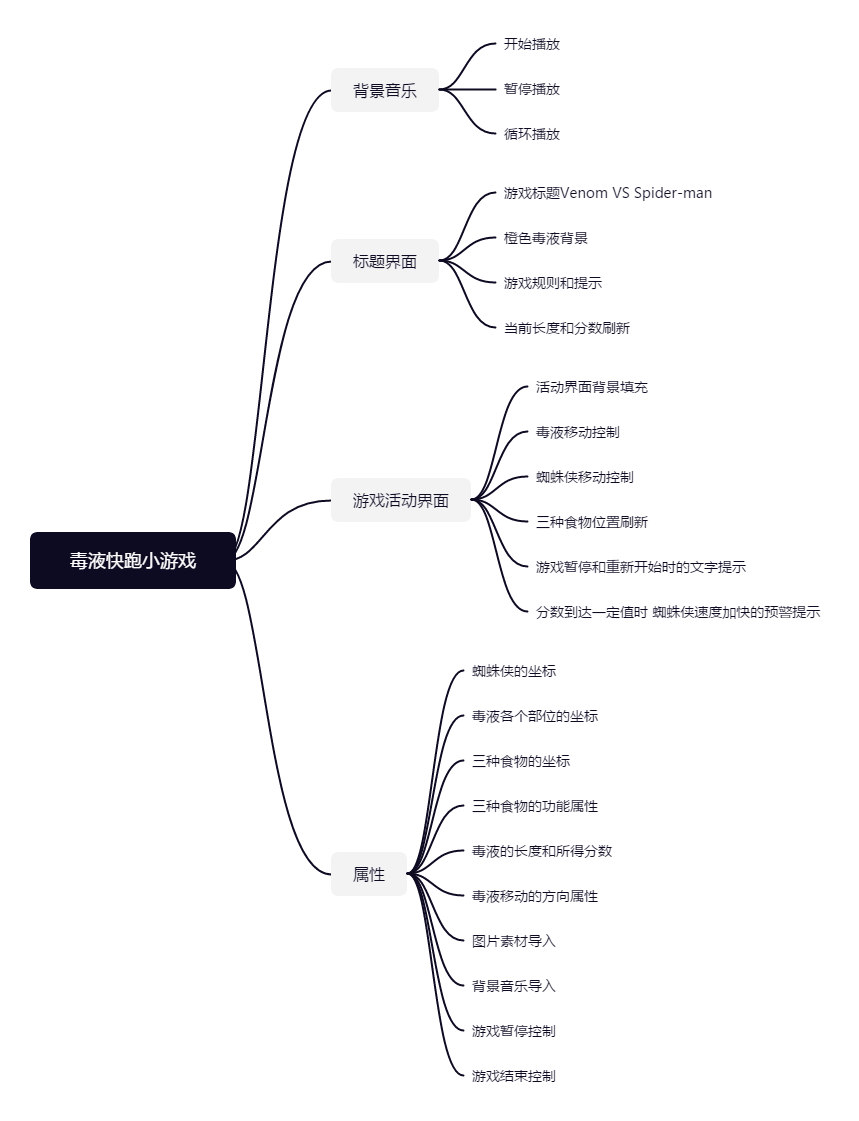
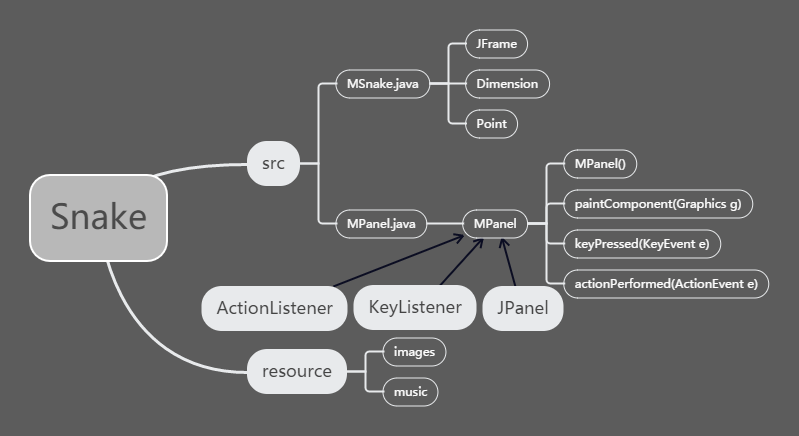
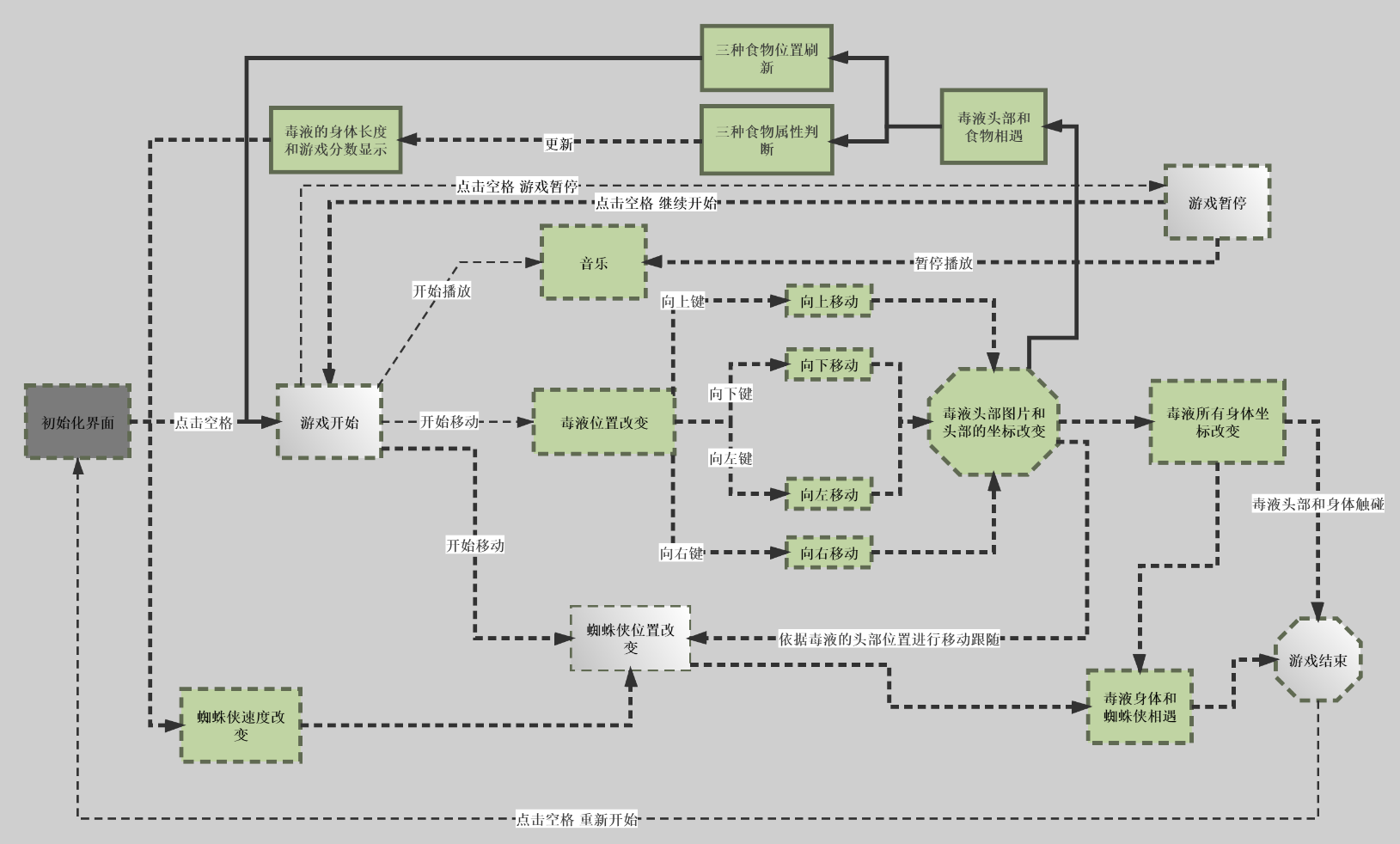
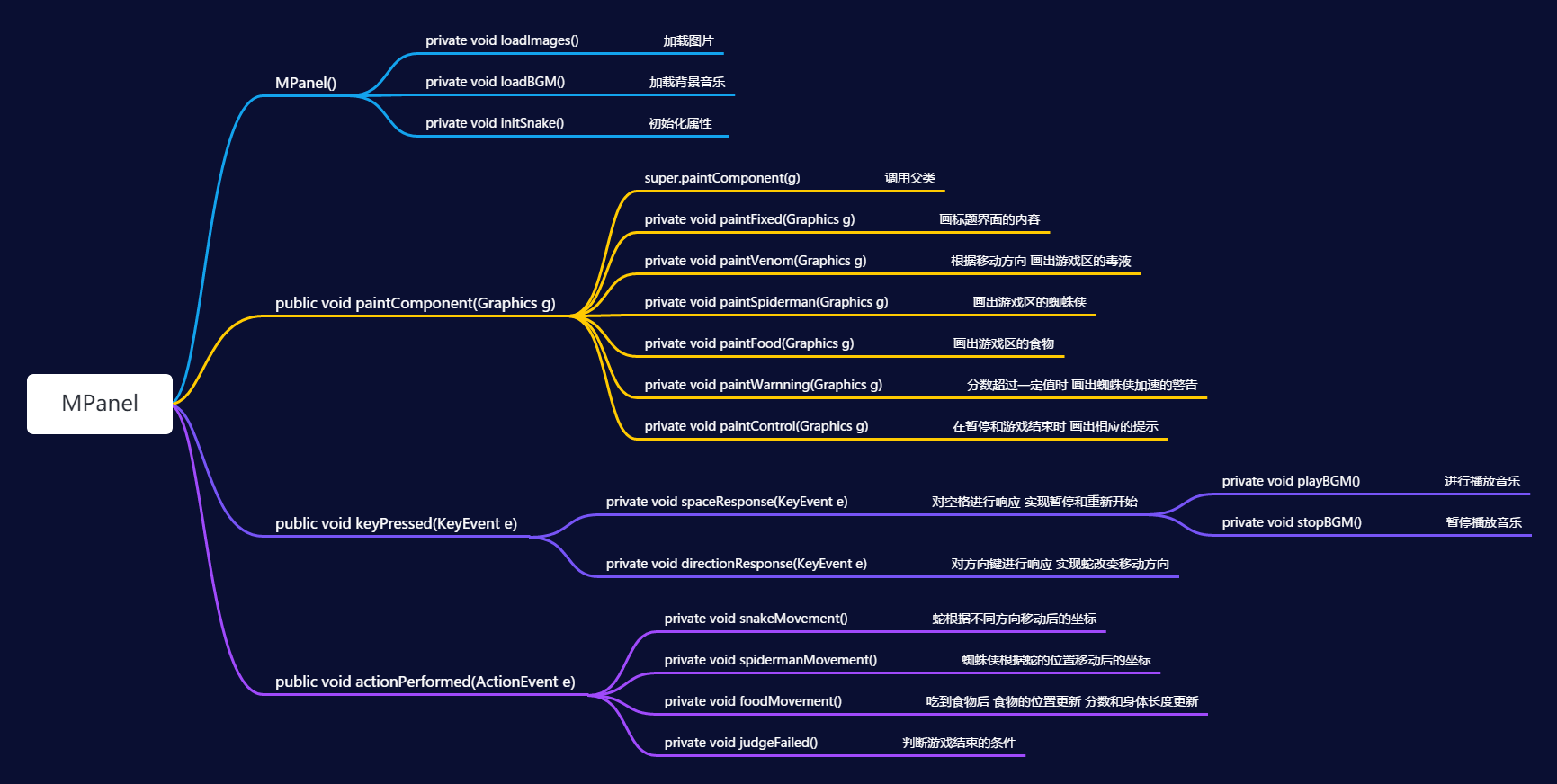
**毒液快跑小游戏**

