TRƯỜNG ĐẠI HỌC MỎ-ĐỊA CHẤT

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO TỔNG KẾT SINH HOẠT HỌC THUẬT

Đề tài: Xây dựng Game

trong Công nghệ đa phương tiện

Người báo cáo: **Dương Chí Thiện**
Bộ môn Hệ thống thông tin- Tri thức

Năm 2022

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Lý do chọn đề tài

Trong những năm gần đây, khoa học và kỹ thuật phát triển mạnh mẽ, công nghệ cũng có những bước tiến vượt bậc đặc biệt là về mảng điện thoại di động. Điện thoại di động là thiết bị tiện ích, dễ sử dụng, nhỏ gọn, có thể sử dụng mọi lúc, mọi nơi.... Các App dần dần xuất hiện và ngày càng phát triển hơn về cả số lượng và chất lượng.

Các ứng dụng điện thoại xuất hiện nhằm phục vụ nhu cầu tất yếu của người dùng như: chơi game, nghe nhạc, chụp ảnh, quay video, xem phim… Nhu cầu của người dùng càng cao, App xuất hiện càng nhiều. Tuy nhiên, không phải App nào được làm ra cũng có chất lượng tốt.

Gần đây nhu cầu giải trí của người dùng ngày càng cao ai cũng muốn giải trí giải tỏa tinh thần sau những ngày làm việc mệt nhọc. Vì thế họ tìm đến các hoạt động giải trí ngoài trời nhằm thư giãn đầu óc. Tuy nhiên không phải ai cũng có thời gian, cơ hội để mà tham gia các hoạt động giải trí ngoài trời hoặc là họ không thích đi xa. Chính vì thế mà họ tìm đến thú vui bằng các game ngay trên điện thoại di động của mình. Do biết được nhu cầu của người dùng mà tôi tham gia xây dựng một ứng dụng game Mario trên Unity nhằm phục vụ nhu cầu giải trí của những người sử dụng.

Game Mario là 1 thể loại game 2d rất thận thiện với người dùng từ các trò chơi trên dòng máy Nintendo và mỗi phiên bản đều có sự độc đáo riêng bạn sẽ đến với anh chàng sửa ống nước, để đi cứu công chúa và phải vượt quá rất nhiều thử thách và các con quái vật.

1.2 Giới hạn và phạm vi của đề tài

Đề tài xây dựng game Mario sử dụng công cụ Unity và Công cụ lập trình Visual Studio Code

1.3 Nội dung thực hiện

Nội dung thực hiện/nghiên cứu cụ thể như sau:

- Xây dựng hệ thông ứng dụng lạ, độc , dễ nhìn.
- Thiết kế đặc tả hệ thống quán lý rõ ràng,nhất quán.
- Xây dựng chắc năng cơ bản của game.
- Thiết kế đắc tả hệ thống
- Kiểm thử hệ thống.
- Triển khai thực nghiệm hệ thống máy tính và Smartphone.

1.4 Phương pháp tiếp cận

- Cách tiếp cận : Trực tiếp tham gia sử dụng các ứng dụng game đã được xây dựng để rút ra các kinh nghiệm để xây dựng game của mình được tốt hơn, chế độ phục vụ tốt hơn.
- Sử dụng các phương pháp nghiên cứu: đọc tài liệu trên internet

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về Unity

2.1.1 Đối tượng tham gia hệ thống

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

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

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

2.1.2 Lịch sử của Unity

Ngày nay, con người dành khá nhiều thời gian giải trí bên những chiếc smartphone cùng những tựa game yêu thích. Trong số đó có không ít trò chơi được lập trình dựa trên engine Unity 3D đã ra đời cách đây hơn một thập kỉ. Trải qua thời gian phát triển lâu dài và luôn update công nghệ mới, giờ đây Unity 3D đã trở thành lựa chọn số 1 cho bất cứ lập trình viên nào muốn xây dựng một tựa game có thể sử

dụng đa nền tảng, chi phí rẻ và dễ thao tác. Tuy rất phổ biến những thực tế ít ai biết được nguồn gốc và lịch sử phát triển của engine này.

Vào đầu những năm 2000, ba lập trình viên trẻ là David Helgason (CEO), Nicholas Francis (CCO), và Joachim Ante (CTO) với nguồn kinh tế eo hẹp đã tập trung tại một tầng hầm và bắt đầu lập trình ra thứ mà sau này trở thành một trong những phần mềm được ứng dụng rộng rãi nhất trong ngành công nghiệp video game.

Năm 2008, với sự gia tăng người dung của iPhone, Unity là một trong những nhà phát triển Engine Game đầu tiên để bắt đầu hỗ trợ các nền tảng đầy đủ. Unity bây giờ hỗ trợ 24 nền tảng, bao gồm cả Oculus Rift , PlayStation 4 và Linux.

Năm 2010, IBM bắt đầu tìm hiểu Unity 3D dựa trên các plug-in trên trình duyệt web, như là một cách để truy cập vào thế giới ảo 3D từ bên trong một trình duyệt web.

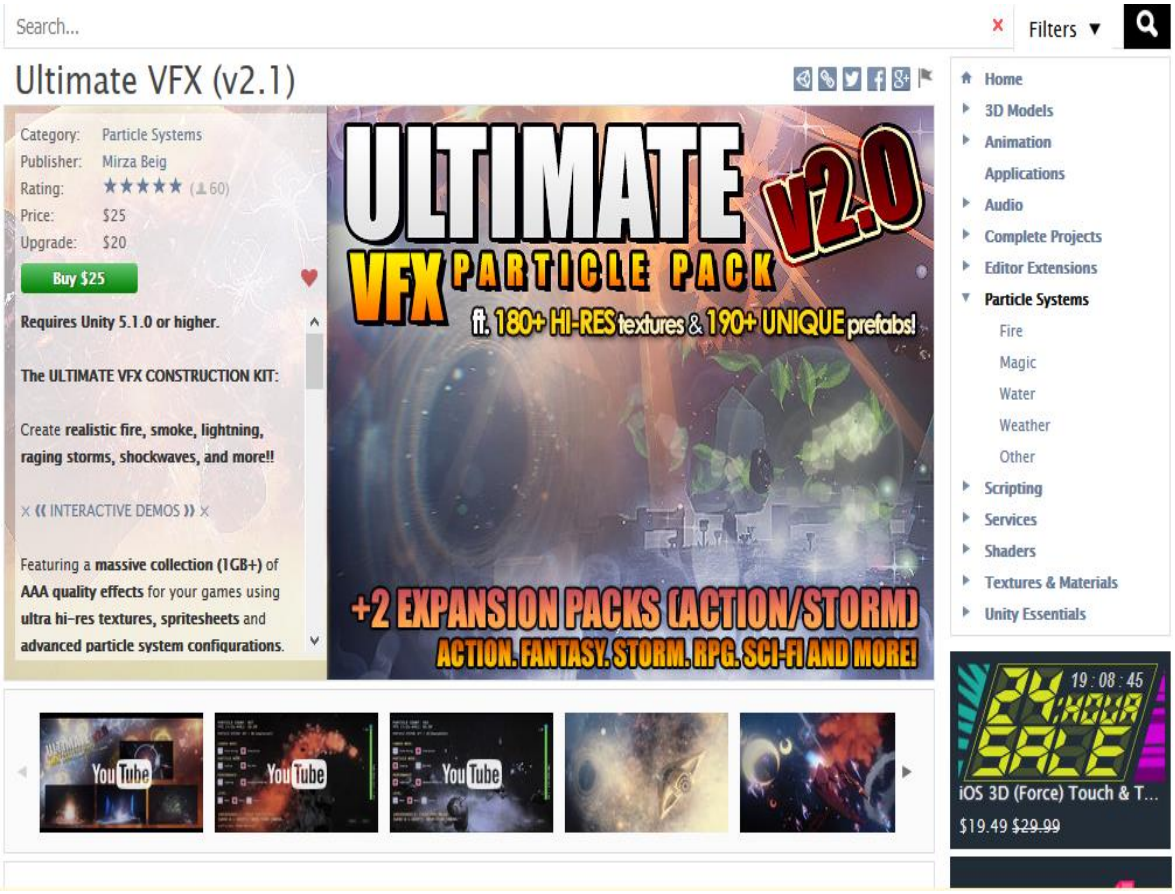
Hình 2-1. Giao diện hiện đại của Unity3D ngày nay

2.2 Tổng quan về các thành phần trong Unity

2.2.1 Assets

Assets là tài nguyên xây dựng nên một dự án trên Unity. Những tài nguyên có thể là hình ảnh, âm thanh, mô hình 2D 3D, chất liệu (material), texture vv hoặc cả một project hoàn chỉnh

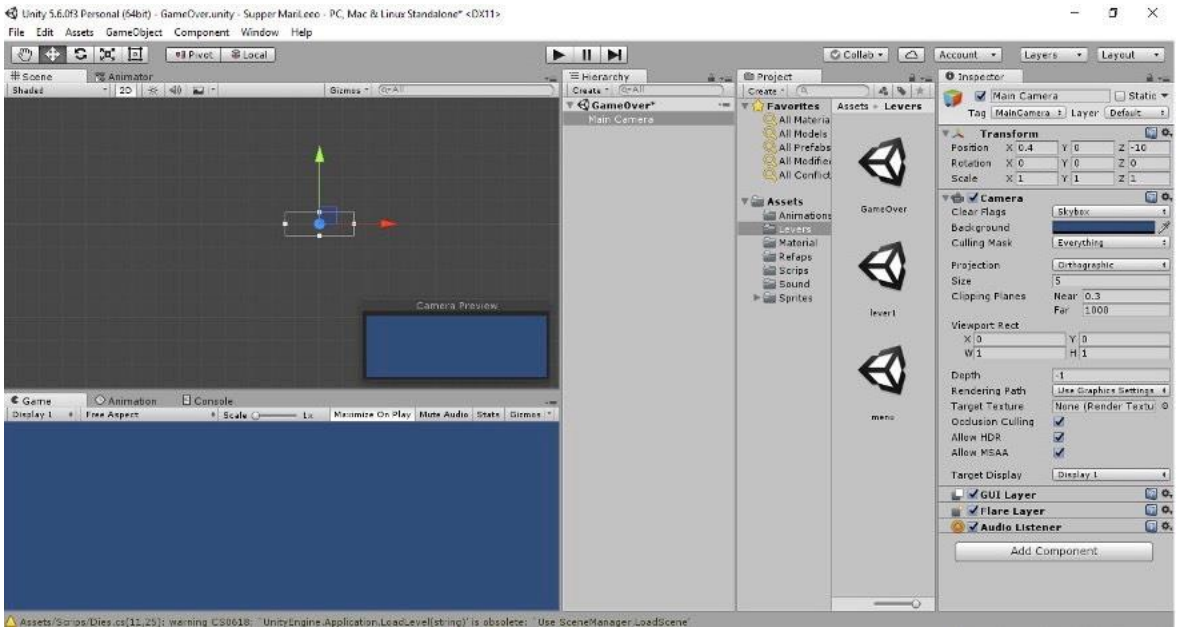
Các asset do chính những nhà phát triển game tạo ra và có thể được download miễn phí hoặc trả phí trên Unity Asset Store. Đây là một trong những tính năng rất hay của Unity. Các asset này sẽ giúp giảm thiểu rất nhiều thời gian cho việc thiết kế và lập trình game.



