ĐẠI HỌC BÁCH KHOA HÀ NỘI

Graduation Research 1

Thiết kế xây dựng trò chơi 2D trên Unity

NGUYỄN ĐÌNH TRUNG

trung.nd20215249@sis.hust.edu.vn

Ngành Công nghệ thông tin

Giảng viên hướng dẫn: Nguyễn Bá Ngọc

Chữ kí GVHD

Lớp: IT – E7

Khoa: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 01/2024

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Giới thiệu về trò chơi và mục tiêu dự án

Trò chơi là một phần không thể thiếu trong cuộc sống hiện đại, mang lại niềm vui và thư giãn cho hàng triệu người trên khắp thế giới. Dự án "Game Test" ra đời với mục tiêu đem đến cho người chơi một trải nghiệm độc đáo và thú vị qua lối chơi đơn giản nhưng gây nghiện.

Mục tiêu chính của dự án "Game Test" là tạo ra một trò chơi giải trí hấp dẫn, thách thức và dễ chơi. Tôi mong muốn mang đến cho người chơi cảm giác hồi hộp khi điều khiển quả bóng đỏ đi qua những thử thách đa dạng.

1.2 Đặt vấn đề và lý do chọn đề tài

Trong thế giới game đa dạng ngày nay, việc tạo ra một trò chơi độc đáo và nổi bật là một thách thức lớn. "Game Test" được lựa chọn với ý định đánh bại thách thức này thông qua việc kết hợp yếu tố hành động nhanh nhẹn, lối chơi sáng tạo và thiết kế màn chơi đa dạng.

Trò chơi "Game Test" được lựa chọn vì tích hợp một loạt yếu tố mà tôi tin rằng sẽ thu hút người chơi. Tính đơn giản và dễ chơi của trò chơi sẽ thuận tiện cho nhiều người chơi, bao gồm cả những người mới bắt đầu. Đồng thời, việc thiết kế màn chơi đa dạng và thú vị sẽ giữ cho người chơi luôn có sự tò mò và hứng thú trong suốt quá trình chơi.

Với niềm đam mê và nhiệt huyết, tôi hy vọng rằng trò chơi "Game Test" sẽ mang lại cho người chơi những giờ phút vui vẻ và thú vị, và đồng thời, đóng góp một phần nhỏ vào thế giới game đa dạng và phong phú.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

2.1 Khảo sát hiện trạng

2.1.1 Mục tiêu khảo sát

Chúng ta cần xác định các yếu tố quan trọng như lối chơi ưa thích, độ khó mong muốn và các tính năng đáng chú ý mà người chơi mong muốn thấy trong trò chơi. Nhờ vào sự tham gia chân thành của người tham gia khảo sát, chúng tôi có thể tạo ra trải nghiệm chơi game tốt nhất dựa trên những thông tin đáng giá này. Bên cạnh đó, độ tuổi, giới tính của người chơi cũng là một yếu tố quan trọng.

2.1.2 Phương Pháp Khảo Sát

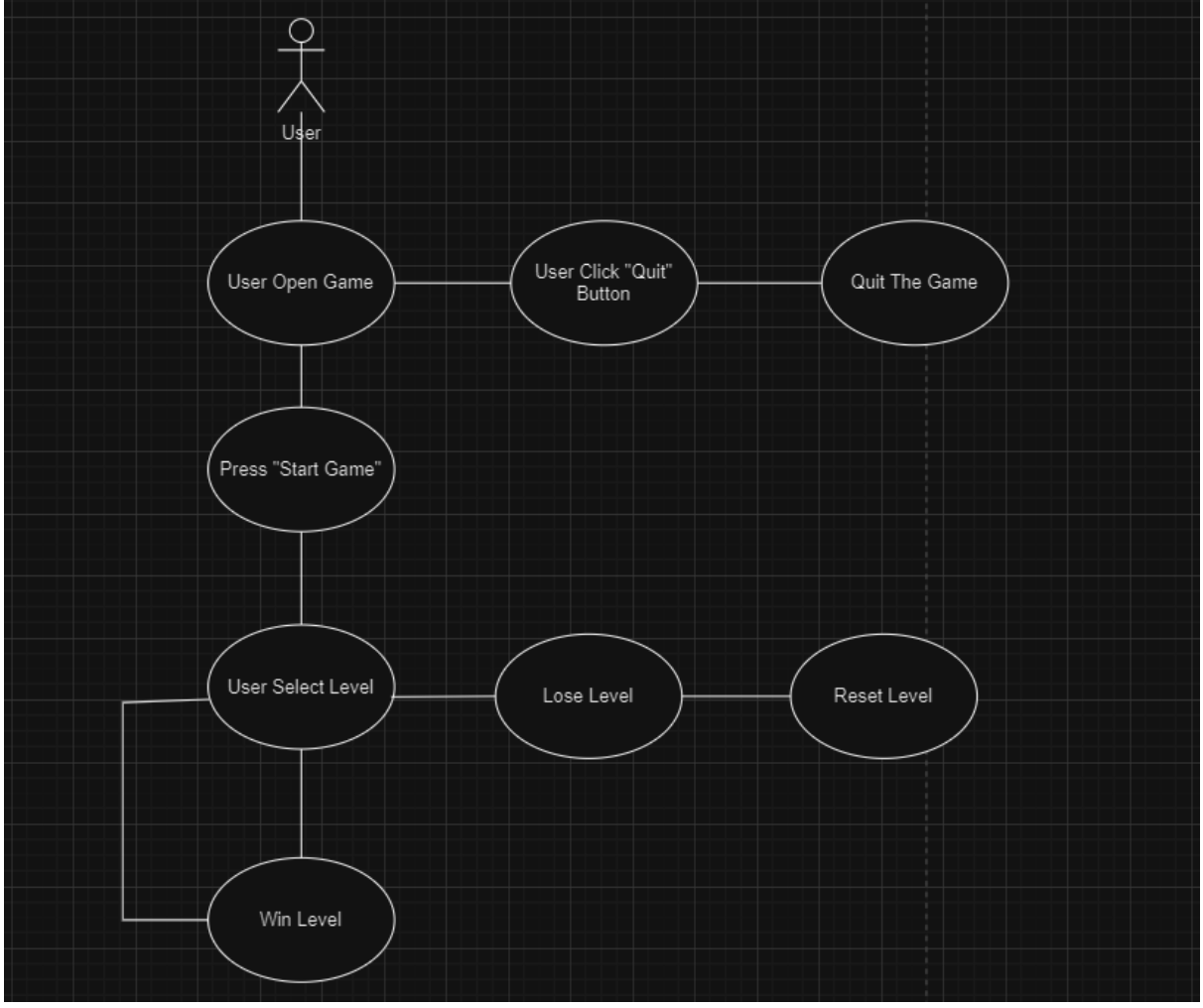
Em đã sử dụng một bộ câu hỏi đa dạng tạo bằng Google Form để đánh giá ý kiến của người chơi về các khía cạnh của trò chơi. Câu hỏi trong khảo sát bao gồm:

- Lối chơi yêu thích: Người chơi được yêu cầu chọn lối chơi yêu thích, ví dụ như hành động, giải đố, phiêu lưu, v.v.
- Độ khó mong muốn: Xác định mức độ khó mà người chơi cảm thấy thoải mái hoặc thách thức.
- Tính năng ưa thích: Người chơi đề xuất những tính năng mà họ muốn thấy trong trò chơi, ví dụ cấp độ tùy chỉnh, đơn giản hóa, v.v.