**摘要：**

本次程序设计的主题是《毒液快跑》，是对贪吃蛇游戏的大拓展。游戏的主要内容大致分为游戏界面设计和游戏功能设计两大部分，其中游戏界面设计是以满足用户的视觉体验为主，并加入了背景音乐，增加用户的体验感。游戏功能设计是在满足原有的贪吃蛇游戏基本功能的基础上，拓展了一些更加新颖的元素，改变了游戏规则。本次的实现代码总共600行左右，主要运用了Java的Swing组件和java.awt包中的GUI设计工具。

1. **系统总体设计**
   1. 需求分析：
2. 三种不同的食物在被毒液吃掉后会在新位置随机生成。
3. 毒液在吃到不同的食物后，长度和所得分数会发生相应的改变。
4. 对于毒液，通过方向键实现移动方向的改变，同时，头部造型也随方向的改变而做出相应的调整。
5. 蜘蛛侠会根据毒液的移动方向，自动做出移动方向的调整，去追赶毒液。
6. 毒液分数超过一定值时，蜘蛛侠移动速度加快，并且在游戏区给出相应的警告。
7. 毒液被蜘蛛侠抓到或者毒液的头部触碰了身体时，游戏结束。
8. 添加了游戏背景音乐，并且在游戏过程中可以循环播放。
9. 添加了控制按钮（空格），可以实现对游戏及音乐的暂停，以及在游戏结束后的重新开始。
10. 在标题界面画出了三种不同食物对应的属性和提示，并实时更新毒液的长度和游戏分数。
11. 打包成jar文件，方便使用。
    1. 概要设计：
12. 编写一个MPanel类，将对应功能的实现都放入这个类中。
13. 通过该类的构造函数，实现游戏初始化，图片和音乐的加载。
14. 在该类中通过方法paintComponent(Graphics g)来调用该类中的所有paint方法，进行绘图。
15. 在该类中通过方法keyPressed(KeyEvent e)来调用该类中的所有Response方法，进行对键盘的监听和响应。
16. 在该类中通过方法actionPerformed(ActionEvent e)来调用该类中的所有Movement方法和judgeFailed方法，来实现内容的移动和游戏结束判断。
17. **功能模块划分**



1. **系统详细设计**
2. **类的设计及类继承关系**
3. **系统流程图**
4. **方法的调用关系**
5. **各个功能模块的实现方法描述**

**一、键盘监听模块**

**public** **void** keyPressed(KeyEvent e) {//键盘监听

spaceResponse(e);

directionResponse(e);

}

//对空格进行响应 实现暂停和重新开始

**private** **void** spaceResponse(KeyEvent e) {

**int** keyCode=e.getExtendedKeyCode();//获取所敲的键盘对应的数字

**if**(keyCode==KeyEvent.***VK\_SPACE***) {//如果敲击的是空格键

**if**(isFailed) {//游戏结束后再次敲击空格 重新开始

isFailed=**false**;//游戏重新开始时将isFailed置false

initSnake();

}

**else** {

isStarted=!isStarted;//取反状态 开始-暂停 暂停-开始

}

repaint(); //修改isStarted参数后重新调用方法

//重新画一遍 使得效果展示出来

**if**(isStarted) {//在空格响应事件中实现声音的播放和暂停

playBGM();

