

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM**

----- o0o -----



BÁO CÁO MÔN HỌC
Nhập môn phát triển game
SE102.J21.PMCL
Đề tài: Game NinjaGaiden 1

Giảng viên hướng dẫn:

ThS. Nguyễn Vĩnh Kha

Sinh viên thực hiện:

Phan Quốc Hưng 15520288

Nguyễn Tấn Diệu 15520116

Tp. Hồ Chí Minh, tháng 06/2019

[illegible]

MỤC LỤC

CHƯƠNG 1: GIỚI THIỆU DIRECTX.....	1
<i>1.1. DirectX là gì?.....</i>	<i>1</i>
CHƯƠNG 2: GIỚI THIỆU GAME NINJA GAIDEN 1	1
<i>2.1. Tổng quan Game Ninja Gaiden 1</i>	<i>1</i>
<i>2.2. Giới thiệu Player</i>	<i>1</i>
2.2.1. Ryu Hayabusa	1
2.2.2. Hệ thống vũ khí.....	1
<i>2.3. Giới thiệu hệ thống Enemy, Boss.....</i>	<i>2</i>
2.3.1. Enemy	2
2.3.2. Boss	4
<i>2.4. Giới thiệu hệ thống item.....</i>	<i>4</i>
2.4.1. Các item cơ bản	4
2.4.2. Item hiển thị.....	5
CHƯƠNG 3: ĐỒ ÁN GAME NINJA GAIDEN 1	6
<i>3.1. Các kỹ thuật áp dụng trong game</i>	<i>6</i>
3.1.1. Sprite	6
3.1.2. Thuật toán xử lý va chạm SweptAABB	6
3.1.3. Viewport (Camera)	7
3.1.4. Texture	7
3.1.5. Tool Tile Map	7
3.1.7. Tile Map	8
3.1.8. Thuật toán phân hoạch không gian Grid	8
<i>3.2. Sơ đồ các class chính trong game</i>	<i>9</i>
3.2.1. Sơ đồ mô phỏng các class object có trong game	9
3.2.2. Sơ đồ mô phỏng các class framework.....	11
3.2.2. Sơ đồ mô phỏng các class màn hình	11
3.2.2. Design pattern	11
<i>3.3. Công cụ hỗ trợ.....</i>	<i>12</i>
<i>3.4. Kết quả thực hiện</i>	<i>12</i>
3.4.1. Đánh giá mức độ hoàn thiện của Game	12

3.4.2. Hướng phát triển (nếu có).....	12
3.4.3. <i>Bảng phân công công việc</i>	12
3.5. <i>Kết luận</i>	13
TÀI LIỆU THAM KHẢO	13

CHƯƠNG 1: GIỚI THIỆU DIRECTX

1.1. DirectX là gì?

Microsoft DirectX là 1 tập hợp các API xử lý các tác vụ liên quan đến hình ảnh, âm thanh. DirectX có thể coi như là 1 chiếc cầu nối liên giữa phần cứng và phần mềm. Nó nhận lệnh liên quan đến hình ảnh, âm thanh từ các phần mềm (game, trình duyệt phim...) rồi xử lý và truyền tải đến phần cứng (chip âm thanh, card đồ họa...). DirectX được Microsoft xây dựng thành 1 thư viện các tác vụ, mã lệnh v.v... ngày càng đồ sộ và thông minh giúp cho các game ngày càng xử lý âm thanh chân thực, hình ảnh mượt mà sắc nét. Ngày nay DirectX có thể xử lý được các hiệu ứng giả lập âm thanh nhiều chiều phức tạp và những hình ảnh chuyển động có độ phân giải cao.

