn nhỏ nơi mà quái vật ngoài hành tinh xuất hiện đầu tiên. Khu rừng có những tòa nhà nơi mà người chơi có thể mua đồ và hồi máu.

Thiết kế Maps:



Hình 3. 60: Môi trường rừng trong trò chơi.



Hình 3. 61: Map trong trò chơi.

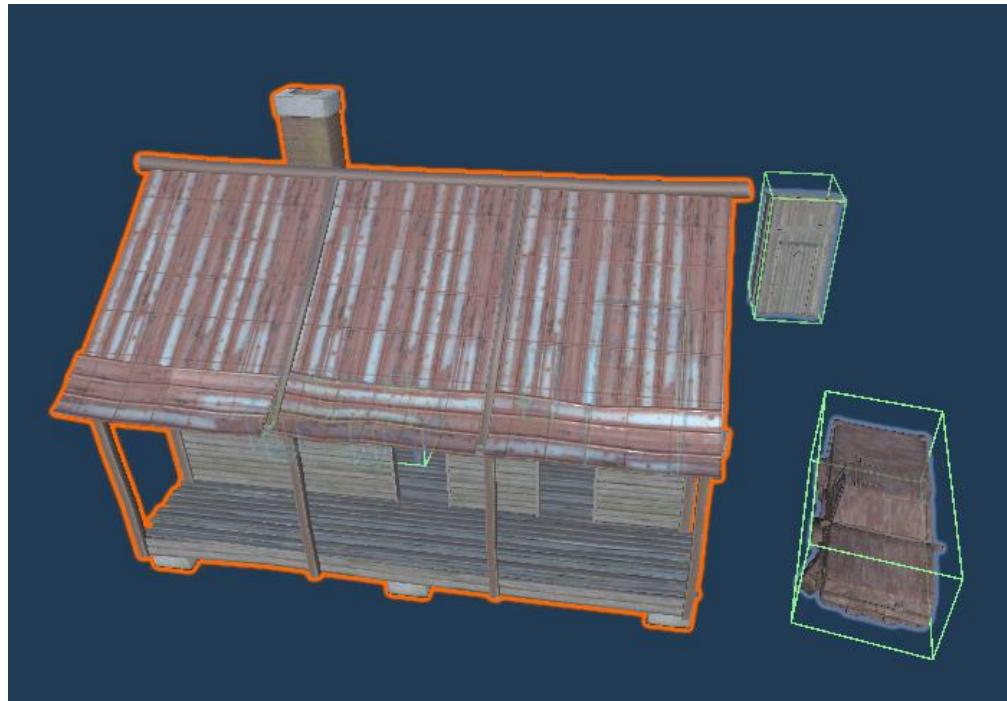
Các model trong Map:

- **Cửa hàng:** Ngôi nhà có chứa cửa hàng bên trong có thể mua súng và đạn.



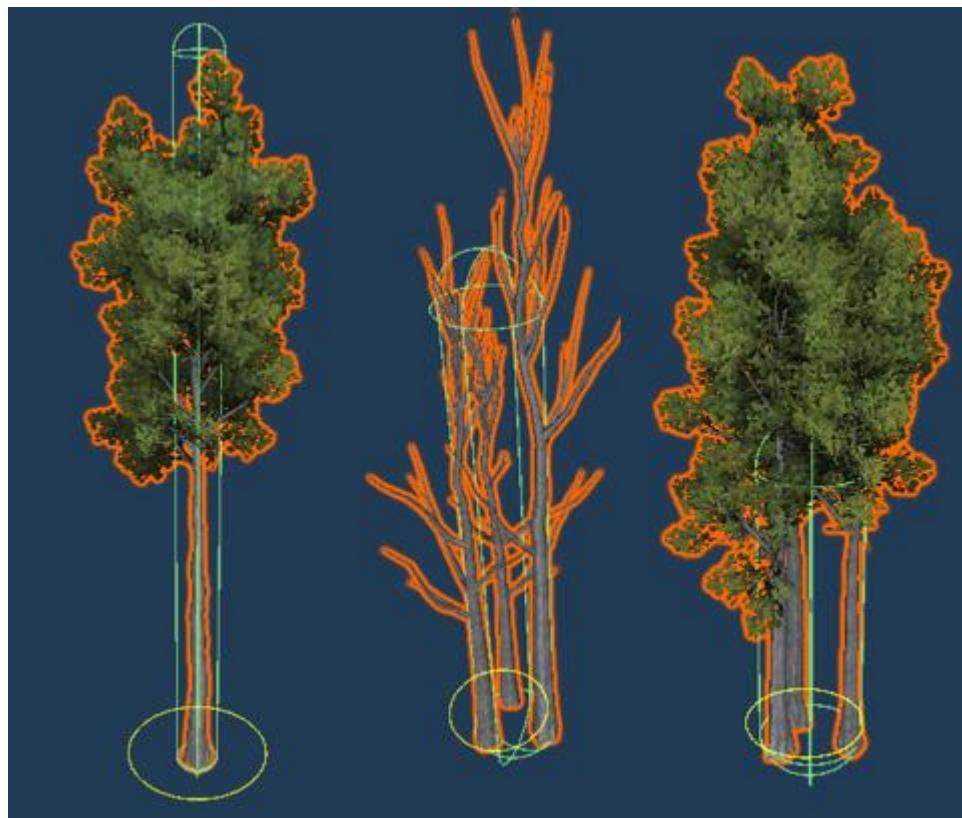
Hình 3. 62: Ngôi nhà chứa cửa hàng.

