BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN NGÀNH

NGÀNH MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU

Đề tài

XÂY DỰNG GAME THU THẬP TRÁI CÂY DỰA TRÊN NỀN TẢNG UNITY 3D KẾT HỢP THUẬT TOÁN A* PATHFIDING

Sinh viên thực hiện : Hồ Anh Kiệt

Mã số : B2004731

Khóa : 46

HK 2 NH 2023 – 2024

Cần Thơ, 5/2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CẦN THƠ

TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

NIÊN LUẬN NGÀNH

NGÀNH MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG DỮ LIỆU

Đề tài

XÂY DỰNG GAME THU THẬP TRÁI CÂY DỰA TRÊN NỀN TẢNG UNITY 3D KẾT HỢP THUẬT TOÁN A* PATHFIDING

Giảng viên hướng dẫn
ThS. Bùi Minh Quân

Sinh viên thực hiện
Hồ Anh Kiệt
Mã số: B2004731
Khoá: 46

HK 2 NH 2023 - 2024

Cần Thơ, 5/2024

NHẬN XÉT CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Đầu tiên, em muốn bày tỏ lòng biết ơn sâu sắc đến Trường Công nghệ Thông tin & Truyền Thông vì đã tạo điều kiện học tập tốt nhất cho em. Đặc biệt, em xin gửi lời cảm ơn chân thành đến Thầy, Cô đã chia sẻ những kiến thức quý báu và truyền đạt những lời khuyên chân thành. Trong quãng thời gian học tập, em đã tích luỹ được nhiều kiến thức bổ ích, là nguồn động viên quan trọng giúp em vững bước trên con đường học nghiên cứu sắp tới.

Em không quên bày tỏ lòng biết ơn sâu sắc đến Thầy Bùi Minh Quân, người đã cho em một cơ hội để thử sức trải nghiệm một đề tài mới lạ và là người tận tâm hỗ trợ và hướng dẫn em trong suốt quá trình thực hiện bài niên luận này. Mặc dù em đã nỗ lực hết mình, nhưng với vốn kiến thức còn hạn chế đặt biết là các nguyên lý hoạt động, bài niên luận không tránh khỏi những khuyết điểm. Em kính mong Thầy xem xét và góp ý, để em có cơ hội rút kinh nghiệm và hoàn thiện bản thân hơn trong những công việc nghiên cứu sắp tới.

Cuối cùng, em xin chân thành cảm ơn tất cả sự hỗ trợ và đóng góp, mang đến cho em những trải nghiệm quý giá. Lời cảm ơn này không chỉ là sự biểu hiện của lòng biết ơn cá nhân mà còn là lời cam kết về việc áp dụng những kiến thức và kỹ năng đã học để phát triển bản thân và đóng góp cho xã hội.

Em xin chân thành cảm ơn!

Cần Thơ, ngày 8 tháng 5 năm 2024

Người viết

Họ tên sinh viên

Hồ Anh Kiệt

MỤC LỤC

DANH MỤC HÌNH

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Trong ngành trò chơi điện tử, việc tạo ra trải nghiệm chơi game độc đáo và thú vị luôn là một thách thức. Trò chơi thu thập trái cây là một thể loại phổ biến, nơi người chơi phải vượt qua các chướng ngại vật và đối mặt với kẻ thù để đạt được mục tiêu của mình. Trong những trò chơi này, nhân vật người chơi chỉ cần tránh kẻ thù và thu thập trái cây. Sử dụng trí tuệ nhân tạo và học máy để kiểm soát hành vi của kẻ thù có thể là giải pháp tạo ra trải nghiệm chơi game đặc biệt và đầy thử thách.

2. Lịch sử giải quyết vấn đề

Trong quá khứ, việc lập trình hành vi của các kẻ địch trong trò chơi thường dựa vào các quy tắc cố định và điều kiện đơn giản. Điều này thường dẫn đến việc các kẻ địch trở nên dễ dàng đoán và ít sáng tạo. Tuy nhiên, với sự tiến bộ của trí tuệ nhân tạo và học máy, các nhà phát triển đã bắt đầu áp dụng các phương pháp này vào việc tạo ra hành vi của các kẻ địch trong trò chơi. Sử dụng các thuật toán khác nhau, các kẻ địch có khả năng thích nghi với hành vi của người chơi đã trở thành một thực tế.

3. Mục tiêu đề tài

Mục tiêu của dự án này là phát triển xây dựng một trò chơi điện tử mà người chơi sẽ phải né tránh các đối thủ và thu thập trái cây. Người chơi chỉ cần sử dụng kỹ năng di chuyển và chiến thuật của mình để hoàn thành mỗi cấp độ. Và nghiên cứu đưa ra các nguyên lý của thuật toán đó

4. Đối tượng và phạm vi nghiên cứu

Nghiên cứu sẽ tập trung vào việc xây dựng phát triển và tích hợp thuật toán vào trò chơi thu thập trái cây trong môi trường Unity 3D. Và nghiên cứu nguyên lý AI với thuật toán A*Pathfinding,nhân vật chơi chỉ cần di chuyển để tránh kẻ địch và thu thập trái cây.

5. Phương pháp nghiên cứu

Thu thập dữ liệu: Trong giai đoạn này, bạn sẽ thu thập dữ liệu về vị trí và hành động của nhân vật chơi và các kẻ địch trong quá trình chơi game. Dữ liệu này có thể bao gồm thông tin về bản đồ, các vật thể trong trò chơi, vị trí xuất phát và đích đến của nhân vật, và các ràng buộc về di chuyển.

Tích hợp vào trò chơi: Sau khi thu thập dữ liệu, bạn sẽ tích hợp thuật toán A* Pathfinding vào trò chơi. Điều này có thể bao gồm việc xây dựng một hệ thống đồ thị hoặc lưới cho bản đồ của trò chơi và triển khai thuật toán A* để tính toán đường đi tối ưu giữa các điểm trong bản đồ.

Đánh giá và điều chỉnh: Cuối cùng, bạn sẽ đánh giá hiệu suất của thuật toán A* Pathfinding trong trò chơi của mình. Điều chỉnh có thể được thực hiện để cải thiện hiệu suất hoặc tối ưu hóa các tham số của thuật toán, như đánh giá lại các hàm heuristic hoặc tối ưu hóa cách sắp xếp các nút trong quá trình tìm kiếm.

6. Kết quả đạt được

Triển khai thuật toán A* Pathfinding: Phát triển và triển khai thuật toán A* Pathfinder trong trò chơi để điều khiển hành vi của các kẻ địch trong môi trường game.

Tích hợp vào trò chơi: Tích hợp thuật toán A* Pathfinding vào trò chơi và kiểm tra tính thực tế và tính chính xác của hành vi. Đảm bảo rằng các kẻ địch có thể di chuyển một cách thông minh và tìm đường đi ngắn nhất để tiếp cận nhân vật chính.

Đánh giá và điều chỉnh: Đánh giá hiệu suất và sự thành công của thuật toán A* Pathfinding trong trò chơi. Điều chỉnh có thể được thực hiện để cải thiện hiệu suất hoặc tối ưu hóa các tham số của thuật toán, như điều chỉnh các hàm heuristic hoặc tinh chỉnh cách xử lý các tình huống đặc biệt.

7. Bố cục luận văn

Phần giới thiệu: Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1 : Mô tả bài toán

Chương 2 : Thiết kế, cài đặt giải thuật, biểu diễn các dữ liệu, trình bày các bước xây dựng ứng dụng.

Chương 3 : Kiểm thử hệ thống và đánh giá độ chính xác, tốc độ của hệ thống.

Phần kết luận: Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG

CHƯƠNG 1

MÔ TẢ BÀI TOÁN

1. Mô tả chi tiết bài toán

Trong trò chơi đang phát triển, người chơi sẽ nhập vai vào một nhân vật trong một môi trường đầy trái cây và kẻ địch. Nhiệm vụ của người chơi là thu thập tất cả các trái cây trên màn hình mà không bị kẻ địch tấn công. Nhân vật chỉ có thể di chuyển qua lại trên màn hình và phải né tránh các kẻ địch để không mất mạng.

2. Vấn đề và giải pháp liên quan đến bài toán

2.1. Điều khiển hành vi của kẻ địch

2.1.1. Điều khiển đường di chuyển của kẻ địch

Kẻ địch cần có khả năng tự động di chuyển trên màn hình một cách linh hoạt, tự định hướng và tìm đường để tiếp cận nhân vật chơi một cách thông minh.

2.1.2. Phản ứng của kẻ địch khi gặp nhân vật chơi

Khi kẻ địch gặp nhân vật chơi, cần phải có một hệ thống logic để quyết định hành động tiếp theo của kẻ địch, bao gồm việc tấn công hoặc tránh xa nhân vật.

2.2 Tối ưu hoá di chuyển của nhân vật chơi trong các địa hình

Nhân vật chơi cần phải có chiến thuật di chuyển thông minh để né tránh các kẻ địch một cách hiệu quả và thu thập trái cây an toàn. Đồng thời, bản đồ cần được thiết kế sao cho phù hợp với các chướng ngại vật, tạo ra các tình huống thách thức cho người chơi. Giải pháp cho vấn đề này bao gồm việc áp dụng thuật toán A* Pathfinding cho nhân vật chính và thiết kế bản đồ một cách cẩn thận để tạo ra trải nghiệm chơi game hấp dẫn.

3. Mô tả giải pháp cho bài toán

A* Pathfinding là một thuật toán tìm đường đi trong lập trình và trí tuệ nhân tạo. Nó áp dụng trong nhiều lĩnh vực, bao gồm:

Trò chơi điện tử: Trong game, A* Pathfinding được sử dụng để điều khiển các nhân vật hoặc đối tượng trong môi trường 3D hoặc 2D. Nó giúp định tuyến đường đi từ vị trí hiện tại của nhân vật đến mục tiêu mà không va chạm với các vật cản.

Robots di động: Trong robot hướng dẫn hoặc tự động hóa, A* Pathfinding được sử dụng để tính toán đường đi an toàn và hiệu quả trong một môi trường không gian 3D hoặc 2D.

Định tuyến trong mạng lưới: A* Pathfinding cũng có thể được sử dụng để tìm đường đi tối ưu giữa các nút trong mạng lưới, ví dụ như tìm đường đi ngắn nhất từ một điểm đến một điểm khác trên bản đồ đường phố.

Trình diễn hình ảnh và đồ họa: Trong xử lý hình ảnh, A* Pathfinding có thể được sử dụng để tìm đường đi tối ưu giữa các điểm trên một bức tranh, chẳng hạn như tìm đường ngắn nhất giữa hai điểm trên một bức ảnh.

Tính toán đường đi trong thời gian thực: A* Pathfinding có thể được sử dụng để tính toán đường đi trong thời gian thực cho các ứng dụng như điều khiển robot hoặc xe tự lái.

⇨ **Như vậy để giải quyết các vấn đề cũng như đề tài trên,AI được áp dụng trên trò chơi điện tử.Phương pháp sử dụng AI là sử dụng thuật toán A* Pathfinding cho đồi thủ trong trò chơi. Điều này đảm bảo rằng cả hai bao gồm đối tượng người chơi và đối thủ đều có thể di chuyển một cách thông minh và hiệu quả trong môi trường game. Đồng thời, cần thiết kế một hệ thống logic điều khiển cho cả hai để đảm bảo tính thực tế và tính chất thách thức của trò chơi.**

CHƯƠNG 2
THIẾT KẾ VÀ CÀI ĐẶT

1. Thiết kế hệ thống

Để xây dựng một trò chơi điện tử hoàn chỉnh và thú vị, ta cần phải có một thiết kế hệ thống chặt chẽ và chi tiết. Trong thiết kế này, mỗi thành phần của trò chơi sẽ được định nghĩa rõ ràng và tương tác với nhau một cách mạch lạc. Dưới đây là các phần chi tiết của thiết kế hệ thống:

+ **Môi trường trò chơi:**

  Môi trường trò chơi là nền tảng để tất cả các sự kiện diễn ra. Nó bao gồm các ô vuông trên màn hình, đại diện cho vị trí của nhân vật và các trái cây.

  Môi trường cũng chứa các đối tượng kẻ địch và các chướng ngại vật khác, tạo ra một không gian sống động và nguy hiểm.

+ **Nhân vật chơi:**

  Nhân vật chơi là người chơi điều khiển và tương tác với môi trường. Nó có khả năng di chuyển và thu thập trái cây.

  Nhân vật cũng có các thuộc tính như điểm số, mạng sống.

+ **Kẻ địch (đối thủ):**

  Kẻ địch là các đối tượng trong trò chơi có thể gây nguy hiểm hoặc thách thức cho nhân vật chính có sử dụng AI, có thể là các loại quái vật hoặc các chướng ngại vật động.

  Hành vi của kẻ địch có thể bao gồm việc di chuyển tấn công nhân vật chính.

+ **Hệ thống điểm số và tiến bộ**

  Hệ thống điểm số và tiến bộ là một phần quan trọng để thúc đẩy sự tiến triển của người chơi trong trò chơi. Trong trò chơi này, điểm số có thể được ghi nhận dựa trên số lượng trái cây thu thập được hoặc thời gian hoàn thành mỗi cấp độ. Tiến bộ của người chơi có thể được đo lường thông qua việc hoàn thành các mục tiêu cụ thể hoặc mở khóa các cấp độ mới.

+ **Các yếu tố khác:**

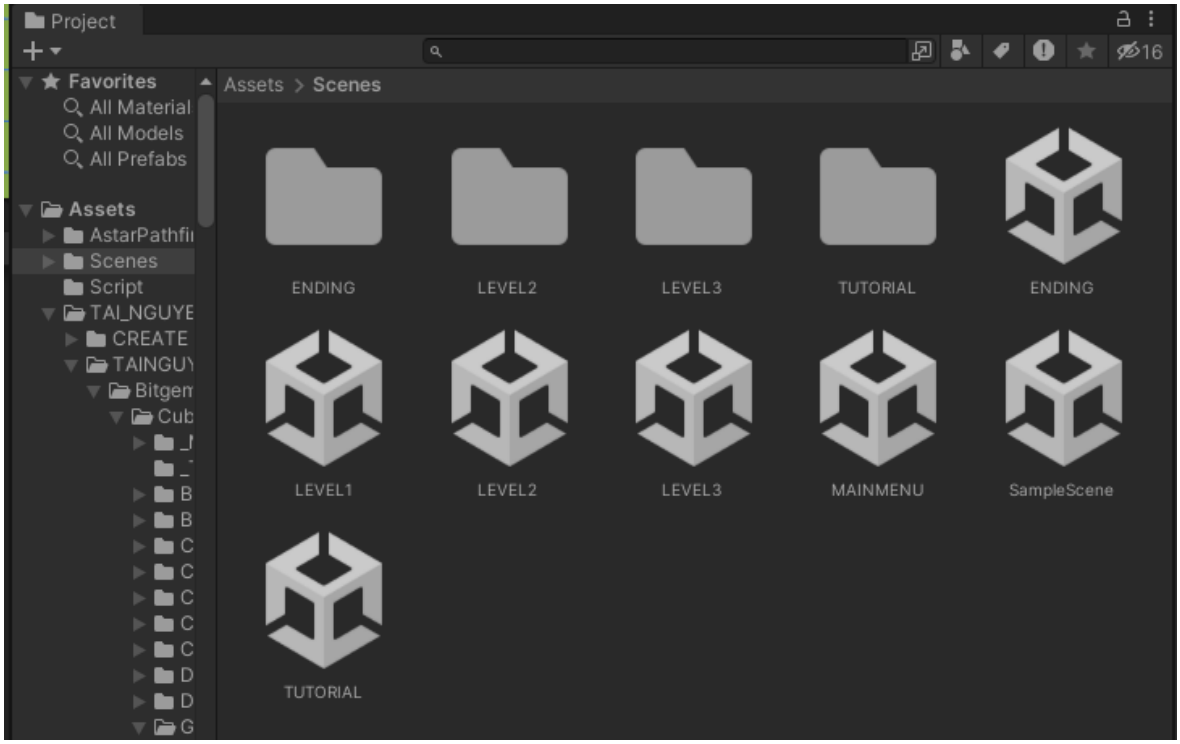
  Ngoài các thành phần chính như môi trường, nhân vật chơi và kẻ địch, còn có nhiều yếu tố khác cần được xem xét như hệ thống giao diện người dùng, âm thanh và hiệu ứng, đồ họa,…. ,tạo sự đa dạng và thú vị cho trò chơi.

2. Thiết kế và cài đặt giải thuật

2.1. Thiết kế bằng công cụ Unity 3D và các khái niệm.

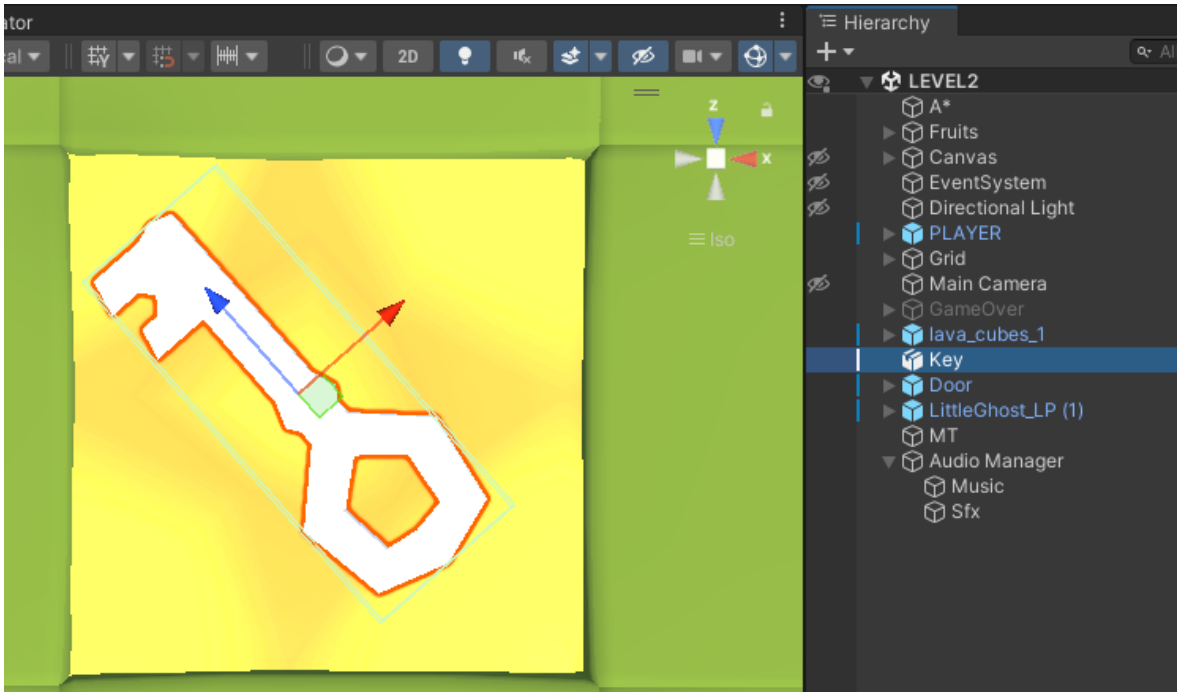
Unity 3D là một công cụ phổ biến được sử dụng rộng rãi trong phát triển trò chơi, đặc biệt là trò chơi 3D. Được biết đến với sự linh hoạt và tiện ích, Unity 3D không chỉ giúp nhà phát triển tạo ra các trò chơi hấp dẫn mà còn cho phép họ dễ dàng triển khai trên nhiều nền tảng khác nhau. Dưới đây là một số khái niệm cơ bản khi làm việc với Unity 3D:

Scenes (Cảnh): Trong Unity, mỗi màn chơi hoặc phần của trò chơi được biểu diễn bằng một Scene. Mỗi Scene có thể chứa nhiều GameObject và định nghĩa cấu trúc, cảnh quan và các tương tác giữa các đối tượng trong trò chơi. Việc sử dụng nhiều Scene giúp quản lý dễ dàng và tách biệt các phần khác nhau của trò chơi, từ màn hình menu đến các cấp độ chơi.



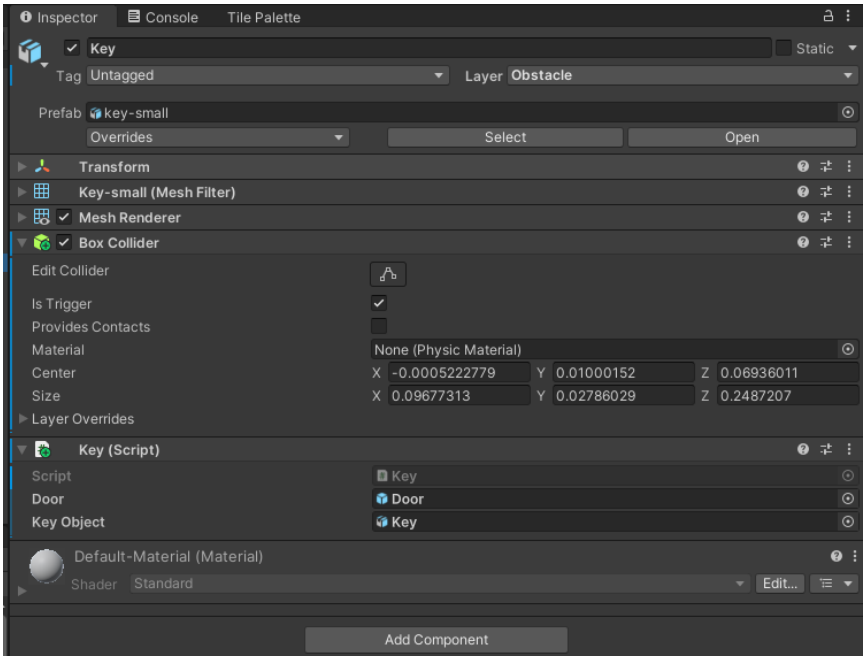
Hình 1: Các scenes trong project unity

GameObject (Đối tượng Game): Là cốt lõi của mỗi trò chơi trong Unity. Một GameObject có thể là bất kỳ đối tượng nào trong trò chơi, bao gồm nhân vật, vật phẩm, cảnh quan, ánh sáng và các đối tượng tương tác khác. Mỗi GameObject có các thành phần (Components) như Transform (định vị, xoay và tỷ lệ), Collider (xác định va chạm), Renderer (hiển thị hình ảnh) và Script (điều khiển hành vi).



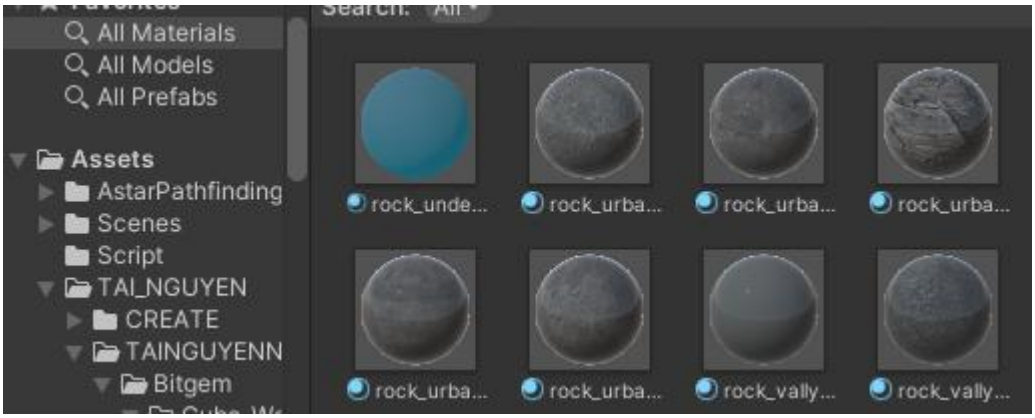
Hình 2: Các đối tượng trong unity

Components (Các thành phần): Là các phần tử mở rộng chức năng của GameObject trong trò chơi. Mỗi Component thực hiện một nhiệm vụ cụ thể, như điều khiển chuyển động, xử lý va chạm, hoặc thậm chí là kết nối với mạng. Việc kết hợp các Components khác nhau vào một GameObject cho phép tạo ra các đối tượng phong phú và đa dạng.



