**A**

**MINI PROJECT REPORT**

**ON**

**“Adventure Lost Game”**

**BY**

**Walunj Pranav Namdev**

**SAVITRIBAI PHULE PUNE UNIVERSITY**

**MASTERS OF COMPUTER APPLICATION**



**Akole Taluka Education Society’s**

**TECHNICAL CAMPUS AKOLE 2021-2022**

**Hardware and Software Requirement**

* **Hardware Required**

Processor: Intel i5 10th gen/Ryzen 5

Memory: Min 8GB

Disk space: Min 1GB

* **Software Required**

Operating System: Windows 7/8/10,

Tool: Java Virtual Machine

IDE: Eclipse

**Project Code:**

**Animation.java**

package aaaaa;

import java.awt.Image;

import java.awt.image.BufferedImage;

import java.util.ArrayList;

public class Animation {

ArrayList frames;

int currentFrame;

boolean loop;

private long animTime;

private long totalDuration;

public Animation(boolean l) {

loop = l;

frames = new ArrayList();

totalDuration = 0;

synchronized (this) {

animTime = 0;

currentFrame = 0;

}

}

public synchronized void addFrame(BufferedImage image, long duration) {

totalDuration += duration;

frames.add(new AnimFrame(image, totalDuration));

}

public synchronized void update(long elapsedTime) {

if (frames.size() > 1) {

animTime += elapsedTime;

if (!(loop && currentFrame == frames.size()-1)){

if (animTime >= totalDuration) {

animTime = animTime % totalDuration;

currentFrame = 0;

}

while (animTime > getFrame(currentFrame).endTime) {

currentFrame++;

}

}

}}

public synchronized BufferedImage getImage() {

if (frames.size() == 0) {

return null;

} else {

return getFrame(currentFrame).image;

}

}

private AnimFrame getFrame(int i) {

return (AnimFrame) frames.get(i);

}

private class AnimFrame {

BufferedImage image;

long endTime;

public AnimFrame(Image image, long endTime) {

this.image = (BufferedImage) image;

this.endTime = endTime;

}

}

}

**Background.java**

package aaaaa;

public class Background {

private int bgX, bgY, speedX;

public Background(int x, int y){

bgX = x;

bgY = y;

speedX = 0;

}

public void update() {

bgX += speedX;

if (bgX < -1920\*3){

bgX = 1920;

}

if (bgX > 1920\*2){

bgX = -1920\*2;

}

}

public int getBgX() {

return bgX;

}

public int getBgY() {

return bgY;

}

public int getSpeedX() {

return speedX;

}

public void setBgX(int bgX) {

this.bgX = bgX;

}

public void setBgY(int bgY) {

this.bgY = bgY;

}

public void setSpeedX(int speedX) {

this.speedX = speedX;

}

}

**Credits.java**

package aaaaa;

import java.awt.event.MouseEvent;

public class credits extends DynamicBackground{

public double orbitX = -50; /\* x-coordinate in orbit's center \*/

public double orbitY = -50; /\* y-coordinate in orbit's center \*/

public double orbitRadius = 20;

@Override

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

void mousePress(MouseEvent e){

int mx=e.getX();

int my= e.getY();

if (mx>60 && mx<210 && my>400 && my<440)

StartingClass.*State*="credits";

if(mx>20 && mx<170 && my>700 && my<740)

StartingClass.*State* = "menu";

}

}

**Death.java**

package aaaaa;

import java.awt.event.MouseEvent;

public class Death extends DynamicBackground{

public double orbitX = -110; /\* x-coordinate in orbit's center \*/

public double orbitY = -60; /\* y-coordinate in orbit's center \*/

@Override

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

void mousePress(MouseEvent e) {

int mx = e.getX();

int my = e.getY();

StartingClass.*restart*();

if (mx>60 && mx<210 && my>350 && my<390){

StartingClass.*State* = "menu";

}

else if(mx>60 && mx<210 && my>400 && my<440){

StartingClass.*State*="game";

}

}

}

**DynamicBackground.java**

package aaaaa;

class DynamicBackground {

public double orbitX; // x-coordinate in orbit's center \*/

public double orbitY; // y-coordinate in orbit's center \*/

public double orbitRadius = 20;

public double orbitSpeed = 0.03;

public double sphereRadius = 5;

public double t;

public double drawX;

public double drawY;

public double radian;

boolean mousePress;

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

}

**Enemy.java**

package aaaaa;

import java.awt.Rectangle;

import java.util.ArrayList;

public class Enemy {

private int maxHealth, currentHealth, power, speedX, centerX, centerY;

private Background bg = StartingClass.getBg1();

public Rectangle r = new Rectangle(0,0,0,0);

public int health = 5;

private boolean isdead;

public static ArrayList<Enemy> enemies = new ArrayList<Enemy>();

private int movementSpeed;

public String direction = "left";

public Enemy(int centerX, int centerY) {

setCenterX(centerX);

setCenterY(centerY);

