VC++实训报告

姓名	余俊晖	学号	1652200204（转专业）
班级	软件工程 4 班	项目名称	星际出击 yjh
实训地点	自主安排	本人 qq	1254094880

桂林电子科技大学信息科技学院

2017 年 12 月 27 日-12 月 30 日

目　录

1. 概述

1.1 项目简介

本次实训项目是做一个飞机大战的游戏，应用 MFC 编程，完成一个界面简洁流畅、游戏方式简单，玩起来易于上手的桌面游戏。该飞机大战项目运用的主要技术即是 MFC 编程中的一些函数、链表思想以及贴图技术。

1.2 实训功能说明

1.2.1 基本功能

（1）通过键盘，方向键和 ASWD 键可控制战机的位置，空格键和鼠标左键发射子弹。

（2）界面中敌机出现的位置，以及敌机和 Boss 炸弹的发射均为随机的，敌机与敌机炸弹、Boss 炸弹均具有一定的速度，且随着关卡难度的增大，数量和速度均随着关卡数增加而增加。

（3）对于随机产生的敌机和敌机炸弹，若超过矩形区域，则释放该对象。

（4）添加碰撞效果，包括战机子弹打中敌机爆炸、敌机炸弹打中战机爆炸、战机与敌机相撞爆炸、战机子弹与敌机炸弹相撞爆炸、战机子弹打中 Boss、战机与 Boss 碰撞以及战机吃到血包七种碰撞效果。且碰撞发生后子弹、炸弹、血包均消失，战机生命值减一，敌机和 Boss 生命值减少当前战机炮弹威力的生命值，若敌机或 Boss 生命值归零，则删除敌机或 Boss。

（5）血包：随着关卡游戏进程的进行，会出现一定量的血包供战机补给生命值，血包会在客户区矩形框内运动，10 秒后消失；若战机在 10 秒内吃到血包，则会增加 5 点生命值知道生命值上限。

（6）每关中战机有三条命，每条命 10 点生命值，生命使用完后，会进入 GameOver 界面显示得分数，并提供重新开始游戏和退出功能。

（7）游戏提供 10 个关卡，每个关卡需要打死相应关卡的敌机数量才能进入 Boss 模式，打败 Boss 之后将会进入下一关。10 关通关后，显示通关界面，并提供重新开始游戏和退出游戏的功能选项。

（8）暂停功能：游戏进行过程中按下 Z 键可进入暂停模式，再按 Z 则返回游戏。

（9）无敌模式：游戏进行过程中按下 Y 键可进入无敌模式，再按 Y 则返回正常游戏。该模式下战机生命值不会减少，可供测试使用。

（10）魔法值：游戏进行过程中，战机魔法值会随着时间递增到上限 10，魔法值供战机道具功能的使用，过一个关卡魔法值不清零。

（11）战机大招：当战机魔法值为 10 满状态时，按下 X 键消耗所有魔法值可发动大招，对屏幕中的敌机进行清屏，Boss 扣 50 点血量。

（12）防护罩：当魔法值不为 0 时，按下 C 键可打开防护罩道具，该状态下战机处于无敌状态，不会损失生命值，但魔法值会随着防护罩开启慢慢降低。

（13）战机升级功能：战机子弹单个威力为 1，在魔法值不为 0 时，按下 V 键开启升级战机模式，战机图标变为动画，子弹威力变成两倍。（若同时开启防护罩和战机升级，则魔法值递减速度翻倍）。

1.2.2 附加功能

（1） 为游戏界面每个关卡添加了滚动背景图片和背景音乐，并在敌机发送炮弹、战机发射子弹、战机击中敌机、敌机击中战机、战机敌机相撞、敌机战机子弹相撞、战机吃到血包、战机大招、战机升级、战机防护罩、游戏结束时均添加了音效。

（2）为美化游戏界面，采用了一部分全民飞机大战图标，并添加了爆炸动画和升级战机动画特效，背景音乐采用微信飞机大战背景音乐和相关特效音效。

（3）为游戏设置了不同的关卡，每个关卡难度不同，敌机与敌机炸弹的速度随着关卡增大而加快，进入第五关以后敌机从上下方均会随机出现，且随机发射炸弹。

（4）前五关卡敌机从上方飞出，速度一定，战机每打掉一架敌机则增加一分，当战机得分超过该关卡所需分数（和关卡数相关）则可进入 Boss 模式，打败 Boss 进入下一关；进入第六关以后，敌机分别从上下两方飞出。随着关卡数增加，敌机数量增加，速度增快，敌机炮弹数量和速度也相应增加，进入 Boss 所需分数增加，Boss 生命值和火力也随着关卡数的增加而增加，游戏难度陡然直升。

（5）游戏界面中显示当前状态下的关卡数、当前命数、当前得分、战机血条、战机魔法条、无敌模式提醒和战机道具提醒，Boss 模式下还有 Boss 血条。

（6）增加了鼠标控制战机位置这一效果，战绩的位置随着鼠标的移动而移动，并且点击鼠标左键可使得战机发射子弹。

（7）进入游戏先进入欢迎界面，欢迎界面中显示游戏使用说明，点击鼠标左键和空格键开始游戏。游戏过程中战机命数使用完、通关均有相应界面进行提醒，用户可选择重新开始游戏或退出游戏。

2. 相关技术

2.1 Windows 定时器技术

Windows 定时器是一种输入设备，它周期性地在每经过一个指定的时间间隔后就通知应用程序一次。程序将时间间隔告诉 Windows，然后 Windows 给您的程序发送周期性发生的 WM_TIMER 消息以表示时间到了。本程序中使用多个定时器，分别控制不同的功能。在 MFC 的 API 函数中使用 SetTimer()函数设置定时器，设置系统间隔时间，在 OnTimer()函数中实现响应定时器的程序。

2.2 透明贴图实现技术

绘制透明位图的关键就是创建一个“掩码”位图(mask bitmap)，这个“掩码”位图是一个单色位图，它是位图中图像的一个单色剪影。

在详细介绍实现过程之前先介绍下所使用的画图函数以及函数参数所代表的功能;整个绘制过程需要使用到 BitBlt()函数。整个功能的实现过程如下：

(1) 创建一张大小与需要绘制图像相同的位图作为“掩码”位图；

(2) 将新创建的“掩码”位图存储至掩码位图的设备描述表中；

(3) 把位图设备描述表的背景设置成“透明色”，不需要显示的颜色；

(4) 复制粘贴位图到“掩码”位图的设备描述表中,这个时候“掩码”位图设备描述表中存放的位图与位图设备描述表中的位图一样；

(5) 把需要透明绘制的位图与对话框绘图相应区域的背景进行逻辑异或操作绘制到对话框上；

(6) 把“掩码”位图与这个时候对话框相应区域的背景进行逻辑与的操作；

(7) 重复步骤 5 的操作,把需要透明绘制的位图与对话框绘图相应区域的背景进行逻辑异或操作绘制到对话框上；

(8) 最后把系统的画笔还给系统，删除使用过的 GDIObject，释放非空的指针，最后把新建的设备描述表也删除。

2.3 CObList 链表

MFC 类库中提供了丰富的 CObList 类的成员函数，此程序主要用到的成员函数如下：

(1) 构造函数，为 CObject 指针构造一个空的列表。

(2) GetHead()，访问链表首部，返回列表中的首元素（列表不能为空）。

(3) AddTail()，在列表尾增加一个元素或另一个列表的所有元素。

(4) RemoveAll(),删除列表中所有的元素。

(5) GetNext(),返回列表中尾元素的位置。

(6) GetHeadPosition()，返回列表中首元素的位置。

(7) RemoveAt()，从列表中删除指定位置的元素。

(8) GetCount()，返回列表中的元素数。

在 CPlaneGameView.h 文件中声明各游戏对象与游戏对象链表：

（1）滚动背景模块

```
CScene  scene;//游戏背景对象
CImageList startIMG;欢迎界面图像列表
```

（2）各游戏对象

```
CMyPlane *myplane;
CEnemy *enemy;
CBoss *boss;
CBomb *bomb;
CBall *ball;
CExplosion *explosion;
CBlood* blood;
```

（3）存储游戏对象的对象链表

```
CObList enemyList;
CObList meList;
CObList bombList;
CObList ballList;
CObList explosionList;
CObList bloodList;
```

（4）客户区窗口矩形（用来动态获取客户区矩形大小）

```
CRect rect;//窗口屏幕矩形
```

（5）游戏运行相关参数：

```
int speed;//战机的速度，方向键控制
int myLife;//为战机设置生命值
int lifeNum;//战机命条数
int myScore;//战机的得分
int passScore;//当前关卡得分数
int lifeCount;//血包产生控制参数
int magicCount;//魔法值，控制能否发大招
int bossBlood;//Boss 血量
int passNum;//记录当前关卡
```

（6）游戏运行相关标志位

```
BOOL bloodExist;//标记屏幕中是否存在血包
int isPass;//是否通关的标志
int isPause;//是否暂停
BOOL isBoss;//标记是否进入 Boss
BOOL bossLoaded;//标记 Boss 出场完成
BOOL isProtect;//标记是否开启防护罩
BOOL isUpdate;//标记战机是否升级
BOOL test;//无敌模式标志位
BOOL isStop;//标记游戏停止
BOOL isStarted;//标记欢迎界面是否加载完成
```

2.4 获取矩形区域

首先，使用 CRect 定义一个对象，然后使用 GetClientRect(&对象名)函数，获取界面的矩形区域 rect.Width() 为矩形区域的宽度，rect.Height()为矩形区域的高度。

