TRƯỜNG ĐẠI HỌC KỸ THUẬT CÔNG NGHIỆP

KHOA ĐIỆN TỬ

BỘ MÔN: CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

LẬP TRÌNH GAME 3D VỚI UNITY

Đề tài : Lập trình game Inosuke

Sinh viên : **Hoàng Minh Chiến**

Nguyễn Đình Đức

Nguyễn Ngọc Hiếu

Chu Thanh Quyết

Lớp : **K55KMT**

Giáo viên hướng dẫn : **Đỗ Duy Cốp**

THÁI NGUYÊN – 2023

TRƯỜNG ĐHKTCN **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

KHOA ĐIỆN TỬ ***Độc lập - Tự do - Hạnh phúc***

BÀI TẬP LỚN

BÀI TẬP LỚN: LẬP TRÌNH GAME 3D VỚI UNITY

BỘ MÔN : CÔNG NGHỆ THÔNG TIN

Sinh viên : Hoàng Minh Chiến *MSSV : K195480106002*

Nguyễn Đình Đức *MSSV : K195480106006*

Nguyễn Ngọc Hiếu *MSSV : K195480106008*

Chu Thanh Quyết *MSSV : K195480106027*

Lớp : K55KMT *Ngành : Kỹ thuật máy tính*

Ngày giao đề : 30/03/2023 *Ngày hoàn thành : 12/06/2023*

1.Tên đề tài : Lập trình game Inoisuke.....

.....

.....

2. Yêu cầu : Phân tích, thiết kế, cài đặt và kiểm thử chương trình. Yêu cầu có giao diện phần mềm người dùng có thể sử dụng được.

TỔ TRƯỞNG BỘ MÔN

(Ký và ghi rõ họ tên)

GIÁO VIÊN HƯỚNG DẪN

(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

Thái Nguyên, ngày 06 tháng 06 năm 2023

GIÁO VIÊN HƯỚNG DẪN

(Ký ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN CHẤM

.....

.....

.....

.....

.....

Thái Nguyên, ngày.....tháng.....năm 2023

GIÁO VIÊN CHẤM

(Ký ghi rõ họ tên)

Bảng phân chia nhiệm vụ các thành viên

STT	Thành viên	Nhiệm vụ
1	Hoàng Minh Chiến	Code
2	Nguyễn Đình Đức	Code
3	Nguyễn Ngọc Hiếu	Design, Word, PowerPoint
4	Chu Thanh Quyết	Design, Word, PowerPoint

MỤC LỤC

CÁC HÌNH ẢNH SỬ DỤNG TRONG BÁO CÁO

LỜI MỞ ĐẦU

Trò chơi đã trở thành một phần không thể thiếu trong cuộc sống hiện đại, không chỉ để giải trí mà còn để kết nối và thúc đẩy sự sáng tạo. Với sự phát triển của công nghệ, lập trình game đã trở thành một lĩnh vực hấp dẫn, cho phép chúng ta tạo ra những thế giới ảo độc đáo và tương tác với người chơi.

Bài tập lớn này là một dự án lập trình game 2D, nơi chúng tôi đã đặt mục tiêu xây dựng một trò chơi hấp dẫn và gây thử thách cho người chơi. Trong quá trình này, chúng tôi sẽ khám phá các khía cạnh khác nhau của lập trình game, từ việc xác định yêu cầu cho đến triển khai và kiểm thử.

Mục tiêu của dự án này là học cách ứng dụng các khái niệm lập trình và các nguyên tắc thiết kế game vào việc xây dựng một trò chơi hoàn chỉnh. Chúng tôi cũng hy vọng rằng thông qua dự án này, chúng tôi sẽ nắm vững các kỹ năng lập trình, cải thiện khả năng xử lý vấn đề và học cách làm việc hiệu quả.

Chúng tôi muốn cảm ơn giảng viên và các bạn đã đóng góp và hỗ trợ trong suốt quá trình thực hiện dự án này. Chúng tôi tin rằng dự án này sẽ mang lại những kinh nghiệm quý giá và là một bước ngoặt trong hành trình phát triển cá nhân của chúng tôi trong lĩnh vực lập trình game.

CHƯƠNG I. GIỚI THIỆU CHUNG VỀ GAME

1.1. Giới thiệu game

a, Cốt truyện

Nội dung game là hành trình của Inoisuke giải của công chúa, trong quá trình giải cứu của mình, Inoisuke đã gặp và tiêu diệt rất nhiều quái vật để tìm đường đến giải cứu công chúa.

Cuộc hành trình của Inoisuke sẽ diễn ra thế nào ?

Liệu Inoisuke có thể giải cứu công chúa hay không ?

Inoisuke sẽ gặp những tên quái vật lợi hại như nào?

Chúng ta hãy cùng giúp cho Inoisuke giải cứu công chúa và quay trở về an toàn.

b, Đồ họa giao diện game

Đồ họa giao diện game được thiết kế với những phông màu cơ bản, tạo cho người chơi cảm giác thoải mát, tránh những chi tiết xen vào gây khó chịu cho người chơi.

Hình 1: Giao diện game

1.2. Mô tả trò chơi

Trò chơi được làm theo thể loại game nhập vai thành một tràng hoàng tử giải cứu công chúa. Sau khi bấm nút vào chơi game, người chơi được đưa đến

cảnh game hướng dẫn. Ở đây người chơi sẽ học cách thao tác cơ bản của game như di chuyển, tấn công quái vật, thu thập xu để tăng điểm số,thu thập kỹ năng,hồi máu cho nhân vật,… Sau khi hoàn thành màn hướng dẫn, người chơi đi qua cánh cửa để đến màn thứ nhất. Tại đây người chơi sẽ lần lượt vượt qua các màn với nhiều loại quái vật và bẫy, thu thập các vật phẩm như tiền xu, bình thuốc,.v.v.. Mỗi khi tiêu diệt được quái vật.

Khi đạp trúng bẫy hoặc bị dính đòn của quái vật, người chơi sẽ mất máu. Khi lượng máu này trở về 0, người chơi sẽ chết hay GameOver. Sau khi chết người chơi sẽ bị đưa về giao diện menu. Tại đây người chơi có thể chơi lại.

1.3. Yêu cầu đối với sản phẩm

- Game phải có dung lượng không quá lớn, tốc độ xử lý nhanh.
- Giao diện game dễ nhìn, thân thiện với người sử dụng.
- Công việc tính toán (va chạm, tấn công, điểm số, …) phải thực hiện chính xác, không có sai sót.
- Phân tích game theo hướng đối tượng cụ thể, rõ ràng.

1.4. Đầu vào đầu ra của sản phẩm

- **Đầu vào (inputs):** Các thao tác từ người chơi như bấm chuột, bấm bàn phím.

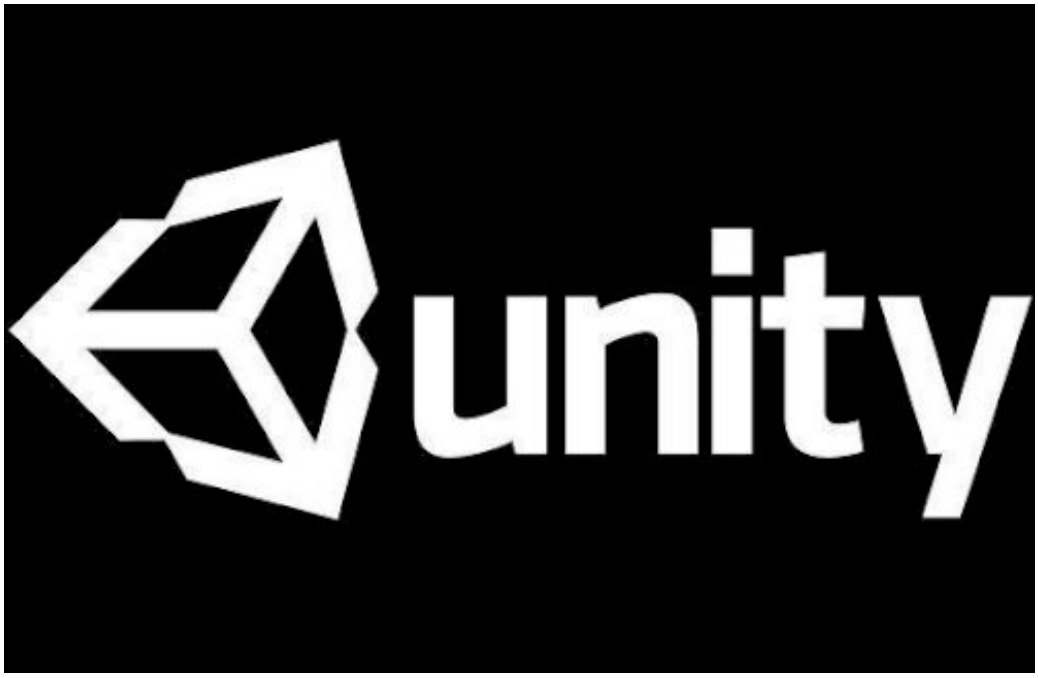
- **Đầu ra (outputs):** Hình ảnh trò chơi được hiển thị lên màn hình, thao tác được nhân vật.

