[日期]

双人飞机大战

概要设计说明书

**目录**

[1.　引言 2](#_Toc2970699)

[1.1　编写目的 2](#_Toc2970700)

[1.2 背景 2](#_Toc2970701)

[1.3 定义 2](#_Toc2970702)

[1.4 参考资料 5](#_Toc2970703)

[2　总体设计 5](#_Toc2970704)

[2.1　需求规定 5](#_Toc2970705)

[2.1.1输入、输出 5](#_Toc2970706)

[2.1.2 功能要求 5](#_Toc2970707)

[2.1.3 性能要求 5](#_Toc2970708)

[2.2　运行环境 6](#_Toc2970709)

[2.3 基本设计概念和处理流程 6](#_Toc2970710)

[2.3.1基本设计概念如下： 6](#_Toc2970711)

[2.3.2处理流程如下： 7](#_Toc2970712)

[2.4　结构 8](#_Toc2970713)

[2.5　功能需求与程序的关系 13](#_Toc2970714)

[3　运行设计 14](#_Toc2970715)

[3.1　运行模块组合 14](#_Toc2970716)

[3.2　运行控制 14](#_Toc2970717)

**概要设计说明书**

# 1.　引言

# 1.1　编写目的

在通过为期两周的Java游戏开发实训之后，我们组将以开发一款独立自主的小游戏来结束本次实训课程，同时我们也会选用比较合理的游戏设计模式来开发我们的游戏，即便是小游戏但是我们也将将游戏引擎独立编写，以此来更深入的理解游戏引擎的概念以及游戏的架构。

## 1.2 背景

系统名称：双人飞机大战

开发者：第七组全体成员：徐士杰201620181529（组长）、胡静雅201620181609、方媛媛201620181605、李丹201620181612、高鸿秋201620181606、郭琳201620181608

用户：NOKIA手机用户

运行该软件机器：Win10 PC

## 定义

1. ArrarList<Bullet>：

每一个玩家类都继承自Plane飞机类，在飞机类中有一个子弹成员是由*ArrayList*构成的，每当用户按下子弹发射键的时候，将会根据玩家所操控的飞机的位置以及方向添加一颗子弹对象到bullets的*ArrayList<Bullet>*中。

1. GameUtil：

该实用工具类是为了处理图片资源而创建的。包含几个静态方法： *public static BufferedImage getImage(String path)* 用于获取指定路径的图片资源。

*static BufferedImage rotateImage(final BufferedImage bufferedimage, final int degree)* 用于给定图片旋转指定角度返回图片。

1. GameObject

该类为所有游戏的抽象类。主要定义了两个带参构造函数以及两个抽象方法：

带参构造函数：

*GameObject(String path)*用于创建默认坐标和角度的图片资源。

*GameObject(int pos\_x, int pos\_y, String path, int angel)*用于创建给定坐标和角度的图片资源。

抽象方法：

*public abstract void draw(Graphics g);*绘制图片资源抽象方法。

*public abstract void update();*更新图像参数抽象方法。

1. InputMsg

该类为处理按键事件功能类。将每个键按下的状态进行保存如*Map<Integer, Boolean> keyStatus*的成员之中，每次刷新时候通过*public boolean getKey(Integer keycode)*方法进行检测按键是否按下。

1. GamePanel

该类继承自*JPanel*类，包含一个存有游戏所有对象要用的*private List<GameObject> allObj*集合。并且通过编写*void updateLogic()* 方法将所有游戏对象进行属性更新。通过重写*JPanel* 的*public void paint(Graphics g)*方法将所有游戏对象进行重绘。

1. Conf

该类是一个属性类，包含了游戏中的一些常量参数，包括窗体的宽高，飞机的直径，子弹的直径，以及帧率等。

1. Director

该类统筹了整个游戏窗体的创建和运行机制。除构造函数和初始化函数之外还有*void runGamePanel(GamePanel gamePanel)*的功能方法来保证游戏运行。该方法不仅初始化了游戏窗体、画布和事件监听对象还将通过创建一个Timer对象，按照一定帧率更新和重绘游戏的所有对象。

1. Plane

该类是游戏中操控飞机的父类，包含了基本的飞机逻辑。飞机将会依照按键的不同转向不同的方向（8个方向），紧随机尾的气流也会因为位置和方向的不统做出相应的更新。并且通过添加*protected boolean boom = false;*成员来判断是否受到攻击和是否需要绘制受到攻击是的动画。并且在初始化的时候设置一个*Timer* 每隔600毫秒，将*boom*设置为false以防止受攻击动画一直展示下去。