使用 IntersectRect(&,&))函数来判断两个源矩形是否有重合的部分。如果有不为空，则返回非零值；否则，返回 0。

2.5 内存释放

在 VC/MFC 用 CDC 绘图时，频繁的刷新，屏幕会出现闪烁的现象，CPU 时间占用率相当高，绘图效率极低，很容易出现程序崩溃。及时的释放程序所占用的内存资源是非常重要的。

在程序中使用到的链表、刷子等占用内存资源的对象都要及时的删除。Delete Brush，List.removeall()等。

2.6 CImageList 处理爆炸效果

爆炸效果是连续的显示一系列的图片。如果把每一张图片都显示出来的话，占用的时间是非常多的，必然后导致程序的可行性下降。CImageList 是一个“图象列表”是相同大小图象的集合，每个图象都可由其基于零的索引来参考。可以用来存放爆炸效果的一张图片，使用 Draw()函数来绘制在某拖拉操作中正被拖动的图象，即可通过 Timer 消息连续绘制出多张图片做成的爆炸效果。

3. 总体设计与详细设计

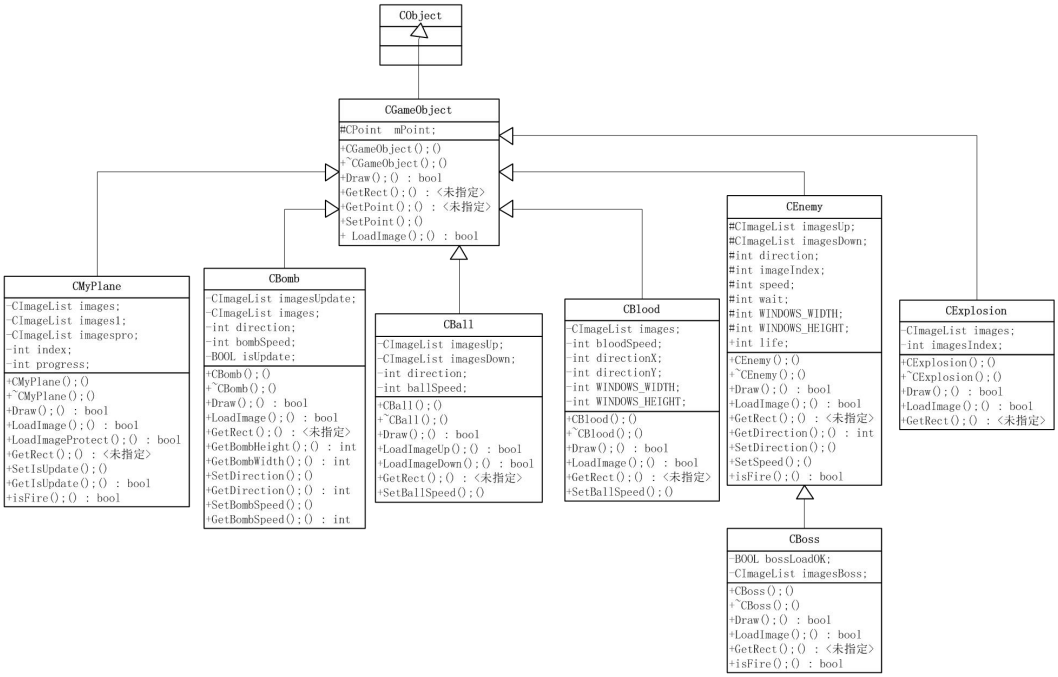
3.1 系统模块划分

该飞机大战游戏程序分为游戏滚动背景绘制模块、各游戏对象绘制模块、游戏对象之间的碰撞模块、爆炸效果产生模块、游戏界面输出玩家得分关卡信息模块、战机道具技能维护模块、消息处理模块、视图生命周期维护模块。

其中在游戏对象绘制模块中，战机是唯一对象，在游戏开始时产生该对象，赋予其固定的生命值，当其与敌机对象、敌机炸弹碰撞时使其生命值减一，直至生命值为零，便删除战机对象。敌机对象与敌机炸弹对象的绘制中采用定时器技术，定时产生。爆炸对象初始化为空，当游戏过程中即时发生碰撞时，在碰撞位置产生爆炸对象，添加到爆炸链表中，并根据爆炸素材图像分八帧进行输出，达到动画特效。

3.2 主要功能模块

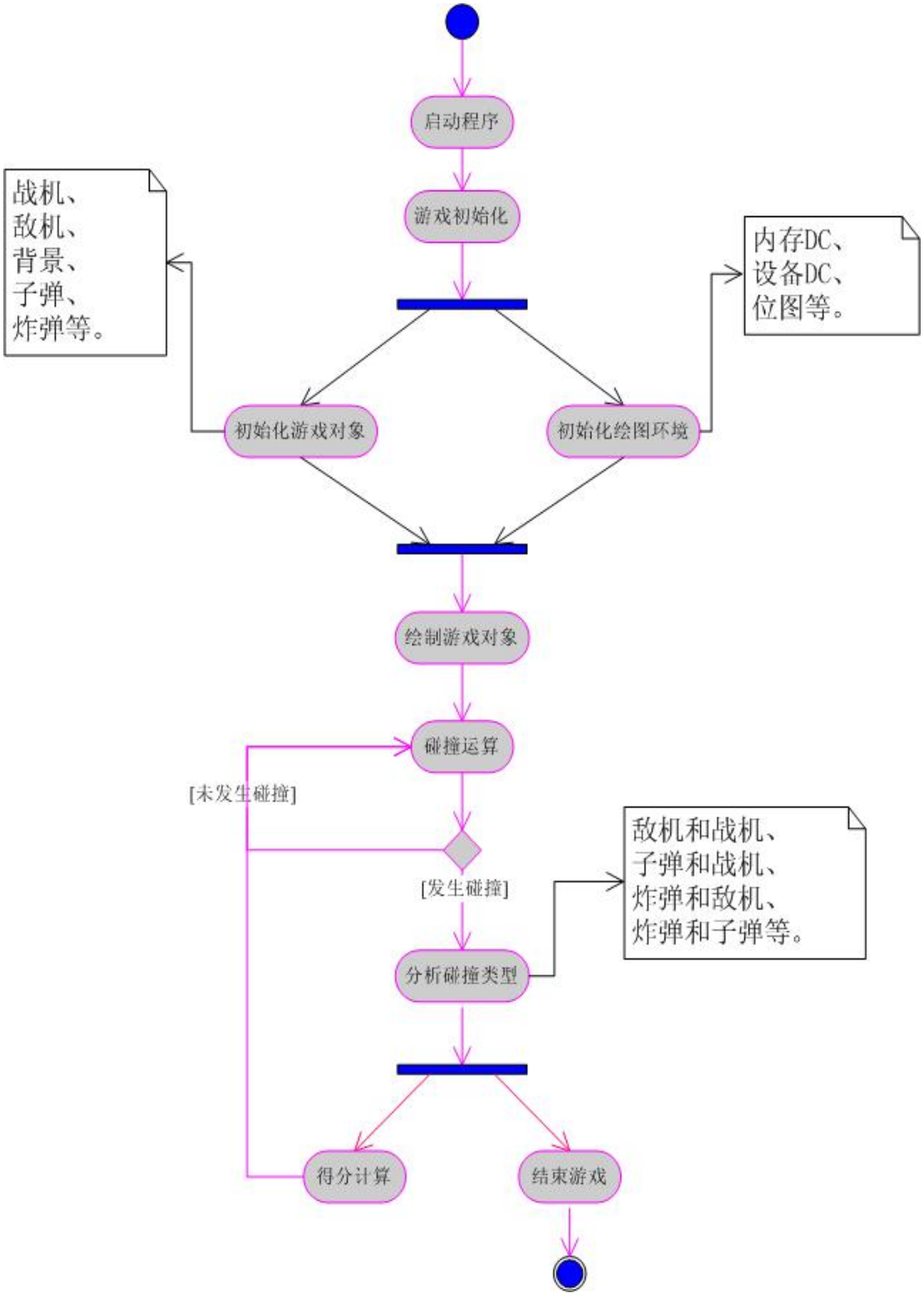
3.2.1 系统对象类图



CGameObject 是各个游戏对象的抽象父类，继承自 CObject 类，其他的类：战机类、敌机类、爆炸类、子弹类、炸弹类、血包类、文字类都继承了此类，CBoss 类继承敌机类。