CHƯƠNG II. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về Engine Unity

2.1.1. Khái niệm

Một Game Engine (hay công cụ tạo Game / động cơ Game) là một phần mềm được viết cho mục đích thiết kế và phát triển video Game. Có rất nhiều loại Game Engine dùng để thiết kế Game cho các hệ máy như hệ Consoles hay máy tính cá nhân (PC). Chức năng cốt lõi của Game Engine phần lớn nằm trong công cụ dựng hình (kết xuất đồ họa) cho các hình ảnh 2 chiều (2D) hay 3 chiều (3D), công cụ vật lý (hay công cụ tính toán và phát hiện va chạm), âm thanh, mã nguồn, hình ảnh động (Animation), trí tuệ nhân tạo, phân luồng, tạo dòng dữ liệu xử lý, quản lý bộ nhớ, dựng ảnh đồ thị, và kết nối mạng. Quá trình phát triển Game tiết kiệm được rất nhiều thời gian và kinh phí vào việc tái sử dụng và tái thích ứng một Engine để tạo nhiều Game khác nhau.



Hình 2: Logo của Unity

2.1.2. Mục đích

Game Engine cung cấp một bộ các công cụ phát triển trực quan và có thể tái sử dụng từng thành phần trong đó. Nói chung các bộ công cụ này cung cấp một môi trường phát triển tích hợp được đơn giản hóa. Phát triển ứng dụng nhanh (Rapid Application Development) cho Game theo cách lập trình hướng dữ liệu. Những Game Engine này đôi khi còn được gọi là các "phần mềm trung gian cho Game" (Game Middleware), như ý nghĩa của thuật ngữ, chúng cung cấp một nền tảng phần mềm linh hoạt và dễ dàng sử dụng lại với mọi chức năng

cốt lõi cần thiết ngay trong nó để có thể phát triển một ứng dụng Game đồng thời giảm giá thành, độ phức tạp, và kịp thời hạn phát hành - tất cả các yếu tố quan trọng trong ngành công nghiệp Game đầy cạnh tranh. Giống như các phần mềm trung gian khác, Game Engine thường cung cấp một nền tảng trừu tượng hóa, cho phép một Game có thể chạy trên nhiều hệ máy bao gồm các hệ console hoặc máy tính cá nhân với một vài thay đổi (nếu cần) trong mã nguồn của Game đó. Thông thường, phần mềm trung gian cho Game được thiết kế với một nền tảng kiến trúc dựa trên các thành phần khác, cho phép các hệ thống khác nhau trong Engine có thể thay thế hoặc mở rộng với các phần mềm trung gian khác chuyên biệt hơn như là Havok cho hệ thống vật lý trong Game, Miles Sound System cho âm thanh, hay Bink cho các đoạn video.

Một số Game Engine chỉ cung cấp khả năng dựng hình (kết xuất) 3D thời gian thực hay một khả năng riêng biệt nào khác thay vì rất nhiều chức năng trong phạm vi rộng mà Game yêu cầu. Loại Engine này thường được gọi là: "Graphics Engine", "Rendering Engine," hay "3D Engine" thay vì thuật ngữ bao quát hơn là "Game Engine". Một vài ví dụ cho các Engine đồ họa là: RealmForge, Truevision3D, OGRE, Crystal Space, Genesis3D, Vision Engine, Irrlicht và JMonkey Engine

2.1.3. Lịch sử

Thuật ngữ "Game Engine" xuất hiện vào giữa những năm 90, đặc biệt là trong mối quan hệ giữa Game 3D và Game bắn súng góc nhìn người thứ nhất (FPS). Như các thương hiệu nổi tiếng của id Sortware: Doom và Quake , thay vì phải làm việc từ đầu, các nhà phát triển khác (nếu được cấp phép) sẽ có quyền truy nhập vào phần lõi (mã nguồn) của Game và thiết kế các hình ảnh, nhân vật, vũ khí, và các màn chơi của riêng họ - gọi là Game Content (nội dung Game) hay "Game Assets" (tài sản Game).

Các Game sau này, như Quake III Arena và sản phẩm năm 1998 của Epic Games: Unreal được thiết kế với cách tiếp cận mới này, Game Engine và nội dung Game được chia thành các phần riêng biệt để phát triển. Engine có thể tái sử dụng khiến việc phát triển Game tiếp theo nhanh hơn và dễ dàng hơn, một thuận lợi to lớn trong ngành công nghiệp cạnh tranh này.

Game Engine hiện đại là một trong những ứng dụng được viết ra (bằng các ngôn ngữ lập trình) phức tạp nhất, thường xuyên phải có rất nhiều tinh chỉnh trong hệ thống để đảm bảo kiểm soát chính xác trải nghiệm người dùng. Sự phát triển liên tục của Game Engine đã tạo ra một sự phân chia mạnh mẽ giữa các công việc dựng hình, viết kịch bản, thiết kế hình ảnh, và thiết kế màn chơi. Hiện nay thông thường một đội ngũ phát triển Game điển hình phải có số lượng họa sĩ gấp vài lần số lượng lập trình viên.

2.2. Game 2D là gì?

2.2.1. Khái Niệm

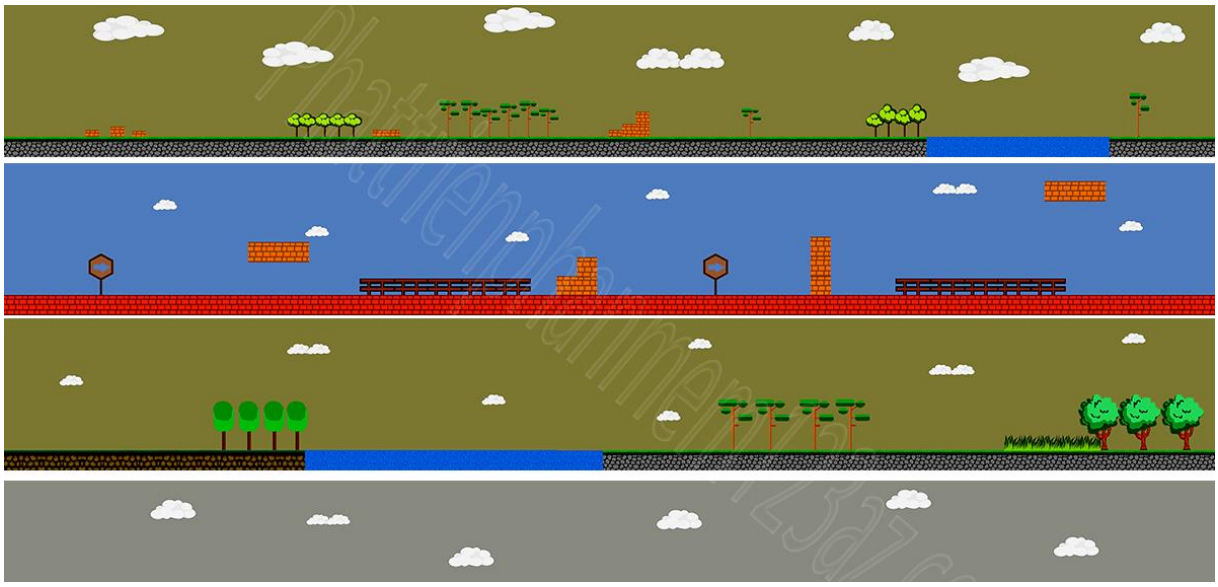
Game 2D (2D game) là một loại trò chơi được thiết kế và hiển thị trên một mặt phẳng hai chiều. Trái ngược với game 3D, trong đó các đối tượng và môi trường được biểu diễn trong không gian ba chiều, game 2D thường sử dụng các hình ảnh và hình vẽ được vẽ trên một mặt phẳng phẳng, tạo ra cảm giác chuyển động và tương tác.

