**《Java程序课程设计》**

**实验报告**

**题 目： 涂鸦跳跃**

**系 别： 计算机与信息工程学院**

**班 级： 22软件班**

**姓 名： 徐正天**

**学 号： 2215080126**

**日 期： 2023 年 12 月**

**计算机与信息工程学院**

**目 录**

[一．课程设计目的 3](#_Toc29041)

[1、知识目标 3](#_Toc2357)

[2、能力目标 3](#_Toc22065)

[二．课程设计的要求 3](#_Toc28438)

[1、题目及设计说明 3](#_Toc11978)

[2、游戏核心玩法和规则 4](#_Toc29364)

[3、游戏特色功能 4](#_Toc30543)

[三、课程设计过程 4](#_Toc11299)

[（一）需求分析 4](#_Toc27307)

[1、游戏需求概述 4](#_Toc26939)

[2、功能分析 4](#_Toc2947)

[（二）系统设计 5](#_Toc9588)

[1、系统流程图 5](#_Toc1329)

[2、模块功能设计 6](#_Toc3456)

[（三）主要功能模块 6](#_Toc11634)

[1、游戏主界面模块 6](#_Toc21674)

[2、平台生成模块 11](#_Toc29897)

[3、碰撞判定模块 13](#_Toc4682)

[4、游戏的无限循环模块 16](#_Toc17052)

[四、总结与体会 16](#_Toc20435)

# 一．课程设计目的

本课程设计项目为学生提供了一个既动手又动脑，独立实践的机会，将书本上的理论知识和实际需求有机地结合起来，从而锻炼学生分析问题、解决实际问题的能力，提高学生的编程序能力和创新意识。

1、知识目标

（1）熟练掌握面向对象的编程的基本思想

（2）掌握java 的相关语法、应用技术及综合运用。

（3）通过综合运用所学的java 的知识，进行一个简单完整的系统的开发，进一步理解和掌握java 开发桌面应用的技术和开发过程，掌握调试技巧及测试方法，增强解决处理问题的能力，积累开发经验。

2、能力目标

（1）培养学生将所学专业知识融会贯通，解决动态网站开发过程中所遇到的实际问题的能力及其基本工作素质。

（2）培养学生理论联系实际的工作作风、严肃认真的科学态度，既具有独立工作的能力，又能与人合作的风格，树立自信心。

# 二．课程设计的要求

1、题目及设计说明

本次课程设计题目是《涂鸦跳跃》。

设计说明：Doodle Jump 是一款经典的跳跃类游戏，玩家通过方向键控制角色不断向上跳跃，躲避障碍物、获得分数。本项目旨在实现一个简化版的 Doodle Jump 游戏，并加入一些基本的元素和功能，在制作过程中掌握java面向对象设计、图形界面开发等技术。

## 2、游戏核心玩法和规则

玩家通过敲击键盘左右键来控制角色的左右移动，在不同的平台间跳跃，包括普通平台、移动平台和不稳定的平台。有些平台会消失，有些则会移动，增加了游戏的挑战性。游戏中会出现各种道具，如弹簧、火箭等可以帮助角色跳得更高；同时也有敌对物体或障碍物，碰到它们会导致游戏失败。角色碰到敌对物体或障碍物时游戏结束，而碰到增益道具会获得加分。游戏根据角色跳跃的高度计算分数。在游戏中不断挑战自己的高分成绩是核心乐趣之一。

## 3、游戏特色功能

（1）无限纵向跳跃：游戏中的角色可以不断向上跳跃，没有固定的终点，玩家的目标是尽可能地跳得更高。

（2）多样化的平台设计：游戏中的平台多种多样，包括普通平台、移动平台、不稳定平台等，每种平台都有不同的特性，给游戏增添了更多挑战。

（3）丰富的道具：游戏中存在多种增益道具，如弹簧、火箭等，它们能够帮助角色跳得更高或者获得额外的分数。

（4）随机性与挑战性：平台的随机生成和障碍物的出现增加了游戏的随机性，每次游戏都有不同的挑战，让玩家难以预料。

（5）简单易上手：游戏操作简单，只需左右移动控制角色跳跃，不需要复杂的操作，容易让玩家上手。

# 三、课程设计过程

## （一）需求分析

1、游戏需求概述

涂鸦跳跃（Doodle Jump），是一款富有趣味的技巧性的游戏，在游戏中玩家要让涂鸦弹簧小怪物不停地往上跳跃，在跳跃中要小心破碎的平台、移动的蓝色平台、黑洞、不明飞行物和坏人，途中有快速上升的火箭和竹蜻蜓，这是一个富有趣味的技巧性游戏。跳得越高，分数就越高。

玩家可以通过键盘上的左右键控制小怪物不断移动，在不同的平台上跳跃，保持向上移动，避免碰撞障碍物和坏人，并且防止角色掉落到屏幕底部。游戏根据玩家跳跃到达的高度生成分数。

