**All Functions and Source Code**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a black screen

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Source Code Below**

**/\***

**\* Audio.java**

**\*/**

import javax.sound.sampled.AudioInputStream;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import java.io.File;

public class Audio {

String[] song = {"Music/1.wav", "Music/2.wav", "Music/3.wav", "Music/4.wav", "Music/5.wav", "Music/background.wav",};

Clip clip;

AudioInputStream audioInputStream;

public Audio() {

}

public Audio(int s) {

try {

audioInputStream = AudioSystem.getAudioInputStream(new File(song[s]).getAbsoluteFile());

clip = AudioSystem.getClip();

clip.open(audioInputStream);

} catch (Exception e) {

e.printStackTrace();

}

}

public void playAudio() {

clip.start();

clip.loop(Clip.LOOP\_CONTINUOUSLY);

}

public void stopAudio() {

clip.stop();

}

}

**/\***

**\* Button.java**

**\*/**

/\*

\* WHITE = 0x000000, rgba: (255,255,255,0/1), (0,0,0,0), or

\* BLACK = 0xFFFFFF, rgba: (0,0,0,1)

\* RED = = 0xFF0000, rgba: (255,0,0,0.00-1.00)

\* GREEN = 0x00FF00

\* BLUE = 0x0000FF

\* LIME\_GREEN =

\* HOT\_PINK = 0xFF00FF

\* PURPLE = 0x800080

\*/

import java.awt.\*;

public class Button {

int x = 0;

int y = 0;

boolean keyPressed = false;

public void drawCircle(Graphics g, int move) {

if (move != 1) {

g.setColor(Color.BLACK); // // g.setColor(new Color(0xFFFFFF));

g.fillOval(504, 234, 41, 41);

g.setColor((Color.GREEN)); // g.setColor((Color.green));

g.fillOval(500, 230, 40, 40);

}

if (move == 1) {

g.setColor(Color.DARK\_GRAY); // g.setColor((Color.red));

g.fillOval(504, 234, 40, 40);

}

if (move != 2) {

g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));

g.fillOval(504, 334, 41, 41);

g.setColor(Color.RED); // g.setColor((Color.green));

g.fillOval(500, 330, 40, 40);

}

if (move == 2) {

g.setColor(Color.DARK\_GRAY); // g.setColor((Color.red));

g.fillOval(504, 334, 40, 40);

}

if (move != 3) {

g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF))

g.fillOval(504, 434, 41, 41);

g.setColor(Color.YELLOW); // g.setColor((Color.green));

g.fillOval(500, 430, 40, 40);

}

if (move == 3) {

g.setColor(Color.DARK\_GRAY); // g.setColor((Color.red));

g.fillOval(504, 434, 40, 40);

}

if (move != 4) {

g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));

g.fillOval(504, 534, 41, 41);

g.setColor(Color.BLUE); // g.setColor((Color.green));

g.fillOval(500, 530, 40, 40);

}

if (move == 4) {

g.setColor(Color.DARK\_GRAY); // g.setColor((Color.red));

g.fillOval(504, 534, 40, 40);

}

if (move != 5) {

g.setColor(Color.BLACK); // g.setColor(new Color(0xFFFFFF));

g.fillOval(504, 634, 41, 41);

g.setColor(Color.ORANGE); // g.setColor((Color.green));

g.fillOval(500, 630, 40, 40);

}

if (move == 5) {

g.setColor(Color.DARK\_GRAY); // g.setColor((Color.red));

g.fillOval(504, 634, 40, 40);

}

}

public void drawTriangle(Graphics g, int move) {

//g.setColor(Color.WHITE); // g.setColor(new Color(0xFFFFFFFF, true));

//g.fillPolygon(new int[]{513,513,533}, new int[]{242,262,252},3);

g.setColor(Color.blue);

if (move == 1) {

g.setColor(Color.WHITE);

g.fillPolygon(new int[] {514, 514, 534}, new int[] {245, 265, 255}, 3);

}

if (move != 1) {

g.setColor(Color.BLACK);

g.fillPolygon(new int[] {510, 510, 530}, new int[] {241, 261, 251}, 3);

}

if (move == 2) {

g.setColor(Color.WHITE);

g.fillPolygon(new int[] {514, 514, 534}, new int[] {345, 365, 355}, 3);

}

if (move != 2) {

g.setColor(Color.BLACK);

g.fillPolygon(new int[] {510, 510, 530}, new int[] {341, 361, 351}, 3);

}

if (move == 3) {

g.setColor(Color.WHITE);

g.fillPolygon(new int[] {514, 514, 534}, new int[] {445, 465, 455}, 3);