}**else** {

stopBGM();

}

}

}

//对方向键进行响应 实现毒液改变移动方向

**private** **void** directionResponse(KeyEvent e){

//新增判断条件 修复毒液直接掉头致死的bug

**int** keyCode=e.getExtendedKeyCode();//获取所敲的键盘对应的数字

**if**(keyCode==KeyEvent.***VK\_LEFT***&&(snakex[0]-25)!=snakex[1]){

direction="L";

}

**else** **if**(keyCode==KeyEvent.***VK\_RIGHT***&&(snakex[0]+25)!=snakex[1]){

direction="R";

}

**else** **if**(keyCode==KeyEvent.***VK\_UP***&&(snakey[0]-25)!=snakey[1]){

direction="U";

}

**else** **if**(keyCode==KeyEvent.***VK\_DOWN***&&(snakey[0]+25)!=snakey[1]){

direction="D";

}

}

键盘监听模块，主要针对空格和方向键进行监听，由于空格可以控制游戏的暂停和游戏结束时的重新开始，因此在spaceResponse方法中，通过成员isFailed和isStarted分别进行对游戏结束和游戏暂停进行控制。在directionResponse方法中，通过成员direction记录监听到的方向键，实现毒液的移动，并将毒液的头部坐标和第一个身体的坐标进行对比，完善了毒液直接掉头致死的bug。

**二、事件处理模块**

**public** **void** actionPerformed(ActionEvent e) {//经过100ms之后调用该方法 **if**(isStarted&&!isFailed) {//引入判断条件 实现暂停

snakeMovement(); //蛇根据不同方向的移动

spidermanMovement();//蜘蛛侠根据蛇的位置移动

foodMovement();//吃到食物后 食物的位置更新 分数和身体长度更新

judgeFailed();//判断游戏结束的条件 蛇的头部和身体的坐标重叠了 或者蜘蛛侠和蛇的身体触碰

repaint();

}//if(isStarted&&!isFailed)

timer.start();//重画之后时钟继续

}//actionPerformed(ActionEvent e)

//毒液根据不同方向的移动

**private** **void** snakeMovement() {

//注意要从后向前移 不然会覆盖掉

**for**(**int** i=len-1;i>0;i--) {//将当前的身体部位移动到其前一身体部位的位置

snakex[i]=snakex[i-1];

snakey[i]=snakey[i-1];

}

//身体移动后 移动头部 需要根据蛇当前的运动方向

/\*注意坐标是方块的左上角 活动右边界=850(活动范围)+25(左边框)=875

\* 横坐标到850时 占据的格子的横坐标是850-875 超过850就超出了右边界

\*/

**if**(direction=="R") {

snakex[0]=snakex[0]+25;

**if**(snakex[0]>850) {//超出右边界 则从左边回来

snakex[0]=25;

}

}

/\*注意坐标是方块的左上角 活动左边界=25(左边框)

\* 横坐标到25时 占据的格子的横坐标是25-50 小于25就超出了左边界

\*/

**else** **if**(direction=="L")

{

snakex[0]=snakex[0]-25;

**if**(snakex[0]<25) {

snakex[0]=850;

}

}

/\*注意坐标是方块的左上角 活动上边界=215

\* 纵坐标坐标到75时 占据的格子的纵坐标是215-240 小于215就超出了上边界

\*/

**else** **if**(direction=="U")

{

snakey[0]=snakey[0]-25;

**if**(snakey[0]<215) {

snakey[0]=740;

}

}

/\*注意坐标是方块的左上角 活动下边界=215+550(活动范围)=765

\* 纵坐标坐标到740时 占据的格子的纵坐标是740-765 超过740就超出了下边界

\*/

**else** **if**(direction=="D")

{

snakey[0]=snakey[0]+25;

**if**(snakey[0]>740) {

snakey[0]=215;

}

}

}

//蜘蛛侠根据毒液的位置移动

**private** **void** spidermanMovement() {

**if**(score>-2000) {

**if**(spiderman1\_x>snakex[0]) {

spiderman1\_x-=(1+(score/500)\*0.5);

}

**if**(spiderman1\_x<snakex[0]) {

spiderman1\_x+=(1+(score/500)\*0.5);

}

**if**(spiderman1\_y<snakey[0]) {

spiderman1\_y+=(1+(score/500)\*0.5);

}

**if**(spiderman1\_y>snakey[0]) {

spiderman1\_y-=(1+(score/500)\*0.5);

}

}

}

//吃到食物后 食物的位置更新 分数和身体长度更新

**private** **void** foodMovement(){

//食物1：吃到身体+1 分数+100

**if**(snakex[0]==foodx&&snakey[0]==foody) {

len++;//吃到食物后长度+1

score=score+100;//吃到食物后增加分数

**if**(score==500||score==1000||score==1500||score==2000||score==550||score==1050||score==1550||score==2050) {

times=30;

}

snakex[len-1]=snakex[len-2];//因为实现的顺序是先打印后移动的 未初始化的队尾会短暂停留在(0,0)处

snakey[len-1]=snakey[len-2];//所以将新队尾的位置初始化为蛇之前队尾的位置

//食物被吃掉后 重新随机生成一个新位置

foodx=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

foody=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

**for**(**int** i=0;i<len;i++)//实现食物的位置尽量不出现在蛇的身体里或者和其它食物重叠

{

**if**(snakex[i]==foodx&&snakey[i]==foody||(food2x==foodx&&food2y==foody)||(foodx==food3x&&foody==food3y))

{

foodx=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

foody=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

}

}

}

//食物2：吃到身体-1 分数+50

**if**(snakex[0]==food2x&&snakey[0]==food2y) {

**if**(len>5)//吃到食物后长度-1 但是不能小于基础长度

len--;

score=score+50;//吃到食物后分数增加

**if**(score==500||score==1000||score==1500||score==2000) {

times=30;

}

//食物被吃掉后 重新随机生成一个新位置

food2x=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

food2y=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

**for**(**int** i=0;i<len;i++)//实现食物的位置尽量不出现在蛇的身体里或者和其它食物重叠

{

**if**((snakex[i]==food2x&&snakey[i]==food2y)||(food2x==foodx&&food2y==foody)||(food3x==food2x&&food3y==food2y))

{

food2x=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

food2y=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

}

}

}

//食物3：吃到身体+2 分数-200

**if**(snakex[0]==food3x&&snakey[0]==food3y) {

score=score-200;

len++;

snakex[len-1]=snakex[len-2];//因为实现的顺序是先打印后移动的 未初始化的队尾会短暂停留在(0,0)处

snakey[len-1]=snakey[len-2];//所以将新队尾的位置初始化为蛇之前队尾的位置

len++;

snakex[len-1]=snakex[len-2];//因为实现的顺序是先打印后移动的 未初始化的队尾会短暂停留在(0,0)处

snakey[len-1]=snakey[len-2];//所以将新队尾的位置初始化为蛇之前队尾的位置

//食物被吃掉后 重新随机生成一个新位置

food3x=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

food3y=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

**for**(**int** i=0;i<len;i++)//实现食物的位置尽量不出现在蛇的身体里或者和其它食物重叠

{

**if**(snakex[i]==food3x&&snakey[i]==food3y||(food3x==foodx&&food3y==foody)||(food3x==food2x&&food3y==food2y))

{

food3x=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

food3y=215+25\*rand.nextInt(22);//活动范围是34\*24个25\*25的格子

}

}

}

}

//判断游戏结束的条件 蛇的头部和身体的坐标重叠了 或者蜘蛛侠和蛇的身体触碰

**private** **void** judgeFailed(){

**for**(**int** i=1;i<len;i++) {

**if**(snakex[i]==snakex[0]&&snakey[i]==snakey[0]) {

isFailed=**true**;

