```cpp
//CFoeBig
#include"CFoeBig.h"
#include"../Config/Config.h"
void CFoeBig::InitFoe() {
    ::loadimage(&m_img, L"./res/foebig.jpg");
    m_x = rd()% (BACK_WIDTH- FOEBIG_WIDTH+1) ;
    m_y = -FOEBIG_HEIGHT;
    m_blood = FOEBIG_BLOOD;
    m_showId = FOEBIG_INIT_ID;
}
void CFoeBig::ShowFoe() {
    //屏蔽图 位或
    ::putimage(m_x, m_y/*显示到窗口的位置*/, FOEBIG_WIDTH, FOEBIG_HEIGHT/*要显示图片的大小*/,
        &m_img/*要显示那个图片*/, FOEBIG_WIDTH, FOEBIG_HEIGHT*(FOEBIG_INIT_ID- m_showId)/* 从图片的哪个位置开始显示  */, SRCPAINT/*传输方式*/);

    ::putimage(m_x, m_y/*显示到窗口的位置*/, FOEBIG_WIDTH, FOEBIG_HEIGHT/*要显示图片的大小*/,
        &m_img/*要显示那个图片*/, 0, FOEBIG_HEIGHT * (FOEBIG_INIT_ID - m_showId)/* 从图片的哪个位置开始显示  */, SRCAND/*传输方式*/);
}
bool CFoeBig::IsHitPlane(int x, int  y) {
    int x1 = x + PLANE_WIDTH / 2;
    //y

    //x
    int y2 = y + PLANE_HEIGHT / 2;

    int x3 = x + PLANE_WIDTH;
    //y2

    if (m_x <= x1 && x1 <= (m_x + FOEBIG_WIDTH) &&
        m_y <= y && y <= (m_y + FOEBIG_HEIGHT)
        ) {
        return true;
    }

    if (m_x <= x && x <= (m_x + FOEBIG_WIDTH) &&
        m_y <= y2 && y2 <= (m_y + FOEBIG_HEIGHT)
        ) {
        return true;
    }


    if (m_x <= x3 && x3 <= (m_x + FOEBIG_WIDTH) &&
        m_y <= y2 && y2 <= (m_y + FOEBIG_HEIGHT)
        ) {
        return true;
    }
    return false;
}

//CFoeList
```

```cpp
#include"CFoeList.h"
#include<typeinfo>
using namespace std;
#include"CFoeBig.h"
#include"CFoeMid.h"
#include"CFoeSma.h"
#include"../Config/Config.h"

CFoeList::CFoeList() {}
CFoeList::~CFoeList() {
    //11. del yuesl, 遍历两个链表，正常链表、爆炸链表,删除里面的每个敌人飞机
    list<CFoe*>::iterator ite = m_lstFoe.begin();
    while (ite != m_lstFoe.end()) {
        if (*ite) {
            delete (*ite);
            (*ite) = nullptr;
        }
        ++ite;
    }

    ite = m_lstBoomFoe.begin();
    while (ite != m_lstBoomFoe.end()) {
        if (*ite) {
            delete (*ite);
            (*ite) = nullptr;
        }
        ++ite;
    }

}
void CFoeList::ShowAllFoe() {
    //12. del yuesl ，遍历两个链表，显示每一个敌人飞机
    for (CFoe* p : m_lstFoe) {
        if (p) p->ShowFoe();
    }

    for (CFoe* p : m_lstBoomFoe) {
        if (p) p->ShowFoe();
    }
}
void CFoeList::MoveAllFoe() {
    //13. del yuesl，遍历链表，移动所有敌人飞机，注意每种敌人飞机移动时，传递的步长不同
    list<CFoe*>::iterator ite = m_lstFoe.begin();
    while (ite != m_lstFoe.end()) {
        if (*ite) {
            //RTTI: RunTime type Id  运行时类型识别
            //typeinfo 包含表达式类型的信息的结构 =typeid( 表达式/类型  );
//sizeof()
            /*int a = 0;
            typeid(a*10) == typeid(int)*/

            if (typeid(**ite) == typeid(CFoeBig)) (*ite)-
>MoveFoe(FOEBIG_MOVE_STEP);
            if (typeid(**ite) == typeid(CFoeMid)) (*ite)-
>MoveFoe(FOEMID_MOVE_STEP);
            if (typeid(**ite) == typeid(CFoeSma)) (*ite)-
>MoveFoe(FOESMA_MOVE_STEP);

```

```
    //移动出界后，需要删除敌人飞机
            if ((*ite)->m_y >= BACK_HEIGHT) {  //判断是否出界
                //删除敌人飞机
                delete (*ite);
                (*ite) = nullptr;

                //删除节点