1. 功能分析

（1）跳跃控制：提供直观的操控方式，通过键盘上的方向键控制角色左右移动和跳跃。

（2）平台生成：随机或预设生成不同特性的平台，如普通平台、弹簧平台、可移动平台等。

（3）角色与场景设计：有趣的角色设计，包括动画效果和生动的场景设计。

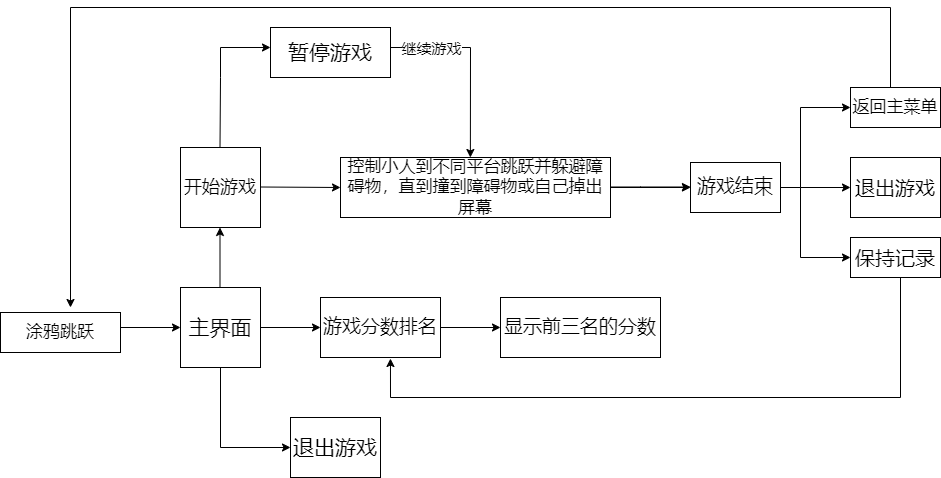
（4）游戏计分系统：显示当前分数，记录玩家最高分并提供排行榜。

（5）用户界面：提供简洁易懂的用户界面，包括开始游戏、查看排名选项。

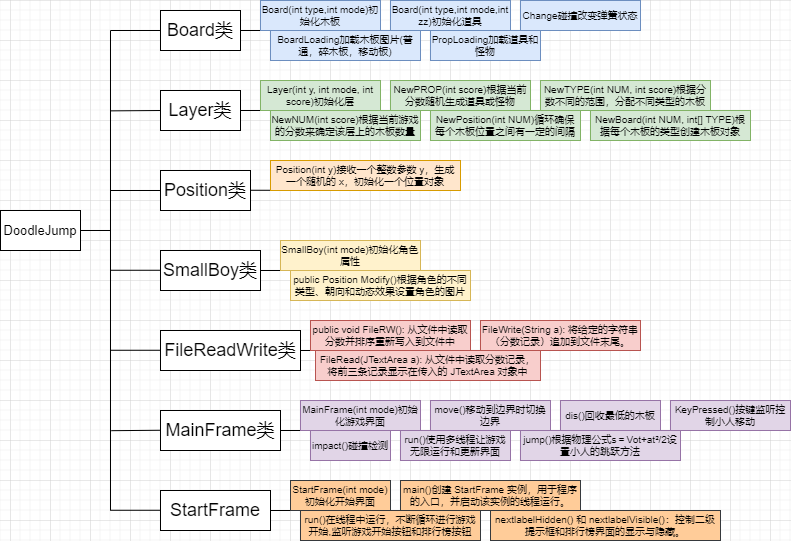
（6）游戏的持续性：提供失败后返回主菜单的选项，同时保存玩家的最高分等数据。

## （二）系统设计

### 1、系统流程图



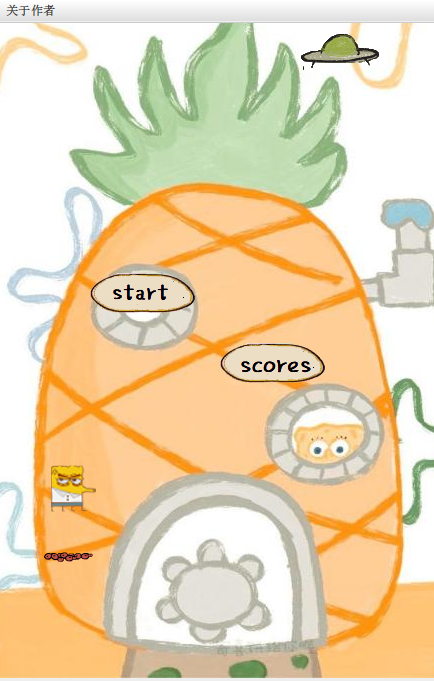
### 2、模块功能设计



## （三）主要功能模块

### 1、游戏主界面模块

效果图：



**初始化主界面：**

|  |
| --- |
| public StartFrame(int mode){//初始化开始界面  **//界面属性设置**  MODE=mode;  this.setIconImage(Toolkit.getDefaultToolkit().getImage("icon.png"));  this.setResizable(false);  this.setLayout(null);  this.setVisible(true);  this.setSize(450,720);  languagepath="image/English/";  this.setTitle("DoodleJump");  **//背景**  JLabel bgLabel=new JLabel(new ImageIcon("image/System/menu4.jpg"));  modeLabel1=new JLabel(new ImageIcon("image/System/basic.png"));  modeLabel2=new JLabel(new ImageIcon("image/System/basic.png"));  modeLabel1.setIcon(new ImageIcon("image/System/basic.png"));  bgLabel.setBounds(-5,-50,450,730);  modeLabel1.setBounds(0,620,450,60);  this.getLayeredPane().add(bgLabel, Integer.valueOf(Integer.MIN\_VALUE+1));  this.getLayeredPane().add(modeLabel1,Integer.valueOf(Integer.MIN\_VALUE));  this.getLayeredPane().add(modeLabel2,Integer.valueOf(Integer.MIN\_VALUE));  this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  JPanel jp=(JPanel)this.getContentPane();  jp.setOpaque(false);  **//开始按钮**  Start=new JButton(new ImageIcon(languagepath+"Button/start.png"));  Start.setContentAreaFilled(false);  Start.setBorder(null);  Start.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  isStart=true;  }});  Start.setBounds(90,250,105,40);  add(Start);  **//排行榜按钮**  Scores=new JButton(new ImageIcon(languagepath+"Button/scores.png"));  Scores.setContentAreaFilled(false);  Scores.setBorder(null);  Scores.