Hình 2.2: Unity Assets Store

2.2.2 Scenes

Trong Unity, một cảnh chơi (hoặc một phân đoạn) là những màn chơi riêng biệt, một khu vực trong game hoặc thành phần có trong nội dung của trò chơi (các menu). Các thành phần này được gọi là Scene. Bằng cách tạo ra nhiều Scenes, chúng ta có thể phân phối thời gian và tối ưu tài nguyên, kiểm tra các phân đoạn trong game một cách độc lập.



Hình 2-3. Một Scenes trong game

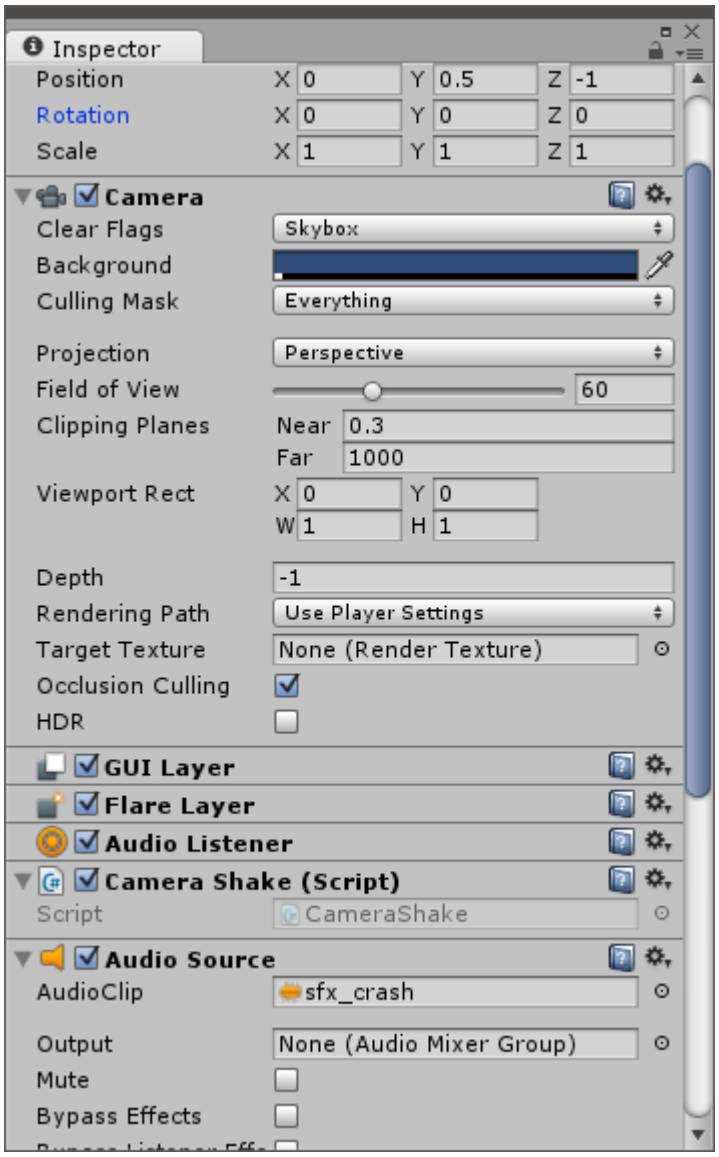
2.2.3 Game Object

Khi Asset được sử dụng trong các Scene, Unity định nghĩa đó là Game Object. Đây là một thuật ngữ thông dụng, đặc biệt trong mảng lập trình. Tất cả các Game Object đều chứa ít nhất một thành phần cơ bản là Transform, lưu trữ thông tin về vị trí, góc xoay và tỉ lệ của Game Object. Thành phần Transform có thể được tuỳ biến và chỉnh sửa trong quá trình lập trình

2.2.4 Components

Components là các thành phần trong game, bổ sung tính năng cho các Game Object. Mỗi Component có chức năng riêng biệt. Đa phần các Component phụ thuộc vào Transform, vì nó lưu trữ các thông số cơ bản của Game Object. Bản chất của

Game Object là không có gì cả, các đặc tính và khả năng của Game Object nằm hoàn toàn trong các Component. Do đó chúng ta có thể xây dựng nên bất kỳ Game Object nào trong game mà chúng ta có thể tưởng tượng được.



Hình 2-4: Cửa sổ Components

2.2.5 Scripts

Scripts được Unity xem như một Component. Đây là thành phần thiết yếu trong quá trình phát triển game. Bất kỳ một game nào, dù đơn giản nhất đều cần đến Scripts để tương tác với các thao tác của người chơi, hoặc quản lý các sự kiện để thay đổi chiều hướng của game tương ứng với kịch bản game.

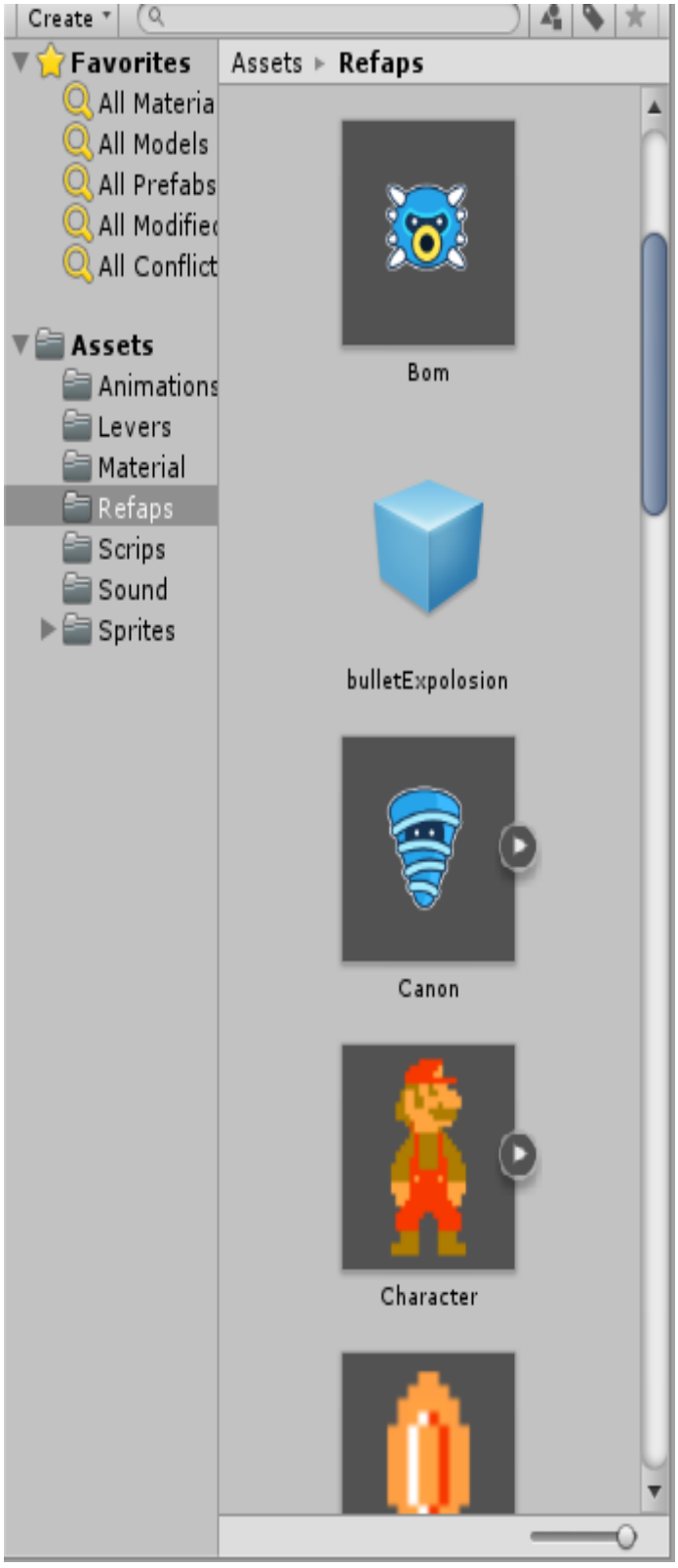
Unity cung cấp cho lập trình viên khả năng viết Script bằng các ngôn ngữ: JavaScript, C#. Unity không đòi hỏi lập trình viên phải học cách lập trình trong Unity, nhưng trong nhiều tình huống, chúng ta cần sử dụng Script trong mỗi phần của kịch bản game.

Để viết Script, chúng ta có thể làm việc với một trình biên tập Script độc lập của Unity, hoặc làm việc trên Mono Developer được tích hợp vào Unity trong những phiên bản gần đây. Mono Developer là một IDE khá tốt, cung cấp nhiều chức năng tương tự Visual Studio chúng ta cũng có thể dùng Visual Studio để viết file C# như bình thường . Mã nguồn viết trên Mono Developer sẽ đươc cập nhật và lưu trữ trong dự án trên Unity

2.2.6 Prefabs

Prefabs thực chất là Game Object được lưu trữ lại để tái sử dụng. Các Game Object được nhân bản từ một prefab sẽ giống nhau hoàn toàn, ngoại trừ thành phần Transform để phân biệt và quản lý được tốt hơn.

Để tạo ra một prefab, ta đơn giản chỉ cần kéo một Game Object vào cửa sổ Project

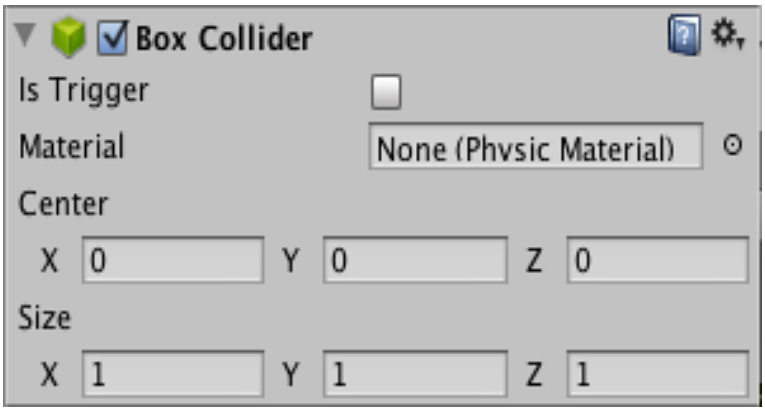


Hình 2-6: Cửa sổ Prefabs

2.2.7 Collider

Hệ thống xử lý va chạm bào gồm 2d và 3d. Được chia ra các va chạm của các hình cơ bản:

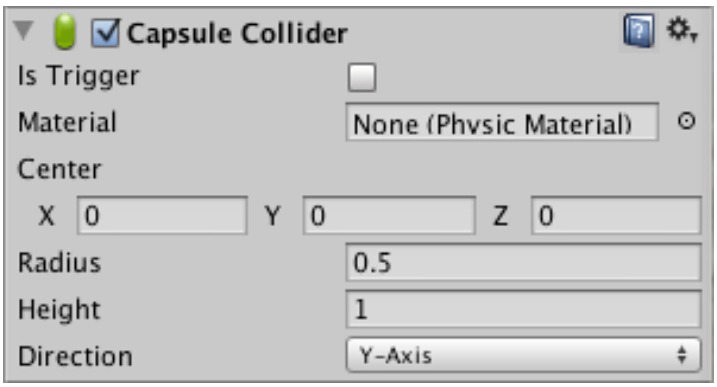
- Box collider: Va chạm cho các vật hình hợp chữ nhật đối với 3d và hình chữ nhật đối với 2d.



