Shape, square

Description automatically generated

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**UNIVERSITY OF TRANSPOST AND COMMUNICATIONS**

Logo

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN**

**MÔN CÔNG NGHỆ JAVA**

***ĐỀ TÀI: Game MegaMan***

**Sinh viên thực hiện : Nhóm 22**

|  |  |
| --- | --- |
| **Họ tên sinh viên** | **MSV** |
| Đỗ Xuân Tùng | 211243774 |
| Mai Thế Công | 211214289 |

**Lớp : CNTT6-K62**

**Năm 2023**

Mục Lục

[**A.** **Tổng quan** 2](#_Toc134044486)

[**I.** **Giới thiệu về Game MegaMan** 2](#_Toc134044487)

[**II.** **Mô tả chung về code** 2](#_Toc134044488)

[**B.** **Nội dung chính** 2](#_Toc134044489)

[**I.** **Package Control** 2](#_Toc134044490)

[**1.** **Button** 2](#_Toc134044491)

[**2.** **RectangleButton** 2](#_Toc134044492)

[**II.** **Package Effect** 3](#_Toc134044493)

[**1.** **Animation** 3](#_Toc134044494)

[**2.** **CacheDataLoader** 3](#_Toc134044495)

[**3.** **FrameImage** 3](#_Toc134044496)

[**III.** **Package GameObject** 4](#_Toc134044497)

[**IV.** **Package State** 5](#_Toc134044498)

[**1.** **GameWorldState** 5](#_Toc134044499)

[**2.** **MenuState** 5](#_Toc134044500)

[**3.** **State** 5](#_Toc134044501)

[**V.** **Package UserInterface** 6](#_Toc134044502)

[**1.** **GameFrame** 6](#_Toc134044503)

[**2.** **GamePanel** 6](#_Toc134044504)

[**3.** **InputManager** 7](#_Toc134044505)

1. **Tổng quan**
2. **Giới thiệu về Game MegaMan**

Khi nhắc đến một dòng game gắn liền với tuổi thơ, chắc chắn sẽ có nhiều người nghĩ ngay đến **MegaMan**, hay còn được biết đến với cái tên quen thuộc hơn là RockMan. MegaMan chính là một trong những dòng game có tuổi đời lâu nhất của Capcom nói riêng và của cả ngành công nghiệp game nói chung. Khởi đầu từ năm 1987 trên hệ máy Nintendo Entertainment System, tính đến ngày 31 tháng 12 năm 2010, đã có tổng cộng trên 28 triệu bản được phát hành trên toàn thế giới và tính đến nay, MegaMan đã trở thành một thương hiệu lớn, một cái tên khá quen thuộc, sở hữu hàng chục tựa game lớn nhỏ khác nhau và xây dựng được một cộng đồng vững mạnh.

MegaMan là 1 tựa game đi cảnh nhập vai, người chơi sẽ hóa thân thành nhân vật Rock - là một người máy giống người được tạo ra trong phòng thí nghiệm bởi một nhà khoa học là Tiến sĩ Light (Dr. Light). Trước sự phản bội của Tiến sĩ Wily (Dr. Wily), Rock được biến đổi thành một người máy chiến đấu để bảo vệ thế giới khỏi sự đe dọa của đội quân độc ác của tiến sĩ Wily. Từ đây, Rock trở thành Mega Man - Cái tên gọi RockMan hay MegaMan cũng được bắt nguồn từ đây.

1. **Mô tả chung về project**

Trong project này chúng tôi chỉ tái hiện lại phần rất nhỏ trong game MegaMan với một bản đồ vật lý 50×156 là một ma trận nhị phân và chỉ xuất hiện 1 Final Boss. Trò chơi sẽ được hoàn thành khi MegaMan có thể đánh bại được Final Boss. Tuy nhiên bạn cũng sẽ thất bại khi sử dụng hết 3 mạng của MegaMan.

Tất cả các dữ liệu liên quan đến text, image, audio đều được lưu trong file data.

Trong project của chúng tôi quản lý theo hướng đối tượng và được chia thành 5 package bao gồm package Control, Effect, GameObject, State, UserInterface. Mỗi package sẽ có 1 chức năng, nhiệm vụ khác nhau:

1. **Control**: Quản lý việc điều khiển sau khi trò chơi kết thúc
2. **Effect**: Quản lý về các hiệu ứng diễn ra trong game
3. **GameObject**: Quản lý các đối tượng, các vật thể trong game
4. **State**: Quản lý các trạng thái của game
5. **UserInterface**: Quản lý giao diện người dùng liên quan đến hiển thị và lấy các sự kiện trên bàn phím
6. **Nội dung chính**
7. **Package Control**

Trong package Control gồm 2 class là Button và RectangleButton

1. **Button**

Đây là 1 lớp trừu tượng, nó dùng để khai báo thuộc tính và phương thức cho các lớp khác sử dụng.