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  isScores=true;  }});  Scores.setBounds(220,320,105,40);  add(Scores);  **//小人动画**  Board myboard=new Board(1,1);  myboard.setBounds(40,527,63,15);  add(myboard);  Stan=new SmallBoy(1);  Stan.setBounds(35,280,62,47);  add(Stan);  **//UFO动画**  UfoLabel=new JLabel(new ImageIcon("image/System/UFO1.png"));  UfoLabel.setBounds(291, 29, 101, 146);  add(UfoLabel,-1);  **//二级界面框**  NextLabel=new JLabel();  NextLabel.setOpaque(true);  NextLabel.setVisible(false);  Next=new JButton(new ImageIcon(languagepath+"Button/next.png"));  Next.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  isNext=true;  }});  NextLabel.setIcon(new ImageIcon(languagepath+"System/rule1.png"));  NextLabel.add(Next);  add(NextLabel);  **//排行榜界面**  nextScoreRecord =new JTextArea();  nextScoreCrown1 =new JLabel();  nextScoreCrown2=new JLabel();  nextScoreCrown3 =new JLabel();  Scores.addActionListener(new ActionListener() {  public void actionPerformed(ActionEvent e) {  NextLabel.setIcon(new ImageIcon(languagepath+"System/rank.png"));  NextLabel.setBounds(13,240,411,368);  nextScoreRecord.setBounds(160,90,200,200);  nextScoreRecord.setFont(new java.awt.Font("Dialog",1,25));  nextScoreRecord.setOpaque(false);  nextScoreRecord.setVisible(false);  NextLabel.add(nextScoreRecord);  nextScoreCrown1.setIcon(new ImageIcon("image/System/crown1.png"));  nextScoreCrown1.setBounds(100,130,30,30);  nextScoreCrown1.setVisible(false);  NextLabel.add(nextScoreCrown1);  nextScoreCrown2.setIcon(new ImageIcon("image/System/crown2.png"));  nextScoreCrown2.setBounds(100,165,30,30);  nextScoreCrown2.setVisible(false);  NextLabel.add(nextScoreCrown2);  nextScoreCrown3.setIcon(new ImageIcon("image/System/crown3.png"));  nextScoreCrown3.setBounds(100,195,30,30);  nextScoreCrown3.setVisible(false);  NextLabel.add(nextScoreCrown3);  Next.setIcon(new ImageIcon(languagepath+"Button/back.png"));  Next.setBounds(180,290,83,40);  FileReadWrite a=new FileReadWrite();  a.FileRW();  nextScoreRecord.setText("ID SCORE");  a.FileRead(nextScoreRecord);  nextScoreRecord.setEditable(false);  isScores=true;  }});  } |

