**ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**

**LỚP KTPM 2013**

**----------oOo----------**

****

**ĐỒ ÁN MÔN NHẬP MÔN PHÁT TRIỂN GAME**

**Đề tài:**

**XÂY DỰNG GAME CASTLEVANIA**

**Giảng viên hướng dẫn:**

**ThS. NGUYỄN VĨNH KHA**

**GV HUỲNH HỒ THỊ MỘNG TRINH**

**Sinh viên thực hiện:**

**CHÂU NGỌC THÁI SƠN - 13520704**

**TRẦN HUY THỊNH - 13520830**

**VÕ ĐÌNH CAO MINH TRÍ - 13520921**

**Thành phố Hồ Chí Minh, ngày 01 tháng 01 năm 2017**

MỤC LỤC

[I/ Giới thiệu về game Castlevania 3](#_Toc471077781)

[II/ Giới thiệu đồ án 3](#_Toc471077782)

[III/ Phân tích và thiết kế 3](#_Toc471077783)

[1. Phân tích 3](#_Toc471077784)

[1.1. Danh sách các class sử dụng trong game 3](#_Toc471077785)

[1.2. Sơ đồ các class kế thừa trong game 5](#_Toc471077786)

[2. Thiết kế map 6](#_Toc471077787)

[2.1. Các thực hiện 6](#_Toc471077788)

[2.2. Kết quả sau khi lưu map 6](#_Toc471077789)

[IV/ Các kiến thức áp dụng 8](#_Toc471077790)

[1. Phương pháp lập trình hướng đối tượng 8](#_Toc471077791)

[1.1. Khái niệm 8](#_Toc471077792)

[1.2. Đặc điểm của lập trình hướng đối tượng 8](#_Toc471077793)

[1.3. Trừu tượng hóa dữ liệu 8](#_Toc471077794)

[1.4. Tính đa hình 8](#_Toc471077795)

[1.5. Tính kế thừa 9](#_Toc471077796)

[1.6. Tính đóng gói dữ liệu 9](#_Toc471077797)

[1.7. Những ưu điểm của lập trình hướng đối tượng 9](#_Toc471077798)

[1.8. Các thành phần trong lập trình hướng đối tượng 9](#_Toc471077799)

[2. Hướng đối tượng trong game Castlevania 10](#_Toc471077800)

[3. Kỹ thuật phân hoạch không gian bằng Quadtree 10](#_Toc471077801)

[3.1. Giới thiệu về cây tứ phần (Quadtree) 10](#_Toc471077802)

[3.2. Sử dụng Quadtree để phân hoạch không gian: 11](#_Toc471077803)

[V/ Giao diện và cách chơi: 13](#_Toc471077804)

[1. Menu: 13](#_Toc471077805)

[2. Màn hình chơi game: 13](#_Toc471077806)

[3. Cách chơi game: 15](#_Toc471077807)

[VI/ Nhận xét, kết luận và hướng phát triển: 15](#_Toc471077808)

[1. Nhận xét: 15](#_Toc471077809)

[2. Kết luận: 16](#_Toc471077810)

[3. Hướng phát triển: 16](#_Toc471077811)

# I/ Giới thiệu về game Castlevania

Castlevania (tên tiếng Nhật là Akumajou Dracula) là một tựa game được phát triển và xuất bản bởi tập đoàn Konami vào tháng 9/1986 tại Nhật Bản. Vào tháng 5/1987, game được phát hành cho hệ máy NES của Nitendo tại Bắc Mỹ và tiếp theo là tại châu Âu vào năm 1988.

Castlevani là tựa game kinh điển của thời đại game 8 bit: game gồm 6 màn chơi, nhân vật vượt qua các thử thách trong game để gặp boss cuối.

Cốt truyện: Trò chơi mở đầu cho cuộc đấu tranh không khoan nhượng giữa gia tộc Belmont và bá tước Dracula - con ma cà rồng đầu tiên và hùng mạnh nhất! Sau mỗi lần tưởng như đã bị tiêu diệt, Dracula lại trỗi dậy, và một Belmont trẻ lại tiếp tục sứ mệnh của gia đình với chiếc roi Vampire Killer. Chiếc roi này là một vũ khí được cường hóa thông qua giả kim thuật - được trao cho Leon Belmont, thành viên đầu tiên trong gia đình đã từng đụng độ với loài quỷ dữ. Trong *Castlevania* 1986, chủ nhân của chiếc roi là một trong các hậu duệ của Leon - Simon Belmont.  Simon Belmont sẽ phiêu lưu trong lâu đài, vượt qua các đày tớ của Dracula và cuối cùng kết liễu hắn.