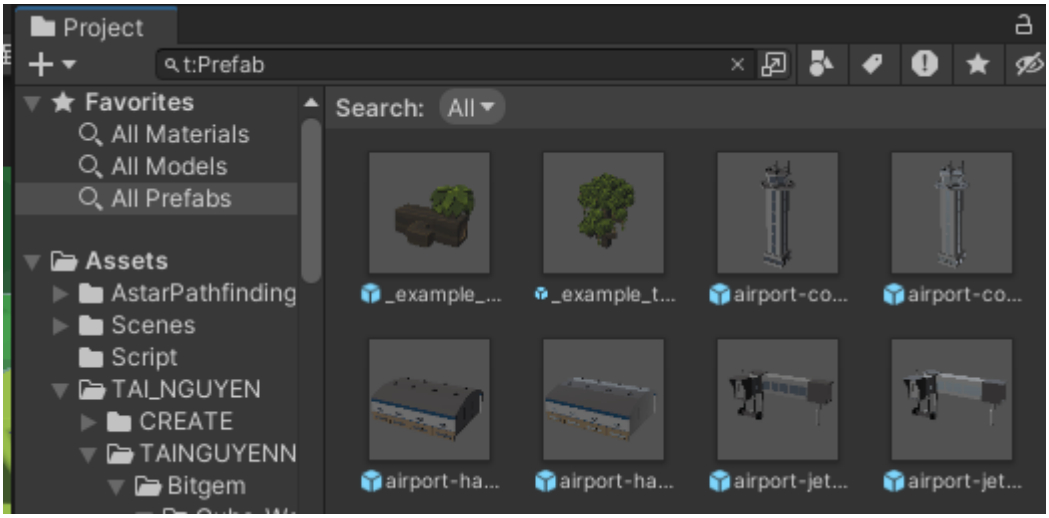
Hình 3: Các thành phần trong đối tượng Key

Materials (Vật liệu): Là các thành phần được sử dụng để định nghĩa các thuộc tính của bề mặt của đối tượng trong trò chơi. Vật liệu có thể xác định màu sắc, ánh sáng, texture, và các hiệu ứng đặc biệt như phản xạ và làm mờ. Sử dụng vật liệu phù hợp giúp tạo ra các đối tượng có vẻ ngoài hấp dẫn và chân thực hơn.



Hình 4: Material

Prefabs (Mẫu đối tượng): Là các đối tượng được lưu trữ dưới dạng mẫu để tái sử dụng trong trò chơi. Prefabs giúp tiết kiệm thời gian và công sức trong việc tạo ra các đối tượng giống nhau hoặc có cùng chức năng, bằng cách tạo một phiên bản và sử dụng lại nó nhiều lần trong trò chơi.



Hình 5: Các mẫu đối tượng

Scripting (Viết mã): Unity sử dụng ngôn ngữ lập trình C# để viết mã điều khiển hành vi của trò chơi. Bằng cách viết Script, bạn có thể điều khiển chuyển động, va chạm, hoạt động của các đối tượng trong trò chơi. Scripting là một phần quan trọng trong việc tạo ra các trò chơi đa dạng và phong phú trong Unity 3D.

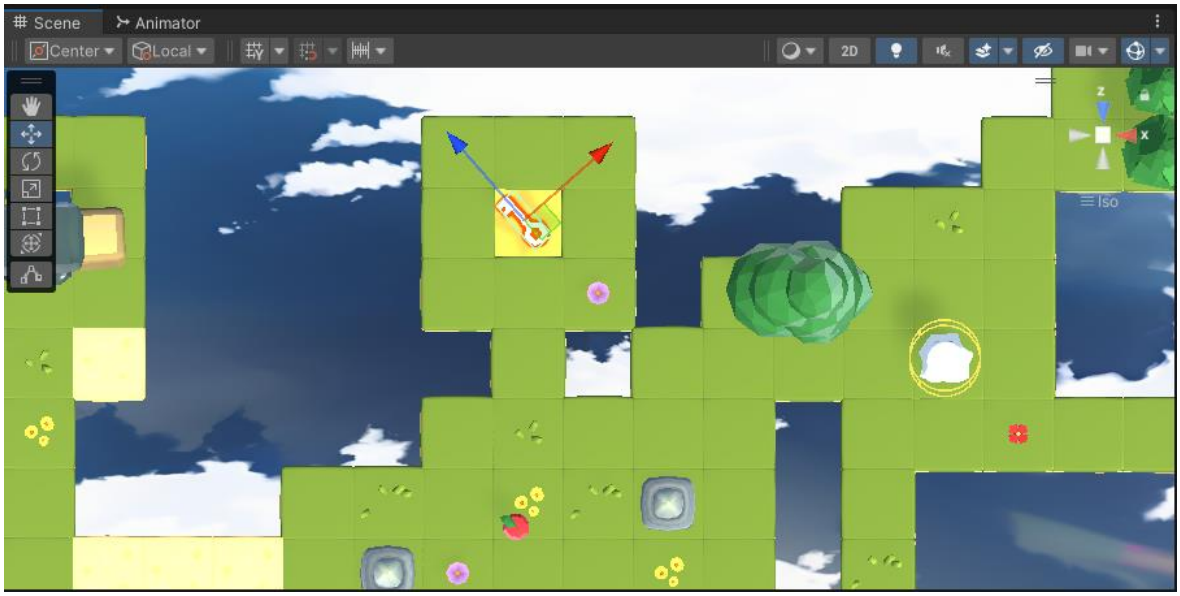
Hình 6: Tạo các script cho các đối tượng phù hợp

2.2. Giao diện của Unity

Unity cung cấp một giao diện người dùng trực quan và dễ sử dụng, giúp người dùng quản lý và tương tác với các thành phần của dự án một cách hiệu quả. Dưới đây là các phần chính của giao diện:

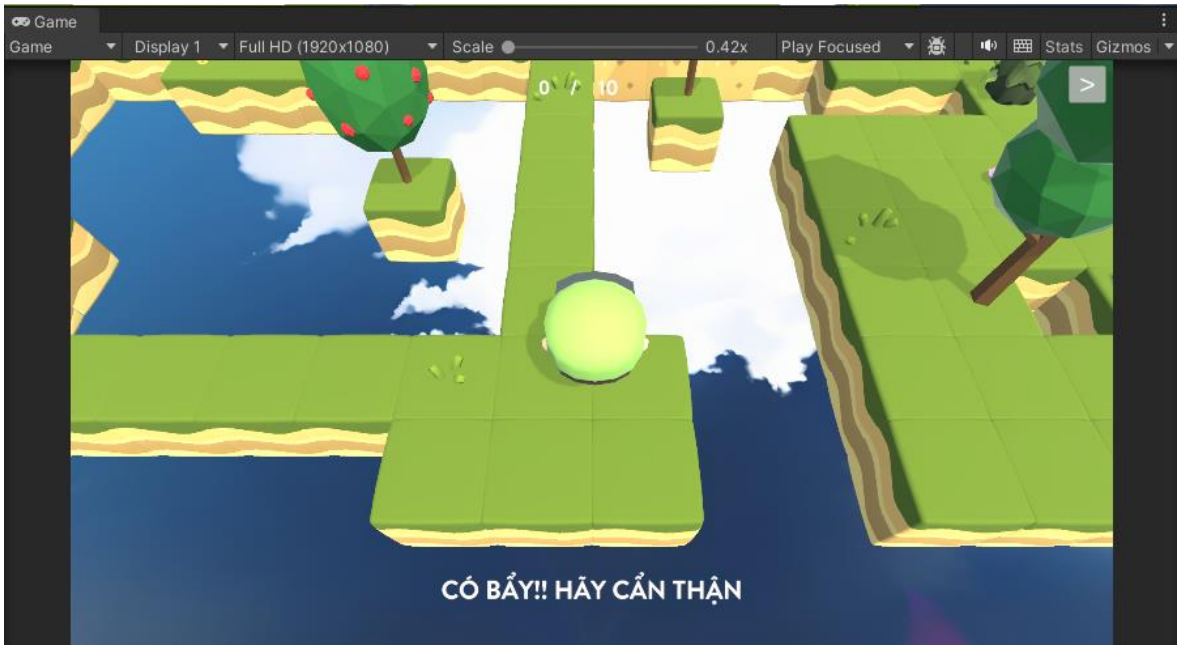
Khung làm việc (Work Area): Đây là phần trung tâm của giao diện, nơi thực hiện mọi công việc liên quan đến phát triển trò chơi. Khung làm việc bao gồm cả cửa sổ Scene, Game, và cửa sổ xem 3D hoặc 2D. Trong đó, cửa sổ Scene cho phép chỉnh sửa và xem các đối tượng trong không gian 3D hoặc 2D, còn cửa sổ Game hiển thị trực tiếp trò chơi đang chạy trong thời gian thực cụ thể hơn:

Cửa sổ Scene (Scene View): Hiển thị Scene hiện tại của dự án. Ở đây, người dùng có thể thao tác với các đối tượng trong Scene bằng cách di chuyển, quay, và thay đổi kích thước chúng, cũng như thêm, xóa và chỉnh sửa các đối tượng. Cửa sổ Scene cung cấp một cái nhìn tổng quan và cho phép người dùng tùy chỉnh trò chơi của mình trong không gian 3D hoặc 2D.



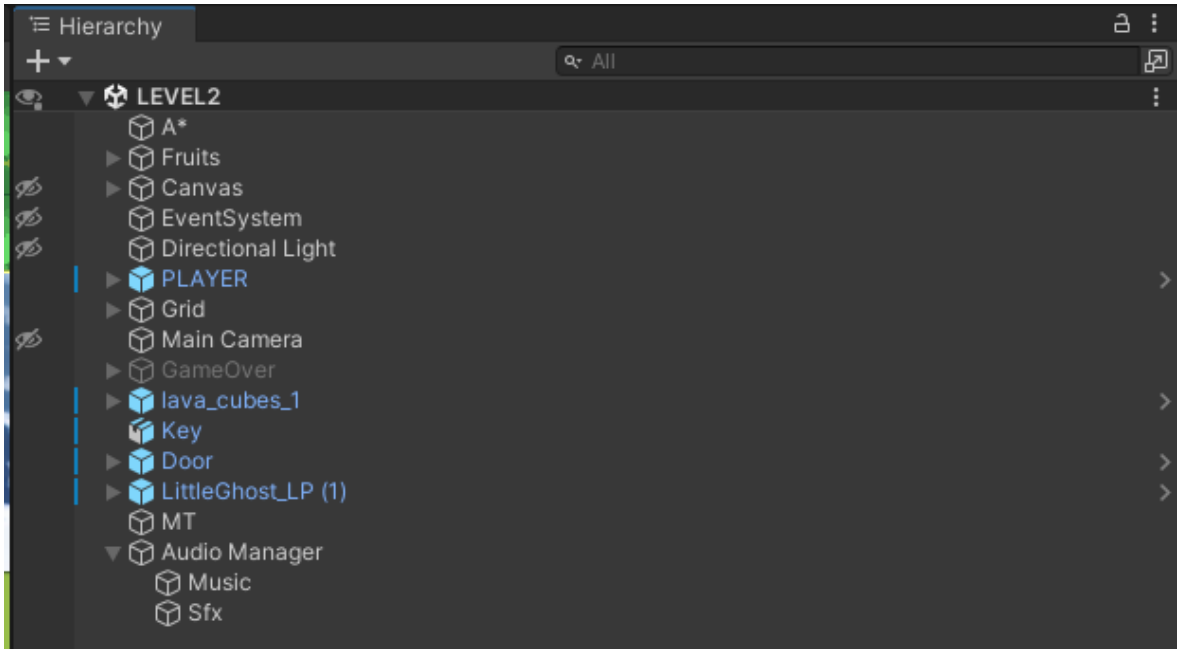
Hình 7: Giao diện cửa sổ Scene

Cửa sổ Game (Game View): Hiển thị trò chơi đang chạy trong thời gian thực. Điều này cho phép người dùng kiểm tra và thử nghiệm trò chơi của mình trong môi trường thực tế để đảm bảo tính ổn định và trải nghiệm chơi game tốt nhất.



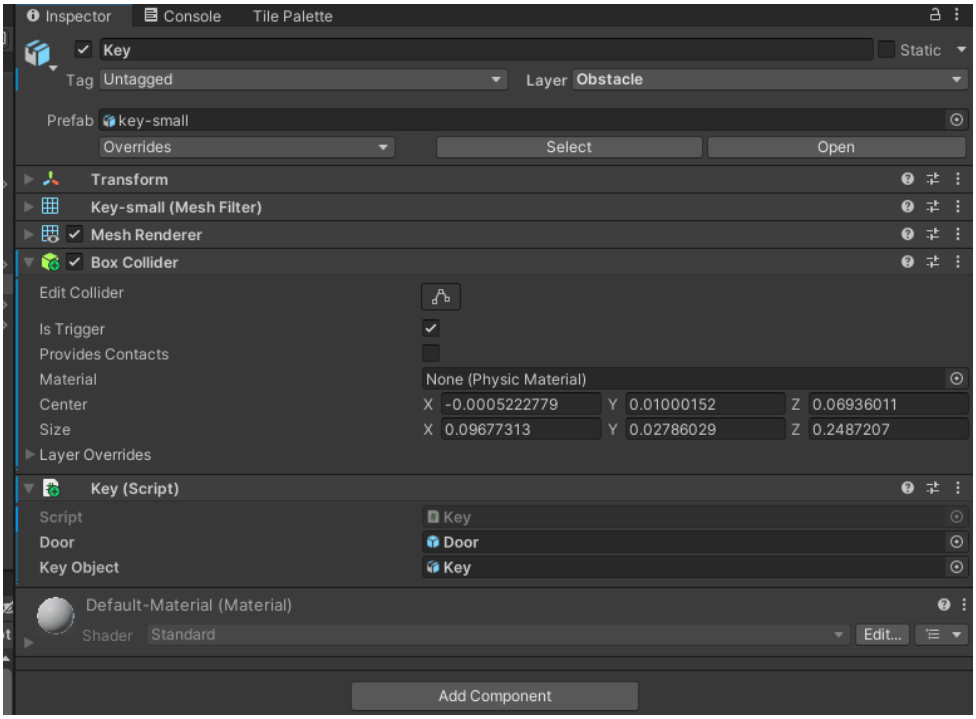
Hình 8: Giao diện cửa sổ Game

Cửa sổ Hierarchy: Đây là nơi hiển thị cấu trúc của tất cả các GameObject trong Scene hiện tại dưới dạng cây. Mỗi GameObject được liệt kê với tên và biểu tượng, cho phép người dùng dễ dàng tìm kiếm, chọn và quản lý chúng. Các hành động như tạo mới, xóa và sắp xếp các đối tượng trong cây Hierarchy cũng có thể được thực hiện từ đây.



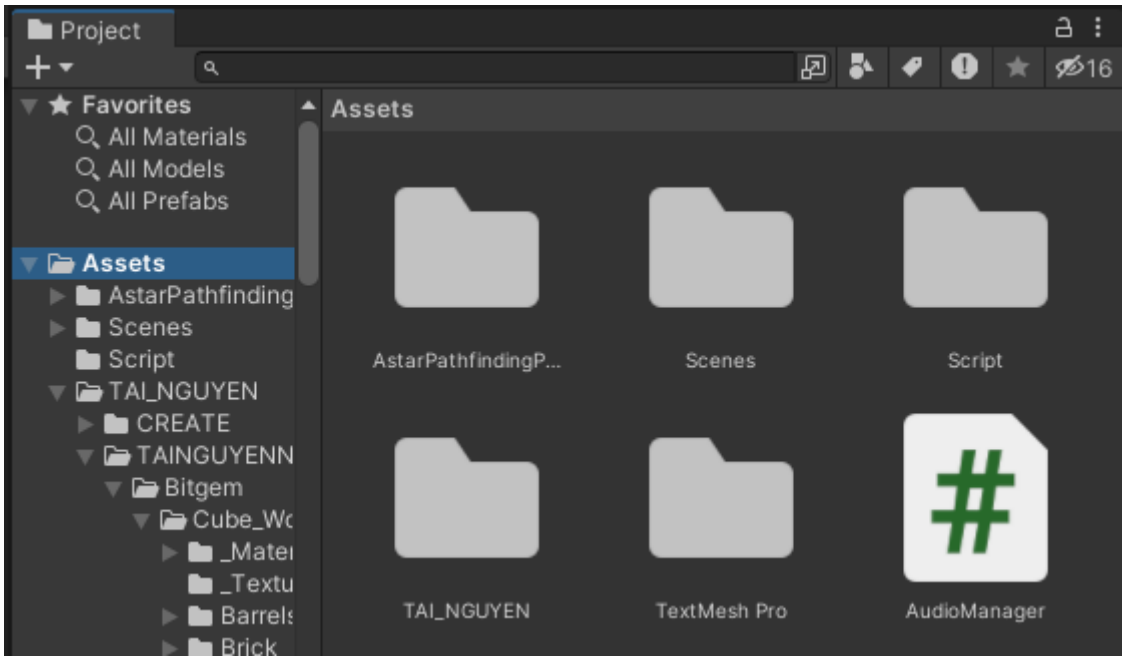
Hình 9: Giao diện cửa sổ Hierarchy

Cửa sổ Inspector: Hiển thị các thuộc tính và thành phần của đối tượng đang được chọn. Người dùng có thể tùy chỉnh và điều chỉnh các thuộc tính của GameObject và các thành phần của nó, bao gồm Transform, Renderer, Collider, và các thành phần tùy chỉnh khác. Cửa sổ này cung cấp một cách tiện lợi để thay đổi các giá trị và tùy chỉnh đối tượng một cách chi tiết.



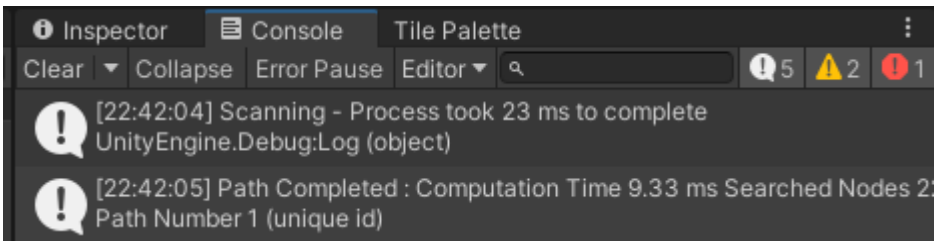
Hình 10: Giao diện cửa sổ Inspector

Cửa sổ Project: Hiển thị toàn bộ cấu trúc thư mục của dự án, bao gồm các tài nguyên như hình ảnh, âm thanh, video và Script. Người dùng có thể dễ dàng tìm kiếm, quản lý và sắp xếp các tài nguyên của dự án trong cửa sổ này. Các thao tác như tạo mới, xóa và di chuyển tệp tin cũng có thể được thực hiện từ đây.



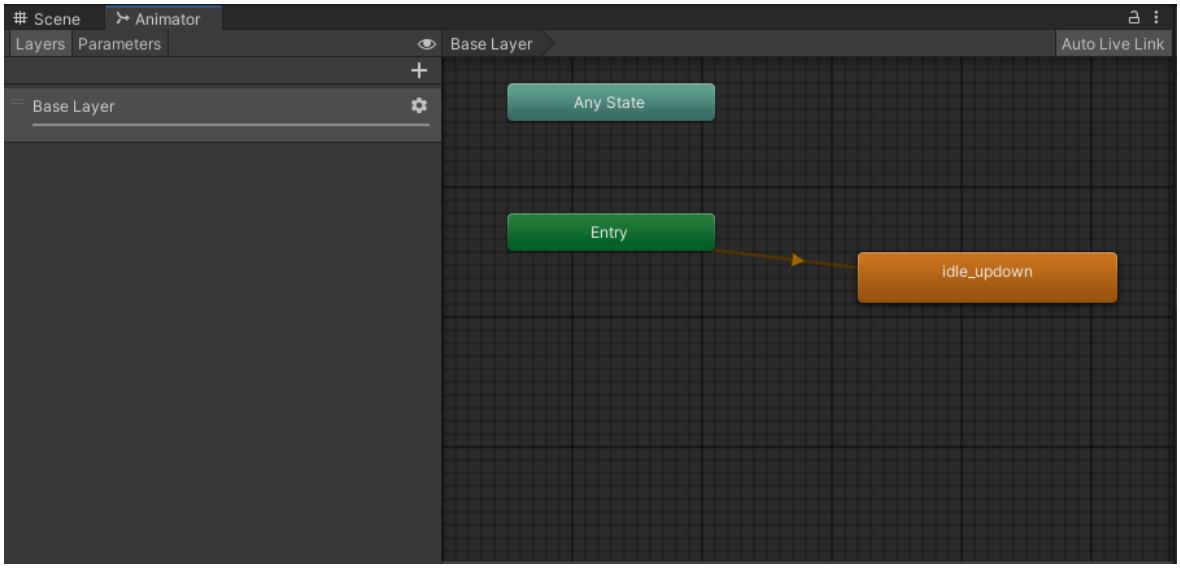
Hình 11: Giao diện cửa sổ Project

Cửa sổ Console: Hiển thị thông tin debug và lỗi trong quá trình chạy trò chơi. Cửa sổ này là công cụ quan trọng để theo dõi và gỡ lỗi các vấn đề trong mã của người dùng, bao gồm cả thông báo lỗi, cảnh báo và in ra từ hàm Debug trong mã. Nó cung cấp thông tin quan trọng để xác định và sửa chữa lỗi trong trò chơi.



Hình 12: Giao diện cửa sổ console

Cửa sổ Animation: Dùng để tạo và chỉnh sửa các hoạt cảnh (Animation) cho các đối tượng trong trò chơi. Người dùng có thể tạo ra các hoạt động động, quay, và hiệu ứng khác cho các đối tượng, và cấu hình các keyframe và các thuộc tính của chúng. Điều này cho phép người dùng tạo ra các động tác phong phú và linh hoạt cho các nhân vật và đối tượng trong trò chơi.



Hình 13: Giao diện cửa sổ Animation

2.3 Kiến trúc tổng quan

Trong quá trình phát triển game, Unity Engine là một công cụ mạnh mẽ, và một phần quan trọng của nó là UnityAPI - một bộ công cụ lập trình được cung cấp bởi Unity để viết script cho trò chơi. UnityAPI chứa các đối tượng và phương thức hỗ trợ hầu hết các thành phần và đối tượng trong Unity, mang lại sức mạnh linh hoạt cho người phát triển.

Trong một scene của trò chơi, có thể có nhiều đối tượng game khác nhau. Mỗi đối tượng này có thể được gắn kèm với một hoặc nhiều đoạn script. Để gắn script vào một đối tượng, người phát triển phải kế thừa class đó từ lớp MonoBehaviour trong UnityAPI, và tên của class cũng phải trùng khớp với tên của file script. Khi một script được gắn kèm vào một đối tượng game, đó được coi như là một thành phần bên trong của đối tượng đó, và được quản lý và cấp phát vùng nhớ khi chạy trò chơi. Điều này tạo ra một cấu trúc linh hoạt và dễ quản lý trong quá trình phát triển và thực thi game.

