**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI  
UNIVERSITY OF TRANSPOST AND COMMUNICATIONS**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN CÔNG NGHỆ JAVA**

**ĐỀ TÀI: Plane Game**

**Giảng viên hướng dẫn : Nguyễn Quốc Huấn**

**Lớp : Công nghệ thông tin 6 – K62**

**Nhóm : 15**

**Thành viên nhóm :**

**Chu Văn Dũng (nhóm trưởng)**

**Vũ Quang Trường(thành viên)**

**Hà Nội 2023**

**Lời nói đầu**

Trong thời đại công nghệ ngày nay, game là một trong những ứng dụng được sử dụng phổ biến nhất. Java là một trong những ngôn ngữ lập trình phổ biến nhất trong việc phát triển game, cho phép lập trình viên xây dựng game với tính năng đa nền tảng và có thể chạy trên nhiều hệ điều hành khác nhau.

Trong báo cáo của tôi hôm nay, tôi sẽ giới thiệu về công dụng của game trong thế giới hiện đại của chúng ta. Game không chỉ đơn thuần là một thứ giải trí, mà còn có nhiều ứng dụng khác trong nhiều lĩnh vực khác nhau.

Việc phát triển game cũng đã trở thành một ngành nghề phát triển nhanh chóng, với nhiều công ty lớn và nhỏ đang hoạt động trong lĩnh vực này. Java là một trong những ngôn ngữ lập trình phổ biến được sử dụng trong việc phát triển game, đặc biệt là các game trên điện thoại di động và máy tính bảng.

Việc sử dụng Java để phát triển game còn mang lại một số lợi ích khác như tính di động cao, khả năng tương thích trên nhiều nền tảng khác nhau, đồng thời Java cũng hỗ trợ quản lý bộ nhớ tự động và có các tính năng bảo mật cao.

Mục lục

1. Tổng quan về đề tài4
2. Giới thiệu đề tài4
3. Chức năng của chương trình 4
4. Phân tích thiết kế chương trình4
5. Cài đặt chương trình5
6. Kết luận và đánh giá18
7. Kết quả đạt được18
8. Hạn chế 18
9. Hướng phát triển18

**I. Tổng quan về đề tài**

**1. Giới thiệu đề tài**

Game đã trở thành một phần không thể thiếu của ngành công nghiệp giải trí, với số lượng người chơi ngày càng tăng đáng kể. Đồng thời, game cũng có khả năng giáo dục và truyền tải thông điệp cho người chơi thông qua các nội dung và câu chuyện phong phú.

Bên cạnh đó, game cũng được sử dụng trong các lĩnh vực như giáo dục, y tế và quân đội. Các ứng dụng của game trong lĩnh vực này giúp cải thiện hiệu quả giáo dục, nghiên cứu y học và huấn luyện quân sự.

Trò chơi bắn (hay còn gọi là Chicken Invaders) được tạo ra bởi công ty InterAction Studios, một công ty độc lập được thành lập bởi công dân Hy Lạp Rod Athanasios Kostas vào năm 1992. Kể từ khi phát hành phiên bản đầu tiên vào năm 1999, game đã trở thành một trong những trò chơi nổi tiếng nhất trên PC.

Mục tiêu của trò chơi bắn gà thường là để người chơi tiêu diệt các con gà bằng cách bắn chúng bằng súng hoặc vật phẩm tương tự. Người chơi sẽ phải né tránh các đối tượng trên màn hình để không bị trừ điểm hoặc mất mạng. Mục tiêu chính của trò chơi bắn gà là giải trí và thú vị, và cũng có thể được sử dụng để rèn luyện kỹ năng tập trung và phản xạ của người chơi.

**2. Chức năng của chương trình: chức năng thực hiện trò chơi**

Trong trò chơi bắn gà, người chơi sẽ điều khiển một chiếc tàu bay và tiêu diệt các con gà đang bay trên không trung bằng cách bắn chúng. Người chơi sẽ được cộng điểm khi tiêu diệt được các con gà và mất máu nếu va chạm vào vật cản.

Bảng điểm ở góc trên màn hình cho biết điểm của người chơi. Ban đầu điểm bằng 0. Khi máy bay hạ được 1 con gà thì sẽ tăng 1 điểm

**II. Phân tích thiết kết chương trình**

**Quá trình thiết kế chương trình:**

Bước 1: Xác định yêu cầu chương trình: Đây là bước đầu tiên và nhóm em thấy đó cũng là bước quan trọng nhất trong quá trình thiết kế chương trình. Nên chúng em đã cố gắng xác định rõ yêu cầu chương trình, bao gồm tính năng chính, đối tượng người chơi, cách chơi và các yêu cầu kỹ thuật khác.

Bước 2: Lập trình: Sau khi đã có bản thiết kế, bước tiếp theo đó là thực hiện lập trình để tạo ra các chức năng và tính năng cho chương trình.