Hình 2-7: Cửa sổ Box collider

  - Is Trigger: dạng true/false, cho phép va chạm có đi xuyên qua không?
  - Material: tham chiếu đến physic material.
  - Center: điều chỉnh thông số tâm của va chậm theo các trục tương ứng.
  - Size: điều chỉnh kích thước theo các trục tương ứng.

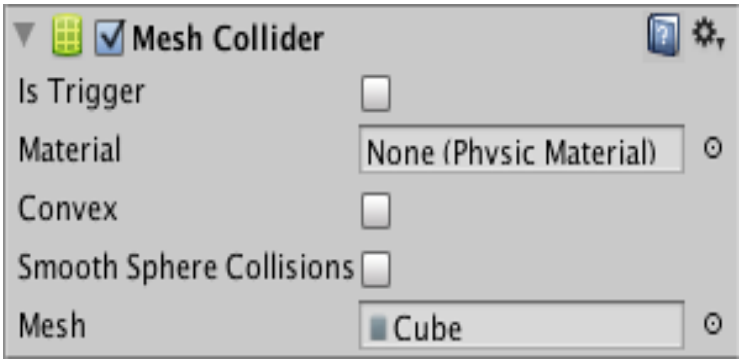
- Capsule Colider: Va chạm dành cho các khói hình trụ.



Hình 2-8: Cửa sổ Capule collider

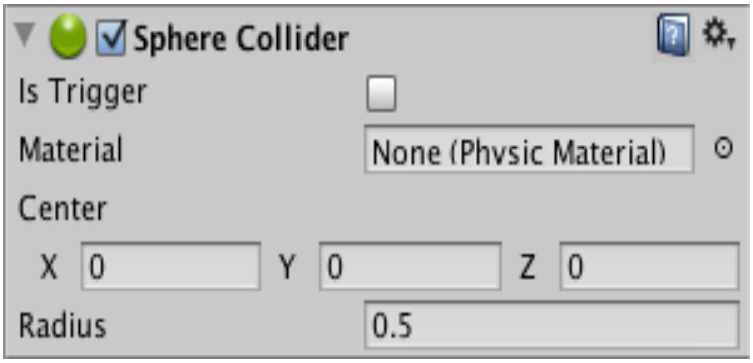
- Is Trigger: dạng true/false, cho phép va chạm có đi xuyên qua không?
- Material: tham chiếu đến physic material.
- Center: điều chỉnh thông số tâm của va chạm theo các trục tương ứng.
- radius: điều chỉnh kích thước bán kính.
- Height: độ cao
- Direction: xoay hình trụ theo các trục tương ứng.Box collider: Va chạm cho các vật hình hợp chữ nhật đối với 3d và hình chữ nhật đối với 2d.

- Mesh Colilider: Va chạm dạng lưới dành các vật thể không xác định hình thể



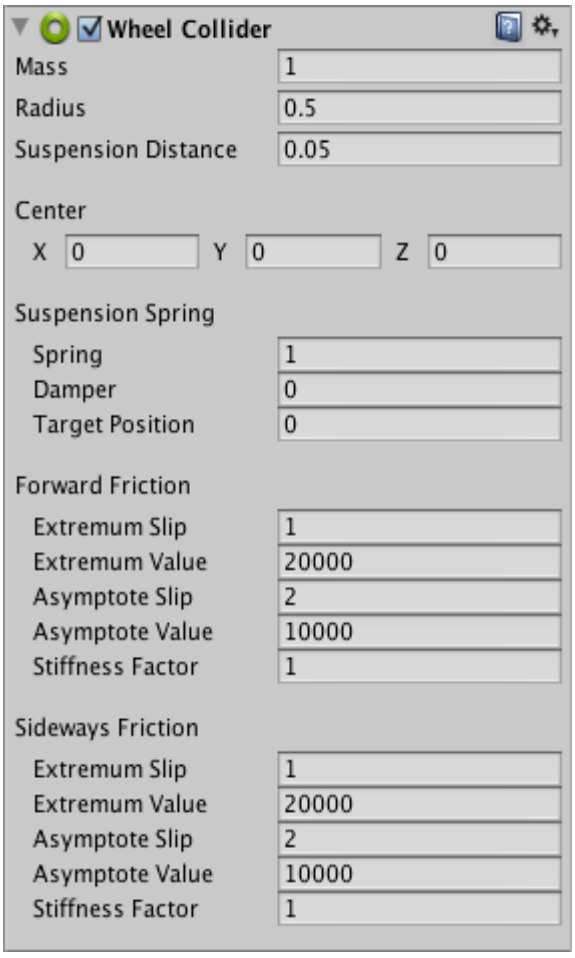
Hình 2-9: Cửa sổ Mesh collider

- Sphere Colider: va chạm áp dụng cho các vật hình cầu..



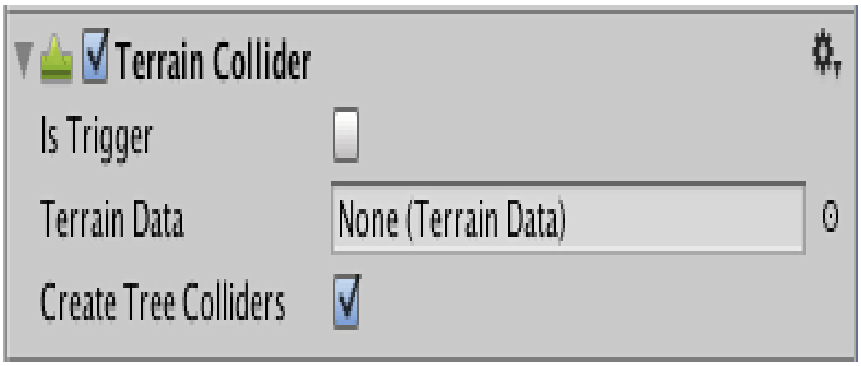
Hình 2-10: Cửa sổ Sphere collider

- Wheel collider: dành cho vật thể hình bánh xe. Nó sẽ mô phỏng hệ thống va chạm giống với các bánh xe.



Hình 2-11: Cửa sổ Wheel collider

- Terrain Collider: dành cho các vật thể địa hình và terrain collider sẽ dựa theo địa hình đó



Hình 2-12: Cửa sổ Terrain collider

2.2.8 Rigidbody

Hệ thống mô phỏng vật lí trong game. Cũng chia ra 2d

- Mass: khối lượng (đơn vị tùy ý).
- Drag: Sức cản không khí ảnh hưởng như thế nào với đối tượng khi di chuyển. 0 có nghĩa là không có sức cản không khí, và vô cùng làm cho các đối tượng di chuyển ngay lập tức dừng lại.
- Angular Drag: Sức cản không khí ảnh hưởng đến các đối tượng khi quay từ mô-men xoắn. 0 có nghĩa là không có sức cản không khí. Không thể làm cho vật dừng quay hẳn chỉ bằng cách thiết lập Angular Drag của nó đến vô cùng.
- Use Gravity: Nếu được kích hoạt, các đối tượng bị ảnh hưởng bởi lực hấp dẫn.
- Is Kinematic: Nếu được kích hoạt, các đối tượng sẽ không được thúc đẩy bởi động cơ vật lý.
- Interpolate: giảm xóc.
- Collision detection: Được sử dụng để ngăn chặn các đối tượng chuyển động nhanh qua các đối tượng khác mà không phát hiện va chạm.

- Constraints: Ràng buộc Những hạn chế về chuyển động của Rigidbody:

2.2.9 Sprite

Là một hình ảnh 2D của một game object có thể là hình ảnh đầy đủ, hoặc có thể là một bộ phận nào đó. Unity cho phép tùy chỉnh màu sắc, kích thước, độ phân giải của một hình ảnh 2d.

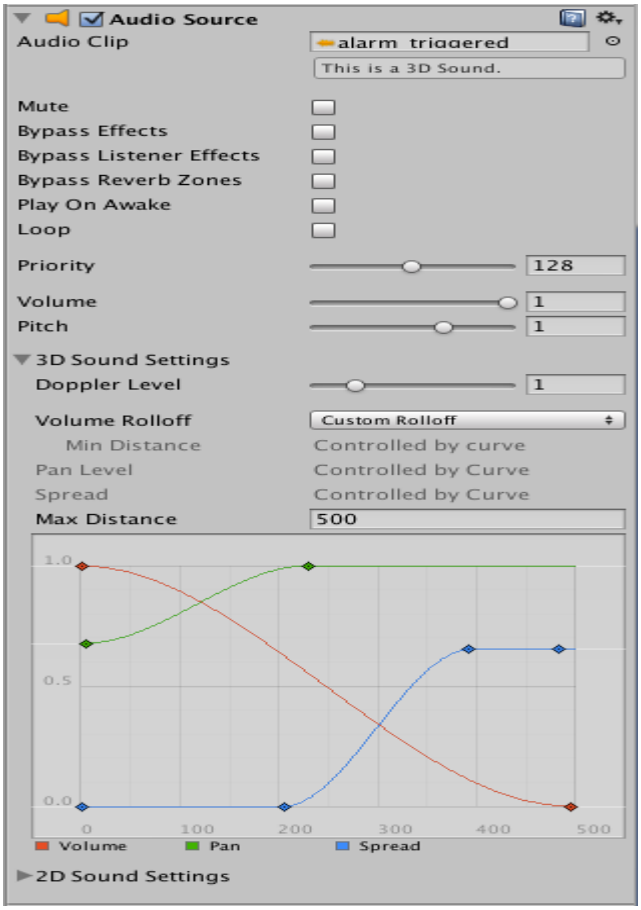
2.2.10 Animator

Trong 2D thi animotion là tập một hình ảnh động dựa trên sự thay đổi liên tục của nhiều sprite khác nhau. Trong 3d là một tập hợp các sự thay đổi theo thời gian của đối tượng trong không gian. Mỗi thay đổi là một key frame.-Key Frame hay Frame là một trạng thái của một animation.