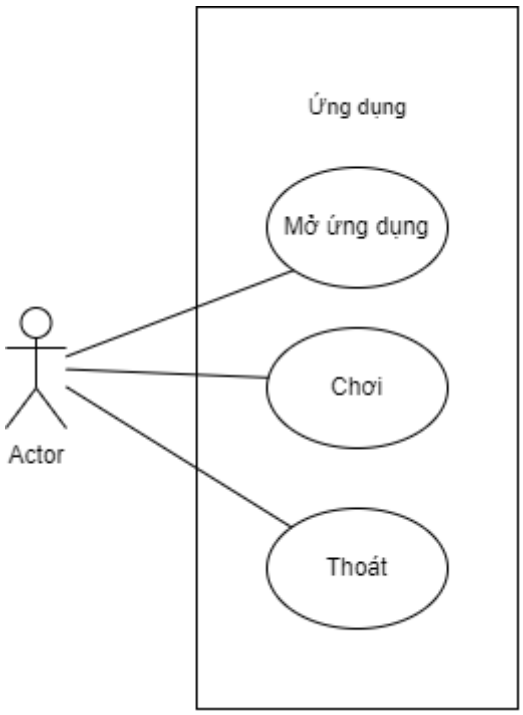
Trong UnityAPI, có một loạt các lớp hỗ trợ lập trình, cung cấp các phương thức và tính năng mạnh mẽ để xây dựng trò chơi. Dưới đây là một số lớp quan trọng và thường được sử dụng:

- **GameObject:** Đại diện cho các đối tượng trong trò chơi. GameObject có thể chứa các thành phần như Renderer, Collider, và Script, và có thể được sắp xếp thành các cấu trúc phức tạp như cây Hierarchy.
- **Transform**: Quản lý vị trí, quay và tỷ lệ của một GameObject. Transform cho phép người phát triển điều chỉnh và kiểm soát vị trí và hình dạng của các đối tượng trong trò chơi.

- **MonoBehaviour:** Là lớp cơ sở cho các script trong Unity. MonoBehaviour chứa các phương thức như Start, Update, và FixedUpdate, cho phép người phát triển tương tác với và điều khiển các đối tượng và hành vi trong trò chơi.
- **Collider:** Đại diện cho các vùng va chạm trong trò chơi, cho phép xác định khi nào hai đối tượng va chạm với nhau.
- **Rigidbody:** Đại diện cho vật lý của một đối tượng trong trò chơi. Rigidbody cho phép áp dụng lực và chuyển động vật lý cho các đối tượng, tạo ra hành vi chuyển động tự nhiên.
- **Camera:** Đại diện cho máy ảnh trong trò chơi, cho phép người phát triển điều chỉnh và kiểm soát góc nhìn của người chơi trong trò chơi.
- **AudioSource:** Cho phép phát âm thanh trong trò chơi. AudioSource cho phép người phát triển tải và phát các tệp âm thanh, cũng như điều chỉnh âm lượng và hiệu ứng âm thanh.

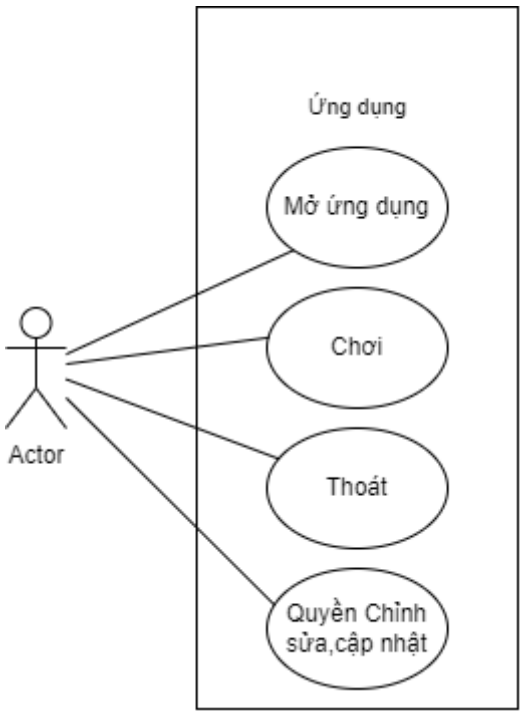
2.4 Các thành phần xử lý

2.4.1 Sơ đồ Usercase (user)



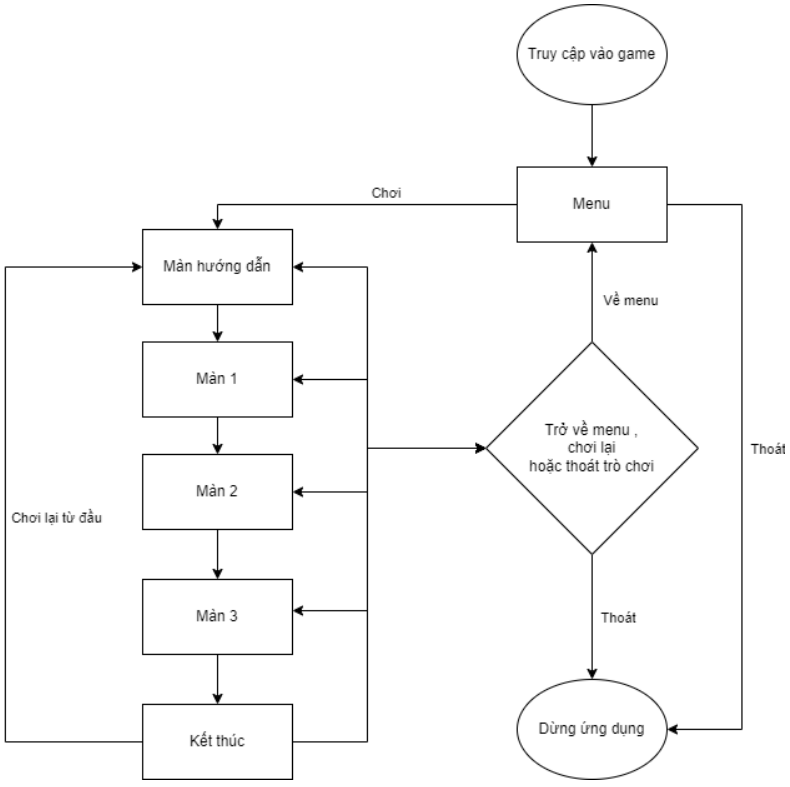
Hình 14: Sơ đồ Use case đối với người dùng

2.4.2 Sơ đồ Usercase (admin)



Hình 15: Sơ đồ Use case đối với admin

2.4.3 Lưu đồ hoạt động



Hình 16: Lưu đồ hoạt động

2.5 Mô tả kịch bản các màn chơi và các giải thuật

2.5.1 Màn hướng dẫn

Mô tả màn chơi: Màn này là bước khởi đầu trong cuộc hành trình của người chơi. Ở đây, người chơi sẽ được hướng dẫn cách di chuyển và tìm các mục tiêu của trò chơi. Mục tiêu của màn này là giúp người chơi làm quen với cách điều khiển nhân vật và thu thập viên kim cương trong màn hướng dẫn.

Đối tượng: Màn này gồm các đối tượng: Nhân vật chơi, một viên kim cương để thu thập và thực hiện qua màn, một chìa khoá, một cánh cửa.Môi trường xung quanh gồm các đối tượng cây cối,hoa,địa hình. Không có kẻ địch và chướng ngại vật có thể xuất hiện ở đây.

Điều kiện qua màn: Người chơi cần chạm vào viên kim cương sau khi đã tìm được chìa khoá để mở cửa.Khi đó màn hướng dẫn đã hoàn thành và sẽ chuyển sang màn 1

2.5.2 Màn 1

Mô tả màn chơi: Trong màn này, người chơi sẽ phải tiến xa hơn trong hành trình của họ, tiếp xúc với một không gian vừa và đối mặt với những thách thức đầu tiên. Người chơi sẽ thực hiện thu thập đầy đủ các trái cây trên sân cụ thể là 8 trái cây nhưng sẽ có một trái cây cuối cùng nằm ở phía sau cánh cửa vậy nên cũng cần tìm chìa khoá khi chạm vào chìa khoá thì cách cửa sẽ biến mất cũng như màn hướng dẫn. Trong màn này sẽ có đối thủ bất ngờ truy tìm người chơi nên trò chơi sẽ không đơn giản để có thể qua màn.

Đối tượng:

Màn này được bổ sung các đối tượng bao gồm: các trái cây như:Táo,khóm và dưa hấu để thu thập,một số đối tượng thuộc yếu tố môi trường như mây, các loại cây cối khác nhau và đặc biệt sẽ có một kẻ địch đuổi theo người chơi, kẻ thù ở đây chỉ di chuyển theo người chơi nên sẽ đi xuyên chướng ngại vật.

Các đối tượng địa hình, một chìa khoá và cửa sẽ được sử dụng lại từ màn hướng dẫn

Điều kiện qua màn: Người chơi cần thu thập đủ số lượng trái cây trên sân theo yêu cầu của màn chơi ở màn này là 8, tìm chìa khoá để mở cửa thu thập trái cây đằng sau nó và tránh bị đối thủ chạm vào.Khi thu thập đủ 8 trái cây thì lập tức chuyển sang màn thứ 2

2.5.3 Màn 2

Mô tả màn chơi: Màn này đánh dấu thử thách tiếp theo trong cuộc phiêu lưu của người chơi. Người chơi sẽ tiếp xúc với một không gian lớn hơn với các đường

dẫn đến một vài khu vực nhỏ để thu thập trái cây tuy nhiên một số khu vực sẽ có hệ thống bẫy vậy nên người chơi cần phải tránh đồng thời di chuyển sao cho mượt mà. Đặt biệt khi không chỉ tránh các bẩy mà còn phải tránh một đối thủ thông minh nó sẽ tìm đường đi ngắn để tiếp cận tuy nhiên trong không gian này sẽ có một số bụi nhỏ người chơi có thể đi qua bụi nhỏ này để tránh bị tiếp cận với bụi nhỏ này thì đối thủ không thể đi qua nó được mà đối thủ sẽ tìm một đường khác để truy đuổi người chơi.

Đối tượng:

Màn này gồm một số các đối tượng của các màn trước như: Địa hình, cây cối,một chìa khoá,một cửa,trái cây,nhân vật chơi.

Các đối tượng được bổ sung trong màn này gồm: Bẫy,Các bụi hoa nhỏ,một đối tượng đối thủ được thiết lập AI.

Điều kiện qua màn: Người chơi cần thu thập đủ số lượng trái cây ở đây là 10, di chuyển tránh các bẩy và cũng tìm chìa khoá để mở cửa và thu thập một trái cây đằng sau nó.Khi thu thập đầy đủ mà không bị đối thủ chạm vào thì lập tức chuyển sang màn cuối cùng của thử thách.

2.5.4 Màn 3

Mô tả màn chơi: Đây là màn thử thách cuối của người chơi.Người chơi sẽ vào không gian rất lớn như mê cung và các con đường rất hẹp dễ rơi xuống tuy vậy người chơi sẽ được đưa về địa diểm cụ thể.Trong màn này có hai đối thủ,một đối thủ AI sử dụng thuật toán A* và không sử dụng thuật toán, không có bẩy tuy nhiên có một vài trái cây chúng không thể thu thập và khi người chơi chạm vào thì lập tức sẽ bị thua cuộc vậy nên cần quan sát kỹ trái cây đó.Nhiệm vụ vẫn như các màn trước.và số lượng cần thu thập lên đến 21.Tuy độ khó cao nhưng người sẽ được tăng tốc dộ di chuyển đồng nghĩa người chơi phải di chuyển khéo léo.

Đối tượng:

Màn này bao gồm các đối tượng:Địa hình,môi trường,các cây cối,các loại trái cây, một viên kim cương, một chìa khoá và một cửa.

Các đối tượng được bổ sung trong màn này gồm: Hai đối thủ.

Điều kiện qua màn: Người chơi cần tìm chìa khoá để mở cửa thu thập kim cương đồng thời thu thập đủ số lượng trái cây ở đây là 20 trái cây tính thêm kim cương là 21 .tránh các vật gây hại và khi thu thập đủ thì chúc mừng người chơi đã hoàn thành trò chơi trong màn (Scene) kết thúc trò chơi

2.5.5 Kết thúc trò chơi

Mô tả màn chơi: Sau khi hoàn thành ba màn trong trò chơi người chơi được đưa về nhà (màn hướng dẫn) .Người chơi có thể chơi lại hoặc ra menu.Màn này không thực hiện gì hết vì người chơi đã hoàn thành tất cả các thử thách trước đó.

Đối tượng: Sử dụng các đối tượng từ màn hướng dẫn như:Môi trường,cây cối,hoa,địa hình,một viên kim cương.

2.6 Giải thuật A* trong Unity

Giải thuật đóng vai trò quan trọng trong việc xử lý các tình huống phức tạp trong trò chơi và đảm bảo trải nghiệm chơi game mượt mà và logic. Trong phần này, chúng ta sẽ trình bày về quá trình thiết kế và cài đặt giải thuật A* (A-star) trong trò chơi thu thập trái cây. Giải thuật A* được sử dụng để tìm đường đi tối ưu từ vị trí hiện tại của kẻ thù đến mục tiêu, tức là người choi gần nhất, đồng thời điều chỉnh đường đi trong các tình huống thay đổi.

Xây dựng bản đồ: Trước hết, cần xây dựng một bản đồ để trình bày môi trường của trò chơi. Bản đồ này sẽ được biểu diễn dưới dạng một lưới ô vuông, trong đó mỗi ô vuông đại diện cho một vị trí trong môi trường. Mỗi ô vuông sẽ có các thuộc tính như chi phí di chuyển, đại diện cho sự khó khăn của việc di chuyển qua ô đó.

Tìm đường đi: Sau khi có bản đồ, thuật toán A* sẽ được áp dụng để tìm đường đi tối ưu từ vị trí hiện tại của kẻ dịch đến mục tiêu, Thuật toán này sử dụng một hàm chi phí tổng hợp để ước lượng chi phí tối ưu từ điểm xuất phát đến điểm đích thông qua mỗi ô vuông.

Điều chỉnh đường đi: Trong quá trình di chuyển, có thể xuất hiện các thay đổi không mong muốn trong môi trường như sự xuất hiện của nhân vật chính hoặc các chướng ngại vật mới. Khi điều này xảy ra, thuật toán A* sẽ được kích hoạt lại để tạo ra đường đi mới cho kẻ thù.

Tích hợp Giải thuật vào trò chơi: Cuối cùng, sau khi đã thiết kế và cài đặt giải thuật A* sẽ tích hợp nó vào trò chơi. Điều này bao gồm việc liên kết giữa kết quả của thuật toán A* và các hành động của kẻ dịch, đồng thời xử lý các tình huống đặc biệt như khi không có đường đi hoặc khi mục tiêu thay đổi.

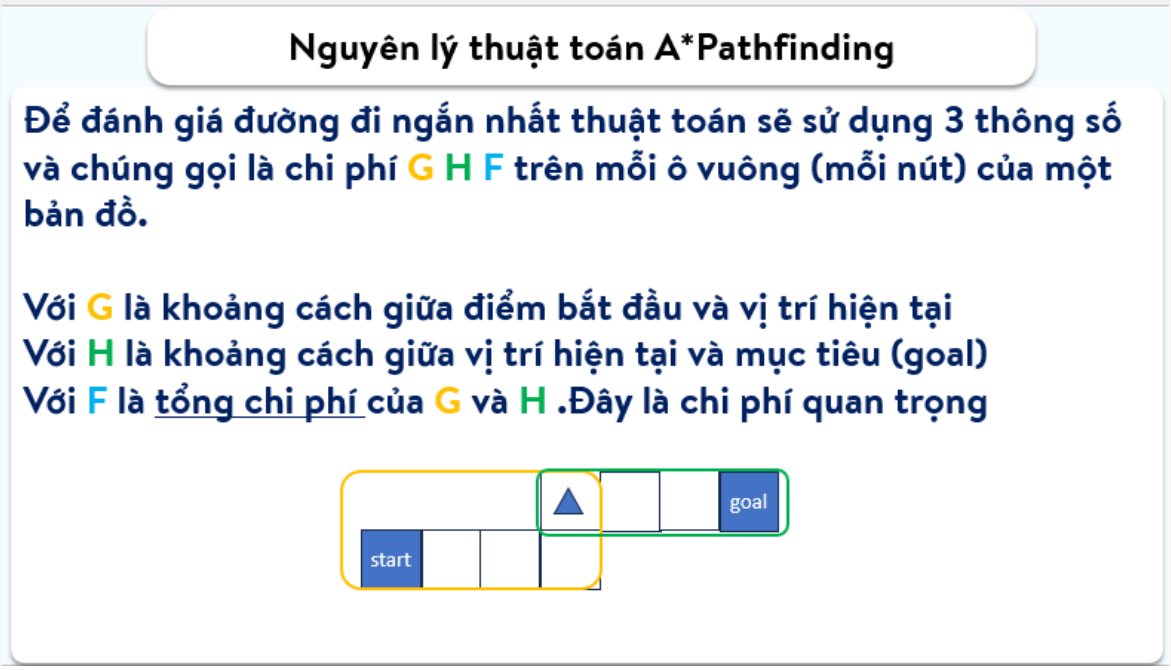
Cụ thể hơn về cơ chế của A* : Thuật toán A* tính toán đường đi ngắn nhất bằng cách sử dụng một kết hợp giữa các thông tin từ hai nguồn:

Chi phí thực tế (g) từ điểm bắt đầu đến điểm hiện tại: Đây là chi phí thực tế để đi từ điểm bắt đầu đến điểm hiện tại trên đường đi đang xem xét. Nó thường được tính bằng tổng các trọng số của các cạnh đã đi qua.

Ước lượng chi phí (h) từ điểm hiện tại đến điểm kết thúc: Đây là một ước lượng về chi phí còn lại để đi từ điểm hiện tại đến điểm kết thúc. Trong thuật toán A*, hàm

ước lượng này thường được gọi là hàm heuristic và phải là một ước lượng tốt nhưng cần phải nhanh chóng tính toán. Hàm heuristic thường được chọn sao cho nó không bao giờ lớn hơn chi phí thực tế từ điểm đó đến điểm kết thúc, mà điều này được gọi là "tính chất hàm heuristic đường đi không đáng tin cậy".

Khi A* chạy, nó duyệt qua các nút trong đồ thị một cách tương đối thông minh. Nó sử dụng tổng của chi phí thực tế đã đi qua và ước lượng chi phí còn lại để xác định nên duyệt các nút nào tiếp theo. Thuật toán chọn nút tiếp theo có tổng chi phí thấp nhất (chi phí thực tế + ước lượng chi phí còn lại) để tiếp tục duyệt. Điều này giúp thuật toán tìm ra đường đi ngắn nhất từ điểm bắt đầu đến điểm kết thúc một cách hiệu quả. Để có thể hình dung thêm thì ta có thể ví dụ ở dưới.



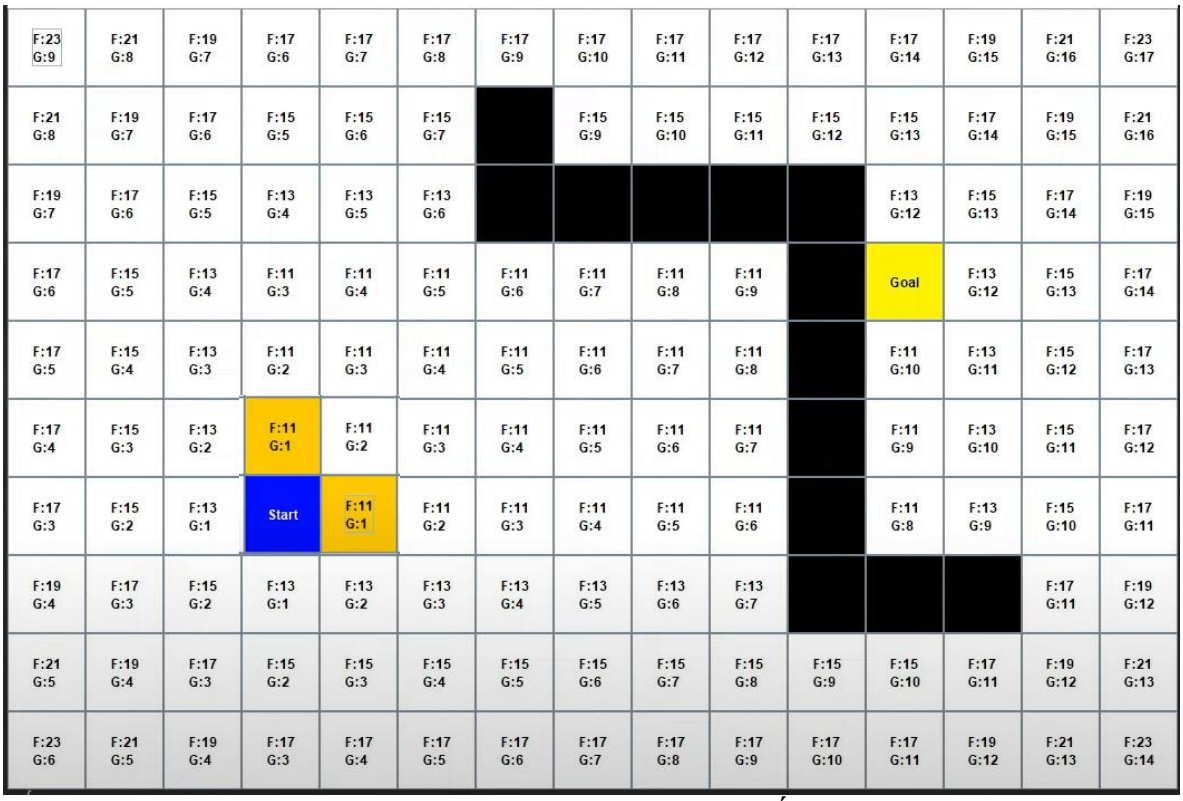
Hình 17: Các thông số của thuật toán.

F:23 G:9	F:21 G:8	F:19 G:7	F:17 G:6	F:17 G:7	F:17 G:8	F:17 G:9	F:17 G:10	F:17 G:11	F:17 G:12	F:17 G:13	F:17 G:14	F:19 G:15	F:21 G:16	F:23 G:17
F:21 G:8	F:19 G:7	F:17 G:6	F:15 G:5	F:15 G:6	F:15 G:7		F:15 G:9	F:15 G:10	F:15 G:11	F:15 G:12	F:15 G:13	F:17 G:14	F:19 G:15	F:21 G:16
F:19 G:7	F:17 G:6	F:15 G:5	F:13 G:4	F:13 G:5	F:13 G:6						F:13 G:12	F:15 G:13	F:17 G:14	F:19 G:15
F:17 G:6	F:15 G:5	F:13 G:4	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8	F:11 G:9		Goal	F:13 G:12	F:15 G:13	F:17 G:14
F:17 G:5	F:15 G:4	F:13 G:3	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8		F:11 G:10	F:13 G:11	F:15 G:12	F:17 G:13
F:17 G:4	F:15 G:3	F:13 G:2	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7		F:11 G:9	F:13 G:10	F:15 G:11	F:17 G:12
F:17 G:3	F:15 G:2	F:13 G:1	Start	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6		F:11 G:8	F:13 G:9	F:15 G:10	F:17 G:11
F:19 G:4	F:17 G:3	F:15 G:2	F:13 G:1	F:13 G:2	F:13 G:3	F:13 G:4	F:13 G:5	F:13 G:6	F:13 G:7				F:17 G:11	F:19 G:12
F:21 G:5	F:19 G:4	F:17 G:3	F:15 G:2	F:15 G:3	F:15 G:4	F:15 G:5	F:15 G:6	F:15 G:7	F:15 G:8	F:15 G:9	F:15 G:10	F:17 G:11	F:19 G:12	F:21 G:13
F:23 G:6	F:21 G:5	F:19 G:4	F:17 G:3	F:17 G:4	F:17 G:5	F:17 G:6	F:17 G:7	F:17 G:8	F:17 G:9	F:17 G:10	F:17 G:11	F:19 G:12	F:21 G:13	F:23 G:14

Hình 18: Bản đồ sau khi tính giá trị F và G.

