AMRITA SCHOOL OF ENGINEERING, BANGALORE
AMRITA VISHWA VIDYAPEETHAM
BANGALORE 560 035

DODGE THE ENEMY

A PROJECT REPORT

Submitted by

BL.EN.U4AIE19054 Ruthvik.K

For the subject

19AIE111- Data Structures and Algorithms -1

Faculty Incharge
Ms. D. Radha
Asst. Prof(SG),
CSE, ASE-BLR

Description of Project

My project is a game called Dodge the Enemy. The GUI used to make the game is Swing which is a part of AWT. Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. In this game, there is a main character of colour white and there will be enemies in different colours. The main aim of the game is that the player last as long as possible. If the player's health goes down to zero, the game is over. The current score and level are shown on the Gave over screen. The score and level are counted, and score is used as currency for the shop for upgrades. Instructions are given in the help icon, difficulty also can be chosen. The difficulty of the game increases with the increase in the level.
The GUI is JFrame(Swing) and awt, the background is black and graphics library is used to create the character and enemies.
Game.java: - The main class that runs the game is the Game class.
GameObject.java: - It contains the location of the object in the game.
Handler.java: - Data structure is used to handle objects.
Menu.java: - Displays the details on the screen when the game is started.
HUD.java : - Heads Up Display: - Displays the health capacity, score and level
KeyInput.java: - The key inputs for the player.
ID.java: - Defines the character of the game for health reduction.
Player.java: - Player is the main character of the game. All controls of the player is in this class.
MenuParticle.java: - Contains the background effects of the menu screen.
Trail.java: - The trail that every character has while moving.
Window.java: - Used to adjust the size of the game screen.
Spawn.java: - The class that spawns enemies.
Enemies: - There are different types of enemies such as basic enemy, fast enemy, boss, smart enemy. Each enemy has different characteristics. Basic Enemy just bounces off the walls, the smart enemy follows the player, the boss shoots out bullets, etc.
Shop.java: -

- Upgrade health increases the health capacity and refills the health.
- Upgrade speed increases the speed of the player.
- Refill health refills the health to maximum.
- The cost of each upgrade increases on every purchase.

I have used LinkedList as the main data structure in the class Handler.

Game.java

```
package Main_Project;
import java.awt.Canvas;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.image.BufferStrategy;
import java.util.Random;
public class Game extends Canvas implements Runnable {
	private static final long serialVersionUID = -6239124249454582O1L
	public static final int WIDTH = 640, HEIGHT = WIDTH / 12 * 9;
	private Thread thread;
	public boolean running = false;
	public static boolean paused = false;
	//normal = 0
	//hard = 1
	public static int diff = 0;
	private Random r;
	private Handler handler;
	private HUD hud;
	private Spawn spawner;
	private Menu menu;
	private Shop shop;
	// an enum class for the display screens of the game as menu and others
	public enum STATE {
		Menu,
		Select,
		Help,
		Shop,
		End,
		Game
	};
	public static STATE gameState = STATE.Menu;
	// the function that will initiate the game
	public Game() {
		handler = new Handler();
		hud = new HUD();
		shop = new Shop(handler,hud);
		menu = new Menu(this, handler, hud);
		this.addKeyListener(new KeyInput(handler, this));
		this.addMouseListener(menu);
		this.addMouseListener(shop)
```

```
                new Window(WIDTH, HEIGHT, "Let's start the Games...", this);
                spawner = new Spawn(handler, hud,this);
                r = new Random();
                if (gameState == STATE.Game) {
                        // player initialize
                        handler.addObject(new Player(WIDTH / 2 - 32, HEIGHT / 2
- 32, ID.Player, handler));
                        handler.addObject(
                                        new BasicEnemy(r.nextInt(Game.WIDTH -
64), r.nextInt(Game.HEIGHT - 64), ID.BasicEnemy, handler));
                } else {
                        for (int i = 0; i < 10; i++) {
                                handler.addObject(new
MenuParticle(r.nextInt(WIDTH), r.nextInt(HEIGHT), ID.MenuParticle, handler));
                        }
                }
        }
        // this function will start the game
        public synchronized void start() {
                thread = new Thread(this);
                thread.start();
                running = true;
        }
        // this function will end the game
        public synchronized void stop() {
                try {
                        thread.join();
                        running = false;

                } catch (Exception e) {
                        e.printStackTrace();
                }
        }
        // loop for the game to rum
        public void run() {
                // loop for all basic games
                long lastTime = System.nanoTime();
                double amountOfTicks = 60.0;
                double ns = 1000000000 / amountOfTicks;
                double delta = 0;
                long timer = System.currentTimeMillis();
                int frames = 0;
                while (running) {
```

```
                long now = System.nanoTime();
                delta += (now - lastTime) / ns;
                lastTime = now;
                while (delta >= 1) {
                        tick();
                        delta--;
                }
                if (running)
                        render();
                frames++;
                // displays the FPS(can be decreased if the 100 is increased to
1000)
                if (System.currentTimeMillis() - timer > 1000) {
                        timer += 1000;
                        frames = 0;
                }
        }
        stop();
    private void tick() {
        if (gameState == STATE.Game)
                if (!paused) {
                        hud.tick();
                        spawner.tick();
                        handler.tick();

                        if (HUD.HEALTH <= 0) {
                                HUD.HEALTH = 100;
                                gameState = STATE.End;
                                handler.clearEnemies();

                        }
        } else if (gameState == STATE.Menu || gameState == STATE.End ||
gameState == STATE.Help || gameState == STATE.Select) {
                menu.tick();
                handler.tick();
        }
    }
    // the function used to remove volatile images and decrease the frames per
    // second
    private void render() {
        // to render the volatile images
        BufferStrategy bs = this.getBufferStrategy();
        if (bs == null) {
```

```
                this.createBufferStrategy(3);
                return;
            }
            // this part is to set the background color
            Graphics g = bs.getDrawGraphics();
            g.setColor(Color.black);
            g.fillRect(0, 0, WIDTH, HEIGHT);
            if (paused) {
                g.setColor(Color.red);
                g.drawString("PAUSED", this.WIDTH/2-36,this.HEIGHT/2-
36);
            }
            if (gameState == STATE.Game) {
                hud.Render(g);
                handler.render(g);
            } else if(gameState == STATE.Shop) {
                shop.render(g);
            } else if (gameState == STATE.Menu || gameState == STATE.Help
|| gameState == STATE.End || gameState == STATE.Select) {
                menu.render(g);
                handler.render(g);
            }
            g.dispose();
            bs.show();
        }
        // to keep the characters in the screen or in its specific value limit
        public static float clamp(float var, float min, float max) {
            if (var >= max)
                return var = max;
            else if (var <= min)
                return var = min;
            else
                return var;
        }
        // main function to run the game
        public static void main(String[] args) {
            new Game();
        }
}
```

GameObject.java

```
package Main_Project;
import java.awt.Graphics;
import java.awt.Rectangle
//enemy and player are examples of game object
public abstract class GameObject {
	//coordinates of the objects
	protected float x,y;
	protected ID id;
	protected float velX,velY;
	//to take in details like coordinates and to differentiate between player and
enemy
	public GameObject(float x, float y, ID id) {
		this.x = x;
		this.y = y;
		this.id = id;
	}
	public abstract void tick();
	public abstract void render(Graphics g);
	//to check if the enemy is touching the player or not
	public abstract Rectangle getBounds();
	public void setX(float x) {
		this.x = x;
	}
	public float getX() {
		return x;
	}
	public void setY(float y) {
		this.y = y;
	}
	public float getY() {
		return y;
	}
	public void setId(ID id) {
		this.id = id;
	}
	public ID getId() {
		return id;
	}
	public float getVelX() {
		return velX;
	}
```

```
	public void setVelX(float velX) {
		this.velX = velX;
	}
	public float getVelY() {
		return velY;
	}
	public void setVelY(float velY) {
		this.velY = velY;
	}
}
```

Handler.java

```
package Main_Project;
import java.awt.Graphics;
import java.util.LinkedList;
import Main_Project.Game.STATE;
//render all objects, controls all objects and individually update them
public class Handler {
	LinkedList<GameObject> object = new LinkedList<GameObject>();
	//speed of the player
	public int move = 5;//tick and render functions keeps taking in temporary
objects for the value of i(ie:-each object at a time)
	public void tick() {
		//this loops for all the game objects present in the game
		for(int i =0;i<object.size();i++)
		{
			GameObject tempObject = object.get(i);
			tempObject.tick();
		}
	}
	public void render(Graphics g) {
		for(int i =0;i<object.size();i++)
		{
			GameObject tempObject = object.get(i);

			tempObject.render(g);
		}
	}
	//for removing all present enemies for the boss level
	public void clearEnemies() {
		for(int i =0;i<object.size();i++)
		{
			GameObject tempObject = object.get(i);
```

```
                        if(tempObject.getId()== ID.Player) {
                                object.clear();
                                addObject(new Player(tempObject.x, tempObject.y,
ID.Player, this));
                        }

                }
        }
        public void addObject(GameObject object) {
                this.object.add(object);
        }
        public void removeObject(GameObject object) {
                this.object.remove(object);
        }
}
```

Window.java

```
package Main_Project
import java.awt.*;
import javax.swing.JFrame;
public class Window extends Canvas{
        private static final long serialVersionUID = -8074370659981218511L;
        //function for the size of the screen of the game
        public Window(int width, int height, String title,Game game) {
                JFrame frame = new JFrame(title);
                //setting the size
                frame.setPreferredSize(new Dimension (width,height));
                frame.setMaximumSize(new Dimension (width,height));
                frame.setMinimumSize(new Dimension (width,height));
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                frame.setResizable(false);
                frame.setLocationRelativeTo(null);
                frame.add(game);
                frame.setVisible(true);

                //game is initialized
                game.start();
        }
}
```

Menu.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.Random;
import Main_Project.Game.STATE;
public class Menu extends MouseAdapter {
	private Handler handler;
	private HUD hud;
	private Spawn spawn;
	Random r = new Random();
	// to initialize this constructor in the main game file
	public Menu(Game game, Handler handler, HUD hud) {
		this.handler = handler;
		this.hud = hud;

	// function to determine what happens when the mouse is clicked
	public void mousePressed(MouseEvent e) {
		int mx = e.getX();
		int my = e.getY();
		if (Game.gameState == STATE.Menu) {
			// play button
			if (mouseOver(mx, my, 205, 150, 200, 64)) {
				Game.gameState = STATE.Select;
				return;
			}
			// help button
			if (mouseOver(mx, my, 205, 250, 200, 64)) {
				Game.gameState = STATE.Help;
			}
			// quit game
			if (mouseOver(mx, my, 205, 350, 200, 64)) {
				System.exit(0);
			}
		}
		// in the select screen
		if (Game.gameState == STATE.Select) {
			// normal button
			if (mouseOver(mx, my, 205, 150, 200, 64)) {
```

```
                        Game.gameState = STATE.Game;
                        handler.addObject(new Player(Game.WIDTH / 2 - 32,
Game.HEIGHT / 2 - 32, ID.Player, handler));
                        handler.clearEnemies();
                        handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
ID.BasicEnemy,
                                        handler));
                        Game.diff = 0;
                }
                // hard button
                if (mouseOver(mx, my, 205, 250, 200, 64)) {
                        Game.gameState = STATE.Game;
                        handler.addObject(new Player(Game.WIDTH / 2 - 32,
Game.HEIGHT / 2 - 32, ID.Player, handler));
                        handler.clearEnemies();
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
ID.BasicEnemy, handler));
                        Game.diff = 1;
                }
                // back button
                if (mouseOver(mx, my, 205, 350, 200, 64)) {
                        Game.gameState = STATE.Menu;
                }
        }
        // back button in the help page
        if (Game.gameState == STATE.Help) {
                if (mouseOver(mx, my, 500, 350, 100, 32)) {
                        Game.gameState = STATE.Menu;
                }
        }
        if (Game.gameState == STATE.End) {
                if (mouseOver(mx, my, 200, 350, 200, 64)) {
                        Game.gameState = STATE.Menu;
                        hud.setLevel(1);
                        hud.setScore(0);
                        spawn.setScoreKeep(0);
                }
        }
    }
    public void mouseReleased(MouseEvent e) {
    }
```

```
	private boolean mouseOver(int mx, int my, int x, int y, int width, int height)
{
		if (mx > x && mx < (x + width)) {
			if (my > y && my < (y + height)) {
				return true;
			} else
				return false;
		} else
			return false;
	}
	public void tick() {
	}
	public void render(Graphics g) {
		if (Game.gameState == STATE.Menu) {
			g.setColor(Color.GREEN);
			Font fnt = new Font("Arial", 1, 50);
			Font fnt2 = new Font("Times New Roman", 1, 40);
			g.setFont(fnt);
			g.drawString("Dodge the Enemy", 120, 100);
			g.setFont(fnt2);
			g.setColor(Color.white);
			g.drawRect(205, 150, 200, 64);
			g.drawString("Play", 265, 193);
			g.setColor(Color.white);
			g.drawRect(205, 250, 200, 64);
			g.drawString("Help", 265, 293);
			g.setColor(Color.white);
			g.drawRect(205, 350, 200, 64);
			g.drawString("Quit", 265, 393);
		} else if (Game.gameState == STATE.Help) {
			g.setColor(Color.GREEN);
			Font fnt2 = new Font("Times New Roman", 1, 30);
			Font fnt3 = new Font("Fraktur", 1, 20);
			Font fnt = new Font("Arial", 1, 50);
			g.setFont(fnt);
			g.drawString("Help", 260, 100);

			g.setColor(Color.white);
			g.setFont(fnt3);
			g.drawString("Main aim in this game is to dodge the
enemies.", 50, 150);
			g.drawString("Use keys W A S D to move your player.", 50,
175);
```

```
                g.drawString("W -> to move your character up.", 50, 200);
                g.drawString("S -> to move your character down.", 50, 225);
                g.drawString("A -> to move your character left.", 50, 250);
                g.drawString("D -> to move your character right.", 50, 275);
                g.drawString("P -> to pause the game.", 50, 300);
                g.setFont(fnt2);
                g.setColor(Color.white);
                g.drawRect(500, 350, 100, 32);
                g.drawString("Back", 518, 375);
            } else if (Game.gameState == STATE.End) {
                g.setColor(Color.GREEN);
                Font fnt2 = new Font("Times New Roman", 1, 30);
                Font fnt3 = new Font("Fraktur", 1, 20);
                Font fnt = new Font("Arial", 1, 50);
                g.setFont(fnt);
                g.drawString("Game Over", 180, 100);
                g.setColor(Color.white);
                g.setFont(fnt3);
                g.drawString("Your Score is " + hud.getScore(), 50, 150);
                g.drawString("Your Level is " + hud.getLevel(), 50, 175);
                g.setFont(fnt2);
                g.setColor(Color.white);
                g.drawRect(200, 350, 200, 64);
                g.drawString("Try Again", 230, 385);
            } else if (Game.gameState == STATE.Select) {
                g.setColor(Color.GREEN);
                Font fnt = new Font("Arial", 1, 50);
                Font fnt2 = new Font("Times New Roman", 1, 40);
                g.setFont(fnt);
                g.drawString("Select Difficulty", 100, 100);
                g.setFont(fnt2);
                g.setColor(Color.white);
                g.drawRect(205, 150, 200, 64);
                g.drawString("Normal", 245, 193);
                g.setColor(Color.white);
                g.drawRect(205, 250, 200, 64);
                g.drawString("Hard", 265, 293);
                g.setColor(Color.white);
                g.drawRect(205, 350, 200, 64);
                g.drawString("Back", 265, 393);
            }
        }
}
```

HUD.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
//heads up display for health
public class HUD {
	public int bounds = 0;
	public static int HEALTH = 100;
	private int greenValue = 255;
	private int score = 0;
	private int level = 1;
	public void tick() {
		HEALTH = (int) Game.clamp(HEALTH, 0, 100 + (bounds/2));
		greenValue = HEALTH * 2;
		greenValue = (int) Game.clamp(greenValue, 0, 255);
		score++;
	}
	public static int getHEALTH() {
		return HEALTH;
	}
	public int getGreenValue() {
		return greenValue;
	}
	public int getScore() {
		return score;
	}
	public void setScore(int score) {
		this.score = score;
	}
	public int getLevel() {
		return level;
	}
	public void setLevel(int level) {
		this.level = level;
	}
	public void Render(Graphics g) {
		g.setColor(Color.DARK_GRAY);
		g.fillRect(15, 15, 200 + bounds, 32);
		g.setColor(new Color(75, greenValue, 0));
		g.fillRect(15, 15, HEALTH * 2, 32);
		g.setColor(Color.WHITE);
		g.drawRect(15, 15, 200 + bounds, 32);
```

```
                g.drawString(HEALTH + "%", 15, 12);
                g.drawString("Score:" + score, 15, 64);
                g.drawString("Level:" + level, 15, 80);
                g.drawString("Press Space for shop", 15, 96);
        }
}
```

KeyInput.java

```
package Main_Project;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import Main_Project.Game.STATE;
public class KeyInput extends KeyAdapter {
        private Handler handler;
        Game game;
        public KeyInput(Handler handler, Game game) {
                this.handler = handler;
                this.game = game;
        }
        public void keyPressed(KeyEvent e) {
                int key = e.getKeyCode();
                for (int i = 0; i < handler.object.size(); i++) {
                        GameObject tempObject = handler.object.get(i);
                        if (tempObject.getId() == ID.Player) {
                                // all the key events from player1
                                if (key == KeyEvent.VK_W)
                                        tempObject.setVelY(-handler.move);
                                if (key == KeyEvent.VK_S)
                                        tempObject.setVelY(handler.move);
                                if (key == KeyEvent.VK_D)
                                        tempObject.setVelX(handler.move);
                                if (key == KeyEvent.VK_A)
                                        tempObject.setVelX(-handler.move);
                        }
                }
                if (key == KeyEvent.VK_P && Game.gameState == STATE.Game)
{
                                if (Game.paused)
                                        Game.paused = false;
                                else
                                        Game.paused = true;
                                return;
                }
```

```
                if (key == KeyEvent.VK_ESCAPE)
                        System.exit(0);
                if (key == KeyEvent.VK_SPACE) {
                        if (Game.gameState == STATE.Game)
                                Game.gameState = STATE.Shop;
                        else if (Game.gameState == STATE.Shop)
                                Game.gameState = STATE.Game;
                }
        }
        public void keyReleased(KeyEvent e) {
                int key = e.getKeyCode();
                for (int i = 0; i < handler.object.size(); i++) {
                        GameObject tempObject = handler.object.get(i);
                        if (tempObject.getId() == ID.Player) {
                                // all the key events from player1
                                if (key == KeyEvent.VK_W)
                                        tempObject.setVelY(0);
                                if (key == KeyEvent.VK_S)
                                        tempObject.setVelY(0);
                                if (key == KeyEvent.VK_D)
                                        tempObject.setVelX(0);
                                if (key == KeyEvent.VK_A)
                                        tempObject.setVelX(0);
                        }
                }
        }
}
```

ID.java

```
package Main_Project;
public enum ID {
        Player(),
        BasicEnemy(),
        FastEnemy(),
        SmartEnemy(),
        HardEnemy(),
        EnemyBoss(),
        MenuParticle(),
        Trail();
}
```

