一、课程设计内容与目标

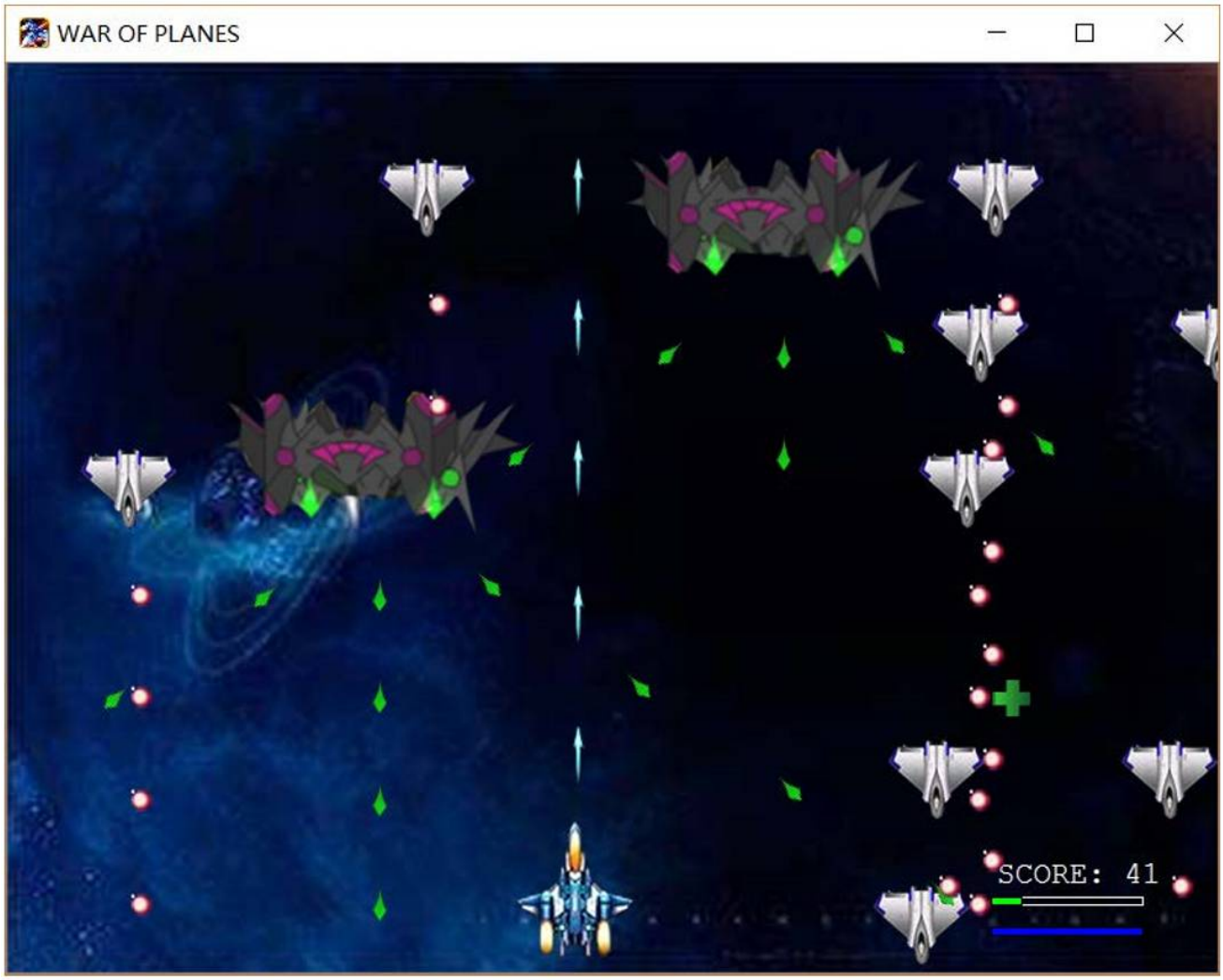
通过类的继承等来实现一个飞机小游戏，首先实现控制台版的飞机游戏，然后修改与显示相关的部分代码完成从控制台到图形界面的迁移过程。