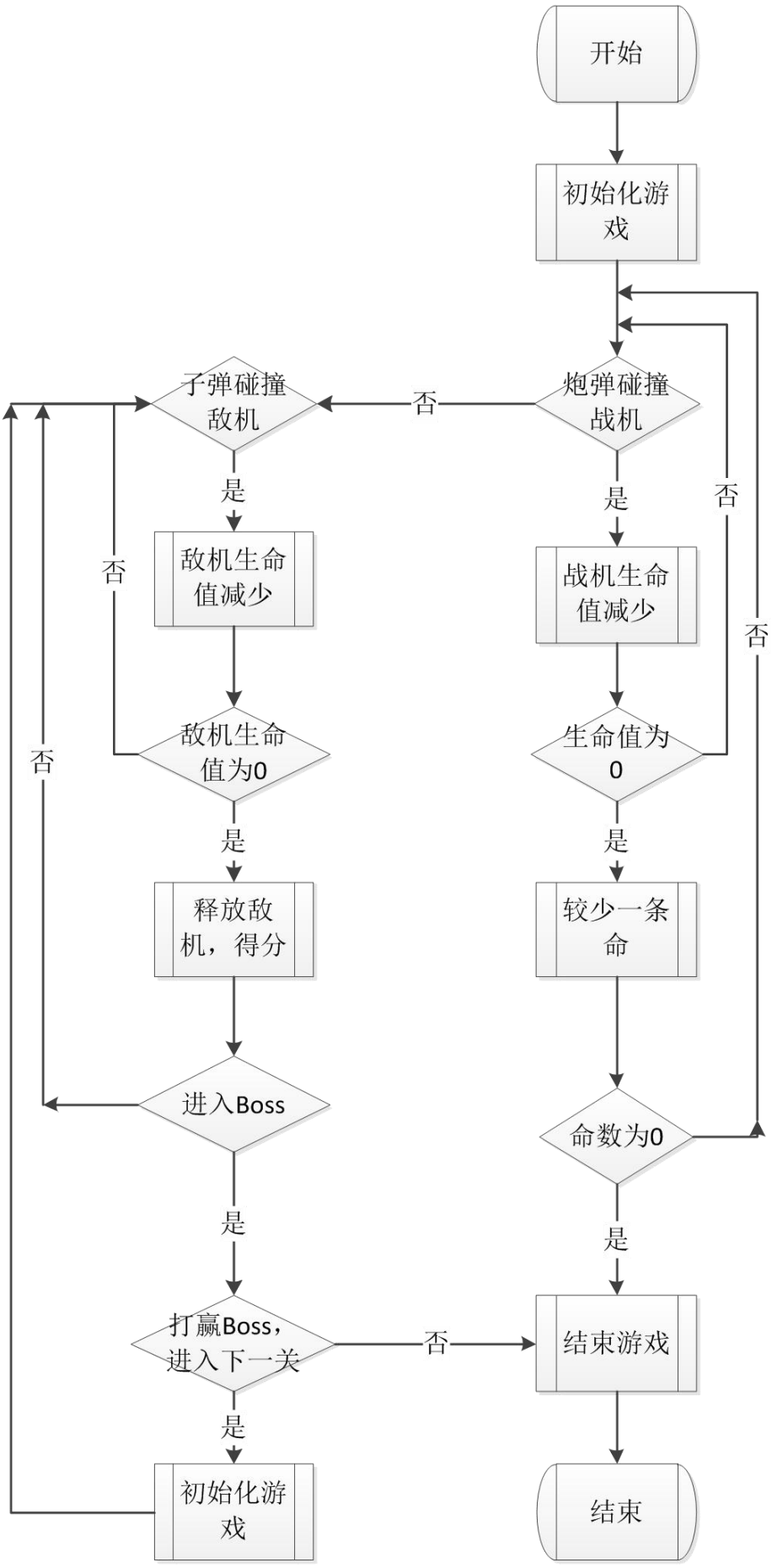
每个游戏对象类中既继承了来自父类 CGameObject 的属性，又有自己的特有属性和方法。

