TTS 7.0 COOKBOOK

（JAVA BIRD UNIT 01）

版本编号 7.0

2013-12

达内IT培训集团

JAVA BIRD Unit01

1. Bird项目

* 问题

这款游戏的起源是越南独立开发者开发的[手机游戏](http://www.3366.com/heji/361_new.shtml)，短时间竟占领了全球各大App Store免费排行榜首位。游戏中，玩家控制一只小鸟飞过一个个柱子的间隙。飞得越远分数越高，看玩家能使小鸟在空中坚持多久。初始界面如图-1所示。

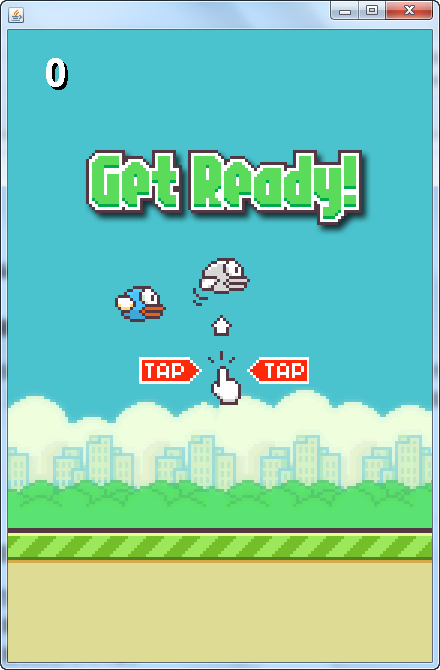


图 – 1

玩家在如图-1所示的界面的任意位置，按下鼠标左键，开始游戏。

游戏开始以后，玩家需要不断控制点击屏幕的频率来调节小鸟的飞行高度和降落速度，让小鸟顺利的通过画面右端的柱子间隙。如果玩家不小心擦碰到了柱子或掉落到地面上，则游戏宣告结束。如图-2所示。

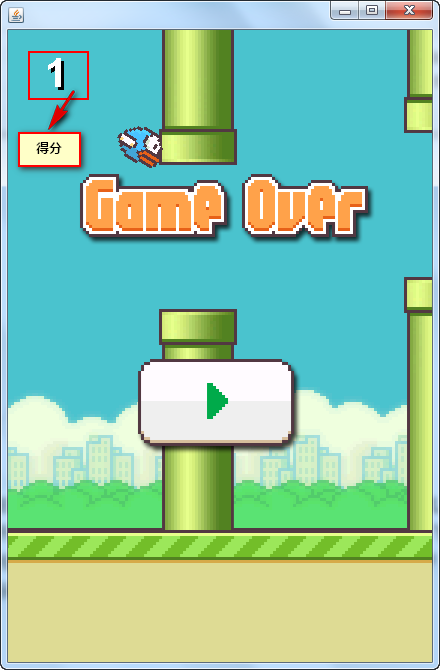


图 – 2

如图-2所示的左上角显示了用户的得分，每通过一个柱子的间隙得1分。另外，此时玩家可以在如图-2所示的界面的任意位置，按下鼠标左键，重新开始游戏。

* 方案

软件的开发过程如下：

1. 需求(软件功能的文字描述)

2. 业务需求分析: 找对象，以及对象之间的关系。本项目中对象如下所示：

BirdGame

|-- 地面Ground

|-- 小鸟Bird

|-- 多个柱子Column

3. 软件概要设计：计算机只能按顺序“处理数据”。

数据建模：使用一个数据模型，描述对象的关系。使用绘图坐标系作为参考模型，鸟是正方形区域；柱子是长方形区域，中间有间隙；地面是矩形。 对象是结构化的“数据”，是一组有关系的数据。

4. 类的设计

本案例中的类，及类之间的关系如图-3所示。



图 – 3

* 步骤

实现此案例需要按照如下步骤进行。

步骤一：新建工程和包

新建名为BirdGrame的Java工程；然后，在工程下的src目录下新建包com.tarena.bird； 最后，将该工程所需的图片拷贝到该包下，工程结构如图-4所示：

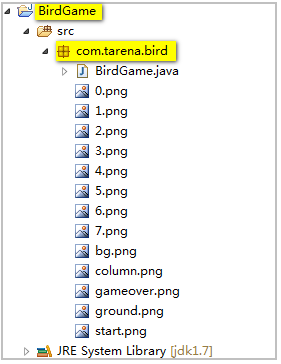


图- 4

在项目中，包的名字一般为公司域名倒过来，再加上项目名称，即为包名。如图-4中中的包名为com.tarena.bird，其中，com.tarena是达内公司的域名倒过来，bird为本项目的名称。

步骤二：构建工程的结构

首先，新建类BirdGame，该类由public修饰并且该类继承自JPanel类，扩展Java AWT的面板功能；然后，新建Groud类，表示游戏中的地面；新建Column表示游戏中的柱子；新建Bird表示游戏中的鸟，游戏中的各个对象如图-5所示。

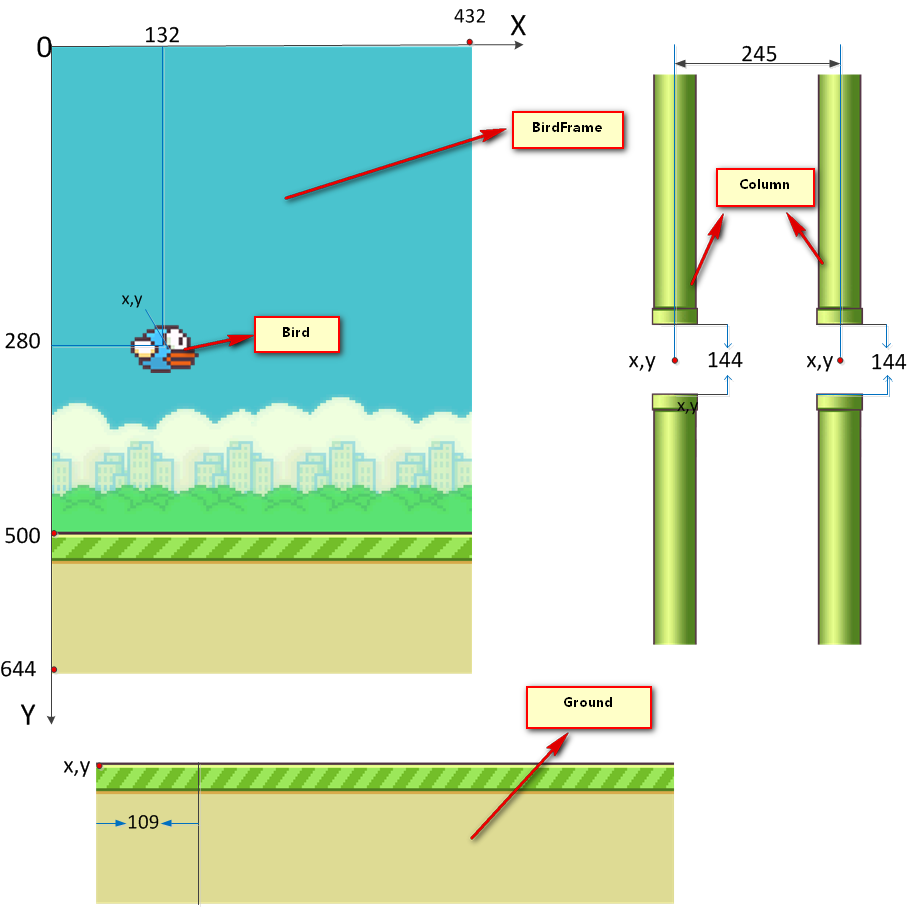


图 – 5

代码如下所示：

package com.tarena.bird;

