

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
ĐỀ TÀI: XÂY DỰNG GAME THỂ LOẠI
SANDBOX BẰNG CÔNG CỤ UNITY

Giảng viên hướng dẫn: ThS. Phạm Thị Vương
Sinh viên thực hiện: Phạm Hoàng Yên
Lớp: Công Nghệ Thông Tin
Khóa: 61
MSSV: 6151071031

Hồ Chí Minh, ngày 17 tháng 06 năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH
BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
ĐỀ TÀI: XÂY DỰNG GAME THỂ LOẠI
SANDBOX BẰNG CÔNG CỤ UNITY

Giảng viên hướng dẫn:	ThS. Phạm Thị Vương
Sinh viên thực hiện:	Phạm Hoàng Yến
Lớp:	Công Nghệ Thông Tin
Khóa:	61
MSSV:	6151071031

Hồ Chí Minh, ngày 17 tháng 06 năm 2024

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

TP. HCM, ngày 17 tháng 06 năm 2024
Giáo viên hướng dẫn
(Ký và ghi rõ họ tên)

LỜI MỞ ĐẦU

Ngành công nghiệp game toàn cầu hiện đang chứng kiến sự tăng trưởng mạnh mẽ, số lượng người chơi cũng không ngừng gia tăng. Sự ra đời của các thiết bị di động thông minh, mạng internet tốc độ cao và các nền tảng chơi game trực tuyến đã tạo điều kiện thuận lợi cho việc tiếp cận game của mọi người. Nhu cầu giải trí ngày càng tăng, đặc biệt là trong bối cảnh thời đại mới. Game trở thành một hình thức giải trí phổ biến được nhiều người lựa chọn. Ngành công nghiệp game ngày càng đa dạng với sự xuất hiện của nhiều thể loại game mới, đáp ứng mọi sở thích và nhu cầu của người chơi.

Trong thế giới game đầy màu sắc với vô số thể loại khác nhau, game Sandbox luôn nổi bật bởi sự tự do và sáng tạo vô hạn mà nó mang lại cho người chơi. Khác với những tựa game gò bó bởi cốt truyện và nhiệm vụ định sẵn, sandbox mở ra một không gian rộng lớn, nơi người chơi có thể thỏa sức khám phá, xây dựng và trải nghiệm theo cách riêng của mình.

Công cụ hỗ trợ phát triển game cũng là một phần không thể thiếu của các tựa Game từ nhỏ tới lớn, việc lựa chọn một công cụ hỗ trợ mạnh mẽ, linh hoạt và phù hợp giúp nhà phát triển xây dựng và hoàn thiện trò chơi một cách nhanh chóng và hiệu quả. Nó cung cấp các khả năng tái sử dụng tài nguyên, hỗ trợ đa nền tảng đồ họa đẹp mắt. Unity Engine là công cụ phát triển game 2D/3D rất thịnh hành hiện nay, đã góp phần tạo nên rất nhiều tựa game đình đám trên toàn thế giới.

Mục tiêu của đồ án là tập trung nghiên cứu và ứng dụng Unity Engine vào phát triển trò chơi thể loại Sandbox, làm nổi bật tính sáng tạo của thể loại này. Qua quá trình thực hiện đồ án này, em đã được trang bị những kiến thức cơ bản đến nâng cao và học hỏi được nhiều kỹ năng, từ đó từng bước đóng góp vào sự nghiệp phát triển ngành công nghiệp Game ở nước ta.

LỜI CẢM ƠN

Quãng thời gian bốn năm tuy ngắn ngủi nhưng đã tràn đầy những kỷ niệm đẹp đẽ và ý nghĩa dưới sự dẫn dắt của thầy cô bộ môn Công Nghệ Thông Tin thuộc trường Đại Học Giao Thông Vận Tải Phân Hiệu tại Thành phố Hồ Chí Minh. Nhờ sự tận tâm, nhiệt huyết với nghề và tình thương đối với học sinh của quý thầy cô, không chỉ bỏ qua những lần sai sót của em trong quá trình học tập mà còn rất tận tình hướng dẫn và chỉ ra lỗi sai. Em xin chân thành cảm ơn các thầy cô đã dùn dắt, truyền đạt những kiến thức chuyên môn cùng kinh nghiệm quý báu của mình để em có đủ kiến thức, kỹ năng thực hiện đồ án này.

Em xin gửi lời cảm ơn và lòng biết ơn sâu sắc đến thầy Th.S Phạm Thị Vương, người đã hướng dẫn và giải đáp các thắc mắc của em trong suốt quãng thời gian thực hiện đồ án. Những lời góp ý và lời khuyên của thầy đã giúp em có cái nhìn khác về những vấn đề mình đang gặp và hoàn thiện hơn cho trò chơi, từ đó giúp em rút ra được nhiều kinh nghiệm quý báu.

Trong quá trình làm đồ án không thể tránh khỏi những thiếu sót, em rất mong được nhận những lời góp ý, chỉ ra lỗi sai để em có cái nhìn trực quan và hoàn thiện hơn về mặt kiến thức và bản thân cũng rút ra được nhiều kinh nghiệm hơn cho tương lai,

Em xin chân thành cảm ơn quý thầy cô!

MỤC LỤC

LỜI MỞ ĐẦU	i
LỜI CẢM ƠN	ii
DANH MỤC HÌNH ẢNH	vi
DANH MỤC CÁC CỤM TỪ VIẾT TẮT	ix
CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI	1
1.1 Giới thiệu về thể loại Sandbox	1
1.1.1 Định nghĩa và các đặc điểm cơ bản của thể loại Sandbox	1
1.1.2 Phân tích các yếu tố trong việc thiết kế một trò chơi Sandbox.	1
1.1.3 Tính giáo dục của thể loại Sandbox.....	2
1.1.4 Các tựa Game Sandbox thành công	4
1.2 Giới thiệu đề tài	6
1.2.1 Mục tiêu đề tài	6
1.2.2 Phương pháp nghiên cứu	7
1.3 Nội dung Game	7
1.3.1 Phạm vi nghiên cứu	7
1.3.2 Mô tả Game.....	8
CHƯƠNG II: CƠ SỞ LÝ THUYẾT CÁC CÔNG CỤ VÀ PHẦN MỀM SỬ DỤNG	9
2.1 Các công cụ thường được sử dụng để phát triển game Sandbox	9
2.1.1 GameMaker Studio 2	9
2.1.2 Unreal Engine	10
2.1.3 Godot.....	10
2.1.4 Unity	12
2.2 Công cụ Unity trong phát triển game	13
2.2.1 Giới thiệu chung	13
2.2.2 Quá trình phát triển của Unity	14
2.2.3 Ưu điểm và nhược điểm của Unity.....	14
2.2.4 Các thành phần trong Unity Editor	16
2.2.5 Các đặc điểm và tính năng trong Unity Editor	23
2.3. Các công cụ sử dụng	26
2.3.1 Visual Studio Code	26
2.3.2 Unity Editor	27

2.3.3 Asset Store	27
2.3.4 Adobe Photoshop	28
2.4. Giới thiệu về ngôn ngữ lập trình C#	29
2.4.1 Định nghĩa và tính năng chính của ngôn ngữ lập trình C#.....	29
2.4.2 Cách sử dụng C# trong trò chơi.....	30
2.4.3 Ưu nhược điểm của C#	32
2.5 Kỹ thuật lập trình hướng đối tượng trong C#	33
2.5.1 Tổng quan về lập trình hướng đối tượng	33
2.5.2 Một số kỹ thuật lập trình hướng đối tượng	34
CHƯƠNG III: PHÂN TÍCH THIẾT KẾ GAME.....	35
3.1 Load hoạt họa	35
3.1.1 Vấn đề	35
3.1.2 Giải pháp	35
3.2 Chuyển động mô hình nhân vật 2D.....	36
3.2.1 Vấn đề	36
3.2.2 Giải pháp	36
3.3 Xây dựng giao diện Game	37
3.3.1 Vấn đề	37
3.3.2 Giải pháp	37
3.4 Âm thanh trong Game.....	41
3.4.1 Vấn đề	41
3.4.2 Giải pháp	41
3.5 Lỗi và Debugging	43
3.5.1 Vấn đề	43
3.5.2 Giải pháp	44
CHƯƠNG IV: TRIỂN KHAI VÀ THỰC HIỆN	45
4.1 Triển khai thế giới game:	45
4.1.1 Giới thiệu hàm PerlinNoise trong khởi tạo địa hình.....	45
4.1.2 Khởi tạo thế giới Game.....	45
4.1.3 Khởi tạo hệ sinh thái	46
4.1.4 Thuật ngữ “Chunk”	50
4.1.5 Thuật ngữ “Bedrock”	52
4.1.6 Tạo Shader bằng Universal Render Pipeline (URP)	52

4.2 Triển khai hệ thống vật phẩm	55
4.2.1 Hệ thống vật phẩm.....	55
4.2.2 Vật phẩm rơi ra	58
4.2.3 Cơ chế đặt và đào khói trong Game	60
4.3 Triển khai công cụ trong game.....	61
4.4 Triển khai nhân vật	62
4.4.1 Xác định và mô tả nhân vật	s62
4.4.2 Các hoạt ảnh Animation	63
4.4.3 Quy định nơi sử dụng hoạt ảnh.....	65
4.4.4 Cơ chế tự động nhảy (Autojump) khi gặp một vật cản	66
4.5 Triển khai hệ thống kho đồ và thanh hotbar	67
4.5.1 Hệ thống kho đồ.....	67
4.5.2 Hệ thống thanh hotbar.....	d71
4.6 Kiểm Tra và Sửa Lỗi:	72
ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN	73
1. Kết Quả Đạt Được.....	73
2. Đề xuất hướng phát triển trong tương lai	73
TÀI LIỆU THAM KHẢO	75

DANH MỤC HÌNH ẢNH

Hình 1.1 Game Minecraft	5
Hình 1.2. Game Terraria	6
Hình 2.3. Unity.....	14
Hình 2.4. Các thành phần cơ bản: Assets.....	16
Hình 2.5. Các thành phần cơ bản: Transform	17
Hình 2.6. Ví dụ Prefab trong Flappy Bird	19
Hình 2.7. Các thành phần cơ bản: Scene View.....	19
Hình 2.8. Các thành phần cơ bản: Gane View.....	20
Hình 2.9. Tỉ lệ khung hình	20
Hình 2.10. Các thành phần cơ bản: Hierachy	21
Hình 2.11. Các thành phần cơ bản: Inspector	22
Hình 2.12. Các thành phần cơ bản: Project.....	23
Hình 2.13. Visual Studio.....	27
Hình 2.14. Unity Asset Store	28
Hình 2.15. Thiết kế nhân vật bằng Adobe Photoshop	29
Hình 2.16. C#	30
Hình 3.17. Prefab ô vật phẩm có thể tái sử dụng	36
Hình 3.18 Tạo Animation cho nhân vật.....	36
Hình 3.19. Quản lý chuyển động bằng Animator	37
Hình 3.20. Bắt phím và quy định cho Animation	37
Hình 3.21. UI Button.....	38
Hình 3.22. Chuyển đổi Font thành Font Asset.....	39
Hình 3.23. Giao diện bắt đầu Game	39
Hình 3.24. Canvas chứa giao diện tải Game	40
Hình 3.25. Phương thức hiện quá trình tải địa hình.....	40
Hình 3.26. Gán các GameObject vào Script	41
Hình 3.27. Gán Scene cho Button.....	41
Hình 3.28. Giao diện File âm thanh hợp lệ	42
Hình 3.29. Kéo thả File âm thanh trên giao diện	43
Hình 4.30. Khởi tạo địa hình 100x100.....	45

Hình 4.31. Tạo địa hình bằng PerlinNoise	46
Hình 4.32. Định nghĩa hệ sinh thái bằng màu sắc	47
Hình 4.33. Dải màu hệ sinh thái và độ nhiễu	47
Hình 4.34. Hệ sinh thái thảo nguyên	48
Hình 4.35. Hệ sinh thái rừng rậm	48
Hình 4.36. Hệ sinh thái tuyết trắng	49
Hình 4.37. Hệ sinh thái sa mạc	49
Hình 4.38. Các hệ sinh thái	50
Hình 4.39. Chunk	50
Hình 4.40. Thuật toán tải Chunk ở nơi nhân vật đứng	51
Hình 4.41. Ví dụ tải Chunk 1	51
Hình 4.42. Ví dụ tải Chunk 2	52
Hình 4.43. Bedrock	52
Hình 4.44. Overlay	54
Hình 4.45. Áp dụng Overlay	54
Hình 4.46. Hệ thống ánh sáng trong Game	55
Hình 4.47. Phân loại vật phẩm	55
Hình 4.48. Vật phẩm nền - Hoá thạch ở sa mạc	56
Hình 4.49. Vật phẩm nền - Cây và cỏ ở môi trường tự nhiên	56
Hình 4.50. Vật phẩm nền - Lớp tường trong hang động	57
Hình 4.51. Vật phẩm mặt trước - Khối đất ở các biomes	57
Hình 4.52. Vật phẩm mặt trước - Các loại quặng	58
Hình 4.53. Khởi tạo vật phẩm rơi ra	59
Hình 4.54. Khối vật phẩm rơi ra khi khai thác	59
Hình 4.55. Nhân vật cầm vật phẩm	60
Hình 4.56. Cơ chế đào khối	60
Hình 4.57. Đặt khối tại vị trí trỏ chuột	61
Hình 4.58. Công cụ búa	61
Hình 4.59. Công cụ rìu	62
Hình 4.60. Công cụ cuốc	62
Hình 4.61. Hoạt ảnh Idle	63

Hình 4.62. Hoạt ảnh Run.....	63
Hình 4.63. Hoạt ảnh Jump.....	64
Hình 4.64. Hoạt ảnh Mining	64
Hình 4.65. Animator Idle, Run, Jump	65
Hình 4.66. Animator Mining.....	65
Hình 4.67. Run và Jump.....	66
Hình 4.68, Quy định nơi dùng Animation Mining	66
Hình 4.69. Sử dụng Raycast xác định vị trí	67
Hình 4.70; Tạo Raycast.....	67
Hình 4.71. Autojump.....	67
Hình 4.72. Khởi tạo ô vật phẩm	68
Hình 4.73. Giao diện túi đồ	68
Hình 4.74. Kiểm tra vật phẩm.....	69
Hình 4.75. Thêm vật phẩm vào kho đồ.....	69
Hình 4.76. Cập nhật giao diện ô chứa vật phẩm.....	70
Hình 4.77. Loại bỏ vật phẩm.....	71
Hình 4.78. Khởi tạo ô vật phẩm ở thanh hotbar.....	71
Hình 4.79. Thanh hotbar	72

DANH MỤC CÁC CỤM TỪ VIẾT TẮT

STT	Từ viết tắt	Chữ viết đầy đủ
1	UI	User Interface (Giao diện người dùng)
2	GPU	Graphics Processing Unit (Bộ xử lý đồ họa)
3	GMS2	GameMaker Studio 2
4	GML	GameMaker Language
5	AI	Artificial Intelligence (Trí tuệ nhân tạo)

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Giới thiệu về thể loại Sandbox

1.1.1 Định nghĩa và các đặc điểm cơ bản của thể loại Sandbox

Game Sandbox, hay còn gọi là trò chơi hộp cát, là thể loại game điện tử tập trung vào sự tự do khám phá và tự do sáng tạo của người chơi. Trong game Sandbox, người chơi thường được đặt vào một thế giới mở rộng lớn với ít sự hướng dẫn hoặc nhiệm vụ cụ thể. Thay vào đó, họ được khuyến khích tự do khám phá thế giới, tương tác với môi trường và các nhân vật, và tự tạo ra mục tiêu và cách chơi cho riêng mình.

Đặc điểm chính của game sandbox:

- **Thế giới mở rộng lớn:** Game sandbox thường có thế giới mở rộng lớn cho phép người chơi tự do khám phá. Thế giới này có thể được thiết kế theo nhiều phong cách khác nhau, từ thời trung cổ huyền bí đến vũ trụ tương lai rộng lớn.
- **Ít sự hướng dẫn hoặc nhiệm vụ cụ thể:** Không giống như các thể loại game khác, game sandbox thường cung cấp ít sự hướng dẫn hoặc nhiệm vụ cụ thể cho người chơi. Thay vào đó, người chơi được tự do khám phá thế giới theo cách của riêng mình và tự tạo ra mục tiêu cho mình.
- **Tự do sáng tạo:** Game sandbox thường cung cấp cho người chơi nhiều công cụ và tài nguyên để họ có thể sáng tạo và xây dựng những thứ của riêng mình. Ví dụ, người chơi có thể xây dựng nhà cửa, chế tạo vũ khí, hoặc thậm chí tạo ra các trò chơi nhỏ trong game.
- **Gameplay phi tuyến tính:** Game sandbox thường có gameplay phi tuyến tính, nghĩa là người chơi có thể hoàn thành các nhiệm vụ và mục tiêu theo nhiều cách khác nhau. Điều này giúp tăng thêm giá trị chơi lại cho game và khiến cho mỗi lần chơi trở nên độc đáo hơn.

1.1.2 Phân tích các yếu tố trong việc thiết kế một trò chơi Sandbox.

- Nguồn gốc xuất phát của thuật ngữ

Thuật ngữ "hộp cát" bắt nguồn từ bản chất của hộp cát cho phép trẻ em tạo ra gần như bất cứ thứ gì chúng muốn trong đó. Từ quan điểm phát triển trò chơi điện tử, sandbox là trò chơi kết hợp các yếu tố của thiết kế với hệ thống trò chơi, có thể được xem là một trò chơi, cũng có thể xem sandbox là một chế độ chơi.

- Yếu tố sáng tạo trong việc thiết kế trò chơi:

Thế giới mở

- **Môi trường đa dạng:** Môc trò chơi Sandbox nên có thể giới mở rộng lớn với nhiều môi trường đa dạng, từ những khu rừng rậm rạp đến những sa mạc nóng bỏng, từ những thành phố sầm uất đến những hòn đảo hoang vắng. Điều này mang đến cho người chơi nhiều không gian để khám phá và trải nghiệm.
- **Tính tương tác:** Môi trường trong game Sandbox nên có tính tương tác cao, cho phép người chơi tương tác với các vật thể. Ví dụ, người chơi có thể phá hủy các vật thể, thu thập tài nguyên, xây dựng nhà cửa.

Hệ thống sáng tạo

- **Công cụ đa dạng:** Môc trò chơi Sandbox nên cung cấp cho người chơi nhiều công cụ đa dạng để sáng tạo, bao gồm các công cụ xây dựng, chế tạo, và thiết kế. Điều này cho phép người chơi tự do sáng tạo bất cứ thứ gì họ tưởng tượng, từ những ngôi nhà đơn giản đến những công trình kiến trúc phức tạp, hay thậm chí chế tạo những vũ khí và vật phẩm độc đáo.
- **Hệ thống sinh thái đa dạng:** Môc trò chơi Sandbox có thể có hệ thống sinh tháu đa dạng ở mỗi biomes. Điều này tạo ra thế giới game sống động và thú vị hơn.

1.1.3 Tính giáo dục của thể loại Sandbox

Không chỉ mang đến trải nghiệm giải trí thú vị mà còn ẩn chứa nhiều giá trị giáo dục to lớn. Dưới đây là một số khía cạnh chính thể hiện tính giáo dục của thể loại Sandbox:

- Phát triển tư duy sáng tạo

Tự do sáng tạo: Game Sandbox mang đến cho người chơi môi trường tự do sáng tạo không giới hạn. Họ có thể xây dựng bất cứ thứ gì họ tưởng tượng, từ những ngôi nhà đơn giản đến những công trình kiến trúc phức tạp, hay thậm chí chế tạo những vũ khí và vật phẩm độc đáo. Quá trình sáng tạo này giúp kích thích tư duy sáng tạo, khả năng giải quyết vấn đề và tư duy logic của người chơi.

Thử nghiệm và học hỏi: Game Sandbox khuyến khích người chơi thử nghiệm và học hỏi thông qua trải nghiệm. Khi xây dựng hoặc chế tạo thứ gì đó, họ có thể mắc sai lầm, nhưng chính những sai lầm này giúp họ học hỏi và cải thiện kỹ năng của mình.

- Kỹ năng giải quyết vấn đề:

Thách thức và nhiệm vụ: Game Sandbox thường có nhiều thử thách và nhiệm vụ đòi hỏi người chơi phải vận dụng kỹ năng giải quyết vấn đề để hoàn thành. Ví dụ, họ

cần thu thập tài nguyên, chế tạo công cụ, và xây dựng những công trình để vượt qua các chướng ngại vật và tiến bộ trong game.

Tư duy phản biện: Quá trình giải quyết vấn đề trong game Sandbox giúp người chơi phát triển tư duy phản biện. Họ cần phân tích tình huống, đánh giá các lựa chọn và đưa ra quyết định phù hợp để đạt được mục tiêu.

- Kỹ năng làm việc nhóm:

Chơi cùng nhau: Nhiều game Sandbox cho phép người chơi chơi cùng nhau trong một thế giới chung. Họ có thể hợp tác để xây dựng công trình, chiến đấu với kẻ thù, hoặc hoàn thành các nhiệm vụ.

Giao tiếp và phối hợp: Để thành công trong chế độ chơi tập thể, người chơi cần giao tiếp hiệu quả và phối hợp hành động với nhau. Họ cần chia sẻ thông tin, hỗ trợ lẫn nhau và đưa ra chiến lược chung để đạt được mục tiêu chung.

- Kiến thức về thế giới:

Môi trường và văn hóa: Một số game Sandbox được lấy bối cảnh trong những thế giới giả tưởng với các nền văn hóa và môi trường độc đáo. Khi chơi game, người chơi có thể học hỏi về các nền văn hóa khác nhau, hệ sinh thái đa dạng và các khía cạnh khác của thế giới.

Lịch sử và khoa học: Một số game Sandbox được lấy bối cảnh trong các giai đoạn lịch sử khác nhau hoặc dựa trên các nguyên tắc khoa học thực tế. Khi chơi game, người chơi có thể học hỏi về lịch sử, khoa học và các lĩnh vực khác một cách thú vị và hấp dẫn.

- Kỹ năng công nghệ:

Lập trình và thiết kế: Một số game Sandbox cho phép người chơi chỉnh sửa và tạo nội dung trong game, bao gồm các bản đồ, vật phẩm, và thậm chí cả mã nguồn. Hoạt động này giúp người chơi học hỏi về lập trình, thiết kế và các kỹ năng công nghệ khác.

Sử dụng công cụ: Game Sandbox thường cung cấp nhiều công cụ và tài nguyên cho người chơi sử dụng để xây dựng và sáng tạo. Việc sử dụng các công cụ này giúp người chơi phát triển kỹ năng sử dụng máy tính và làm quen với các công nghệ mới.

Nhìn chung, game Sandbox mang đến nhiều lợi ích giáo dục cho người chơi, giúp họ phát triển tư duy sáng tạo, kỹ năng giải quyết vấn đề, kỹ năng làm việc nhóm, kiến thức về thế giới và kỹ năng công nghệ. Do đó, game Sandbox có thể được xem như một công cụ giáo dục hiệu quả để bổ sung cho phương pháp giảng dạy truyền thống.

1.1.4 Các tựa Game Sandbox thành công

Đầu tiên phải kể đến là tượng đài của thể loại game Sandbox đó chính là Minecraft. Minecraft là một trong những trò chơi bán chạy nhất mọi thời đại, với hơn 238 triệu bản được bán ra tính đến năm 2020. Nó đã nhận được nhiều giải thưởng và lời khen ngợi từ giới phê bình vì lối chơi sáng tạo, thế giới mở và khả năng tùy chỉnh cao. Đè tài đã lấy cảm hứng lối chơi từ việc xây dựng công trình bằng các khối khai thác được để xây dựng thành một công trình với sự sáng tạo của người chơi.

Đặc điểm:

- **Thế giới mở rộng lớn** được tạo ra theo thủ tục.
- **Hệ thống chế tạo** cho phép người chơi tạo ra nhiều vật phẩm và công trình khác nhau.
- **Chế độ chơi đa dạng**, bao gồm sinh tồn, sáng tạo và phiêu lưu.
- Hỗ trợ **cộng đồng mạnh mẽ** với nhiều bản mod và nội dung do người dùng tạo.
- **Lối chơi sáng tạo**: Khuyến khích người chơi thử nghiệm và đưa ra các giải pháp độc đáo cho các vấn đề.
- **Có tính giáo dục cao**: Minecraft không chỉ là một trò chơi giải trí đơn thuần mà còn có giá trị giáo dục cao.
 - Minecraft đặt người chơi vào một thế giới mở rộng lớn, nơi họ phải tự mình khám phá và tìm cách tồn tại. Điều này giúp trẻ em phát triển kỹ năng giải quyết vấn đề, tư duy phản biện và ra quyết định.

Ví dụ: người chơi phải tìm kiếm thức ăn và nơi trú ẩn, xây dựng công cụ và vũ khí, và bảo vệ bản thân khỏi những kẻ thù. Để làm được điều này, họ cần phải suy nghĩ sáng tạo và đưa ra các giải pháp cho các vấn đề khác nhau.

◦ Minecraft cung cấp cho người chơi vô số công cụ và vật liệu để xây dựng bất cứ thứ gì họ có thể tưởng tượng. Điều này giúp trẻ em phát triển kỹ năng sáng tạo, tư duy logic và khả năng giải quyết vấn đề không gian.

Ví dụ: người chơi có thể xây dựng nhà cửa, lâu đài, thành phố thậm chí là những cỗ máy phức tạp. Quá trình này đòi hỏi họ phải lên kế hoạch, thiết kế và xây dựng công trình của mình một cách cẩn thận, đồng thời giải quyết các vấn đề về kỹ thuật và thẩm mỹ.



Hình 1.1 Game Minecraft

Tiếp đến là tựa game cũng đã tạo cảm hứng trong quá trình thực hiện đề tài này, đó chính là **Terraria**. Terraria là một trò chơi phiêu lưu hành động 2D thế giới mở được đánh giá cao, đã bán được hơn 35 triệu bản kể từ khi phát hành vào năm 2011. Nó được khen ngợi vì lối chơi khám phá và xây dựng sáng tạo, thế giới rộng lớn và nhiều nội dung.

Đặc điểm:

- Thế giới mở được tạo theo thủ tục trải dài từ trái sang phải và từ trên xuống dưới.
- Hệ thống chế tạo cho phép người chơi tạo ra nhiều vật phẩm và công trình khác nhau.
- Nhiều loại kẻ thù và trùm để chiến đấu.
- Nhiều sự kiện và thử thách khác nhau để khám phá.
- Lối chơi sáng tạo: Khuyến khích người chơi thử nghiệm và đưa ra các giải pháp độc đáo cho các vấn đề.



Hình 1.2. Game Terraria

1.2 Giới thiệu đề tài

1.2.1 Mục tiêu đề tài

Thể loại Sandbox là một thể loại video game dựa trên mô hình hộp cát của trẻ em, đem đến yếu tố sáng tạo cho người chơi thông qua các trải nghiệm khi tương tác một cách tự do trong thế giới trong game thường là thế giới mở. Vì thế mà các game Sandbox là sự kết hợp giữa yếu tố sáng tạo với các thể loại game khác để trò chơi tăng thêm phần nổi bật.

Các tựa game thể loại này hiện nay thường lồng ghép yếu tố sinh tồn, thế giới mở để khiến nội dung game đa dạng và thú vị hơn, gameplay phi tuyến tính nghĩa là người chơi có thể chọn chơi theo nhiều mục đích khác nhau như thám hiểm địa hình, chiến đấu với độ khó tăng dần, hoặc đơn giản là xây dựng thoả mãn trí sáng tạo của mình. Điều này giúp tăng thêm giá trị chơi lại cho game và khiến cho mỗi lần chơi trở nên độc đáo hơn. Đồ họa 2D đơn giản nhưng rất được nhiều người chơi yêu thích như các tựa game tiêu biểu Minecraft, Terraria, Grand Theft Auto V,...

