# 第六周工作报告

这周主要完成了以下工作：

1、实现了趣味模式，并且玩家能够保存自己的地图。

2、在原有的游戏内容中添加了以下新的趣味元素：对抗模式

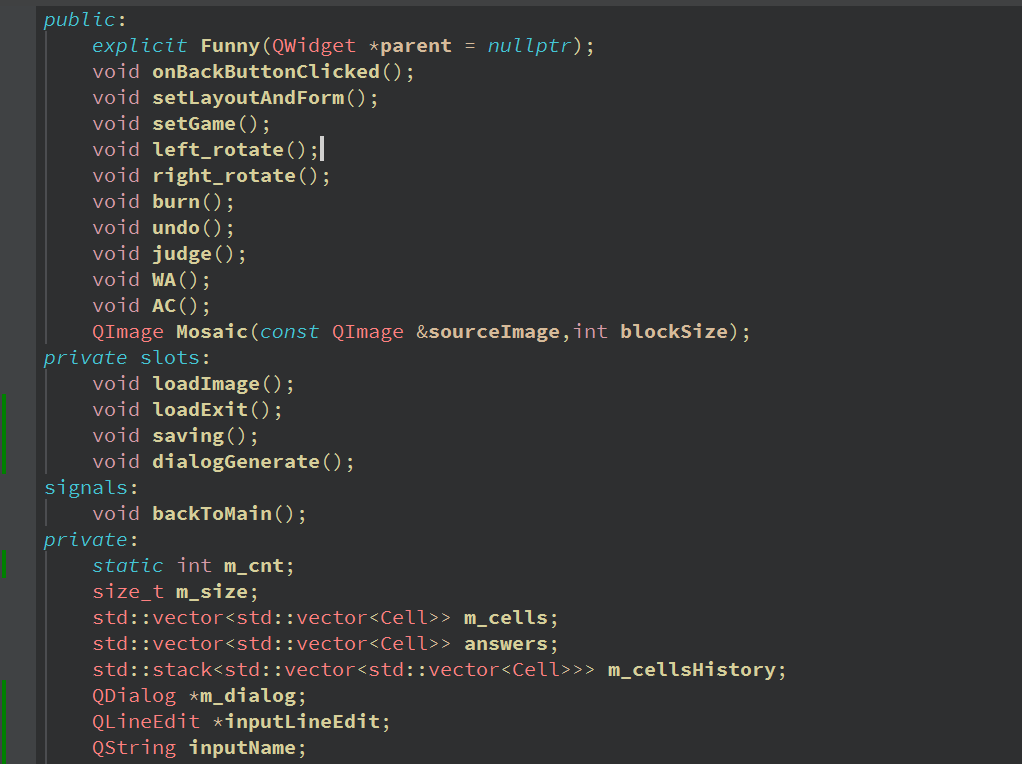
3、对随机模式的难度选择进行了修改和实现

## 趣味模式的实现

趣味模式的内容是：玩家通过导入自己喜欢的图片，系统会生成对应的地图。

玩家也可以将目前的地图进行命名并且存储，这样可以保存当前游戏的进度。

Funny类的设计



关键的核心在于Mosaic函数，它可以实现将导入的图片转换为马赛克分格，进而利用随机生成地图，供给玩家进行游玩。

QImage mosaicImage=sourceImage;   
 int width=sourceImage.width();   
 int height=sourceImage.height();   
   
 for(int y=0; y<height;y+=blockSize)   
 {   
 for(int x=0;x<width;x+=blockSize)   
 {   
 // 计算块内的平均颜色   
 int red=0, green=0,blue=0,pixelCount=0;   
 for(int yy=y;yy<y+blockSize && yy<height;yy++)   
 {   
 for(int xx = x; xx<x+blockSize && xx<width;xx++)   
 {   
 QColor color=sourceImage.pixelColor(xx,yy);   
 red+=color.red();   
 green+=color.green();   
 blue+=color.blue();   
 ++pixelCount;   
 }   
 }   
 red/=pixelCount;   
 green/=pixelCount;   
 blue/=pixelCount;   
   
 QColor averageColor(red,green,blue);   
   
 // 将块内所有像素设置为平均颜色   
 for(int yy=y;yy<y+blockSize && yy<height;yy++)   
 {   
 for(int xx=x; xx<x+blockSize && xx<width;xx++)   
 {   
 mosaicImage.setPixelColor(xx,yy,averageColor);   
 }   
 }   
 }   
 }   
 return mosaicImage;