                ite = m_lstFoe.erase(ite);   //自带了++效果
                continue;
            }
        }
        ++ite;
    }
}

//CGunList
#include"../Config/Config.h"
#include"CGunList.h"


CGunList::CGunList() {}
CGunList::~CGunList() {
    list<CGunner*>::iterator ite = m_lstGun.begin();
    while (ite != m_lstGun.end()) {
        if (*ite) {  //如果炮弹的指针不为空
            delete (*ite);
            (*ite) = nullptr;
        }
        ++ite;
    }
}
void CGunList::ShowAllGun() {
    for (CGunner* p : m_lstGun) {  //遍历链表显示每个炮弹
        if (p) p->ShowGun();
    }
}
void CGunList::MoveAllGun() {
    //8. del yuesl, 遍历炮弹链表移动每个炮弹，注意出界要删除
    list<CGunner*>::iterator ite = m_lstGun.begin();
    while (ite != m_lstGun.end())
    {
        if (*ite)//炮弹不为空的情况下 移动每一个炮弹
        {
            (*ite)->MoveGun();
            //判断是否出界，出界则删除
            if ((*ite)->m_y <= -GUNNER_HEIGHT)
            {
                delete (*ite);  //先删除炮弹本身
                (*ite) = nullptr;
                ite = m_lstGun.erase(ite); //再删除节点，必须用迭代器接，返回删除的下一个，自带++的效果
                continue;
            }
        }
        ++ite;
    }
}

```

```cpp
//CGunner
#include"CGunner.h"
#include"../Config/Config.h"
#include"../FoeList/CFoeBig.h"
#include"../FoeList/CFoeMid.h"
#include"../FoeList/CFoeSma.h"
#include<typeinfo>
using namespace std;

CGunner::CGunner():m_x(0),m_y(0){}
CGunner::~CGunner() {}

void CGunner::InitGun(int x, int y) {
    m_x = x;
    m_y = y;
    ::loadimage(&m_img, L"./res/gunner.jpg");

}
void CGunner::ShowGun()
{
    //6. del yuesl,利用原图和屏蔽图共同显示，并去除白边
    //屏蔽图 位或
    ::putimage(m_x, m_y/*显示到窗口的位置*/, GUNNER_WIDTH, GUNNER_HEIGHT/*要显示图片的大小*/,
        &m_img/*要显示那个图片*/, GUNNER_WIDTH, 0/* 从图片的哪个位置开始显示  */, SRCPAINT/*传输方式*/);
    //原图位于
    ::putimage(m_x, m_y/*显示到窗口的位置*/, GUNNER_WIDTH, GUNNER_HEIGHT/*要显示图片的大小*/,
        &m_img/*要显示那个图片*/, 0, 0/* 从图片的哪个位置开始显示  */, SRCAND/*传输方式*/);
}
void CGunner::MoveGun() {
    m_y -= GUNNER_MOVE_STEP;
}

bool CGunner::isHitFoe(CFoe*  pfoe) {
    if (!pfoe)  return false;
    //7. del yuesl, 获取炮弹中心点，将中心点判断是否撞击到敌人飞机，注意三种敌人飞机判断的范围不同
    int x = m_x + GUNNER_WIDTH / 2;
    int y = m_y + GUNNER_HEIGHT / 2;

    if (typeid(*pfoe) == typeid(CFoeBig)) {

        if (pfoe->m_x <= x && x <= pfoe->m_x + FOEBIG_WIDTH &&
            pfoe->m_y <= y && y <= pfoe->m_y + FOEBIG_HEIGHT
            ) {
            return true;
        }
    }
    else if (typeid(*pfoe) == typeid(CFoeMid)) {

        if (pfoe->m_x <= x && x <= pfoe->m_x + FOEMID_WIDTH &&
            pfoe->m_y <= y && y <= pfoe->m_y + FOEMID_HEIGHT
            ) {
            return true;
        }
```

```
    }
    else if (typeid(*pfoe) == typeid(CFoeSma)) {

        if (pfoe->m_x <= x && x <= pfoe->m_x + FOESMA_WIDTH &&
            pfoe->m_y <= y && y <= pfoe->m_y + FOESMA_HEIGHT
            ) {
            return true;
        }
    }
    return false;
}

//CPlane
#include"CPlane.h"
#include"../Config/Config.h"

CPlane::CPlane():m_x(0),m_y(0){}
