

《Java程序设计实践》报告

专业 软件工程 指导老师 李旺、陈恺萌、郭晓曦

项 目 名 称： I Don’t Wanna Be The Bugger

组长： 游嘉升（201921122050） 自评等级 1 成绩

组员： 吴旭婷（201921122034） 自评等级 1 成绩

组员： 自评等级 成绩

组员： 自评等级 成绩

评阅：

2021年1月22日

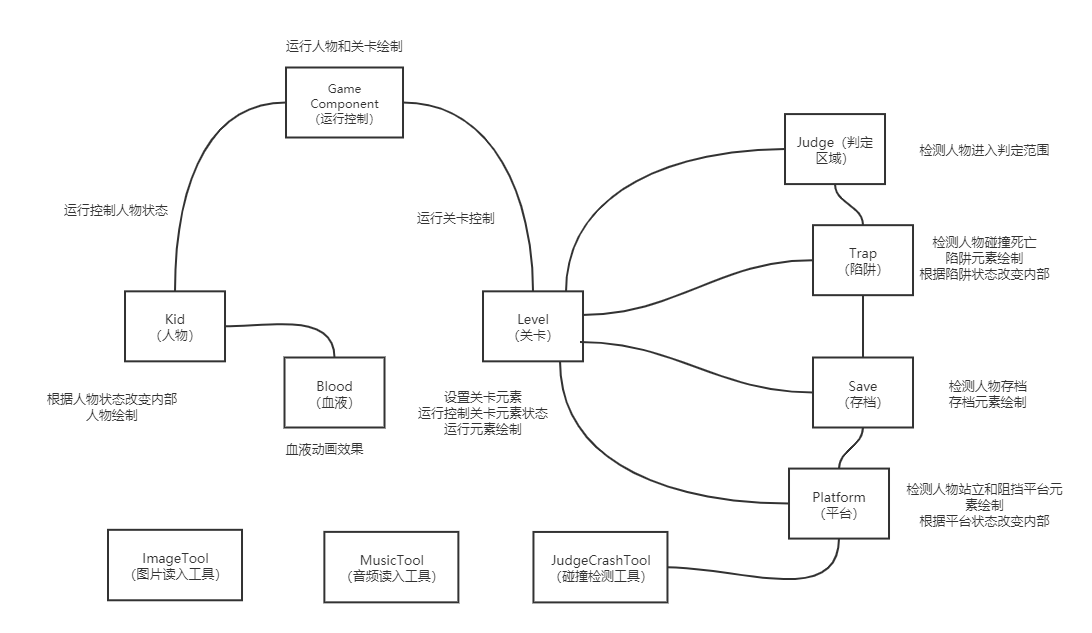
# 一、课程设计项目简单描述

在美国早期游戏业比较宁静的时代，I Wanna Be The Guy 作为一款难度较高的冒险类游戏，在当年很多人玩过。玩家将体验一个年轻、懵懂的孩子为了追寻理想中的角色所经历的过程。变幻莫测的完美剧情只是这项无限创意中的一部分，令人沮丧的跳跃和射击的挑战会使你对这个主角小孩产生敬意。

I Wanna出名后，一些地方开始举行I Wanna 大会，并利用GM8和MMF2等工具制作I Wanna。这些可在百度I\_wanna\_be\_the\_guy吧或日本的wiki或维基百科I Wanna专题上找到并下载。除此之外，在PIXIV上也有I Wanna相关的作品。

该课程设计项目属于I Wanna Be The Guy游戏同人作品，构建了人物、关卡、关卡元素等模型；获取了相应的素材图片与音乐，丰富了游戏的内容；实现了基本的人物移动跳跃死亡，关卡和各种多种机关的创建，关卡切换和重开，存档等多种功能。基于java语言封装使关卡的布局更加方便，精心设计了四个关卡使游戏更加具有乐趣。