Bước 3: Kiểm thử và sửa lỗi: Kiểm tra và thử nghiệm chương trình, sửa các lỗi phát sinh trong quá trình thử nghiệm.

**Các đối tượng:**

Bullet: để xác định tọa độ, hình dạng, màu sắc, kích thước và tốc độ của đối tượng viên đạn và cung cấp các phương thức để cập nhật vị trí của viên đạn, kiểm tra xem nó có di chuyển ra khỏi màn hình hay không, vẽ viên đạn lên màn hình và truy xuất các thuộc tính của viên đạn như tọa độ và kích thước.

Chicken: để lưu trữ thông tin về một con gà trong trò chơi, bao gồm vị trí, hình dạng và hình ảnh.

Hp: để lưu trữ thông tin về máu hiện tại và máu tối đa của một đối tượng trong game. Mục đích của class này là để quản lý và cập nhật trạng thái máu của đối tượng đó khi bị tấn công.

HPRender: để vẽ thanh máu cho nhân vật trong trò chơi và ngoài ra nó cũng để lưu trữ điểm máu cho nhân vật.

Player: Mục đích để định nghĩa người chơi trong game bao gồm Các thuộc tính như vị trí của người chơi trên màn hình, tốc độ di chuyển, góc xoay, hình dáng của người chơi và hình ảnh của người chơi và các phương thức như cập nhật vị trí của người chơi dựa trên tốc độ và góc xoay hiện tại, vẽ người chơi trên màn hình và tính toán hình dáng của người chơi dựa trên vị trí và góc xoay hiện tại của nó.

Sound: được sử dụng để phát nhạc nền và âm thanh khi người chơi thực hiện các hành động như bắn, va chạm, hủy bỏ hoặc hiển thị âm thanh cụ thể của trò chơi.

**III. Cài đặt chương trình**

Hướng dẫn cài đặt:

**Tạo class Main kế thừa lớp Jframe:**

public class Main extends JFrame{

public Main()

{

init();

}

private void init()

{

setTitle("ShootingPlane!!");

setSize(1366, 768);

setLocationRelativeTo(null); // Đặt giữa màn hình

setResizable(false); // Không cho resize window bằng mouse

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

}

public static void main(String[] args) {

Main main = new Main();

main.setVisible(true);

}

}

Tạo package game.component chứa:

Class PanelGame kế thừa lớp JComponent đưa vào lớp JFrame bằng phương thức add trong class Main:

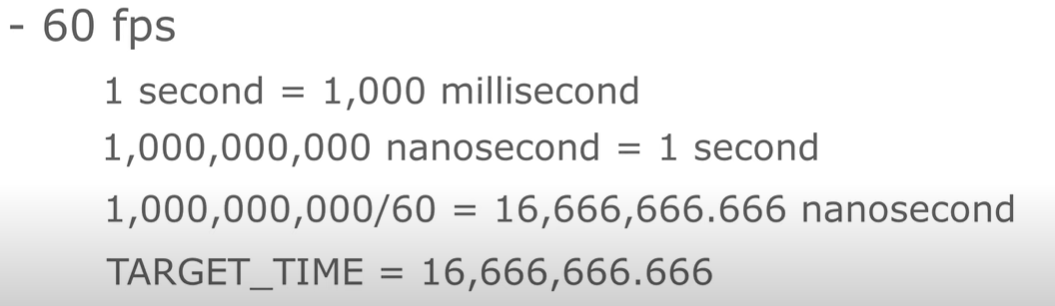
PanelGame panelGame = new PanelGame();

add(panelGame);

**Tạo gameFPS trong class panel:**

Trong trò chơi này, FPS = 60

Tạo biến FPS = 60, TARGET\_TIME = 1000000000 / FPS; (1s = 1e9 nano\_s)



Tạo các biến int width, height để lưu size của JComponent.

Tạo biến thread để chia các quá trình vòng lặp game và để render graphics.

Khởi tạo biến Boolean start = true;

Tạo phương thức public start() để bắt đầu trò chơi:

Tạo thread object với runnable. Trong phương thức run ta khởi tạo vòng lặp While để xử lý hình ảnh game

Tạo phương thức sleep để vòng lặp ngủ. được định nghĩa với khai báo private nghĩa là nó chỉ được truy cập trong phạm vi của lớp hiện tại. Tên phương thức là sleep, nhận đầu vào là một tham số kiểu long với tên là speed, đại diện cho thời gian ngủ của thread.

Trong thân phương thức, chúng ta sử dụng lệnh Thread.sleep(speed) để cho thread đang chạy ngủ trong speed millisecond (ms). Để bắt các ngoại lệ có thể xảy ra trong lúc thread đang ngủ, chúng ta sử dụng khối try-catch để bắt và xử lý ngoại lệ InterruptedException. Khi ngoại lệ này xảy ra, phương thức sẽ in ra thông báo lỗi trong console với câu lệnh System.err.println(ex).

