

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Programming Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 2nd | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering Class | | | | | **Class:** | 2017 |
|  |  | | | | |  |  |
| **Student name:** | Song Xingjian | | **Student No.:** | | 222018321062006 | | |
|  |  | |  | |  | | |
| **Teacher:** | Wang Xiaomeng | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp 7 GUI | | |
| Time | 2019.12.18 | Type | □Verification □Design □Synthetical |
| **1. Objective**  Through this training, the students should master GUI design using JavaFX and event handling.  **2. Requirement**  Programming the following exercise  **Problem 1:** Sokoban Game.    **3. Content and design of experiment(main content, operations, algorithm description or code of program)**  **Problem 1:**  **Solutions:**   1. 首先创建Path类，用来存放图片素材的路径，方便调用和修改。其中所有变量都设置为public静态变量，不可修改； 2. 创建MainProgram类，作为游戏的入口及主页面，包含“开始游戏”和“编辑地图”两个按钮。其中较为复杂的是Button控件的调用，以及按钮事件的动态反映； 3. 创建MapEdit类，用作新建地图。菜单栏含有两个选项，分别可以做到重置地图和保存当前地图； 4. 重置地图后页面的四面边界设置为stone属性，通过键盘方向键实现小人的移动，通过多次点击空格键实现石头、箱子、目标点的放置与切换； 5. 保存地图后将画面中的内容顺序读入，并将其坐标和属性写入文本文档中； 6. 返回主页面点击开始游戏，弹出开始游戏的页面，在菜单栏点击加载地图，找到刚才储存的地图文件，然后将其打开。地图读入是保存的逆操作，将文本文档中的坐标和属性信息转换到页面，从而完成地图的加载； 7. 游戏时通过方向键实现小人上下左右的移动，通过判定路径上的元素属性完成推箱子的动作，最后检验每个目标点是否均非空，来判定游戏是否结束。   **Codes:**   * **图片路径类Path的创建：**   **package** box;  **public** **class** Paths {  **public** **final** **static** String ***MAIN*** = "file:E:/Java/实验课/Exp7/main.png";  **public** **final** **static** String ***BOX*** = "file:E:/Java/实验课/Exp7/box.png";  **public** **final** **static** String ***STONE*** = "file:E:/Java/实验课/Exp7/stone.png";  **public** **final** **static** String ***MAN*** = "file:E:/Java/实验课/Exp7/man.png";  }   * **主函数及开始页面：**   **package** box;  **import** javafx.application.Application;  **import** javafx.geometry.Pos;  **import** javafx.scene.Scene;  **import** javafx.scene.control.Button;  **import** javafx.scene.image.Image;  **import** javafx.scene.image.ImageView;  **import** javafx.scene.layout.VBox;  **import** javafx.stage.Stage;  **public** **class** MainProgram **extends** Application {  @Override // 重写  **public** **void** start(Stage primaryStage) **throws** Exception {  VBox root = **new** VBox();  // 放置图片  ImageView img = **new** ImageView(Paths.***MAIN***);  img.setFitWidth(400);  img.setFitHeight(200);  // 开始游戏按钮  Button playBtn = **new** Button("开始游戏");  playBtn.setPrefWidth(400);  playBtn.setPrefHeight(100);  playBtn.setOnAction(e -> {  **new** PlayStage().show();  });  // 编辑地图按钮  Button editBtn = **new** Button("编辑地图");  editBtn.setPrefWidth(400);  editBtn.setPrefHeight(100);  editBtn.setOnAction(e -> {  **new** MapEdit().show();  });  // 把图片和两个按钮放在VBox里面  root.getChildren().addAll(img, playBtn, editBtn);  root.setAlignment(Pos.***CENTER***);  // 创建Scene  Scene scene = **new** Scene(root, 400, 400);  // 设置Stage相关参数  primaryStage.setScene(scene);  primaryStage.setTitle("推箱子");// 题目  primaryStage.getIcons().add(**new** Image(Paths.