### 平台生成模块

效果图：



**switch 语句根据不同的属性值进行绘制**

**k[i].PROP 是木板/道具的属性值，根据不同的属性值，它会执行不同的操作来设置 k[i].prop 的位置和图像**

**k[i].prop.setLocation() 用于设置道具或木板的位置。**

**add(k[i].prop) 将道具或木板添加到界面上。**

**根据属性值不同，选择不同的图像，例如：**

**对于属性值为4的道具，根据 propt 的值不同设置不同的图像。**

**对于属性值为5的道具，根据 propt 的值设置位置，并根据条件选择不同的图像。**

**属性值为6和7的道具会被添加到界面，并移除 k[i].myBoard[0]。**

|  |
| --- |
| for(int i=0;i<30;i++) {//重新绘制木板,道具  switch(k[i].PROP){  case 1:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-9);  add(k[i].prop);  break;  case 2:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-23);  add(k[i].prop);  break;  case 3:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-37);  add(k[i].prop);  break;  case 4:  k[i].prop.setLocation(k[i].myPosition[0].x,k[i].myPosition[0].y);  if(propt%30<=10)  k[i].prop.setIcon(new ImageIcon(modepath+"Fly1.png"));  if(propt%30<=20&&propt%30>10)  k[i].prop.setIcon(new ImageIcon(modepath+"Fly2.png"));  if(propt%30<=30&&propt%30>20)  k[i].prop.setIcon(new ImageIcon(modepath+"Fly3.png"));  add(k[i].prop);  break;  case 5:  k[i].prop.setLocation(k[i].myPosition[0].x+(int)24\*propt/400,k[i].myPosition[0].y-50);  if(propt==0)  k[i].prop.setIcon(new ImageIcon(modepath+"Move2.png"));  if(propt==400)  k[i].prop.setIcon(new ImageIcon(modepath+"Move1.png"));  add(k[i].prop);  break;  case 6:  k[i].prop.setLocation(k[i].myPosition[0].x,k[i].myPosition[0].y);  add(k[i].prop);  remove(k[i].myBoard[0]);  break;  case 7:  k[i].prop.setLocation(k[i].myPosition[0].x,k[i].myPosition[0].y);  add(k[i].prop);  remove(k[i].myBoard[0]);  break;  case 8:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-29);  add(k[i].prop);  break;  }  for(int j=0;j<k[i].NUM;j++){  if(k[i].TYPE[j]==4)  k[i].myPosition[j].y=k[i].y+(int)100\*propt/400-50;  if(k[i].TYPE[j]==5)  k[i].myPosition[j].x=k[i].x+(int)100\*propt/400-50;  k[i].myBoard[j].setLocation(k[i].myPosition[j].x,k[i].myPosition[j].y);  }  } |

### 碰撞判定模块

**判断跳跃状态下的碰撞：**

**当 jumpStatus 为 true 且道具类型为4、5、6或7时（这些类型表示可以跳跃），进行以下操作：**

**根据道具类型设置一个特定的 x 和 y 值，表示小人和道具之间的碰撞区域。**

**判断小人的位置是否和道具的位置重合，如果是，根据道具类型执行相应的操作。例如，如果道具类型是7，即黑洞，将 isJump 设置为 false，并让 Stan 不可见。**

**判断非跳跃状态下的碰撞：**

**当 jumpStatus 为 false 时，也就是小人下降状态，对木板和道具的碰撞进行检测。**

**如果小人落在木板上，重置了一些参数（跳跃状态、垂直初速度、加速度），并根据木板的类型进行不同的处理，比如移除木板、改变小人的属性等。**

**如果小人和道具相撞，根据道具类型进行不同的操作，修改小人属性、加分。**

|  |
| --- |
| public void impact(){  for(int i=0;i<30;i++) {  for(int j=0;j<k[i].NUM;j++){  if(jumpStatus==true&&(k[i].PROP==4||k[i].PROP==5||k[i].PROP==6||k[i].PROP==7)&&(Stan.TYPE!=3)) {  int x=0,y=0;  switch(k[i].PROP) {  case 4:  x=79;  y=45;  break;  case 5:  x=39;  y=50;  break;  case 6:  x=83;  y=53;  break;  case 7:  x=64;  y=62;  break;  }  if((X>k[i].myPosition[0].x-StanWidth/3\*2&&X<k[i].myPosition[0].x+x-StanWidth/3&&Y>k[i].myPosition[0].y&&Y<k[i].myPosition[0].y+y))  {isJump=false;  if(k[i].PROP==7)Stan.setVisible(false);//碰到黑洞，Stan消失  switch(k[i].PROP) {  }  }  }  if(jumpStatus==false&&(X>k[i].myPosition[j].x-StanWidth/3\*2&&X<k[i].myPosition[j].x+BdWidth-StanWidth/3&&Y>k[i].myPosition[j].y-StanHeight\*2/3-15&&Y<k[i].myPosition[j].y-StanHeight\*2/3+15)){  jumpStatus=true;  t=0;  StanBasePOS=k[i].myPosition[j].y-StanHeight;  V0=4;  a=25;  switch(k[i].TYPE[j]){//木板碰撞  case 1:  if(k[i].PROP!=4||k[i].PROP!=5||k[i].PROP!=6)  break;  case 2:  remove(k[i].myBoard[j]);  k[i].myPosition[j].x=-100;  break;  case 3:  jumpStatus=false;  t=V0\*a\*2;  remove(k[i].myBoard[j]);  k[i].myPosition[j].x=-100;  }  if(k[i].PROP!=0){//道具碰撞  Stan.TYPE=k[i].PROP;  if(k[i].PROP==4||k[i].PROP==5||k[i].PROP==6) {  Stan.TYPE=1;  }  switch(k[i].PROP){  case 1:  V0=8;  a=25;  k[i].Change();  break;  case 2:  V0=10;  a=30;  k[i].Change();  break;  case 3:  V0=20;  a=20;  k[i].Change();  break;  case 4:  k[i].prop.setVisible(false);  k[i].myPosition[0].x=-100;  SCORE=SCORE+250;  break;  case 5:  k[i].prop.setVisible(false);  k[i].myPosition[0].x=-100;  SCORE=SCORE+500;  break;  case 6:  k[i].prop.setVisible(false);  k[i].myPosition[0].x=-100;  SCORE=SCORE+1000;  break;  case 7:  isJump=false;  Stan.setVisible(false);  break;  }  }  }  }  }  } |