}

// Behavioral Methods

public static void update() {

for (Enemy i: enemies) {

i.follow();

i.centerX += i.speedX;

i.speedX = i.bg.getSpeedX() \* 5 + i.movementSpeed;

i.r.setBounds(i.centerX - 30, i.centerY-10, 85, 60);

if (i.r.intersects(Robot.yellowRed))

i.checkCollision();

}

}

public void follow() {

if (centerX - StartingClass.getRobot().getCenterX() >650){

this.movementSpeed = 0;

}

else if (StartingClass.getRobot().getCenterX() - centerX > 650){

this.movementSpeed = 0;

}

else if (Math.abs(StartingClass.getRobot().getCenterX() - centerX) < 5) {

this.movementSpeed = 0;

}

else {

if (StartingClass.getRobot().getCenterX() >= centerX) {

this.direction = "right";

this.movementSpeed = 2;

}

else {

this.direction = "left";

this.movementSpeed = -2;

}

}

}

private void checkCollision() {

if (r.intersects(Robot.rect) || r.intersects(Robot.rect2)){

StartingClass.State = "dead";

}

}

public void attack() {

}

public int getMaxHealth() {

return maxHealth;

}

public int getCurrentHealth() {

return currentHealth;

}

public int getPower() {

return power;

}

public int getSpeedX() {

return speedX;

}

public int getCenterX() {

return centerX;

}

public int getCenterY() {

return centerY;

}

public Background getBg() {

return bg;

}

public void setMaxHealth(int maxHealth) {

this.maxHealth = maxHealth;

}

public void setCurrentHealth(int currentHealth) {

this.currentHealth = currentHealth;

}

public void setPower(int power) {

this.power = power;

}

public void setSpeedX(int speedX) {

this.speedX = speedX;

}

public void setCenterX(int centerX) {

this.centerX = centerX;

}

public void setCenterY(int centerY) {

this.centerY = centerY;

}

public void setBg(Background bg) {

this.bg = bg;

}

public boolean getIsDead(){

return isdead;

}

public void setIsDead(boolean s){

isdead = s;

}}

**Guide.java**

package aaaaa;

import java.awt.event.MouseEvent;

public class Guide extends DynamicBackground{

public double orbitX = -20; /\* x-coordinate in orbit's center \*/

public double orbitY = -20; /\* y-coordinate in orbit's center \*/

@Override

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

void mousePress(MouseEvent e) {

int mx = e.getX();

int my = e.getY();

if (mx>60 && mx<210 && my>350 && my<390){

StartingClass.*State* = "guide";

}

else if(mx>20 && mx<170 && my>700 && my<740){

StartingClass.*State*="menu";

}

}

}

**Intro.java**

package aaaaa;

import java.awt.event.MouseEvent;

public class Intro extends DynamicBackground{

public double orbitX = -50; /\* x-coordinate in orbit's center \*/

public double orbitY = -50; /\* y-coordinate in orbit's center \*/

@Override

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

void mousePress(MouseEvent e) {

int mx = e.getX();

int my = e.getY();

if (mx>60 && mx<210 && my>300 && my<340){

StartingClass.*State* = "intro";

}

else if(mx>20 && mx<170 && my>700 && my<740){

StartingClass.*State*="menu";

}

}

}

**Menu.java**

package aaaaa;

import java.awt.event.MouseEvent;

public class Menu extends DynamicBackground{

public double orbitX = -700; /\* x-coordinate in orbit's center \*/

public double orbitY = -600; /\* y-coordinate in orbit's center \*/

@Override

public void update(){

radian = orbitSpeed \* t;

drawX = orbitX + orbitRadius \* Math.*cos*(radian);

drawY = orbitY + orbitRadius \* Math.*sin*(radian);

t+=1;

}

void mousePress(MouseEvent e) {

int mx = e.getX();

int my = e.getY();

if (mx>60 && mx<210 && my>250 && my<290) //Play button

StartingClass.*State* = "game";

else if (mx>60 && mx<210 && my>300 && my<340) // intro button

StartingClass.*State* = "intro";

else if (mx>60 && mx<210 && my>350 && my<390) //guide button

StartingClass.*State* = "guide";

else if (mx>60 && mx<210 && my>400 && my<440) // credits button

StartingClass.*State* = "credits";

else if(mx>60 && mx<210 && my>450 && my<490)

System.*exit*(1);