Ở đề tài này, em muốn làm nổi bật cốt lõi của một game thể loại Sandbox là yếu tố sáng tạo thông qua việc thám hiểm địa hình và xây dựng công trình, thu hút cộng đồng người chơi trực tuyến cùng chia sẻ tác phẩm của mình với nhau, đồng thời mang tính giáo dục rất cao khi áp dụng vào hoạt động ngoại khoá hay giảng dạy.

1.2.2 Phương pháp nghiên cứu

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

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

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

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

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

Xác định các yếu tố cần thiết để tạo ra yếu tố sáng tạo trong trò chơi Sandbox

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

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

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

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

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

1.3 Nội dung Game

1.3.1 Phạm vi nghiên cứu

Đề tài này sẽ tập trung vào yếu tố sáng tạo của thể loại Sandbox trên địa hình ngang sử dụng Unity Engine 2D. Phạm vi nghiên cứu sẽ bao gồm các giai đoạn từ phát triển ý tưởng đến triển khai và kiểm tra:

- Phát triển ý tưởng:** Nghiên cứu sẽ bắt đầu bằng việc nghiên cứu các trò chơi Sandbox hiện tại để thu thập ý tưởng và yêu cầu. Điều này bao gồm cả việc xác

định các yếu tố gameplay như cơ chế tạo thế giới, tương tác công cụ với môi trường, tương tác người chơi với các khối (block)

- **Thiết kế và phát triển:** Sau khi có ý tưởng cơ bản, giai đoạn này sẽ tập trung vào việc thiết kế và phát triển game. Điều này bao gồm việc tạo các mô hình 2D cho nhân vật, địa hình, và môi trường chơi, cũng như việc lập trình các tính năng gameplay như chuyển động, tương tác công cụ, phá huỷ khối và đặt khối, hệ thống kho đồ.
- **Tối ưu hóa và kiểm tra:** Phần quan trọng của quá trình phát triển là tối ưu hóa game để đảm bảo hiệu suất tốt trên nhiều loại thiết bị. Giai đoạn này bao gồm kiểm tra và sửa lỗi để cải thiện trải nghiệm người chơi.
- **Đánh giá và phân tích:** Cuối cùng, nghiên cứu sẽ đánh giá trò chơi đã phát triển thông qua các phương tiện như cuộc thử nghiệm người dùng và phản hồi từ cộng đồng người chơi. Phần này sẽ giúp xác định điểm mạnh, điểm yếu và tiềm năng cải thiện trong tương lai của trò chơi.

1.3.2 Mô tả Game

Người chơi sẽ nhập vai vào nhân vật trong game với 3 công cụ mặc định mang theo người trong cả hành trình khám phá các hệ sinh thái trên mặt đất và dưới lòng đất, dùng công cụ thích hợp để khai thác tài nguyên, dùng tài nguyên thu thập được để xây dựng công trình theo trí tưởng tượng. Có nhiều loại tài nguyên và đồ trang trí, rất ít cơ hội khai thác được vật phẩm quý hiếm.

CHƯƠNG II: CƠ SỞ LÝ THUYẾT CÁC CÔNG CỤ VÀ PHẦN MỀM SỬ DỤNG

2.1 Các công cụ thường được sử dụng để phát triển game Sandbox

2.1.1 GameMaker Studio 2

GameMaker Studio 2 (GMS2) là một công cụ tạo game 2D phổ biến được sử dụng để tạo ra nhiều loại trò chơi khác nhau, bao gồm game sandbox. Nó cung cấp một giao diện người dùng trực quan, nhiều tính năng mạnh mẽ. GMS2 cung cấp nhiều tính năng mạnh mẽ để phát triển game sandbox, bao gồm:

- **Rooms:** GMS2 sử dụng Rooms để tạo ra các thế giới trò chơi. Mỗi Room có thể chứa các đối tượng, sprite, nền và các yếu tố khác.
- **Physics:** GMS2 sử dụng hệ thống Physics để mô phỏng vật lý trong game, giúp tạo ra các tương tác vật lý chân thực giữa các vật thể trong thế giới trò chơi.
- **GML:** GMS2 sử dụng GML (GameMaker Language) để lập trình game. GML là một ngôn ngữ lập trình đơn giản và dễ học, với nhiều tính năng hỗ trợ phát triển game.
- **Asset Marketplace:** GMS2 có Asset Marketplace cung cấp nhiều tài nguyên miễn phí và trả phí cho các nhà phát triển game, bao gồm sprite, tilemaps, âm thanh và mã.

Ưu điểm:

- **Dễ sử dụng:** GMS2 có giao diện người dùng trực quan và dễ học, ngay cả đối với những người mới bắt đầu.
- **Nhiều tính năng:** GMS2 cung cấp nhiều tính năng mạnh mẽ để phát triển game, bao gồm mô phỏng vật lý, AI, âm thanh, đồ họa và mạng.
- **Linh hoạt:** GMS2 có thể được sử dụng để tạo ra nhiều loại trò chơi 2D khác nhau, bao gồm game sandbox, game nhập vai, game platformer, game puzzle và nhiều hơn nữa.

Nhược điểm của GMS2:

- **Chỉ hỗ trợ 2D:** GMS2 chỉ hỗ trợ phát triển game 2D, không hỗ trợ game 3D.
- **Có thể hạn chế:** GMS2 có thể hạn chế cho các dự án lớn và phức tạp.
- **Hiệu suất:** Hiệu suất của game GMS2 có thể bị ảnh hưởng bởi phần cứng máy tính.

2.1.2 Unreal Engine

Unreal Engine là một công cụ tạo game 3D phổ biến được sử dụng để tạo ra nhiều loại trò chơi khác nhau, bao gồm game sandbox. Nó cung cấp đồ họa tiên tiến, hiệu suất cao và nhiều tính năng mạnh mẽ để hỗ trợ việc phát triển game sandbox. Unreal Engine cung cấp nhiều tính năng mạnh mẽ để phát triển game sandbox, bao gồm:

- **World Composition:** World Composition cho phép tạo ra những thế giới mở rộng lớn và chi tiết.
- **Physics:** Unreal Engine sử dụng PhysX để mô phỏng vật lý thực tế trong game, giúp tạo ra các tương tác vật lý chân thực giữa các vật thể trong thế giới trò chơi.
- **C++:** Unreal Engine sử dụng C++ để lập trình game, là một ngôn ngữ mạnh mẽ và linh hoạt với nhiều tính năng hỗ trợ phát triển game.
- **Marketplace:** Unreal Engine Marketplace cung cấp nhiều tài nguyên miễn phí và trả phí cho các nhà phát triển game, bao gồm mô hình 3D, kết cấu, âm thanh và mã.

Ưu điểm:

- **Đồ họa tiên tiến:** Unreal Engine sử dụng công nghệ đồ họa tiên tiến nhất, cho phép tạo ra những thế giới trò chơi đẹp và chân thực.
- **Hiệu suất cao:** Unreal Engine được tối ưu hóa cho hiệu suất cao, giúp tạo ra những game mượt mà và nhanh chóng trên nhiều nền tảng khác nhau.
- **Nhiều tính năng:** Unreal Engine cung cấp nhiều tính năng mạnh mẽ để phát triển game, bao gồm mô phỏng vật lý, AI, âm thanh, đồ họa, mạng và nhiều hơn nữa.

Nhược điểm:

- **Có thể phức tạp:** Unreal Engine có thể phức tạp để học hỏi và sử dụng cho các nhà phát triển mới bắt đầu.
- **Yêu cầu phần cứng mạnh:** Unreal Engine yêu cầu phần cứng mạnh để chạy trơn tru.
- **Kích thước lớn:** Các dự án Unreal Engine có thể có kích thước lớn và tốn nhiều thời gian để phát triển.

2.1.3 Godot

Godot là một công cụ tạo game 2D và 3D miễn phí và mã nguồn mở, được đánh giá cao bởi tính linh hoạt, dễ sử dụng và cộng đồng hỗ trợ tích cực. Nó là lựa

chọn tuyệt vời cho các nhà phát triển game, đặc biệt là những người mới bắt đầu, muốn tạo ra các game sandbox hấp dẫn và đầy thử thách.

Ưu điểm:

- **Mã nguồn mở:** Godot hoàn toàn miễn phí để sử dụng và phát triển, cho phép bạn tạo ra game thương mại mà không phải lo lắng về phí bản quyền. Mã nguồn mở của nó cũng mang đến sự minh bạch và khả năng tùy chỉnh cao, giúp bạn dễ dàng điều chỉnh công cụ theo nhu cầu cụ thể của mình.
- **Dễ sử dụng:** Godot sở hữu giao diện trực quan và thân thiện với người dùng, giúp bạn dễ dàng học hỏi và bắt đầu tạo game ngay cả khi không có kinh nghiệm lập trình trước đây. Các công cụ chỉnh sửa cảnh trực quan, hệ thống kịch bản đơn giản và hệ thống tài liệu hướng dẫn chi tiết sẽ hỗ trợ bạn trong suốt quá trình phát triển.
- **Tính linh hoạt:** Godot là công cụ đa năng, có thể được sử dụng để tạo ra nhiều loại game khác nhau, bao gồm game 2D, game 3D, game nhập vai, game phiêu lưu, game giải đố và tất nhiên là game sandbox. Nó cung cấp nhiều tính năng mạnh mẽ để hỗ trợ việc phát triển game sandbox,

Nhược điểm:

- **Khả năng 3D còn hạn chế:** Godot đang được phát triển và cải thiện liên tục, nhưng khả năng 3D của nó vẫn chưa mạnh mẽ bằng Unity hay Unreal Engine. Nếu bạn muốn tạo ra game 3D với đồ họa cao cấp và hiệu ứng phức tạp, Godot có thể không phải là lựa chọn phù hợp nhất.
- **Hiệu suất có thể bị ảnh hưởng:** Hiệu suất của game Godot có thể bị ảnh hưởng bởi phần cứng máy tính. Nếu bạn muốn tạo ra game 3D nặng hoặc game 2D với nhiều tính năng phức tạp, bạn cần đảm bảo có phần cứng đủ mạnh để hỗ trợ.
- **Thiếu tài liệu và hướng dẫn** cho một số tính năng: Godot có hệ thống tài liệu và hướng dẫn chi tiết cho nhiều tính năng, nhưng một số tính năng mới hoặc ít sử dụng có thể chưa có tài liệu đầy đủ. Điều này có thể khiến bạn gặp khó khăn trong việc học cách sử dụng các tính năng đó.
- **Hạn chế về tính năng tích hợp mạng:** Godot hỗ trợ phát triển game mạng, nhưng các tính năng tích hợp mạng của nó có thể không mạnh mẽ hoặc linh hoạt bằng các công cụ khác như Unity. Nếu muốn tạo ra game mạng với nhiều người chơi hoặc các tính năng mạng phức tạp, người làm game có thể cần sử dụng các thư viện bên thứ ba hoặc tự viết mã cho các tính năng đó.

2.1.4 Unity

Unity là một công cụ tạo game 2D/3D phổ biến được sử dụng để tạo ra nhiều loại trò chơi khác nhau, bao gồm game sandbox. Nó cung cấp một giao diện người dùng trực quan, nhiều tính năng mạnh mẽ và một cộng đồng lớn để phát triển game sandbox, bao gồm:

- **Terrain:** Unity cung cấp một hệ thống Terrain cho phép tạo ra các thế giới mở rộng lớn và chi tiết.
- **PhysX:** Unity sử dụng PhysX để mô phỏng vật lý thực tế trong game, giúp tạo ra các tương tác vật lý chân thực giữa các vật thể trong thế giới trò chơi.
- **C#:** Unity sử dụng C# để lập trình game, là một ngôn ngữ mạnh mẽ và linh hoạt với nhiều tính năng hỗ trợ phát triển game.
- **Asset Store:** Unity có một Asset Store cung cấp nhiều tài nguyên miễn phí và trả phí cho các nhà phát triển game, bao gồm mô hình 3D, kết cấu, âm thanh và mã.

Ưu điểm:

- **Dễ sử dụng:** Unity có giao diện người dùng trực quan và dễ học, ngay cả đối với những người mới bắt đầu.
- **Nhiều tính năng:** Unity cung cấp nhiều tính năng mạnh mẽ để phát triển game, bao gồm mô phỏng vật lý, AI, âm thanh, đồ họa và mạng.
- **Linh hoạt:** Unity có thể được sử dụng để tạo ra nhiều loại trò chơi khác nhau, bao gồm game sandbox, game nhập vai, game FPS, game RTS và nhiều hơn nữa.

Nhược điểm:

- **Có thể phức tạp:** Unity có thể phức tạp để học hỏi và sử dụng cho các dự án lớn.
- **Hiệu suất:** Hiệu suất của game Unity có thể bị ảnh hưởng bởi phần cứng máy tính.

Có thể thấy, Unity là công cụ hữu hiệu nhất khi có thể đáp ứng lập trình cả game 2D và 3D nhưng không yêu cầu cấu hình cao, có thể lập trình game đa nền tảng cùng bộ công cụ và cộng đồng người sử dụng khổng lồ. Chính vì vậy Unity là lựa chọn hàng đầu để bắt tay vào thực hiện đồ án lần này

2.2 Công cụ Unity trong phát triển game

2.2.1 Giới thiệu chung

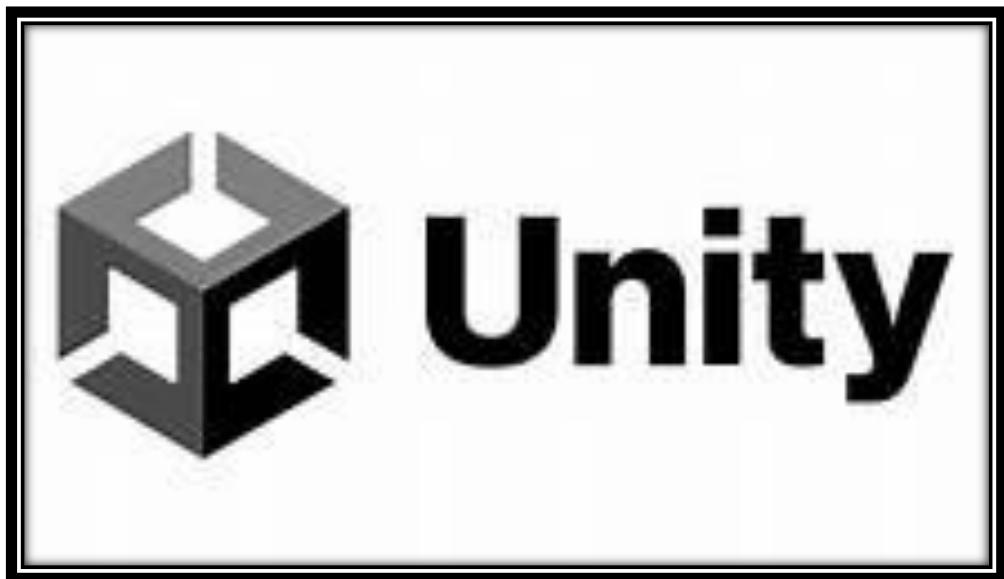
Unity là một Game Engine đa nền tảng được phát triển bởi công ty Unity Technologies, chủ yếu được dùng để phát triển video game cho 21 nền tảng bao gồm máy tính, game consoles (như PlayStation) và điện thoại,...

Hầu hết các nền tảng chơi game phổ biến hiện nay đều được Unity hỗ trợ như: mobile (Android, iOS, WindowPhone), PC (Window, MacOS), Consoles (Nintendo, PlayStation), Browsers (sử dụng WebGL), ... Không chỉ vậy, Unity cũng sẵn có các tính năng công nghệ thực tế ảo VR hay thực tế tăng cường AR có thể chạy tốt trên các thiết bị như Google Cardboard, GearVR, Hololens, Oculus, ...

Về mặt đồ họa, Unity hỗ trợ cả 2D và 3D, có thể sử dụng các đồ họa APIs phổ biến như Direct3D, OpenGL, WebGL, ... Unity cung cấp cho các lập trình viên những công cụ dựng hình, kết xuất đồ họa, xử lý về cả âm thanh, hình ảnh, đồng thời có sẵn công cụ vật lý (tính toán và phát hiện va chạm).

Hơn 50% số lượng game trên thị trường được sản xuất bởi Unity. Một vài tựa game vô cùng nổi tiếng được tạo ra bởi Unity có thể được kể đến như Pokémon Go, Hearthstone, Ori And The Blind Forest, Monument Valley, Axie Infinity,... Độ “phủ sóng” của Unity rất rộng, có thể được áp dụng phổ biến trong nhiều dòng game khác nhau từ game “hạng nặng” Triple A (AAA) cho đến game giáo dục đơn giản cho con nít.

Lập trình Unity 2D và 3D được lập trình dựa vào 3 ngôn ngữ chính là C#, Boo và UnityScript. Thông thường, ngôn ngữ chính mà lập trình viên Unity sử dụng phổ biến nhất hiện nay là C#



Hình 2.3. Unity

2.2.2 Quá trình phát triển của Unity

Ra mắt đầu tiên vào năm 2005 tại sự kiện Apple's Worldwide Developer Conference bởi nhà sáng lập David Helgason, trải qua hơn 12 năm phát triển, nay Unity đã có version 5.5 hoàn thiện hơn về rất nhiều mặt. Tháng 5-2012 theo cuộc khảo sát Game Developer Megazine được công nhận là Game engine tốt nhất cho mobile. Năm 2014 Unity thắng giải “Best Engine” tại giải UK’s annual Develop Industry Excellence.

- **Những năm đầu**

Trong những năm đầu, Unity chủ yếu được sử dụng để tạo trò chơi dựa trên web và trò chơi di động đơn giản. Tuy nhiên, khi công cụ này phát triển và các tính năng mới được thêm vào, nó ngày càng trở nên phổ biến để phát triển các trò chơi phức tạp và chất lượng cao cho nhiều nền tảng, bao gồm PC, bảng điều khiển, thiết bị di động, thực tế ảo (VR) và thực tế tăng cường (AR) nền tảng.

- **Cập nhật lớn**

Trong những năm qua, Unity đã trải qua một số cập nhật và cải tiến lớn, giới thiệu các tính năng mới như trình chỉnh sửa trực quan, hỗ trợ trò chơi 2D, mô phỏng vật lý, trí tuệ nhân tạo và kết nối mạng nhiều người chơi. Ngày nay, Unity là một trong những công cụ phát triển trò chơi phổ biến nhất trên thế giới với cộng đồng nhà phát triển và người dùng rộng lớn và tích cực.

2.2.3 Ưu điểm và nhược điểm của Unity

Ưu điểm của Unity:

- **Editor:** Với Editor, nhà phát triển không cần thiết phải viết Code để sắp đặt các đối tượng trong Game như những Engine khác mà Developer có thể kéo thả, thay đổi vị trí của từng đối tượng trong Game trực tiếp trên Editor.
- **Đa nền tảng** là lợi ích thứ 2 rất quan trọng với nhiều công ty cũng như developer. Vì với việc bạn tạo ra Game mà Game đó có thể chạy được trên hầu hết những hệ điều hành quan trọng như Desktop (Mac, Window và Linux) hay Mobile (iOS, Android) hoặc Web (WebGL) thì cũng đã tiết kiệm công sức cũng như chi phí rất nhiều cho doanh nghiệp đó.
- **Miễn phí:** Và yếu tố cuối cùng chính là chi phí. Với Unity, miễn phí là một điểm thu hút rất nhiều Developer chọn làm việc với game engine này. Tuy nhiên, với các game được tạo ra miễn phí thì bắt buộc phải có Logo Unity trong Game.
- **Cộng đồng hỗ trợ lớn:** Unity có cộng đồng để hỗ trợ mỗi khi người dùng có thắc mắc về cách sử dụng hoặc về các phiên bản cập nhật, khi gặp lỗi, bug. Ngoài ra, Unity còn có “chợ” plug-in vô cùng phong phú. Trên chợ có những gói package do chính Unity hoặc các nhà phát triển khác sản xuất. Thậm chí, có những nhà phát triển đăng tải toàn bộ một trò chơi hoặc một mô hình 3D hoàn chỉnh

Nhược điểm của Unity:

- **Dung lượng Unity game bundle khá lớn**

So với những game engine khác, Unity sản xuất game có dung lượng nặng nên đây là một điểm trừ lớn. Thậm chí, game web do Unity sản xuất có thể có dung lượng lên đến cả trăm MB nên web chạy không nổi. Chính vì thế, cũng cùng một game đó thì game mobile lại chạy tốt trong khi game web lại giật, lag.

Lý giải nguyên nhân cho vấn đề này, có thể nhận định rằng những game engine nhiều tool hỗ trợ để mau ra sản phẩm thì sẽ phải có nhiều layer, structure phức tạp nên khiến cho sản phẩm game nặng, dư thừa nhiều tính năng không cần thiết.

- **Các phiên bản cập nhật liên tục**

Unity cho ra mắt nhiều phiên bản cập nhật liên tục trong một năm và nhiều năm liên tục. Nếu phiên bản cập nhật được ra mắt trong cùng một năm thì vẫn có thể sử dụng song song nhiều phiên bản cùng năm.

Còn nếu trong trường hợp bản cập nhật khác năm, và trong nội bộ một team, hoặc giữa team phát triển và khách hàng, sử dụng các phiên bản Unity khác nhau với nhau thì khi push code, đẩy code lên cho member thì sẽ bị lỗi hình ảnh, script, code,...

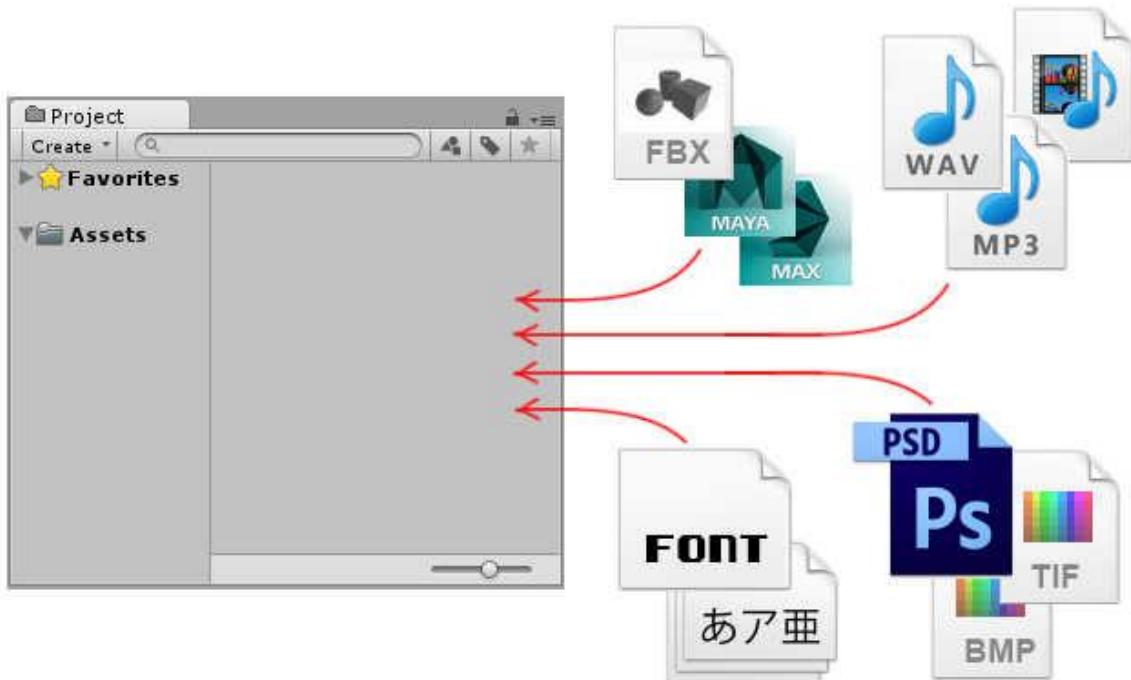
thậm chí là mất hết tiến độ.

Tuy nhiên, điều khó là nếu như nhà phát triển nhận nhiều dự án cùng một lúc mà mỗi khách hàng lại sử dụng một phiên bản Unity khác nhau để phát triển game thì bạn bắt buộc phải thay đổi qua lại.

2.2.4 Các thành phần trong Unity Editor

- Assets

Một asset là bất kỳ tài nguyên nào: âm thanh, texture (hình ảnh), model (vật thể 3D)... mà người dùng cần để cấu thành một GameObject.



Hình 2.4. Các thành phần cơ bản: Assets

- Sprite

Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó.

- Animation Keyframe

Animation Keyframe hay Frame là một trạng thái của một animation. Có thể được tạo nên từ 1 sprite hay nhiều sprite khác nhau.

- Camera

Là một game object đặc biệt trong scene, dùng để xác định tầm nhìn, quan sát các đối tượng khác trong game.

- Transform

Là 3 phép biến đổi tịnh tiến, quay theo các trục, và phóng to thu nhỏ một đối tượng



Hình 2.5. Các thành phần cơ bản: Transform

- Scenes

Một scene chứa các vật thể trong game (game objects) tương tác với thế giới, trong đó có player của chúng ta.

- Game Object

Các Game Objects là toàn bộ các thứ trong Scene, có thể nhìn thấy hoặc không nhìn thấy.

Không chỉ các objects thực như bàn, ghế, rìu, kiếm mà còn các objects hỗ trợ như audio sources, cameras, light sources (nguồn sáng),... Các game object rỗng rất hữu dụng, đặc biệt là trong hệ thống cấp bậc parent-child (bạn có thể xem trong Hierarchy tab).

Tất cả các game objects (kể cả game objects rỗng) đều có vị trí và độ xoay trong thế giới game, nó có thể di chuyển, xoay và scale. Lưu ý rằng các yếu tố này sẽ chịu ảnh hưởng bởi cấp bậc như hình trên.

Có thể tắt (disabled) hoặc bật (enabled) gameObject tùy ý, các thành phần cấu tạo nên một GameObject là Components.

- Components

Một GameObject thường sẽ được cấu tạo từ nhiều thành phần (components) nên nó có thể là các hình ảnh, những hành động của nhân vật, mã điều khiển,... Functions thường kế thừa từ MonoBehaviour class và có thể ghi đè bên trong những class còn có thể thực hiện cho những sự kiện quan trọng nào đó.

Hai sự kiện này thường được dùng phổ biến trong một Component đó là: Start() tiến hành chạy thêm 1 lần duy nhất trước khi hàm update. Update() là sẽ thực hiện sau một vòng lặp chạy liên tục. Nó sẽ được gọi một lần dành cho mỗi khung hình (thường là 25 khung hình mỗi giây). Cứ mỗi thứ như vậy gọi sẽ được gọi là một component của GameObject.

Ví dụ về component:

- Mesh Filter và Mesh Renderer là những components định nghĩa các bề mặt của khối vật thể 3D.
- Rigidbody component làm nhiệm vụ định nghĩa cách mà một object của thế di chuyển một cách “vật lý” trong Game như cân nặng, sức cản không khí, trọng lượng,...
- Collider component định nghĩa một khối bao quanh GameObject để xử lý va chạm với các GameObject cũng có Collider khác trong scene.
- Vị trí, độ xoay và scale của một GameObject được định nghĩa bởi component có tên là Transform.

Có thể điều chỉnh các thông số của component ở ngay trên Unity Editor, trong cửa sổ Inspector.

- Scripts

Script là dạng tập tin có chứa những đoạn mã nguồn được sử dụng với mục đích khởi tạo cũng như xử lý đối tượng trong game.