import javax.swing.JPanel;

public class BirdGame extends JPanel {

/\*\* 启动软件的方法 \*/

public static void main(String[] args) throws Exception {

}

}

/\*\* 地面 \*/

class Ground {

}

/\*\* 柱子类型，x,y是柱子的中心点的位置 \*/

class Column {

}

/\*\* 鸟类型, x,y是鸟类型中心的位置 \*/

class Bird {

}

步骤三：为 BirdGame类添加属性

添加的代码如图-6所示。

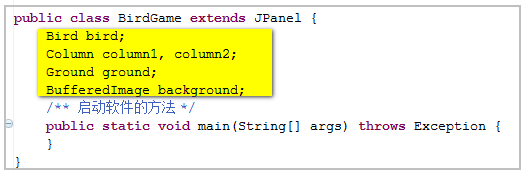


图 – 6

步骤四：为 Ground类添加属性

Ground类中，各个属性表示的含义如图-7所示。



图 – 7

在Ground类中添加的代码如图-8所示。

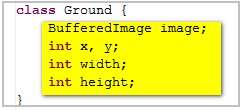


图 – 8

另外，从图-8中可以看出，还有一个image属性，该属性表示为Ground类的贴图。

步骤五：为 Column类添加属性

Column类中，各个属性表示的含义如图-9所示。

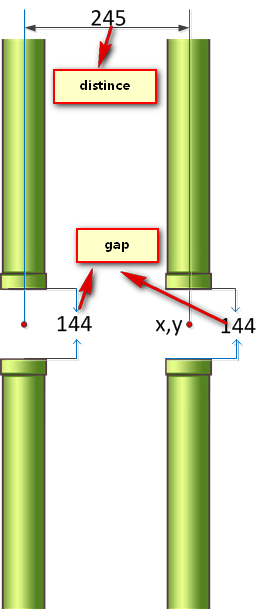


图 – 9

在Column类中添加的代码如图-10所示。

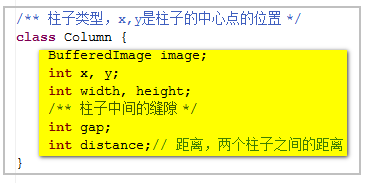


图 – 10

另外，从图-10中可以看出，还有一个image属性，该属性表示为Column类的贴图。

步骤六：为 Bird类添加属性

代码如图-11所示。

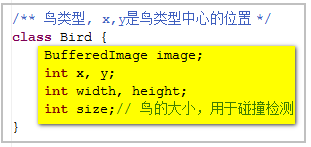


图 – 11

Bird的属性image，表示为Bird类的贴图；x，y表示鸟类型的中心位置；width，height指的是鸟图片的宽和高；本案例中，小鸟的大小用一个正方形区域来表示，size即表示该正方形的边长，用于后续的碰撞检测。

步骤七：为 BirdGrame类添加构造方法，初始化属性

代码如图-12所示。



图 – 12

从图-12中，可以看出实例化Column类的时候，出现了编译错误。这是因为，还没有为Column类编写带一个参数的构造器。

步骤八：为Column类添加构造方法，初始化属性

代码如图-13所示。



图 – 13

下面介绍一下x、y坐标的计算方式，请看图-14。



图 – 14

从图-14可以看出，第一个柱子的x坐标x1=432+118=550，第二根柱子的x坐标x2=550+245，因此，柱子的x坐标可以总结为：x=550+(n-1)\*245，其中n表示第几根柱子，例如：1，2。

y坐标的计算，请看图-15。

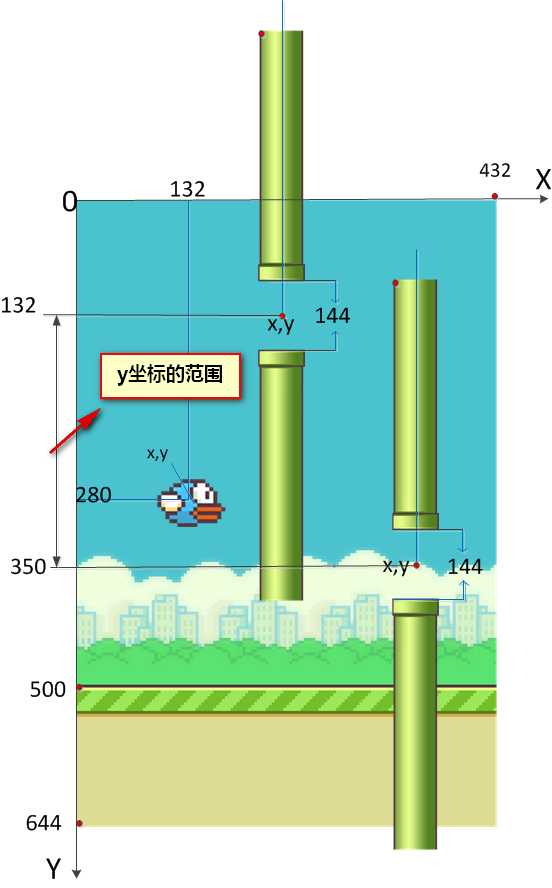


图 – 15

另外，从图-15可以看出，柱子y坐标范围规定为132到350之间的随机数。使用Random类的nextInt方法生产[0，218）之间的随机数，再加上132，因此计算出来的范围即为132到350之间。代码如下所示：

y = random.nextInt(218) + 132;