+ Đầu tiên đánh giá các nút liền kề xung quanh nút start

+ Xét chi phí F có số nhỏ nhất sau đó xét G, trong trường hợp này thuật toán sẽ chọn cả hai nút và đặt chúng là đã kiểm tra.



Hình 19: Xét các nút có giá trị F nhỏ nhất sau đó xét G

+ Tiếp theo là nó sẽ đánh giá các nút liền kề với các nút đã kiểm tra

+ Ta thấy có 3 nút có giá trị F là 11 và tất cả có chi phí G là 2 nên ta đặt chúng là đã kiểm tra và tiếp tục công việc như vậy.

F:23 G:9	F:21 G:8	F:19 G:7	F:17 G:6	F:17 G:7	F:17 G:8	F:17 G:9	F:17 G:10	F:17 G:11	F:17 G:12	F:17 G:13	F:17 G:14	F:19 G:15	F:21 G:16	F:23 G:17
F:21 G:8	F:19 G:7	F:17 G:6	F:15 G:5	F:15 G:6	F:15 G:7		F:15 G:9	F:15 G:10	F:15 G:11	F:15 G:12	F:15 G:13	F:17 G:14	F:19 G:15	F:21 G:16
F:19 G:7	F:17 G:6	F:15 G:5	F:13 G:4	F:13 G:5	F:13 G:6						F:13 G:12	F:15 G:13	F:17 G:14	F:19 G:15
F:17 G:6	F:15 G:5	F:13 G:4	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8	F:11 G:9		Goal	F:13 G:12	F:15 G:13	F:17 G:14
F:17 G:5	F:15 G:4	F:13 G:3	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8		F:11 G:10	F:13 G:11	F:15 G:12	F:17 G:13
F:17 G:4	F:15 G:3	F:13 G:2	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7		F:11 G:9	F:13 G:10	F:15 G:11	F:17 G:12
F:17 G:3	F:15 G:2	F:13 G:1	Start	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6		F:11 G:8	F:13 G:9	F:15 G:10	F:17 G:11
F:19 G:4	F:17 G:3	F:15 G:2	F:13 G:1	F:13 G:2	F:13 G:3	F:13 G:4	F:13 G:5	F:13 G:6	F:13 G:7				F:17 G:11	F:19 G:12
F:21 G:5	F:19 G:4	F:17 G:3	F:15 G:2	F:15 G:3	F:15 G:4	F:15 G:5	F:15 G:6	F:15 G:7	F:15 G:8	F:15 G:9	F:15 G:10	F:17 G:11	F:19 G:12	F:21 G:13
F:23 G:6	F:21 G:5	F:19 G:4	F:17 G:3	F:17 G:4	F:17 G:5	F:17 G:6	F:17 G:7	F:17 G:8	F:17 G:9	F:17 G:10	F:17 G:11	F:19 G:12	F:21 G:13	F:23 G:14

Hình 20: Xét các nút có giá trị G tăng dần

+ Nhưng bây giờ đây là các nút rắn (chướng ngại vật) nên ko thể tiến xa hơn điều đó chương trình nhận ra đây ko phải là đường dẫn đến mục tiêu

+ Nên nó tìm nút triển vọng tiếp theo trong số các nút liền kề này

+ Ta thấy không có giá trị F:11 nhưng có 1 số F13 và 2 nút có chi phí G 1 nên chọn 2 nút này kiểm tra

F:15 G:5	F:13 G:4	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8	F:11 G:9
F:15 G:4	F:13 G:3	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8
F:15 G:3	F:13 G:2	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7
F:15 G:2	F:13 G:1	Start	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6
F:17 G:3	F:15 G:2	F:13 G:1	F:13 G:2	F:13 G:3	F:13 G:4	F:13 G:5	F:13 G:6	F:13 G:7

Hình 21: Xét các nút có giá trị F:13 và G:1

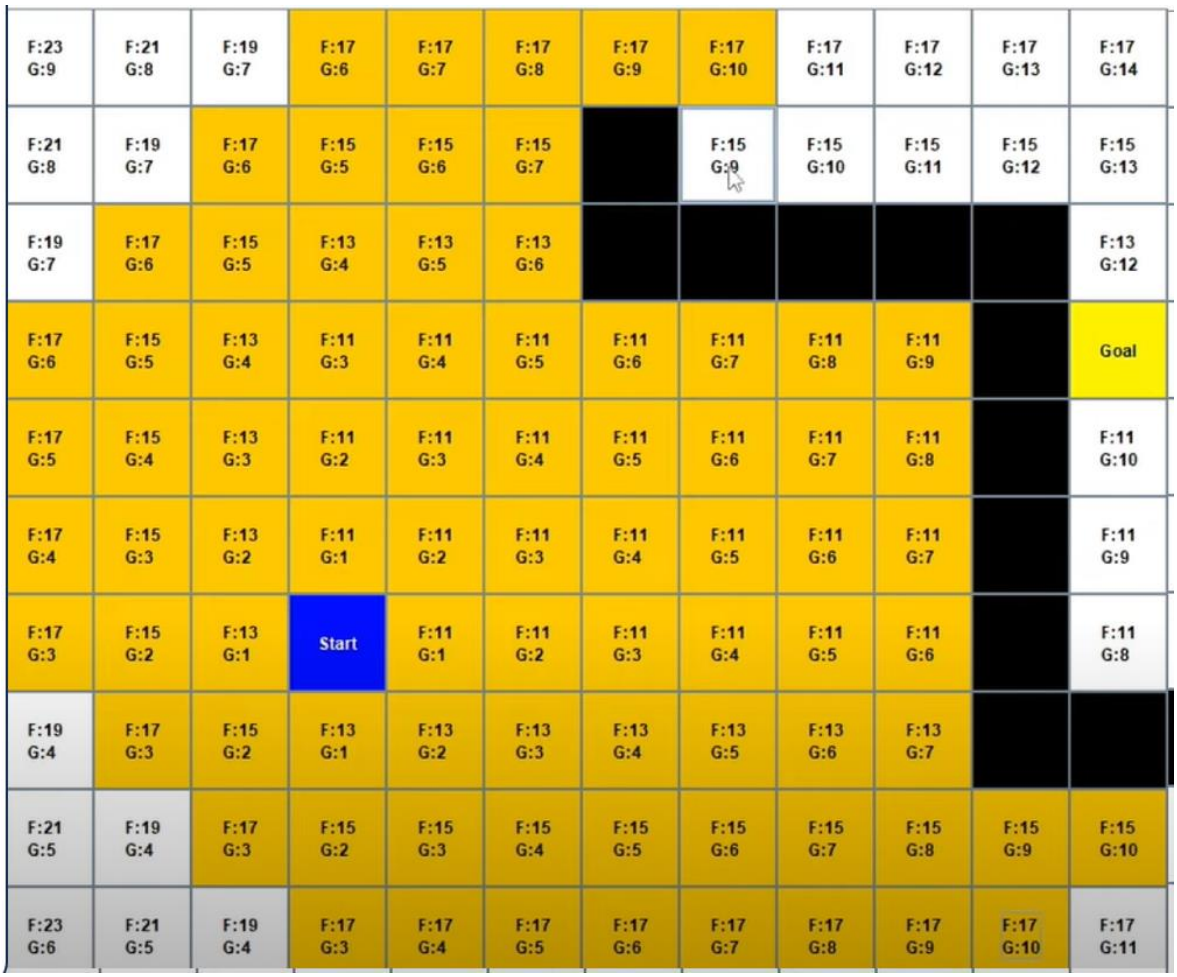
+ Tương tự như vậy thì xét có F:13 và G có giá trị tăng dần.

F:21 G:8	F:19 G:7	F:17 G:6	F:15 G:5	F:15 G:6	F:15 G:7		F:15 G:9	F:15 G:10	F:15 G:11	F:15 G:12	F:15 G:13	F:17 G:14
F:19 G:7	F:17 G:6	F:15 G:5	F:13 G:4	F:13 G:5	F:13 G:6						F:13 G:12	F:15 G:13
F:17 G:6	F:15 G:5	F:13 G:4	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8	F:11 G:9		Goal	F:13 G:12
F:17 G:5	F:15 G:4	F:13 G:3	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7	F:11 G:8		F:11 G:10	F:13 G:11
F:17 G:4	F:15 G:3	F:13 G:2	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6	F:11 G:7		F:11 G:9	F:13 G:10
F:17 G:3	F:15 G:2	F:13 G:1	Start	F:11 G:1	F:11 G:2	F:11 G:3	F:11 G:4	F:11 G:5	F:11 G:6		F:11 G:8	F:13 G:9
F:19 G:4	F:17 G:3	F:15 G:2	F:13 G:1	F:13 G:2	F:13 G:3	F:13 G:4	F:13 G:5	F:13 G:6	F:13 G:7			
F:21 G:5	F:19 G:4	F:17 G:3	F:15 G:2	F:15 G:3	F:15 G:4	F:15 G:5	F:15 G:6	F:15 G:7	F:15 G:8	F:15 G:9	F:15 G:10	F:17 G:11

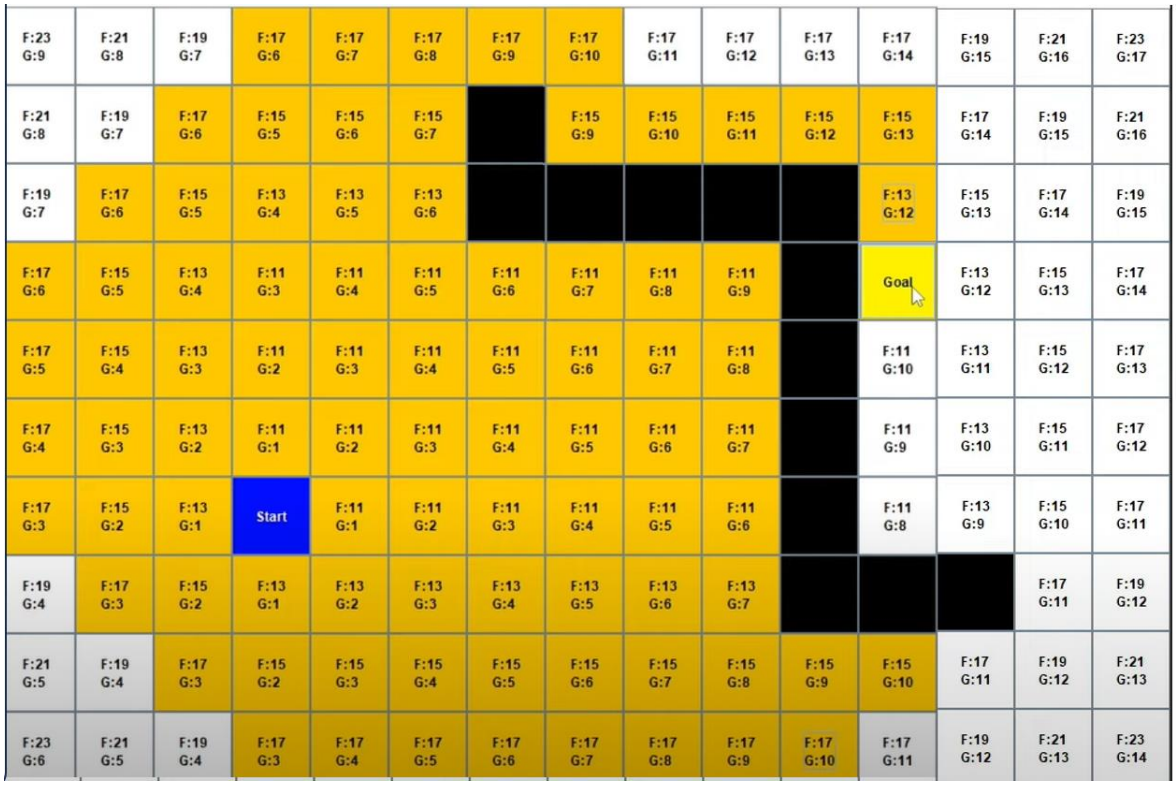
Hình 22: Thực hiện xét các nút xung quanh

+ Khi hết F:13 thì ta thấy có F:15 nhưng G nhỏ nhất là 2 thì ta xét nó và xét xung quanh có F:15 ứng với G tăng dần.

+ Đến khi F:17 Ta thấy rằng F giảm xuống còn 15 vì không có nút cố định và nó có thể di chuyển theo hướng mục tiêu nên F:15 G:9 sẽ là nút tiếp theo và xét F:15 và các thao tác xét các nút tương tự như vậy

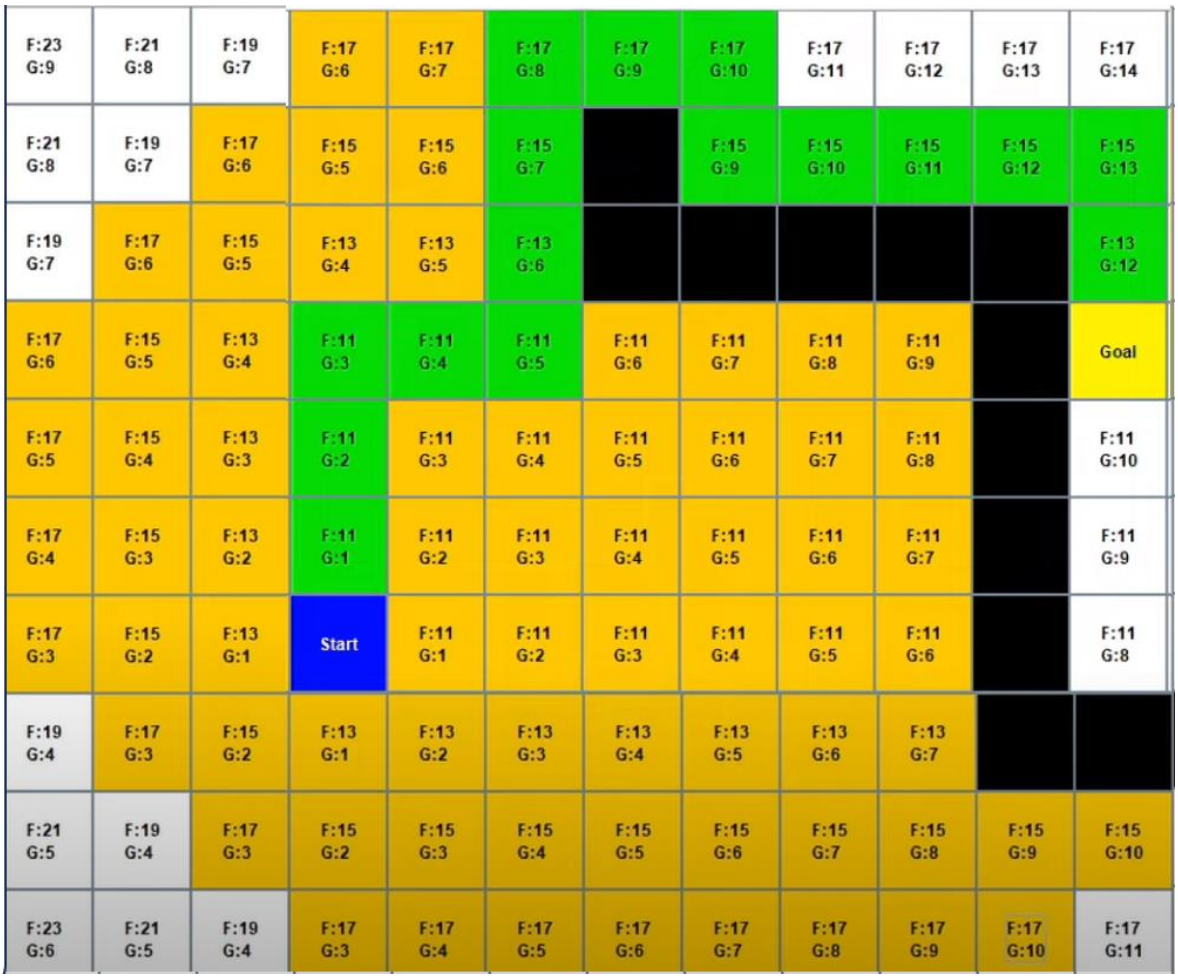


Hình 23: Xét F:15 và G:9



Hình 24: Xét các nút liền kề cuối cùng đi đến Goal

+ Dựa theo các thông số G và F tăng ta tìm được đường đi ngắn nhất.



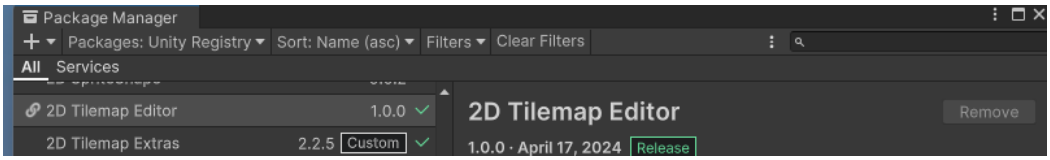
Hình 25: Thu thập và cho ra kết quả đường đi ngắn nhất

2.7 Xây dựng ứng dụng game

2.7.1 Thiết kế bản đồ

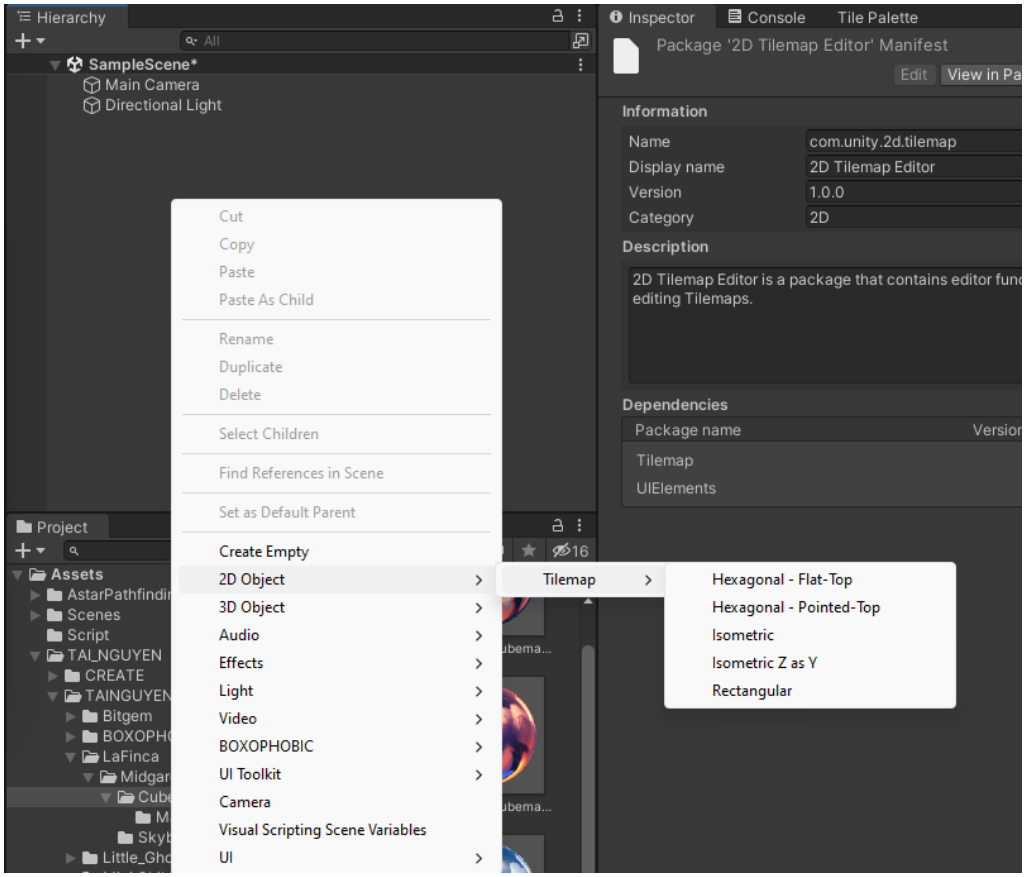
Trong Unity có tính năng là Prefab Brush.Prefab Brush là một tính năng trong Unity được sử dụng trong quá trình thiết kế bản đồ cho phép vẽ hoặc thả các Prefab vào scene một cách nhanh chóng và tiện lợi và Prefab là một đối tượng hoặc một nhóm các đối tượng đã được thiết lập và được lưu trữ trong một tệp riêng biệt.

+ Đầu tiên là cài đặt 2D Tilemap Editor.



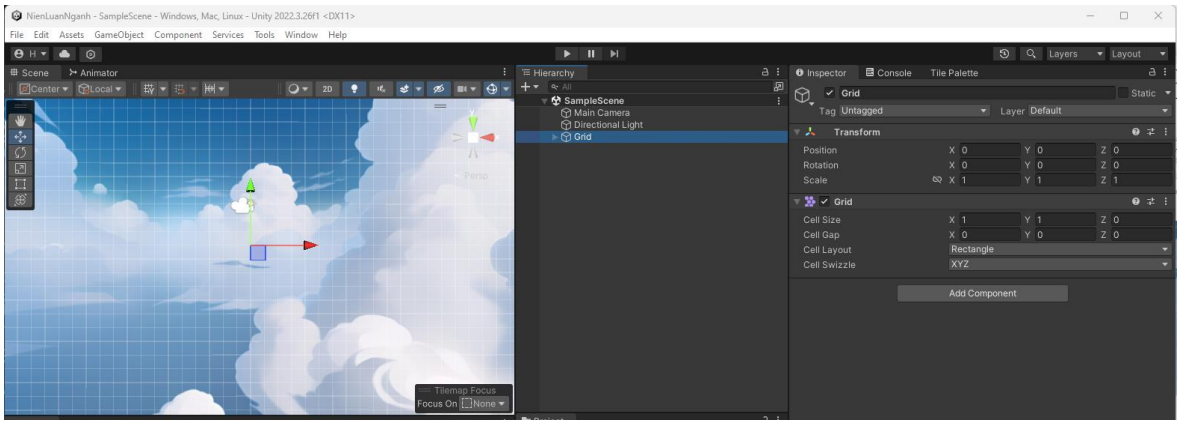
Hình 26: Cài đặt tilemap

+ Sau khi cài đặt tiến hành tạo một tilemap.