### 4、游戏的无限循环模块

|  |
| --- |
| **首先是在run()方法中的while(true)循环，这个循环会一直运行，直到游戏结束或者用户选择返回。在这个循环中，游戏的各种状态（如是否暂停、是否开始、是否死亡等）都会被检查，并且相应的行为（如移动、跳跃、碰撞检测等）也会被执行。**  **其次是在dis()方法中，当一个木板的位置超过屏幕底部时，这个木板会被移除，并且会在屏幕顶部生成一个新的木板。这样就实现了木板的无限循环，让玩家可以一直跳跃。**  if(PAUSE==false) {  if(isStart==true){  move(); //检测移动  Stan.Modify();//更改小人样式  StanWidth=Stan.Modify().x;  StanHeight=Stan.Modify().y;  dis(); //刷新多余木板  if(moveBg==false){  Y = StanBasePOS - jump(t);  t++;  if(propflag==true)  propt++;  else  propt--;  }  else{ //移动背景  for(int i=0;i<30;i++) {  k[i].y=k[i].y+V0-t/a;  for(int j=0;j<k[i].NUM;j++) {  k[i].myPosition[j].y=k[i].y;  }  }  StanBasePOS=270+a\*V0\*V0/2;  SCORE+=V0-t/a;  Score.setText("SCORE:"+SCORE);  t++;  if(propflag==true)  propt++;  else  propt--;  }  if(t==V0\*a){//下降时  jumpStatus=false;  Stan.TYPE=1;  Stan.FLASH=0;  }  if(Stan.TYPE==2){  if(t%20<10)  Stan.FLASH=1;  else  Stan.FLASH=2;  }  if(Y<270&&jumpStatus==true&&moveBg==false){//判断能否移动背景  moveBg=true;  }  else if(t==V0\*a&&moveBg==true){  moveBg=false;  }  public void dis(){  for(int i=0;i<30;i++){  if(k[i].y>=690){  if(k[i].PROP!=0)  remove(k[i].prop);//如果当前元素的 PROP 属性不为 0，则移除 k[i] 的 prop 组件。  for(int j=0;j<k[i].NUM;j++)  remove(k[i].myBoard[j]);//移除所有相关的板块  k[i]=new Layer(findMin()-50-(int)SCORE/4000,MODE,SCORE);  switch(k[i].PROP){  case 1:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-9);  add(k[i].prop);  break;  case 2:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-23);  add(k[i].prop);  break;  case 3:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-37);  add(k[i].prop);  break;  case 7:  k[i].prop.setLocation(k[i].myPosition[0].x+22,k[i].myPosition[0].y-29);  add(k[i].prop);  break;  //如果 PROP 是 1、2、3 或 7，设置 prop 组件在特定位置。  case 4:  k[i].prop.setLocation(k[i].myPosition[0].x,k[i].myPosition[0].y);  add(k[i].prop);  k[i].myBoard[0].setVisible(false);//如果 PROP 是 4，设置 prop 组件在另一个位置，并将相应的 myBoard 不可见。  break;  }  for(int j=0;j<k[i].NUM;j++) {  k[i].myBoard[j].setBounds(k[i].myPosition[j].x, k[i].myPosition[j].y,57, 15);  add(k[i].myBoard[j]);//遍历 myBoard 数组中的每个元素，并使用 setBounds 方法将它们放置到新的坐标位置上，将板块重新添加到界面中。  }  }  }  } |

