SketchRPG 设计报告

徐伟　1110310123

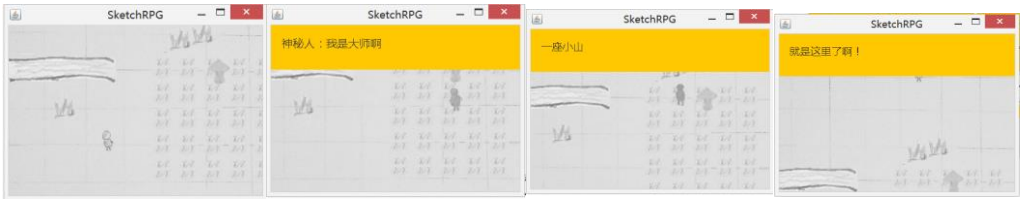
题目

SketchRPG 手绘角色扮演游戏

使用过程

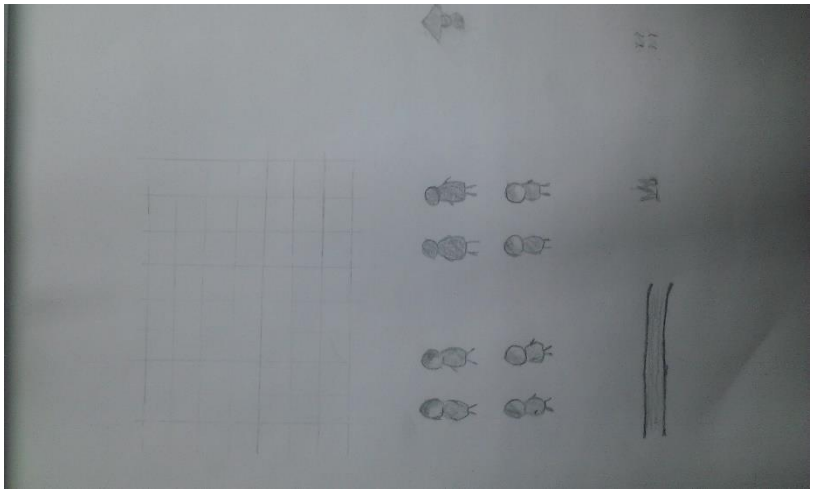
1.加载时现象：

2.使用上下左右来移动，使用 w 键来调查，其中有 NPC，有结束点，有其他对象



3.结束

4，附手绘原稿：



总体设计思想

想要做一个 RPG 类型的游戏，包含地图、对话、战斗等元素，但是由于时间和能力原因，最终只完成了地图和对话元素，战斗元素没有完成。

地图元素：首先手绘背景地图、河流、山、草、人物等元素，通过照相机将其转入到计算机中，再用 PS 处理，分别抠图处理成各个单独部分，构成基本的地图元素。然后代码中通过重写 JPanel 的 paintComponent 方法，绘制这些元素，构成地图。同时，响应方向键按下事件，改变人的位置以及地图的位置，使得人能够在地图中移动，同时当人移动到距窗口一定临界值时，同时也将地图位置移动，保证人物在一个合理的范围内能够走遍所有的地图。

对话：考虑了两种情景，一种是发现模式，一种是与 NPC 的对话模式。发现模式及在 RPG 中按下调查按键后给出当前是否发现物品的提示，这往往只是一句话；对话模式在与 NPC 对话或者一些剧情中出现，往往是几组问答或几句话。为了方便，对话模式采用读文件的方式，文件中每行为一句话，而发现模式仅仅只需要一个字符串即可。读文件的方式在文件读完后即说明对话模式结束，转回到地图模式；而发现模式显示完字符串后即说明本次对话已经结束。

综上，由于我将按钮事件绑定在主窗体上，故为每种元素建立了一个状态，分为地图状态、对话状态，同时还加入了结束状态。在地图状态下使用上下左右方向键移动任务，使用 w 键来触发转入到对话状态。

