Project Title:

Block Game

Sebastian and Immanuel

Introduction to Game

- Our aim for the game, is to shoot the enemy which is on the screen.

- The blue block is the character which the player will be controlling, which the red block is the enemy which the player is trying to shoot.

Features of the game

- The features which have been coded within the game, are shooting, movement, reloading the game and collision.

- The shooting feature within the game when the player presses 'spacebar' the blue block will shoot, from its left side, a small red block.

- The blue block has been coded, for its movement controls to be the up, down, left, right arrow keys on the keyboard.

- The movement speed of the player is much faster, compared to the enemy, making the game easier to play.

- When the player successfully hits the red block with the shot, the game will pause and allow the player to start again. As seen here on the right.

- The collision feature within the game, makes the red block follow the player to try and get closer.

Design

Initial sketch ideas for the game

This is the initial sketch idea for the game. The idea was that the wizard, the character which the player will be controlling, will be shooting waves of enemies.
A score bar is also shown at the top of the screen too.

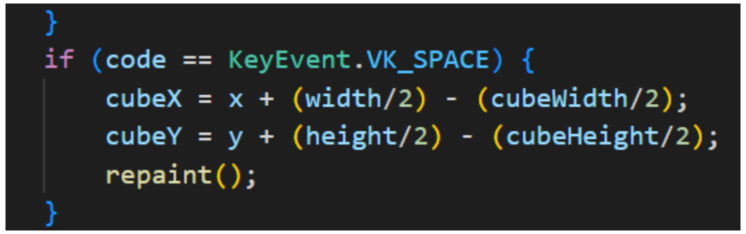
Comparison for final idea

Development process

When it came to the development of the game, we decided that Immanuel was going to code the game, and for me to do the presentation.

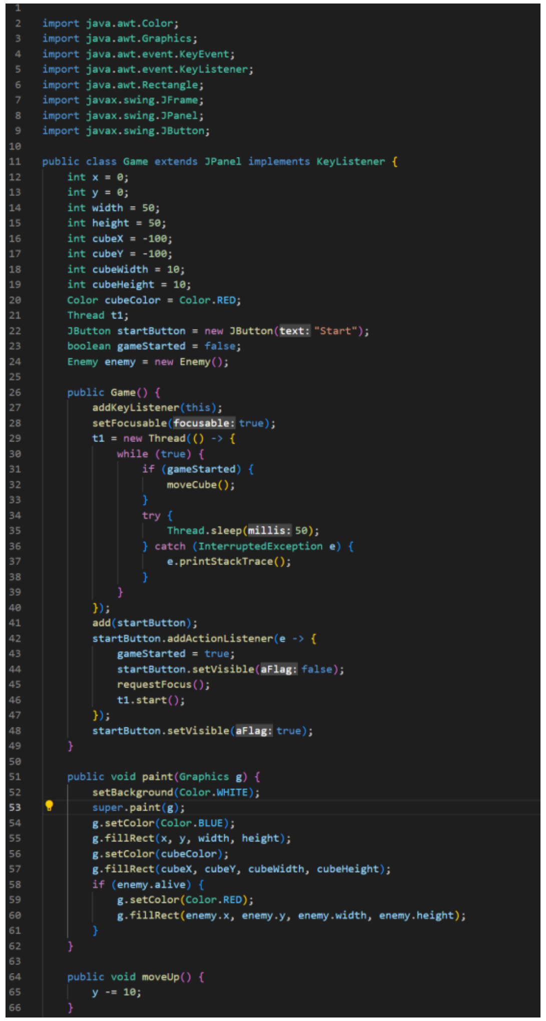
The issues which came up whilst coding the code, was the collision system not fully wording. For example, when the red block touches the blue block the player does not die, but instead the red block rapidly moves within the blue block.

An other issue which happened whilst coding the game, is that the blue block would not shoot. Leading to the entire game not working. This issue was overcome from changing the shooting button to spacebar, from double pressing left.