步骤九：为Ground类添加构造方法，初始化属性

代码如图-16所示。



图 – 16

地面初始的坐标位置如图-17所示。

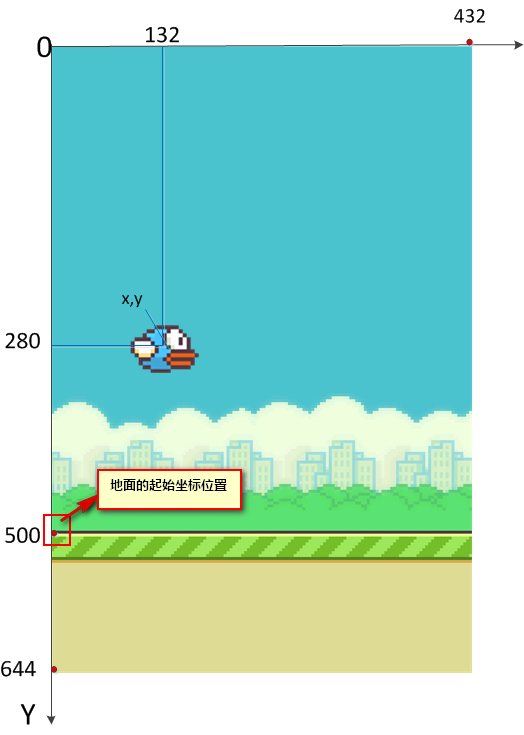


图 – 17

步骤十：为Bird类添加构造方法，初始化属性

代码如图-18所示。



图 – 18

鸟的初始坐标位置如图-19所示。

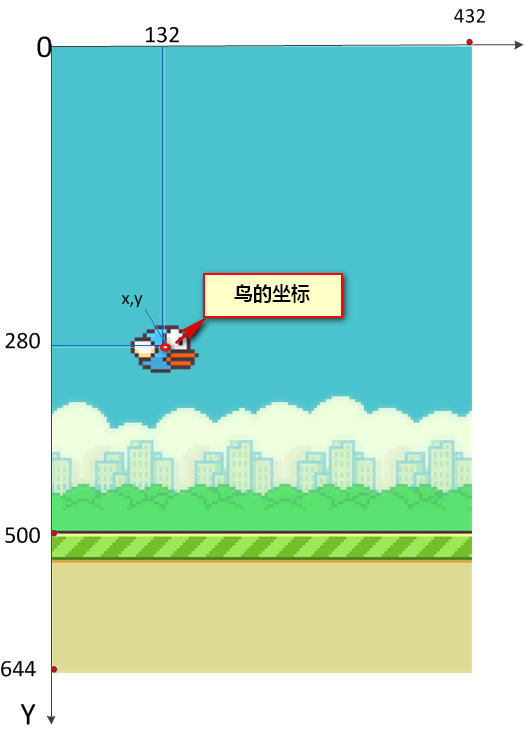


图 – 19

步骤十一：编写main方法

在main方法中，设置窗口的大小、居中、点击窗口的右上角“X”关闭窗口以及设置窗口可见，代码如图-20所示。

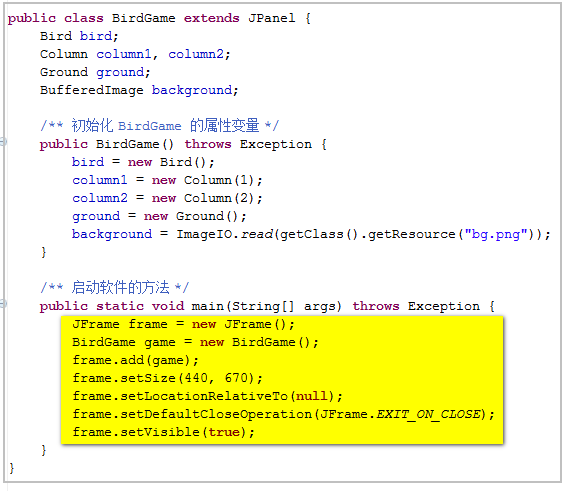


图 – 20

步骤十二：绘制界面

在BirdGame类中重写paint方法，绘制界面。首先绘制地面，代码如图-21所示。

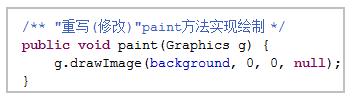


图 – 21

使用Graphics的drawImage方法，绘制背景界面时，绘制的起始坐标点为（0，0）。

然后，绘制两个柱子，代码如图-22所示。

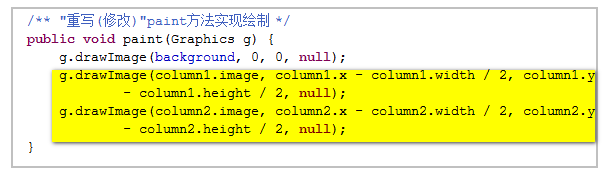


图 – 22

绘制两个柱子时，绘制的起始坐标与Column类的属性x、y的值不同，需要做一些更改，请看图-23。

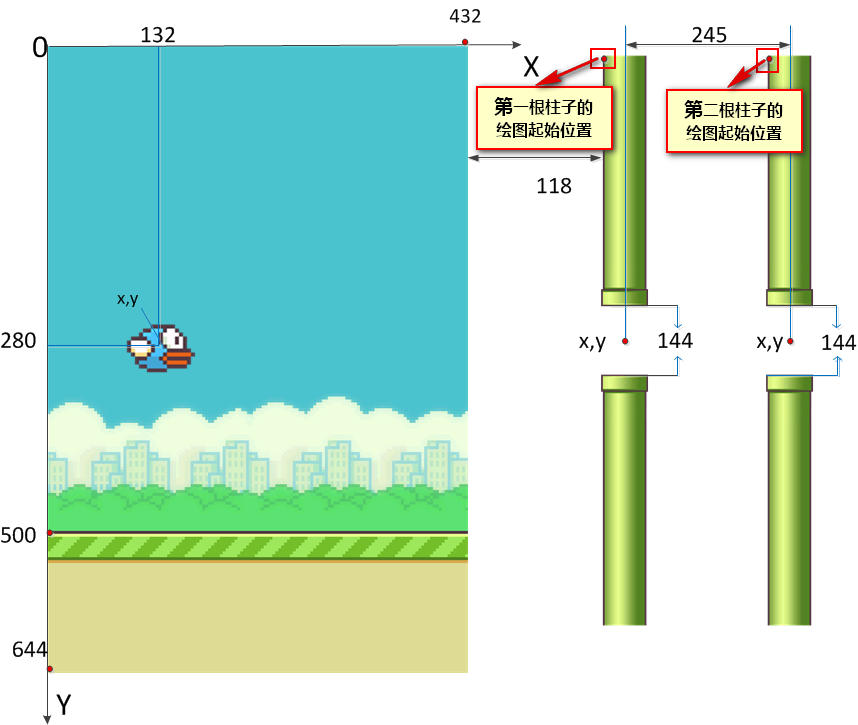


图 – 23

从图-23可以看出，绘图起始位置x=柱子.x-1/2柱子.width; 绘图起始位置y=柱子.y-1/2柱子.height。

接着，绘制地面，代码如图-24所示。

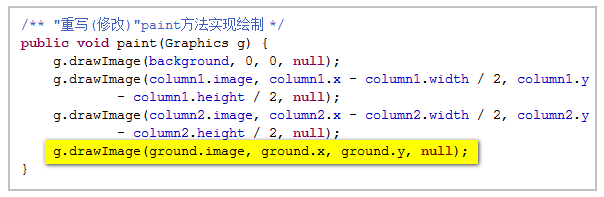


图 – 24

最后，绘制小鸟，代码如图-25所示。

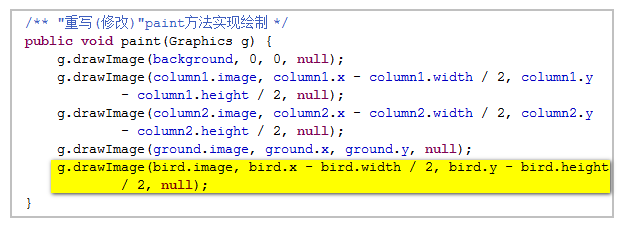


图 – 25

从图-25可以看出，绘制小鸟的时候，起始坐标位置也需要换算一下，原理请参考柱子的绘制原理。

步骤十三：运行

运行BirdFrame类，显示的界面效果如图-26所示。

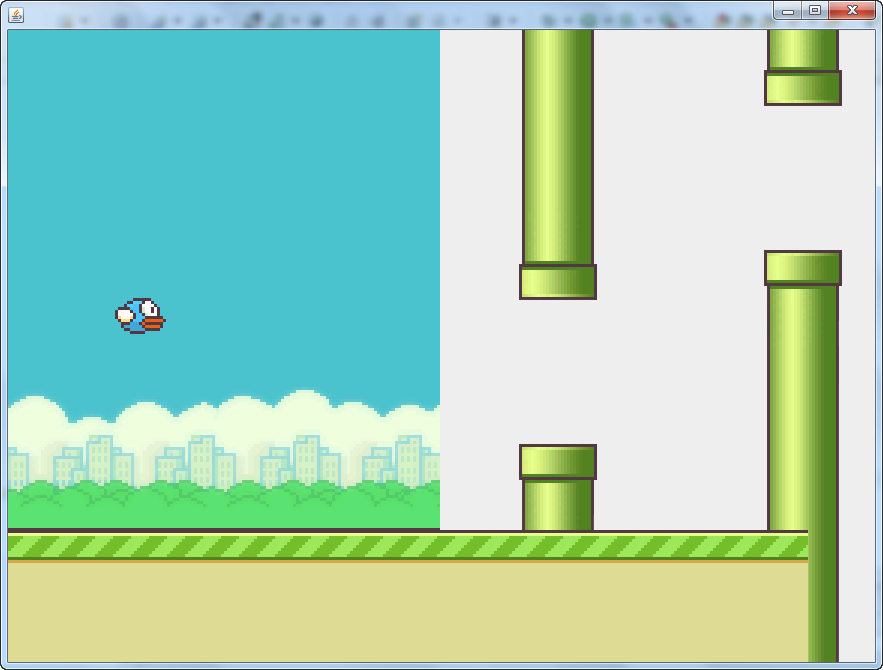


图 – 26

步骤十四：实现地面的运动

首先，在Ground类中添加step方法，代码如图-27所示。



图 – 27

step方法的作用为使地面移动一步，如果移动到x坐标为-109，则将x坐标设置为0，形成滚动的效果。地面的坐标位置如图-28所示。



图 – 28

然后，在BirdGame类中添加action()方法，在方法中利用死循环，在该循环中做以下操作来实现，地面的运动：

1）修改地面的位置，调用Ground类的step方法来实现。

2）重新绘制界面，调用JPanel的repaint方法来实现。

3）休眠 1/30秒，即设置屏幕的刷新率为1秒30次，调用Thread类的sleep方法来实现。

在BirdFrame中添加的代码如图-29所示。

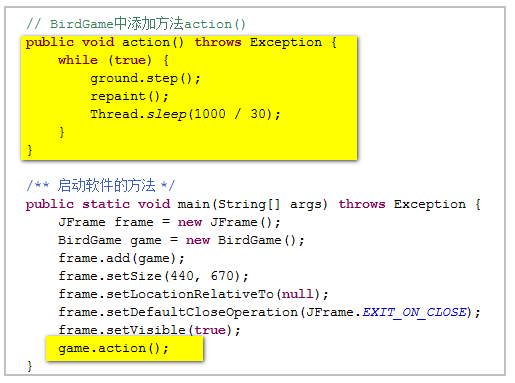


图 – 29

步骤十五：实现柱子的移动

首先，要实现柱子的移动，要计算出将柱子移动回出场位置的坐标，请看图-30。

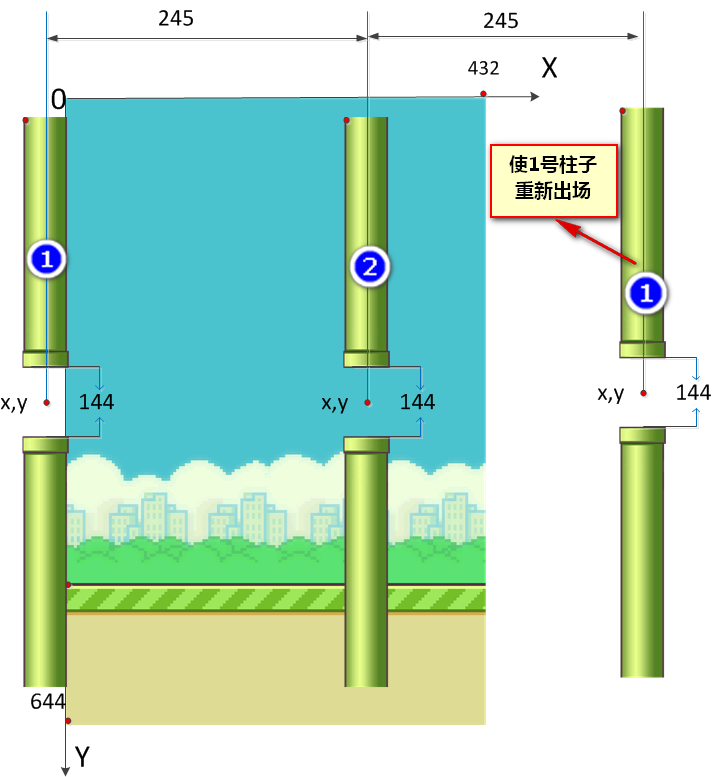


图 – 30

从图-30可以看出，移回出场位置的1号柱子的x坐标为distance \* 2 - width / 2，y坐标的范围不变，依然是132到350之间的随机数。在Colunm类中添加的代码如图-31所示。

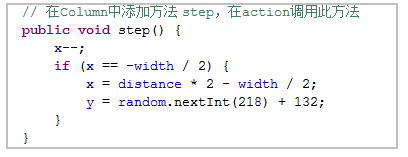


图 – 31

另外，从图-30可以看出，当柱子的x坐标为-width/2时，柱子移动出界面。

然后，在BirdGame类的action方法，添加对step方法的调用，代码如图-32所示。

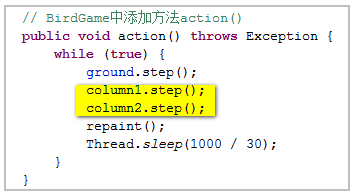


图 – 32

步骤十六：实现鸟的运动

首先，实现上抛运行。计算经过时间t的位移s的计算公式，请看图-33。



图 – 33

经过时间t的位置s的公式为：