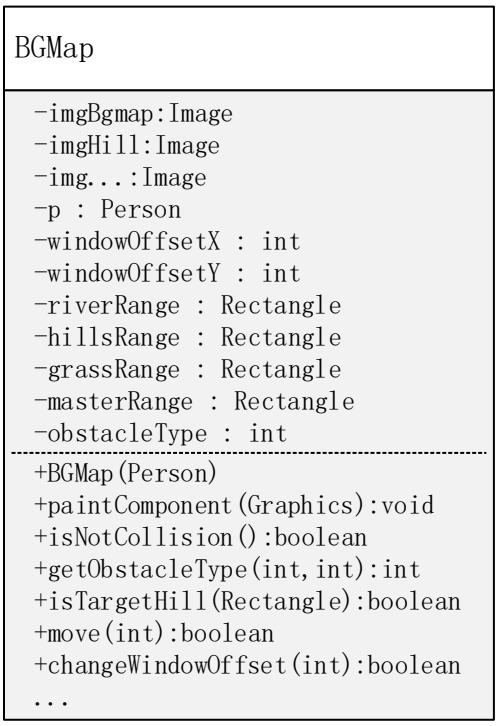
最后还要考虑在 RPG 开头和结束时加入说明文字。

详细设计

1. 类图

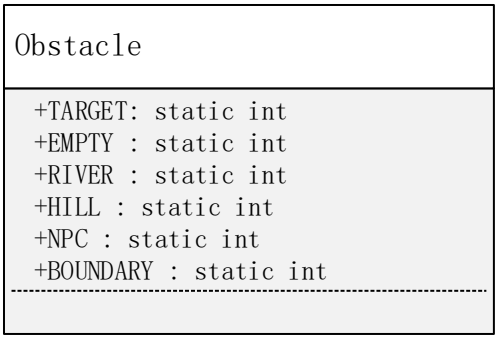
包含 5 个类，分别是：BGMap ， Obstacle ， Person ， SketchRPG ， WordsPanel 以下分别介绍各类：