Đối với Unity, có thể sử dụng C#, Java Script, Boo để thực hiện lập trình Script. Ngoài ra, có thể điều chỉnh thuộc tính cho component khác thông qua script. Và script thường được xem như là “não” của con người bởi nó có thể điều khiển được những bộ phận còn lại.

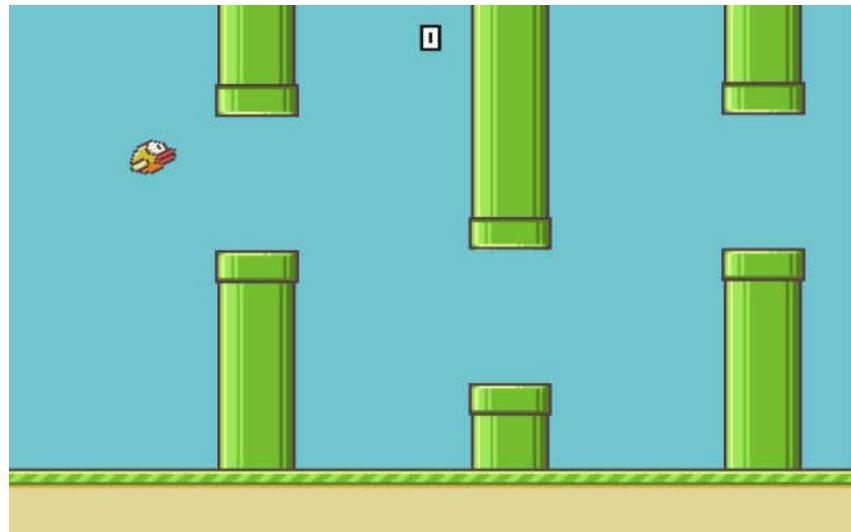
- Prefabs

Prefab là một gameObject được “cache” lại nhằm tái sử dụng, thường được dùng để tạo nhiều instances của một object nào đó.

Giả sử tạo một gameObject cần ghép từ 5 bộ phận, sẽ rất mất thời gian nếu mỗi lần tạo gameObject đó phải ngồi ghép lại 5 bộ phận đó. Hãy biến nó thành Prefab và tạo ra nhiều bản sao.

Prefabs thường được sử dụng theo 2 cách:

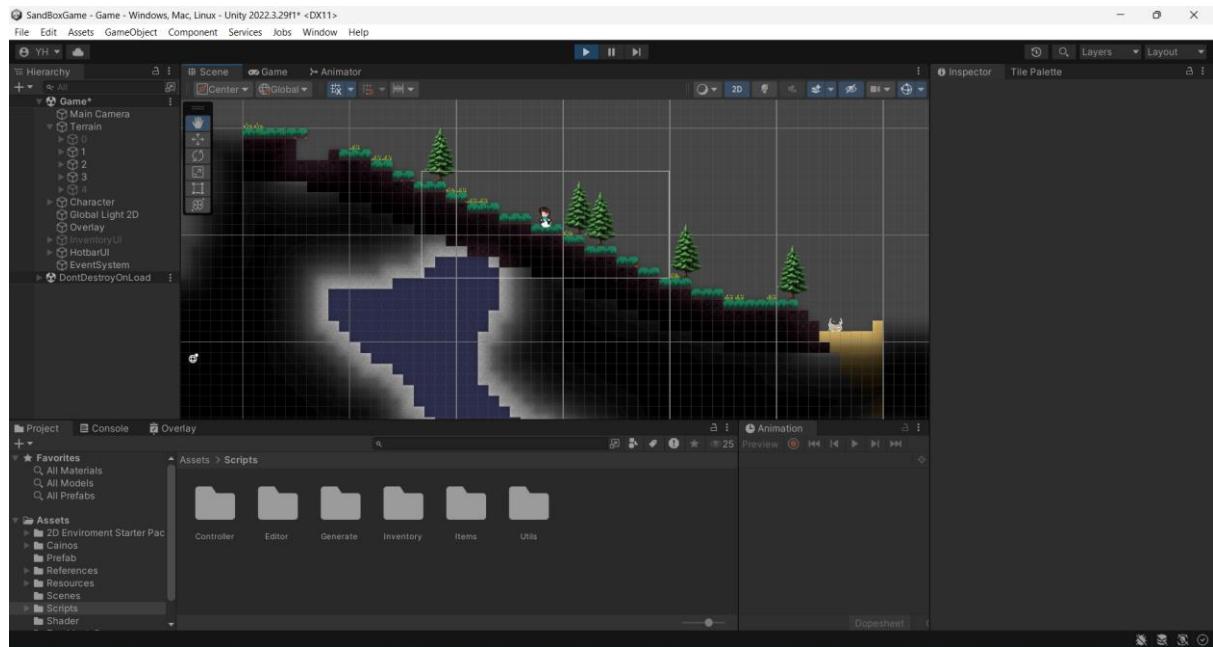
- Thiết kế level cho game, có thể tạo một cái cột đèn và biến nó thành prefab, tạo ra nhiều bản sao và đặt nó khắp mọi nơi trong map. Nếu muốn đổi độ sáng của toàn bộ cột đèn, thay vì đổi từng cái một, thì chỉ cần đổi độ sáng của “cột đèn prefab”
- Cách dùng thứ 2: tạo GameObject ngay trong màn chơi, giống như Flappy Bird chẳng hạn, các cột màu xanh hiện ra liên tục, mỗi lần hiện như vậy tức là đang “tạo bản sao” của cột.



Hình 2.6. Ví dụ Prefab trong Flappy Bird

- Scene View

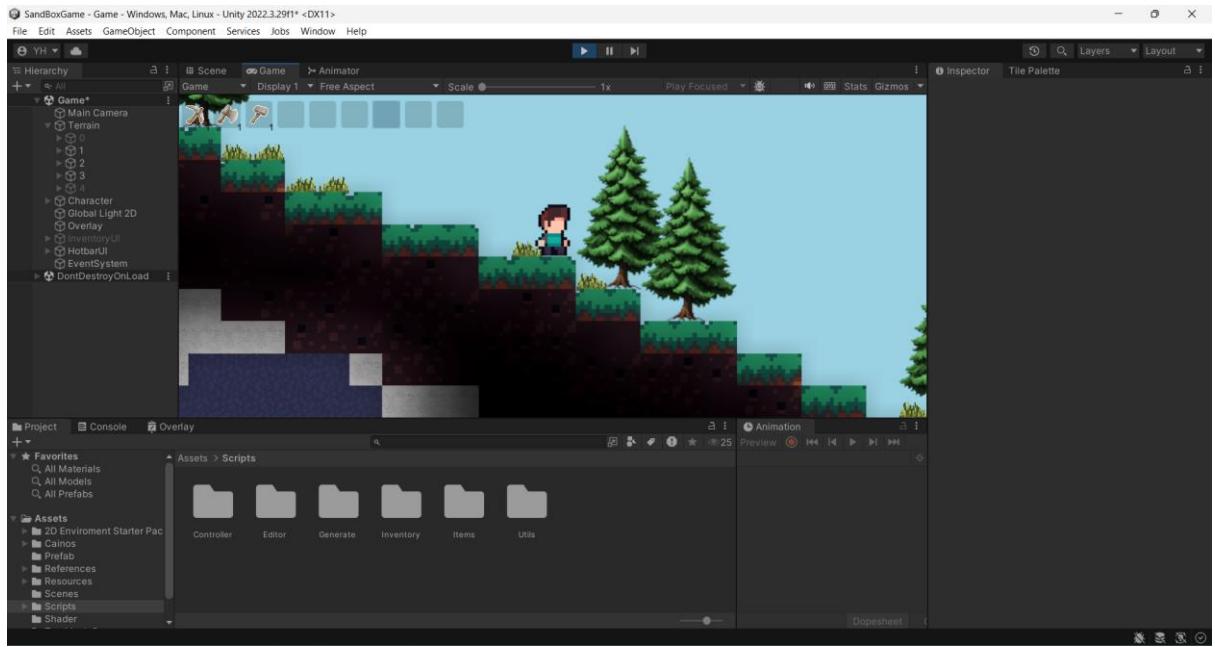
Đây là cửa sổ của thế giới game trong scene hiện tại, nơi lưu trữ toàn bộ các elements (có thể nhìn thấy hoặc không) của scene. Có thể theo dõi được vị trí tương đối của chúng cũng như thực hiện một số chỉnh sửa đơn giản.



Hình 2.7. Các thành phần cơ bản: Scene View

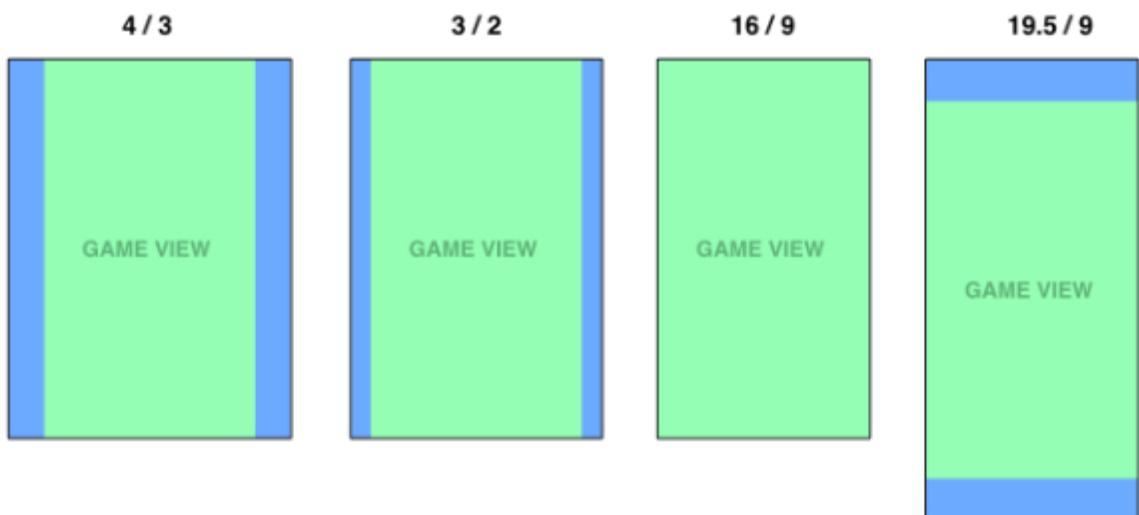
- Game View

Đây là màn hình mà player (người dùng) sẽ nhìn thấy khi chơi game của bạn. các vật thể được hiển thị nhò camera.



Hình 2.8. Các thành phần cơ bản: Game View

Chúng ta cũng có thể thay đổi tỉ lệ khung hình của game như 5:4, 4:3, 3:2:



Hình 2.9. Tỉ lệ khung hình

- Hierarchy

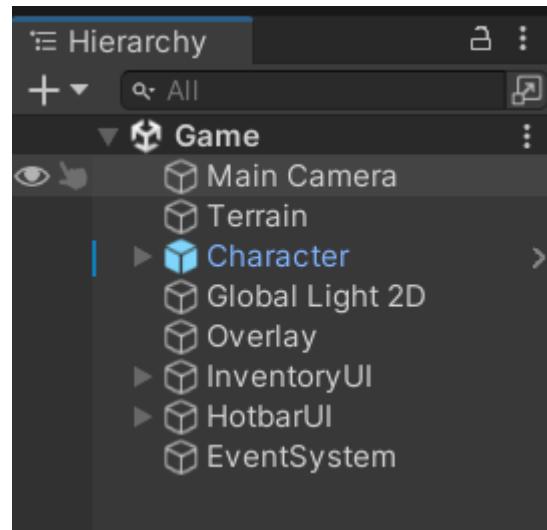
Tab này sẽ lưu trữ toàn bộ các gameObjects có trong scene của bạn, đồng thời có một đặc điểm về gameObjects mà bạn cần lưu ý đó là tính phân cấp (hay phân bậc) thành cây.

Khi di chuyển hay xoay gameObject cha, các children của nó cũng sẽ di chuyển và xoay tương ứng. Điều này cũng dễ hiểu, giả sử:

gameObject Player nhặt được 2 gameObjects giày, như vậy bạn sẽ muốn 2

gameObjects giày làm con (child) của Player. Như vậy khi player di chuyển, giày của bạn sẽ luôn dính vào chân của Player.

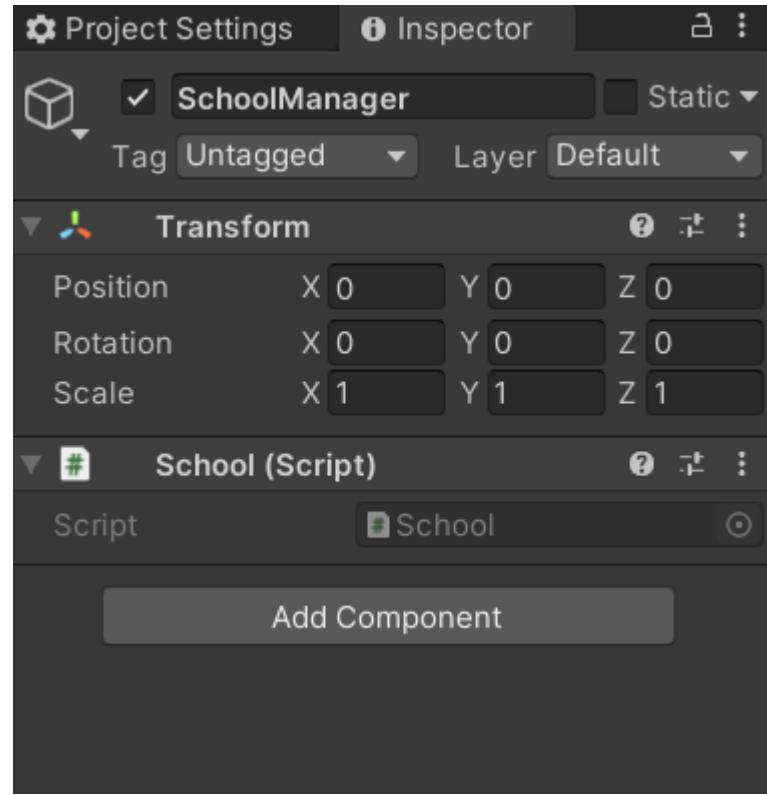
Trong gameObject Building chứa nhiều gameObjects Room, mỗi gameObject Room lại có nhiều gameObject Furnitures như vậy khi di chuyển Building bạn sẽ có thể di chuyển toàn bộ trong một lần thay vì di chuyển từng cái.



Hình 2.10. Các thành phần cơ bản: Hierachy

- Inspector

Inspector giúp theo dõi các thông số của một/ nhiều gameObject khi nhấn vào một gameObject bất kỳ trên tab Hierarchy.



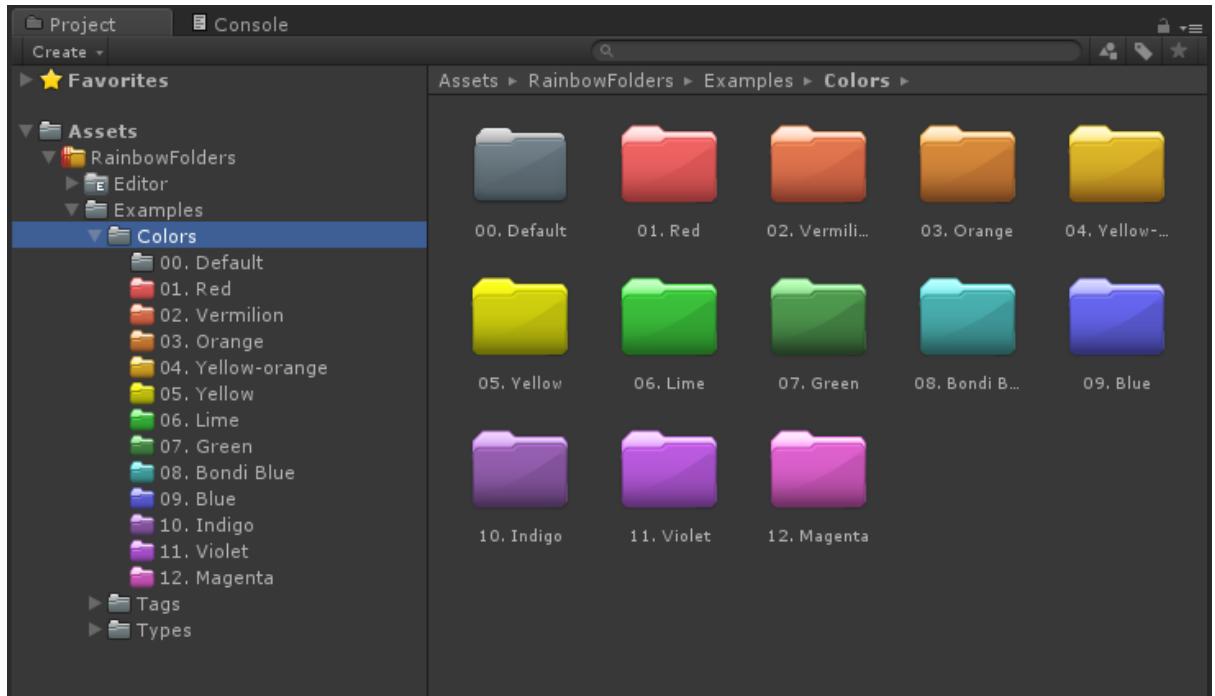
Hình 2.11. Các thành phần cơ bản: Inspector

Giả dụ có một gameObject gọi là Player, khi click vào nó trên tab Hierachy, chúng ta sẽ theo dõi được vị trí x, y và z trong không gian, tốc độ, hình ảnh hiển thị,... trên tab Inspector.

Tối thiểu nhất, khi click vào một gameObject bất kỳ sẽ theo dõi được thông số của component Transform (hoặc RectTransform). Tất nhiên nếu bạn theo nhiều components khác như Animator, MeshRenderer, Rigidbody,... thì cũng theo dõi được thông số của các components tương ứng với gameObject đã chọn.

- Project

Tab project sẽ lưu trữ toàn bộ các assets mà lập trình viên import vào để sử dụng cho game của mình.



Hình 2.12. Các thành phần cơ bản: Project

Có thể tổ chức và quản lý assets bằng các folders, hoặc tìm kiếm theo asset type (models, materials, audio, prefabs, scripts,...)

- Packages

Một package là tập hợp các game objects, assets và các file meta-data liên quan.

Chúng là các object liên quan với nhau: models, scripts, materials, etc,...:

- Một tập các shaders để hỗ trợ hiệu ứng hình ảnh.
- Các prefabs hiệu ứng particle systems.
- Các models xe cho game đua xe.
- Các models cây và vật cản, ...

2.2.5 Các đặc điểm và tính năng trong Unity Editor

- Rendering

Unity cung cấp nhiều đường dẫn kết xuất (render) khác nhau để đáp ứng nhu cầu của nhiều dự án, bao gồm:

- **Forward Rendering:** Đây là đường dẫn kết xuất mặc định trong Unity, phù hợp cho các dự án có hiệu suất cao với đồ họa đơn giản.
- **Deferred Rendering:** Đường dẫn kết xuất này phù hợp cho các dự án có đồ họa phức tạp với nhiều nguồn sáng và hiệu ứng.

- **Scriptable Render Pipelines (SRPs)**: SRPs cung cấp cho bạn sự linh hoạt cao nhất để kiểm soát quy trình rendering, nhưng đòi hỏi kiến thức lập trình.

- **Lightning**

Unity cung cấp một hệ thống ánh sáng linh hoạt cho phép bạn tạo ra nhiều loại ánh sáng khác nhau, bao gồm:

- Ánh sáng điểm: Loại ánh sáng này phát ra từ một điểm duy nhất và chiếu sáng theo mọi hướng.
- Ánh sáng hướng: Loại ánh sáng này phát ra từ một hướng cụ thể và chiếu sáng theo một đường thẳng.
- Ánh sáng khu vực: Loại ánh sáng này phát ra từ một khu vực hình chữ nhật hoặc hình vuông và chiếu sáng theo mọi hướng trong khu vực đó.

Hỗ trợ nhiều loại bóng đổ khác nhau để tạo ra hình ảnh chân thực hơn, bao gồm:

- Bóng đổ cứng: Bóng đổ cứng có cạnh sắc nét và được xác định rõ ràng.
- Bóng đổ mềm: Bóng đổ mềm có cạnh mờ và hòa quyện vào môi trường xung quanh.
- Bóng đổ đổ bóng: Bóng đổ đổ bóng là bóng đổ được chiếu lên các bề mặt khác, tạo ra hiệu ứng chân thực hơn.

Hỗ trợ ánh sáng phản xạ, cho phép mô phỏng cách thức ánh sáng phản xạ từ các bề mặt khác nhau. Điều này có thể giúp tạo ra hình ảnh chân thực hơn và trải nghiệm chơi game nhập vai hơn.

Unity hỗ trợ ánh sáng tán xạ, cho phép mô phỏng cách thức ánh sáng tán xạ qua các môi trường như sương mù hoặc bụi. Điều này có thể giúp tạo ra bầu không khí và hiệu ứng chân thực hơn.

- **Physics**

- **Hệ thống Rigidbody**: Là nền tảng của hệ thống vật lý (physics) trong Unity. Nó cho phép bạn mô phỏng chuyển động của các đối tượng trong thế giới trò chơi, bao gồm lực, va chạm và trọng lực.
- Các loại **Collider**: Unity cung cấp nhiều loại Collider khác nhau để xác định hình dạng của các đối tượng vật lý, bao gồm:
 - Box Collider: Hình dạng hộp chữ nhật.
 - Sphere Collider: Hình dạng hình cầu.
 - Capsule Collider: Hình dạng hình trụ.

- Mesh Collider: Hình dạng lưới tùy chỉnh.
- **Trigger:** Cho phép phát hiện khi hai đối tượng vật lý va chạm nhau mà không ảnh hưởng đến chuyển động của chúng.
- Hệ thống **Raycasting:** Cho phép bắn tia từ một điểm trong thế giới trò chơi và phát hiện các đối tượng vật lý mà nó va chạm.
- Cài đặt vật lý **nâng cao:** Unity cung cấp nhiều cài đặt vật lý nâng cao cho phép kiểm soát chi tiết cách thức hệ thống vật lý hoạt động, bao gồm:
 - Khối lượng: Khối lượng của đối tượng vật lý.
 - Ma sát: Lượng ma sát giữa các đối tượng vật lý.
 - Độ đòn hồi: Mức độ đòn hồi của các đối tượng vật lý.
 - Bước thời gian: Khoảng thời gian giữa các lần mô phỏng vật lý.
- Audio và Sound Effects
- Hệ thống âm thanh tích hợp: Cho phép phát âm thanh và nhạc trong dự án của mình. Hệ thống này hỗ trợ nhiều định dạng âm thanh phổ biến, bao gồm MP3, WAV và OGG.
- Âm thanh 3D: Unity hỗ trợ âm thanh 3D, tạo ra trải nghiệm âm thanh chân thực hơn bằng cách mô phỏng vị trí của các nguồn âm thanh trong thế giới trò chơi.
- Hiệu ứng âm thanh: Unity cung cấp nhiều hiệu ứng âm thanh tích hợp sẵn để sử dụng trong dự án của mình, bao gồm tiếng ồn, tiếng vang và hiệu ứng Doppler.
- Hệ thống trộn âm: Cho phép kiểm soát âm lượng, độ cao và hiệu ứng của nhiều nguồn âm thanh cùng một lúc.
- Âm thanh theo sự kiện: Phát âm thanh dựa trên các sự kiện trong trò chơi, chẳng hạn như khi người chơi nhấp vào nút hoặc khi nhân vật di chuyển.
- Âm thanh môi trường: Tạo ra âm thanh môi trường để tạo bầu không khí cho cảnh của mình.
- Cài đặt âm thanh nâng cao: Unity cung cấp nhiều cài đặt âm thanh nâng cao cho phép bạn kiểm soát chi tiết cách thức hệ thống âm thanh hoạt động, bao gồm:
 - Tần số lấy mẫu: Tần số mà âm thanh được lấy mẫu.
 - Độ sâu bit: Số bit được sử dụng để lưu trữ mỗi mẫu âm thanh.
 - Bộ nhớ đệm âm thanh: Kích thước của bộ nhớ đệm âm thanh.
 - Khoảng cách âm thanh: Khoảng cách mà âm thanh có thể được nghe thấy.

- **Programming**

Cấu trúc của một đoạn mã bao gồm 3 thành phần chính như sau:

- Biến (variable) thường có chứa bất kì giá trị kiểu dạng số đặc thù hoặc kiểu kí tự.
- Hàm (function) thường được sử dụng để có thể thực ti những công việc thông thường có cùng 1 biến và các biểu thức toán học khác.

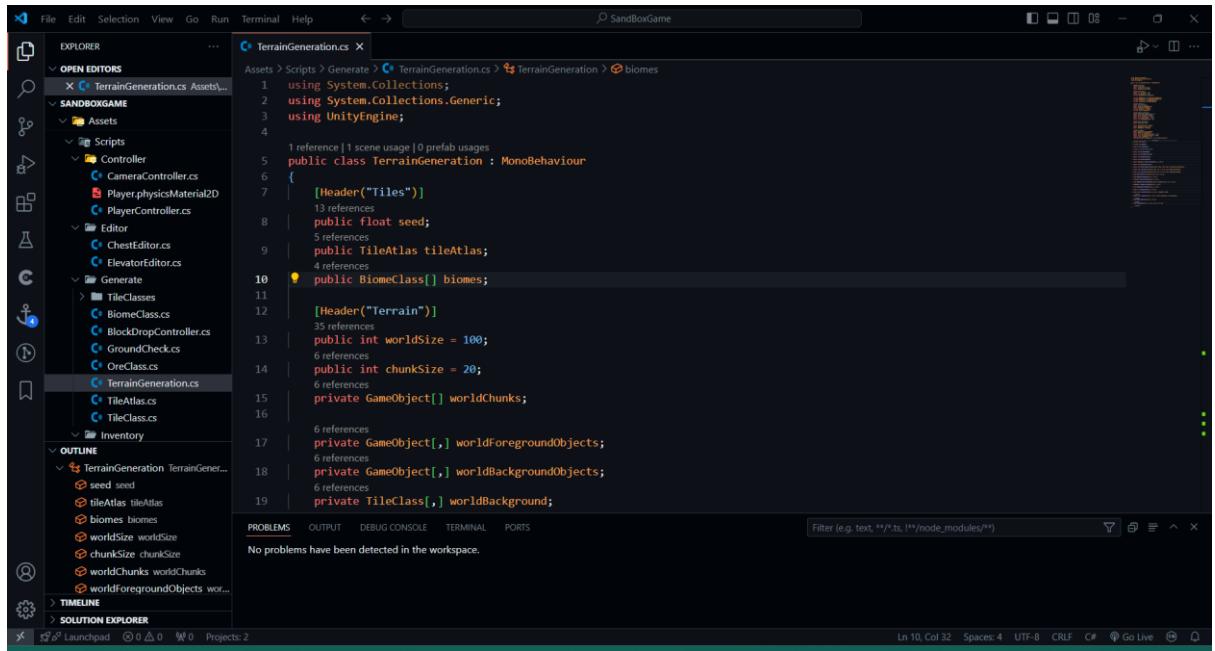
Những function bên trong Unity thường được phân chia thành 2 nhóm bao gồm: liên quan đến game – game relate (như system, input, network), đối tượng đặc trưng – object specific, người dùng định nghĩa – user defined.

Chú thích (comment) sẽ bị các chương trình dịch bỏ qua ngay khi chạy chương trình và nó cho phép người sử dụng có thể ghi chép lại những điều cần nhớ hoặc vô hiệu hóa một dòng mã lệnh nào đó.