# II/ Giới thiệu đồ án

Phiên bản gốc của game có rất nhiều màn chơi rất hấp dẫn, tuy nhiên trong phạm vi đồ án môn học này chỉ thực hiện 2 màn chơi của game. Yêu cầu chương trình của chúng ta ở đây là làm sao thiết kế được 1 chương trình game giống (hoặc gần giống) như các màn chơi trong game CastleVania trên trên nền tảng DirectX, dựa vào các phương pháp lập trình đã học. Cụ thể là sử dụng phương pháp lập trình hướng đối tượng kết hợp với các kĩ thuật khác như phân chia không gian sử dụng cây tứ phân (Quadtree), các kĩ thuật chuyển đổi tọa độ và sử dụng các thư viện đồ họa, thư viện âm thanh trong DirectX.

# III/ Phân tích và thiết kế

## 1. Phân tích

### 1.1. Danh sách các class sử dụng trong game

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Class** | **Chức năng** |
| 1 | Game | Là lớp chứa các chuỗi sự kiện chính của game, khởi tạo màn hình, directx, keyboard, …. |
| 2 | SceneGame | Là lớp chứa các hàm chính khi chơi game như xử lý camera, xử lý stage và level, load game resources,.. |
| 3 | SceneMenu | Là lớp chứa các hàm để tạo và xử lý menu chính của game. |
| 4 | SceneMain | Là lớp render menu và các scene của game. |
| 5 | GameScore | Là lớp có chức năng tạo và xửa lý bảng điểm trong game chứa các thông tin như HP, điểm, loại vũ khí, item, thời gian… |
| 6 | Qbackground | Có chức năng load tilemap từ file text trước đó đã tạo để tạo background gameplay, load file Object chứa thong tin các object game và file quadtree chứ vị trí các object/node để vẽ lên màn hình một cách tối ưu. |
| 7 | Qnode |
| 8 | Tile |
| 9 | Camera | Là lớp định nghĩa camera chính của game, tạo viewport, chuyển đổi toạ độ thế giới thực. |
| 10 | Caudio | Là lớp định nghĩa khởi tạo thiết bị và xửa lý âm thanh. |
| 11 | dsutil |
| 12 | dxutil |
| 13 | Ctext | Là lớp định và khởi tạo Font trong game. |
| 14 | Csprite | Lớp xỷ lý các sprite trong game |
| 15 | Ctextture | Lớp xử lý các hình ảnh tạo nên các sprite. |
| 16 | SweptAABB | Chứa hàm xét va chạm trong game. |
| 17 | QGameObject | Lớp có chức năng đọc file Object từ file map và khởi tạo theo đúng từng object ==> list object game, và list object trong viewport. |
| 18 | GameObject | Là lớp cha của toàn bộ các object trong game, định nghĩa loại, các thuộc tính và hàm cơ bản. |
| 19 | DynamicObject | Là lớp được kế thừa từ lớp GameObject, định nghĩa các loại Object có khả năng di chuyển trong game. |
| 20 | LKRect | Lớp chứa các thuộc tính của khung bao object |
| 21 | Singleton | Là lớp load các file hình ảnh sprite của các object. |
| 22 | SoundManager | Lớp xử lý âm thanh trong game. |
| 23 | Cenum | Là lớp khai báo biến enum quy định ID của các object. |
| 24 | Global | Chứa các thông số cơ bản của game như các biến về kích thướt màn hình, biến thiết bị,… |
| 25 | Utils | Lớp chứa hàm tạo surface từ 1 file hình ảnh. |
| 26 | Main | Lớp gọi các hoạt động của game. |
| 27 | QueenMedusa | Lớp định nghĩa boss level 2 của game |
| 28 | LittleSnake | Lớp định nghĩa đạn của boss level 2 |
| 29 | BlackKnight | Lớp định nghĩa hiệp sĩ bóng tối - enemy |
| 30 | DragonSkullCanon | Lớp định nghĩa trụ đầu rồng - enemy |
| 31 | FireBall | Lớp định nghĩa đạn của DragonSkull |
| 32 | Flyingblock | Lớp định nghĩa con ruồi - enemy |
| 33 | Medusa | Lớp định nghĩa đầu rắn biết bay - enemy |
| 34 | VampireBat | Lớp định nghĩa dơi hút máu - enemy |
| 35 | Brick | Lớp định nghĩa Viên gạch. |
| 36 | BrokenBrick | Lớp định nghĩa viên gạch đánh vỡ được |
| 37 | Candle | Lớp định nghĩa cây nến nhỏ. |
| 38 | Door | Lớp định nghĩa các loại cửa của game. |
| 39 | MovingPlatform | Lớp định nghĩa bệ di chuyển |
| 40 | OpenDoor | Lớp chứa hàm xử lý khi mở cửa. |
| 41 | Stair | Lớp định nghĩa Cầu thang. |
| 42 | StupidDoor | Lớp định nghĩa bẫy nhọn đâm xuống đất. |
| 43 | Simon | Lớp định nghĩa nhân vật chính Simon của game, chứa các trạng thái, thuộc tính, hàm va chạm, … |
| 44 | Axe | Lớp định nghĩa vũ khí rìu - weapon |
| 45 | Boomerang | Lớp định nghĩa boomerang |
| 46 | Dagger | Lớp định nghĩa dao găm. |
| 47 | FireBoom | Lớp định nghĩa quả bom - weapon |
| 48 | Morningstar | Lớp định nghĩa thanh roi của simon. |
| 49 | Weapon | Lớp chứa các thuộc tính cơ bản của weapon, xử lý va chạm. |
| 50 | Crown | Lớp định nghĩa Vương Miện |
| 51 | DoubleShot | Lớp định nghĩa vật phẩm nhân đôi sát thương |
| 52 | Firedie | Lớp định nghĩa hiệu ứng khi đánh chết enemy hay cây nến,… |
| 53 | Food | Lớp định nghĩa vật phẩm đồ ăn tăng hp cho simon. |
| 54 | GoldChest | Lớp định nghĩa Rương vàng |
| 55 | MagicalCrystal | Lớp định nghĩa vật phẩm rơi ra khi giết boss |