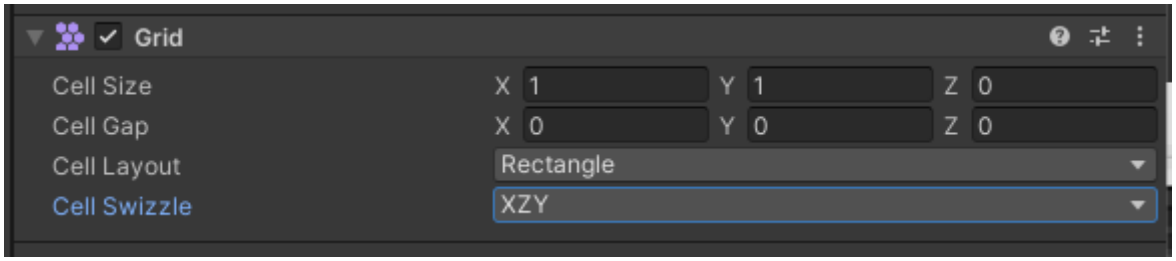
Hình 27: Tạo tilemap

+ Có thể tuỳ chỉnh kích thước lưới sao cho phù hợp

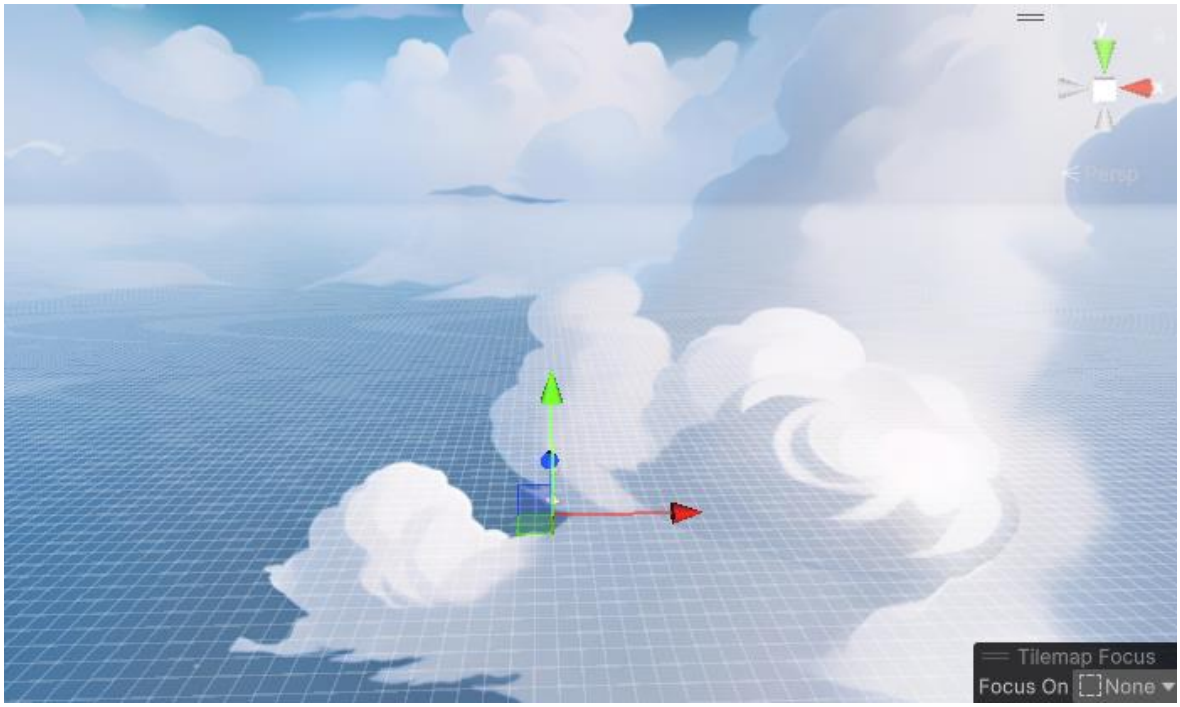


Hình 28: Thiết lập tilemap

+ Do hướng camera và trọng lực ta chỉnh Cell Swizzle về XZY

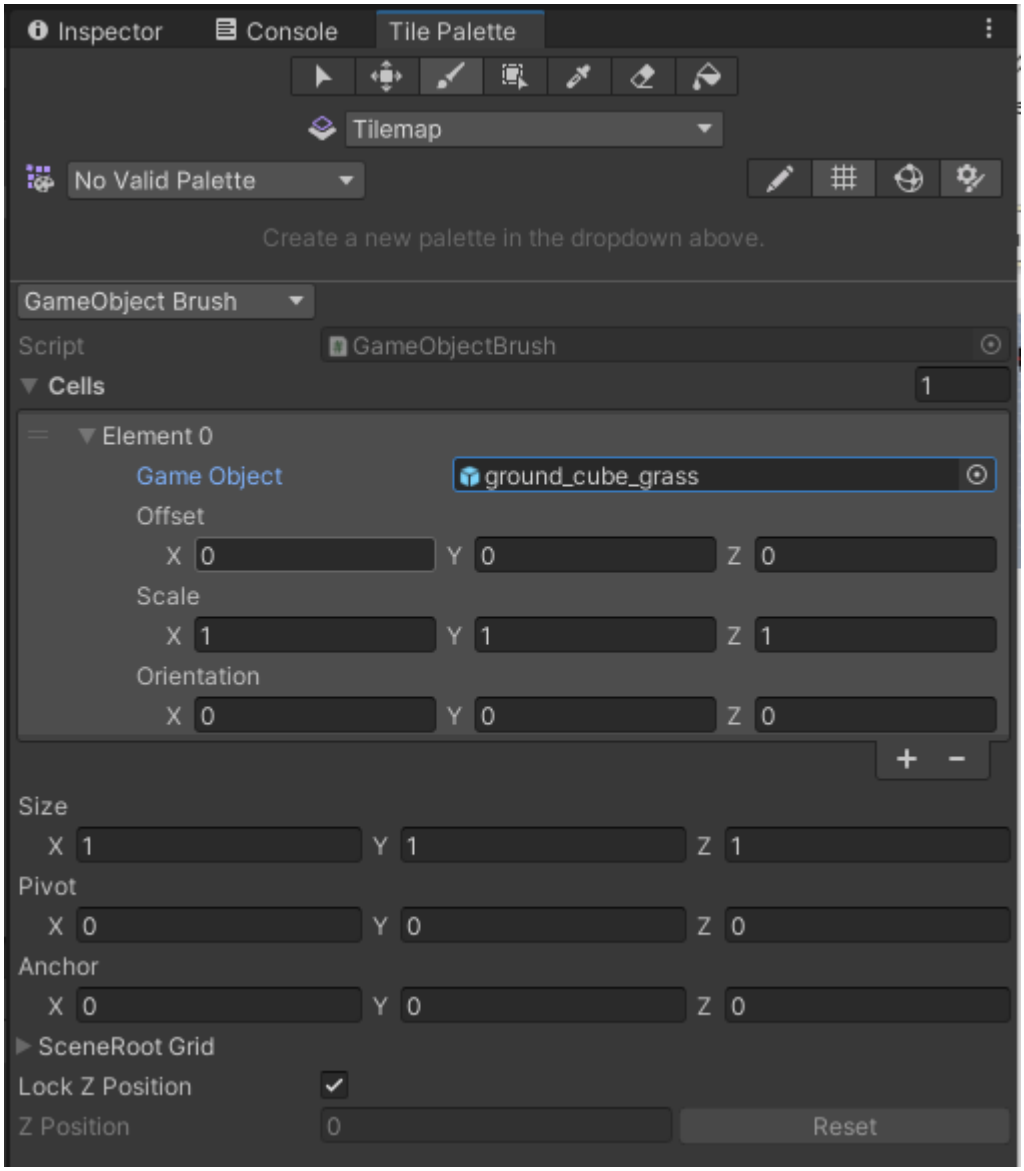


Hình 29: Chỉnh sửa toạ độ của tilemap



Hình 30 :Toạ độ của tilemap đã được chỉnh sửa

+ Trong Tile Palette có để điều chỉnh thông số vị trí. Ở đây có thể chọn Game object (đối tượng) bất kỳ để vẽ và tạo ra.



Hình 31: Chọn đối tượng để tạo địa hình

Hình 32: Vẽ các đối tượng trên tilemap

+ Cho mỗi đối tượng 2 thành phần Rigidbody và Collider để tuân theo quy tắc trọng lực và cấu trúc vật lý.

+ Tương tự như vậy sẽ thiết kế ra các màn chơi khác nhau.

2.7.2 Thiết kế các cơ chế tương tác trong trò chơi

Cơ chế của game này đều có một điểm chung đó là thu thập tất cả các trái cây theo yêu cầu, tránh các kẻ thù hay các bẫy để có thể qua màn. Để có thế thiết lập được ta cần gắn script vào các đối tượng : Trái cây, nhân vật,kẻ thù,giao diện và nhiệm vụ

+ **Thành phần trái cây:** Là đối tượng các trái cây trong game. Ta gắn vào cho các đối tượng trái cây đó một file script để xử lý người chơi thu thập.

```
using UnityEngine;

public class Apple : MonoBehaviour
{
    private AudioManager audioManager;

    void Start()
    {
        // Tìm và gán AudioManager
        audioManager = GameObject.FindGameObjectWithTag("Audio").GetComponent<AudioManager>();
    }

    public void OnTriggerEnter(Collider other)
    {
        // Sử dụng kiểu dữ liệu đúng cho thành phần muốn truy xuất
        PlayerInventory playerInventory = other.GetComponent<PlayerInventory>();

        if (playerInventory != null)
        {
            // Phát âm thanh khi nhặt được táo
            audioManager.PlaySFX(audioManager.collectClip);

            // Gọi phương thức AppleCollected từ PlayerInventory
            playerInventory.AppleCollected();

            // Tắt đối tượng táo
            gameObject.SetActive(false);
        }
    }
}
```

Hình 33: Script cho đối tượng Apple

Đoạn script trên để xử lý việc nhặt quả táo trong trò chơi. Khi một đối tượng người chơi tiếp xúc với quả táo, nó sẽ thông báo cho PlayerInventory và phát âm thanh. Sau đó, đối tượng quả táo sẽ được tắt đi.

+ **Thành phần người chơi:** Là đối tượng Player cho phép người chơi điều kiển và thực hiện các nhiệm vụ.Ở đây thêm script NVDichuyen và PlayerInventory.

```
public class NVDichuyen : MonoBehaviour
{
    public float speed = 5.0f;
    public float rotationSpeed;

    void Update()
    {
        // Lấy giá trị di chuyển ngang và dọc từ Input
        float horizontalMove = Input.GetAxis("Horizontal");
        float verticalMove = Input.GetAxis("Vertical");

        // Tạo một vectơ hướng di chuyển
        Vector3 moveDirection = new Vector3(horizontalMove, 0, verticalMove);

        // Chuẩn hóa vectơ hướng di chuyển
        moveDirection.Normalize();

        // Tính độ lớn của vectơ hướng di chuyển
        float magnitude = moveDirection.magnitude;

        // Giới hạn độ lớn của vectơ hướng di chuyển trong khoảng 0 và 1
        magnitude = Mathf.Clamp01(magnitude);

        // Di chuyển nhân vật theo vectơ hướng di chuyển, độ lớn và tốc độ trong thời gian delta
        transform.Translate(moveDirection * magnitude * speed * Time.deltaTime, Space.World);

        // Kiểm tra xem nhân vật có di chuyển hay không
        if (moveDirection != Vector3.zero)
        {
            // Tạo quaternion xoay
            Quaternion toRotate = Quaternion.LookRotation(moveDirection, Vector3.up);

            // Xoay nhân vật theo quaternion xoay, tốc độ xoay và thời gian delta
            transform.rotation = Quaternion.RotateTowards(transform.rotation, toRotate, rotationSpeed * Time.deltaTime);
        }
    }
}
```

Hình 34: Script di chuyển cho đối tượng Player

Script trên điều khiển di chuyển của một nhân vật trong trò chơi. Dưới đây là một số điểm quan trọng trong mã:

- Biến speed: Đây là tốc độ di chuyển của nhân vật.
- Biến rotationSpeed: Đây là tốc độ xoay của nhân vật.
- Phương thức Update(): Được gọi mỗi frame để cập nhật tình trạng của nhân vật.
- Lấy giá trị di chuyển từ Input: Sử dụng hàm Input.GetAxis("Horizontal") và Input.GetAxis("Vertical") để lấy giá trị di chuyển ngang và dọc từ bàn phím hoặc joystick.
- Tạo vectơ hướng di chuyển: Sử dụng các giá trị di chuyển để tạo một vectơ hướng di chuyển trong không gian thế giới.
- Chuẩn hóa và giới hạn vectơ hướng di chuyển: Chuẩn hóa vectơ hướng di chuyển để đảm bảo rằng nhân vật di chuyển với tốc độ cố định. Sau đó, giới hạn giá trị của vectơ hướng di chuyển trong khoảng từ 0 đến 1.
- Di chuyển nhân vật: Sử dụng transform.Translate để di chuyển nhân vật theo vectơ hướng di chuyển với tốc độ và thời gian delta.
- Xoay nhân vật: Nếu nhân vật đang di chuyển, sử dụng Quaternion.LookRotation để tạo quaternion xoay dựa trên hướng di chuyển. Sau đó, sử dụng Quaternion.RotateTowards để xoay nhân vật theo quaternion này với tốc độ xoay và thời gian delta.

```
public class PlayerInventory : MonoBehaviour
{
    public int Soapple { get; private set; }
    public UnityEvent<PlayerInventory> OnAppleCollected;
    public UnityEvent OnReachedMaxApples;

    // Số táo cần thu thập để chuyển đến Scene
    private const int RequiredApples = 8;

    public void AppleCollected()
    {
        Soapple++;
        OnAppleCollected.Invoke(this);

        // Kiểm tra nếu số lượng táo đã đạt đến số táo cần thiết để chuyển Scene
        if (Soapple >= RequiredApples)
        {
            // Gọi sự kiện khi đạt đến số táo tối đa
            OnReachedMaxApples.Invoke();
        }
    }

    // Hàm để chuyển đến Scene
    public void ChangeSceneToSecond()
    {
        SceneManager.LoadScene(3); // Thay "Scene2" bằng tên Scene
    }
}
```

Hình 35: Script quản lý kho cho đối tượng Player

Với Script này nó sẽ quản lý kho của người chơi. Mỗi khi người chơi nhặt được một quả táo, số lượng táo sẽ tăng lên. Khi đạt đến một số lượng táo nhất định, người chơi sẽ được chuyển đến một scene mới.Cụ thể hơn:

- Sự kiện OnAppleCollected được kích hoạt mỗi khi một quả táo được nhặt.
- Sự kiện OnReachedMaxApples được kích hoạt khi số lượng táo đạt đến giới hạn tối đa.
- Hàm AppleCollected tăng số lượng táo, kích hoạt sự kiện OnAppleCollected, và kiểm tra xem số lượng táo đã đạt đến giới hạn tối đa chưa.
- Hàm ChangeSceneToSecond dùng để chuyển đến scene thứ hai khi số lượng táo đạt đến giới hạn.

+ **Thành phần giao diện:** là một đối tượng dùng để làm hiển thị các thông tin trên màn hình.Ở đây sẽ gắn script để hiển thị số lượng thu thập.

```
public class InventoryUI : MonoBehaviour
{
    private TextMeshProUGUI appletext;
    // Start is called before the first frame update
    void Start()
    {
        appletext = GetComponent<TextMeshProUGUI>();
    }

    public void Updateappletext(PlayerInventory playerInventory)
    {
        appletext.text = playerInventory.Soapple.ToString();
    }
}
```

Hình 36: Script hiển thị số lượng thu thập

Đoạn script này để cập nhật giao diện người dùng (UI) của kho của người chơi. Nó sử dụng một TextMeshProUGUI để hiển thị số lượng táo trong kho của người chơi.

- Biến appletext lưu trữ tham chiếu đến TextMeshProUGUI được sử dụng để hiển thị số lượng táo.
- Trong hàm Start(), nó gán TextMeshProUGUI từ GameObject hiện tại và biến appletext.
- Phương thức Updateappletext(PlayerInventory playerInventory) được gọi để cập nhật số lượng táo trong kho của người chơi lên giao diện người dùng.

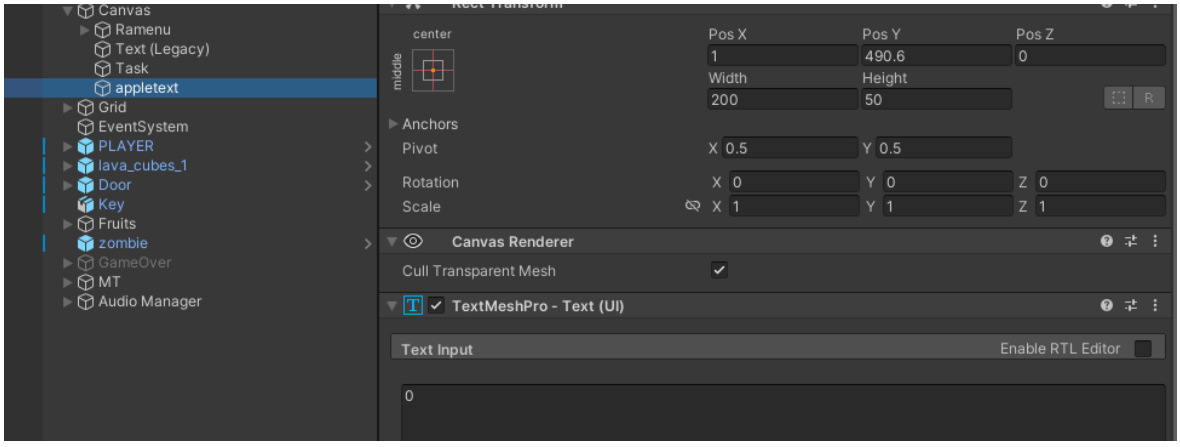
=> Như vậy cả ba đoạn script trên đều liên quan và có liên kết với nhau trong việc quản lý và hiển thị kho của người chơi trong trò chơi.Hãy xem xét các mối liên kết cụ thể:

- Apple và PlayerInventory:

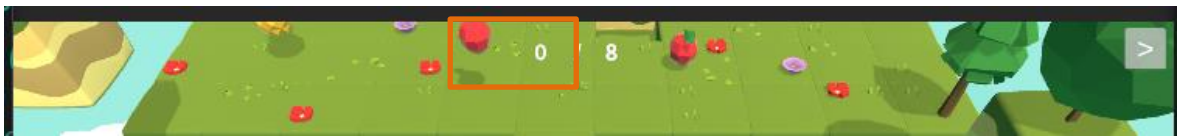
- Khi một đối tượng "Apple" được nhặt bởi người chơi, phương thức AppleCollected() trong PlayerInventory được gọi.
- Phương thức này tăng số lượng táo trong kho của người chơi và kiểm tra nếu số lượng đủ để kích hoạt sự kiện chuyển scene.
- PlayerInventory và InventoryUI:
- Khi số lượng táo trong kho của người chơi thay đổi (qua việc nhặt táo), phương thức Updateappletext(PlayerInventory playerInventory) trong InventoryUI được gọi.
- Phương thức này cập nhật giao diện người dùng để hiển thị số lượng táo mới.
- InventoryUI và Apple:
- Dù không có một liên kết trực tiếp, nhưng thông qua PlayerInventory, số lượng táo được cập nhật sau mỗi lần nhặt táo. Sau đó, thông qua InventoryUI, giao diện người dùng được cập nhật để hiển thị số lượng táo mới đó.

❖ **Các bước thực hiện liên kết:**

Phần UI(giao diện) tạo thêm một text và đặt tên điều chỉnh sao cho phù hợp.

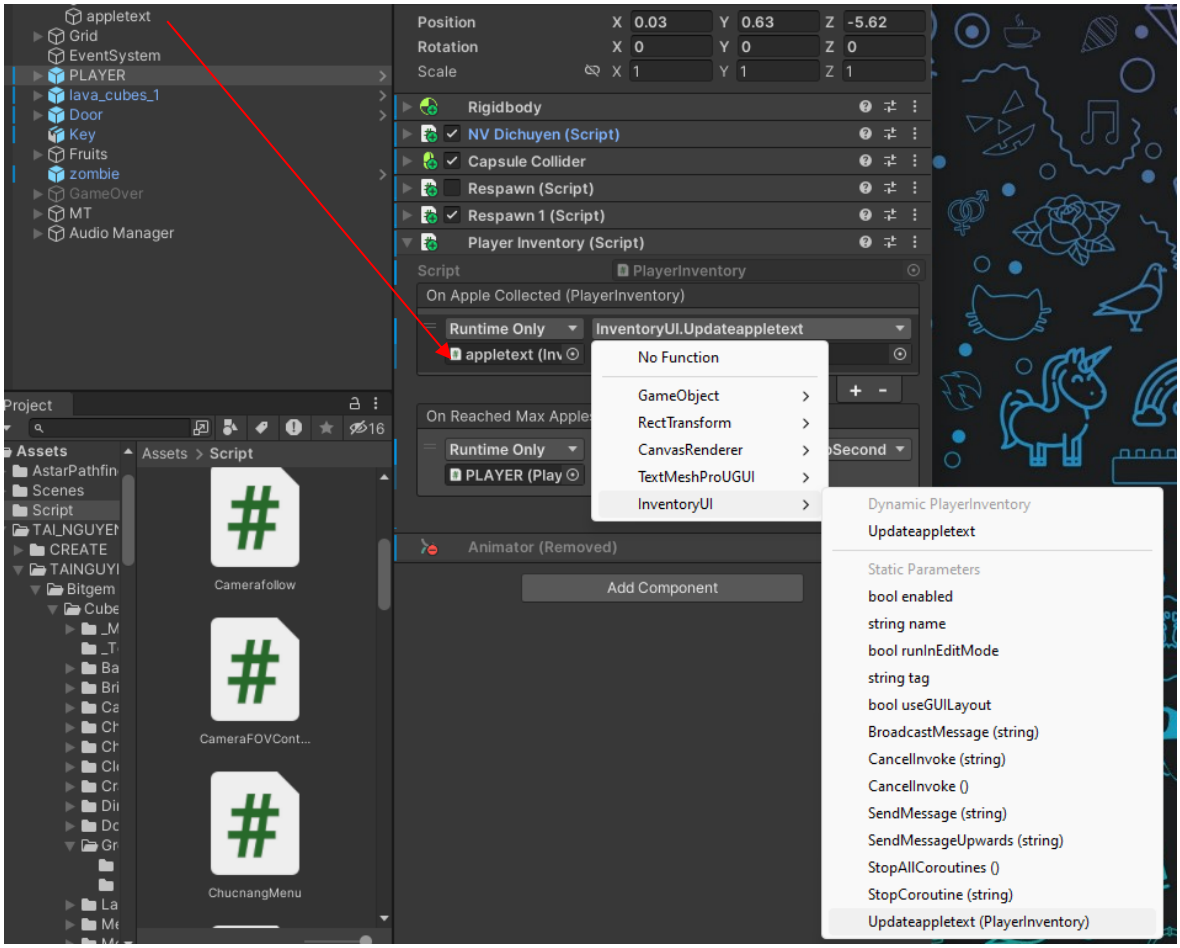


Hình 37: Tạo text và thêm script

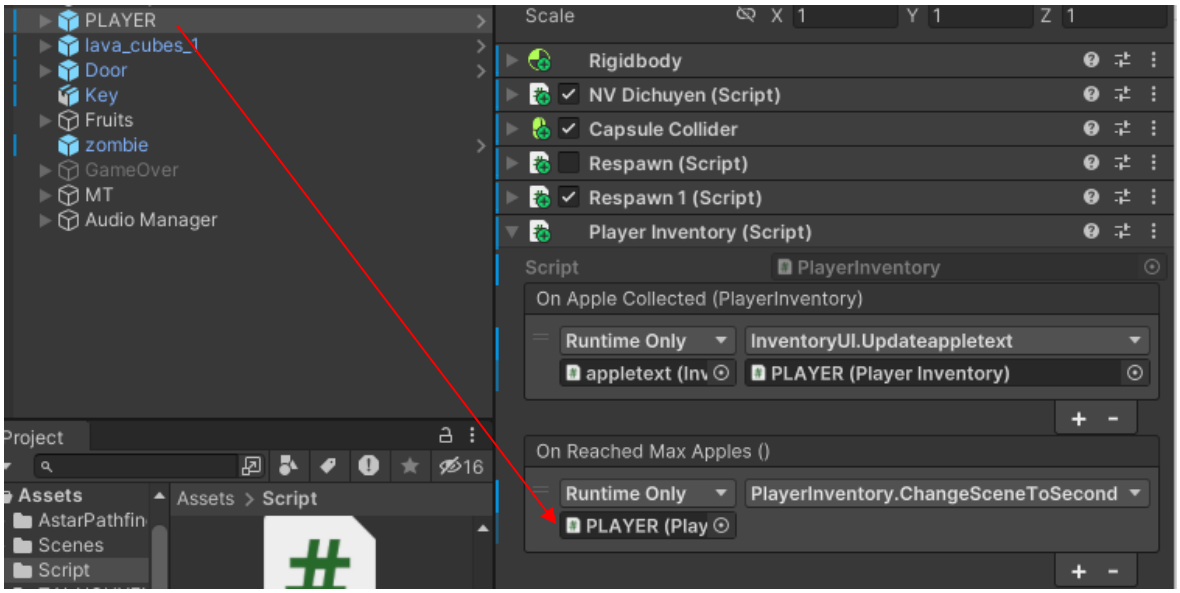


Hình 38: Số lượng trái cây thu thập sẽ được hiển thị tăng dần

Phần Player (người chơi): sau khi gắn script sẽ có các thiết lập ta kéo thả và chọn hàm để hoạt động và các tuỳ chỉnh tốc độ di chuyển và xoay.