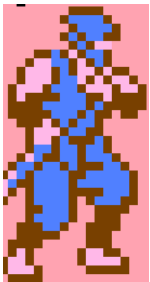
CHƯƠNG 2: GIỚI THIỆU GAME NINJA GAIDEN 1

2.1. Tổng quan Game Ninja Gaiden 1

Ninja Gaiden 1 được xây dựng trên nền tảng máy NES, một trò chơi nổi tiếng với độ khó cực kì cao vào thời điểm ra mắt (12/1988). Chúng ta nhập vai vào Ryu Hayabusa, câu chuyện bắt đầu khi cha của Ryu bị giết chết bởi các sát thủ, anh đã bắt đầu đi trả thù, tìm ra ai là người giết cha mình. Với đồ họa 2D và góc nhìn ngang, game mang lại trải nghiệm rất tốt, game chỉ có chế độ single player. Game đã đạt các giải thưởng nổi bật như: Best Challenge and Best Ending (1989), Best Game of the Year (NES),...

2.2. Giới thiệu Player

2.2.1. Ryu Hayabusa



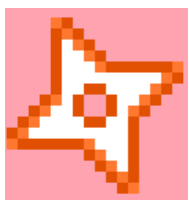
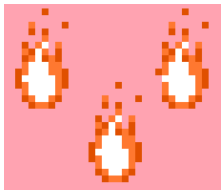
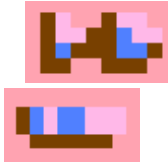




Người chơi sẽ nhập vai vào Ryu Hayabusa, một ninja với nhiều kĩ năng đặc biệt.

Trạng thái: đứng yên, đứng chém, nhảy quay vòng chém, sử dụng vũ khí đặc biệt, ngồi, ngồi chém, chết, bị thương.

2.2.2. Hệ thống vũ khí

Hình ảnh	Tên	Mô tả
----------	-----	-------



	Katana	Đây là vũ khí gắn liền với Ryu, dùng để giết quái mà không hao tốn điểm kỹ năng.
	Small Shuriken	Phi tiêu nhỏ, khi Ryu ném thì sẽ giết được con quái đầu tiên chạm vào và sau đó mất.
	Big Shuriken	Phi tiêu lớn, khi Ryu ném, nó sẽ giết hết tất cả con quái trên đường bay, quỹ đạo bay như một Boomerang. Khi chạm vào người Ryu hoặc ném vào boss thì biến mất.
	Circle Fire	Vòng lửa tròn, đây là vũ khí mạnh nhất của Ryu, phóng ra 3 vòng lửa, giết mọi con quái trên đường bay. Khi boss chạm cả 3 vòng lửa thì mất 3 hp. Chỉ biến mất khi ra khỏi viewport hoặc chạm boss.
	Bullet	Viên đạn của những con quái cầm súng, với sát thương là 1 hp ứng với 1 viên. Có score: 100
	Sword's witch	Vũ khí của witch, quăng những cây kiếm gây sát thương, 1 hp/1 cây. Có score: 100
	Boom	Bom của boss, ném ra 3 viên. Sát thương 1 hp/1 viên Có score: 200

2.3. Giới thiệu hệ thống Enemy, Boss

2.3.1. Enemy

Enemy	Hình thức tấn công/di chuyển	Trạng thái
-------	------------------------------	------------

<div>Soldier Sword</div> <table><tr><td>HP</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>200</td></tr></table> <div></div>	HP	1	Damage	1	Score	200	Lính cầm đao, di chuyển qua lại theo đường thẳng	Di chuyển Tấn công Chết
HP	1							
Damage	1							
Score	200							
<div>Eagle</div> <table><tr><td>HP</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>300</td></tr></table> <div></div>	HP	1	Damage	1	Score	300	Đây là con enemy khó chịu nhất game, thường xuất hiện những nơi vực thẳm, người chơi dễ “lọt hố”	Dĩ theo người chơi (Ryu) Chết
HP	1							
Damage	1							
Score	300							
<div>Bat</div> <table><tr><td>HP</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>200</td></tr></table> <div></div>	HP	1	Damage	1	Score	200	Con dơi bay theo quỹ đạo sin, thường xuất hiện ở những nơi vực thẳm, gây chế khi nhảy	Di chuyển Chết
HP	1							
Damage	1							
Score	200							
<div>Witch</div> <table><tr><td>HP</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td></td></tr></table> <div></div>	HP	1	Damage	1	Score		Phù thủy ném những cây kiếm, gây ức chế cho người chơi khi nhảy qua các cột.	Di chuyển Tấn công Chết
HP	1							
Damage	1							
Score								
<div>Soldier Run</div> <table><tr><td>HP</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>300</td></tr></table> <div></div>	HP	1	Damage	1	Score	300	Xuất hiện bất ngờ, chạy nhanh làm người chơi dễ bị mất máu.	Di chuyển Chết
HP	1							
Damage	1							
Score	300							