3.2.2 系统主程序活动图

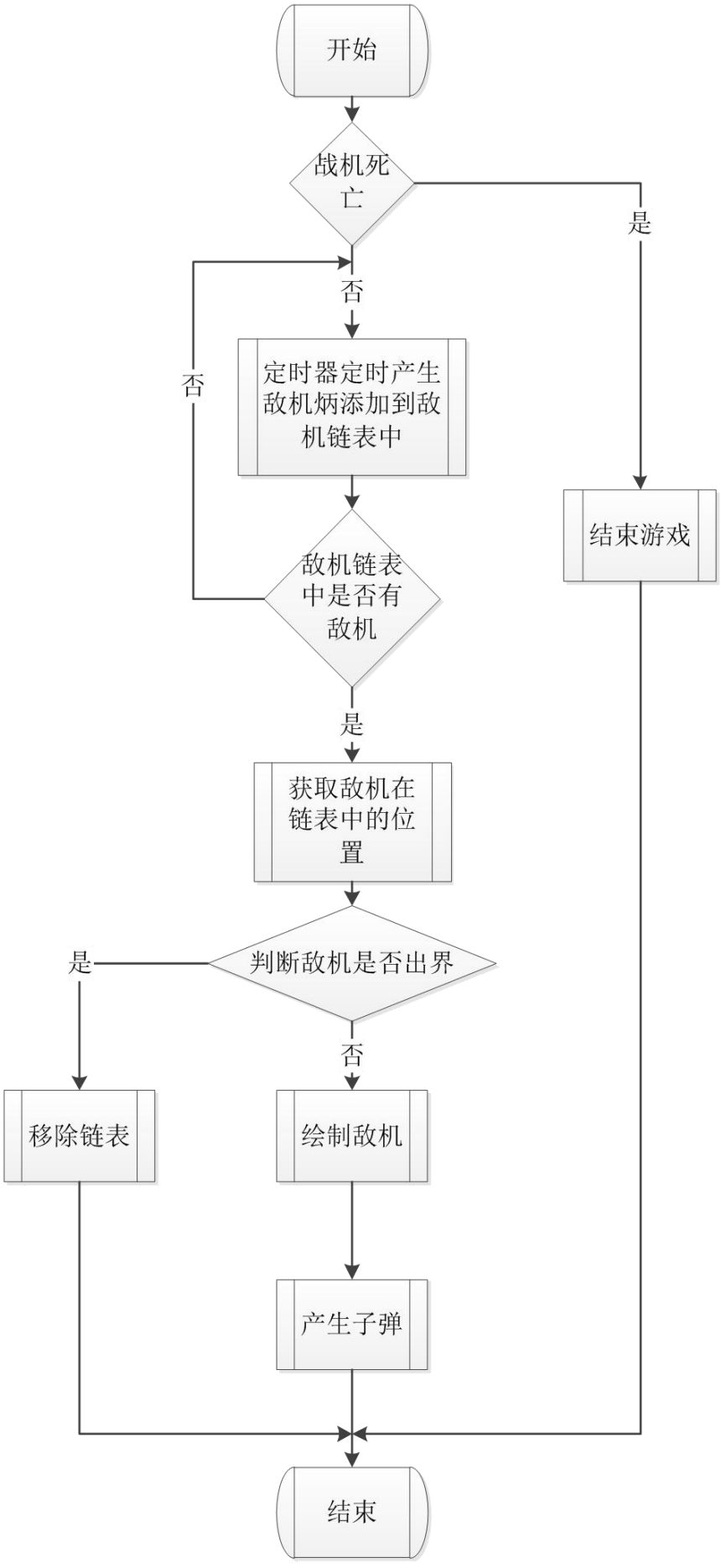


3.2.3 系统部分流程图

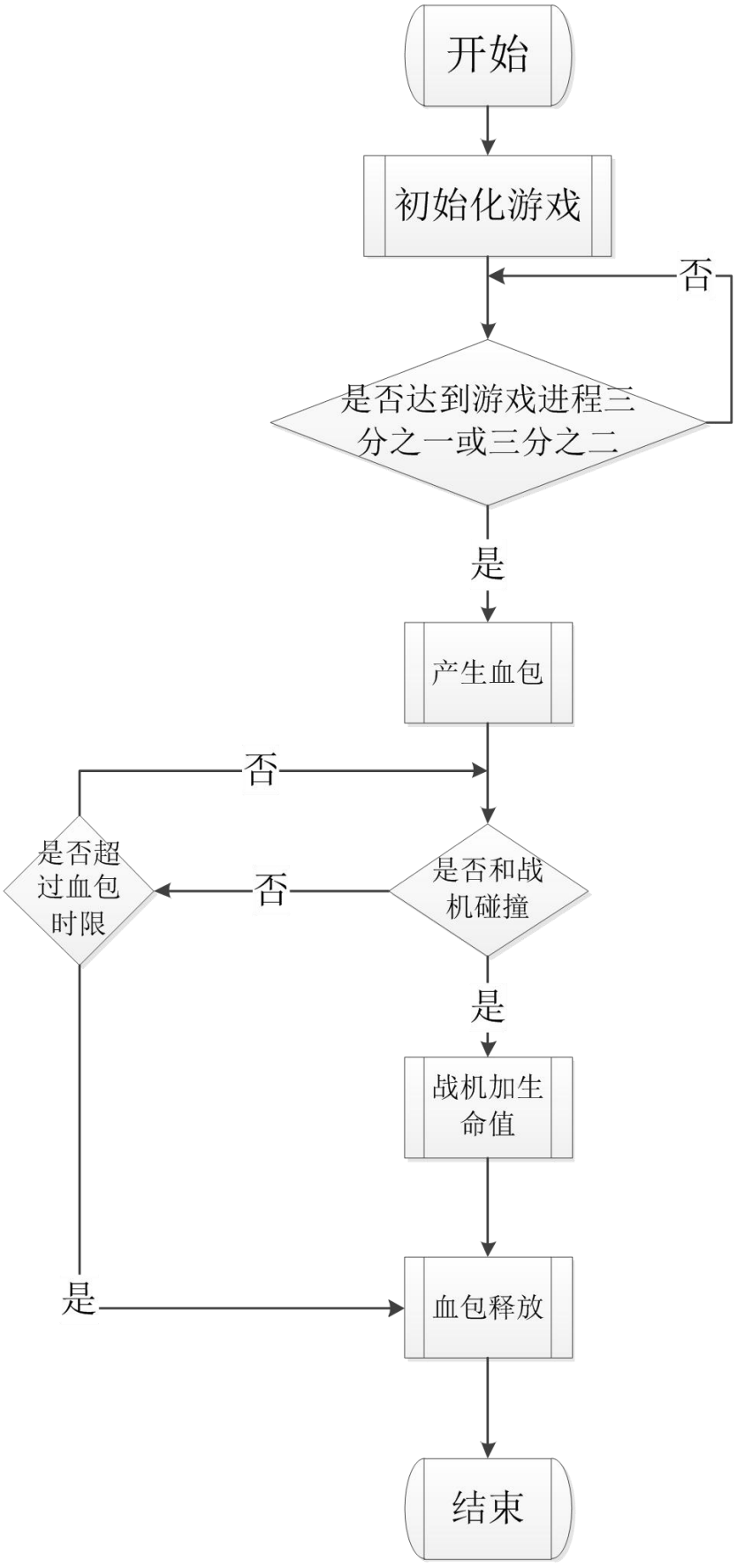
（1）飞机大战游戏执行流程图


开始
初始化游戏
炮弹碰撞战机
否
子弹碰撞敌机
是
否
否
否
战机生命值减少
敌机生命值减少
生命值为0
敌机生命值为0
是
是
是
否
较少一条命
释放敌机，得分
命数为0
进入Boss
是
是
否
结束游戏
打赢Boss，进入下一关
是
结束
初始化游戏

（2）定时器产生敌机和炸弹流程图


开始
战机死亡
是
否
否
定时器定时产生敌机炳添加到敌机链表中
结束游戏
敌机链表中是否有敌机
是
获取敌机在链表中的位置
是
判断敌机是否出界
否
移除链表
绘制敌机
产生子弹
结束

（3）血包执行流程图


开始
初始化游戏
否
是否达到游戏进程三分之一或三分之二
是
产生血包
否
是否超过血包时限
否
是否和战机碰撞
是
战机加生命值
是
血包释放
结束

4. 编码实现

4.1 双缓冲

```
//双缓冲
    CDC *pDC = GetDC();
    //获得客户区矩形区域
    GetClientRect(&rect);
    //内存缓冲 CDC
    CDC cdc;
    //内存中承载临时图像的缓冲位图
    CBitmap* cacheBitmap = new CBitmap;
    //用当前设备 CDC 初始化缓冲 CDC
    cdc.CreateCompatibleDC(pDC);
    //绑定 pDC 和缓冲位图的关系，cdc 先输出到缓冲位图中，输出完毕之后再一次性将缓冲位图输出到屏幕
    cacheBitmap->CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
    //替换 cdc 原本的缓冲区为缓冲位图，这样 cdc 输出的内容就写到了缓冲位图中
    CBitmap *pOldBit = cdc.SelectObject(cacheBitmap);

//将二级缓冲 cdc 中的数据推送到一级级缓冲 pDC 中，即输出到屏幕中
    pDC->BitBlt(0, 0, rect.Width(), rect.Height(), &cdc, 0, 0, SRCCOPY);
    //释放二级 cdc
    cdc.DeleteDC();
    //释放缓冲位图
    cacheBitmap->DeleteObject();
    //释放一级 pDC
    ReleaseDC(pDC);
```

4.2 滚动背景，并在滚动背景的初始化函数和释放函数添加背景音乐播放和释放

```
class CScene
{
```

```
    //成员变量
private:
    CImage  images[8];//滚动背景,0 位为开始图片，1-7 为七张不同的背景
    int     beginY;//背景的 Y 坐标

    bool    isStart;//是否开始

//成员函数
public:
    bool InitScene();//初始化场景
    void MoveBg();//移动背景
                ////绘制场景(注：这里 bufferDC 是引用参数)
    void StickScene(CDC* pDC, int index, CRect rClient);//传入 index-1 表示输出开始图片
    void ReleaseScene();//释放内存资源

//构造与析构
public:
    CScene(void);
    ~CScene(void);
};
//初始化场景
bool CScene::InitScene()
{
    //加载开始图片
    this->images[0].Load(_T("image\\start.bmp"));
    CString str;
    //如果加载失败，返回 false
    for (int i = 1; i <= 7; i++) {
        str.Format(_T("image\\background%d.bmp"), i);
```

```
        this->images[i].Load(str);
        if (images[i].IsNull())
            return false;
    }

    //开始为真，背景起始坐标为0
    this->isStart = true;
    this->beginY = 0;

    //播放背景音乐
    mciSendString(L"open sound\\background.mp3 alias bgm", NULL, 0, NULL);
    mciSendString(L"play bgm repeat", NULL, 0, NULL);
    return true;
}
//绘制场景
void CScene::StickScene(CDC* pDC,int index, CRect rClient)
{
    if (index == -1)
        index = 0;
    else
        index = index % 7 + 1;
    //设置缩放图片的模式为:COLORONCOLOR，以消除像素重叠
    pDC->SetStretchBltMode(COLORONCOLOR);
    //如果到了下边界，回到起点
    if (beginY >= rClient.Height())
    {
        beginY = 0;
        if (isStart)
            isStart = false;
    }
```

```
    //客户区高度
    int cltHeight = rClient.Height();
    rClient.bottom = cltHeight + beginY;
    rClient.top = beginY;
    //如果是开始就绘制起始背景
    if (isStart)
    {
        this->images[index].StretchBlt(*pDC, rClient, SRCCOPY);
    }
    //将下一张背景作为起始背景
    else
    {
        this->images[index].StretchBlt(*pDC, rClient, SRCCOPY);
    }
    //绘制下一张背景
    rClient.top -= cltHeight;
    rClient.bottom -= cltHeight;
    images[index].StretchBlt(*pDC, rClient, SRCCOPY);
}
//移动背景
void CScene::MoveBg()
{
    //移动背景
    beginY += 1;
}
//释放内存资源
void CScene::ReleaseScene()
{
    for (int i = 0; i <8; i++)
        if (!images[i].IsNull())
```

```
            images[i].Destroy();
        mciSendString(L"close bgm", NULL, 0, NULL);
    }
```

onTimer 中控制背景滚动：

```
    //输出背景
        if (isStarted == FALSE)
            scene.StickScene(&cdc, -1, rect);
        else
            scene.StickScene(&cdc, passNum, rect);
        if (nIDEvent == 4) {
            //滚动背景
            scene.MoveBg();
        }
```

4.3 显示战机

```
    if (myplane != NULL) {
            //reDraw(myplane, TRUE);
            myplane->Draw(&cdc,FALSE,isProtect);
        }
```