}

}

**for**(**int** i=0;i<len;i++) {

**if**(snakex[i]>=spiderman1\_x&&snakex[i]<=spiderman1\_x+50&&snakey[i]>=spiderman1\_y&&snakey[i]<=spiderman1\_y+50) {

isFailed=**true**;

}

}

}

事件处理模块，程序每经过100ms，就会因键盘监听模块中成员的改变，而做出相应的处理，在snakeMovement方法中，毒液的坐标根据direction成员的改变而做出对应的调整，同时，考虑到毒液移动到边缘时可以从另一端回到游戏区，在游戏区边缘处引入了新的判断条件；在spidermanMovement方法中，通过将蜘蛛侠的坐标与毒液的头部坐标进行比较，从而判断下一步蜘蛛侠的移动方向，并给出移动的坐标；在foodMovement方法中，将毒液的头部坐标和三种不同的食物坐标进行对比，如果两者的坐标相同，则说明毒液吃到了对应的食物，根据每种食物的不同属性，对分数和毒液的身体长度进行修改更新，并且在游戏区中重新随机生成该食物的位置坐标；在judgeFailed方法中，通过将毒液头部坐标与身体坐标，以及蜘蛛侠坐标进行对比，如果蜘蛛侠的坐标和毒液某个位置的坐标相同，或者毒液头部坐标和身体的某个位置坐标相同，游戏结束。

**三、界面设计模块**

**public** **void** paintComponent(Graphics g) {//画组件,参数g相当于画笔

**super**.paintComponent(g);//调用父类

paintFixed(g); //画标题和一些固定不变的内容

paintVenom(g); //根据方向 画出不同的Venom

paintSpiderman(g);//画出移动的蜘蛛侠

paintFood(g); //画出三种不同的食物

paintWarnning(g); //分数超过一定值时，给出蜘蛛侠加速的警告

paintControl(g); //画出在暂停和重新开始时的提示

}

//画标题和一些固定不变的内容

**private** **void** paintFixed(Graphics g) {

**this**.setBackground(Color.***WHITE***);

title.paintIcon(**this**,g,25,0);/\*第一个参数代表画的位置

第二个参数是方法传入的画笔

第三和第四个参数是左上角的横纵坐标\*/

//fillRect函数用指定的画刷填充矩形

g.setColor(Color.***DARK\_GRAY***);

g.fillRect(25,215, 850, 550);//横坐标25 纵坐标215

//width=850(去掉左右各25留空) 纵向坐标：215->765

food\_1.paintIcon(**this**, g, 675, 55);

food\_3.paintIcon(**this**, g, 675, 80);

food\_2.paintIcon(**this**, g, 675, 105);

g.setColor(Color.***black***);

g.setFont(**new** Font("Microsoft YaHei UI",Font.***BOLD***,20));

g.drawString("Venom", 500, 38);

venom.paintIcon(**this**, g, 570, 11);

g.setColor(Color.***BLUE***);

g.setFont(**new** Font("Microsoft YaHei UI",Font.***BOLD***,30));

g.drawString("VS", 612, 42);

spidermanU.paintIcon(**this**, g, 655, 11);

g.setColor(Color.***RED***);

g.setFont(**new** Font("Microsoft YaHei UI",Font.***BOLD***,20));

g.drawString("Spider-man", 695, 38);

g.setColor(Color.***white***);

g.setFont(**new** Font("Microsoft JhengHei",Font.***BOLD***,15));

g.drawString(":len+1 socre+100",700,73);

g.drawString(":len - 1 socre+50",700,123);

g.drawString(":len+2 socre - 200",700,98);

g.drawString("Tips:Don't eat",550,158);

food\_3.paintIcon(**this**, g, 650, 140);

g.drawString("and watch out for", 675, 158);

spiderman1.paintIcon(**this**, g,800,128);

g.setFont(**new** Font("方正粗黑宋简体",Font.***PLAIN***,20));

g.drawString("Len:"+len, 680, 195);

g.drawString("Score:"+score, 750, 195);

//spiderman1.paintIcon(this, g, 25, 260);

zhizhuxiaL.paintIcon(**this**, g, 0, 215);//左边装饰的蜘蛛侠

}

//分数超过一定值时，给出蜘蛛侠加速的警告

**private** **void** paintWarnning(Graphics g) {

**if**(score>=500&&score<1000&&times>0)

{

--times;

g.setColor(Color.***RED***);

g.setFont(**new** Font("arial",Font.***BOLD***,30));

g.drawString("WARNNING:Spider-man is getting angry!!!", 130, 300);

}

**if**(score>=1000&&times>0&&score<2000)

{

--times;

**if**(score<1500){

g.setColor(Color.***pink***);

}

**else** {

g.setColor(Color.***red***);

}

g.setFont(**new** Font("arial",Font.***BOLD***,30));

g.drawString("WARNNING:You are driving Spider-man crazy!!!", 130, 300);

}

**if**(score>=2000&&times>0)

{

--times;

g.setColor(Color.***PINK***);

g.setFont(**new** Font("arial",Font.***BOLD***,30));

g.drawString("Hahaha...Spider-man sucksss!!!", 225, 300);

}

}

//根据方向 画出不同的Venom

**private** **void** paintVenom(Graphics g) {

//先填充矩阵再初始化 不然会覆盖

**if**(direction=="R") {//根据一组if else结构打印出所需要的头

right.paintIcon(**this**, g, snakex[0], snakey[0]);//初始时在左上角 且使用向右的头部

}

**else** **if**(direction=="L") {

left.paintIcon(**this**, g, snakex[0], snakey[0]);

}

**else** **if**(direction=="U") {

up.paintIcon(**this**, g, snakex[0], snakey[0]);

}

**else** **if**(direction=="D") {

down.paintIcon(**this**, g, snakex[0], snakey[0]);

}

**for**(**int** i=1;i<len;i++) {//通过循环，打印出蛇的body

body.paintIcon(**this**, g, snakex[i], snakey[i]);

}

}

//画出移动的蜘蛛侠

**private** **void** paintSpiderman(Graphics g) {

spiderman1.paintIcon(**this**, g, spiderman1\_x, spiderman1\_y);

}

//画出三种不同的食物

**private** **void** paintFood(Graphics g) {

food.paintIcon(**this**, g, foodx, foody);

food2.paintIcon(**this**, g, food2x, food2y);

food3.paintIcon(**this**, g, food3x, food3y);

}

//画出在暂停和重新开始时的提示

**private** **void** paintControl(Graphics g) {

**if**(!isStarted) {//isStarted初始化为false（暂停）

g.setColor(Color.***WHITE***);

g.setFont(**new** Font("arial",Font.***BOLD***,40));

g.drawString("Press Space to Start",250,300);

}

**if**(isFailed) {//在游戏结束时给出提示

g.setColor(Color.***RED***);

g.setFont(**new** Font("arial",Font.***BOLD***,40));

g.drawString("Failed:Press Space to Restart",200,300);

}

}

界面设计模块，每经过100ms，事件处理模块在修改对应的成员后，都会调用界面设计模块对界面进行重画。方法paintFixed主要用于画标题界面的固定规则和提示，并用于更新修改后的长度和分数属性；方法paintWarnning对分数进行判断，如果分数到达蜘蛛侠加速的临界处，会在游戏区中央画出蜘蛛侠加速的警告；方法paintVenom在游戏区画出毒液，由于毒液的移动方向会改变，为了视觉上的美观，根据不同的移动方向，我们画出不同的毒液头部进行区分，增加用户体验感。方法paintSpiderman和paintFood根据蜘蛛侠和食物的坐标，画出对应的内容。方法paintControl在监听到空格或者游戏结束时，事件处理模块改变了对应的成员（isStarted和isFailed），在游戏区中央画出提示。