}

if (move != 3) {

g.setColor(Color.BLACK);

g.fillPolygon(new int[] {510, 510, 530}, new int[] {441, 461, 451}, 3);

}

if (move == 4) {

g.setColor(Color.WHITE);

g.fillPolygon(new int[] {514, 514, 534}, new int[] {545, 565, 555}, 3);

}

if (move != 4) {

g.setColor(Color.BLACK);

g.fillPolygon(new int[] {510, 510, 530}, new int[] {541, 561, 551}, 3);

}

if (move == 5) {

g.setColor(Color.WHITE);

g.fillPolygon(new int[] {514, 514, 534}, new int[] {645, 665, 655}, 3);

}

if (move != 5) {

g.setColor(Color.BLACK);

g.fillPolygon(new int[] {510, 510, 530}, new int[] {641, 661, 651}, 3);

}

}

// Version 4

public void gameButton(Graphics g, boolean[] key) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

String[] letters = {"1", "2", "3", "4"};

String[] buttonColors = {"GREEN", "Color.RED", "Color.YELLOW", "Color.BLUE"};

Color[] colors = {Color.GREEN, Color.YELLOW, Color.RED, Color.BLUE};

Color[] darkColors = {Color.GREEN, Color.YELLOW, Color.RED, Color.BLUE};

int[] x = {45, 195, 345, 495};

int y = 625;

for (int i = 0; i < 4; i++) {

g2.setColor(key[i] ? colors[i].darker() : colors[i]);

g2.fillOval(x[i], y, 60, 60);

g2.setColor(Color.BLACK);

g2.setFont(new Font("SansSerif", Font.BOLD, 32));

g2.drawString(letters[i], x[i] + 20, y + 42);

}

}

/\*

// Version 3

public void gameButton(Graphics g, boolean[] key) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

String[] letters = {"A","S","J","K"};

int[] x = {45,195,345,495};

int y = 625;

for(int i=0;i<4;i++) {

g2.setColor(key[i] ? new Color(40,40,40) : new Color(220,220,220));

g2.fillOval(x[i], y, 60, 60);

g2.setColor(Color.BLACK);

g2.setFont(new Font("SansSerif",Font.BOLD,32));

g2.drawString(letters[i], x[i]+20, y+42);

}

}

\*/

/\*

// Version 2

public void gameButton (Graphics g, boolean[] key) {

String[] letters = {"A","S","J","K"};

int[] x = {45,195,345,495};

int y = 625;

}

g.setFont(new Font("monospaced",Font.BOLD,50));

if(!key[0]) {

g.setColor(new Color(0x000000)); // 0xFFFFFFF = RED

g.fillOval(45, 625, 60, 60);

g.setColor((Color.green));

g.fillOval(40, 620, 60, 60);

g.setColor(Color.GREEN);

g.drawString("1",57,665);

}

if(key[0]) {

g.setColor((Color.red));

g.fillOval(45, 625, 60, 60);

g.setColor(Color.BLUE);

g.drawString("1",62,670);

}

if(!key[1]) {

g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED

g.fillOval(195, 625, 60, 60);

g.setColor((Color.green));

g.fillOval(190, 620, 60, 60);

g.setColor(Color.BLUE);

g.drawString("2",207,665);

}

if(key[1]) {

g.setColor((Color.red));

g.fillOval(195, 625, 60, 60);

g.setColor(Color.BLUE);

g.drawString("2",212,670);

}

if(!key[2]) {

g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED

g.fillOval(345, 625, 60, 60);

g.setColor((Color.green));

g.fillOval(340, 620, 60, 60);

g.setColor(Color.BLUE);

g.drawString("3",357,665);

}

if(key[2]) {

g.setColor((Color.red));

g.fillOval(345, 625, 60, 60);

g.setColor(Color.BLUE);

g.drawString("3",362,670);

}

if(!key[3]) {

g.setColor(new Color(0xFFFFFF)); // 0xFFFFFFF = RED

g.fillOval(495, 625, 60, 60);

g.setColor((Color.green));

g.fillOval(490, 620, 60, 60);

g.setColor(Color.BLUE);

g.drawString("4",507,665);

}

if(key[3]) {

g.setColor((Color.red));

g.fillOval(495, 625, 60, 60);

g.setColor(Color.BLUE);

g.drawString("4",512,670);