- **Ngôi nhà bình thường:** Ngôi nhà bình thường có các vật dụng thông thường bên trong.



Hình 3. 63: Ngôi nhà bình thường.

- **Cây:** Các loại cây khác nhau trong khu rừng.

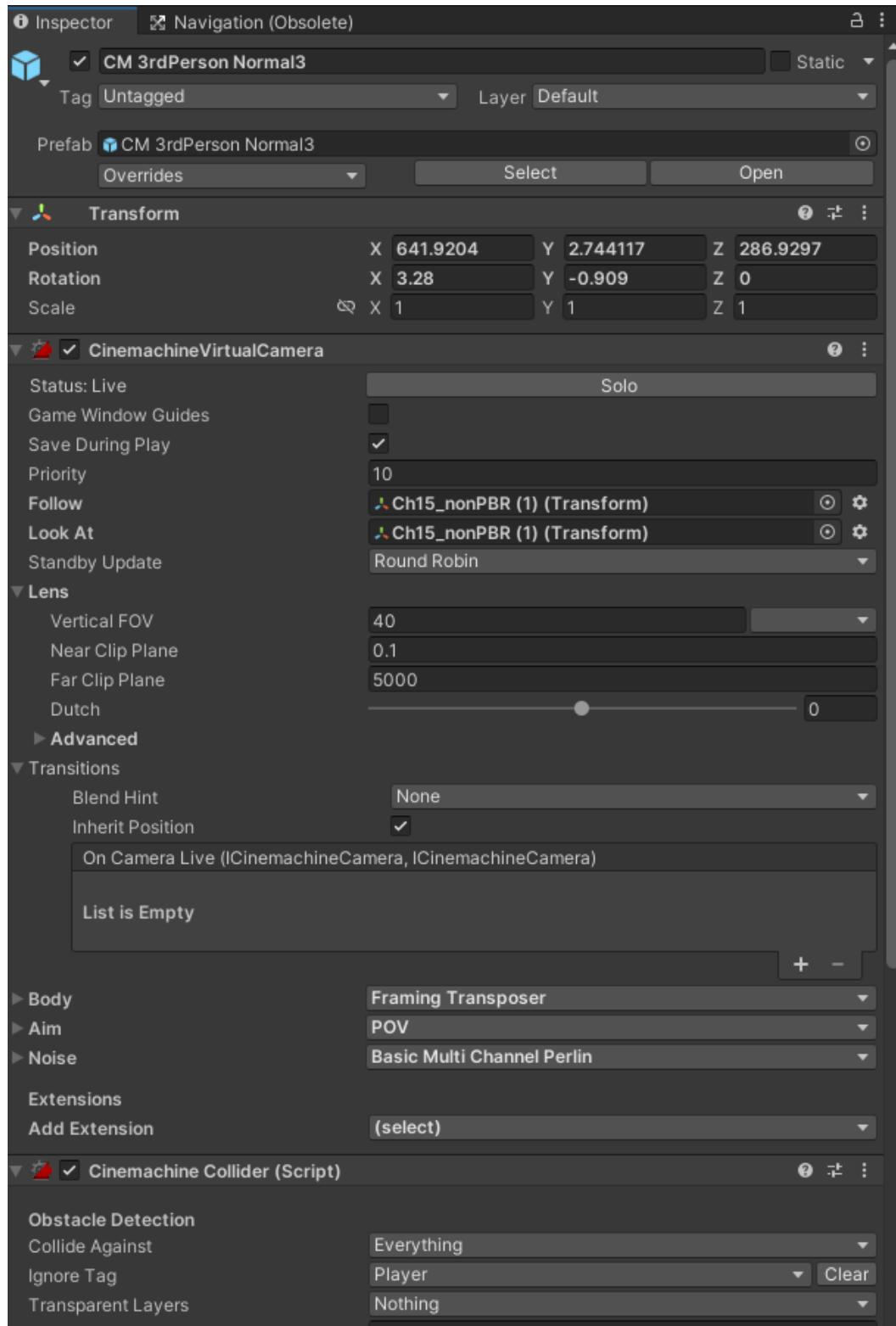


Hình 3. 64: Các loại cây trong trò chơi.