飞机游戏中要实现以下几个基本要素：

- 玩家飞机发射子弹攻击敌机
- 敌机可以发射子弹攻击玩家
- 玩家飞机可以自由移动，敌机可以定向移动
- 玩家飞机与敌机只有在真正边界相撞时才算相撞，且相撞时双方均要受到一定伤害

在课程设计一中，我实现了上述的基本功能，并提供了向图形界面的接口函数，在课程设计一中，玩家飞机和敌机均可以通过简单的代码修改任意更改形态

在课程设计二中，我添加了程序启动界面（就是此实验报告的封面），欢迎界面、背景音乐、暂停、玩家飞机技能、补给掉落、boss 等功能，游戏界面如下图：



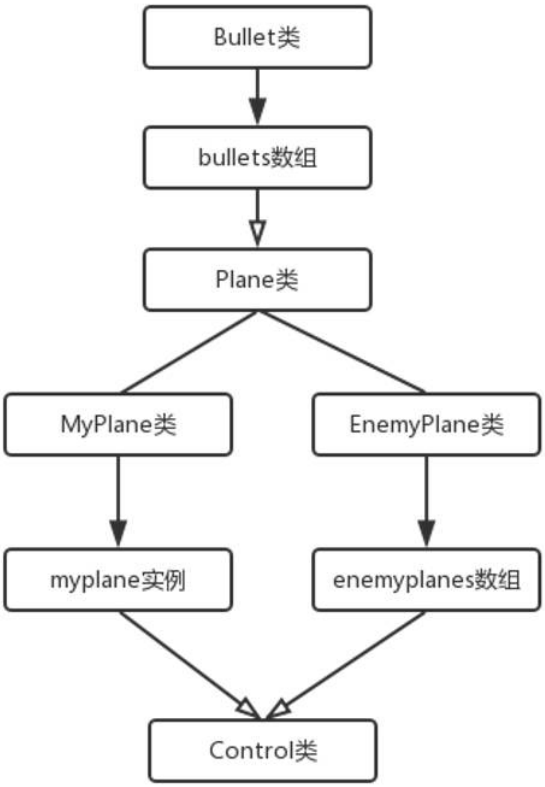
在此游戏中，右下角的 SCORE 为玩家当前击落的敌机数，绿色进度条为玩家飞机当前的血条，下面的蓝色进度条为玩家飞机当前的技能条。玩家可以使用消耗技能值实现一技能，如按 Q 可以三连发（如上图），按 E 可以一次性击落当前所有敌机，按 R 可以一次性消掉当前所有子弹。

二、类层次关系和实现

本次实验中，我总共设计了如下几个两个基类 Plane 和 Bullet。Plane 派生出MyPlane和 EnemyPlane 两个子类。还有一个 Control类管理全局。两次课程设计中主要使用的类没有发生变化，只是类之间的关系稍有变化。

2.1 第一次课程设计（控制台版）

第一次课程设计（控制台版）中各类关系如下：



如图，Plane 类中聚集了 Bullet 类的实例 bullets 数组，用于记录该飞机发出去的子弹；Plane 类派生出两个子类，MyPlane 与 EnemyPlane，分别是玩家飞机与敌机。这两个类均聚集在 Control 类中，由 Control 类对所有飞机的状态进行管理。由于 Control 与 Plane、MyPlane、EnemyPlane 关系密切，故我将 Control类设置为这几个类的友元。

