C 语言项目报告

项目名称：校园 CS

小组成员：吴志涛（物理方向）

郭运佳（化学方向）

填写日期：2020 年 9 月 9 日

一、摘要

相信很多人都玩过 4399 上的坦克大战，虽然简单但和同学朋友一起玩的时候有无限乐趣，里面最有意思的机制便是子弹反弹和子弹杀伤自己的可能，我们做的这款游戏与这款游戏游戏机制相似，但我们为其加入了背景故事，同时利用 Easyx 使画面更加精美，使游戏更加丰富，更具有可玩性。于此同时，我们也注意课堂上学习的 C 语言知识的应用，如链表、迭代等，争取在 C 语言下完成整个项目。

二、问题描述

游戏背景说明：2050 年，学校正在举行校园 CS 比赛，参赛的同学分为两队，一对一单独对战，两个人将被同时传送到一个异空间进行比赛，每次 bb 弹击中任何一个人的时候另外一个人的比分将加一，同时两个人将继续被传送到下一个异空间进行比赛，达到一定比分后两人将决出胜负。异空间将自动检验每个人发射的子弹数量，每个人在空间中存在的子弹最多只有五颗，只有当子弹出界时那个人物才能继续发射子弹。需要说明的是，子弹会在墙壁上弹射，同时弹射的子弹如果发射到自己身上也会杀伤到自己，同时上一局未出界的子弹会留到下一局，玩家应该谨慎发射子弹，避免子弹射到自己或者陷入无法发射子弹的困境。

游戏特点：

（1）随机地图

（2）子弹会在墙壁上弹射并有杀伤到自己的可能

（3）全向移动方式

（4）上一局的子弹会留到下一局

待解决的问题：

1.透明化：

（1）三元光栅操作码（位操作和掩码图）

（2）支持 Alpha 信息旋转的函数（基于直接操作显示缓冲区）

2. 人物的动画效果：

（1）设置各种状态参量

（2）批量绘图函数解决频繁刷新闪烁的问题

3. 子弹在墙壁上的反弹和地形对人物的限制：

（1）用链表记录子弹信息，结构体记录人物信息

（2）首先遍历保存随机地图的数组，再根据人物或子弹与墙壁的相对位置进行反弹或限制坐标

4. 如何保证随机生成的地图的可玩性

三、组内分工

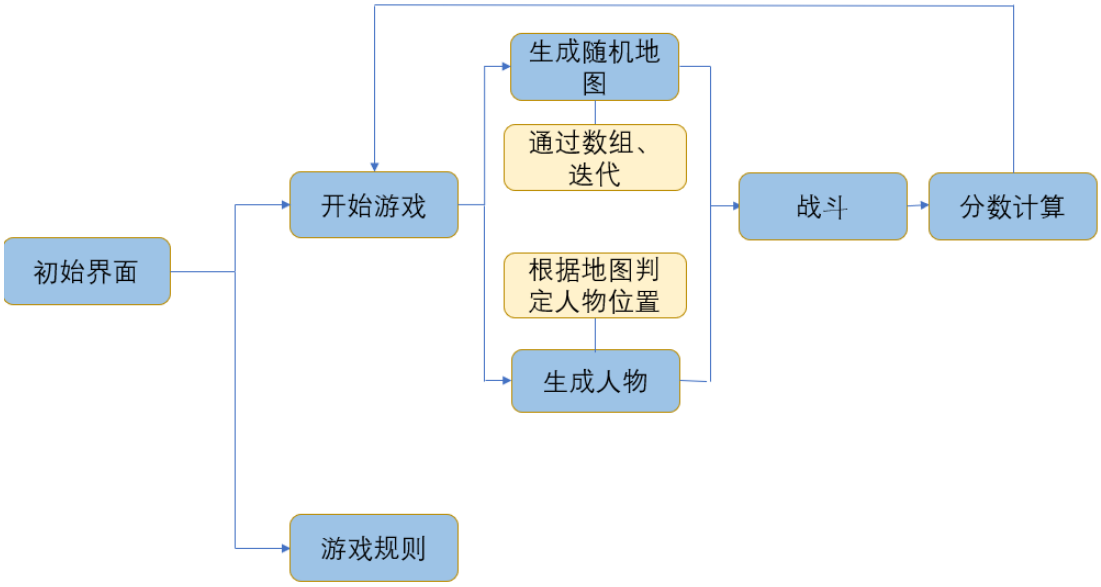
吴志涛（物理）：角色的移动，音乐，射击，子弹和人物的判定，各个模块的整合（60%）