4.4 随机产生敌机和敌机炮弹、Boss 炮弹

```
    //随机添加敌机,敌机随机发射炸弹，此时敌机速度与数量和关卡有关
        if (myplane != NULL && isPause == 0&&isBoss==FALSE)
        {
            //敌机产生定时器触发
            if (nIDEvent == 2) {
                //根据关卡数产生敌机
                if (passNum <=5) {
                    //前五关只有一个方向的敌机
                    int direction = 1;//设置敌机的方向，从上方飞出
                    CEnemy *enemy = new CEnemy(direction, rect.right,
rect.bottom);
```

```
                enemyList.AddTail(enemy);//随机产生敌机
            }
            else if (passNum >5) {//第五关之后，两个方向的敌机
                int direction1 = 1; //设置敌机的方向，从上方飞出
                CEnemy *enemy1 = new CEnemy(direction1,rect.right,rect.bottom);
                enemy1->SetSpeed(ENEMY_SPEED+(rand() % 2 + passNum-1));
                enemyList.AddTail(enemy1);//随机产生敌机
                int direction2 = -1;//设置敌机的方向，从下方飞出
                CEnemy *enemy2 = new CEnemy(direction2, rect.right, rect.bottom);
                enemy2->SetSpeed(ENEMY_SPEED + (rand() % 2 + passNum - 1));
                enemyList.AddTail(enemy2);//随机产生敌机
            }
        }

        //超出边界的敌机进行销毁
        POSITION stPos = NULL, tPos = NULL;
        stPos = enemyList.GetHeadPosition();
        int direction = 1;
        while (stPos != NULL)
        {
            tPos = stPos;
            CEnemy *enemy = (CEnemy *)enemyList.GetNext(stPos);
            //判断敌机是否出界
            if (enemy->GetPoint().x<rect.left || enemy->GetPoint().x>rect.right
                || enemy->GetPoint().y<rect.top || enemy->GetPoint().y>rect.bottom)
            {
```

```
                    enemyList.RemoveAt(tPos);
                    delete enemy;
                }//if
                else
                {
                    //没出界，绘制
                    enemy->Draw(&cdc,passNum, FALSE);
                    //敌机炸弹产生定时器触发
                    if (nIDEvent == 3) {
                        //设置定时器产生敌机炸弹
                        PlaySound((LPCTSTR)IDR_WAV_BALL, AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
                        CBall * ball = new CBall(enemy->GetPoint().x +
ENEMY_HEIGHT / 2, enemy->GetPoint().y + ENEMY_HEIGHT, enemy->GetDirection());
                        ball->SetBallSpeed(BALL_SPEED + (passNum - 1) );
                        ballList.AddTail(ball);
                    }
                }//else
            }//while

             //判断产生的敌机炸弹是否出界，若已经出界，则删除该敌机炸弹
            POSITION stBallPos=NULL, tBallPos=NULL;
            stBallPos = ballList.GetHeadPosition();
            while (stBallPos != NULL)
            {
                tBallPos = stBallPos;
                ball = (CBall *)ballList.GetNext(stBallPos);
                if (ball->GetPoint().x<rect.left ||
ball->GetPoint().x>rect.right
                    || ball->GetPoint().y<rect.top ||
```

```
ball->GetPoint().y>rect.bottom)
                {
                    ballList.RemoveAt(tBallPos);
                    delete  ball;
                }//if
                else
                {
                    ball->Draw(&cdc,passNum, FALSE);
                }//else
            }//while
        }
        //Boss 产生炮弹，一次五发炮弹
        if (myplane != NULL && isPause == 0 && isBoss == TRUE) {
            //Boss 发射子弹
            //敌机炸弹产生定时器触发
            if (nIDEvent == 3) {
                //设置定时器产生敌机炸弹
                PlaySound((LPCTSTR)IDR_WAV_BALL,        AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
                CBall * ball1 = new CBall(boss->GetPoint().x + BOSS_WIDTH / 2,
boss->GetPoint().y + BOSS_HEIGHT, 1);
                ball1->SetBallSpeed(BALL_SPEED + (passNum - 1) * 2);
                ballList.AddTail(ball1);
                CBall   *   ball2   =   new   CBall(boss->GetPoint().x   +5,
boss->GetPoint().y + BOSS_HEIGHT, 1);
                ball2->SetBallSpeed(BALL_SPEED + (passNum - 1) * 2);
                ballList.AddTail(ball2);
                CBall * ball3 = new CBall(boss->GetPoint().x + BOSS_WIDTH -5,
boss->GetPoint().y + BOSS_HEIGHT,1);
                ball3->SetBallSpeed(BALL_SPEED + (passNum - 1) * 2);
```

```
                ballList.AddTail(ball3);
                CBall * ball4 = new CBall(boss->GetPoint().x + BOSS_WIDTH / 2+85,
boss->GetPoint().y + BOSS_HEIGHT, 1);
                ball4->SetBallSpeed(BALL_SPEED + (passNum - 1) * 2);
                ballList.AddTail(ball4);
                CBall * ball5 = new CBall(boss->GetPoint().x + BOSS_WIDTH / 2-85,
boss->GetPoint().y + BOSS_HEIGHT, 1);
                ball5->SetBallSpeed(BALL_SPEED + (passNum - 1) * 2);
                ballList.AddTail(ball5);
            }
            //显示 Boss 炸弹
            //判断产生的敌机炸弹是否出界，若已经出界，则删除该敌机炸弹
            POSITION stBallPos = NULL, tBallPos = NULL;
            stBallPos = ballList.GetHeadPosition();
            while (stBallPos != NULL)
            {
                tBallPos = stBallPos;
                ball = (CBall *)ballList.GetNext(stBallPos);
                if               (ball->GetPoint().x<rect.left               ||
ball->GetPoint().x>rect.right
                    ||              ball->GetPoint().y<rect.top              ||
ball->GetPoint().y>rect.bottom)
                {
                    ballList.RemoveAt(tBallPos);
                    delete  ball;
                }//if
                else
                {
                    ball->Draw(&cdc, FALSE);
                }//else
```

```
            }//while
        }
```

4.5 显示战机发射子弹

```
    if (myplane != NULL&& isPause == 0)
        {
            //声明战机子弹位置
            POSITION posBomb = NULL, tBomb = NULL;
            posBomb = bombList.GetHeadPosition();
            while (posBomb != NULL)
            {
                tBomb = posBomb;
                bomb = (CBomb *)bombList.GetNext(posBomb);
                if                (bomb->GetPoint().x<rect.left                ||
bomb->GetPoint().x>rect.right
                    ||                bomb->GetPoint().y<rect.top                ||
bomb->GetPoint().y>rect.bottom)
                {
                    //删除越界的子弹
                    bombList.RemoveAt(tBomb);
                    delete  bomb;
                }//if
                else
                    bomb->Draw(&cdc, FALSE);
            }//while
        }//if
```

4.6 碰撞检测，以战机子弹集中敌机为例

```
    if (myplane != NULL&& isPause == 0)
        {
            //声明战机子弹位置，敌机位置
            POSITION bombPos, bombTemp, enemyPos, enemyTemp;
```

```
            for (bombPos = bombList.GetHeadPosition(); (bombTemp = bombPos) !=
NULL;)
            {
                bomb = (CBomb *)bombList.GetNext(bombPos);
                for (enemyPos = enemyList.GetHeadPosition(); (enemyTemp =
enemyPos) != NULL;)
                {
                    enemy = (CEnemy *)enemyList.GetNext(enemyPos);
                    //获得战机子弹的矩形区域
                    CRect bombRect = bomb->GetRect();
                    //获得敌机的矩形区域
                    CRect enemyRect = enemy->GetRect();
                    //判断两个矩形区域是否有交接
                    CRect tempRect;
                    if (tempRect.IntersectRect(&bombRect, enemyRect))
                    {
                        //将爆炸对象添加到爆炸链表中
                        CExplosion          *explosion          =          new
CExplosion((bomb->GetPoint().x + BOMB_WIDTH / 2 - EXPLOSION_WIDTH / 2),
(bomb->GetPoint().y + BOMB_HEIGHT / 2 - EXPLOSION_WIDTH / 2));
                        explosionList.AddTail(explosion);
                        PlaySound((LPCTSTR)IDR_WAV_EXPLOSION,
AfxGetInstanceHandle(), SND_RESOURCE | SND_ASYNC);
                        //爆炸后删除子弹
                        bombList.RemoveAt(bombTemp);
                        delete  bomb;
                        //指向下一个
                        bomb = NULL;
                        //敌机生命值减少
                        enemy->life -= (1 + isUpdate);
```

```
                    if (enemy->life <= 0) {
                        //增加得分
                        passScore++;
                        //删除敌机
                        enemyList.RemoveAt(enemyTemp);
                        delete  enemy;
                    }
                    //炮弹已删除，直接跳出本循环
                    break;
                }
            }

            if (isBoss == TRUE&&bomb != NULL) {
                //获得战机子弹的矩形区域
                CRect bombRect = bomb->GetRect();
                //获得 Boss 的矩形区域
                CRect bossRect = boss->GetRect();
                //判断两个矩形区域是否有交接
                CRect tempRect;
                if (tempRect.IntersectRect(&bombRect, bossRect))
                {
                    //将爆炸对象添加到爆炸链表中
                    CExplosion          *explosion          =          new
CExplosion((bomb->GetPoint().x  +  BOMB_WIDTH  /  2  -  EXPLOSION_WIDTH  /  2),
(bomb->GetPoint().y + BOMB_HEIGHT / 2 - EXPLOSION_WIDTH / 2));
                    explosionList.AddTail(explosion);
                    PlaySound((LPCTSTR)IDR_WAV_EXPLOSION,
AfxGetInstanceHandle(), SND_RESOURCE | SND_ASYNC);
                    //爆炸后删除子弹
                    bombList.RemoveAt(bombTemp);
```

```
                    delete  bomb;
                    bomb = NULL;
                    //是 Boss，不删除敌机，只扣血
                    bossBlood -= (1 + isUpdate);
                    if (bossBlood <= 0) {
                        delete boss;
                        boss = NULL;
                        //过关的标志变量
                        delete  myplane;
                        myplane = NULL;
                        isPass = 1;
                        isBoss = FALSE;
                    }
                }
            }
        }//for
    }//if
```

4.7 显示爆炸效果

```
if (myplane != NULL&&isPause == FALSE) {
        POSITION explosionPos, explosionTemp;
        explosionPos = explosionList.GetHeadPosition();
        //检索爆炸链表，非空时在所在位置显示
        while (explosionPos != NULL)
        {
            explosionTemp = explosionPos;
            explosion = (CExplosion *)explosionList.GetNext(explosionPos);
            BOOL flag = explosion->Draw(&cdc, FALSE);
            //爆炸 8 帧结束，删除爆炸对象
            if (flag == EXPLOSION_STATUS_STOP) {
                explosionList.RemoveAt(explosionTemp);
```

```
                delete explosion;
            }
        }//while
    }
```

4.8 血包功能

游戏三分之一和三分之二进程时刻出现血包

```
//开启血包
    if (myplane != NULL && myLife > 0)
    {
        //关卡打了三分之一三分之二处出现血包
        if (passScore > (PASS_SCORE + passNum * 5)*lifeCount/3)
        {
            //若屏幕中有未吃掉的血包，这次不产生血包
            if (bloodExist == FALSE) {
                lifeCount++;
                //产生血包
                blood = new CBlood(rect.right,rect.bottom);
                bloodList.AddTail(blood);
                bloodExist = TRUE;
                SetTimer(6, 10000, NULL);
            }else lifeCount++;
        }
    }
//血包定时器，10 秒后血包消失
    if (nIDEvent == 6&&isPause==0) {
        KillTimer(6);
        bloodExist = FALSE;
        //声明血包位置
        POSITION bloodPos, bloodTemp;
        for (bloodPos = bloodList.GetHeadPosition(); (bloodTemp = bloodPos) !=
```

```
NULL;)
            {
                blood = (CBlood *)bloodList.GetNext(bloodPos);
                bloodList.RemoveAt(bloodTemp);
                delete  blood;
            }
        }
    //显示血包
        if (myplane != NULL&&isPause == FALSE)
        {
            POSITION bloodPos;
            bloodPos = bloodList.GetHeadPosition();
            //检索血包链表，非空时在所在位置显示
            while (bloodPos != NULL)
            {
                blood = (CBlood *)bloodList.GetNext(bloodPos);
                blood->Draw(&cdc, FALSE);
            }//while
        }
    //血包碰撞检测
    if (myplane != NULL&& isPause == 0)
        {
            //声明血包位置
            POSITION bloodPos, bloodTemp;
            for (bloodPos = bloodList.GetHeadPosition(); (bloodTemp = bloodPos) !=
NULL;)
            {
                blood = (CBlood *)bloodList.GetNext(bloodPos);
                //获得血包矩形
                CRect bloodbRect = blood->GetRect();
```

```
                //获得战机矩形
                CRect planeRect = myplane->GetRect();
                //判断两个矩形区域是否有交接
                CRect tempRect;
                if (tempRect.IntersectRect(&bloodbRect, planeRect))
                {
                    //加血效果
                    myLife += 5;
                    if (myLife > DEFAULT_LIFE)
                        myLife = DEFAULT_LIFE;
                    // TODO 声音
                    PlaySound((LPCTSTR)IDR_WAV_BLOOD,   AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
                    //加血后血包删除
                    bloodList.RemoveAt(bloodTemp);
                    delete  blood;
                    break;
                }//if
            }//for
        }//if
```