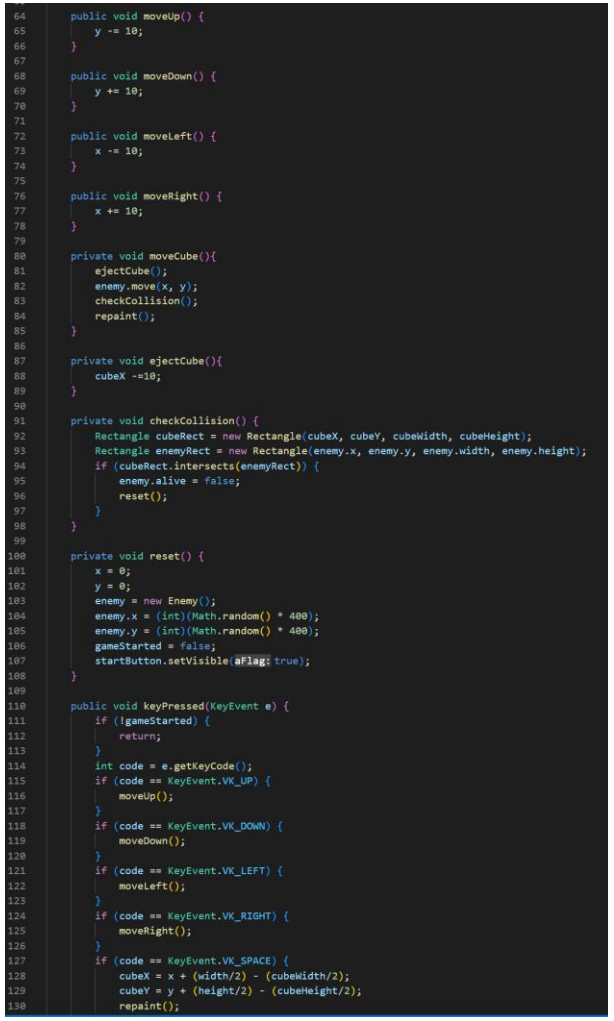


```
}
if (code == KeyEvent.VK_SPACE) {
    cubeX = x + (width/2) - (cubeWidth/2);
    cubeY = y + (height/2) - (cubeHeight/2);
    repaint();
}
```