郭运佳（化学）：初始界面，随机地图的生成，角色的绘制（40%）

四、分析

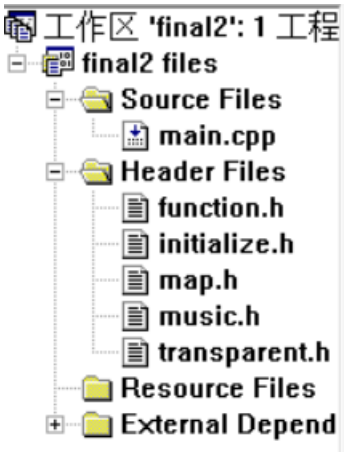
我们的游戏每局会生成一个随机地图，这个随机地图只有百分之八十及以上的路能走通时才会生成，玩家会生成在能走通的路的两侧。玩家移动采取的是全向移动方式，以玩家一为例，使用 A 和 D 两个键来改变前进的方向，使用 W 和 S 前进或者后退，这种移动方式不复杂有一定的操作难度，在双人对战时会增添一定的趣味性。玩家发射出去的子弹会在墙上弹射同时反弹的子弹也有杀伤到自己的可能。而且上一局生成的子弹也会留到下一局，和随机地图

一起造成一定的随机性和可玩性，也就是说上一局发射的子弹下一局有可能直接把自己杀死。每当有一个玩家被击中时则自动开始下一局，一个玩家被击中时则另一个玩家增加一分，最后比较累计得分。



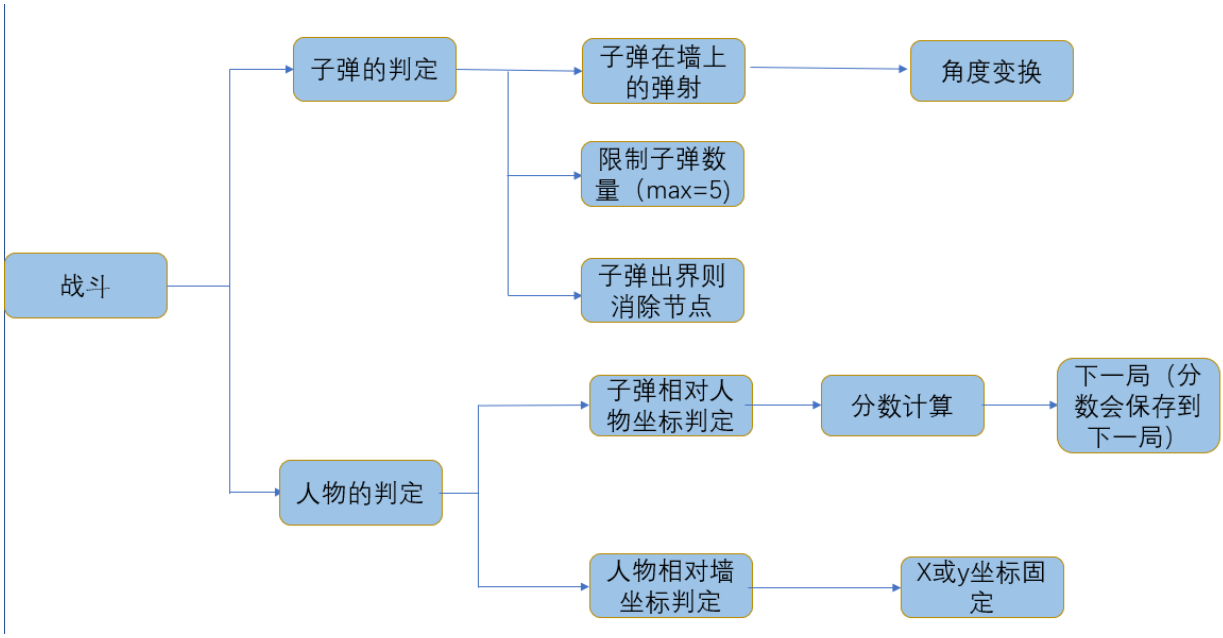
五、设计

项目的分块:

