BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

Bài tập lớn môn học
Công Nghệ JaVa

TÊN ĐỀ TÀI
Lập trình game space invader

Giảng viên:
Sinh viên thực hiện: Đỗ Đức Việt
Mã sinh viên: 211200831

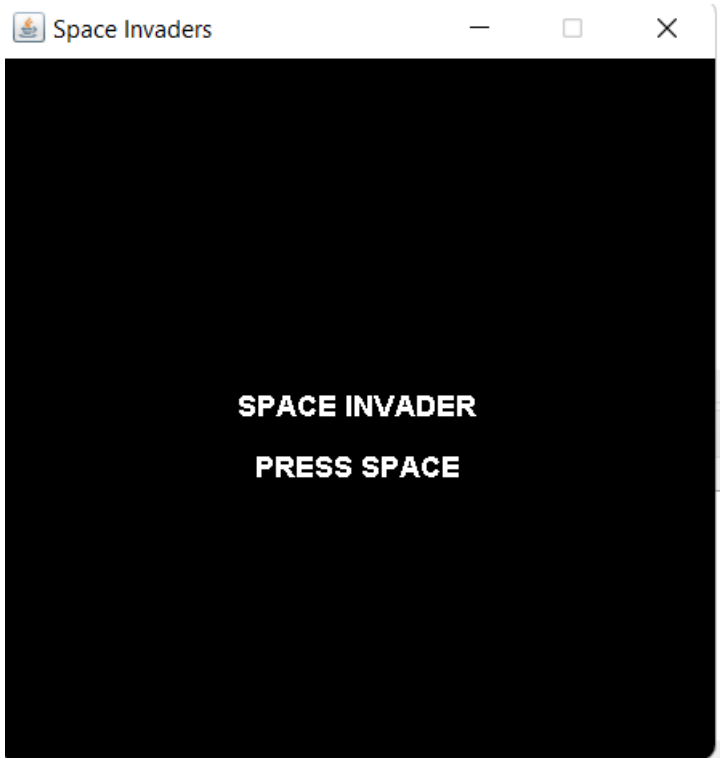
Ngày 2, tháng 5, năm 2023

I Đề bài

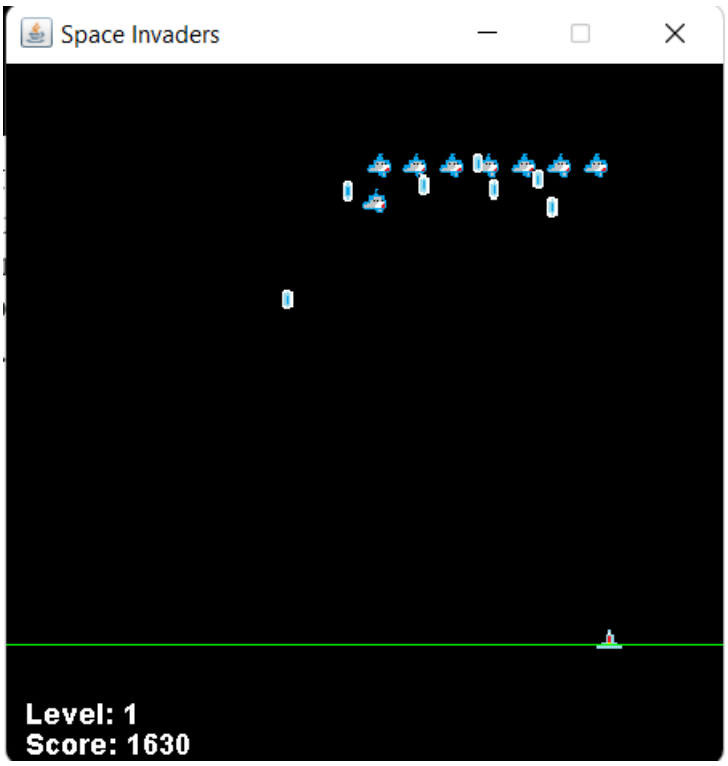
Giới thiệu: Space Invaders hay còn được biết là game bắn ruồi là tựa game bắn súng điện tử thùng được phát triển bởi Tomohiro Nishikado vào năm 1978. Trong game, bạn sẽ điều khiển phi thuyền và hạ gục những đợt người ngoài hành tinh để giành được nhiều điểm nhất có thể. Chính hệ thống tính điểm này đã trở thành yếu tố chuẩn mực cho các game điện tử thùng về sau.