}

}

\*/

}

**/\***

**\* GameText.java**

**\*/**

import java.awt.\*;

public class GameText {

private String TITLE = "Manuvo";

public void gameName(Graphics g) {

g.setFont(new Font(Font.DIALOG\_INPUT, Font.BOLD, 90));

g.setColor(Color.GRAY); // g.setColor(new Color(0xFFFFFF00, true));

g.drawString(TITLE, 264, 100);

g.setColor(Color.BLUE); // g.setColor(new Color(156, 112, 248, 255));

g.drawString(TITLE, 260, 100);

g.setColor(new Color(0xFFFFFF00, true)); // 0xFFFFFFFF = Black

/\*

g.drawString("Tiles",264,190);

g.setColor(new Color(156, 112, 248, 255)); // Color(156, 112, 248, 255) = Purple

g.drawString("Tiles",260,190);

\*/

}

public void songText(Graphics g) {

g.setColor(new Color(114, 222, 210, 255)); // Color(156, 112, 248, 255) = Ugly Cyan\_Green

g.setFont(new Font("monospaced Bold", Font.ITALIC, 30));

g.drawString("Someone you Loved", 180, 230);

g.drawString("Memories", 180, 330);

g.drawString("Play Date", 180, 430);

g.drawString("Dance Monkey", 180, 530);

g.drawString("Counting Stars", 180, 630);

}

// Guitar = G, R, Y, Bl, O

public void difficultyText(Graphics g) {

g.setFont(new Font("serif", Font.BOLD, 20));

// Green

//g.setColor(new Color(0x0FFD0FF, true)); // 0xFFD0FF00 = Pink

g.setColor(new Color(0xCC00FF00, true)); // 0xFFD0FF00 = Purple

g.drawString("Very Easy", 400, 280);

g.setColor(Color.RED);

g.drawString("Easy", 400, 380);

g.setColor(new Color(0xFFFFFF00, true)); // 0x099202 = Green

g.drawString("Medium", 400, 480);

g.setColor(Color.BLUE);

g.drawString("Hard", 400, 580);

g.setColor(Color.ORANGE);

g.drawString("Very Hard", 400, 680);

}

public void gameOver(Graphics g, int score) {

g.setFont(new Font("serif", Font.BOLD, 50));

g.setColor(new Color(0xFFFFFFFF, true)); // g.setColor(new Color(0xFFE600));

g.drawString("Score: " + score, 180, 255);

g.setColor(new Color(0xFFFFFFFF)); // g.setColor(new Color(0x800000));

g.drawString("Game Over", 150, 325);

g.setColor(new Color(0xFFFFFFFF)); // g.setColor(new Color(0x26FF00));

g.drawString("Press Enter", 160, 400);

}

// Version 4

public void score(Graphics g, int score, String compliment, int complimentSize) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_TEXT\_ANTIALIASING, RenderingHints.VALUE\_TEXT\_ANTIALIAS\_ON);

g2.setColor(Color.BLACK);

g2.setFont(new Font("SansSerif", Font.BOLD, 48));

g2.drawString(String.valueOf(score), 270, 70);

g2.setFont(new Font("SansSerif", Font.PLAIN, complimentSize));

g2.setColor(new Color(140, 140, 140)); // soft gray

g2.drawString(compliment, 230, 140);

}

/\*

// Version 3

public void score(Graphics g,int score, String compliment, int complimentSize) {

Graphics2D g2 = (Graphics2D) g;

g2.setFont(new Font("SansSerif",Font.BOLD,50));

g2.setColor(Color.white);

g2.drawString(String.valueOf(score),270,70);

g2.setFont(new Font("SansSerif",Font.BOLD,complimentSize));

g2.setColor(new Color(255, 255, 255, 200));

g2.drawString(compliment,230,140);

}

\*/

/\*

// Version 2

public void score(Graphics g,int score, String compliment, int complimentSize) {

g.setColor(new Color(0xFFFFFF)); // 0xFFFFFF = White

g.setFont(new Font("serif",Font.BOLD,50));

g.drawString(String.valueOf(score),300,70);

if(compliment.equals("Perfect"))

g.setColor(new Color(0x630061)); // 0x630061 = Purple

else if(compliment.equals("Great"))

g.setColor(new Color(0x001E99)); // 0x001E99 = Blue

else

g.setColor(new Color(0x008787)); // 0x008787 = Cyan

g.setFont(new Font("serif",Font.BOLD,complimentSize));

if(complimentSize == 50)

g.drawString(compliment,280,120);

else

g.drawString(compliment,230,140);

}

\*/

}

**/\***

**\* Main.java**

**\*/**

import javax.swing.\*;

public class Main {

public static void main(String[] args) {

JFrame frame = new JFrame();

MainScreen mainScreen = new MainScreen();

//GamePlay gamePlay = new GamePlay();

frame.setBounds(400,10,600,750); // frame.setBounds(x,y,width, height);

frame.setTitle("Manuvo");

frame.setResizable(true);

frame.setVisible(true);

frame.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

frame.add(mainScreen);

}

}

**/\***

**\* MainScreen.java**

**\*/**

import javax.imageio.ImageIO;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import java.util.Arrays;

import java.util.Random;

public class MainScreen extends JPanel implements KeyListener, ActionListener {

private Timer time;

private int delay = 15;

private boolean play = false;

private boolean over = false;

Random rand = new Random();

private int[] backBallX = new int[40];

private int[] backBallY = new int[40];

private boolean check = true;

private final int MISTAKE = 50;

//private final int bottomBond = 440;

private final int bottomBond = 450; // Hit Tile Out-Of-Bounds now

private int x = 0;

private int score = 0;

private int foulY = 0;

private int foulPlace;

private int totalTiles = 0;

private int move = 1;

private int selectRectY = 200;

private int speed = 5;

private boolean go = false;

private boolean key[] = new boolean[5];

private boolean tilesCheck[] = new boolean[5];

private boolean checkTilesProduce[] = new boolean[5];

private int tilesY[] = new int[5];

private String compliment = "";

private int complimentSize = 50; // Compliment {"Perfect","Great"}

private boolean foul = false;

private boolean startSong = false;

private boolean audioStatus = false;

private boolean backgroundAudioStatus = false;

Audio audio;

Audio backgroundAudio;

//Audio2 audio2;

//Audio3 audio3;

//Audio4 audio4;

//Audio5 audio5;

//BackgroundAudio backgroundAudio;

String[] images = {

"Images/someoneYouLoved.jpeg",

"Images/memories.jpg",

"Images/playDate.jpeg",

"Images/danceMonkey.jpeg",

"Images/countingStars.jpeg",

"Images/HandGrip2.jpg"

};

BufferedImage[] songImage = new BufferedImage[5];

BufferedImage iconImage = null;

/\* BufferedImage song1 = null;

BufferedImage song2 = null;

BufferedImage song3 = null;

BufferedImage song4 = null;

BufferedImage song5 = null;

BufferedImage iconImage = null; \*/

public MainScreen() {

initialValues();

addKeyListener(this);

setFocusable(true);

setFocusTraversalKeysEnabled(false);

time = new Timer(delay, this);

time.start();

}

public void initialValues() {

compliment = "";

totalTiles = 0;

score = 0;

delay = 15;

play = false;

over = false;

check = true;

x = 0;

move = 1;

selectRectY = 200; // height of Song and Tile Hit rectangles

go = false;

foulY = 0;

audioStatus = false;

startSong = false;

backgroundAudioStatus = false;

foul = false;

//songImage = null;

/\* song1 = null;

song2 = null;

song3 = null;

song4 = null;

song5 = null; \*/

for (int i = 0; i < 5; i++) {

key[i] = false;

tilesCheck[i] = false;

checkTilesProduce[i] = false;

tilesY[i] = -150; // Hit Tile Height = 150px tall

}

for (int i = 0; i < 40; i++) {

backBallX[i] = rand.nextInt(550);

backBallY[i] = rand.nextInt(700);

}

}

public void paint(Graphics graphics) {

//BackGround

backGround(graphics);

//backBalls(graphics);

//name

GameText gameText = new GameText();

gameText.gameName(graphics);

//icon

icon(graphics);

//song

//playDate, danceMonkey, memories, countingStars, someoneYouLoved.

drawSongImage(graphics);

getSongImage(check);

check = false;

//songName

gameText.songText(graphics);

//difficulty

gameText.difficultyText(graphics);

//playButtonIn

Button button = new Button();

button.drawCircle(graphics, move);

//buttonTriangleIn

button.drawTriangle(graphics, move);

Song1 song1 = new Song1();

if (!go && !backgroundAudioStatus) {

backgroundAudio = new Audio(5);

backgroundAudio.playAudio();

backgroundAudioStatus = true;

}

if (go) {

backgroundAudio.stopAudio();

}

if (move == 1) {

speed = 2;

if (startSong) {

audio = new Audio(0);

audio.playAudio();

audioStatus = true;

startSong = false;

}

if (over && audioStatus) {

audio.stopAudio();

audioStatus = false;

}

}

if (move == 2) {

speed = 4;

if (startSong) {

audio = new Audio(1);

audio.playAudio();

audioStatus = true;

startSong = false;

