TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

——————— * ———————

BÀI TẬP LỚN

MÔN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Xây Dựng Một Game Nhập Vai Đơn Giản SimpleRPG

Sinh viên thực hiện : Nhóm 2

STT	Họ và tên	MSSV
1	Hoàng Đặng Tuấn Đạt	20183701
2	Lê Hà Hưng	20183757
3	Nguyễn Quang Hưng	20183760
4	Nguyễn Duy Khai	20183771

Lớp : Kĩ thuật máy tính – **Khoá 63**
Giáo viên hướng dẫn : ThS. **Nguyễn Mạnh Tuấn**

Hà Nội, tháng 6 năm 2020

MỤC LỤC

LỜI NÓI ĐẦU

Công nghệ thông tin trong giai đoạn hiện nay đang có những bước phát triển như vũ bão trên mọi lĩnh vực của đời sống xã hội. Đối với Việt Nam, tuy ngành công nghệ thông tin mới chỉ phát triển mạnh trong vòng 10 năm trở lại đây nhưng những bước phát triển đó đã cho thấy Việt Nam có tiềm năng rất lớn. Trong đó, ngành công nghiệp game của nước ta đang được các công ty, tập đoàn, doanh nghiệp đầu tư vô cùng mạnh mẽ. Các phần mềm game càng lúc càng đòi hỏi nhiều sự khắt khe, sáng tạo, đồ họa đẹp mắt để đáp ứng được yêu cầu giải trí, thậm chí là đam mê của hàng tỷ người dùng.

Qua thời gian tìm hiểu thị trường game Việt Nam và nhận thấy tiềm năng to lớn trên, nhóm chúng em đã quyết định cùng nhau phát triển một game nhập vai với mục tiêu đáp ứng nhu cầu giải trí của người chơi và xa hơn nữa là hướng tới các game thủ chuyên nghiệp. Tuy nhiên, vì thời gian không có nhiều nên nhóm chúng em quyết định chỉ thiết kế một game nhập vai đơn giản.

Quá trình tiếp cận đề tài: đầu tiên là phác thảo biểu đồ Use Case từ tổng quát đến cụ thể với nhiều mức phân rã khác nhau, sau đó thiết kế cấu trúc dữ liệu, lập biểu đồ trình tự, biểu đồ lớp và thiết kế chi tiết cho từng lớp. Sau cùng là xây dựng hệ thống mã nguồn và cấu trúc lưu trữ dữ liệu cũng thuật toán cho phần mềm.

Mặc dù đã cố gắng hoàn thiện sản phẩm nhưng không thể tránh khỏi những thiếu hụt về kiến thức và sai sót trong kiểm thử phần mềm. Chúng em mong muốn nhận được những nhận xét thẳng thắn, chi tiết đến từ thầy và các bạn để tiếp tục hoàn thiện hơn nữa. Cuối cùng, nhóm chúng em xin được gửi lời cảm ơn đến thầy Nguyễn Mạnh Tuấn đã hướng dẫn bọn em trong suốt quá trình hoàn thiện phần mềm. Xin trân trọng cảm ơn thầy.

PHÂN CÔNG THÀNH VIÊN TRONG NHÓM

Họ và tên	**Email**	**Điện thoại**	**Công việc thực hiện**	**Đánh giá**
Hoàng Đặng Tuấn Đạt	dat.hdt183701@sis.hust.edu.vn	0846379998	• Phát triển mã nguồn • Viết báo cáo	
Lê Hà Hưng	hung.lh183757@sis.hust.edu.vn	0976594801	• Thiết kế đồ họa cho game • Viết mã nguồn xây dựng hệ thống Character • Làm slide	
Nguyễn Quang Hưng	hung.nq183760@sis.hust.edu.vn	0949376925	• Phân tích yêu cầu bài toán, thiết kế các biểu đồ Use Case • Xây dựng mã nguồn hệ thống Screens • Viết báo cáo	
Nguyễn Duy Khai	khai.nd183771@sis.hust.edu.vn		• Phát triển mã nguồn cho phần mềm • Kiểm thử phần mềm	

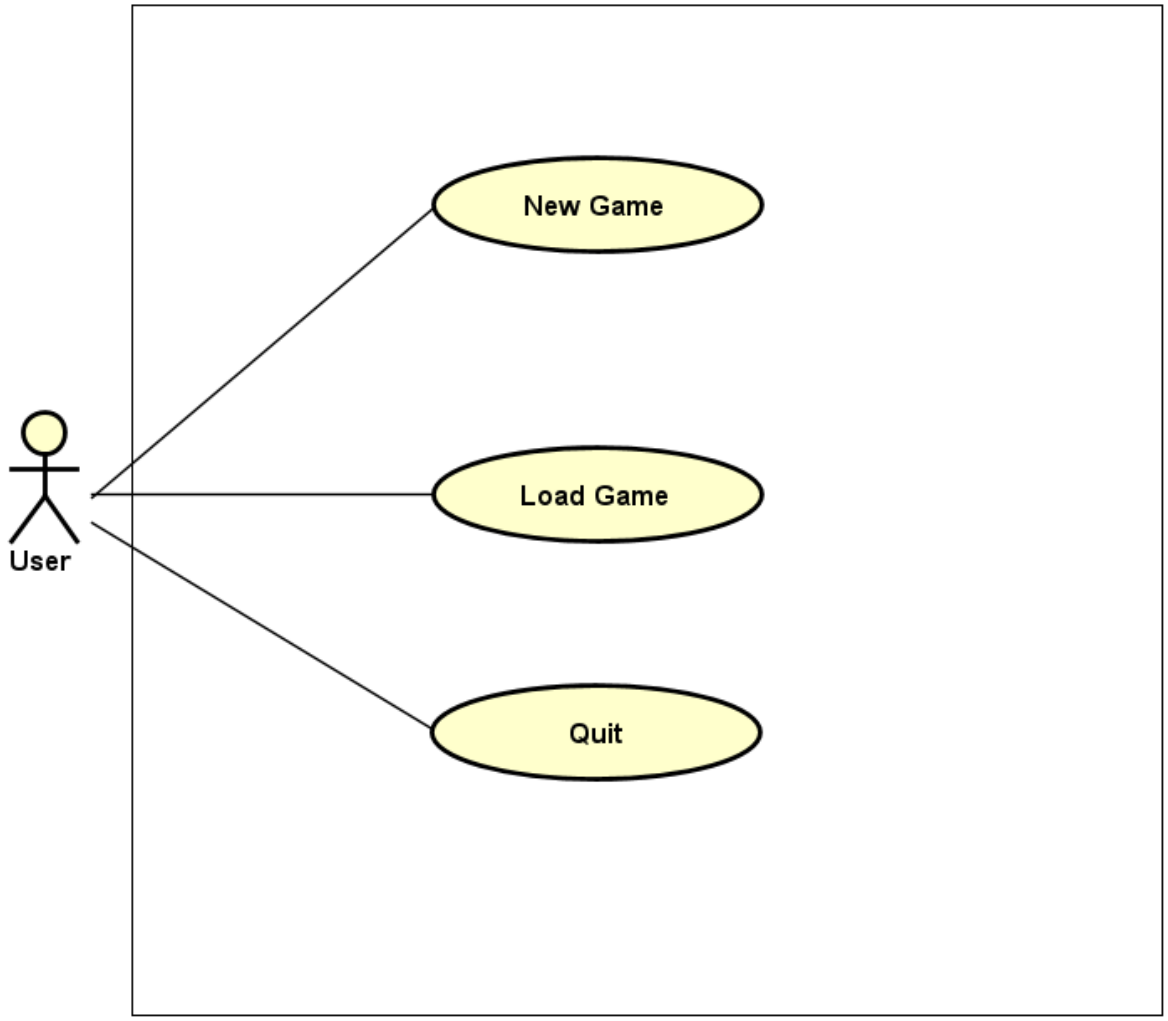
CHƯƠNG 1. KHẢO SÁT, ĐẶC TẢ YÊU CẦU BÀI TOÁN

1.1. Mô tả yêu cầu bài toán

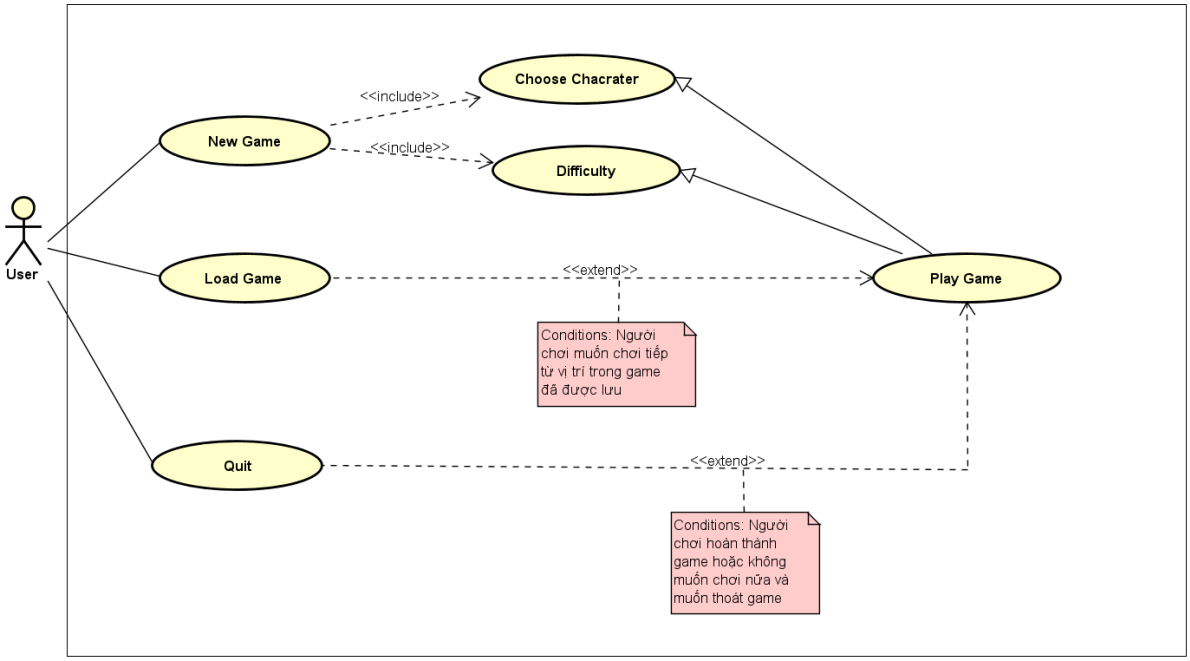
- Người chơi điều khiển một nhân vật trong một bản đồ được lưu trong một cấu trúc dữ liệu, trong đó mỗi ô tương ứng với một dạng bản đồ khác nhau (đất, cỏ, nước...)
- Trên bản đồ có các quái vật có thể di chuyển được. Các nhân vật người chơi điều khiển và quái vật có các chỉ số xác định tình trạng và thể lực (ví dụ HP, MP, Attack, Defense, Speed...).
- Người chơi có thể tấn công quái vật và sử dụng các kỹ năng đặc biệt. Tương tự, quái vật cũng có thể tìm đến và tấn công người chơi.
- Người chơi có thể di chuyển qua lại giữa các bản đồ khác nhau (ví dụ khi đi vào vùng M0, M1, M2... trên bản đồ) hoặc đi đến kết thúc của trò chơi (ví dụ khi đi vào vùng END trên bản đồ).
- Quái vật phải được chia thành nhiều loại như NPC, Boss, ...
- Trò chơi có 3 mức độ easy, normal, hard

1.2. Biểu đồ Use Case

1.2.1. Biểu đồ Use Case tổng quan



1.2.2. Biểu đồ Use Case phân rã mức 2



1.3. Đặc tả Use Case

1.3.1. Use Case Play Game

Use Case ID	UC-1.1
Use Case Name	New Game
Description	Là người dùng, tôi muốn truy cập vào ứng dụng để bắt đầu game
Actor(s)	Người dùng
Priority	Must Have
Trigger	Người dùng muốn truy cập vào game

Pre-Condition(s):	Thiết bị của người dùng có cài đặt game và thỏa mãn các yêu cầu về phần cứng.
Post-Condition(s):	Giao diện Game hiện ra, người dùng bắt đầu chơi
Basic Flow	1. Người dùng mở game 2. Người dùng ấn vào button Play Game 3. Người dùng chọn nhân vật 4. Người dùng chọn độ khó 5. Hệ thống bắt đầu game với nhân vật và độ khó đã chọn
Alternative Flow	
Exception Flow	
Business Rules	
Non-Functional Requirement	

1.3.2. Use case Load Game

Use Case ID	UC-1.2
Use Case Name	Load Game
Description	Là người dùng, tôi muốn tiếp tục game từ 1 bản lưu có sẵn

Actor(s)	Người dùng
Priority	Không bắt buộc
Trigger	Người dùng muốn tiếp tục 1 game từ 1 bản cho trước
Pre-Condition(s):	Thiết bị của người dùng có cài đặt game và thỏa mãn các yêu cầu về phần cứng, có ít nhất 1 bản lưu mà người dùng đã lưu từ trước
Post-Condition(s):	Giao diện Game hiện ra với tất cả những thuộc tính cũ mà người dùng đã lưu lại từ trước
Basic Flow	1. Người dùng mở game 2. Người dùng ấn vào button Load Game 3. Người dùng chọn bản lưu muốn chơi 4. Hệ thống load file bản ghi và thuộc tính có trong bản lưu cũ
Alternative Flow	
Exception Flow	Không có bản lưu game
Business Rules	
Non-Functional Requirement	

1.3.3. Use case Quit

Use Case ID	UC-1.3
Use Case Name	Quit
Description	Là người dùng, tôi muốn thoát khỏi ứng dụng
Actor(s)	Người dùng
Priority	Must Have
Trigger	Người dùng muốn thoát khỏi game
Pre-Condition(s):	Thiết bị của người dùng có cài đặt game và thỏa mãn các yêu cầu về phần cứng và đã mở game từ trước
Post-Condition(s):	Game kết thúc
Basic Flow	1. Người dùng ấn vào button Exit 2. Hệ thống kết thúc toàn bộ chương trình game
Alternative Flow	
Exception Flow	
Business Rules	
Non-Functional Requirement	

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ BÀI TOÁN

2.1. Thiết kế Cơ sở dữ liệu hoặc Cấu trúc tệp dữ liệu

2.1.1. Dữ liệu hình ảnh

Để thực hiện việc tạo các menu chính của game, menu chọn nhân vật, chọn độ khó cho game; chúng em đã thiết kế sẵn các file ảnh tĩnh và lưu riêng vào 1 thư mục. Khi chạy chương trình, chúng em sử dụng thư viện có sẵn của ngôn ngữ C# là Drawings để vẽ các ảnh lên màn hình.