BGMap：



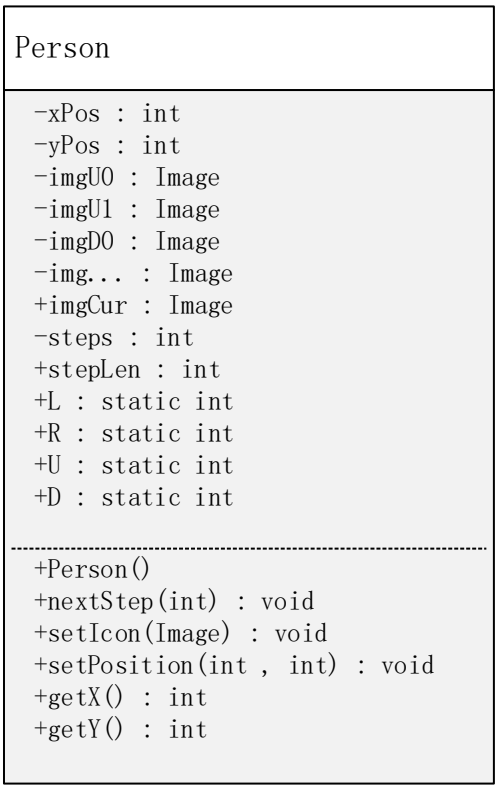
主要完成地图的绘制(人物移动、地图位置移动)、碰撞检测等。

Obstacle ：



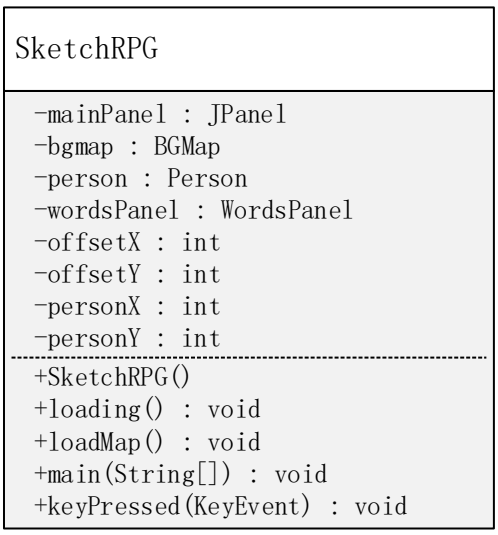
定义了障碍物类型

Person



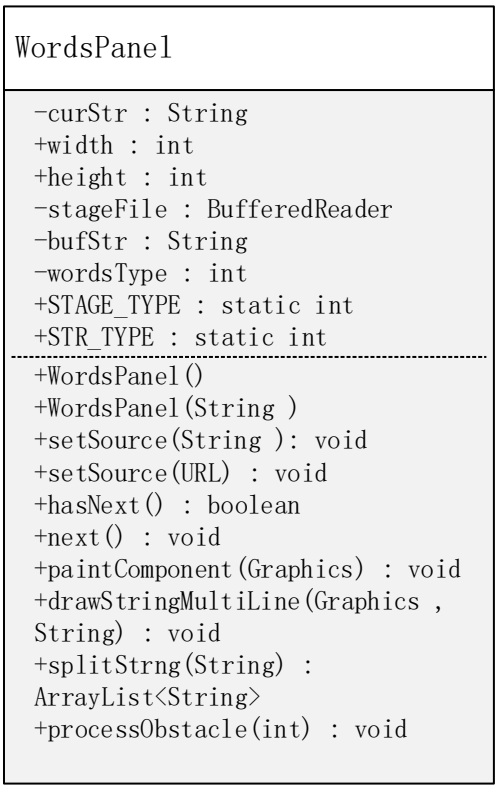
定义了人物的位置、使用的图片

SketchRPG ：



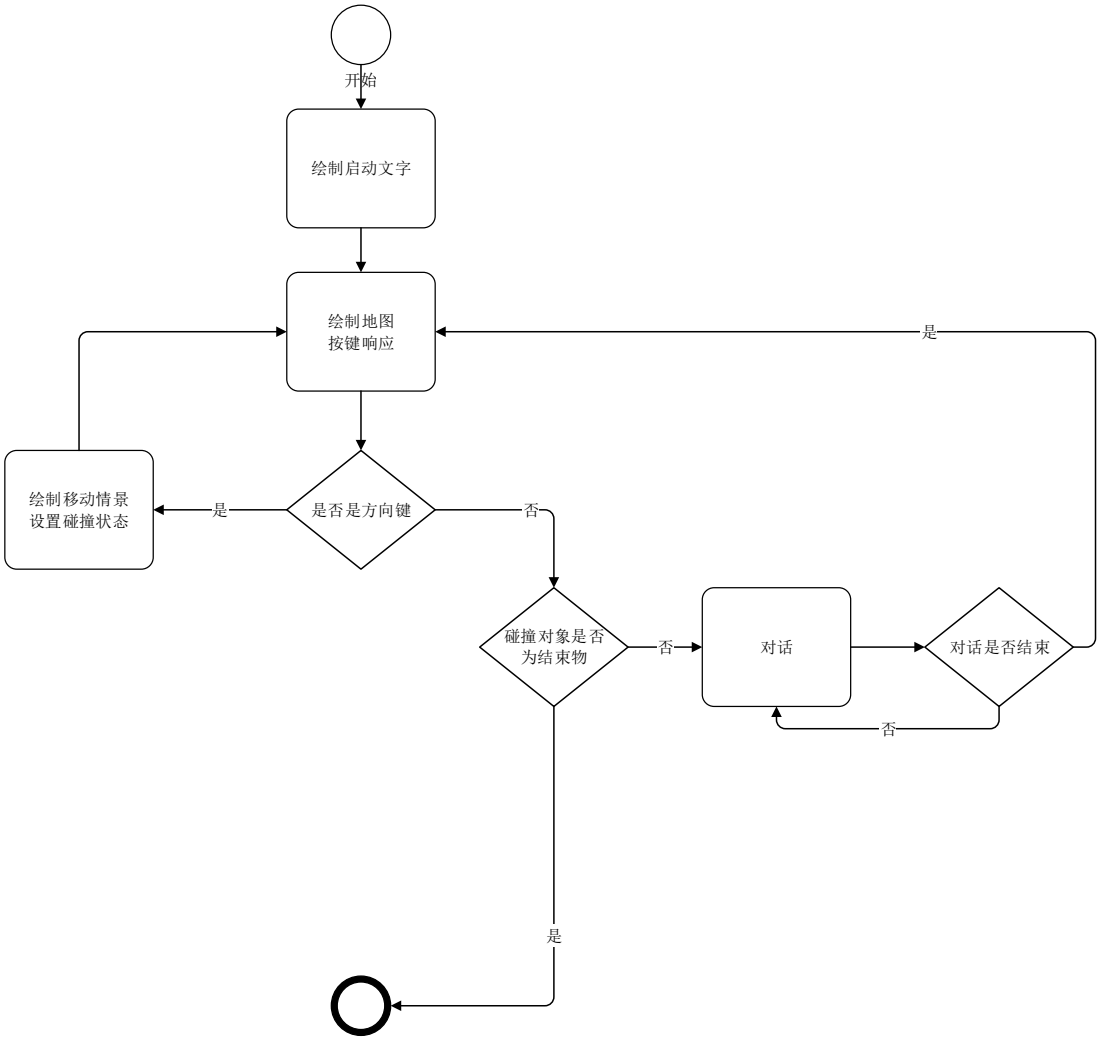
主类，控制程序流程，绑定按钮事件。

WordsPanel



主要完成对话的处理及绘制，也对障碍类型进行处理

2. 结构图



具体实现

1.BGMap.java

```
import javax.swing.* ;



import java.awt.* ;

import java.net.URL;



public class BGMap extends JPanel {

protected Person p = null ;
```

```
private Image imgBgmap = null ;

private Image imgHill = null ;

private Image imgRiver = null ;

private Image imgGrass = null ;

private Image imgMaster = null ;



public final static int RIVER_WIDTH = 200 ;

public final static int RIVER_HEIGHT = 50 ;

public final static int HILL_WIDTH = 45 ;

public final static int HILL_HEIGHT = 45 ;

public final static int GRASS_WIDTH = 45 ;

public final static int GRASS_HEIGHT = 45 ;

public final static int MASTER_HEIGHT = 45 ;

public final static int MASTER_WIDTH = 45 ;

public final static int BGMAP_WIDTH = 500 ;

public final static int BGMAP_HEIGHT = 500 ;



protected int windowOffsetX = 0 ;

protected int windowOffsetY = 0 ;



protected Rectangle riverRange = null ;

protected Rectangle [] hillsRange = null ;

protected Rectangle grassRange = null ;
```

```
protected Rectangle masterRange = null ;

//recode the obstacle

public int obstacleType = Obstacle.EMPTY ;



public BGMap(Person p){

    super(null) ;



    this.p = p ;

    windowOffsetY = SketchRPG.WINDOW_HEIGHT - BGMAP_HEIGHT ;

    //river

    riverRange = new Rectangle(-5 , 250 , RIVER_WIDTH ,
RIVER_HEIGHT) ;

    // hills position

    hillsRange = new Rectangle[5] ;

    hillsRange[0] = new Rectangle(75,320 , HILL_WIDTH , HILL_HEIGHT) ;

    hillsRange[1] = new Rectangle(280,200 , HILL_WIDTH ,
HILL_HEIGHT) ;

    hillsRange[2] = new Rectangle(320,190 , HILL_WIDTH ,
HILL_HEIGHT) ;

    hillsRange[3] = new Rectangle(175,0 , HILL_WIDTH , HILL_HEIGHT) ;
// target hill

    hillsRange[4] = new Rectangle(275,0 , HILL_WIDTH , HILL_HEIGHT) ;
// target hill

    //grass

    grassRange = new Rectangle(250,250,225,225) ;

    //master
```

```
    masterRange = new Rectangle(350 , 260 , MASTER_WIDTH ,
MASTER_HEIGHT) ;

    //images

    ClassLoader curClassLoader = this.getClass().getClassLoader();

    Toolkit defaultToolkit = Toolkit.getDefaultToolkit() ;

    URL urlMap = curClassLoader.getResource("res/img/map.png") ;

