**数据结构课程设计**

**设计题目：迷宫旅行游戏**

**目录**

1. **课程设计题目…………………………………………………1**
2. **需求分析………………………………………………………1**
3. **测试数据………………………………………………………1**
4. **概要设计………………………………………………………1**
5. **调用关系图……………………………………………………2**
6. **程序代码………………………………………………………2**
7. **测试结果………………………………………………………11**
8. **心得体会及总结………………………………………………12**

**数据结构课程设计**

**一、课程设计题目**

迷宫旅行游戏

**二、需求分析**

1、迷宫旅行游戏的功能是：

1.1 自动生成迷宫；

1.2 找出离开迷宫的路线；

1.3 用键盘控制角色走迷宫；

1.4 游戏结束可选择重新生成一局；

1. 设计思路:用Java语言实现游戏编程。用Prim算法生成迷宫，用深度优先搜索找到离开迷宫的路线，在JPanel画迷宫，然后挂载到JFrame;
2. 设计思路分析：

3.1 Java的GUI图形界面较容易实现;

3.2 JPanel() 创建具有[双缓冲](https://baike.baidu.com/item/%E5%8F%8C%E7%BC%93%E5%86%B2" \t "https://baike.baidu.com/item/_blank)和流布局的新 JPanel。因为需要经常更新角色在迷宫的位置，画面会有卡顿现象，因此用JPanel 就可以解决UI更新问题 ;

**三、测试数据**

boolean map[][]=new PMap().prim(0, 0, 20, 19, true)

PaintMap p=new PaintMap(map,new EMap(map).exitmap())

**四、概要设计**

PMap类：随机Prim算法实现用于生成迷宫

public boolean[][] prim(int startX,int startY,int widthLimit,int heightLimit,boolean haveBorder)

EMap类：DSF算法实现用于求解离开迷宫路线

算法实现方法:private void dfs(int x,int y,int c)

离开迷宫实现方法:public ArrayList<Integer> exitmap()

判断边界方法:private boolean ise(int dx,int dy)

PaintMap类：绘制迷宫，实现角色移动更新

画迷宫方法:public void paint(Graphics g)

向上移动方法:public void moveUp()

向下移动方法:public void moveDown()

向左移动方法:public void moveLeft()

向右移动方法:public void moveRight()

空格按下显示路线方法:public void PressSp()

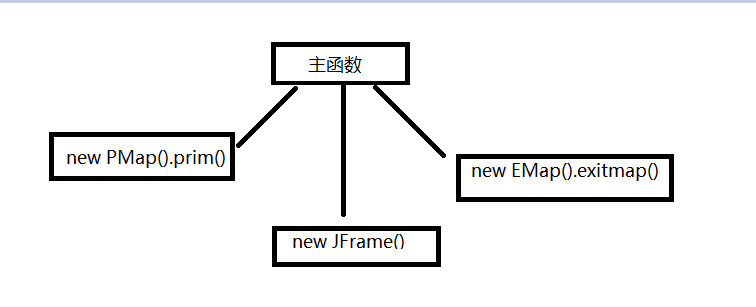
判断边界方法:private boolean IsEdge(int x,int y)

判断胜利方法:private void Win(int x,int y)

Text类：主函数类

键盘监听回调方法:public void keyPressed(KeyEvent key)

**五、调用关系图**



**六、程序代码**

package kcsj;

import java.util.ArrayList;

import javax.swing.JPanel;

@SuppressWarnings("serial")

/\*

让迷宫全都是墙.。

选一个格，作为迷宫的通路，然后把它的邻墙放入列表。

当列表里还有墙时:

从列表里随机选一个墙，如果它对面的格子不是迷宫的通路:

把墙打通，让对面的格子成为迷宫的通路。

把那个格子的邻墙加入列表。

如果对面的格子已经是通路了，那就从列表里移除这面墙。

\*/

public class PMap extends JPanel {

public boolean[][] prim(int startX,int startY,int widthLimit,int heightLimit,boolean haveBorder){

final boolean block=false,unblock=true;

if(widthLimit<1)

widthLimit=1;

if(heightLimit<1)

heightLimit=1;

if(startX<0||startX>=widthLimit)

startX=(int)Math.round(Math.random()\*(widthLimit-1));

if(startY<0||startY>=heightLimit)

startY=(int)Math.round(Math.random()\*(heightLimit-1));

if(!haveBorder) {

--widthLimit;