<table><tr><th colspan="2">Soldier Gun</th></tr><tr><td>Hp</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>200</td></tr></table> 	Soldier Gun		Hp	1	Damage	1	Score	200	Tên lính bắn súng, bắn 1 lần ra 3 viên.	Di chuyển Tấn công Chết
Soldier Gun										
Hp	1									
Damage	1									
Score	200									
 <table><tr><th colspan="2">Panther</th></tr><tr><td>Hp</td><td>1</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>200</td></tr></table>	Panther		Hp	1	Damage	1	Score	200	Xuất hiện bất ngờ, chạy nhanh, làm người chơi khó né tránh.	Di chuyển Chết
Panther										
Hp	1									
Damage	1									
Score	200									

2.3.2. Boss



Boss	Hình thức tấn công/di chuyển	Trạng thái								
<table><tr><th colspan="2">Boss</th></tr><tr><td>HP</td><td>16</td></tr><tr><td>Damage</td><td>1</td></tr><tr><td>Score</td><td>6100</td></tr></table> 	Boss		HP	16	Damage	1	Score	6100	Nhảy qua nhảy lại, nhảy 3 lần thì lần thứ 3 đáp đất sẽ thả ra 3 viên bom	Nhảy Chết
Boss										
HP	16									
Damage	1									
Score	6100									

2.4. Giới thiệu hệ thống item

2.4.1. Các item cơ bản

Hình ảnh	Loại item	Mô tả
	Item point blue	Cho người chơi 5 điểm kỹ năng.
	Item point red	Cho người chơi 10 điểm kỹ năng.
	Small Shuriken	Cho người chơi vũ khí phi tiêu nhỏ.
	Big Shuriken	Cho người chơi vũ khí phi tiêu lớn.
	Circle Fire	Cho người chơi vũ khí chường ra 3 vòng lửa.
	Timer	Đóng băng tất cả object trong viewport khoảng 3s.
	Blood	Hồi đầy máu cho người chơi.
	Vase blue	Cho người chơi 500 điểm. (score)
	Vase red	Cho người chơi 1000 điểm. (score)

2.4.2. Item hiển thị

Hình ảnh	Tên	Mô tả
	Food	Khi đánh thì rơi item (tham chiếu 2.4.1). Item ở Stage 2-1.
	Food	Khi đánh thì rơi item (tham chiếu 2.4.1). Item ở Stage 2-2.

CHƯƠNG 3: ĐỒ ÁN GAME NINJA GAIDEN 1

3.1. Các kỹ thuật áp dụng trong game

3.1.1. Sprite

- Giúp quản lý sprite sheet, tổng số frame, kích thước của 1 frame, update frame (next).
- Một hành động có nhiều frame, khi thực hiện 1 hành động ta sẽ cập nhật lại chỉ số frame.

