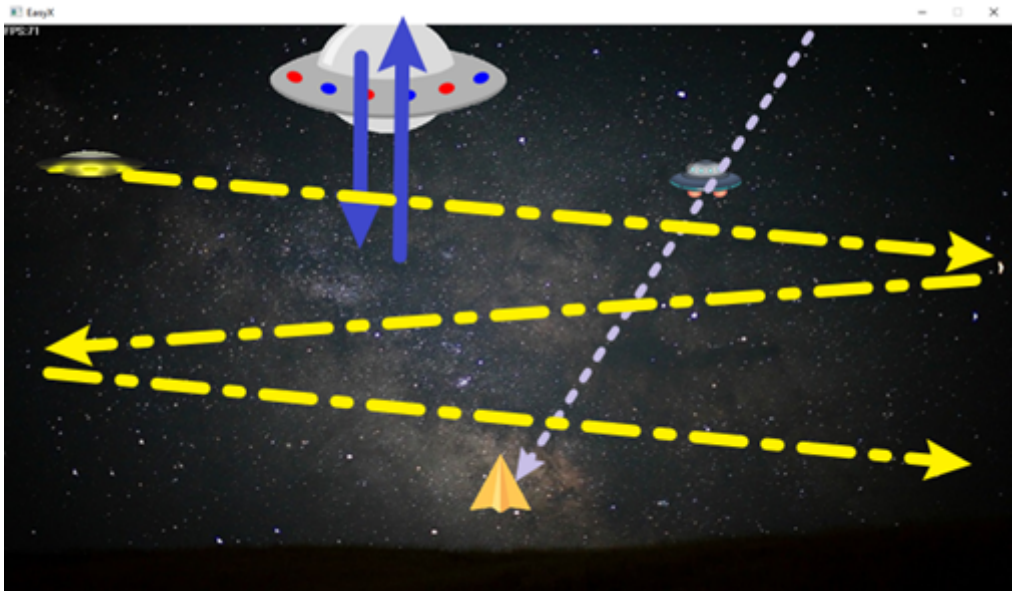
游戏中有3种UFO，每种UFO都有不同的移动方式，大型UFO缓慢向前，一段时间后再退回去；中型UFO向前蛇形走位，遇边折返；小型UFO快速撞向玩家飞机。



增加属性

为结构体添加新的属性，图片的宽高，飞行器速度，是否重生。

```
//定义飞机的结构体
struct aircraft
{
    int x ;
    int y;
    int width;
    int height;
    int speed;
    int new_born_flg;
};
```

修改初始化代码

```
//飞机数据初始化
void dataInit()
{
    plane = { 150,150,80,80,10,1 };
    ufoa  = { 0,0,300,150,2,1 };
    ufob  = { 350,0,150,50,4,1 };
    ufoc  = { 450,200,100,60,10,1 };

}
```

UFOA的移动

编写一个函数，ufoaMove，里边有个局部静态变量dir，表示前进方向。

重生之后水平位置随机生成。

```
//大UFO缓慢向前移动，到达一定的位置回去
void ufoaMove()
{
    static int dir = 1;//前进
    if (1 == ufoa.new_born_flg)//新出生的ufoa ，重置x,y的坐标
    {
        ufoa.new_born_flg = 0;
        ufoa.x = rand() % (WIDTH - ufoa.width);
        ufoa.y = -50 ;
    }
    if(ufoa.y > 200)//纵坐标大于300，改为后退
    {
        dir = 0; //后退

    }
    else if(ufoa.y < -150)
    {
        dir = 1;
        ufoa.new_born_flg = 1;
    }

    if(dir == 1) //前进
        ufoa.y += ufoa.speed;
    else  //后退
        ufoa.y -= ufoa.speed;
}
```

主函数需要设置个随机数的种子srand(time(NULL));

```
int _tmain(int argc, _TCHAR* argv[])
{
    dataInit();//初始化所有飞机，设置飞机坐标
    initgraph(WIDTH, HEIGHT);// 创建绘图窗口
    loadRes();
    clock_t start_time;
    BeginBatchDraw();
    srand(time(NULL));
    while(1)
    {
        ufoaMove();
        start_time = clock();
        getInput(); //获取输入
        dealInput();//调整飞机位置
        showAircraft();//显示飞机
        putimage(0, 0, &temp_img);
        ctrlFps(start_time);//控制Fps在60左右
        FlushBatchDraw();
    }
    EndBatchDraw();
    _getch();
    closegraph();

    return 0;
}
```