    //System.out.println(urlMap);

    imgBgmap = defaultToolkit.getImage(urlMap) ;



    URL urlRiver = curClassLoader.getResource("res/img/river.png") ;

    imgRiver = defaultToolkit.getImage(urlRiver) ;



    URL urlHill = curClassLoader.getResource("res/img/hill.png") ;

    imgHill = defaultToolkit.getImage(urlHill) ;



    URL urlGrass = curClassLoader.getResource("res/img/grass.png") ;

    imgGrass = defaultToolkit.getImage(urlGrass) ;



    URL urlMaster = curClassLoader.getResource("res/img/master.png") ;

    imgMaster = defaultToolkit.getImage(urlMaster) ;

}

public void paintComponent(Graphics g){

    super.paintComponent(g) ;

    g.drawImage(imgBgmap, 0 , 0 , this) ;
```

```
    //river
    g.drawImage(imgRiver, riverRange.x , riverRange.y , this) ;
    //hill
    for(int i = 0 ; i < hillsRange.length ; i++){
        g.drawImage(imgHill , hillsRange[i].x , hillsRange[i].y ,
this) ;
    }
    //grass
    for(int x = 0 ; x < grassRange.width ; x += GRASS_WIDTH){
        for(int y = 0 ; y < grassRange.height ; y += GRASS_HEIGHT){
            g.drawImage(imgGrass ,  x + grassRange.x,  y +
grassRange.y , this) ;
        }
    }
    //master
    g.drawImage(imgMaster , masterRange.x , masterRange.y , this) ;
    //people
    g.drawImage(p.imgCur , p.getX(), p.getY(),this) ;
}
// to detech if has collision , and set the obstacle type
public boolean isNotCollision(int tmpX , int tmpY){

    //to test the bounds of map
    if(tmpX <= 0 || tmpX >= BGMAP_WIDTH - Person.PERSON_WIDTH){
        obstacleType = Obstacle.BOUNDARY ;
```

```
        return false ;

    }

    if(tmpY <= 0 || tmpY >= BGMAP_HEIGHT - Person.PERSON_HEIGHT){

        obstacleType = Obstacle.BOUNDARY ;

        return false ;

    }

    //to test the objects of hills and river

    Rectangle newPos = new Rectangle(tmpX , tmpY , Person.PERSON_WIDTH
- 20, Person.PERSON_HEIGHT - 20) ;

    if(newPos.intersects(riverRange)){

        obstacleType = Obstacle.RIVER ;

        return false ;

    }

    if(newPos.intersects(masterRange)){

        obstacleType = Obstacle.NPC ;

        return false ;

    }

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

        if(newPos.intersects(hillsRange[i])){

            if(isTargetHill(hillsRange[i])){

                obstacleType = Obstacle.TARGET ;

            }

            else{

                obstacleType = Obstacle.HILL ;
```

```
            }