}

}

**Projectile.java**

package aaaaa;

import java.awt.Rectangle;

import java.util.ArrayList;

public class Projectile {

private int x, y, speedX;

private boolean visible;

private Rectangle r;

public Projectile(int startX, int startY, boolean right){

x = startX;

y = startY;

if (right)

speedX =30;

else

speedX = -30;

visible = true;

r = new Rectangle(0, 0, 0, 0);

}

public void update(){

x += speedX;

r.setBounds(x, y, 10, 5);

if (x > 1366 || x<0){

visible = false;

r = null;

}

else

checkCollision();

}

private void checkCollision() {

ArrayList tiles = StartingClass.getTileArray();

for (int i = 0; i < tiles.size(); i++) {

Tile p = (Tile) tiles.get(i);

if (r != null){

if (r.intersects(p.getRectangle()) && (p.getType()==2||p.getType()==3||p.getType()==6)){

if (Robot.getDirection() == "right")

x = p.getTileX()-35;

else if (Robot.getDirection() == "left")

x = p.getTileX()+50;

visible = false;

}

}

}

for (Enemy i: Enemy.enemies){

if(r.intersects(i.r)){

visible = false;

if (i.health > 0) {

i.health -= 1;

}

if (i.health == 0) {

i.setIsDead(true);

}

}

}

}

public int getX() {

return x;

}

public int getY() {

return y;

}

public int getSpeedX() {

return speedX;

}

public boolean isVisible() {

return visible;

}

public void setX(int x) {

this.x = x;

}

public void setY(int y) {

this.y = y;

}

public void setSpeedX(int speedX) {

this.speedX = speedX;

}

public void setVisible(boolean visible) {

this.visible = visible;

}

}

**Robot.java**

package aaaaa;

import java.awt.Rectangle;

import java.util.ArrayList;

public class Robot {

// Constants are Here

final int JUMPSPEED = -17;

final int MOVESPEED = 5;

private int centerX = 500;

private int centerY = 560;

private boolean jumped = false;

private boolean movingLeft = false;

private boolean movingRight = false;

private boolean ducked = false;

private static String direction = "right";

private int speedX = 0;

private int speedY = 0;

public static Rectangle rect = new Rectangle(0, 0, 0, 0);

public static Rectangle rect2 = new Rectangle(0, 0, 0, 0);

public static Rectangle yellowRed = new Rectangle(0, 0, 0, 0);

public static Rectangle footleft = new Rectangle(0,0,0,0);

public static Rectangle footright = new Rectangle(0,0,0,0);

private Background bg1 = StartingClass.getBg1();

private Background bg2 = StartingClass.getBg2();

private Background bg3 = StartingClass.getBg3();

private Background bg4 = StartingClass.getBg4();

private ArrayList<Projectile> projectiles = new ArrayList<Projectile>();

public void update() {

if (speedX == 0){

bg1.setSpeedX(0);

bg2.setSpeedX(0);

bg3.setSpeedX(0);

bg4.setSpeedX(0);

}

if (centerX <= 800 && speedX > 0) {

centerX += speedX;

}

if (centerX >=400 && speedX<0){

centerX += speedX;

}

if (speedX > 0 && centerX > 800) {

bg1.setSpeedX(-MOVESPEED / 5);

bg2.setSpeedX(-MOVESPEED / 5);

bg3.setSpeedX(-MOVESPEED / 5);

bg4.setSpeedX(-MOVESPEED / 5);

}

if (speedX < 0 && centerX < 400){

bg1.setSpeedX(MOVESPEED / 5);

bg2.setSpeedX(MOVESPEED / 5);

bg3.setSpeedX(MOVESPEED / 5);

bg4.setSpeedX(MOVESPEED / 5);

}

// Updates Y Position

centerY += speedY;

// Handles Jumping

speedY += 1;

if (speedY > 3){

jumped = true;

}

rect.setRect(centerX -55, centerY - 63 , 35, 70);

rect2.setRect(rect.getX(), rect.getY() + 70, 35, 70);

yellowRed.setRect(centerX - 92, centerY - 80, 110, 180);

footleft.setRect(centerX -60, centerY - 33, 5, 95);

footright.setRect(centerX -20, centerY - 33, 5, 95);

}

public void moveRight() {

direction = "right";

if (ducked == false) {

speedX = MOVESPEED;

}

}

public void moveLeft() {

direction = "left";

if (ducked == false) {

speedX = -MOVESPEED;

}

}

public void stopRight() {

setMovingRight(false);

stop();

}

public void stopLeft() {

setMovingLeft(false);

stop();

}

private void stop() {

if (isMovingRight() == false && isMovingLeft() == false) {

speedX = 0;