在这个版本中，我通过一个循环来检测用户的按键输入和定时动作，如下：

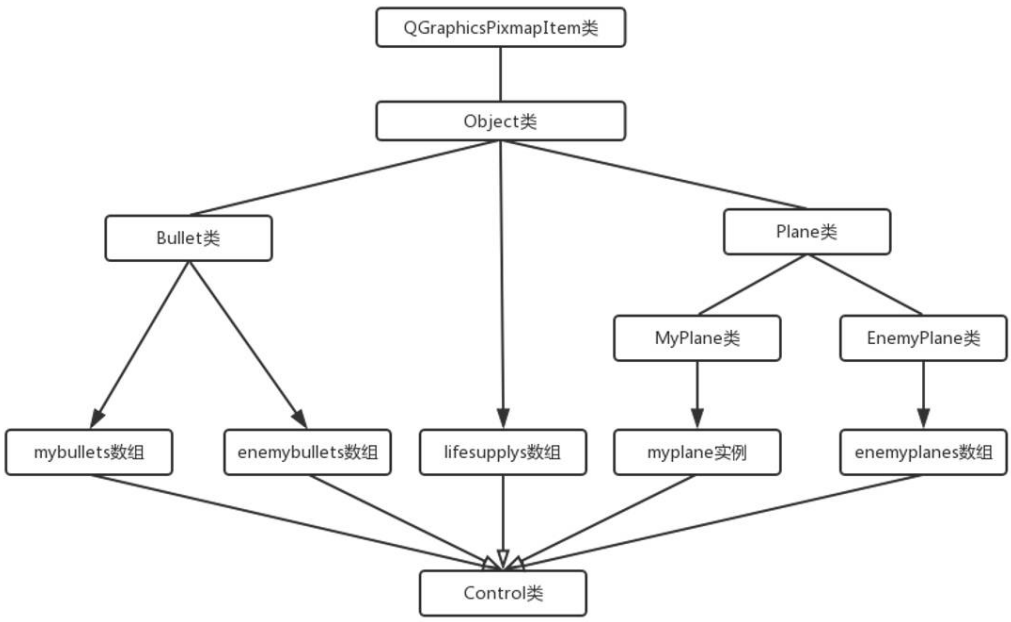
```
clock_t bulletPositionClock = clock();
clock_t newBulletClock = clock();
clock_t planePositionClock = clock();
while (1)
{
    if (clock() - bulletPositionClock >= 300) //每300毫秒刷新一次子弹位置
    {
        if (!updateBullets())
            break;
        bulletPositionClock = clock();
    }

    if (clock() - newBulletClock >= 500) //每500毫秒发射一次新子弹
    {
        shootEnemyBullets();
        newBulletClock = clock();
    }

    if (clock() - planePositionClock >= 1000) //每1000毫秒更新一次敌机的位置
    {
        if (!updateEnemyPlanes())
            break;
        planePositionClock = clock();
    }

    if (kbhit() && !keyboardHandle(getch()))
        break;

}
```

keyboardHandle是我自己写的一个函数，用于根据用户的不同按键获取用户输入。

2.2 第二次课程设计（图形界面版）

第二次课程设计（图形界面版）中对类的关系进行了一些优化。因为 Plane和 Bullet 类都有一些共同的属性，所以设计了一些共同的基类 Object，以实现一些共同的性质，如 synScreen，delScreen 等。第二次课程设计是用的是 Qt 的QGraphicsScene、QGraphicsItem、QGraphicsView 三组件，其中 Control 继承自QGraphicsScene，Plane 与 Bullet 继承自 QGraphicsItem，在 main 函数中创建QGraphicsView与QGraphicsScene 相关联，主要操作都在 Control中完成。

类结构图如下：



第二次课程设计中，Bullet 类不再聚集在 Plane 类中，而是直接实例化为两个数组，聚集在 Control中，便于管理。Object 类均需要使用图片来初始化，所以都是 Qt 提供的 QGraphicsPixmapItem 类的子类。而 Control 类抽象成为QGraphicsScene 类的子类，用于管理 Scene 中的各个 Item。

除此之外，为了丰富游戏内容，还在增加了其他的一些组件，用以实现游戏启动界面、欢迎界面、背景音乐、血条、技能条、游戏暂停、游戏结束等功能。

类名及相关游戏组件如下：

- QGraphicsTextItem 游戏标题、提示信息等
- QGraphicsWidget 游戏按钮、遮罩面板（用于在游戏暂停时达到场景变暗的效果）
- QGraphicsRectItem 血条、技能条。