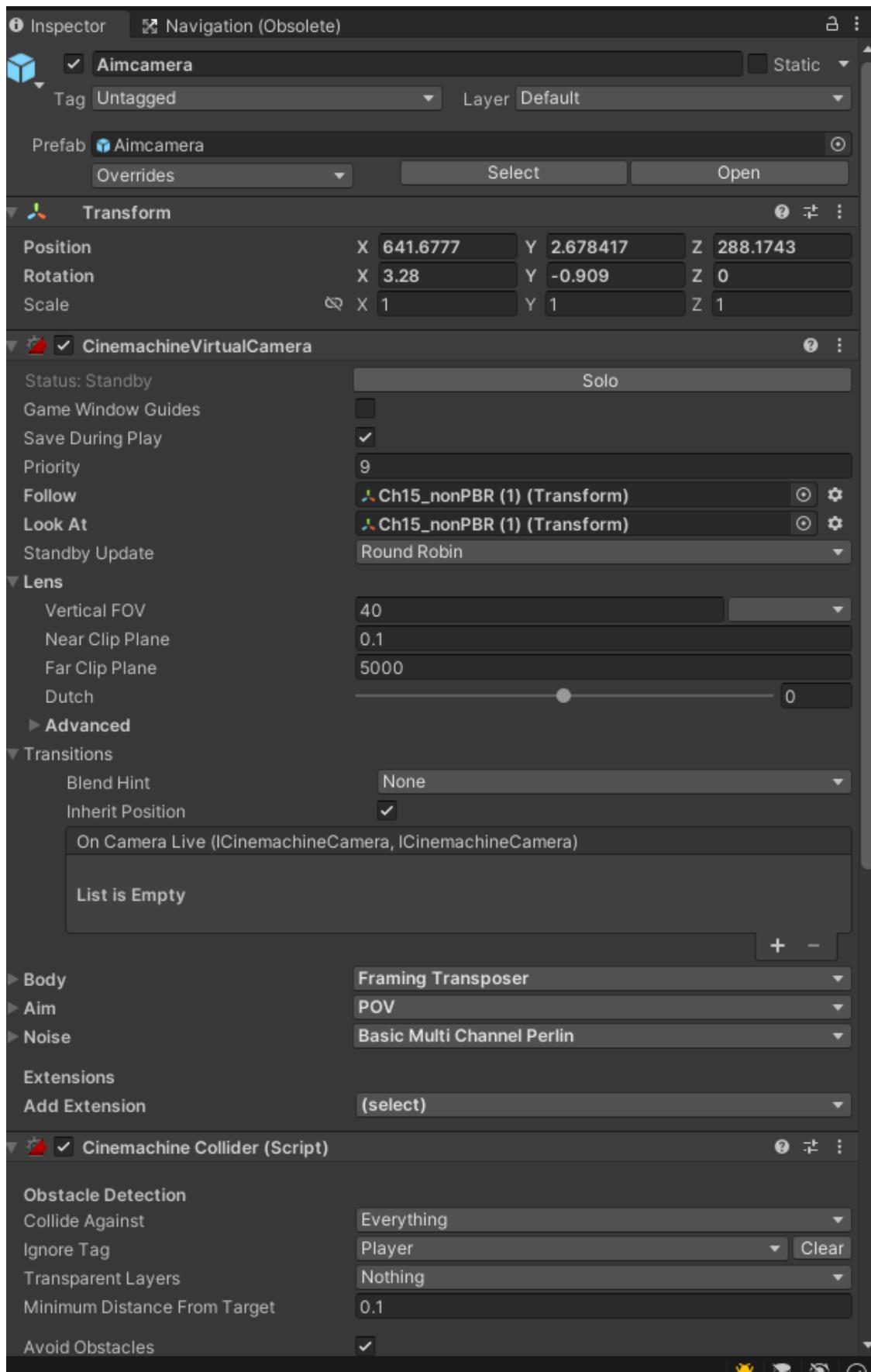
### 3.1.4. Xây dựng camera trong game.

Ý tưởng: Camera có thể xoay, đi theo nhân vật. Mỗi khi nhân vật ngắm bắn thì camera sẽ phóng to theo tâm ngắm bắn.

Xây dựng camera: Sử dụng CameraCinemachine để tạo camera trong game.  
Chỉnh thông số cho camera.



Hình 3. 65: Camera góc nhìn thứ 3.



Hình 3. 66: Camera lúc ngắm bắn.

Lập trình Code cho camera:

```
public class SwitchVCamera : MonoBehaviour
{
    [SerializeField] private PlayerInput playerInput;
    [SerializeField] private int priorityBoosAmount = 10;
    [SerializeField] private Canvas thirdPersonCanvas;
    [SerializeField] private Canvas aimCanvas;
    private CinemachineVirtualCamera virtualCamera;
    private InputAction aimAction;

    #region Unity Message | 0 references
    private void Awake()
    {
        virtualCamera = GetComponent<CinemachineVirtualCamera>();
        aimAction = playerInput.actions["Aim"];
    }
    #endregion

    #region Unity Message | 0 references
    private void OnEnable()
    {
        aimAction.performed += _ => StartAim();
        aimAction.canceled += _ => CancelAim();
    }
    #endregion

    #region Unity Message | 0 references
    private void OnDisable()
    {
        aimAction.performed -= _ => StartAim();
        aimAction.canceled -= _ => CancelAim();
    }
    #endregion

    #region Unity Message | 2 references
    private void StartAim()
    {
        virtualCamera.Priority += priorityBoosAmount;
        aimCanvas.enabled = true;
        thirdPersonCanvas.enabled = false;
    }
    #endregion

    #region Unity Message | 2 references
    private void CancelAim()
    {
        virtualCamera.Priority -= priorityBoosAmount;
        aimCanvas.enabled = false;
        thirdPersonCanvas.enabled = true;
    }
}
```

Hình 3. 67: Code camera.