3.1.2. Thuật toán xử lý va chạm SweptAABB

- Nhóm em chọn xét va chạm trên hình chữ nhật.
- Giả sử ta có Rect1, Rect2. Nếu $!(\text{Rect1.left} > \text{Rect2.right} \parallel \text{Rect1.right} < \text{Rect2.left} \parallel \text{Rect1.top} > \text{Rect2.bottom} \parallel \text{Rect1.bottom} < \text{Rect2.top})$ thì xảy ra hiện tượng chồng lên nhau giữa 2 hình nhật.
- Tuy nhiên, nếu vật chuyển động quá nhanh sẽ có hiện tượng không va chạm. SweptAABB ra đời, để khắc phục chuyện đó, đầu tiên ta sẽ vẽ thử 1 broadphase và xét va chạm bằng cách tính collisionTime - thời gian va chạm, giá trị collisionTime từ 0->1, bằng 0 là chuẩn bị va chạm, bằng 1 là vừa kết thúc va chạm, trong khoảng 0-> 1 là đang va chạm. Đồng thời ta phải xác định hướng va chạm.
- Phương thức SweptAABB được mô tả:

```
SweptAABB(  
    float ml, float mt, float mr, float mb,  
    float dx, float dy,  
    float sl, float st, float sr, float sb,  
    float &t, float &nx, float &ny)
```

ml, mt, mr, mb là kích thước của vật di chuyển

dx, dy là quãng đường đi được của vật di chuyển

sl, st, sr, sb là kích thước của vật tĩnh, vật đang được làm mốc để xét va chạm.

t là thời gian va chạm (collisionTime)

nx, ny dùng để xét hướng va chạm và chỉ va chạm theo Ox hoặc Oy, không đồng thời xảy ra cả 2.

Có 2 TH xảy ra:

- Nếu không va chạm: $nx = ny = 0$.
- Nếu có va chạm:
 - Xét trục Ox có 3 TH:
 - Nếu $nx > 0$ thì vật chuyển động va chạm bên phải vật tĩnh.
 - Nếu $nx < 0$ thì vật chuyển động va chạm bên trái vật tĩnh.
 - Nếu $nx = 0$ không va chạm theo trục Ox, chắc chắn va chạm theo trục Oy
 - Xét trục Oy có 3 TH:
 - Nếu $ny < 0$ thì vật chuyển động va chạm top của vật tĩnh.

Nếu $n_y > 0$ thì vật chuyển động chạm bottom của vật tĩnh.
 Nếu $n_y = 0$ thì không va chạm theo trục Oy, chắc chắn va chạm theo trục Ox.

3.1.3. Viewport (Camera)

- Bản chất camera trong game là một ma trận 4x4

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

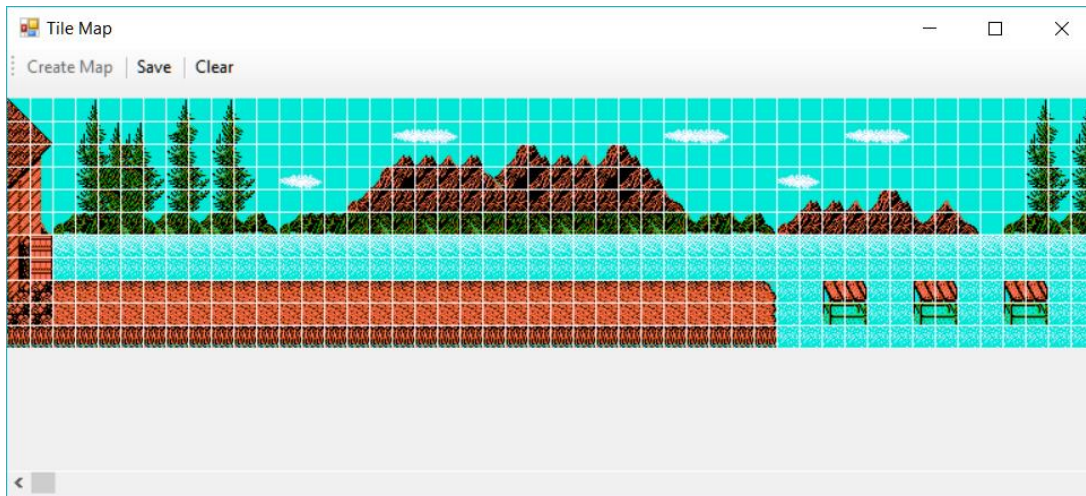
- Có nhiều phép biến đổi: scale, translate, rotate. Nhóm em chỉ dùng translate, cụ thể gán 3 tham số T_x , T_y , T_z . Tuy nhiên đây là game 2D, trục Z có thể bỏ qua.
- Sau mỗi lần di chuyển của nhân vật ta phải update lại camera, gán lại giá trị cho matrix transform.