# 二、需求分析



# 三、系统设计

1.碰撞检测算法



该算法为较为重要的算法，所有的关卡元素以及人物都继承自Thing类，judgeCrash方法通过判断传入两个类的图片是否重叠进而返回是否碰撞。

矩形重叠判断公式

X轴：[（x1+w1/2）-（x2+w2/2）]<（w1+w2）/2

Y轴：[（y1+h1/2）-（y2+h2/2）]<（h1+h2）/2

当判断公式返回真值时说明矩形重叠。

2.GameComponent类

1）Timer类和ActionListener接口

通过ActionListener监听器和Timer结合定时调用Kid类的run方法和Level类中的scan方法，使画面每隔固定的一段时间改变并调用repaint刷新画面显示从而达到动效。

2）KeyListener接口

使用KeyListenr对键盘输入监听，并对相应键的KeyPress调用Kid或者Level的方法对其的状态进行改变。

3）窗体的绘制

GameComponent继承于JComponent重写了paintComponent方法，调用Kid类和Level类的draw方法绘制出当前的画面。

4）重新游戏

当监听到键盘键入‘R’或‘r’时调用restart方法，进而调用Kid对象的reset和Level的reset重置关卡。

3.Level类

1）Level类的数据域



2）关卡的设置与重置

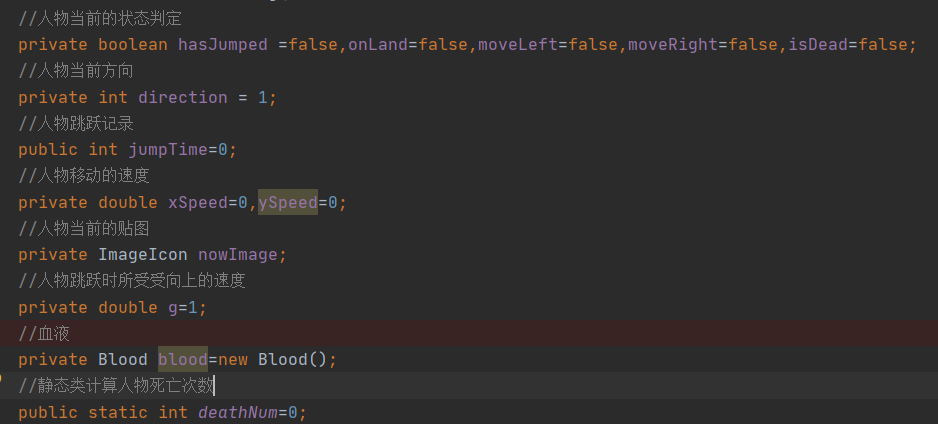
Level中的reset具有关卡设置与重置的功能，将对应关卡的元素载入对应的ArrayList中，在draw类被调用时会根据载入的元素的坐标和贴图绘制出所有载入的元素，从而形成关卡，重置亦是如此。

3）关卡元素的检测和动态变化

Scan方法会对所有的关卡元素进行扫描，如果满足触发条件，则调用关卡元素的触发内容。还可以定义关卡进行时的动态变化（触发后某个机关才会出现的某个机关需要在这里设置）。