CPlane::~CPlane() {}

void CPlane::InitPlane() {
    ::loadimage(&m_img,L"./res/player.jpg");
    ::loadimage(&m_maskImg,L"./res/player_mask.jpg");

    //9. del yuesl,初始化玩家飞机的坐标，让其在窗口下方居中位置
    m_x = (BACK_WIDTH - PLANE_WIDTH) / 2;
    m_y = BACK_HEIGHT - PLANE_HEIGHT;
}
void CPlane::ShowPlane() {
    ::putimage(m_x, m_y, &m_maskImg,SRCPAINT);   //先  屏蔽图 位或
    ::putimage(m_x, m_y, &m_img, SRCAND);    //后  原图 位与
}
void CPlane::MovePlane(int direct) {
    if (direct == VK_UP) {
        //if ((m_y - PLANE_MOVE_STEP) >= 0) {
        //  m_y -= PLANE_MOVE_STEP;
        //}
        //else {
        //  m_y = 0;
        //}
        (m_y - PLANE_MOVE_STEP) >= 0? m_y -= PLANE_MOVE_STEP: m_y = 0;

    }
    else if (direct == VK_DOWN) {
        (m_y + PLANE_MOVE_STEP) <= (BACK_HEIGHT - PLANE_HEIGHT) ?
            m_y += PLANE_MOVE_STEP : m_y = (BACK_HEIGHT - PLANE_HEIGHT);
    }
    else if (direct == VK_LEFT) {
        (m_x - PLANE_MOVE_STEP) >= 0 ? m_x -= PLANE_MOVE_STEP : m_x = 0;
    }
    else if (direct == VK_RIGHT) {
        (m_x + PLANE_MOVE_STEP) <= (BACK_WIDTH - PLANE_WIDTH) ?
            m_x += PLANE_MOVE_STEP : m_x = (BACK_WIDTH - PLANE_WIDTH);
    }

}

CGunner* CPlane::SendGun() {
    CGunner* pGun = new CGunner;  //创建炮弹 和 初始化
```

```
    pGun->InitGun(m_x + (PLANE_WIDTH - GUNNER_WIDTH) / 2, m_y - GUNNER_HEIGHT);
    return pGun;
}


CPlane& CPlane::GetPlane() {
    //10. del yuesl  ，单例模式创建玩家飞机
    static CPlane plane;
    return plane;  // 返回该实例的引用

}

//CPlaneApp
#include"CPlaneApp.h"
#include"../Config/Config.h"
//#include"../FoeList/CFoeBig.h"
//#include"../FoeList/CFoeMid.h"
//#include"../FoeList/CFoeSma.h"


WND_PARAM(600+16,800+39,500,50,L"飞机大战")
CREATE_OBJECT(CPlaneApp)


CPlaneApp::CPlaneApp() :m_score(0), m_plane(CPlane::GetPlane()) {}
CPlaneApp::~CPlaneApp() {}

void CPlaneApp::On_Init() {
    SetTimer();

    ::loadimage(&m_img, L"./res/card.png", 120, 50);   //加载图片，将图片加载到指定大小


    m_back.InitBack();
    m_plane.InitPlane();
}
void CPlaneApp::On_Paint() {

    m_back.ShowBack();
    m_plane.ShowPlane();

    m_lstGun.ShowAllGun();

    m_lstFoe.ShowAllFoe();

    ShowScore();
}
void CPlaneApp::On_Close() {}

void CPlaneApp::InitMsgMap(){
//2. del yuesl ,初始化消息映射，绑定消息和处理函数（你认为需要哪些就绑定哪些）
    INIT_MSGMAP(WM_TIMER, EX_WINDOW, CPlaneApp) //添加这一条的作用是用来绑定消息和处理函数
    INIT_MSGMAP(WM_KEYDOWN, EX_KEY, CPlaneApp)
}

```

```
328
329 void CPlaneApp::SetTimer() {
330     //设定定时器  WM_TIMER
331     ::SetTimer(m_hwnd, BACK_MOVE_ID, BACK_MOVE_INTERVAL,nullptr);
332     //定时检测方向键是否按下
333     ::SetTimer(m_hwnd, CHECK_KEYDOWN_ID, CHECK_KEYDOWN_INTERVAL,nullptr);
334
335     //定时创建炮弹
336     ::SetTimer(m_hwnd, CREATE_GUN_ID, CREATE_GUN_INTERVAL, nullptr);
337
338     //炮弹定时移动
339     ::SetTimer(m_hwnd, GUN_MOVE_ID, GUN_MOVE_INTERVAL, nullptr);
340
341     //定时创建敌人飞机
342     ::SetTimer(m_hwnd, FOE_CREATE_ID, FOE_CREATE_INTERVAL, nullptr);
343
344     //敌人飞机定时移动
345     ::SetTimer(m_hwnd, FOE_MOVE_ID, FOE_MOVE_INTERVAL, nullptr);
346