3.1.4. Texture

- Dùng để đọc file hình ảnh và transparent color của hình ảnh.

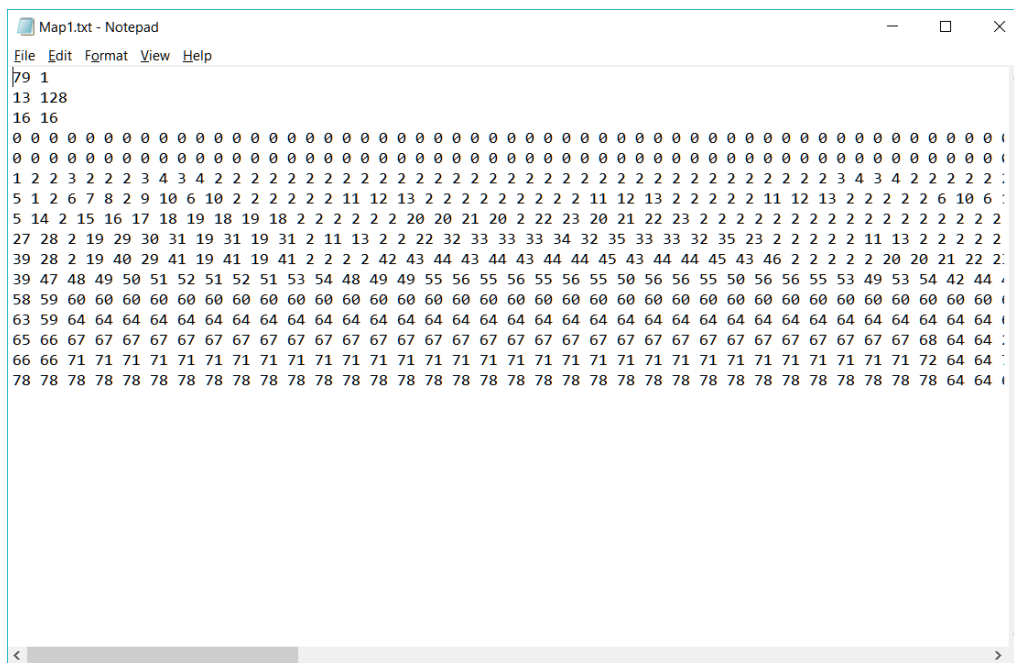
3.1.5. Tool Tile Map

- Theo yêu cầu giảng viên là viết 1 tool xuất ra 1 tấm ảnh tile map từ background. Nhóm em đã xây dựng 1 tool xuất 1 tấm ảnh tile map và 1 file.txt chứa tham số để vẽ map.



- Bấm save và chọn đường dẫn sẽ export ra 1 file.png, 1 file.txt





3.1.7. Tile Map

- Tile map là một kỹ thuật giúp chúng ta vẽ map những không cần phải vẽ hết cả tấm ảnh. Chúng ta chỉ việc update theo viewport, đi tới đâu vẽ tới đó, tiết kiệm chi phí cho GPU.
- Chúng ta định nghĩa 1 mảng 2 chiều trong ảnh tile map, kích thước 1 tile.
- Chúng ta xác định kích thước ô cửa sổ từ đó ta xác định vẽ n dòng, m cột. Tại vị trí dòng thứ i, cột thứ j là 1 tile.
- Ta dựa vào góc trái của camera, lấy vị trí top/tileWidth, left/tileHeight, từ đây ta được vị trí bắt đầu vẽ, ta duyệt mảng 2 chiều theo n dòng, m cột và vẽ map.