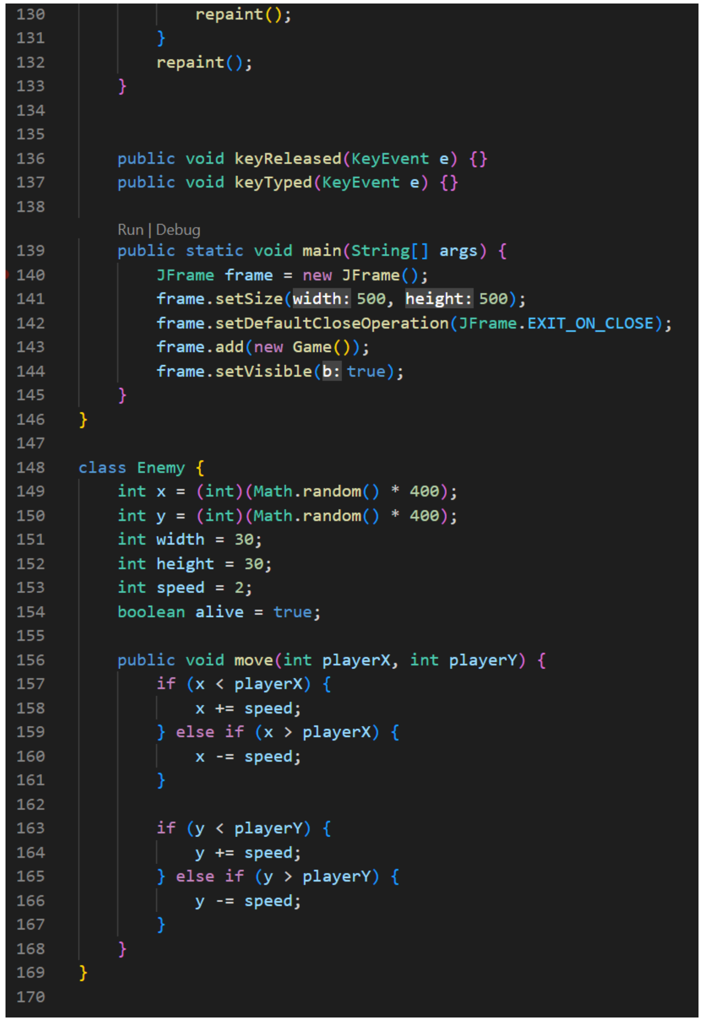
Games code



```java

import java.awt.Color;
import java.awt.Graphics;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.Rectangle;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JButton;

public class Game extends JPanel implements KeyListener {
    int x = 0;
    int y = 0;
    int width = 50;
    int height = 50;
    int cubeX = -100;
    int cubeY = -100;
    int cubeWidth = 10;
    int cubeHeight = 10;
    Color cubeColor = Color.RED;
    Thread t1;
    JButton startButton = new JButton(text: "Start");
    boolean gameStarted = false;
    Enemy enemy = new Enemy();

    public Game() {
        addKeyListener(this);
        setFocusable(focusable: true);
        t1 = new Thread(() -> {
            while (true) {
                if (gameStarted) {
                    moveCube();
                }
                try {
                    Thread.sleep(millis: 50);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });
        add(startButton);
        startButton.addActionListener(e -> {
            gameStarted = true;
            startButton.setVisible(aFlag: false);
            requestFocus();
            t1.start();
        });
        startButton.setVisible(aFlag: true);
    }

    public void paint(Graphics g) {
        setBackground(Color.WHITE);
        super.paint(g);
        g.setColor(Color.BLUE);
        g.fillRect(x, y, width, height);
        g.setColor(cubeColor);
        g.fillRect(cubeX, cubeY, cubeWidth, cubeHeight);
        if (enemy.alive) {
            g.setColor(Color.RED);
            g.fillRect(enemy.x, enemy.y, enemy.width, enemy.height);
        }
    }

    public void moveUp() {
        y -= 10;
    }
```