### 1.2. Sơ đồ các class kế thừa trong game

****

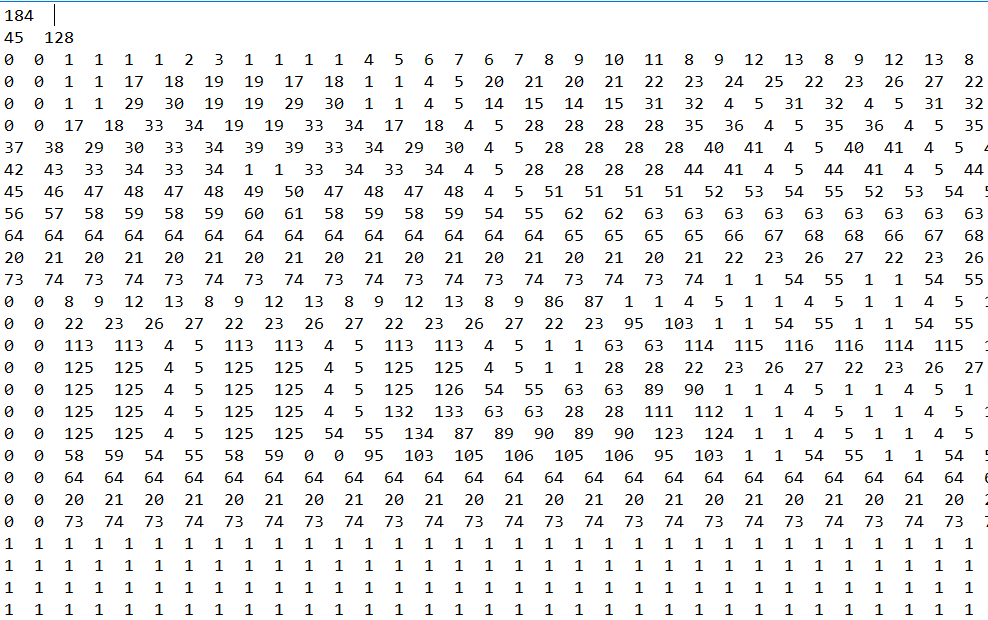
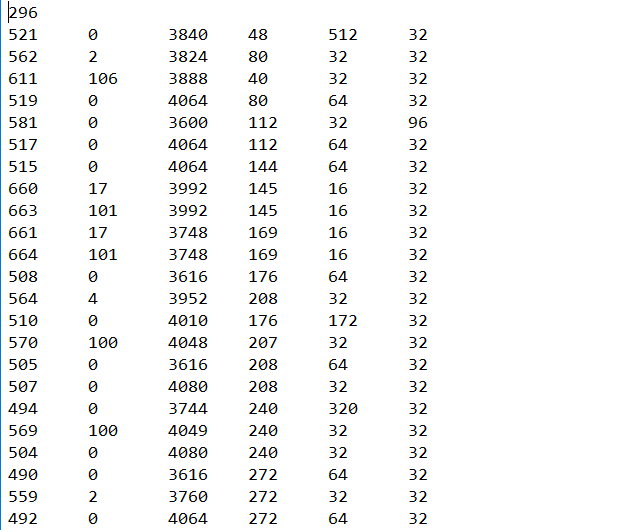
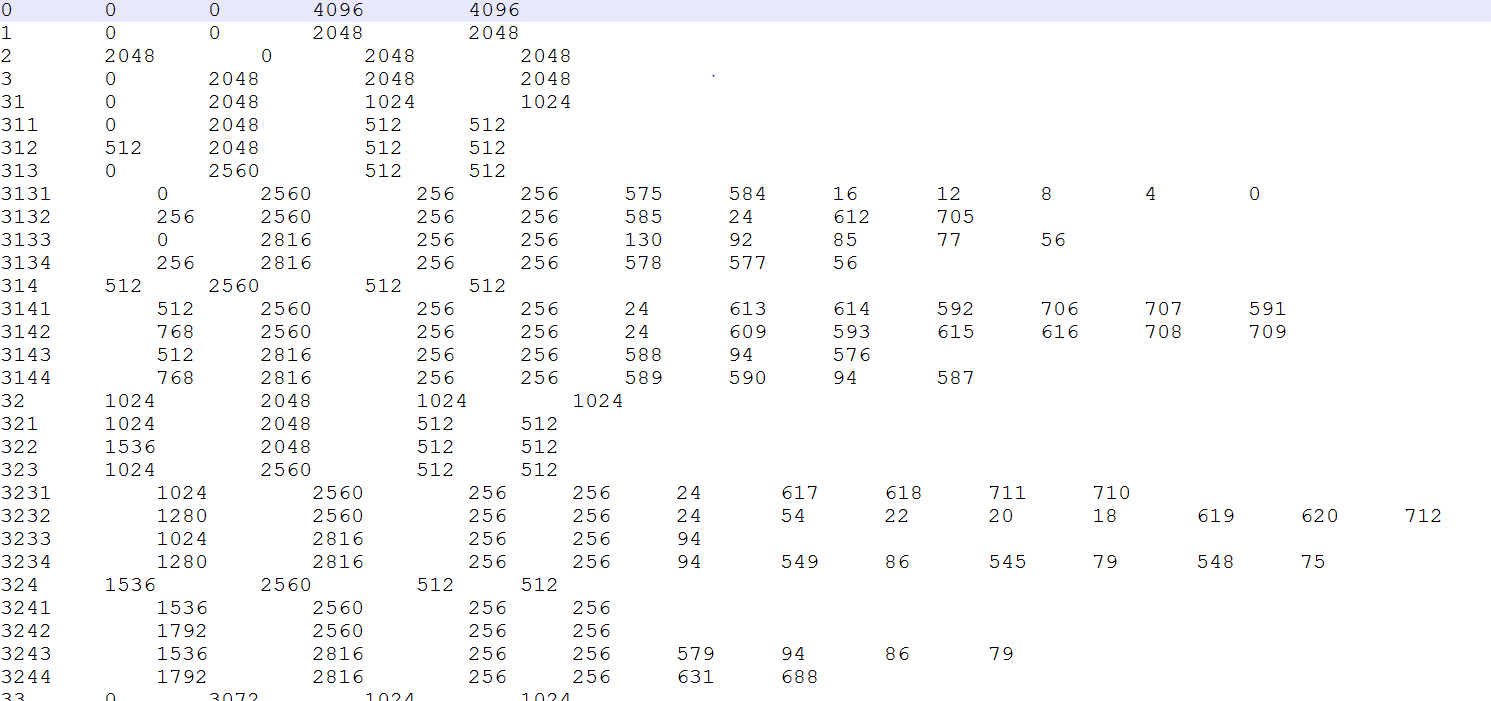
## 2. Thiết kế map