- Aminator: gồm các thành phần:
- Controller: Bộ điều khiển animation gắn liền với nhân vật này. Nó sẽ quản lý các animation clip, các thông số tốc độ của animation, thứ tự các clip…
- Avatar: thành phần tạo hình ảnh cho object.
- Apply Root Motion: dạng true-false, cho phép thiết lập animation có di chuyển theo không gian đã được tạo khi cấu hình animation.
- Animate Physics: dạng true-false, khi được chọn, các hình ảnh trong animation sẽ có thể tương tác vật lý với nhau.
- Culling mode: chọn chế độ cho hình ảnh động.

2.2.11 Audio Source:

Âm thanh trong game. Gồm cả âm thanh 2d và 3d.



Hình 2-12: Cửa sổ Audio Source

- Audio clip: tham chiếu điến file âm thanh.
- Mute: chơi ở chế độ như tắt tiếng.
- Bypass effects: bộ lọc hiệu ứng áp dụng cho các nguồn âm thanh.
- Bypass listener effects: Điều này là để nhanh chóng chuyển tất cả các hiệu ứng Listener on / off.
- Pass Reverb Zones: Điều này là để nhanh chóng chuyển tất cả các khu Reverb on / off.
- Play on awake: Nếu được kích hoạt, âm thanh sẽ bắt đầu chơi lúc cảnh ra mắt. Nếu vô hiệu hóa, cần phải bắt đầu nó bằng cách sử dụng lệnh Play () từ kịch bản script.
- Loop: Kích hoạt tính năng này để làm cho Clip âm thanh lặp lại.

- Pitch: Xác định ưu tiên của nguồn âm thanh này trong số tất cả những nguồn âm cùng tồn tại trong bối cảnh đó. (Ưu tiên: 0 = quan trọng nhất, 256 = ít quan trọng nhất Mặc định = 128).
- 3D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh nếu Audio Clip là một âm thanh 3D.
- 2D Sound Setting: Cài đặt được áp dụng cho các nguồn âm thanh nếu Audio Clip là một âm thanh 2D.

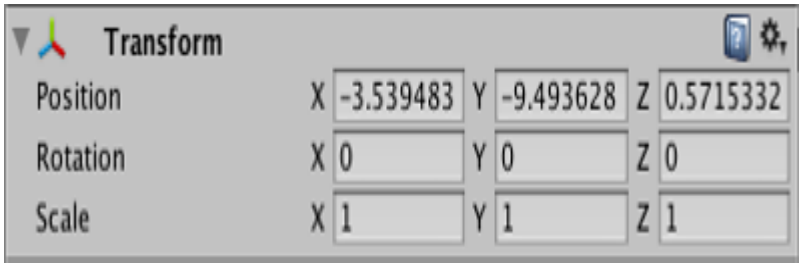
2.2.12 Camera

Là một game object đặc biệt trong scene, dùng để xác định tầm nhìn, quansát các đối tượng khác trong game. Bao gồm các thuộc tính:

- Clear flags: Xác định phần nào của màn hình sẽ bị xóa. Đây là tiện dụng khi sử dụng nhiều máy ảnh để vẽ các yếu tố trò chơi khác nhau.
- Bachground: Màu áp dụng cho các màn hình còn lại sau khi tất cả các yếu tố trong quan điểm đã được rút ra và không có skybox.
- Culling Mask: Bao gồm hoặc bỏ qua lớp của các đối tượng được đưa ra bởi các Camera.
- Projection: khả năng của máy ảnh để mô phỏng góc nhìn.
- Field of view (thuộc tính chỉ xuất hiện khi chọn Perspective trong mục Projection): Chiều rộng của góc nhìn của Camera, đo bằng độ dọc theo trục Y.
- HDR: Cho phép High Dynamic Range dựng hình cho camera này.

2.2.13 Transform

Trasform: quản lý object trong không gian ba chiều, theo ba thông số:

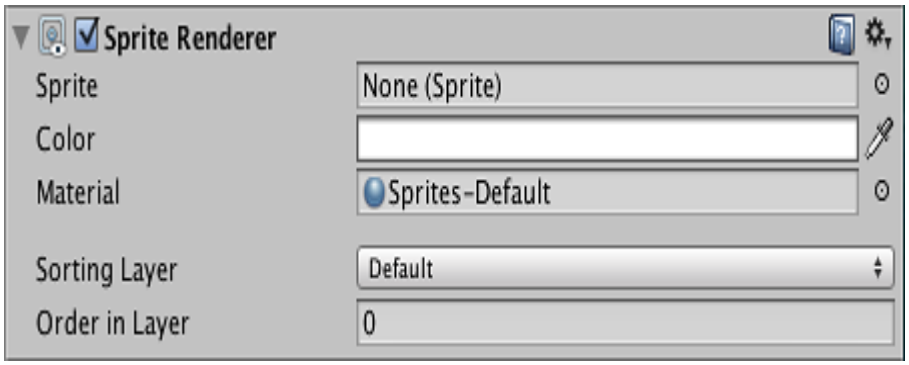


Hình 2-12: Cửa sổ Transform

- Postion: quản lí vị trí hiện tại của object.
- Rotation: quản lí cá thông số quay của object theo các trục x, y, z.
- Scale: quản lí các thông số phóng to, thu nhỏ theo các trục x, y, z.

2.2.14 Renderer

Các SpriteRenderer thànhphần cho phép bạn hiển thị hình ảnh như Sprites để sử dụng trong cả hai cảnh 2D và 3D.



Hình 2-12: Cửa sổ Renderer

- Sprite: Các đối tượng Sprite để render. Đối tượng Sprite có thể được tạo ra từ textures bằng cách sử dụng các thiết lập Sprite.
- Color: Vertex màu.
- Material: Chất liệu được sử dụng để làm sprite.
- Sorting Layer: Các layer được sử dụng để xác định các ưu tiên của sprite này trong khi hiển thị.

- Order in Layer: Các ưu tiên của sprite trong layer của nó. Con số thấp hơn được kết xuất đầu tiên và con số tiếp theo phủ bên dưới.

2.3 Nguyên tắc thiết kế Game Mario

2.3.1 Nguyên tắc 1: Thiết kế giao diện game chặt chẽ và dễ sử dụng

Cách thiết kế giao diện cho ứng dụng là một trong những vấn đề rất quan trọng đối với người chơi. Ngoài ra, bạn còn cần phải quan tâm đến bố cục của game sao cho hợp lý để người dùng dễ dàng tiếp cận các thông tin cần thiết khi sử dụng. Bởi ứng dụng thiết kế không tốt thì người dùng sẽ khó tiếp cận được những nội dung theo đúng ý muốn của người chơi. Từ đó, người chơi sẽ nhanh chóng thoát ra khỏi ứng dụng và cũng sẽ không muốn quay lại khi có nhu cầu.

Về cách phân chia và tổ chức, trước khi Game được thiết kế, nhóm đề tài đã định hình một số khung giao diện thường gặp và thiết kế chúng trở thành giao diện.

2.3.2 Nguyên tắc 2: Game phải được sử dụng một cách mượt mà

Các cử chỉ, hành động của nhân vật chính cũng như kẻ thù phải được thực thi một cách trơn tru, do đó tác giả đã tận dụng để xây dựng cách điều khiển bằng nút bấm trên bàn phím nhằm giúp cho người dùng có thể thao tác dễ dàng nhất.

2.3.3 Nguyên tắc 3: Cách cài đặt game phải dễ dàng

Nhờ công cụ Unity, việc cài đặt và chơi trên Windows hết sức dễ dàng, chỉ cần chọn nền mà mình mong muốn game được chạy trên đó và build.

Hệ thống sẽ tự động build cho chúng ta 1 file .exe để chúng ta có thể click double vào và game sẽ hoạt động.

CHƯƠNG 3: NỘI DUNG THỰC HIỆN

3.1 Phân tích đề tài

3.1.1 Khái niệm game Side scroller?

Side scroller là một trò chơi điện tử trong đó gameplay sẽ bắt đầu từ góc nhìn của nhân vật chính bắt đầu từ bên trái rồi chuyển động sang bên phải, và các nhân vật phụ trên màn hình thường di chuyển từ phía bên trái của màn hình sang phải để đáp ứng một mục tiêu nào đó của trò chơi.

Hình 3-1: Ví dụ về các game sử dụng side scroller

Cách sử dụng phổ biến của side scroller ở trong các trò chơi đa nền tảng là các trò chơi có tính năng như: leo trèo, chạy, nhảy các chướng ngại vật.

3.1.2 Cách chơi

Mục tiêu của trò chơi là hòa mình vào anh chàng thợ sửa ông nước đi cứu công chúa, yêu cầu di chuyển nhân vật chính vượt qua các chướng ngại vật cản trở trên đường bằng các nút di chuyển nhảy và bắn để vượt qua các con quái vật trên đường đi để cứu công chúa.

Khi nhân vật chạm vào quái vật thì sẽ mất máu nếu hết máu nhân vật sẽ chết, và trên đường đi sẽ có rất nhiều vàng rơi trên đường nếu nhân vật nhặt được vàng sẽ được cộng điểm, hoặc giết các con quái vật cũng sẽ nhận được điểm nếu nhân vật chết các con Boss sẽ rơi ra các vật phẩm, vật phẩm có thể là máu để hồi máu cho nhân vật,vàng để tăng điểm cho nhân vật,càng những màn về sau trò chơi sẽ càng tăng mức khó cho các màn về sau,nhiều quái vật hơn,thử thách dần dần cao lên .

3.2 Xác định yêu cầu

3.2.1 Giao tiếp hệ thống

Hệ thống cần phải có đầy đủ chức năng của một ứng dụng chơi game.

Các nút điều khiển phải được liên kết chặt chẽ.

3.2.2 Giao tiếp về điều khiển

Các điều khiển phải đầy đủ, dễ dàng sử dụng để người dùng không bị ngượng khi sử dụng ứng dụng.

Các điều khiển cần phải có hệ thống quản lý rõ ràng, liên kết chặt chẽ với nhau và đặc biệt phải phục vụ được đầy đủ các chức năng khi người dùng chơi game.

3.2.3 Giao tiếp về giao diện

Giao diện phải sử dụng các hình ảnh sắc nét, mượt mà, mới lạ, dễ nhìn.

Giao diện phải sắp xếp các nút điều khiển, các hình ảnh, các chữ một cách ngăn nắp, gọn gàng và đặc biệt là phải phân bố hợp lý không bị loạn.

Đặc biệt giao diện cần phải phù hợp với màn hình các thiết bị điện thoại khác nhau.

3.3 Kịch bản game

Game Mario là một game side scrolling với bối cảnh là một chú thợ sửa ông nước có tên là Mario, với nhiệm vụ là giải cứu công chúa khỏi các con quái vật chúng bắt đi.

Nhưng mọi chuyện không đơn giản như thế, ở mỗi một con đường đều có những quái vật của nơi đó bảo vệ và ngăn cản Mario, nhiệm vụ của Mario là sống sót đến cuối cùng để tìm được công chúa và cứu công chúa.