1. **RectangleButton**

Lớp RectangleButton là lớp kế thừa từ lớp Button, quản lý tọa độ, màu sắc, phông chữ, kích cỡ cho 2 button NEW GAME và EXIT sau khi trò chơi kết thúc.

1. **Package Effect**

Trong package Effect gồm 3 class là Animation, CacheDataLoader và FrameImage

1. **Animation**

Animation được biết đến là nghệ thuật diễn hoạt hình ảnh có trong các chương trình quảng cáo, phim hoạt hình, các trò chơi điện tử… - đó là nghệ thuật làm cho hình ảnh được xuất hiện và chuyển động một cách sống động trên màn hình, dựa theo một câu chuyện, nội dung cụ thể nào đó. Cơ chế tạo chuyển động của Animation khá giống với phim điện ảnh, truyền hình, ở việc chuyển động được tạo thành bằng nhiều hình ảnh khác nhau, được nối tiếp nhau thành một chuỗi hình ảnh. Những hình ảnh ấy sẽ tạo thành ảo ảnh thị giác về chuyển động, khi nó được xâu chuỗi trong một khoảng thời gian nhất định.

Lớp Animation trong project có chức năng quản lý các đối tượng là các FrameImage. Các FrameImage được lưu trữ trong ArrayList tiện cho việc sử dụng để tạo các animation. Đặc biệt ở trong lớp animation có 1 phương thức flipAllImage giúp các hình ảnh được lật lại theo chiều ngang so với chiều vốn có ban đầu của nó. Thao tác lật ảnh gồm có 2 bước, đầu tiên chúng ta scale nó theo chiều ngược lại tức là tỉ lệ -1 sau đó tịnh tiến tâm vẽ lùi về bên trái với khoảng cách bằng với kích thước của ảnh. Sau khi đã định nghĩa thao tác lật ảnh trong đối tượng AffineTransform, chúng ta copy từng pixel trong ảnh vào một image, trong Java có sẵn lớp AffineTransformOp cho phép chúng ta làm điều này dễ dàng với phương thức filter ().

For (int i = 0; i < frameImages.size(); i++) {

BufferedImage image = frameImages.get(i). getImage ();

AffineTransform tx = AffineTransform.getScaleInstance(-1, 1);

tx.translate(-image.getWidth(), 0);

AffineTransformOp op = new AffineTransformOp (tx,

AffineTransformOp.TYPE\_BILINEAR);

image = op.filter(image, null);

frameImages.get (i). setImage(image);

1. **CacheDataLoader**

Lớp CacheDataLoader có chức năng tải lên các dữ liệu từ file data. Việc lưu các dữ liệu frame, animation, audioclip, … trong file giúp giảm nhẹ code và đảm bảo bảo mật thông tin dữ liệu có trong game. Trong lớp CacheDataLoader sử dụng Hashtable để lưu FrameImage, Animation và AudioClip. Hashtable lưu giữ các cặp key/value trong một hash table. Khi sử dụng một Hashtable, bạn xác định một đối tượng mà được sử dụng như là một key, và value mà bạn muốn liên kết tới key đó. Key này sau đó được hash, và hash code kết quả được sử dụng như là chỉ mục, tại đó value được lưu giữ bên trong table đó.

1. **FrameImage**

Lớp FrameImage mục đích được tạo ra nhằm để vẽ các bức ảnh nhỏ từ 1 bức ảnh lớn có trong file data. Từ nhiều bức ảnh nhỏ chạy trong một thời gian ngắn có thể tạo ra được 1 ảnh động (animation). Lớp FrameImage có chức năng quản lý đối tượng là các hình ảnh nhỏ và thuộc tính name giúp việc quản lý trở nên dễ dàng hơn.