UFOB的移动

中型UFO从顶部向右下方倾斜移动，水平方向上遇到边框反弹，向下移动到末端直到中型UFO消失在窗口中，在顶部重新生成。

```
//UFOB左右快速移动，慢速向下移动
void ufobMove()
{
    static int step = ufob.speed;//spep表示方向与速度
    if (1 == ufob.new_born_flg)//新出生的ufob ，重置x,y的坐标
    {
        ufob.new_born_flg = 0;
        ufob.x = rand() % (WIDTH - ufob.width);
        ufob.y = -ufob.height;
    }
    //水平撞墙返回
    if ((ufob.x <= 0) || (ufob.x + ufob.width >= WIDTH))
        step = -step;
    ufob.x += step;
    ufob.y++;
    //超出下边界，重生，y坐标重置，x坐标随机
    if (ufob.y >= HEIGHT)
        ufob.new_born_flg = 1;
}
```

第4行，step表示每次调用此函数时，ufob在水平方向上移动的坐标。Ufob在竖直方向上每次在移动1像素，水平方向上移动多个像素，所以其水平运动速度更快。第13行，判断ufob的横坐标，当ufob的左侧小于0，或者右侧大于屏幕宽度后，速度取反。第18行，当ufob超越下边界，使得ufob重新生成。

UFOC的移动

小型UFO将撞向飞机，它的移动轨迹是一条直线，直线的一个点是初始化时，小型UFO随机生成的坐标，另一个点是飞机的坐标。让小型UFO移动的函数，主要逻辑就是根据2个点的坐标画一条直线。

```
//ufoc撞向飞机
void ufocMove()
{
    static float dist_x = 0, dist_y = 0;//ufoc出生时，记录与飞机的横竖距离
    static float tmp_x = 0, tmp_y = 0; //储存x,y坐标的临时变量，浮点型方便计算
    static float vx = 0, vy = 0;
    float step = 1000 / ufoc.speed;       //调整UFO速度
    if (1 == ufoc.new_born_flg)
    {
        ufoc.new_born_flg = 0;
        tmp_x = rand() % (WIDTH - ufoc.width);
        tmp_y = -ufoc.height;
        dist_x = plane.x - tmp_x;
        dist_y = plane.y - tmp_y;
        vx = dist_x / step;
        vy = dist_y / step;
    }
    tmp_x += vx;
    tmp_y += vy;
    ufoc.x = (int)(tmp_x + 0.5);
    ufoc.y = (int)(tmp_y + 0.5);
    //边界判断，可以超出画面，但不超出太多
```

```
    if (ufoc.x < -ufoc.width)
        ufoc.new_born_flg = 1;
    else if (ufoc.x > WIDTH+ufoc.width)
        ufoc.new_born_flg = 1;
    //超出下边界，重生，y坐标重置，x坐标随机
    if (ufoc.y >= HEIGHT)
        ufoc.new_born_flg = 1;
}
```