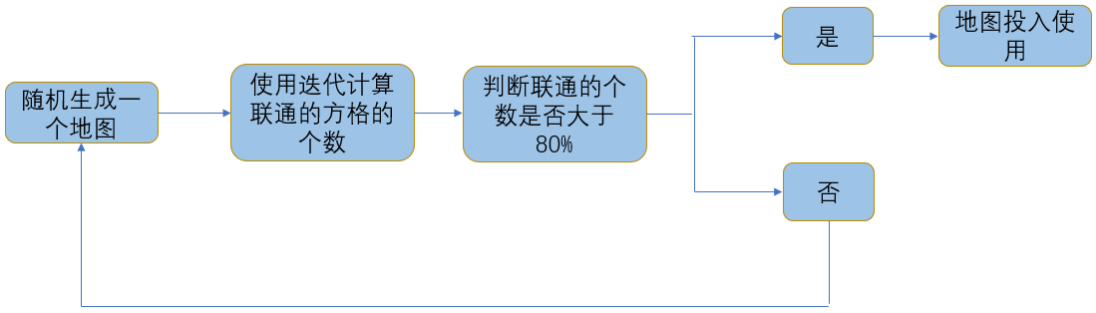


我们的项目主要分为战斗和地图两部分，我们通过头文件的方式将一些函数单独封装了起来，比如播放音乐的函数（music.h）和透明化图片的函数(transparent.h)。

战斗系统中的主要模块及流程图:



随机生成地图的流程图：



六、实施

一、软件的选择：

Easyx+VC6++

最开始我们在 b 站上寻找做 C 语言游戏的课，最后发现了河海大学童晶老师的课，讲了 Easyx 和 VC6++的使用，同时我们看了那个老师的往届学生做的游戏的演示，觉得效果也很好，于是我们选择了比较容易操作的 Easyx 和 VC6++，但 Easyx 和 VC6++也有很多问题，比如 Easyx 有图片透明化的问题，VC6++有调试不方便和代码显示颜色都一样的问题。(之后的同学如果遇到 Easyx 上图片透明化的问题可以参考我们的两个解决方案，VC6++建议用 VS 替代，VS 的 debug 更方便一些，而且代码显示也便于编写。)

二、运用 Easyx 时碰到的问题：

1.透明化

因为 easyx 不支持 png 格式的透明图片的透明背景显示，我们开始用了三元光栅操作通过位操作使原图和掩码图相互配合实现了背景的透明化，之后发现如果还通过三元光栅操作的方法当人物的枪旋转时 Easyx 自带的 rotateimage()函数会使原图失真，同时在旋转过后的空

白地方会填充背景颜色导致覆盖地图或者另一个人物，于是我们通过查阅资料实现了基于直接操作显示缓冲区的支持 Alpha 信息旋转的函数，最终成功实现了图片旋转过后依旧能实现背景透明化的效果。
三元光栅操作

```
void c(double x,double y,IMAGE *black,IMAGE *character)
{
        putimage(x,y,black,SRCAND);//显示IMAGE对象
        putimage(x,y,character,SRCPAINT);//显示IMAGE对象


}
```

基于直接操作显示缓冲区

```
// 变量初始化
    DWORD *dst = GetImageBuffer(dstimg);
    DWORD *src = GetImageBuffer(srcimg);
   // 实现透明贴图
   for (int iy = 0; iy < iheight; iy++)
   {
       for (int ix = 0; ix < iwidth; ix++)
       {
           if (src[ix] != transparentcolor)
               dst[ix] = src[ix];
       }
       dst += dst_width;
       src += src_width;
```

2.人物移动时的动画效果
要实现移动时的动画效果就要不断切换显示的函数，我们采用了各种状态函数来记录人物的状态，比如人物此时的左右方向通过此时的角度来判断，而人物跑动时腿动的效果则不断增加人物的某个状态参量的值，然后通过对 2 取余数来使这个参量在几个值之间不断切换，最终在 show()函数中根据状态参量不断切换显示的图片，从而实现人物移动时腿一直处于奔跑的状态，而静止时人物的腿则静止不动。因为这种方法需要不断刷新屏幕，会造成屏幕闪烁的问题，我们用批量绘图函数解决了这个问题。

三、控制方面的问题：

1.人物和子弹的判定
我们项目的人物和子弹的判定的难点主要是我们的判定不是对某条或某几条固定的线进行判定，在于我们每局生成的随机地图都是不一样的，但是我们的地图是用数组来保存的，于是每次对数组进行遍历，分成四种情况，对上面、下面、左面和右面的墙进行坐标判定，然后对人物坐标进行限制。子弹的反弹与人物的判定类似，只不过对坐标的限制改成了对角度的变换。
用链表储存子弹的信息

```
typedef struct bullet
    {   double bullet_x,bullet_y;
        double bullet_angle;
        int i;                                  //i为1则为stu发射的子弹，为0则为inv发射的子弹，每个人最多发射五个子弹
        struct  bullet *next;
        struct  bullet *before;
        int v;


    }B;
```

