**Lucrare de laborator nr. 7**

**Panfil Ion Grupa: CR-221 GitHub:** https://github.com/Panfil-Ion/TAP.git

**Realizarea unui joc lucrarii de laborator**

**Codul**

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.util.ArrayList;  
import java.util.List;  
import java.util.Random;  
  
public class SpaceShooterGame extends JPanel implements KeyListener, Runnable {  
  
 private static final int WIDTH = 800;  
 private static final int HEIGHT = 600;  
 private static final int PLAYER\_WIDTH = 50;  
 private static final int PLAYER\_HEIGHT = 50;  
 private static final int ENEMY\_WIDTH = 30;  
 private static final int ENEMY\_HEIGHT = 30;  
  
 private boolean running;  
 private Thread gameThread;  
 private int playerX, playerY;  
 private List<Point> enemies;  
 private int level;  
 private int score;  
 private boolean leftPressed, rightPressed;  
 private int enemySpeed;  
  
 private int playerSpeed;  
  
 private String[] levelMessages = {  
 "Începătorul curajos",  
 "Urcătorul de dealuri",  
 "Eroul neînfricat",  
 "Conducătorul de nave spațiale",  
 "Maestrul evitării",  
 "Supraviețuitorul cosmic",  
 "3.2.1",  
 "Boom ",  
 "...",  
 };  
 public SpaceShooterGame() {  
 setPreferredSize(new Dimension(WIDTH, HEIGHT));  
 setBackground(Color.BLACK);  
 setFocusable(true);  
 addKeyListener(this);  
  
 playerX = WIDTH / 2 - PLAYER\_WIDTH / 2;  
 playerY = HEIGHT - 2 \* PLAYER\_HEIGHT;  
  
 enemies = new ArrayList<>();  
 level = 1;  
 score = 0;  
  
 leftPressed = false;  
 rightPressed = false;  
  
 playerSpeed = 5;  
 enemySpeed = 2;  
  
 spawnEnemies(); // Spawn enemies at level 1  
  
 startGame();  
 }  
  
  
 private void startGame() {  
 running = true;  
 gameThread = new Thread(this);  
 gameThread.start();  
 }  
  
 private void stopGame() {  
 running = false;  
 try {  
 gameThread.join();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
  
 private void update() {  
 // Move player  
 if (leftPressed && playerX > 0) {  
 playerX -= playerSpeed; // Use playerSpeed instead of 5  
 }  
 if (rightPressed && playerX < WIDTH - PLAYER\_WIDTH) {  
 playerX += playerSpeed; // Use playerSpeed instead of 5  
 }  
  
 // Move enemies  
 List<Point> toRemove = new ArrayList<>();  
 for (Point enemy : enemies) {  
 enemy.y += enemySpeed;  
 if (enemy.y > HEIGHT) {  
 toRemove.add(enemy);  
 score++; // Increase score when an enemy is removed  
 }  
 }  
 enemies.removeAll(toRemove);  
  
 // Check for collision  
 Rectangle playerRect = new Rectangle(playerX, playerY, PLAYER\_WIDTH, PLAYER\_HEIGHT);  
 for (int i = 0; i < enemies.size(); i++) {  
 Point enemy = enemies.get(i);  
 Rectangle enemyRect = new Rectangle(enemy.x, enemy.y, ENEMY\_WIDTH, ENEMY\_HEIGHT);  
 if (playerRect.intersects(enemyRect)) {  
 stopGame();  
 JOptionPane.showMessageDialog(this, "Game Over! Your Score: " + score);  
 break;  
 }  
 }  
 }  
  
 private void draw(Graphics g) {  
 // Draw player  
 g.setColor(Color.BLUE);  
 g.fillRect(playerX, playerY, PLAYER\_WIDTH, PLAYER\_HEIGHT);  
  
 if (level <= levelMessages.length) {  
 g.setColor(Color.WHITE); // Set color to white  
 Font originalFont = g.getFont(); // Save the original font  
 g.setFont(new Font("Arial", Font.BOLD, 27)); // Set font size to 20  
 Graphics2D g2d = (Graphics2D) g;  
 g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_OVER, 0.3f)); // Set transparency to 30%  
 g.drawString(levelMessages[level - 1], WIDTH - 475, 50);  
 g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC\_OVER, 1f)); // Reset transparency to 100%  
 g.setFont(originalFont); // Reset the font to the original  
 }  
  