            return false ;

        }

    }

    obstacleType = Obstacle.EMPTY ;

    return true ;

}

public int getObstacleType(){

    return obstacleType ;

}

public boolean isTargetHill(Rectangle hill){

    if(hill == hillsRange[3] || hill == hillsRange[4]){

        return true ;

    }

    else{

        return false ;

    }

}

public boolean move(int direct){

    int tmpX = p.getX() ;

    int tmpY = p.getY() ;

    if(direct == Person.L){

        tmpX -= p.stepLen ;

    }
```

```
    else if(direct == Person.R){

        tmpX += p.stepLen ;

    }

    else if(direct == Person.U){

        tmpY -= p.stepLen ;

    }

    else if(direct == Person.D){

        tmpY += p.stepLen ;

    }

    boolean isNotCol = isNotCollision(tmpX , tmpY) ;

    if(isNotCol){

        p.setPosition(tmpX, tmpY);

        p.nextStep(direct) ;

        return true ;

    }

    else{

        return false ;

    }

}

public int getWindowOffsetX(){

    return windowOffsetX ;

}

public int getWindowOffsetY(){

    return windowOffsetY ;
```

```
}

public boolean changeWindowOffset(int direct){

    int pWindowX = p.getX() + windowOffsetX ;

    int pWindowY = p.getY() + windowOffsetY ;

    if(direct == Person.R){

        if(windowOffsetX + BGMap.BGMAP_WIDTH <=
SketchRPG.WINDOW_WIDTH){

            windowOffsetX = SketchRPG.WINDOW_WIDTH -
BGMap.BGMAP_WIDTH ;

            return false ;

        }

        if(pWindowX > 325){

            windowOffsetX -= p.stepLen ;

            return true ;

        }

    }

    else if(direct == Person.L){

        if(windowOffsetX >= 0){

            windowOffsetX = 0 ;

            return false ;

        }

        if(pWindowX < 125){

            windowOffsetX += p.stepLen ;

            return true ;

        }
```

```
    }

    else if(direct == Person.U){

        if(windowOffsetY >= 0){

            windowOffsetY = 0 ;

            return false ;

        }

        if(pWindowY < 100){

            windowOffsetY += p.stepLen ;

            return true ;

        }

    }

    else if(direct == Person.D){

        if(windowOffsetY + BGMap.BGMAP_HEIGHT <=
SketchRPG.WINDOW_HEIGHT){

            windowOffsetY = SketchRPG.WINDOW_HEIGHT -
BGMap.BGMAP_HEIGHT  ;

            return false ;

        }

        if(pWindowY >= 200){

            windowOffsetY -= p.stepLen ;

            return true ;

        }

    }

    //otherwise , don't change

    return false ;
```

```
}

}
```