Mặc dù cũng có một số game bắn súng ra đời trước nó nhưng sự thành công của nó đến từ các tính năng mới mẻ, đồ họa hấp dẫn cùng một số yếu tố khác, chính những điều đó đã tạo nên cơn sốt game điện tử thùng tại Nhật và khắp thế giới sau đó, thúc đẩy các công ty Nhật khác tham gia vào thị trường game. Bài tập lớn lần này em làm dựa trên trò chơi đó

Cách chơi: Người chơi bấm phím space để bắt đầu game



sử dung 2 phím mũi tên trái phải để điều khiển máy bay dùng nút space để bắn hạ invader đang muốn xâm lược nếu để invader xuống dưới vạch chỉ định hoặc người chơi bị bắn hạ thì game over.



Bắn hạ invader sẽ được điểm còn invader xuống càng thấp sẽ càng ít điểm. Nếu người chơi bắn hạ hết invader thì sẽ sang màn mới độ khó sẽ tăng lên và số invader sẽ nhiều hơn.

II Các thành phần của game

Có 3 package chính trong game

1.Entities (dùng để chứa các thực thể trong game)

a. Class Entity

b. Class Alien

c. Class AlienArmy

d. Class Bullet

e. Class Player

2. Inputs (chứa các đầu vào của game)

a. Class KeyboardInput

3. Main (nơi game vận hành)

a. Class Commons

b. Class GamePanel

c. Class GameWindow

III Các lớp của bài toán

1.Entities (dùng để chứa các thực thể trong game)

a. Class Entity

Thuộc tính:

```
private boolean visible =true;(xem thực thể có được hiện thị)
private Image image; (Lưu ảnh của thực thể)
protected int x; (toạ độ x y trên cửa sổ game)
protected int y;
protected boolean dying; (xem thực thể đã chết chưa)
```

Phương thức:

Getter setter

```
public void die() {
      visible = false;
}
```

(Tất cả các lớp dưới đều kế thừa Entites trừ AlienArmy)

b. Class Alien

Thuộc tính:

```
private Bomb bomb;
Random generator = new Random();
private final String ALIEN = "res/alien.png";
private int alienSpeed =1,alienCount =0;
private int score =250;
```

Phương thức:

Draw() để vẽ người chơi

Update() để cập nhập sự thay đổi

UpdateY() cập nhập y của alien

Class con:

Class bomb kế thừa Entities

Thuộc tính:

```
private final String bomb = "res/bomb.png";
```

```
private boolean destroyed;
```

Phương thức: hàm tạo , getter, setter

c. Class Bullet

Thuộc tính:

```
private String shot = "res/shot.png";
private final int H_SPACE = 3;
private final int V_SPACE = 1;
```

Phương thức:

Bullet(int x,int y) hàm tạo set vị trí và dying bằng false

Draw() để vẽ người chơi

Update() để cập nhập sự thay đổi

d. Class Player

Thuộc tính:

```
private final int START_Y = 280;
private final int START_X = 270;
private boolean attack = false;
private final String player = "res/player.png";
private int width;
private boolean left, right;
```

Phương thức:

Các hàm geter setter

Draw() để vẽ người chơi

Update() để cập nhập sự thay đổi

UpdatePos kiểm tra diều kiện left right để thay đổi vị trí người chơi sang 2 bên bằng cách cộng trừ x

e. Class AlienArmy

Thuộc tính:

```
private ArrayList<Alien> aliens;
private int alienX = 150;
private int alienY = 5;
private int quantity; (số lượng của đội alien)
```

Phương thức:

Hàm tạo

Hàm getter, setter

Draw()

Update()

2. Inputs (chứa các đầu vào của game)

a. Class KeyboardInput(implements KeyListener)

Thuộc tính:

```
private GamePanel gamePanel;
private int x,y;
public KeyBoardInputs(GamePanel gamePanel) {
	this.gamePanel = gamePanel;
}
```

Phương thức:

```
@Override
```

KeyPressed()

KeyReleased()

3. Main

a. Class Commons (interface lưu các hằng số của game)

b. Class GamePanel (Nơi game hoạt động kế thừa JPanel và implement runable)

Thuộc tính:

```
private AlienArmy alienAmry;
private Player player;
private Bullet bullet;
private Boolean ingame = true;
private int row =2,col=8;
private int level=1;
private int score =0;
private int State =1;
private Dimension d;
private final String expl = "res/explosion.png";
private Thread animator;
```