2.1.3 Kết quả khảo sát

Kết quả từ khảo sát cung cấp cái nhìn sâu rộng về sự mong đợi và nguyện vọng của người chơi. Những thông tin này đã đóng góp quan trọng vào quá trình thiết kế và phát triển trò chơi "Game Test". Tôi đã áp dụng các phản hồi và ý kiến từ khảo sát để tạo ra một trò chơi phù hợp với mong muốn của người chơi và đồng thời đảm bảo trải nghiệm chơi game thú vị và đa dạng.

2.2 Tổng quan chức năng



2.3 Đặc tả chức năng

Đối tượng sử dụng: Người chơi.

Use case này mô tả các bước để vào màn chơi và các kết quả màn chơi.

Các bước thực hiện:

- User thực hiện ấn vào icon mở game, game sẽ mở và đưa User đến màn hình chính.
- User ấn nút Start Game để vào chọn màn chơi.
- User vượt qua các chướng ngại vật, thu thập vật phẩm, chạy đến cuối màn chơi sẽ chiến thắng. Nếu như User bị đánh bại bởi địch hoặc đâm vào chướng ngại vật thì sẽ thua.
- Khi thắng sẽ hiện màn hình thắng với lựa chọn chơi level tiếp theo hoặc về màn hình chính. Nếu thua thì sẽ tự động chơi lại.\

2.4 Yêu cầu phi chức năng

- **Hiệu suất và Tính ổn định:** Trò chơi phải hoạt động mượt mà và không có sự gián đoạn lớn trong trải nghiệm người chơi.Thời gian tải trò chơi và chuyển giữa các màn chơi cần được giữ ở mức thấp.
- **Giao diện Người dùng:**

Giao diện trò chơi cần phải thân thiện với người dùng và dễ sử dụng.

Yêu cầu sử dụng biểu tượng, hình ảnh và màu sắc hài hòa để tạo sự thú vị và hấp dẫn.

- **Thiết kế cấp độ:**

Yêu cầu việc thiết kế các cấp độ phải có sự thay đổi về độ khó, sự thách thức và sự đa dạng để giữ cho trò chơi hấp dẫn và không đơn điệu.

- **Bảo mật và Quyền riêng tư:**

Trò chơi không được thu thập thông tin cá nhân của người chơi mà không có sự đồng ý.

3.1 Giới thiệu về công cụ làm game Unity

3.1.1 Tổng quan về Unity

Hình 3.1: Unity là gì?

Unity là một nền tảng phát triển trò chơi đa năng và mạnh mẽ. Được phát triển bởi công ty Unity Technologies, Unity cung cấp một môi trường làm việc tích hợp để tạo ra các trò chơi 2D, 3D và thực tế ảo (VR) hoặc thực tế tăng cường (AR). Điều đặc biệt về Unity là khả năng hỗ trợ cho nhiều nền tảng khác nhau như máy tính, điện thoại di động, máy chơi game và thậm chí cả nền tảng web.

Unity cung cấp một loạt công cụ sáng tạo, từ trình chỉnh sửa hình ảnh và âm thanh đến các công cụ quản lý đối tượng và thiết kế môi trường trò chơi. Nền tảng này cũng hỗ trợ việc lập trình bằng nhiều ngôn ngữ như C Sharp và JavaScript.

Với khả năng linh hoạt và cộng đồng phát triển mạnh mẽ, Unity đã trở thành một trong những công cụ phát triển trò chơi phổ biến và ưa chuộng nhất trên thị trường.

3.1.2 Ưu điểm

Unity có nhiều ưu điểm hấp dẫn, đó là lý do tại sao nền tảng này trở thành một công cụ phát triển trò chơi phổ biến:

Đa năng: Unity cho phép phát triển trò chơi 2D, 3D, thực tế ảo (VR) và thực tế tăng cường (AR) trên nhiều nền tảng khác nhau, bao gồm cả máy tính, điện thoại di động, máy chơi game và web.

Dễ học và sử dụng: Unity3D được built trong một môi trường phát triển tích hợp, cung cấp một hệ thống toàn diện cho các lập trình viên, từ soạn thảo mã nguồn, xây dựng công cụ tự động hóa đến trình sửa lỗi. Do được hướng đến đồng thời cả lập trình viên không chuyên và studio chuyên nghiệp, nên Unity khá dễ sử dụng.

Lập trình đa dạng: Unity hỗ trợ việc lập trình bằng nhiều ngôn ngữ như C Sharp và JavaScript, cho phép lập trình viên sử dụng ngôn ngữ mà họ đã quen thuộc.

Cộng đồng mạnh mẽ: Unity có một cộng đồng phát triển lớn và tích cực. Người dùng có thể chia sẻ kiến thức, tài liệu và tài nguyên miễn phí, giúp giải quyết vấn đề nhanh chóng.

Các công cụ sáng tạo: Unity cung cấp nhiều công cụ hỗ trợ sáng tạo như trình chỉnh sửa hình ảnh, âm thanh và môi trường trò chơi.

Kết xuất chất lượng cao: Unity cho phép tạo ra đồ họa và hiệu ứng chất lượng cao, giúp trò chơi trở nên hấp dẫn hơn.

Cross-Platform: Khả năng phát triển trên nhiều nền tảng giúp đạt tới một lượng người chơi rộng rãi.

Thư viện tài nguyên: Unity có cửa hàng tài nguyên cộng đồng và cửa hàng tài nguyên trả phí, giúp tìm kiếm và tích hợp các tài nguyên dễ dàng.

Khả năng tùy chỉnh: Unity cho phép tùy chỉnh các chức năng và tích hợp công cụ bên ngoài để đáp ứng nhu cầu cụ thể của dự án.

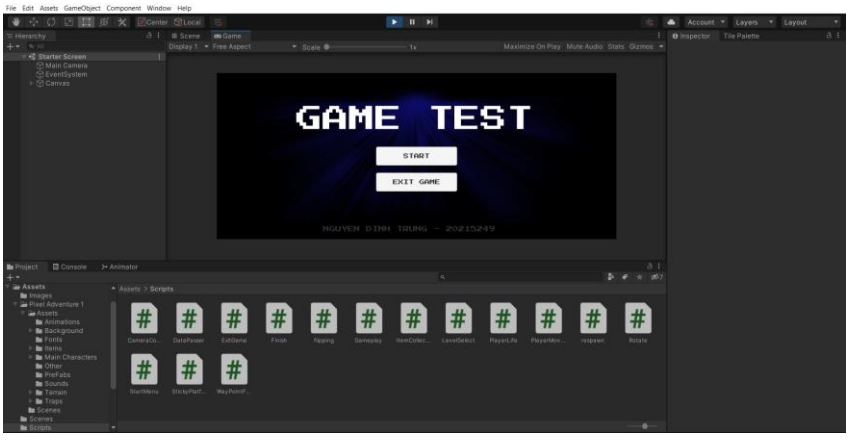
Tóm lại, Unity là một công cụ mạnh mẽ và linh hoạt, thích hợp cho cả người mới bắt đầu và các nhà phát triển chuyên nghiệp trong việc tạo ra các trò chơi chất lượng cao trên nhiều nền tảng khác nhau.