s = v0 \* t + g \* t \* t / 2;

从图-33可以看出，经过时间t后小鸟的纵坐标位置y1的计算公式为：

y1 = y - s;

经过时间t后小鸟的运行速度v为：

v = v0 - g \* t;

此时，在Bird类添加的代码如下所示：

/\*\* 鸟类型, x,y是鸟类型中心的位置 \*/

class Bird {

BufferedImage image;

int x, y;

int width, height;

int size;// 鸟的大小，用于碰撞检测

// 在Bird类中增加属性，用于计算鸟的位置

double g;// 重力加速度

double t;// 两次位置的间隔时间

double v0;// 初始上抛速度

double speed;// 是当前的上抛速度

double s;// 是经过时间t以后的位移

double alpha;// 是鸟的倾角 弧度单位

public Bird() throws Exception {

image = ImageIO.*read*(getClass().getResource("0.png"));

width = image.getWidth();

height = image.getHeight();

x = 132;

y = 280;

size = 40;

g = 4;

v0 = 20;

t = 0.25;

speed = v0;

s = 0;

alpha = 0;

}

// 在Bird中添加鸟的移动方法

public void step() {

double v0 = speed;

s = v0 \* t + g \* t \* t / 2;// 计算上抛运动位移

y = y - (int) s;// 计算鸟的坐标位置

double v = v0 - g \* t;// 计算下次的速度

speed = v;

}

}

然后，实现鸟的动画处理。使用数组，存储要显示的图片，共存储8张图片，鸟每运动一次换一张图片显示，以达到鸟的动画效果。此时，Bird类中添加的代码如下所示：

/\*\* 鸟类型, x,y是鸟类型中心的位置 \*/

class Bird {

BufferedImage image;

int x, y;

int width, height;

int size;// 鸟的大小，用于碰撞检测

// 在Bird类中增加属性，用于计算鸟的位置

double g;// 重力加速度

double t;// 两次位置的间隔时间

double v0;// 初始上抛速度

double speed;// 是当前的上抛速度

double s;// 是经过时间t以后的位移

double alpha;// 是鸟的倾角 弧度单位

// 定义一组（数组）图片，是鸟的动画帧

BufferedImage[] images;

// 是动画帧数组元素的下标位置

int index;

public Bird() throws Exception {

image = ImageIO.*read*(getClass().getResource("0.png"));

width = image.getWidth();

height = image.getHeight();

x = 132;

y = 280;

size = 40;

g = 4;

v0 = 20;

t = 0.25;

speed = v0;

s = 0;

alpha = 0;

// 创建数组,创建8个元素的数组

// 是8个空位置，没有图片对象，

// 8个位置的序号: 0 1 2 3 4 5 6 7

images = new BufferedImage[8];

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

// i = 0 1 2 3 4 5 6 7

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

}

index = 0;

}

// 在Bird中添加飞翔(fly)的代码

public void fly() {

index++;

image = images[（index/12） % 8];