Trong game 2D, người chơi thường chỉ có thể di chuyển và tương tác với các đối tượng trong mặt phẳng ngang và dọc, thường là bên trái và bên phải hoặc trên và dưới màn hình. Game 2D có thể có nhiều thể loại khác nhau, bao gồm game platformer, game RPG (role-playing game), game puzzle và nhiều hơn nữa.

Các trò chơi 2D thường sử dụng các công nghệ và công cụ phát triển như ngôn ngữ lập trình, thư viện đồ họa và công cụ phát triển game. Các ngôn ngữ phổ biến để lập trình game 2D bao gồm C++, C#, Java và Python. Ngoài ra, các thư viện và framework như Pygame, Unity và Unreal Engine cũng được sử dụng để xây dựng game 2D một cách dễ dàng và hiệu quả.

Game 2D thường có thiết kế đơn giản và gần gũi với người chơi, với các đồ họa và hiệu ứng đơn giản. Tuy nhiên, điều quan trọng là gameplay và cách thức tương tác của người chơi, để tạo ra một trải nghiệm thú vị và gây thử thách.

Với sự phát triển của công nghệ và nền tảng di động, game 2D vẫn là một thể loại phổ biến và đa dạng, thu hút mọi lứa tuổi và sở thích. Cùng với sự sáng tạo và kỹ năng lập trình, game 2D có thể mở ra cơ hội để tạo ra những trò chơi độc đáo và đáng nhớ.



Hình 3: Game 2D

2.2.2. Các yếu tố hình thành thể loại game

Cốt truyện: Cốt truyện là tâm điểm của trò chơi, nó mang lại sự kết nối giữa các sự kiện và nhân vật trong game. Một cốt truyện tốt giúp người chơi thấy hứng thú và tạo ra sự gắn kết với trò chơi. Cốt truyện có thể là một cuộc phiêu lưu, một hành trình tìm kiếm, hoặc một câu chuyện về sự vượt qua khó khăn và thách thức.

Nhân vật: Nhân vật trong game 2D là các thực thể mà người chơi sẽ điều khiển hoặc tương tác trong trò chơi. Các nhân vật có thể là anh hùng, nhân vật chính, kẻ phản diện hoặc những con thú đáng yêu. Mỗi nhân vật nên có đặc điểm và kỹ năng riêng, từ đó tạo ra sự đa dạng và lựa chọn cho người chơi.

Thế giới và môi trường: Thế giới và môi trường trong game 2D là nền tảng để người chơi trải nghiệm. Điều quan trọng là tạo ra một thế giới độc đáo với các địa điểm, cảnh quan và môi trường tương tác. Bối cảnh có thể là một thành phố hiện đại, một vùng nông thôn hoặc một thế giới hư cấu đầy phép thuật.

Thời gian và không gian: Xác định thời gian và không gian trong game 2D là quan trọng để tạo ra sự truyền cảm và logic cho câu chuyện. Thời gian có thể là hiện tại, quá khứ hoặc tương lai, trong khi không gian có thể là một vùng đất rộng lớn hoặc một cấu trúc nhỏ hẹp.

Tương tác và nhiệm vụ: Bối cảnh và cốt truyện trong game 2D thường được xây dựng xung quanh các nhiệm vụ mà người chơi phải hoàn thành. Những nhiệm vụ này có thể là câu đố, chiến đấu với kẻ thù, hoặc tương tác với các nhân vật khác trong game. Các nhiệm vụ thúc đẩy sự tiến triển của cốt truyện, cung cấp phần thưởng và tạo ra sự thử thách cho người chơi.

Trạng thái và sự phát triển: Trong game 2D, người chơi thường trải qua sự phát triển và tiến độ qua các trạng thái khác nhau. Có thể là sự tiến bộ qua các cấp độ khác nhau, mở khóa vật phẩm mới, nâng cấp nhân vật hoặc khám phá các khu vực mới trong trò chơi. Sự phát triển này tạo ra sự hứng thú và khám phá liên tục, đồng thời thách thức người chơi để tiếp tục chơi.

Tạo cảm xúc và trải nghiệm: Bối cảnh và cốt truyện cùng nhau tạo ra cảm xúc và trải nghiệm cho người chơi. Một game 2D có thể mang lại cảm giác hồi hộp, phấn khích, thư giãn hoặc sợ hãi. Bằng cách sử dụng âm nhạc, hình ảnh, âm thanh và tương tác, game 2D tạo ra một trải nghiệm đầy mê hoặc cho người chơi.

Tương tác đa dạng: Game 2D cung cấp nhiều hình thức tương tác cho người chơi. Điều này có thể là di chuyển, nhảy, tấn công, tương tác với đối

tượng trong game hoặc tương tác với các nhân vật khác. Việc tạo ra tương tác đa dạng giúp tăng tính tương tác và sự thú vị của trò chơi.

2.3. UML

UML là ngôn ngữ mô hình hoá, ngôn ngữ đặc tả và ngôn ngữ xây dựng mô hình trong quá trình phát triển phần mềm, đặc biệt là trong phân tích và thiết kế hệ thống hướng đối tượng. UML là ngôn ngữ hình thức, thống nhất và chuẩn hoá mô hình hệ thống một cách trực quan. Nghĩa là các thành phần trong mô hình được thể hiện bởi các ký hiệu đồ hoạ, biểu đồ và thể hiện đầy đủ mối quan hệ giữa các chúng một cách thống nhất và có logic chặt chẽ.

Hình 4. UML – Ngôn ngữ mô hình hóa các yêu cầu phần mềm

2.4. Phân tích thiết kế hướng đối tượng và UML

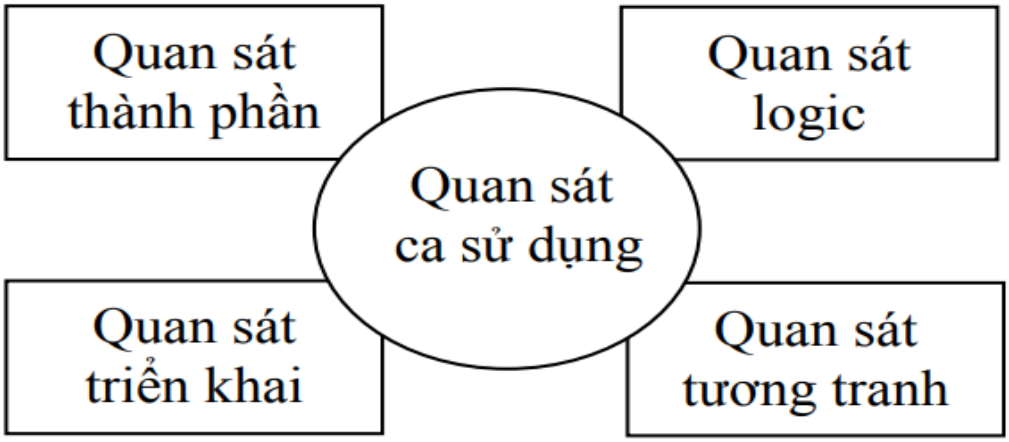
Phân tích thiết kế hướng đối tượng cần bản vẽ để mô tả hệ thống được thiết kế còn UML là ngôn ngữ mô tả các bản vẽ nên cần nội dung để thực hiện. Do vậy, chứng ta phân tích và thiết kế theo hướng đối tượng và sử dụng UML để biểu diễn các thiết kế đó nên chúng thường đi đôi với nhau.

