高级程序设计语言程序设计语言

班级：软件工程三班

学号：

姓名： 王镜鑫

目录

项目简介和摘要-----------------------------------------3

需求分析-----------------------------------------------3

系统设计-----------------------------------------------3

系统实现-----------------------------------------------4

调试给错-----------------------------------------------5

美工素材-----------------------------------------------5

参考资料------------------------------------------------5

收获体会------------------------------------------------5

关键代码------------------------------------------------5

项目简介

刚进入的时候会有一个界面，为地图编辑器。可以使用此编辑器进行地图编辑，地图编辑器的内容包括：关卡。向左箭头，带有金币的砖块，带有花朵的砖块带有蘑菇的砖块，带有星星的砖块，普通砖块，向左运动的板栗仔，向右运动的板栗仔，向左运动的乌龟，向右运动的乌龟，金币。带有食人花的管道。普通管道，洞。向右的箭头。橡皮擦，可以使用鼠标点击图标然后拖动到面板上点击面板进行地图编辑。橡皮擦可以擦除已经建立好的模型。部署完地图之后可以选择下一关进行下一个关卡的编辑，也可以点击开始游戏开始游戏。游戏开始后从编辑的第一关卡开始进行闯关，人物可以移动通过ad键进行控制，可以跳跃，通过k控制，跳的时候可以跳到管子和砖块上面。人物有两种状态。大马里奥和小马里奥。小玛丽奥可以撞普通的砖块或者带有包含物的砖块使得砖块可以向上稍微移动。砖块上的一些包含物也会随着砖块移动。大马里奥可以顶破普通砖块。怪物分为三种，分别为板栗仔，乌龟和食人花。马里奥可以通过跳跃的方式踩死怪物。板栗仔在被踩的时候会变扁，乌龟被踩的时候。走动状态会变成龟壳状态，龟壳状态被碰到可以变成跑动的龟壳状态，跑动的龟壳可以杀死马里奥。板栗仔和其他的乌龟。运动的龟壳在运动的时候被马里奥踩到会变成静止的龟壳，食人花长在管道中。会定时出现对管道上方的物体进行攻击。当马里奥踩在管道上的时候不会出现。还有三种物体是包含在砖块中的。分别是星星。蘑菇。花朵。马里奥自下向上顶砖块之后砖块上方会生长出相应的植物星星和蘑菇会向右方向行走。花朵会在原地，。马里奥可以通过触碰的方式吃掉植物，不同植物有不同的加成效果，其中，吃掉蘑菇之后会变成大马里奥。吃掉星星之后会变成无敌状态。吃掉花朵之后会有发射子弹的技能。任何物品。尤其是可移动物。包括子弹，在碰到洞之后会掉落到洞中人物掉落之后会损失一命，人物一共有五条生命。每次正面碰到乌龟或者板栗仔，或者掉落到洞中之后便会损失一条生命，每次损失生命则该关卡从头开始当五条生命全部损失之后便会到gameover状态。当马里奥走到地图的最后一个模型之后的位置的时候说明本关通过。本关通过时会有马里奥跳下拉动旗帜旗帜拉倒底端的时候会向右跑到城堡位置。跑到城堡位置即属于本关卡已经通过。则消除所有的加成状态转到下一关卡。最后通关所有的关卡即为game ends.跳跃的时候有重力效应，降落的会越来越快游戏界面上方会有剩余生命，当前时间为0的时候会损失一命，还有计分系统。当玩家杀死怪物，或者吃掉某种可生长物，或者过关的时候都会获取相应的分数加成。分数显示在面板上方。吃掉金币会有金币数量统计，统计结果在生命右边。本项目的亮点在于应用ioc技术的地图编辑器和精美的人物模型。

需求分析

人物跳跃的重力，条约落下的时候碰到其他的硬物可以停止下落。踩死怪物。怪物死亡方式不同展现的画面不同。吃东西，有属性加成。大马里奥顶破砖块。小玛丽奥顶砖块砖块可跟随移动。砖块上方的东西可以跟随移动。地图编辑功能。声音功能。人物胜利拉旗进城堡。