2.3. Các công cụ sử dụng

2.3.1 Visual Studio Code

- Visual Studio Code (VS Code) là trình soạn thảo mã nguồn miễn phí và mã nguồn mở phổ biến, được nhiều lập trình viên Unity ưa chuộng. Nó cung cấp nhiều tính năng mạnh mẽ giúp bạn viết và gỡ lỗi mã C# cho Unity một cách hiệu quả hơn, từ đó tăng tốc độ phát triển game.
- Hỗ trợ Intellisense, giúp lập trình viên tự động hoàn thành mã, gợi ý các biến, hàm và phương thức phù hợp, cũng như kiểm tra lỗi cú pháp khi đang viết code. Nhờ vậy có thể viết code nhanh hơn và chính xác hơn.
- VS Code có hệ sinh thái mở rộng không lồ với hàng ngàn tiện ích mở rộng miễn phí và trả phí. Lập trình viên có thể cài đặt các tiện ích mở rộng để bổ sung thêm các tính năng cho VS Code, ví dụ như tự động hóa các tác vụ lập trình, quản lý dự án, so sánh mã code, v.v.
- Có thể tích hợp trực tiếp với Unity Editor, cho phép mở và chỉnh sửa các tập tin script C# trong Unity một cách dễ dàng. Nhờ vậy, lập trình viên có thể chuyển đổi qua lại giữa VS Code và Unity Editor một cách nhanh chóng và mượt mà.



Hình 2.13. Visual Studio

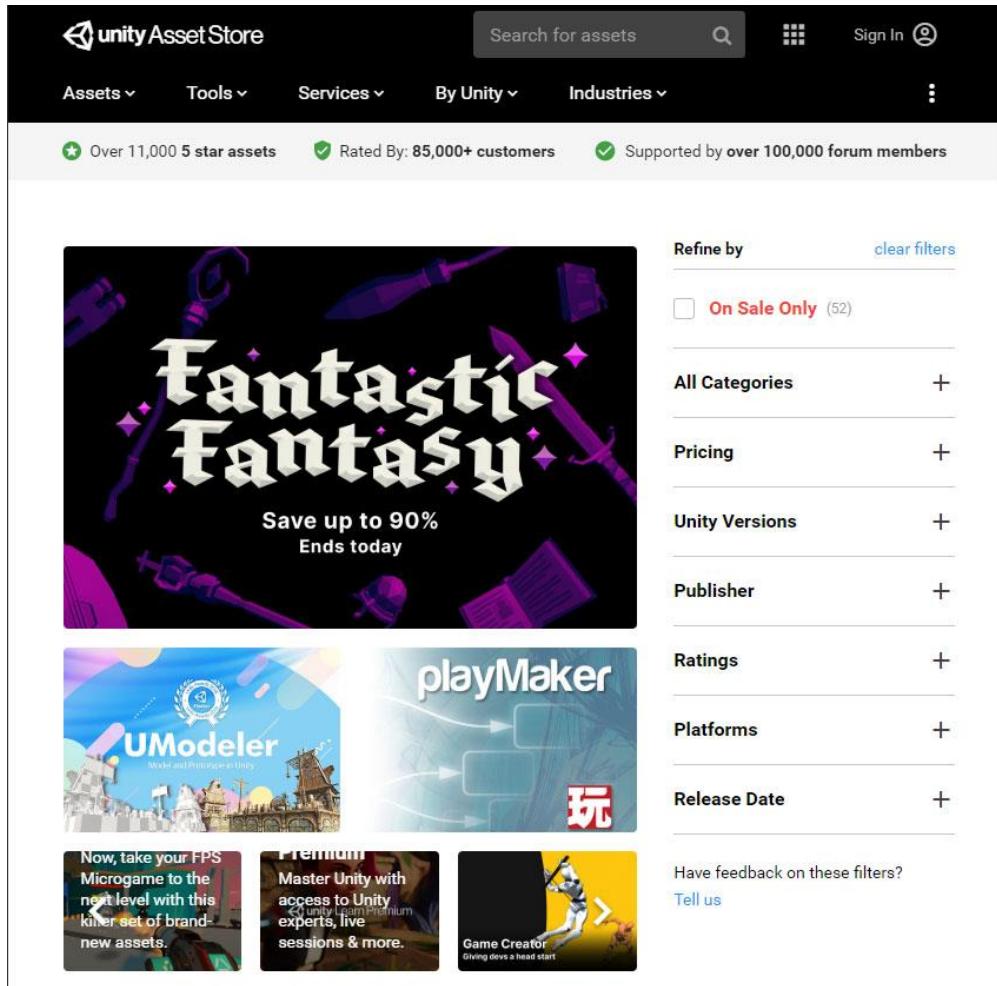
2.3.2 Unity Editor

Unity Editor là ứng dụng trung tâm sử dụng để xây dựng trò chơi điện tử hoặc các trải nghiệm tương tác khác trong nền tảng phát triển Unity. Nó cung cấp một bộ công cụ toàn diện để tạo ra các khía cạnh khác nhau của trò chơi.

2.3.3 Asset Store

Unity Asset Store là một kho lưu trữ trực tuyến cung cấp nhiều tài nguyên kỹ thuật số cho các nhà phát triển Unity, bao gồm:

- **Mô hình 3D:** Các mô hình 3D được sử dụng để tạo ra các nhân vật, vật thể, cảnh quan và các yếu tố khác trong môi trường game.
- **Kết cấu:** Kết cấu là những hình ảnh được áp dụng lên mô hình 3D để tạo ra chi tiết và độ chân thực cho bề mặt.
- **Âm thanh:** Âm thanh bao gồm hiệu ứng âm thanh, nhạc nền và lồng tiếng cho game.
- **Shader:** Shader là các chương trình nhỏ được sử dụng để tạo ra các hiệu ứng hình ảnh cho các vật thể trong game.
- **Công cụ:** Các công cụ hỗ trợ việc phát triển game như script, plugin và hệ thống AI.
- **Gói mẫu:** Gói mẫu bao gồm các tài nguyên và hướng dẫn để giúp bạn bắt đầu tạo game nhanh chóng.

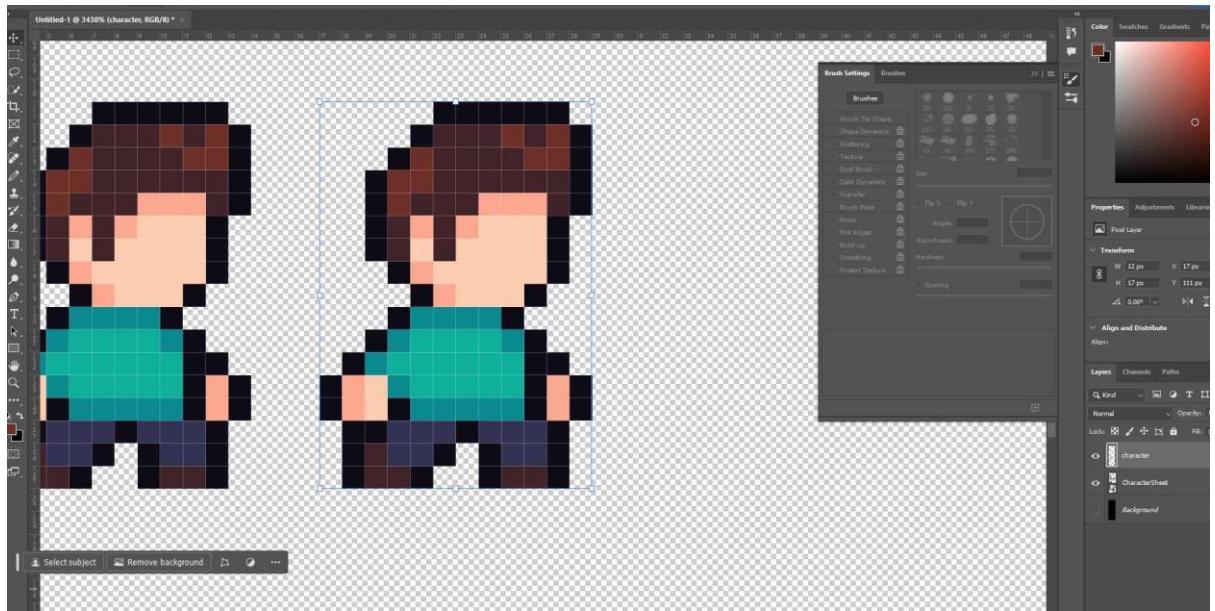


Hình 2.14. Unity Asset Store

2.3.4 Adobe Photoshop

Adobe Photoshop là một phần mềm chỉnh sửa ảnh và thiết kế đồ họa mạnh mẽ được sử dụng rộng rãi trong nhiều ngành công nghiệp, bao gồm cả phát triển game. Trong Unity, Photoshop có thể được sử dụng để tạo ra nhiều loại tài sản cho game, bao gồm:

- **Texture:** Texture là những hình ảnh được áp dụng lên bề mặt của các mô hình 3D để tạo chi tiết và màu sắc. Photoshop cung cấp nhiều công cụ để tạo và chỉnh sửa texture, bao gồm bộ lọc, cọ vẽ và công cụ chọn.
- **Sprite:** Sprite là những hình ảnh 2D được sử dụng để tạo ra các nhân vật, vật thể và các yếu tố giao diện người dùng (UI) trong game. Photoshop có thể được sử dụng để vẽ và chỉnh sửa sprite, cũng như tạo hiệu ứng động.



Hình 2.15. Thiết kế nhân vật bằng Adobe Photoshop

Tích hợp Photoshop với Unity:

- Xuất file: Xuất các tệp Photoshop sang định dạng được Unity hỗ trợ, chẳng hạn như PNG hoặc JPG. Sau đó, nhập các tệp này vào Unity và sử dụng chúng làm texture, sprite hoặc các tài sản khác.
- Plugin: Có một số plugin của bên thứ ba cho phép tích hợp Photoshop với Unity một cách liền mạch hơn. Các plugin này có thể cung cấp các tính năng như nhập và xuất texture, chỉnh sửa sprite trực tiếp trong Photoshop và xem trước các thay đổi của bạn trong Unity theo thời gian thực.

Lợi ích sử dụng Photoshop trong Unity:

- Chất lượng cao: Photoshop là một công cụ mạnh mẽ với nhiều tính năng giúp bạn tạo ra các tài sản chất lượng cao cho game của mình.
- Tính linh hoạt: Photoshop có thể được sử dụng để tạo ra nhiều loại tài sản khác nhau, từ texture và sprite đến concept art và GUI.
- Dễ sử dụng: Photoshop có giao diện người dùng trực quan và dễ sử dụng, ngay cả đối với những người mới bắt đầu.

2.4. Giới thiệu về ngôn ngữ lập trình C#

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

C# là ngôn ngữ kịch bản chính được sử dụng trong Unity để phát triển trò chơi. Nó là một ngôn ngữ lập trình hướng đối tượng mạnh mẽ, cung cấp nền tảng để tạo ra các hệ thống trò chơi tương tác và phức tạp. C# có một số ưu điểm khi sử dụng trong

Unity:

- Hỗ trợ đa nền tảng: C# cho phép tạo ra các trò chơi có thể chạy trên nhiều nền tảng khác nhau, bao gồm PC, Mac, Linux, thiết bị di động và bảng điều khiển trò chơi.
- Bảo mật: C# là ngôn ngữ lập trình an toàn và bảo mật, giúp bảo vệ trò chơi của bạn khỏi phần mềm độc hại và các mối đe dọa bảo mật khác.
- Khả năng mở rộng: C# là ngôn ngữ lập trình có thể mở rộng, có nghĩa là bạn có thể dễ dàng thêm các tính năng mới và chức năng vào trò chơi của mình.



Hình 2.16. C#

2.4.2 Cách sử dụng C# trong trò chơi

- Hành vi đơn sắc

Trong Unity, các tập lệnh thường được lấy từ lớp MonoBehaviour. MonoBehaviour là lớp cơ sở cho các tập lệnh tương tác với GameObject trong Unity. Nó cung cấp một loạt các phương thức và chức năng để xử lý các sự kiện và hành vi update theo thời gian.

- Unity API (Giao diện lập trình ứng dụng)

Unity hiển thị API rộng lớn cho phép bạn tương tác với các tính năng, thành phần và hệ thống của công cụ. API Unity cung cấp các lớp và phương thức để thao tác GameObject, truy cập thông tin đầu vào từ người chơi, xử lý các mô phỏng vật lý và hơn thế nữa. Bạn có thể sử dụng API để kiểm soát các khía cạnh khác nhau trong hoạt động và giao diện của trò chơi.

- Thứ tự thực thi tập lệnh

Unity thực thi các tập lệnh theo một thứ tự cụ thể được xác định bởi thuộc tính execution order của tập lệnh hoặc vị trí của tập lệnh trong danh sách thứ tự thực thi

tập lệnh. Hiểu thứ tự thực thi tập lệnh là rất quan trọng khi xử lý các phụ thuộc giữa các tập lệnh hoặc khi các hành vi cụ thể cần diễn ra theo một thứ tự cụ thể.

- Biến và kiểu dữ liệu

C# hỗ trợ nhiều kiểu dữ liệu khác nhau, bao gồm số nguyên, số dấu phẩy động, chuỗi, boolean, v.v. Bạn có thể khai báo variables bằng cách sử dụng các kiểu dữ liệu này để lưu trữ và thao tác các giá trị. Các biến có thể là cục bộ của một phương thức cụ thể hoặc các biến thành viên có thể truy cập được trong toàn lớp.

- Phương thức và chức năng

Các phương thức là các khối mã thực hiện các tác vụ cụ thể. Bạn có thể xác định các phương thức trong tập lệnh của mình để đóng gói chức năng và làm cho mã của bạn có tổ chức hơn và có thể sử dụng lại được. Unity cung cấp các phương thức được xác định trước như Awake, Start, Update và FixedUpdate mà bạn có thể ghi đè để thực hiện các hành động trong các giai đoạn cụ thể của trò chơi.

- Kiểm soát dòng chảy

C# hỗ trợ các cấu trúc luồng điều khiển như câu lệnh if-else, vòng lặp (for, while, do-while) và câu lệnh switch. Các cấu trúc này cho phép bạn đưa ra quyết định, lặp lại các bộ sưu tập và thực hiện các hành động khác nhau dựa trên các điều kiện cụ thể.

- Sự kiện và đại biểu

C# hỗ trợ các sự kiện và đại biểu, cho phép lập trình theo hướng sự kiện. Sự kiện cho phép bạn xác định và kích hoạt các sự kiện tùy chỉnh trong mã của mình, trong khi các đại biểu tạo điều kiện thuận lợi cho việc giao tiếp giữa các đối tượng và phương thức.

- Gỡ lỗi

Unity cung cấp các công cụ để gỡ lỗi tập lệnh C# của bạn. Bạn có thể sử dụng Debug.Log để in thông báo tới bảng điều khiển nhằm mục đích gỡ lỗi. Ngoài ra, trình gỡ lỗi tích hợp cho phép bạn đặt điểm dừng, kiểm tra các biến và duyệt qua mã của bạn để xác định và giải quyết vấn đề.

- Tuần tự hóa nội dung

Unity sử dụng hệ thống tuần tự hóa để lưu và tải nội dung, bao gồm cả tập lệnh C#. Khi tạo các lớp tùy chỉnh, bạn cần đánh dấu chúng bằng thuộc tính '[System.Serializable]' để đảm bảo giá trị của chúng được tuần tự hóa chính xác.

2.4.3 Ưu nhược điểm của C#

Ưu điểm của C#:

- **Dễ học và sử dụng:** C# có cú pháp rõ ràng và dễ hiểu, gần gũi với nhiều ngôn ngữ lập trình khác như C++, Java, giúp người mới học dễ dàng tiếp cận.
- **Hỗ trợ mạnh mẽ từ .NET Framework:** C# hoạt động trên .NET Framework, cung cấp nhiều thư viện và công cụ hỗ trợ phát triển ứng dụng mạnh mẽ, từ ứng dụng desktop, web đến dịch vụ web và ứng dụng di động.
- **Quản lý bộ nhớ tự động:** Với cơ chế garbage collection, C# giúp quản lý bộ nhớ một cách tự động, giảm bớt gánh nặng cho lập trình viên trong việc quản lý tài nguyên.
- **Hỗ trợ lập trình hướng đối tượng (OOP):** C# hỗ trợ đầy đủ các tính năng của lập trình hướng đối tượng, như kế thừa, đa hình, và đóng gói, giúp cấu trúc mã nguồn rõ ràng và dễ bảo trì.
- **Đa nền tảng:** Với .NET Core và .NET 5/6, C# có thể chạy trên nhiều nền tảng khác nhau như Windows, macOS, và Linux.
- **Bảo mật:** C# tích hợp nhiều tính năng bảo mật, giúp bảo vệ ứng dụng khỏi các lỗ hổng và tấn công từ bên ngoài.
- **Hỗ trợ lập trình đồng bộ và bất đồng bộ:** Với các từ khóa async và await, C# cung cấp cách tiếp cận dễ dàng và mạnh mẽ cho lập trình bất đồng bộ, tối ưu hóa hiệu suất của ứng dụng.

Nhược điểm của C#:

- **Hiệu suất:** Mặc dù C# có hiệu suất khá tốt, nhưng so với các ngôn ngữ lập trình thấp cấp như C hay C++, C# thường chậm hơn do phụ thuộc vào .NET runtime.
- **Sự phụ thuộc vào hệ sinh thái Microsoft:** Mặc dù đã có nhiều cải tiến trong việc hỗ trợ đa nền tảng, nhưng C# vẫn chủ yếu phát triển mạnh mẽ trong hệ sinh thái của Microsoft, điều này có thể gây khó khăn cho những ai không sử dụng công nghệ của Microsoft.
- **Độ phức tạp của hệ sinh thái:** .NET Framework và các công nghệ liên quan khá phức tạp và có thể gây khó khăn cho những người mới bắt đầu hoặc không quen thuộc với hệ sinh thái Microsoft.

- **Kích thước ứng dụng:** Các ứng dụng viết bằng C# thường có kích thước lớn hơn so với các ngôn ngữ lập trình khác do sự phụ thuộc vào .NET runtime và các thư viện kèm theo.
- **Cập nhật liên tục:** Microsoft thường xuyên cập nhật C# và .NET, đòi hỏi lập trình viên phải liên tục học hỏi và cập nhật kiến thức để theo kịp các thay đổi và cải tiến mới.

2.5 Kỹ thuật lập trình hướng đối tượng trong C#

2.5.1 Tổng quan về lập trình hướng đối tượng

Lập trình hướng đối tượng (OOP) là một phương pháp lập trình sử dụng các "đối tượng" để mô hình hóa các thực thể trong thế giới thực. Mỗi đối tượng có trạng thái (được biểu thị bằng các thuộc tính) và hành vi (được biểu thị bằng các phương thức). OOP cung cấp nhiều lợi ích cho việc phát triển phần mềm, bao gồm:

- **Khả năng bảo trì:** Mã OOP dễ bảo trì hơn vì nó được tổ chức thành các module riêng biệt (đối tượng).
- **Khả năng tái sử dụng:** Mã OOP có thể được tái sử dụng dễ dàng hơn vì các đối tượng có thể được sử dụng lại trong các dự án khác nhau.
- **Khả năng mở rộng:** Mã OOP dễ mở rộng hơn vì các đối tượng mới có thể được dễ dàng thêm vào hệ thống hiện có.

C# hỗ trợ đầy đủ các nguyên tắc của lập trình hướng đối tượng, bao gồm:

- **Đóng gói:** Đóng gói là việc nhóm các thuộc tính và phương thức liên quan vào một đơn vị (đối tượng). Điều này giúp che giấu các chi tiết triển khai bên trong của đối tượng và chỉ cho phép truy cập vào các thành viên công khai của nó.
- **Trùu tượng:** Trùu tượng là việc tập trung vào các đặc điểm thiết yếu của một đối tượng và bỏ qua các chi tiết không quan trọng. Điều này giúp đơn giản hóa mã và dễ dàng hiểu hơn.
- **Kế thừa:** Kế thừa là khả năng của một đối tượng để thừa hưởng các thuộc tính và phương thức từ một đối tượng khác (đối tượng cha). Điều này giúp thúc đẩy sự tái sử dụng mã và cho phép tạo ra các lớp phân cấp.
- **Đa hình:** Đa hình là khả năng của một đối tượng phản hồi các tin nhắn khác nhau theo những cách khác nhau. Điều này giúp cho mã linh hoạt hơn và dễ dàng mở rộng hơn.

2.5.2 Một số kỹ thuật lập trình hướng đối tượng

Dưới đây là một số kỹ thuật lập trình hướng đối tượng phổ biến trong C#:

- **Lớp:** Lớp là bản mẫu để tạo ra các đối tượng. Nó xác định các thuộc tính và phương thức của đối tượng.
- **Đối tượng:** Đối tượng là một thể hiện cụ thể của một lớp. Nó có trạng thái riêng và có thể tương tác với các đối tượng khác.
- **Phương thức:** Phương thức là một hành động mà một đối tượng có thể thực hiện. Nó được xác định trong lớp và có thể truy cập được từ các đối tượng khác.
- **Thuộc tính:** Thuộc tính là một thuộc tính của một đối tượng. Nó được xác định trong lớp và lưu trữ trạng thái của đối tượng.
- **Kế thừa:** Kế thừa cho phép một lớp thừa hưởng các thuộc tính và phương thức từ một lớp khác. Lớp con có thể ghi đè các phương thức được thừa kế để cung cấp hành vi cụ thể hơn.
- **Giao diện:** Giao diện là một tập hợp các phương thức mà một lớp phải triển khai. Nó thúc đẩy lập trình hướng đối tượng và cho phép các lớp khác nhau tương tác với nhau một cách nhất quán.
- **Đa hình:** Đa hình cho phép cùng một phương thức được gọi trên các đối tượng khác nhau và tạo ra hành vi khác nhau cho mỗi đối tượng. Điều này giúp cho mã linh hoạt hơn và dễ dàng mở rộng hơn.

CHƯƠNG III: PHÂN TÍCH THIẾT KẾ GAME

3.1 Load hoạt họa

3.1.1 Vấn đề

Game 2D được xây dựng từ nhiều mô hình 2D được đặt lên không gian 2 chiều sao cho hài hoà với nhau để tạo thành cảnh vật trong game. Do đó việc nạp và hiển thị được mô hình 2D trong game là vô cùng quan trọng.

Mô hình 2D được cấu tạo từ rất nhiều đa giác để tạo nên khối vật thể. Ngày nay, trong một mô hình 2D không chỉ đơn thuần chứa một khối vật thể mà nó bao gồm nhiều khối vật thể được gắn kết với nhau trên một khung xương. Điều này giúp cho mô hình không bị gãy chết một chuyển động vào bên trong và dễ dàng thay đổi chuyển động cho mô hình

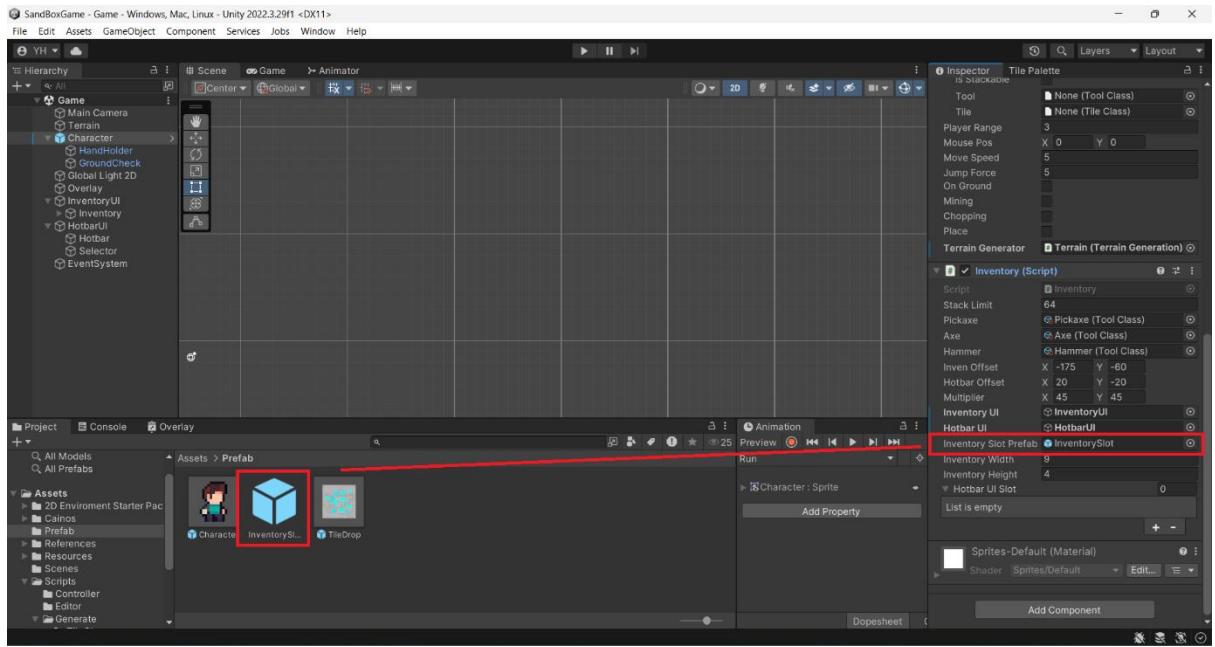
3.1.2 Giải pháp

Các mô hình 2D, 3D thông thường được thiết kế sẵn bằng các phần mềm thiết kế 3D chuyên dụng như Adobe Photoshop, Adobe InDesign, CorelDRAW,... sau đó mô hình sẽ được đưa vào Game Engine để sử dụng

Engine Unity hỗ trợ rất nhiều định dạng mô hình 2D, 3D khác nhau như: PNG, JPG... khi mô hình được load vào project ta sẽ chuyển mô hình sang mục Prefab để có thể tái sử dụng nhiều lần

Unity có hỗ trợ load mô hình bằng cách kéo thả Prefab vào vị trí bất kỳ trong Scene. Tuy nhiên để linh hoạt hơn thì ta có thể xử lý bằng code

Trước tiên, ta tạo ra một file script và gắn nó vào một đối tượng trong game bất kì để đoạn script có thể thực thi. Trong file script này, ta khai báo một đối tượng kiểu GameObject để lưu mô hình và dùng hàm Instantiate() để khởi tạo mô hình này ở vị trí góc quay mong muốn. Trên cửa sổ Inspector của đối tượng game được gắn script vào xuất hiện thuộc tính Obj. Ta chọn Prefab mong muốn và kéo thả vào thuộc tính Obj



Hình 3.17. Prefab ô vật phẩm có thể tái sử dụng

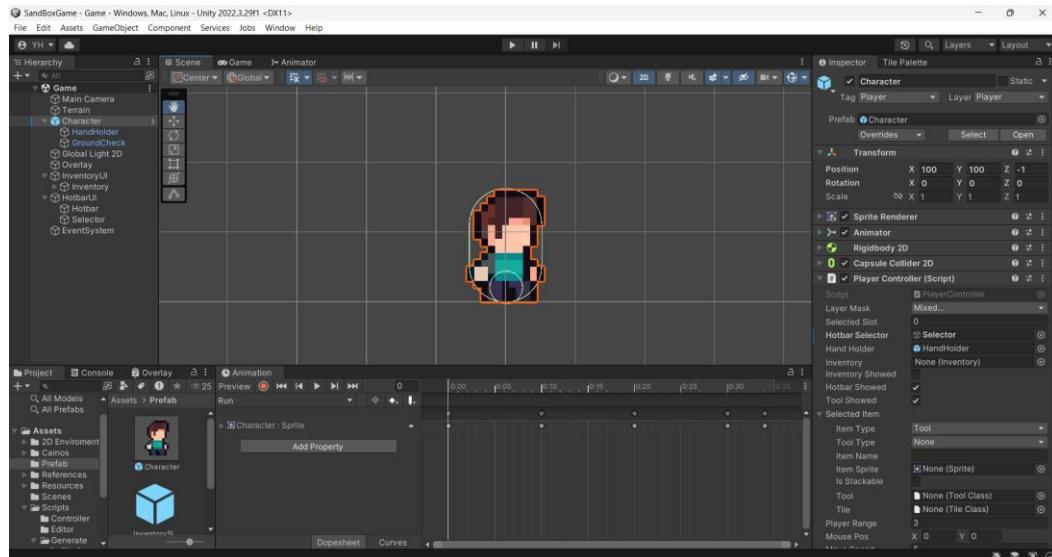
3.2 Chuyển động mô hình nhân vật 2D

3.2.1 Vấn đề

Chúng ta đã load được mô hình 2D vào trong game, vậy làm sao để mô hình 2D này có thể chuyển động trong Game.