Phân tích thiết kế hướng đối tượng sử dụng UML bao gồm các thành phần:

- Các sự vật
- Các mối quan hệ
- Các biểu đồ
- Các quan sát

2.4.1. Các quan sát

Các quan sát (góc nhìn) theo các phương diện khác nhau của hệ thống cần phân tích, thiết kế. Dựa vào các quan sát để thiết lập kiến trúc cho hệ thống cần phát triển. Có năm loại quan sát: quan sát theo ca sử dụng, quan sát logic, quan sát thành phần, quan sát tương tranh và quan sát triển khai. Mỗi quan sát tập trung khảo sát và mô tả một khía cạnh của hệ thống và thường được thể hiện trong một số biểu đồ nhất định.



Hình 5. Các quan sát khác nhau

- **Quan sát các ca sử dụng** (hay trường hợp sử dụng): Mô tả các chức năng, nhiệm vụ của hệ thống. Quan sát này thể hiện mọi yêu cầu của hệ thống, do vậy nó phải được xác định ngay từ đầu và nó được sử dụng để điều khiển, thúc đẩy và thẩm định hay kiểm tra các công việc của tất cả các giai đoạn của cả quá trình phát triển phần mềm.
- **Quan sát logic:** Biểu diễn cách tổ chức logic của các lớp và các quan hệ của chúng với nhau. Nó mô tả cấu trúc tĩnh của các lớp, đối tượng và sự liên hệ của chúng thể hiện mối liên kết động thông qua sự trao đổi các thông điệp. Quan sát được thể hiện trong các biểu đồ lớp, biểu đồ đối tượng, biểu đồ tương tác, biểu đồ biến đổi trạng thái. Quan sát logic tập trung vào cấu trúc của hệ thống. Trong quan sát này ta nhận ra các bộ phận cơ bản cấu thành hệ thống thể hiện mọi quá trình trao đổi, xử lý thông tin cơ bản trong hệ thống.
- **Quan sát thành phần** (Quan sát cài đặt): Xác định các mô đun vật lý hay tệp mã chương trình và sự liên hệ giữa chúng để tổ chức thành hệ thống phần mềm. Trong quan sát này ta cần bổ sung: chiến lược cấp phát tài nguyên cho từng thành phần, và thông tin quản lý như báo cáo tiến độ thực hiện công việc, v.v. Quan sát thành phần được thể hiện trong các biểu đồ thành phần và các gói
- **Quan sát tương tranh** (quan sát tiến trình): Biểu diễn sự phân chia các luồng thực hiện công việc, các lớp đối tượng cho các tiến trình và sự đồng bộ giữa các luồng trong hệ thống. Quan sát này tập trung vào các nhiệm vụ tương tranh, tương tác với nhau trong hệ thống đa nhiệm.
- **Quan sát triển khai**: Mô tả sự phân bổ tài nguyên và nhiệm vụ trong hệ thống. Nó liên quan đến các tầng kiến trúc của phần mềm, thường là kiến trúc ba tầng: tầng giao diện (tầng trình diễn hay tầng sử dụng), tầng logic tác nghiệp và tầng lưu trữ CSDL được tổ chức trên một hay nhiều máy

tính khác nhau. Quan sát triển khai bao gồm các luồng công việc, bộ xử lý và các thiết bị. Biểu đồ triển khai mô tả các tiến trình và chỉ ra những tiến trình nào trên máy nào

2.4.2. Các sự vật

UML có bốn phần tử mô hình, đó là cấu trúc, hành vi, nhóm và chú thích.

- **Phần tử cấu trúc:** là các danh từ trong mô hình UML, biểu diễn cho các thành phần khái niệm hay vật lý của hệ thống. UML có bảy phần tử cấu trúc được mô tả như sau:
  - **Lớp**: Lớp là tập các đối tượng cùng chia sẻ với nhau về các thuộc tính, thao tác, quan hệ và ngữ nghĩa
  - **Giao diện**: Giao diện là tập các thao tác làm dịch vụ cho lớp hay thành phần. Giao diện mô tả hành vi quan sát được từ bên ngoài thành phần. Giao diện chỉ khai báo các phương thức xử lý nhưng không định nghĩa nội dung thực hiện. Nó thường không đứng một mình mà thường được gắn với lớp hay một thành phần
  - **Phần tử cộng tác**: Phần tử cộng tác mô tả ngữ cảnh của sự tương tác trong hệ thống. Nó thể hiện một giải pháp thi hành trong hệ thống, bao gồm các lớp, quan hệ và sự tương tác giưa chúng để thực hiện một ca sử dụng như mong đợi.
  - **Ca sử dụng**: Ca sử dụng mô tả một tập các hành động mà hệ thống sẽ thực hiện để phục vụ cho các tác nhân ngoài. Tác nhân ngoài là những gì bên ngoài có tương tác, trao đổi với hệ thống
  - **Lớp tích cực**: Lớp tích cực được xem như là lớp có đối tượng làm chủ một hay nhiều tiến trình, luồng hành động.
  - **Thành phần**: Thành phần biểu diễn vật lý mã nguồn, các tệp nhị phân trong quá trình phát triển hệ thống.
  - **Nút**: Nút thể hiện thành phần vật lý tồn tại khi chương trình chạy và biểu diễn cho các tài nguyên được sử dụng trong hệ thống.
  -
- **Phần tử nhóm:** là bộ phận tổ chức của mô hình UML. Phần tử nhóm có gói, mô hình và khung công việc
  - **Gói** (Package): Gói là phần tử đa năng được sử dụng để tổ chức các lớp, hay một số nhóm khác vào trong một nhóm. Không giống với thành phần (component), phần tử gói hoàn toàn là khái niệm, có nghĩa là chúng chỉ tồn tại trong mô hình vào thời điểm phát triển hệ thống chứ không tồn tại vào thời điểm chạy chương trình. Gói giúp ta quan sát hệ thống ở mức tổng quát
  - **Mô hình**: Mô hình là những mô tả về các đặc tính tĩnh và/hoặc động của các chủ thể trong hệ thống

- **Khung công việc**: Khung công việc là một tập các lớp trừu tượng hay cụ thể được sử dụng như là các khuôn mẫu để giải quyết một họ các vấn đề tương tự

2.4.3. Các mối quan hệ

UML cho phép biểu diễn cả bốn mối quan hệ giữa các đối tượng trong các hệ thống. Đó là các quan hệ: phụ thuộc, kết hợp, tổng quát hoá và hiện thực hoá.

- **Quan hệ phụ thuộc:** Đây là quan hệ ngữ nghĩa giữa hai phần tử, trong đó sựu thay đổi của một tử sẽ tác động đến ngữ nghĩa của phần tử phụ thuộc.
- **Quan hệ kết hợp:** Kết hợp là quan hệ cấu trúc xác định mối liên kết giữa các lớp đối tượng. Khi có một đối tượng của lớp này gửi/nhận thông điệp đến/từ chỗ đối tượng của lớp kia thì hai lớp đó có quan hệ kết hợp. Một dạng đặc biệt của quan hệ kết hợp là quan hệ kết nhập, biểu diễn mối quan hệ giữa toàn thể và bộ phận.
- **Quan hệ tổng quát hoá:** Đây là quan hệ mô tả sự khái quát hoá mà trong đó một số đối tượng cụ thể (của lớp con) sẽ được kế thừa các thuộc tính, các phương thức của các đối tượng tổng quát (lớp cơ sở).
- **Hiện thực hoá:** Hiện thực hoá là quan hệ ngữ nghĩa giữa giao diện và lớp (hay thành phần) để thực hiện cài đặt các dịch vụ đã được khai báo trong các giao diện

2.4.4. Các biểu đồ

Biểu đồ là đồ thị biểu diễn đồ họa về tập các phần tử trong mô hình và mối quan hệ của chúng. Biểu đồ chứa đựng các nội dung của các quan sát dưới các góc độ khác nhau, một thành phần của hệ thống có thể xuất hiện trong một hay nhiều biểu đồ. UML cung cấp những biểu đồ trực quan để biểu diễn các khía cạnh khác nhau của hệ thống, bao gồm:

- **Biểu đồ ca sử dụng:** Mô tả sự tương tác giữa các tác nhân ngoài và hệ thống thông qua các ca sử dụng. Các ca sử dụng là những nhiệm vụ chính, các dịch vụ, những trường hợp sử dụng cụ thể mà hệ thống cung cấp cho người sử dụng và ngược lại.
- **Biểu đồ lớp mô tả cấu trúc tĩnh:** Mô tả mô hình khái niệm bao gồm các lớp đối tượng và các mối quan hệ của chúng trong hệ thống hướng đối tượng. Biểu đồ trình tự thể hiện sự tương tác của các đối tượng với nhau, chủ yếu là trình tự gửi và nhận thông điệp để thực thi các yêu cầu, các công việc theo thời gian.
- **Biểu đồ cộng tác:** Tương tự như biểu đồ trình tự nhưng nhấn mạnh vào sự tương tác của các đối tượng trên cơ sở cộng tác với nhau bằng cách trao đổi các thông điệp để thực hiện các yêu cầu theo ngữ cảnh công việc.