Tạo 3 phương thức:

private void drawBackground()

private void drawGame()

private void render()

Gọi các phương thức trong vòng lặp While

long startTime = System.nanoTime();

drawBackground();

drawGame();

render();

long time = System.nanoTime() - startTime;

if (time < Target\_TIME) {

long sleep = (Target\_TIME - time) / 1000000;

sleep(sleep);

}

Dòng đầu tiên khởi tạo biến startTime với giá trị là thời điểm hiện tại được lấy bằng hàm System.nanoTime(). Hàm này trả về thời điểm hiện tại tính bằng nanosecond (1/1,000,000,000 giây).

Tiếp theo, trong vòng lặp game, ta thực hiện các thao tác drawBackground(), drawGame() và render() để vẽ nền, vẽ game và hiển thị lên màn hình.

Sau đó, ta tính thời gian đã trôi qua bằng cách lấy thời điểm hiện tại trừ đi thời điểm khởi tạo biến startTime.

Sau đó, ta so sánh thời gian này với thời gian mục tiêu (được định nghĩa trước đó với biến Target\_TIME) để xác định thời gian ngủ của thread.

Nếu thời gian đã trôi qua nhỏ hơn thời gian mục tiêu, ta tính thời gian ngủ cần thiết bằng cách lấy hiệu của thời gian mục tiêu với thời gian đã trôi qua, chia cho 1,000,000 để chuyển đổi thành đơn vị millisecond. Kết quả này sẽ được gán cho biến sleep.

Cuối cùng, ta gọi hàm sleep(sleep) để ngủ thread trong khoảng thời gian sleep tính bằng millisecond. Hàm sleep() đã được định nghĩa trước đó trong đoạn mã nguồn, nhận đầu vào là thời gian ngủ được truyền vào với đơn vị millisecond.

**Tạo đồ họa, màu nền:**

Thêm vào phương thức drawBackground():

g2.setColor(new Color(30, 30, 30));

g2.fillRect(0, 0, width, height);

Thêm vào phương thức render():

Graphics g = getGraphics();

g.drawImage(image, 0, 0, null);

g.dispose();

Đầu tiên, đoạn mã sử dụng phương thức getGraphics() để lấy đối tượng Graphics từ JFrame, đây là đối tượng được sử dụng để vẽ các thành phần trên JFrame.

Tiếp theo, đoạn mã sử dụng phương thức drawImage() của đối tượng Graphics để vẽ hình ảnh đã được xử lý lên JFrame. Tham số thứ nhất là hình ảnh cần vẽ, tham số thứ hai và ba là vị trí x, y của hình ảnh trên JFrame, và tham số cuối cùng là đối tượng ImageObserver, được sử dụng để thông báo cho ứng dụng rằng hình ảnh đã được vẽ lên đối tượng Graphics. Trong trường hợp này, giá trị null được truyền vào đối số này để không sử dụng ImageObserver.

Cuối cùng, đoạn mã sử dụng phương thức dispose() của đối tượng Graphics để giải phóng tài nguyên của nó khi việc vẽ đã hoàn tất.

**Tạo player máy bay và cho máy bay xoay theo keyboard**

Tạo class Player:

public static final double PLAYER\_SIZE = 64;

private double x, y;

private final float MAX\_SPEED = 1f;

private float speed = 0f;

private float angle = 0f;

Tạo phương thức public void changeAngle(float angle). Phương thức changeAngle(float angle) được dùng để thay đổi giá trị góc của đối tượng. Đối số angle được truyền vào phương thức đại diện cho giá trị góc mới cần thiết.

Trước khi gán giá trị góc mới vào biến angle, đoạn mã kiểm tra nếu giá trị angle nhỏ hơn 0 thì gán nó bằng 359 (tức giá trị góc lớn nhất có thể có), ngược lại nếu giá trị angle lớn hơn 359 thì gán nó bằng 0 (tức giá trị góc nhỏ nhất có thể có). Điều này đảm bảo rằng giá trị góc luôn nằm trong khoảng từ 0 đến 359 độ.

Cuối cùng, giá trị angle đã được xử lý được gán vào biến this.angle, để thực hiện thay đổi góc của đối tượng.

Tạo phương thức draw():

public void draw(Graphics2D g2)

{

AffineTransform oldTransform = g2.getTransform();

g2.translate(x, y);

AffineTransform tran = new AffineTransform();

tran.rotate(Math.toRadians(angle + 90), PLAYER\_SIZE/2, PLAYER\_SIZE/2);

g2.drawImage(image, tran, null);

g2.setTransform(oldTransform);

}

Trước tiên, một đối tượng AffineTransform mới được tạo ra để lưu trữ phép biến đổi (trong trường hợp này là phép xoay). Đối tượng này được đặt tên là tran.