}

// 在Bird中添加鸟的移动方法

public void step() {

double v0 = speed;

s = v0 \* t + g \* t \* t / 2;// 计算上抛运动位移

y = y - (int) s;// 计算鸟的坐标位置

double v = v0 - g \* t;// 计算下次的速度

speed = v;

}

}

从上述代码中可以看出，每循环到第8张图片，则将index的值设置为0，即从第1张图片再次开始显示。

另外，fly方法实现了鸟的飞翔。一个数与8进行取余数，可以获取到0到7之间的余数，正好是images中图片的下标。index/12的目的是使鸟的运动速度放慢，即切换照片的频率放慢。

接着，在BirdFrame类的action方法中，调用Bird的fly方法和step方法。运行后，会发现鸟可以动起来了。代码如图-34所示。

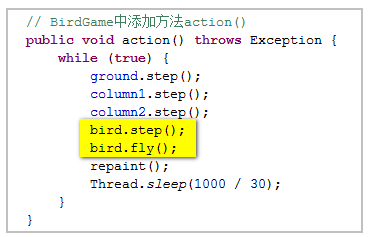


图 – 34

最后，计算倾角。请看图-35，了解什么是倾角。

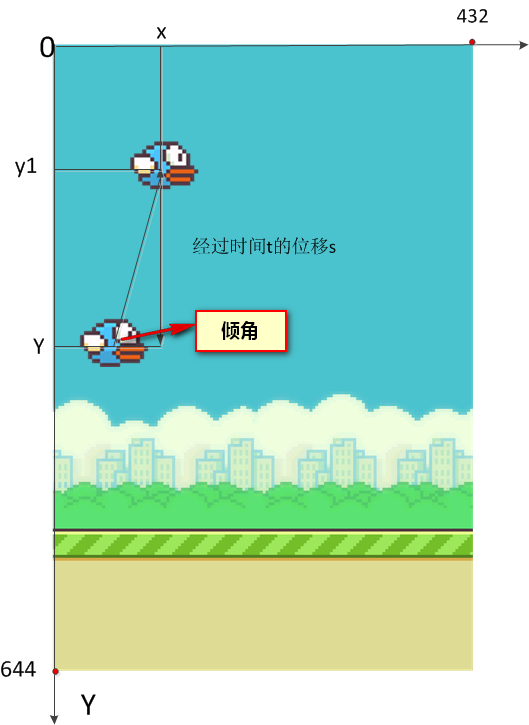


图 – 35

假设水平移动的位移为固定数8，那么倾角=反正切s/8。在Bird类添加的代码如下所示：

/\*\* 鸟类型, x,y是鸟类型中心的位置 \*/

class Bird {

BufferedImage image;

int x, y;

int width, height;

int size;// 鸟的大小，用于碰撞检测

// 在Bird类中增加属性，用于计算鸟的位置

double g;// 重力加速度

double t;// 两次位置的间隔时间

double v0;// 初始上抛速度

double speed;// 是当前的上抛速度

double s;// 是经过时间t以后的位移

double alpha;// 是鸟的倾角 弧度单位

// 在Bird类中定义

// 定义一组（数组）图片，是鸟的动画帧

BufferedImage[] images;

// 是动画帧数组元素的下标位置

int index;

public Bird() throws Exception {

image = ImageIO.*read*(getClass().getResource("0.png"));

width = image.getWidth();

height = image.getHeight();

x = 132;

y = 280;

size = 40;

g = 4;

v0 = 20;

t = 0.25;

speed = v0;

s = 0;

alpha = 0;

// 创建数组,创建8个元素的数组

// 是8个空位置，没有图片对象，

// 8个位置的序号: 0 1 2 3 4 5 6 7

images = new BufferedImage[8];

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

// i = 0 1 2 3 4 5 6 7

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

}

index = 0;

}

// 在Bird中添加鸟的移动方法

public void step() {

double v0 = speed;

s = v0 \* t + g \* t \* t / 2;// 计算上抛运动位移

y = y - (int) s;// 计算鸟的坐标位置

double v = v0 - g \* t;// 计算下次的速度

speed = v;

// 调用Java API提供的反正切函数，计算倾角

alpha = Math.*atan*(s / 8);

}

}

上述代码中，调用Java API提供的反正切方法*atan*，计算了倾角。

接着，在BirdFrame类的paint中，为达到鸟的飞翔效果，使用Java API，使小鸟旋转，代码如图-36所示。



图 – 36

步骤十七：实现鼠标事件

要实现鼠标的按下事件，步骤如下：

1．使用“匿名内部类”声明事件"处理方法"。即在BirdFrame类的action方法声明事件处理方法；

2. 首先，在Bird中添加flappy方法，以实现鸟的飞扬。然后，在处理方法中，调用Bird类的flappy方法；

3. 注册事件监听。即在BridFrame类的action方法中，注册事件监听。

在BirdFrame类的添加的代码如图-37所示。



图 – 37

在Bird类添加的代码如图-38所示。

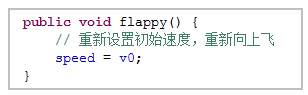


图 – 38

从图-38可以看出，实现flappy方法，只需将速度设置为初始速度v0即可。

步骤十八：实现记分

要实现记分，需要进行以下操作：

1. 在BirdFrame类中增加属性score，用于记分，score初始化为0。

2. 在BirdFrame类的action方法的主循环中，添加判分逻辑。当柱子的x坐标与鸟的x坐标重合时，则加一分。

3. 在BirdFrame类的paint方法中将分数画出来。

在BirdFrame类中增加的代码如下所示：

public class BirdGame extends JPanel {

Bird bird;

Column column1, column2;

Ground ground;

BufferedImage background;

// 分数

int score;

/\*\* 初始化 BirdGame 的属性变量 \*/

public BirdGame() throws Exception {

bird = new Bird();

column1 = new Column(1);

column2 = new Column(2);

ground = new Ground();

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

}

/\*\* "重写(修改)"paint方法实现绘制 \*/

public void paint(Graphics g) {

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

g.drawImage(column1.image, column1.x - column1.width / 2, column1.y

- column1.height / 2, null);

g.drawImage(column2.image, column2.x - column2.width / 2, column2.y

- column2.height / 2, null);

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

// 旋转(rotate)绘图坐标系，是API方法

Graphics2D g2 = (Graphics2D) g;

g2.rotate(-bird.alpha, bird.x, bird.y);

g.drawImage(bird.image, bird.x - bird.width / 2, bird.y - bird.height

/ 2, null);

g2.rotate(bird.alpha, bird.x, bird.y);

// 在paint方法中添加绘制分数的算法

Font f = new Font(Font.*SANS\_SERIF*, Font.*BOLD*, 40);

g.setFont(f);

g.drawString("" + score, 40, 60);

g.setColor(Color.*WHITE*);

g.drawString("" + score, 40 - 3, 60 - 3);

}

// BirdGame中添加方法action()

public void action() throws Exception {

MouseListener l = new MouseAdapter() {

//鼠标按下

public void mousePressed(MouseEvent e) {

bird.flappy();

}

};

// 将l挂接到当前的面板（game）上

addMouseListener(l);

while (true) {

ground.step();

column1.step();

column2.step();

bird.step();

bird.fly();

// 计分逻辑

if (bird.x == column1.x || bird.x == column2.x) {

score++;

}

repaint();

Thread.*sleep*(1000 / 30);

}

}

/\*\* 启动软件的方法 \*/

public static void main(String[] args) throws Exception {

JFrame frame = new JFrame();

BirdGame game = new BirdGame();

frame.add(game);

frame.setSize(440, 670);

frame.setLocationRelativeTo(null);