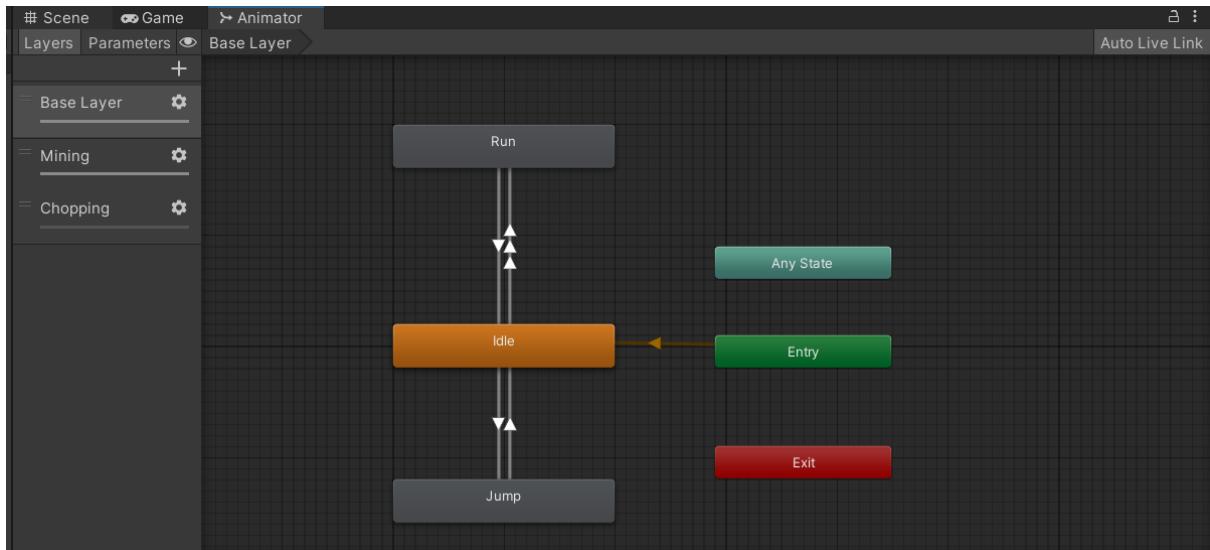
3.2.2 Giải pháp

Trước tiên mô hình 2D cần phải có sẵn animation bên trong. Khi import mô hình vào Unity, ta có thể tạo **Animation** cho từng hành động như Idle (đứng tại chỗ), Run (chạy), Jump (nhảy).



Hình 3.18 Tạo Animation cho nhân vật

Sau đó dùng **Animator** để quản lý chuyển động, nó hoạt động bằng cách kết hợp các animation clip (đoạn phim hoạt hình) với các trạng thái (state) và chuyển tiếp (transition) để tạo ra các hành vi phức tạp.



Hình 3.19. Quản lý chuyển động bằng Animator

Bắt phím và quy định nơi sử dụng animation bằng code

```

private void Update()
{
    horizontal = Input.GetAxis("Horizontal");
    mining = Input.GetMouseButton(0);
    chopping = Input.GetMouseButton(0);
    place = Input.GetMouseDown(1);

    if (selectedItem != null && selectedItem.itemType == ItemClass.ItemType.tool)
    {
        animator.SetBool("Mining", mining);
    }
}

```

Hình 3.20. Bắt phím và quy định cho Animation

3.3 Xây dựng giao diện Game

3.3.1 Vấn đề

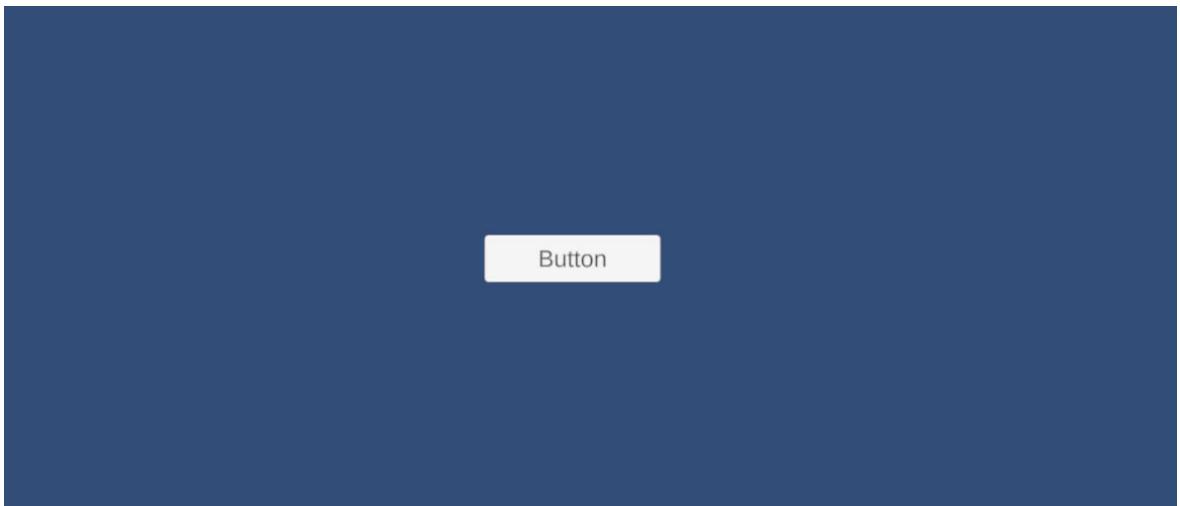
Giao diện đồ họa người dùng là một phần quan trọng không thể thiếu trong khi xây dựng một ứng dụng game hay bất cứ một ứng dụng nào để vẽ các đối tượng đồ họa như Button, Label, Checkbox, Slider,... lên màn hình

3.3.2 Giải pháp

Để làm được điều này ta có thể dùng UI Button để chuyển cảnh giữa các Scene với nhau và các Scene được quản lý bằng hệ thống Scene Manager. Hệ thống UI liên

kết với Scene bằng code để chuyển cảnh. Hệ thống UI trong Unity đóng vai trò quan trọng trong việc tạo ra giao diện người dùng (UI) cho trò chơi hoặc ứng dụng. Nó cung cấp một bộ công cụ và tính năng mạnh mẽ giúp thiết kế giao diện,

Để sử dụng được các tính năng trong UI ta phải tạo trực tiếp từ màn hình Inspector. Ví dụ sau đây tạo ra một Button đơn giản

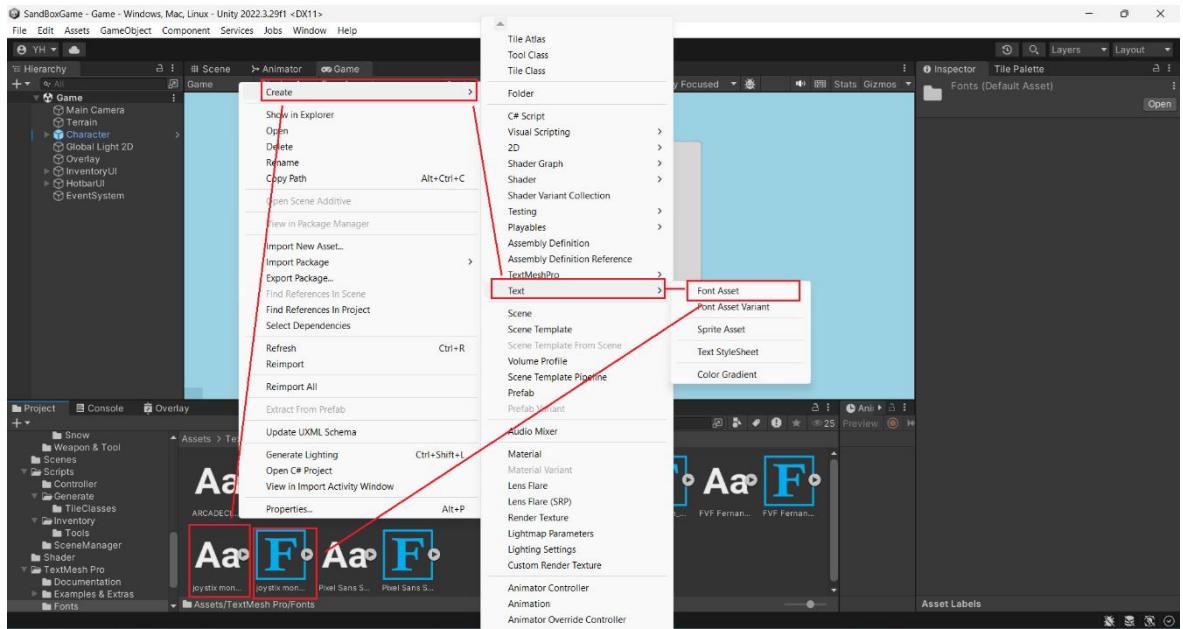


Hình 3.21. UI Button

Áp dụng vào giao diện của Game, ta dùng hàng loạt các UI để bắt đầu tạo. Đầu tiên dùng Canvas một để chứa giao diện Game và một để chứa màn hình tải Game

Ở Canvas giao diện Game, ta dùng UI Text để thiết kế tên trò chơi, dùng UI Button để chuyển cảnh và giữa quá trình chuyển cảnh có thể thấy được quy trình tải địa hình diễn ra bằng UI Slider

Vì bộ Fonts có sẵn của Unity có hạn nên Unity có công cụ hỗ trợ chuyển đổi Font ngoài thành Font Asset sử dụng trong Unity

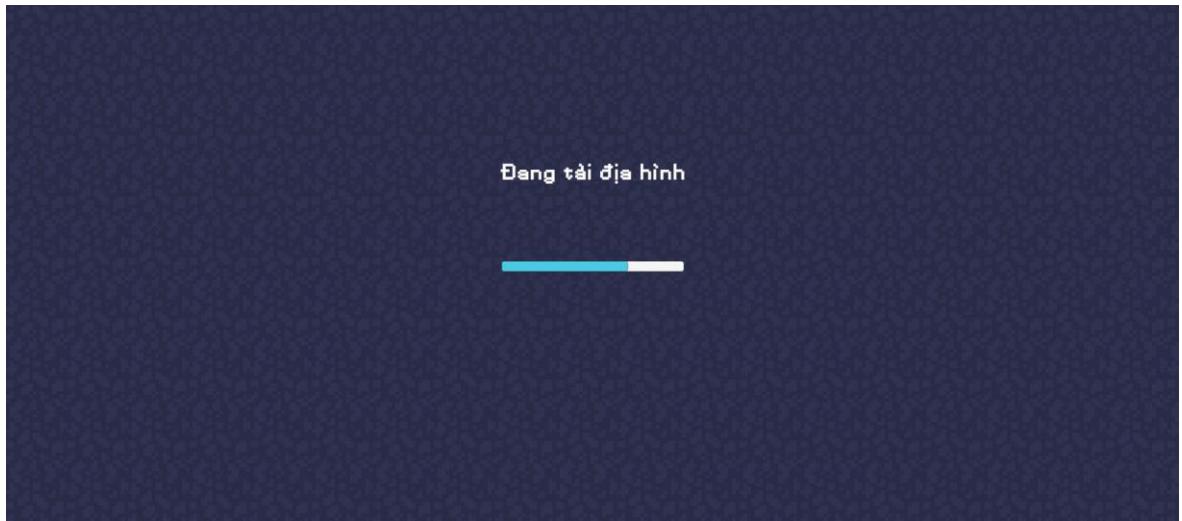


Hình 3.22. Chuyển đổi Font thành Font Asset



Hình 3.23. Giao diện bắt đầu Game

Ở Canvas giao diện tải địa hình, thanh Slider được gán cho trùng với tốc độ tải địa hình trong Game, khi đó người chơi sẽ thấy được tiến độ tải thông qua UI Slider.



Hình 3.24. Canvas chứa giao diện tải Game

Phương thức để gán giao diện tải địa hình vào giữa 2 Scene được thực hiện như sau:

```
0 references
public void SceneLoader(int scene)
{
    StartCoroutine(LoadScene_Coroutine(scene));
}

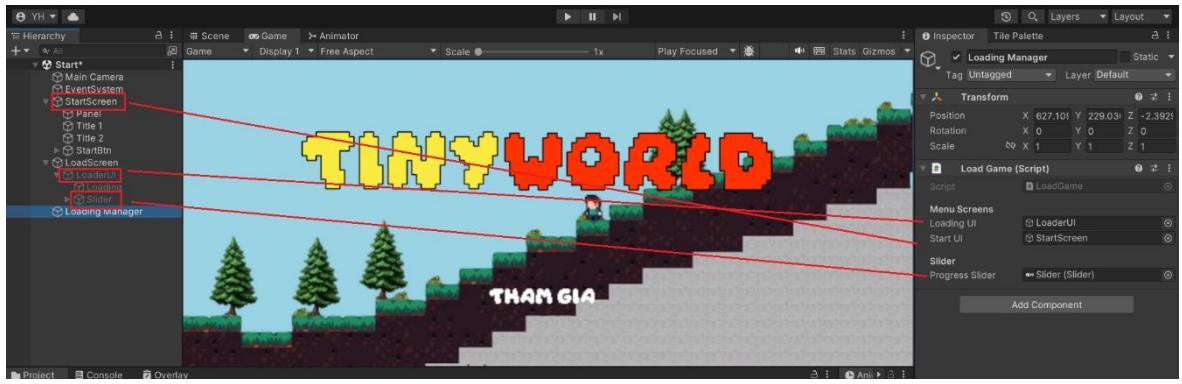
1 reference
public IEnumerator LoadScene_Coroutine(int scene)
{
    AsyncOperation asyncOperation = SceneManager.LoadSceneAsync(scene);
    loadingUI.SetActive(true);

    while (!asyncOperation.isDone)
    {
        float progress = Mathf.Clamp01(asyncOperation.progress / 0.9f);
        progressSlider.value = progress;

        yield return null;
    }
}
```

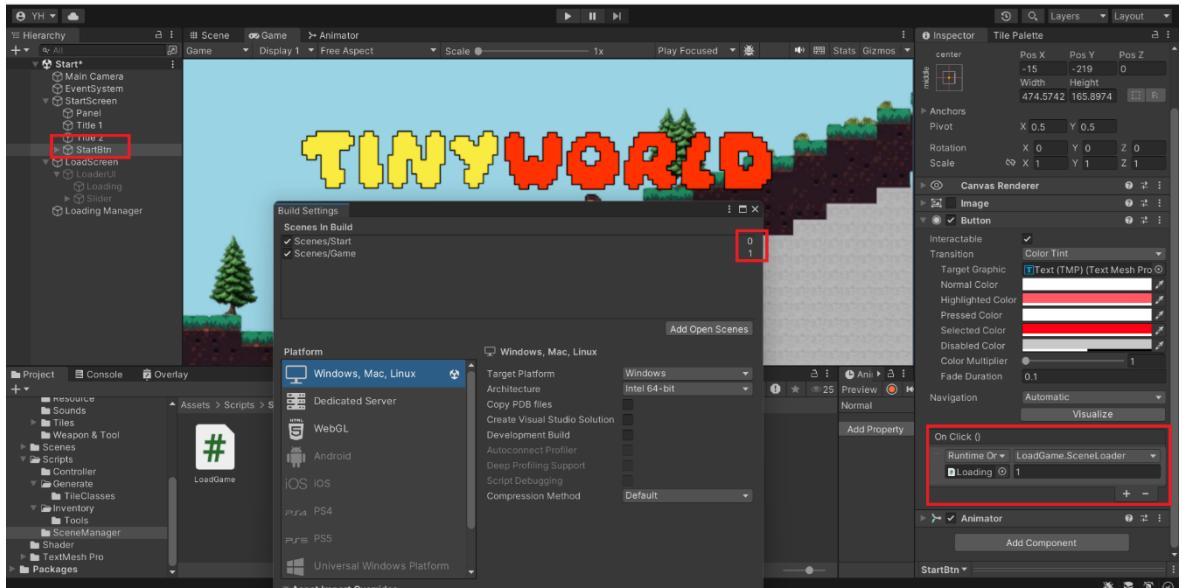
Hình 3.25. Phương thức hiện quá trình tải địa hình

Như vậy khi nhấn “Tham gia”, giao diện tải sẽ hiện lên và hiện tiến trình, và để UI Button nhận biết được cần tải Scene nào, ta thiết lập một GameObject mới rỗng để gán các GameObject được khởi tạo từ Script vào



Hình 3.26. Gán các GameObject vào Script

Cuối cùng ở Button “Tham gia”, vì ta đã cài đặt trong Script rằng Button sẽ tải Scene ở vị trí thứ n (người dùng truyền vào) trong Build Settings vì vậy ta cần kiểm tra kĩ đã thêm đủ Scene vào Build Settings chưa và tiến hành gán GameObject và giá trị vào Button



Hình 3.27. Gán Scene cho Button

3.4 Âm thanh trong Game

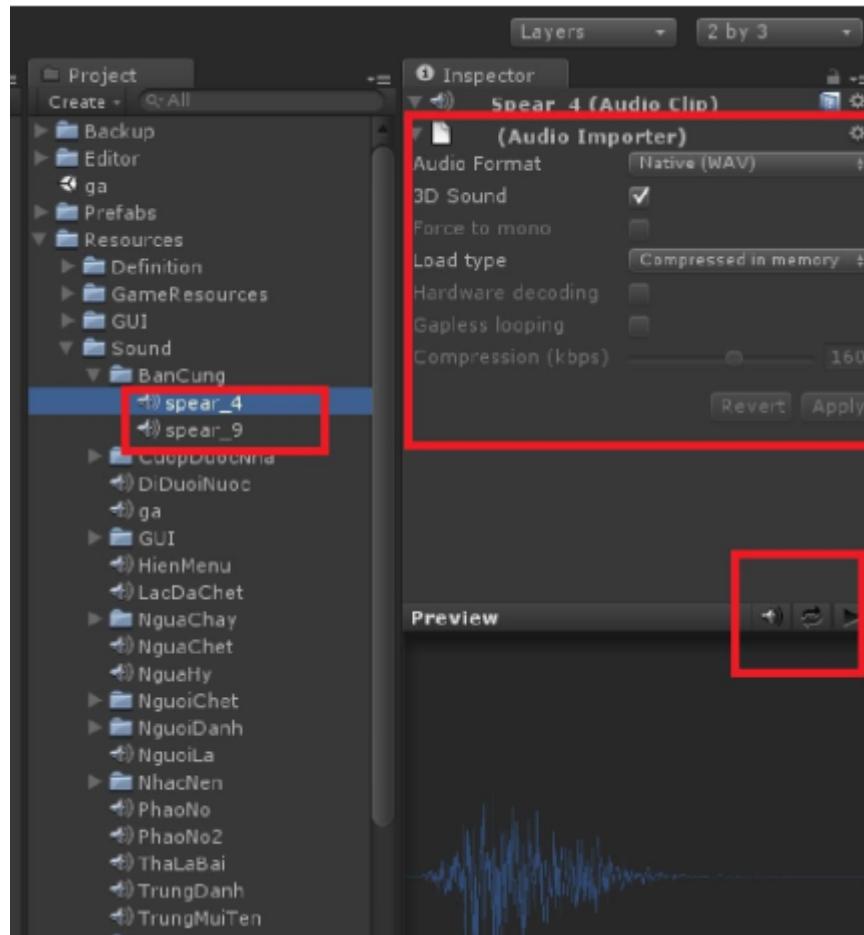
3.4.1 Vấn đề

Âm thanh là yếu tố không kém phần quan trọng trong ứng dụng game. Thật nhảm chán khi trải nghiệm game nhưng không có nhạc nền hay hiệu ứng âm thanh khi tương tác với môi trường

3.4.2 Giải pháp

Để thêm được một file âm thanh trong Unity có 2 cách: bằng code hoặc trên giao diện. Dù chọn cách nào thì trước hết chúng ta phải có sẵn các file âm thanh và

import vào project. Sau khi import âm thanh vào project, nếu file hợp lệ ta sẽ thấy như hình sau và có thể nhấn nút play để nghe thử



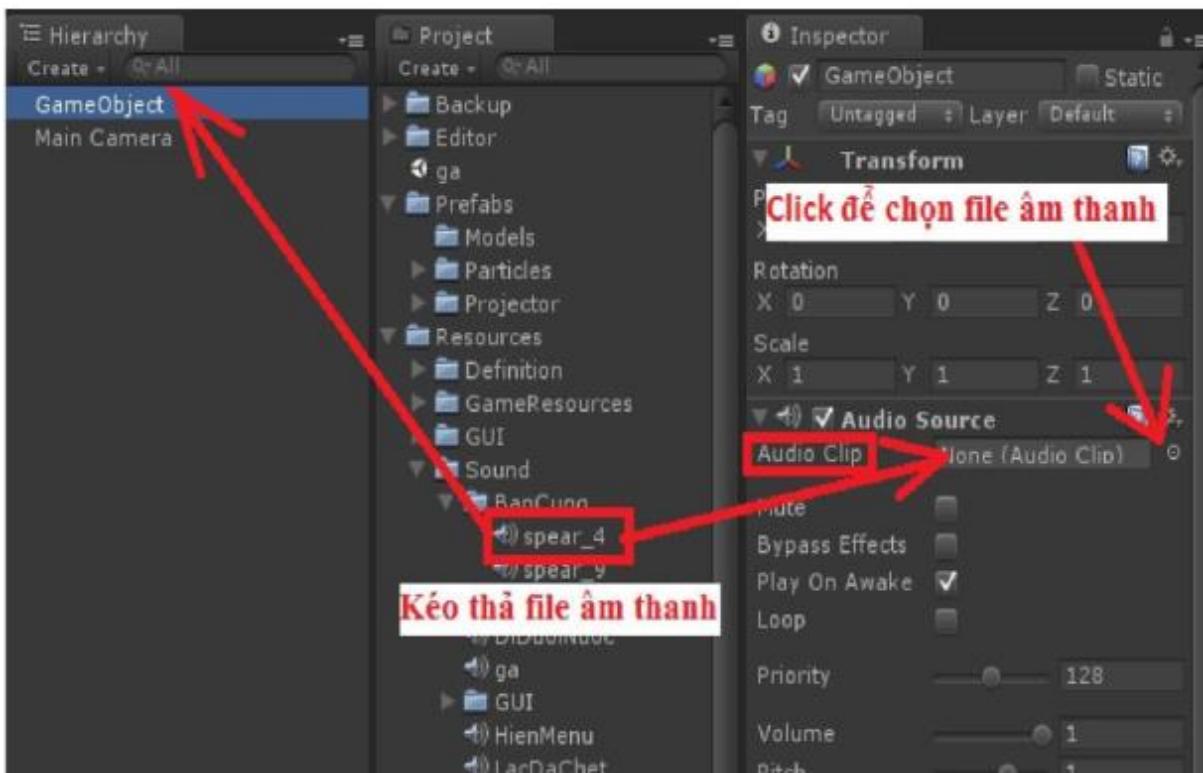
Hình 3.28. Giao diện File âm thanh hợp lệ

- Cách 1: Tạo trên giao diện

Trên menu của Unity, vào GameObject → Create Empty

Chọn đối tượng vừa tạo và gắn thành phần “Audio Source” cho đối tượng này. AudioSource là một đối tượng âm thanh. Muốn Play hay Stop, thay đổi cách lặp, tăng giảm volume nhạc thì phải thông qua đối tượng này

Sau khi gắn thành phần âm thanh cho đối tượng vừa tạo, chúng ta dễ dàng chỉnh sửa các thông số và gán file âm thanh cho thành phần AudioSource này.



Hình 3.29. Kéo thả File âm thanh trên giao diện

- Cách 2: Gán bằng code

Để linh hoạt hơn trong việc áp dụng âm thanh vào vị trí ta cần, có thể bật nhạc bằng code. Ta khởi tạo đối tượng game âm thanh bằng cách gán đường dẫn file nhạc cho âm thanh. Sau đó chỉ cần gọi các phương thức Play() để chạy nhạc.

3.5 Lỗi và Debugging

3.5.1 Vấn đề

Trong quá trình phát triển, sẽ có những vấn đề trong code khiến cho chương trình không hoạt động như mong muốn hoặc hoàn toàn không hoạt động.

Các loại lỗi phổ biến trong Unity:

- **Lỗi cú pháp (Syntax errors):** Đây là những lỗi xảy ra do sai chính tả trong code, hoặc do vi phạm các quy tắc cú pháp của ngôn ngữ lập trình.
- **Lỗi logic (Logical errors):** Đây là những lỗi xảy ra do logic trong code sai, khiến cho chương trình không hoạt động như mong muốn.
- **Lỗi runtime (Runtime errors):** Đây là những lỗi xảy ra khi chương trình đang chạy, thường do truy cập vào bộ nhớ không hợp lệ hoặc do thực hiện các thao tác không được phép.

3.5.2 Giải pháp

Debugging là một kỹ năng quan trọng đối với bất kỳ lập trình viên Unity nào. Bằng cách debug hiệu quả có thể tiết kiệm thời gian và cải thiện chất lượng code của mình.

Quy trình Debugging:

- **Xác định lỗi:** Bước đầu tiên là xác định lỗi xảy ra ở đâu và biểu hiện của nó như thế nào. Có thể xem lỗi từ **màn hình Console**
- **Tái tạo lỗi:** Tiếp theo cần tái tạo lỗi để có thể debug nó. Điều này có nghĩa là cần thực hiện các bước cần thiết để khiến lỗi xảy ra một lần nữa. Chẳng hạn như sử dụng hàm **Debug.Log** giúp theo dõi từng bước thực thi chương trình, xuất log ở mỗi bước chương trình chạy và kiểm tra giá trị của biến
- **Sửa lỗi:** Sau khi xác định được vị trí xảy ra lỗi thì sửa lỗi trong code.
- **Kiểm tra lại:** Sau khi sửa lỗi, kiểm tra lại chương trình để đảm bảo rằng lỗi đã được sửa.

CHƯƠNG IV: TRIỂN KHAI VÀ THỰC HIỆN

4.1 Triển khai thế giới game:

4.1.1 Giới thiệu hàm PerlinNoise trong khởi tạo địa hình

Hàm Perlin Noise là một kỹ thuật toán học được sử dụng để tạo ra nhiễu ngẫu nhiên có cấu trúc trong Unity. Nó thường được sử dụng để tạo ra các cảnh quan tự nhiên như núi, mây, và nước.

Hàm Perlin Noise hoạt động bằng cách tính toán giá trị nhiễu cho mỗi điểm trong không gian 2D và 3D. Giá trị nhiễu được dựa trên vị trí của điểm và một số tham số ngẫu nhiên. Các tham số này có thể được điều chỉnh để thay đổi độ nhấp nhô, tần suất và chi tiết của nhiễu.