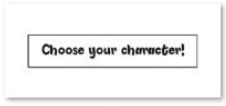

Characters


Credit


GameOver

Mask

ModeSelect



Text

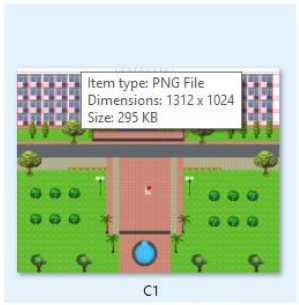

Title1


Title2

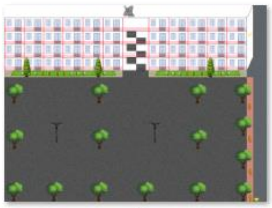

Title3

TitleTheme

Victory



C1


C9


D3

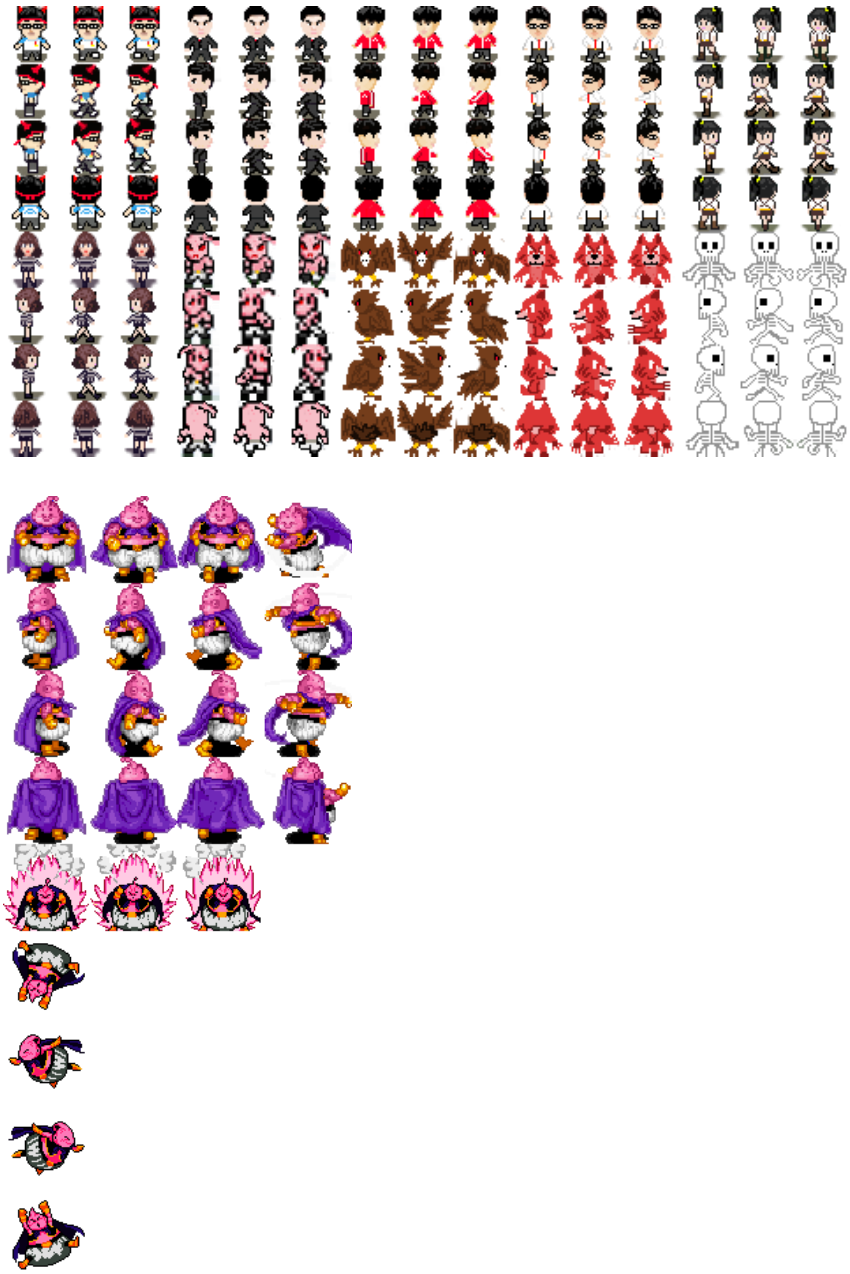

StartMap


Tiny


TQB


truong

Tiếp đó, chúng em cũng sử dụng các tấm ảnh Textures như bên dưới, gọi là sprite, để vẽ các Tile của bản đồ, nhân vật, các quái vật, con boss, hiệu ứng skill,… lên màn hình.



2.1.2. Dữ liệu âm thanh

Chúng em cũng tạo hệ thống các file âm thanh, video dùng trong game. Khi người chơi đánh quái, đánh boss,… sẽ có các âm thanh hoặc các video khi thua, khi clear game.

2.1.3. Dữ liệu hệ thống save/load

Nhóm em tạo một chương trình phụ trong phần mềm gọi là hệ thống hệ thống Map Editor. Tại đây, developer có thể tạo các tile của map, tạo kiểu nhân vật, tạo scene,... bằng GUI. Sau khi tạo, các object của class map, kiểu nhân vật,

scene,... sẽ được ghi ra thành file binary lưu vào trong ổ cứng. Trong quá trình chơi game, những file này sẽ được đọc trở lại thành những vật thể trong game.

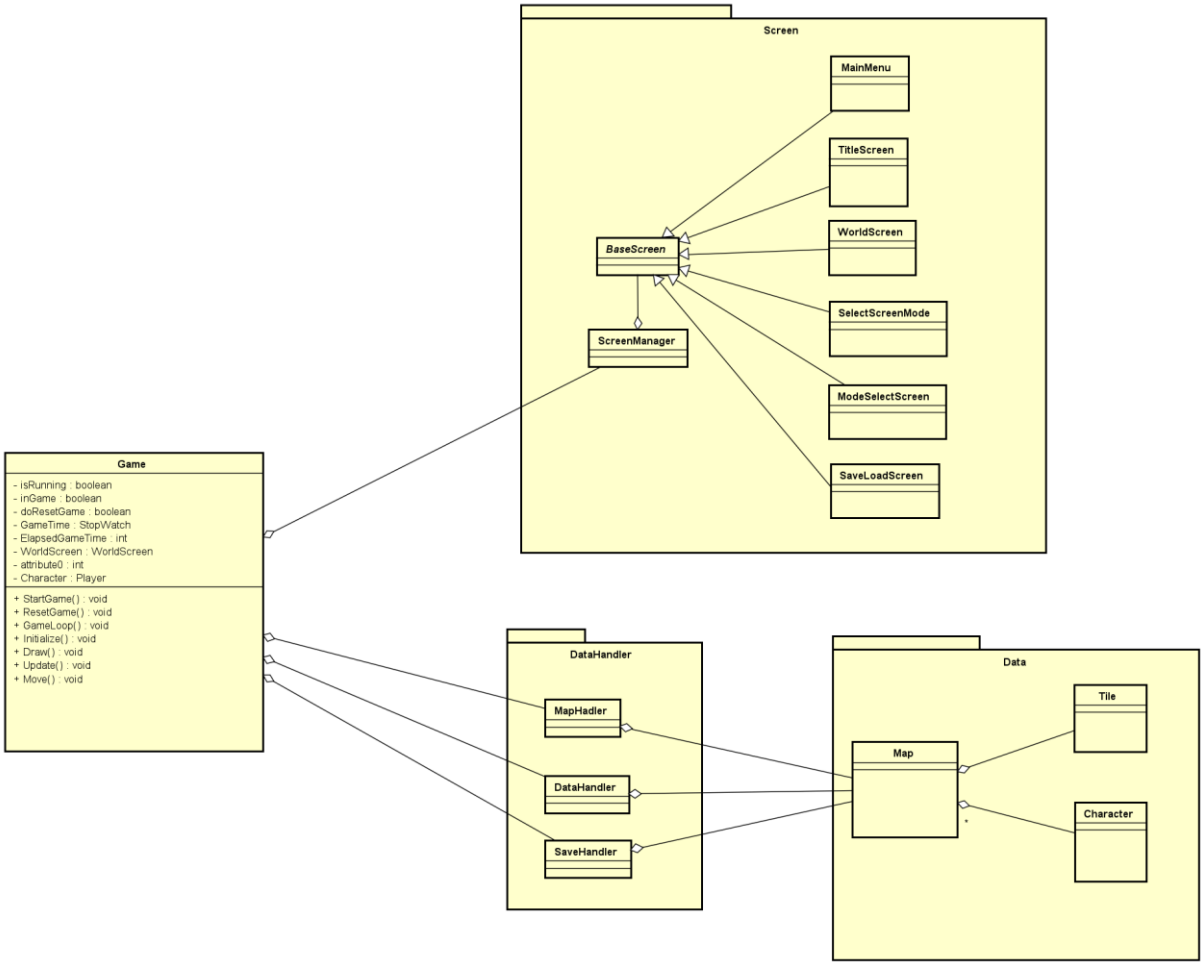
2.2. Biểu đồ lớp

2.2.1. Hệ thống các package

- Class Game: Là Class chính, thực thi quá trình chạy chương trình Game, xử lý thắng thua, thực hiện gọi các Class khác
- Package Screen: Dùng để tạo và hiển thị các giao diện trên màn hình, cũng như tạo điều kiện để người dùng có thể trải nghiệm các hình ảnh trên màn hình một cách tối ưu. Khi người dùng sử dụng thì chương trình sẽ hiển thị lại màn hình sau mỗi thao tác của người chơi theo mỗi chuyển động của nhân vật trong game.
- Package Data: Bao gồm các folder và các class, các folder sẽ chứa các file dữ liệu đầu vào (hình ảnh, âm thanh) được import và save/load để phục vụ cho việc hiển thị trong game. Bên cạnh các folder đó có các class để xử lý các đối tượng đầu vào tương ứng thể hiện cho từng đối tượng như item, character và map, …
- Package Globals: Chứa các class thể hiện những gì chung nhất của game. Những thứ dùng chung cho cả phần mềm như âm thanh, enumeration để quy định các kiểu dữ liệu tự tạo như hướng di chuyển, và công thức tính sát thương, cùng với đó là các hằng số quy định các frame ảnh và số frame ảnh được load trong 1s được sử dụng ở đây.