4.Kid类

1）Kid类中的数据域



2）Kid类状态的改变

通过jump，walk，fall，idle，die方法，使Kid对象进入某种状态，并改变Kid对象的数据域。

3）Kid类的坐标根据状态的改变

人物根据坐标数据定位，在run方法中定义了当Kid处于某种状态时，实际的坐标根据随之改变。由于GameComponent的定时调用，从而达到人物状态改变的效果。

4）Kid类的绘画

Draw方法根据人物当前的状态载入对应的图片，并显示在窗体上。

5）Kid的重置

当Kid经过死亡状态后，触发重开游戏后调用reset方法将Kid设置为初始状态。

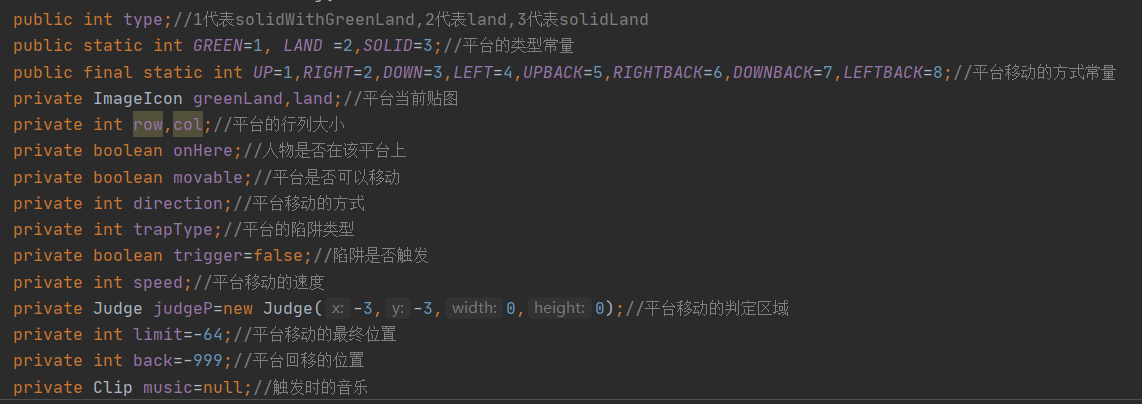
6）Blood类

在Kid进入死亡状态时，将人物死亡坐标传入Blood类，初始化500个血液坐标，并使用随机数方法随机出500个横纵轴速度，使500个血液坐标改变，最后调用绘制在屏幕上。

5.Platform，Trap，Judge，Save类

（四类具有大部分相似的数据域与方法，以Platform为例分析）

1）Platform类的数据域



2）Platform类的站立与阻挡

judgeKidCrash方法通过对Kid对象和Platform对象当前坐标进行一系列碰撞的判定，并对站立与阻挡两种不同的状态对Kid对象内部数据进行改变，从而达到站立和阻挡效果。其余三类均有通过碰撞检测实现的一些功能。

3）Platform类的移动

Platform对象在Level类中检测是否触发自身判定区域，若已触发判定区域，则调用move方法对Platform的坐标进行改变。由于GameLevel会定时调用Level类的检测方法，从而达到移动的效果。