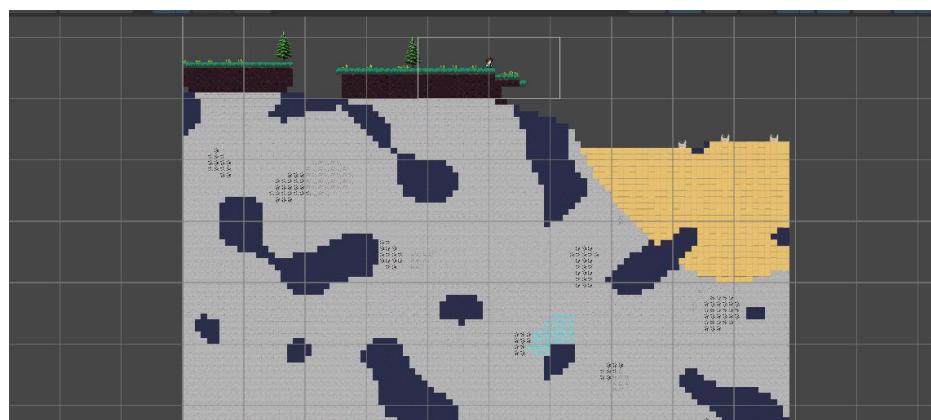
Giá trị nhiễu cao càng cao thì độ nhấp nhô cao hơn, tần suất tạo ra nhiễu thấp hơn tức là khoảng cách các thay đổi nhiễu lớn hơn và ít chi tiết hơn. Ngược lại giá trị nhiễu càng thấp thì độ nhấp nhô thấp hơn nhưng liên tục và gần nhau hơn, tạo ra nhiều chi tiết, các chi tiết nhỏ cũng được hiển thị rõ ràng hơn.

4.1.2 Khởi tạo thế giới Game

Địa hình trò chơi được xây dựng theo dạng màn hình ngang rộng lớn bao gồm nhiều hệ sinh thái khác nhau, với các đồi núi và hang động dưới lòng đất được khởi tạo một cách ngẫu nhiên không cố định mỗi khi khởi tạo game.

- Địa hình

Được cấu tạo bởi lớp đất và mặt cỏ nhấp nhô như đồi núi ở trên mặt đất cùng các tài nguyên thiên nhiên như cây thông, cỏ dại. Dưới lòng đất được bao phủ bởi đá và hang động, rải rác là các loại quặng từ dễ tìm đến rất hiếm. Dưới đây là ví dụ cho một thế giới khởi tạo 100x100



Hình 4.30. Khởi tạo địa hình 100x100

Hàm PerlinNoise đã được áp dụng để tạo độ nhiễu cho hang động, hệ sinh thái, tạo các tầng độ cao ngẫu nhiên trên mặt đất tạo thành đồi núi. Ứng dụng này đã giúp các đồi núi và hang động được đặt một cách tự nhiên, không thể đoán trước được.

Để phân biệt nơi sẽ đặt block, nơi nào sẽ là hang động, dùng độ nhiễu làm thước đo và đánh dấu bằng màu sắc. Sau đó ta chỉ cần gán block chỉ định ở những nơi có màu sắc đó.

```
public void GenerateNoiseTexture(float freq, float limit, Texture2D noiseTexture)
{
    float v;
    for (int x = 0; x < noiseTexture.width; x++)
    {
        for (int y = 0; y < noiseTexture.height; y++)
        {
            v = Mathf.PerlinNoise((x + seed) * freq, (y + seed) * freq);

            if (v > limit)
            {
                noiseTexture.SetPixel(x, y, Color.white);
            }
            else
            {
                noiseTexture.SetPixel(x, y, Color.black);
            }
        }
    }
    noiseTexture.Apply();
}
```

Hình 4.31. Tạo địa hình bằng PerlinNoise

- Hang động

Như đã giải thích cách hoạt động của thuật toán tạo địa hình ở trên, hang động được tạo ra một cách tự động với độ nhiễu thấp giúp hang động được khởi tạo một cách tự nhiên nhất. Ở mỗi hệ sinh thái, lớp tường hang động của nó là khác nhau và số lượng hang động cũng khác nhau. Người chơi có thể dùng công cụ thích hợp để phá tường của hang để lấy ánh sáng hoặc dùng như một vật phẩm trang trí.

4.1.3 Khởi tạo hệ sinh thái

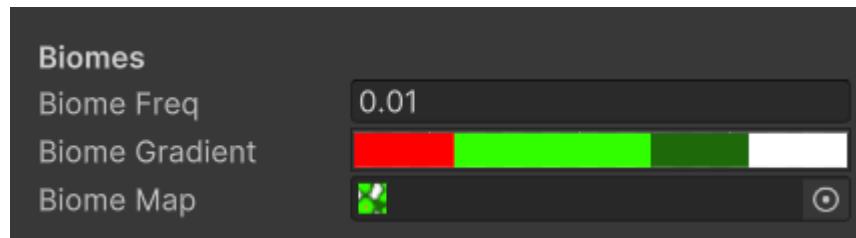
Hệ sinh thái (biomes) đóng vai trò quan trọng trong một tựa game Sandbox, tạo nên sự đa dạng và phong phú cho thế giới ảo mà người chơi khám phá. Mỗi hệ sinh thái sở hữu những đặc điểm riêng biệt về địa hình, khí hậu, thảm thực vật và động vật, mang đến những trải nghiệm độc đáo cho người chơi.

Có tất cả 4 hệ sinh thái trong thế giới, được khởi tạo ở các vị trí ngẫu nhiên bằng hàm PerlinNoise. Để phân biệt các hệ sinh thái với nhau, định nghĩa các hệ sinh thái bằng dải màu RGBA, mỗi màu được định nghĩa sẽ tương ứng với màu của hệ

sinh thái đó. Như vậy ta có thể dễ dàng điều chỉnh các thuộc tính như thông số độ nhiễu, độ cao địa hình, đồ họa khác nhau dễ dàng cho mỗi hệ sinh thái

```
1 reference
public void drawBiomeMap()
{
    float b;
    Color col;
    biomeMap = new Texture2D(worldSize, worldSize);
    for (int x = 0; x < biomeMap.width; x++)
    {
        for (int y = 0; y < biomeMap.height; y++)
        {
            b = Mathf.PerlinNoise((x + seed) * biomeFreq, (y + seed) * biomeFreq);
            col = biomeGradient.Evaluate(b);
            biomeMap.SetPixel(x, y, col);
        }
    }
    (field) Texture2D TerrainGeneration.biomeMap
    biomeMap.Apply();
}
```

Hình 4.32. Định nghĩa hệ sinh thái bằng màu sắc



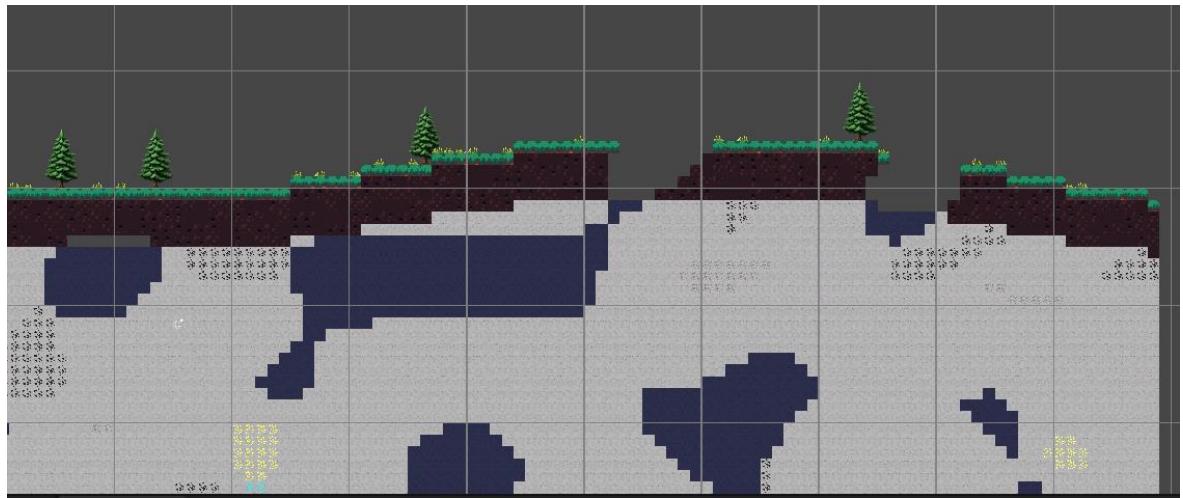
Hình 4.33. Dải màu hệ sinh thái và độ nhiễu

Các thuộc tính có thể được cài đặt riêng cho mỗi hệ sinh thái:

- Tên
- Màu giúp phân biệt hệ sinh thái
- Block
- Độ nhiễu hang động/nhiễu địa hình
- Cho phép tạo hang động hay không
- Độ cao lớp đất, địa hình trên mặt đất
- Vật phẩm, tài nguyên, quặng

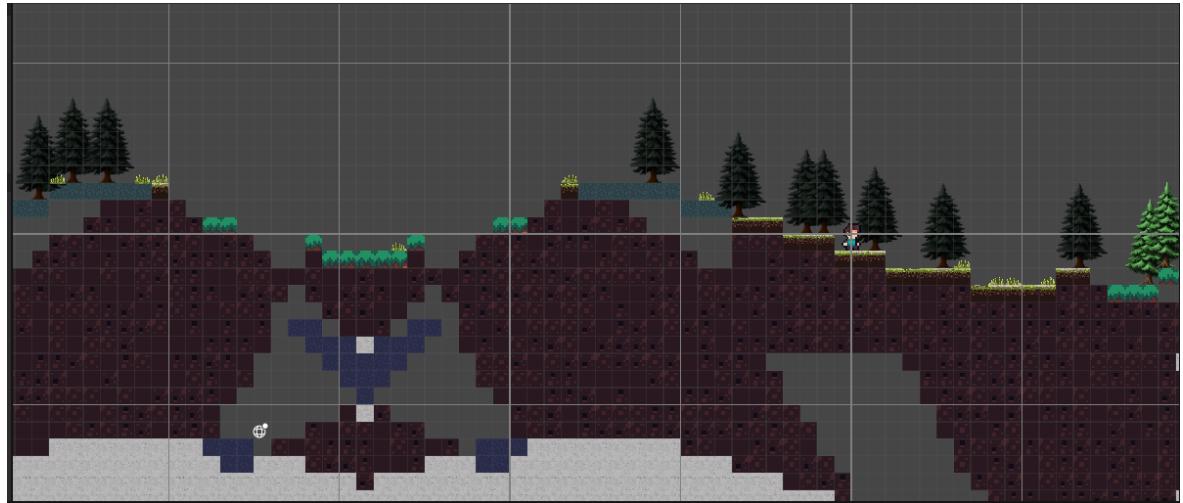
Các hệ sinh thái trong Game:

- Vùng thảo nguyên: Là hệ sinh thái chiếm phần lớn nhất trong thế giới với 40% tỉ lệ xuất hiện. Vùng thảo nguyên có xu hướng nhiều đồi núi nhỏ hơn các nơi khác với tỉ lệ kiểm được khoáng sản quý cao nhất.



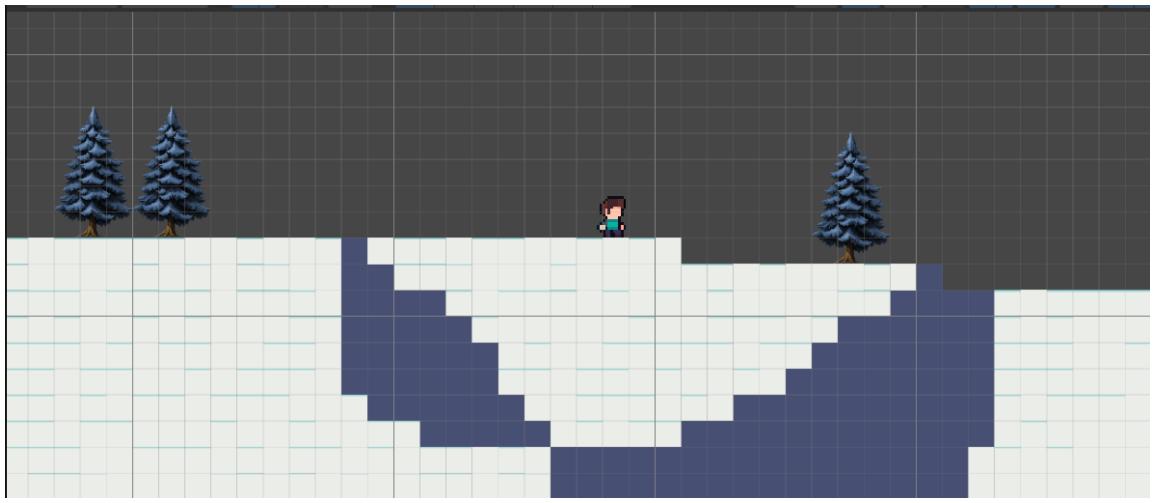
Hình 4.34. Hệ sinh thái thảo nguyên

- Vùng rừng rậm: Chiếm 20% địa hình, gồ ghề hơn so với các hệ sinh thái khác và cũng là nơi có nhiều cây cối nhất



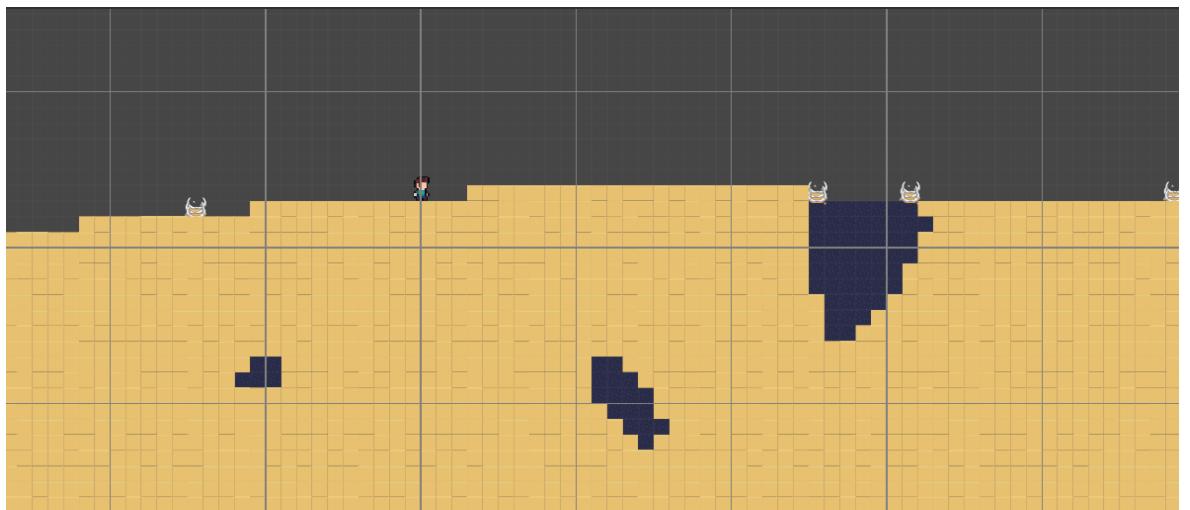
Hình 4.35. Hệ sinh thái rừng rậm

- Vùng tuyết trắng: Chiếm 20% địa hình, là vùng cao nhất đồng thời cũng là nơi cây cối thưa thớt nhất, hầu như không có gì ở đây ngoài tuyết trắng. Nhưng người chơi cần phải vượt qua hệ sinh thái này để đến với các hệ sinh thái nhiều vật phẩm khác



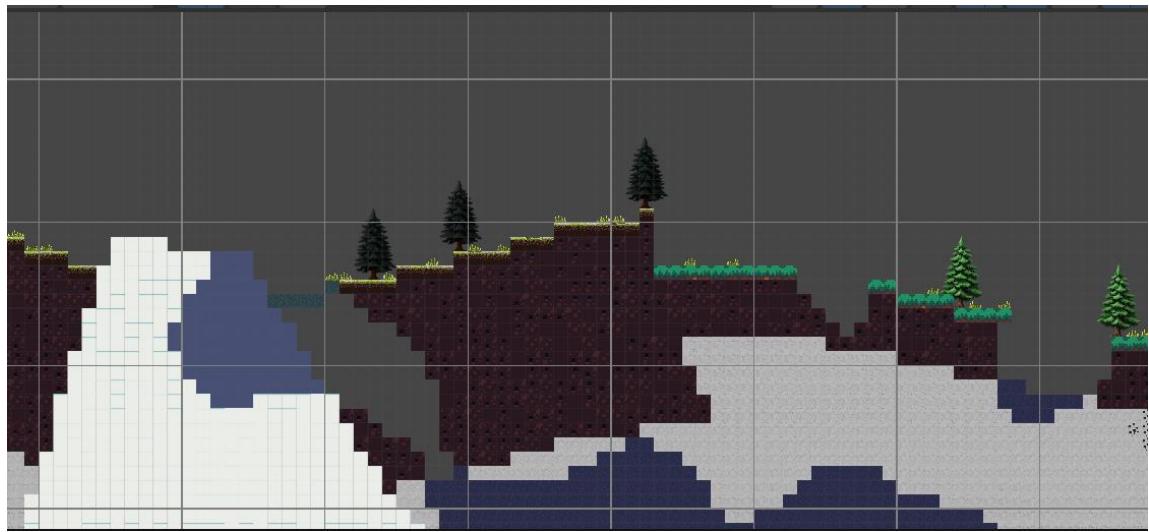
Hình 4.36. Hệ sinh thái tuyết trắng

- Vùng sa mạc: Chiếm 20% địa hình, là vùng thấp nhất trong địa hình, bằng phẳng nhất. Ở sa mạc không tìm được cây cối nhưng sẽ tìm được một vài vật phẩm thú vị nơi đây



Hình 4.37. Hệ sinh thái sa mạc

Dưới đây là ví dụ cho một địa hình được khởi tạo với các hệ sinh thái liền kề nhau nhưng chúng là độc lập và khác biệt

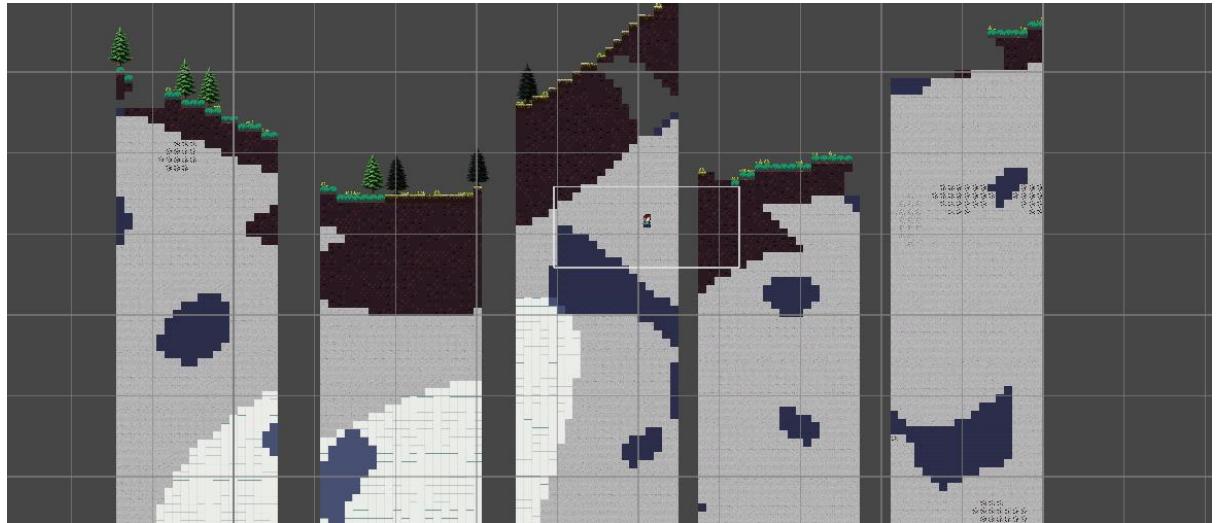


Hình 4.38. Các hệ sinh thái

4.1.4 Thuật ngữ “Chunk”

“Chunk” là một thuật ngữ được sử dụng trong game Sandbox để mô tả một phần nhỏ của thế giới game. Việc sử dụng chunk giúp cải thiện hiệu suất và khả năng quản lý thế giới game trong các game sandbox rộng lớn. Trong game, chunk đang được mặc định tính bằng độ lớn của thế giới chia cho 20.

Như trong ảnh dưới đây, các chunk được chia đều dựa theo độ lớn của địa hình thành các chunk nhỏ



Hình 4.39. Chunk

Tính từ vị trí của người chơi, các chunk ở 2 bên gần nhất (được tính bằng thuật toán đo khoảng cách sau đây) sẽ hiển thị còn các chunk xa hơn được ẩn đi với mục đích cải thiện hiệu suất, giảm độ nặng và tăng trải nghiệm mượt mà cho người chơi khi khởi tạo map to.

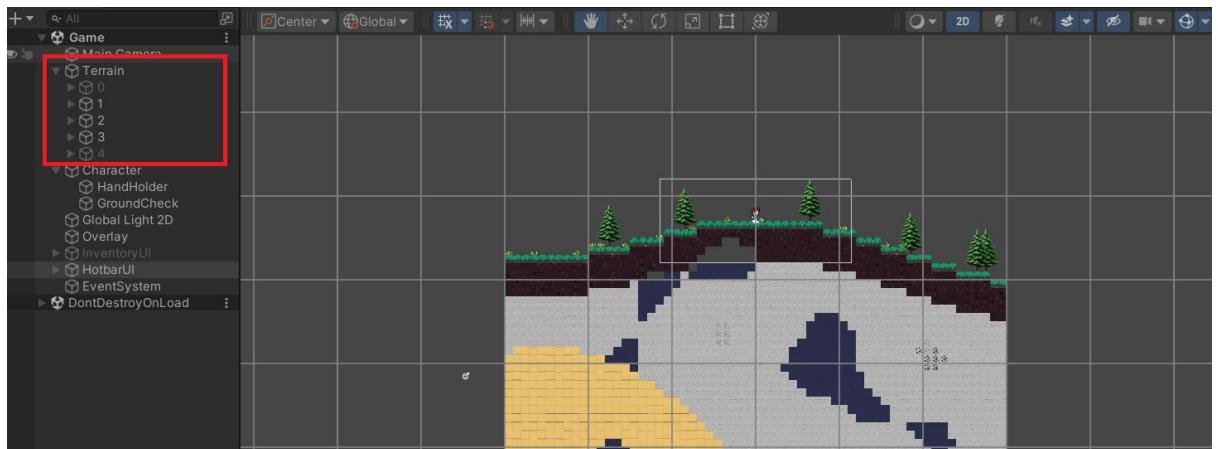
Orthographic là một chế độ camera trong Unity được sử dụng để hiển thị các

đối tượng như thể chúng được nhìn từ một góc nhìn trực giao (vuông góc với mặt phẳng). Tăng orthographic size của camera giúp chuyển chunk mượt mà hơn, không bị mất địa hình trong khi di chuyển

```
2 references
public void LoadChunks()
{
    for (int i = 0; i < worldChunks.Length; i++)
    {
        if (Vector2.Distance(new Vector2((i * chunkSize) + (chunkSize / 2), 0),
                             new Vector2(player.transform.position.x, 0)) > Camera.main.orthographicSize * 6f)
        {
            worldChunks[i].SetActive(false);
        }
        else
        {
            worldChunks[i].SetActive(true);
        }
    }
}
```

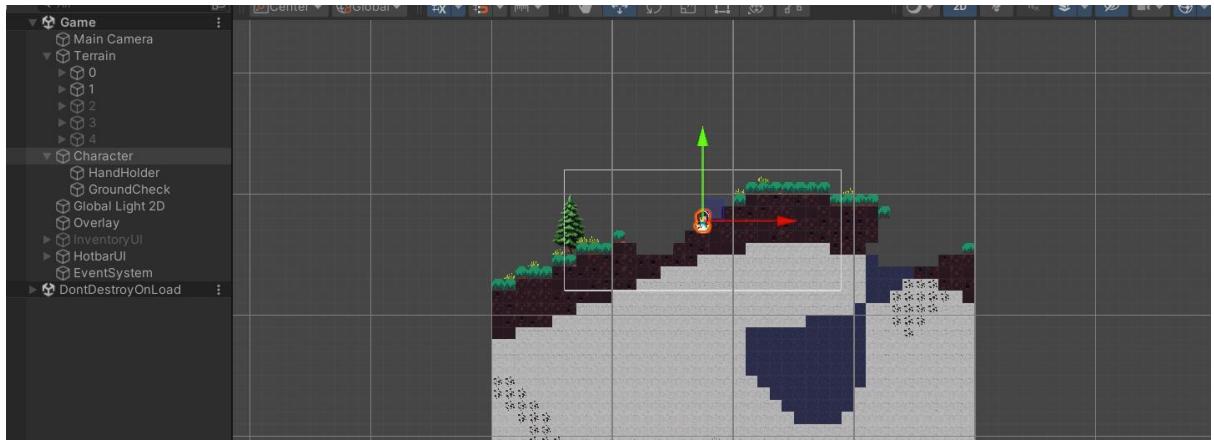
Hình 4.40. Thuật toán tải Chunk ở nơi nhân vật đứng

Cùng quan sát địa hình mẫu sau đây với kích thước khởi tạo địa hình là 100x100, ta sẽ thấy từ vị trí của nhân vật, các chunk xa tầm sẽ bị ẩn đi



Hình 4.41. Ví dụ tải Chunk 1

Nếu thử di chuyển nhân vật qua phía khác, ta sẽ thấy các chunk nhân vật đi tới hiện lên, các chunk xa hơn tự động ẩn đi



Hình 4.42. Ví dụ tải Chunk 2

4.1.5 Thuật ngữ “Bedrock”

Bedrock (tảng đá gốc) là thuật ngữ phổ biến trong các game Sandbox, dùng để chỉ lớp đáy cứng của thế giới game. Nó là lớp đá không thể phá hủy và thường được tạo ra bằng một loại vật liệu đặc biệt. Bedrock đóng vai trò quan trọng trong các game sandbox, được dùng để giới hạn thế giới game, giúp xác định độ cao thấp nhất của thế giới game, ngăn người chơi đào sâu hơn nữa. Điều này giúp đảm bảo tính ổn định của cấu trúc thế giới và tránh các lỗi phát sinh do người chơi đào sâu quá mức.



Hình 4.43. Bedrock

4.1.6 Tạo Shader bằng Universal Render Pipeline (URP)

Shader đóng vai trò quan trọng trong việc tạo đồ họa 2D/3D đẹp mắt và chân thực trong Unity. Shader là những chương trình nhỏ được viết bằng ngôn ngữ lập trình shader, chạy trên GPU (bộ xử lý đồ họa) để tạo ra các hiệu ứng hình ảnh cho các đối tượng 2D/3D. Shader cho phép mô phỏng các hiệu ứng ánh sáng tự nhiên như tán xạ ánh sáng, phản xạ, đổ bóng, v.v. Trong phạm vi đồ án này sẽ thiết kế một

Shader vùng sáng khi không bị che khuất và tối dần đi khi có vật che khuất. Hai loại shader phổ biến nhất trong Unity là **Unlit Shader** và **Lit Shader**.

Unlit Shader: Unlit Shader là loại shader không tính toán ánh sáng. Điều này có nghĩa là các vật thể được render với Unlit Shader sẽ không bị ảnh hưởng bởi nguồn sáng trong scene và cũng không tạo ra bóng đổ.

- **Đặc điểm:** Unlit Shader thường **nhanh hơn** Lit Shader vì nó bỏ qua việc tính toán ánh sáng. Do đó, Unlit Shader là lựa chọn tốt cho các đối tượng không cần hiệu ứng ánh sáng phức tạp hoặc cho các dự án nhắm mục tiêu đến phần cứng cấu hình thấp.