1. **Package GameObject**

Trong package GameObject gồm 21 class BackgroundMap, BlueFire, Bullet, BulletManager, Camera, DarkrRaise, DarkRaiseBullet, FinalBoss, GameObject, Human, MegaMan, ParticularObject, ParticularObjectManager, PhysicalMap, RedEyeDevil, RedEyeBullet, RobotR, RobotRBullet, RocketBullet, SmallRedGun, YellowFlowerBullet sử dụng mô hình hướng đối tượng để có thể tái sử dụng code. Trong đó GameObject là lớp cơ bản của mọi đối tượng trong game. Từ GameObject sẽ có 2 nhánh, một nhánh là các đối tượng ở dạng vĩ mô như BackGroundMap, PhysicalMap, Camera và một nhánh còn lại là các đối tượng cụ thể ParticularObject. Trong ParticularObject sẽ đi đến trực tiếp các lớp như Bullet, Human và các lớp của quái vật. Lớp MegaMan sẽ kế thừa từ lớp Human. Các lớp khác thuộc về các loại đạn như BlueFire, DarkRiseBullet, … sẽ kế thừa từ lớp Bullet.

Đối với các vật thể như MegaMan, viên đạn, quái vật ta sẽ sử dụng các hình chữ nhật để biểu diễn sự va chạm, tiếp xúc giữa các vật thể.

Nhân vật MegaMan sẽ bị mất máu khi va chạm với quái vật hoặc va chạm với đạn từ quái vật. Khi vừa va chạm với phe địch, MegaMan sẽ tiến vào trạng thái NoBeHurt (không bị dính sát thương) trong vòng 2 giây. Đối với việc va chạm của MegaMan với tường sẽ khiến tốc độ của MegaMan bị dừng lại. Việc MeganMan bị rơi tự do xuống khi đang ở trên không trung hay bị chạm trần nhờ vào việc giảm tốc độ theo phương Y.

Đối với lớp Bullet, khi viên đạn va chạm với đối tượng khác null hoặc ở trạng thái ALIVE, đặt lại lượng máu cho viên đạn bằng 0 và vật thể va chạm với viên đạn sẽ nhận 1 lượng sát thương tùy theo mỗi loại đạn:

ParticularObject object = getGameWorld(). particularObjectManager. getCollisionWidthEnemyObject (this);

if (object! = null && object.getState() == ALIVE){

setBlood (0);

object.beHurt(getDamage());

System.out.println("Bullet set behurt for enemy");

}

Đối với lớp Camera, Camera là một đối tượng luôn đi theo MegaMan và khi gặp FinalBoss, Camera mới không bị thay đổi vị trí (bị khóa). Ngoài ra sau khi hiển thị các đoạn hội thoại Camera cũng sẽ bị thay đổi tọa độ mới. Việc vẽ các hình ảnh có trong Game cũng chỉ vẽ tất cả các hình ảnh ở trong khu vực Camera chứ không vẽ hết toàn bộ Image trong Game để tránh tình trạng Lag do phải vẽ quá nhiều hình ảnh.

Đối với các quái vật, DarkRaise là 1 quái nhỏ nhưng có độ thú vị khá lớn do đạn của DarkRise có khả năng thay đổi đuổi theo MegaMan.

float megaManX = getGameWorld().megaMan.getPosX();

float megaManY = getGameWorld().megaMan.getPosY();

// tính khoảng cách giữa người chơi và quái

float deltaX = megaManX - getPosX();

float deltaY = megaManY - getPosY();

// xác định tốc độ đạn

float speed = 3;

float a = Math.abs(deltaX/deltaY);

// thay đổi vận tốc 2 phương X và Y của đạn theo hướng MegaMan

float speedX = (float) Math.sqrt(speed\*speed\*a\*a/ (a\*a + 1));

float speedY = (float) Math.sqrt(speed\*speed/ (a\*a + 1));

// bắn đạn theo hướng xác định

Bullet bullet = new DarkRaiseBullet(getPosX(), getPosY(), getGameWorld());

If (deltaX < 0)

bullet.setSpeedX(-speedX);

else bullet.setSpeedX(speedX);

bullet.setSpeedY(speedY);

bullet.setTeamType(getTeamType());

getGameWorld(). bulletManager.addObject(bullet);

1. **Package State**

Trong package State gồm 3 class là GameWorldState, MenuState và State

1. **GameWorldState**

Lớp GameWorldState kế thừa từ lớp State. Lớp này quản lý trạng thái của tất cả các object bao gồm PhysicalMap, BackGroundMap, Camera, MegaMan, BulletManager, ParticularObjectManager và cả phần hướng dẫn (tutorial).

Việc quản lý trạng thái của ParticularObject ở đây bao gồm tọa độ (x,y), hướng quay (trái hoặc phải), phe (đối địch hay đồng minh), ….