### 2.1. Các thực hiện

Sử dụng công cụ hỗ trợ làm map viết bằng ngôn ngữ C#, với chức năng chính là hỗ trợ đưa các đối tượng vào map, canh chỉnh vị trí cho phù hợp và sau đó lưu file map dưới định dạng text. Công cụ đã được cài đặt sẵn tính năng buid quadtree trước khi lưu. Về phần các đối tượng, xem mỗi đối tượng vẽ là một game object, có ID, Type (loại đối tượng), X, Y, Width, Height để thuận tiện cho việc load map trong game.

### 2.2. Kết quả sau khi lưu map

Kết quả sau khi lưu ta sẽ có các file dưới định dạng text như sau :

* TileInMap: Chứa các tile dùng để load Background
* MatrixTile: Dòng đầu là số các tile khác nhau, dòng thứ 2 chứa số dòng và số cột, các dòng ở dưới là ma trận các id của tile.  
  
* ObjectInMap: Dòng đầu là số lượng Object có trong map. Các dòng dưới theo thứ tự là Id, tên Object, tọa độ x, tọa độ y, chiều rộng, chiều cao của object.  
  
* QuadTree: Cột đầu tiên chứa id của Node, cột 2 và 3 chứa tọa độ x, y của Node, cột 4 và 5 chứa chiều rộng và chiều cao của Node, các cột tiếp theo chứa danh sách Id của các Object có trong Node.  
  