3.1.3 Các thành phần và khái niệm cơ bản

Asset: là những tài nguyên xây dựng nên một dự án Unity. Từ những tập tin hình ảnh, mô hình 3D đến các tập tin âm thanh. Unity gọi các tập

tin mà chúng ta dùng để tạo nên trò chơi là tài sản (Assets). Điều này lí giải tại sao tất cả các tập tin, thư mục của các dự án Unity đều được lưu trữ trong một thư mục có tên là “Assets”.

Scenes: là các màn chơi hoặc cảnh quan mà người chơi có thể trải nghiệm. Mỗi scene trong Unity có thể chứa các đối tượng, hình ảnh, âm thanh và các tài nguyên khác liên quan đến trải nghiệm trò chơi cụ thể. Ví dụ:



Hình 3.2: Main Menu Scene

Đây thường là scene mà người chơi sẽ thấy khi khởi động trò chơi.

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.

Game Object là một thực thể cơ bản trong Unity, có thể là bất cứ thứ gì trong trò chơi như nhân vật, vật phẩm, môi trường, ánh sáng, camera... Game Object có thể có một hoặc nhiều tính năng.

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

Scripts là các đoạn mã viết bằng ngôn ngữ lập trình (thường là C Sharp hoặc JavaScript) được gắn vào các Game Object để thêm hành vi và logic vào trò chơi.

Đặc điểm: Scripts là các tập hợp các câu lệnh và chức năng được viết để thực hiện các hành động cụ thể trong trò chơi. Chúng có thể điều khiển chuyển động, tương tác, hiệu ứng và hầu hết các khía cạnh của trò chơi.

Kết nối với Game Object: Mỗi Script được gắn liền với một Game Object. Khi Script được gắn vào Game Object, nó có khả năng truy cập và thao tác với các thành phần và thuộc tính của Game Object đó.

Chức năng và Phương thức: Trong một Script, bạn có thể định nghĩa các hàm (phương thức) để thực hiện các chức năng cụ thể. Ví dụ: hàm để điều khiển di chuyển của nhân vật, hàm để kiểm tra va chạm.

Tương tác: Scripts cho phép tạo ra các tương tác giữa các Game Object và với người chơi. Ví dụ: khi người chơi nhấn nút, Script có thể thực hiện hành động như bắn đạn hoặc mở cửa. Sự kiện và Gọi hàm: Scripts có thể phản hồi vào các sự kiện như nhấn nút, va chạm, thời gian, v.v. Bạn có thể kích hoạt các hàm trong Script dựa trên các sự kiện này.

Tùy chỉnh: Scripts cho phép bạn tạo ra các hành vi tùy chỉnh cho các Game Object. Điều này giúp tạo ra tính năng riêng biệt và độc đáo cho trò chơi của bạn.

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

3.1.4 Đặc điểm và tính năng

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

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

Terrains còn gọi chung là địa hình bao gồm phần đất nền của môi trường trong trò chơi cùng các đối tượng gắn liền như cây, cỏ, . . . Unity cung cấp một công cụ hỗ trợ rất tốt khả năng này với tên gọi là Terrains Tools cho phép chúng ta thiết kế địa hình với các công cụ vẽ dưới dạng Brush có nhiều thông số tùy chỉnh để tạo hình và lát Texture cho địa hình. Cùng với Terrain Tools là Tree Creator, một công cụ mạnh mẽ cho phép chúng ta tạo ra cây cối với hình dạng, kích thước và kiểu cách đa dạng.

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

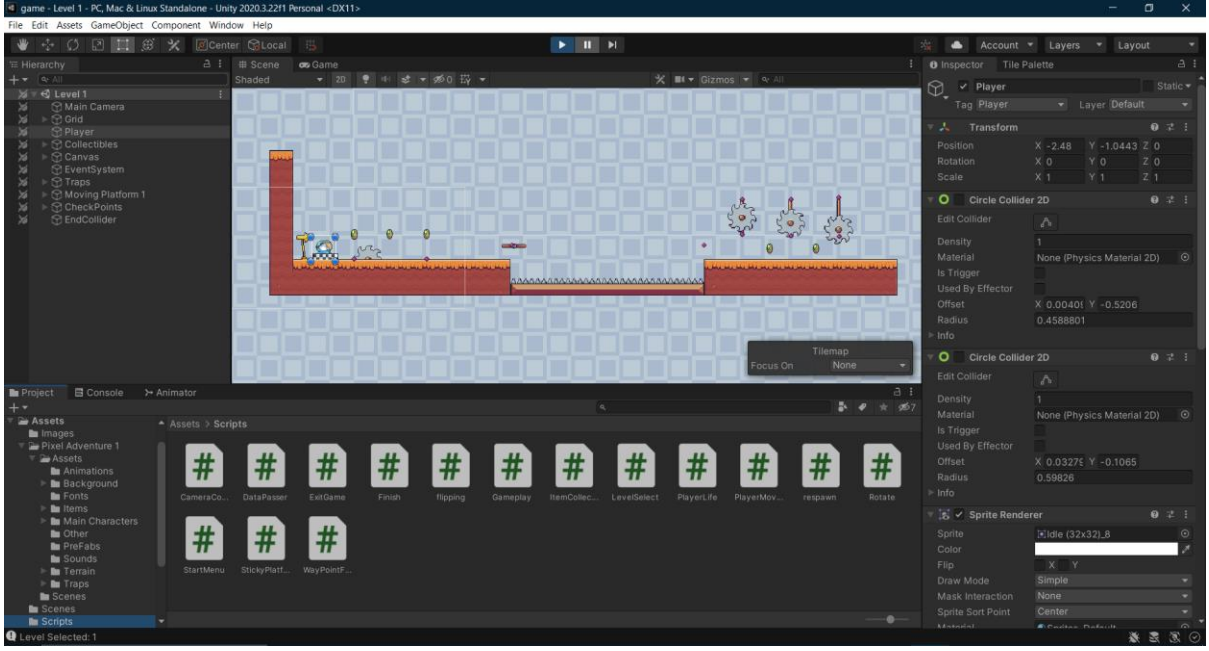
Physics: PhysX là một Engine mô phỏng và xử lí vật lý cực kỳ mạnh mẽ được phát triển bởi nhà sản xuất card đồ họa hàng đầu thế giới NVIDIA. Unity đã tích hợp Engine này vào để đảm nhận mọi vấn đề vật lý. Một số vấn đề vật lý được hỗ trợ bởi Unity như: Soft Bodies, Rigid Bodies, Joints, Cars, . . .