}

if (over && audioStatus) {

audio.stopAudio();

audioStatus = false;

}

}

if (move == 3) {

speed = 6;

if (startSong) {

audio = new Audio(2);

audio.playAudio();

audioStatus = true;

startSong = false;

}

if (over && audioStatus) {

audio.stopAudio();

audioStatus = false;

}

}

if (move == 4) {

speed = 8;

if (startSong) {

audio = new Audio(3);

audio.playAudio();

audioStatus = true;

startSong = false;

}

if (over && audioStatus) {

audio.stopAudio();

audioStatus = false;

}

}

if (move == 5) {

speed = 10;

if (startSong) {

audio = new Audio(4);

audio.playAudio();

audioStatus = true;

startSong = false;

}

if (over && audioStatus) {

audio.stopAudio();

audioStatus = false;

}

}

if (go) {

song1.gameInBackGround(graphics);

button.gameButton(graphics, key);

if (!over) {

Tiles tiles = new Tiles();

tiles.drawTiles(graphics, tilesCheck, tilesY, play);

GameText gameText1 = new GameText();

gameText1.score(graphics, score, compliment, complimentSize);

}

}

if (foul) {

Tiles tiles = new Tiles();

tiles.drawFoul(graphics, foulPlace, foulY);

}

if (over) {

GameText gameTextOver = new GameText();

gameTextOver.gameOver(graphics, score);

try {

Thread.sleep(500);

} catch (InterruptedException e) {

e.printStackTrace();

}

foul = false;

}

graphics.dispose();

repaint();

}

public void backGround(Graphics g) {

// Start Screen Background - Black with rising bubbles

g.setColor(new Color(0, 0, 0, 255)); // Black Background

g.fillRect(0, 0, 600, 750);

// Select Song Block Perimeter

g.setColor(Color.WHITE); // g.setColor(new Color(255, 255, 255, 163)); = White

g.draw3DRect(0, selectRectY, 580, 100, true); // Select Song Outline size/pos

}

/\*

// Start Screen - Background Rising Bubbles

private void backBalls(Graphics g) {

g.setColor(Color.RED); // g.setColor(new Color(255, 255, 255, 131)); = Gray

for(int i = 0; i < 40; i++) {

g.fillOval(backBallX[i], backBallY[i], 10, 10); // Bubble Size

}

}

\*/

// Start screen - Song Image

public void getSongImage(boolean check) {

if (check) {

try {

for (int i = 0; i <= 4; i++) {

songImage[i] = ImageIO.read(new File(images[i]));

}

//song1 = ImageIO.read(new File("Images/someoneYouLoved.jpeg"));

//song2 = ImageIO.read(new File("Images/memories.jpg"));

//song3 = ImageIO.read(new File("Images/playDate.jpeg"));

//song4 = ImageIO.read(new File("Images/danceMonkey.jpeg"));

//song5 = ImageIO.read(new File("Images/countingStars.jpeg"));

} catch (IOException e) {

e.printStackTrace();

}

}

}

// Start Screen - Song Select Image (image,x,y,width,height)

public void drawSongImage(Graphics g) {

int y = 210;

for (int i = 0; i <= 4; i++) {

g.drawImage(songImage[i], 20, y, 150, 80, this);

y += 100;

}

/\* g.drawImage(song1, 20, 210, 150, 80, this);

g.drawImage(song2, 20, 310, 150, 80, this);

g.drawImage(song3, 20, 410, 150, 80, this);

g.drawImage(song4, 20, 510, 150, 80, this);

g.drawImage(song5, 20, 610, 150, 80, this); \*/

}

// Song Select Page Icon Image

public void icon(Graphics g) {

if (check) try {

//iconImage = ImageIO.read(new File("Images/HandGrip2.jpg"));

iconImage = ImageIO.read(new File(images[5]));

} catch (IOException e) {

e.printStackTrace();

}

// Image (image,x,y,width,height)

g.drawImage(iconImage, 20, 10, 200, 190, this);

}

// create tiles from top of screen

public void tilesProduce() {

totalTiles++;

x = rand.nextInt(4);

tilesCheck[x] = true;

}

@Override

public void actionPerformed(ActionEvent e) {

for (int i = 0; i < 40; i++) { // when bubbles float off top of screen

backBallY[i] -= 1;

if (backBallY[i] == 0) {

backBallY[i] = 700; // create new bubble y bottom of screen

backBallX[i] = rand.nextInt(550); // random x position

}

}

for (int i = 0; i <= 3; i++) {

if (tilesCheck[i]) {

tilesY[i] += speed;

}

if (tilesY[i] >= 80 && !checkTilesProduce[i]) {

tilesProduce();

checkTilesProduce[i] = true;

}

if (tilesY[i] >= bottomBond) {

over = true;

play = false;

tilesCheck[i] = false;

}

}

/\*

if(tilesCheck[0]) {

tilesY[0]+=speed;

}

if(tilesCheck[1]) {

tilesY[1]+=speed;

}

if(tilesCheck[2]) {

tilesY[2]+=speed;

}

if(tilesCheck[3]) {

tilesY[3]+=speed;

}

if(tilesY[0]>=80 && !checkTilesProduce[0]) {

tilesProduce();

checkTilesProduce[0] = true;

}

if(tilesY[1]>=80 && !checkTilesProduce[1]) {

tilesProduce();

checkTilesProduce[1] = true;

}

if(tilesY[2]>=80 && !checkTilesProduce[2]) {

tilesProduce();

checkTilesProduce[2] = true;

}

if(tilesY[3]>=80 && !checkTilesProduce[3]) {

tilesProduce();

checkTilesProduce[3] = true;

}

if(tilesY[0]>=bottomBond) {

over = true;

play = false;

tilesCheck[0]=false;

}

if(tilesY[1]>=bottomBond) {

over = true;

play=false;

tilesCheck[1]=false;

}

if(tilesY[2]>=bottomBond) {

over = true;

play=false;

tilesCheck[2]=false;

}

if(tilesY[3]>=bottomBond) {

over = true;

play = false;

tilesCheck[3]=false;

}

\*/

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_DOWN) {

move += 1;

selectRectY += 100;

if (move == 6) move = 1;

if (selectRectY == 700) selectRectY = 200;

}

if (e.getKeyCode() == KeyEvent.VK\_UP) {

move -= 1;

selectRectY -= 100;

if (move == 0) move = 5;

if (selectRectY == 100) selectRectY = 600;

}

if (go && e.getKeyCode() == KeyEvent.VK\_1) {

play = true;

if (!audioStatus) startSong = true;

tilesProduce();

}

if (e.getKeyCode() == KeyEvent.VK\_ENTER) {

go = true;

}

if (over && e.getKeyCode() == KeyEvent.VK\_ENTER) {

initialValues();

}

if (e.getKeyCode() == KeyEvent.VK\_1 && !over) {

complimentSize = 80;

if (tilesY[0] < tilesY[1] - MISTAKE || tilesY[0] < tilesY[2] - MISTAKE || tilesY[0] < tilesY[3] - MISTAKE) {

over = true;

foul = true;

foulPlace = 0;

foulY = sortTilesY(tilesY);

}

key[0] = true;

}

if (e.getKeyCode() == KeyEvent.VK\_2 && !over) {

complimentSize = 80;

if (tilesY[1] < tilesY[0] - MISTAKE || tilesY[1] < tilesY[2] - MISTAKE || tilesY[1] < tilesY[3] - MISTAKE) {

over = true;

foul = true;

foulPlace = 1;

foulY = sortTilesY(tilesY);

}

key[1] = true;

}

if (e.getKeyCode() == KeyEvent.VK\_3 && !over) {

complimentSize = 80;

if (tilesY[2] < tilesY[0] - MISTAKE || tilesY[2] < tilesY[1] - MISTAKE || tilesY[2] < tilesY[3] - MISTAKE) {

over = true;

foul = true;

foulPlace = 2;

foulY = sortTilesY(tilesY);

}

key[2] = true;

}

if (e.getKeyCode() == KeyEvent.VK\_4 && !over) {

complimentSize = 80;

if (tilesY[3] < tilesY[0] - MISTAKE || tilesY[3] < tilesY[1] - MISTAKE || tilesY[3] < tilesY[2] - MISTAKE) {

over = true;

foul = true;

foulPlace = 3;

foulY = sortTilesY(tilesY);

}

key[3] = true;

}

}

@Override

public void keyReleased(KeyEvent e) {

if (e.getKeyCode() == KeyEvent.VK\_1) {

complimentSize = 50; // Compliment Player Size = 50px

key[0] = false;

}

if (e.getKeyCode() == KeyEvent.VK\_2) {

complimentSize = 50;

key[1] = false;

}

if (e.getKeyCode() == KeyEvent.VK\_3) {

complimentSize = 50;

key[2] = false;

}

if (e.getKeyCode() == KeyEvent.VK\_4) {

complimentSize = 50;

key[3] = false;