2.Obstacle.java

```
public class Obstacle {

public final static int TARGET = 0 ;

public final static int EMPTY = 1 ;

public final static int RIVER = 2 ;

public final static int HILL = 3 ;

public final static int NPC = 4 ;

public final static int BOUNDARY = 5 ;









}
```

3.Person.java

```
import javax.swing.* ;



import java.awt.* ;

import java.net.URL;



public class Person {



public static final int PERSON_WIDTH = 45 ;

public static final int PERSON_HEIGHT = 45 ;
```

```
// the postion at the map !!
protected int xPos = 0 ;
protected int yPos = 0 ;

private Image imgU0 = null ;
private Image imgU1 = null ;
private Image imgD0 = null ;
private Image imgD1 = null ;
private Image imgL0 = null ;
private Image imgL1 = null ;
private Image imgR0 = null ;
private Image imgR1 = null ;

public Image imgCur = null ;
private int steps = 0 ;
public int stepLen = 10 ;

public final static int L = 0 ;
public final static int R = 1 ;
public final static int U = 2 ;
public final static int D = 3 ;
public Person(){
    super() ;
```

```
    ClassLoader curClassLoader = this.getClass().getClassLoader();

    Toolkit defaultToolkit = Toolkit.getDefaultToolkit();



    URL urlU0 = curClassLoader.getResource("res/img/U0.png") ;

    imgU0 = defaultToolkit.getImage(urlU0) ;



    URL urlU1 = curClassLoader.getResource("res/img/U1.png") ;

    imgU1 = defaultToolkit.getImage(urlU1) ;



    URL urlD0 = curClassLoader.getResource("res/img/D0.png") ;

    imgD0 = defaultToolkit.getImage(urlD0) ;



    URL urlD1 = curClassLoader.getResource("res/img/D1.png") ;

    imgD1 = defaultToolkit.getImage(urlD1) ;



    URL urlR0 = curClassLoader.getResource("res/img/R0.png") ;

    imgR0 = defaultToolkit.getImage(urlR0) ;



    URL urlR1 = curClassLoader.getResource("res/img/R1.png") ;

    imgR1 = defaultToolkit.getImage(urlR1) ;



    URL urlL0 = curClassLoader.getResource("res/img/L0.png") ;

    imgL0 = defaultToolkit.getImage(urlL0) ;
```

```
    URL urlL1 = curClassLoader.getResource("res/img/L1.png") ;

    imgL1 = defaultToolkit.getImage(urlL1) ;



    setIcon(imgL0) ;

}

public void nextStep(int d ){

    steps = (++ steps ) % 2 ;

    if(d == D){

        if(steps % 2 == 0) setIcon(imgD0) ;

        else setIcon(imgD1) ;

    }

    else if(d == U){

        if(steps % 2 == 0) setIcon(imgU0) ;

        else setIcon(imgU1) ;

    }

    else if(d == R){

        if(steps%2 == 0) setIcon(imgR0) ;

        else setIcon(imgR1) ;

    }

    else{

        if(steps%2 == 0) setIcon(imgL0) ;

        else setIcon(imgL1) ;

    }

}
```

```
public void setIcon(Image cur){

    imgCur = cur ;

}

public void setPosition(int x , int y){

    xPos = x ;

    yPos = y ;

}

public int getX(){

    return xPos ;

}

public int getY(){

    return yPos ;

}

}
```