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

Programming: Lập trình là một trong những yếu tố quan trọng nhất trong phát triển Game. Lập trình cho phép nhà phát triển tạo nên khả năng tương tác, trí thông minh và yếu tố Gameplay cho trò chơi. Unity cho phép chúng ta lập trình bằng nhiều ngôn ngữ mạnh mẽ và phổ biến với các lập trình viên như: C Sharp, Javascript và Boo.

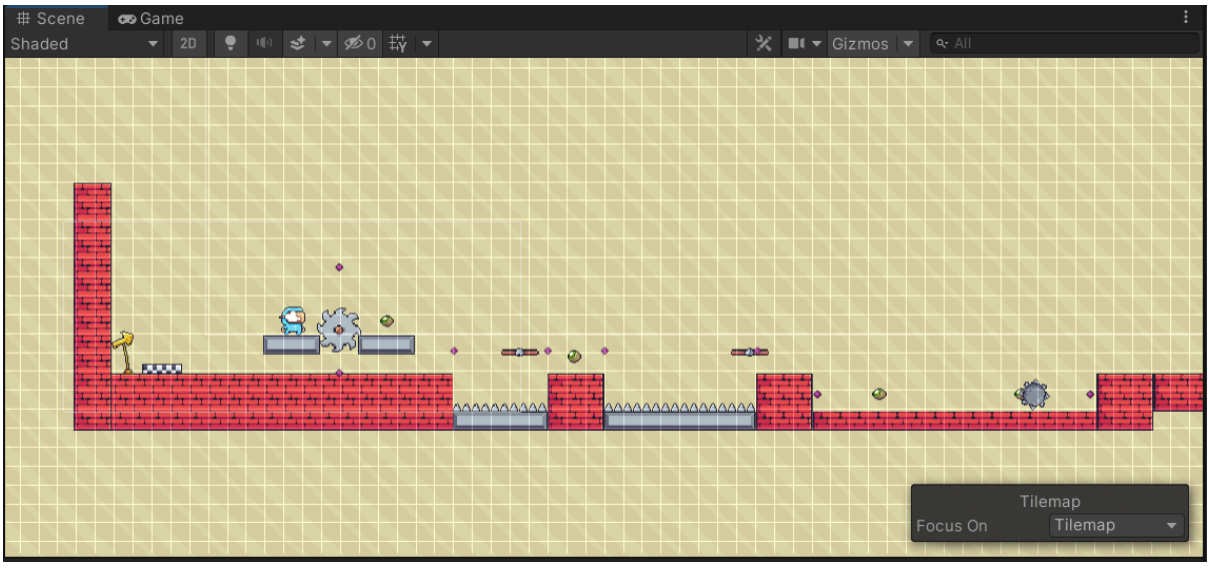
3.1.5 Giao diện cơ bản của Unity

Giao diện Unity được thiết kế một cách rõ ràng và dễ sử dụng để giúp bạn dễ dàng quản lý và phát triển trò chơi của mình.



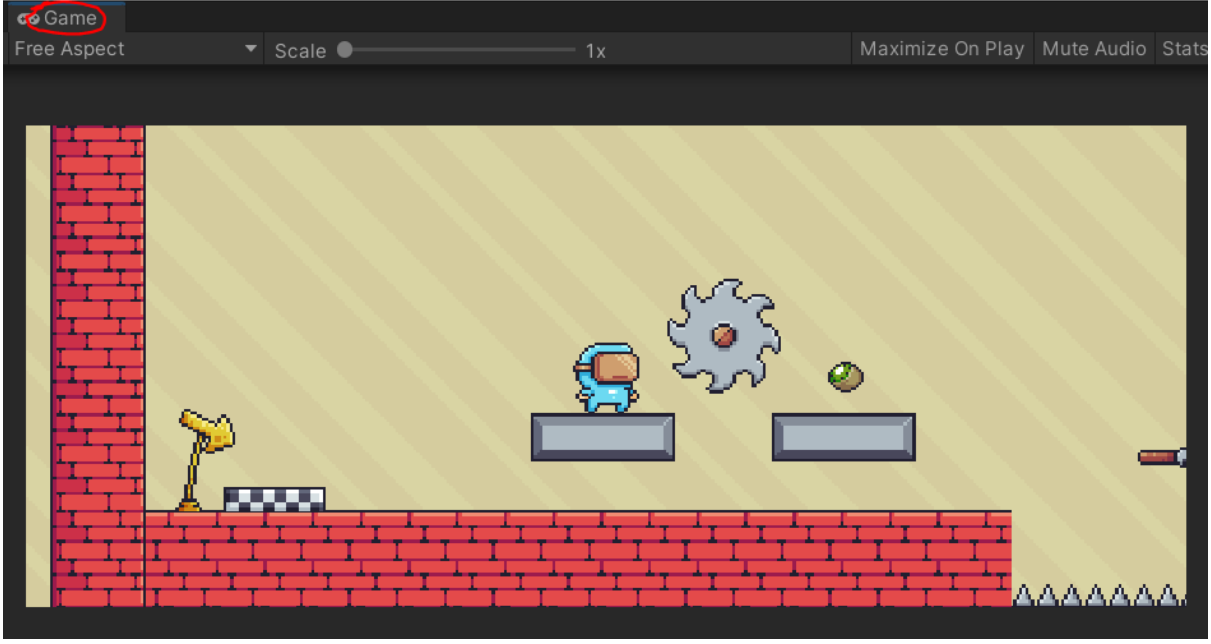
Hình 3.3: Tổng quan giao diện Unity

Scene View (Cửa sổ Hiển Thị Scene): Phần này phần hiển thị các đối tượng trong Scenes một cách trực quan, có thể lựa chọn các đối tượng, kéo thả, phóng to, thu nhỏ, xoay các đối tượng...Phần này có để thiết lập một số thông số như hiển thị ánh sáng, âm anh, cách nhìn 2D hay 3D... Khung nhìn Scene là nơi bố trí các Game Object như cây cối, cảnh quan,enemy, player, camera,. . . trong game. Sự bố trí hoạt cảnh là một trong những chức năngquan trọng nhất của Unity



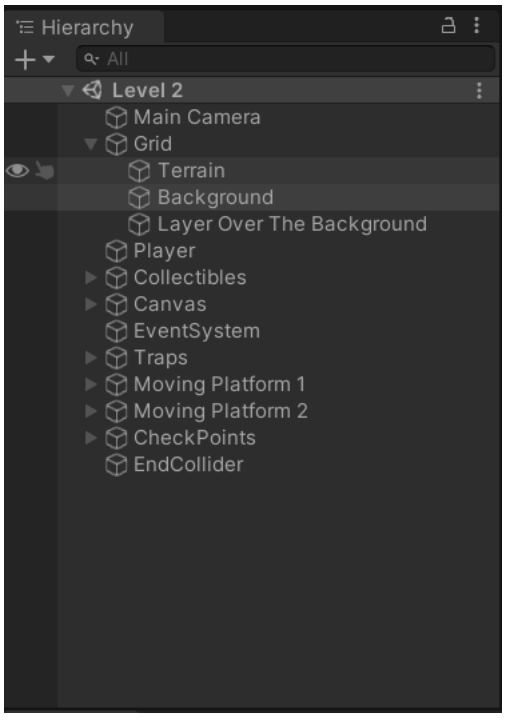
Hình 3.4: Scene View trong Unity

Game View (Cửa sổ Hiển Thị Game): Đây là cửa sổ mà bạn có thể xem trực tiếp trò chơi của mình khi đang phát triển. Điều này giúp bạn kiểm tra trực tiếp hiệu suất và hình ảnh của trò chơi.