347     //定时检测是否碰撞的定时器
348     ::SetTimer(m_hwnd, CHECK_HIT_ID, CHECK_HIT_INTERVAL, nullptr);
349
350     //定时切换爆炸图片
351     ::SetTimer(m_hwnd, CHANGE_PIC_ID, CHANGE_PIC_INTERVAL, nullptr);
352
353
354 }
355 void CPlaneApp::KillTimer() {
356     ::KillTimer(m_hwnd, BACK_MOVE_ID);
357     ::KillTimer(m_hwnd, CHECK_KEYDOWN_ID);
358     //定时创建炮弹
359     ::KillTimer(m_hwnd, CREATE_GUN_ID);
360     //炮弹定时移动
361     ::KillTimer(m_hwnd, GUN_MOVE_ID);
362     //定时创建敌人飞机
363     ::KillTimer(m_hwnd, FOE_CREATE_ID);
364     //敌人飞机定时移动
365     ::KillTimer(m_hwnd, FOE_MOVE_ID);
366     //定时检测是否碰撞的定时器
367     ::KillTimer(m_hwnd, CHECK_HIT_ID);
368     //定时切换爆炸图片
369     ::KillTimer(m_hwnd, CHANGE_PIC_ID);
370 }
371 void CPlaneApp::ShowScore() {
372     ::putimage(0, 0, &m_img);  //先显示图片
373
374     wstring  wstr = L"分数为：";
375     wchar_t sc[5] = {0};
376     _itow_s(m_score, sc, 10);//itoa
377     wstr += sc;
378
379     RECT rect = { 0,0,120, 50 };
380     settextcolor(RGB(10,20,200));  //设置文字的颜色
381     drawtext(wstr.c_str(),&rect, DT_VCENTER| DT_SINGLELINE| DT_CENTER);
382 }
383
384 //用于处理所有的定时器的处理函数
385 void CPlaneApp::On_WM_TIMER(WPARAM w, LPARAM l) {
```

```
386
387     switch (w)
388     {
389     case BACK_MOVE_ID:
390     {  //背景移动
391         m_back.MoveBack();
392     }
393     break;
394     case CHECK_KEYDOWN_ID:
395     {
396         if (::GetAsyncKeyState(VK_UP)) {   //检测方向键up  是否按下，如果按下则返回非0值
397             m_plane.MovePlane(VK_UP);
398         }
399         if (::GetAsyncKeyState(VK_DOWN)) {
400             m_plane.MovePlane(VK_DOWN);
401         }
402         if (::GetAsyncKeyState(VK_LEFT)) {
403             m_plane.MovePlane(VK_LEFT);
404         }
405         if (::GetAsyncKeyState(VK_RIGHT)) {
406             m_plane.MovePlane(VK_RIGHT);
407         }
408     }
409     break;
410     case CREATE_GUN_ID:
411     {
412         //3. del yuesl  定时发射炮弹，并添加到链表中
413         CGunner* pGun = m_plane.SendGun();  //发射炮弹
414         //装进链表
415         m_lstGun.m_lstGun.push_back(pGun);
416     }
417     break;
418     case GUN_MOVE_ID:
419     {
420         m_lstGun.MoveAllGun();
421     }
422     break;
423     case FOE_CREATE_ID:
424     {
425         int num = CFoe::rd() % 101;
426
427         CFoe* pFoe = nullptr;
428
429         //根据概率  创建
430         if (num < 50) {   //小
431             //pFoe = new  CFoeSma;
432             pFoe = m_fac.CreateFoe("foesma");   //使用简单工厂模式
433         }
434         else if (50 <= num && num < 80) {
435             pFoe = m_fac.CreateFoe("foemid");
436         }
437         else {
438             pFoe = m_fac.CreateFoe("foebig");
439         }
440
441         pFoe->InitFoe();   //统一初始化
442         m_lstFoe.m_lstFoe.push_back(pFoe);  //添加链表
```

```
443     }
444     break;
445     case FOE_MOVE_ID:
446     {
447         m_lstFoe.MoveAllFoe();
448     }
449     break;
450     case CHECK_HIT_ID:
451     {
452         //4. del yuesl  ,定时检测是否碰撞，
453 
454         //遍历敌人飞机,判断每个敌人飞机是否撞击到了玩家飞机，撞击了，则游戏结束....
