

**Course Experiment Report**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Course:** | Java Language | | | | | | |
|  |  | | | | | | |
| **Semester:** | 1-18th | **week** | 16th | **year** | | 1st | **term** |
|  |  |  |  |  | |  |  |
| **Major:** | Software Engineering | | | | | **Class:** | 2019 |
|  |  | | | | |  |  |
| **Student name:** | 冯春霖 | | **Student No.:** | | 222019321062074 | | |
|  |  | |  | |  | | |
| **Teacher:** | Wang Xiaomeng | | | | | | |

College of Computer and Information Science

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Exp7 GUI | | |
| Time | 2020.12.21 | Type | □Verification □Design □Synthetical |
| 1. Answer the questions  (1) What else can be improved in the program?  A: If the number of maps provided is large, an input box can be designed for jumping. A function of zooming window can be added.  (2) How to develop a map editing program for this game?  A: Let users design maps by dragging and dropping, and change the map array of logical layers at the same time as placing, and then export the logical map array to map files in format.  2. All Codes  The implementation of creating elements:  switch(type) {  case Element.***MAN***:  e = this.man = new Man(icons[Element.***MAN***]);  break;  case Element.***BOX***:  e = new Box(icons[Element.***BOX***]);  break;  case Element.***WALL***:  e = new Wall(icons[Element.***WALL***]);  break;  case Element.***TARGET***:  e= new Target(icons[Element.***TARGET***]);  break;  default:break;  }  The implementation of judging whether the game is terminated:  public boolean judge() {  boolean win = true;  for(ImageView img: targets) {//TBF  int x = (int) (img.getX() / ***CELL\_SIZE***);  int y = (int) (img.getY() / ***CELL\_SIZE***);  if(!(this.map[x][y] instanceof Box)) {//TBF  win = false;  break;  }  }  return win;  }  The implementation of moveMan():  public void moveManDown(){  int manX = (int) (man.getX() / ***CELL\_SIZE***);  int manY = (int) (man.getY() / ***CELL\_SIZE***);  if(manY < ylength - 1) {  if(map[manX][manY + 1] == null) {  man.down();  map[manX][manY] = null;  map[manX][manY + 1] = man;  }else if(map[manX][manY + 1] instanceof Box) {  if(manY + 1 < ylength - 1 && map[manX][manY + 2] == null) {  Box b = (Box) map[manX][manY + 1];  b.down();  map[manX][manY + 2] = b;  man.down();  map[manX][manY] = null;  map[manX][manY + 1] = man;  }  }//if  }//if  }    public void moveManLeft(){  int manX = (int) (man.getX() / ***CELL\_SIZE***);  int manY = (int) (man.getY() / ***CELL\_SIZE***);  if(manX > 0) {  if(map[manX - 1][manY] == null) {  man.left();  map[manX][manY] = null;  map[manX - 1][manY] = man;  }else if(map[manX - 1][manY] instanceof Box) {  if(manX - 1 > 0 && map[manX - 2][manY] == null) {  Box b = (Box) map[manX - 1][manY];  b.left();  map[manX - 2][manY] = b;  man.left();  map[manX][manY] = null;  map[manX - 1][manY] = man;  }  }//if  }//if  }    public void moveManRight(){  int manX = (int) (man.getX() / ***CELL\_SIZE***);  int manY = (int) (man.getY() / ***CELL\_SIZE***);  if(manX < xlength - 1) {  if(map[manX + 1][manY] == null) {  man.right();  map[manX][manY] = null;  map[manX + 1][manY] = man;  }else if(map[manX + 1][manY] instanceof Box) {  if(manX + 1 < xlength - 1 && map[manX + 2][manY] == null) {  Box b = (Box) map[manX + 1][manY];  b.right();  map[manX + 2][manY] = b;  man.right();  map[manX][manY] = null;  map[manX + 1][manY] = man;  }  }//if  }//if  }  The implementation of switching map:  Button nextBtn = new Button("Next");  nextBtn.setOnAction(e -> {  if(currentLevel < mapFiles.size()) {  currentLevel++;  currentMap.loadMap(mapFiles.get(currentLevel));  label.setText("Current Level:" + currentLevel);  primaryStage.sizeToScene();  }  });    Button resetBtn = new Button("Reset");  resetBtn.setOnAction(e -> {  currentMap.loadMap(mapFiles.get(currentLevel));  label.setText("Current Level:" + currentLevel);  primaryStage.sizeToScene();  });  Handing of input keys from keyboard:  switch(e.getCode()) {  case ***LEFT***:currentMap.moveManLeft();break;  case ***RIGHT***: currentMap.moveManRight();;break;  case ***UP***: currentMap.moveManUp();break;  case ***DOWN***: currentMap.moveManDown();;break;  default: break;  }  Screenshots of program execution: | | | |

|  |  |  |
| --- | --- | --- |
| Evaluation | Code Correctness (60%): |  |
| Experience (40%): |  |
| Score： | |