此外，检测玩家按键输入和定时动作的方式也使用了 QObject 提供的消息机制来实现，使用了如下事件函数：

- timeEvent() 用于接收计时器消息，实现飞机、子弹定时移动，敌机的定时生成
- keyPressEvent() 用于接收键盘控制，实现玩家控制

三、课程设计一和二的关联与衔接

课程设计一为了向课程设计二过渡，设计了一些共同的接口函数，在迁移过程中仅需修改参数即可。

在过渡中优化了一部分类的设计，比如发现 Bullet、Plane、LifeSupply 都有共同的属性，故让这三个类均继承自 Object 类。

Object 类与显示相关的函数变动如下表：

函数名	控制台版	图形界面版
delScreen	delScreen(char **screen)	delScreen(QGraphicsScene *scene)
synScreen	synScreen(char **screen)	synScreen(QGraphicsScene *scene)

delScreen 用于将子弹在屏幕上抹去，synScreen 用于将子弹显示在屏幕上，hit 用于子弹与飞机相撞时降低子弹生命值，生命值降为 0 时抹去子弹。

Bullet 类只需要改动一个函数：

hit	hit(char **screen)	hit(QGraphicsScene *scene)

其中 char **screen 为控制台版中 Control 类的一个成员变量，用于保存要在屏幕上显示的所有字符；在图形界面版中 Control类继承自 QGraphicsScene，本身就保存了所有要显示的信息，所以可以直接将 Control类指针传进去。

Plane 类也只需改动一个函数：

函数名	控制台版	图形界面版
crash	crash(char **screen)	crash(QGraphicsScene *scene)

crash用于飞机与飞机或子弹相撞时降低生命值，生命值降为0时抹去飞机。

Control类变化较多，因为 Qt中有一些现成的库函数，所以有些地方可以直接使用库函数，变化如下：（参数略去）

函数功能	控制台版	图形界面版
刷新屏幕	Control::refreshScreen()	QGraphicsItem::update()
碰撞检测	Control::judgeCrash()	QGraphicsItem:: collidesWithItem ()
键盘响应	Control::keyboardHandle()	QObject:: keyPressEvent()
定时更新	Control::run()	QObject:: timerEvent()

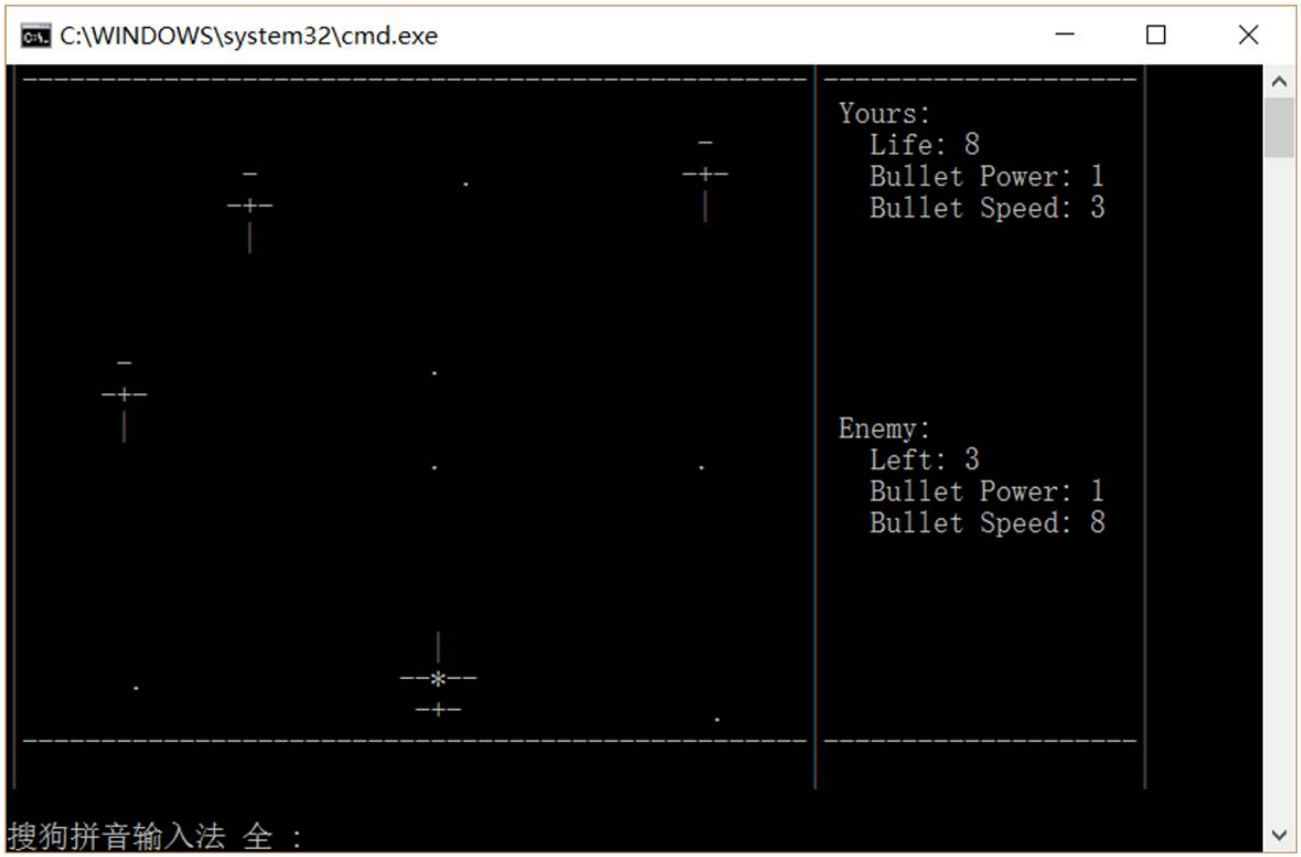
另外，为了更好地适应图形界面，也对一些功能做出了修改，比如控制台版本的子弹相撞可以消掉，但是图形界面中玩家飞机子弹与敌机子弹不能消掉。