***BOX***));// 图标  primaryStage.setResizable(**false**);  primaryStage.show();  }  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  *launch*(args);  }  }   * **编辑地图页面：**   **package** box;  **import** java.io.File;  **import** java.io.FileNotFoundException;  **import** java.io.PrintWriter;  **import** javafx.scene.Node;  **import** javafx.scene.Scene;  **import** javafx.scene.control.Menu;  **import** javafx.scene.control.MenuBar;  **import** javafx.scene.control.MenuItem;  **import** javafx.scene.image.Image;  **import** javafx.scene.image.ImageView;  **import** javafx.scene.layout.Pane;  **import** javafx.scene.layout.VBox;  **import** javafx.scene.shape.Circle;  **import** javafx.scene.paint.Color;  **import** javafx.stage.FileChooser;  **import** javafx.stage.Stage;  **public** **class** MapEdit **extends** Stage {  **int** mapw = 32 \* 20;  **int** maph = 32 \* 20;  Node[][] elements = **new** Node[20][20];  **int** index = 0;  @SuppressWarnings("deprecation")  **public** MapEdit() {  VBox root = **new** VBox();  Pane p = **new** Pane();  // 创建人，并将其放在画布中间  ImageView man = **new** ImageView(Paths.***MAN***);  man.setX(32 \* 10);  man.setY(32 \* 10);  // 创建菜单栏  MenuBar menuBar = **new** MenuBar();  Menu menu1 = **new** Menu("菜单"); // 将菜单添加到菜单栏  menuBar.getMenus().add(menu1);  // 菜单中的“保存地图”按钮  MenuItem menu11 = **new** MenuItem("保存地图");  menu1.getItems().add(menu11); // 将“保存地图”添加到菜单  menu11.setOnAction(e -> {  FileChooser directoryChooser = **new** FileChooser();  File file = directoryChooser.showSaveDialog(**this**);  **try** {  PrintWriter pw = **new** PrintWriter(file);  **for** (**int** i = 0; i < elements.length; i++) {  **for** (**int** j = 0; j < elements[i].length; j++) {  **if** (elements[i][j] != **null**) { // 如果该位置不为null  **if** (elements[i][j] **instanceof** Circle) { // 如果这个位置属于circle类  // output i,j,circle  pw.print(i + "," + j + ",circle\n");  } **else** { // 如果这个位置是一个stone  ImageView img = (ImageView) elements[i][j];  **if** (img.getImage().~~impl\_getUrl~~().equals(Paths.***STONE***)) {  // output i,j,stone;  pw.print(i + "," + j + ",stone\n");  } **else** { // 如果这个位置是一个box  // output i,j,box;  System.***out***.println("box");  pw.print(i + "," + j + ",box\n");  }  }  } **else** {  pw.print(i + "," + j + ",null\n");  }  }  }  pw.close();  } **catch** (FileNotFoundException e1) {  e1.printStackTrace();  }  });  // 菜单中的“重置地图”按钮  MenuItem menu12 = **new** MenuItem("重置地图");  menu1.getItems().add(menu12);  menu12.setOnAction(e -> {  emptyMap();  p.getChildren().clear();  **for** (**int** i = 0; i < elements.length; i++) {  **for** (**int** j = 0; j < elements[i].length; j++) {  **if** (elements[i][j] != **null**) {  p.getChildren().add(elements[i][j]);  }  }  }  p.getChildren().add(man);  });  root.getChildren().addAll(menuBar, p);  Scene scene = **new** Scene(root, mapw, maph + 32);  scene.setOnKeyPressed(e -> {  **switch** (e.getCode()) {  **case** ***UP***: {  index = 0;  man.setY(man.getY() - 32);  }  **break**;  **case** ***DOWN***: {  index = 0;  man.setY(man.getY() + 32);  }  **break**;  **case** ***LEFT***: {  index = 0;  man.setX(man.getX() - 32);  }  **break**;  **case** ***RIGHT***: {  index = 0;  man.setX(man.getX() + 32);  }  **break**;  **case** ***SPACE***: {  Node item = **null**;  // 多次点击空格切换物品  **switch** ((**int**) (index++ % 3)) {  // 第一次是石头  **case** 0: {  ImageView img = **new** ImageView(Paths.***STONE***);  img.setX(man.getX());  img.setY(man.getY());  item = img;  }  ;  **break**;  // 第二次是箱子  **case** 1: {  // BOX  ImageView img = **new** ImageView(Paths.***BOX***);  img.setX(man.getX());  img.setY(man.getY());  item = img;  }  ;  **break**;  // 第三次是终点（圆形）  **case** 2: {  // Circle  Circle circle = **new** Circle();  // 注意这里是圆心的位置，所以纵横坐标都要加16  circle.setCenterX(man.getX() + 16);  circle.setCenterY(man.getY() + 16);  circle.setRadius(5.0f);  circle.setFill(Color.***RED***);  item = circle;  }  ;  **break**;  }  **int** manx = (**int**) man.getX() / 32;  **int** many = (**int**) man.getY() / 32;  **if** (elements[manx][many] != **null**) {  p.getChildren().remove(elements[manx][many]);  }  elements[manx][many] = item;  p.getChildren().add(item);  }  ;  **break**;  **default**:  **break**;  }  });  // 设置Stage相关参数  **this**.setScene(scene);  **this**.setTitle("地图编辑"); // 题目  **this**.getIcons().add(**new** Image(Paths.***BOX***)); // 图标  **this**.setResizable(**false**);  }  // 创建一个空地图  **public** **void** emptyMap() {  **for** (**int** i = 0; i < elements.length; i++) {  **for** (**int** j = 0; j < elements[i].length; j++) {  // 四边为stone，其他全为null  **if** (i == 0 || j == 0 || i == elements.length - 1 || j == elements[i].length - 1) {  ImageView stone = **new** ImageView(Paths.***STONE***);  stone.setX(i \* 32);  stone.setY(j \* 32);  elements[i][j] = stone;  } **else** {  elements[i][j] = **null**;  }  }  }  }  }   * **开始游戏页面：**   **package** box;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.IOException;  **import** java.util.ArrayList;  **import** java.util.List;  **import** java.util.Scanner;  **import** javafx.scene.Node;  **import** javafx.scene.Scene;  **import** javafx.scene.control.Alert;  **import** javafx.scene.control.Alert.AlertType;  **import** javafx.scene.control.Menu;  **import** javafx.scene.control.MenuBar;  **import** javafx.scene.control.MenuItem;  **import** javafx.scene.image.Image;  **import** javafx.scene.image.ImageView;  **import** javafx.scene.layout.Pane;  **import** javafx.scene.layout.VBox;  **import** javafx.scene.paint.Color;  **import** javafx.scene.shape.Circle;  **import** javafx.stage.FileChooser;  **import** javafx.stage.Stage;  **public** **class** PlayStage **extends** Stage {  **int** mapw = 32 \* 20;  **int** maph = 32 \* 20;  Node[][] elements = **new** Node[20][20]; // 地图格子  List<Circle> circleList = **new** ArrayList<Circle>();  @SuppressWarnings({ "incomplete-switch", "deprecation" })  **public** PlayStage() {  VBox root = **new** VBox();  Pane p = **new** Pane();  // 创建人  ImageView man = **new** ImageView(Paths.