Phương thức:

Hàm tạo

GameInit() (khởi tạo các thực thể và chạy luồng)

GameOver() (hiện thị màn hình game over)

Upadte()

Pain() (vẽ màn hình game)

NEW() (tạo ra màn mới)

Hit() , hit2() (xử lý va chạm giữa các thực thể)

Run() (vòng lặp game)

c. Class GameWindow (kế thừa Jframe, là hàm thread chính)

Phương thức:

hàm tạo để tạo cửa sổ

III Mô tả các thao tác

1. Giao diện game

GameWindow kế thừa Jframe để tạo ra cửa sổ game và trong GameWindow sẽ add GamePanel kế thừa JPanel class xử lý mọi hình ảnh hiển thị của trò chơi. Dùng hàm pack() để kích thước cửa sổ bằng với GamePanel.
Code như sau:

```
public class GameWindow extends JFrame implements Commons{
    public GameWindow(){
        add(new GamePanel());
        setTitle("Space Invaders");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        pack();
        setVisible(true);
        setResizable(false);


    }
     public static void main(String[] args) {
            GameWindow gameWindow =new GameWindow();
        }
}
```

2. Vòng lặp game

GamePanel là một thread riêng để vận hành trò chơi

Cố định 1s có 120 cập nhập vị trí của các thực thể và repain vẽ lại màn hình bằng cách khai báo 2 biến để lưu khoảng cách giữa 2 lần update và repain

```
double timePerFrame = 1000000000.0/ 120;
double timePerUpdate = 1000000000.0/ 200;
```

Trong mỗi vòng lặp có biến curtentime lưu thời gian hiện tại và previoustime để lưu thời gian vòng lặp trước lấy hiệu của cả 2 để tính ra khoảng cách thời gian mỗi vòng lặp chia cho 2 timePerFrame và timePerUpdate và cộng thêm vào biến deltaU và deltaF khi 2 biến này lớn hơn 1 thì chạy hàm Update và repain

```
while(ingame) {

    long curtentTime = System.nanoTime();

    deltaU += (curtentTime - previousTime)/timePerUpdate;
    deltaF += (curtentTime - previousTime)/timePerFrame;
    previousTime = curtentTime;

    if(deltaU >= 1) {
        update();
        updates++;
        deltaU--;
    }


    if(deltaF >= 1) {
        repaint();
        frames++;
        deltaF--;
    }
```

Hàm update sẽ chứa các phương thức update của thực thế khác và 1 số phương thức khác

```
private void update() {
    if(State == 2) {
        player.update();

        alienAmry.update();
        hit();
        bullet.update();
        hit2();
        NEW();
    }
}
```

Hàm pain sẽ chứa những hàm draw() của các thực thể khác.

3. **Player di chuyển**

-Thêm KeyBoardInput vào game panel nếu e.getKeyCode() == KeyEvent.***VK_RIGHT*** thì đặt player.right == true ngược lại khi thả đặt player.right == false. Tương tự với nút sang trái

```
public void keyPressed(KeyEvent e) {

    if(gamePanel.getState() == 2) {
        // TODO Auto-generated method stub
        if(e.getKeyCode() == KeyEvent.VK_LEFT) {
            gamePanel.getPlayer().setLeft(true);
        }
        if(e.getKeyCode() == KeyEvent.VK_RIGHT) {
            gamePanel.getPlayer().setRight(true);
        }
```

```
public void keyReleased(KeyEvent e) {
    // TODO Auto-generated method stub
    if(e.getKeyCode() == KeyEvent.VK_LEFT) {
        gamePanel.getPlayer().setLeft(false);
    }
    if(e.getKeyCode() == KeyEvent.VK_RIGHT) {
        gamePanel.getPlayer().setRight(false);
    }
}
```

-Và trong hàm update của player nếu left == true and !right == true thì toạ độ của player x-- ngược lại x++.

```
public void update() {
    updatePos();

}
```

```
private void updatePos() {
        if(left  && !right) {
            if(x >= 0 + 2) x--;
        }
        if(right  && !left) {
            if(x <= BOARD_WIDTH-PLAYER_WIDTH) x++;
        }

    }
```

4. Player bắn

Nếu người chơi nhấn space và nếu viên đạn cũ đã chết thì một viên đạn mới sẽ được một viênj đạn sẽ được tạo tại vị trí của player.

```
if (e.getKeyChar() == KeyEvent.VK_SPACE) {
    if(gamePanel.getBullet().isDying()) {
        x = gamePanel.getPlayer().getX();
        y = gamePanel.getPlayer().getY();
        gamePanel.setBullet(new Bullet(x, y)) ;
    }
}
```