}

if (e.getKeyCode() == KeyEvent.VK\_1 && tilesCheck[0] && tilesY[0] > 200) {

ScoreCalculate scoreCalculate = new ScoreCalculate();

compliment = scoreCalculate.score(tilesY[0]);

if (compliment.equals("Perfect")) score += 3;

else if (compliment.equals("Great")) score += 2;

else score += 1;

key[0] = false;

tilesCheck[0] = false;

checkTilesProduce[0] = false;

tilesY[0] = -150;

tilesProduce();

}

if (e.getKeyCode() == KeyEvent.VK\_2 && tilesCheck[1] && tilesY[1] > 200) {

ScoreCalculate scoreCalculate = new ScoreCalculate();

compliment = scoreCalculate.score(tilesY[1]);

if (compliment.equals("Perfect")) score += 3;

else if (compliment.equals("Great")) score += 2;

else score += 1;

key[1] = false;

tilesCheck[1] = false;

checkTilesProduce[1] = false;

tilesY[1] = -150;

tilesProduce();

}

if (e.getKeyCode() == KeyEvent.VK\_3 && tilesCheck[2] && tilesY[2] > 200) {

ScoreCalculate scoreCalculate = new ScoreCalculate();

compliment = scoreCalculate.score(tilesY[2]);

if (compliment.equals("Perfect")) score += 3;

else if (compliment.equals("Great")) score += 2;

else score += 1;

key[2] = false;

tilesCheck[2] = false;

checkTilesProduce[2] = false;

tilesY[2] = -150;

tilesProduce();

}

if (e.getKeyCode() == KeyEvent.VK\_4 && tilesCheck[3] && tilesY[3] > 200) {

ScoreCalculate scoreCalculate = new ScoreCalculate();

compliment = scoreCalculate.score(tilesY[3]);

if (compliment.equals("Perfect")) score += 3;

else if (compliment.equals("Great")) score += 2;

else score += 1;

key[3] = false;

tilesCheck[3] = false;

checkTilesProduce[3] = false;

tilesY[3] = -150;

tilesProduce();

}

}

public int sortTilesY(int[] tilesY) {

int[] sort = new int[4];

for (int i = 0; i < 4; i++) {

sort[i] = tilesY[i];

}

Arrays.sort(sort);

return sort[3];

}

}

**/\***

**\* ScoreCalculate.java**

**\*/**

public class ScoreCalculate {

public String score(int place) {

if (place >= 330 && place <= 370) {

return "Perfect";

} else if (place >= 300 && place < 320 || place > 360 && place <= 380) {

return "Great";

} else {

return "Cool";

}

}

}

**/\***

**\* Song1.java**

**\*/**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

class Song1 extends JPanel implements KeyListener, ActionListener {

/\*

// Version 4

public void gameInBackGround(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

// Pure white background

g2.setColor(Color.WHITE);

g2.fillRect(0, 0, 600, 600); // Glowing lane separators

// Very subtle lane separators

g2.setColor(new Color(200, 200, 200));

g2.setStroke(new BasicStroke(2f));

g2.drawLine(150, 0, 150, 600);

g2.drawLine(300, 0, 300, 600);

g2.drawLine(450, 0, 450, 600); // Hit zone at bottom

// Hit zone (light neutral)

g2.setColor(new Color(240, 240, 240));

g2.fillRect(0, 450, 600, 200);

}

\*/

/\*

// version 3

public void gameInBackGround(Graphics g) {

// Smooth rendering

Graphics2D g2 = (Graphics2D) g;

// Gradient background

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

GradientPaint bg = new GradientPaint(0, 0, new Color(70,0,120), 0, 600, new Color(140,0,255));

g2.setPaint(bg);

g2.fillRect(0, 0, 600, 600); // Glowing lane separators

g2.setStroke(new BasicStroke(6f));

g2.setColor(new Color(255,255,255,90));

g2.drawLine(150,0,150,600);

g2.drawLine(300,0,300,600);

g2.drawLine(450,0,450,600); // Hit zone at bottom

g2.setPaint(new Color(0,0,0,160));

g2.fillRect(0, 450, 600, 200);

}

\*/

// version 2

public void gameInBackGround(Graphics g) {

// Smooth rendering

Graphics2D g2 = (Graphics2D) g;

// Gradient background

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

GradientPaint bg = new GradientPaint(0, 0, new Color(70,0,120), 0, 600, new Color(140,0,255));

g2.setPaint(bg);

g2.fillRect(0, 0, 600, 600); // Glowing lane separators

g2.setStroke(new BasicStroke(6f));

g2.setColor(new Color(255,255,255,90));

g2.drawLine(150,0,150,600);

g2.drawLine(300,0,300,600);

g2.drawLine(450,0,450,600); // Hit zone at bottom

g2.setPaint(new Color(0,0,0,160));

g2.fillRect(0, 450, 600, 200);