4.9 通关和死亡消息页面

```
    if (isStop == FLAG_RESTART) {
            HFONT textFont;
            textFont = CreateFont(18, 9, 0, 0, 0, 0, 0, 0, 0, 0, 0, 100, 10, 0);
            cdc.SelectObject(textFont);
            //设置透明背景
            cdc.SetBkMode(TRANSPARENT);
            cdc.SetTextColor(RGB(255, 255, 255));
            cdc.TextOutW(rect.right/2-150, rect.bottom/2-30, _T("哇，恭喜你已通
关！是否重新开始？Y/N"));
```

```
        }
        else if (isStop == FLAG_STOP) {
            HFONT textFont;
            textFont = CreateFont(18 , 9, 0, 0, 0, 0, 0, 0, 0, 0, 100, 10, 0);
            cdc.SelectObject(textFont);
            //设置透明背景
            cdc.SetBkMode(TRANSPARENT);
            cdc.SetTextColor(RGB(255, 255, 255));
            //显示最后结果
            CString str;
            cdc.TextOutW(rect.right / 2 - 100, rect.bottom / 2 - 30, _T("GAME OVER!
"));
            str.Format(_T("您的得分为：%d"), myScore);
            cdc.TextOutW(rect.right / 2 - 100, rect.bottom / 2 - 10, str);
            cdc.TextOutW(rect.right / 2 - 100, rect.bottom / 2 +10, _T("COME ON !
重新开始？Y/N"));
        }
```

4.10 魔法值控制维护

```
    //控制魔法值
        if (nIDEvent == 5) {
            if (myplane != NULL&&isPause == 0) {
                //防护罩和战机升级没打开，魔法值递增
                if (isProtect==FALSE&&isUpdate==FALSE) {
                    magicCount++;
                    if (magicCount > 10)
                        magicCount = 10;
                }
                //判断是否打开防护罩
                if (isProtect == TRUE) {
                    //开启防护罩魔法值递减
```

```
                magicCount --;
                if (magicCount <= 0) {
                    magicCount = 0;
                    isProtect = FALSE;
                }
            }
            //判断是否升级战机
            if (isUpdate == TRUE) {
                //战机升级，魔法值递减
                magicCount--;
                if (magicCount <= 0) {
                    magicCount = 0;
                    isUpdate = FALSE;
                    myplane->SetIsUpdate(isUpdate);
                }
            }
        }
    }
```

4.11 得分到达关卡需求，进入 Boss

```
// 进入 Boss
    int pScore = PASS_SCORE + passNum * 5;
    // TODO 调试条件
    //if (myplane != NULL && passScore >= 3 && isPause == 0&&isBoss==FALSE)
    if (myplane != NULL && passScore >= pScore && isPause == 0 && isBoss == FALSE)
    {
        //进入 Boss
        isBoss = TRUE;
        boss = new CBoss(1, rect.right, rect.bottom);
        boss->SetSpeed(BOSS_SPEED+passNum-1);
        boss->life = BOSS_LIFE + passNum * 50;//Boss 总血量
```

```
            bossBlood= BOSS_LIFE + passNum * 50;//当前 Boss 血量
            //Boss 出场，暂停游戏
            bossLoaded = FALSE;

            //重新设置 Boss 的子弹产生频率，增强 Boss 子弹发射频率
            KillTimer(3);
            SetTimer(3, 2000 - passNum * 180, NULL);
        }
    //显示 Boss
        if (myplane != NULL &&boss!=NULL&& isPause == 0 && isBoss == TRUE) {
            BOOL status = boss->Draw(&cdc,passNum, FALSE);
            if (status == TRUE)
                bossLoaded = TRUE;
        }
```

4.12 检测标记位 isPass，判断打赢 Boss，进入下一关

```
    if (isPass == 1)
        {
            isPass = FALSE;
            if (passNum ==10)//10 关
            {
                //重新初始化数据
                KillTimer(1);
                KillTimer(2);
                KillTimer(3);
                //KillTimer(4);
                KillTimer(5);
                myplane = new CMyPlane(FALSE);
                isPause = TRUE;

                isStop = FLAG_RESTART;
```

```
                //清屏
                CBitmap* tCache = cacheBitmap;
                cacheBitmap = new CBitmap;
                cacheBitmap->CreateCompatibleBitmap(pDC,          rect.Width(),
rect.Height());
                cdc.SelectObject(cacheBitmap);
                delete tCache;
            }//if
            else
            {
                KillTimer(1);
                KillTimer(2);
                KillTimer(3);
                //KillTimer(4);
                KillTimer(5);
                isPause = TRUE;
                //保存所需数据
                int tScore = myScore + passScore;
                int tPassNum = passNum + 1;
                int tTest = test;
                int magic = magicCount;
                //重新开始游戏
                Restart();
                myScore = tScore;
                passNum = tPassNum;
                magicCount = magic;
                test = tTest;
            }//else
        }//if
```

4.13 消息监听

4.13.1 方向上键或 W 键：if (myplane != NULL && (GetKeyState(VK_UP) < 0 || GetKeyState('W') < 0) && isPause == 0)…

4.13.2 方向下键或 S 键：if (myplane != NULL && (GetKeyState(VK_DOWN) < 0 || GetKeyState('S') < 0) && isPause == 0)…

4.13.3 方向左键或 A 键：if (myplane != NULL && (GetKeyState(VK_LEFT) < 0 || GetKeyState('A') < 0) && isPause == 0)…

4.13.4 方向右键或 D 键：if (myplane != NULL && (GetKeyState(VK_RIGHT) < 0 || GetKeyState('D') < 0) && isPause == 0)…

4.13.5 空格键发射子弹：

```
if (myplane != NULL && (GetKeyState(VK_SPACE) < 0) && isPause == 0 && bossLoaded == TRUE) {
            //空格键发射子弹
            CBomb *Bomb1 = new CBomb(myplane->GetPoint().x+20, myplane->GetPoint().y, 1,isUpdate);
            PlaySound((LPCTSTR)IDR_WAV_BOMB, AfxGetInstanceHandle(), SND_RESOURCE | SND_ASYNC);
            bombList.AddTail(Bomb1);
            CBomb *Bomb2 = new CBomb(myplane->GetPoint().x + PLANE_WIDTH-50, myplane->GetPoint().y, 1, isUpdate);
            bombList.AddTail(Bomb2);
            //子弹自动飞行，在 Timer 中绘制
        }
```

4.13.6 Z 键暂停：if (myplane != NULL&&GetKeyState('Z') < 0)…

4.13.7 X 键发大招清屏：

```
if (myplane != NULL&&GetKeyState('X') < 0 && isPause == 0 && bossLoaded == TRUE)
        {
            //战机发大招
            if (magicCount >= 10) {
                magicCount -= 10;
```

```
                    //清空敌机
                    POSITION enemyPos, enemyTemp;
                    for (enemyPos = enemyList.GetHeadPosition(); (enemyTemp =
enemyPos) != NULL;)
                    {
                        enemy = (CEnemy *)enemyList.GetNext(enemyPos);
                        //将爆炸对象添加到爆炸链表中
                        CExplosion          *explosion          =          new
CExplosion((enemy->GetPoint().x + ENEMY_WIDTH / 2), (enemy->GetPoint().y +
ENEMY_HEIGHT / 2));
                        explosionList.AddTail(explosion);
                        PlaySound((LPCTSTR)IDR_WAV_DAZHAO,
AfxGetInstanceHandle(), SND_RESOURCE | SND_ASYNC);

                        //删除敌机
                        enemyList.RemoveAt(enemyTemp);
                        delete  enemy;
                        //增加得分
                        passScore++;
                    }//for
                if(isBoss==TRUE) {
                    //将爆炸对象添加到爆炸链表中
                    CExplosion *explosion = new CExplosion(boss->GetPoint().x +
BOSS_WIDTH / 2, boss->GetPoint().y + BOSS_HEIGHT / 2);
                    explosionList.AddTail(explosion);
                    PlaySound((LPCTSTR)IDR_WAV_DAZHAO,  AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
                    bossBlood -= 50;
                    if (bossBlood <= 0) {
                        //boss死，过关
```

```
                    //过关的标志变量
                    delete boss;
                    boss = NULL;
                    //过关的标志变量
                    isPause = TRUE;
                    CMyPlane* temp = myplane;
                    myplane = new CMyPlane(FALSE);
                    delete  temp;
                    temp = NULL;
                    isPass = 1;
                    isBoss = FALSE;
                }
            }
             //清空敌机炮弹
            POSITION ballPos, ballTemp;
            for (ballPos = ballList.GetHeadPosition(); (ballTemp = ballPos) !=
NULL;)
            {
                ball = (CBall *)ballList.GetNext(ballPos);
                //删除敌机炮弹
                ballList.RemoveAt(ballTemp);
                delete  ball;
            }//for
        }
    }//if
```