Và trong phương thức update() sẽ có phương thức update của đạn. toạ độ y của viên đạn xẽ là y--;

```
public void update() {
    if(isVisible()) {
        y--;
    }
}
```

Trong phương thức update của gamePanel cũng có phương thức hit sẽ kiểm tra xem toạ độ của viên đạn và alien để nếu bằng nhau thì alien sẽ bị bắn hạ

Sẽ kiểm tra nếu viên đạn vẫn còn được hiển thị sẽ tạo một Iterator duyệt toàn bộ alien nếu alien cũng được nhìn thấy và toạ độ bằng nhau sẽ tiêu diệt alien bằng cách set alien đã chết và không được hiển thị

```
public void hit() {

    if (bullet.isVisible()) {
        Iterator it = alienAmry.getAliens().iterator();
        int bulletX = bullet.getX();
        int bulletY = bullet.getY();
        while (it.hasNext()) {
            Alien alien = (Alien) it.next();
            int alienX = alien.getX();
            int alienY = alien.getY();

            if (alien.isVisible() && bullet.isVisible()) {
                if (bulletX >= (alienX) && bulletX <= (alienX + ALIEN_WIDTH) && bulletY >= (alienY) && bulletY <= (alienY+ALIEN_HEIGHT) )
                        ImageIcon ii =  new ImageIcon(expl);
                        alien.setImage(ii.getImage());
                        alien.setDying(true);
                         //deaths++;
                        bullet.setDying(true);
                        bullet.die();
                        alienAmry.setQuantity(alienAmry.getQuantity()-1);
                        score += alien.getScore();
                }
            }
            if(bullet.getY()<=0) {
                bullet.setDying(true);
            }
        }

    }
```

5. Alien

Class Alien sẽ có một class con là Class bomb và có thuộc tính bomb,
Một alien sinh bomb thì trong hàm update sẽ có một biến shot là một số sinh ngẫu nhiên nếu nó bằng **chance**(là hằng số được quy định trước) và bomb của alilen đã bị phá huỷ thì bomb sẽ được khởi tạo và lưu vào thuộc tính bomb

```
public void update() {
    alienCount++;
    if(alienCount == 5) {
        x += alienSpeed;
        alienCount =0;
    }

    int shot = generator.nextInt(15);
     if (shot == CHANCE && this.isVisible() && bomb.isDestroyed()) {
         bomb.setDestroyed(false);
         bomb.setVisible(true);
         bomb.setX(this.getX());
         bomb.setY(this.getY());
     }

     if (!bomb.isDestroyed()) {
         bomb.count++;
         if(bomb.count == 4) {
             bomb.setY(bomb.getY() + 1);
             bomb.count =0;
         }

         if (bomb.getY() >= GROUND - BOMB_HEIGHT) {
             bomb.setDestroyed(true);
         }
     }
}
```

và nếu bomb cũng có hàm update để bomb di chuyển bằng cách đặt y++
và nếu toạ độ y của bom quá vạch đích (GROUND là hằng số đặt từ trước)
thì set bomb.destroyed = true

và để bomb bắn trúng player thì cơ chế tương tư phương thức hit,

```
public void hit2() {
    Iterator i3 = alienAmry.getAliens().iterator();
    while (i3.hasNext()) {
        Alien a = (Alien) i3.next();
        Alien.Bomb b = a.getBomb();

        int bombX = b.getX();
        int bombY = b.getY();
        int playerX = player.getX();
        int playerY = player.getY();

        if (player.isVisible() && !b.isDestroyed()) {
            if ( bombX >= (playerX) &&
                bombX <= (playerX+PLAYER_WIDTH) &&
                bombY >= (playerY) &&
                bombY <= (playerY+PLAYER_HEIGHT) ) {
                    ImageIcon ii = new ImageIcon(expl);
                    player.setImage(ii.getImage());
                    player.setDying(true);
                    b.setDestroyed(true);
                    ingame = false;
                }
        }
        if(a.getY() == playerY && a.getX() == playerX) {
            ingame = false;
        }
        if(a.getY() >= GROUND) {
            ingame = false;
        }
    }
```

nhưng khi player bị bắn trúng ingame đặt bằng false vòng lặp game sẽ dừng lại

Nhưng để quản lý nhiều alien thì em dùng class AlienArmy
AlienArmy sẽ có hàm tạo để tạo và lưu nhiều alien vào danh sanh liên kết quantity để lưu số lượng alien.

```
public AlienArmy(int col, int row) {
    aliens =new ArrayList<Alien>();
    for(int i=0 ;i < col; i++) {
        for(int j=0; j<row; j++) {
            Alien alien = new Alien(alienX + 18*j, alienY + 18*i);
            aliens.add(alien);
        }
    }
    quantity =col*row;
}
```