 // Draw enemies  
 g.setColor(Color.RED);  
 for (Point enemy : enemies) {  
 g.fillRect(enemy.x, enemy.y, ENEMY\_WIDTH, ENEMY\_HEIGHT);  
 }  
  
 // Draw level and score  
 g.setColor(Color.WHITE);  
 g.setFont(new Font("Arial", Font.BOLD, 20));  
 g.drawString("Level: " + level, 20, 20);  
 g.drawString("Score: " + score, 20, 40);  
 }  
  
 @Override  
 public void paintComponent(Graphics g) {  
 super.paintComponent(g);  
 draw(g);  
 }  
  
 @Override  
 public void keyPressed(KeyEvent e) {  
 int key = e.getKeyCode();  
 if (key == KeyEvent.VK\_LEFT) {  
 leftPressed = true;  
 } else if (key == KeyEvent.VK\_RIGHT) {  
 rightPressed = true;  
 }  
 }  
  
 @Override  
 public void keyTyped(KeyEvent e) {}  
  
 @Override  
 public void keyReleased(KeyEvent e) {  
 int key = e.getKeyCode();  
 if (key == KeyEvent.VK\_LEFT) {  
 leftPressed = false;  
 } else if (key == KeyEvent.VK\_RIGHT) {  
 rightPressed = false;  
 }  
 }  
  
 @Override  
 public void run() {  
 while (running) {  
 try {  
 Thread.sleep(20);  
 update();  
 repaint();  
 if (enemies.isEmpty()) {  
 level++;  
 enemySpeed += 1; // Increase enemy speed by 1 with each level  
 playerSpeed += 1; // Increase player speed by 1 with each level  
 spawnEnemies();  
 }  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
 private void spawnEnemies() {  
 Random rand = new Random();  
 int numEnemies = level \* 5;  
 for (int i = 0; i < numEnemies; i++) {  
 int x = rand.nextInt(WIDTH - ENEMY\_WIDTH);  
 int y = -rand.nextInt(HEIGHT);  
 enemies.add(new Point(x, y));  
 }  
 }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("Space Shooter");  
 SpaceShooterGame game = new SpaceShooterGame();  
 frame.getContentPane().add(game);  
 frame.pack();  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 frame.setLocationRelativeTo(null);  
 frame.setVisible(true);  
 }  
}

Codul furnizat reprezintă implementarea unui simplu joc de tip shooter spațial scris în limbajul Java, utilizând biblioteca grafică Swing. Iată o descriere generală a ceea ce face acest cod:

Codul furnizat reprezintă implementarea unui simplu joc de tip shooter spațial scris în limbajul Java, utilizând biblioteca grafică Swing

**Setări și inițializare:** În constructorul clasei SpaceShooterGame, sunt efectuate setările inițiale ale jocului, cum ar fi dimensiunile ferestrei, culoarea de fundal, inițializarea jucătorului și a inamicilor, nivelul și scorul, precum și alte variabile și configurări necesare.