Sau đó, phép biến đổi xoay được áp dụng trên đối tượng tran bằng cách sử dụng phương thức rotate(). Giá trị góc của phép xoay được tính bằng cách cộng thêm 90 độ vào giá trị góc ban đầu angle (vì đối tượng được vẽ ban đầu đã được xoay trước đó 90 độ), và tính toán tọa độ của tâm quay là (PLAYER\_SIZE/2, PLAYER\_SIZE/2).

Sau khi đã áp dụng phép biến đổi xoay lên đối tượng tran, hình ảnh được vẽ lên màn hình bằng cách sử dụng phương thức drawImage() của đối tượng g2 (một đối tượng Graphics2D đã được tạo trước đó). Đối số thứ hai của phương thức drawImage() là đối tượng tran (chứa phép biến đổi xoay), để vẽ đối tượng đồ họa với phép xoay đã được áp dụng.

Xây dựng phương thức: public void changeLocation(double x, double y) để thay đổi tọa độ player

Truy cập class PanelGame:

Tạo phương thức private void initObjectGame() để khởi tạo đối tượng player. Gọi phương thức initObjectGame() trong hàm start() của PanelGame.

Truy cập package game.component tạo class Key để lưu các keyboard actions. Trong đo khởi tạo các giá trị Boolean key\_left, key\_right, key\_j, key\_space.

Trong class PanelGame khởi tạo thuộc tính private Key key và phương thức initKeyboard().

khởi tạo một đối tượng Key để lưu trữ trạng thái của các phím được bấm trên bàn phím. Sau đó, nó yêu cầu trung tâm tập trung (focus) của các sự kiện bàn phím bằng cách gọi requestFocus(), và gắn một KeyListener vào component hiện tại để theo dõi sự kiện phím được bấm và phím được thả.

Trong keyPressed(), nếu người dùng bấm phím A, D, Space, J hoặc Enter, nó sẽ gọi phương thức tương ứng của đối tượng Key với tham số là true để đánh dấu phím đã được bấm.

Trong keyReleased(), nếu người dùng thả phím A, D, Space, J hoặc Enter, nó sẽ gọi phương thức tương ứng của đối tượng Key với tham số là false để đánh dấu phím đã được thả.

Tạo một thread mới trong hàm initKeyboard() với một vòng lặp while được thực thi trong khi giá trị biến start là true. Trong vòng lặp này, đoạn mã kiểm tra các phím được nhấn trên bàn phím để thay đổi góc quay của đối tượng player.

Nếu phím "left" được nhấn (key.isKey\_left() == true), góc quay sẽ giảm đi một giá trị s. Ngược lại, nếu phím "right" được nhấn (key.isKey\_right() == true), góc quay sẽ tăng lên một giá trị s. Sau đó, phương thức changeAngle() được gọi để thay đổi góc quay của đối tượng player thành góc được tính toán mới.

Gọi hàm initKeyboard trong hàm start của class PanelGame.

**Tạo biến Player có thể di chuyển và bắn đạn**

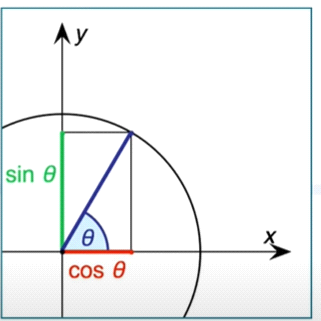
* **Di chuyển**

Thêm biến MAX\_SPEED = 1f, speed = 0f, biến Boolean speedUp.

Trong class Player, tạo các phương thức speedUp(), speedDown().

Trong class PanelGame thì ta check xem phím space có được bấm không:

if(key.isKey\_space()) { player.speedUp(); } else { player.speedDown(); }

Nếu người dùng đã nhấn phím space, phương thức "speedUp()" của đối tượng "player" sẽ được gọi để tăng tốc độ của người chơi. Nếu không, phương thức "speedDown()" của đối tượng "player" sẽ được gọi để giảm tốc độ của người chơi.

Trong class Player tạo phương thức update() để máy bay di chuyển

public void update()

{

x += Math.cos(Math.toRadians(angle))\*speed;

y += Math.sin(Math.toRadians(angle))\*speed;

}

Gọi phương thức update của player trong PanelGame.

* **Bắn đạn:**

Tạo class Bullet các thuộc tính của lớp này:

private double x, y: Đây là tọa độ của đối tượng đạn trên trục x và trục y trong không gian 2 chiều.

private final Shape shape: Đây là hình dạng của đối tượng đạn được định nghĩa trong lớp Shape, một lớp đối tượng của Java để biểu diễn các hình dạng hình học.

private final Color color = new Color(255, 255, 255): Đây là màu sắc của đối tượng đạn, được định nghĩa dưới dạng một đối tượng Color. Trong trường hợp này, đối tượng đạn có màu trắng (#FFFFFF).

private final float angle: Đây là góc quay của đối tượng đạn trong đơn vị độ (degree).

private double size: Đây là kích thước của đối tượng đạn, được định nghĩa dưới dạng một số thực.

private float speed: Đây là tốc độ di chuyển của đối tượng đạn, được định nghĩa dưới dạng một số thực.