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

frame.setVisible(true);

game.action();

}

}

步骤十九：实现鸟的碰撞检测

1. 实现鸟碰撞地面。

1) 为BirdGame增加boolean变量 gameOver，该变量用于标识游戏是否结束，如果gameOver的值为true，则表示游戏结束了；如果gameOver的值为false，则表示游戏还没有结束。gameOver的默认值为false。

2）当检测到碰到地面时，即 gameOver的值为true。 在主循环中检测是否碰到地面。在Bird类中增加方法hit(Ground)检测鸟是否碰地面。当鸟的y坐标减去size/2大于等于地面的y坐标时，则与地面发生碰撞。

3） 修改BirdFrame类的action方法中的主循环，当游戏没有结束时，执行鸟和柱子的移动。也就是说，如果游戏结束了，鸟和柱子就不移动了。

4） 在BirdFrame类的paint方法添加当 gameOver的值为 true时，显示游戏结束界面。

在BirdFrame类添加的代码如下所示：

public class BirdGame extends JPanel {

Bird bird;

Column column1, column2;

Ground ground;

BufferedImage background;

boolean gameOver;

BufferedImage gameOverImage;

// 分数

int score;

/\*\* 初始化 BirdGame 的属性变量 \*/

public BirdGame() throws Exception {

gameOver = false;

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

bird = new Bird();

column1 = new Column(1);

column2 = new Column(2);

ground = new Ground();

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

}

/\*\* "重写(修改)"paint方法实现绘制 \*/

public void paint(Graphics g) {

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

g.drawImage(column1.image, column1.x - column1.width / 2, column1.y

- column1.height / 2, null);

g.drawImage(column2.image, column2.x - column2.width / 2, column2.y

- column2.height / 2, null);

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

// 旋转(rotate)绘图坐标系，是API方法

Graphics2D g2 = (Graphics2D) g;

g2.rotate(-bird.alpha, bird.x, bird.y);

g.drawImage(bird.image, bird.x - bird.width / 2, bird.y - bird.height

/ 2, null);

g2.rotate(bird.alpha, bird.x, bird.y);

// 在paint方法中添加绘制分数的算法

Font f = new Font(Font.SANS\_SERIF, Font.BOLD, 40);

g.setFont(f);

g.drawString("" + score, 40, 60);

g.setColor(Color.WHITE);

g.drawString("" + score, 40 - 3, 60 - 3);

if (gameOver) {

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

}

}

// BirdGame中添加方法action()

public void action() throws Exception {

MouseListener l = new MouseAdapter() {

//鼠标按下

public void mousePressed(MouseEvent e) {

bird.flappy();

}

};

// 将l挂接到当前的面板（game）上

addMouseListener(l);

while (true) {

if (!gameOver) {

ground.step();

column1.step();

column2.step();

bird.step();

bird.fly();

}

if (bird.hit(ground)) {

gameOver = true;

}

// 计分逻辑

if (bird.x == column1.x || bird.x == column2.x) {

score++;

}

repaint();

Thread.sleep(1000 / 30);

}

}

/\*\* 启动软件的方法 \*/

public static void main(String[] args) throws Exception {

JFrame frame = new JFrame();

BirdGame game = new BirdGame();

frame.add(game);

frame.setSize(440, 670);

frame.setLocationRelativeTo(null);

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

frame.setVisible(true);

game.action();