4）Platform的绘制

Draw方法根据平台当前的状态载入对应的图片，并显示在窗体上。

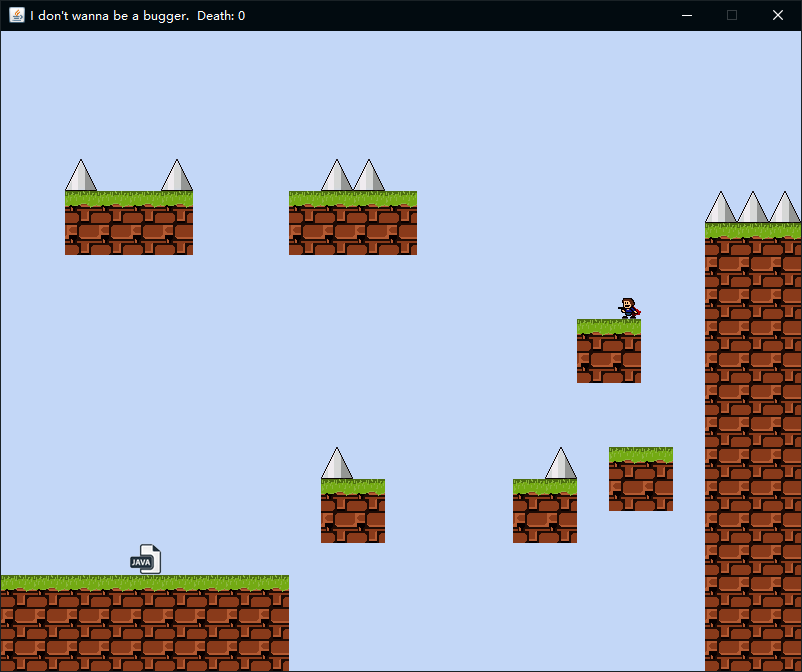
6.ImageTool和MusicTool

在GameComponent创建时就将图片数据和音乐数据载入内存中，提高读写速度的同时也方便之后的读取。

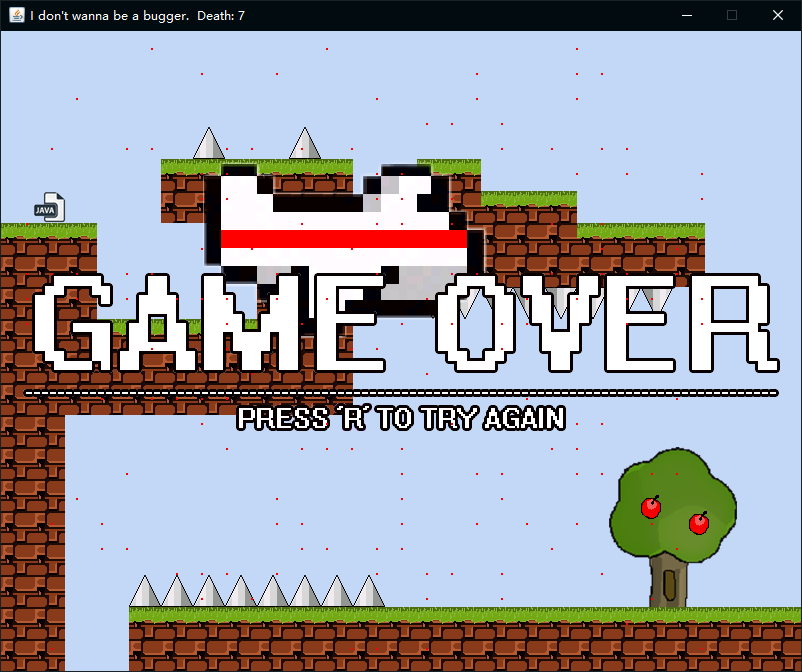
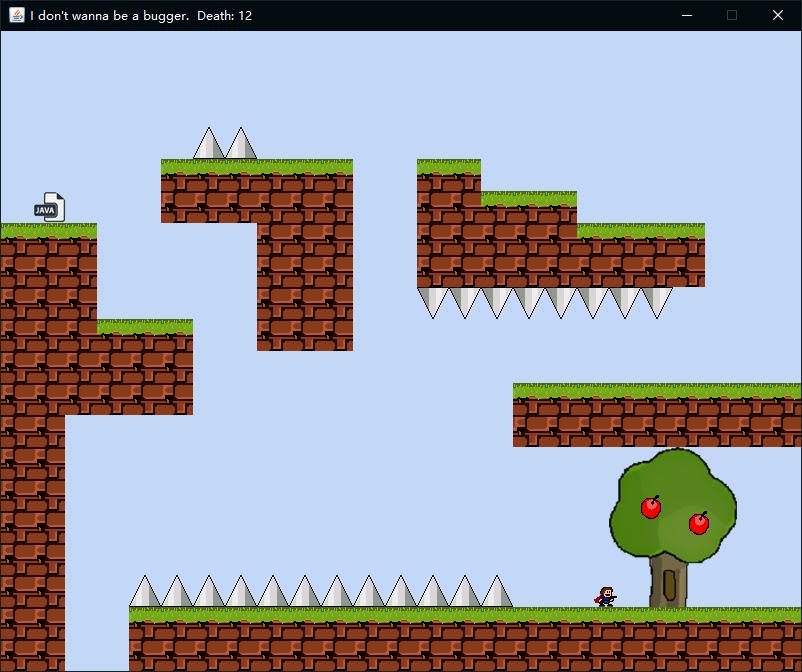
# 四、效果及存在问题