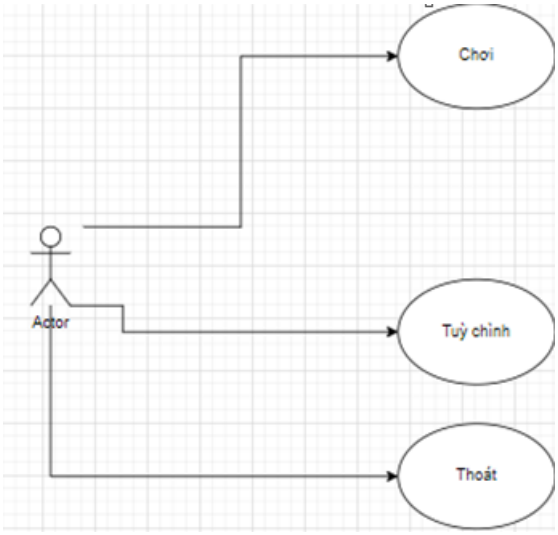
- **Biểu đồ trạng thái:** Thể hiện chu kỳ hoạt động của các đối tượng, của các hệ thống con và của cả hệ thống. Nó là một loại ôtômát hữu hạn trạng thái, mô tả các trạng thái, các hành động mà đối tượng có thể có và các sự kiện gắn với các trạng thái theo thời gian.
- **Biểu đồ hành động:** Chỉ ra dòng hoạt động của hệ thống, bao gồm các trạng thái hoạt động, trong đó từ một trạng thái hoạt động sẽ chuyển sang trạng thái khác sau khi một hoạt động tương ứng được thực hiện. Nó chỉ ra trình tự các bước, tiến trình thực hiện cũng như các điểm quyết định và sự rẽ nhánh theo luồng sự kiện.
- **Biểu đồ thành phần:** Chỉ ra cấu trúc vật lý của các thành phần trong hệ thống, bao gồm: các thành phần mã nguồn, mã nhị phân, thư viện và các thành phần thực thi.
- **Biểu đồ triển khai:** Chỉ ra cách bố trí vật lý các thành phần theo kiến trúc được thiết kế của hệ thống.

CHƯƠNG III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1. Các chức năng của hệ thống

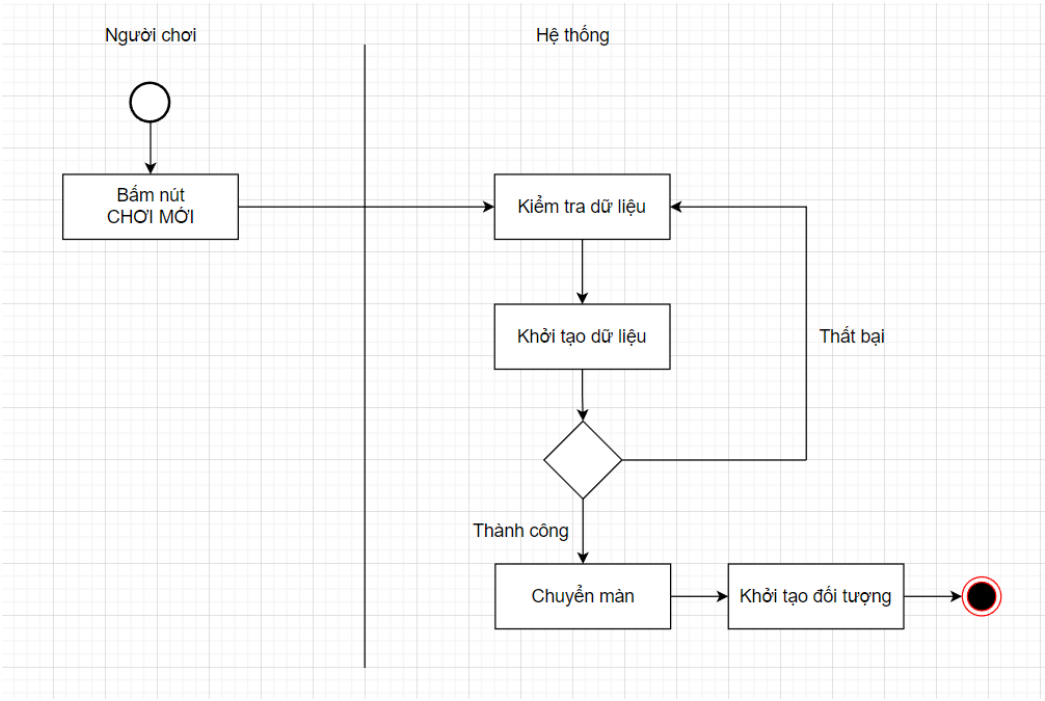
STT	Tên chức năng	Mô tả
1	Bắt đầu và thoát game	Khi vào game sẽ có các tùy chọn như chơi game, cài đặt và thoát game
2	Điều khiển nhân vật	Di chuyển nhân vật theo các hướng trong môi trường 2D như trái, phải, nhảy, sử dụng kỹ năng.
3	Tương tác với các đối tượng	Có thể tương tác với các đối tượng như chuyển màn, hồi máu, nhặt xu…
4	Tấn công	Tấn công gây sát thương lên quái vật, phá hủy các đồ vật như thùng gỗ, tường…