Tạo hàm constructor tương ứng:

Trong constructor này, đối tượng "Bullet" được khởi tạo với các giá trị tương ứng của các thuộc tính của nó.

Cụ thể, các bước thực hiện của constructor là:

Đầu tiên, tọa độ x, y của đối tượng đạn được điều chỉnh sao cho nó xuất hiện giữa trung tâm của player bằng cách thêm một nửa kích thước của player và trừ đi một nửa kích thước của đạn. Điều này đảm bảo rằng đạn xuất hiện ở giữa trung tâm của player.

Sau đó, các thuộc tính angle, size, speed được gán với các giá trị tương ứng được truyền vào hàm khởi tạo.

Tiếp theo, thuộc tính shape được khởi tạo là một hình elip có kích thước là size x size.

Cuối cùng, tọa độ x, y được gán cho thuộc tính x, y của đối tượng "Bullet".

Constructor này được sử dụng để tạo ra một đối tượng "Bullet" mới với các thuộc tính được khởi tạo sẵn khi một đối tượng "Bullet" được tạo ra.

Tạo các phương thức draw(), update(), các hàm get, set tương tự class Player, phương thức check để kiểm tra viên đạn có bị bắn ra ngoài không – return false nếu bắn ra ngoài, ngược lại return true.

Truy cập class PanelGame:

+ Khởi tạo List các bullets, và biến shortTime để lưu khoảng thời gian khi người chơi nhấn phím J để bắn.

+ Tạo phương thức initBullets() Phương thức initBullets() được sử dụng để khởi tạo một ArrayList có tên là bullets và tạo một Thread mới để cập nhật trạng thái của các đối tượng đạn trong bullets.

Cụ thể, trong thread, vòng lặp while(start) sẽ chạy liên tục để cập nhật các thuộc tính của đối tượng đạn và kiểm tra xem nó có chạm vào các vật thể khác trong trò chơi hay không.

Nếu một đối tượng đạn đã chạm vào một vật thể khác, phương thức checkBullets() sẽ được gọi để xử lý va chạm này.

Nếu một đối tượng đạn đã di chuyển ra khỏi khu vực chơi game, nó sẽ bị loại bỏ khỏi danh sách bullets.

Cuối cùng, phương thức sleep() được sử dụng để tạm dừng thread trong 1ms trước khi tiếp tục vòng lặp.

Phương thức initBullets() này được gọi trong hàm khởi tạo của class, đảm bảo rằng danh sách bullets được khởi tạo ngay từ đầu và thread được bắt đầu chạy ngay khi game được bắt đầu.

Trong phương thức initKeyBoard thêm dòng code kiểm tra phím J có được bấm không. Nếu người chơi ấn phím "j" trên bàn phím, điều kiện key.isKey\_j() sẽ trả về true, và chương trình sẽ kiểm tra giá trị của biến shortTime. Nếu shortTime bằng 0, đoạn mã sẽ thực hiện việc thêm một đối tượng đạn mới vào danh sách bullets, với các thông số được thiết lập như tọa độ và góc quay của người chơi, kích thước và tốc độ của đạn.

Biến shortTime được tăng lên một đơn vị, và nếu nó bằng 15, nó sẽ được đặt lại về 0 để đảm bảo rằng người chơi không thể liên tục bắn đạn.

Nếu người chơi không ấn phím "j", shortTime sẽ được đặt lại về 0 để đảm bảo rằng người chơi không thể bắn đạn khi không cần thiết.

Ta vẽ đạn trong phương thức draw của PanelGame:

for(int i = 0; i < bullets.size(); ++i)

{

Bullet bullet = bullets.get(i);

if(bullet != null) bullet.draw(g2);

}

Đây là một vòng lặp để vẽ tất cả các đối tượng đạn trong danh sách bullets lên màn hình. Vòng lặp chạy từ đầu đến cuối danh sách bullets bằng cách sử dụng biến đếm i.

Đối với mỗi phần tử bullet trong danh sách, điều kiện bullet != null được sử dụng để đảm bảo rằng phần tử đó không bị xóa khỏi danh sách bên trong vòng lặp while của phương thức initBullets(). Sau đó, phương thức draw() được gọi với đối số g2 để vẽ đối tượng đạn lên màn hình.

Đoạn mã này được sử dụng để vẽ tất cả các đối tượng đạn trong danh sách bullets lên màn hình trong game.

**Tạo các chướng ngại vật gà và khi đạn bắn vào thì gà biến mất**

Tạo class PanelGame với các thuộc tính:

image: Đối tượng Image được tạo bằng cách tải hình ảnh của gà từ tệp tin chicken.png trong thư mục image.