--heightLimit;

}

//迷宫尺寸换算成带墙尺寸

widthLimit\*=2;

heightLimit\*=2;

//迷宫起点换算成带墙起点

startX\*=2;

startY\*=2;

if(haveBorder) {

++startX;

++startY;

}

//初始化迷宫

boolean[][]mazeMap=new boolean [widthLimit+1][heightLimit+1];

for(int i=0;i<=widthLimit;i++)

for(int j=0;j<=heightLimit;j++)

mazeMap[i][j]=block;

mazeMap[0][1]=unblock;//入口

mazeMap[widthLimit][heightLimit-1]=unblock;//出口

ArrayList<Integer> blockPos = new ArrayList<Integer>();

int targetX=startX,targetY=startY;

mazeMap[targetX][targetY]=unblock;

if(targetY>1) {

blockPos.add(targetX);blockPos.add(targetY-1);blockPos.add(0);

}

if (targetX < widthLimit)

{

blockPos.add(targetX+1);blockPos.add(targetY);blockPos.add(1);

}

if (targetY < heightLimit)

{

blockPos.add(targetX);blockPos.add(targetY+1);blockPos.add(2);

}

if (targetX > 1)

{

blockPos.add(targetX-1);blockPos.add(targetY);blockPos.add(3);

}

while(!blockPos.isEmpty()) {

int blockIndex=(int)Math.round(Math.random()\*(blockPos.size()/3-1))\*3;

if(blockIndex+2<blockPos.size()) {

if(blockPos.get(blockIndex+2).equals(0)) {

targetX= blockPos.get(blockIndex);

targetY= blockPos.get(blockIndex+1)-1;

}

else if(blockPos.get(blockIndex+2).equals(1)) {

targetX= blockPos.get(blockIndex)+1;

targetY= blockPos.get(blockIndex+1);

}

else if(blockPos.get(blockIndex+2).equals(2)) {

targetX= blockPos.get(blockIndex);

targetY= blockPos.get(blockIndex+1)+1;

}

else if(blockPos.get(blockIndex+2).equals(3)) {

targetX= blockPos.get(blockIndex)-1;

targetY= blockPos.get(blockIndex+1);

}

}

if(mazeMap[targetX][targetY]==block) {

//打通墙

if(blockIndex+1<blockPos.size())

mazeMap[blockPos.get(blockIndex)][blockPos.get(blockIndex+1)]=unblock;

else

System.out.println("error");

mazeMap[targetX][targetY]=unblock;

//添加当前目标的邻墙

if (targetY > 1 && mazeMap[targetX][targetY - 1] == block && mazeMap[targetX][targetY - 2] == block)

{

blockPos.add(targetX);blockPos.add(targetY-1);blockPos.add(0);

}

if (targetX < widthLimit -1&& mazeMap[targetX + 1][targetY] == block && mazeMap[targetX + 2][targetY] == block)

{

blockPos.add(targetX+1);blockPos.add(targetY);blockPos.add(1);

}

if (targetY < heightLimit-1 && mazeMap[targetX][targetY + 1] == block && mazeMap[targetX][targetY + 2] == block)

{

blockPos.add(targetX);blockPos.add(targetY+1);blockPos.add(2);

}

if (targetX > 1 && mazeMap[targetX - 1][targetY] == block && mazeMap[targetX - 1][targetY] == block)

{

blockPos.add(targetX-1);blockPos.add(targetY);blockPos.add(3);

}

}

for(int l=blockIndex,k=0;k<3;k++) {

blockPos.remove(l);

}

}

return mazeMap;

}

}

package kcsj;

import java.util.ArrayList;

public class EMap {

private ArrayList<Integer> blockPos = new ArrayList<Integer>();

private int d[][]= {{0,-1},{1,0},{0,1},{-1,0}};

private boolean a[][];

private int width;

private int height;

private boolean fl=false;

public EMap(boolean b[][]) {

width=(b.length-1)/2;

height=(b[0].length-1)/2;

a=new boolean [b.length][b[0].length];

for(int i=0;i<b.length;i++)

for(int j=0;j<b[0].length;j++)

a[i][j]=b[i][j];