1. Collide

该类为碰撞检测类。由于该项目中的飞机以及子弹都近似与圆，所以该类中添加了*private boolean testFunc(int x1, int y1, int r1, int x2, int y2, int r2)* 方法用于圆形碰撞检测。并且通过在GamePanel 类中的void updateLogic() 方法中不断调用Collide类中的public void test()方法进行检测。一旦检测到发审到碰撞是，将会从ArrarList<Bullet>中抹除该子弹对象，并且将Player\_01对象或者Player\_02对象的表示生命值成员life进行减一操作。

1. 坐标系：

以左上角那点为（0,0），向右则x递增，向下则y递增。

## 参考资料

《Java从入门到精通》

# 2　总体设计

## 2.1　需求规定

### 2.1.1输入、输出

输入：W, A, S, D, H, UP, LEFT, DOWN, RIGHT, G

输出：飞机移动以及子弹发射

### 2.1.2 功能要求

1. 基于Java开发
2. 飞机将有八个方向移动
3. 飞机在停止或移动过程中可以发射子弹
4. 飞机在受到攻击的时候显示受攻击特效以及生命值减一的动画

### 2.1.3 性能要求

本游戏在设计方面本着方便、实用及娱乐性高的宗旨，在对界面进行设计的过程中，始终坚持清晰明了，在性能方面能够实现效率高，不易出错等优点。

游戏主界面应该力求美观，赏心悦目。游戏控制模块应做到易懂、易操作，而且准确率高，不易出错。

## 2.2　运行环境

本次设计的系统环境及工具：

运行环境：Win 10

编程语言：Java

使用工具：IntelliJ IDEA

## 基本设计概念和处理流程

### 2.3.1基本设计概念如下：

该游戏设计的关键是碰撞检测以及页面更新的和重绘。我们通过在游戏运行之初添加一个Timer对象来不断调用GamePanel类的v*oid updateLogic()*方法以及 *public void paint(Graphics g)*方法来保证页面的不断更新。通过事件监听对象来不断更新游戏对象属性。每一次页面更新的时候将其中的一架飞机的位置以及另一架飞机所发射的所有子弹进行碰撞检测。当判定发生碰撞时将会改变飞机的生命值life且抹除子弹在ArrayList中的集合。每一次绘制玩家生命值的时候都会依据飞机的生命值属性来绘制🖤个数。当生命值为0的时候，游戏结束弹出End图片。