Hình 3.5: Game View trong Unity

Hierarchy (Cây đối tượng): Tab Hierarchy là nơi hiển thị các Game Object trong Sences hiện hành. Khi các đối tượng được thêm hoặc xóa trong Sences, tương ứng với các đối tượng đó trong cửa sổ Hierarchy. Tương tự trong tab Project, Hierarchy cũng có một thanh tìm kiếm giúp quản lý và thao tác với các Game Object hiệu quả hơn đặc biệt là với các dự án lớn.



Hình 3.6: Hierarchy trong Unity

Project (Dự án): Trong cửa sổ này, bạn có thể thấy tất cả các tài nguyên của dự án, chẳng hạn như hình ảnh, âm thanh, các tệp mã nguồn, và nhiều thứ khác. Điều này giúp bạn quản lý tất cả các tài nguyên của trò chơi.

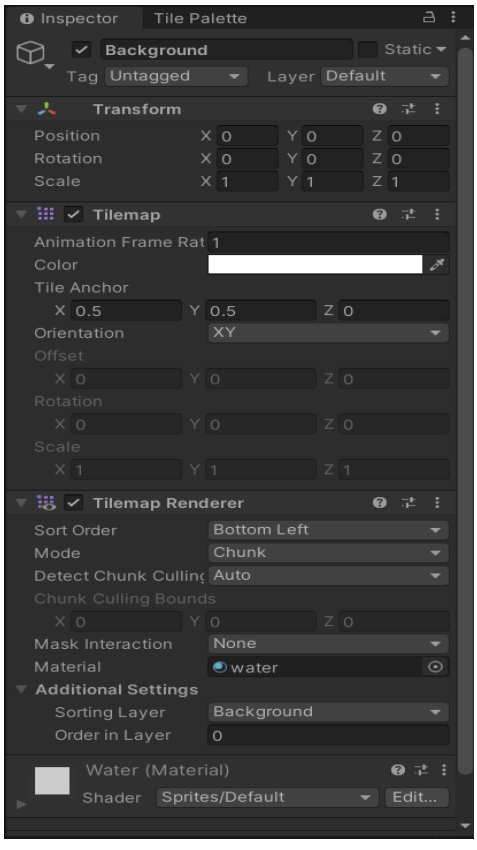
Inspector (Thanh kiểm tra): Cửa sổ Inspector hiển thị chi tiết các thông tin về Game Object đang làm việc, kể cả những component được đính kèm và thuộc tính của nó.

Bạn có thể điều chỉnh, thiết lập mọi thông số và chức năng của Game Object thông qua cửa sổ Inspector.

Mọi thuộc tính thể hiện trong Inspector đều có thể dễ dàng tuỳ chỉnh trực tiếp mà không cần thông qua một kịch bản định trước.

Các thiết lập của từng component được đặt trong menu. Các bạn có thể click chuột phải, hoặc chọn icon hình bánh răng nhỏ để xuất hiện menu.

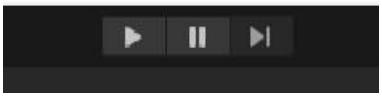
Ngoài ra Inspector cũng thể hiện mọi thông số Import Setting của asset đang làm việc như hiển thị mã nguồn của Script, các thông số animation,…



Hình 3.7: Inspector trong Unity

Toolbar (Thanh công cụ): Thanh công cụ nằm ở phía trên cùng của giao diện – cung cấp các công cụ và chức năng phổ biến như lưu dự án, phát triển trò chơi, và quản lý các sơ đồ.

Play Controls (Các nút điều khiển chơi): Bạn có thể nhấn vào các nút Play, Pause, và Stop để thực hiện việc chạy và kiểm tra trò chơi của mình trong Editor.



Hình 3.8: Play Controls trong Unity

3.2 Giới thiệu về ngôn ngữ C Sharp

C Sharp là một ngôn ngữ lập trình mạnh mẽ, linh hoạt và đa năng, thích hợp cho việc phát triển đa dạng loại ứng dụng, từ các ứng dụng máy tính đến game và ứng dụng di động. Ngôn ngữ Lập Trình Hướng Đối Tượng: C Sharp được thiết kế chủ yếu để hỗ trợ lập trình hướng đối tượng (OOP). Điều này có nghĩa là bạn có thể tạo các lớp, đối tượng, và tương tác giữa chúng để tổ chức và quản lý mã nguồn một cách hiệu quả.

Đa Dạng Ứng Dụng: C Sharp được sử dụng rộng rãi trong phát triển ứng dụng đa dạng, từ ứng dụng Windows cho máy tính, ứng dụng di động, ứnh dụng web, đến các ứng dụng trò chơi.

Sự An Toàn Về Kiểu Dữ Liệu: C Sharp có kiểm tra kiểu dữ liệu mạnh mẽ, giúp tránh các lỗi phổ biến liên quan đến kiểu dữ liệu trong quá trình biên dịch và thực thi.

Tích Hợp Tốt Với Công Cụ Phát Triển Microsoft: C Sharp được tích hợp mạnh mẽ với các công cụ phát triển của Microsoft như Visual Studio, giúp tạo ra môi trường phát triển mạnh mẽ và dễ dàng sử dụng.

Cú Pháp Đơn Giản: Cú pháp của C Sharp được thiết kế để dễ đọc và dễ hiểu, giúp người lập trình tập trung vào việc giải quyết vấn đề thay vì mất thời gian với cú pháp phức tạp.

Xử Lý Ngoại Lệ: C Sharp cung cấp cơ chế xử lý ngoại lệ để quản lý và xử lý các tình huống không mong muốn trong quá trình thực thi chương trình.

Phát Triển Ứng Dụng Game: C Sharp cũng được sử dụng rộng rãi trong phát triển game, đặc biệt là trong Unity, một môi trường phát triển trò chơi phổ biến.

Hỗ Trợ Cộng Đồng Lớn: C Sharp có một cộng đồng phát triển lớn và nhiều tài liệu học tập, giúp bạn dễ dàng tìm kiếm giải pháp cho các vấn đề cụ thể.