}

if (isMovingRight() == false && isMovingLeft() == true) {

moveLeft();

}

if (isMovingRight() == true && isMovingLeft() == false) {

moveRight();

}

}

public void jump() {

if (jumped == false) {

speedY = JUMPSPEED;

jumped = true;

}

}

public void shoot() {

Projectile p;

if (getDirection() == "right")

p = new Projectile(centerX-5, centerY-5,true);

else

p = new Projectile(centerX-5, centerY-5,false);

projectiles.add(p);

}

public int getCenterX() {

return centerX;

}

public int getCenterY() {

return centerY;

}

public boolean isJumped() {

return jumped;

}

public int getSpeedX() {

return speedX;

}

public int getSpeedY() {

return speedY;

}

public void setCenterX(int centerX) {

this.centerX = centerX;

}

public void setCenterY(int centerY) {

this.centerY = centerY;

}

public void setJumped(boolean jumped) {

this.jumped = jumped;

}

public void setSpeedX(int speedX) {

this.speedX = speedX;

}

public void setSpeedY(int speedY) {

this.speedY = speedY;

}

public boolean isDucked() {

return ducked;

}

public void setDucked(boolean ducked) {

this.ducked = ducked;

}

public boolean isMovingRight() {

return movingRight;

}

public void setMovingRight(boolean movingRight) {

this.movingRight = movingRight;

}

public boolean isMovingLeft() {

return movingLeft;

}

public void setMovingLeft(boolean movingLeft) {

this.movingLeft = movingLeft;

}

public ArrayList getProjectiles() {

return projectiles;

}

public static String getDirection(){

return direction;

}

}

**StartingClass.java**

package aaaaaa;

import java.awt.Color;

import java.awt.Graphics;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.image.BufferedImage;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import javax.imageio.ImageIO;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class StartingClass extends JPanel implements Runnable, KeyListener, MouseListener {

private boolean ctrl\_press;

ArrayList projectiles;

static StartingClass starter = new StartingClass();

private static Robot robot;

private BufferedImage currentSprite, c0,c1, c2, c3,c4,c5,c6,c7,c8,c9, c10,c11,

s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,crouch\_r0,crouch\_r1,crouch\_r2,crouch\_r3,

crouch\_l0,crouch\_l1,crouch\_l2,crouch\_l3,menu,bulletr,bulletl,

characterJumped\_r, characterJumped\_l, background1,background2,

el,e2l,er,e2r,play,exit,creditsButton,credits,backButton,credits1,introback,

introfront,guideback,guidefront,guideButton,introButton,logo,teamlogo,start,

deathimg,menubutton,playagain;

private static Background bg1, bg2,bg3,bg4;

public Animation anim\_r,anim\_l,anim,anim\_still,anim\_still\_r,anim\_still\_l,

crouchdown\_r,crouchdown\_r1,crouchdown\_l;

public static Animation hanim\_l,hanim\_r;

public static BufferedImage tiledirt,grasstop, tilestone,tiletree,tilerock;

private static ArrayList<Tile> tilearray = new ArrayList<Tile>();

static String State = "start";

Menu MENU;

credits CREDITS;

Intro INTRO;

Guide GUIDE;

Death DEATH;

public static void restart(){

robot.getProjectiles().clear();

tilearray.clear();

Enemy.enemies.clear();

bg1 = new Background(0, 0);

bg2 = new Background(1920, 0);

bg3 = new Background(-1920,0);

bg4 = new Background(-1920\*2,0);

robot = new Robot();

try {

starter.loadMap("data/map1.txt");

}

catch (IOException e) {

e.printStackTrace();

}

}

public void init() throws Exception{

setFocusable(true);

addKeyListener(this);

addMouseListener(this);

// Image Setups

characterJumped\_r = ImageIO.read(new File("data/jumpedr.png"));

characterJumped\_l = ImageIO.read(new File("data/jumpedl.png"));

background1 = ImageIO.read(new File("data/bg1.png"));

background2 = ImageIO.read(new File("data/bg2.png"));

tiledirt = ImageIO.read(new File("data/tiledirt.png"));

grasstop = ImageIO.read(new File("data/grasstop.png"));

tilestone = ImageIO.read(new File("data/tilestone.png"));

tiletree = ImageIO.read(new File("data/tiletree.png"));

tilerock = ImageIO.read(new File("data/tilerock.png"));;

bulletr = ImageIO.read(new File("data/bulletr.png"));

bulletl = ImageIO.read(new File("data/bulletl.png"));

c0= ImageIO.read(new File("data/walk/walk-r (1).png"));

c1 = ImageIO.read(new File("data/walk/walk-r (2).png"));