除此之外，为了丰富游戏，增强游戏可玩性和趣味性，还添加了其他的函数以实现：游戏启动界面、游戏欢迎界面及帮助、游戏暂停、背景音乐、玩家技能、随机掉落补给等等功能。

四、实验成果展示

4.1 第一次课程设计（控制台版）

控制台版的飞机大战较为简陋，但是可拓展性较好。敌机和玩家飞机的形状可以直接通过更改代码中定义的一个静态数组来改变，且碰撞也是按照实际形状来检测的。玩家可以通过一些按键来控制右侧栏中一系列参数：生命值、子弹威力、子弹速度等等。玩家子弹和敌机子弹相撞时会同时销毁。运行效果如下图：

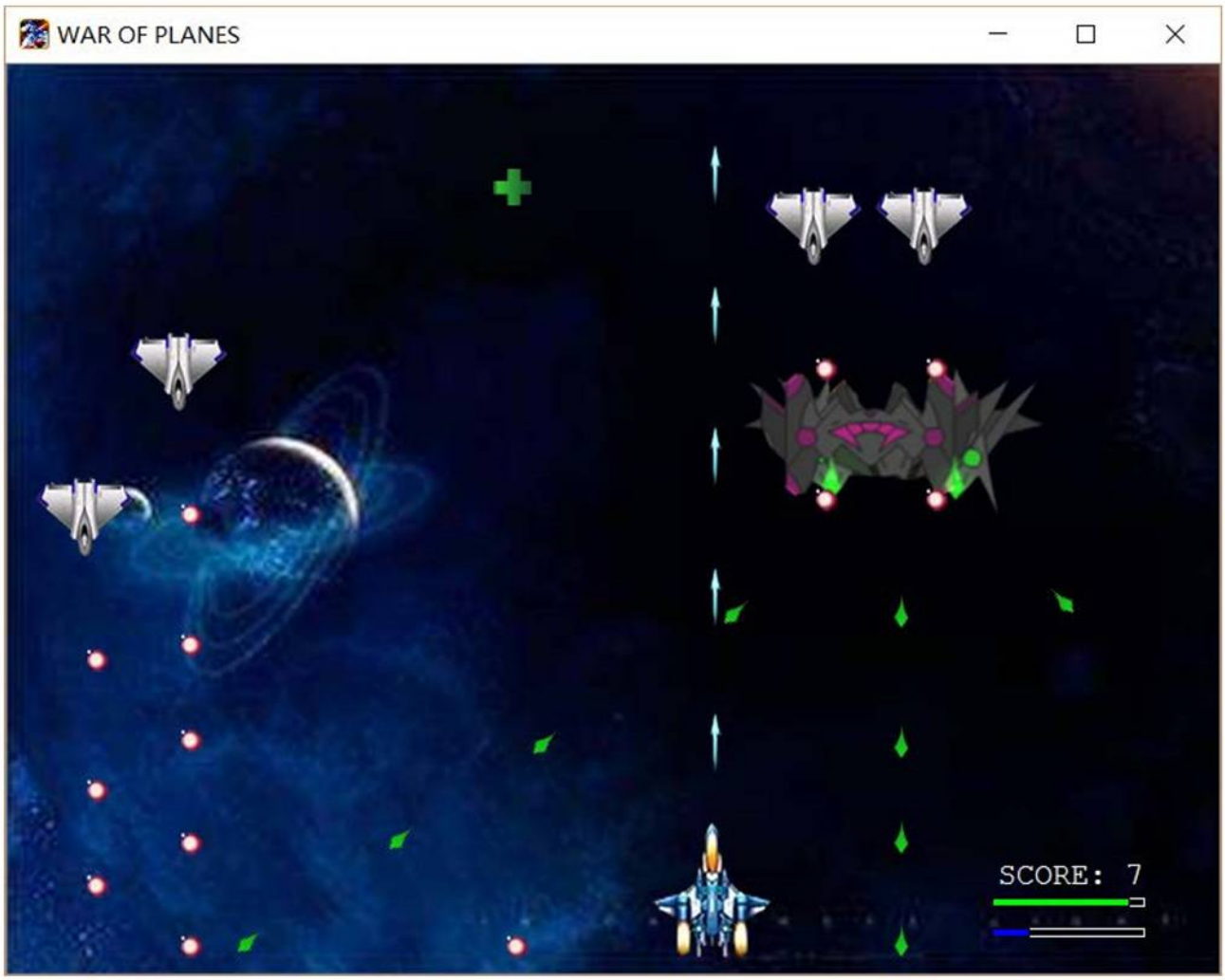


4.2 第二次课程设计（图形界面版）

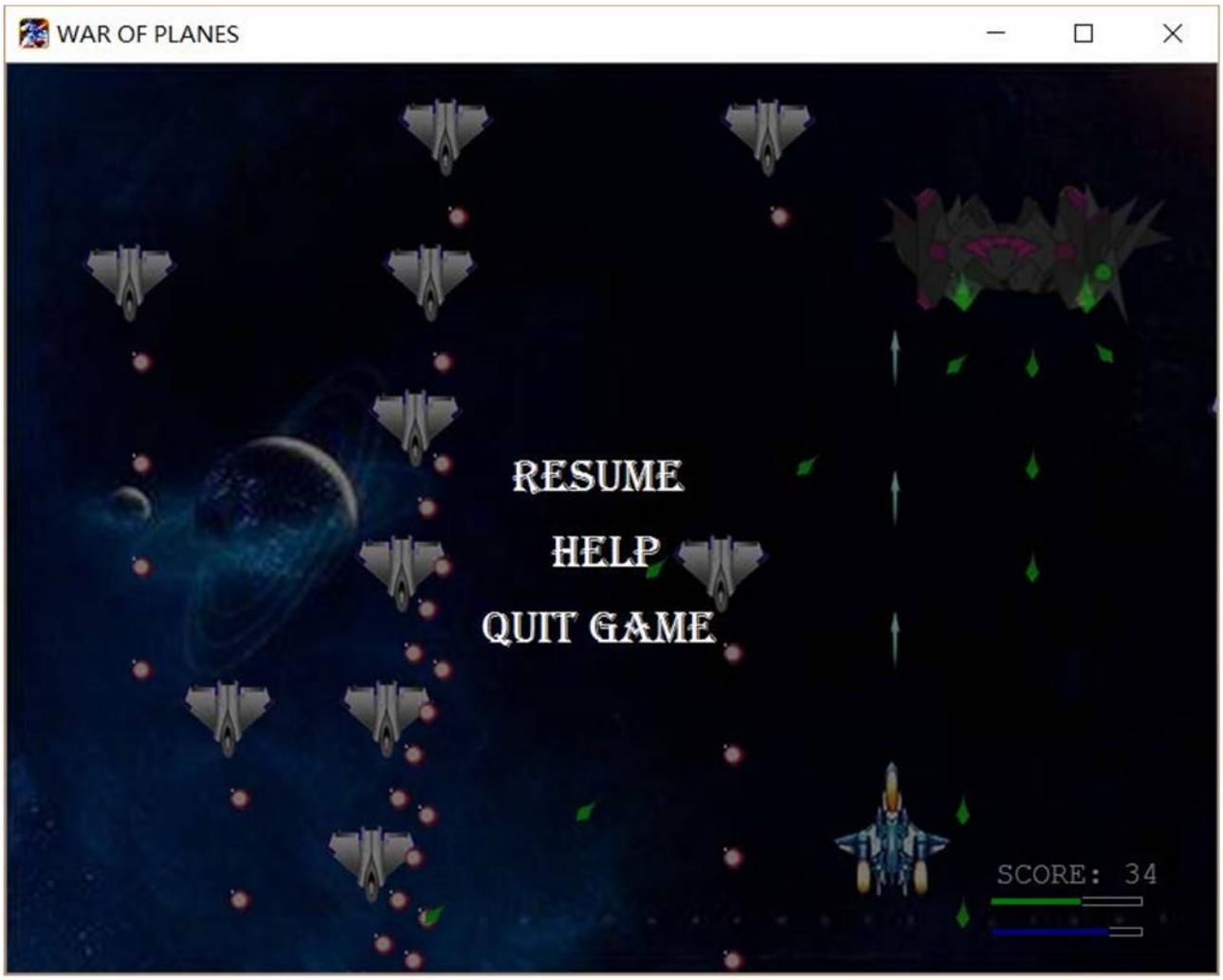
我为我的程序设计了一个安装包，已在附件中。安装之后会在桌面创建一个快捷方式，双击便可运行。如下是启动画面：