4.13.8 C键开启防护罩：`if (myplane != NULL&&GetKeyState('C') < 0 && isPause == 0)`

4.13.9 V键战机升级：`if (myplane != NULL&&GetKeyState('V') < 0 && isPause == 0)`

4.13.10 Y键开启无敌模式或响应死亡、重开游戏界面操作：

```
if (GetKeyState('Y') < 0 )
```

```
{
        if (isStop == FALSE) {
            //无敌模式开关
            if (test == FALSE)
                test = TRUE;
            else test = FALSE;
        }
        else {
            isStop = FALSE;
            Restart();
        }
    }
```

4.13.11 N键响应结束、重开游戏界面操作：

```
if (GetKeyState('N') < 0) {
        if (isStop!=FALSE) {
            MyDialog dialog;
            dialog.DoModal();
        }
    }
```

4.13.12 开始界面按空格键进入游戏：

```
//按空格进入游戏
    if (isStarted == FALSE && (GetKeyState(VK_SPACE) < 0)) {
        isStarted = TRUE;
    }
```

4.13.13 鼠标移动监听：

```
void CPlaneWarView::OnMouseMove(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值

    if (myplane!=NULL && isPause == 0 ) {
```

```
            //绘制新游戏对象
            myplane->SetPoint(point.x,point.y);
        }
        CView::OnMouseMove(nFlags, point);
    }
```

4.13.14 鼠标左键发射子弹和开始界面进入游戏:

```
    void CPlaneWarView::OnLButtonDown(UINT nFlags, CPoint point)
    {
        // TODO: 在此添加消息处理程序代码和/或调用默认值

        if (myplane!=NULL&&isPause == 0 && bossLoaded == TRUE)
        {
            CBomb *Bomb1 = new CBomb(myplane->GetPoint().x, myplane->GetPoint().y,
1, isUpdate);
            PlaySound((LPCTSTR)IDR_WAV_BOMB,             AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
            bombList.AddTail(Bomb1);
            CBomb  *Bomb2  =  new  CBomb(myplane->GetPoint().x  +  BOMB_DISTANCE,
myplane->GetPoint().y, 1, isUpdate);
            bombList.AddTail(Bomb2);
            //子弹自动飞行，在 Timer 中绘制
        }
        if (isStarted == FALSE) {
            isStarted = TRUE;
        }

        CView::OnLButtonDown(nFlags, point);
    }
```

4.14 生命周期:

4.14.1 窗口销毁

```
void CPlaneWarView::OnDestroy()
{
    // 销毁指针资源
    if (myplane != NULL)
        delete myplane;
    if (boss != NULL)
        delete boss;
        //释放内存资源
    scene.ReleaseScene();
    CView::OnDestroy();
}
```

4.14.2游戏重新开始

```
void CPlaneWarView::Restart()
{
    // TODO: 在此处添加游戏重新开始初始化参数
    //战机重新加载
    myplane = new CMyPlane(FALSE);

    //清空敌机链表
    if (enemyList.GetCount() > 0)
        enemyList.RemoveAll();
    //清空战机链表
    if (meList.GetCount() > 0)
        meList.RemoveAll();
    //清空战机子弹链表
    if (bombList.GetCount() > 0)
        bombList.RemoveAll();
    //清空敌机炸弹链表
    if (ballList.GetCount() > 0)
        ballList.RemoveAll();
```

```
    //清空爆炸链表
    if (explosionList.GetCount() > 0)
        explosionList.RemoveAll();
    //清空血包列表
    if (bloodList.GetCount() > 0)
        bloodList.RemoveAll();

    //参数重新初始化
    myLife = DEFAULT_LIFE;
    lifeNum = DEFAULT_LIFE_COUNT;
    myScore = 0;
    passScore = 0;
    passNum = DEFAULT_PASS;
    isPass = 0;
    isPause = 0;
    lifeCount = 1;
    magicCount = 0;
    bloodExist = FALSE;
    bossBlood = BOSS_LIFE;
    isBoss = FALSE;
    bossLoaded = TRUE;
    isProtect = FALSE;
    isUpdate = FALSE;
    test = FALSE;
    boss = NULL;
    SetMyTimer();
}
```

4.14.3游戏暂停

```
void CPlaneWarView::Pause()
{
```

```
        // TODO: 在此处添加游戏暂停操作
        isPause = TRUE;
        Sleep(1000);
    }
```

4.14.4 生命值归零，游戏结束

```
    void CPlaneWarView::gameOver(CDC* pDC,CDC& cdc,CBitmap* cacheBitmap)
    {
        //结束游戏界面
        //释放计时器
        KillTimer(1);
        KillTimer(2);
        KillTimer(3);
        //KillTimer(4);
        KillTimer(5);
        //计算最后得分
        myScore += passScore;
        //播放游戏结束音乐
        PlaySound((LPCTSTR)IDR_WAV_GAMEOVER,             AfxGetInstanceHandle(),
SND_RESOURCE | SND_ASYNC);
        //清屏
        CBitmap* tCache = cacheBitmap;
        cacheBitmap = new CBitmap;
        cacheBitmap->CreateCompatibleBitmap(pDC, rect.Width(), rect.Height());
        cdc.SelectObject(cacheBitmap);
        //内存中承载临时图像的缓冲位图
        delete tCache;
        isStop = FLAG_STOP;
    }
```

5. 实训中遇到的主要问题及解决方法

5.1 滚动背景实现问题

要实现滚动背景，需要在背景图上取出客户区矩形大小的区域，并按照 Timer 进行递进，并且要在背景图边界处衔接好返回背景图开头位置，实现一张背景图的循环反复，达到滚动背景图的目的，刚开始由于不懂得如何实现开头结尾处的衔接问题，导致滚动背景图实现难度大，后来经由网上查阅的资料得知要把背景图加载两份，第二份背景图开头衔接在第一份结尾处，让客户区矩形在该两份背景图上进行滑动，当滑动到第二份背景图时，再把第一份接到第二份结尾处，从而实现循环反复滚动背景。其中实现的难点在于控制好客户区矩形坐标和背景图上要显示的图片块位于图片上的坐标的对应关系。

5.2 背景音乐循环播放和游戏音效同时输出问题

由于平时接触 MFC 太少，对 MFC 操作多媒体文件颇为陌生，刚开始使用的 PlaySound 可以简单的实现音频的输出，但是该函数只支持单个音轨的播放，不支持背景音乐和游戏操作音效的同时播放，后来经过大量的查找资料和代码样例，终于找到了使用 mciSendString 进行背景音乐的播放，而使用原来的 PlaySound 播放操作音效，达到了背景音乐和操作音效的同时播放的目的。

5.3 多帧位图素材的动画特效实现问题

飞机大战中的爆炸显示是通过一张带有八个帧的位图进行连续输出实现爆炸特效，刚开始只是简单的在一个 while 循环中循环八次，结果不尽如人意，后来联想到帧和时间的对应关系，在每一次 onTimer 调用时输出一帧，并在爆炸对象中用 progress 标记位记录播放位置，等到八个帧播放结束时，返回播放结束标记位，在 onTimer 中检测并删除播放完成的该爆炸对象。

5.4 游戏结束和游戏重新开始的游戏资源维护问题

该游戏由于实现了多个额外功能，且都是带有全局意义的，因此放置了较多标记位来标记每个状态，通过这些标记位来控制游戏进程中的各个状态，因此在游戏结束和重新开始游戏时，各个标记位的正确重置将会决定重新开始游戏之后的游戏状态。还由于这些操作可能会在 onTimer 函数调用过程中进行中断，这和 onTimer 中使用的双缓冲的 pDC 和 cdc 的申请和释放息息相关，中断操作必须对 pDC 和 cdc 进行正确的处理才会防止游戏过程中的意外崩溃出现。

6. 实训体会

由于对 MFC 接触不多，那次本次实训开发飞机大战相当于是零基础，因此在开发过程中走了很多弯路，但是也是这次的实训，让我了解了 MFC 的事件处理机制，了解了如何在屏幕中绘制需要向用户展示的图形信息，学会了使用基本的操作对屏幕中绘制的图形进行控制，也了解了 Windows 的屏幕定时刷新的机制，学习了微软封装好的一些 C++的类库，学会遇到问题到 MSDN 中进行 API 查询，并对 VS 的使用和编程也进行了提高。

在开发过程中遇到了大量的异常和中断，通过此次开发，也学会了遇到错误如何慢慢通过 IDE 的错误信息进行错误查找和代码排错更正，通过添加断点调试程序一步步监视程序运行的具体过程，从而找到程序运行出错的原因。