至此已经实现了3种敌机的移动

完整代码

```
#include "stdafx.h"

#define HEIGHT 720  // 游戏画面尺寸
#define WIDTH 1280

//定义飞机的结构体
struct aircraft
{
    int x ;
    int y;
    int width;
    int height;
    int speed;
    int new_born_flg;
};
aircraft plane, ufoa, ufob, ufoc;

IMAGE img_bk, img_plane,temp_img,
    img_ufoa,img_ufob,img_ufoc,
    img_plane_bullet,img_ufoa_bullet,img_ufob_bullet;


//按键输入的枚举列表
enum GAMEINPUT
{
    NOINPUT = 0X0,
    UPINPUT = 0X1,
    DOWNINPUT = 0X2,
    LEFTINPUT = 0X4,
    RIGHTINPUT = 0X8,
    FIREINPUT = 0X10
};
int input = NOINPUT;//判断输入变量
int speed = 10;

void dataInit();
void loadRes();
void drawAlpha(IMAGE* dstimg, int x, int y, IMAGE* srcimg);
void showAircraft();
void getInput();
void dealInput();
void ctrlFps(int start_time);
void ufoaMove();
```

```
void ufobMove();
void ufocMove();

int _tmain(int argc, _TCHAR* argv[])
{
    dataInit();//初始化所有飞机，设置飞机坐标
    initgraph(WIDTH, HEIGHT);// 创建绘图窗口
    loadRes();
    clock_t start_time;
    BeginBatchDraw();
    srand(time(NULL));
    while(1)
    {
        ufoaMove();
        ufobMove();
        ufocMove();
        start_time = clock();
        getInput(); //获取输入
        dealInput();//调整飞机位置
        showAircraft();//显示飞机
        putimage(0, 0, &temp_img);
        ctrlFps(start_time);//控制Fps在60左右
        FlushBatchDraw();
    }
    EndBatchDraw();
    _getch();
    closegraph();

    return 0;
}

//飞机数据初始化
void dataInit()
{
    plane = { 150,150,80,80,10,1 };
    ufoa  = { 0,0,300,150,2,1 };
    ufob  = { 350,0,150,50,4,1 };
    ufoc  = { 450,200,100,60,10,1 };

}

//以相对路径载入所有素材
void loadRes()
{
    loadimage(&img_bk, _T("res\\background.png"));
    loadimage(&temp_img, _T("res\\background.png"));
    loadimage(&img_plane, _T("res\\plane.png"));
    loadimage(&img_ufoa, _T("res\\ufoa.png"));
    loadimage(&img_ufob, _T("res\\ufob.png"));
    loadimage(&img_ufoc, _T("res\\ufoc.png"));
    loadimage(&img_plane_bullet, _T("res\\plane_bullet.png"));
    loadimage(&img_ufoa_bullet, _T("res\\ufoa_bullet.png"));
    loadimage(&img_ufob_bullet, _T("res\\ufob_bullet.png"));
}

// 根据透明度绘图
void drawAlpha(IMAGE *dstimg, int x, int y, IMAGE *srcimg)
{
```

```
    // 变量初始化
    DWORD *dst = GetImageBuffer(dstimg);
    DWORD *src = GetImageBuffer(srcimg);
    int src_width = srcimg->getwidth();
    int src_height = srcimg->getheight();
    int dst_width = (dstimg == NULL ? getwidth() : dstimg->getwidth());
    int dst_height = (dstimg == NULL ? getheight() : dstimg->getheight());

    // 计算贴图的实际长宽
    int iwidth = (x + src_width > dst_width) ? dst_width - x : src_width;
// 处理超出右边界
    int iheight = (y + src_height > dst_height) ? dst_height - y : src_height;
// 处理超出下边界
    if (x < 0) { src += -x;             iwidth -= -x;   x = 0; }
// 处理超出左边界
    if (y < 0) { src += src_width * -y; iheight -= -y;  y = 0; }
// 处理超出上边界

    // 修正贴图起始位置
    dst += dst_width * y + x;
    // 实现透明贴图
    for (int iy = 0; iy < iheight; ++iy)
    {
        for (int i = 0; i < iwidth; ++i)
        {
            int sa = ((src[i] & 0xff000000) >> 24);//获取阿尔法值
            if (sa != 0)//假如是完全透明就不处理
            if (sa == 255)//假如完全不透明则直接拷贝
            dst[i] = src[i];
            else//真正需要阿尔法混合计算的图像边界才进行混合
            dst[i] = ((((src[i] & 0xff0000) >> 16) + ((dst[i] & 0xff0000) >> 16)
* (255 - sa) / 255) << 16) |((((src[i] & 0xff00) >> 8) + ((dst[i] & 0xff00) >>
8) * (255 - sa) / 255) << 8) | ((src[i] & 0xff) + (dst[i] & 0xff) * (255 - sa) /
255);
        }
        dst += dst_width;
        src += src_width;
    }
}
//绘制所有的飞机
void showAircraft()