# IV/ Các kiến thức áp dụng

## 1. Phương pháp lập trình hướng đối tượng

### 1.1. Khái niệm

Lập trình hướng đối tượng (gọi tắt là OOP, từ chữ Anh ngữ object-oriented programming), hay còn gọi là lập trình định hướng đối tượng, là kĩ thuật lập trình hỗ trợ công nghệ đối tượng. OOP được xem là giúp tăng năng suất, đơn giản hóa độ phức tạp khi bảo trì cũng như mở rộng phần mềm bằng cách cho phép lập trình viên tập trung vào các đối tượng phần mềm ở bậc cao hơn. Ngoài ra, nhiều người còn cho rằng OOP dễ tiếp thu hơn cho những người mới học về lập trình hơn là các phương pháp trước đó.

### 1.2. Đặc điểm của lập trình hướng đối tượng

Tập trung vào dữ liệu thay cho các hàm. Chương trình được chia thành các đối tượng độc lập. Cấu trúc dữ liệu được thiết kế sao cho đặc tả được các đối tượng. Dữ liệu được che giấu, bao bọc. Các đối tượng trao đổi với nhau thông qua các hàm. Chương trình được thiết kế theo hướng tiếp cận từ dưới lên.

### 1.3. Trừu tượng hóa dữ liệu

Khi bắt đầu phát triển phần mềm, lập trình viên sẽ không đi vào viết lệnh ngay lập tức. Công việc đầu tiên cần nghiên cứu dữ án, phân tích các đặc tả phần mềm, các thành phần cần có. Theo đó, lập trình viên sẽ phải xác định những thông tin, dữ liệu cần thiết cho từng thành phần. Như vậy, trừu tượng hóa dữ liệu là quá trình xác định, tập hợp những thuộc tính, hành xử cho một thực thể cụ thể liên quan trong ứng dụng. Trong quá trình đó, có thể có những chi tiết sẽ được cài đặt nhưng ngược lại sẽ có những chi tiết cần loại bỏ vì chúng không nằm trong yêu cầu.

Những ưu điểm của việc trừu tượng hóa dữ liệu bao gồm:

* Tập trung vào vấn đề.
* Xác định rõ những thành phần và chức năng cần thiết.
* Loại trừ những chi tiết không cần thiết.

### 1.4. Tính đa hình

Đa hình là khả năng cho phép một hoạt động có những hành xử khác nhau trong cùng một đối tượng. khái niệm này ám chỉ việc đối tượng có nhiều phương thức cùng tền nhưng lại thực hiện những nhiệm vụ khác nhau. Trình biên dịch và hệ thống thực thi sẽ so khớp mỗi lời gọi phương thức, đối tượng. Khả năng tìm kiếm đúng để sử dụng trong những tình huống phức tạp là bản chất của tính đa hình. Đa hình là một trong những đặc trưng quan trọng của hệ thông hướng đối tượng.

### 1.5. Tính kế thừa

Kế thừa là khả năng cho phép một lớp chia sẻ thuộc tính, hành xử được cài đặt trong nó. Nguyên lý kế thừa hỗ trợ việc tạo ra cấu trúc phần cấp các lớp. Nó được hiểu như một cở chế sao chép ảo không đơn điệu và có chắt lọc. Một lớp có thể thừa kế từ một hay nhiều lớp khác. Theo đó, lớp được kế thừa là lớp cha và lớp kế thừa là lớp con.

### 1.6. Tính đóng gói dữ liệu

Đóng gói dữ liệu là quá trình che dấu những cài đặt cụ thể của một đối tượng với một đối tượng sử dụng khác. Dữ liệu được tổ chức có thể phơi bày với thế giới bên ngoài hoặc che dấu vào bên trong đối tượng. Những thuộc tính, hành xử được phởi bày để được truy cập, triệu gọi từ một đối tượng khác. Ngược lại, những thuộc tính, hành xử được che giấu chỉ đảm nhận hành xử hay lưu trữ dữ liệu cho chính đối tượng đó, chúng không có khả năng truy cập từ ngoài vào.

### 1.7. Những ưu điểm của lập trình hướng đối tượng

Loại bỏ được những dư thừa, trùng lặp trong việc xây dựng ứng dụng. Cài đặt đối tượng giúp xức tiến việc sử dụng lại, trao đổi giữa các đối tượng với nhau do đó sẽ giảm kích thước, thời gian xử lý,… thời gian phát triển hệ thống, tăng năng xuất lao động. Dễ bảo trì, nâng cấp, giảm lỗi.

### 1.8. Các thành phần trong lập trình hướng đối tượng

- Class: các chức năng, thành phần của một thực thể được định nghĩa một cách chung nhất trong một mẫu cụ thể được gọi là class.

- Object: là một đối tượng cụ thể của class

- Attribute: là những thành phần được định nghĩa trong một class.