}

/\*

// version 1

public void gameInBackGround(Graphics g) {

g.setColor(new Color(0x0000FF)); // 0x842EDC = purple

g.fill3DRect(0, 0, 600, 600, true); // glowing lane separators

g.setColor(new Color(0xFFFFFF));

g.drawLine(150, 0, 150, 600);

g.drawLine(300, 0, 300, 600);

g.drawLine(450, 0, 450, 600); // bottom hit zone

g.setColor(Color.BLACK);

g.fill3DRect(0, 600, 600, 150, true);

g.setColor(new Color(0x34000000, true)); // 0x34000000 = RED

g.fill3DRect(0, 450, 600, 200, true);

}

\*/

@Override

public void actionPerformed(ActionEvent e) {

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyPressed(KeyEvent e) {

}

@Override

public void keyReleased(KeyEvent e) {

}

}

**/\***

**\* Tiles.java**

**\*/**

import javax.swing.\*;

import java.awt.\*;

import java.util.Random;

public class Tiles extends JPanel {

private int pos;

private int YHEIGHT = 200;

Random random = new Random();

// Version 4

public void drawTiles(Graphics g, boolean[] tilesCheck, int[] tilesY, boolean play) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

int w = 150, h = 200;

if (!play) {

g2.setColor(Color.BLUE.darker());

g2.fillRect(225, 300, w, h);

g2.setColor(Color.WHITE);

g2.setFont(new Font("SansSerif", Font.BOLD, 24));

g2.drawString("Press 1\n", 249, 361);

g2.drawString("to START", 243, 415);

return;

}

g2.setColor(Color.BLUE.darker());

for (int i = 0; i < 4; i++) {

if (tilesCheck[i]) {

g2.fillRect(i \* w, tilesY[i], w, h);

}

}

}

/\*

// Version 3

public void drawTiles(Graphics g, boolean[] tilesCheck, int[] tilesY, boolean play) {

Graphics2D g2 = (Graphics2D) g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING, RenderingHints.VALUE\_ANTIALIAS\_ON);

int tileWidth = 150;

int tileHeight = 200;

Color tileColor = new Color(10,10,10,220);

Color tileHighlight = new Color(255,255,255,45);

if (!play) {

g2.setColor(tileColor);

g2.fillRoundRect(225, 300, 150, 200, 30, 30);

g2.setColor(Color.white);

g2.setFont(new Font("SansSerif", Font.BOLD, 40));

g2.drawString("START", 245, 415);

return;

}

for(int i = 0; i < 4; i++) {

if(tilesCheck[i]) {

int x = i \* tileWidth;

int y = tilesY[i];

// tile rectangle g2.setColor(tileColor);

g2.fillRoundRect(x, y, tileWidth, tileHeight, 30, 30);

// highlight glow g2.setColor(tileHighlight);

g2.fillRoundRect(x+10, y+10, tileWidth-20, tileHeight-20, 30, 30);

}

}

}

\*/

/\*

// Version 2

public void drawTiles (Graphics g, boolean[] tilesCheck, int[] tilesY, boolean play) {

g.setColor(Color.BLACK);

if(!play) {

g.fillRect(150,350,150,200);

g.setColor(Color.white);

g.setFont(new Font(Font.DIALOG\_INPUT,Font.BOLD,30));

g.drawString("Start",180,460);

}

if(tilesCheck[0]) {

g.fillRect(0,tilesY[0],150,YHEIGHT); // YHEIGHT = 200

}

if(tilesCheck[1]) {

g.fillRect(150,tilesY[1],150,YHEIGHT);

}

if(tilesCheck[2]) {

g.fillRect(300,tilesY[2],150,YHEIGHT);

}

if(tilesCheck[3]) {

g.fillRect(450,tilesY[3],150,YHEIGHT);

}

}

\*/

public void drawFoul(Graphics g, int foulPlace, int foulY) {

g.setColor(Color.red);

if (foulPlace == 0) {

g.fillRect(0, foulY, 150, YHEIGHT);

}

if (foulPlace == 1) {

g.fillRect(150, foulY, 150, YHEIGHT);

}

if (foulPlace == 2) {

g.fillRect(300, foulY, 150, YHEIGHT);

}

if (foulPlace == 3) {

g.fillRect(450, foulY, 150, YHEIGHT);

}

}

}