Player.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
public class Player extends GameObject {
	Handler handler;
	// controls of the player are here
	public Player(float x, float y, ID id, Handler handler) {
		super(x, y, id);
		this.handler = handler;
	}
	public Rectangle getBounds() {
		return new Rectangle((int) x, (int) y, 32, 32);
	}
	// this is for the movement of the blocks velX and velY are initiated in the
	// player
	public void tick() {
		x += velX;
		y += velY;
		handler.addObject(new Trail(x, y, ID.Trail, Color.WHITE, 32, 32,
0.08f, handler));
		x = Game.clamp((int) x, 0, Game.WIDTH - 44);
		y = Game.clamp((int) y, 0, Game.HEIGHT - 64);
		collision();

	public void collision() {
		for (int i = 0; i < handler.object.size(); i++) {
			GameObject tempObject = handler.object.get(i);
			if (tempObject.getId() == ID.BasicEnemy ||
tempObject.getId() == ID.FastEnemy
					|| tempObject.getId() == ID.SmartEnemy) {
				// in the above condition, temp object is the basic
enemy
				if (getBounds().intersects(tempObject.getBounds())) {
					// code for what happens if there is a collision
					HUD.HEALTH -= 2;
				}
			}
			if (tempObject.getId() == ID.EnemyBoss) {
```

```
                                        // in the above condition, temp object is the
basic enemy
                                        if
(getBounds().intersects(tempObject.getBounds())) {
                                                // code for what happens if there is a
collision
                                                HUD.HEALTH -= 20;
                                        }
                                }
                        }
                }
                // character is formed here
                public void render(Graphics g) {
                        g.setColor(Color.white);
                        g.fillRect((int) x, (int) y, 32, 32);

}
```

MenuParticle.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Random;
//this class is just for background effects
public class MenuParticle extends GameObject{
        private Handler handler;
        Random r = new Random();
        private Color col;
        public MenuParticle(float x, float y, ID id,Handler handler) {
                super(x, y, id);
                this.handler = handler;
                velX = (r.nextInt(5 - -5) + -5);
                velY = (r.nextInt(5 - -5) + -5);
                if(velX == 0) velX = 1;
                if(velY == 0) velY = 1;
                col = new Color( r.nextInt(255), r.nextInt(255), r.nextInt(255));
        }
        public void tick() {
                x += velX;
                y += velY;
                if(y < 0 || y > Game.HEIGHT-48) velY*=-1;
```

```
            if(x < 0 || x > Game.WIDTH-32) velX*=-1;
            //for trail
            handler.addObject(new Trail(x,y,ID.Trail,col,16,16,0.02f,handler));
        }
        public void render(Graphics g) {
            g.setColor(Color.red);
            g.fillRect((int)x,(int) y, 16, 16);
        }
        public Rectangle getBounds() {
            return new Rectangle((int)x,(int)y,32,32);
        }
}
```

Trail.java

```
package Main_Project;
import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
public class Trail extends GameObject{
        private float alpha = 1;
        //life = 0.001 to 0.1
        private float life;
        private Handler handler;
        private Color color;
        private int width,height;
        public Trail(float x, float y, ID id,Color color,int width,int height,float life,
Handler handler) {
            super(x, y, id);
            this.handler = handler;
            this.color = color;
            this.width = width;
            this.height = height;
            this.life = life;
        }
        public void tick() {
            if(alpha>life) {
                alpha -= (life - 0.001f);
            }else handler.removeObject(this);
        }
        public void render(Graphics g) {
            Graphics2D g2d = (Graphics2D) g;
```

```
                g2d.setComposite(makeTransparent(alpha));
                g.setColor(color);
                g.fillRect((int)x,(int) y, width, height);
                g2d.setComposite(makeTransparent(1));
        }

        private AlphaComposite makeTransparent(float alpha) {
                int type = AlphaComposite.SRC_OVER;
                return (AlphaComposite.getInstance(type, alpha));
        }
        public Rectangle getBounds() {
                return null;
        }
}
```

Spawn.java

```
package Main_Project;
import java.util.Random
public class Spawn {
        private Handler handler;
        private HUD hud;
        private Random r = new Random();
        private Game game;
        int scoreKeep = 0;
        public Spawn(Handler handler, HUD hud, Game game) {
                this.handler = handler;
                this.hud = hud;
                this.game = game;
        }
        public int getScoreKeep() {
                return scoreKeep;
        }
        public void setScoreKeep(int scoreKeep) {
                this.scoreKeep = scoreKeep;
        }
        public void tick() {
                scoreKeep++;

                if (scoreKeep >= 750) {
                        scoreKeep = 0;
                        hud.setLevel(hud.getLevel() + 1);
                        if (game.diff == 0) {
                                if (hud.getLevel() == 2)
```

```
                handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
            if (hud.getLevel() == 3)
                handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
            if (hud.getLevel() == 4)
                handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                          ID.SmartEnemy, handler));
            if (hud.getLevel() == 5) {
                handler.clearEnemies();
                handler.addObject(new
EnemyBoss2((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
            }
            if (hud.getLevel() == 8) {
                handler.clearEnemies();
                handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
                handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
            }
            if (hud.getLevel() == 9) {
                handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
                handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.BasicEnemy, handler));
            }
            if (hud.getLevel() == 10) {
                handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.SmartEnemy, handler));
                handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                        ID.SmartEnemy, handler));
            }
            if (hud.getLevel() == 11) {
                handler.clearEnemies();
```

```
                        handler.addObject(new
EnemyBoss((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
                    }
                    if (hud.getLevel() == 14) {
                        handler.clearEnemies();
                        handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                        handler.addObject(new
BasicEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));

                    if (hud.getLevel() == 15) {
                        handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                        handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                    }
                    if (hud.getLevel() == 16) {
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.SmartEnemy, handler));
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.SmartEnemy, handler));
                    }
                    if (hud.getLevel() == 17) {
                        handler.clearEnemies();
                        handler.addObject(new
EnemyBoss((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
                        handler.addObject(new
EnemyBoss2((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
                    }
                } else if (game.diff == 1) {
                    if (hud.getLevel() == 2)
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                    if (hud.getLevel() == 3)
                        handler.addObject(new
FastEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
```

```
                                        ID.BasicEnemy, handler));

                    if (hud.getLevel() == 4)
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.SmartEnemy, handler));
                    if (hud.getLevel() == 5) {
                        handler.clearEnemies();
                        handler.addObject(new
EnemyBoss((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));

                    if (hud.getLevel() == 8) {
                        handler.clearEnemies();
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.BasicEnemy, handler));
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.BasicEnemy, handler));
                    }
                    if (hud.getLevel() == 9) {
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.BasicEnemy, handler));
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                                ID.BasicEnemy,
handler));
                    }
                    if (hud.getLevel() == 10) {
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.SmartEnemy, handler));
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.SmartEnemy, handler));
                    }
                    if (hud.getLevel() == 11) {
                        handler.clearEnemies();
                        handler.addObject(new
EnemyBoss2((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
                    }
                    if (hud.getLevel() == 14) {
```

```
                        handler.clearEnemies();
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                    }
                    if (hud.getLevel() == 15) {
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.BasicEnemy, handler));
                        handler.addObject(new
HardEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                        ID.BasicEnemy,
handler));
                    }
                    if (hud.getLevel() == 16) {
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.SmartEnemy, handler));
                        handler.addObject(new
SmartEnemy(r.nextInt(Game.WIDTH - 64), r.nextInt(Game.HEIGHT - 64),
                                ID.SmartEnemy, handler));
                    }
                    if (hud.getLevel() == 17) {
                        handler.clearEnemies();
                        handler.addObject(new
EnemyBoss2((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));
                        handler.addObject(new
EnemyBoss((Game.WIDTH / 2) - 64, -140, ID.EnemyBoss, handler));

                    }
                }
            }
        }
}
```