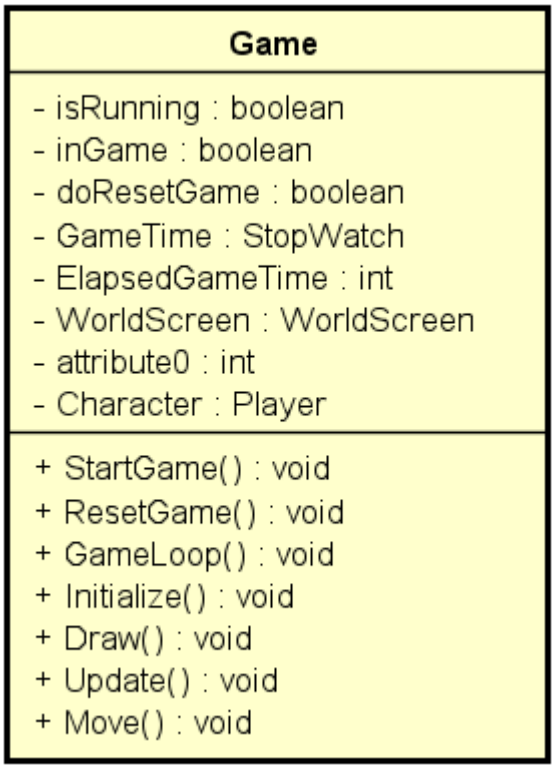
2.2.2. Biểu đồ lớp

Biểu đồ lớp tổng quát:


Screen
MainMenu
TitleScreen
WorldScreen
BaseScreen
SelectScreenMode
ScreenManager
ModeSelectScreen
SaveLoadScreen
Game
- isRunning : boolean
- inGame : boolean
- doResetGame : boolean
- GameTime : StopWatch
- ElapsedGameTime : int
- WorldScreen : WorldScreen
- attribute0 : int
- Character : Player
+ StartGame() : void
+ ResetGame() : void
+ GameLoop() : void
+ Initialize() : void
+ Draw() : void
+ Update() : void
+ Move() : void
DataHandler
MapHadler
DataHandler
SaveHandler
Data
Tile
Map
Character
*

2.3. Thiết kế chi tiết lớp

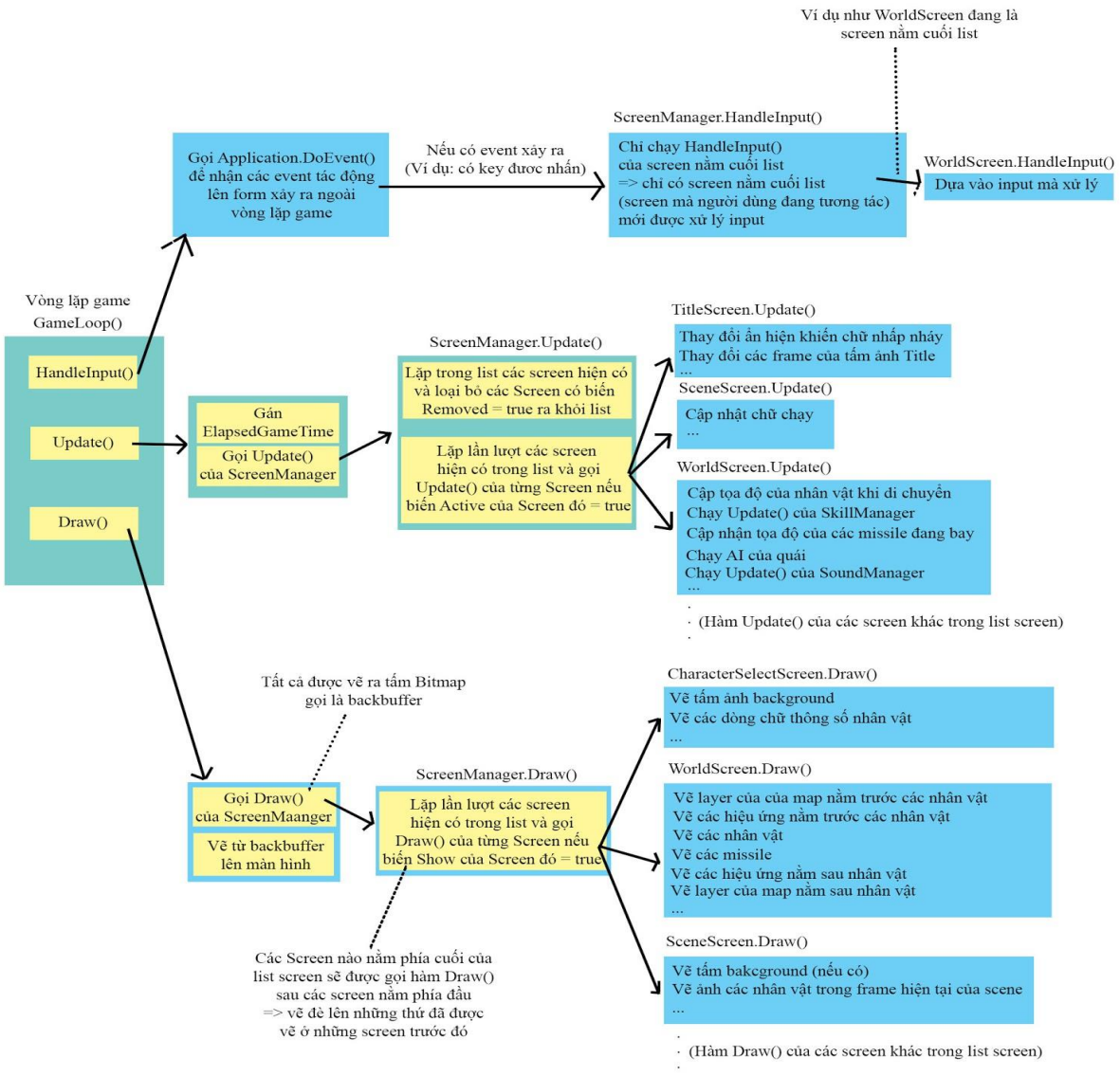
2.3.1. Class Game



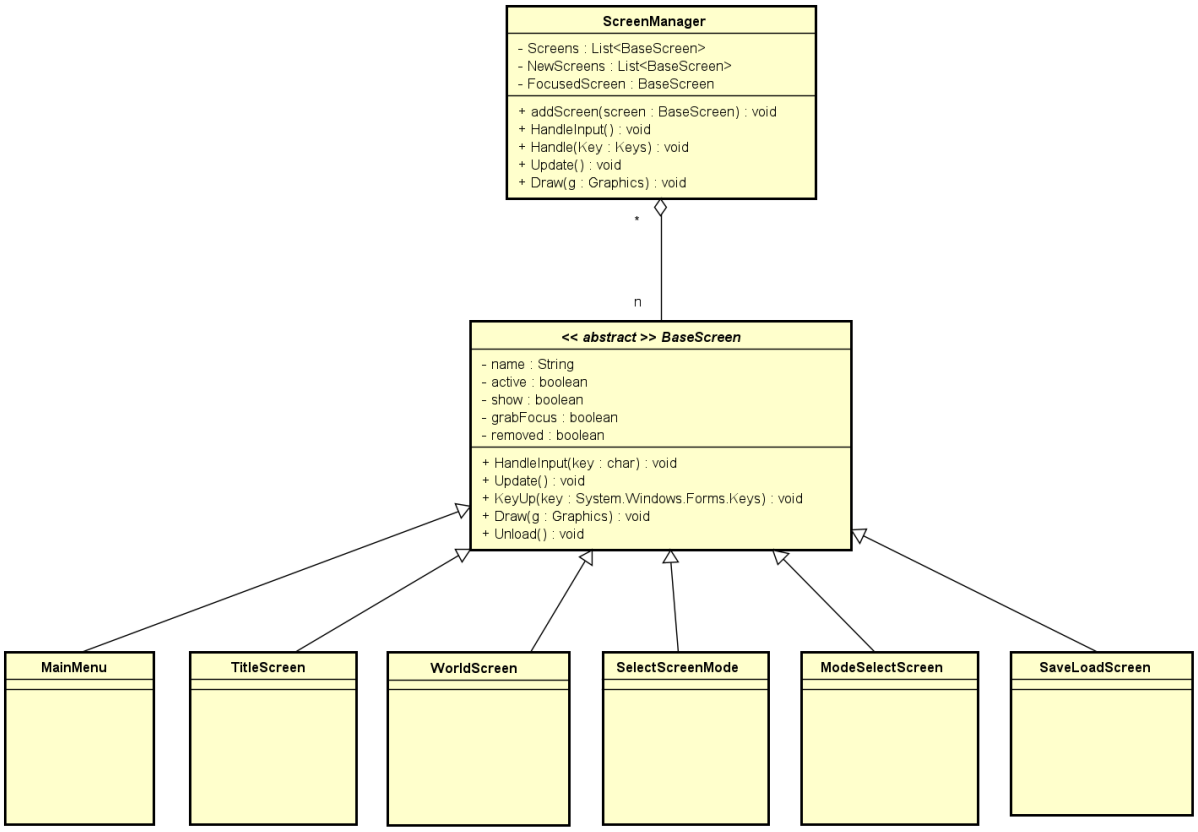
Một Game bất kỳ đơn giản chỉ gồm 1 vòng lặp (hàm GameLoop()), có một biến là isRunning ban đầu đặt là bằng true. Khi người chơi thoát game thì biến isRunning sẽ bằng false và thoát game. Trong vòng lặp GameLoop(), ta có 3 phần quan trọng chính của game, là xử lý input từ người chơi, cập nhật các thông số của đối tượng trong game, và vẽ các đối tượng ra ngoài màn hình.

- Hàm xử lý input là HandleInput() là nhận các input từ người chơi, ví dụ như là Click chuột hoặc nhấn phím trên bàn phím, rồi xử lý event cho nó.
- Hàm cập nhật thông số game là Update(), có nhiệm vụ gọi các hàm Update() của class ScreenManager, để class này xử lý và chạy hàm Update() của những object của các class Screen mà nó quản lý.
- Và hàm quan trọng nữa trong game là Draw(), thể hiện tất cả những Object đã khởi tạo trong RAM, và hiển thị ra ngoài màn hình, ví dụ như cái map có những hình ảnh này, thông số này, nhân vật được tạo lập có thông số kia,… chúng ta sẽ load tất cả và vẽ ra ngoài màn hình.

Trong quá trình chơi game, những timer là không thể thiếu. Ví dụ như timer cho chữ ẩn hiện sau 0.5 giây, timer cho frame của tấm background màn hình title thay đổi mỗi 0.2 giây,.... Do đó, bọn em tạo 1 object của class Stopwatch, và cứ đầu mỗi lần chạy của hàm Update() trong vòng lặp game, bọn em sẽ gán thời gian chạy được của Stopwatch vào biến ElapsedGameTime, sau đó reset cái Stopwatch và cho nó chạy tiếp. Từ việc này, biến ElapsedGameTime sẽ chứa thời gian trôi qua kể từ lần hàm Update() được gọi trước đó (lần cái Stopwatch được reset về 0) cho đến lần nó được gọi hiện tại. Khi cần tạo 1 timer, chúng em sẽ tạo 1 biến dùng để đếm thời gian, tạm gọi là counter, và biến chu kỳ, tạm gọi là interval. Khi hàm update của object nào đó chạy, bọn em sẽ lấy biến counter cộng với ElapsedGameTime để cho biết tổng cộng đã trôi qua bao lâu rồi. Nếu biến counter có giá trị lớn hơn interval, tức là đã đến lúc thực hiện code cần thiết sau chu kỳ chúng ta muốn, thì biến counter sẽ được set về 0, và các code sẽ được chạy sau đó.



2.3.2. Package Screens



Chúng em tạo hệ thống screen để load các ảnh, map, nhân vật, skill và hiển thị lên màn hình. Trong thư mục Screen, chúng em tạo các Class BaseScreen (lớp cha), class ScreenManger để quản lý các screen tương ứng với các phần của game. Và có 1 thư mục screens chứa các Class chi tiết hơn là MainMenu, TitleMenu, WorldScreen, … đều là lớp con kế thừa từ lớp BaseScreen.

- Lớp BaseScreen : Là một lớp abstract, định nghĩa các hàm abstract là HandleInput(), Update(), KeyUp(), Draw(), Unload() để các lớp con kế thừa có thể overriding lại. Trong đó, hàm HandleInput có chức năng nhận tất cả các thao tác của từ dùng. Hàm Update có chức năng liên tục load hình ảnh và hiển thị lên màn hình.

- Lớp ScreenManager: Quản lý thực thi, điều khiển, hiển thị tất cả các màn hình có trong game ba phần chính là HandleInput(), Update() và Draw() của các screen. Class ScreenManager sẽ kết tập từ các class khác là Mainmenu, TitleScreen, WorldScreen, ModeSelectScreen, SaveLoadScreen. Lớp chứa 2 List các đối tượng BaseScreen là screens chứa các screen hiện tại, và newScreens chứa các screen mới để chuẩn bị hiển thị trên màn hình trong vòng lặp sau. Mọi screen con dưới đây đều phải thông qua class này để có thể hiển thị lên màn hình. Hàm Draw() đơn giản lặp tất cả các screen trong List screens và vẽ lên màn hình.

Bởi vì trong 1 vòng lặp Update của game, nếu chúng ta thay đổi trực tiếp các screen trong list thì sẽ gây ảnh hưởng đến phần Update() của các screen đó trong vòng lặp. Vì vậy nên Trong lớp BaseScreen có 1 thuộc tính là removed, có tác dụng là 1 screen được hiển thị ra màn hình và chuyển đến screen tiếp theo thì biến removed sẽ được gán và false, và bị loại bỏ ra khỏi List screen để giải phóng bộ nhớ Ram. Ở vòng lặp tiếp theo, ScreenManager sẽ loại bỏ tất cả các screen nào có biến này = true ra khỏi list trước khi chạy hàm Update() của từng screen, từ đó tránh được lỗi ảnh hưởng đến game từ việc loại bỏ trực tiếp screen đó từ list khi đang trong phần chạy Update của game. Khi 1 screen đến giai đoạn kết thúc và không hiện nữa, nó sẽ gọi hàm Unload(). Tại đây, biến removed sẽ được set = true, đồng thời Dispose các component dùng trong screen đó (nếu có) để giải phóng bộ nhớ RAM, chống tình trạng memory leak.

- Lớp TitleScreen: Lớp này có ý nghĩa là hiển thị màn hình chung nhất của game khi người dùng bắt đầu Click bật trò chơi. Ở lớp này, hàm HandleInput được overriding sẽ thực hiện lấy input là 1 phím bất kỳ người dùng ấn trên bàn phím để chuyển tiếp tới màn hình tiếp theo. Hàm Update sẽ có chức năng là set vị trí của tấm ảnh (đoạn tấm ảnh chạy ra), thay đổi hiển thị liên tục dòng chữ sau một chu kỳ thời gian, khiến cho nó nhấp nháy, và thay liên tục các Frame ảnh sau một chu kỳ thời gian để tạo hiệu ứng ảnh động. Đồng thời, chúng em cũng thêm 1 phương thức cho phép load và chạy nhạc có sẵn.