{

    drawAlpha(&temp_img,0, 0, &img_bk);
    drawAlpha(&temp_img,plane.x, plane.y, &img_plane);
    drawAlpha(&temp_img,ufoa.x, ufoa.y, &img_ufoa);
    drawAlpha(&temp_img,ufob.x, ufob.y, &img_ufob);
    drawAlpha(&temp_img,ufoc.x, ufoc.y, &img_ufoc);
    //暂时也绘制子弹
    drawAlpha(&temp_img,0, 300, &img_plane_bullet);
    drawAlpha(&temp_img,50, 300, &img_ufoa_bullet);
    drawAlpha(&temp_img,100, 300, &img_ufob_bullet);
}

//同时获取多个输入，操作飞机
void getInput()
{
    if (GetAsyncKeyState('W') & 0x8000)
```

```
    {
        input |= UPINPUT;
    }
    if (GetAsyncKeyState('S') & 0x8000)
    {
        input |= DOWNINPUT;
    }
    if (GetAsyncKeyState('A') & 0x8000)
    {
        input |= LEFTINPUT;
    }
    if (GetAsyncKeyState('D') & 0x8000)
    {
        input |= RIGHTINPUT;
    }
    if (GetAsyncKeyState('K') & 0x8000)
    {
        input |= FIREINPUT;
    }
}

//同时处理多个输入，调整飞机的位置
void dealInput()
{
    if ((input & UPINPUT) && (plane.y >= 0))
    {
        plane.y -= speed;
    }
    if ((input & DOWNINPUT) && (plane.y <= HEIGHT - 120))
    {
        plane.y += speed;
    }
    if ((input & LEFTINPUT) && (plane.x >= 0))
    {
        plane.x -= speed;
    }
    if ((input & RIGHTINPUT) && (plane.x <= WIDTH - 120))
    {
        plane.x += speed;
    }
    input = NOINPUT;
}

//根据从开始到现在的时间，设置休眠的时间
void ctrlFps(int start_time)
{
    clock_t running_time = clock() - start_time;
    if((13 - running_time) >= 0)//防止睡眠函数使用负数
      Sleep(13 - running_time);//动态睡眠
    TCHAR time_text[50];
    int FPS = 1000 / (clock() - start_time);
    _stprintf_s(time_text, _T("FPS:%d"), FPS);
    settextstyle(60, 0, _T("黑体")); //为了演示，显示fps字体大小不宜太大
    outtextxy(0, 0, time_text);
}

//UFOA缓慢向前移动，到达一定的位置回去
void ufoaMove()
```

```
{
    static int dir = 1;//前进
    if (1 == ufoa.new_born_flg)//新出生的ufoa ， 重置x,y的坐标
    {
        ufoa.new_born_flg = 0;
        ufoa.x = rand() % (WIDTH - ufoa.width);
        ufoa.y = -50 ;
    }
    if(ufoa.y > 200)//纵坐标大于300，改为后退
    {
        dir = 0; //后退

    }
    else if(ufoa.y < -150)
    {
        dir = 1;
        ufoa.new_born_flg = 1;
    }

    if(dir == 1) //前进
        ufoa.y += ufoa.speed;
    else  //后退
        ufoa.y -= ufoa.speed;
}


//UFOB左右快速移动，慢速向下移动
void ufobMove()
{
    static int step = ufob.speed;//spep表示方向与速度
    if (1 == ufob.new_born_flg)//新出生的ufob ， 重置x,y的坐标
    {
        ufob.new_born_flg = 0;
        ufob.x = rand() % (WIDTH - ufob.width);
        ufob.y = -ufob.height;
    }
    //水平撞墙返回
    if ((ufob.x <= 0) || (ufob.x + ufob.width >= WIDTH))
        step = -step;
    ufob.x += step;
    ufob.y++;
    //超出下边界，重生，y坐标重置，x坐标随机
    if (ufob.y >= HEIGHT)
        ufob.new_born_flg = 1;
}

//ufoc撞向飞机
void ufocMove()
{
    static float dist_x = 0, dist_y = 0;//ufoc出生时，记录与飞机的横竖距离
    static float tmp_x = 0, tmp_y = 0; //储存x,y坐标的临时变量，浮点型方便计算
    static float vx = 0, vy = 0;
    float step = 1000 / ufoc.speed;       //调整UFO速度
    if (1 == ufoc.new_born_flg)
    {
        ufoc.new_born_flg = 0;
        tmp_x = rand() % (WIDTH - ufoc.width);
        tmp_y = -ufoc.height;
```

```
        dist_x = plane.x - tmp_x;
        dist_y = plane.y - tmp_y;
        vx = dist_x / step;
        vy = dist_y / step;
    }
    tmp_x += vx;
    tmp_y += vy;
    ufoc.x = (int)(tmp_x + 0.5);
    ufoc.y = (int)(tmp_y + 0.5);
    //边界判断，可以超出画面，但不超出太多
    if (ufoc.x < -ufoc.width)
        ufoc.new_born_flg = 1;
    else if (ufoc.x > WIDTH+ufoc.width)
        ufoc.new_born_flg = 1;
    //超出下边界，重生，y坐标重置，x坐标随机
    if (ufoc.y >= HEIGHT)
        ufoc.new_born_flg = 1;
}
```