CHICKEN\_SIZE: Một hằng số double đại diện cho kích thước của gà.

x, y: Vị trí hiện tại của gà trên màn hình.

speed: Tốc độ di chuyển của gà.

angle: Góc xoay của gà tính theo độ.

chickenShap: Một đối tượng Area để lưu trữ hình dạng của đối tượng gà.

Trong phương thức khởi tạo, hình ảnh gà được tải và đối tượng Path2D được sử dụng để xác định hình dạng của đối tượng gà. Sau đó, đối tượng Area được tạo ra để lưu trữ hình dạng của gà.

Tạo các phương thức changeLocation(), update(), changeAngle(), draw(), get, set tương tự class Player.

Truy cập class PanelGame tạo 1 list lưu các đối tượng chickens. Tạo phương thức addChicken() - Đây là phương thức được sử dụng để thêm đối tượng gà vào danh sách các đối tượng gà của trò chơi. Trong phương thức này, đầu tiên, một đối tượng Random được tạo ra để tạo vị trí ngẫu nhiên cho đối tượng gà.

Sau đó, tạo ra hai đối tượng gà chicken và chicken2. Chicken được đặt ở vị trí x = 0 và một vị trí ngẫu nhiên trên trục y. Chicken2 được đặt ở vị trí x = width (chiều rộng của màn hình) và một vị trí ngẫu nhiên trên trục y.

Để đối tượng chicken2 di chuyển ngược lại hướng di chuyển của chicken, giá trị góc angle của chicken2 được đặt là 180 độ (tức là quay ngược lại). Sau đó, hai đối tượng gà được thêm vào danh sách chickens.

Trong phương thức initObjectGame() khởi tạo mảng ArrayList chickens. Sau đó, một thread mới được tạo để thêm đối tượng gà mới vào danh sách chickens mỗi 3 giây (bằng cách gọi phương thức addChicken() và chờ 3 giây bằng phương thức sleep(3000)).

Vòng lặp while với biến điều kiện start được sử dụng để cho phép việc thêm đối tượng gà diễn ra trong khi trò chơi đang chạy. Khi biến start bị đặt là false, thread sẽ dừng lại và phương thức run() kết thúc.

Tiếp ta update vị trí chicken hoặc di chuyển chicken ở thread trong phương thức initKeyBoard().

Mỗi đối tượng gà sẽ được cập nhật thông tin mới bằng cách gọi phương thức update() của lớp Chicken.

Nếu vị trí của đối tượng gà hiện tại vượt quá giới hạn kích thước của màn hình (chiều rộng width và chiều cao height) thì đối tượng đó sẽ được loại bỏ khỏi danh sách chickens bằng phương thức remove().

Nếu đối tượng gà hiện tại trong danh sách chickens đã bị loại bỏ (có thể do va chạm với đạn hoặc bị giết bằng cách khác), thì đối tượng đó sẽ bị bỏ qua và không được cập nhật thông tin nữa.

Tạo phương thức checkBullets để kiểm tra đạn va vào chicken thì chicken biến mất. Phương thức checkBullets dùng để kiểm tra va chạm giữa đối tượng Bullet và Chicken trong trò chơi. Đầu vào là một đối tượng Bullet được truyền vào để kiểm tra. Phương thức sẽ lặp qua tất cả các đối tượng Chicken trong danh sách chickens, sau đó tính toán phần giao của hình dạng của Bullet và Chicken sử dụng lớp Area và kiểm tra xem kết quả có rỗng không. Nếu không rỗng, nghĩa là Bullet và Chicken đã va chạm và đối tượng Chicken tương ứng sẽ bị xóa khỏi danh sách chickens. Cuối cùng, đối tượng Bullet cũng sẽ được xóa khỏi danh sách bullets.

**Tạo máu player, máu chicken**

Tạo class HP trong package gameObject:

Thuộc tính:

* MAX\_HP: giá trị tối đa của HP.
* currentHp: giá trị hiện tại của HP.

Phương thức:

* getMAX\_HP(): trả về giá trị tối đa của HP.
* setMAX\_HP(double MAX\_HP): thiết lập giá trị tối đa của HP.
* getCurrentHp(): trả về giá trị hiện tại của HP.
* setCurrentHp(double currentHp): thiết lập giá trị hiện tại của HP.
* HP(double MAX\_HP, double currentHp): constructor có tham số để khởi tạo giá trị của MAX\_HP và currentHp.
* HP(): constructor không tham số để khởi tạo đối tượng HP với giá trị mặc định.

Tạo class HPRender để vẽ đồ họa:

* Constructor: Nhận vào một object HP và lưu trữ lại
* Phương thức hpRender: Vẽ thanh máu dựa trên thông tin của object HP truyền vào. Tham số đầu vào gồm graphics context (g2), shape của nhân vật và vị trí y của thanh máu trên trục y.
* Phương thức updateHP: Trừ đi một lượng máu cutHp từ máu hiện tại của nhân vật và trả về true nếu nhân vật còn máu, false nếu đã hết máu.
* Phương thức getHp: Trả về giá trị máu hiện tại của nhân vật.
* Phương thức resetHp: Thiết lập lại máu hiện tại của nhân vật bằng giá trị MAX\_HP.