Hình 3.9: Hình ảnh Visual Studio

3.3 Giới thiệu Visual Studio

Visual Studio là một môi trường phát triển tích hợp (IDE) mạnh mẽ được phát triển bởi Microsoft. Nó cung cấp một loạt các công cụ và tài nguyên để hỗ trợ việc phát triển ứng dụng đa năng, bao gồm ứng dụng di động, web, máy tính và cả game. Visual Studio hỗ trợ Unity một cách rất mạnh mẽ. Unity là một nền tảng phát triển game phổ biến, và Visual Studio được tích hợp chặt chẽ với nó để cung cấp môi trường lập trình tốt nhất cho việc phát triển game trong Unity. Dưới đây là một số cách mà Visual Studio hỗ trợ Unity:

- **Tích Hợp Mượt Mà:** Khi bạn cài đặt Unity, bạn có thể dễ dàng liên kết nó với Visual Studio làm trình biên tập mã mặc định. Điều này giúp bạn mở các tệp mã và tài nguyên trong Unity bằng cách sử dụng Visual Studio.
-
- **Debugging Mạnh Mẽ:** Visual Studio cho phép bạn debug mã của mình trong Unity một cách hiệu quả. Bạn có thể thực hiện debugging theo dõi và xem biến, hàm và lớp trong quá trình thực thi game.
- **Tích Hợp Source Control:** Visual Studio hỗ trợ tích hợp các hệ thống quản lý phiên bản như Git cho dự án Unity của bạn. Điều này giúp bạn theo dõi lịch sử thay đổi, hợp nhất mã và quản lý phiên bản dễ dàng.
- **Công Cụ Unity:** Visual Studio cung cấp các công cụ đặc biệt dành cho Unity như Console, Animator và Hierarchy. Điều này giúp bạn dễ dàng đặt dấu chỗ lỗi, tìm kiếm và chỉnh sửa các thành phần trong Unity.
- **Tùy Chỉnh Môi Trường:** Có thể tùy chỉnh môi trường lập trình của mình trong Visual Studio để phù hợp với cách bạn làm việc với Unity.

3.4 Giới thiệu các công cụ quản lý mã nguồn

3.4.1 Git lap

GitLab là một nền tảng quản lý mã nguồn và dự án phân tán, được xây dựng trên nền tảng Git. Nó cung cấp môi trường hoàn chỉnh để quản lý, lưu trữ và hợp tác phát triển mã nguồn.

Hình 3.10: Hình ảnh Gitlap

Tính năng chính:

- Lưu trữ mã nguồn: Quản lý mã nguồn bằng hệ thống quản lý phiên bản Git.
- Quản lý dự án: Theo dõi tiến độ công việc, phân chia nhiệm vụ, quản lý vấn đề và báo cáo.
- CI/CD: Tích hợp tính năng Continuous Integration (CI) và Continuous De-ployment (CD) để tự động hóa kiểm tra và triển khai mã.
- Hợp tác nhóm: Cho phép nhiều người cùng làm việc trên cùng một dự án, với các tính năng như merge request và nhận xét mã.
- Quản lý vấn đề: Tạo và quản lý các vấn đề, báo cáo lỗi và yêu cầu tích hợp.

3.4.2 Source Tree

SourceTree là một ứng dụng giao diện đồ họa (GUI) dành cho việc quản lý mã nguồn và thao tác với hệ thống quản lý phiên bản như Git và Mercu

Hình 3.11: Hình ảnh Source Tree

Tính năng chính:

- Cung cấp giao diện trực quan để thực hiện các thao tác với mã nguồn, không cần sử dụng dòng lệnh.
- Xem lịch sử và thay đổi: Hiển thị lịch sử commit và thay đổi mã nguồn một cách rõ ràng.
- Quản lý nhánh: Tạo, chuyển đổi và hợp nhất các nhánh mã nguồn một cách dễ dàng.
- Giải quyết xung đột: Hỗ trợ giải quyết xung đột khi hợp nhất mã nguồn từ các nhánh khác nhau.
- Hợp tác nhóm: Cho phép thực hiện các thao tác hợp nhất và tương tác với các nền tảng như GitLab.

CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI HỆ THỐNG

4.1 Thiết kế trò chơi

4.1.1 Mô tả cách trò chơi hoạt động, quy tắc và cơ chế chơi.

Tên trò chơi: Game Test (Em chưa nghĩ ra tên ạ ☹)

Core loop: Play level - Colect Coin

Màn hình: Ngang.

Hệ điều hành: PC

Di chuyển: Sử dụng phím A, D để di chuyển sang trái phải, ấn Space để nhảy

Tấn công: Nhảy lên đầu kẻ địch để đánh bại chúng.

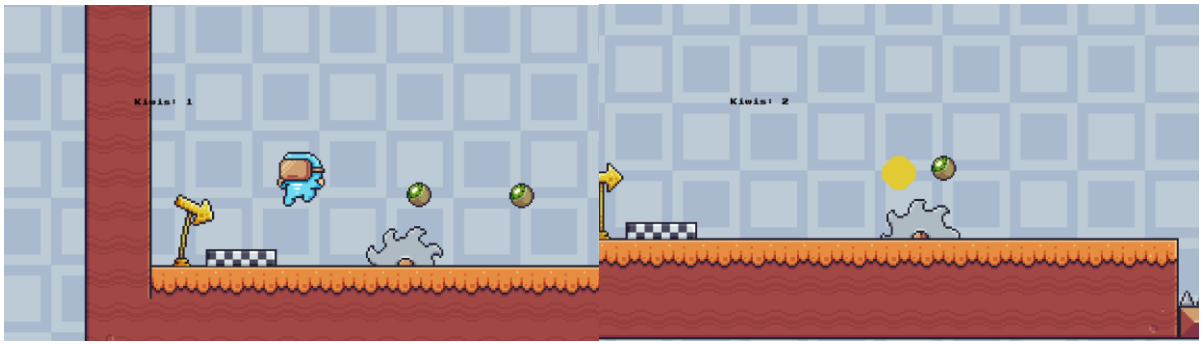
Tính điểm: Điểm tính theo số Kiwis lấy được.

Điều kiện chiến thắng: Người chơi đến được cổng chiến thắng ở điểm cuối màn chơi.

Điều kiện thua: Người chơi bị kẻ địch đánh bại, ngã xuông hố hoặc bị gai đâm.

Màn chơi: Xây dựng màn chơi, đầy đủ đối tượng chính, môi trường. Có điểm đầu điểm cuối.