系统设计

本项目共有20个公共类。5个接口。分成三种。分别是控制类。模型类。工具类。功能方面有两大块。分别是地图编辑器和正式游戏。地图编辑器部分使用了spring框架的ioc技术。关于三种类。控制类中有两个类分别为Main和Control作用分别为控制地图编辑，地图编辑的思路如下：玩家点击图标之后鼠标的状态变成点击的图标的状态值，本类中有一个map键值分别为Integer和model鼠标移动到某位置点击之后会使integer加一。构造出相应的model然后put到map中如果用户点击的是橡皮擦。那么会计算哪一个类在橡皮擦点击的位置，并且把相应的位置的model设为忽略。最后解析这个map构造xml文件和保存着各个类的数量的一个properties文件Control类通过解析xml和properties文件解析关卡信息还原用户编辑的地图。还有一个全局的properties为game.properties保存着关卡数目。在点击开始游戏之后开始运行Control类中的work方法。Work方法的作用是初始化整个游戏的完整页面读入xml中的内容实例化对象存到容器中。然后启动paintThread线程画出面板，启动其他的必要线程进行工作，根据用户的操作对容器中的对象的一些参数进行改变呈现不同的视觉效果。

系统实现

本项目由于需要实现用户自由设计地图，所以应该尽量降低耦合度，从全局的角度出发，对类的设计应该分为实物模型，统筹控制，工具，抽象出来的接口。接下来一一介绍：统筹控制包含Main和Control在上一部分已经介绍。接口被抽象出来以下几种：Dangerous类所有的可以杀死主角的实物模型类应该去实现该接口，Flint类，砖块和管子应该去实现该接口。Growable接口，所有的可以被马里奥从砖块中顶出的实物模型应该实现该接口。Kill接口，所有的可以伤害到别的实物模型类的类都应该实现该接口。Moveable接口，所有的可以移动的物体都应该实现该接口。实物模型类一共有12个。Badflower类是食人花类，实现Dangerous接口，Bullet类是子弹类，实现Moveable和Kill接口。Flower类是吃了以后可以发射子弹的类，实现了Growable接口，Hole类是地面上存在的洞类。没有去实现任何接口，Mario类，是主角类，实现了Moveable，Kill接口。Money类，因为可以直接被主角吃掉并且在砖块上被顶出之后不需要生长移动过程所以不实现任何接口。Monster类是板栗仔，实现了Dangerous，Moveable接口Mushroom类是吃了以后变大的蘑菇类实现了Growable和Moveable接口Pipe是管道类实现了Flint接口。Star是吃了以后变成无敌状态的星星，实现了Growable和Moveable接口Turtle是乌龟类，实现了Dangerous,Kill,Moveable接口。Wall是砖块类，实现了Flint方法。还有6个工具类，其中的方法和字段大部分是静态的和final的，ApplicationUtil类可以通过传入的关卡值去加载spring上下文。为程序提供对象实例，CrashType定义了一些物体之间的碰撞类型的常量。ImageTool类中包含了程序用到的所有的图片以及为了克服延迟加载而写的事先加载所有图片的方法Null类是一个Growable类的空实现，因为在构造砖块的时候定义构造方法里面应当传入所包含的可生长物，而在使用spring框架进行实例化得时候不允许出现null.本人又极不愿意在写另外一种构造方法，所以索性构造一个Growable接口的空实现类，通过传出特殊的Type值进行识别，Property类，用于解析配置文件。获取数据，SoundTool类包含所有的使用到的音乐，以及静态的播放音乐的方法。

调试给错

在实现的过程中出现了很多错误。比如声音播放问题和人物碰撞检测的问题等。不过最后解决的还算满意。

美工素材

本项目是一个人写的，代码和图片美工都是自己实现的。由于互联网上找不到相关素材，所以本人现学的ps，通过录制游戏中的人物动作分帧截图，使用抠图等技术自己做的图。

参考资料

完全原创。没有参考任何人的类似项目。

收获体会

要有统筹整个项目的意识。对整个项目有总体的把握。类的编写应当事先分好类，分清楚每种类的任务。最大化解耦，以免改动的时候涉及到过多的地方。分清楚每个类的任务。合理设计避免冗余。

关键代码

碰撞检测部分代码：

**public** **int** getCrashType(**int** down,**int** direction,Rectangle rec1,Rectangle rec2)//rec1为wall，rec2为Mario 获取撞击类型

{

//rec1是硬物。rec2是移动物

**if**(die)**return** CrashType.***NO\_CRASH***;

**int** rec1X=(rec1.x+rec1.x+rec1.width)>>1;

**int** rec1Y=(rec1.y+rec1.y+rec1.height)>>1;

**int** rec2X=(rec2.x+rec2.x+rec2.width)>>1;

**int** rec2Y=(rec2.y+rec2.y+rec2.height)>>1;

**int** width=rec1.width+rec2.width;

**int** hight=rec1.height+rec2.height;

**if**(rec2Y>=rec1Y)

{

**if**(rec1X>=rec2X)

{

**if**(down!=1||(rec1X-rec2X)/((**double**)width)>((rec2Y-rec1Y)/(**double**)hight)+CrashType.***POINT***)

**return** CrashType.***WALL\_L***;**else**

{

**if**(control.getMario().isCanWork())work();

control.getMario().down();