### 3.1.5. Xây dựng trang bị và vật phẩm hỗ trợ.

Ý tưởng xây dựng trang bị: Gồm nhiều loại súng khác nhau và đạn:

- **Súng lục:** Loại vũ khí cơ bản với tốc độ bắn trung bình và dễ sử dụng.

- **Shotgun:** Loại súng bắn ở cự li gần.
- **Súng máy hạng nhẹ:** Loại súng tự động với sức mạnh tấn công liên tục.
- **Súng trường:** Loại vũ khí tiên tiến với các tính năng ưu việt và đặc biệt, có sức mạnh và hiệu quả vượt trội.
- **Súng bắn tỉa:** Loại súng bắn tỉa với tầm bắn xa và độ chính xác cao, lý tưởng để tiêu diệt kẻ thù từ xa.
- **Đạn:** Là một loại đạn gây sát thương lên quái vật.

Thiết kế model:



Hình 3. 68: Súng lục.



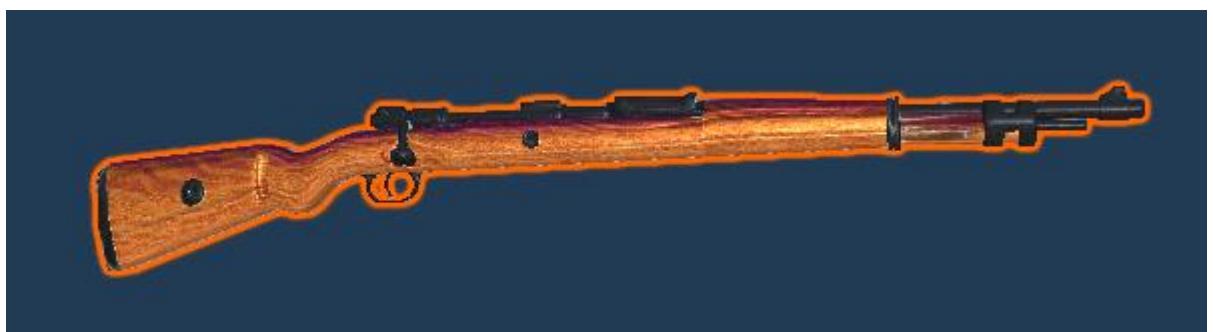
Hình 3. 69: Súng Shotgun.



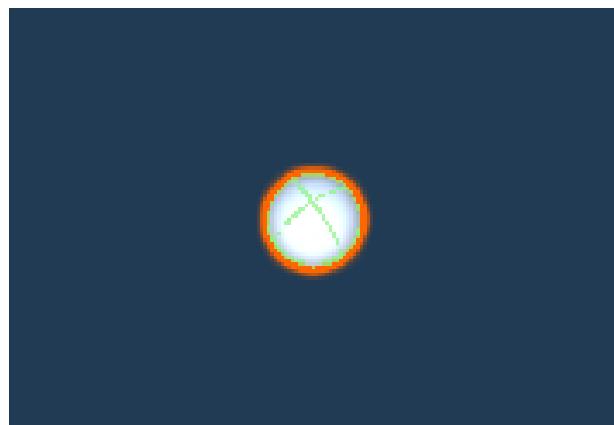
Hình 3. 70: Súng máy hạng nhẹ.



Hình 3. 71: Súng trường.



Hình 3. 72: Súng bắn tỉa.



Hình 3. 73: Viên đạn.

Lập trình Code cho súng và đạn:

- Code hàm tạo.

```
public Gun(string name, float dmg, float rate, float speed)
{
    gunName = name;
    damage = dmg;
    fireRate = rate;
    bulletSpeed = speed;
}
```

Hình 3. 74: Code hàm tạo.

- Code bắn súng chung.

```

public void Shoot(Transform barrelTransform, GameObject bulletPrefab, Transform bulletParent, float bulletHitMiss
{
    if (barrelTransform == null)
    {
        Debug.LogError("Barrel transform is null!");
        return;
    }
    if (Time.time >= nextFireTime)
    {
        GameObject bullet = GameObject.Instantiate(bulletPrefab, barrelTransform.position, Quaternion.identity, b
        BulletController bulletController = bullet.GetComponent<BulletController>();
        RaycastHit hit;
        if (Physics.Raycast(barrelTransform.position, barrelTransform.forward, out hit, Mathf.Infinity))
        {
            bulletController.target = hit.point;
            bulletController.hit = true;
        }
        else
        {
            bulletController.target = barrelTransform.position + barrelTransform.forward * bulletHitMissDistance;
            bulletController.hit = true;
        }
        if (bulletController != null)
        {
            bulletController.SetSpeed(bulletSpeed);
            bulletController.SetDamage(damage);
        }
        else
        {
            Debug.LogError("Bullet prefab does not contain a BulletController component!");
        }
        bulletnum -= 1;
        nextFireTime = Time.time + 1f / fireRate;
    }
}

```

Hình 3. 75: Code bắn súng chung.