- Operation: giống như một dịch vụ được triệu gọi bởi một đối tượng

- Method: còn gọi là phương thức định nghĩa cụ thể một operation

-Message: là những đòi hỏi, những thông tin cần có cho một operation.

- Event: là những kích hoạt để thực thi một hay nhiều operation

- Construction: là triệu gọi đầu tiên khi một object được tạo. Constructor là một method đặc biệt được gọi đầu tiên trước khi một method hay attribute có thể được sử dụng trong một object của một class. Constructor có thể khỏi gán nhữn thuộc tính, cấp phát bộ nhớ nếu cần thiết.

- Destruction: là phương thức đặc biệt được gọi để hủy một object. Destruction sẽ xóa object, giải phóng bộ nhớ mà constructor cấp phát. Theo đó, mọi khả năng sử dụng truy cập tới object này sẽ không còn.

## 2. Hướng đối tượng trong game Castlevania

Nguyên tắc kế thừa cho phép các lớp dẫn xuất sử dụng các phương thức từ lớp cơ sở như những phương thức của chính nó, mà không cần thiết phải định nghĩa lại.

Để thuận tiện cho việc tái sử dụng lại các phương thức hay dùng trong các đối tượng khác nhau, việc tự động so khớp và thực thi các phương thức và các đối tượng. Trong game, chúng em sử dụng 1 class GameObject là class cha và các class dẫn xuất.

Việc sử dụng chung con trỏ của lớp cha và tự động trỏ đến đúng đối tượng dẫn xuất giúp cho việc xử lý trở nên dễ dàng hơn.

Mỗi vật trong game được đưa vào một class quản lý. Việc chia nhỏ này giúp cho việc quản lý cũng như xử lý trong game dễ dàng hơn.

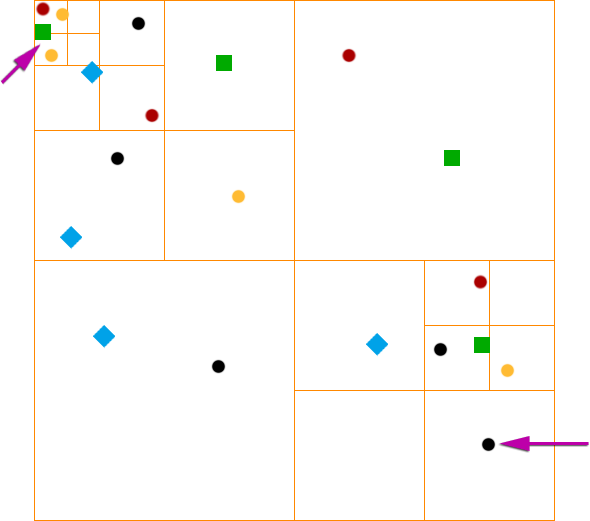
Danh sách các class sử dụng kế thừa trong game:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Class** | **Chức Năng** |
| 1 | GameObject | Đây là lớp cha, chứa các hàm ảo và các thuộc tính cần thiết cho việc lưu trữ, tất cả các đối tượng trong game đều kế thừa từ lớp này. |
| 2 | DynamicObject | Đây là lớp cha của tất cả các đối tượng có thể di chuyển, kế thừa từ lớp GameObject |
| 3 | Scene | Lớp nền cho các màn hình trong game như menu, scenegame…. |

### 3. Kỹ thuật phân hoạch không gian bằng Quadtree

### 3.1. Giới thiệu về cây tứ phần (Quadtree)

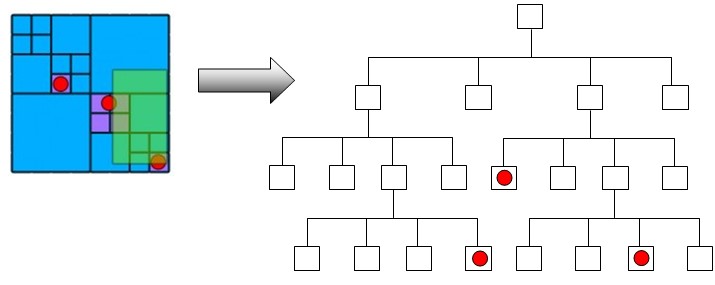
* Quadtree là một cấu trúc dữ liệu phân nhánh dạng cây, được sử dụng để phân hoạch vùng không gian hai chiều hiện tại thành các vùng nhỏ hơn và dễ quản lý hơn. Khác với cây nhị phân, mỗi lần phân hoạch, quadtree sẽ chia vùng hiện tại thành 4 vùng. Do đó mỗi node trong quadtree có thể có 4 node dẫn xuất hoặc không có node dẫn xuất nào, không có ngoại lệ.
* Việc sử dụng Quadtree giúp chúng ta quản lý các đối tượng trong game một cách hiệu quả hơn, bằng cách phân chia vùng cần xử lý các đối tượng thành những vùng con, các đối tượng sẽ được đưa vào các vùng tương ứng và được quản lý riêng biệt. Quá trình phân chia cứ thế tiếp diễn cho đến khi mỗi vùng chỉ chứa một số lượng đối tượng nhất định hoặc mức độ phân chia đạt mức “chấp nhận được”.