- Lớp MainMenu: Đây là lớp quy định, biểu diễn các thuộc tính của màn hình Menu chọn lựa các chức năng chính của Game. Thông qua phím các mũi tên và phím Enter để chọn chức năng, hàm HandleInput sẽ nhận các input đầu vào để thay đổi SelectedIndex với các phím mũi tên, hoặc chuyển đến màn hình tiếp theo nếu là phím Enter. Ở đây chúng em tạo list các button để người chơi có thể dễ dàng tương tác bằng chuột.

- Lớp CharacterSelectScreen: có chức năng lựa chon nhân vật chính chơi trong game. Chúng em chia màn hình ra làm 4 phần hiển thị tương ứng với 4 nhân vật đã được tạo sẵn. Cơ chế tương tự như MainMenu.

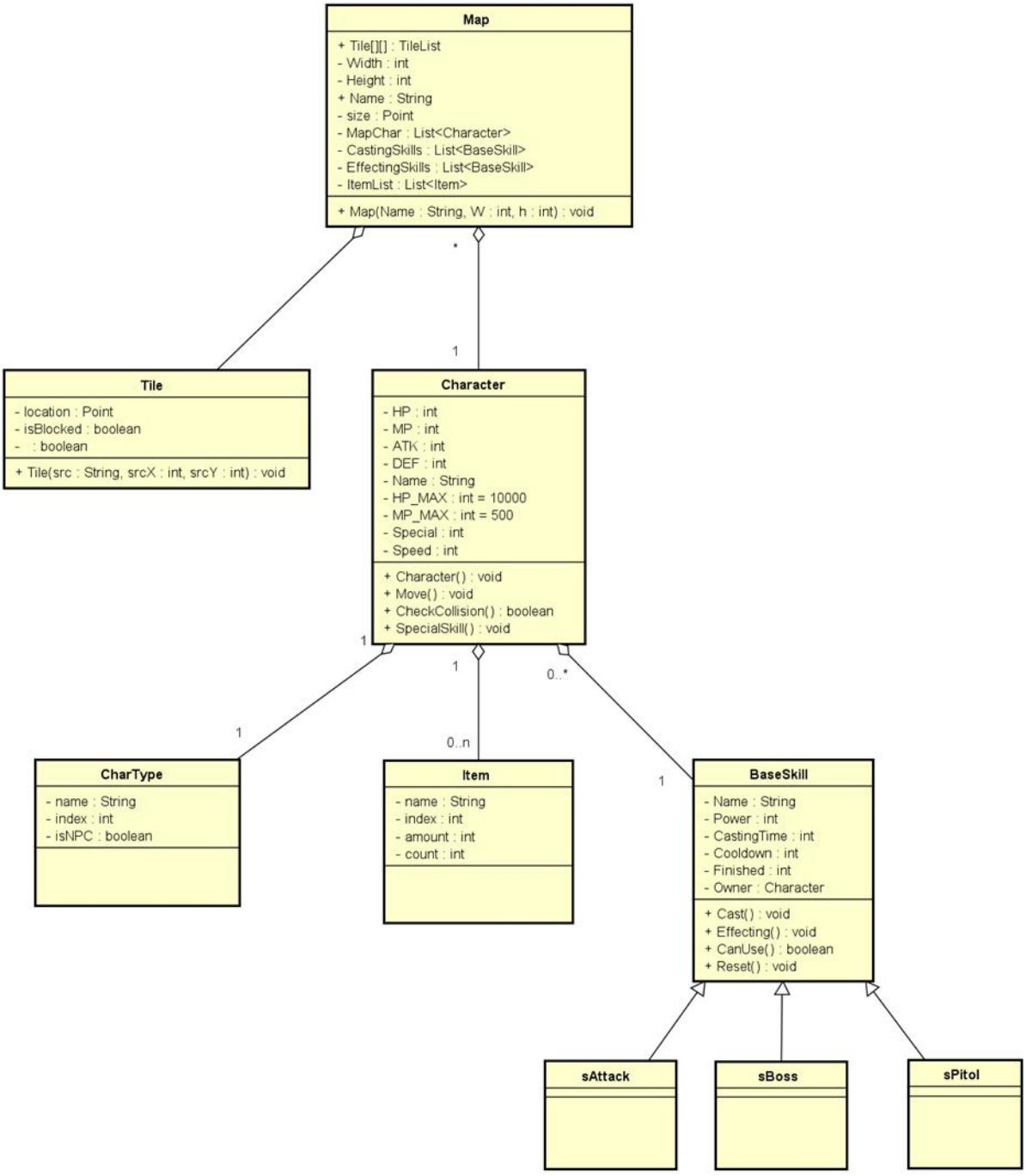
- ModeSelectScreen: Có chức năng lựa chọn mức độ khó trong game. Có 3 mức độ là easy, normal, hard. Cơ chế tương tự như MainMenu.

- Lớp WorldScreen: Thể hiện gio diện chính của game khi chơi, hiển thị toàn bộ các đối tượng như là map, các NPC, character, boss, skill, thanh máu,

… . Hàm update có nhiệm vụ liên tục cập nhật các nhân vật, skill, chuyển động các nhân vật theo thuật toán đã triển khai sẵn , là nơi chính vận hành game trong lúc chơi.

- Lớp SaveLoadScreen: Hiển thị màn hình người chơi chọn các bản saved game đã lưu trước đó slot để load, hoặc để save. Khi tạo object mới của class này, nếu chúng ta trao cho constructor parameter là “save” thì nó sẽ hoạt động như màn hình save, còn nếu constructor nhận parameter là “load” thì sẽ hoạt động như màn hình load game.

2.3.3. Package Data



- Lớp Art: là class effect (hiệu ứng) của game. Trong class chứa các thông số cho biết vị trí xuất hiện effect, hình ảnh của effect được lấy từ tấm ảnh nào, và hàm Update() được gọi mỗi vòng lặp game để cập nhật thông số của effect.

- Lớp Character: Là lớp dùng để tạo các đối tượng nhân vật trong Game. Các character trong game, kể cả các NPC, nhân vật chính, boss đều là Object của lớp này. Class này chứa 1 biến của class CharType (trình bày bên dưới), sẽ quy định kiểu nhân vật của nhân vật này. Class này còn có

các biến chứa thông số cho biết toạ độ hiện tại trong map của nhân vật, list của BaseSkill cho biết các chiêu thức nhân vật sở hữu,....

- Lớp Tile: được hiểu như là một ô hình ảnh kích thước 32px x 32px, chứa thông số của tấm ảnh nguồn và vị trí lấy từ tấm ảnh nguồn để vẽ ra nó. Class này còn chứa biến IsBlocked để cho biết tile đó có thể đi vào hay ko, và biến StepTrigger để nhận biết tile đó có xảy ra event gì khi đi vào hay không. Nếu có, biến Script trong tile đó sẽ được xử lý bởi class TriggerHandler. Hiểu đơn giản là 1 hình ảnh sẽ được chia thành nhiều ô vuông nhỏ, mỗi ô vuông có kích thước 32x32 px.

- Lớp Map: Chứa mảng hai chiều của các object class Tile, gọi là TileList. (phải nói được ý này mới đáp ứng yêu cầu đề chứ!) Index của các Tile trong mảng 2 chiều TileList sẽ là toạ độ của Tile đó nằm trong map. Ví dụ TileList[2, 3] sẽ là Tile nằm ở vị trí (2, 3) trong map. Trong class map còn có các list chứa các object nằm trong map, như list của class character để chứa các nhân vật đang ở trong map đó, list của skill chứa các skill đang được triển khai trong map đó, …

- Lớp Missile: là class của các vật bay (projectile) trong map, như viên đạn, quả bom, mũi tên,.... Class này cũng sẽ có các thông số cho biết vị trí hiện tại của vật bay đó đang ở đâu trong map, hình ảnh được vẽ từ phần nào của tấm ảnh nguồn, góc bay của vật bay,.... Một missile sẽ có một biến FromSkill cho biết nó được tạo từ skill nào, và khi va chạm nó sẽ gọi hàm MissileHit của skill đó, báo hiệu rằng vật bay đã dính vật thể hay nhân vật nào đó, từ đó skill mới xử lý các bước tiếp theo như tính damage, gây damage hay tạo các effect nổ,.... Phần kiểm tra va chạm của missile tương tự với kiểm tra collision của nhân vật, sẽ được trình bày ở phần kỹ thuật.

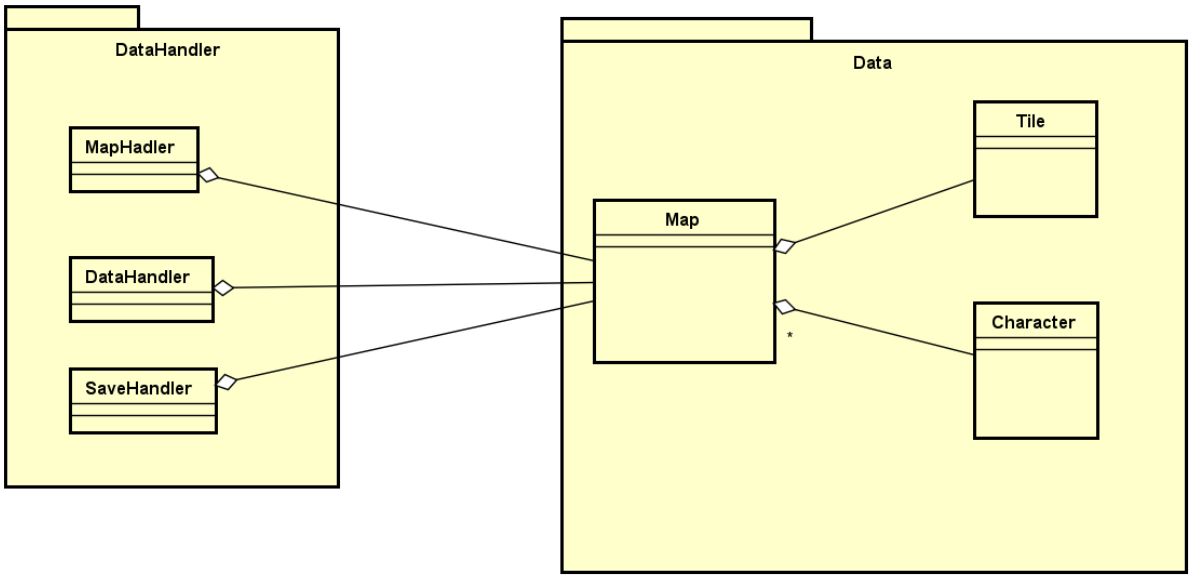
- Lớp BaseSkill: Là lớp cha của toàn bộ các skill được tạo ra, bao gồm skill của nhân vật, boss hay quái. Nói cách khác đây là lớp trừu tượng hóa để định nghĩa ra các skill. Mỗi skill sẽ có hàm Cast, chạy 1 lần khi skill được sử dụng, và hàm Casting, chạy trong hàm Update() của game, để cập nhật thông số và trạng thái của skill qua các chu kỳ.

- Lớp CharType: Là lớp định nghĩa ra các loại nhân vật. Nói dễ hiểu là class “kiểu nhân vật”. bao gồm quái, boss, các nhân vật chính, phụ cũng

như người bán các item trong game. Đây là lớp chứa các thông số quy định độ mạnh yếu của kiểu nhân vật đó. Khi một object của class Character được tạo, các thông số của nhân vật đó sẽ được tính dựa vào các thông số trong class CharType này. Ở Map Editor, chúng em có tạo một form dành riêng để tạo, chỉnh sửa các object kiểu nhân vật của class này. List kiểu nhân vật sau khi được tạo sẽ được ghi ra file lưu vào ổ cứng, khi chạy game sẽ được nạp trở lại trong phần sơ kỳ hoá? (Initialize) của game.

- Lớp Item: Lớp này định nghĩa ra các vật phẩm được sử dụng trong game, được từ việc tiêu diệt quái vật. Đây cũng là class gần như abstract, để cho các class con cụ thể hơn override hàm Use().

2.3.4. Package Handler



- Lớp DataHandler: Là lớp chứa các hàm dùng để ghi ra hoặc đọc dữ liệu của object các class từ file lưu trong ổ cứng. Các object của class scene, list của kiểu nhân vật,... sẽ được tạo trong Map Editor, và lưu vào trong ổ cứng thông qua các hàm trong class này. Tất cả đều được ghi dưới dạng file Binary. Chúng em dùng class BinaryWriter và BinaryReader có trong thư viện System.IO của C# để ghi và đọc từ file.

- Lớp MapHandler: Cơ chế giống với class DataHandler, chứa các hàm dùng để đọc và ghi ra file các object của class map. Ví dụ chúng ta tạo 1 map có tên StartMap, khi nhấn save trong Map Editor thì thông qua hàm SaveMap trong class này, các thông số của map đó sẽ được ghi ra file và lưu trong ổ cứng. Sau đó, trong khi chơi game, nếu nhân vật đi vào một tile có trigger là Load Map, TriggerHandler sẽ lấy phần giá trị của trigger đó và chạy hàm

LoadMap trong class này để đọc các giá trị từ file đã lưu, tạo object mới của class map và gán các giá trị đó vào lại, phục hồi nguyên trạng object map giống như khi tạo trong editor.