- Code bắn súng shotgun.

```

public void ShootShotGun(Transform barrelTransform, GameObject bulletPrefab, Transform bulletParent, float bulletH
{
    if (barrelTransform == null)
    {
        Debug.LogError("Barrel transform is null!");
        return;
    }
    if (Time.time < nextFireTime)
    {
        return;
    }
    for (int i = 0; i < 10; i++)
    {
        Vector3 randomDirection = Random.insideUnitSphere;
        randomDirection.Normalize();
        randomDirection *= 0.2f;
        Vector3 newPosition = barrelTransform.position + new Vector3(randomDirection.x, randomDirection.y, random
        GameObject bullet = GameObject.Instantiate(bulletPrefab, barrelTransform.position, Quaternion.identity, b
        BulletController bulletController = bullet.GetComponent<BulletController>();
        Vector3 targetPosition = newPosition + barrelTransform.forward * 10f ;
        bulletController.target = targetPosition;
        bulletController.hit = true;
        if (bulletController != null)
        {
            bulletController.SetSpeed(bulletSpeed);
            bulletController.SetDamage(damage);
        }
        else
        {
            Debug.LogError("Bullet prefab does not contain a BulletController component!");
        }
    }
    bulletnum -= 10;
    nextFireTime = Time.time + 1f / fireRate;
}

```

Hình 3. 76: Code bắn súng Shotgun.

- Code cho viên đạn.

```
public class BulletController : MonoBehaviour
{
    private float speed = 50f;
    private float timeToDestroy = 2f;
    public float damage;
    5 references
    public Vector3 target { get; set; }
    4 references
    public bool hit { get; set; }
    0 references
    private void OnEnable()
    {
        Destroy(gameObject, timeToDestroy);
    }
    0 references
    private void Update()
    {
        transform.position = Vector3.MoveTowards(transform.position, target, speed * Time.deltaTime);
        if (!hit && Vector3.Distance(transform.position, target) < .01f)
        {
            Destroy(gameObject);
        }
    }
    0 references
    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Wolf"))
        {
            MonsterBase monster = other.GetComponentInParent<MonsterBase>();
            if (monster != null)
            {
                monster.PlayerAttack(damage);
            }
            Destroy(gameObject);
        }
    }
    2 references
    public void SetDamage(float damageNew){ damage = damageNew; }
    2 references
    public void SetSpeed(float speedNew){ speed = speedNew; }
}
```

Hình 3. 77: Code viên đạn.

Ý tưởng xây dựng item hỗ trợ: Khi người chơi chạm vào các trụ này máu sẽ được hồi đầy, giúp họ tiếp tục cuộc chiến mà không phải lo lắng sức khỏe.  
Thiết kế model:



Hình 3. 78: Trụ hồi máu.

### 3.1.6. Xây dựng cổng tạo quái vật.

Ý tưởng xây dựng cổng tạo quái vật: Là một loại cổng khi nhân vật chính tới gần thì sẽ tự động tạo ra quái vật để tấn công người chơi.

Thiết kế model: Là một Particle System.



Hình 3. 79: Cổng tạo quái vật.

Lập trình Code cho cổng:

```
public class GateController : MonoBehaviour
{
    public GameObject monsterPrefab;
    public Transform player;
    public ParticleSystem par;
    private float timer = 0f;
    private float spawnInterval = 10f;
    private int monsterCount = 0;
    void Start()
    {
        timer = spawnInterval;
    }

    void Update()
    {
        timer += Time.deltaTime;
        if (timer >= spawnInterval)
        {
            timer = 0f;
            float distance = Vector3.Distance(player.position, transform.position);
            if (distance < 50f && monsterCount <= 3)
            {
                par.Play();
                GameObject monster = Instantiate(monsterPrefab, transform.position,
                    NavMeshAgent navAgent = monster.GetComponent<NavMeshAgent>();
                monsterCount++;
            }
            else
            {
                par.Stop();
            }
        }
        if (monsterCount == 3)
        {
            Destroy(gameObject);
        }
    }
}
```

Hình 3. 80: Code Gate.

### 3.1.7. Xây dựng Menu trò chơi.

Ý tưởng xây dựng menu chính: gồm các chức năng.

- **Chơi mới:** Bắt đầu chơi từ đầu (loại bỏ tất cả dữ liệu đã lưu từ lần chơi trước đó).
- **Tiếp tục:** Tiếp tục chơi dựa theo dữ liệu đã lưu từ lần chơi trước đó.
- **Cài đặt:** Bao gồm 2 chức năng điều chỉnh âm thanh và điều chỉnh nhạc nền.
- **Điều chỉnh âm thanh:** Di chuyển theo thanh slider để tăng giảm âm thanh trò chơi bao gồm tiếng bước chân và tiếng súng.
- **Điều chỉnh nhạc nền:** Di chuyển theo thanh slider để tăng giảm nhạc nền trò chơi.
- **Thoát:** Đóng trò chơi.