1.运行效果

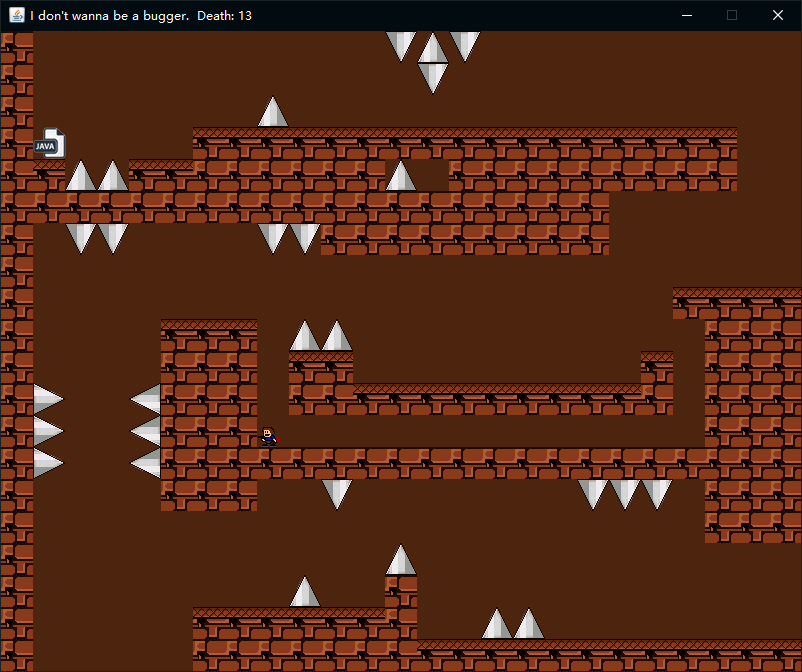
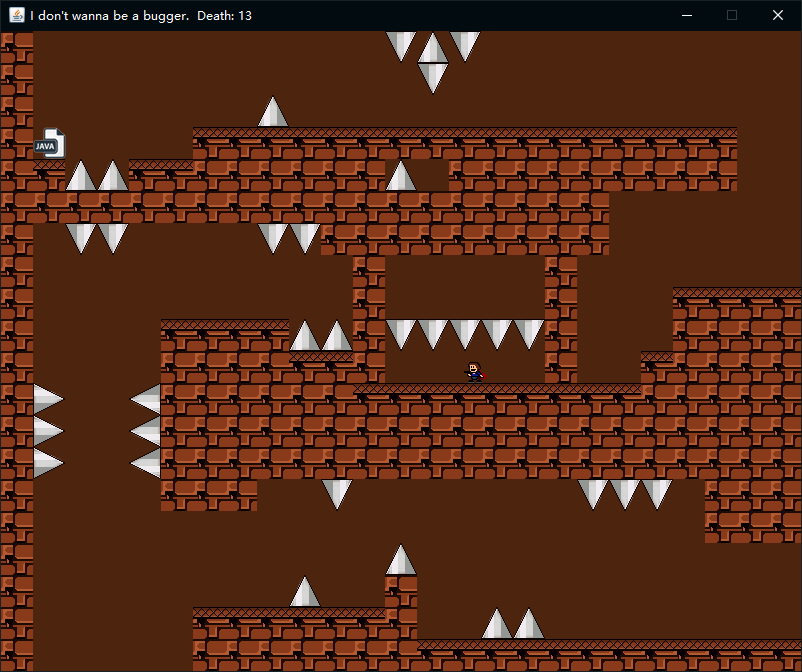
Level 1