**return** CrashType.***WALL\_D***;

}

}**else**

{

**if**(down!=1||(rec2X-rec1X)/((**double**)width)>((rec2Y-rec1Y)/(**double**)hight)+CrashType.***POINT***)

**return** CrashType.***WALL\_R***;**else**

{

**if**(control.getMario().isCanWork())work();

control.getMario().down();

**return** CrashType.***WALL\_D***;

}

}

}**else**

{

**if**(rec1X>=rec2X)

{

**if**((rec1X-rec2X)/((**double**)width)>=((rec1Y-rec2Y)/(**double**)hight)+CrashType.***POINT***)

**return** CrashType.***WALL\_L***;**else**

{

**return** CrashType.***WALL\_U***;

}

}**else**

{

**if**((rec2X-rec1X)/((**double**)width)>=((rec1Y-rec2Y)/(**double**)hight)+CrashType.***POINT***)**return** CrashType.***WALL\_R***;**else** **return** CrashType.***WALL\_U***;

}

}

}

马里奥跳跃代码：.

**private** **class** JumpThread **extends** Thread//跳的线程

{

**private** **int** n=jumpHight;

**public** **void** run()

{

**if**(down==-1)**return**;

**if**(downThread!=**null**)

{

downThread.~~stop~~();

downThread=**null**;

}

**if**(jumpThread!=**null**)

{

jumpThread.~~stop~~();

jumpThread=**null**;

}

jumpThread=**this**;

//while(!Control.isALL\_START()){try{sleep(Control.TIME);} catch (InterruptedException e){}}

SoundTool.*play*(SoundTool.*jumpSound*);

**try**

{

**int** site=locaY;

**double** count=Math.*sqrt*((2\*0.085\*n));

**for**(**int** i=site;count>0;i-=(count-=0.1))//向上跳的时候改变状态

{

**if**(getCrashType()&&(crashType==CrashType.***WALL\_D***||crashType==CrashType.***WUWL***||crashType==CrashType.***WUWR***))//如果发现从下撞击了硬物则跳出向上的过程改为向下

{

down=-1;

**break**;

}

locaY=i;

*sleep*(10);

down=1;

}

**new** Down().start();

}**catch**(Exception e)

{

e.printStackTrace();

}

down=0;

}

}

马里奥降落代码：

**private** **class** Down **extends** Thread//二类下落线程 //控制最终下落的线程如果在正常下落时由于碰撞被打断则启动该线程监视是否需要再次落下

{

**public** **void** run()

{

**if**(down==1)**return**;

//while(!Control.isALL\_START()){try{sleep(Control.TIME);} catch (InterruptedException e){}}

**if**(downThread!=**null**)

{

downThread.~~stop~~();

}

downThread=**this**;

// System.out.println("enter down!");

// while(down!=-1&&locaY!=control.getCutLine())//没有落地面上一直在循环

**while**(locaY<control.getCutLine())

{

**boolean** flag=**false**;//

**int** site=locaY;

// System.out.println(crashType+" "+down+" "+canWork);

**if**((getCrashType()&&(crashType==CrashType.***NO\_CRASH***)&&(down!=1))||canWork==**false**)//多线程重要判断应该靠紧

{

flag=**true**;//已经下落

**double** count=1;

**for**(**int** i=site;i<=control.getCutLine();i+=(count+=0.1))

{

down=-1;

locaY=i;

downSpeed=count;

// System.out.println(i+" "+control.getCutLine());

**if**(getCrashType()&&(crashType==CrashType.***WALL\_U***||crashType==CrashType.***PIPE\_U***||crashType==CrashType.***WUWL***||crashType==CrashType.***WUWR***))//如果在某个硬物的上方、启动二类下落线程且退出本线程

{

down=0;//落道硬物上面则运动状态为0

downSpeed=0;

downThread=**null**;

**new** Down().start();

setCanWork(**true**);

**return**;

}

**try**

{

*sleep*(10);

} **catch** (InterruptedException e)

{

down=0;

downThread=**null**;

downSpeed=1;

setCanWork(**true**);

**new** Down().start();

}

}

locaY=control.getCutLine();

}

**if**(flag)//在该线程中如果已经下落则跳出

{

**if**(!control.getMario().isDownDie())

**for**(**int** j=0;j<control.getHoles().size();j++)

{

**if**(control.getHoles().get(j).canPaint())

control.getHoles().get(j).DownDie(control.getMario());

}

**if**(!control.getMario().isDownDie())downSpeed=1;

down=0;

downThread=**null**;

setCanWork(**true**);

**return**;

}

}

}