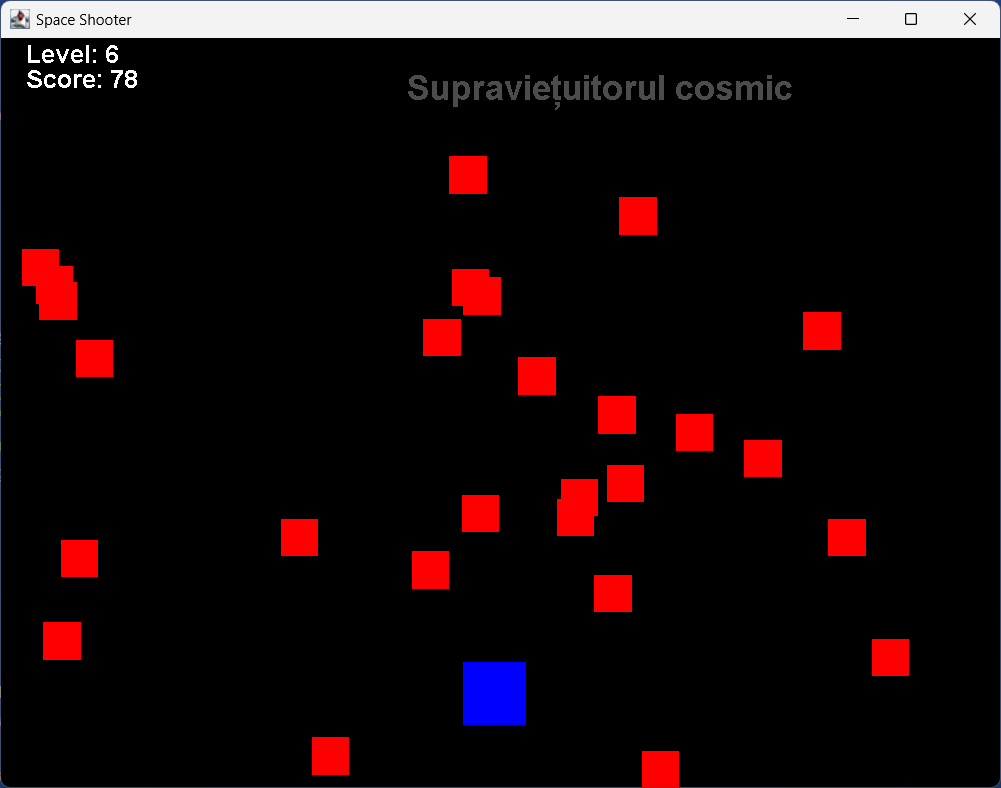
**Desenare și interacțiune:** Sunt suprascrise metodele paintComponent(Graphics g), keyPressed(KeyEvent e), keyReleased(KeyEvent e) și run() pentru a desena elementele jocului, pentru a gestiona intrările de la tastatură și pentru a actualiza starea jocului într-un thread separat. Astfel, jucătorul poate mișca o navă spațială în stânga și în dreapta, iar inamicii sunt generați și mișcați în jos pe ecran.

**Actualizare și verificare:** Metoda update() este responsabilă pentru actualizarea pozițiilor jucătorului și inamicilor, detectarea coliziunilor și eliminarea inamicilor care au ieșit din ecran. În timpul rulării jocului, se verifică dacă toți inamicii au fost eliminați pentru a trece la nivelul următor, crescând viteza și generând noi inamici.

**Text și mesaje:** Este desenat text pe ecran pentru a afișa nivelul curent al jocului și scorul jucătorului. De asemenea, sunt afișate mesaje specifice nivelului la începutul fiecărui nivel.

**Terminare și restartare:** Jocul se oprește când jucătorul se ciocnește cu un inamic, afișând un mesaj de Game Over împreună cu scorul obținut. Apoi, jocul poate fi repornit făcând clic pe fereastra de mesaj.

Rezultatul codului

****

**Concluzie**

În acest laborator, am explorat crearea unui joc de tip shooter spațial în Java, folosind biblioteca grafică Swing. Am învățat cum să gestionăm desenarea elementelor grafice, interacțiunea cu utilizatorul și logica jocului. Prin lucrul cu firuri de execuție și detectarea coliziunilor, am putut crea o experiență de joc dinamică și captivantă. Integrarea mesajelor de avertizare și informațiilor despre joc a îmbunătățit experiența utilizatorului. Această activitate ne-a oferit o temelie solidă pentru dezvoltarea ulterioară a aplicațiilor interactive și a jocurilor în Java, explorând concepte importante precum abstractizarea, encapsularea și polimorfismul.