3.2. Biểu đồ Use-Case

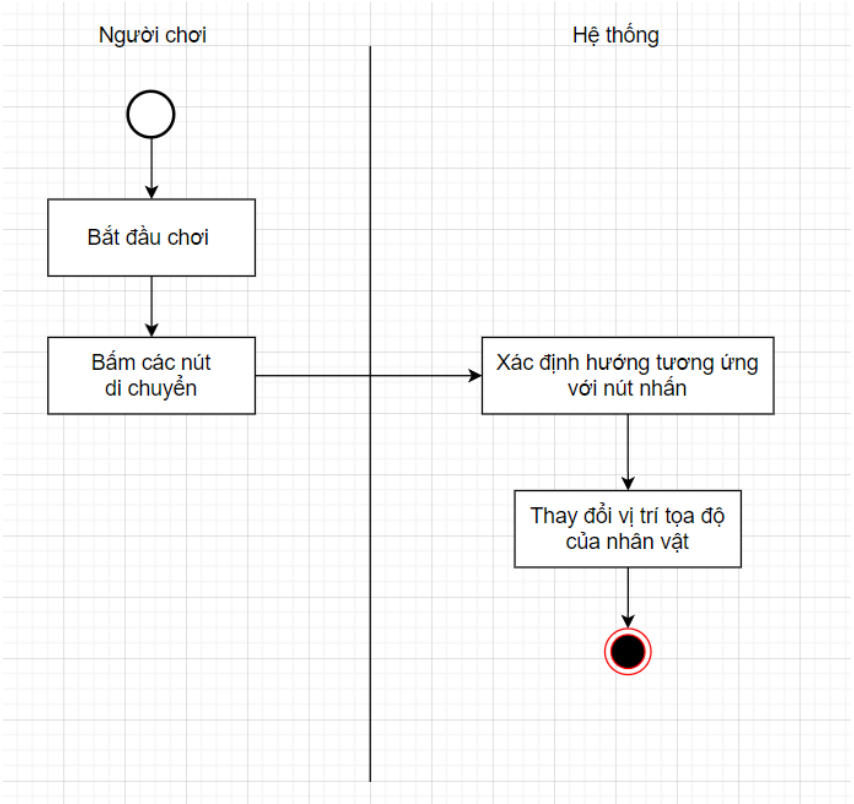


Hình 6. Biểu đồ Use-Case

3.3. Biểu đồ hoạt động

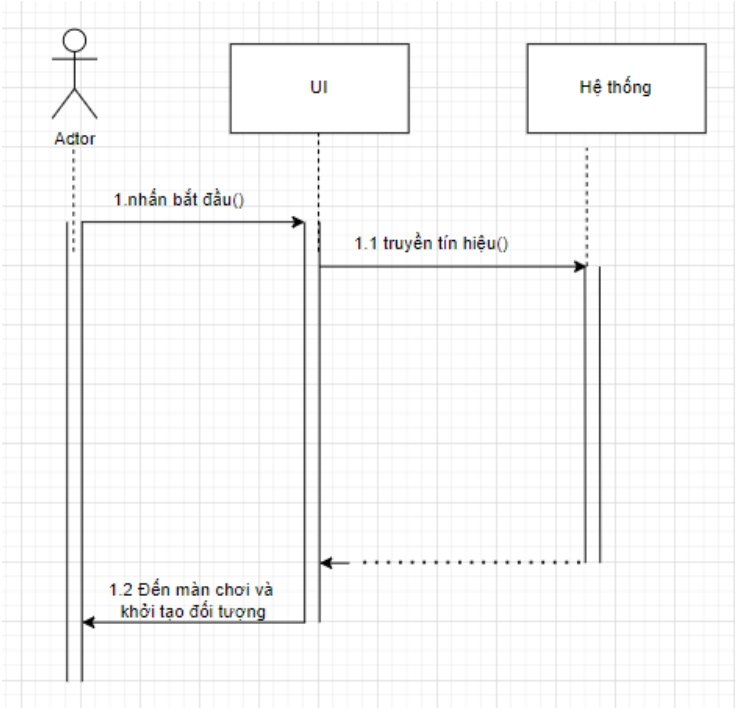


Hình 7. Biểu đồ hoạt động bắt đầu game

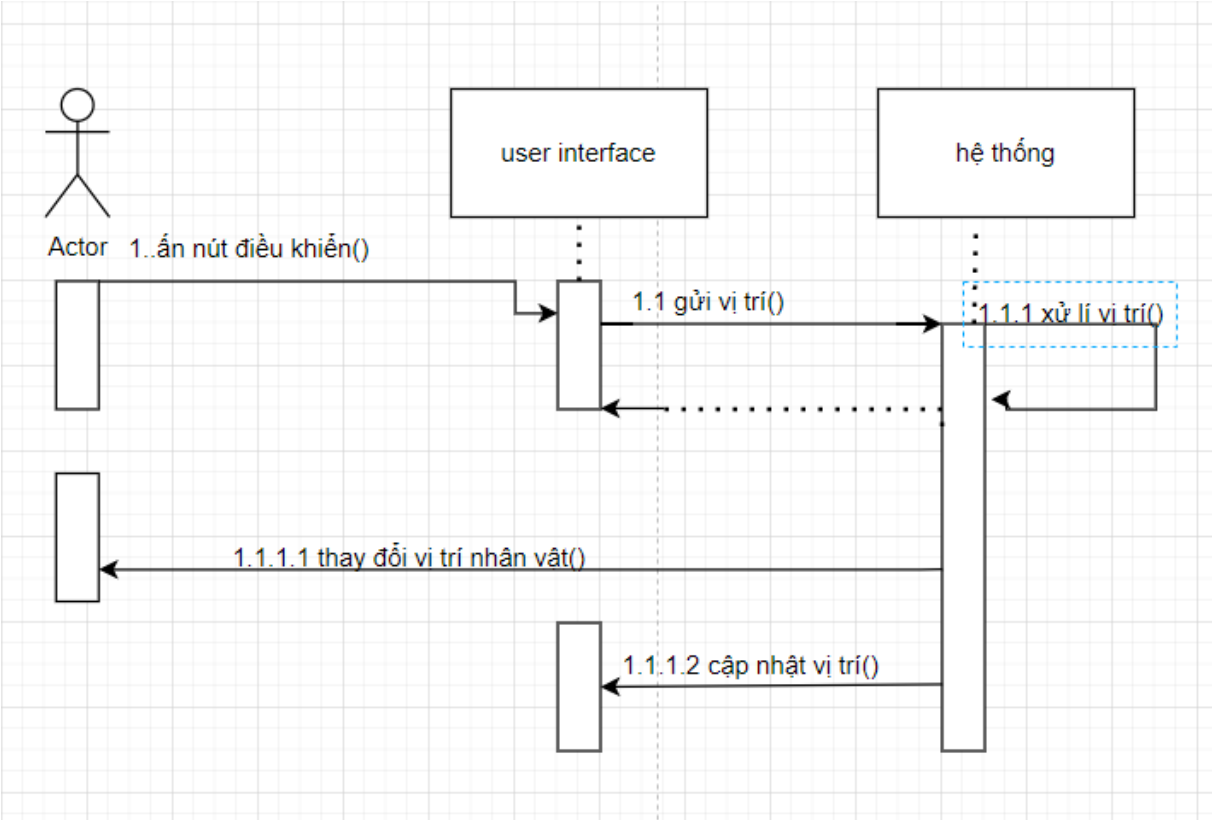


Hình 8. Biểu đồ điều khiển nhân vật

3.4. Biểu đồ tuần tự

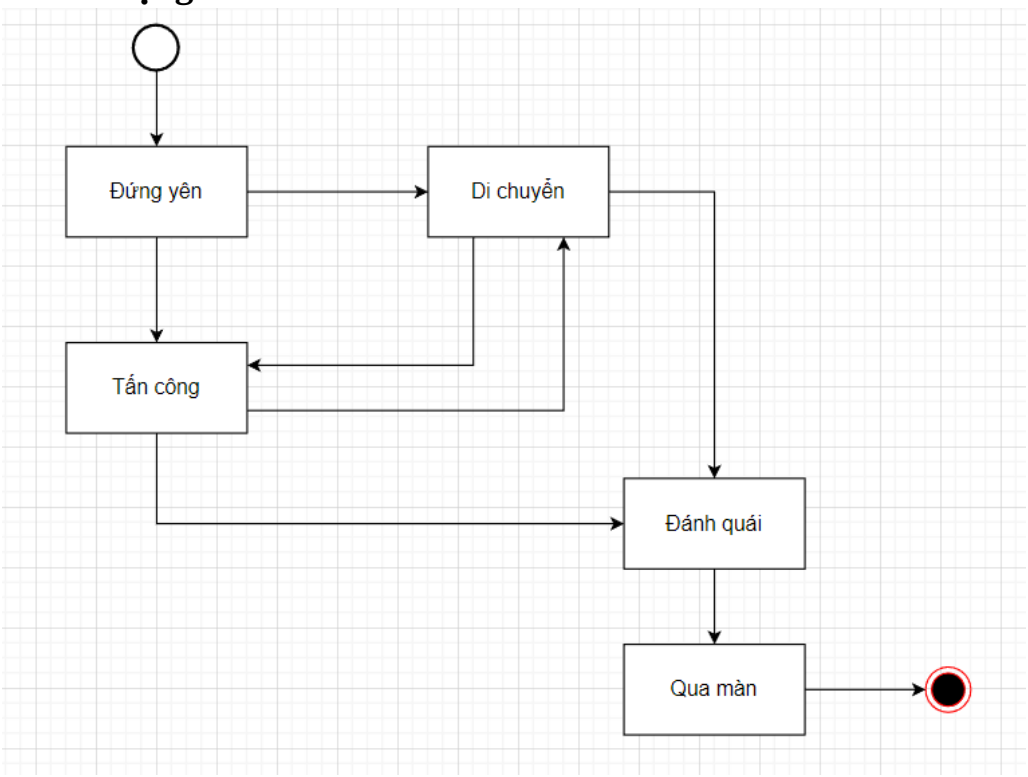


Hình 9. Biểu đồ tuần tự bắt đầu chơi game



Hình 10. Biểu đồ tuần tự điều khiển nhân vật

3.5. Biểu đồ trạng thái



Hình 11. Biểu đồ trạng thái

CHƯƠNG IV. LẬP TRÌNH VÀ KIỂM THỬ

4.1. Microsoft Visual Studio

4.1.1. Visual Studio là gì ?

Microsoft Visual Studio là một môi trường phát triển tích hợp (IDE) từ Microsoft. Microsoft Visual Studio còn được gọi là "Trình soạn thảo mã nhiều người sử dụng nhất thế giới ", được dùng để lập trình C++ và C# là chính. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

Hình 12. Microsoft Visual Studio

Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C, C++ và C++/CLI, VB.NET , C# và F#. Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ

cài đặt riêng rẽ. Nó cũng hỗ
trợ XML/XSLT, HTML/XHTML, JavaScript và CSS.