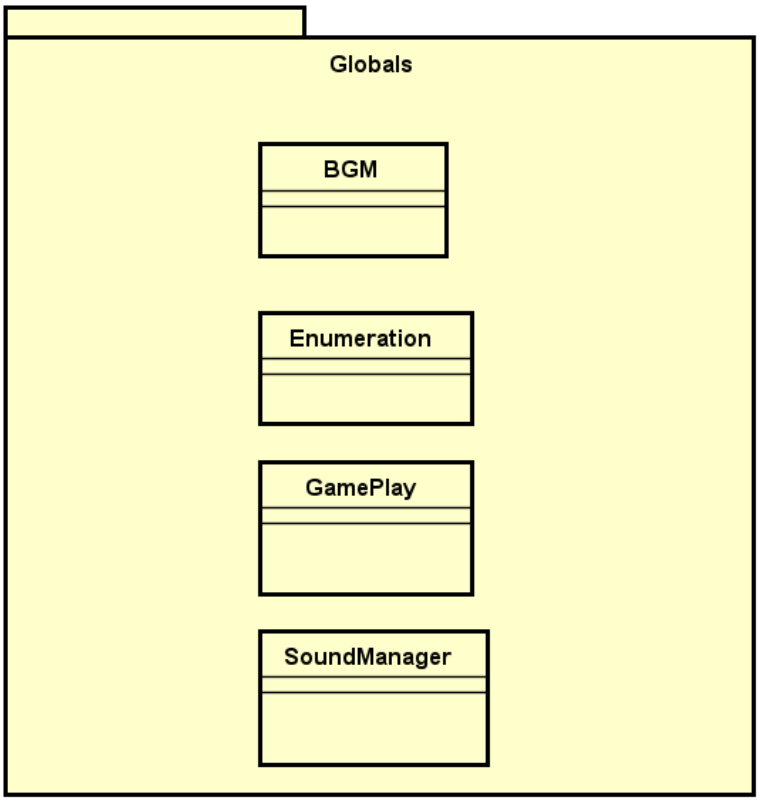
- Lớp SaveHandler: là class thực thi Serialize và Deserialize class Save. Khi người chơi save, hàm SaveGame trong class này sẽ được gọi. Lúc đó, object mới của class Save sẽ được tạo, nạp vào các giá trị cần lưu, chuẩn bị sẵn sàng, sau đó được Serialize thành file binary lưu vào ổ cứng. Tương tự khi load game, hàm LoadGame được gọi, trả về giá trị là một object của class Save được Deserialize từ file ở ổ cứng. Sau đó, các thông số của game sẽ được nạp trở lại dựa vào thông số trong object của class Save vừa được Deserialize này.

- Lớp TriggerHandler: Là class dùng để xử lý các script của các trigger nằm trong tile. 1 script sẽ gồm 3 phần:
    - Action: hành động cần thực hiện.
    - Value: giá trị (nếu có) khi thực hiện hành động.
    - Condition: điều kiện để thực hiện hành động đó.

  Ba phần này được chúng em quy định thành một chuỗi dạng string để dễ tạo và lưu trữ. Một script sẽ có dạng là:
  "Action|Value1,Value2,...|Condition1,Condition2,..."
  Khi TriggerHandler xử lý một script, nó sẽ cắt chuỗi này ra để lấy các giá trị như Action, Value1,... rồi sau đó xử lý. Điều kiện là phần được kiểm tra đàu tiên. Nếu điều kiện đáp ứng đủ, script sẽ được thực hiện.
  Các Action gồm có:
  Load Map: Chuyển sang 1 map hoàn toàn mới. Action này chúng em quy định nó cần có 3 value. Value thứ nhất sẽ là tên map cần load, hai value sau sẽ là toạ độ x và y của nhân vật khi chuyển sang map đó. Ví dụ một script có dạng: Load "Map|C9,12,23|" khi được xử lý sẽ gọi hàm LoadMap của class MapHandler, trao value thứ 1 là tên map C9cho để hàm LoadMap load file C9.map đã được tạo và lưu từ Map Editor lên, sau đó set vị trí hiện tại của nhân vật bằng value thứ 2 và thứ 3 của script.
  Gọi hàm LoadScene của DataHandler để load scene cần thiết lên từ file. Hành động này chỉ cần 1 value là tên file scene. Sau khi scene được load lên, một object mới của class SceneScreen được tạo ra và được trao cho scene vừa load để hiển thị. Object mới của class SceneScreen sau đó được thêm vào list screen để có thể được chạy và hiện ra màn hình.
  Play Sound, Play BGM, Stop BGM, Resume BGM: Quy định việc phát, dừng và tiếp tục các file âm thanh trong chương trình.
  - Để kiểm tra điều kiện, chúng em tạo một biến Flag Sở Globals, là một HashMap (Dictionary) của string và string. Một điều kiện sẽ có dạng "Key=Giá trị cần đáp ứng". Hàm CheckCondition sẽ lấy Key được cắt ra từ script, đem đi tra và lấy Value của key đó từ Flags. Nếu giá trị lấy từ

Flags đúng bằng với giá trị cần đáp ứng thì hàm sẽ trả về giá trị true, ngược lại thì là false. Giá trị của các Flag trong Flags sẽ được thay đổi tại từng event xảy ra trong game.

2.3.5. Package Globals



- Lớp BGM: Lớp quy định việc phát các trình âm thanh nền. Chúng em sử dụng Interface WindowMediaPlayer có sẵn từ thư viện WMPLib của Microsoft để phát các file nhạc MP3.
-
- Lớp Enumeration: Định nghĩa các kiểu dữ liệu đặc thù do lập trình viên tự quy định, bao gồm các hướng của nhân vật (dir), team của nhân vật (enemy hay player), các chế độ chơi (Hard Mode - Easy, Normal, Hard), Status của nhân vật (Poisoned,Paralyzed, Burned) và SkillType, quy định xem một skill khi sử dụng sẽ tính damage theo thông số Attack hay SpecialAttack của nhân vật.
- Lớp GamePlay: Quy định việc người chơi sẽ chơi game như nào? Ví dụ các công thức tính sát thương, kiểm tra chướng ngại, hàm gây damage và sau đó kiểm tra một nhân vật đã chết hay chưa,....
- Lớp Globals: Nơi khởi tạo các giá trị để dùng xuyên suốt trong Project. Tất cả các biến trong class này đều là static và public để có thể truy cập từ bất cứ nơi nào của project.

- Lớp Settings: Quy định các giá trị mà người chơi thiết lập, như độ lớn nhỏ khi phát nhạc, phát âm thanh, có hiện cây máu hay không, tốc độ chữ chạy của scene,... Vì không đủ thời gian nên chúng em chưa tạo xong OptionScreen để người chơi có thể tuỳ chỉnh các thông số này.

- Lớp Sound Manager: Là class quản lý, phát và chạy các file âm thanh dạng WAV. Chúng em dùng class MediaPlayer có sẵn trong C# để phát các file âm thanh. Khi cần phát một file âm thanh, một object mới của class MediaPlayer sẽ được tạo để chạy file âm thanh cần thiết. Nếu chỉ dùng 1 object của class này để phát nhiều file âm thanh thì khi file âm thanh sau được phát, nó sẽ dừng file âm thanh trước, cho nên việc tạo mới object của class này mỗi lần phát file âm thanh là cần thiết. Từ việc này, chúng em cần có một hàm Update() chạy trong hàm Update() của mỗi vòng lặp game. Hàm này có tác dụng kiểm tra trong list các object của class MediaPlayer được phát, cái nào đã chạy xong thì chạy hàm Close() của nó để đóng nó lại và dispose nó, giải phóng bộ nhớ, từ đó chống được tình trạng memory leak.

- Lớp Texture: Là class chứa tất cả các biến chứa file hình ảnh của game. Khi game được khởi chạy, tất cả file hình ảnh cần thiết sẽ được nạp vào các biến trong class này (nạp vào bộ nhớ RAM) để được sử dụng bởi hàm DrawImage().

2.3.6. Package Form

Là package do Windows hỗ trợ, được C# định nghĩa sẵn. Gồm Form chính và các form dùng trong Map Editor, như Character Palette dùng để tạo và chỉnh sửa list kiểu nhân vật, Scene Palette dùng để tạo và chỉnh sửa các scene, TilePalette dùng để tạo và chỉnh sửa các Tile của map.

CHƯƠNG 3. CÔNG NGHỆ VÀ THUẬT TOÁN SỬ DỤNG

3.1. Ngôn ngữ lập trình và các nền tảng công nghệ

Để lập trình một phần mềm hay hệ thống phần mềm tuyệt nhiên không thể không kể đến ngôn ngữ lập trình. Với một game 2D nhập vai cho desktop Windows như BK's Knights của nhóm, chúng em đã thống nhất và đi đến quyết định lựa chọn sử dụng **ngôn ngữ lập trình C# (Csharp)** - một trong những ngôn ngữ lập trình hướng đối tượng mạnh mẽ và được tổ chức chặt chẽ với các kiểu dữ liệu, cấu trúc dữ liệu được định nghĩa sẵn, các từ khóa cũng không quá xa lạ nên dễ dàng có thể tiếp cận cũng như một hệ thống thư viện mã nguồn mở đồ sộ hỗ trợ rất nhiều cho lập trình viên trong quá trình xây dựng phần mềm. Với những đặc điểm tối ưu của C# cùng với sự đồng thuận của cả nhóm khi mà cả 4 thành viên trong nhóm đều có thể sử dụng ngôn ngữ lập trình này tối thiểu là ở mức cơ bản thì ngôn ngữ C# là lựa chọn tối ưu nhất cho giải pháp phần mềm được đặt ra. Và phục vụ cho phần mềm này nhóm chúng em đã sử dụng công cụ lập trình **Visual Studio 2019**.

Với ngôn ngữ C#, để lập trình game BK's Knights theo ý tưởng đề ra là giao diện hiển thị 2D sử dụng trên hệ thống desktop hệ điều hành Windows, chúng em đã sử dụng nền tảng **Windows Forms Application**. Chắc hẳn với những ai đã quen sử dụng C# thì không còn lạ lẫm gì với nền tảng này khi đây là một platform được sử dụng rất phổ biến để lập trình các phần mềm cơ bản cho máy tính Windows một cách đơn giản và hiệu quả. Đây là một nền tảng thuộc **.NET Framework** của **Microsoft**. Với việc sử dụng các tính năng đã được hỗ trợ để dễ dàng thao tác và lập trình với giao diện mong muốn, bên cạnh nền tảng **WinForms** chúng ta có thể sử dụng linh hoạt các bộ thư viện mã nguồn mở dồi dào từ Microsoft. Trong phần mềm của chúng em các thư viện được sử dụng đến bao gồm:

System, System.Collections.Generic, System.Linq, System.Text, System.Threading.Tasks, System.Drawing, System.IO, System.IO.Compression, System.Runtime.Serialization.Formatters.Binary, System.Windows.Input, System.Windows.Forms, System.Data, System.ComponentModel, WMPLib, System.Diagnostics, System.Windows.Media.

Với các bộ thư viện kể trên chúng ta có thể sử dụng rất nhiều những tính năng, phương thức đã được xây dựng sẵn và nhờ đó công việc lập trình phần mềm trở nên đơn giản hơn rất nhiều. Đáng chú ý như các hàm DrawImage() của class Graphics trong thư viện System.Drawing; hay các trình đọc và ghi ra file của thư viện System.IO, phát âm thanh hay video,... nhờ đó chúng ta đi đến được một hệ thống phần mềm như mong muốn.

3.2. Kỹ thuật, cấu trúc dữ liệu và thuật toán

Để thực thi một giải pháp phần mềm cho một ý tưởng thì không thể thiếu các kỹ thuật, cấu trúc dữ liệu và các giải thuật được áp dụng. Về cơ bản một phần mềm như BK's Knights trong quá trình lập trình cần sử dụng rất nhiều những kỹ thuật lập trình và nổi bật hơn cả chúng ta có thể để ý đến là kỹ thuật hướng đối

tượng và kỹ thuật xử lý dữ liệu đầu vào trong dự án này. Đầu tiên nói về hướng đối tượng, để có một giao diện và các tính năng hoàn chỉnh như mong đợi trước hết chúng ta cần phân tích và thiết kế để hoạch định những tính năng cần làm và những vấn đề cần giải quyết. Qua việc học môn lập trình hướng đối tượng chúng ta có được tư duy phân tích thiết kế cơ bản cho hệ thống phần mềm với việc chia nhỏ các nhóm tính năng, công việc thành các package, class để vừa dễ dàng phân công trong một nhóm nhiều người làm - tăng năng suất làm việc, lại vừa có thể tổ chức kiểm soát, quản lý và bảo trì mã nguồn. Với việc sử dụng một ngôn ngữ thuần hướng đối tượng như C#, việc tổ chức thực thi lập trình cho phần mềm của chúng em giảm bớt được rất nhiều khó khăn, vướng mắc. Nhờ lập trình hướng đối tượng với ngôn ngữ C# chúng ta thể hiện được các đối tượng, các ý tưởng vào mã nguồn và xử lý để đưa chúng ra thành giao diện thực tế trên máy tính một cách dễ dàng. Các package và class cụ thể cũng như việc vận dụng kỹ thuật hướng đối tượng ra sao đã được trình bày ở chương 2, bao gồm chủ yếu là Encapsulation (tính đóng gói của 1 class), Inheritance (tính kế thừa của một class), Overriding (tái định nghĩa hàm của class), và Polymorphism (tính đa hình của một đối tượng).