Shop.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
public class Shop extends MouseAdapter{
	Handler handler;
	HUD hud;
	private int B1 = 500;
	private int B2 = 500;
	private int B3 = 200;
	public Shop(Handler handler,HUD hud) {
		this.handler = handler;
		this.hud = hud;
	}
	public void render(Graphics g) {
		g.setColor(Color.green);
		g.setFont(new Font("Arial",0,48));
		g.drawString("SHOP", 250, 50);
		g.setFont(new Font("Arial",0,21));
		g.setColor(Color.white);
		//Box 1
		g.drawString("Upgrade Health", 60, 130);
		g.drawString("Cost: " + B1, 60, 160);
		g.drawRect(55, 105, 160, 70);
		//Box 2
		g.drawString("Upgrade Speed", 240, 130);
		g.drawString("Cost: " + B2, 240, 160);
		g.drawRect(235, 105, 160, 70);

		//Box 3
		g.drawString("Refill Health", 420, 130);
		g.drawString("Cost: " + B3, 420, 160);
		g.drawRect(415, 105, 160, 70);
		g.drawString("Your Score: " + hud.getScore(), 240, 80);
	}
	public void mousePressed(MouseEvent e) {
		int mx = e.getX();
```

```
                int my = e.getY();
                //for box 1
                if(mx >= 55 && mx <= 215) {
                        if(my >= 105 && my <= 175) {
                                if(hud.getScore() >= B1) {
                                        hud.setScore(hud.getScore() - B1);
                                        B1 += 500;
                                        hud.bounds += 30;
                                        hud.HEALTH = (100 +(hud.bounds/2));
                                }
                        }
                }
                //for box 2
                if(mx >= 235 && mx <= 385) {
                        if(my >= 105 && my <= 175) {
                                if(hud.getScore() >= B2) {
                                        hud.setScore(hud.getScore() - B2);
                                        B2 += 500;
                                        handler.move++;
                                }
                        }
                }
                //for box 3
                if(mx >= 415 && mx <= 575) {
                        if(my >= 105 && my <= 175 ) {
                                if(hud.getScore() >= B3) {
                                        hud.setScore(hud.getScore() - B3);
                                        B3 += 250;
                                        hud.HEALTH = (100 +(hud.bounds/2));
                                }
                        }
                }
        }
}
```

BasicEnemy.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
public class BasicEnemy extends GameObject{
```

```
	private Handler handler;
	public BasicEnemy(float x, float y, ID id,Handler handler) {
		super(x, y, id);
		this.handler = handler;
		velX = 5;
		velY = 5;
	}
	public void tick() {
		x += velX;
		y += velY;
		//to make the enemy bounce of  the walls
		if(y < 0 || y > Game.HEIGHT-48) velY*=-1;
		if(x < 0 || x > Game.WIDTH-32) velX*=-1;
		//for trail
		handler.addObject(new
Trail(x,y,ID.Trail,Color.red,16,16,0.02f,handler));
	}
	public void render(Graphics g) {
		g.setColor(Color.red);
		g.fillRect((int)x, (int)y, 16, 16);
	}
	public Rectangle getBounds() {
		return new Rectangle((int)x,(int)y,32,32);
	}
}
```

FastEnemy.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
public class FastEnemy extends GameObject{
	private Handler handler;
	public FastEnemy(float x, float y, ID id,Handler handler) {
		super(x, y, id);
		this.handler = handler;
		velX = 2;
		velY = 9;
	}
	public void tick() {
		x += velX;
```

```
		y += velY;
		//to make the enemy bounce of  the walls
		if(y < 0 || y > Game.HEIGHT-48) velY*=-1;
		if(x < 0 || x > Game.WIDTH-32) velX*=-1;
		//for trail
		handler.addObject(new
Trail(x,y,ID.Trail,Color.CYAN,16,16,0.02f,handler));
	}
	public void render(Graphics g) {
		g.setColor(Color.red);
		g.fillRect((int)x,(int) y, 16, 16);
	}
	public Rectangle getBounds() {
		return new Rectangle((int)x,(int)y,32,32);
	}
}
```

SmartEnemy.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
public class SmartEnemy extends GameObject{

	private Handler handler;
	private GameObject player;
	public SmartEnemy(float x, float y, ID id,Handler handler) {
		super(x, y, id);
		this.handler = handler;
		for(int i = 0;i < handler.object.size();i++) {
			if(handler.object.get(i).getId() == ID.Player)
				player = handler.object.get(i);
		}
	}
	public void tick() {
		x += velX;
		y += velY;
		//code to follow the player
		float diffX = x - player.getX() - 16;
		float diffY = y - player.getY() - 16;
```

```
                float distance = (float) Math.sqrt((x - player.getX()) * (x -
player.getX()) + (y - player.getY()) * (y - player.getY()));
                velX = ((-1/distance) * diffX);
                velY = ((-1/distance) * diffY);
                //for trail
                handler.addObject(new
Trail(x,y,ID.Trail,Color.MAGENTA,16,16,0.02f,handler));
        }
        public void render(Graphics g) {
                g.setColor(Color.MAGENTA);
                g.fillRect((int)x,(int) y, 16, 16);
        }

        public Rectangle getBounds() {
                return new Rectangle((int)x,(int)y,32,32);
        }
}
```

HardEnemy.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Random;
public class HardEnemy extends GameObject{
        private Handler handler;
        private Random r = new Random();
        public HardEnemy(float x, float y, ID id,Handler handler) {
                super(x, y, id);
                this.handler = handler;
                velX = 5;
                velY = 5;
        }
        public void tick() {
                x += velX;
                y += velY;
                //to make the enemy bounce of  the walls
                if(y < 0 || y > Game.HEIGHT-64) {
                        if(velY<0)
                                velY = -(r.nextInt(7)+1)*-1;
                        else
```