Lập trình Code menu chính:

- Code menu chính.

```
public void NewGame()
{
    SceneManager.LoadScene(1);
}

0 references
public void Continue()
{
    isContinue = true;
    SceneManager.LoadScene(1);
}

0 references
public void Setting()
{
    if(!isSetting)
    {
        pSetting.SetActive(isSetting);
        isSetting = true;
    }
    else
    {
        pSetting.SetActive(isSetting);
        isSetting = false;
    }
}

0 references
public void Exit()
{
    Application.Quit();
}
```

Hình 3. 81: Code menu chính.

- Code cài đặt chính.

```

private PlayerAudio pa;

public Slider volumeSlider;
Unity Message | 0 references
private void Start()
{
    pa = FindObjectOfType<PlayerAudio>();
    if (volumeSlider != null)
    {
        volumeSlider.onValueChanged.AddListener(pa.SetSoundVolume);
        volumeSlider.value = pa.Getvolume();
    }
    else
    {
        Debug.LogError("Volume Slider is not assigned in the inspector!");
    }
}
0 references
public void AdjustVolume(float volume)
{
    if (pa != null)
    {
        pa.SetSoundVolume(volume);
    }
    else
    {
        Debug.LogError("AudioController is not assigned.");
    }
}

```

Hình 3. 82: Code điều chỉnh nhạc nền.

```

private AudioController audioController;

public Slider volumeSlider;
Unity Message | 0 references
private void Start()
{
    audioController = FindObjectOfType<AudioController>();
    if (volumeSlider != null)
    {
        volumeSlider.onValueChanged.AddListener(audioController.SetMusicVo
        volumeSlider.value = audioController.Getvolume();
    }
    else
    {
        Debug.LogError("Volume Slider is not assigned in the inspector!");
    }
}
0 references
public void AdjustVolume(float volume)
{
    if (audioController != null)
    {
        audioController.SetMusicVolume(volume);
    }
    else
    {
        Debug.LogError("AudioController is not assigned.");
    }
}

```

Hình 3. 83: Code điều chỉnh nhạc nền và âm thanh.

Ý tưởng menu trong game: gồm các chức năng.

- **Chơi tiếp:** Quay lại trò chơi và tiếp tục chơi.
- **Cài đặt:** Bao gồm 3 chức năng điều chỉnh âm thanh, điều chỉnh nhạc nền và quay lại.
- **Điều chỉnh âm thanh:** Di chuyển slider để tăng giảm âm thanh trò chơi.
- **Điều chỉnh nhạc nền:** Di chuyển slider để tăng giảm nhạc nền trò chơi.
- **Quay lại:** Quay lại menu tạm dừng.
- **Thoát:** Thoát chế độ chơi và quay lại menu chính.

Lập trình Code menu trong game:

```
public void PlayinGame()
{
    TogglePauseGame();
}

0 references
public void SettinginGame()
{
    if (isSettingInGame)
    {
        pMenu.SetActive(!isSettingInGame);
        pSettingInGame.SetActive(isSettingInGame);
        isSettingInGame = false;
    }
    else
    {
        pSettingInGame.SetActive(isSettingInGame);
        pMenu.SetActive(!isSettingInGame);
        isSettingInGame = true;
    }
}

0 references
public void ExitinGame()
{
    Vector3 playerPosition = pm.GetTranform();
    float score = headController.Killcoin;
    float heath = headController.Head;
    Dictionary<int, bool> buyGuns = gunsMenu.BuyGuns;
    Dictionary<int, Gun> gundictionary = pm.gunDictionary;
    List<string> destroyedObject = GetDestroyedObjects();
    Dictionary<string, int> gateOnMonster = GetGates();
    /*Dictionary<string, SerializableVector3> Monsterp = GetMonster();*/
    saveGamePlayer = new SaveGamePlayer(new SerializableVector3(playerPo
    file = File.Create(Application.persistentDataPath + "/save.dat");
    formatter.Serialize(file, saveGamePlayer);
    file.Close();
    Time.timeScale = 1;
    SceneManager.LoadScene(0);
}
```

Hình 3. 84: Code menu trong game.

### 3.1.8. Xây dựng cửa hàng trong game.

Ý tưởng xây dựng cửa hàng trong game: Tạo ra một cửa hàng có thể hiển thị thông tin nhân vật, mua súng và mua đạn.

- **Hiển thị thông tin nhân vật:** Hiển thị máu nhân vật.
- **Mua súng:** Mua các loại súng khác nhau dự trên công dụng từng loại súng.
- **Mua đạn:** Mua số đạn tương ứng với 1 băng đạn của súng.