### 2.3.2处理流程如下：

开始

显示窗口以及加载游戏对象

玩家各种指令操作

更新游戏对象属性并重绘

判断是否存在玩家生命值为0

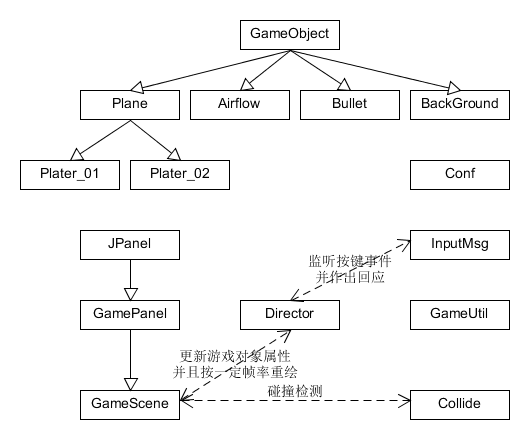
显示结束图片

结束

是

否

## 2.4　结构



*public abstract class* ***GameObject*** *{*

*int pos\_x;*

*int pos\_y;*

*Image image;*

*int angel;*

*BufferedImage IMAGE;*

*GameObject(String path) {*

*this.pos\_y = this.pos\_x = 0;*

*this.image = GameUtil.getImage(path);*

*}*

*GameObject(int pos\_x, int pos\_y, String path, int angel) {*

*this.pos\_x = pos\_x;*

*this.pos\_y = pos\_y;*

*this.IMAGE = GameUtil.getImage(path);*

*this.image = GameUtil.rotateImage(IMAGE, angel);*

*this.angel = angel;*

*}*

*public abstract void draw(Graphics g);*

*public abstract void update();*

*}*

*public class* ***Conf*** *{*

*public final static int WIDTH = 1200;*

*public final static int HEIGHT = 800;*

*final static int plane\_R = 100;*

*final static int bullet\_R = 30;*

*final static int airflow\_R = 50;*

*final static int heart\_R = 25;*

*final static int endImg\_WIDTH = 150;*

*final static int endImg\_HEIGHT = 150;*

*final static int Boom\_R = 75;*

*public static int TIMEINTERVAL = 1000 / 60;*

*}*

*public class* ***GamePanel*** *extends JPanel {*

*private List<GameObject> allObj = new ArrayList<>();*

*protected BackGround backGround;*

*protected Player\_01 plane1;*

*protected Player\_02 plane2;*

*protected Collide collide;*

*protected void addGameObject(GameObject go) {*

*allObj.add(go);*

*}*

*@Override*

*public void paint(Graphics g) {*

*super.paint(g);*

*for (GameObject game : allObj) {*

*game.draw(g);*

*}*

*}*

*void updateLogic() {*

*collide.test();*

*for (GameObject game : allObj) {*

*game.update();*

*}*

*}*

*}*

*public class* ***InputMsg*** *{*

*private Map<Integer, Boolean> keyStatus = new HashMap<>();*

*private static InputMsg inst;*

*private InputMsg() {*

*init();*

*}*

*private void init() {*

*for (int i = 0; i < 127; i++) {*

*keyStatus.put(i, false);*

*}*

*}*

*public static InputMsg getInst() {*

*if (inst == null) {*

*inst = new InputMsg();*

*}*

*return inst;*

*}*

*void setKeyStatus(KeyEvent e, boolean stat) {*

*keyStatus.put(e.getKeyCode(), stat);*

*}*

*public boolean getKey(Integer keycode) {*

*return keyStatus.get(keycode);*

*}*

*}*

*public class* ***GameUtil*** *{*

*private GameUtil() {*

*}*

*public static BufferedImage getImage(String path) {*

*BufferedImage bi = null;*

*try {*

*bi = ImageIO.read(new File(path));*

*} catch (IOException e) {*

*e.printStackTrace();*

*}*

*return bi;*

*}*

*static BufferedImage rotateImage(final BufferedImage bufferedimage,*

*final int degree) {*

*int w = bufferedimage.getWidth();*

*int h = bufferedimage.getHeight();*

*int type = bufferedimage.getColorModel().getTransparency();*

*BufferedImage img;*

*Graphics2D graphics2d;*

*(graphics2d = (img = new BufferedImage(w, h, type))*

*.createGraphics()).setRenderingHint(*

*RenderingHints.KEY\_INTERPOLATION,*

*RenderingHints.VALUE\_INTERPOLATION\_BILINEAR);*

*graphics2d.rotate(Math.toRadians(degree), w / 2, h / 2);*

*graphics2d.drawImage(bufferedimage, 0, 0, null);*

*graphics2d.dispose();*

*return img;*

*}*

*}*

*public class* ***Collide*** *{*

*Player\_01 player01;*

*Player\_02 player02;*

*public Collide(Player\_01 player01, Player\_02 player02) {*

*this.player01 = player01;*

*this.player02 = player02;*

*}*

*public void test() {*

*test\_01();*

*test\_02();*

*}*

*private void test\_01() {*

*int pos\_x = player01.pos\_x + Conf.plane\_R / 2;*

*int pos\_y = player01.pos\_y + Conf.plane\_R / 2;*

*ArrayList<Bullet> temp = new ArrayList<>();*

*for (Bullet \_bullet :*

*player02.bullets) {*

*int p\_x = \_bullet.pos\_x;*

*int p\_y = \_bullet.pos\_y;*

*if (testFunc(pos\_x, pos\_y, Conf.plane\_R / 2, p\_x, p\_y, Conf.bullet\_R / 2)) {*

*temp.add(\_bullet);*

*}*

*}*

*for (Bullet \_bullet :*

*temp) {*

*player02.bullets.remove(\_bullet);*

*if (player01.life != 0) {*

*player01.life--;*

*player01.boom = true;*

*}*

*}*

*}*

*private void test\_02() {*

*int pos\_x = player02.pos\_x + Conf.plane\_R / 2;*

*int pos\_y = player02.pos\_y + Conf.plane\_R / 2;*

*ArrayList<Bullet> temp = new ArrayList<>();*

*for (Bullet \_bullet :*

*player01.bullets) {*

*int p\_x = \_bullet.pos\_x;*

*int p\_y = \_bullet.pos\_y;*

*if (testFunc(pos\_x, pos\_y, Conf.plane\_R / 2, p\_x, p\_y, Conf.bullet\_R / 2)) {*