用结构体储存人物的信息

```
struct character1
    {   double cx,cy;//记录坐标
        double angle;//枪的角度
        double last_cx,last_cy;
        int i;//记录是否是只转枪
        int wl,wr,sl,sr;//记录状态
        int bullet;//是否发射子弹
        int bullet_v;//子弹速度
        double bullet_x,bullet_y;//子弹的坐标
        double lastBullet_angle; //存放上一个子弹的方向
        int score;  //分数
        int bullet_number;//子弹数量
        char s[];
};
```

地形对人物限制的部分代码

```
//人物走到墙边的判定
for(i=0;i<8;i++)
{   for(j=0;j<12;j++)
if(maps[i][j]==0){

 //左墙
    if(inv.cx+95>=100*j&&inv.cx+55<=100*j&&(((inv.cy+40)>=100*i)&&(inv.cy+40<=100*(i+1)))||((inv.cy+100)>=100*i)&&(inv.cy+100<=100*(i+1)))))
            inv.cx=inv.last_cx;
 //右墙
    if(inv.cx+95>=100*(j+1)&&inv.cx+55<=100*(j+1)&&(((inv.cy+40)>=100*i)&&(inv.cy+40<=100*(i+1)))||((inv.cy+100)>=100*i)&&(inv.cy+100<=100*(i+1)))))
            inv.cx=inv.last_cx;
 //上墙
    if(inv.cy+100>=100*i&&inv.cy+40<=100*i&&(((inv.cx+55)>=100*j)&&(inv.cx+55<=100*(j+1)))||((inv.cx+95)>=100*j)&&(inv.cx+95<=100*(j+1)))))
            inv.cy=inv.last_cy;
 //下墙
    if(inv.cy+40<=100*(i+1)&&inv.cy+100>=100*(i+1)&&(((inv.cx+55)>=100*j)&&(inv.cx+55<=100*(j+1)))||((inv.cx+95)>=100*j)&&(inv.cx+95<=100*(j+1)))))
            inv.cy=inv.last_cy;
```

子弹出界时删除子弹所在链表的部分代码

```
 //前不空，后不空
 if(pBullet->next!=NULL&&pBullet->before!=NULL)
 {      pBullet->before->next=pBullet->next;
        pBullet->next->before=pBullet->before;
        pBullet0=pBullet;
        pBullet=pBullet->next;
        pBullet0->before=NULL;
        pBullet0->next=NULL;
        free(pBullet0);
        continue;

 }
```

2.主界面的设计

主界面设计：采用 EasyX 并结合自己制作的贴图效果，完成对主界面显示的实现，并且在主界面操作上，为了增加游戏用户的游戏体验，引用 GetMouseMsg()函数，以此来确定鼠标在程序界面中的位置以及是否点击鼠标进行操作。使用户可以用鼠标直接对程序进行操作，更加方便，体验更好，同时在初始界面也增加了游戏规则选项，阅读完毕后也可以选择返回到主界面，此部分主要采用 while 循环语句和 break 跳出循环。在鼠标触碰到选项键时，选项键会变换颜色，来达到提醒玩家可以按下按钮的目的，同时又增加了美观度。

3.如何保证生成随机地图的可玩性

随机地图的设计：随机地图的显示部分也主要依靠 EasyX 和贴图效果，使得画面得以呈现，在随机地图中，运用数组来储存坐标信息，通过数组储存的数字不同，在相应的位置显示不同的图片，来达到显示地图的效果，采用数组进行地图数据储存还有的好处就是方便移动和子弹进行相关的判定，数组中不同的元素对应地图上不同的像素点，通过数组元素中保存的数据可以判断出地图上应该显示的状态，在移动和子弹反弹上的判定都会非常方便。

随机地图的随机性是通过 srand((int)time(0));赋予种子来实现的，避免了伪随机数带来的每次重复都采用相同数值的问题，使地图在创建过程中不会出现一直没有生成，或生成的都是一样的地图的情况。