Class Chicken kế thừa class HpRender. Gọi phương thức super(new HP(20, 20)) trong constructor. Sau đó gọi phương thức hpRender() trong phương thức draw().

Vào class PanelGame thêm câu lệnh if(!chicken.updateHP(bullet.getSize())) trong phương thức checkBullets.

Máu của Player cũng làm giống như với class Chicken.

Trong class Player tạo biến private final Area playerShap để lưu shape máy bay. Đối tượng Player cũng có một đường dẫn hình dạng được tạo bằng cách sử dụng lớp Path2D để tạo ra một đa giác được xác định bởi tập hợp các điểm. Sau đó, đường dẫn hình dạng này được chuyển thành một đối tượng hình dạng (Area) được lưu trữ trong thuộc tính "playerShap". Tạo theme phương thức getShape. Phương thức getShape() được sử dụng để lấy hình dạng của đối tượng Player dưới dạng Area (một lớp được sử dụng để đại diện cho một hình dạng hình học), sau đó được áp dụng một số phép biến đổi hình học trên nó. Các phép biến đổi này được thực hiện bằng cách sử dụng lớp AffineTransform, đóng vai trò là một ma trận biến đổi để quay, dịch và co giãn hình dạng.

Cụ thể, phương thức này trả về một Area mới được tạo bằng cách áp dụng các phép biến đổi sau trên playerShap (một Area khác đại diện cho hình dạng ban đầu của Player):

Dịch chuyển hình dạng theo tọa độ x và y của Player bằng cách sử dụng phương thức translate(x, y) của AffineTransform.

Quay hình dạng theo góc quay hiện tại của Player (được xác định bởi thuộc tính angle) bằng cách sử dụng phương thức rotate(angle, centerX, centerY) của AffineTransform, với centerX và centerY là tọa độ trung tâm của hình dạng.

Tạo một hình dạng mới bằng cách áp dụng các phép biến đổi trên playerShap bằng cách sử dụng phương thức createTransformedShape(shape) của AffineTransform.

Sau khi hoàn thành, phương thức trả về hình dạng mới này, đại diện cho hình dạng hiện tại của Player đã được áp dụng các phép biến đổi hình học tương ứng.

**Máu Player hết chương trình kết thúc nhấn enter để chơi lại**

Trong class Player, thêm thuộc tính private boolean alive = true, các hàm isAlive(), setAlive(). Trong đoạn mã này, có một biến alive được khai báo với giá trị mặc định là true. Phương thức isAlive() được sử dụng để truy xuất giá trị của biến alive, nó sẽ trả về true nếu alive vẫn đang là true, ngược lại nó sẽ trả về false.

Phương thức setAlive() được sử dụng để thay đổi giá trị của biến alive. Nếu tham số truyền vào là true, thì giá trị của biến alive sẽ được cập nhật thành true, ngược lại nếu tham số truyền vào là false, thì giá trị của biến alive sẽ được cập nhật thành false.

Tạo phương thức reset() để khởi tạo lại trò chơi. Phương thức reset() được sử dụng để thiết lập lại trạng thái ban đầu của đối tượng Player. Cụ thể, nó đặt giá trị của biến alive thành true, gọi phương thức resetHp() để đặt lại giá trị của HP về giá trị ban đầu, đặt góc quay angle và tốc độ speed về giá trị 0. Tức là phương thức này có tác dụng khởi tạo lại trạng thái ban đầu của đối tượng Player để bắt đầu một ván chơi mới.

Trong class PanelGame:

* Tạo phương thức checkPlayer(Chicken chicken) kiểm tra va chạm giữa player và một con gà.  
  Trong đó, trước tiên, nó tạo ra một Area với hình dạng của player và tương tác với hình dạng của gà. Nếu khu vực này không rỗng, tức là player và gà đang va chạm, phương thức sẽ lấy hp của con gà và cập nhật lại hp của player. Nếu hp của con gà nhỏ hơn hoặc bằng 0 sau khi cập nhật, phương thức sẽ loại bỏ con gà đó khỏi danh sách các con gà. Nếu hp của player nhỏ hơn hoặc bằng 0, phương thức sẽ đặt cờ alive của player thành false.
* Trong phương thức drawGame() thêm điều kiện if(player.isAlive()){player.draw(g2);} (Máy bay không hiện khi hết máu) và thêm điều kiện if(player.isAlive()) vào thread của phương thức initKeyBoard.
* Tạo text “Game over” khi kết thúc trò chơi trong phương thức drawGame(). Chúng ta sử dụng một phần của hàm paintComponent(Graphics g) để vẽ trò chơi lên màn hình.  
  Nếu người chơi đã chết (!player.isAlive()), thì trò chơi sẽ hiển thị thông báo "GAME OVER" và hướng dẫn người chơi ấn phím Enter để tiếp tục chơi.  
  Đoạn code sử dụng đối tượng Graphics2D để vẽ chữ lên màn hình với font chữ đậm, kích thước 50 cho tiêu đề "GAME OVER", và kích thước 15 cho phần hướng dẫn "Press key enter to continue...".  
  Phần vẽ chữ được tính toán vị trí dựa trên kích thước của màn hình và kích thước của văn bản, sử dụng đối tượng FontMetrics và Rectangle2D.
* Thêm phím enter để reset() trò chơi.