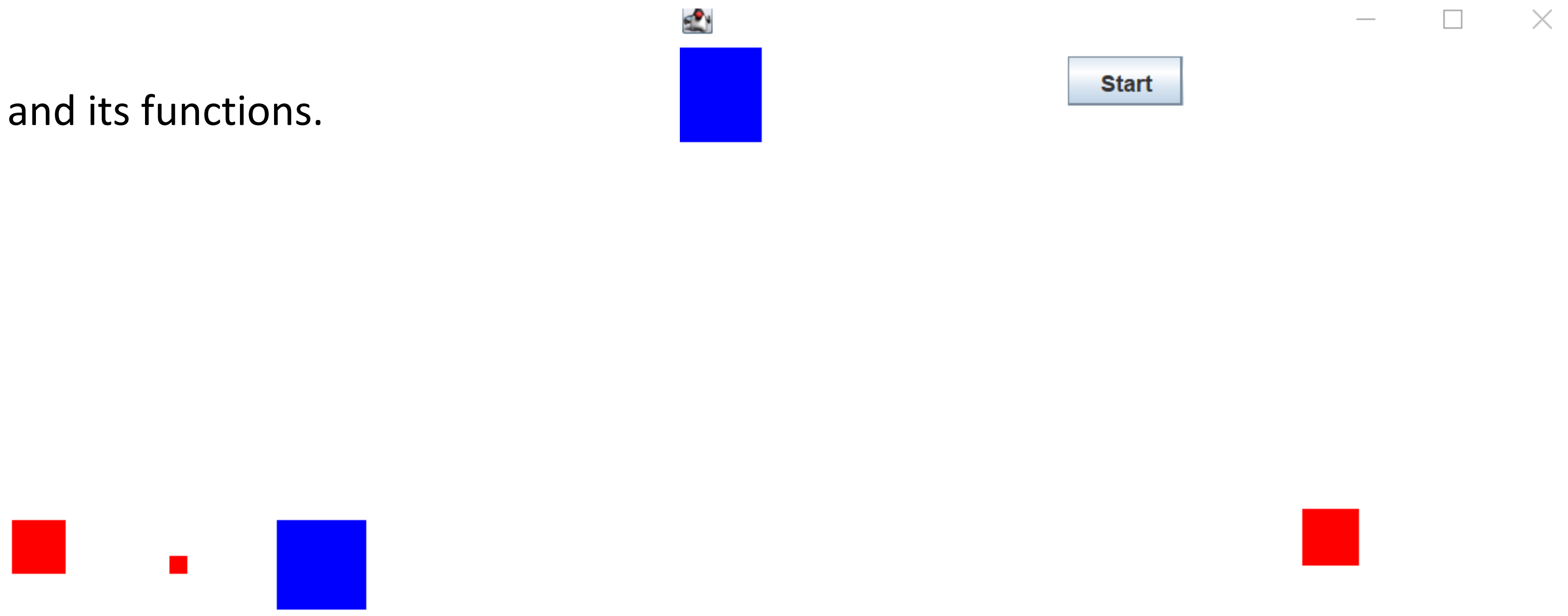
```java
    public void moveUp() {
        y -= 10;
    }

    public void moveDown() {
        y += 10;
    }

    public void moveLeft() {
        x -= 10;
    }

    public void moveRight() {
        x += 10;
    }

    private void moveCube(){
        ejectCube();
        enemy.move(x, y);
        checkCollision();
        repaint();
    }

    private void ejectCube(){
        cubeX -=10;
    }

    private void checkCollision() {
        Rectangle cubeRect = new Rectangle(cubeX, cubeY, cubeWidth, cubeHeight);
        Rectangle enemyRect = new Rectangle(enemy.x, enemy.y, enemy.width, enemy.height);
        if (cubeRect.intersects(enemyRect)) {
            enemy.alive = false;
            reset();
        }
    }

    private void reset() {
        x = 0;
        y = 0;
        enemy = new Enemy();
        enemy.x = (int)(Math.random() * 400);
        enemy.y = (int)(Math.random() * 400);
        gameStarted = false;
        startButton.setVisible(aFlag: true);
    }

    public void keyPressed(KeyEvent e) {
        if (!gameStarted) {
            return;
        }
        int code = e.getKeyCode();
        if (code == KeyEvent.VK_UP) {
            moveUp();
        }
        if (code == KeyEvent.VK_DOWN) {
            moveDown();
        }
        if (code == KeyEvent.VK_LEFT) {
            moveLeft();
        }
        if (code == KeyEvent.VK_RIGHT) {
            moveRight();
        }
        if (code == KeyEvent.VK_SPACE) {
            cubeX = x + (width/2) - (cubeWidth/2);
            cubeY = y + (height/2) - (cubeHeight/2);
            repaint();
```



```java
            repaint();
        }
        repaint();
    }


    public void keyReleased(KeyEvent e) {}
    public void keyTyped(KeyEvent e) {}

    Run | Debug
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setSize(width: 500, height: 500);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.add(new Game());
        frame.setVisible(b: true);
    }
}

class Enemy {
    int x = (int)(Math.random() * 400);
    int y = (int)(Math.random() * 400);
    int width = 30;
    int height = 30;
    int speed = 2;
    boolean alive = true;

    public void move(int playerX, int playerY) {
        if (x < playerX) {
            x += speed;
        } else if (x > playerX) {
            x -= speed;
        }

        if (y < playerY) {
            y += speed;
        } else if (y > playerY) {
            y -= speed;
        }
    }
}

```

Demonstration

Present the game and its functions.



Evaluation

What we believe went well whilst coding the game, is that the game is able to run and play.

We could improve the game next time by adding avatars to the blue and red blocks. This will bring more life to the game and overall improve the experience for the player.

What we have learnt is that, coding a game requires a lot more time, to finish and make sure it is fully working.