*Hình 1.1.1 - Sử dụng cây tứ phân trong phân hoạch không gian*

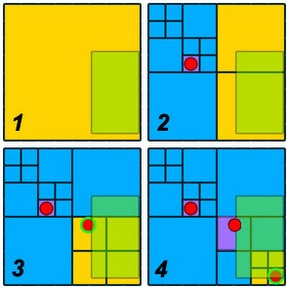
### 3.2. Sử dụng Quadtree để phân hoạch không gian:

* Một vấn đề lớn khi làm game Castlevania cũng như một số game 2d khác đó là số lượng các đối tượng trong một màn chơi là khá lớn (khoảng 500). Nếu khi cần xét va chạm, mà ta phải duyệt qua tất cả các đối tượng thì sẽ rất lãng phí tài nguyên.Trong khi đó, hầu như ta chỉ cần xét va chạm với các đối tượng trong màn hình, như vậy việc xét hết các đối tượng là không cần thiết. Vì vậy, ta phải làm cách nào đó để số lượng đối tượng phải duyệt qua giảm đi một lượng đáng kể.
* Để giảm lượng đối tượng được xét, ta phải sử dụng phương pháp phân hoạch không gian. Phương pháp được cho là hiệu quả nhất với một map lớn là QuadTree – phân chia không gian làm bốn phần. QuadTree sẽ chia vùng không gian ra thành những vùng bằng nhau, và nếu đối tượng có thể chứa được ở những vùng không gian nhỏ hơn thì không gian con sẽ được chia nhỏ hơn nữa. Việc phân chia sẽ diễn ra cho đến khi không có không gian con nhỏ hơn chứa được đối tượng đó.
* Khi nhắc đến cây, người ta thường nghĩ ngay đến phương pháp đệ quy. Nếu không sử dụng đệ quy có thể sử dụng hàng đợi để duyệt cây, nhưng việc cài đặt khá là phức tạp. Sử dụng đệ quy giúp cho việc lập trình trở nên đơn giản hơn nhiều, mà không làm giảm hiệu suất của chương trình nhiều.

**

Hình 4. Giới thiệu về cách chia node trong Quadtree

* Các bước thực hiện:

**

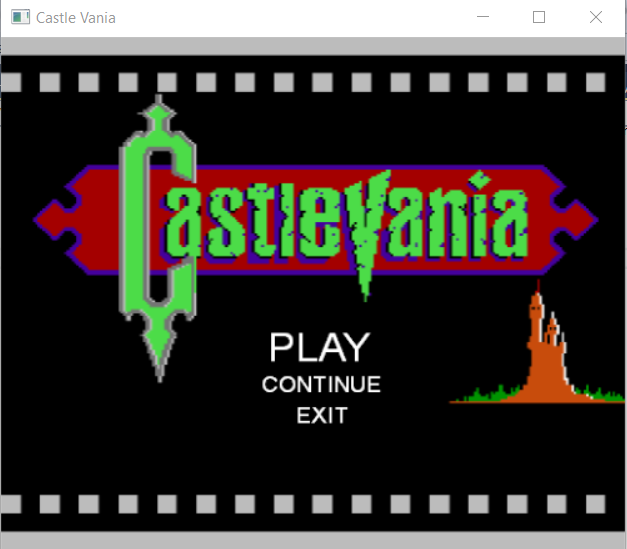
Hình 5. Các bước thực hiện việc kiểm tra va chạm trong Quadtree

* Hình chữ nhật xanh va chạm với hình chữ nhật vàng lớn (node mức 1), nhưng không có đối tượng nào ở mức 1.
* Hình chữ nhật xanh va chạm với 2 hình chữ nhật vàng nhỏ ở mức 2 (2 node ở mức 2): cũng không có đối tượng nào, ta tiếp tục kiểm tra các node con.
* Hình chữ nhật xanh va chạm với 4 hình chữ nhật vàng nhỏ ở mức 3 (4 node ở mức 3): có 1 đối tượng, ta đưa đối tượng vào danh sách trả về (danh sách các đối tượng có xảy ra va chạm với hình chữ nhật xanh).
* Hình chữ nhật xanh va chạm với 4 hình chữ nhật vàng nhỏ ở mức 4 (6 node ở mức 4): có 1 đối tượng, ta đưa đối tượng vào danh sách trả về.

