**需求及类分析**

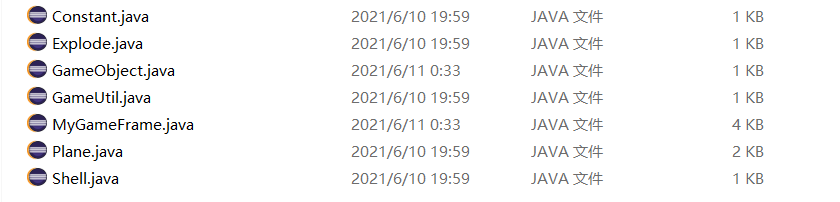
## 1.系统功能需求分析

本游戏的需求是首先生成一个游戏界面，然后载入背景图和小鸡的图片，并且设置小鸡由上下左右键控制其移动。在界面的200\*200像素处生成数十只炸弹气，并让炸弹向随机方向移动，当碰到游戏界面边缘时反弹向另一个方向移动。并且检测小鸡和炸弹的坐标是否有重合，如有重合则表示小鸡被炸弹炸到，则游戏结束。

小鸡躲炸弹游戏规则如下：游戏中的数十只炸弹在游戏界面内上任意方向行走，当炸弹碰到游戏界面边缘时则反弹向另一方向行走，游戏玩家通过上、下、左、右控制键控制小鸡向相应的方向移动。当炸弹碰到小鸡时小鸡则被炸游戏结束。

plane

系统的功能需求主要分为以下几类：



1. 游戏主界面需要呈现游戏背景、炸弹以及小鸡。

### 小鸡类

小鸡类是游戏的主体角色类，通过drawSelf(Graphics g)进行绘制。同时通过判断用户的键盘操作来挪动自己的位置，基于用户的上下左右操作，位置挪动如下：

g.drawImage(img, (int)x,(int) y, null);

if(left){

x -=speed;

}

if(right){

x += speed;

}

if(up){

y -=speed; //y = y-speed;

}

if(down){

y += speed;

}

### 炸弹类

Shell.java定义了炸弹类。炸弹类我们用实心的黄色圆形实现。是在窗口固定位置（200,200）处生成炸弹，炸弹飞行的方向通过Math.sin(degree)定义，角度是随机的，速度定义为3，并且遇见边界会反弹。

x += speed\*Math.cos(degree);

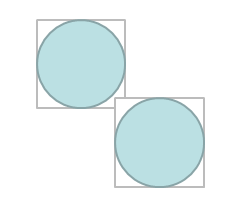
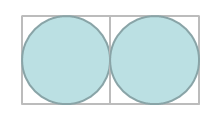
y += speed\*Math.sin(degree);

degree = Math.random()\*Math.PI\*2;

1. 游戏可以随机产生炸弹，炸弹能够按照随机产生的角度飞行，并能够在碰撞到面板边缘后自动返回。
2. 游戏中有一只小鸡，小鸡可以通过游戏玩家的操作进行上下左右的移动，来躲避随机飞行的炸弹。
3. 游戏需要能够监测到小鸡和炸弹是否相撞，如果没有相撞，则继续进行游戏。如果相撞，则游戏结束，显示游戏的时长。

### 矩形检测原理

在游戏中，多个元素是否碰在一起，实际上，通常是用“矩形检测”原理实现的。在该游戏中所有物体都可以抽象为“矩形”，我们只需要判断两个矩形是否相交即可。



Java提供了intersects()方法，可以直接判断矩形是否相交。

boolean peng = shells[i].getRect().intersects(plane.getRect());

具体的实现方式是通过定义的类的width、height以及它的x坐标和y坐标实现。

在本游戏中所有物体包括小鸡和炸弹都有它们自己的矩形对象。圆形的定义也是有其长与宽。小鸡的图像定义虽然是小鸡形状，但是其只是绘制了小鸡形状的色彩，真实的图像保持了矩形形状，仍然包含长度与宽度。

当系统检测到它们的矩形元素有交集时，我们就判定小鸡和炸弹相撞，则游戏结束。

## 2系统非功能需求分析

## 1) 系统可靠性需求

该游戏系统具有较高的可靠性，游戏的后台系统需要稳定，并能够对用户的启动游戏、关闭游戏等及时作出反应，使得游戏的运作非常可靠。

**2) 系统响应时间需求**

用户打开页面以及提交相关操作的响应时间一般应低于1.5秒。如果游戏的加载速度过慢，会给系统的运行结果带来极大的不确定性，同时系统响应时间会较长，影响用户体验。因此，游戏需要进一步提升响应时间。

由于在游戏的过程中，存在大量绘制图案的操作，并同时运用到爆炸、小鸡等图片。由于此因素的存在，在游戏设计与开发过程中，应该采用体积较小的PNG的图片格式，并将图片适当进行压缩有效地提高系统的响应时间。

1. **类的具体展示**