通过这个Mosaic函数，可以实现将图片转换为马赛克风格，后续导入地图时，直接提取每个方格块左上角的颜色rgb值即可。

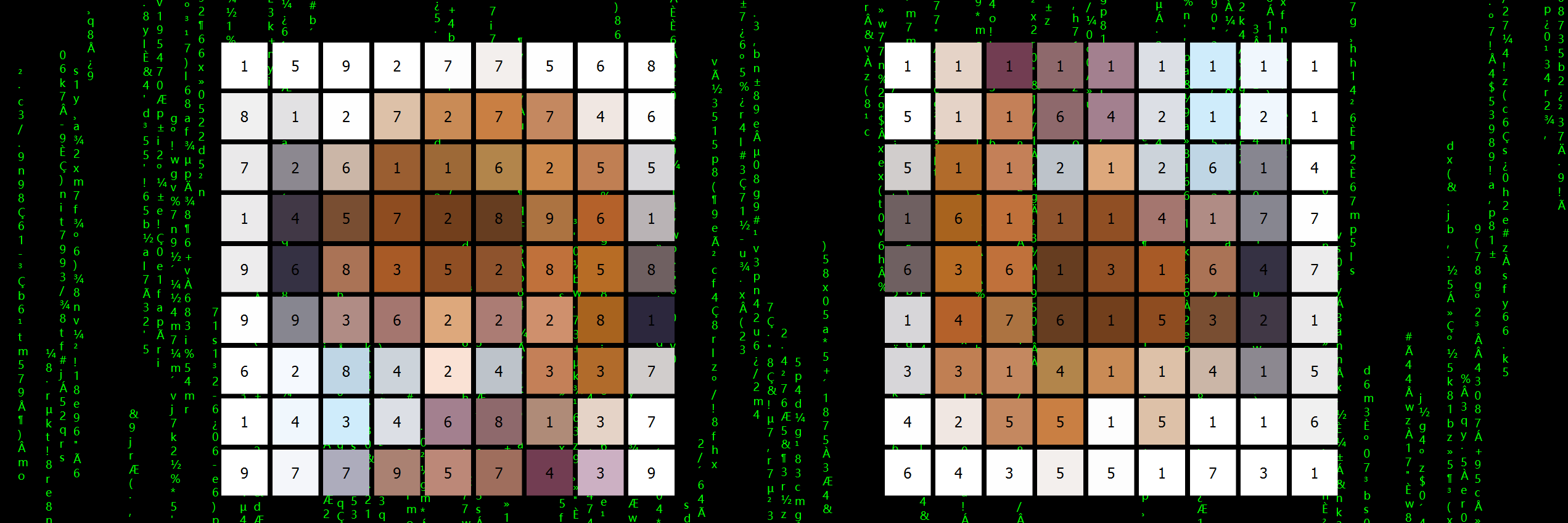
玩家可以有两个选择：可以导入新地图，只需要选择自己想要的图片即可。也可以选择已经导入的地图继续游玩，这两个功能分别由loadImage和loadExit函数实现。