Mặc dù Unlit Shader không xử lý ánh sáng, nhưng chúng thường có các thuộc tính để kiểm soát giao diện của vật liệu, chẳng hạn như:

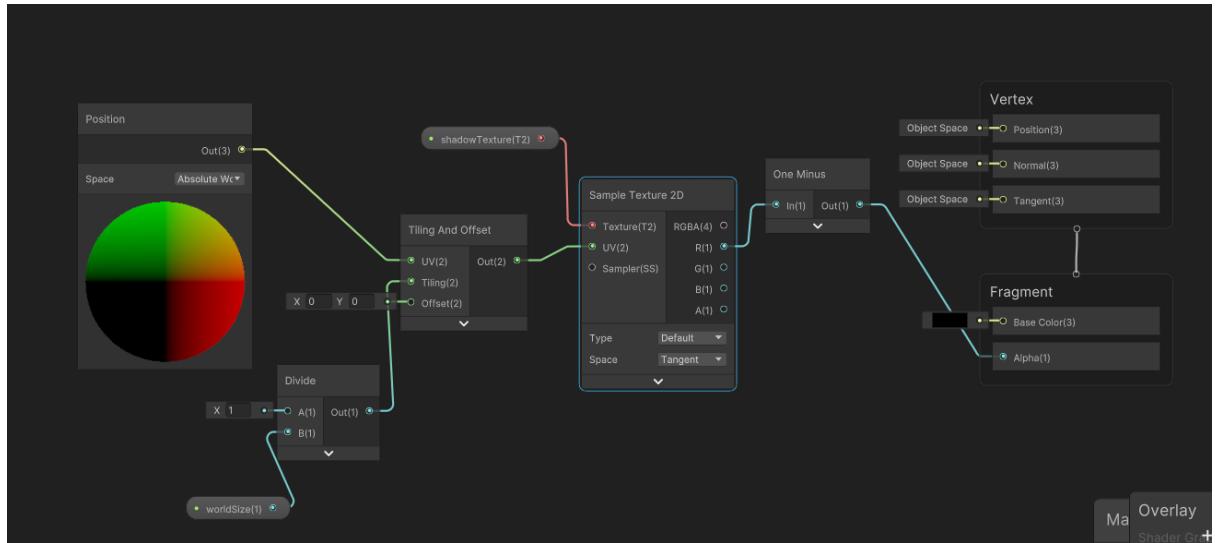
- **Main Texture:** Cài đặt kết cấu chính cho vật liệu.
- **Color:** Cài đặt màu cơ bản của vật liệu.
- Các thuộc tính khác có thể thay đổi tùy theo Unlit Shader cụ thể bạn đang sử dụng.

Lit Shader: Lit Shader là loại shader tính toán ánh sáng. Điều này có nghĩa là các vật thể được render với Lit Shader sẽ bị ảnh hưởng bởi nguồn sáng trong scene của bạn và cũng tạo ra bóng đổ. Lit Shader mô phỏng cách ánh sáng tương tác với các vật thể trong thế giới thực, giúp tạo ra hình ảnh 3D chân thực và sống động hơn.

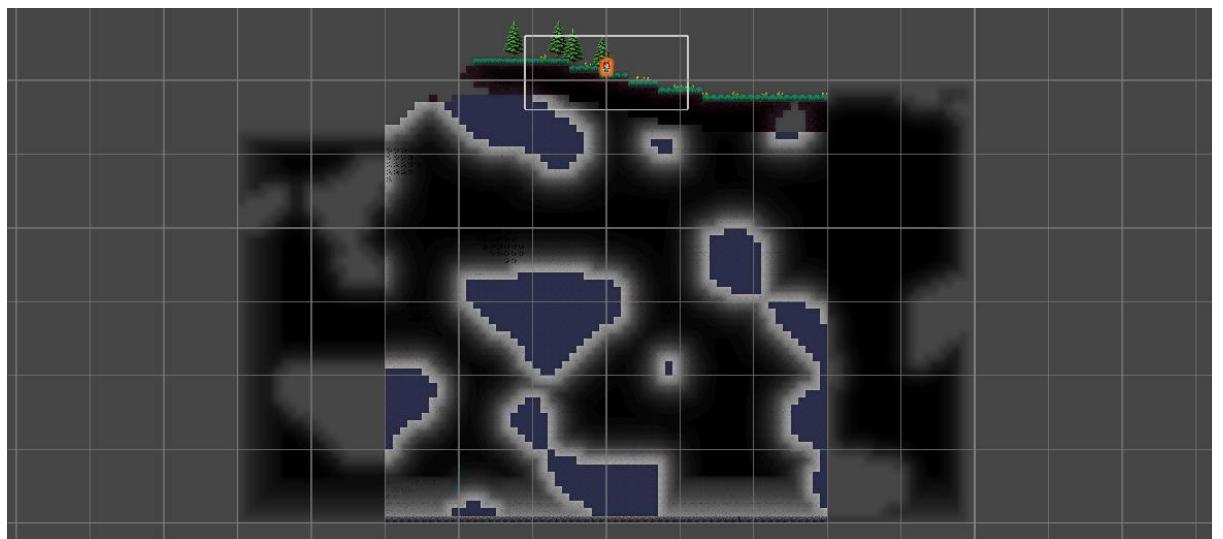
- **Đặc điểm:** Lit Shader mang lại hình ảnh 3D chân thực và sống động hơn nhờ mô phỏng ánh sáng.
- **Hiệu ứng ánh sáng:** Lit Shader cho phép bạn tạo ra nhiều hiệu ứng ánh sáng phức tạp như:
 - **Bóng đổ:** Các vật thể sẽ tạo ra bóng đổ thực tế dựa trên vị trí của nguồn sáng.
 - **Ánh sáng tán xạ:** Ánh sáng sẽ tán xạ trên bề mặt vật thể, tạo ra hiệu ứng chân thực hơn.
 - **Phản xạ:** Vật thể có thể phản xạ ánh sáng từ các vật thể khác, tạo ra hiệu ứng bóng mượt.

Universal Render Pipeline (URP) là một hệ thống render được tích hợp sẵn trong Unity, cung cấp cho người dùng khả năng tạo đồ họa chất lượng cao với hiệu suất tốt trên nhiều nền tảng khác nhau. URP được giới thiệu lần đầu trong Unity 2019 và nhanh chóng trở thành lựa chọn phổ biến cho các nhà phát triển game, thay thế cho hệ thống render mặc định (Built-in Render Pipeline).

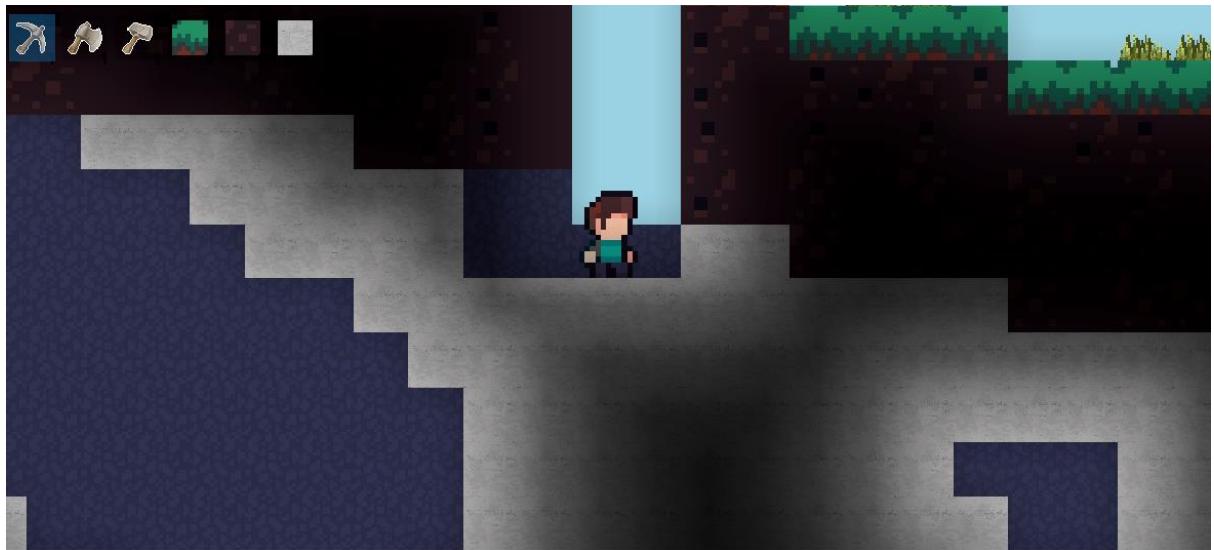
Để tạo hiệu ứng ánh sáng, ta thiết kế một Overlay như một tấm màn bao phủ lên toàn bộ bề mặt địa hình và điều chỉnh lại điều kiện được tháp sáng hay tắt đi bằng code



Hình 4.44. Overlay



Hình 4.45. Áp dụng Overlay



Hình 4.46. Hệ thống ánh sáng trong Game

4.2 Triển khai hệ thống vật phẩm

4.2.1 Hệ thống vật phẩm

Có 2 loại vật phẩm: block (khối), tool (công cụ)

```
6 references
public enum ItemType
{
    2 references
    tool,
    3 references
    block
};
```

Hình 4.47. Phân loại vật phẩm

- **Block** là loại vật phẩm có thể được khai thác bằng các công cụ thích hợp về túi đồ sau đó dùng những khối này để xây dựng công trình.

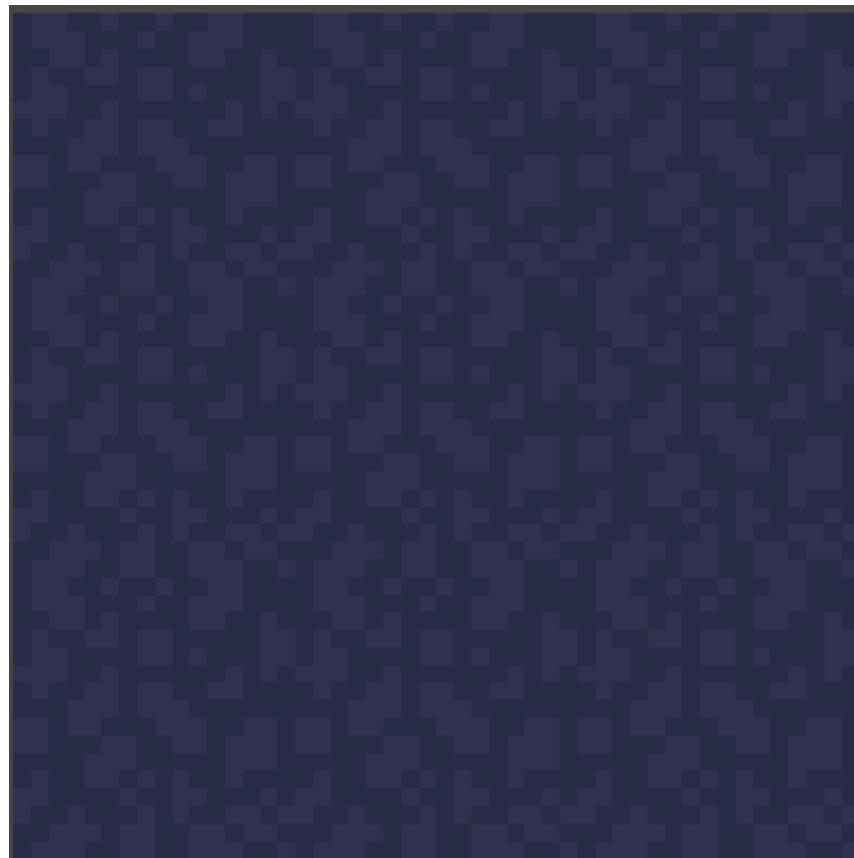
Các vật phẩm ở **mặt nền** như cây, cỏ, tường đá có thể đi xuyên qua được và phải dùng công cụ thích hợp để tương tác được với nó.



Hình 4.48. Vật phẩm nền - Hoá thạch ở sa mạc

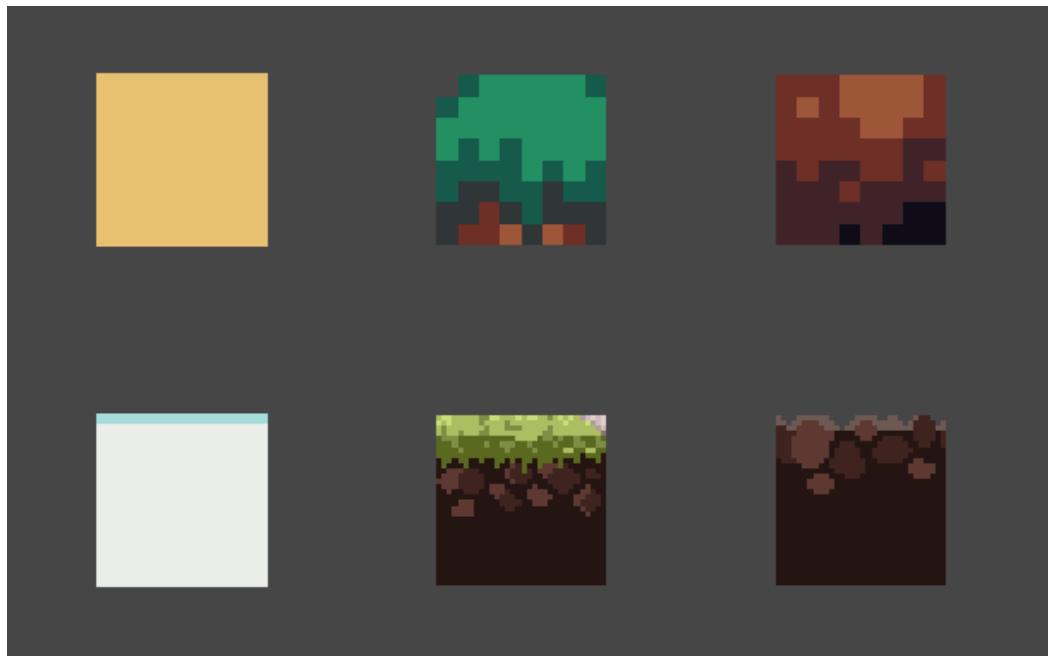


Hình 4.49. Vật phẩm nền - Cây và cỏ ở môi trường tự nhiên

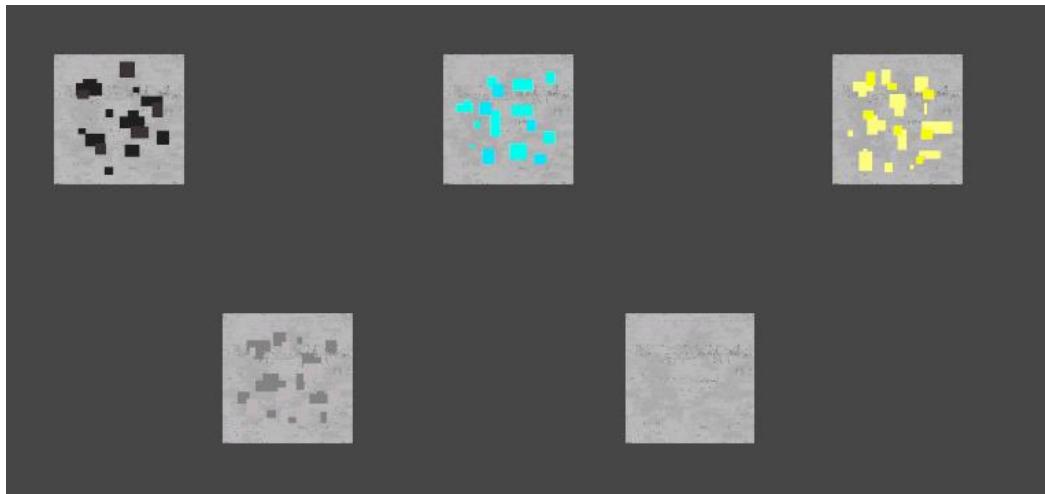


Hình 4.50. Vật phẩm nền - Lớp tường trong hang động

Các vật phẩm ở **mặt trước** có thể chặn đường đi của nhân vật, có va chạm vật lý và trọng lực



Hình 4.51. Vật phẩm mặt trước - Khối đất ở các biomes



Hình 4.52. Vật phẩm mặt trước - Các loại quặng

Mỗi khối khi được khai thác sẽ rót ra các vật phẩm tương ứng với khối đó và lưu vào túi đồ, cài đặt loại công cụ được phép khai thác cho mỗi khối là khác nhau, vì vậy nếu dùng sai công cụ sẽ không tương tác được với vật phẩm ở nền, cũng không tương tác được với khối ở mặt trước. Cuối cùng, dựa theo các tựa game đình đám như Minecraft hay Terraria, các vật phẩm được lưu trữ trong túi đồ nhân vật cũng có một số lượng nhất định gọi là “Stack”, một ô vật phẩm chứa tối đa vật phẩm cùng loại là 64 cái, tức là 1 stack bằng 64 vật phẩm cùng loại trong 1 ô chứa.

- **Tool** là loại vật phẩm dạng công cụ đặc biệt để tương tác với khối, sẽ được đề cập ở phần sau

4.2.2 Vật phẩm rơi ra

Khi tương tác với khối bằng công cụ, hàm Destroy() sẽ được gọi và khối khối bị huỷ, xoá khỏi khỏi danh sách địa hình, sau đó tạo ra một vật phẩm có kích thước nhỏ hơn 1/3 đồng thời khởi tạo các thuộc tính như trong *Hình. 46*

Khi người chơi đến gần, khối rơi ra sẽ biến mất và được nạp vào túi đồ của nhân vật đồng thời cũng nhận các thuộc tính của khối đó

```

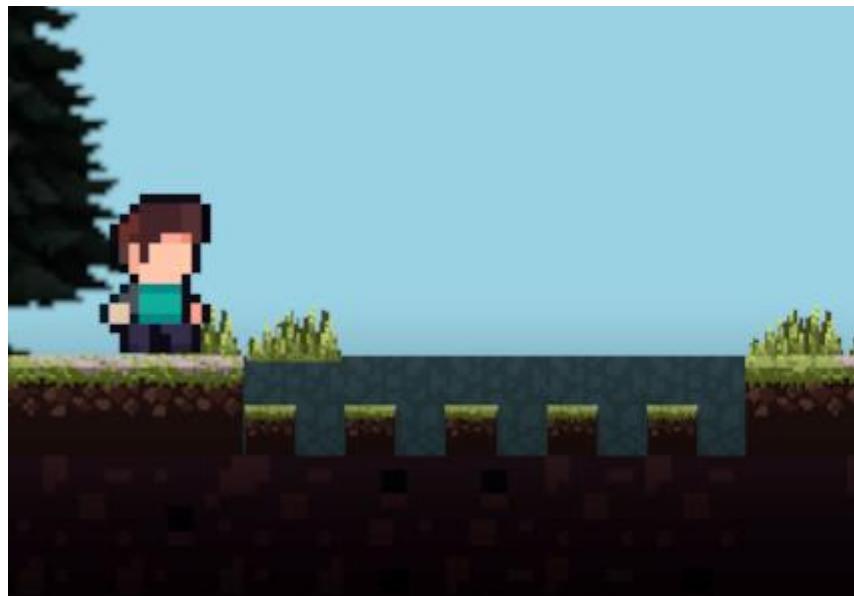
1 reference
public static TileClass CreateInstance(TileClass tile, bool isNaturallyPlaced)
{
    var thisTile = ScriptableObject.CreateInstance<TileClass>();
    thisTile.Init(tile, isNaturallyPlaced);

    return thisTile;
}

1 reference
public void Init(TileClass tile, bool isNaturallyPlaced)
{
    tileName = tile.tileName;
    wallVariant = tile.wallVariant;
    tileSprites = tile.tileSprites;
    isBgElement = tile.isBgElement;
    toolToBreak = tile.toolToBreak;
    tileDrop = tile.tileDrop;
    isStackable = tile.isStackable;
    naturallyPlaced = isNaturallyPlaced;
}

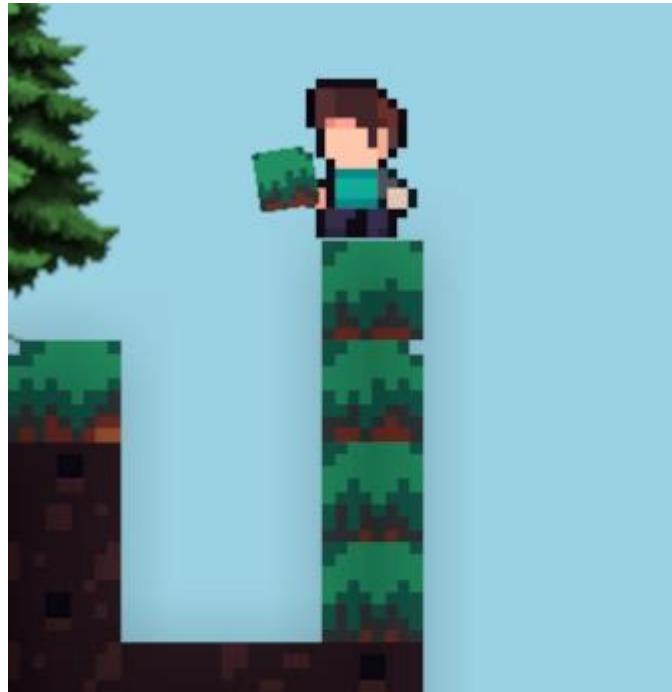
```

Hình 4.53. Khởi tạo vật phẩm rơi ra



Hình 4.54. Khởi vật phẩm rơi ra khi khai thác

Để đặt khối trên địa hình, nhấn chuột phải để thao tác, không thể đặt khối nếu ở phía dưới của nó không có các vật tựa. Khi người chơi chọn lựa khối muốn đặt trên thanh hotbar, ở vị trí tay nhân vật cũng sẽ hiển thị vật phẩm đó đang được cầm



Hình 4.55. Nhân vật cầm vật phẩm

4.2.3 Cơ chế đặt và đào khói trong Game

Mặc định, nhân vật không thể tương tác với các khối ở quá xa mình, chỉ có thể tương tác với những vật phẩm cách bản thân 2 blocks, sau đó tạo hai thuộc tính x, y nhận giá trị toạ độ ở vị trí con trỏ chuột đang hướng tới, từ đó khi người chơi nhấn sẽ chạy hàm huỷ phá huỷ khối ở vị trí con trỏ chuột và nạp vật phẩm rơi ra vào túi đồ.

```
// NOTE Tools interacts
if (Vector2.Distance(transform.position, mousePos) <= playerRange)
{
    if (mining)
    {
        terrainGenerator.toolBreakTile(mousePos.x, mousePos.y, selectedItem);
    }
}

mousePos.x = Mathf.RoundToInt(Camera.main.ScreenToWorldPoint(Input.mousePosition).x - 0.5f);
mousePos.y = Mathf.RoundToInt(Camera.main.ScreenToWorldPoint(Input.mousePosition).y - 0.5f);
```

Hình 4.56. Cơ chế đào khói

Đặt khối lên địa hình cũng như vậy, khi nơi người chơi muốn đặt khối hợp lệ, lấy vị trí toạ độ tại nơi con trỏ chuột thông qua hai biến x, y được gán như *Hình 49*, đặt khối được chọn lên địa hình thành công khi có một khối khác ở vị trí bất kỳ xung quanh nó.

```

// NOTE Place block
if (Vector2.Distance(transform.position, mousePos) <= playerRange &&
    Vector2.Distance(transform.position, mousePos) > 0.5f)
{
    if (place)
    {
        if (selectedItem != null)
        {
            if (selectedItem.itemType == ItemClass.ItemType.block)
            {
                if (terrainGenerator.CheckTile(selectedItem.tile, mousePos.x, mousePos.y, false))
                    inventory.Remove(selectedItem);
            }
        }
    }
}

```

Hình 4.57. Đặt khối tại vị trí trỏ chuột

4.3 Triển khai công cụ trong game

Công cụ là vật phẩm mặc định sẽ đi cùng với nhân vật từ lúc khởi tạo trò chơi. Mỗi công cụ có một mục đích khác nhau, cần phải sử dụng đúng mới có thể khai thác vật phẩm loại khối được.

- Công cụ búa: là loại công cụ dùng để tương tác với tường đất trong hang động, dùng khi người chơi cần ánh sáng dưới lòng đất, đập lớp đất đi để có ánh sáng hoặc cần khai thác số lượng lớn tường đất về để xây dựng công trình.



Hình 4.58. Công cụ búa

- Công cụ rìu: là loại công cụ dùng để tương tác với cây, gỗ. Các vật phẩm cây được đặt là vật phẩm nền, người chơi có thể đi xuyên qua được vì vậy để chặt được cây, người chơi phải dùng rìu để tương tác và nhận được gỗ.



Hình 4.59. Công cụ rìu

- Công cụ cuốc: là loại công cụ khá nhiều công dụng khi có thể tương tác với hầu hết các vật phẩm ở mặt trước địa hình, từ đào lớp đất trên bề mặt đến đào đá và quặng ở dưới mặt đất. Tuy nhiên không thể lấy gỗ và đào tường được.



Hình 4.60. Công cụ cuốc

Nhấn chuột trái để dùng công cụ, animation được thiết kế cho việc khai thác sẽ được thực thi cùng lúc. Nếu khối vượt quá phạm vi của nhân vật thì animation sẽ không được thực thi.

4.4 Triển khai nhân vật

4.4.1 Xác định và mô tả nhân vật

- Tên nhân vật: Tuunu
- Giới tính: Không xác định

- Tuổi: 5
- Giới thiệu: Tuunu là một đứa trẻ 5 tuổi đang dạo chơi trong thế giới tí hon của mình, thế giới trong mơ của Tuunu được tạo nên bởi những sở thích ở thế giới thật của đứa trẻ. Với ước mơ được chu du thế giới và tính tò mò khám phá, Tuunu thích sưu tập các vật phẩm ở những nơi mình đặt chân đến và đánh dấu nó bằng cách xây dựng một công trình to bự ở đây đính kèm vài viên đá quý. Một giấc mơ tự do mà Tuunu yêu thích vào mỗi giấc ngủ dài.