**四、初始化模块**

MPanel() {

**this**.setLayout(**null**);

loadImages();//加载图片

initSnake();

**this**.setFocusable(**true**);//可以获取键盘的事件

**this**.addKeyListener(**this**); //使用implements KeyListener

//实现利用自身进行监听

timer.start();//时钟开始

loadBGM();//播放音乐

}//构造函数

**private** **void** initSnake() {//该方法用于初始化操作

len=3;

times=0;

score=0;//每次开始 分数清0

snakex[0]=100;

snakey[0]=240;

snakex[1]=75;

snakey[1]=240;

snakex[2]=50;

snakey[2]=240;

spiderman1\_x=825;

spiderman1\_y=715;

direction="R";//方向:R,L,U,D

foodx=25+25\*rand.nextInt(34);//rand.nextInt的范围是[0,34)

foody=215+25\*rand.nextInt(22);//活动范围是34\*22个25\*25的格子

food2x=25+25\*rand.nextInt(34);

food2y=215+25\*rand.nextInt(22);

food3x=25+25\*rand.nextInt(34);

food3y=215+25\*rand.nextInt(22);

}

初始化模块通过构造函数，实现初始化。方法initSnake中，将毒液的初始长度设置为3，坐标为游戏区的左上角，初始运动方向为向右。蜘蛛侠的位置坐标初始为游戏区的右下角，三种食物的坐标随机生成。

1. **异常处理的设计**

**1.在加载图片的方法中，通过catch (IOException e)捕获IO异常并打印。**

**private** **void** loadImages() {//加载图片

InputStream is;

**try** {//通过类加载器读入图片

//导入标题图片 is=getClass().getClassLoader().getResourceAsStream("images/duye3.jpg");

title=**new** ImageIcon(ImageIO.*read*(is));

//导入蛇的身体图片

is=getClass().getClassLoader().getResourceAsStream("images/body3.png");

body=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/up2.png");

up=**new** ImageIcon(ImageIO.*read*(is));

//导入蛇的四种不同形态的头部图片

is=getClass().getClassLoader().getResourceAsStream("images/down2.png");

down=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/right2.png");

right=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/left2.png");

left=**new** ImageIcon(ImageIO.*read*(is));

//导入食物的图片

//食物1:

is=getClass().getClassLoader().getResourceAsStream("images/food2.png");

food=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/food2.png");

food\_1=**new** ImageIcon(ImageIO.*read*(is));

//食物2:

is=getClass().getClassLoader().getResourceAsStream("images/food3.png");

food2=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/food3.png");

food\_2=**new** ImageIcon(ImageIO.*read*(is));

//食物3：

is=getClass().getClassLoader().getResourceAsStream("images/food5.png");

food3=**new** ImageIcon(ImageIO.*read*(is));

is=getClass().getClassLoader().getResourceAsStream("images/food5.png");

food\_3=**new** ImageIcon(ImageIO.*read*(is));

//导入蜘蛛侠图片

is=getClass().getClassLoader().getResourceAsStream("images/pp2.png");

spiderman1=**new** ImageIcon(ImageIO.*read*(is));

//蜘蛛侠左边挂绳图

is=getClass().getClassLoader().getResourceAsStream("images/spidermanL.jpg");

zhizhuxiaL=**new** ImageIcon(ImageIO.*read*(is));

//导入毒液图片

is=getClass().getClassLoader().getResourceAsStream("images/duyeH.png");

venom=**new** ImageIcon(ImageIO.*read*(is));

//蜘蛛侠VS右边的logo

is=getClass().getClassLoader().getResourceAsStream("images/spidermanH.png");

spidermanU=**new** ImageIcon(ImageIO.*read*(is));

} **catch** (IOException e) {

e.printStackTrace();

}

}

**2.在加载音乐的方法中，加入了异常处理并打印。**

**private** **void** loadBGM() {//加载音乐

**try** {

bgm=AudioSystem.*getClip*();//创建一段音乐

/\*下面一步实现加载文件 做成字节流

先找到自己当前的类 然后通过类加载器ClassLoader 将对应的内容加载上来\*/

InputStream is=**this**.getClass().getClassLoader().getResourceAsStream("music/bgmf.wav"); //注意只能导wav格式的

/\*将字节流转为音频字节流

\*getAudioInputStream：read enough data to determine whether they support the stream

\*BufferedInputStream可以把stream中的数据先读出一部分，放在缓存区中 够AudioInputStream做检查

\*解决了jar文件没有声音的情况\*/

AudioInputStream ais=AudioSystem.*getAudioInputStream*(**new** BufferedInputStream(is));

bgm.open(ais);//打开

//bgm.start();//播放

// bgm.loop(Clip.LOOP\_CONTINUOUSLY);//可以实现循环重复播放

} **catch** (LineUnavailableException e) {

e.printStackTrace();

} **catch** (UnsupportedAudioFileException e) {

e.printStackTrace();

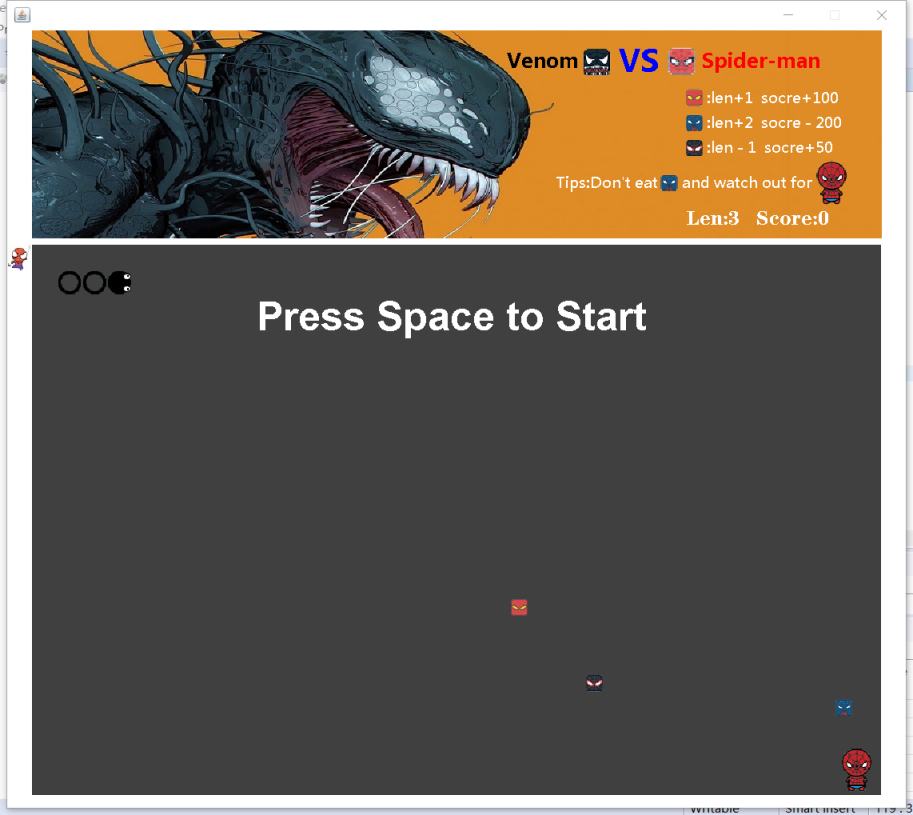
} **catch** (IOException e) {

e.printStackTrace();