```
                        velY = (r.nextInt(7)+1)*-1;
                }
                if(x < 0 || x > Game.WIDTH-48) {
                        if(velX<0)
                        velX = -(r.nextInt(7)+1)*-1;
                else
                        velX = (r.nextInt(7)+1)*-1;
                 }
                //for trail
                handler.addObject(new
Trail(x,y,ID.Trail,Color.yellow,16,16,0.02f,handler));
        }
        public void render(Graphics g) {
                g.setColor(Color.yellow);
                g.fillRect((int)x, (int)y, 16, 16);
        }
        public Rectangle getBounds() {
                return new Rectangle((int)x,(int)y,32,32);
        }
}
```

EnemyBoss.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Random;
public class EnemyBoss extends GameObject
        private Handler handler;
        Random r = new Random();
        private int timer = 100;
        private int timer2 = 50;
        public EnemyBoss(float x, float y, ID id,Handler handler) {
                super(x, y, id);
                this.handler = handler;
                velX = 0;
                velY = 2;
        }
        public void tick() {
                x += velX;
                y += velY;
```

```
		if(timer<=0)
		{
			velY = 0;
			timer2--;
		}
		else
			timer--;

		if(timer2 <= 0) {
			if(velX == 0) { velX = 2;}
			int spawn = r.nextInt(10);
			if(spawn == 0) handler.addObject(new
EnemyBossBullet((int)x + 48, (int)y + 48, ID.BasicEnemy, handler));
		}
		if(x < 0 || x > Game.WIDTH-112) velX*=-1;
	}
	public void render(Graphics g) {
		g.setColor(Color.red);
		g.fillRect((int)x, (int)y, 96, 96);
	}
	public Rectangle getBounds() {
		return new Rectangle((int)x,(int)y,96,96);
	}
}
```

EnemyBoss2.java

```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Random
public class EnemyBoss2 extends GameObject{
	private Handler handler;
	Random r = new Random();
	private int timer = 100;
	private int timer2 = 50;
	public EnemyBoss2(float x, float y, ID id,Handler handler) {
		super(x, y, id);
		this.handler = handler;
		velX = 0;
		velY = 2;
```

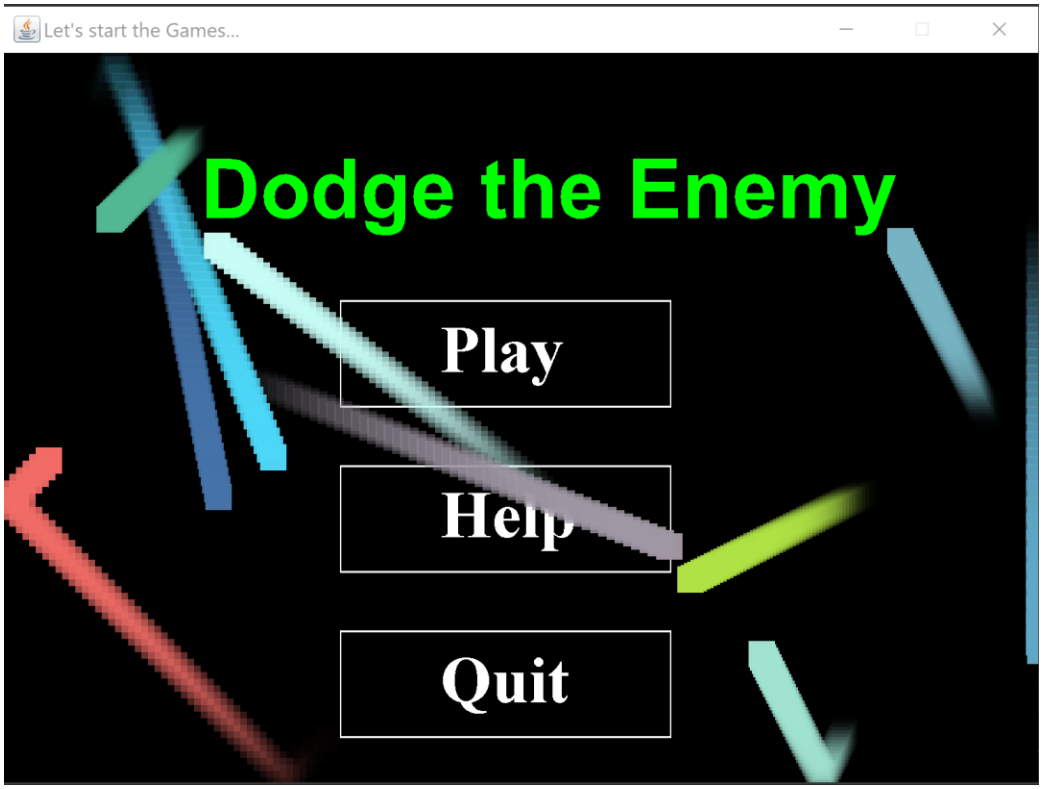
```
}
	public void tick() {
		x += velX;
		y += velY;
		if(timer<=0)
		{
			velY = 0;
			timer2--;
		}
		else
			timer--;
		if(timer2 <= 0) {
			if(velX == 0) { velX = -2;}
			int spawn = r.nextInt(10);
			if(spawn == 0) handler.addObject(new
EnemyBossBullet((int)x + 48, (int)y + 48, ID.BasicEnemy, handler));
		}
		if(x < 0 || x > Game.WIDTH-112) velX*=-1;
	}
	public void render(Graphics g) {
		g.setColor(Color.red);
		g.fillRect((int)x, (int)y, 96, 96);
	}
	public Rectangle getBounds() {
		return new Rectangle((int)x,(int)y,96,96);
	}
}
```

