**Flappy bird小游戏设计实验报告**

[一、 实验目的 2](#_Toc44276895)

[二、 实验环境 2](#_Toc44276896)

[三、 实现功能 2](#_Toc44276897)

[四、 实验内容 2](#_Toc44276898)

[1. 游戏窗体实现 2](#_Toc44276899)

[2. 随机生成柱子 4](#_Toc44276900)

[3. 切换小鸟飞翔的不同状态 6](#_Toc44276901)

[1) 点击后小鸟的仰俯角的变化： 6](#_Toc44276902)

[2) 小鸟飞翔的状态切换： 7](#_Toc44276903)

[4. 游戏结束： 8](#_Toc44276904)

**Flappy bird小游戏设计实验报告**

# 实验目的

了解如何将程序导入Linux系统中，并在Linux环境下运行，熟悉Linux系统操作环境。

利用eclipse实现小游戏运行。

# 实验环境

Linux虚拟机

Java配置：Jdk

开发工具：eclipse

# 实现功能

实现一个flappy bird小游戏的游戏窗体

开始游戏后，在游戏窗体内随机生成高低不同的柱子

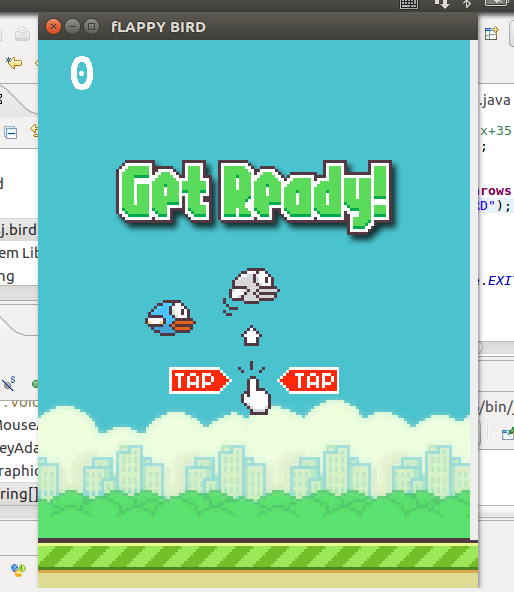
应用鼠标监听创建需要的类，鼠标每点击一次，小鸟就会抬头并向上飞起一定的高度，以此控制小鸟安全飞过柱子

利用循环实现游戏积分，每过一根柱子就累加积分，落地或撞到柱子，游戏结束

游戏结束时查看分数

# 实验内容

## 游戏窗体实现



实现代码：

public class World extends JPanel {

Column column1;

Column column2;

Bird bird;

Ground ground;

BufferedImage background;

BufferedImage gameoverImg;

BufferedImage startImg;

boolean start;

int score;

boolean gameOver;

int index = 0;

public World() throws IOException {

background = ImageIO.read(getClass().getResource("bg.png"));

gameoverImg = ImageIO.read(getClass().getResource("gameover.png"));

startImg = ImageIO.read(getClass().getResource("start.png"));

start();

}

public void start(){

try {

start = false;

gameOver = false;

bird = new Bird();

ground = new Ground();

column1 = new Column(1);

column2 = new Column(2);

score = 0;

} catch (Exception e) {

e.printStackTrace();

}

}

public void paint(Graphics g) {

//抗锯齿代码

// Graphics2D g2 = (Graphics2D)g;

// RenderingHints qualityHints = new RenderingHints(

// RenderingHints.KEY\_ANTIALIASING,

// RenderingHints.VALUE\_ANTIALIAS\_ON);

// qualityHints.put(RenderingHints.KEY\_RENDERING,

// RenderingHints.VALUE\_RENDER\_QUALITY);

// g2.setRenderingHints(qualityHints);

//绘制背景

g.drawImage(background, 0, 0, null);

//绘制柱子

column1.paint(g);

column2.paint(g);

//绘制地面

ground.paint(g);

//绘制分数

Font font = new Font(Font.MONOSPACED, Font.BOLD, 45);

g.setFont(font);

g.setColor(Color.white);

g.drawString(score+"", 30, 50);

//绘制小鸟

bird.paint(g);

//绘制结束状态

if(gameOver){

//g.drawString("Game Over!", 70 , 190);

g.drawImage(gameoverImg, 0, 0, null);

return;

}

if(! start){

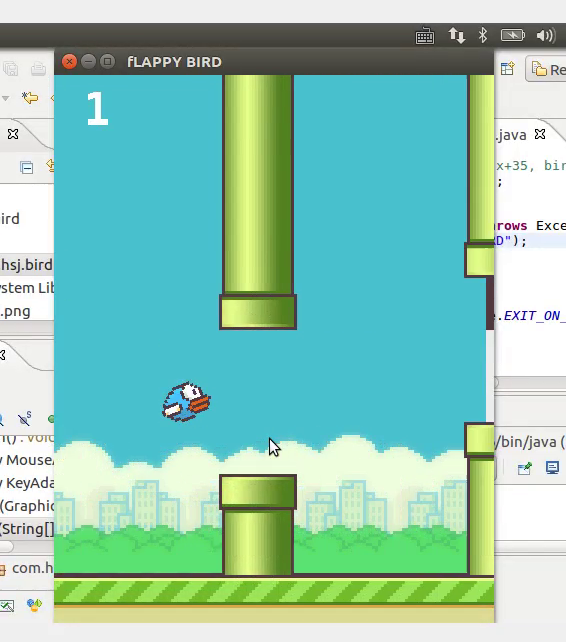
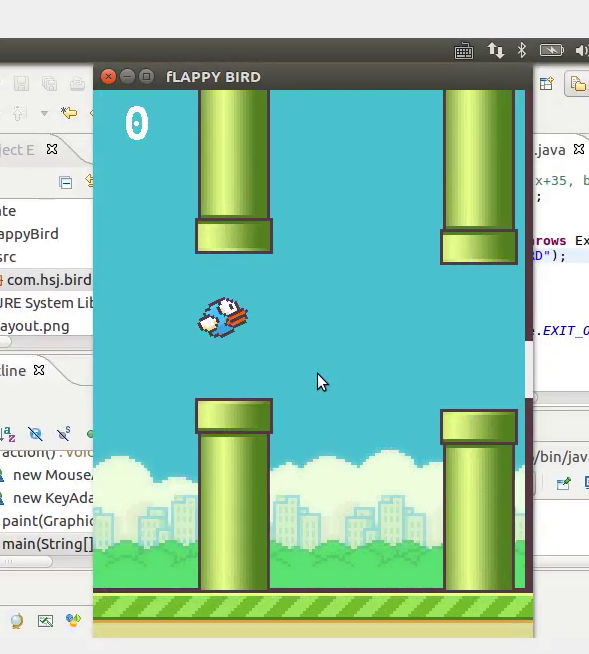
//g.drawString("Start >>>", bird.x+35, bird.y);

g.drawImage(startImg, 0, 0, null);

}

}

## 随机生成柱子



实现代码：

public class Column {

BufferedImage image;

//以柱子的中间作为柱子的位置

int x;

int y; // 140 ~ 280

int width;

int height;

int gap = 144;

Random r = new Random();

int distance = 245;

public Column(int n) throws IOException {

image = ImageIO.read(getClass().getResource("column.png"));

width = image.getWidth();

height = image.getHeight();

this.x = distance \* n + 80 + 432;

this.y = r.nextInt(120) + 220;

}

public void step(){

x--;

if(x<=-width/2){

x = distance\*2-width/2;

this.y = r.nextInt(120) + 220;

}

}

public void paint(Graphics g){

//g.drawRect(x-width/2, y-height/2, width, height/2-gap/2);

//g.drawRect(x-width/2, y+gap/2, width, height/2-gap/2);

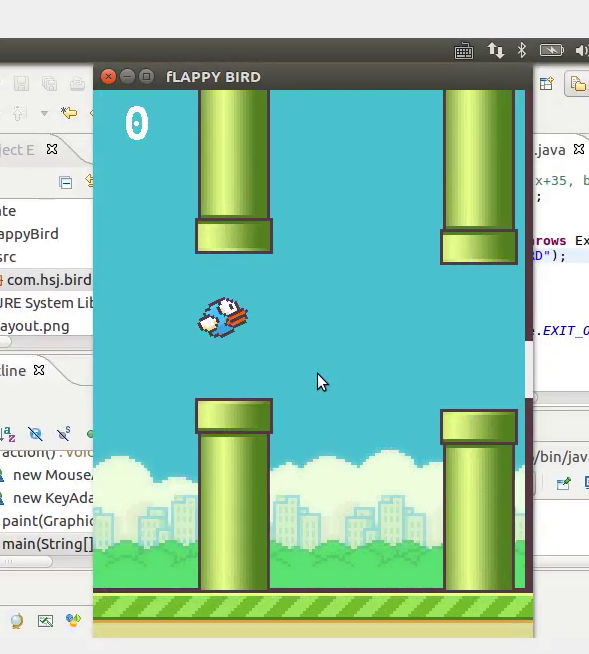
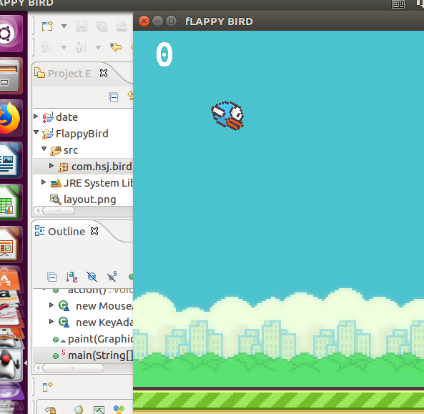
g.drawImage(image, x-width/2, y-height/2, null);

}

}

## 切换小鸟飞翔的不同状态

### 点击后小鸟的仰俯角的变化：



实现代码：

public class Bird {

/\*\* 鸟飞行位置, 这个位置是鸟的中心位置 \*/

int x; int y;

/\*\* 飞行角度 \*/

double angle;

/\*\* 动画帧 \*/

BufferedImage[] images;

/\*\* 当前图片 \*/

BufferedImage image;

/\*\* 当前图片序号 \*/

int index = 0;

/\*\* 重力加速度 \*/

final double g;

/\*\* 时间间隔秒 \*/

final double t;

/\*\* 初始上抛速度 \*/

final double v0;

/\*\* 当前上抛速度 \*/

double speed;

/\*\* 移动距离 \*/

double s;

/\*\* 鸟的范围, 鸟的范围是一个正方形区域, 中心点是x,y \*/

int size = 40;

public Bird() throws Exception {

this.g = 4; //重力加速度

this.t = 0.25; //每次计算的间隔时间

this.v0 = 20; //初始上抛速度

x = 132; //鸟的初始位置

y = 275; //鸟的初始位置

//加载动画帧

images = new BufferedImage[8];

for(int i=0; i<8; i++){

images[i] = ImageIO.read(getClass().getResource(i+".png"));

}

image = images[0];

}

/\*\* 飞行一步

\* 竖直上抛位移计算

\* (1) 上抛速度计算 V=Vo-gt

(2) 上抛距离计算 S=Vot-1/2gt^2

\* \*/

public void step(){

//V0 是本次

double v0 = speed;

//计算垂直上抛运动, 经过时间t以后的速度,

double v = v0 - g\*t;

//作为下次计算的初始速度

speed = v;

//计算垂直上抛运动的运行距离

s = v0\*t - 0.5 \* g \* t \* t;

//将计算的距离 换算为 y坐标的变化

y = y - (int)s;

//计算小鸟的仰角,

angle = -Math.atan(s/8);

}

### 小鸟飞翔的状态切换：



实现代码：

public void fly(){

//更换小鸟的动画帧图片, 其中/4 是为了调整动画更新的频率

index++;

image = images[(index/8)%images.length];

}

/\*\* 小鸟向上飞扬 \*/

public void flappy(){

//每次小鸟上抛跳跃, 就是将小鸟在当前点重新以初始速度 V0 向上抛

speed = v0;

}

//绘制时并发执行的, 要同步避免 旋转坐标系对其他绘制方法的影响

public synchronized void paint(Graphics g){

//g.drawRect(x-size/2, y-size/2, size, size);

Graphics2D g2 = (Graphics2D)g;

//旋转绘图坐标系, 绘制

g2.rotate(angle, this.x, this.y);

//以x,y 为中心绘制图片

int x = this.x-image.getWidth()/2;

int y = this.y-image.getHeight()/2;

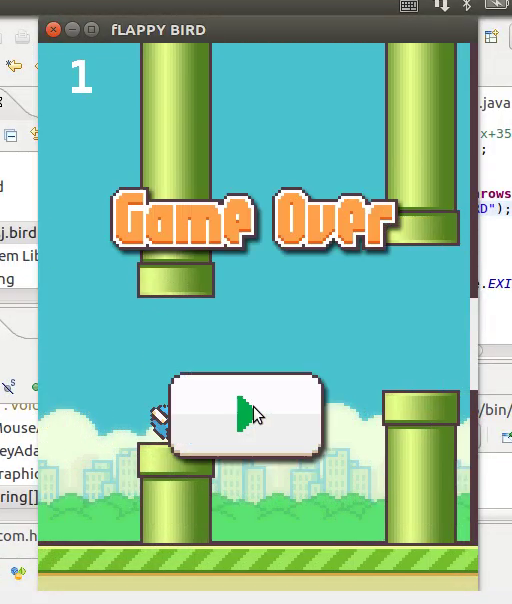
g.drawImage(image, x, y, null);

//旋转回来

g2.rotate(-angle, this.x, this.y);

}

## 游戏结束：



判断游戏结束代码：

public String toString() {

return "Bird [x=" + x + ", y=" + y + ", g=" + g + ", t=" + t + ", v0="

+ v0 + ", speed=" + speed + ",s="+s+"]";

}

/\*\* 判断鸟是否通过柱子 \*/

public boolean pass(Column col1, Column col2) {

return col1.x==x || col2.x==x;

}

/\*\* 判断鸟是否与柱子和地发生碰撞 \*/

public boolean hit(Column column1, Column column2, Ground ground) {

//碰到地面

if(y-size/2 >= ground.y){

return true;

}

//碰到柱子

return hit(column1) || hit(column2);

}

/\*\* 检查当前鸟是否碰到柱子 \*/

public boolean hit(Column col){

//如果鸟碰到柱子: 鸟的中心点x坐标在 柱子宽度 + 鸟的一半

if( x>col.x-col.width/2-size/2 && x<col.x+col.width/2+size/2){

if(y>col.y-col.gap/2+size/2 && y<col.y+col.gap/2-size/2 ){

return false;

}

return true;

}

return false;

}

}

实验总结：

通过这次Linux 大作业程序编写，让我更好的了解到Linux环境下如何运行代码，更加熟悉了Linux系统的环境，也认识到许多编程中出现的错误，学到了很多东西，很感谢老师一学期的知识讲授，也很感激同学的耐心讲解。