*temp.add(\_bullet);*

*}*

*}*

*for (Bullet \_bullet :*

*temp) {*

*player01.bullets.remove(\_bullet);*

*if (player02.life != 0) {*

*player02.life--;*

*player02.boom = true;*

*}*

*}*

*}*

*private boolean testFunc(int x1, int y1, int r1, int x2, int y2, int r2) {*

*int length = (int) Math.sqrt((x2 - x1) \* (x2 - x1) + (y2 - y1) \* (y2 - y1));*

*if (length > r1 + r2)*

*return false;*

*else*

*return true;*

*}*

*}*

*public class* ***Director*** *{*

*public JFrame frame = new JFrame();*

*public GamePanel gamePanel;*

*public InputMsg inputMsg;*

*public int timeIntervale;*

*public void setTimeIntervale(int timeIntervale) {*

*this.timeIntervale = timeIntervale;*

*}*

*public Director(String title) {*

*frame.setTitle(title);*

*init();*

*inputMsg = InputMsg.getInst();*

*}*

*public void init() {*

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

*frame.setSize(Conf.WIDTH, Conf.HEIGHT);*

*frame.setLocationRelativeTo(null);*

*frame.setVisible(true);*

*}*

*void setIcon(String path) {*

*frame.setIconImage(GameUtil.getImage(path));*

*}*

*private void initKeyListener() {*

*this.gamePanel.addKeyListener(new KeyAdapter() {*

*@Override*

*public void keyPressed(KeyEvent e) {*

*inputMsg.setKeyStatus(e, true);*

*}*

*@Override*

*public void keyReleased(KeyEvent e) {*

*inputMsg.setKeyStatus(e, false);*

*}*

*});*

*this.gamePanel.requestFocus();*

*}*

*void runGamePanel(GamePanel gamePanel) {*

*this.gamePanel = gamePanel;*

*frame.add(gamePanel);*

*initKeyListener();*

*final GamePanel finalGamePanel = this.gamePanel;*

*Timer timer = new Timer();*

*timer.schedule(new TimerTask() {*

*@Override*

*public void run() {*

*finalGamePanel.updateLogic();*

*finalGamePanel.repaint();*

*}*

*}, 0, this.timeIntervale);*

*}*

*}*

## 2.5　功能需求与程序的关系

1. 游戏界面:
   1. 两架飞机以及在各自方向飞机后的气流
   2. 玩家名字以及生命值显示
2. 与程序的关系：
3. 初始界面将显示两架飞机和两位玩家的名字以及个5颗🖤的生命值。
4. 当按下方向键的时候飞机将会按照指定方向发生移动。
5. 当按下攻击键的时候，飞机将会朝指定方向发射子弹。
6. 当飞机被敌方子弹击中的时候飞机将会产生爆炸动画并且玩家生命值减少一颗🖤。
7. 当有玩家生命值掉至为0的时候游戏界面跳出End图片，游戏结束。

## 3　运行设计

## 3.1　运行模块组合

1. 启动游戏后程序会创建一个每隔1000/60s的计时器用以更新游戏画面。并且初始化按键监听器。
2. 当玩家按下相应按键的时候，游戏引擎将会做出相应判断，之后会更新相应游戏对象的属性，并且在下一次重绘函数调用的时候重绘新的更新后的画面。
3. 为了防止晚间按下攻击键的时候释放连片的子弹，所以每当按下攻击键之后飞机将不允许创建子弹对象，但每个飞机都有一个计时器用来每隔600毫秒将控制创建子弹对象重置为允许。
4. 对于游戏界面中的子弹对象，每一次游戏界面的更新属性，都会与飞机中心坐标计算碰撞检测，一旦检测到碰撞存在，立刻将子弹对象在其集合中抹除，防止下一次重绘对于的子弹，并且对于玩家及飞机属性作出相应调整。
5. 在每次重绘游戏对象的时候都会进行生命值检测，一旦检测到存在生命值为0的时候，结束画面弹出。

## 3.2　运行控制

玩家1移动：

W（上）, S（下）, A（左）, D（右）

玩家1攻击：H

玩家2移动：

↑（上），↓（下），←（左），→（右）

玩家2攻击：G