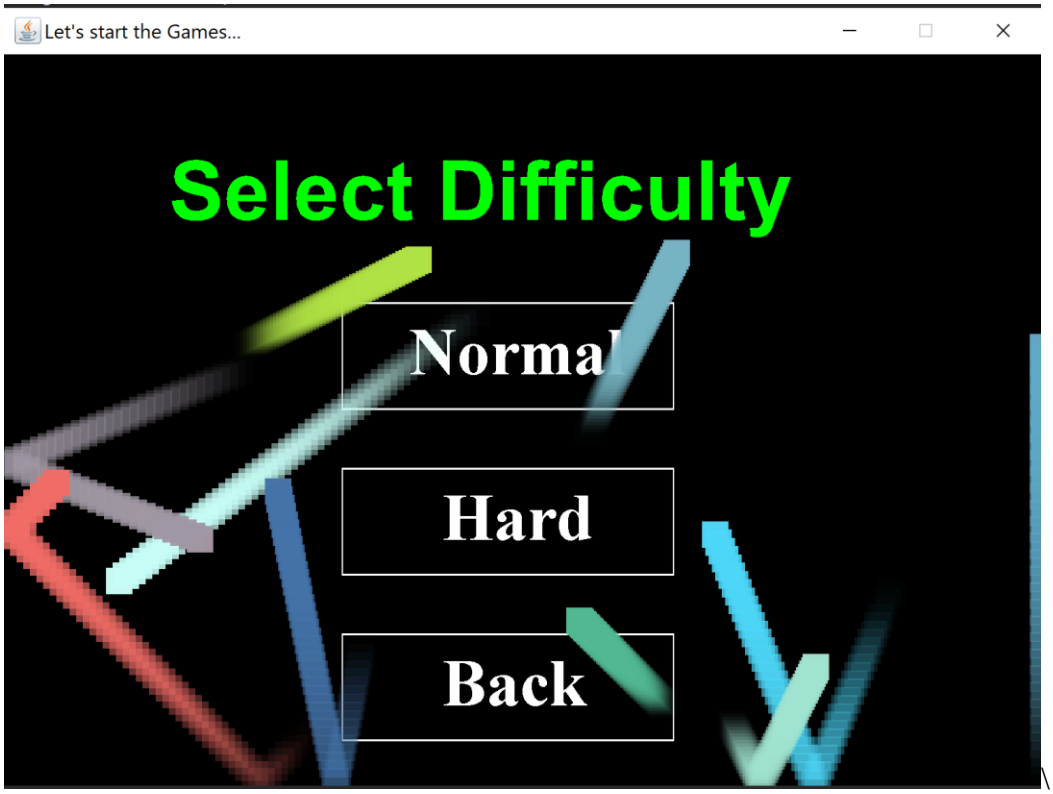
EnemyBossBullet.java

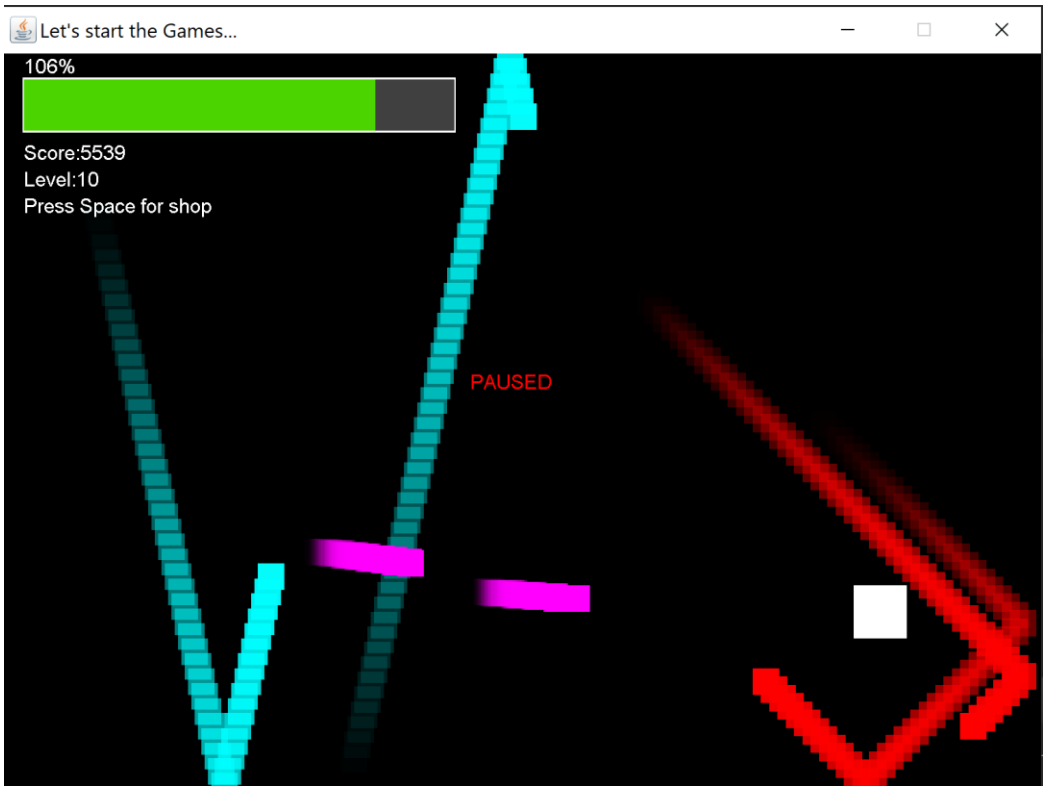
```
package Main_Project;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.util.Random;
public class EnemyBossBullet extends GameObject{
	private Handler handler;
	Random r = new Random();
	public EnemyBossBullet(float x, float y, ID id,Handler handler) {
		super(x, y, id);
		this.handler = handler;
		velX = (r.nextInt(5 - -5) + -5);
```

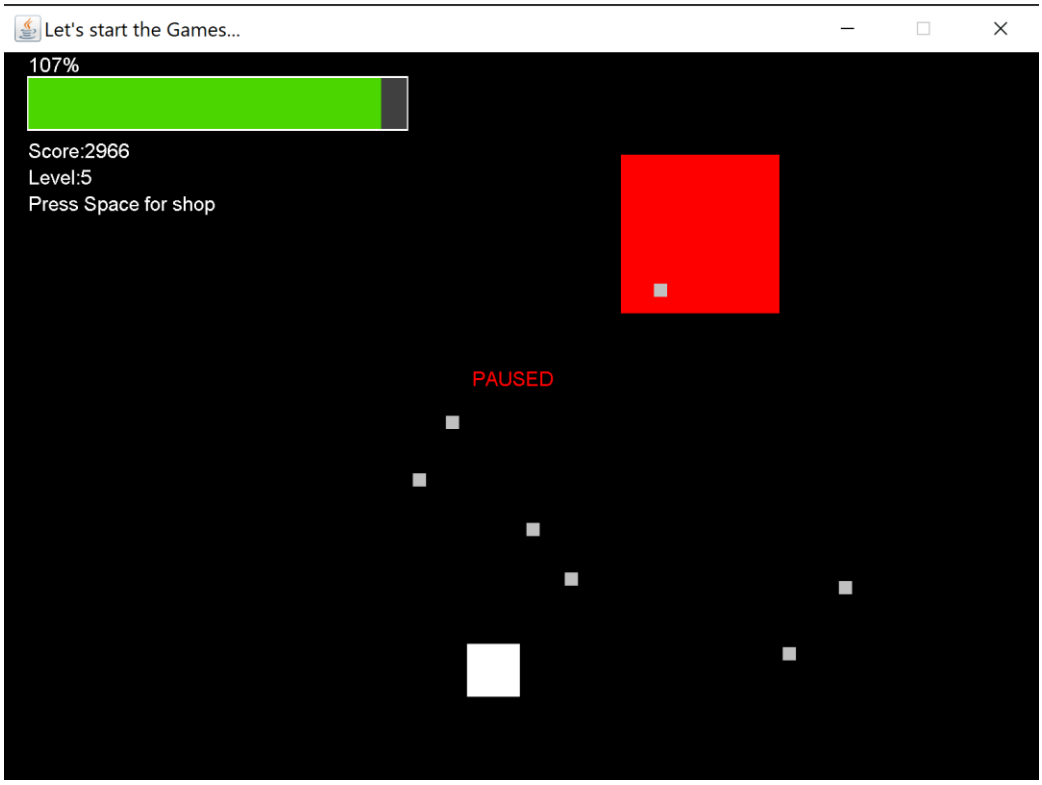
```
            velY = 5;
        }
        public void tick() {
            x += velX;
            y += velY;
            //to make the bullet disappear
            if(y>=Game.HEIGHT) handler.removeObject(this);
            //for trail
            //handler.addObject(new
Trail(x,y,ID.Trail,Color.LIGHT_GRAY,8,8,0.05f,handler));
        public void render(Graphics g) {
            g.setColor(Color.LIGHT_GRAY);
            g.fillRect((int)x, (int)y, 8, 8);
        }
        public Rectangle getBounds() {
            return new Rectangle((int)x,(int)y,8,8);
        }
}
```