4.1.2. Một số tính năng của Visual Studio

Editor mã nguồn: Visual Studio cung cấp một trình soạn thảo mã nguồn thông minh và mạnh mẽ, hỗ trợ nhiều ngôn ngữ lập trình như C++, C#, Python, JavaScript, và nhiều ngôn ngữ khác. Nó cung cấp gợi ý mã, kiểm tra lỗi cú pháp, định dạng mã tự động và nhiều tính năng khác giúp tăng năng suất và độ chính xác trong việc viết mã.

Gỡ lỗi (Debugging): Visual Studio cung cấp chức năng gỡ lỗi mạnh mẽ giúp lập trình viên tìm ra và khắc phục lỗi trong mã nguồn. Nó cho phép theo dõi giá trị biến, bước qua từng dòng mã, theo dõi gọi hàm và thậm chí xem lại lịch sử gỡ lỗi để phân tích và khắc phục sự cố.

IntelliSense: Đây là tính năng tự động hoàn thành mã trong quá trình viết. IntelliSense của Visual Studio cung cấp các gợi ý mã thông minh dựa trên ngữ cảnh, giúp lập trình viên tiết kiệm thời gian và giảm số lỗi cú pháp.

Quản lý phiên bản: Visual Studio tích hợp tính năng quản lý phiên bản, cho phép lập trình viên làm việc với các hệ thống quản lý phiên bản phổ biến như Git và SVN. Điều này giúp theo dõi sự thay đổi trong mã nguồn, quản lý nhánh và hợp nhất mã nguồn từ nhiều nguồn khác nhau.

Kiểm tra và Tự động hóa: Visual Studio cung cấp các công cụ để kiểm tra mã tự động và xác thực. Lập trình viên có thể viết các bài kiểm tra tự động và chạy chúng để đảm bảo tính đúng đắn của mã nguồn và tránh các lỗi tiềm ẩn.

Thiết kế giao diện người dùng: Visual Studio đi kèm với các công cụ thiết kế giao diện người dùng để tạo và chỉnh sửa các giao diện người dùng cho ứng dụng. Lập trình viên có thể sử dụng trình kéo và thả để tạo các thành phần giao diện như nút, hộp văn bản, danh sách, và nhiều hơn nữa. Các công cụ thiết kế cho phép tùy chỉnh và định dạng giao diện người dùng một cách trực quan, giúp tăng tính thẩm mỹ và trải nghiệm người dùng của ứng dụng.

Hỗ trợ đa nền tảng: Visual Studio hỗ trợ phát triển ứng dụng đa nền tảng, cho phép lập trình viên xây dựng ứng dụng cho nhiều nền tảng khác nhau như Windows, macOS, iOS, Android và Linux. Điều này giúp tiết kiệm thời gian và công sức trong việc phát triển và duy trì ứng dụng trên nhiều nền tảng, đồng thời mở rộng đối tượng người dùng mục tiêu.

IntelliCode: Tính năng IntelliCode trong Visual Studio sử dụng trí tuệ nhân tạo để đề xuất mã nguồn dựa trên dựa trên dữ liệu từ hàng ngàn dự án mã

nguồn mở. Nó cung cấp gợi ý và đề xuất các đoạn mã phổ biến, giúp lập trình viên tăng tốc độ viết mã và cải thiện chất lượng của mã.

Tích hợp công cụ bên thứ ba: Visual Studio hỗ trợ tích hợp và sử dụng các công cụ bên thứ ba như bộ kiểm tra và phân tích mã, công cụ gỡ lỗi và hỗ trợ phân tích các ngôn ngữ lập trình khác nhau. Điều này cho phép lập trình viên mở rộng khả năng và hiệu suất của môi trường phát triển bằng cách tích hợp các công cụ và tiện ích mạnh mẽ từ các nhà cung cấp khác.

Hỗ trợ công cụ kiểm thử: Visual Studio cung cấp các công cụ và khung kiểm thử tích hợp để giúp lập trình viên kiểm tra và đảm bảo chất lượng của ứng dụng. Các công cụ kiểm thử bao gồm kiểm tra đơn vị, kiểm tra tích hợp, kiểm tra hiệu năng và các công cụ kiểm tra tự động khác. Điều này giúp đảm bảo ứng dụng hoạt động một cách chính xác và ổn định trước khi được triển khai.

Hỗ trợ cộng tác: Visual Studio cho phép nhiều lập trình viên làm việc cùng nhau trên cùng một dự án thông qua tính năng cộng tác. Công cụ này cho phép lập trình viên chia sẻ mã nguồn, quản lý phiên bản và gửi nhận phản hồi từ đồng đội một cách dễ dàng, tăng cường khả năng làm việc nhóm và tăng cường hiệu suất phát triển.

Hỗ trợ DevOps: Visual Studio tích hợp tốt với quy trình phát triển và triển khai DevOps, giúp tạo ra một quy trình phát triển liền mạch và tự động. Các tính năng tích hợp DevOps bao gồm triển khai liên tục, xây dựng tự động, kiểm tra tự động và triển khai tự động. Điều này giúp giảm thời gian và công sức cần thiết để triển khai ứng dụng và đảm bảo sự ổn định và chất lượng của nó.

Hỗ trợ mở rộng: Visual Studio cho phép lập trình viên mở rộng và tùy chỉnh môi trường phát triển bằng cách sử dụng các phần mở rộng và tiện ích. Có sẵn một cộng đồng lớn phát triển các phần mở rộng, người dùng có thể tìm và cài đặt các công cụ và tính năng bổ sung để đáp ứng nhu cầu cụ thể của dự án và phong cách lập trình.