***MAN***);  // 创建菜单栏  MenuBar menuBar = **new** MenuBar();  Menu menu1 = **new** Menu("菜单"); // 将菜单添加到菜单栏  menuBar.getMenus().add(menu1);  // 菜单中的“加载地图”按钮  MenuItem menu11 = **new** MenuItem("加载地图");  menu1.getItems().add(menu11);  menu11.setOnAction(e -> {  // 打开文件路径  FileChooser directoryChooser = **new** FileChooser();  File file = directoryChooser.showOpenDialog(**this**);  **try** {  Scanner input = **new** Scanner(**new** FileInputStream(file));  p.getChildren().clear(); // 清空当前窗口  **while** (input.hasNext()) { // 逐行读取数据，直到没有数据  String[] items = input.next().split(","); // 用“，”号分割字符串，前两位存入坐标，第三位进行switch判断  **int** i = Integer.*parseInt*(items[0]);  **int** j = Integer.*parseInt*(items[1]);  **switch** (items[2]) {  **case** "circle": { // 如果是circle，则画一个半径为5的圆，但是element棋盘数组中依然为null  Circle c = **new** Circle(i \* 32 + 16, j \* 32 + 16, 5);  c.setFill(Color.***RED***);  circleList.add(c);  elements[i][j] = **null**;  p.getChildren().add(c);  }  ;  **break**;  **case** "box": { // 如果是box，则显示box图片，并将element棋盘数组的这个位置设置为box  ImageView img = **new** ImageView(Paths.***BOX***);  img.setX(i \* 32);  img.setY(j \* 32);  elements[i][j] = img;  p.getChildren().add(img);  }  ;  **break**;  **case** "stone": { // 如果是stone，则显示stone图片，并将element棋盘数组的这个位置设置为stone  ImageView img = **new** ImageView(Paths.***STONE***);  img.setX(i \* 32);  img.setY(j \* 32);  elements[i][j] = img;  p.getChildren().add(img);  }  ;  **break**;  **case** "null": // 如果是null，则将element棋盘数组的这个位置设置为null  elements[i][j] = **null**;  **break**;  }  }  // 随机添加人的位置  **boolean** manExist = **false**;  **while** (!manExist) { // 多次随机尝试  **int** i = (**int**) (Math.*random*() \* 20);  **int** j = (**int**) (Math.*random*() \* 20);  **if** (elements[i][j] == **null**) { // 随机生成的位置如果为null，则放置人，并跳出循环；反之继续随机生成，直到找到第一个null  man.setX(i \* 32);  man.setY(j \* 32);  p.getChildren().add(man);  manExist = **true**;  }  }  input.close(); // 关闭输入流文件  } **catch** (IOException e1) {  e1.printStackTrace();  }  });  root.getChildren().addAll(menuBar, p);  Scene scene = **new** Scene(root, mapw, maph + 32);  scene.setOnKeyPressed(e -> {  **int** x = (**int**) (man.getX() / 32);  **int** y = (**int**) (man.getY() / 32);  **switch** (e.getCode()) {  **case** ***UP***: {  Node node = elements[x][y - 1]; // node为小人上面的位置  System.***out***.println(node);  **if** (node **instanceof** ImageView) { // 如果小人上面的位置是一石头或箱子  ImageView img = (ImageView) node;  **if** (img.getImage().~~impl\_getUrl~~().equals(Paths.***BOX***)) { // 如果小人上面是箱子  **if** (elements[x][y - 2] == **null**) { // 如果箱子上面是空  elements[x][y - 2] = img; // 把箱子推到上面  elements[x][y - 1] = **null**; // 原来的位置箱子拿掉  img.setY(img.getY() - 32);  man.setY(man.getY() - 32);  }  }  } **else** {  man.setY(man.getY() - 32); // 如果小人上面的位置为空则上移  }  }  ;  **break**;  **case** ***DOWN***: {  // down  Node node = elements[x][y + 1]; // node为小人左面的位置  System.