Hình 39: Gắn script và cấu hình sự kiện thu thập cho đối tượng PLAYER



Hình 40: Cấu hình sự kiện số lượng cần hoàn thành cho đối tượng PLAYER

Hình 41: Thêm script di chuyển và điều chỉnh tốc độ cho đối tượng PLAYER

Phần trái cây: Vì script của nó Sử dụng kiểu dữ liệu đúng cho thành phần muốn truy xuất nên sẽ không có cấu hình.

+ **Thành phần kẻ địch và bẫy:** là đối tượng mà người choi cần phải né tránh,Ở đây ta sẽ gắn script xử lý khi Player (người chơi) chạm phải những đối tượng mang script “trap” thì trò chơi sẽ kết thúc.

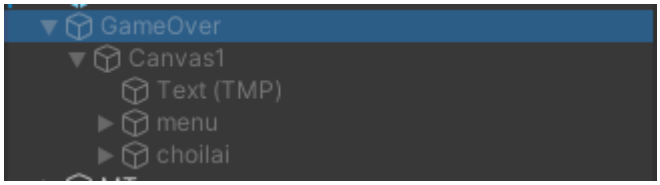
```
public class Trap : MonoBehaviour
{
    public GameObject gameOverObject;
    private AudioManager audioManager;

    void Start()
    {
        // Tìm và gán AudioManager
        audioManager = GameObject.FindGameObjectWithTag("Audio").GetComponent<AudioManager>();
    }

    private void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            // Phát âm thanh khi người chơi va chạm vào cạm
            audioManager.PlaySFX(audioManager.winClip);

            // Tắt đối tượng người chơi
            other.gameObject.SetActive(false);

            // Kích hoạt đối tượng game over
            gameOverObject.SetActive(true);
        }
    }
}
```

Hình 42: script bẫy dành cho các đối tượng gây hại

Ở script này nó sẽ kích hoạt âm thanh và Kích hoạt đối tượng game over: Khi đối tượng người chơi va chạm với "Trap", đối tượng gameOverObject sẽ được kích hoạt, cho phép hiển thị màn hình game over hoặc thực hiện các hành động liên quan đến kết thúc trò chơi.

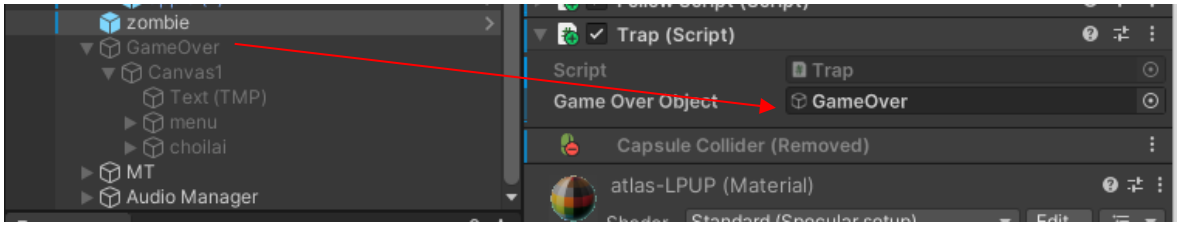
❖ **Thực hiện cấu hình script:**

- Tạo một đối tượng tên GameOver sao đó tạo thêm các giao hiện cho nó như nút chơi lại hoặc thoát và do nó là đối tượng khi chạm vào các đối tượng bẫy hay kẻ thù giao diện mới xuất hiện nên đối tượng này sẽ bị ẩn.



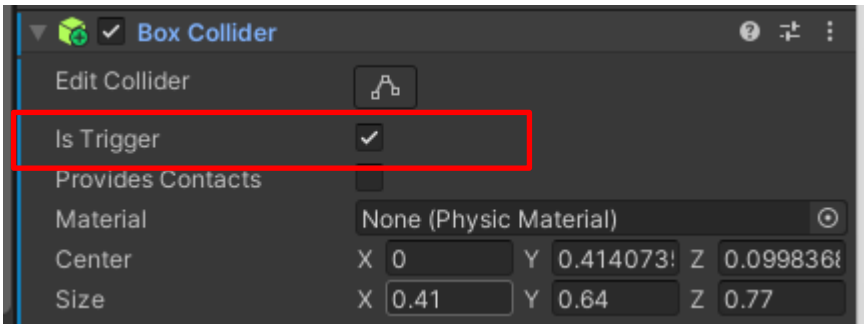
Hình 43: Tạo đối tượng GameOver và thêm đối tượng giao diện

- Sao đó gắn script "Trap" vào các đối tượng gây nguy hiểm cho người chơi và cần phải tránh né. Và kéo tối tượng GameOver vào cấu hình để hoạt động.

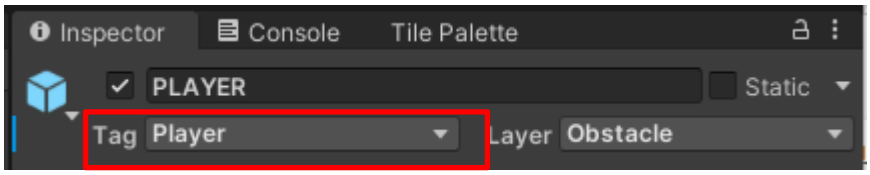


Hình 44: Gắn script và cấu hình cho các đối tượng mang đặc tính bẫy hoặc đối thủ

- Trong script có OnTrigger là một loại sự kiện được kích hoạt khi một đối tượng với Collider (được cài đặt để làm Trigger) va chạm với một đối tượng CompareTag("Player") có Collider nên phải cấu hình thêm.



Hình 45: Kích hoạt sự kiện kích hoạt cho đối tượng trap



Hình 46: Đặt Tag Player cho đối tượng PLAYER

+ **Các thành phần nhiệm vụ:** Là các đối tượng nhằm mục đích để mở lối đi và thu thập trái cây để hoàn thành mục tiêu,thành phần này bao gồm hai đối tượng chìa khoá và đối tượng cửa ở đây ta tạo một script cho chìa khoá và kích hoạt chung với đối tượng cửa.

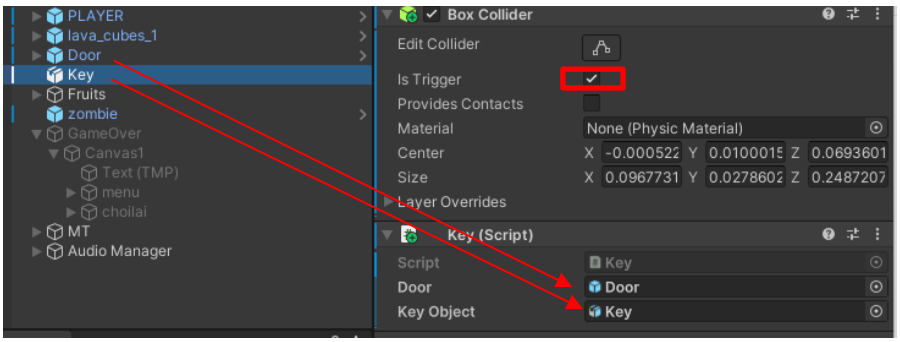
```
public class Key : MonoBehaviour
{
    private AudioManager audioManager;

    private void Awake()
    {
        audioManager = GameObject.FindGameObjectWithTag("Audio").GetComponent<AudioManager>();
    }

    [SerializeField]
    GameObject door;
    [SerializeField]
    GameObject keyObject;

    bool isOpened = false;

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player") && !isOpened)
        {
            isOpened = true;
            // Phát âm thanh khi người chơi chạm vào chìa khóa
            audioManager.PlaySFX(audioManager.coinClip);

            // Vô hiệu hóa cánh cửa bằng cách đặt game object door không hoạt động
            door.SetActive(false);
            // Vô hiệu hóa chìa khóa bằng cách đặt game object keyObject không hoạt động
            keyObject.SetActive(false);

            // Xóa chìa khóa sau khi sử dụng
            Destroy(gameObject);
        }
    }
}
```

Hình 47: Script cho các đối tượng nhiệm vụ cần làm

Script trên sẽ quản lý hành vi của chìa khóa trong trò chơi Unity. AudioManager là lớp quản lý âm thanh, được tìm bằng FindGameObjectWithTag với tag "Audio" và lưu vào biến audioManager. Biến door và keyObject là đối tượng cánh cửa và chìa khóa. Biến isOpened là cờ chỉ trạng thái cửa đã mở hay chưa. Phương thức OnTriggerEnter kiểm tra va chạm với người chơi và trạng thái cửa, mở cửa và làm biến mất chìa khóa nếu cần. Cuối cùng, hủy đối tượng chìa khóa sau khi sử dụng.

❖ Thực hiện cấu hình Script

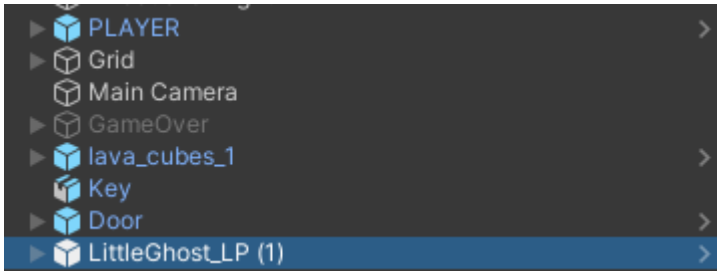
- Kéo thả hai đối tượng và ở đối tượng “Key” chọn sự kiện để kích hoạt.



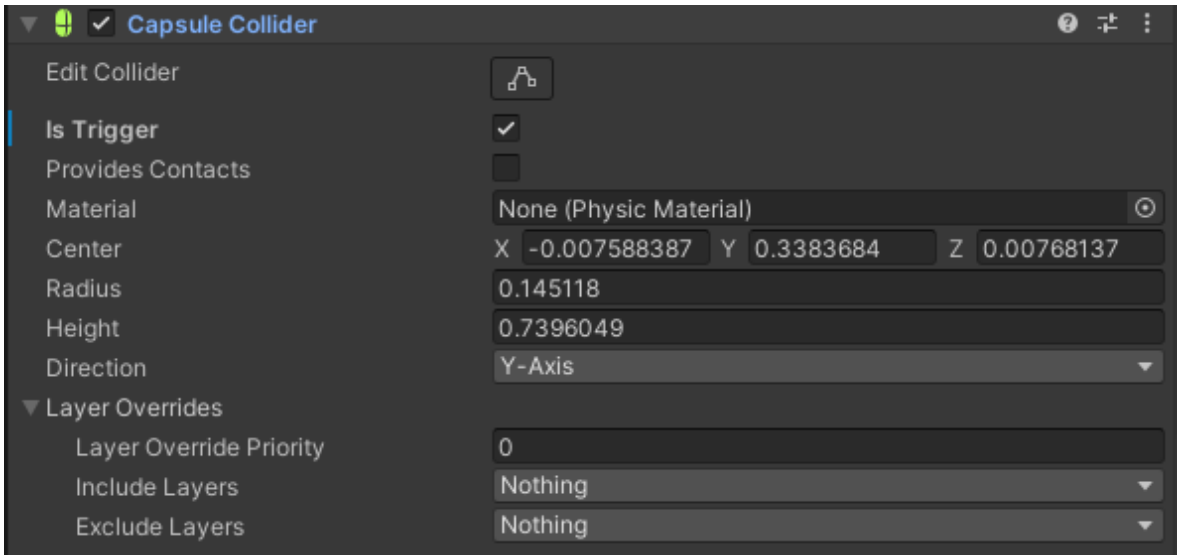
Hình 48: Gắn script cho đối tượng Key và cấu hình

2.7.3 Thiết kế cơ chế AI sử dụng A*Pathfinder cho đối thủ trong trò chơi

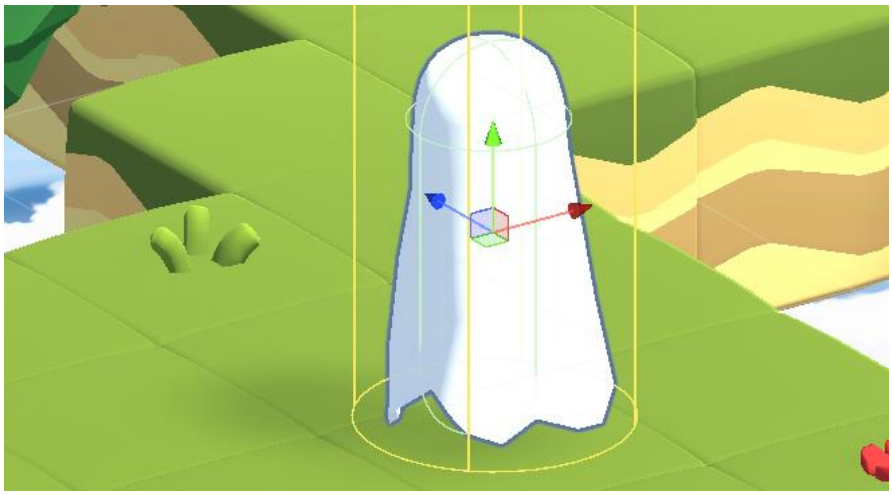
Đầu tiên sẽ có một nhân vật người chơi và trong phần này ta sẽ làm việc nhiều nhất trên đối thủ.Đầu tiên là đối thủ này sẽ có một collider



Hình 49: Đối tượng đối thủ



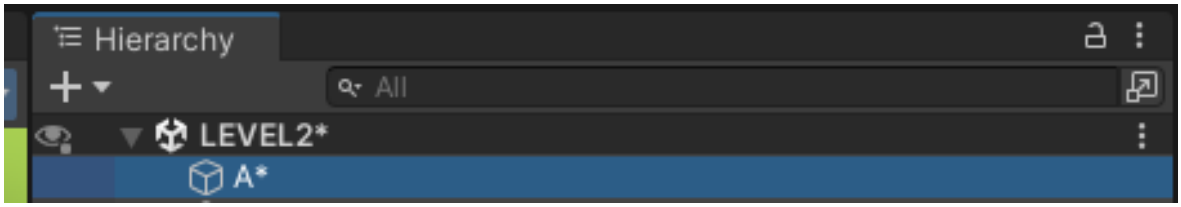
Hình 50: Đặt collider đối tượng đối thủ



Hình 51: Collider đối tượng đối thủ hiển thị màu xanh lá

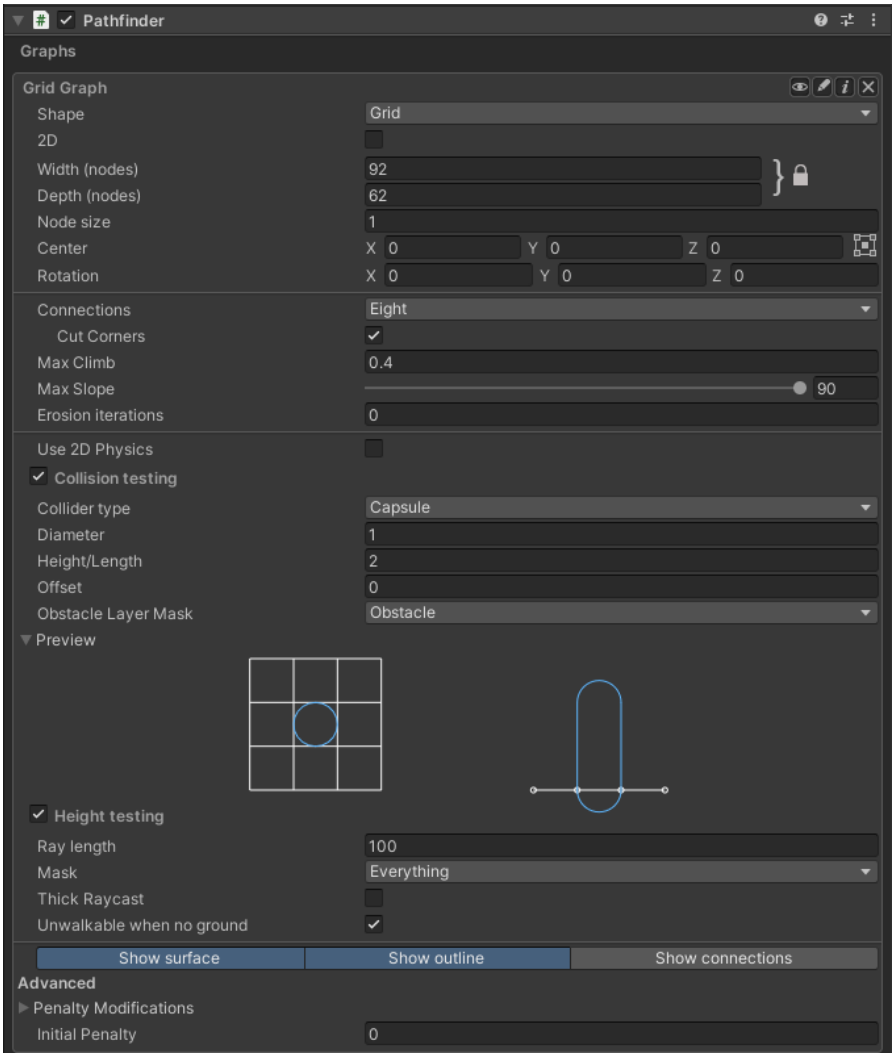
Sử dụng một packet được gọi là A* pathfinding sau đó tải về Package và import Package đó.

Tiếp theo với bất kỳ thuật toán nào thì phải định nghĩa trước bản đồ sẽ trong như thế nào nên tiếp theo sẽ tạo một GameObject (một đối tượng) mới có tên là A*, đây là đối tượng trọng tâm, nó sẽ quản lý việc bản đồ sẽ trong như thế nào và định nghĩa đối tượng nào là vật cản.



*Hình 52: Tạo đối tượng A**

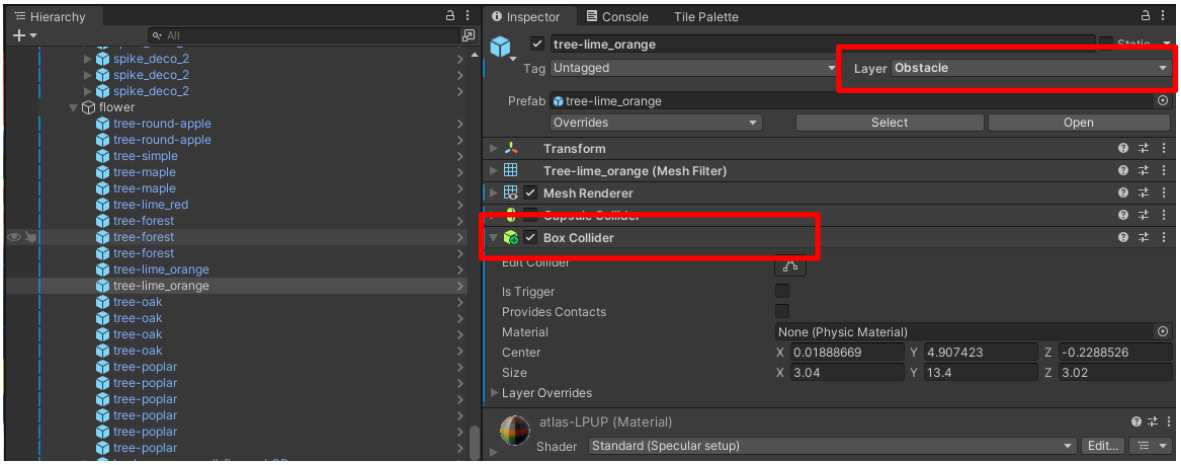
Ở tab Inspector tìm kiếm component tên là Pathfinder



Hình 53: component Pathfinder

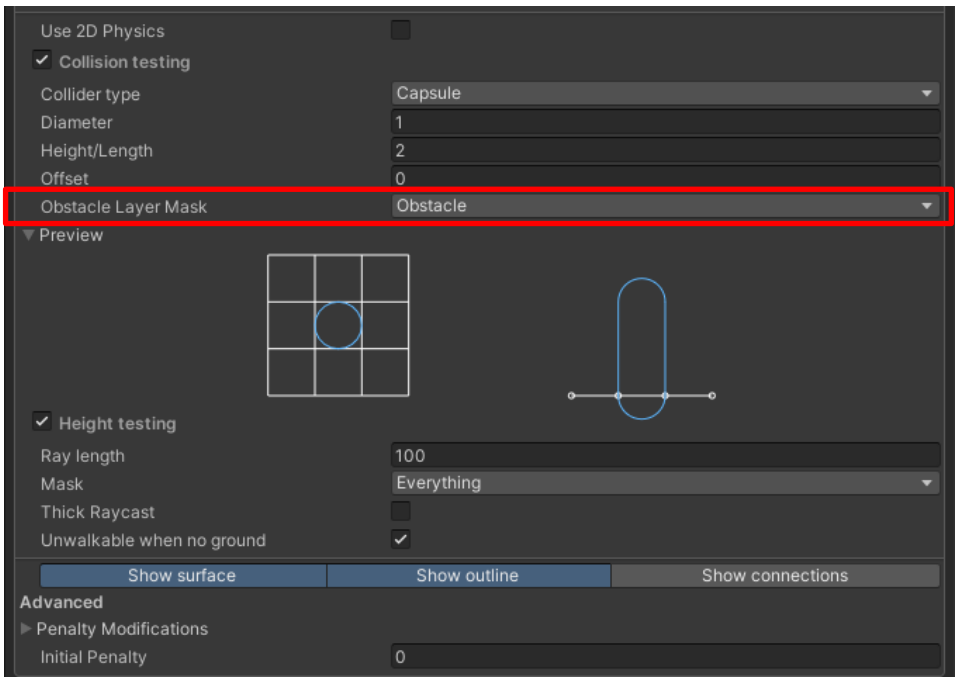
Tạo Grid Graph (lưới bản đồ) sao đó điều chỉnh độ dài và chiều cao của bản đồ. Có thể điều chỉnh cho nó giữa bản đồ bằng cách cho điều chỉnh mục center sao cho phù hợp.

Tiếp theo định nghĩa các vật cảng bằng cách cho các đối tượng đã tạo mang tính vật cản và định nghĩa lại Layer (lớp) mang tên Obstacle sau khi đã thêm Collider cho các vật cản đó.