Lập trình Code cửa hàng trong game:

```
public void BuyGun()
{
    if (!BuyGuns[currentGun] && headController.Killcoin >= PriceGu
    {
        headController.Killcoin -= PriceGuns[currentGun];
        BuyGuns[currentGun] = true;
    }
}

1 reference
public bool IsGunBought(int gunIndex)
{
    return BuyGuns.ContainsKey(gunIndex) && BuyGuns[gunIndex];
}

0 references
public void BuyBullet()
{
    if(headController.Killcoin >= PriceBullet[currentGun])
    {
        headController.Killcoin -= PriceBullet[currentGun];
        player.SetBulletGun(currentGun, BulletNum[currentGun]);
    }
}
```

Hình 3. 85: Code mua súng và mua đạn.

## 3.2. Điều khiển I/O khi chơi trò chơi.

### 3.2.1. Điều khiển trên Bàn phím.

- **Di chuyển nhân vật:** Sử dụng các phím WSAD để di chuyển nhân vật lên, xuống, trái và phải trên màn hình.
- **Nhảy:** Sử dụng phím Space để nhảy.
- **Đổi súng:** Sử dụng phím Tab để thay đổi súng đang có.
- **Chạy nhanh:** Sử dụng phím Shift để nhân vật tập trung chạy khiến nhân vật chạy nhanh hơn.
- **Sử dụng item:** Di chuyển nhân vật vào item để sử dụng.
- **Mở menu tạm dừng:** Sử dụng phím Esc để mở menu tạm dừng.

### 3.2.2. Điều khiển trên Chuột.



Hình 3. 86: Nhân vật nhắm bắn.

- **Nhắm bắn:** Nhấn giữ chuột phải để nhắm bắn.
- **Bắn súng:** Nhấn và nhả giữ chuột trái để bắn súng.
- **Xoay nhân vật:** Di chuyển chuột sang trái phải hoặc lên xuống để nhân vật xoay theo hướng di chuột.

### 3.3. Giao diện GUI.

#### 3.3.1. Giao diện menu chính.

Giao diện menu chính bao gồm các chức năng như: Chơi mới, Tiếp tục, Cài đặt, Thoát.



Hình 3. 87: Giao diện menu chính.



Hình 3. 88: Giao diện chức năng cài đặt trong menu chính.

### 3.3.2. Giao diện menu tạm dừng.

Giao diện menu tạm dừng bao gồm các chức năng: Chơi tiếp, cài đặt, thoát.



Hình 3. 89: Giao diện meunu tạm dừng.



Hình 3. 90: Giao diện chức năng cài đặt trong menu tạm dừng.

### 3.3.3. Giao diện cửa hàng.

Giao diện cửa hàng bao gồm các chức năng: Hiển thị thông tin nhân vật, mua súng, mua đạn.



Hình 3. 91: Giao diện cửa hàng.

### 3.3.4. Giao diện GameOver.

Giao diện GameOver bao gồm 2 chức năng: Chơi lại và thoát.



Hình 3. 92: Giao diện GameOver.

- **Chơi lại:** Bắt đầu chơi lại từ đầu.
- **Thoát:** Thoát chế độ chơi và quay lại menu chính.

## 3.4. Các âm thanh trong trò chơi.

- Nhạc nền
- Tiếng nhán chuột
- Tiếng bắn đạn
- Tiếng bước chân

## 3.5. Kiểm thử và đánh giá.

### 3.5.1. Kiểm Thử.

**Kiểm Tra Tích Hợp:** Dự án đã trải qua quá trình kiểm tra tích hợp kỹ lưỡng để đảm bảo rằng mọi thành phần của game được tích hợp một cách mượt mà và không gây ra lỗi.

**Kiểm Tra Tính Năng:** Mỗi tính năng của game đã được kiểm tra kỹ lưỡng dưới nhiều trường hợp sử dụng khác nhau để đảm bảo hoạt động đúng đắn và mượt mà.

**Kiểm Tra Tương Thích:** Game đã được kiểm tra trên nhiều nền tảng và thiết bị khác nhau để đảm bảo rằng hoạt động một cách mượt mà và không gặp phải vấn đề tương thích.

**Kiểm Tra Hiệu Năng:** Em đã tiến hành kiểm tra hiệu suất của game và tối ưu hóa nếu cần thiết để đảm bảo trải nghiệm chơi game mượt mà và không gặp phải giật lag.

### 3.5.2. Đánh Giá.

**Trải Nghiệm Người Chơi:** Người chơi đã cung cấp phản hồi tích cực về trải nghiệm chơi game, đặc biệt là về điều khiển, gameplay và đồ họa.

**Độ Mượt Mà:** Dự án đã được đánh giá là mượt mà và không gặp phải giật lag trên các nền tảng và thiết bị khác nhau.

**Đồ Họa và Âm Thanh:** Chất lượng đồ họa và hiệu ứng âm thanh đã nhận được sự đánh giá cao từ phía người chơi và nhận được nhiều lời khen ngợi.