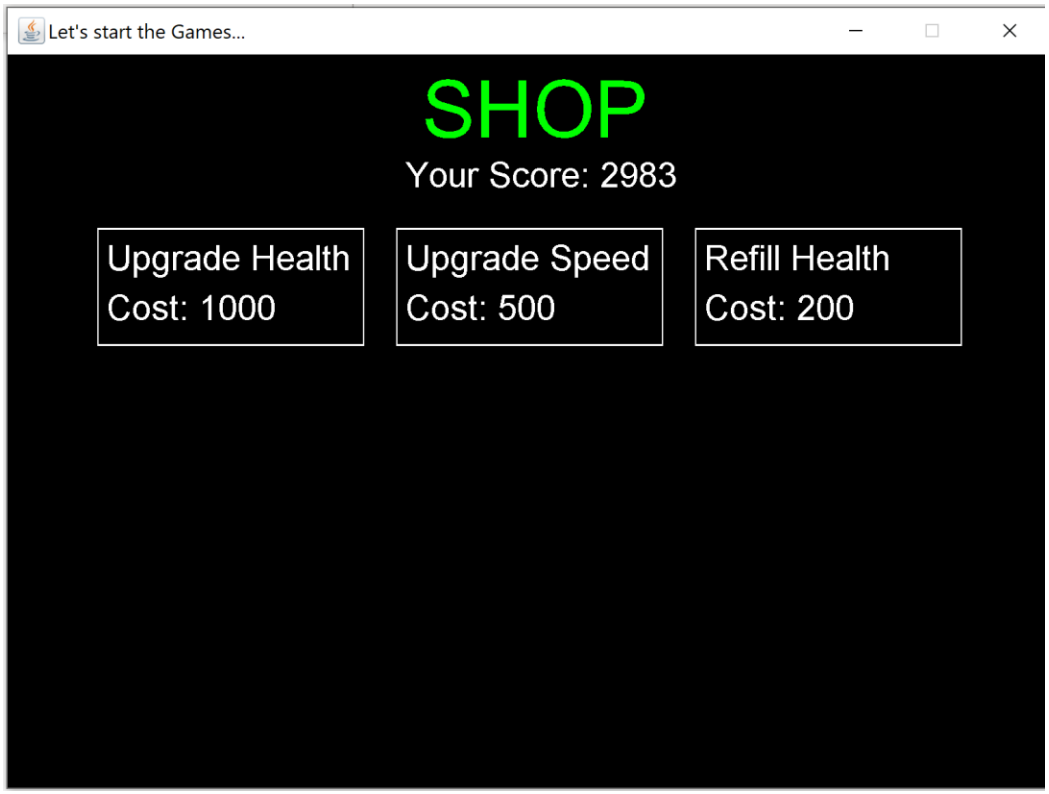
Screenshots of the Game

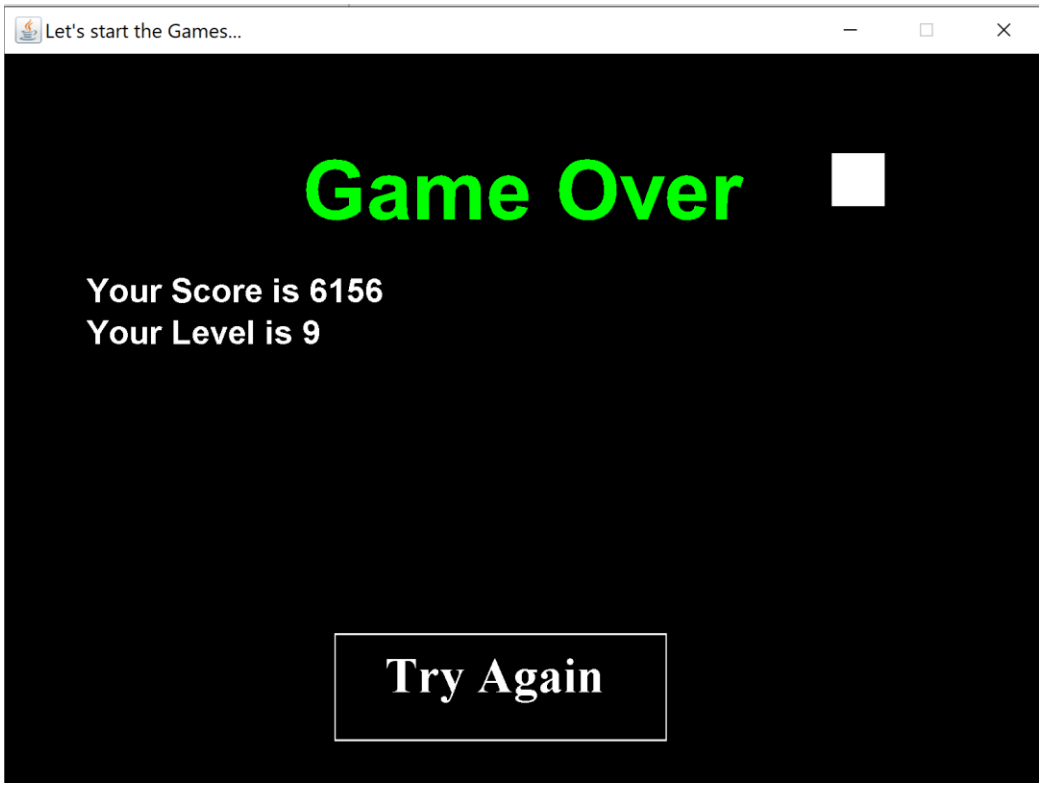





Let's start the Games...
106%
Score:5539
Level:10
Press Space for shop
PAUSED


Let's start the Games...
107%
Score:2966
Level:5
Press Space for shop
PAUSED


Let's start the Games...
SHOP
Your Score: 2983
Upgrade Health
Cost: 1000
Upgrade Speed
Cost: 500
Refill Health
Cost: 200


Let's start the Games...
Game Over
Your Score is 6156
Your Level is 9
Try Again