# V/ Giao diện và cách chơi:

## 1. Menu:

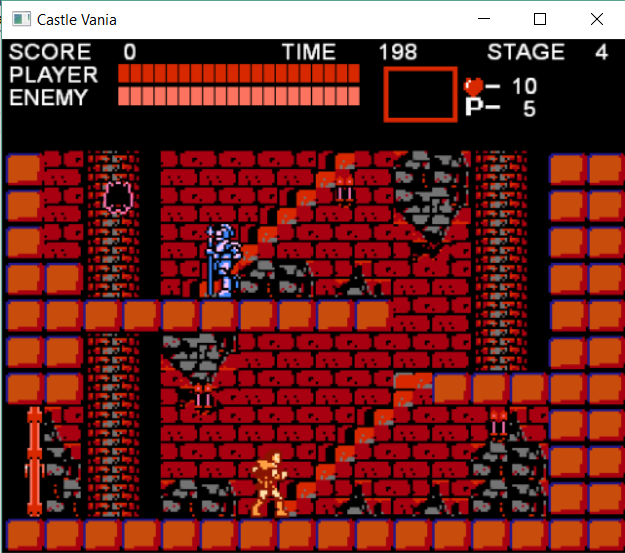
Giao diện menu của game đơn giản, gồm các lựa chọn, logo của game:



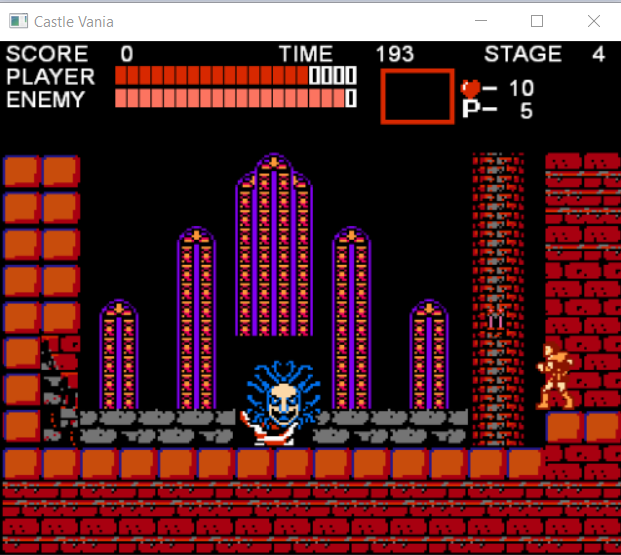
*Menu*

## 2. Màn hình chơi game:

Hiển thị điểm, thời gian, các vật phẩm, một số thông tin khác:



*Màn hình chơi game – cảnh lúc bắt đầu*



*Gặp boss Queen Medusa*

## 3. Cách chơi game:

Nhìn chung, game có cách chơi khá đơn giản:

* Di chuyển: Người chơi sử dụng các nút mũi tên trái phải để di chuyển simon, nút mũi tên lên và xuống để điều khiển simon lên hoặc xuống cầu thang, nút space để khiến simon nhảy lên.
* Tấn công: Người chơi nhấn phím Enter để khiến simon quất roi, nhấn đè phím Q và nhấn phím Enter để simon sử dụng các loại vũ khí khác.

# VI/ Nhận xét, kết luận và hướng phát triển:

## 1. Nhận xét:

Các chức năng trong game đa phần đã hoạt động tốt. Simon, các item, quái, boss di chuyển và hoạt động tốt, các đối tượng va chạm tương đối ổn với nhau. Tuy nhiên, do xử lý va chạm chưa thật sự tốt nên sự va chạm trong game chưa được ổn định và chính xác cho lắm.

## 2. Kết luận:

So với yêu cầu đồ án đặt ra thì game của nhóm đã hoàn thành tương đối. Nhóm đã áp dụng được các kỹ thuật xử lý va chạm AABB, sử dụng Quadtree, mapeditor cho game. Nhóm cũng đã học được rất nhiều từ đồ này, từ các kỹ thuật làm game cơ bản, cho đến việc cải thiện kỹ năng phối hợp làm việc nhóm.

Mặc dù đã có cố gắng rất nhiều và bỏ công sức nghiên cứu, cũng như sự hỗ trợ nhiệt tình của thầy cô và bạn bè, nhóm vẫn không tránh khỏi những thiếu sót. Nhóm mong nhận được sự góp ý của thầy cô về đồ án này.

## 3. Hướng phát triển:

Trong thời gian tới, nhóm sẽ tiếp tục xây dựng và hoàn thiện đồ án hơn nữa, xử lý va chạm tối ưu hơn và làm những màn tiếp theo. Nhóm xin chân thành cảm ơn sự đóng góp ý kiến của thầy cô và bạn bè trong suốt quá trình thực hiện đồ án.