### 3.5.3. Phản Hồi và Cải Tiến.

Sau khi thu thập và xem xét phản hồi từ người chơi, em nhận thấy rằng việc cải thiện và mở rộng các màn chơi là một phần quan trọng để tăng tính hấp dẫn của trò chơi. Hiện tại, dự án chỉ có một màn chơi, điều này có thể dẫn đến sự đơn điệu và giảm khả năng chơi lại của game.

Để cải thiện trải nghiệm chơi game, có thể xem xét các biện pháp sau:

- **Thêm Các Màn Chơi Mới:** Em sẽ phát triển và thêm vào các màn chơi mới với cấu trúc và thách thức khác nhau để tăng tính đa dạng và sự hấp dẫn của game.
- **Đa Dạng Hóa Màn Chơi Hiện Tại:** Em sẽ nâng cấp và thay đổi các yếu tố trong màn chơi hiện tại để tạo ra những trải nghiệm mới mẻ và thú vị cho người chơi.
- **Tối Ưu Hóa Cơ Chế Gameplay:** Em sẽ kiểm tra và cải thiện cơ chế gameplay để đảm bảo tính chân thực và độ mượt mà trong từng màn chơi.
- **Tăng Khả Năng Tương Tác:** Em sẽ bổ sung các yếu tố tương tác mới như nhiệm vụ phụ, bí ẩn, hoặc chế độ chơi đa người để tăng sự hấp dẫn và tương tác trong game.
- **Thu Thập Phản Hồi Liên Tục:** Em sẽ tiếp tục thu thập phản hồi từ người chơi về các màn chơi mới và cải tiến để đảm bảo rằng dự án luôn đáp ứng được mong đợi và yêu cầu của cộng đồng người chơi.

Qua việc thực hiện các biện pháp trên, em hy vọng sẽ cải thiện và mở rộng trải nghiệm chơi game, đồng thời giữ cho dự án luôn hấp dẫn và mượt mà theo thời gian.

## KẾT LUẬN VÀ KIẾN NGHỊ

Trong quá trình thực hiện dự án Alien War 3D, em đã trải qua một hành trình học hỏi đầy ý nghĩa và đáng nhớ. Từ những ngày đầu tiên của việc lập kế hoạch cho dự án đến khi hoàn thiện sản phẩm cuối cùng, em đã học được rất nhiều từ trải nghiệm làm việc thực tế và sự hướng dẫn từ các giảng viên. Dự án này không chỉ là một bước đệm quan trọng trong sự nghiệp của em mà còn là một cơ hội tuyệt vời để áp dụng những kiến thức lý thuyết đã học vào thực tế.

Trong kết luận, em muốn nhấn mạnh những điểm chính sau:

**Thành Công của Dự Án:** Dự án Alien War 3D đã hoàn thành với một sản phẩm game hoàn chỉnh, kịch tính và thú vị. Sự hợp tác giữa các thành viên trong nhóm đã tạo nên một trải nghiệm chơi game đáng nhớ, từ các pha hành động đến đồ họa và âm thanh chân thực.

**Những Kinh Nghiệm Học Được:** Qua dự án này, em đã học được rất nhiều về quy trình phát triển game, quản lý thời gian và giải quyết vấn đề. Từ việc phân tích yêu cầu đến thiết kế, lập trình và kiểm thử, mỗi bước trong quy trình phát triển game đều đóng góp vào sự phát triển kỹ năng của em.

**Kiến Nghị Cho Tương Lai:** Dựa trên kinh nghiệm từ dự án này, em đề xuất một số điểm cải thiện và kiến nghị cho các dự án tương lai. Điều này bao gồm việc cải thiện quy trình làm việc, tăng cường kiến thức kỹ thuật và cải thiện quản lý dự án. Bằng cách áp dụng những bài học từ dự án Alien War 3D, em tin rằng các dự án tương lai sẽ được thực hiện một cách hiệu quả và chất lượng hơn.

**Tổng Kết:** Em rất tự hào về những gì em đã đạt được với dự án Alien War 3D và hy vọng rằng sản phẩm này sẽ mang lại trải nghiệm tuyệt vời cho người chơi. Em xin chân thành cảm ơn sự hỗ trợ và hướng dẫn từ các giảng viên và đồng đội trong dự án này. Đây là một trải nghiệm đáng trân trọng và em sẽ tiếp tục áp dụng những kinh nghiệm học được vào những thách thức sắp tới.

Xin chân thành cảm ơn!

## **DANH MỤC TÀI LIỆU THAM KHẢO**

1. Technologies, U. (no date) Unity user manual 2022.3 (LTS), Unity. Available at: <https://docs.unity3d.com/Manual/index.html> (Accessed: 20 March 2024).
2. The best assets for game making (no date) Unity Asset Store. Available at: <https://assetstore.unity.com/?q=grass&orderBy=1> (Accessed: 10 April 2024).
3. (No date) Mixamo. Available at: <https://www.mixamo.com/#/?page=1&type=Character> (Accessed: 20 April 2024).
4. (No date a) Mixamo. Available at: <https://www.mixamo.com/#/?page=1&query=shooter+park&type=Motion%2CMotionPack> (Accessed: 29 April 2024).