}

1. **测试与结果**

1.打开游戏，游戏未开始时的界面图（暂停时音乐不播放）



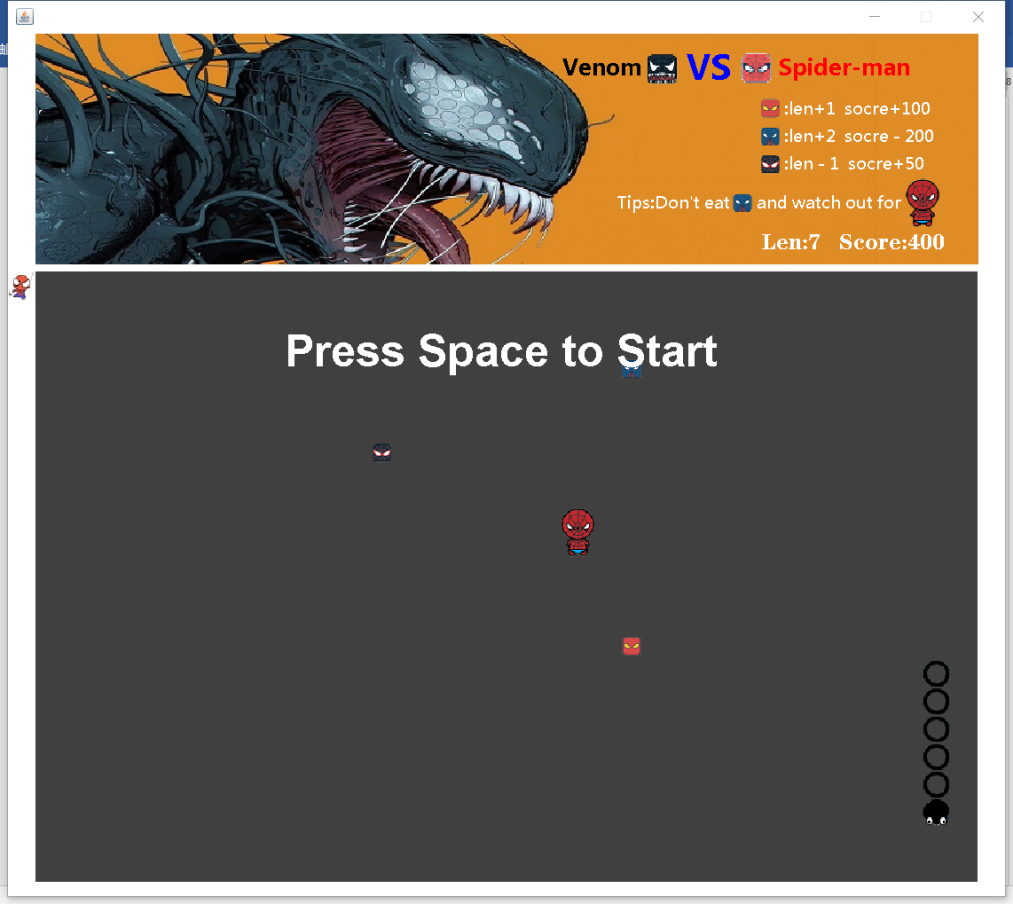
2.游戏开始，音乐响起，吃到一定的红色食物后，毒液身体变长，分数对应增多，蜘蛛侠向毒液的方向进行移动。



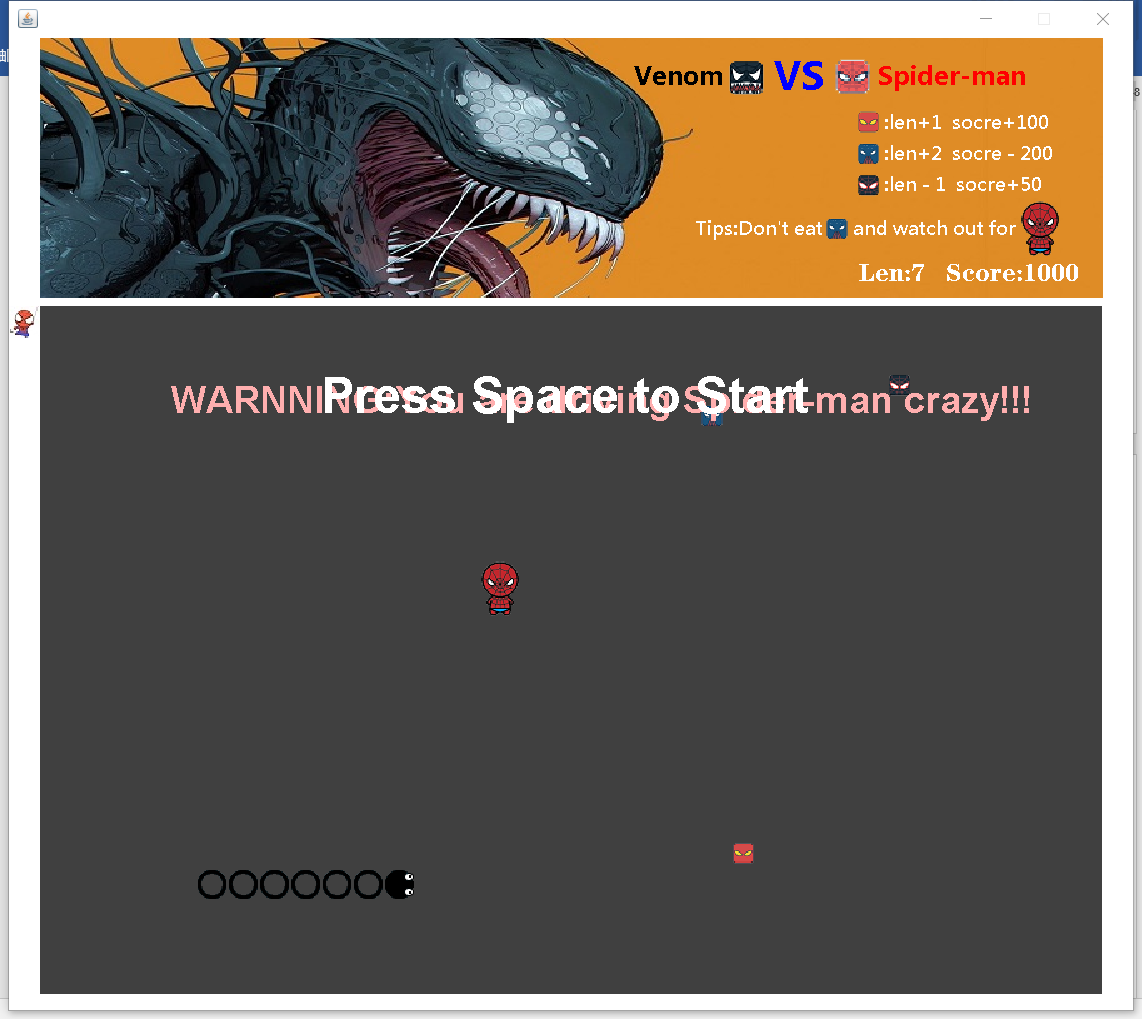
3.对比上一张图，毒液吃到了黑色的食物，身体变短1节，分数继续增大，到达了500分，是蜘蛛侠加速的临界分数，给出了蜘蛛侠加速的警告（由于要将游戏暂停进行截图，一部分被白色字挡住）