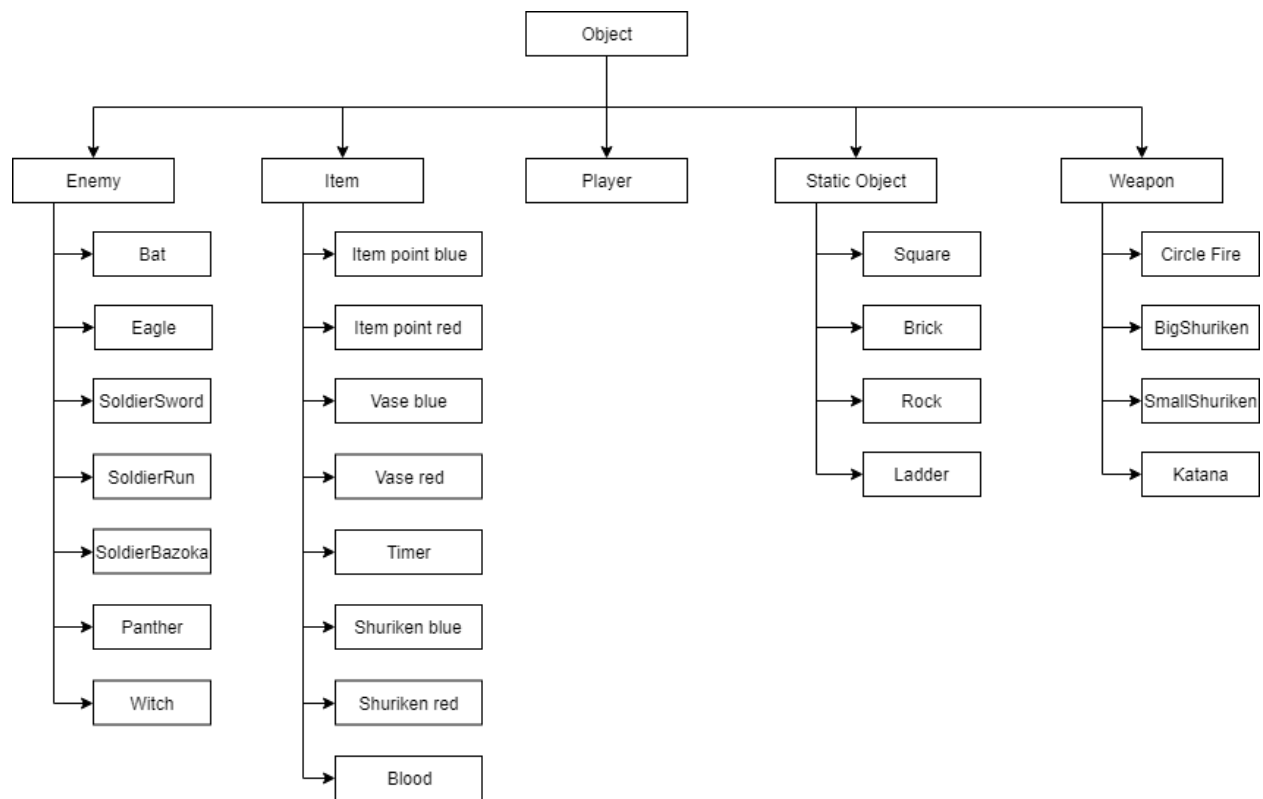
3.1.8. Thuật toán phân hoạch không gian Grid

- Xét và chạm đòi hỏi tốn nhiều tài nguyên, nếu các đối tượng quá xa nhân vật thì ta có thể bỏ xét và chạm để giảm chi phí CPU.
- Grid là một kỹ thuật dùng để phân hoạch không gian, đầu tiên ta xác định vùng nhìn đang overlap các cell nào của grid, từ đó ta chỉ xét và chạm các đối tượng trong các cell đó và bỏ qua các cell còn lại.
- Cách thực hiện:
 - Định nghĩa Cell:
 - Thuộc tính: dòng, cột, mảng object (dòng-cột là vị trí của cell)
 - Kích của 1 cell: cellHeight, cellWidth. Kích thước một cell không đổi.
 - Định nghĩa Grid:
 - Mảng 2 chiều Cell, số dòng, số cột
 - Đầu tiên ta khởi tạo Grid dựa trên kích thước map
 - Số dòng = $\text{ceil}(\text{mapHeight}/\text{cellHeight})$