欢迎界面如下：

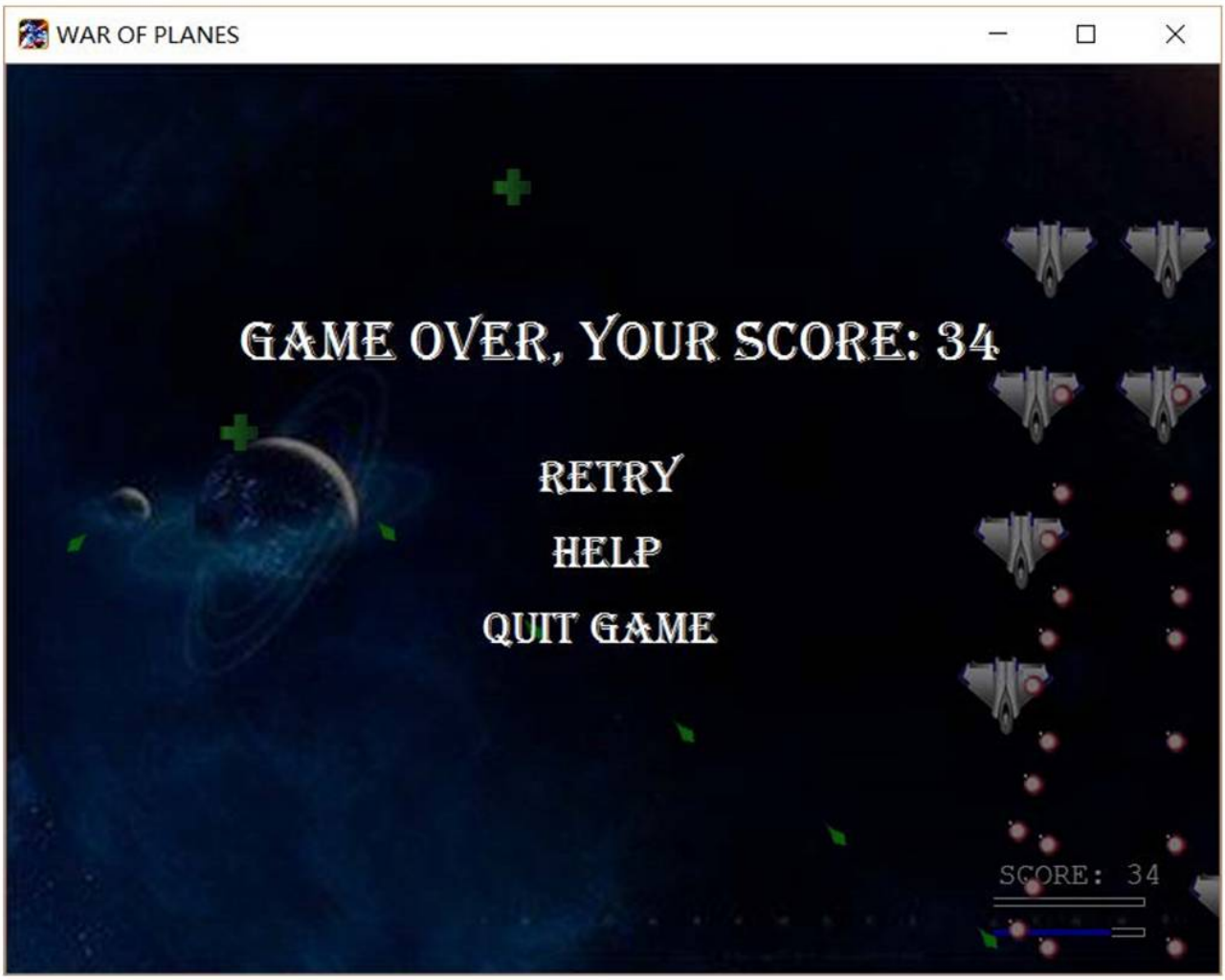
点击开始游戏，右下方 SCORE 为分数，绿色进度条为生命值，蓝色进度条为技能值，可以使用 J、K、L 三个键释放技能。同时，敌机被击落的时候也会随机掉落声明补给（绿色的加号）。



按空格键可以暂停，暂停时背景变暗。



生命值耗尽时游戏结束，玩家可以选择重玩或者退出游戏。



五、遇到的问题和思考

5.1 代码复用的思考

在这第一次设计中，类的继承主要体现在 Plane 派生出 MyPlane 与EnemyPlane 上，起到了不错的效果。但是还有可以改进的地方。

在完成第二次课程设计的过程中，我考虑到在 Bullet 类和 Plane 类中都有delScreen、synScreen 函数功能也非常接近，且有一些共同的成员变量，如果要更好地复用的话，应该让 Bullet 类与Plane 类都继承自同一个 Object 类，于是设计了 Object 类，这样在后续添加新的游戏道具时也会更加方便。

5.2 字符界面过渡图形界面遇到的困难

由于之前没有图形界面编程的经验，所以不知道应该怎样为图形界面留出接口，后来决定用 Qt 之后，查了一些关于 Qt 的资料了解到 QGraphicsScene 与QGraphicsItem 提供的库函数，于是设计了上述的一些函数便于代码迁移。

5.3 对文档-视结构的思考

在 MFC 的文档-视结构中，程序的数据储存在文档类中，文档类是对数据的抽象表示。数据显示由视图负责，视图是程序窗口的客户区，框架窗口是客户区的框架，程序数据显示在窗口，用户通过视图与程序交互。

在第二次课程设计中，我的代码结构与此类似。我使用的 QGraphicsScene（Control的基类）类似于文档-视中的 CDocument 类，保存的是程序的数据（飞机、子弹、提示信息等）；QGraphicsItem 就是文档-视中的数据，通过QGraphicsScene 的 addItem 函数加入到场景中，由 QGraphicsScene 管理；QGraphicsView 类似于文档-视中的 CView 类，通过其 setScene 函数与QGraphicsScene 建立关联。