**Tạo điểm trò chơi**

Trong class PanelGame thêm biến score = 0. Truy cập phương thức checkBullets() kiểm tra if(!chicken.updateHP(bullet.getSize())) thì score++.

Truy cập phương thức drawGame() để hiển thị điểm trên màn hình:

g2.setColor(Color.WHITE);

g2.setFont(getFont().deriveFont(Font.BOLD, 15f));

g2.drawString("Score : " + score, 10, 20);

**Chèn âm thanh**

Lớp sound sử dụng thư viện javax.sound.sampled để tạo và phát các tệp âm thanh. Bên trong lớp Sound, có 4 biến URL để lưu trữ các tệp âm thanh được sử dụng trong game.

4 biến URL trong lớp Sound được khởi tạo và đặt tên là:

* Shoot: đại diện cho tệp âm thanh "shot.wav", được sử dụng khi người chơi bắn đạn trong game.
* Collide: đại diện cho tệp âm thanh "collide.wav", được sử dụng khi hai đối tượng va chạm trong game.
* Destroy: đại diện cho tệp âm thanh "destroy.wav", được sử dụng khi một đối tượng bị hủy trong game.
* Cuckoo: đại diện cho tệp âm thanh "chicken.wav", được sử dụng khi một con gà xuất hiện trong game (có thể là một phần của cốt truyện hoặc là một phần của hiệu ứng âm thanh trong game).

Các biến URL này được khởi tạo trong phương thức khởi tạo của lớp Sound bằng cách sử dụng phương thức getClassLoader () và getResource ().

Phương thức play () được sử dụng để phát các tệp âm thanh. Nó nhận một đối tượng URL làm đối số và sau đó tạo một đối tượng AudioInputStream từ URL này. Tiếp theo, nó tạo một đối tượng Clip bằng cách sử dụng phương thức getClip () của AudioSystem. Đối tượng Clip này được mở bằng cách sử dụng phương thức open () và sau đó thêm một LineListener để đóng Clip khi nó phát xong. Sau đó, tệp AudioInputStream được đóng và Clip được bắt đầu bằng cách sử dụng phương thức start () của nó.

Khi Clip đã phát xong, phương thức LineListener được kích hoạt và phương thức update () được gọi. Trong phương thức này, nếu sự kiện LineEvent có kiểu là STOP, đối tượng Clip được đóng bằng cách sử dụng phương thức close () của nó. Nếu có bất kỳ ngoại lệ nào xảy ra trong phương thức play (), nó sẽ được in ra bằng cách sử dụng phương thức println () của đối tượng System.err.

Link source code:

<https://github.com/ChuDung2k3/planeGameJava?fbclid=IwAR0P4xIBU91KZyfWouSVwM7S0AHqiZ4q6Mz9oZ1E8_XVJ6f0Sh8KryCUXak>

IV. Kết luận và đánh giá

1. Kết quả đạt được

* Tính bao phủ: Các class được phân chia rõ ràng và mỗi class chỉ đảm nhận một nhiệm vụ cụ thể trong game, giúp cho code có tính bao phủ tốt.
* Sử dụng hình ảnh và hiệu ứng đồ họa: Game được thiết kế với đồ họa khá đẹp mắt và sử dụng các hiệu ứng động để tạo ra sự sinh động và hấp dẫn cho người chơi.
* Viết code theo quy ước: Các đoạn code tuân theo quy ước đặt tên và định dạng code, giúp cho code đẹp mắt và dễ đọc hơn.

2. Hạn chế

* Trò chơi vẫn chưa có các cấp bậc màn chơi khác nhau theo từng độ khó cho người chơi
* Game chưa tạo được giao diện đẹp mắt cho người chơi

3. Hướng phát triển

* Nhóm em có dự định thêm nhiều loại vũ khí và tăng tính đa dạng của gameplay. Việc thêm nhiều loại vũ khí khác như tên lửa, đại bác, bom nổ, hay các loại vũ khí laser, sẽ giúp tăng thêm tính đa dạng của gameplay, tạo ra nhiều trải nghiệm mới cho người chơi.
* Thêm nhiều mức độ khó khác nhau và chế độ chơi: Để tăng tính thử thách và độ phong phú cho trò chơi, có thể thêm nhiều mức độ khó khác nhau như mức dễ, mức trung bình và mức khó, v.