4.SketchPRG.java

```
import javax.swing.* ;



import java.awt.* ;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;



public class SketchRPG extends JFrame implements KeyListener{

public final static int WINDOW_WIDTH = 450 ;

public final static int WINDOW_HEIGHT = 300  ;
```

```
private JPanel mainPanel = null ;

protected BGMap bgmap = null ;

protected Person person = null ;

protected WordsPanel wordsPanel = null ;



//map position

private int offsetX = 0 ;

private int offsetY = 0 ;

//person position at the map

private int personX = 0 ;

private int personY = 0 ;



public final static int GAME_STATE_MAP = 0 ;

public final static int GAME_STATE_TALK = 1 ;

public final static int GAME_STATE_OVER = 2 ;



private int gameState = GAME_STATE_MAP ;

public SketchRPG(){

    super("SketchRPG") ;

    mainPanel = new JPanel() ;

    mainPanel.setLayout(null);

    mainPanel.setPreferredSize(new
Dimension(WINDOW_WIDTH,WINDOW_HEIGHT));
```

```
    setContentPane(mainPanel) ;

    setBounds(400,300,WINDOW_WIDTH , WINDOW_HEIGHT) ;

    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE) ;

    pack() ;

    setVisible(true) ;

    loading() ;

    loadMap() ;

}

public void loading(){

    wordsPanel = new WordsPanel("SketchRPG\n 描述了一个少年寻找更广阔世界
入口的故事。\n"

                                    + "虽然游戏情节简单，但真实世界中却又是
何其的困难。\n"

                                    + "不断努力，才能看到更加开阔的世界吧\n"

                                    + "by 11103101 徐伟\n") ;

    setContentPane(wordsPanel) ;

    try{

        Thread.sleep(3000) ;

    }

    catch(Exception e){

        e.printStackTrace() ;

    }

    wordsPanel.setSource("使用方向键来行走，使用'w'键来触发情景\n 有问题麻烦
老师联系\n18004515446") ;

    wordsPanel.next() ;
```

```
    setContentPane(wordsPanel) ;

    try{

        Thread.sleep(3000) ;

    }

    catch(Exception e){

        e.printStackTrace() ;

    }

    setContentPane(mainPanel) ;

    wordsPanel = null ;

}

public void loadMap(){

    person = new Person() ;

    bgmap = new BGMap(person) ;

    wordsPanel = new WordsPanel() ;

    //set init position

    personY = 420 ;

    personX = 80 ;

    person.setPosition(personX , personY );

    mainPanel.removeAll();

    mainPanel.add(bgmap) ;

    offsetY = bgmap.getWindowOffsetY() ;

    bgmap.setBounds(offsetX , offsetY , BGMap.BGMAP_WIDTH ,
BGMap.BGMAP_HEIGHT);

    repaint() ;
```

```
    addKeyListener(this) ;
}
public static void main(String[] argv){
    SketchRPG xx = new SketchRPG() ;
}


@Override
public void keyTyped(KeyEvent e) {
    // TODO Auto-generated method stub


}
@Override
public void keyPressed(KeyEvent e) {
    // TODO Auto-generated method stub
    int keyCode = e.getKeyCode();
    char keyChar = e.getKeyChar() ;
    if(gameState == GAME_STATE_MAP){
        int direct ;
        switch(keyCode){
            case KeyEvent.VK_UP :
                direct = Person.U ;
                break ;
            case KeyEvent.VK_DOWN :
                direct = Person.D ;
```

```
                break ;

            case KeyEvent.VK_LEFT :

                direct = Person.L ;

                break ;

            case KeyEvent.VK_RIGHT :

                direct = Person.R ;

                break ;

            default :

                direct = -1 ;

        }

        if(direct != -1){

            bgmap.move(direct) ;

            //decide if moving the bgmap

            if(bgmap.changeWindowOffset(direct)){

                bgmap.setLocation(bgmap.getWindowOffsetX() ,
bgmap.getWindowOffsetY()) ;

            }

        }

        else if(keyChar == 'W' || keyChar == 'w'){

            if(wordsPanel == null) return ;

            wordsPanel.processObstacle(bgmap.getObstacleType());

            wordsPanel.next();

            mainPanel.add(wordsPanel) ;


wordsPanel.setBounds(0,0,wordsPanel.width,wordsPanel.height);
```

```
            mainPanel.setComponentZOrder(wordsPanel,0) ; // make it
visible

            gameState = GAME_STATE_TALK ;

            repaint() ;

        }

    }

    else if(gameState == GAME_STATE_TALK){

        if(wordsPanel.hasNext()){

            wordsPanel.next();

        }

        else{

            if(bgmap.getObstacleType() == Obstacle.TARGET){

                gameState = GAME_STATE_OVER ;

                mainPanel.removeAll() ;

                wordsPanel = null ;

                wordsPanel = new WordsPanel("OVER\nby 11103101 徐伟
") ;

                mainPanel.add(wordsPanel) ;

                wordsPanel.setBounds(0,0,WINDOW_WIDTH,WINDOW_HEIGHT);

                repaint() ;

            }

            else{

                gameState = GAME_STATE_MAP ;

                mainPanel.remove(wordsPanel) ;

            }
```

```
        }

    }

    repaint() ;

}

@Override

public void keyReleased(KeyEvent e) {

    // TODO Auto-generated method stub



}

}
```