c2 = ImageIO.read(new File("data/walk/walk-r (3).png"));

c3 = ImageIO.read(new File("data/walk/walk-r (4).png"));

c4 = ImageIO.read(new File("data/walk/walk-r (5).png"));

c5 = ImageIO.read(new File("data/walk/walk-r (6).png"));

c6 = ImageIO.read(new File("data/walk/walk-r (7).png"));

c7 = ImageIO.read(new File("data/walk/walk-r (8).png"));

c8 = ImageIO.read(new File("data/walk/walk-r (9).png"));

c9 = ImageIO.read(new File("data/walk/walk-r (10).png"));

c10 = ImageIO.read(new File("data/walk/walk-r (11).png"));

c11 = ImageIO.read(new File("data/walk/walk-r (12).png"));

s0 = ImageIO.read(new File("data/walk/0.png"));

s1 = ImageIO.read(new File("data/walk/1.png"));

s2 = ImageIO.read(new File("data/walk/2.png"));

s3 = ImageIO.read(new File("data/walk/3.png"));

s4 = ImageIO.read(new File("data/walk/4.png"));

s5 = ImageIO.read(new File("data/walk/5.png"));

s6 = ImageIO.read(new File("data/walk/6.png"));

s7 = ImageIO.read(new File("data/walk/7.png"));

s8 = ImageIO.read(new File("data/walk/8.png"));

s9 = ImageIO.read(new File("data/walk/9.png"));

s10 = ImageIO.read(new File("data/walk/10.png"));

s11 = ImageIO.read(new File("data/walk/11.png"));

crouch\_r0 = ImageIO.read(new File("data/crouch/r0.png"));

crouch\_r1 = ImageIO.read(new File("data/crouch/r1.png"));

crouch\_r2 = ImageIO.read(new File("data/crouch/r2.png"));

crouch\_r3 = ImageIO.read(new File("data/crouch/r3.png"));

crouch\_l0 = ImageIO.read(new File("data/crouch/1.png"));

crouch\_l1 = ImageIO.read(new File("data/crouch/2.png"));

crouch\_l2 = ImageIO.read(new File("data/crouch/3.png"));

crouch\_l3 = ImageIO.read(new File("data/crouch/4.png"));

// Menu Images

menu = ImageIO.read(new File("data/menu.png"));

logo= ImageIO.read(new File("data/logo.png"));

teamlogo = ImageIO.read(new File("data/teamlogo.png"));

play= ImageIO.read(new File("data/play.png"));

exit= ImageIO.read(new File("data/exit.png"));

creditsButton= ImageIO.read(new File("data/creditsButton.png"));

guideButton= ImageIO.read(new File("data/guideButton.png"));

introButton=ImageIO.read(new File("data/introButton.png"));

start = ImageIO.read(new File("data/intro1.png"));

deathimg = ImageIO.read(new File("data/deathimg.png"));

menubutton = ImageIO.read(new File("data/menuButton.png"));

playagain = ImageIO.read(new File("data/playagainButton.png"));

//Credits Stuff

credits= ImageIO.read(new File("data/credits.png"));

credits1= ImageIO.read(new File("data/credits1.png"));

backButton= ImageIO.read(new File("data/backButton.png"));

//Intro stuff

introback=ImageIO.read(new File("data/introback.png"));

introfront=ImageIO.read(new File("data/introfront.png"));

//Guide stuff

guideback=ImageIO.read(new File("data/guideback.png"));

guidefront=ImageIO.read(new File("data/guidefront.png"));

el = ImageIO.read(new File("data/heliboy.png"));

e2l =ImageIO.read(new File("data/heliboy2.png"));

er = ImageIO.read(new File("data/heliboyl.png"));

e2r =ImageIO.read(new File("data/heliboy2l.png"));

hanim\_l = new Animation(false);

hanim\_l.addFrame(el, 500);

hanim\_l.addFrame(e2l, 500);

hanim\_r = new Animation(false);

hanim\_r.addFrame(er, 500);

hanim\_r.addFrame(e2r, 500);

crouchdown\_r = new Animation(true);

crouchdown\_r.addFrame(crouch\_r0, 20);

crouchdown\_r.addFrame(crouch\_r1, 20);

crouchdown\_r.addFrame(crouch\_r2, 20);

crouchdown\_r.addFrame(crouch\_r3, 20);

crouchdown\_l = new Animation(true);

crouchdown\_l.addFrame(crouch\_l0, 20);

crouchdown\_l.addFrame(crouch\_l1, 20);

crouchdown\_l.addFrame(crouch\_l2, 20);

crouchdown\_l.addFrame(crouch\_l3, 20);

anim\_r = new Animation(false);