***out***.println(node);  **if** (node **instanceof** ImageView) { // 如果小人下面的位置是一石头或箱子  ImageView img = (ImageView) node;  **if** (img.getImage().~~impl\_getUrl~~().equals(Paths.***BOX***)) { // 如果小人下面是箱子  **if** (elements[x][y + 2] == **null**) { // 如果箱子下面是空  elements[x][y + 2] = img; // 把箱子推到下面  elements[x][y + 1] = **null**; // 原来的位置箱子拿掉  img.setY(img.getY() + 32);  man.setY(man.getY() + 32);  }  }  } **else** {  man.setY(man.getY() + 32); // 如果小人下面的位置为空则下移  }  }  ;  **break**;  **case** ***LEFT***: {  // left  Node node = elements[x - 1][y]; // node为小人左面的位置  System.***out***.println(node);  **if** (node **instanceof** ImageView) { // 如果小人左面的位置是一石头或箱子  ImageView img = (ImageView) node;  **if** (img.getImage().~~impl\_getUrl~~().equals(Paths.***BOX***)) { // 如果小人左面是箱子  **if** (elements[x - 2][y] == **null**) { // 如果箱子左面是空  elements[x - 2][y] = img; // 把箱子推到左面  elements[x - 1][y] = **null**; // 原来的位置箱子拿掉  img.setX(img.getX() - 32);  man.setX(man.getX() - 32);  }  }  } **else** {  man.setX(man.getX() - 32); // 如果小人左面的位置为空则左移  }  }  ;  **break**;  **case** ***RIGHT***: {  // right  Node node = elements[x + 1][y]; // node为小人右面的位置  System.***out***.println(node);  **if** (node **instanceof** ImageView) { // 如果小人右面的位置是一石头或箱子  ImageView img = (ImageView) node;  **if** (img.getImage().~~impl\_getUrl~~().equals(Paths.***BOX***)) { // 如果小人右面是箱子  **if** (elements[x + 2][y] == **null**) { // 如果箱子右面是空  elements[x + 2][y] = img; // 把箱子推到右面  elements[x + 1][y] = **null**; // 原来的位置箱子拿掉  img.setX(img.getX() + 32);  man.setX(man.getX() + 32);  }  }  } **else** {  man.setX(man.getX() + 32); // 如果小人右面的位置为空则右移  }  }  ;  **break**;  }  // 判断游戏是否完成  **boolean** gameover = **true**;  **for** (Circle c : circleList) {// 判断是不是所有的Circle的位置都有不为null，若是则游戏完成  **int** cx = (**int**) ((c.getCenterX() - 16) / 32);  **int** cy = (**int**) ((c.getCenterY() - 16) / 32);  **if** (elements[cx][cy] == **null**) {  gameover = **false**;  }  }  **if** (gameover) {  Alert alert = **new** Alert(AlertType.***INFORMATION***);  alert.setTitle("信息提示");  alert.setHeaderText(**null**);  alert.setContentText("游戏完成!");  alert.showAndWait();  }  });  **this**.setScene(scene);  **this**.setTitle("游戏界面");  **this**.getIcons().add(**new** Image(Paths.***BOX***));  }  }  **Results:**   * **开始游戏页面：**      * **编辑地图页面：**      * **保存地图：**      * **加载地图并开始游戏：**      * **游戏完成：**     **Summary:**   1. 该游戏通过20 X 20二维数组element来存放整张棋盘； 2. 创建目标点时即Circle类的示例时，要注意将横竖坐标位置+16，因为这里是圆心的位置； 3. 通过FileChooser这个类创建文件选择路径； 4. 在通过space键放置物品的时候，创建了一个计数变量index，每次点击space键，index变量自增1，通过对index%3的结果判定放置物品的属性是stone、box还是目标点；另外，因为大多数情况下放置的是石头，所以，在每次点击方向键的时候index会复位到0； 5. 在推箱子的操作中，先判定路径的前方第二位是否为空，若非空则不能推箱子（前方是石头），若不为空，则将箱子前推，并消除原来位置的箱子； 6. 判断游戏是否完成，即判断每个circle是否都非空即可。 | | | |

|  |  |  |
| --- | --- | --- |
| Teacher’s  comments | content and design of experiment（A-E）： |  |
| operations, algorithm description or code of program（A-E）： |  |
| results（A-E）： |  |
| summary and analysis of experiment（A-E）： |  |
| Grade（A-E）： | |