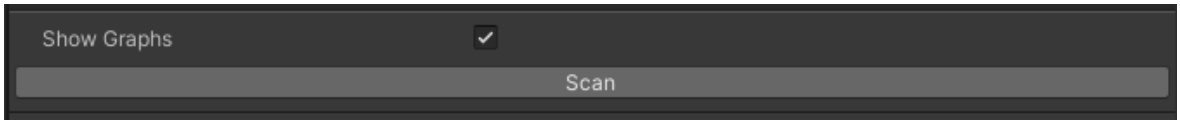


Hình 54: Định nghĩa các vật cản

Vào lại A* ở mục Obstacle Layer Mask mặc định của nó là Nothing nên sẽ đổi thành Obstacle và Scan lại bản đồ



Hình 55: Chọn layer Obstacle

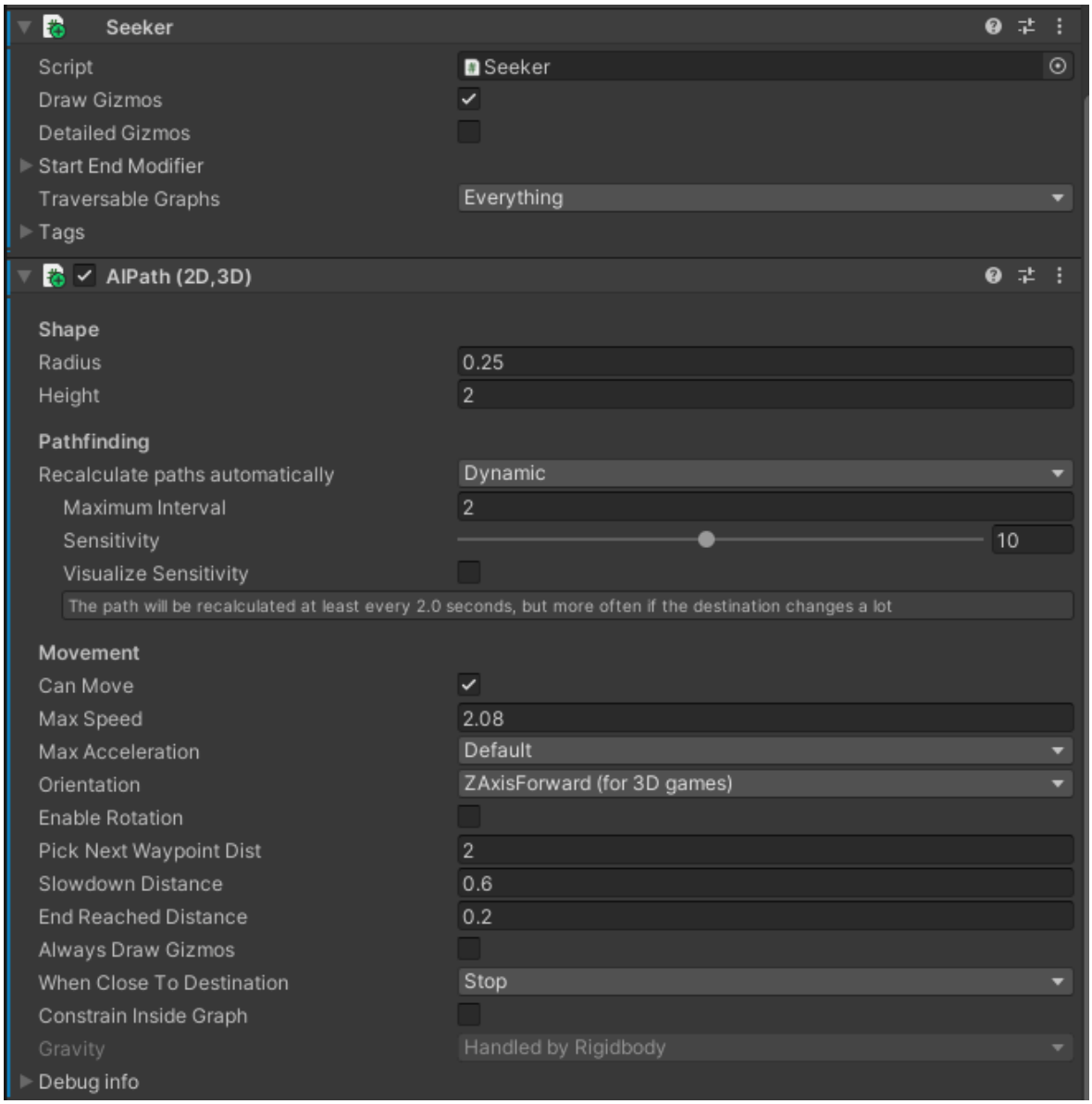


Hình 56: Nhấn scan để quét bản đồ

*Hình 57: Bản đồ A**

⇨ Như vậy có thể thấy rằng bản đồ đã được định nghĩa rằng các đối tượng cây cối hay các bụi nhỏ sẽ là vật cản và vùng màu xanh sẽ là vùng mà đối thủ có thể di chuyển.

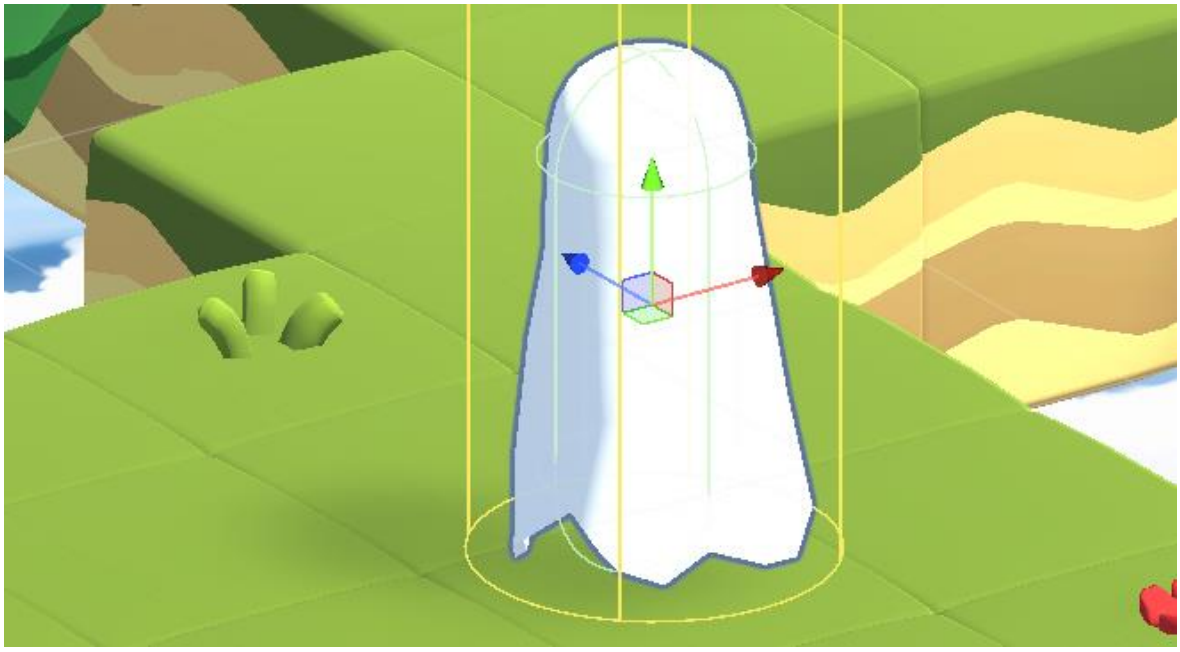
Tiếp theo sẽ thêm một component để đối thủ tự động di chuyển theo nhân vật.Sử dụng component có tên là AIPath và khi thêm vào nó sẽ tự động thêm một component có tên là Seeker thì Seeker này nó sẽ quản lý việc tìm đường đi cũng như vẽ cái dường đi đó.



Hình 58: Component AIPath và Seeker

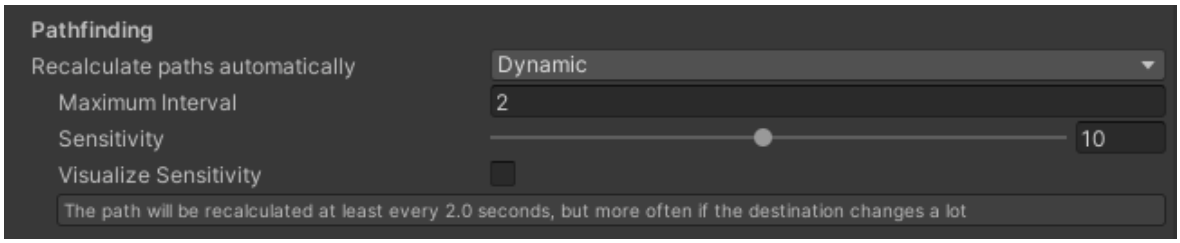
Các thành phần chính của component này:

+ **Orientation:** nó sẽ tạo hình dạng xung quanh đối thủ và nó sẽ hoạt động giống như Collider của đối thủ nhưng mà sẽ hoạt động trong bản đồ A* và có thể thay đổi mục Shape sao cho phù hợp.



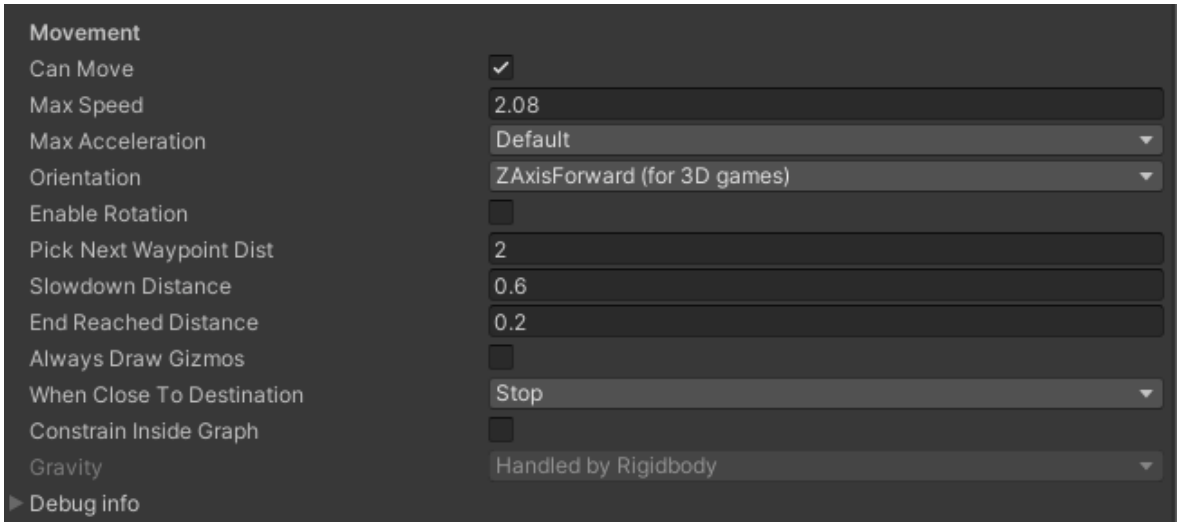
Hình 59: Orientation cho đối thủ được hiển thị vàng xung quanh

+ Pathfinding: Nó sẽ định nghĩa tính toán lại đường theo một chế độ mà người tạo đã sử dụng. Ví dụ có thể sử dụng Never có nghĩa nó sẽ không tính toán lại đường đi này nữa. Trong game kẻ thù sẽ đuổi theo người chơi thì người chơi ban đầu sẽ ở một vị trí cố định thì đối thủ nó sẽ tự động di chuyển đến vị trí ban đầu của người chơi.Nhưng trong khi chơi người chơi sẽ di chuyển đối tượng Player sang một một vị trí khác và nếu thiết lập như vậy thì đối thủ sẽ không biết được. Khi đó ta phải tính toán lại đường đi để kẻ thù có thể di chuyển nơi mà đối tượng Player đang đứng ở vị trí mới.Trong mặc định thì sau 2 giây nó sẽ tính toán lại đường đi.



Hình 60: Mục Pathfinding

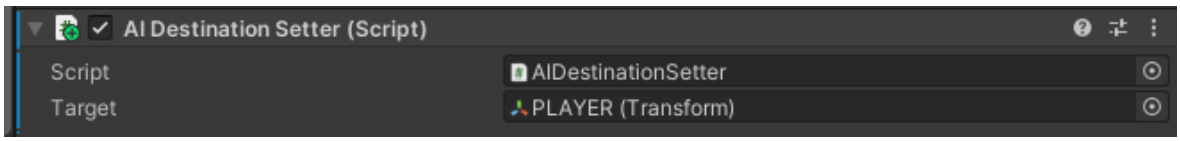
+ Movement: Cho đối thủ có thể di chuyển và chỉnh sửa tốc độ di chuyển sao cho cân bằng trong game. Mục gravity là trọng lực của đối tượng nên sẽ không tắt nó



Hình 61: Mục Movement

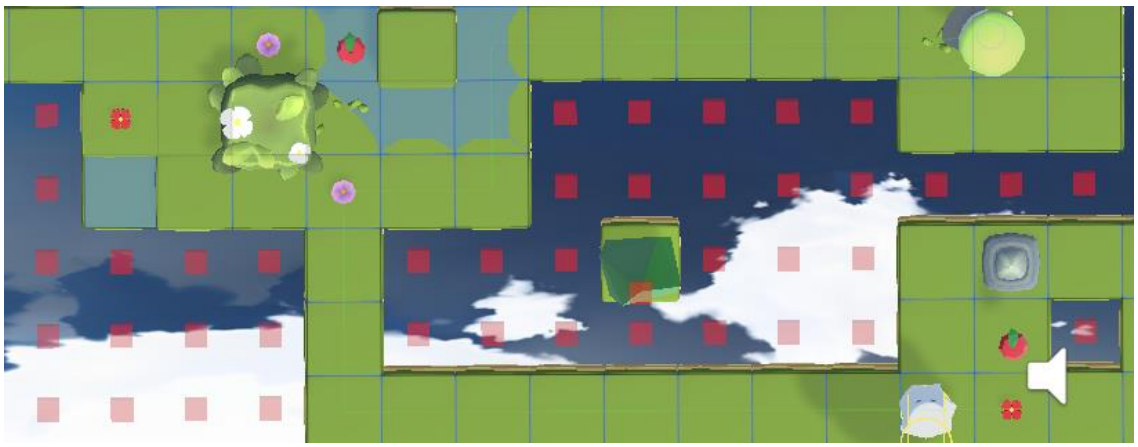
+ Pick Next Waypoint Dist: Khoảng cách mà nó sẽ tìm đến điểm tiếp theo mà nó sẽ di chuyển trong bản đồ.Điểm tiếp theo ở đây đó là nó sẽ di chuyển một ô trong bản đồ và nó sẽ đi hết,nó không di theo thẳng một đường.

Thêm một component tên là AIDestination Setter giúp con AI đối thủ biết được là nó sẽ đi theo Player.Để thiết lập thì ta sẽ kéo đối tượng Player vào cấu hình.



Hình 62: Component AIDestination Setter giúp AI đối thủ xác định đi theo đối tượng nào

Sau khi thiết lập xong ta chạy ứng dụng lên và kết quả đạt được là đối thủ sẽ di chuyển theo nhân vật.Khi nhân vật di chuyển thì sẽ thấy một đường xanh là đối thủ sẽ di chuyển theo đường đó.

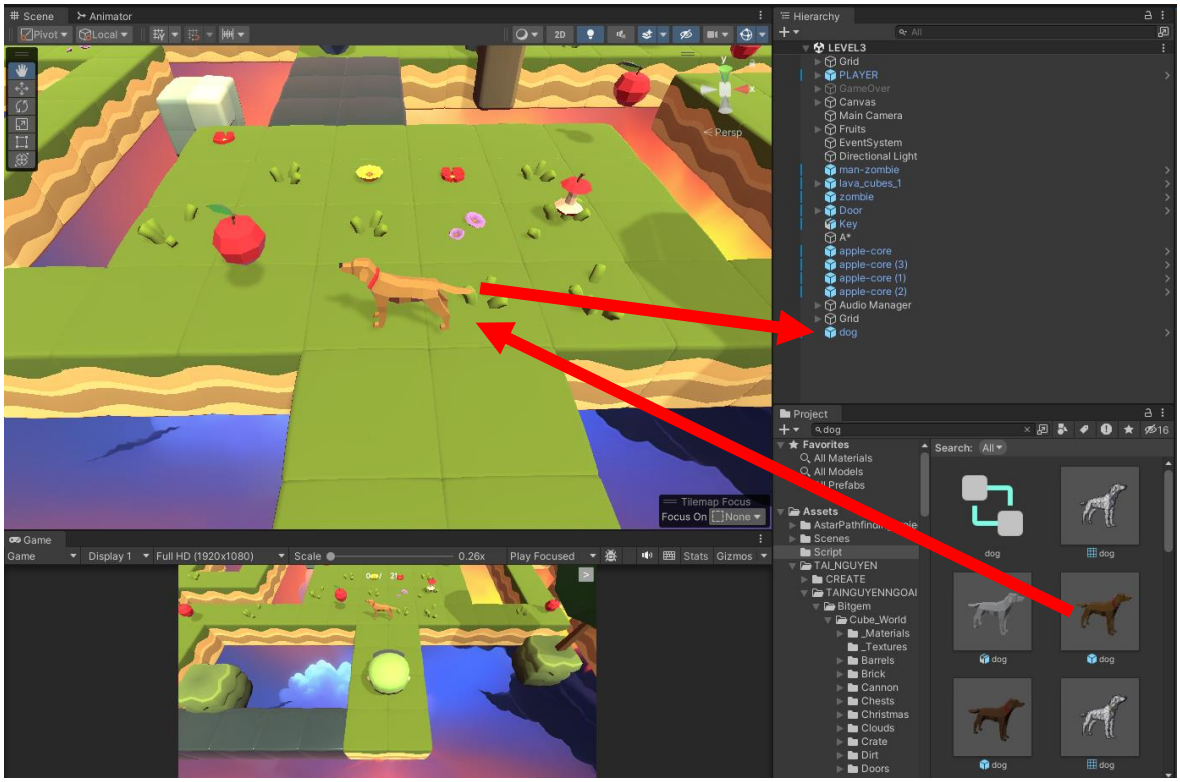


Hình 63: Hiển thị đường đi ngắn khi khởi động game

2.7.4 Thiết kế định nghĩa một đối tượng mới

Đối với các admin người thiết kế các màn chơi có thể chỉnh sửa,định nghĩa thêm một đối tượng mới từ đó trò chơi có thể được cập nhật sau khi xuất ra trò chơi chính thức .Ở đây sử dụng màn cuối cùng để định nghĩa một đối tượng mới sẽ như thế nào.

+ Thêm đối tượng là mô hình một chú chó (dog) vào bằng cách kéo thả vào trong địa hình.Điều đó đối tượng được xuất hiện trên tab Hierarchy.



Hình 64: Thêm một đối tượng vào địa hình ở màn cuối

❖ Đối tượng này ta cho nó một vài định nghĩa thêm như:

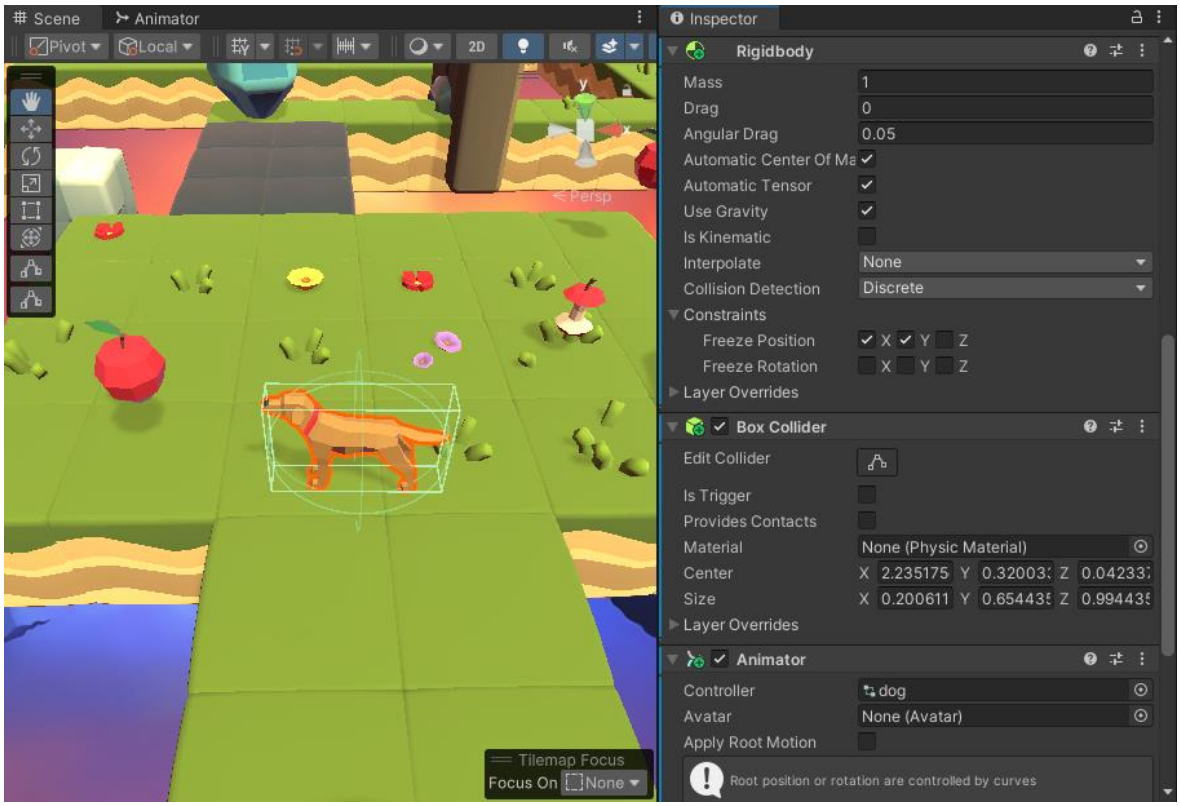
+ Box Collider là một thành phần của đối tượng trong một trò chơi hoặc môi trường 3D trong công nghệ trò chơi điện tử. Nó định nghĩa một hình hộp (box) xung quanh đối tượng, và được sử dụng để xác định khu vực nào của không gian mà đối tượng đó chiếm dụng và tương tác được trong trò chơi. Box Collider thường được sử dụng để xác định va chạm giữa các đối tượng, điều này rất quan trọng trong việc xây dựng hệ thống vật lý và tương tác trong một trò chơi. Khi hai Box Collider xung quanh hai đối tượng va chạm vào nhau, hành động xử lý va chạm có thể được kích hoạt, cho phép các hành động như đẩy lùi, gây sát thương hoặc kích hoạt các hiệu ứng khác.

+ Rigidbody là một thành phần quan trọng trong các trò chơi và môi trường 3D trong lập trình trò chơi điện tử. Nó được sử dụng để mô phỏng vật lý của các đối

tượng trong trò chơi, cho phép chúng di chuyển, quay và tương tác với các yếu tố khác trong môi trường 3D. Khi một đối tượng được gắn với một Rigidbody, nó trở nên ảnh hưởng bởi lực vật lý như trọng lực, ma sát và va chạm. Rigidbody cung cấp các thuộc tính như vận tốc, gia tốc và moment xoắn để điều khiển chuyển động của đối tượng. Điều này cho phép các đối tượng di chuyển tự nhiên trong không gian 3D dựa trên các nguyên tắc vật lý.

+ Animator là một công cụ trong lập trình trò chơi điện tử, giúp tạo ra các hoạt ảnh cho các đối tượng trong trò chơi.

+ Ngoài ra có thể gắn script " Trap " cho đối tượng giúp người chơi chạm vào nó thì lập tức màn hình GameOver xuất hiện.