anim\_r.addFrame(c0, 50);

anim\_r.addFrame(c1, 50);

anim\_r.addFrame(c2, 50);

anim\_r.addFrame(c3, 50);

anim\_r.addFrame(c4, 50);

anim\_r.addFrame(c5, 50);

anim\_r.addFrame(c6, 50);

anim\_r.addFrame(c7, 50);

anim\_r.addFrame(c8, 50);

anim\_r.addFrame(c9, 50);

anim\_r.addFrame(c10, 50);

anim\_r.addFrame(c11, 50);

anim\_l = new Animation(false);

anim\_l.addFrame(s0, 50);

anim\_l.addFrame(s1, 50);

anim\_l.addFrame(s2, 50);

anim\_l.addFrame(s3, 50);

anim\_l.addFrame(s4, 50);

anim\_l.addFrame(s5, 50);

anim\_l.addFrame(s6, 50);

anim\_l.addFrame(s7, 50);

anim\_l.addFrame(s8, 50);

anim\_l.addFrame(s9, 50);

anim\_l.addFrame(s10, 50);

anim\_l.addFrame(s11, 50);

anim = anim\_r;

currentSprite = c0;

}

public void start() {

bg1 = new Background(0, 0);

bg2 = new Background(1920, 0);

bg3 = new Background(-1920,0);

bg4 = new Background(-1920\*2,0);

robot = new Robot();

MENU = new Menu();

CREDITS= new credits();

INTRO= new Intro();

GUIDE= new Guide();

DEATH = new Death();

// Initialize Tiles

try {

loadMap("data/map1.txt");

}

catch (IOException e) {

e.printStackTrace();

}

Thread thread = new Thread(this);

thread.start();

}

@Override

public void run() {

while (true) {

switch (State){

case "start":

break;

case "menu":

MENU.update();

break;

case "dead":

DEATH.update();

break;

case "credits":

CREDITS.update();

break;

case "intro":

INTRO.update();

break;

case "guide":

GUIDE.update();

break;

case "game":

gameUpdate();

break;

}

repaint();

try {

Thread.sleep(17);

}

catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public void gameUpdate(){

robot.update();

bg1.update();

bg2.update();

bg3.update();

bg4.update();

Enemy.update();

updateTiles();

hanim\_l.update(50);

hanim\_r.update(50);

ArrayList projectiles = robot.getProjectiles();

for (int i = 0; i < projectiles.size(); i++) {

Projectile p = (Projectile) projectiles.get(i);

if (p.isVisible() == true) {

p.update();

} else {

projectiles.remove(i);

}

}

for (int i = 0; i < Enemy.enemies.size(); i++){

if (Enemy.enemies.get(i).getIsDead())

Enemy.enemies.remove(i);

}

if (robot.isJumped() && robot.getDirection() == "right") {

currentSprite = characterJumped\_r;

}

else if (robot.isJumped() && robot.getDirection() == "left") {

currentSprite = characterJumped\_l;

}

else if ((robot.getDirection() == "right") && (robot.getSpeedX() == 0)&& robot.isDucked()==false){

currentSprite = c0;

}

else if ((robot.getDirection() == "left") && (robot.getSpeedX() == 0) && robot.isDucked()==false){

currentSprite = s0;

}

else {

if (robot.getSpeedX()<0){

anim = anim\_l;

}

else if (robot.getSpeedX()>0){

anim = anim\_r;

}

else if (robot.isDucked() && robot.getDirection() == "right"){

anim = crouchdown\_r;

}

else if (robot.isDucked() && robot.getDirection() == "left"){

anim = crouchdown\_l;

}

currentSprite = anim.getImage();

anim.update(10);

}

if (robot.getCenterY()>1400){

State = "dead";

}

}

@Override

public void paint(Graphics g) {

switch (State) {

case "start":

g.drawImage(start, 0, 0, this);

break;

case "menu":

try{

g.drawImage(menu, (int) MENU.drawX,(int)MENU.drawY,this);

g.drawImage(logo, 510, 32, this);

g.drawImage(play,60,250,this);

g.drawImage(introButton, 60, 300, this);

g.drawImage(guideButton,60,350, this);

g.drawImage(creditsButton,60,400,this);

g.drawImage(exit,60,450,this);

g.drawImage(teamlogo,60,680,this);