loadExit函数的实现：

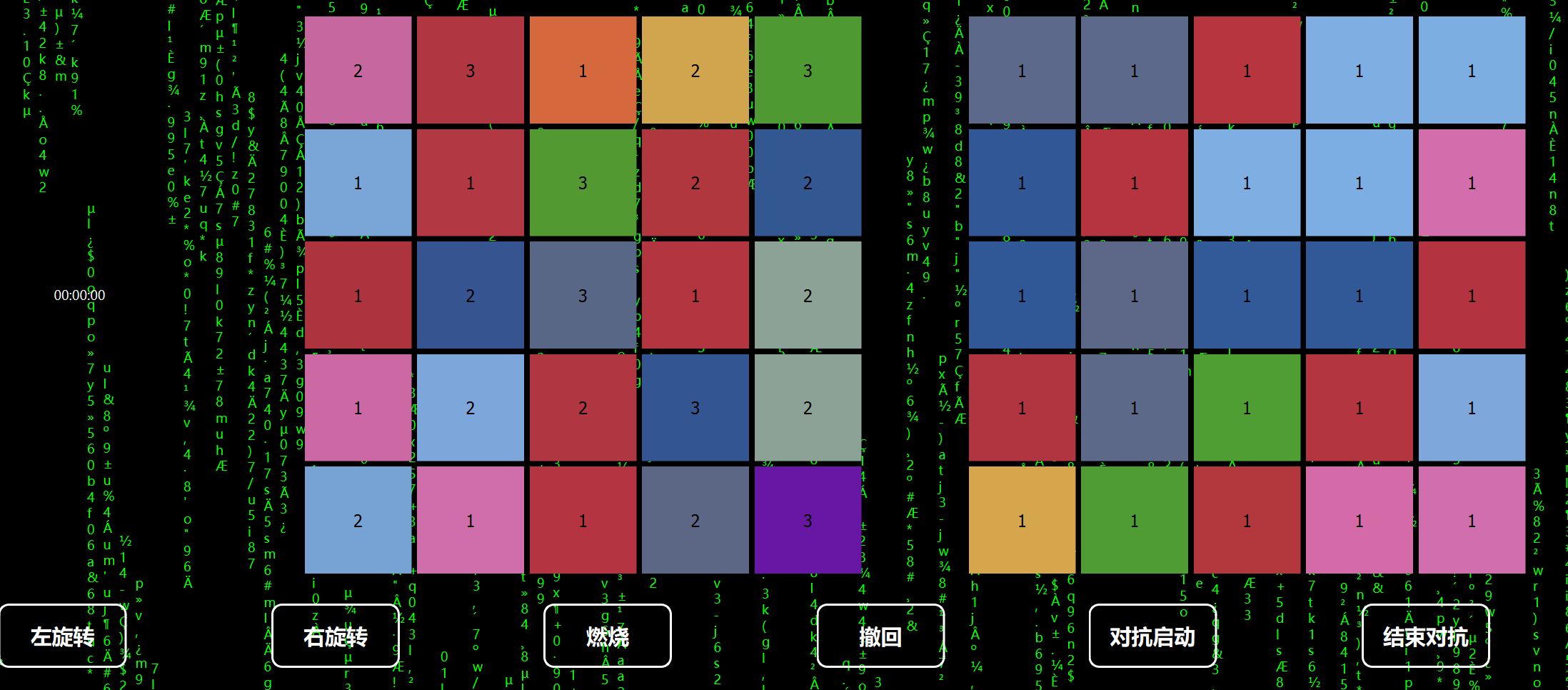


总体的实现效果如下：



## 对抗模式的实现

对原有的游戏模式进行一点点小小的修改：每过一定的时间，系统会对玩家当前的地图进行操作，对玩家的行为造成干扰，这个是作为可供选择的模式，如果玩家点击对抗启动按钮确认，则会启动对抗模式，反之，则不会启动。另外，当玩家觉得当前的游戏难度过大时，玩家也可以通过点击结束对抗按钮来取消，这样就可以实现停止对抗模式。