Khi chạy game, hệ thống sẽ chuyển đến giao diện menu, ở đây người dùng chỉ cần ấn vào nút Play là hệ thống sẽ chuyển tới một giao diện khác, ở đó là giao diện chọn màn chơi.

Khi đã chọn được màn chơi thích hợp, Mario sẽ bắt đầu ở màn đấy và điều đầu tiên người chơi có thể ấn những nút như S D để di trái, phải và Space để nhảy lên cao.

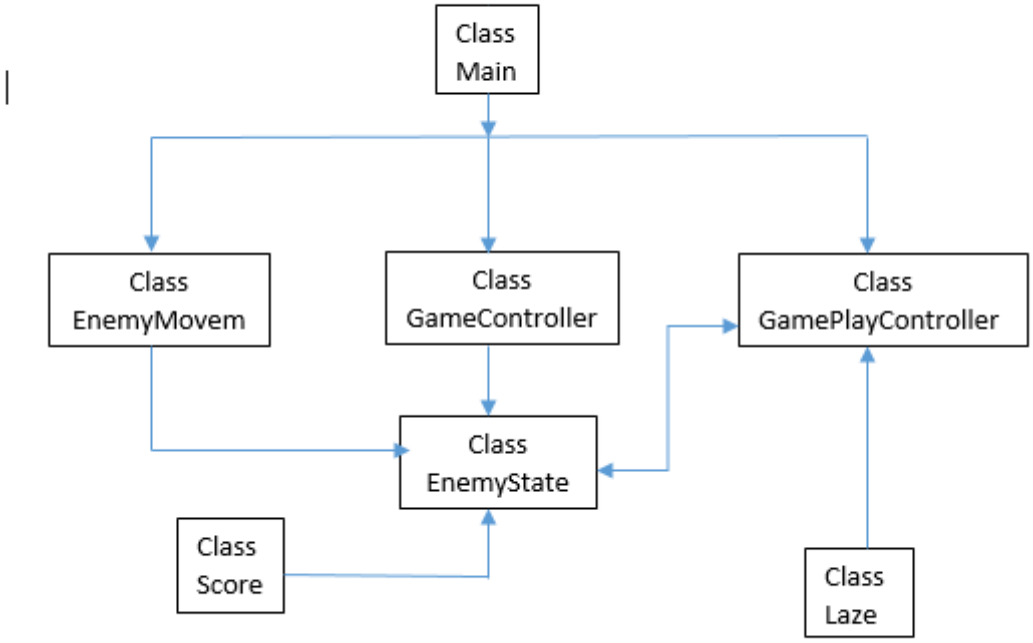
Ngoài ra Mario còn có thể bắn đạn qua chiếc súng chú đang cầm bằng chuột trái.

Về kẻ địch nhưng con quái vật sẽ di chuyển hoặc bắn ra những viên đạn khiến cho Mario va vào chúng sẽ mất máu dần dần nếu máu hết thì phải chơi lại từ đầu.

Trong màn chơi có những vật phẩm mà Mario có thể nhặt được, đó là đồng xu giúp tăng điểm, trái tim để hồi máu.

Khi sống sót đến hết màn chơi, Mario sẽ chạm vào một cánh cửa, ở đó cánh cửa sẽ đưa Mario đến một hành tinh tiếp theo.

3.4 Sơ đồ quan hệ giữa các lớp



Hình 3-2 : Sơ đồ quan hệ giữa các lớp

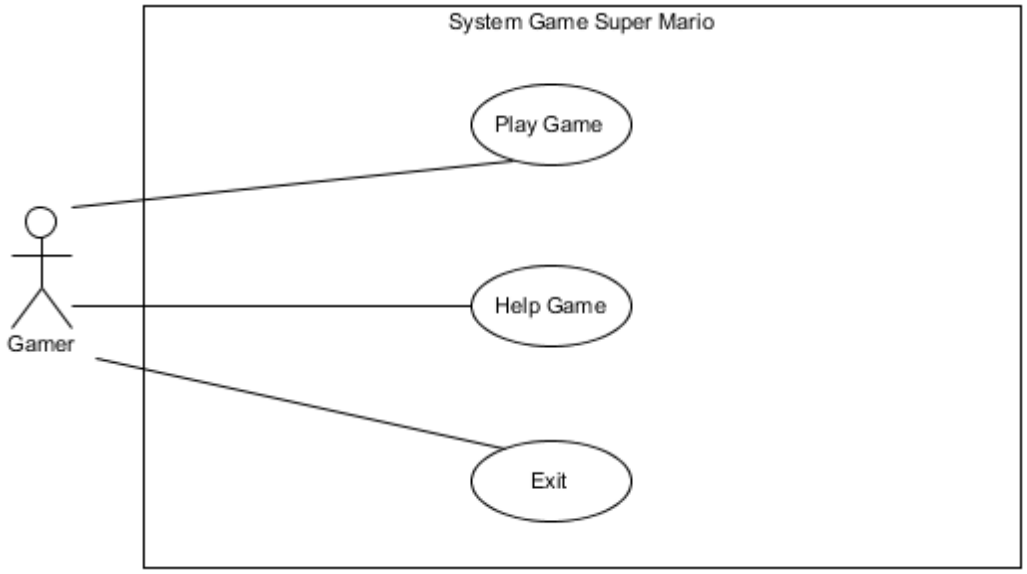
- **Class Main**: Đây là class để xử lý tiến trình bắt đầu game cũng như kết thúc game
- **Class EnemyMove:** Đây là class xử lý di chuyển của Enemy
- **Class GameController:** Đây là class lưu trữ các enemy và xử lý random Enemy.
- **Class GamePlayController:** Đây là class dùng để xử lý di chuyển nhân vật, kịch hoạt Laze và các xử lý va chạm với enemy , HP, âm thanh …
- **Class EnemyState** : Đây là class dùng để xử lý va chạm của enemy, xử lý số điểm, gây sát thương, âm thanh , chuyển động…
- **Class Laze:** Là class để xử lý bay của viên đạn cũng như xử lý va chạm với enemy

3.5 Thiết kế đặc tả chức năng

Để hiểu rõ hơn về yêu cầu của đề tài, nhóm nghiên cứu đề xuất phương án thiết kế các biểu đồ để hiểu thêm về cách thức hoạt động, cách xử lý của đề tài.

3.5.1 Biểu đồ Use-case:

a) Biểu đồ Use-case tổng quát:



Hình 3-3 :Biểu đồ Use Case tổng quát

b) Đặc tả Use-case "Play"

- *Tên use-case:* Use-case Play game.
- *Người sử dụng:* Người dùng.
- *Mục đích:* Chức năng này là chức năng chơi game, là chức năng sử dụng chính của ứng dụng.
- *Dòng sự kiện:*

Hành động của tác nhân	Phản ứng của hệ thống
1. User vào ứng dụng.	Hệ thống đưa ra giao diện chính.
2. User nhấn vào button "Play".	Hệ thống hiển thị ra màn hình chơi game.

- *Các yêu cầu đặc biệt:*

  Không có

- *Trạng thái hệ thống trước khi bắt đầu thực hiện Use-case:*

  Hệ thống đang ở màn hình menu game.

- *Trạng thái hệ thống sau khi thực hiện Use-case:*

  Hệ thống hiển thị ra màn hình chơi game chính.

- *Điểm mở rộng:*

  Không có

c) Đặc tả Use-case "Help"

- *Tên use-case:* Use-case Help.
- *Người sử dụng:* Người dùng.
- *Mục đích:* Chức năng giới thiệu về thông tin ứng dụng.
- *Dòng sự kiện:*

Hành động của tác nhân	Phản ứng của hệ thống
1. User vào ứng dụng.	Hệ thống đưa ra giao diện chính.
2. User nhấn vào button "Hướng Dẫn".	Hệ thống hiển thị màn hình giới thiệu các chức năng trong game.

- *Các yêu cầu đặc biệt:*

  Không có.

- *Trạng thái hệ thống trước khi bắt đầu thực hiện Use-case:*

  Hệ thống đang ở màn hình chính.

- *Trạng thái hệ thống sau khi thực hiện Use-case:*

  Hệ thống hiển thị màn hình giới thiệu các chức năng trong game.

- *Điểm mở rộng:*

Không có.