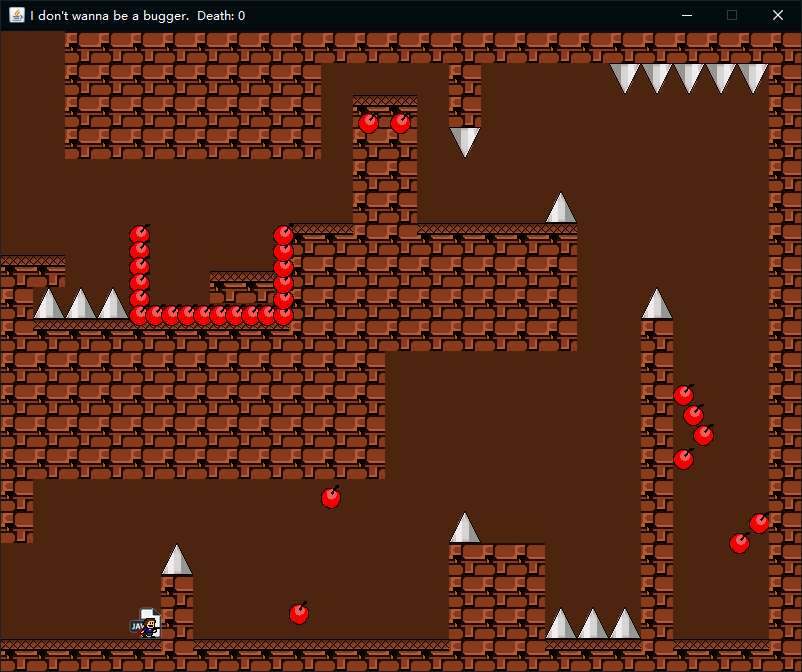
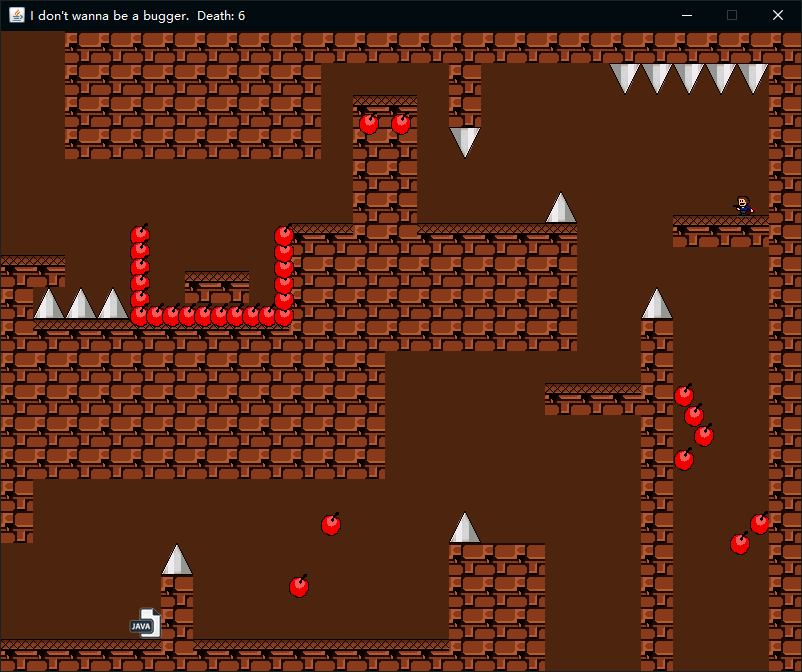
Level 2