}

}

在Bird类添加的碰撞方法的代码如图-39所示。

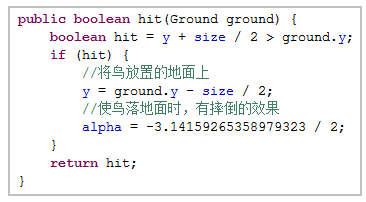


图 – 39

2. 实现鸟碰撞柱子。请看如-40。

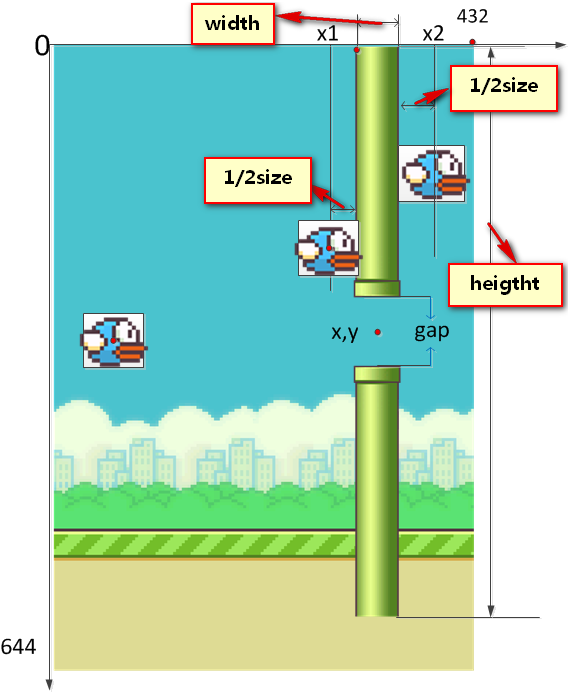


图 – 40

从图-40可以看出，小鸟与柱子发生碰撞的x坐标计算如下：

x1=column.x-width/2-size/2

x2=column.x+width/2+size/2

当鸟的x坐标大于x1（x>x1）并且小于x2（x<x2）时与柱子发生碰撞。但是如果在缝隙中，则不发生碰撞，请看如图-39。

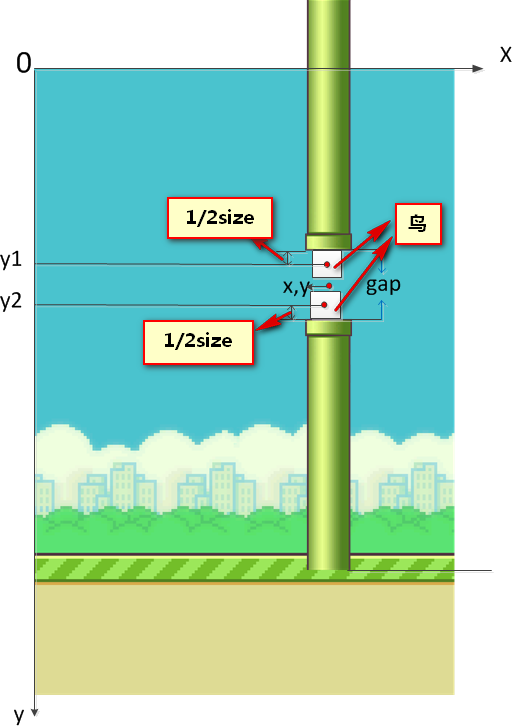


图 – 41

从图-41可以总结出，小鸟在柱子的缝隙中时y坐标计算如下：

y1=column.y-gap/2+size/2

y2=column.y+gap/2-size/2

当鸟的y坐标大于y1（y>y1）并且小于y2（y<y2）时，鸟在缝隙中。

在Bird类添加hit（Column）方法，用于判断鸟是否碰撞柱子，代码如图-42所示。

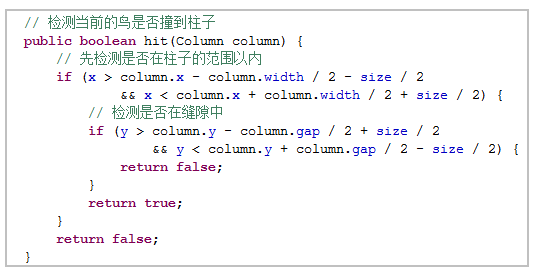


图 – 42

修改BirdFrame类的action方法，修改的代码如图-43所示：



图 – 43

完成上述代码，即完成了判断鸟的碰撞功能。

步骤二十：实现游戏的开始、结束以及重新开始

首先，把游戏分成三种状态，分别是start、running以及game\_over，start状态的界面效果如图-44所示。

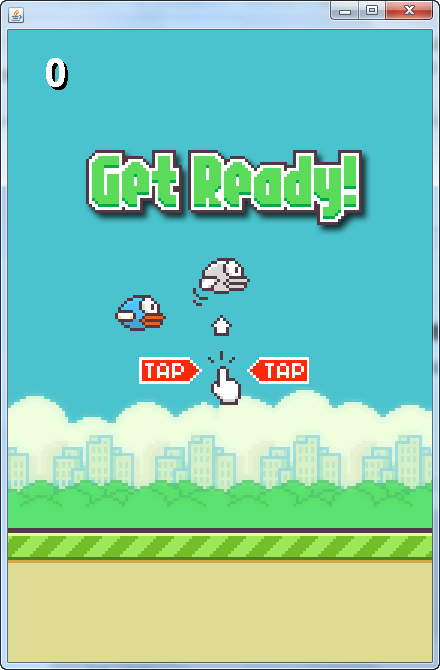


图 – 44

当游戏为start状态时，小鸟的翅膀是挥舞的、地面是移动的。即在该状态时，调用的Bird的fly方法以及Ground的step方法。

running状态的界面效果如图-45所示。

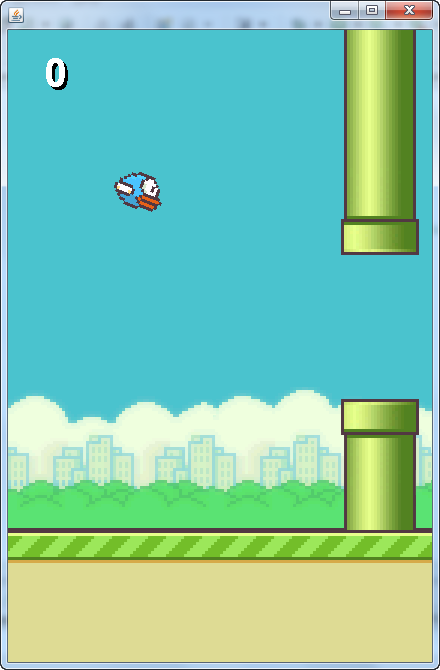


图 – 45

当游戏为running状态时，小鸟的翅膀是挥舞的、小鸟是上下移动的、地面是移动的、柱子是移动的。即在该状态时，调用的Bird的fly方法、Bird的step方法、Ground的step方法、Column的step方法。并在这个状态中进行记分以及判断是否撞到柱子。

当游戏为game\_over状态时的界面效果如图-46所示。

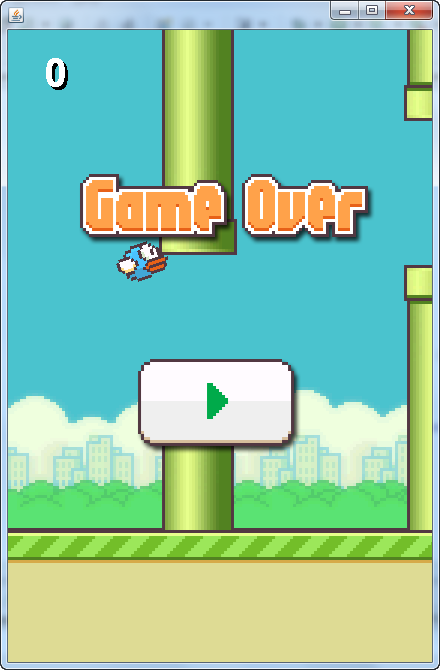


图 – 46

当游戏为game\_over状态时，只是显示的游戏结束的图片。

下面介绍一下当鼠标按下时，各个状态是如何进行转换的。

首先，当鼠标按下时如果当前游戏的状态为game\_over，则将下列信息进行重新初始化，并把状态更新为start，即实现了重新开始功能。

column1 = new Column(1);

column2 = new Column(2);

bird = new Bird();

score = 0;

state = START;

然后，当鼠标按下时如果当前游戏的状态为start，则将状态更新为running。

state = RUNNING;

最后，当鼠标按下时如果当前游戏的状态为running，则使小鸟飞扬，即调用Bird类的flappy方法。

完成游戏的开始、结束以及重新开始功能，在BirdFrame类中添加和修改的代码如下所示：

public class BirdGame extends JPanel {

Bird bird;

Column column1, column2;

Ground ground;

BufferedImage background;

//boolean gameOver;

/\*\* 游戏状态 \*/

int state;

public static final int *START* = 0;

public static final int *RUNNING* = 1;

public static final int *GAME\_OVER* = 2;

BufferedImage startImage;

BufferedImage gameOverImage;

// 分数

int score;

/\*\* 初始化 BirdGame 的属性变量 \*/

public BirdGame() throws Exception {

state = *START*;

//gameOver = false;

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

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

bird = new Bird();

column1 = new Column(1);

column2 = new Column(2);

ground = new Ground();

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

}

/\*\* "重写(修改)"paint方法实现绘制 \*/

public void paint(Graphics g) {

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

g.drawImage(column1.image, column1.x - column1.width / 2, column1.y

- column1.height / 2, null);

g.drawImage(column2.image, column2.x - column2.width / 2, column2.y

- column2.height / 2, null);

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

// 旋转(rotate)绘图坐标系，是API方法

Graphics2D g2 = (Graphics2D) g;

g2.rotate(-bird.alpha, bird.x, bird.y);

g.drawImage(bird.image, bird.x - bird.width / 2, bird.y - bird.height

/ 2, null);

g2.rotate(bird.alpha, bird.x, bird.y);

// 在paint方法中添加绘制分数的算法

Font f = new Font(Font.*SANS\_SERIF*, Font.*BOLD*, 40);

g.setFont(f);

g.drawString("" + score, 40, 60);

g.setColor(Color.*WHITE*);

g.drawString("" + score, 40 - 3, 60 - 3);

//if (gameOver) {

// g.drawImage(gameOverImage, 0, 0, null);

//}

// 在paint方法中添加显示游戏结束状态代码

switch (state) {

case *GAME\_OVER*:

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

break;

case *START*:

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

break;

}

}

// BirdGame中添加方法action()

public void action() throws Exception {

MouseListener l = new MouseAdapter() {

public void mousePressed(MouseEvent e) {

//bird.flappy();

try {

switch (state) {

case *GAME\_OVER*:

column1 = new Column(1);

column2 = new Column(2);

bird = new Bird();

score = 0;

state = *START*;

break;

case *START*:

state = *RUNNING*;

case *RUNNING*:

// 鸟向上飞扬

bird.flappy();

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

};

// 将l挂接到当前的面板（game）上

addMouseListener(l);

while (true) {

// if (!gameOver) {

// ground.step();

// column1.step();

// column2.step();

// bird.step();

// bird.fly();

// }

// if ((bird.hit(ground) || bird.hit(column1) || bird.hit(column2))) {

// gameOver = true;

// }

// // 计分逻辑

// if (bird.x == column1.x || bird.x == column2.x) {

// score++;

// }

switch (state) {

case *START*:

bird.fly();

ground.step();

break;

case *RUNNING*:

column1.step();

column2.step();

bird.step();// 上下移动

bird.fly();// 挥动翅膀

ground.step();// 地面移动

// 计分逻辑

if (bird.x == column1.x || bird.x == column2.x) {

score++;

}

// 如果鸟撞上地面游戏就结束了

if (bird.hit(ground) || bird.hit(column1) || bird.hit(column2)) {

state = *GAME\_OVER*;