}

catch(Exception e){ }

break;

case "dead":

g.drawImage(deathimg, (int) DEATH.drawX,(int)DEATH.drawY,this);

g.drawImage(menubutton,60,350, this);

g.drawImage(playagain,60,400, this);

break;

case "credits":

g.drawImage(credits, (int)CREDITS.drawX, (int)CREDITS.drawY, this);

g.drawImage(credits1,183,84, this);

g.drawImage(backButton,20,700, this);

break;

case "intro":

g.drawImage(introback,(int)INTRO.drawX, (int)INTRO.drawY, this);

g.drawImage(introfront,52, 70, this);

g.drawImage(backButton,20,700, this);

break;

case "guide":

g.drawImage(guideback,(int)GUIDE.drawX,(int)GUIDE.drawY, this);

g.drawImage(guidefront,478,170, this);

g.drawImage(backButton, 20,700, this);

break;

case "game":

g.drawImage(background1, bg1.getBgX(), bg1.getBgY(), this);

g.drawImage(background2, bg2.getBgX(), bg2.getBgY(), this);

g.drawImage(background2, bg3.getBgX(), bg3.getBgY(), this);

g.drawImage(background1, bg4.getBgX(), bg4.getBgY(), this);

paintTiles(g);

paintProjectiles(g);

paintEnemies(g);

g.drawImage(currentSprite, robot.getCenterX() - 61,robot.getCenterY() - 63, this);

break;

}

}

private void updateTiles() {

for (int i = 0; i < tilearray.size(); i++) {

Tile t = (Tile) tilearray.get(i);

t.update();

}

}

private void paintEnemies(Graphics g){

for (Enemy i: Enemy.enemies){

if (i.getCenterX()>-50 && i.getCenterX()<1366){

if (i.direction == "right")

g.drawImage(hanim\_r.getImage(), i.getCenterX() - 48, i.getCenterY() - 48, this);

else if (i.direction == "left")

g.drawImage(hanim\_l.getImage(), i.getCenterX() - 48, i.getCenterY() - 48, this);

}

}

}

private void paintProjectiles(Graphics g){

projectiles = robot.getProjectiles();

for (int i = 0; i < projectiles.size(); i++) {

Projectile p = (Projectile) projectiles.get(i);

if (robot.getDirection() == "right")

g.drawImage(bulletr, p.getX(), p.getY(), this);

else if (robot.getDirection() == "left")

g.drawImage(bulletl, p.getX(), p.getY(), this);

}

}

private void paintTiles(Graphics g) {

for (int i = 0; i < tilearray.size(); i++) {

Tile t = (Tile) tilearray.get(i);

g.drawImage(t.getTileImage(), t.getTileX(), t.getTileY(), this);

}

}

private void loadMap(String filename) throws IOException {

ArrayList lines = new ArrayList();

int width = 0;

int height = 0;

BufferedReader reader = new BufferedReader(new FileReader(filename));

while (true) {

String line = reader.readLine();

// no more lines to read

if (line == null) {

reader.close();

break;

}

if (!line.startsWith("!")) {

lines.add(line);

width = Math.max(width, line.length());

}

}

height = lines.size();

for (int j = 0; j < 20; j++) {

String line = (String) lines.get(j);

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

if (i < line.length()) {

char ch = line.charAt(i);

if (ch == '\*'){

Enemy.enemies.add(new Enemy(i\*40, j\*40-80));

}

else{

Tile t = new Tile(i, j, Character.getNumericValue(ch));

tilearray.add(t);

}

}

}

}

}

@Override

public void keyPressed(KeyEvent e) {

if (State == "start"){

if (e.getKeyCode() == KeyEvent.VK\_SPACE)

State = "menu";

}

if (State=="credits"||State == "intro"||State=="guide"){

switch(e.getKeyCode()){

case KeyEvent.VK\_ESCAPE:

State="menu";

break;

}

}

if (State=="game"){

switch (e.getKeyCode()) {

case KeyEvent.VK\_ESCAPE:

State="menu";

break;

case KeyEvent.VK\_UP:

System.out.println("Move up");

break;

case KeyEvent.VK\_DOWN:

robot.setDucked( true);

break;

case KeyEvent.VK\_LEFT:

robot.moveLeft();

break;

case KeyEvent.VK\_RIGHT:

robot.moveRight();

break;

case KeyEvent.VK\_SPACE:

robot.jump();

break;

case KeyEvent.VK\_ENTER:

State="game";

break;

case KeyEvent.VK\_CONTROL:

ctrl\_press = true;

}

}

}

@Override

public void keyReleased(KeyEvent e) {

switch (e.getKeyCode()) {

case KeyEvent.VK\_UP:

break;

case KeyEvent.VK\_DOWN:

if (State == "game"){

crouchdown\_r.currentFrame = 0;

crouchdown\_l.currentFrame = 0;

robot.setDucked(false);