5.WordsPanel.java

```
import javax.swing.* ;



import java.awt.* ;

import java.net.URL;

import java.util.ArrayList;

import java.io.* ;



public class WordsPanel extends JPanel {

String curStr = null ;

public int width = 0 ;

public int height = 0 ;

private BufferedReader stageFile = null ;

private String bufStr = null ;
```

```
private int wordsType = -1 ;

public final static int STAGE_TYPE = 0 ;

public final static int STR_TYPE = 1 ;



public WordsPanel(){

    super() ;

    width = SketchRPG.WINDOW_WIDTH ;

    height = 80 ;

}

public WordsPanel(String str){

    super() ;

    width = SketchRPG.WINDOW_WIDTH ;

    height = SketchRPG.WINDOW_HEIGHT ;

    curStr = str ;

    setPreferredSize(new Dimension(width , height)) ;

    repaint() ;

}

public void setSource(String str){

    this.bufStr = str ;

    wordsType = STR_TYPE ;

}



public void setSource(URL url){

    try {
```

```
        //System.out.println(url.toURI()) ;

        stageFile = new BufferedReader(new FileReader(new
File(url.toURI()))) ;

        wordsType = STAGE_TYPE ;

        bufStr = stageFile.readLine();

        if(bufStr == null)

            stageFile.close();

    } catch (Exception e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    }

}

public boolean hasNext(){

    if( bufStr != null ) return true ;

    else return false ;

}

public void next(){

    curStr = bufStr ;

    if(wordsType == STR_TYPE){

        bufStr = null ;

    }

    else{

        try{

            bufStr = stageFile.readLine();
```

```
            if(bufStr == null){

                stageFile.close();

            }

        }

        catch(Exception e){

            e.printStackTrace() ;

        }

    }

    repaint() ;

}

public void paintComponent(Graphics g){

    super.paintComponent(g);

    setBackground(Color.orange);

    Font font = new Font("Microsoft Yahei" , Font.PLAIN , 20) ;

    g.setFont(font);

    if(curStr != null)

        drawStringMultiLine(g,curStr) ;

}

public void drawStringMultiLine(Graphics g , String str){

    ArrayList<String> rstList = splitStr(str) ;

    int lineHeight = height / (rstList.size() + 1)  ;

    int baseHeight = 0 ;

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

        baseHeight += lineHeight ;
```

```
        g.drawString(rstList.get(i), 20, baseHeight);

    }

}

public ArrayList splitStr(String str){

    ArrayList<String> rstList = new ArrayList<String>() ;

    int startPos = 0 ;

    int maxLen = Math.min(15, str.length());

    while(startPos < str.length()){

        int subLen = maxLen ;

        int endPos = Math.min(startPos + subLen, str.length()) ;

        String subStr = str.substring(startPos , endPos) ;

        //detect if has the char '\n'

        int LFPos = subStr.indexOf('\n') ;

        if(LFPos != -1){

            subLen = LFPos ;

            endPos = Math.min(startPos + subLen, str.length()) ;

            subStr = str.substring(startPos , endPos ) ;

            subLen += 1 ; // skip '\n' for the next time

        }

        //System.out.println(subStr) ;

        rstList.add(subStr) ;

        startPos += subLen ;

    }

    return rstList ;
```

```
}

public void processObstacle(int obstacle){

    if(obstacle == Obstacle.BOUNDARY){

        setSource("地图边界") ;

    }

    else if(obstacle == Obstacle.EMPTY){

        setSource("没有发现什么") ;

    }

    else if(obstacle == Obstacle.HILL){

        setSource("一座小山") ;

    }

    else if(obstacle == Obstacle.RIVER){

        setSource("一条小河") ;

    }

    else if(obstacle == Obstacle.NPC){

        URL urlNPCStage =
this.getClass().getClassLoader().getResource("res/stage/npc.dat") ;

        setSource(urlNPCStage) ;

    }

    else if(obstacle == Obstacle.TARGET){

        URL urlT =
this.getClass().getClassLoader().getResource("res/stage/target.dat") ;

        setSource(urlT) ;

    }

    else{
```

```
        setSource("不明物体出现！") ;

    }

}

}
```