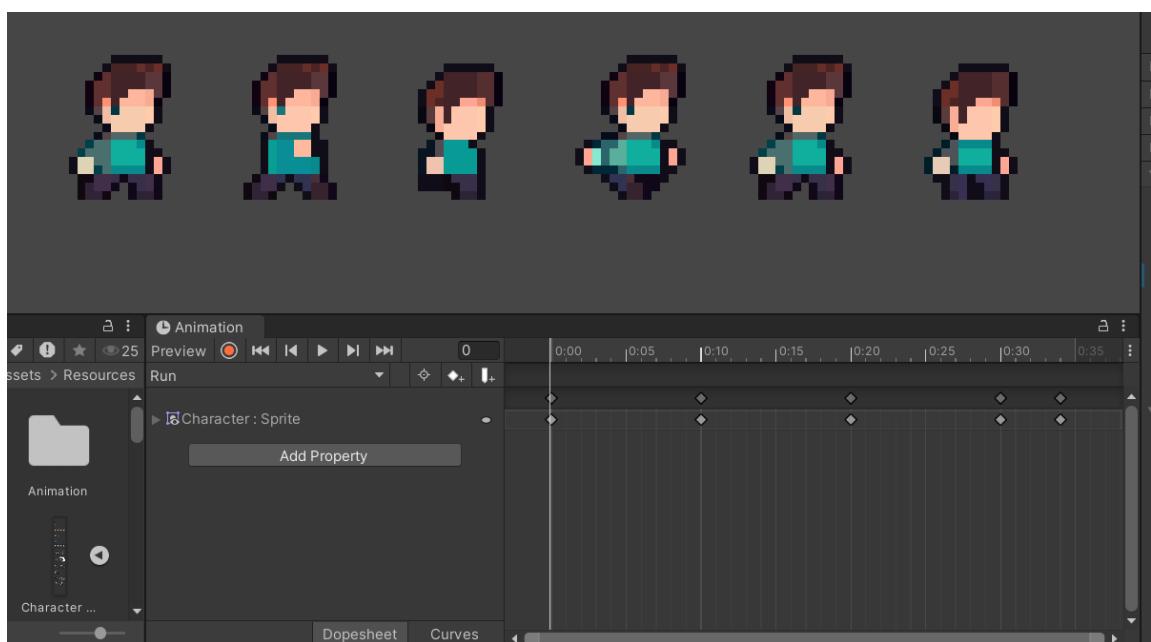
4.4.2 Các hoạt ảnh Animation

- Idle (Đứng yên)



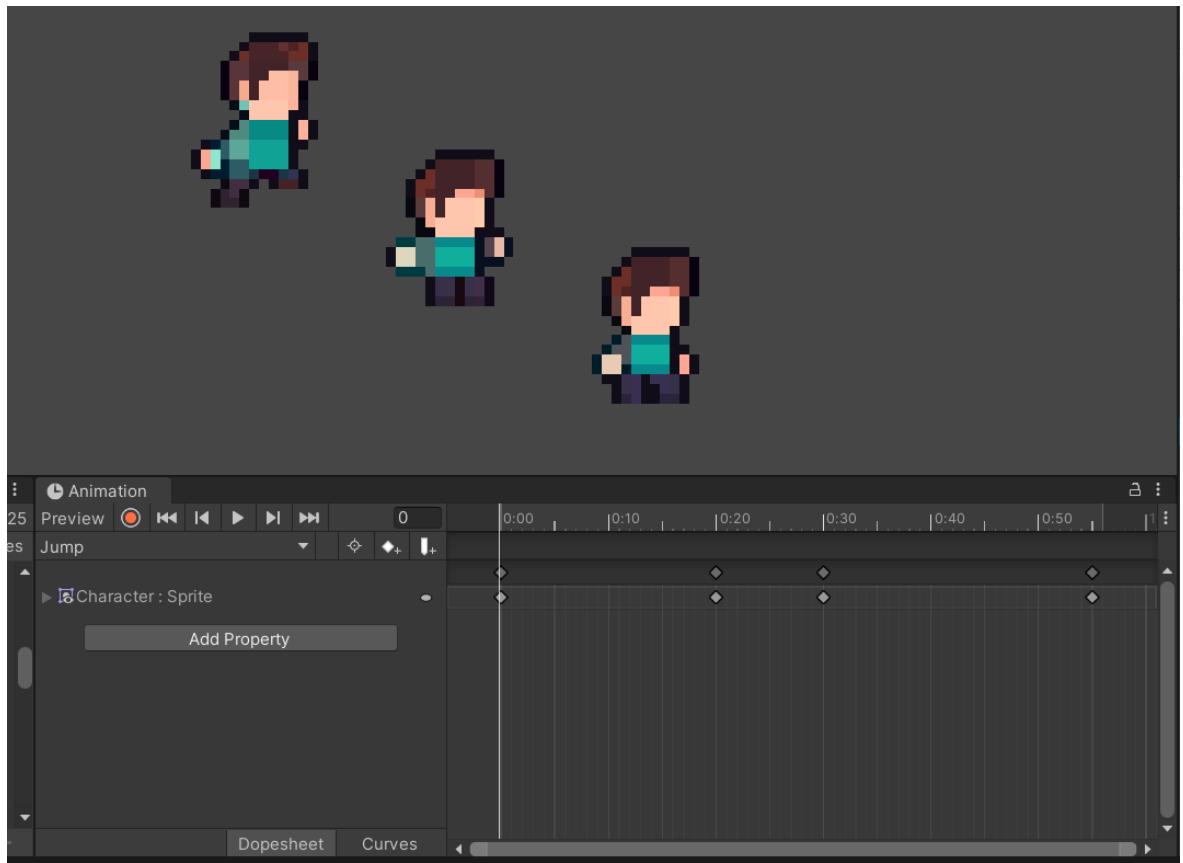
Hình 4.61. Hoạt ảnh Idle

- Run (Chạy)



Hình 4.62. Hoạt ảnh Run

- Jump (Nhảy)



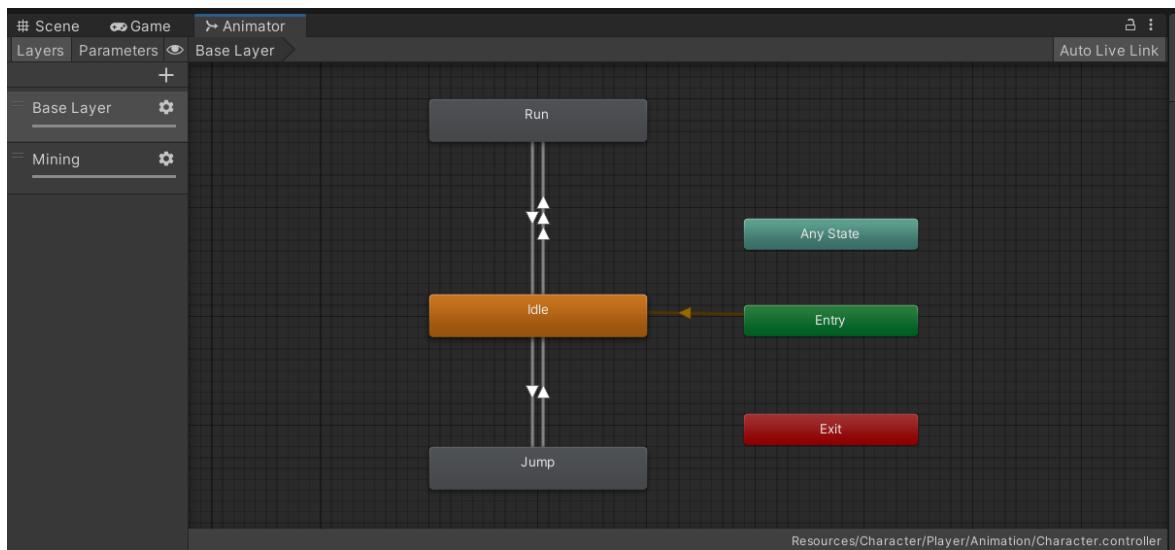
Hình 4.63. Hoạt ảnh Jump

- Khai thác



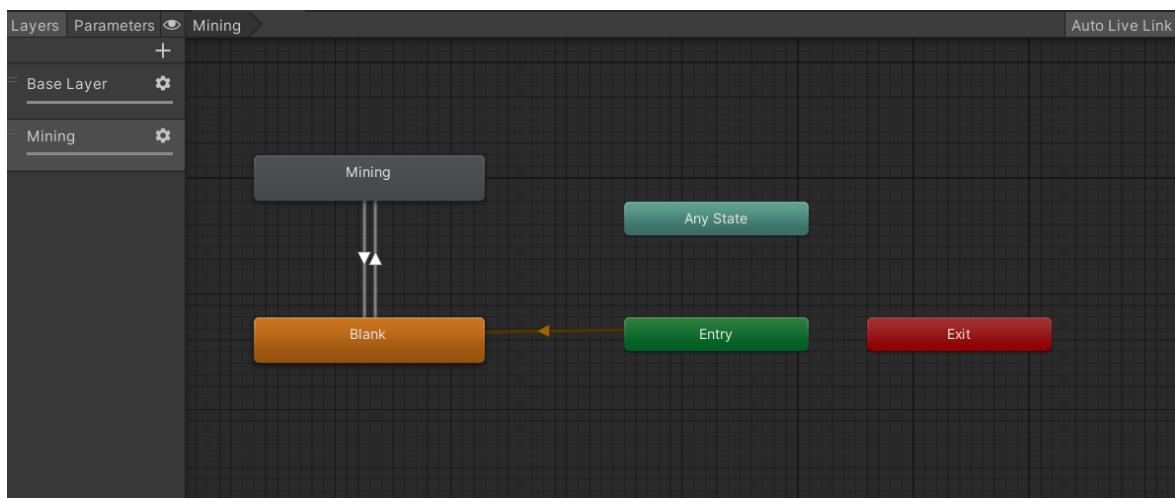
Hình 4.64. Hoạt ảnh Mining

- **Animator Idle, Run và Jump**



Hình 4.65. Animator Idle, Run, Jump

- **Animator Mining**



Hình 4.66. Animator Mining

4.4.3 Quy định nơi sử dụng hoạt ảnh

- **Idle:** Khi không có bất cứ hoạt động gì xảy ra, animation Idle sẽ được thực thi, người chơi sẽ đứng yên tại chỗ
- **Run:** Để điều khiển nhân vật di chuyển, dùng phím “A” để chạy qua bên trái và phím “D” để chạy qua bên phải. Trong khi di chuyển có thể chạy hoạt ảnh Mining cùng lúc
- **Jump:** Để điều khiển nhân vật nhảy, dùng phím “Space” để nhảy lên trên các khối cao. Trong khi nhảy có thể chạy hoạt ảnh Mining và Run cùng lúc

```

// NOTE: Run
if (horizontal > 0)
{
    transform.localScale = new Vector3(1, 1, 1);
}
else if (horizontal < 0)
{
    transform.localScale = new Vector3(-1, 1, 1);
}

// NOTE: Jump
if (vertical > 0.1f || jump > 0.1f)
{
    if (onGround)
    {
        movement.y = jumpForce;
    }
}

```

Hình 4.67. Run và Jump

- **Mining:** Thêm vào trò chơi một vài tính thực tế khi chọn đúng công cụ thì animation Mining mới được thực thi. Tạo một điều kiện ràng buộc người chơi phải chọn đúng loại công cụ trên thanh hotbar thì animation mới hoạt động.

```

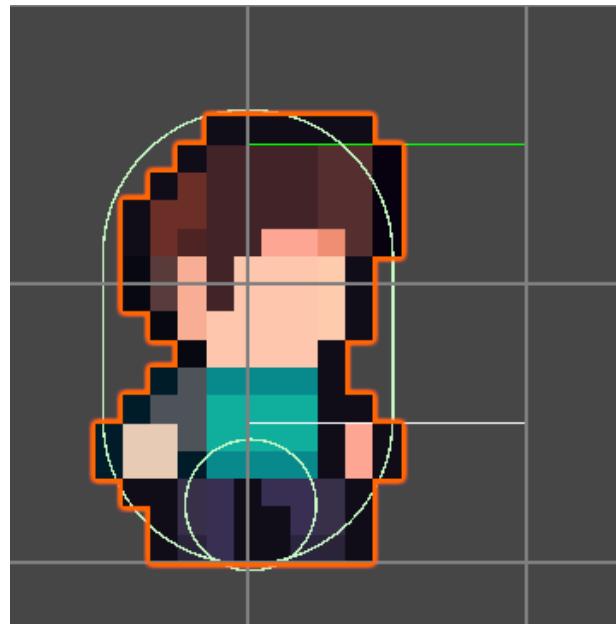
if (selectedItem != null && selectedItem.itemType == ItemClass.ItemType.tool)
{
    animator.SetBool("Mining", mining);
}

```

Hình 4.68, Quy định noi dùng Animation Mining

4.4.4 Cơ chế tự động nhảy (Autojump) khi gặp một vật cản

Sử dụng Raycast để xác định vị trí sẽ tự động nhảy khi va chạm vật lý với một khối bất kỳ. Nếu trên đầu nhân vật bị cản sẽ không thực hiện tự động nhảy



Hình 4.69. Sử dụng Raycast xác định vị trí

```

1 reference
public bool FootRayCast()
{
    RaycastHit2D hit = Physics2D.Raycast(transform.position - (Vector3.up * -0.5f), Vector2.right * transform.localScale.x, 1f, layerMask);
    return hit;
}

1 reference
public bool HeadRayCast()
{
    RaycastHit2D hit = Physics2D.Raycast(transform.position + (Vector3.up * 1.5f), Vector2.right * transform.localScale.x, 1f, layerMask);
    return hit;
}

```

Hình 4.70; Tạo Raycast

```

// NOTE: Autojump
if (FootRayCast() && !HeadRayCast() && movement.x != 0)
{
    if (onGround)
    {
        movement.y = jumpForce * 0.6f;
    }
}

rigid.velocity = movement;

```

Hình 4.71. Autojump

4.5 Triển khai hệ thống kho đồ và thanh hotbar

4.5.1 Hệ thống kho đồ

Hệ thống kho đồ đóng vai trò quan trọng trong các game sandbox, giúp người

chơi lưu trữ, quản lý và sử dụng các vật phẩm thu thập được trong quá trình chơi game. Tối đa mỗi ô chứa được 64 vật phẩm cùng loại, nếu tiếp tục khai thác cùng vật phẩm sẽ được lưu vào ô tiếp theo trong túi đồ nếu còn trống

- **Kho vật phẩm và ô chứa vật phẩm:** Để tạo một hệ thống kho đồ, ta sử dụng các UI được Unity Editor cung cấp để tạo một giao diện kho đồ, vị trí ô vật phẩm được khởi tạo bằng code để có thể linh hoạt trong việc điều chỉnh số lượng ô vật phẩm, mỗi vị trí ô đồ sẽ nắm giữ toàn bộ thuộc tính của vật phẩm được nạp vào nó.

```
// Inventory
for (int x = 0; x < inventoryWidth; x++)
{
    for (int y = 0; y < inventoryHeight; y++)
    {
        GameObject invenSlot = Instantiate(inventorySlotPrefab, inventoryUI.transform.GetChild(0).transform);
        invenSlot.GetComponent<RectTransform>().localPosition = new Vector3((x * multiplier.x) + invenOffset.x,
            (y * multiplier.y) + invenOffset.y);
        invenUIslot[x, y] = invenSlot;
        inventorySlot[x, y] = null;
    }
}
```

Hình 4.72. Khởi tạo ô vật phẩm

Thao tác mở kho đồ bằng cách nhấn phím “E”, tắt bằng cách nhấn phím “E” lần nữa, khi bật kho đồ, thanh hotbar sẽ ẩn đi và ngược lại. Tất cả vật phẩm nhân vật thu được sẽ hiện ở đây



Hình 4.73. Giao diện túi đồ

- **Thêm vật phẩm vào túi đồ:** Để thêm vật phẩm vào túi đồ sau khi nhặt vật phẩm rơi, đầu tiên ta cần kiểm tra xem ô chứa vật phẩm đã có vật phẩm nào chưa? Nếu có thì có cùng loại không? Cùng loại thì đã đầy Stack chưa? Vì vậy việc đầu tiên cần làm là viết hàm kiểm tra vật phẩm

```

1 reference
public Vector2Int Contains(ItemClass item)
{
    for (int y = inventoryHeight - 1; y >= 0; y--)
    {
        for (int x = 0; x < inventoryWidth; x++)
        {
            if (inventorySlot[x, y] != null)
            {
                if (inventorySlot[x, y].item.itemName == item.itemName)
                {
                    if (item.isStackable && inventorySlot[x, y].quantity < stackLimit)
                        return new Vector2Int(x, y);
                }
            }
        }
    }

    return Vector2Int.one * -1;
}

```

Hình 4.74. Kiểm tra vật phẩm

Sau khi kiểm tra kĩ lưỡng xong, ta tiến hành thêm vào ô đồ, nếu ô đồ đã tồn tại vật phẩm thì ta kiểm tra có cùng loại vật phẩm không, nếu cùng loại thì đến số lượng Stack tối đa chưa, nếu không đạt điều kiện thì vật phẩm sẽ được thêm vào ô kế tiếp và tiếp tục kiểm tra điều kiện.

```

Vector2Int itemPos = Contains(item);
bool added = false;

if (itemPos != Vector2Int.one * -1)
{
    inventorySlot[itemPos.x, itemPos.y].quantity += 1;
    added = true;
}

if (!added)
{
    for (int y = inventoryHeight - 1; y >= 0; y--)
    {
        if (added)
            break;

        for (int x = 0; x < inventoryWidth; x++)
        {
            if (inventorySlot[x, y] == null)
            {
                inventorySlot[x, y] = new InventorySlot { item = item, slot = new Vector2Int(x, y), quantity = 1 };
                added = true;
                break;
            }
            else
            {
                if (inventorySlot[x, y].item.itemName == item.itemName)
                {
                    inventorySlot[x, y].quantity += 1;
                    added = true;
                    break;
                }
            }
        }
    }
}

```

Hình 4.75. Thêm vật phẩm vào kho đồ

Cuối cùng gọi hàm Update để cập nhật giao diện kho đồ và thanh hotbar (được để cập ngay sau đây)

- **Cập nhật túi đồ**

Đây là hàm được dùng để cập nhật lại tất cả giao diện và thuộc tính, số lượng cho ô vật phẩm. Nếu vật phẩm được dùng hết, ô vật phẩm sẽ trả lại giá trị null như ban đầu, nếu vật phẩm được thêm vào được thực thi thông qua hàm Add() hoặc được loại bỏ bằng hàm Remove(), số lượng trên giao diện sẽ được cập nhật theo

```
for (int x = 0; x < inventoryWidth; x++)
{
    for (int y = 0; y < inventoryHeight; y++)
    {
        if (inventorySlot[x, y] == null)
        {
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().sprite = null;
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().enabled = false;

            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().text = "0";
            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().enabled = false;
        }
        else
        {
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().enabled = true;
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().sprite = inventorySlot[x, y].item.itemSprite;

            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().text = inventorySlot[x, y].quantity.ToString();
            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().enabled = true;
        }
    }
}
```

Hình 4.76. Cập nhật giao diện ô chứa vật phẩm

- **Loại bỏ vật phẩm**

Khi sử dụng vật phẩm trong túi đồ, số lượng sẽ được loại bỏ dần, cho tới khi sử dụng hết, hàm Update() sẽ đặt lại giao diện ô vật phẩm về null không chứa vật phẩm nào và tiếp tục nạp vào các vật phẩm khác sau đó

```

// Inventory
for (int x = 0; x < inventoryWidth; x++)
{
    for (int y = 0; y < inventoryHeight; y++)
    {
        if (inventorySlot[x, y] == null)
        {
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().sprite = null;
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().enabled = false;

            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().text = "0";
            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().enabled = false;
        }
        else
        {
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().enabled = true;
            invenUIslot[x, y].transform.GetChild(0).GetComponent<Image>().sprite = inventorySlot[x, y].item.itemSprite;

            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().text = inventorySlot[x, y].quantity.ToString();
            invenUIslot[x, y].transform.GetChild(1).GetComponent<Text>().enabled = true;
        }
    }
}

```

Hình 4.77. Loại bỏ vật phẩm

4.5.2 Hệ thống thanh hotbar

Thanh hotbar là thanh công cụ tiện ích cho người chơi khi có thể dùng trực tiếp trên giao diện Game mà không cần phải bật tắt túi đồ to. Để tạo thanh hotbar, ta sử dụng Prefab và vị trí ô vật phẩm được khởi tạo bằng code để có thể linh hoạt trong việc điều chỉnh số lượng, vị trí ô vật phẩm, mỗi vị trí ô đồ sẽ nắm giữ toàn bộ thuộc tính của vật phẩm tương ứng với hàng đầu tiên trong kho đồ.

```

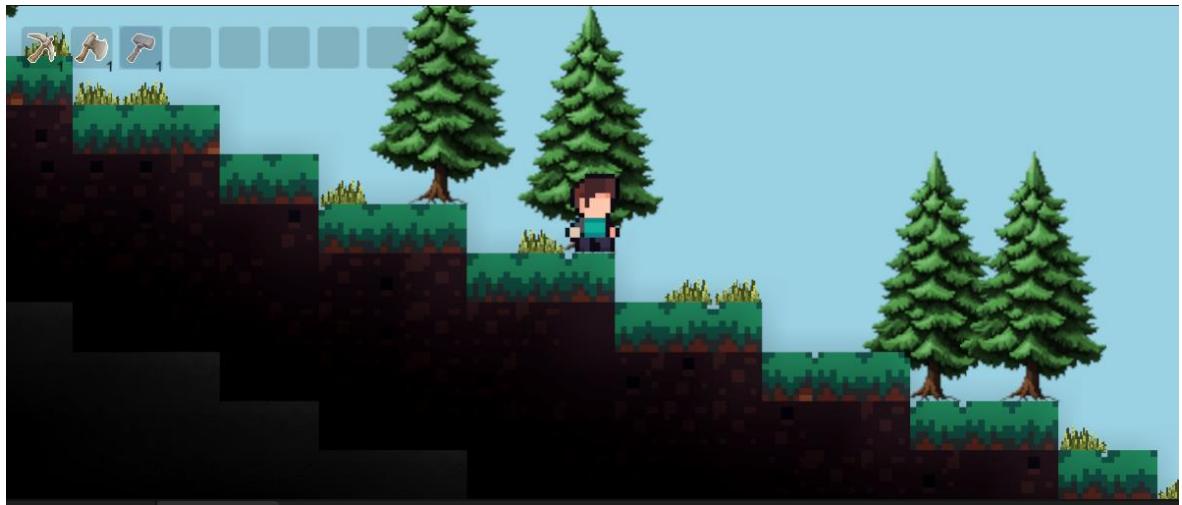
// Hot bar
for (int x = 0; x < inventoryWidth; x++)
{
    GameObject hotbar_Slot = Instantiate(inventorySlotPrefab, hotbarUI.transform.GetChild(0).transform);
    hotbar_Slot.GetComponent<RectTransform>().localPosition = new Vector3((x * multiplier.x) + hotbarOffset.x,
                                                                        hotbarOffset.y);

    hotbarUISlot[x] = hotbar_Slot;
    hotbarSlot[x] = null;
}

```

Hình 4.78. Khởi tạo ô vật phẩm ở thanh hotbar

Thao tác chọn vật phẩm bằng cách lăn chuột, thanh hotbar sẽ ẩn đi khi mở giao diện túi đồ và ngược lại. Và ba công cụ ở đầu tiên chính là ba công cụ khởi đầu sẽ đi cùng với người chơi.



Hình 4.79. Thanh hotbar

4.6 Kiểm Tra và Sửa Lỗi:

- **Mục tiêu:** Đảm bảo trò chơi hoạt động đúng như dự kiến, không có lỗi kỹ thuật hay lỗi giao diện nào xảy ra, đem lại trải nghiệm tốt nhất cho người chơi.

- **Phương pháp:**

Kiểm tra hệ thống (System Testing):

- Đảm bảo khi khởi tạo trò chơi, địa hình được tạo một cách phong phú, có kết hợp đủ bốn hệ sinh thái, kiểm tra số lượng và độ lớn các hang động là vừa đủ.
- Thủ tương tác với trò chơi để đảm bảo các lỗi chơi hoạt động hiệu quả, không gặp lỗi.
- Xác minh hệ thống kho đồ hoạt động hiệu quả khi liên tục thu thập và loại trừ vật phẩm
- Điều chỉnh các lỗi từ hệ thống Console và các gợi ý trong Visual Studio Code

Kiểm tra người dùng (User Testing):

- Gửi game cho bạn bè để trải nghiệm và nhận những phản hồi góp ý về game để xác định các vấn đề về trải nghiệm người dùng và từ đó có những điều chỉnh phù hợp hơn.

ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

1. Kết Quả Đạt Được

Hoàn thành một tựa Game với các yếu tố cơ bản của thể loại Sandbox và cũng là thể loại trò chơi yêu thích của bản thân, luôn cố gắng tìm hiểu và xây dựng trò chơi tốt nhất có thể trong khả năng của mình. Dù trò chơi không hoàn hảo nhưng những kiến thức học hỏi được và đam mê trong quá trình làm đồ án lần này là vô giá đối với bản thân.

Điểm mạnh:

- Trò chơi có đạt được các **yếu tố cơ bản** của thể loại Sandbox đó là thế giới rộng lớn và tính sáng tạo, tự do
- **Sự đồng nhất** trong hình ảnh và âm thanh mang lại trải nghiệm thú vị cho người chơi.
- Giao diện người dùng **thân thiện** và dễ sử dụng giúp tối ưu hóa trải nghiệm người dùng.

Điểm yếu:

- **Nội dung chưa phong phú và đa dạng:** Ít hoạt động để tương tác.
- **Chưa tối ưu hóa code:** Vẫn còn hiện tượng giật lag đôi chút khi chạy Game
- **Thiếu tính hoàn thiện:** Mặc dù trò chơi đã đạt được mức độ hoàn thiện ở mức 80%, nhưng vẫn còn một số tính năng và nội dung cần được cải thiện và phát triển thêm.
- **Hạn chế về mặt kỹ thuật:** Kiến thức về ngôn ngữ lập trình và Unity ở mức cơ bản, ít kinh nghiệm làm việc nên các chức năng còn hạn chế. Giao diện khá đơn giản, chưa có tính sáng tạo.
- **Thiếu nhiều tính năng dự tính:** Không thể hoàn thành các dự tính khi thiết kế kế hoạch xây dựng game, thời gian nghiên cứu và sửa lỗi chiếm khá nhiều trong quá trình thực hiện đề tài này.

2. Đề xuất hướng phát triển trong tương lai

- **Thêm chế độ chơi sinh tồn:** Mở rộng tựa Game này bằng cách thêm một chế độ mới song song với chế độ sáng tạo đó là sinh tồn.
- **Mở rộng thế giới và nội dung:** Tạo thêm nhiều hệ sinh thái và các vật phẩm mới. Thêm các tính năng như chế tạo vật phẩm, vũ khí, quái vật và chức năng lưu Game

- **Cải Thiện Giao Diện Người Dùng:** Thêm các hoạt ảnh ở giao diện tham gia Game, tạo ra trải nghiệm tương tác thú vị hơn và thuận tiện hơn cho người chơi.

TÀI LIỆU THAM KHẢO

[1] - Matt Smith, Shaun Ferns - "Unity 2021 Cookbook: Over 140 recipes to take your Unity game development skills to the next level" - Packt Publishing – 2021.

[2] - Harrison Ferrone - "Learning C# by Developing Games with Unity 2021" - Packt Publishing – 2021.

[3] - Alan Thorn - "Pro Unity Game Development with C#: Leverage the power of Unity and become a pro at creating amazing games" - Apress – 2021.

[4] - Will Goldstone - "Unity 3.x Game Development Essentials" - Packt Publishing – 2011.

[5] - Mike Geig - "Unity Game Development in 24 Hours, Sams Teach Yourself" - Sams Publishing – 2015.

[6] - Andersson, R. - "Procedural Content Generation in Games Using Unity" - Proceedings of the 2013 International Conference on Game Development – 2013.

[7] - Whitehead, T., & Spencer, A. - "A Survey of Real-Time 3D Graphics with Unity" - Journal of Game Development - Vol. 10, No. 3 – 2020.

[8] - Smith, J. - "Implementing Realistic AI in Unity for RPG Games" - International Journal of Game Design and Development - Vol. 8, Issue 2 – 2019.

[9] - Jeremy Gibson Bond - "Introduction to Game Design, Prototyping, and Development: From Concept to Playable Game with Unity and C#" - Addison-Wesley Professional – 2014.

[10] - Youtube ErenCode – Truy cập 20/03/2024

Link truy cập: <https://www.youtube.com/@ErenCode>

[11] - Unity Official Documentation. - Truy cập 20/03/2024

Link truy cập: <https://docs.unity3d.com>.

[12] - Unity Learn. - Truy cập 25/03/2024

Link truy cập: <https://learn.unity.com>.

[13] - Shader và các material PBR – Truy cập 01/06/2024

Link truy cập: <https://www.youtube.com/watch?v=EHDYssPe3HQ>

[14] - Hướng dẫn Rendering trong Unity: Shaders Truy cập 01/06/2024

Link truy cập: <https://hackernoon.com/vi/huong-dan-render-trong-unity-shader-phần-1>

[15] - Showing Progress While Loading Scene Truy cập 10/06/2024

Link truy cập: <https://www.youtube.com/watch?v=qYoec93bZx0>