}

break;

case KeyEvent.VK\_LEFT:

if (State == "game")

robot.stopLeft();

break;

case KeyEvent.VK\_RIGHT:

if (State == "game")

robot.stopRight();

break;

case KeyEvent.VK\_SPACE:

if (State == "game"){

robot.setMovingLeft(false);

robot.setMovingRight(false);

}

break;

case KeyEvent.VK\_CONTROL:

if (State == "game"){

if (ctrl\_press){

robot.shoot();

}

}

break;

}

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void mouseClicked(MouseEvent me) {

}

@Override

public void mousePressed(MouseEvent me) {

switch (State){

case "credits":

CREDITS.mousePress(me);

break;

case "dead":

DEATH.mousePress(me);

break;

case "menu":

MENU.mousePress(me);

break;

case "intro":

INTRO.mousePress(me);

break;

case "guide":

GUIDE.mousePress(me);

}

}

@Override

public void mouseReleased(MouseEvent me) {

if(State=="menu")

MENU.mousePress = false;

if (State=="credits")

CREDITS.mousePress=false;

if(State=="intro")

MENU.mousePress = false;

if (State=="guide")

CREDITS.mousePress=false;

}

@Override

public void mouseEntered(MouseEvent me) {

}

@Override

public void mouseExited(MouseEvent me) {

}

// Getters

public static Background getBg1() {

return bg1;

}

public static Background getBg2() {

return bg2;

}

public static Background getBg3(){

return bg3;

}

public static Background getBg4(){

return bg4;

}

public static Robot getRobot(){

return robot;

}

public static ArrayList getTileArray(){

return tilearray;

}

// Main Function

public static void main(String[] args)

JFrame frame = new JFrame("LOST");

frame.setSize(1366, 768);

frame.setBackground(Color.black);

frame.add(starter);

frame.setUndecorated(true);

frame.setVisible(true);

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

try{

starter.init();

starter.start();

}

catch (Exception e){

e.printStackTrace();

}}}

**Tile.java**

package aaaaa;

import java.awt.Image;

import java.awt.Rectangle;

public class Tile {

private int tileX, tileY, speedX, type;

public Image tileImage;

private Robot robot = StartingClass.getRobot();

private Background bg = StartingClass.getBg1();

private Rectangle r;

public Tile(int x, int y, int typeint){

tileX = x\*40;

tileY = y\*40;

type = typeint;

r = new Rectangle();

if (type == 2)

tileImage = StartingClass.grasstop;

else if (type == 3)

tileImage = StartingClass.tiledirt;

else if (type == 4)

tileImage = StartingClass.tilestone;

else if (type == 5)

tileImage = StartingClass.tiletree;

else if (type == 6)

tileImage = StartingClass.tilerock;

else {

type = 0;

}

}

public void update(){

speedX = bg.getSpeedX() \* 5;

tileX += speedX;

r.setBounds(tileX, tileY, 40, 40);

if (r.intersects(Robot.yellowRed) && type != 0) {

checkVerticalCollision(Robot.rect, Robot.rect2);

checkSideCollision(Robot.footleft, Robot.footright);

}

for (Enemy i: Enemy.enemies){

if(r.intersects(i.r) && ( type == 2 || type == 3 || type == 6 )){

i.setSpeedX(0);

if (i.getCenterX()<tileX)

i.setCenterX(tileX-55);

else if (i.getCenterX()>tileX)

i.setCenterX(tileX+56);

}

}

}

public int getTileX() {

return tileX;

}

public void setTileX(int tileX) {

this.tileX = tileX;

}

public int getTileY() {

return tileY;

}

public void setTileY(int tileY) {

this.tileY = tileY;

}

public Rectangle getRectangle(){

return r;

}

public int getType(){

return type;

}public Image getTileImage() {

return tileImage;

}

public void setTileImage(Image tileImage) {

this.tileImage = tileImage;

}

public void checkVerticalCollision(Rectangle rtop, Rectangle rbot) {

if ( type == 2 || type == 3 || type == 6 ){

if (rtop.intersects(r)) {

robot.setCenterY(tileY+100);

robot.setSpeedY(-robot.JUMPSPEED);

}

if (rbot.intersects(r)) {

robot.setJumped(false);

robot.setSpeedY(0);

robot.setCenterY(tileY - 75);

}

}

}

public void checkSideCollision(Rectangle leftfoot, Rectangle rightfoot) {

if (type == 2 || type == 3 || type == 6 ){

if (leftfoot.intersects(r)) {

robot.setCenterX(tileX + 100);

robot.setSpeedX(0);

}

else if (rightfoot.intersects(r)) {

robot.setCenterX(tileX + 15);

robot.setSpeedX(0);

}}}}

Project Model View

**Home Page:**

****

**Starting Page:**



**End Game:**



**Intro Page:**



**Guide Page:**