主要原理是通过计时器实现的

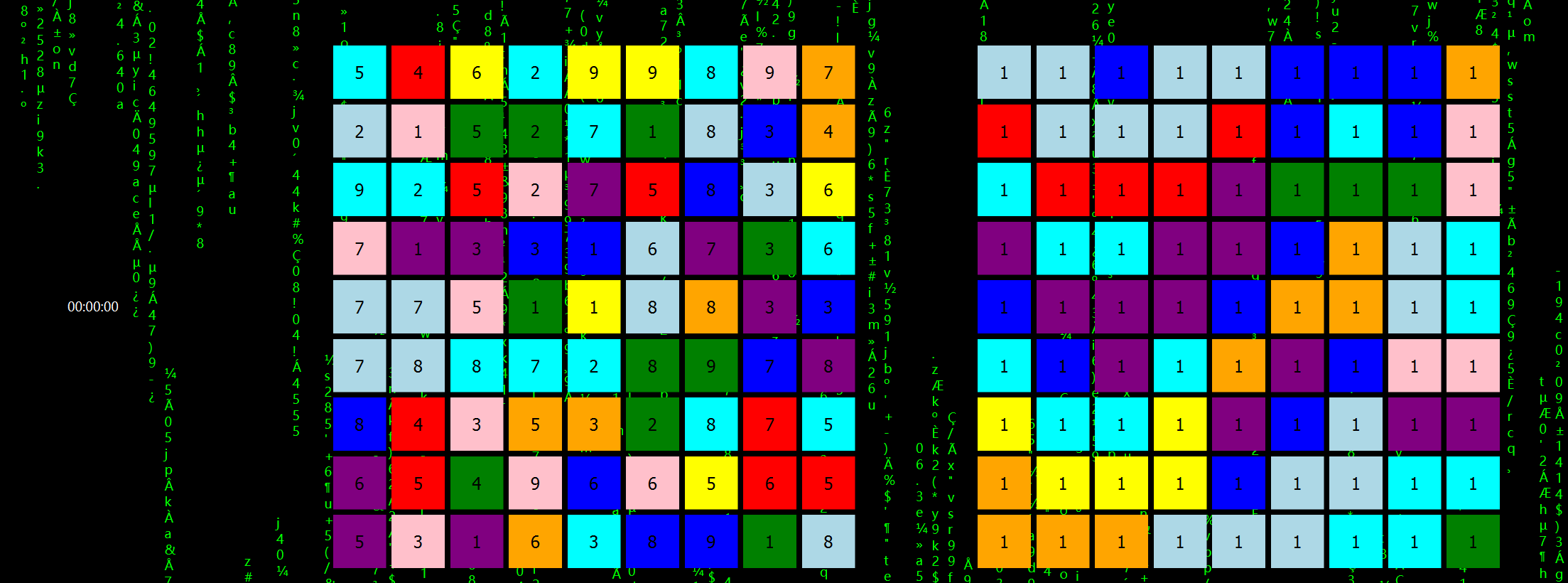


## 随机模式难度的实现

原本的随机模式难度是固定的，现在实现了玩家可以选择三个难度：难度1、难度2、难度3



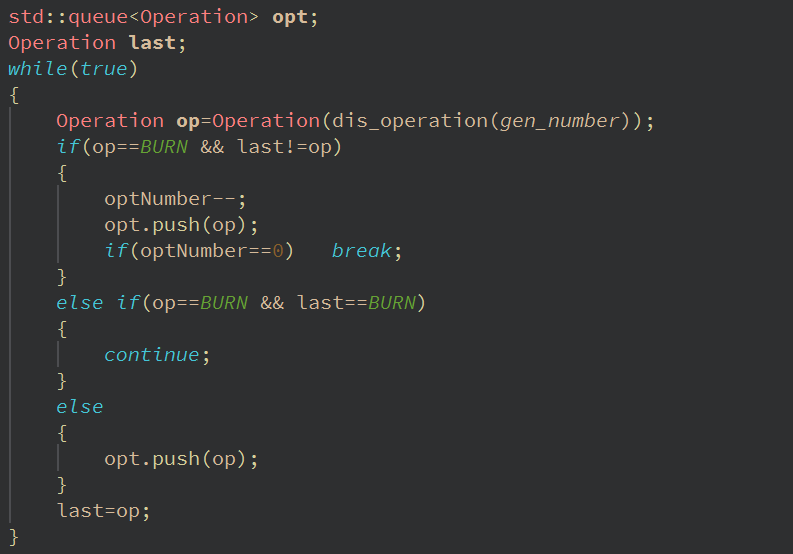
难度一对应的是5 \* 5规模的地图，相应的，每个小方格中的数字也是比较小的，操作次数也是比较少的，难度相对较低。而难度二对应的是7 \* 7的地图，相应的难度较大，操作次数也会有所提高，难度三也在难度二的基础上上升了一个档次。可以参考以下难度三的地图



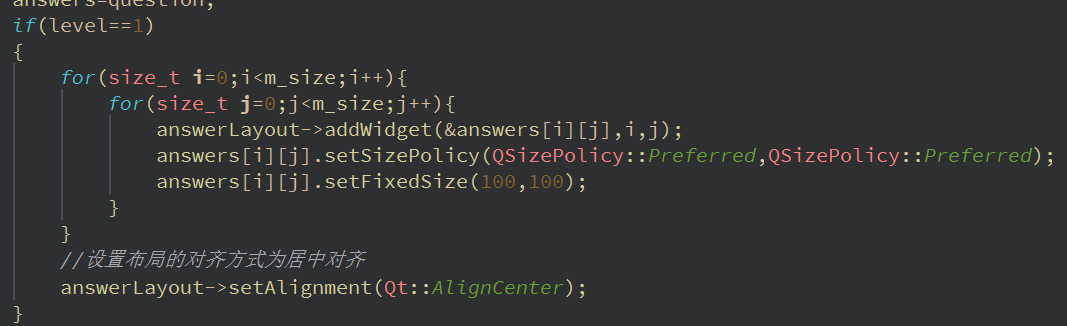
难度一生成代码：



根据选择的难度，生成对应的操作数，并且对每个方格内的数字有一定的要求，不能太大。



生成一系列的操作，同时注意避免一直生成燃烧操作，这样可以保证游戏一定的难度。接下来就是对这些操作的执行操作。



最后就是设置好答案，并且更新即可。

原计划是打算在个人界面添加一个音乐播放器，但是在添加模块时出了一些问题，导致整个项目全面崩溃，还好最后抢救回来，但是又花费了一晚上的时间导入地图。

这里的导入地图也是改用了马赛克算法，这样提高了效率，但是也存在一些问题，由于扫描的地图的原因，导致看上去的颜色非常相近的块的值的rgb值不同，这就导致了judge函数在判断时出现了问题。

大致解决方法就是设定进行judge时，对m\_cells和answer的同一个位置块的rgb值进行计算：分别计算red，green，blue三个的差值，并且只要在一定范围内，就认为是正确的：