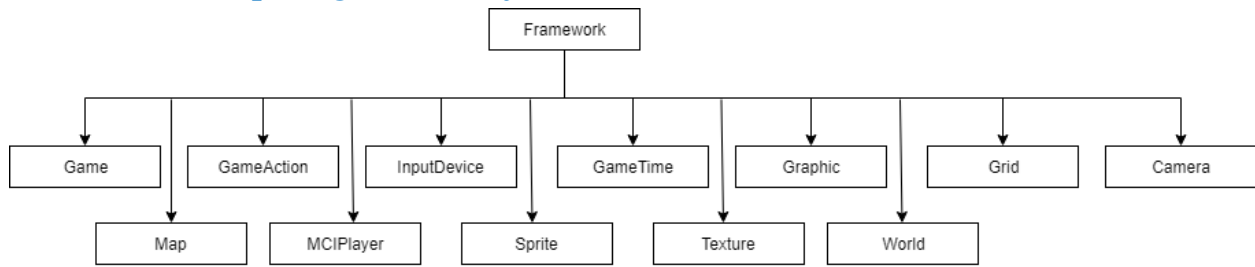
- Số cột = $\text{ceil}(\text{mapWidth}/\text{cellWidth})$
- Thêm object vào Grid:
 - Ta dựa vào vị trí của object là x,y để xác định Cell mà object này thuộc.
 - Ta có object này thuộc tại Cell có dòng = $\text{floor}(x/\text{cellWidth})$, cột = $\text{floor}(y/\text{cellHeight})$. Tuy nhiên ta có thể thêm object này tại cell ta muốn và vị trí có thể không thuộc cell đó.
- Cập nhật các cell:
 - Bước 1: lấy tọa độ góc trên bên trái ViewPort chia lấy nguyên cho kích thước mỗi Cell, ta nhận được cặp số thứ tự của Cell đầu tiên chứa ViewPort, tạm gọi (x1, y1).
 - Bước 2: lấy tọa độ góc dưới bên phải tương tự B1 ta có (x2, y2)
 - Bước 3: ta duyệt mảng 2 chiều từ dòng y1 -> y2, cột từ x1->x2. Lấy mảng object trong cell.
 - Bước 4: Kiểm tra object có nằm trong viewport, nếu nằm trong thì update và render, vì nếu kích thước cell lớn, có thể viewport chỉ chạm đc 1/2 của cell, các object không nằm trong viewport thì không update, render mặc dù nằm trong cell đó.
- Sau khi cập nhật các cell ta có 1 mảng object từ các cell, ta chỉ xét va chạm trong các object.

3.2. Sơ đồ các class chính trong game

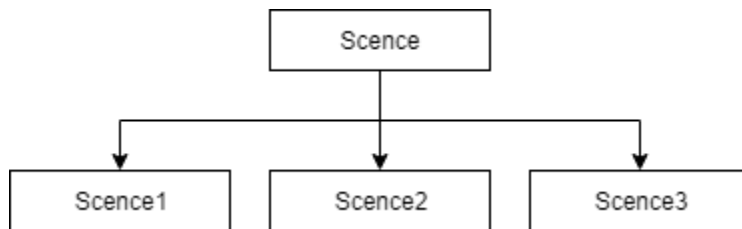
3.2.1. Sơ đồ mô phỏng các class object có trong game



3.2.2. Sơ đồ mô phỏng các class framework



3.2.2. Sơ đồ mô phỏng các class màn hình



3.2.2. Design pattern

- Ngoài việc áp dụng OOP, nhóm em đã sử dụng 1 pattern là singleton.
- Các class áp dụng pattern singleton: World, Camera, Grid, Player, Texture
- Việc áp dụng mẫu trên có lợi ích như:
 - Texture chỉ load lần đầu, khi khởi tạo object, map không cần phải đọc lại file, chỉ việc gọi các texture ra dùng.
 - Grid chỉ khởi tạo lần đầu, các lần sau khi chuyển scence chỉ việc gọi ra, không cần đọc file.
 - Khi chuyển scence, không cần khởi tạo lại Camera, chỉ việc cập nhật vị trí và tiếp tục.
 - World nơi khởi tạo từ các resource, chỉ khởi tạo lần đầu chạy, đồng thời quản lý các scence khi chuyển màn.
 - Player: giúp cho việc gọi player dễ dàng hơn. Đặc biệt các xử lý tương tác giữa player và object, môi trường.