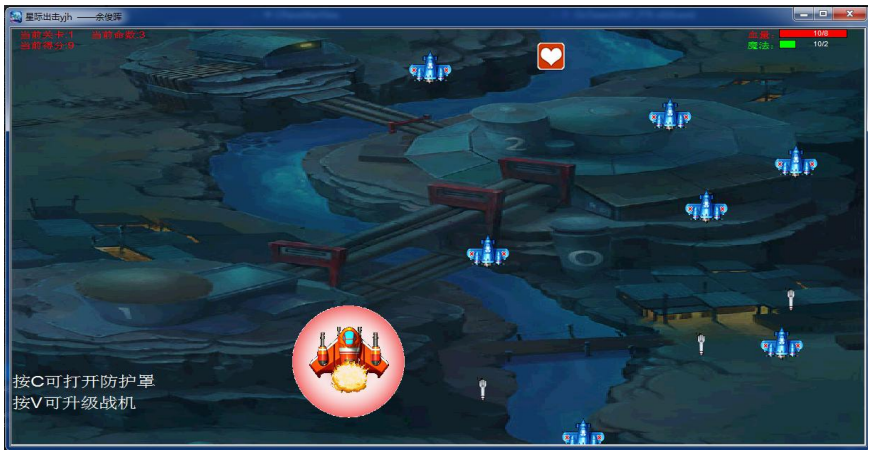
通过此次的实训，对 MFC 入了门，真正实现了图形界面的底层操作，消除了以前对于 MFC 的一些恐惧心理，让我也对 MFC 产生了兴趣，在之后的学习中，我会多找机会使用 MFC 进行一些开发，做一些实用的程序出来。

附录：

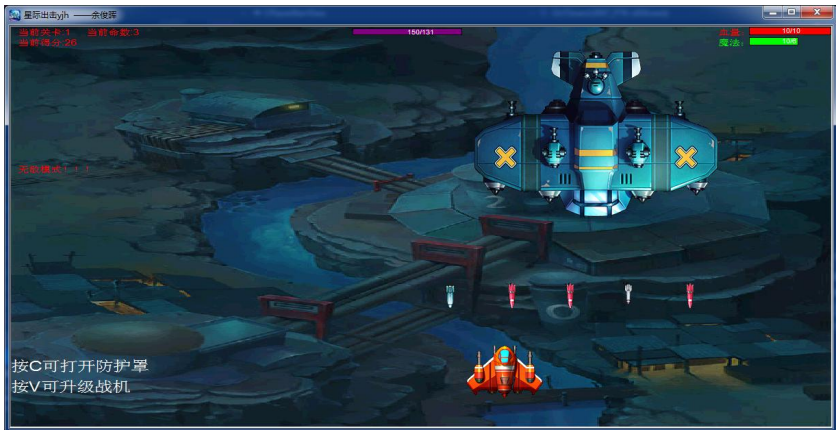
游戏相关截图：

游戏开始界面：

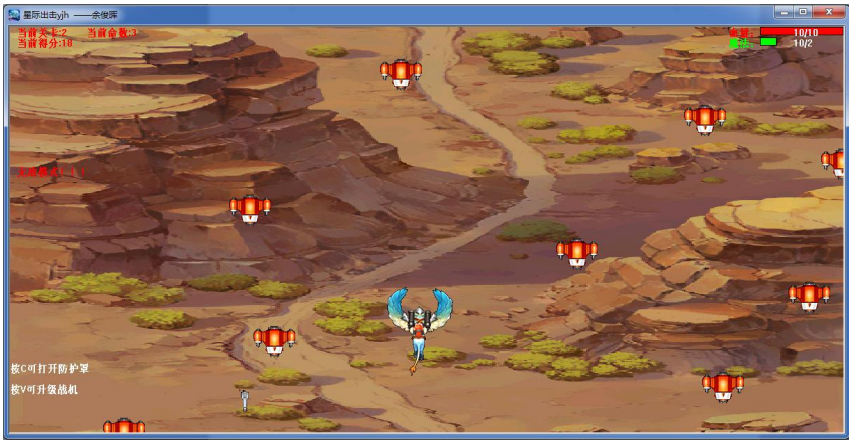
护盾：



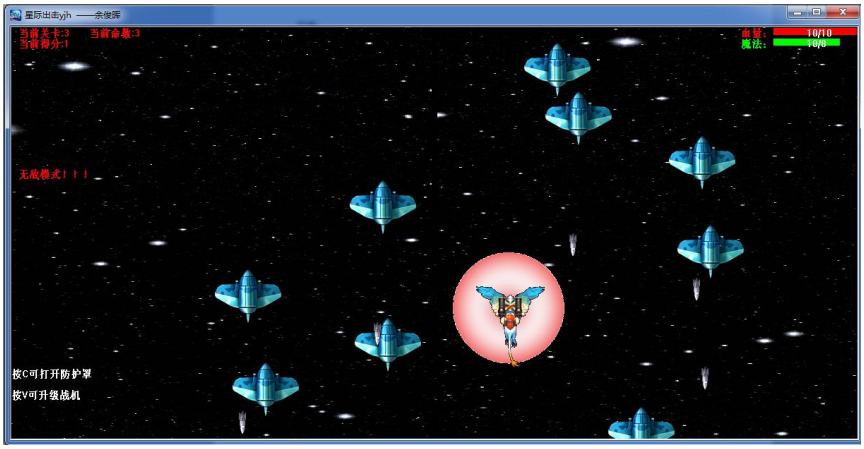
Boss：



变身：

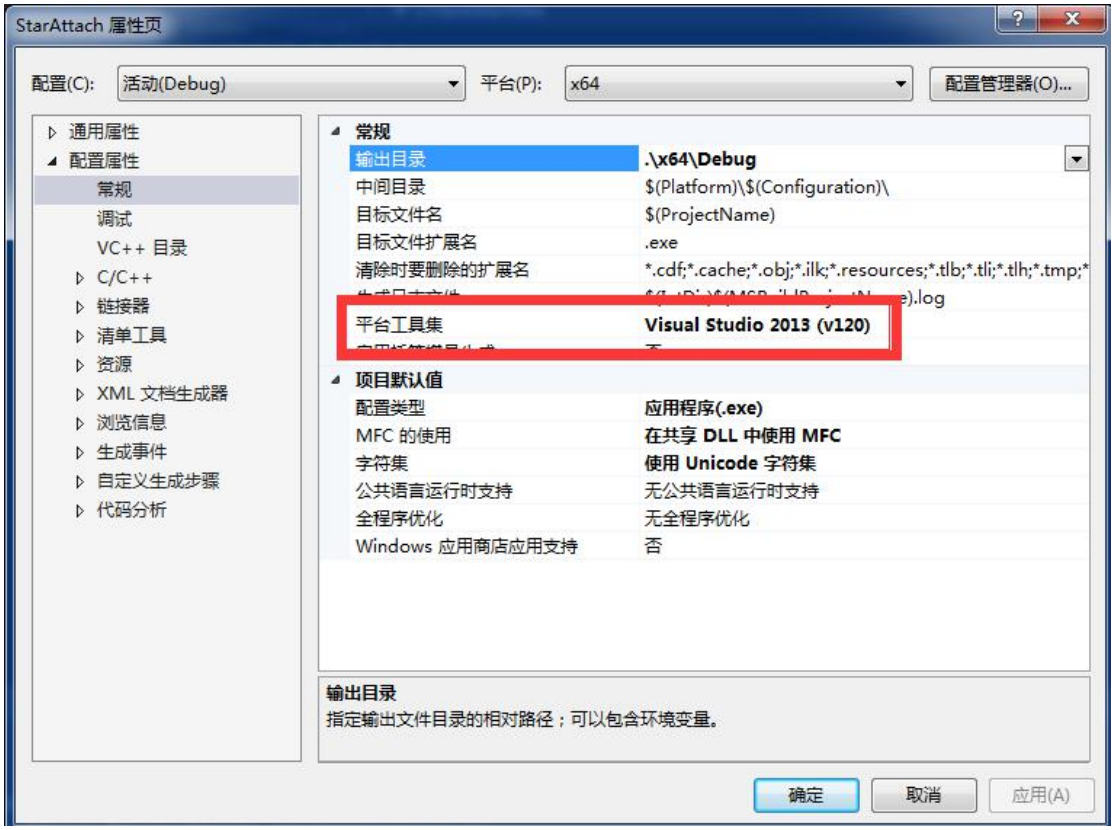


分数达到一定要求进入下一关：



特别说明：

本项目在 vs2013 下面编写，其平台工具集需要改成相对应的版本-v120,，在项目属性中修改


StarAttach 属性页
配置(C): 活动(Debug)
平台(P): x64
配置管理器(O)...
通用属性
配置属性
常规
调试
VC++ 目录
C/C++
链接器
清单工具
资源
XML 文档生成器
浏览信息
生成事件
自定义生成步骤
代码分析
常规
输出目录 .\x64\Debug
中间目录 $(Platform)\$(Configuration)\
目标文件名 $(ProjectName)
目标文件扩展名 .exe
清除时要删除的扩展名 *.cdf;*.cache;*.obj;*.ilk;*.resources;*.tlb;*.tli;*.tlh;*.tmp;*
平台工具集 Visual Studio 2013 (v120)
项目默认值
配置类型 应用程序(.exe)
MFC 的使用 在共享 DLL 中使用 MFC
字符集 使用 Unicode 字符集
公共语言运行时支持 无公共语言运行时支持
全程序优化 无全程序优化
Windows 应用商店应用支持 否
输出目录
指定输出文件目录的相对路径；可以包含环境变量。
确定
取消
应用(A)