Hàm update của alienArmy sẽ có một iterator duyệt tất cả alien trong danh sách nếu có bất kỳ alien nào chưa chết có toạ độ x == cạnh cửa sổ trò chơi thì sẽ có một iterator nữa duyệt tất cả alien update y và trừ điểm của alien

```
public void update() {
    Iterator it = aliens.iterator();
    while (it.hasNext()) {
        Alien a1= (Alien) it.next();
        a1.update();
        int x = a1.getX();
        if (x  >= BOARD_WIDTH - ALIEN_WIDTH && a1.isVisible()) {
            Iterator i1 = aliens.iterator();
            while (i1.hasNext()) {
                Alien a2 = (Alien) i1.next();
                a2.setScore(a2.getScore()-10);
                a2.updateY();
            }
        }
        if (x <= BORDER_LEFT) {

            Iterator i2 = aliens.iterator();
            while (i2.hasNext()) {
                Alien a = (Alien)i2.next();
                a.updateY();
            }
        }
    }
```

Hàm update y sẽ chỉnh tăng toạ độ y của alien để cho alien đi xuống và chỉnh alienspeed = -1 để alien đi sang phía ngược lại hướng đang đi

```
public void updateY() {
    y += GO_DOWN;
    alienSpeed *= -1;
}
    private int alienSpeed
```

Hàm NEW() được đặt trong update() cua class panel khi quatity của alienArmy == 0 thì sẽ tạo mới danh sách Alien với số hàng tăng

```
public void NEW() {
    if(alienAmry.getQuantity() ==0) {
        row++;
        level++;
        alienAmry= new AlienArmy(row, col);
    }
}
```

6. Màn hình pause và bắt đầu

Thuộc tính state trong GamePanel sẽ lưu các kiểu màn hình trong game
State == 1 là đang ở game menu ==2 là đang trong trò chơi ==3 là đang pause
Ban đầu vào game state sẽ bằng 1 hàm pain sẽ vẽ game menu.

```
if(State == 1) {

    Font small = new Font("Helvetica", Font.BOLD, 14);
    FontMetrics metr = this.getFontMetrics(small);
    g.setColor(Color.white);
    g.setFont(small);
    g.drawString("SPACE INVADER", (BOARD_WIDTH - metr.stringWidth("SPACE INVADER"))/2, BOARD_WIDTH/2);
    g.drawString("PRESS SPACE", (BOARD_WIDTH - metr.stringWidth("PRESS SPACE"))/2, BOARD_WIDTH/2+30);
}
```

Khi đó nhấn space sẽ đặt lại state bằng 2 và bắt đầu game

```
}
if(gamePanel.getState() == 1) {
    // TODO Auto-generated method stub
    if (e.getKeyChar() == KeyEvent.VK_SPACE) {
        gamePanel.setState(2);
    }
}
```

Nếu nhấn Esc trong lúc đang chơi game thì hàm pain sẽ vẽ màn hình pause

Game và hàm update chỉ được chạy khi state ==2 nên sẽ các thực thể sẽ không được update

```
if(gamePanel.getState() == 2) {
    // TODO Auto-generated method stub
    if(e.getKeyCode() == KeyEvent.VK_LEFT) {
        gamePanel.getPlayer().setLeft(true);
    }
    if(e.getKeyCode() == KeyEvent.VK_RIGHT) {
        gamePanel.getPlayer().setRight(true);
    }
    if (e.getKeyChar() == KeyEvent.VK_SPACE) {
        if(gamePanel.getBullet().isDying()) {
            x = gamePanel.getPlayer().getX();
            y = gamePanel.getPlayer().getY();
            gamePanel.setBullet(new Bullet(x, y)) ;
        }
    }
    if (e.getKeyCode() == KeyEvent.VK_ESCAPE) {
        gamePanel.setState(3);
    }
}
```

Nếu nhấn Ptrong lúc đang pause game game state sẽ chuyển về 2