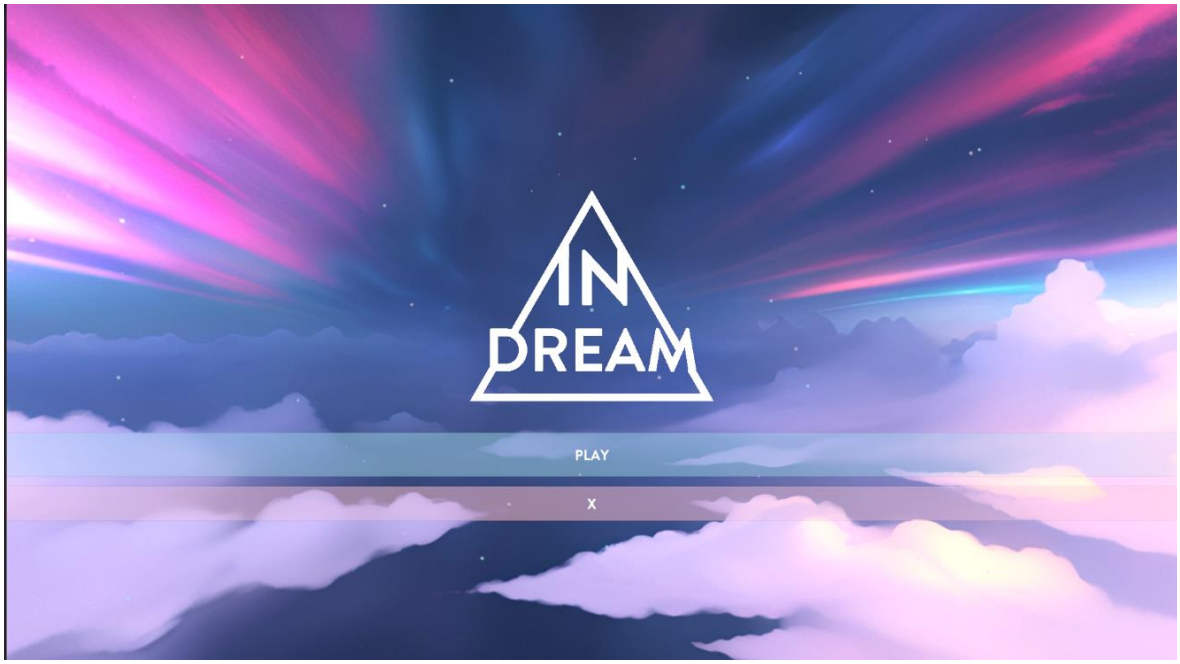


Hình 65:Thêm các định nghĩa về vật lý,vật cản.

3. Giao diện ứng dụng

3.1 Giao diện main menu

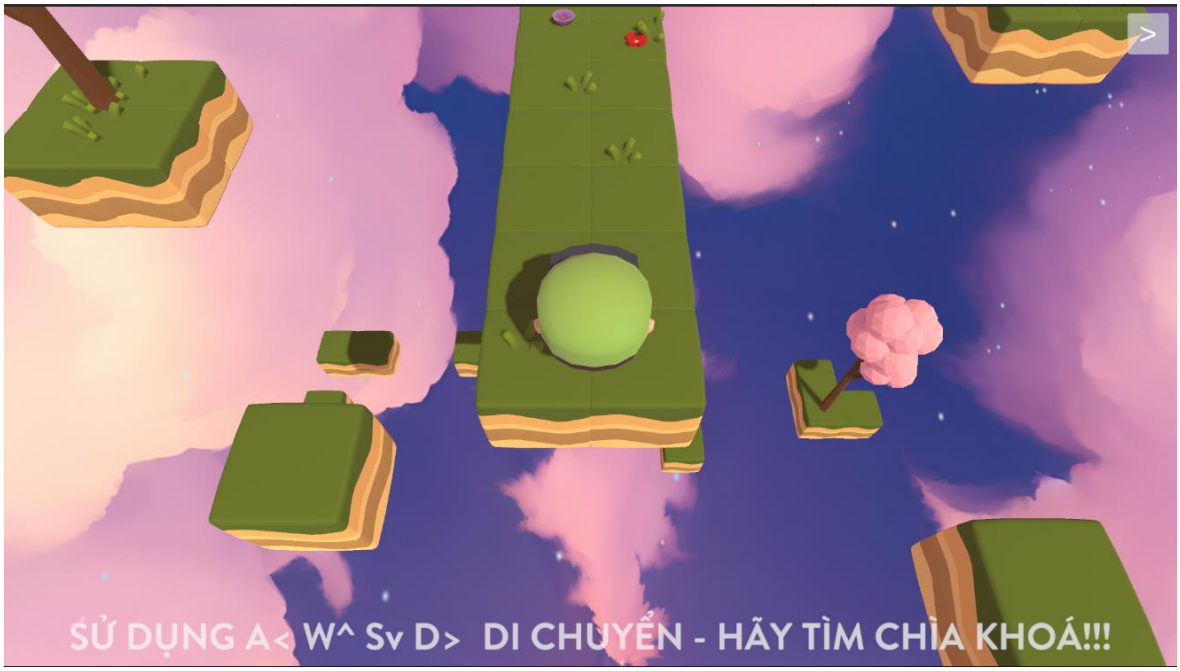
Khi bắt đầu vào trò chơi màn hình chính sẽ có hai nút tương ứng hai chức năng đó là nút chơi và nút thoát.Khi nhấn vào nút chơi (PLAY) thì sẽ bắt đầu vào màn hướng dẫn và nút thoát (X) nó sẽ thoát khỏi màn hình và đóng ứng dụng lại.



Hình 66: Giao diện menu

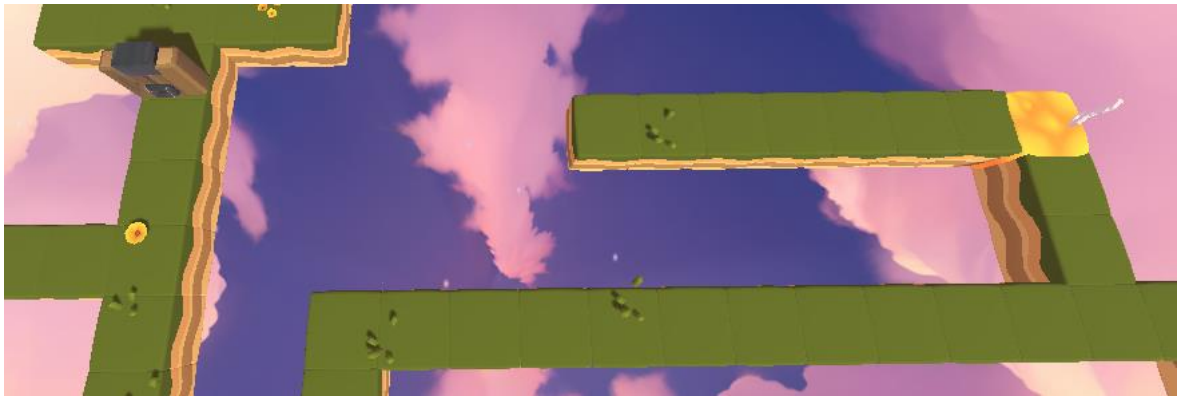
3.2 Giao diện màn hướng dẫn

Sau khi vào game cụ thể màn hướng dẫn sẽ cho ta biết được cách di chuyển và thực hiện nhiệm vụ như tìm chìa khoá và thu thập.



Hình 67: Giao diện màn hướng dẫn

Hình 68: Giao diện môi trường xung quanh màn hướng dẫn



Hình 69: Giao diện nhiệm vụ tìm chìa khoá và đối tượng cửa

3.3 Giao diện màn 1

Sau khi thu thập một vật phẩm cụ thể là kim cương thì mọi thứ sẽ rung chuyển và sẽ chuyển sang đến màng 1.Ở màn này nhiệm vụ sẽ thu thập các trái cây theo yêu cầu để có thể vượt qua màn chơi nhưng đồng thời có một đối thủ sẽ di chuyển theo người chơi và người chơi cần phải tránh đối thủ.

Hình 70: Giao diện màn 1

3.4 Giao diện màn 2

Sau khi thu thập đầy đủ thì nhân vật mà chúng ta điều khiển sẽ dịch chuyển đến màn tiếp theo.Màn này độ khó sẽ cao hơn do có sự xuất hiện của các bẫy và một đối thủ có sử dụng thuộc tính tìm đường đi đến nhân vật ngắn nhất để tiếp cận.Tuy vậy màn này sẽ có các bụi nhỏ người chơi có thể đi qua nó và đối thủ sẽ không thể qua bụi nhỏ nên sẽ tìm đường khác.Nên là nhân vật cần phải tránh bẫy và quan sát xung quanh để có thể thu thập.

Hình 71: Giao diện màn 2

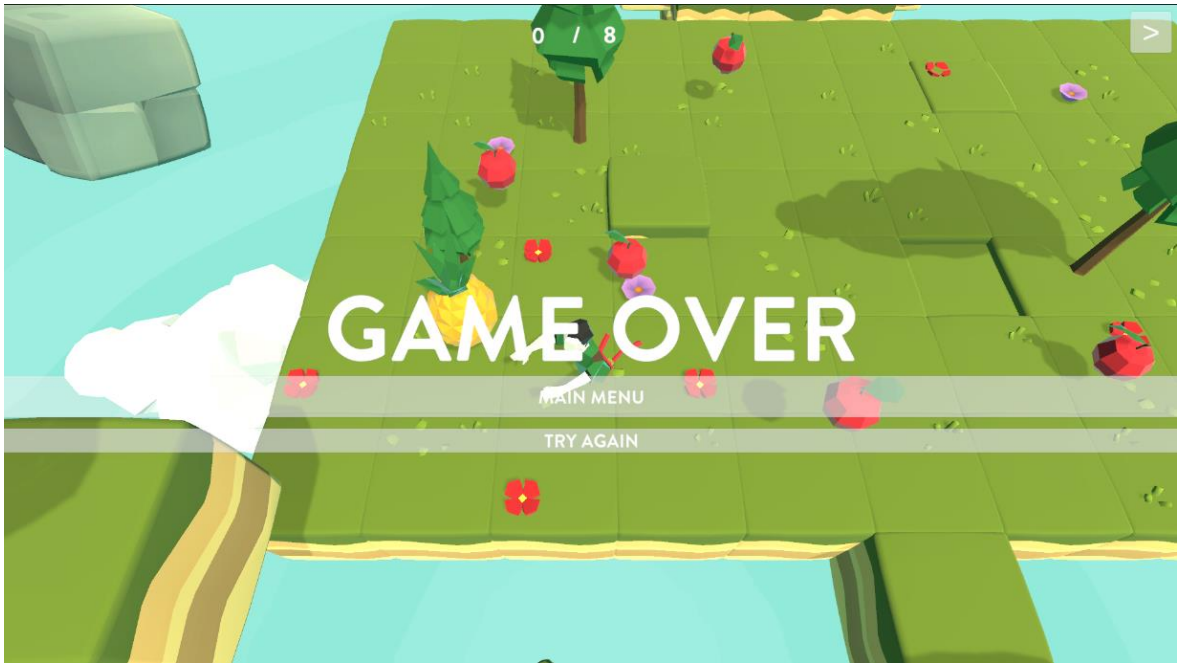
3.5 Giao diện màn 3

Sau khi nhân vật thu thập đầy đủ người chơi sẽ dịch chuyển đến màn cuối cùng.Màn này sẽ khó nhất vì sẽ xuất hiện các trái cây khi chạm vào người chơi sẽ thua cuộc do trái cây có dính độc và khi chạm vào thì lập tức thua đồng thời sẽ có hai đối thủ, một đối thủ có thể di chuyển trên không và một đối thủ sẽ tìm đường ngắn nhất để tiếp cận. Tuy vậy bạn nhận được tốc độ di chuyển nhanh hơn nên người chơi sẽ phản xạ tốt hơn.

Hình 72: Giao diện màn 3

3.6 Giao diện GameOver

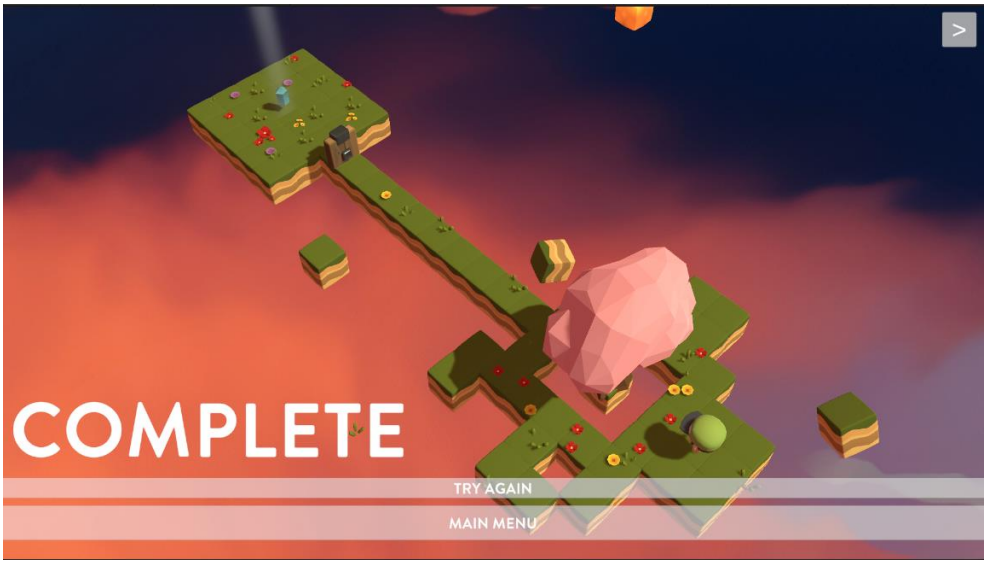
Khi bạn bị đối thủ hay các bẫy chạm vào bạn sẽ bị thua cuộc nhưng bạn vẫn có thể chơi lại màn hoặc có thể thoát ra menu để có thể chơi lại từ đầu hoặc thoát khỏi trò chơi.



Hình 73: Giao diện GameOver

3.7 Giao diện kết thúc

Sau khi vượt qua tất cả các màn và trả lại kim cương mọi thứ sẽ trở lại bình thường và bạn sẽ hoàn thành xong trò chơi.Bạn có thể chơi lại từ đầu hoặc ra menu.



Hình 74: Giao diện kết thúc hoàn thành trò chơi

CHƯƠNG 3
KIỂM THỬ VÀ ĐÁNH GIÁ

1. Mục tiêu

Mục tiêu chính đó là xem có lỗi hay có vấn đề về trò chơi đó hay không, nếu có lỗi thì cần khắc phục và sửa chữa kịp thời nhằm phát triển theo đúng kế hoạch và mục tiêu được đề ra.

2. Nghi thức kiểm tra

Một số nội dung kiểm thử:

+ Kiểm tra các giao diện có hoạt động khi tương tác không.
+ Kiểm tra các nút các phím để điều khiển nhân vật có di chuyển hay không.
+ Kiểm tra hiển thị các con số thu thập có chính xác không.
+ Kiểm tra số lần chơi lại xem ứng dụng có bị văng hay không.
+ Kiểm tra hiệu năng của trò chơi.
+ Kiểm tra các thành phần nhiệm vụ của trò chơi khi tương tác có hoạt động hay không.
+ Kiểm tra các đối thủ có hoạt động được không.
+ Kiểm tra đối thủ có đi theo nhân vật hay không.
+ Kiểm tra hiệu năng khi di chuyển các màn chơi.
+ Kiểm tra các hiệu ứng có hoạt động hay không.

3. Kết quả kiểm tra

Kết thúc quá trình kiểm thử các chức năng hay giao diện của ứng dụng trò choi ,từ các kiểm thử trên cho thấy trò chơi hoạt động khá tốt và đã đạt yêu cầu.Tuy vậy hiệu năng của mỗi máy sẽ ảnh hưởng không nhiều đến chất lượng khi chơi.

PHẦN KẾT LUẬN

1. Kết quả đạt được

Qua quá trình nghiên cứu và thực hiện đề tài tôi đã mang lại các kết quả sau:

- Về ưu điểm

+ Thiết kế,cài đặt, và xây dựng trò chơi đáp ứng yêu cầu đề ra.

+ Giao diện thân thiện.

+ Cơ chế trò chơi dễ tiếp cận nhưng cũng tạo nên thử thách.

+ Tạo nên tính phản xạ nhanh và cách quan sát một cách tốt nhất.

+ Có thể chơi để thư giãn sau buổi học tập hoặc hoàn thành xong công việc.

+ Biết được các thiết lập xây dựng cơ bản trên Unity và xây dựng các Script xử lý các đối tượng cơ bản.

- Hạn chế

+ Do Unity là một đề tài khá mới lạ và đây cũng là lần đầu được thực hiện đề tài nên còn phụ thuộc về tài liệu hướng dẫn về công cụ, sử dụng tài nguyên hay các mô hình bên ngoài để thực hiện và chưa có kinh nghiệm cũng như am hiểu sâu về các kiến thức hay các nguyên lý về các giải thuật. Và còn rất nhiều kiến thức các công cụ về Unity vẫn chưa tìm hiểu hoàn toàn hết.

+ Do thời lượng tạo ra một trò chơi không ngắn cũng không dài nên còn có một số hạn chế nhất định như: Thời lượng chơi ngắn, Chưa có nhiều màn chơi, Các hiệu ứng chưa được bắt mắt.

2. Hướng phát triển

Dù ứng dụng trò chơi này hoạt động tốt tuy vậy vẫn còn hạn chế rất nhiều cho nên trò chơi và bản thân cần thêm thời gian để cải thiện thêm như:

+ Bổ sung nhiều màn chơi hơn và điều chỉnh độ khó cho mỗi màn sao cho phù hợp.

+ Điều chỉnh giao diện màn hình phù hợp với các thiết bị

+ Thêm điểm sao lưu khi thoát trò chơi

+ Thêm tính năng chơi cùng bạn bè để hỗ trợ vượt qua màn dễ hơn

+ Đối với bản thân cần có nhiều thời gian để tìm hiểu các nguyên lý hoạt động của các thuật toán hơn. Và cần học hỏi thêm về công cụ Unity và các phần mềm tạo các mô hình để tránh bản quyền hình ảnh.

TÀI LIỆU THAM KHẢO

[1]. Giải thuật tìm kiếm A* Documents

https://vi.wikipedia.org/wiki/Giải_thuật_tìm_kiếm_A*#:~:text=Trong%20khoa%20học%20máy%20tính,mãn%20một%20điều%20kiện%20đích).

[2]. A*Pathfinding Documents:

https://viblo.asia/p/a-pathfinding-nwmkyEjlkoW

[3]. A*Pathfinding Project:

https://arongranberg.com/astar/

[4]. Learn Unity online:

https://unity.com/learn/

[5]. Learning Pathway Unity:

https://learn.unity.com

[6]. Youtube learn A*:

https://www.youtube.com/watch?v=ZkFyA5y7_xQ

[7]. Youtube learn Unity :

https://www.youtube.com/@SaiGame/videos

[8]. Learn collect item and custom:

https://www.youtube.com/watch?v=EfUCEwKmcjc&t=9s

PHỤ LỤC

1. AI Được áp dụng với nội dung nào.

Trong thuật toán A* Pathfinder, AI được áp dụng để tìm kiếm đường đi tối ưu từ một điểm đến một điểm khác trên một bản đồ hoặc một mạng lưới. Điều này có nghĩa là AI được sử dụng để xác định con đường tối ưu dựa trên các hệ số như chi phí và độ ưu tiên để di chuyển từ điểm bắt đầu đến điểm kết thúc.

2. Các nội dung có trên hệ thống đã có hỗ trợ.

- **Phần camera và phần ánh sánh**:Khi tạo mới một cảnh sẽ có sẵn hai đối tượng này. Camera đại diện cho màn hình sẽ hiển thị và ánh sáng đại diện ánh sáng của mặt trời.

- **Transform:** Component này định nghĩa vị trí, quay và tỉ lệ của một đối tượng trong không gian 3D hoặc 2D.

- **Renderer:** Renderer quản lý cách một đối tượng được hiển thị trong trò chơi hoặc ứng dụng, bao gồm việc vẽ hình ảnh và đồ họa.

- **Collider:** Collider định nghĩa khu vực không gian mà một đối tượng chiếm giữ, để xác định va chạm với các đối tượng khác.

- **Scripting Language:** Unity hỗ trợ việc phát triển ứng dụng bằng nhiều ngôn ngữ lập trình khác nhau, như C#, JavaScript và Boo. C# là ngôn ngữ lập trình chính được khuyến nghị sử dụng bởi Unity.

- **Rigidbody:** Rigidbody là một Component để mô phỏng vật lý của đối tượng trong không gian 3D, bao gồm trọng lực, ma sát và va chạm.

- **Animator:** Component này được sử dụng để tạo và điều khiển hoạt ảnh (animation) cho các đối tượng, bao gồm cả nhân vật và đối tượng không động.

- **Audio Engine:** Unity cung cấp một bộ công cụ để quản lý và phát âm thanh trong trò chơi, bao gồm hỗ trợ cho âm thanh 2D và 3D, âm thanh đa kênh và hiệu ứng âm thanh.

- **Particle System:** Particle System là một Component để tạo ra và điều khiển các hiệu ứng hạt (particle effects) như hỏa, khói, bụi, vv.

- **User Interface (UI):** Unity cung cấp các thành phần để thiết kế và điều khiển giao diện người dùng.

- **Unity Tilemap:** Unity Tilemap là một công cụ mạnh mẽ cho việc xây dựng các bản đồ với lưới các ô vuông.

- Asset Store (Cửa hàng Unity): Unity cung cấp một cửa hàng trực tuyến cho phép người dùng tải về và chia sẻ tài nguyên như mẫu mô hình 3D, mẫu 2D, hiệu ứng âm thanh, và mã nguồn mở.

2. Các nội dung em đã đóng góp

- Xây dựng bản đồ, địa hình và môi trường trong trò chơi:Sử dụng các mô hình và tự thiết kế các màn chơi.

- Thiết kế giao diện trò chơi bao gồm: Tự thiết kế logo, thêm các nút chơi và thoát, giao diện GameOver,các text trong trò chơi.

- Viết Script "Respawn" : Trở về một vị trí khi nhân vật rơi xuống bằng cách Kiểm tra xem vị trí hiện tại của đối tượng có nhỏ hơn ngưỡng respawn hay không. Ở đây, transform.position đại diện cho vị trí của GameObject mà script được gắn vào, và ".y" là tọa độ theo trục y của vị trí đó và sẽ được đặt lại về ví trị mà đối tượng đã rơi xuống.

- Viết Script "Trap" : Xử lý va chạm dùng chung cho hai đối tượng đối thủ và bẫy giúp xử lí va chạm và hiện lên giao diện GameOver.

- Viết Script "ChucnangMenu" : Chuyển cảnh khi bấm vào nút chơi hoặc thoát trong giao diện menu.

- Viết Script "Camerafollow", "CameraFOV" : Các hiệu ứng của tầm nhìn về giao diện và camera di chuyển theo người chơi giúp trên màn hình khi nhân vật di chuyển vị trí nào thì màn hình sẽ di chuyển theo nhân vật đó.

- Viết Script "SimpleRotation" ,"TextEffect", "Diamond" : Hiệu ứng xoay hay lên xuống của vật thể, hiệu ứng chữ, Hiệu ứng rung.

- Canh chỉnh bản đồ A* khi sử dụng công cụ : Thiết lập định nghĩa vật cản của đối tượng phù hợp và canh chỉnh thông số cho đối thủ AI sao cho phù hợp.

- Viết Script "PlayerInventory" và "Apple" Quản lý kho thu thập: khi thu thập số lượng cái cây nhất định lập tức qua màn chơi tiếp theo và xử lý va chạm cho các trái cây để tính điểm.

- Tạo ra cơ chế đối tượng cửa và chìa khoá và Viết script xử lý.Đây là hai đối tượng được thiết kế ra nhằm thêm một nhiệm vụ để có thể thu thập thêm trái cây .Bằng cách xử lí va chạm khi đối tượng người chơi chạm vào nó sẽ biến mất cả hai đối tượng

- Viết script "NVDchuyen" Làm nhân vật di chuyển khi sử dụng phím A,W,S,D

- Làm các hiệu ứng chuyển động với Animation cho các đối tượng môi trường.

- Gán âm thanh nhạc nền, âm thanh khi xử lý va chạm.