Bên cạnh kỹ thuật hướng đối tượng chúng ta có thể chú ý đến một kỹ thuật mà theo chúng em là khá tối ưu được vận dụng trong phần mềm của chúng em đó là kỹ thuật xử lý dữ liệu đầu vào. Với các kiểu dữ liệu cơ bản thì không có gì đặc biệt, song với kiểu dữ liệu là hình ảnh, để tạo nên giao diện hình ảnh với bản đồ (map) và các nhân vật trong game sao cho các nhân vật có thể chuyển động và tạo ra các tương tác vật lý được thể hiện. Về ý tưởng sơ bộ, chúng ta sẽ vẽ liên tục hình ảnh của các đối tượng (object) lên trên màn hình để tạo hiệu ứng động, giống như nguyên lý hoạt động của một video. Hơn nữa, mỗi lần vẽ, frame của nhân vật (vị trí lấy từ tấm ảnh nguồn để vẽ) sẽ thay đổi, tạo ra hiện ứng nhân vật di chuyển.

Và một kỹ thuật không thể thiếu nữa trong một game, đó là đưa hệ tọa độ vào để tính toán. Dựa vào tọa độ, chúng ta có thể kiểm tra được va chạm giữa các nhân vật và vật thể, biết được nhân vật đang ở vị trí nào trong bản đồ, vẽ tất cả các đối tượng lên màn hình đúng với vị trí của nó,....

Một khía cạnh quan trọng khác không thể không kể đến là các cấu trúc dữ liệu và giải thuật. Nhìn chung trong toàn bộ mã nguồn của phần mềm các cấu trúc dữ liệu và thuật toán cơ bản lớn nhỏ mà các lập trình viên đã quen mặt được sử dụng rất nhiều như việc sử dụng các biến, các cấu trúc lập for, foreach, cấu trúc rẽ nhánh if/else,... trong đó nổi bật lên là cấu trúc ArrayList (List trong C#) và HashMap (Dictionary trong C#) để khởi tạo cho chuỗi các đối tượng được xây dựng và mô tả trong game như Characters, Maps, Skills,.... Đó là những cấu trúc dữ liệu mà chúng ta đã được tìm hiểu trong lập trình tổng quát các ngôn ngữ lập trình bậc cao thông dụng. Còn về phía giải thuật thì nổi trội lên là thuật toán CheckCollision và thuật toán tìm đường A Star pathfinding.

Về việc xây dựng ý tưởng và vận dụng thuật toán CheckCollision - mục tiêu là để kiểm tra các va chạm vật lý giữa các thực thể được mô tả trong game. Việc va chạm này là muốn thể hiện một cách thực tế nhất có thể rằng nhân vật, cây cối, tường,... là những thứ có khả năng va chạm vật lý, có thể ngăn cản nhau trong các bước di chuyển. Về ý tưởng thì chỉ đơn giản là chúng ta sử dụng tọa độ hiện tại trong map và kích thước của nhân vật để tính ra được một hitbox có dạng dữ liệu là Rectangle của nhân vật đó. Sau đó, với mỗi tile không thể đi sang được hoặc với mỗi nhân vật khác trong map, chúng ta cũng dựa vào tọa độ và kích thước để tính ra hitbox của chúng. Sau đó, nhờ vào hàm IntersectWith() của Rectangle mà chúng ta có thể biết được hai Rectangle có giao nhau hay không. Nếu có tức là hai vật thể đó collide (đè?) lên nhau, từ đó có thể ngăn chặn không cho nhân vật đi vào những khu vực không thể đi được, hoặc đi xuyên qua các nhân vật khác.

Tiếp đến là thuật toán tìm đường A* (A Star pathfinding) mục tiêu là chúng ta sẽ cài đặt thuật toán này cho các nhân vật qua phần AI. Mục tiêu là các quái và boss của hệ thống được thể hiện sẽ có AI để tự tìm được đường đi ngắn nhất để tìm đến nhân vật của người chơi và tấn công người chơi mà không bị đi vào ngõ cụt có chướng ngại vật rồi bị mắc kẹt ở đó. Việc xử lý cho việc tìm đường này qua giải thuật với các tile trong game (các tile có tọa độ tương ứng) và các hướng lên, xuống, trái, phải được xét. Mô tả về ý tưởng giải thuật như sau: Một tile trong thuật toán sẽ có 3 giá trị, G, H, F và một tile gọi là parent, là tile mà từ nó tính sang tile hiện tại. Giá trị G là khoảng cách từ tile đang xét đến tile xuất phát, giá trị H là khoảng cách từ title đang xét đến tile cần đến, và giá trị F là tổng hai giá trị G và H này. F = G + H.

- Tile sẽ được chia ra làm 3 loại, OpenedTiles là những tile đã được tính các giá trị, ClosedTiles là những tile đã được xét hoặc không thể đi đến (bị chặn), và UnvisitedTiles là những tile chưa được tính giá trị hay xét đến.

- Bắt đầu từ tile xuất phát, chúng ta sẽ bắt đầu xét nó. Khi xét 1 tile, chúng ta sẽ xem các tile xung quanh nó. Nếu một trong cách tile xung quanh là ClosedTile, chúng ta sẽ bỏ qua nó, là UnvisitedTile thì chúng ta sẽ tính các giá trị của nó, đặt parent của nó là tile đang xét, rồi thêm vào danh sách OpenedTiles, còn nếu là Opened Tile thì chúng ta sẽ xét xem đi từ tile hiện tại đãng xét sang nó thì giá trị G có ngắn hơn giá trị G hiện có của nó không. Nếu ngắn hơn, chúng ta sẽ tính lại các giá trị G, H và F của tile đó, đặt parent làm tile đang xét, còn ngược lại thì giữ nguyên.

- Sau khi xét xong một tile, chúng ta đánh dấu nó thành ClosedTile, sau đó kiểm tra xem trong tất cả các OpenedTiles, tile nào có giá trị F nhỏ nhất sẽ được xét tiếp theo. Nếu có nhiều tile cùng giá trị nhỏ nhất F, chúng ta sẽ

lấy tile nào có giá trị H nhỏ nhất để xét. Nếu có nhiều tile có giá trị F và H đều nhỏ nhất và bằng nhau thì chúng ta có thể lấy tuỳ ý một trong số đó để xét.

- Lặp lại quá trình trên cho đến khi tile xét là tile cần đến. Lúc đó, từ tile cần đến, chúng ta lần ngược về parent của những tile trước đó, thì sẽ có được con đường đi từ điểm xuất phát cho đến điểm đích ngắn nhất mà không vào các ngõ cụt hay mắc kẹt giữa các chướng ngại vật.

CHƯƠNG 4. XÂY DỰNG CHƯƠNG TRÌNH MINH HỌA

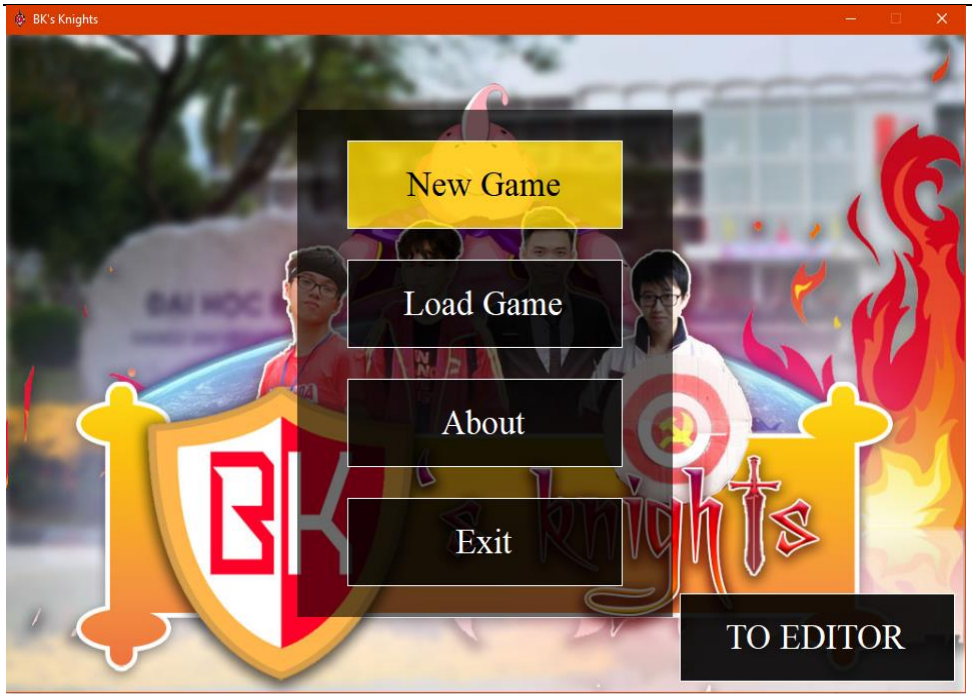
4.1. Kết quả chương trình minh họa

- Kết quả đạt được:
  - Đạt được kỹ năng phân tích, thiết kế, vẽ các sơ đồ miêu tả đề tài
  - Đạt được kỹ năng làm việc nhóm
  - Phát triển tư duy lập trình hướng đối tượng và lập trình đồ họa bằng ngôn ngữ C#
  - Phát triển các kỹ năng thiết kế các đồ họa cho game
- Những chức năng chính
  - Thể hiện được bản đồ và các nhân vật cùng các skill tương ứng
  - Tạo được bản đồ, các kiểu nhân vật, các scene thông qua Map Editor.
  - Triển khai thuật toán A* giúp các nhân vật tự tìm được đường đi ngắn nhất để giao tranh lẫn nhau
  - Chức năng chọn các nhân vật tương ứng
  - Game chia thành nhiều cấp độ khó
  - Chức năng save tình trạng chơi hiện tại và load các file đã được người chơi lưu sẵn.

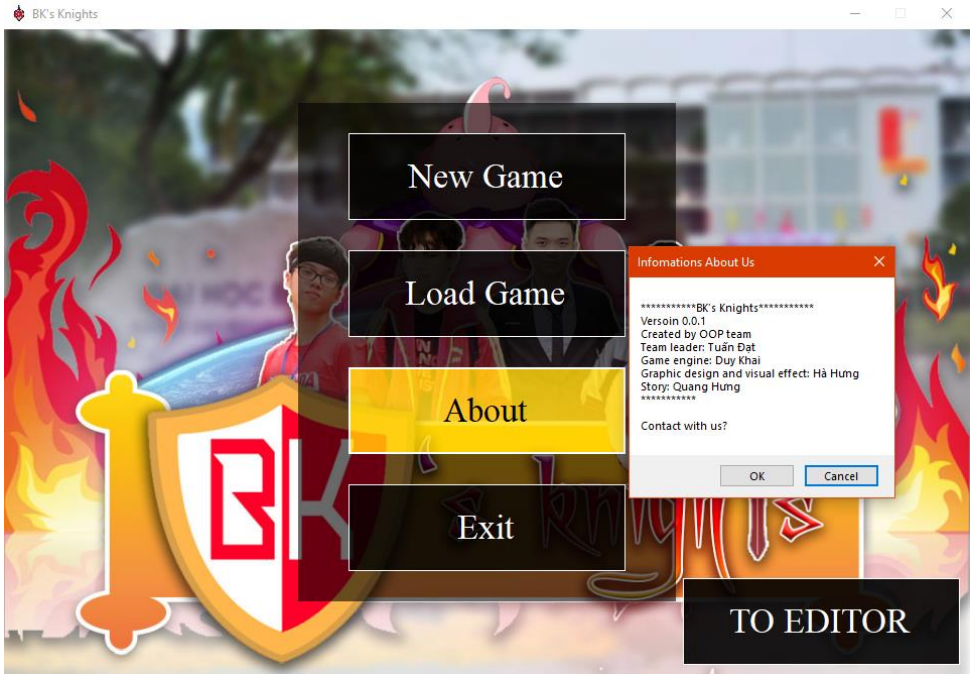
4.2. Giao diện chương trình

- Giao diện khi chạy chương trình:

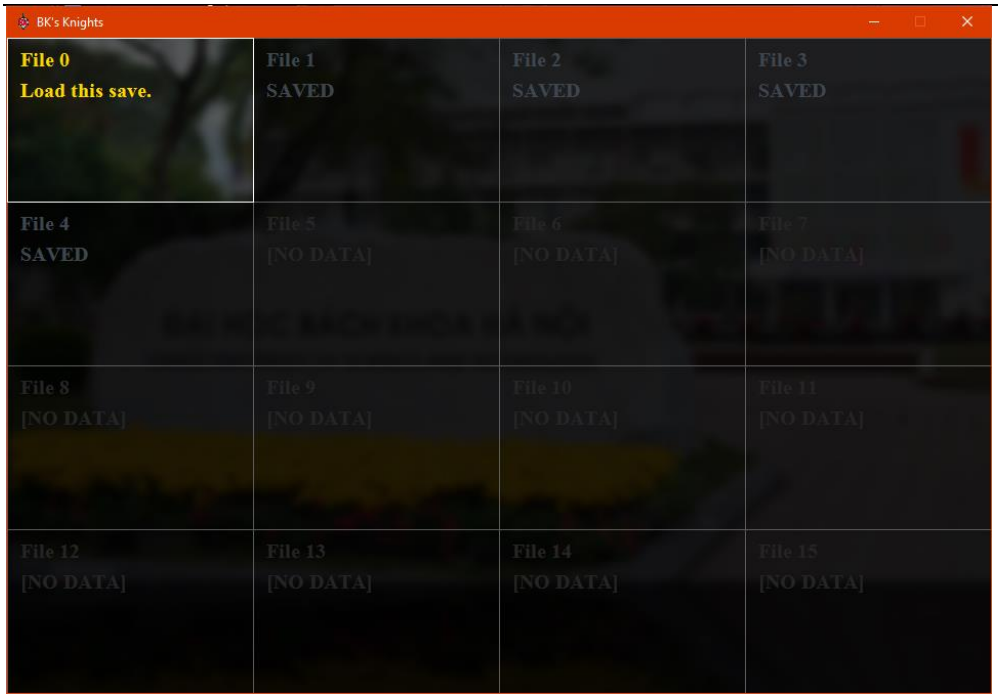
- Giao diện Menu:



- Giao diện khi bấm vào "About":

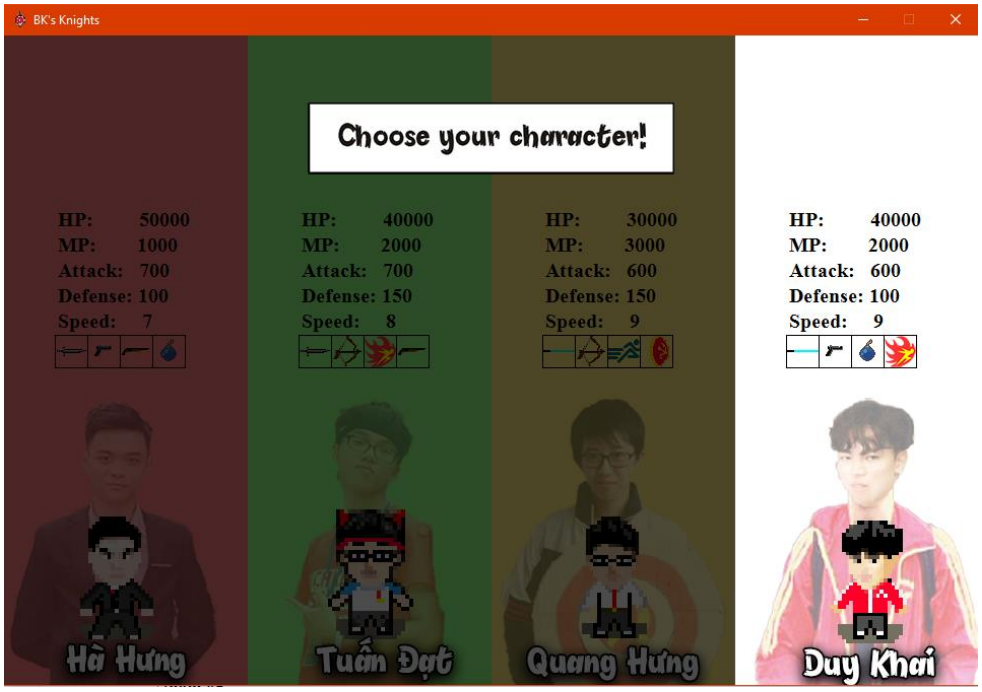


- Giao diện khi bấm vào "Load Game":
  Cửa sổ sẽ hiện lên các file mà người chơi đã lưu lại trong quá trình chơi game.



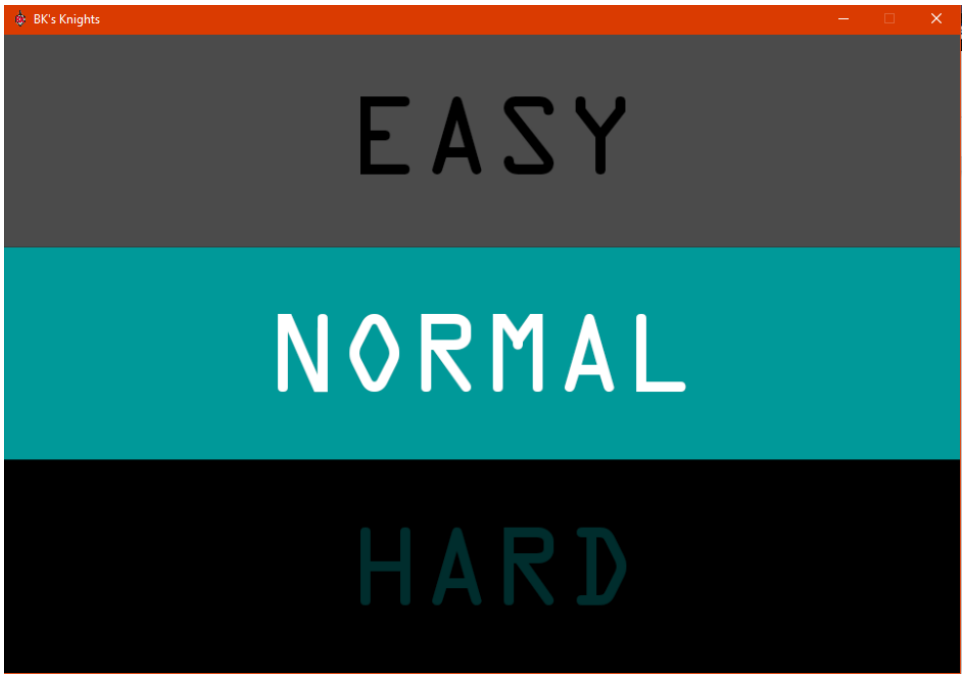
Khi bấm "Enter" vào một file bất kỳ, cửa sổ sẽ chuyển đến cảnh mà người chơi đã lưu lại trong khi chơi game.

- Giao diện khi chọn "New Game":

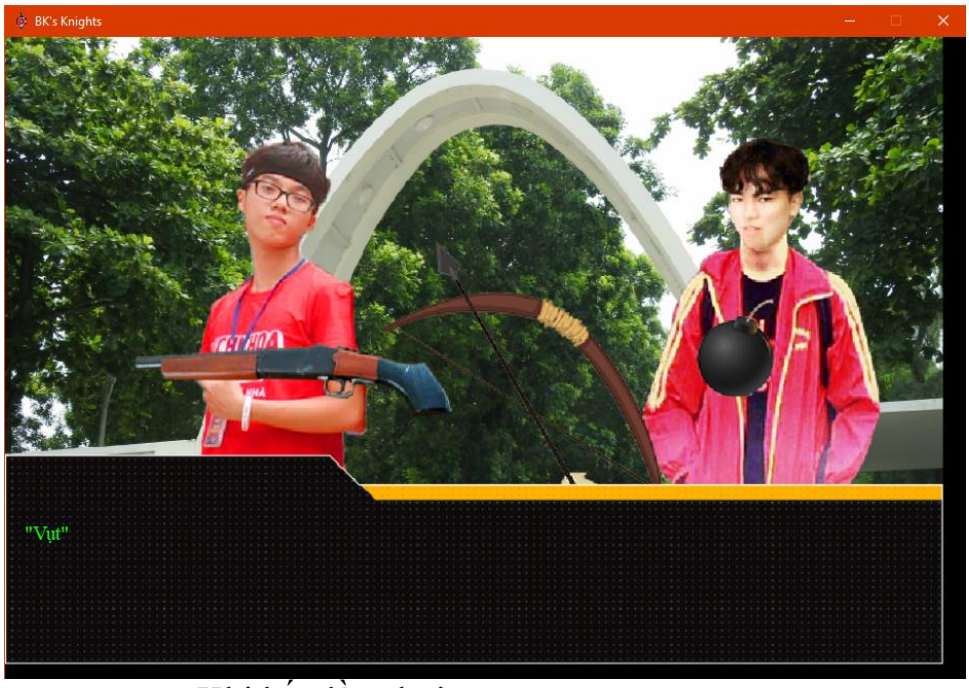


Dùng các phím mũi tên trái phải để chọn nhân vật, sau đó bấm "Enter" hoặc thanh "Spacebar"

- Giao diện chọn mức độ cho game (dễ, trung bình, khó):



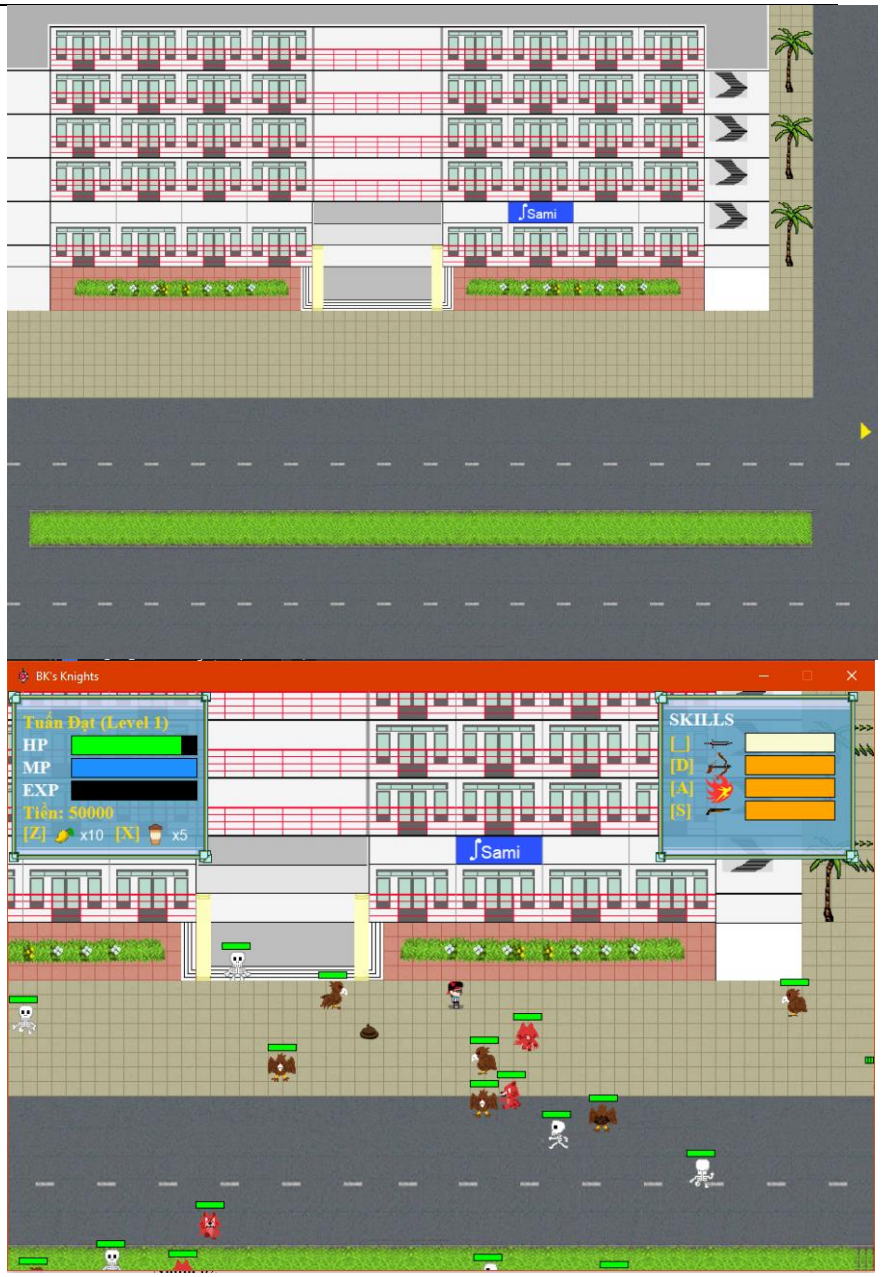
- Khi chọn xong mức chơi, story game sẽ hiện lên. Bấm phím "Ctrl" để tua nhanh qua:



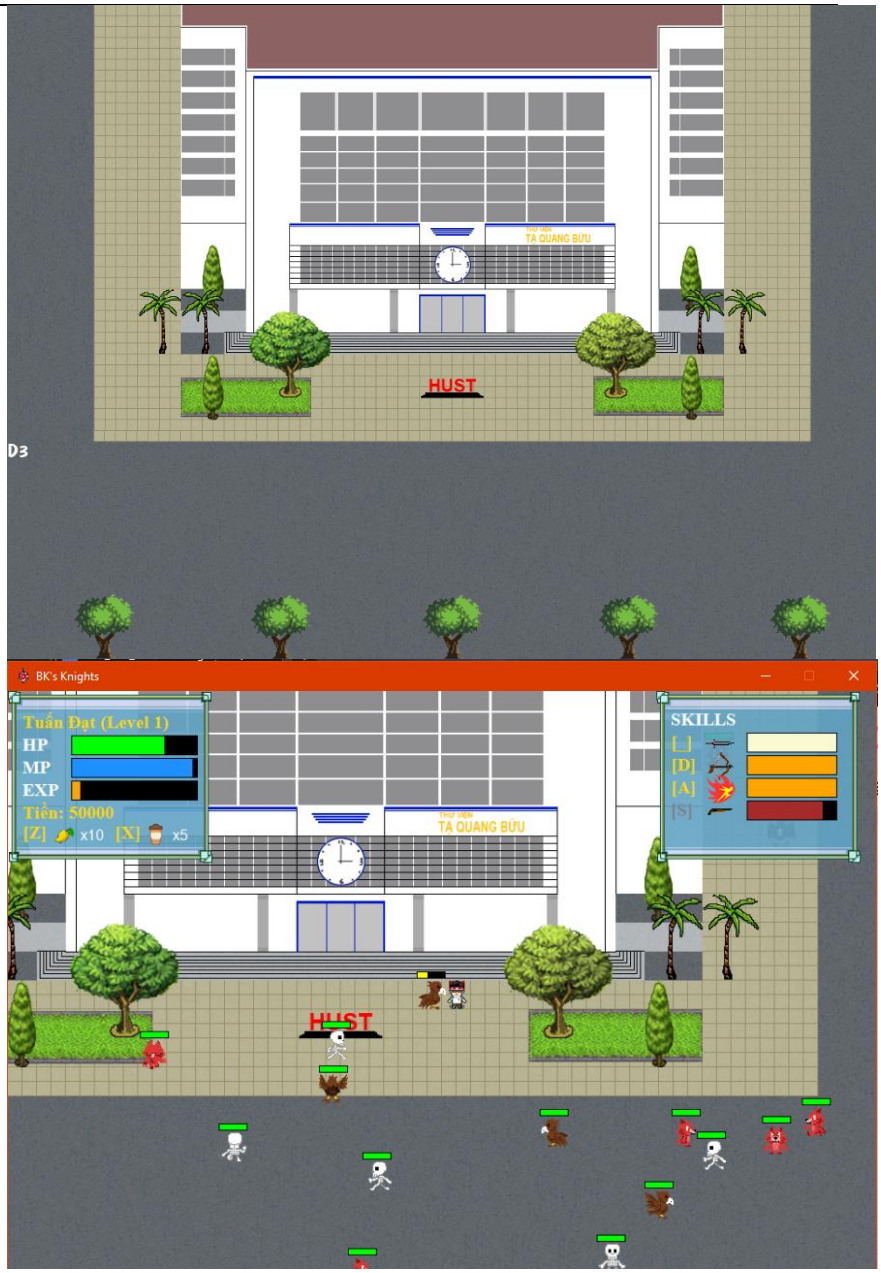
- Khi bắt đầu chơi game:
  - Game gồm có 4 stage, 5 map bao gồm:

- Start map: Khu vực cổng Parabol, dùng để di chuyển tới 4 stage còn lại trong ga

- D3: Sau khi người chơi di chuyển đến ô có ghi chữ D3, game sẽ chuyển nhân vật đến map tên D3:

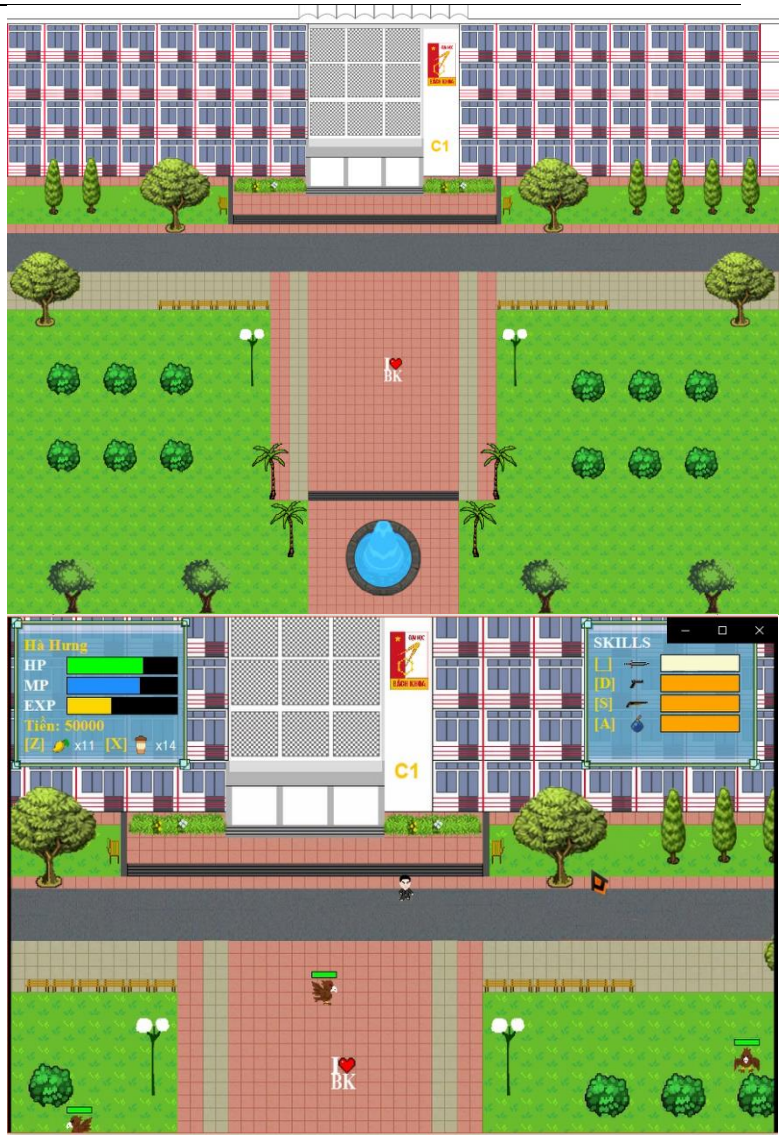


- Thư viện Tạ Quang Bửu: Khi người chơi đang ở D3, di chuyển sang phải đến cuối map D3 sẽ được chuyển sang map này. Khi người chơi ở Start map, di chuyển đến ô TBQ Lib để di chuyển tới map này:



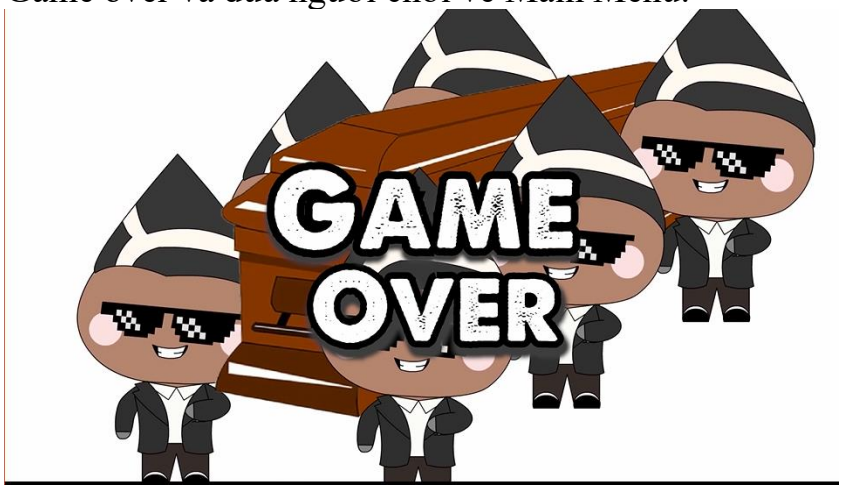
- C9: Ở Start map, người chơi di chuyển đến ô có chữ C9 sẽ được chuyển tới map này:

- C1: Để đến được map này, người chơi phải dành đánh bại được boss phụ ở 3 map D3, C9 và thư viện để mở khoá:



- Boss: Sau khi đánh bại các mini boss, boss cuối sẽ xuất hiện. Sau khi chiến thắng boss cuối, video credit hiện lên, trò chơi sẽ kết thúc:

- Game Over: Khi thua cuộc, màn hình sẽ hiển thị video Game over và đưa người chơi về Main Menu:



- Nhân vật trong game:

- Được vẽ theo phong cách pixel art, kích thước từ 32x32 pixel đến 64x64 pixel (boss), mô phỏng các chuyển động cơ bản

- Nhân vật được chia làm 3 loại: Players (4 nhân vật người chơi sẽ điều khiển), Enemy (Các boss) và Nhân vật phụ (sinh viên chạy quanh trường).

4.3. Kiểm thử các chức năng đã thực hiện

Các chức năng đã xây dựng trong chương trình:

- Save Game and Load Game
- Di chuyển và phát hiện va chạm
- Hệ thống skill, HP và Mana của người chơi và máy
- Game Over và Credit

4.3.1. Kiểm thử cho chức năng 1

Chức năng: Save Game and Load Game

Kết quả kiểm thử: Game không tự lưu tiến trình

- Khi người chơi thoát bất ngờ, game sẽ không lưu tiến trình lại, khi quay lại game, chức năng Load Game không lưu được file nào
- Muốn lưu tiến trình chơi, người chơi cần phải bấm nút F4, và chọn file để lưu tiến trình chơi.

4.3.2. Kiểm thử cho chức năng 2

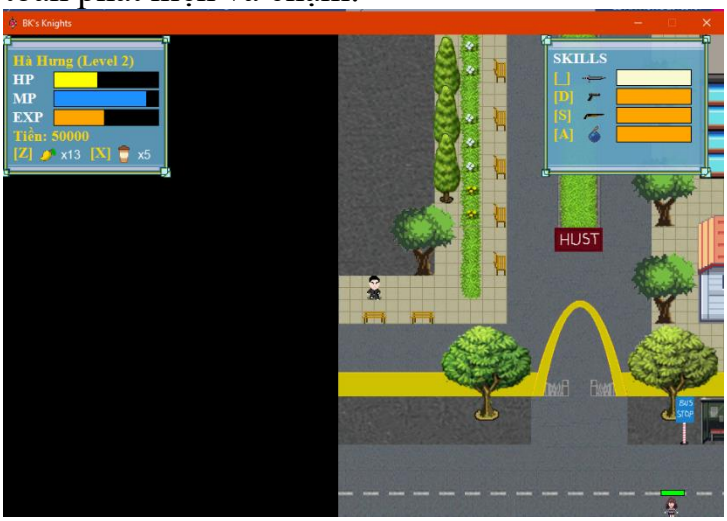
Chức năng: Di chuyển và phát hiện va chạm

Kết quả kiểm thử:

- Việc di chuyển của nhân vật khá mượt mà, không mắc lỗi không nhận bàn phím.
- Tuy nhiên, khi chạy phần mềm gõ tiếng Việt như Unikey bằng quyền adminstrator, chương trình sẽ không nhận bàn phím.

- Vì thuật toán phát hiện va chạm xử lý theo ô vuông, nên các vật thể đặt trong lòng ô vuông sẽ có các khoảng trống xung quanh, người chơi sẽ không thể di chuyển vào các khoảng trống ấy được.

Ví dụ: Ở hình vẽ, người chơi sẽ không thể di chuyển đến phần không gian sát mép trên của ghế đá được, vì ghế đá nằm ở trong ô vuông chưa thuật toán phát hiện va chạm.



4.3.3. Kiểm thử cho chức năng 3

Chức năng: Hệ thống skill, HP và Mana của người chơi và máy

Kết quả kiểm thử:

- Hệ thống skill hoạt động ổn định. Khi người chơi hết mana thì sẽ không thể dùng skill được nữa.
- Skill của các quái cũng hoạt động tốt, xử lý tốt khi chọn mức dễ, trung bình, khó. Ở mức dễ skill sẽ ít làm tiêu hao HP người chơi. Càng tăng dần độ khó, HP người chơi khi dính skill tụt càng nhanh.
- Khi người chơi Level up, các skill cũng được tăng thêm sức mạnh.
- Tuy nhiên, đối với skill lao về phía trước của người chơi, nhân vật trong game dễ bị lao về phía trước và dính vào các toà nhà đã có hệ thống phát hiện va chạm. Khi đó người chơi không thể điều khiển nhân vật được nữa.

4.3.4. Kiểm thử cho chức năng 4

Chức năng: Game over và Credit

Kết quả kiểm thử:

- Khi người chơi hết HP, một video sẽ hiện lên kèm dòng chữ Game Over và đưa người chơi về Main Menu. Tuy nhiên, video sẽ phát nhanh hay chậm tuỳ thuộc vào cấu hình máy để chạy chương trình.
- Màn hình Credit game: Khi người chơi chiến thắng boss cuối, video Credit sẽ hiện lên chúc mừng người chơi thắng cuộc, và đưa người chơi về Main Menu. Tuy nhiên, khi đã về Main Menu, người chơi sẽ không thể thực hiện thao tác New Game được mà phải thoát chương trình ra rồi vào lại để có thể chơi được New Game.

4.3.5. Kết luận

Tổng kết về project, chúng em nhận thấy đề tài còn tồn tại một số ưu nhược điểm sau:

- Ưu điểm: đáp ứng, hoàn thành đầy đủ các yêu cầu của đề tài, phần mềm chạy hoạt động ổn định, cốt truyện của game hấp dẫn, thiết kế giao diện đẹp mắt, trực quan, tương tác tốt với người dùng.

- Nhược điểm: phần mềm chiếm dụng khá nhiều tài nguyên máy tính do hàm DrawImage của class Graphics trong thư viện System.Drawing dùng GDI+, nên không thích hợp để làm các game lớn. Khi vào map được thiết kế to sẽ có chậm rõ rệt. Mặt khác, vì thời gian gấp rút nên chúng em chưa tối ưu hóa được code, vẫn còn tồn tại nhiều bug và nhiều tính năng chưa kịp phát triển.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

TÀI LIỆU THAM KHẢO

[1] Slide bài giảng môn Lập trình hướng đối tượng, thầy Nguyễn Mạnh Tuấn.
[2] Lập trình Windows Form và Web Form với C#, Nguyễn Tất Bảo Thiện
[3] https://vi.wikipedia.org/wiki/Giải-thuật-tìm-kiếm_A*
[4] https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban-1