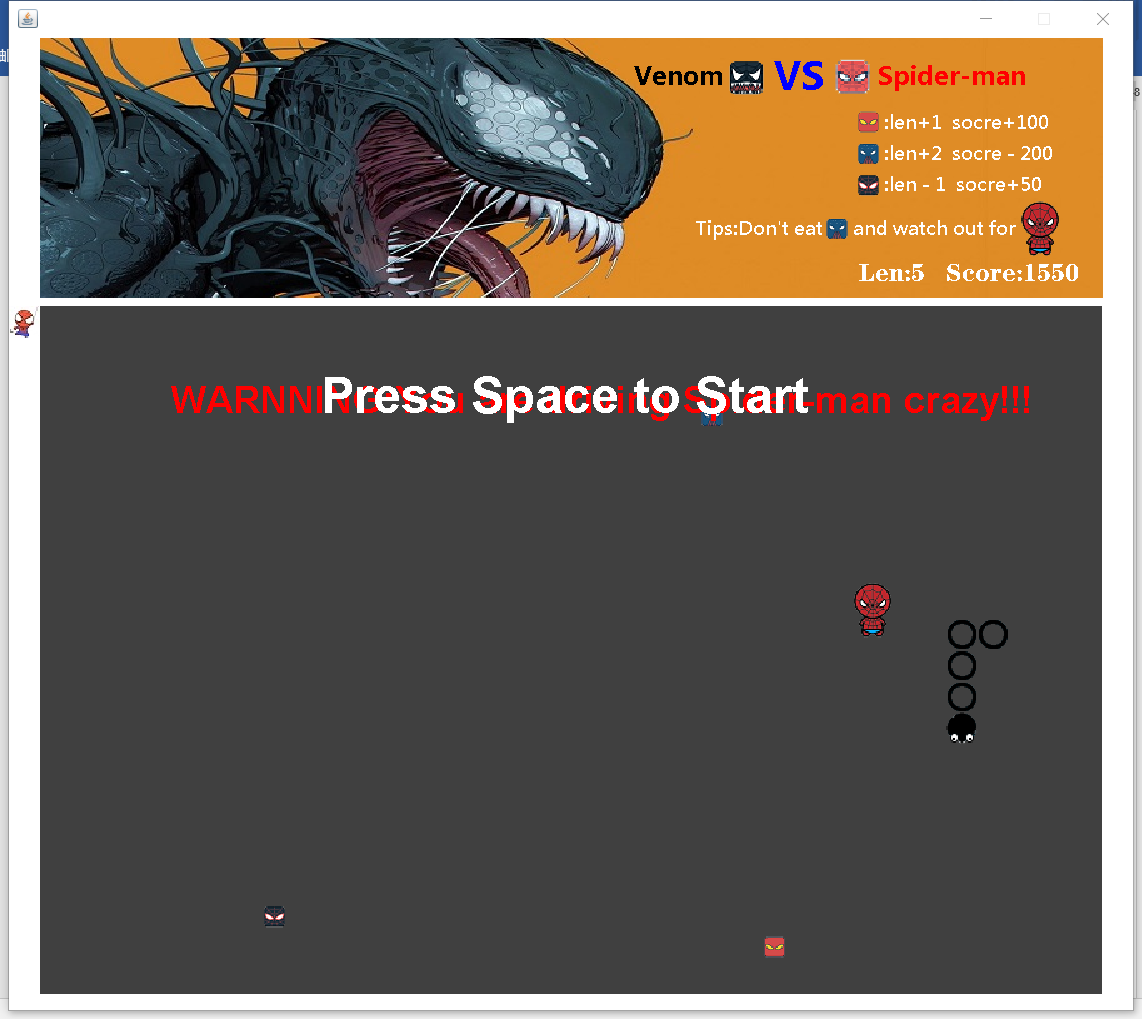
4.在吃到蓝色食物后，身体变长，分数减少



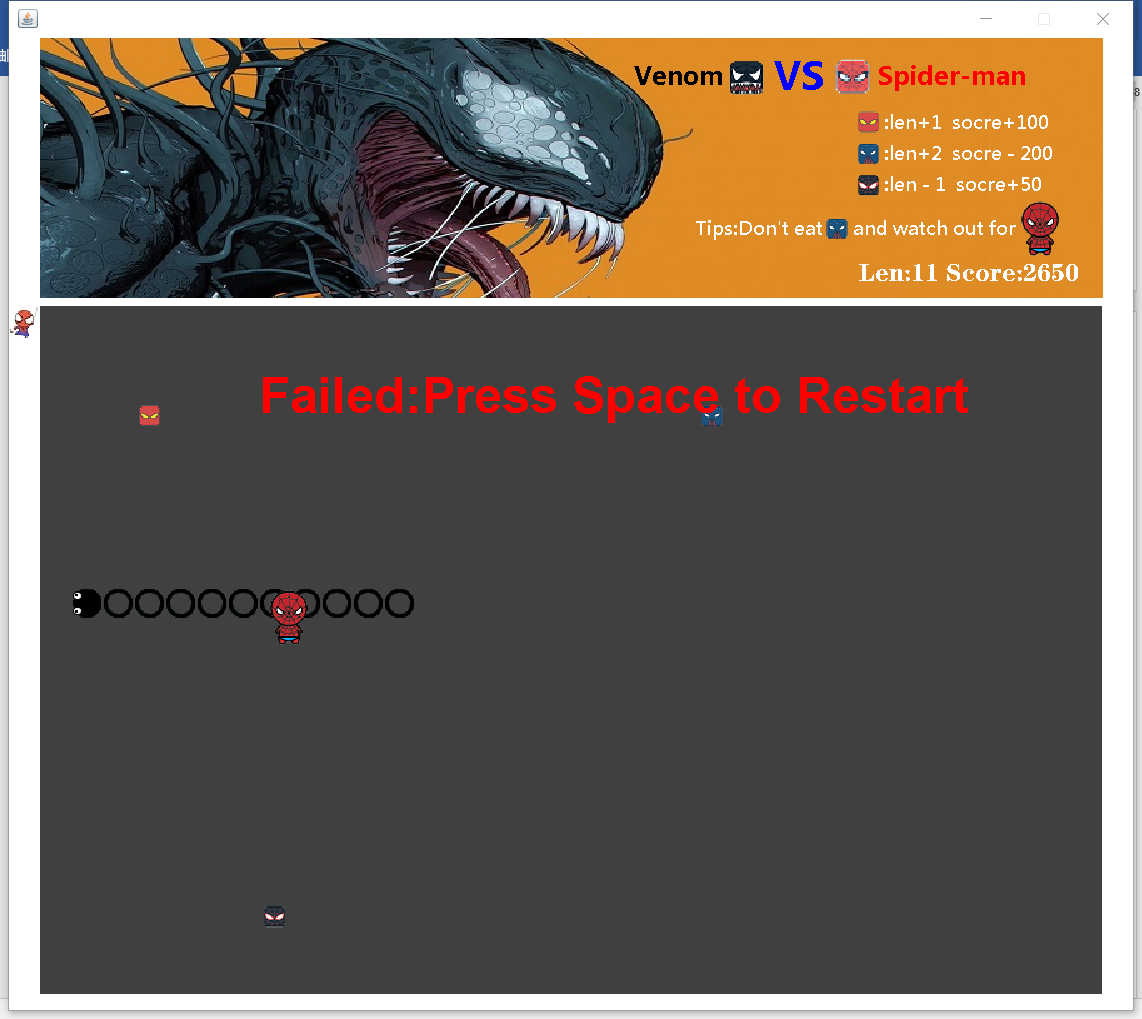
5.分数到达1000，给出蜘蛛侠第二次加速警告，蜘蛛侠速度继续加快。



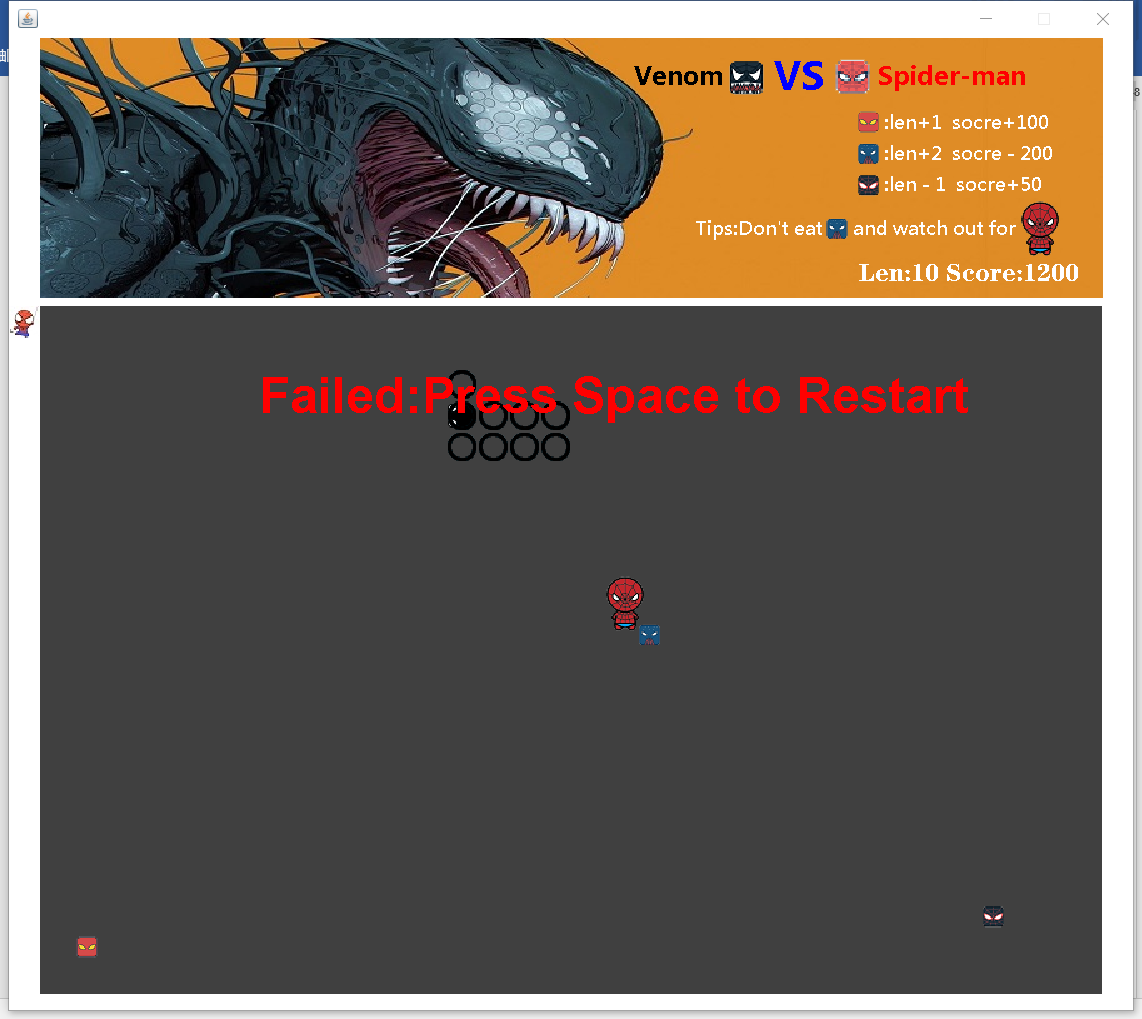
6.分数达到1500-1550，给出蜘蛛侠第三次加速警告，蜘蛛侠速度继续加快



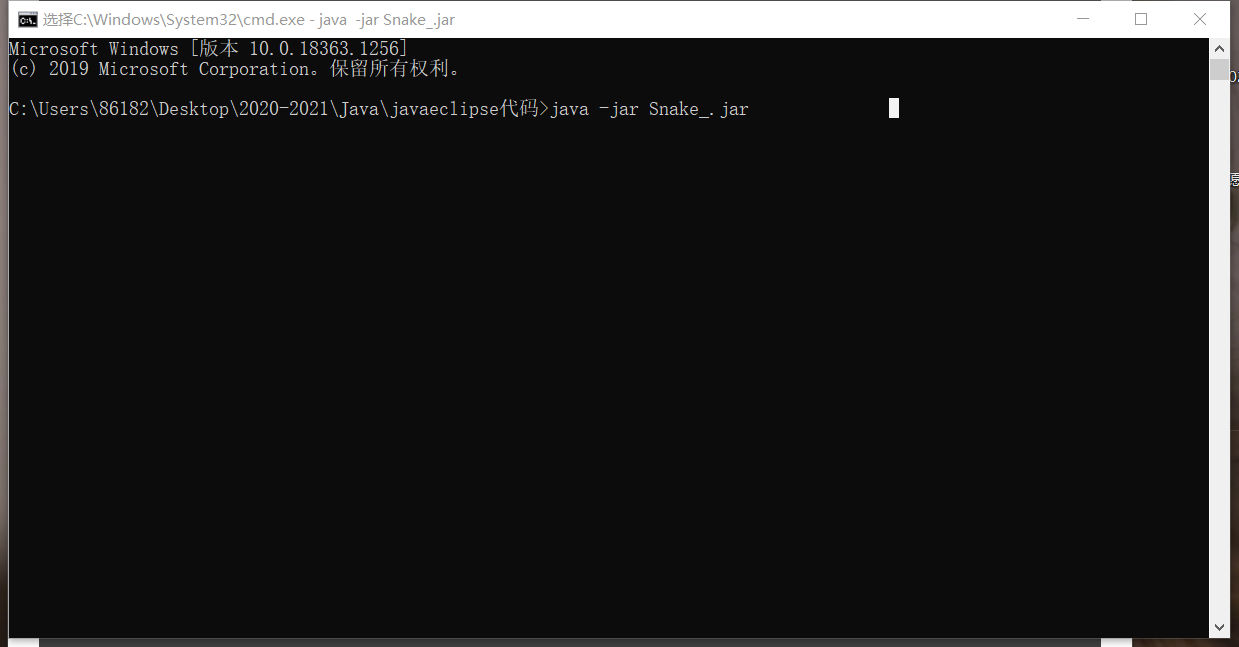
7.随着分数的增大，蜘蛛侠的速度不断加快，最后被蜘蛛侠抓到，游戏结束。



8.或者由于毒液的头部碰到毒液的身体 游戏结束



9.打包成jar文件运行



1. **总结**

**优点：**

1. 对界面做了较为美观的设计，并导入了背景音乐，很大程度上提高了用户的游戏体验感。
2. 引入了较为新颖的风格和元素，并通过增加食物的属性，丰富了游戏的玩法和策略。
3. 标注了大量的提示和说明，使得游戏的上手难度小，游戏使用感强。
4. 打包成jar文件，方面用户使用。

**不足：**

1. 游戏的记录没有得到保存，没有办法得到游戏的历史最高分数等数据，可以通过连接数据库的方式，对游戏数据进行保存。
2. 由于游戏所使用到的大量图片素材都是从网上收集后进行处理得到的，所以很多图片素材的清晰度不高，在图片素材的筛选和处理方面的能力还有待提高。
3. **参考文献**

<https://www.bilibili.com/video/BV1TJ411L742?from=search&seid=15500087244987279619>

<https://www.bilibili.com/video/BV1vz4y1k7JQ?from=search&seid=15500087244987279619>

https://www.bilibili.com/video/BV1db411w76i/?spm\_id\_from=333.788.videocard.4