# 四、总结与体会

**难点简述：**首先是人物的移动和跳跃逻辑，最初使用匀速上升和下降的逻辑，但动画有些违和，最后采用了物理学中路程与加速度的公式来规定人物的跳跃方式，使人物动画更合理，在主界面的UFO动画中，最初的想法是利用椭圆公式让UFO做椭圆运动，但是公式较为复杂难以用代码实现，最后用sin和cos函数来实现UFO的曲线周期运动。其次是木板道具与人物的碰撞逻辑，通过设定x和y的范围判断小人是否撞到道具或怪物，这个过程需要不断地调整参数，也重温了很多高中数学和物理公式。但目前仍存在运行游戏时会出现主界面图片加载不全需要等待一段时间才会加载完全的问题，目前还不知道该如何解决。

**系统总结：**游戏中通过设置七个类来操作不同的功能实现，通过类和类之间具有一定的联系和作用实现了整个程序的有机结合，是整个程序得以正常运行的保证。Board类和Layer类相互联系实现了木板和道具的初始化，Position类则是负责位置的随机生成，MainFrame负责初始化游戏界面，实现了平台的动态生成、小人的跳跃、道具的加分功能等游戏逻辑。StartFrame负责实现游戏的主菜单界面。通过单独设置小功能的类再将其在程序运行中调用方法，减轻了代码的冗杂度和有利于系统的维护循环利用，也减少了不必要的关联性。同时，也保证了类的分工明确，便于代码的检查和修改。

**个人体会：**在整个游戏制作过程中，起初我对Swing和按键监听等组件了解不多，对游戏框架也模糊不清。最初，我的构想很多，想要加入音乐、更多的游戏模式，甚至是一个射击功能。但随着制作的深入，代码逐渐变得臃肿，而图片的制作也给我的负担增添不少。最终，我做出了一个决定，砍掉了多余的模式和功能，只保留了一个我喜欢的SpongeBob模式和最经典的玩法，但代码层面还是有些许欠缺，类与类相互之间的逻辑关系还是非常混乱的，这也导致了在碰撞判定部分和平台生成部分花费了很多调整和尝试的时间，并且项目逻辑上还有很多可以优化的空间。

确认游戏模式和玩法后，我开始系统地学习API中各个类库、组件的相关特性和方法。通过上网学习Swing库中各个组件的用法，探究了Java游戏开发的基本框架和流程。我学会了通过多线程让游戏界面无限循环下去，同时也努力学习使用Photoshop进行图像抠图处理，以提升游戏画面的质感和美观度。

游戏内的图片素材部分来自于网上，但也有一部分是我亲自手绘的。然而，在游戏背景图和人物图片方面，我遇到了一些困难，因为无法将它们处理成我满意的图片。于是，我请了钟晓铃这位土管专业的同学请求帮忙，感谢她在百忙之中抽空帮我重新绘制了游戏背景图和人物图。在此特别感谢钟晓铃同学的帮助，她的绘画让游戏画面更加完美。

通过这段时间的学习和实践，我不仅获得了技术上的进步，更是在解决问题、思考方式、创意发挥等方面有了不小的成长。我希望未来能够继续深入学习游戏开发，探索更多创新的点子和实践新技术，将自己所学运用到未来的项目中。

总的来说，这段时间的游戏制作课程是一段难忘的旅程。它不仅是技术上的学习，更是对自己能力和潜力的挖掘。这门课程不仅让我理解了游戏开发的本质，更让我明白了追求梦想和坚持不懈的重要性。这段经历将成为我未来学习和职业生涯的宝贵财富。