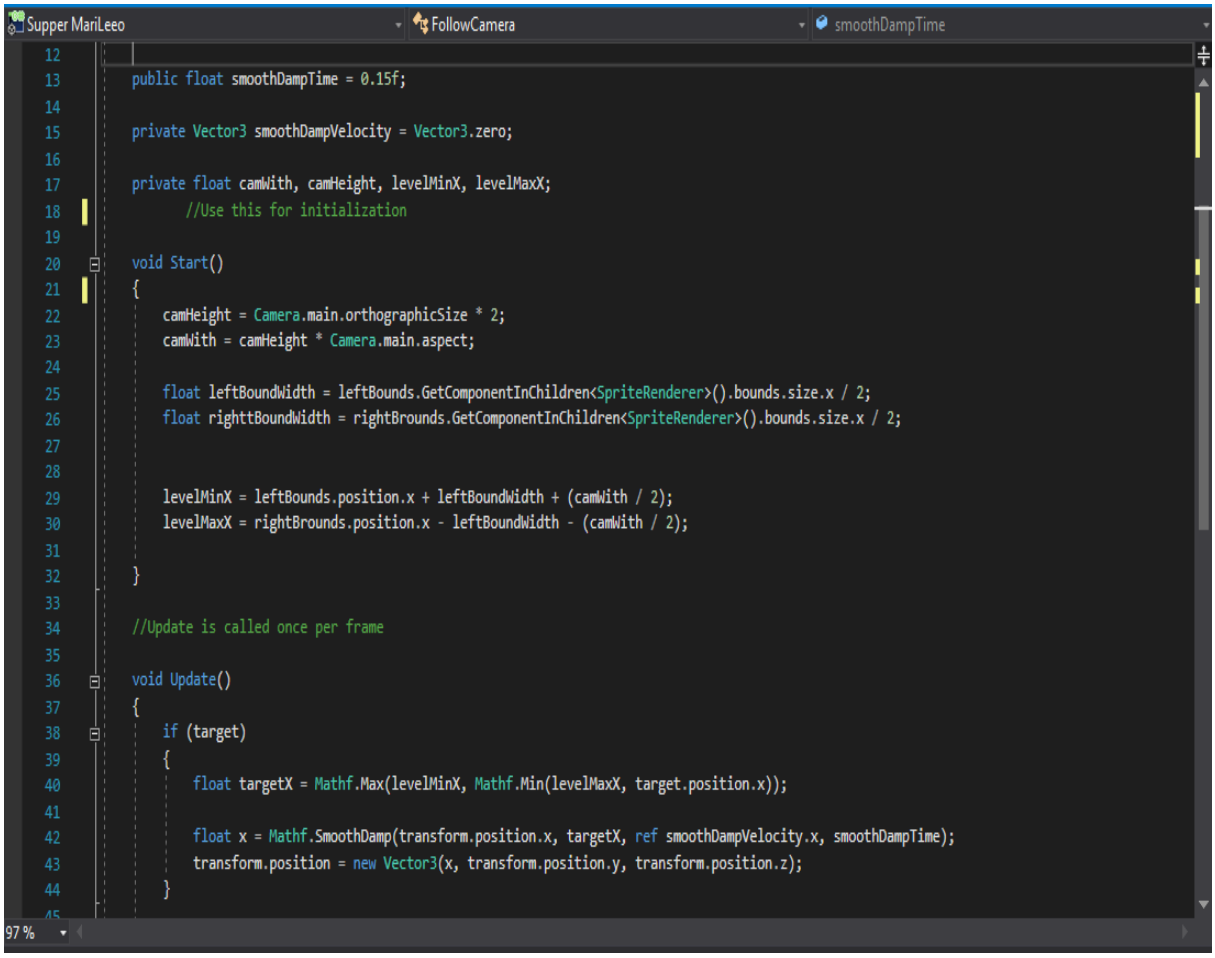
3.6 Xây dựng game

3.6.1 Xây dựng nhân vật di chuyển

```
void Start()
{
    myAnim = GetComponent<Animator>();
    myBody = GetComponent<Rigidbody2D>();

    facingRight = true;
}

// Update is called once per frame
void FixedUpdate()
{
    float move = Input.GetAxis("Horizontal");
    float jump = Input.GetAxis("Jump");

    myAnim.SetFloat("speed", Mathf.Abs(move));
    myAnim.SetFloat("jump", Mathf.Abs(jump));

    myBody.velocity = new Vector2(move * maxSpeed, myBody.velocity.y);
    if (move > 0 && !facingRight)
    {
        flip();
    }
    else if (move < 0 && facingRight)
    {
        flip();
    }
    if (Input.GetKeyDown(KeyCode.Space))
    {
        if (grounded)
        {
            grounded = false;
            myBody.velocity = new Vector2(myBody.velocity.x, jumHeight);
        }
        //GetComponent<Rigidbody2D>().velocity = new Vector3(0, jumpHeight, 0);
    }
    //chức năng bắn từ bàn phím
    if (Input.GetAxisRaw("Fire1") > 0)
        fireBullet();
}
void flip()
{
```

Hình 3-4 : Xây dựng nhân vật di chuyển

3.6.2 Xây dựng camera theo dõi nhân vật



Hình 3-5 : Xây dựng camera theo dõi nhân vật

3.6.3 Xây dựng máu nhân vật

```
    }
    public void addDamage(float damage)
    {
        if (damage <= 0)
            return ;
        curentHealth -= damage;

        playerHealthSlider.value = curentHealth;

        if (curentHealth <= 0)
            makeDate();
    }
    //  tạo ra chức năng hồi máu khi ăn viên máu
    public void addhealth (float healthAmount)
    {
        curentHealth += healthAmount;
        if (curentHealth > maxHealth)
            curentHealth = maxHealth;
        playerHealthSlider.value = curentHealth;
    }
    void Awake()
    {
        MakeInstance();
    }

    void MakeInstance()
    {
        if (instance == null)
        {
            instance = this;
        }
    }
    public void makeDate()
    {
        Instantiate(bloodEffect, transform.position, transform.rotation);
        gameObject.SetActive(false);
        GamePause.instance.CharterDiedShowPanal();
    }
}
```

Hình 3-6 : Xây dựng máu nhân vật

3.6.4 Xây dựng tính điểm cho nhân vật

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

public class ScoreManager : MonoBehaviour {
    public static int score;
    Text text;


    // Use this for initialization
    void Start () {
        text = GetComponent<Text>();
        score = 0;

    }

    // Update is called once per frame
    void Update()
    {
        if (score < 0)

            score = 0;
            text.text = "" + score;

    }
    public static void Addpoint(int p)
    {
        score += p;
    }

    public static void Reset()
    {
        score = 0;
    }
}
```

Hình 3-7 : Xây dựng tính điểm cho nhân vật

3.6.5 Xây dựng máu cho quái vật

```
    public bool drop;
    public GameObject theDrop; // muốn rơi ra cái gì thì kéo vào đây
    void Start () {
        curenhealth = maxHealth;

        enemyHealthSlider.maxValue = maxHealth;
        enemyHealthSlider.value = maxHealth;


    }

    // Update is called once per frame
    void Update () {

    }
    public void addDamge(float damage)
    {
        enemyHealthSlider.gameObject.SetActive(true);// khi nhhận viên đạn mới hiện máu

        curenhealth -= damage;

        enemyHealthSlider.value = curenhealth;

        if (curenhealth <= 0)
            makeDead();
    }
    void makeDead()
    {
        gameObject.SetActive(false);
        Instantiate(enemyHealththEF, transform.position, transform.rotation);
        //chức năng rơi ra viên máu
        if (drop)
        {
            Instantiate(theDrop, transform.position, transform.rotation);
        }
    }
}
```

Hình 3-8 :Xây dựng máu cho quái vật

3.6.6 Xây dựng khi ăn trái tim sẽ tăng máu cho nhân vật

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

public class heartPickup : MonoBehaviour {
    public float healthAmount;

    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag== "Player")
        {
            playerHealth thePlayerHealth = other.gameObject.GetComponent<playerHealth>();
            thePlayerHealth.addhealth(healthAmount);
            Destroy(gameObject);
        }
    }
}
```

Hình 3-9 : Khi ăn trái tim sẽ tăng máu cho nhân vật

3.6.7 Xây dựng sát thương quái vật gây ra

```
{
    // gây sát thương cho nhân vật
    // Use this for initialization
    public float damege;
    float damerate = 0.5f; // sau time 05s gay sat thuong

    public float pushBackFore; //khi bi gay sat thuong nhan vat nhay len
    float nextDamage; // gay sat thuong nay lap tuc
    void Start()
    {
        nextDamage = 0f;
    }
    // Update is called once per frame
    void Update()
    {
    }
    private void OnTriggerStay2D(Collider2D other)
    {

        if (other.gameObject.tag == "Player" && nextDamage < Time.time) // khi player
        {
            playerHealth thePlayerHealth = other.gameObject.GetComponent<playerHealth>();
            thePlayerHealth.addDamage(damege);
            nextDamage = damerate + Time.time;

            pushBack(other.transform);
        }
    }
    void pushBack(Transform pushObject)
    {
        Vector2 pushDirrection = new Vector2(0, (pushObject.position.y - transform.position.y)).normalized; // vi tri cua nhan vat tru di vi tri cua enemy
        pushDirrection *= pushBackFore;
        Rigidbody2D pushRB = pushObject.gameObject.GetComponent<Rigidbody2D>();
        pushRB.velocity = Vector2.zero;
        pushRB.AddForce(pushDirrection, ForceMode2D.Impulse);// forceMode lap tuc cho n 1 cai luc
    }
}
```

Hình 3-10** : **Xây dựng sát thương quái vật gây ra

3.6.8 Xây dựng hiệu ứng bắn

```
}

// Update is called once per frame
void Update()
{

}
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.tag == "ShootTable")
    {
        myPC.removeFore();
        //tao hieu ung no
        Instantiate(bulletExpolosion, transform.position, transform.rotation);
        Destroy(gameObject);
        if (other.gameObject.layer == LayerMask.NameToLayer("enemy"))
        {
            enemyHelth hurtEnemy = other.gameObject.GetComponent<enemyHelth>();
            hurtEnemy.addDamge(weaponDamage);
        }
    }
}
void OnTriggerStay2D(Collider2D other)
{
    if (other.gameObject.tag == "ShootTable")
    {
        myPC.removeFore();
        //tao hieu ung no
        Instantiate(bulletExpolosion, transform.position, transform.rotation);
        Destroy(gameObject);
        if (other.gameObject.layer == LayerMask.NameToLayer("enemy"))
        {
            enemyHelth hurtEnemy = other.gameObject.GetComponent<enemyHelth>();
            hurtEnemy.addDamge(weaponDamage);
        }
    }
}
```

Hình 3-11 : Xây dựng hiệu ứng bắn

3.6.9 Xây dựng quái vật

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

public class canonShoot : MonoBehaviour {
    public GameObject theboom;
    public Transform shootForm;
    public float shotTime;
    float nexShoot = 0f;
    Animator canonAmi;

     void Awake()
    {
        canonAmi = GetComponentInChildren<Animator>();

    }
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
    private void OnTriggerStay2D(Collider2D other)
    {
        if (other.tag == "Player"&& Time.time > nexShoot)
        {
            nexShoot = Time.time + shotTime;
            Instantiate(theboom, shootForm.position, Quaternion.identity);
            canonAmi.SetTrigger("canonShoot");
        }
    }
}
```

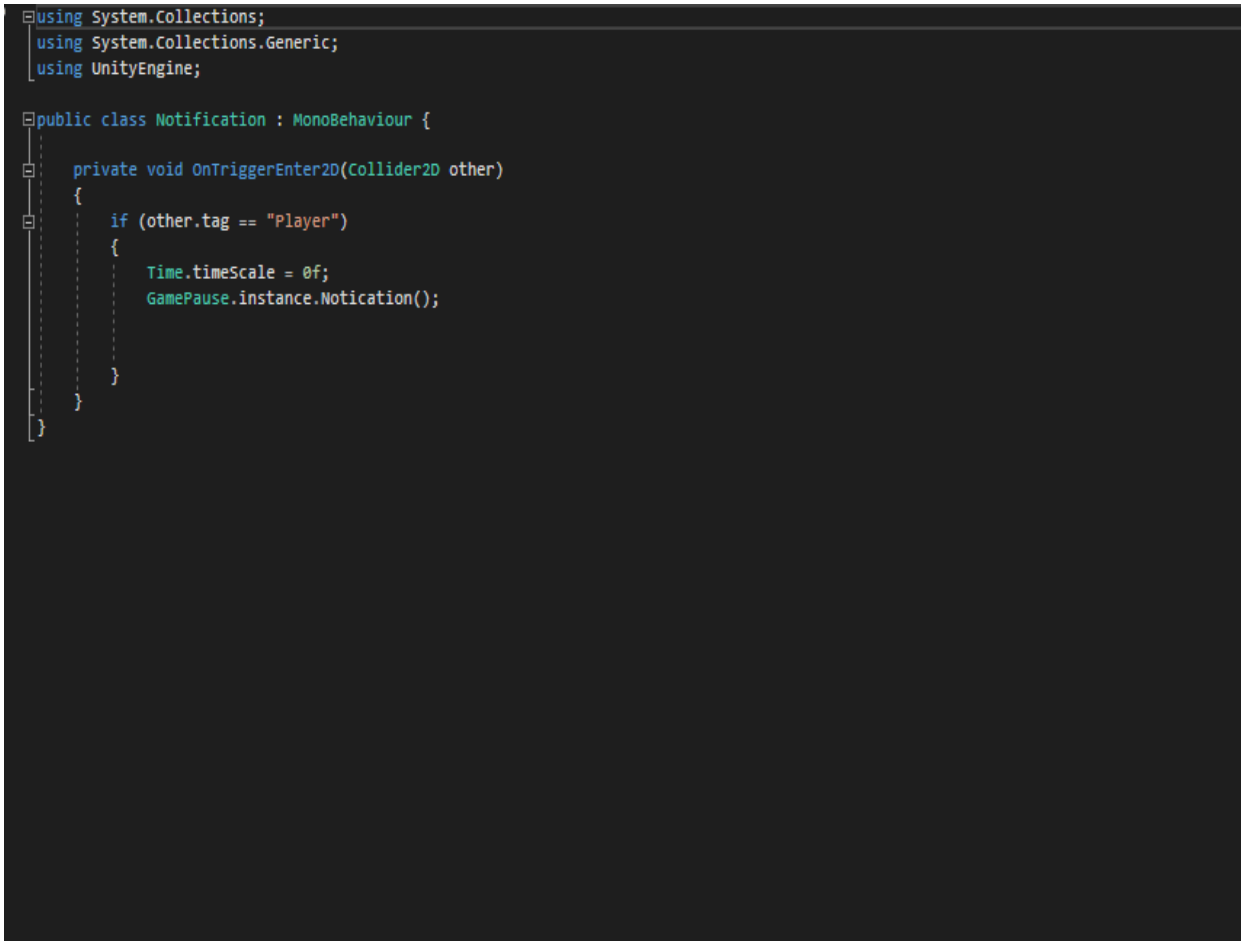
Hình 3-12 : Xây dựng quái vật

3.6.10 Xây dựng vùng nhảy cao lên so với bình thường

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

public class up : MonoBehaviour {
    public float fore = 500f;
    private Animator anim;
    void Awake()
    {
        anim = GetComponent<Animator>();

    }
    // Use this for initialization
    void Start() {

    }
    IEnumerator AnimateBouncy()
    {
        anim.Play("Up");
        yield return new WaitForSeconds(.5f);
        anim.Play("Down");
    }

     void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {
            other.gameObject.GetComponent<PlayerController>().Uplayer(fore);
            StartCoroutine(AnimateBouncy());
        }
    }
    // Update is called once per frame
    void Update() {

        }
    }
```