}

private void dfs(int x,int y,int c) {

if(x==(width\*2)&&y==(height\*2-1)) {

fl=true;

return ;

}

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

if(c==i)continue;

int dx=x+d[i][0];

int dy=y+d[i][1];

if(ise(dx,dy)&&a[dx][dy]) {

if(fl)break;

blockPos.add(dx);blockPos.add(dy);

a[dx][dy]=false;

dfs(dx,dy,(i+2)%4);

}

}

if(!fl) {

blockPos.remove(blockPos.size()-1);

blockPos.remove(blockPos.size()-1);

}

}

@SuppressWarnings("unused")

public ArrayList<Integer> exitmap() {

blockPos.add(0);blockPos.add(1);//初始位置

dfs(0,1,3);

return blockPos;

}

private boolean ise(int dx,int dy) {

return (0 <= dx && dx <= width\*2 && 0 <= dy && dy <= height\*2);

}

}

package kcsj;

import java.awt.Color;

import java.awt.Graphics;

import java.util.ArrayList;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

@SuppressWarnings("serial")

public class PaintMap extends JPanel{

final int unitSize =10;

private int width;

private int height;

private int startX;

private int startY;

private boolean block;

private boolean b[][];

private boolean IsDisplay;

private ArrayList<Integer> ToExit = new ArrayList<Integer>();

public PaintMap(boolean b[][],ArrayList<Integer>a) {

ToExit=a;

this.b=b;

width=b.length;

height=b[0].length;

startX=0; //初始位置

startY=1;

block=true;

IsDisplay=false;

}

public void paint(Graphics g) {

//墙的颜色

g.setColor(Color.green);

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

for(int j=0;j<height;j++)

if(!b[i][j])

g.fill3DRect(30+i\*unitSize, 30+j\*unitSize, unitSize, unitSize, true);

//出口路线颜色

if(IsDisplay) {

g.setColor(Color.red);

for(int i=0;i<ToExit.size();i+=2) {

g.fill3DRect(30+ToExit.get(i)\*unitSize, 30+ToExit.get(i+1)\*unitSize, unitSize, unitSize, true);

}

}

//控制格子颜色

g.setColor(Color.yellow);

if(IsEdge(startX, startY)) {

g.fill3DRect(30+startX\*unitSize, 30+startY\*unitSize, unitSize, unitSize, true);

}

else

g.fill3DRect(30+unitSize,30, unitSize, unitSize, true);

}

public void moveUp() {

startY-=1;

if(IsEdge(startX, startY)) {

if(!b[startX][startY]) {

block=false;

startY+=1;

}

if(block)

repaint();

else

block=true;

Win(startX,startY);

}

else

startY+=1;

}

public void moveDown() {

startY+=1;

if(IsEdge(startX, startY)) {

if(!b[startX][startY]) {

block=false;

startY-=1;

}

if(block)

repaint();

else

block=true;

Win(startX,startY);

}

else

startY-=1;

}

public void moveLeft() {

startX-=1;

if(IsEdge(startX, startY)) {

if(!b[startX][startY]) {

block=false;

startX+=1;

}

if(block)

repaint();

else

block=true;

Win(startX,startY);

}

else

startX+=1;

}

public void moveRight() {

startX+=1;

if(IsEdge(startX, startY)) {

if(!b[startX][startY]) {

block=false;

startX-=1;

}

if(block)

repaint();

else

block=true;

Win(startX,startY);

}

else

startX-=1;

}

public void PressSp() {

if(IsDisplay)

IsDisplay=false;

else

IsDisplay=true;

repaint();

}

private boolean IsEdge(int x,int y) {

return (x<width&&y<height&&x>=0&&y>=0) ;

}

private void Win(int x,int y) {

if(x==width-1&&y==height-2) {

Object[] options = {"再来一局","退出"};

int response=JOptionPane.showOptionDialog ( this, "出来了","Game Over",JOptionPane.YES\_OPTION ,JOptionPane.PLAIN\_MESSAGE, null,

options, options[0] ) ;

if (response == 0){

b=new PMap().prim(0, 0, (width-1)/2,(height-1)/2, true);

ToExit=new EMap(b).exitmap();

startX=0;

startY=1;

block=true;

IsDisplay=false;

repaint();

}

else

System.exit(0);