3.3. Công cụ hỗ trợ

- Để phục vụ cho việc thực hiện đồ án, nhóm đã sử dụng các công cụ hỗ trợ như:
 - + Visual Studio 2017
 - + Tool tile map
 - + Paint

3.4. Kết quả thực hiện

3.4.1. Đánh giá mức độ hoàn thiện của Game

- Đánh giá mức độ hoàn thiện của Game: 85%
- Về cơ bản, nhóm chúng em cũng đã xây dựng nên một game tương đối hoàn chỉnh đáp ứng đủ các yêu cầu mà Thầy đã đặt ra:
 - + Xây dựng game ninja gaiden 1, hoàn thành stage 3-1, stage 3-2, stage 3-3.
 - + Xây dựng tool tile map.
 - + Áp dụng các kỹ thuật trong game: Swept AABB, grid, camera, sprite.
- Một số hạn chế của game :
 - + Chưa làm được logic Eagle giống trong game gốc, chỉ thực hiện được 70% về logic

3.4.2. Hướng phát triển (nếu có)

- Dành thêm thời gian để test và fix nếu có xảy ra lỗi tìm ẩn trong game
- Tối ưu hóa các thuật toán.
- Cải thiện việc quản lý các resource.
- Xây dựng thêm các chế độ chơi khác.

3.4.3. Bảng phân công công việc

Họ tên	MSSV	Tiến độ	Công việc được giao
Phan Quốc Hưng	15520288	100%	Nhóm trưởng, phân công công việc cho các thành viên
			Xây dựng framework
			Coding stage 3-1, stage 3-3.

			Coding logic các object: panther, SoldierSword, SoldierRun, Bat, Eagle, Player, boss.
Nguyễn Tấn Diệu	15520116	100%	Viết tool tile map
			Coding stage 3-2
			Xây dựng hệ thống quản lý Object (Bullet, Item).
			Quản lý grid
			Báo cáo file word.

3.5. Kết luận

- Nhập môn phát triển game được coi là một môn học tương đối khó với mỗi sinh viên học khoa Công nghệ phần mềm. Tuy nhiên, nhờ sự giảng dạy nhiệt tình từ Thầy Nguyễn Vĩnh Kha với những bài học hữu ích trên lớp đã giúp nhóm biết cách làm ra một game là như thế nào và tự làm thành một game hoàn chỉnh. Các kỹ thuật trong game như: kỹ thuật xử lý va chạm swept AABB, kỹ thuật phân hoạch không gian grid,...
- Thông qua môn học này, nhóm em đã hiểu rõ việc quan trọng trong thiết kế class hướng đối tượng, nếu thiết kế tệ thì code không được, gây ra nhiều lỗi ầm, xử lý khó khăn. Nhóm em cũng hiểu rõ cách xây dựng một game: vòng đời game, framework, các thuật toán kinh điển trong game, xử lý hình ảnh (rotate, translate, scale), xử lý input keyboard.

TÀI LIỆU THAM KHẢO

- [1]. <http://tranminhtuan11a1.blogspot.com/2014/06/conceit-space-partitioning-argorithms.html>
- [2]. https://www.youtube.com/watch?v=7HY_SqgaoL4
- [3]. <https://www.sprisers-resource.com/nes/ninjagaiden/>
- [4]. <https://nesmaps.com/maps/NinjaGaiden/sprites/NinjaGaidenSprites.html>
- [5]. <https://www.sounds-resource.com/nes/ninjagaidennes/>
- [6]. <https://www.uroborostudio.com/blog/aabb-collision/>