}

break;

}

repaint();

Thread.*sleep*(1000 / 30);

}

}

* 完整代码

package com.tarena.bird;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.awt.image.BufferedImage;

import java.util.Random;

import javax.imageio.ImageIO;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class BirdGame extends JPanel {

Bird bird;

Column column1, column2;

Ground ground;

BufferedImage background;

/\*\* 游戏状态 \*/

int state;

public static final int START = 0;

public static final int RUNNING = 1;

public static final int GAME\_OVER = 2;

BufferedImage gameOverImage;

BufferedImage startImage;

// 分数

int score;

/\*\* 初始化 BirdGame 的属性变量 \*/

public BirdGame() throws Exception {

state = START;

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

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

score = 0;

bird = new Bird();

column1 = new Column(1);

column2 = new Column(2);

ground = new Ground();

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

}

/\*\* "重写(修改)"paint方法实现绘制 \*/

public void paint(Graphics g) {

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

g.drawImage(column1.image, column1.x - column1.width / 2, column1.y

- column1.height / 2, null);

g.drawImage(column2.image, column2.x - column2.width / 2, column2.y

- column2.height / 2, null);

// 在paint方法中添加绘制分数的算法

Font f = new Font(Font.SANS\_SERIF, Font.BOLD, 40);

g.setFont(f);

g.drawString("" + score, 40, 60);

g.setColor(Color.WHITE);

g.drawString("" + score, 40 - 3, 60 - 3);

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

// 旋转(rotate)绘图坐标系，是API方法

Graphics2D g2 = (Graphics2D) g;

g2.rotate(-bird.alpha, bird.x, bird.y);

g.drawImage(bird.image, bird.x - bird.width / 2, bird.y - bird.height

/ 2, null);

g2.rotate(bird.alpha, bird.x, bird.y);

// 在paint方法中添加显示游戏结束状态代码

switch (state) {

case GAME\_OVER:

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

break;

case START:

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

break;

}

// 添加调试的方框

// g.drawRect(bird.x-bird.size/2,

// bird.y-bird.size/2,

// bird.size, bird.size);

// g.drawRect(column1.x-column1.width/2,

// column1.y-column1.height/2,

// column1.width,

// column1.height/2-column1.gap/2);

// g.drawRect(column1.x-column1.width/2,

// column1.y+column1.gap/2,

// column1.width,

// column1.height/2-column1.gap/2);

}// paint方法的结束

// BirdGame中添加方法action()

public void action() throws Exception {

MouseListener l = new MouseAdapter() {

public void mousePressed(MouseEvent e) {

try {

switch (state) {

case GAME\_OVER:

column1 = new Column(1);

column2 = new Column(2);

bird = new Bird();

score = 0;

state = START;

break;

case START:

state = RUNNING;

case RUNNING:

// 鸟向上飞扬

bird.flappy();

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

};

// 将l挂接到当前的面板（game）上

addMouseListener(l);

while (true) {

switch (state) {

case START:

bird.fly();

ground.step();

break;

case RUNNING:

column1.step();

column2.step();

bird.step();// 上下移动

bird.fly();// 挥动翅膀

ground.step();// 地面移动

// 计分逻辑

if (bird.x == column1.x || bird.x == column2.x) {

score++;

}

// 如果鸟撞上地面游戏就结束了

if (bird.hit(ground) || bird.hit(column1) || bird.hit(column2)) {

state = GAME\_OVER;

}

break;

}

repaint();

Thread.sleep(1000 / 60);

}

}

/\*\* 启动软件的方法 \*/

public static void main(String[] args) throws Exception {

JFrame frame = new JFrame();

BirdGame game = new BirdGame();

frame.add(game);

frame.setSize(440, 670);

frame.setLocationRelativeTo(null);

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

frame.setVisible(true);

game.action();

}

}

/\*\* 地面 \*/

class Ground {

BufferedImage image;

int x, y;

int width;

int height;

public Ground() throws Exception {

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

width = image.getWidth();

height = image.getHeight();

x = 0;

y = 500;

}// 地面的构造器结束

// 地面的类体中,添加方法，地面移动一步

public void step() {

x--;

if (x == -109) {

x = 0;

}

}

}// 地面类的结束

/\*\* 柱子类型，x,y是柱子的中心点的位置 \*/

class Column {

BufferedImage image;

int x, y;

int width, height;

/\*\* 柱子中间的缝隙 \*/

int gap;

int distance;// 距离，两个柱子之间的距离

Random random = new Random();

/\*\* 构造器：初始化数据，n代表第几个柱子 \*/

public Column(int n) throws Exception {

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

width = image.getWidth();

height = image.getHeight();

gap = 144;

distance = 245;

x = 550 + (n - 1) \* distance;

y = random.nextInt(218) + 132;

}

// 在Column中添加方法 step，在action调用此方法

public void step() {

x--;

if (x == -width / 2) {

x = distance \* 2 - width / 2;

y = random.nextInt(218) + 132;

}

}

}// Column类的结束

/\*\* 鸟类型, x,y是鸟类型中心的位置 \*/

class Bird {

BufferedImage image;

int x, y;

int width, height;

int size;// 鸟的大小，用于碰撞检测

// 在Bird类中增加属性，用于计算鸟的位置

double g;// 重力加速度

double t;// 两次位置的间隔时间

double v0;// 初始上抛速度

double speed;// 是当前的上抛速度

double s;// 是经过时间t以后的位移

double alpha;// 是鸟的倾角 弧度单位

// 在Bird类中定义

// 定义一组（数组）图片，是鸟的动画帧

BufferedImage[] images;

// 是动画帧数组元素的下标位置

int index;

public Bird() throws Exception {

image = ImageIO.read(getClass().getResource("0.png"));

width = image.getWidth();

height = image.getHeight();

x = 132;

y = 280;

size = 40;

g = 4;

v0 = 20;

t = 0.25;

speed = v0;

s = 0;

alpha = 0;

// 创建数组,创建8个元素的数组

// 是8个空位置，没有图片对象，

// 8个位置的序号: 0 1 2 3 4 5 6 7

images = new BufferedImage[8];

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

// i = 0 1 2 3 4 5 6 7

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

}

index = 0;

}

// 在Bird中添加飞翔(fly)的代码

public void fly() {

index++;

image = images[(index / 12) % 8];

}

// 在Bird中添加鸟的移动方法

public void step() {

double v0 = speed;

s = v0 \* t + g \* t \* t / 2;// 计算上抛运动位移

y = y - (int) s;// 计算鸟的坐标位置

double v = v0 - g \* t;// 计算下次的速度

speed = v;

// if(y>=500){//如果到达地面，就重新抛起

// y = 280;

// speed = 35;

// }

// 调用Java API提供的反正切函数，计算倾角

alpha = Math.atan(s / 8);

}

// 在Bird中添加方法

public void flappy() {

// 重新设置初始速度，重新向上飞

speed = v0;

}

// 在鸟中添加方法 hit

// 检测当前鸟是否碰到地面ground

// 如果返回true表示发生碰撞

// 否则返回false表示没有碰撞

public boolean hit(Ground ground) {

boolean hit = y + size / 2 > ground.y;

if (hit) {

y = ground.y - size / 2;

alpha = -3.14159265358979323 / 2;

}

return hit;

}

// 检测当前的鸟是否撞到柱子

public boolean hit(Column column) {

// 先检测是否在柱子的范围以内

if (x > column.x - column.width / 2 - size / 2

&& x < column.x + column.width / 2 + size / 2) {

// 检测是否在缝隙中

if (y > column.y - column.gap / 2 + size / 2

&& y < column.y + column.gap / 2 - size / 2) {

return false;

}

return true;

}

return false;

}

}