Level 3

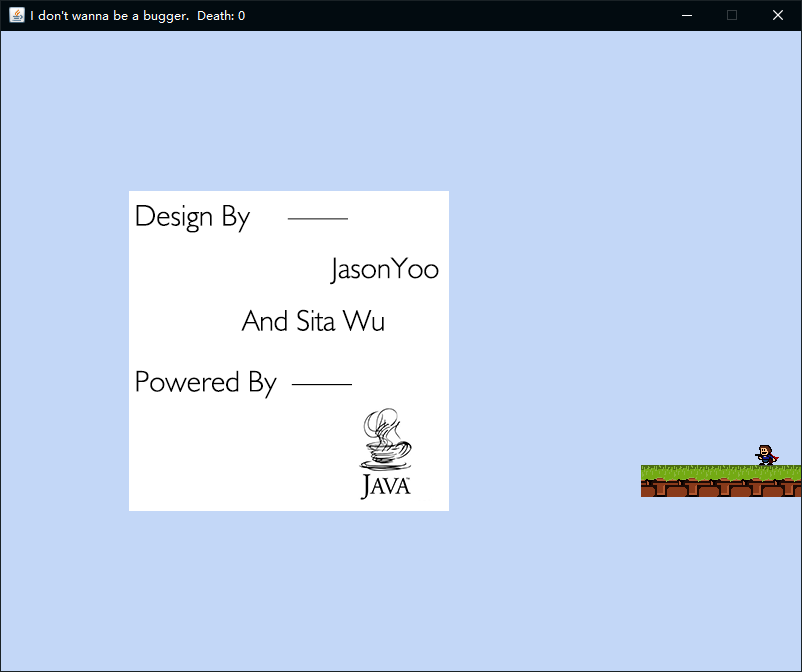
Level 4

Level 5



Level 0(author page)



2.存在的问题

1. 碰撞算法精度问题

由于能力暂时的有限，所有的碰撞检测都只采取了矩形检测重叠的方法来判断是否碰撞，导致碰撞判定的精度并不高（会导致还没触碰机关元素到就死亡的现象）。

预期解决方法：引入检测像素碰撞方法，在一些需要精度比较高的地方使用。

1. 人物物理引擎的死板

原作的物理引擎可以根据按下键盘时的停留时间对人物跳跃的高度和速度做相应的处理，本作并没有做到比较死板，会造成有时候关卡的设计难度增大（因为需要人物本身是否能够通过）。

预期解决方法：监听键盘按键时间，并对其做出不同的反应。

1. 陷阱形式比较单一

只设计了最基本的移动陷阱方法，并没有设计出更多的陷阱方式（虽然已经可以衍生做到很多不同的陷阱效果），游玩过久可能会觉得枯燥。

预期解决方法：多设计一些不同的陷阱类型并应用。

1. 关卡放置方面过于底层，不是很好操作

关于关卡的放置还是处于代码层，新设计的关卡都需要重新写进代码里才可以显示，参数过于繁琐，不够方便。

预期解决方法：制作一个具有交互界面的游戏制作器，可以直接生成数据并传入内部程序运行。

# 五、心得体会

游嘉升：

经过本次课程设计后，在团队协作方面更加知道小组工作时配合的重要性，很多时候小组如果没有配合好会出现不协调的现象。比如某个地方需要另一个成员的某段代码，而另个成员还没写就会导致工程滞留，这是很严重的问题。还有成员之间的沟通需要相互理解，沟通是建立在理解的基础上的，如果互相不理解就容易造成更严重的误会。身为组长还需要做好分工协调的工作，提前将任务明细布置。

在技术层面，更加体会到MVC和封装的优越性。根据MVC模式构建的代码比较清晰明了，各个模块各司其事，相互使用又互不影响。封装后的代码若是想要修改很快就可以找得到其作用的位置。所以在写代码前需要设计框架，再将其填充。这次课设算是累计了不少有用的经验。

吴旭婷：

最大的体会是分工合作的重要性，这也是和之前所做课设的一大不同。明确的分工能避免不必要的争执和推脱，所以从一开始就需要和队友做好沟通，在做的过程中也需要不断磨合。在合作的时候曾起过小纠纷，让我误以为队友对于我设计的机关不满，好在最后解释清楚，也让我在后面制作的时候参考了更多元素，设计得更好。在这期间我也了解到有些组合在合作方面就做的很差，也没有好好沟通，导致最后心中不满，觉得还不如自己完成就好。相比之下我只想说，队友好给力！！！

# 六、参考文献

[1][美] 凯.S.霍斯特曼（Cay S. Horstmann）.Java核心技术·卷 I（原书第10版）[M].机械工业出版社