455 
456 
457         //遍历炮弹链表，  判断每个敌人飞机和每个炮弹是否撞击了,撞击了，则...
458 
459 
460         list<CFoe*>::iterator iteFoe = m_lstFoe.m_lstFoe.begin();  //遍历正常的敌人飞机
461         while (iteFoe != m_lstFoe.m_lstFoe.end()) {
462             if (*iteFoe) {
463                 list<CGunner*>::iterator iteGun = m_lstGun.m_lstGun.begin(); // 遍历炮弹链表
464                 while (iteGun != m_lstGun.m_lstGun.end()) {
465                     if (*iteGun && (*iteGun)->isHitFoe(*iteFoe)) { // 炮弹与敌机碰撞
466                         // 删除炮弹对象
467                         delete (*iteGun);
468                         (*iteGun) = nullptr;
469                         // 从炮弹链表中删除节点
470                         iteGun = m_lstGun.m_lstGun.erase(iteGun);
471 
472                         // 敌人飞机掉血
473                         (*iteFoe)->m_blood--;
474 
475                         // 判断敌机血量是否为0
476                         if ((*iteFoe)->m_blood <= 0) {
477                             // 敌机移到爆炸链表（假设存在爆炸链表管理）
478                             m_lstFoe.m_lstBoomFoe.push_back(*iteFoe);
479                             // 从正常敌机链表中删除节点
480                             iteFoe = m_lstFoe.m_lstFoe.erase(iteFoe);
481                             m_score++; // 加分
482                             break; // 跳出炮弹循环，避免重复检测
483                         }
484                         continue;
485                     }
486                     ++iteGun;
487                 }
488                 // ----------------------------------------------------
489 
490                 //判断是否撞击了玩家飞机
491                 if (iteFoe != m_lstFoe.m_lstFoe.end() && (*iteFoe)->IsHitPlane(m_plane.m_x, m_plane.m_y)) {
492 
493                     //游戏结束 ，停止游戏运行，
494                     KillTimer();
495 
496                     //阻塞的方法
```

```
                    ::MessageBox(m_hwnd, L"GameOver~", L"提示", MB_OK);

                    //使程序退出
                    ::PostMessage(m_hwnd, WM_CLOSE, 0, 0);  //投递一个消息
                    return;
                }
            }
            // 只有当iteFoe未被erase修改时，才手动++
            if (iteFoe != m_lstFoe.m_lstFoe.end())
                ++iteFoe;

        }
    }
    break;
    case CHANGE_PIC_ID:
    {
        //5. del yuesl,定时切换爆炸效果，
        //遍历爆炸链表，将每个敌人飞机用于切换的显示的变量进行减小，直到爆炸效果一个不漏的显示完毕，最后删除敌人飞机

        list<CFoe*>::iterator ite = m_lstFoe.m_lstBoomFoe.begin();
        while (ite != m_lstFoe.m_lstBoomFoe.end()) {
            if (*ite) {
                if (--(*ite)->m_showId < 0) {
                    //是否显示完，显示完后，要回收
                    delete (*ite);
                    (*ite) = nullptr;

                    //回收节点
                    ite = m_lstFoe.m_lstBoomFoe.erase(ite);
                    continue;
                }
            }
            ++ite;
        }
    }
    break;
    }

}


void CPlaneApp::On_WM_KEYDOWN(int key) {
    //m_plane.MovePlane(key);
}
```