Đối với MegaMan việc quản lý trạng thái bao gồm tất cả các hành động như ngồi, chạy, nhảy, tấn công hay các trạng thái của nhân vật như ALIVE, DEATH, NOBEHURT

Ngoài ra còn có các quản lý khác liên quan đến việc bảng hội thoại xuất hiện khi bắt đầu trò chơi, việc camera thay đổi khi xuất hiện hội thoại hay xuất hiện boss cũng nằm trong phần quản lý của lớp GameWorldState.

1. **MenuState**

Lớp MenuState là 1 lớp kế thừa(extends) từ lớp State. Lớp này có chức năng quản lý bảng thông báo cuối cùng sau khi kết thúc trò chơi. Ở bảng thông báo cuối cùng sẽ có 2 button là New Game và Exit. Người chơi sẽ lựa chọn 1 trong 2 button đó. Nếu lựa chọn New Game thì một trò chơi mới sẽ bắt đầu thay vào đó khi lựa chọn Exit sẽ thoát trò chơi tương tự EXIT\_ON\_CLOSE của GameFrame. Việc lựa chọn có thể sử dụng bằng bàn phím nhờ nút di chuyển lên (VK\_UP) và di chuyển xuống (VK\_DOWN) và nút ENTER thay cho click chuột.

1. **State**

Lớp State là 1 lớp trừu tượng, nó dùng để khai báo thuộc tính và phương thức cho các lớp khác sử dụng.

1. **Package UserInterface**

Trong package UserInterface gồm 3 class là GameFrame, GamePanel và InputManager

1. **GameFrame**

Lớp GameFrame được kế thừa (extends) từ lớp JFrame giúp tương thích giao diện với các hệ điều hành khác nhau. Lớp GameFrame có chức năng tạo 1 khung chứa cho game. Giá trị chiều rộng và chiều cao của khung là một hằng số được cố định bằng static final có giá trị lần lượt là 1000pixel và 600pixel.

Frame sẽ được đặt ở chính giữa màn hình tùy theo kích cỡ của màn hình nhờ vào hàm setBounds:

this.setBounds((solution.width - SCREEN\_WIDTH)/2, (solution.height - SCREEN\_HEIGHT)/2, SCREEN\_WIDTH, SCREEN\_HEIGHT);

Trong đó SCREEN\_WIDTH và SCREEN\_HEIGHT là giá trị cố định chiều rộng và chiều cao của Frame. Còn solution.width và solution.height là kết quả có được từ việc tính chiều rộng và chiều cao của thiết bị nhờ hàm Toolkit:

Toolkit toolkit = this.getToolkit();

Dimension solution = toolkit.getScreenSize();

1. **GamePanel**

Lớp GamePanel được kế thừa (extends) JPanel có chức năng là vùng để vẽ game MegaMan, và cũng là 1 thread chứa các chức năng trong game như game loop hay input của game.

Để cho game mượt hơn, tránh bị giật lag, vòng lặp (game loop) của game cần có sự đồng đều về chu kì. Chính vì đó, chúng tôi đã sử dụng:

long previousTime = System.nanoTime();

long currentTime;

long sleepTime;

long period = 1000000000/80;

while(isRunning) {

gameState.Update();

gameState.Render();

repaint ();

currentTime = System.nanoTime();

sleepTime = period - (currentTime - previousTime);

try {

if (sleepTime > 0)

Thread.sleep(sleepTime/1000000);

else Thread.sleep(period/2000000);

} catch (Exception e) {}

previousTime = System.nanoTime();

}

}

Trong đó thuộc tính previousTime là thời gian bắt đầu sẽ được gán bằng giá trị thời gian hiện tại của hệ thống bằng hàm System.nanoTime() đổi sang đơn vị ns. Thuộc tính period là chu kì được tính bằng công thức 1000000000(ns) / 80(FPS). Thuộc tính currentTime là thời gian của hệ thống sau khi thực hiện sau 2 quá trình Update và Render. Thuộc tính sleepTime được tính bằng công thức: Chu kì(period) – thời gian thực thi 2 quá trình Update và Render (currentTime – previousTime). Nếu sleepTime > 0, hệ thống sẽ ngủ (sleep) theo sleepTime. Ngược lại nếu sleepTime < 0, ta để hệ thống ngủ trong thời gian bằng chu kì(period) / 2000000 (ms).

Ngoài ra GamePanel còn được implements Runnable, KeyListener để lắng nghe sự kiện từ bàn phím. Vì KeyListener là 1 interface nên khi chúng ta implements, ta buộc phải implements tất cả các phương thức trừu tượng (abstract methods) của nó.

1. **InputManager**

Lớp InputManager có chức năng quản lý việc nhận thông tin đầu vào của game để các class khác xử lý.