4.1.2 Phác thảo giao diện người dùng



4.2 Phát triển trò chơi

4.2.1 Thiết kế nhân vật

Tên nhân vật	Mô tả
Người chơi	Một phi hành gia (?) màu xanh dương
Lưỡi cưa	Di chuyển phải, trái/ Trên, dưới trong một vùng nhật định.
Gai	Cố định 1 chỗ, người chơi sẽ hi sinh khi chạm vào
Ếch Dojo	Di chuyển trái phải, đánh người chơi khi vào gần
Lưỡi cưa nhỏ	Di chuyển phải trái với tốc độ cao

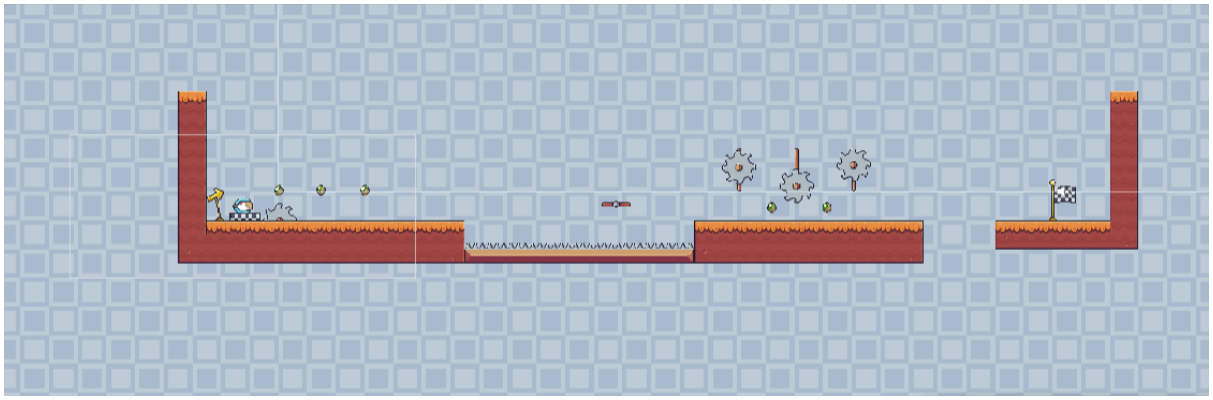
Bảng 4.1: Bảng mô tả các nhân vật

4.2.2 Thiết kế môi trường

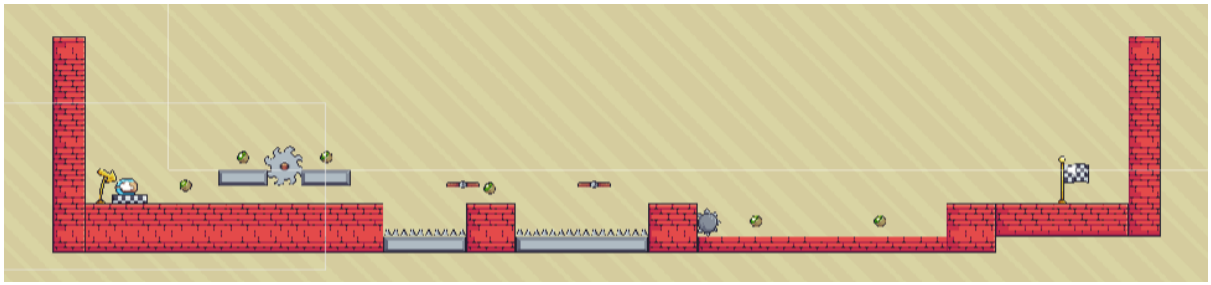
Tên môi trường	Mô tả
Background	Ảnh nền đằng sau
Ground	Gồm các ô đất. Người chơi sẽ di chuyển trên đây.
Bệ gỗ	Giúp người chơi di chuyển trên bản đồ
Cờ, vạch đích	Nơi bắt đầu/ kết thúc ván chơi

4.2.3 Thiết kế màn chơi

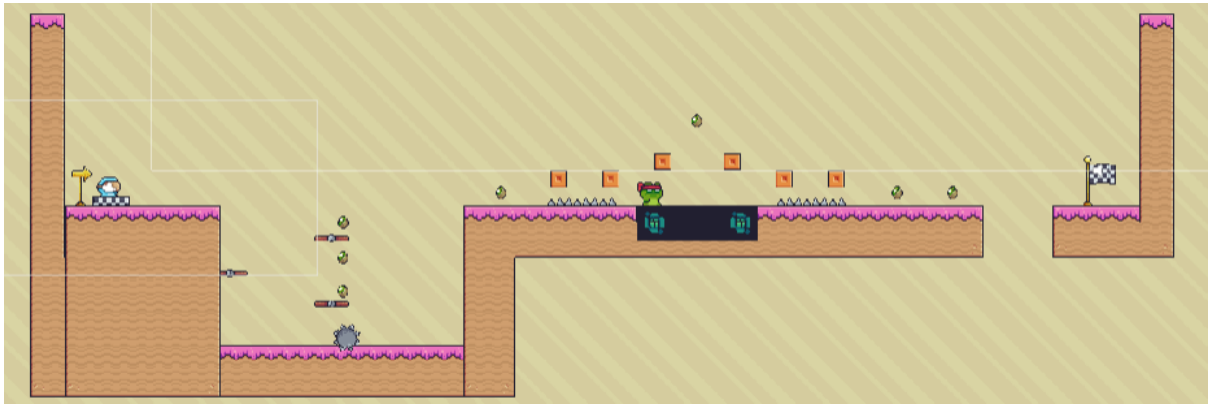
Lv 1:



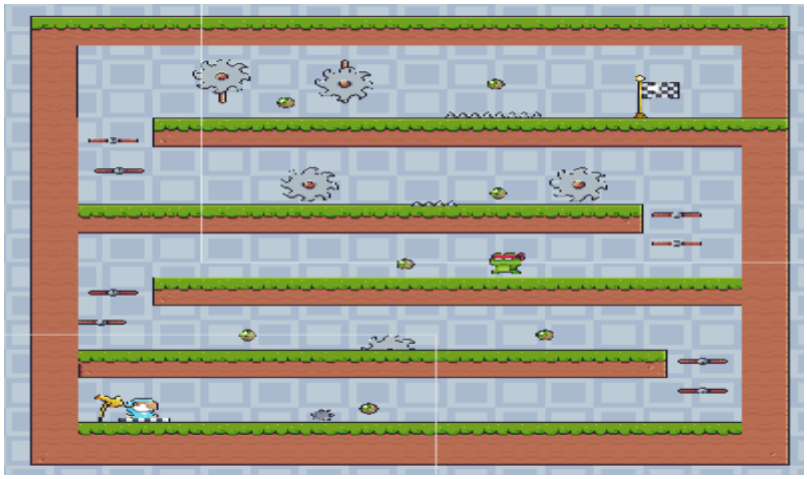
Lv 2



Lv 3:



Lv 4:



4.3 Kết quả đạt được

Xây dựng nhân vậT: Nhân vật có thể di chuyển trái phải và nhảy. Khi va chạm với vật cản sẽ dừng lại. Va vào địch hoặc chông/gai sẽ hi sinh.

Logic di chuyển của nhân vật: Đẩy một lực vào object nhân vật để nó di chuyển.