Hình 3-13 : Xây dựng vùng nhay cao lên so với bình thường

3.6.11 Xây dựng thông báo trong game



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

public class Notification : MonoBehaviour {

    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {
            Time.timeScale = 0f;
            GamePause.instance.Notication();


        }
    }
}
```

Hình 3-14 : Xây dựng thông báo trong game

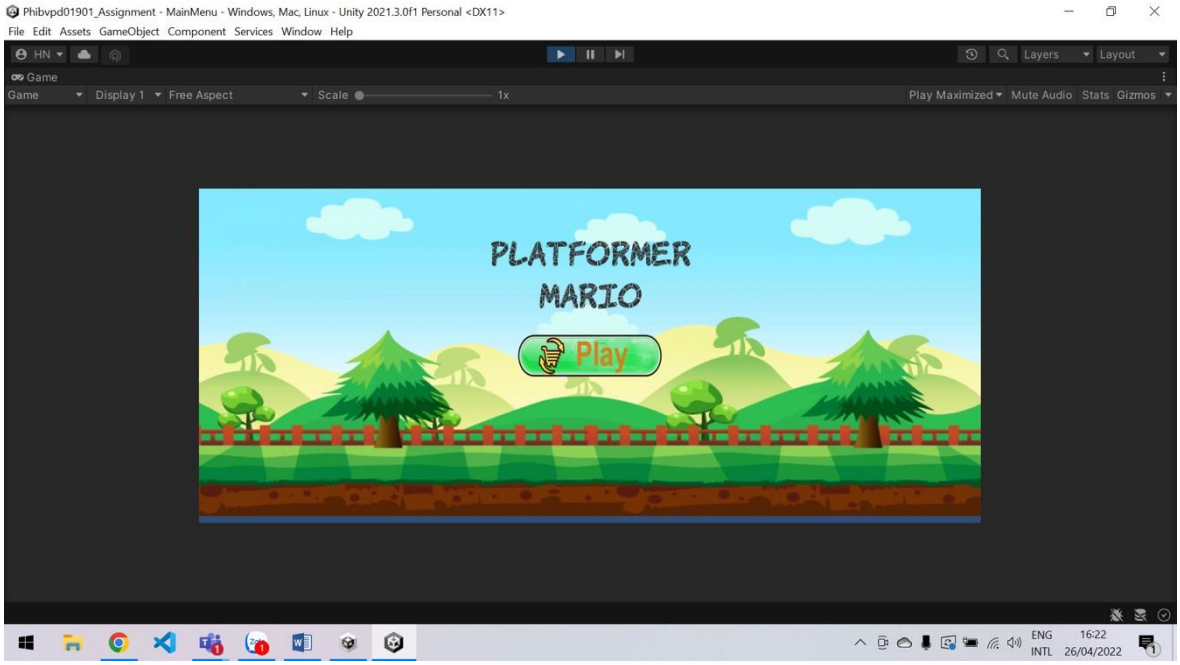
3.6.12 Xây dựng khi tiêu diệt 1 đơn vị sẽ rơi ra một vật phẩm

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

public class Drop : MonoBehaviour {
    public bool drop;
    public GameObject theDrop;
    // Use this for initialization
    void Start () {

    }

    // Update is called once per frame
    void Update () {

    }
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.tag == "Player")
        {
            gameObject.SetActive(true);
            Instantiate(theDrop, transform.position, transform.rotation);
            //lam roi ra item
            if (drop)
            {

                Instantiate(theDrop,transform.position, transform.rotation);

            }
        }
    }
}
```

Hình 3-15: Xây dựng khi tiêu diệt 1 đơn vị sẽ rơi ra một item

3.7 Demo Game

3.7.1 Giao diện màn hình chờ Menu

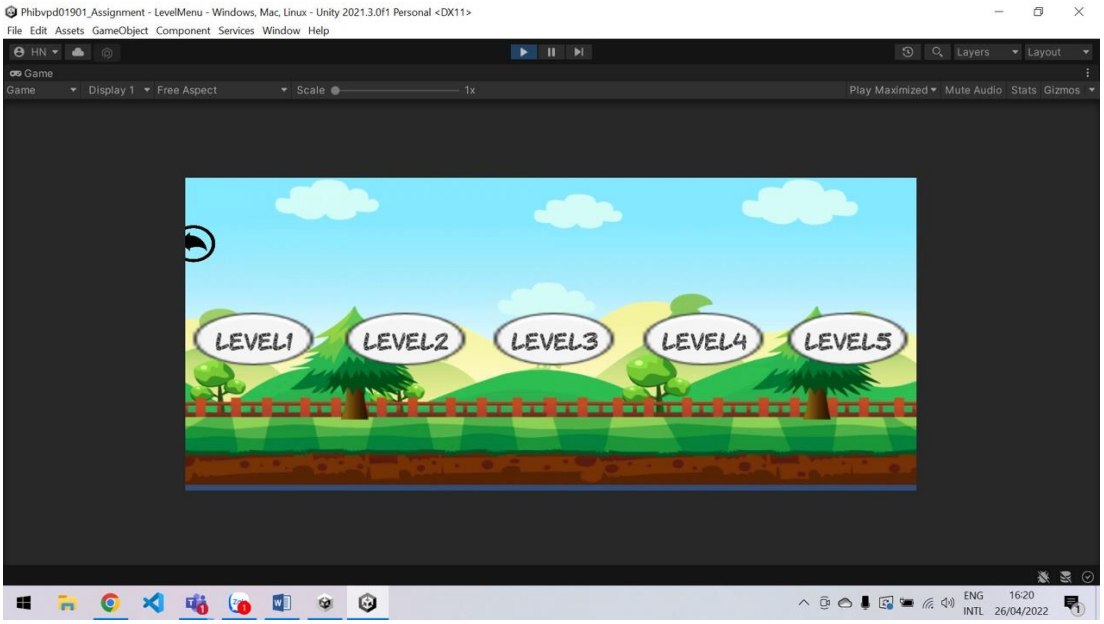


Hình 3-16 : Màn hình Menu Game

- Khi người chơi vào game màn hình sẽ hiện chức năng trong game.
    - Play : Khi click vào Play màn hình sẽ trực tiếp chuyển đến màn hình chọn map chơi cho người dùng

3.7.2 Màn hình chọn map chơi cho người dùng

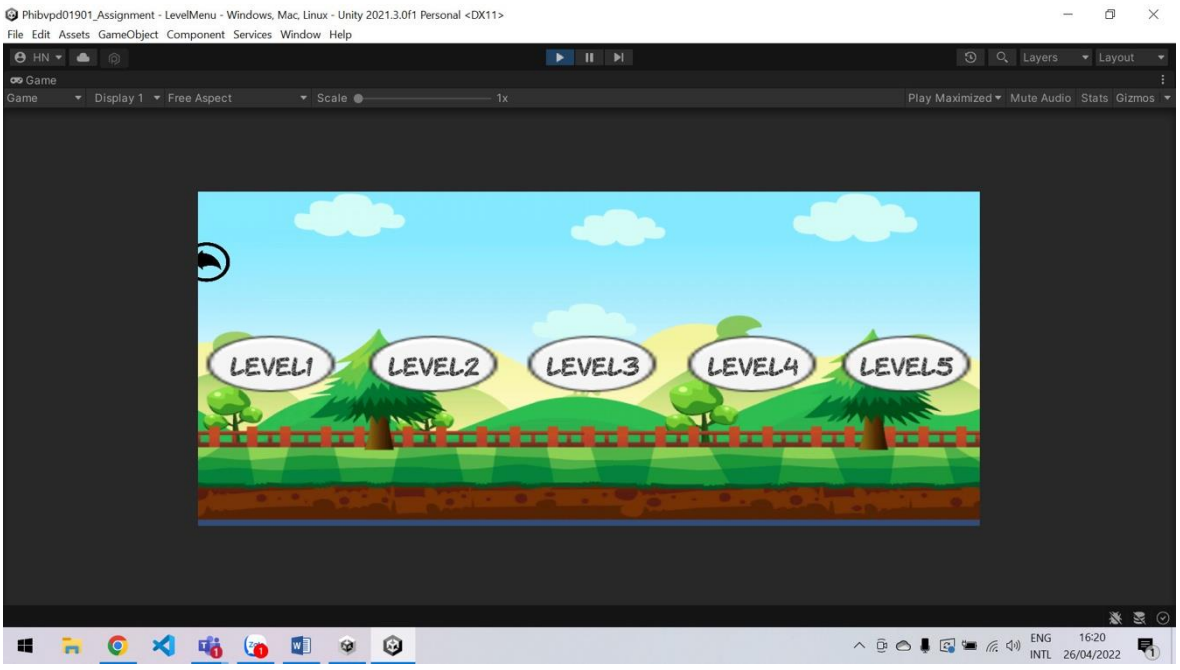
-Người chơi sẽ chọn map chơi theo ý muốn