4.1.3. Các phiên bản phổ biến của Visual Studio

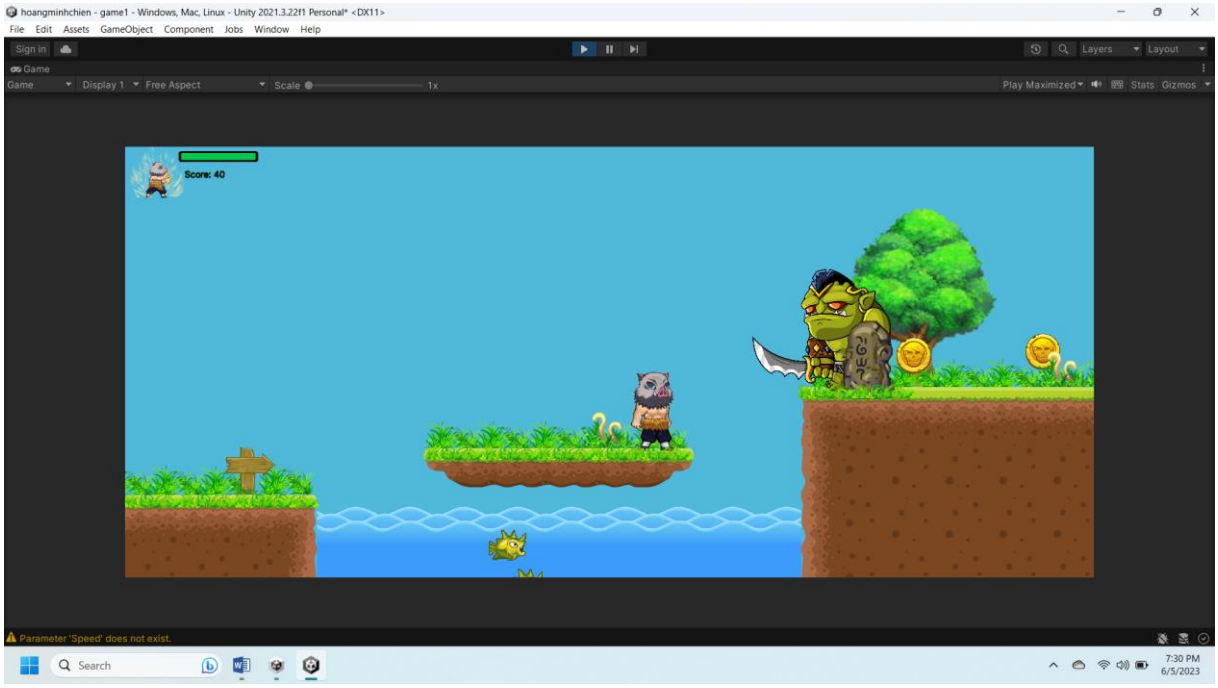
- **Visual Studio Community:** Đây là phiên bản miễn phí có đầy đủ tính năng, có thể mở rộng. Phiên bản dành cho sinh viên, nhà phát triển nguồn mở và cá nhân, để tạo các ứng dụng hiện đại cho Android, IOS, Windows, cũng như các ứng dụng web và dịch vụ đám mây. Mục đích chính của nó là cung cấp hỗ trợ Ecosystem (hàng nghìn tiện ích mở rộng) và Language (có thể lập trình bằng C#, C++, HTML, JavaScript, Python, v.v.).
- **Visual Studio Professional:** Đây là phiên bản thương mại của Visual Studio, hỗ trợ XML và XSLT editing và cả công cụ như Server Explorer, tích hợp với Microsoft SQL Server. Với phiên bản này, người dùng được

dùng thử miễn phí và sau đó cần trả phí để tiếp tục sử dụng. Mục đích chính của phiên bản này là cung cấp các công cụ dành cho nhà phát triển chuyên nghiệp để xây dựng bất kỳ loại ứng dụng nào, các tính năng mạnh mẽ như CodeLens cải thiện năng suất của team, các công cụ lập kế hoạch dự án (Agile, biểu đồ, …) và các lợi ích của Subscriber như phần mềm Microsoft, cùng với Azure, Pluralsight, v.v.

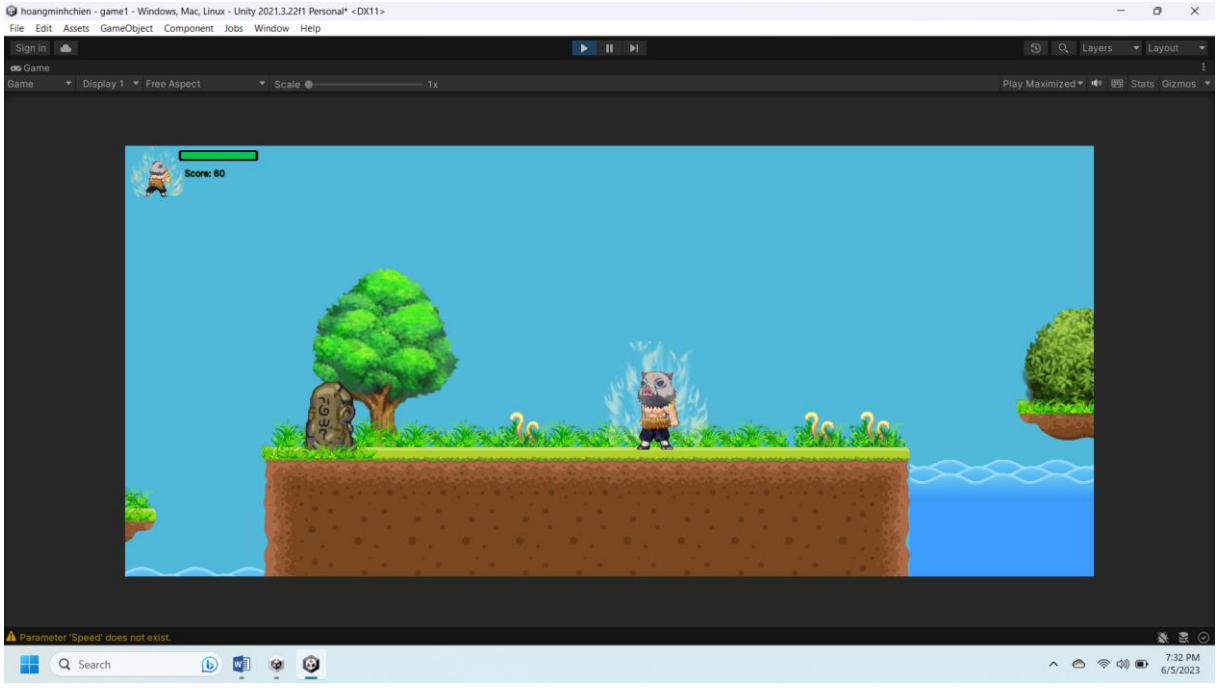
- **Visual Studio Enterprise:** Đây là một giải pháp tích hợp end-to-end cho các team thuộc bất kỳ quy mô nào với nhu cầu mở rộng cũng như yêu cầu chất lượng khắt khe. Với phiên bản này, người dùng được dùng thử miễn phí 90 ngày và sau đó cần trả phí để tiếp tục sử dụng. Lợi ích chính của phiên bản này là có khả năng mở rộng tốt và cung cấp phần mềm chất lượng cao.

4.2. Giao diện trò chơi

Hình 13. Giao diện Menu trò chơi



Hình 14. Giao diện màn chơi



Hình 15. Giao diện nhân vật có kỹ năng mới

Hình 16. Giao diện tổng quan màn 1

CHƯƠNG V. KẾT LUẬN

4.1. Nhận xét và kết luận

Sản phẩm tồn tại những:

- **Ưu điểm:** Hệ thống đã hoàn thiện các bước cơ bản như phân tích thiết kế, cài đặt kiểm thử, bước đầu tạo nên một trò chơi hoàn chỉnh. Trò chơi đáp ứng được những chức năng cơ bản, có dung lượng nhẹ, các lỗi bug của trò chơi hầu hết đã được xử lý.

- **Nhược điểm:** Việc phân tích vẫn còn chưa được trực quan, hình ảnh trong trò chơi vẫn chưa được mượt mà và đẹp mắt. Trò chơi chưa có âm thanh, thời lượng chơi ngắn do chưa đáp ứng được đủ thời gian để thực hiện. Các chức năng của trò chơi cần được hoàn thiện thêm và thêm nhiều kỹ năng, quái vật cũng như độ khó mới, chưa design cho nhân có động tác mượt mà.

4.2. Hướng phát triển của bài tập

Trò chơi sẽ phát triển thêm âm thanh, thêm các màn chơi mới, vũ khí mới, kỹ năng mới, các quái vật mới, các nhân vật và độ khó. Trò chơi cần tối ưu về hình ảnh và thuật toán.

Do trình độ của em là có hạn và chưa có nhiều kinh nghiệm trong việc làm game và việc sử dụng hai công cụ Unity và ngôn ngữ lập trình C# nên bài tập vẫn chưa thể hoàn thành trọn vẹn, còn nhiều những thiếu sót hạn chế nhất định.

TÀI LIỆU THAM KHẢO

https://visualstudio.microsoft.com/
https://www.jetbrains.com/
https://www.udemy.com/vi/topic/unity/
https://codegym.vn/blog/2021/03/30/tron-bo-tai-lieu-lap-trinh-game-unity-co-ban-den-nang-cao/
https://chat.openai.com/
https://viblo.asia/