Logic nhảy: Kiểm tra xem Nhân vật có ở trên mặt đất không, nếu có thì sẽ được phép nhảy. Đẩy một lực từ dưới lên để nhân vật nhảy.

```
private void Update()
    {
        dirX = Input.GetAxisRaw("Horizontal");

        rb.velocity = new Vector2(dirX * 7f, rb.velocity.y);

        if (Input.GetKeyDown("space") && IsGrounded())
        {
            rb.velocity = new Vector3(rb.velocity.x,jumpForce);
        }
        UpdateAnimationState();
    }
```

Xử lý được việc Load hoạt ảnh nhân vật: Game 2D được xây dựng từ nhiều mô hình 2D được đặt lên không gian 2 chiều sao cho hài hòa 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 gắn chết một chuyển động vào bên trong và dễ dàng thay đổi chuyển động cho mô hình.

Giải pháp: Tạo ra một file script và gắn nó vào một đối tượng trong game bất kì để đoạn script có thể thức thi. Trong file script này, ta khai báo một đối tượng kiểu GameObject để lưu mô hình và dung hàm Instantiate() để khởi tạo mô hình này ở vị trí góc quay mong muốn. Sau đó Load mô hình từ Prefab chứa trong resource.

Xử lý được chuyển động nhân vật: 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. Tôi đã sử dụng cach sau: - Kỹ thuật keyframe Đối với kỹ thuật key frame, người ta sử dụng một sprite cho một keyframe của hành động. Để tạo ra chuyển động, ta sẽ vẽ một keyframe tại thời điểm đầu và thay đổi tuần tự các keyframe sau, chúng ta sẽ có được một animation. Đây là phương pháp đơn giản nhất để tạo chuyển động, nhưng lại tốn kém về bộ nhớ, vì ta phải tốn nhiều sprite cho nhiều chuyển động khác nhau.

```
private void UpdateAnimationState()
    {
        MovementState state;

        if (dirX > 0f)
        {
            state = MovementState.running;
            sprite.flipX = false;
        }
        else if (dirX < 0f)
        {
            state = MovementState.running;
            sprite.flipX = true;
        }
        else
        {
            state = MovementState.idle;
        }

        if (rb.velocity.y > .1f)
        {
            state = MovementState.jumping;
        }
        else if(rb.velocity.y < -.1f)
        {
            state = MovementState.falling;
        }
        anim.SetInteger("state", (int)state);

    }
```

Xử lý được va chạm: Colliders là vật mà engine vật lý sẽ dung để có thể nhận ra sự va chạm, không giống như các mesh, chúng nhận biết được mỗi khi

va chạm với nhau. Đa số collider có hình dạng đơn giản nhằm mục đích tính toán đơn giản và dễ dàng hơn, phần lớn các object trong Unity sẽ được gắn collider mỗi khi tạo ra, với Cube là Box Collider, Sphere là Sphere Collider, Cylinder là Capsule Collider. Unity cung cấp những API sau đâu để phát hiện sự va chạm của các collider. void OnCollisionEnter(Collision collision) – Chạy 1 lần tại thời điểm va chạm giữa 2 vật. void OnCollisionStay(Collision collision) – Chạy trong mỗi khung hình tại thời điểm 2 vật còn chạm vào nhau. void OnCollisionExit(Collision collision) – Chạy tại khung hình cuối cùng khi 2 vật không còn chạm vào nhau nữa. Với class Collision chúng ta có thể lấy ra những thuộc tính như: Contacts – điểm va chạm giữa 2 vật, tính bằng vector3. GameObject – game object va chạm với object gốc.

Xử lý Camera đi theo nhân vật: Di chuyển camera theo một đối tượng

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CameraController : MonoBehaviour
{

    [SerializeField] private Transform player;

    private void Update()
    {
        transform.position = new Vector3(player.position.x, player.position.y,
transform.position.z);
    }
}
```

Xử lý sự kiện Nhân Vật ăn Kiwis:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class ItemCollector : MonoBehaviour
{
    private int kiwis = 0;
    [SerializeField] private TextMeshProUGUI kiwisText;
```

```
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.CompareTag("kiwi"))
        {
            Destroy(collision.gameObject);
            kiwis++;
            kiwisText.text = "Kiwis: " +kiwis;
            PlayerPrefs.SetInt("Kiwis", kiwis);
        }

    }
}
```

Xử lý các Object khác:

Lưỡi cưa: Di chuyển 2 bên:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class WayPointFollower : MonoBehaviour
{
    [SerializeField] private GameObject[] waypoints;
    public int currentWaypointIndex = 0;
    [SerializeField] private float speed = 2f;

    private void Update()
    {
        if
(Vector2.Distance(waypoints[currentWaypointIndex].transform.position,
transform.position) < .1f)
        {
            currentWaypointIndex++;
            if (currentWaypointIndex >= waypoints.Length)
            {
            currentWaypointIndex = 0;
            }
        }
        transform.position = Vector2.MoveTowards(transform.position,
waypoints[currentWaypointIndex].transform.position, speed * Time.deltaTime);

    }
}
```

Level Select:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;


public class LevelSelect : MonoBehaviour
{
    public int noOfLevels;
    public GameObject levelButton;
    public RectTransform ParentPanel;
    public int levelReached;
    public int kiwis;

    private int index = 0;

    public int neededKiwis = 5;

    List<GameObject> buttons = new List<GameObject>();

    private void Awake()
    {
        LevelButtons();
    }



    void LevelButtons()
    {
        if (PlayerPrefs.HasKey("level"))
        {
            levelReached = PlayerPrefs.GetInt("level");
        }
        else
        {
            PlayerPrefs.SetInt("level", 1);
            levelReached = PlayerPrefs.GetInt("level");
        }

        for (int i = 0; i < noOfLevels; i++)
        {
            int x = new int();
            x = i + 1;
            GameObject lvlbutton = Instantiate(levelButton);
            lvlbutton.transform.SetParent(ParentPanel, false);
            Text buttontext = lvlbutton.GetComponentInChildren<Text>();
            buttontext.text = (i + 1).ToString();
```

```
lvlbutton.gameObject.GetComponent<Button>().onClick.AddListener(delegate
            {
                LevelSelected(x);
            });

            //if (i + 1 > levelReached)
            //{
            //    lvlbutton.GetComponent<Button>().interactable = false;
            //}

            buttons.Add(lvlbutton);
        }
    }




    void LevelSelected(int index)
    {
        PlayerPrefs.SetInt("levelSelected", index);
        Debug.Log("Level Selected: " + index.ToString());
        StartCoroutine(LoadDelay(index));
    }

    private IEnumerator LoadDelay(int index)
    {
        yield return new WaitForSeconds(2f);
        LoadLevels(index);
    }

    void LoadLevels(int index)
    {
        SceneManager.LoadScene("Level " + index);
    }
```

Platform Gỗ (Dính nhân vật cố định lên platform khi nó di chuyển)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StickyPlatform : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D collision)
    {
        if (collision.gameObject.name == "Player")
```

```
        {
            collision.gameObject.transform.SetParent(transform);
        }
    }

    private void OnTriggerExit2D(Collider2D collision)
    {
        if (collision.gameObject.name == "Player")
        {
            collision.gameObject.transform.SetParent(null);
        }
    }
}
```