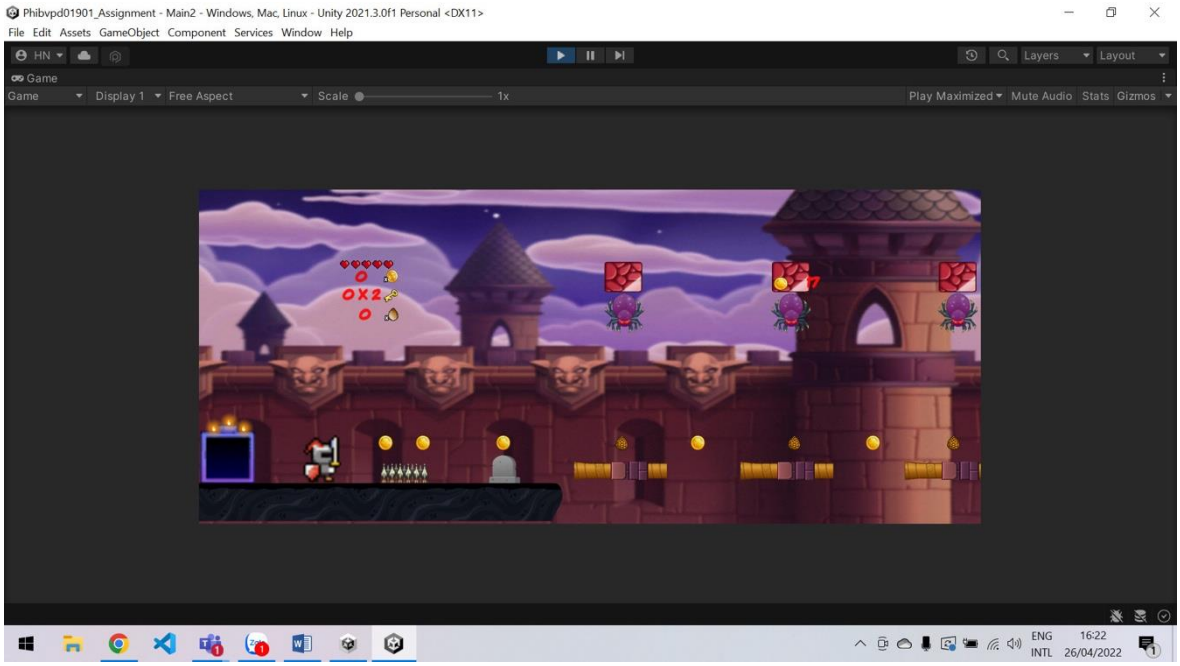


Hình 3-17-Màn hình chọn map theo ý muốn

3.7.3 Màn hình Game Play

- Khi click vào Play màn hình sẽ hiển thị:

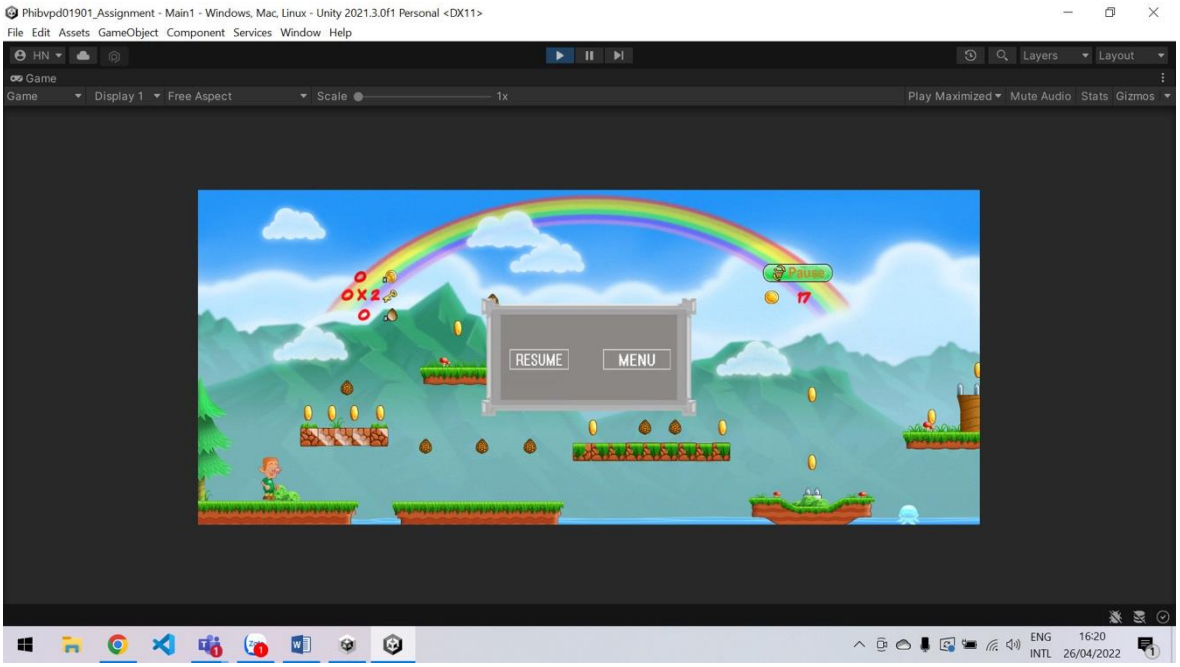




Hình 3-18 : Màn Hình Game Play

3.7.4 Màn hình Pause Game

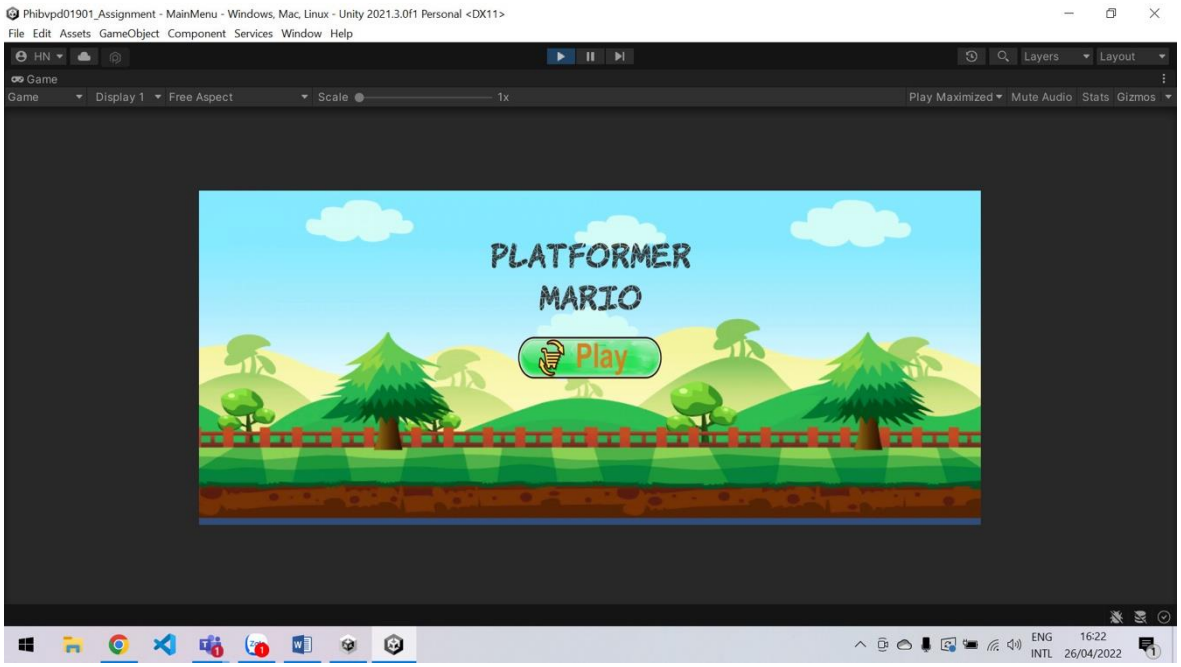
- Khi màn hình ấn nút Pause :



Hình 3-19 : Màn Hình Game Pause

3.7.5 Màn Hình Game Over

- Khi nhân vật chết màn hình sẽ chuyển đến



Hình 3-20 : Màn Hình Game Over

CHƯƠNG 4 : KẾT LUẬN

4.1 Kết quả đạt được của đề tài

- ✓ Trình bày được tổng quan về công nghệ Unity Engine
- ✓ Hiểu rõ được tác dụng của Animation.
- ✓ Hiểu rõ được cách làm game trên Unity.
- ✓ Xử lý được các lỗi cơ bản trong unity.
- ✓ Xử lý được âm thanh trong game.
- ✓ Xử lý được nhân vật trong game.
- ✓ Xử lý được các quái vật trong game.
- ✓ Xử lý được các vật phẩm trong game.
- ✓ Hoàn thành được game Mario.
- ✓ Ứng dụng công nghệ Unity xử lí các bài toán xây dựng lên hệ thống trò chơi hiệu quả. Sử dụng ngôn ngữ C# lập trình lên ứng dụng.
- ✓ Xây dựng được trò chơi "Mario" có tính giải trí cao và hiệu quả giúp người chơi thoải mái sau nhưng ngày làm việc mệt nhọc.
- ✓ Chức năng đơn giản dễ sử dụng phù hợp với mọi lứa tuổi.

4.2 Hạn chế của đề tài

- Chưa xử lý được tối ưu các rằng buộc, dữ liệu chưa được sắp xếp linh hoạt hợp lý.
- Còn nhiều chức năng chưa được hoàn thiện
- Chưa bắt được hết các lỗi của hệ thống.
- Chưa xử lý được trạng thái hệ thống bị dừng khi đang thao tác và còn một số tồn tại trong việc đặt tên và sử dụng linh hoạt các điều khiển.

4.3 Hướng phát triển của đề tài

- Tương tác được giữa người chơi thông qua hệ thống.
- Đồng bộ hóa dữ liệu giữa ứng dụng offline và hệ thống trực tuyến.
- Dữ liệu được tối ưu hóa đến mức chi tiết nhất.
- Tối ưu hóa nhân vật hơn giúp nhân vật chuyển được nhiều trạng thái nhân vật.

TÀI LIỆU THAM KHẢO

[1]. Janine Suvak - *Lập Trình Game Với Unity*

[2] Unity for Absolute Beginners

[3] Unity 2D Game Development

[4] Learn Unity for 2D Game Developmen

[5] Learning C# Programming With Unity 3D - Alex Okita

[6] Learn Unity3D Programming with UnityScript

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

[8] Terry Norton - Learning C# by Developing Games with Unity 3D Beginner's Guide - 2013

[9] Jeff Murray - C# Game Programming Cookbook for Unity 3D - 2014

[10] Charles Bernardoff - NGUI for Unity - 2014

[11]http://www.unity3dstudent.com/

[11]http://unity3d.com/learn

[11]http://vietgamedev.net

[11]http://forum.unity3d.com/threads/26785-Unity-Jump-Start-Video-Tutorials