}

}

}

package kcsj;

import java.awt.EventQueue;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.JFrame;

@SuppressWarnings("serial")

public class Test extends JFrame implements ActionListener ,KeyListener{

boolean map[][]=new PMap().prim(10, 20, 20, 19, true);

PaintMap p=new PaintMap(map,new EMap(map).exitmap());

public Test() {

this.setTitle("Prim迷宫");

this.add(p);

this.setSize(500,500);

this.setVisible(true);

this.setLocationRelativeTo(null);

addKeyListener(this);//监听键盘

}

public static void main(String[] args) {

//Swing 不是线程安全的。

EventQueue.invokeLater(new Runnable() {

public void run() {

new Test().setVisible(true);

}

});

}

@Override

public void keyPressed(KeyEvent key) {

// TODO Auto-generated method stub

switch(key.getKeyCode()) {

case KeyEvent.VK\_UP:p.moveUp();break;

case KeyEvent.VK\_DOWN:p.moveDown();break;

case KeyEvent.VK\_LEFT:p.moveLeft();break;

case KeyEvent.VK\_RIGHT:p.moveRight();break;

case KeyEvent.VK\_SPACE:p.PressSp();break;

}

this.repaint();

}

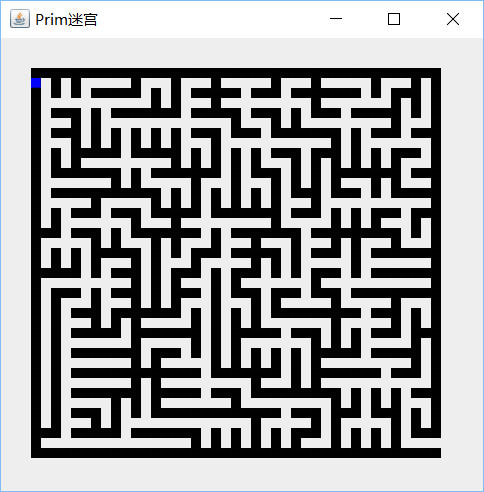
public void keyReleased(KeyEvent arg0) {}

public void keyTyped(KeyEvent arg0) {}

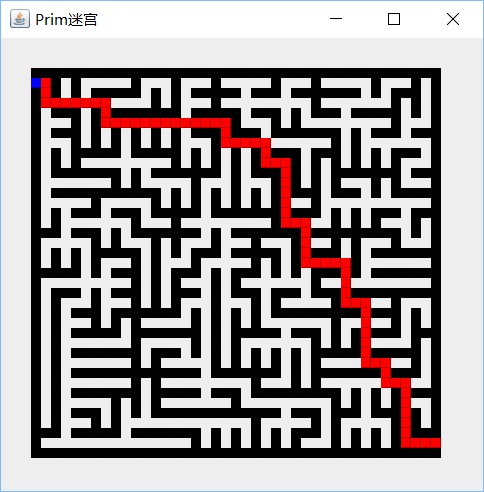
public void actionPerformed(ActionEvent arg0) {}

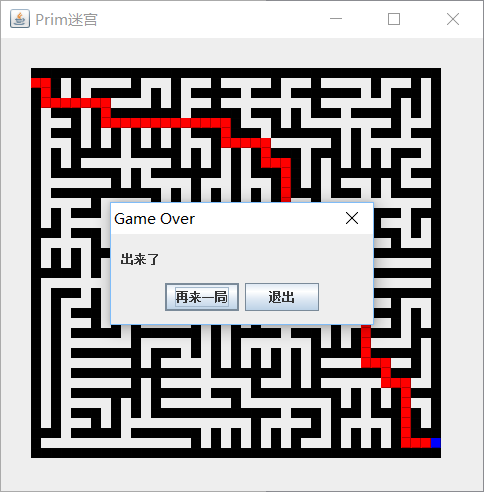
}

1. **测试结果**

 7.1游戏开始:

7.2按空格显示离开迷宫路线，再按一次空格路线消失:



7.3游戏胜利:

**八、心得体会及总结**

通过这次课程设计，更熟练的使用Java语言进行编程，对随机Prim算法和DFS算法理解更深。

在这个过程遇到不少问题，通过查找资料和断点调试基本能解决。印象比较深刻的是做这类有边框限制的程序，要检查好边界，比较容易出错。