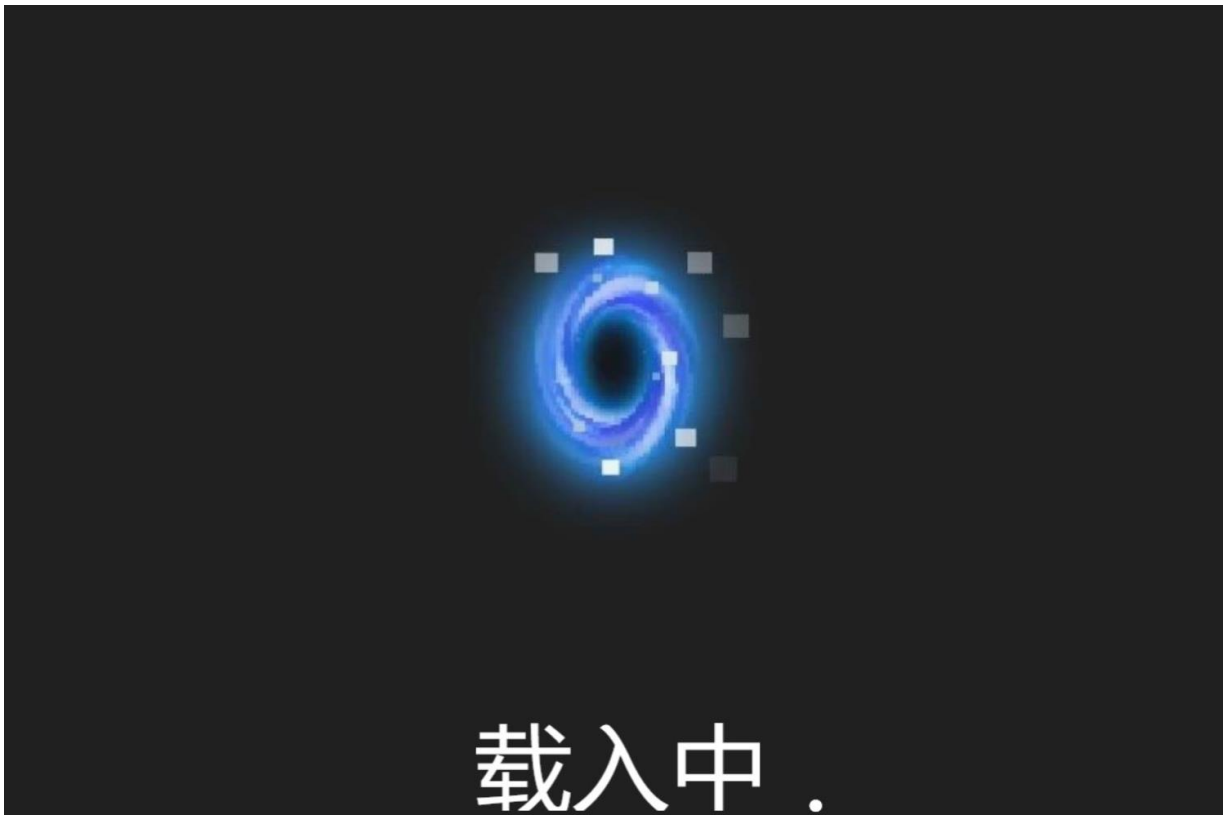
单单是有地图的生成和随机数是不够的，为了确保游戏能够正常的进行，地图的大部分要求是相互连通的，在这里，我采用迭代的方法，先找到一个起始的过道坐标，在他的基础上，分别向上下左右不同的方向寻找是否还有通道，如果找到过道坐标那么继续以这个坐标为基准继续向四周寻找，同时，为了避免下一个过道又找回上一个坐标，我又将已经寻找到附近有过道的坐标对应的数组元素赋予一个无关值，这样就可以很好的避免重复计数，保证了随机地图的正常运转。

```
int passnum(int i,int j)
{
    int pass;//在函数内部记录连续过道的数量
    int a,b;//设置两个变量，用于在将数组前一个数值记录过后，转化为无关数值，避免重复计算
    pass = 0;
    if (maps2[i][j] == 1)
    {
        a = i;
        b = j;
        maps2[a][b] = 0;
        pass++;
        if (maps[i-1][j] == 1)
        {
            pass += passnum(i-1,j);
        }
         if (maps[i+1][j] == 1)
        {
            pass += passnum(i+1,j);
        }
        if (maps[i][j-1] == 1)
        {
            pass += passnum(i,j-1);
        }
         if (maps[i][j+1] == 1)
        {
            pass += passnum(i,j+1);
        }
    }
    return pass;
```

七、测试

初始界面:

载入界面:



游戏界面: