课程设计一 坦克大战

一、游戏简介

相信大部分同窗都玩过或看过“坦克大战”这款典型游戏。目前，就由我们自己动手来开发它。只要人们具有了C++语言和面向对象的基本知识，然后按照实验指南的指引一步一步进行下去，相信我们每个同窗都能把这款典型游戏做出来。

二、实验目的

综合运用C++及其面向对象的知识开发一款小游戏。

三、实验内容

在一种战场上，玩家控制坦克，消灭敌方坦克，并避免敌方坦克摧毁我方基地。游戏的具体规定如下：

1、游戏有一种初始页面，如下图。屏幕上最内部的黑色区域为玩家坦克的活动区域，左上角坐标为(-26,-22)，右下角坐标为(26,22)。当坦克运动到该区域边界时，坦克不能继续迈进。

2、按下任意键开始游戏，玩家控制坦克在战场上穿梭，碰到墙时，不能通过。

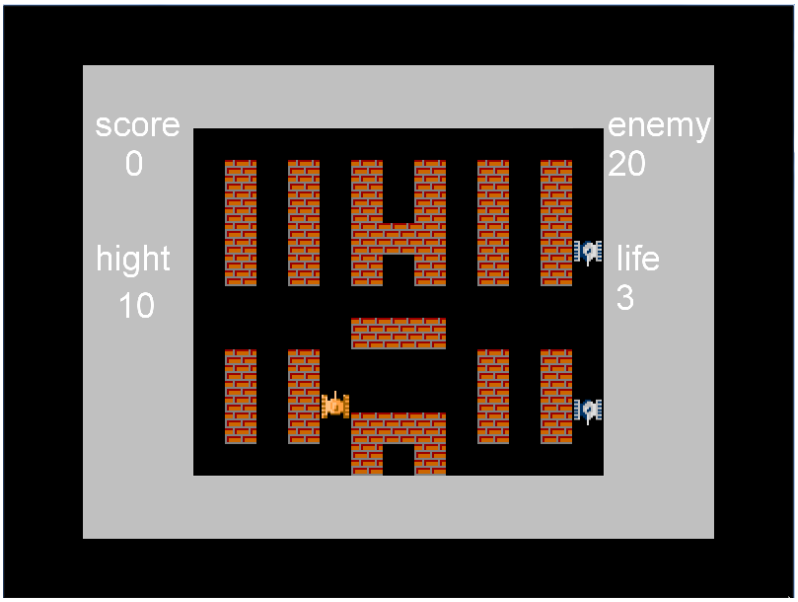
3、敌方坦克自由移动，每隔2秒变化一种方向，每隔3秒发射一发子弹。

4、敌方坦克每隔5秒浮现一辆，从屏幕上方的左、中、右三个位置依次浮现。

5、当玩家被消灭或者我方基地被摧毁或者游戏时间不小于30秒的时候，游戏结束。

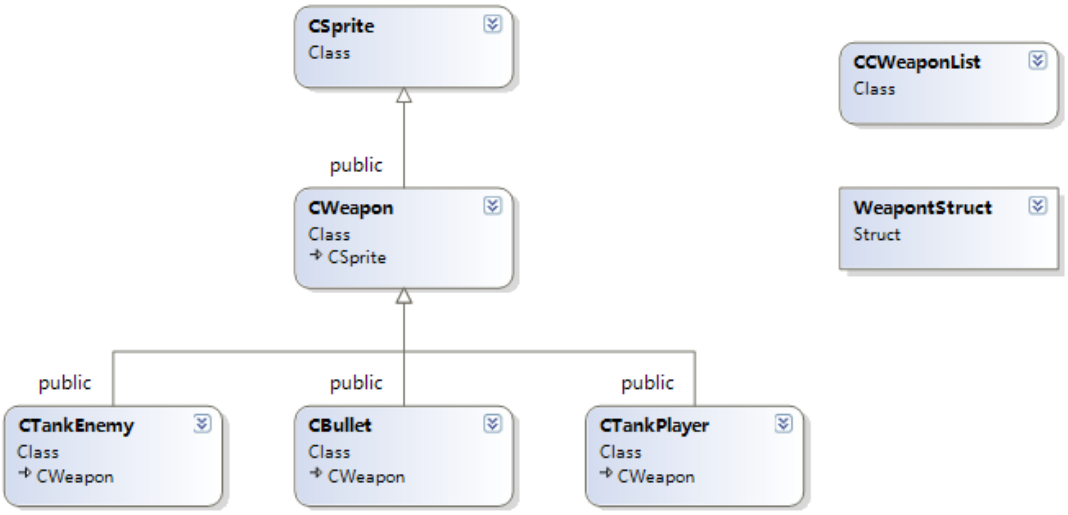

坦克大战
score
0
hight
0
BATTLE
CITY
按任意键开始
enemy
0

游戏开始前


score
0
hight
10
enemy
20
life
3

进入游戏

四、游戏的整体框架



五、实验指南

实验准备

打开 FunCode，创建一种新的 C++项目。注意：项目名称必须为英文和数字，且不能有空格。

点击“项目”→“导入地图模板”，从对话框中选用名称为 TankWar 的模板导入。导入成功后，界面如下：

实验一 游戏开始

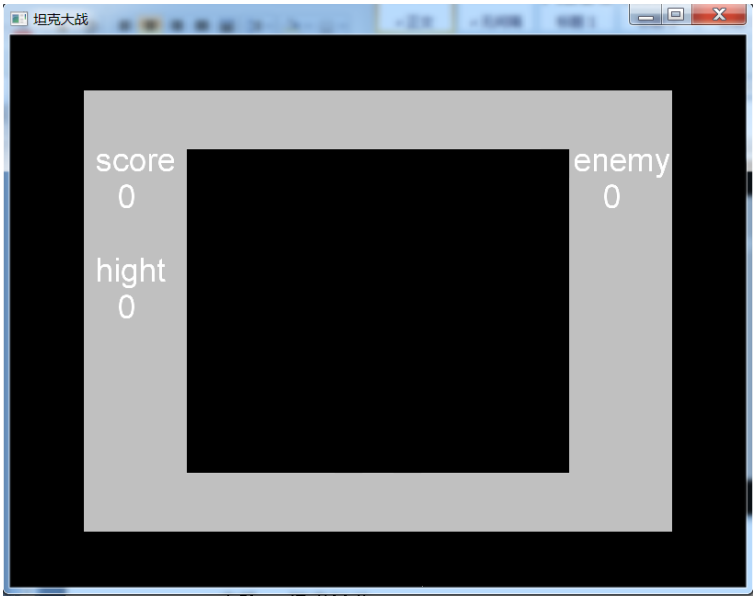
【实验内容】

1、设立游戏标题

2、按空格键,提醒图片消失,游戏进入开始状态.

【实验运营成果】

游戏开始前



按下空格键后

【实验思绪】

要解决 FunCode 中旳图片，我们需要声明 CSprite 类型对象来指向相应图片，然后调用精灵类旳相应函数进行解决。

按下空格键是键盘按下事件，系统调用 CSystem::OnKeyDown 函数进行响应。该函数中调用 CGameMain::OnKeyDown 函数。因此我们可以在 CGameMain 类旳 OnKeyDown 函数完毕相应代码。

按下键盘后，需要变化游戏旳状态，游戏从未开始进入开始状态。成员变量 m_iGameState 用来表达游戏状态。

【实验指引】

1、C++程序旳执行入口是主函数。FunCode 旳主函数名称叫 WinMain，写在 Main.cpp 文献中。CSystem::SetWindowTitle 是设立程序运营窗口标题旳函数，修改如下：

```
CSystem::SetWindowTitle("坦克大战");
```

2、FunCode 程序运营时，当发生键盘按下事件，程序一方面调用并执行 CSystem::OnKeyDown 函数，然后由 CSystem::OnKeyDown 函数调用并执行

CGameMain::OnKeyDown 函数。因此，键盘按下事件的响应代码我们在 CGameMain::OnKeyDown 函数中编写即可。

1、我们要解决的两个精灵如下图，它们的名称分别是 splash 和 start。我们需要创建两个 CSprite 类对象与这两个精灵绑定。

2、在 CGameMain 类中声明两个 CSprite*类型，根据面向对象的封装性原理，成员变量的访问权限应当是 private。代码应当写 LessonX.h 文献中。

```
CSprite* m_pSplash;

CSprite* m_pStart;
```

在 CGameMain 类的构造函数中，对上面两个指针变量进行初始化。通过调用 CSprite 的构造函数，将精灵和精力对象绑定在一起。

```
m_pSplash= new CSprite("splash");

m_pStart= new CSprite("start");
```

3、最后，我们在 CGameMain::OnKeyDown 函数中解决空格键按下事件。

```
if( 0 ==GetGameState() )

{

        if(iKey ==KEY_SPACE)

        {

        m_iGameState = 1;

        }
```

```
}
```

KEY_SPACE 是枚举变量 KeyCodes 的成员，表达空格键。KeyCodes 定义在 CommonClass.h 文献中。该枚举变量定义了所有的键盘值。注意：**必须将 m_iGameState 的值改为 1。**当 GameMainLoop 函数再次执行时，根据 m_iGameState 的值调用并执行 GameInit 函数，游戏进入开始状态。

进入游戏开始状态时，调用 CSprite 类的 SetSpriteVisible 函数将精灵图片设立为不可见。

在 GameInit 函数中添加代码：

```
m_pSplash->SetSpriteVisible(false);

m_pStart->SetSpriteVisible(false);
```

4、程序执行完 CGameMain::GameInit 函数后，执行 CGameMain:: SetGameState(2)，将 m_iGameState 的值改为 2，游戏进入运营状态。

5、在 CGameMain 类的析构函数中 delete 本实验创建的指针对象，以释放分派的内存。

```
CGameMain::~CGameMain()

{

    delete m_pSplash;

    delete m_pStart;

}
```

6、编译并运营程序，然后按下空格键，看看运营效果。

实验二 坦克运动

【实验内容】

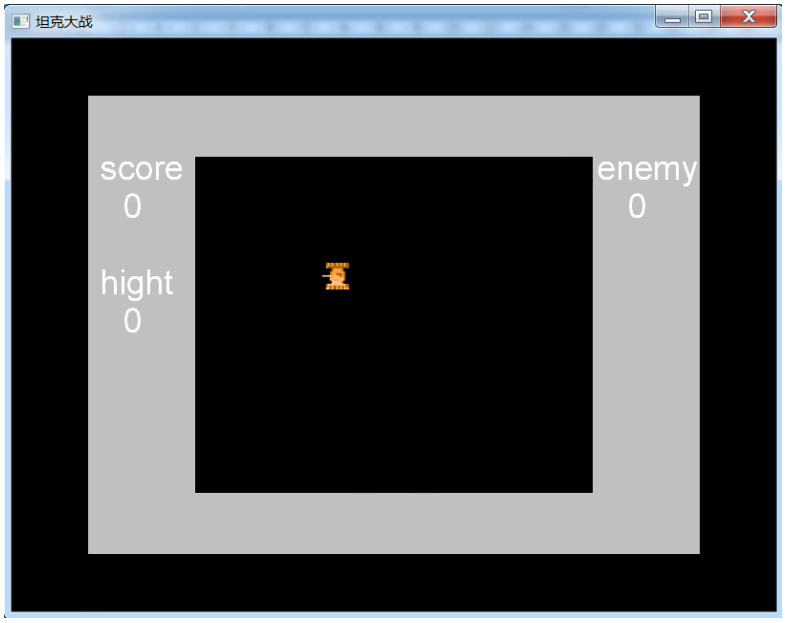
1、创建坦克类 CTankPlayer；

2、游戏开始时，将坦克放置于(0,0)旳坐标上；

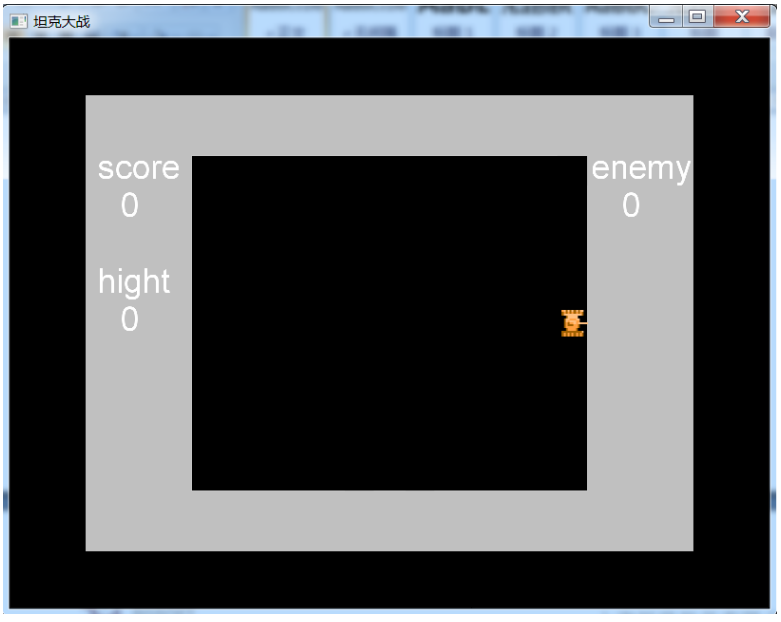
3、通过按键 WSAD 控制坦克上下左右运动；

4、坦克运营到黑色区域边界时不能继续迈进。

【实验运营成果】



在区域内运动



运动到区域边界

【实验思绪】

坦克也是精灵，但是它具有某些自己独特的功能，例如通过键盘控制运动、开炮等。因此我们可以创建一种新的类 CTankPlayer 类，并让它继承 CSprite 类。

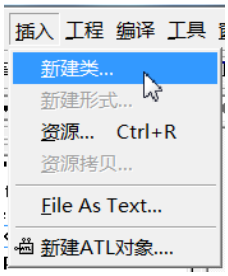
通过 WASD 键，控制坦克做相应的的上左下右运动。按下某个键，给坦克设立相应方向的运动速度；松开时，将该方向的速度设为 0，表达停止运动。

屏幕上最内部的黑色区域为玩家坦克的活动区域，左上角坐标为(-26,-22)，右下角坐标为(26,22)。当坦克运动到该区域边界时，坦克不能继续迈进。

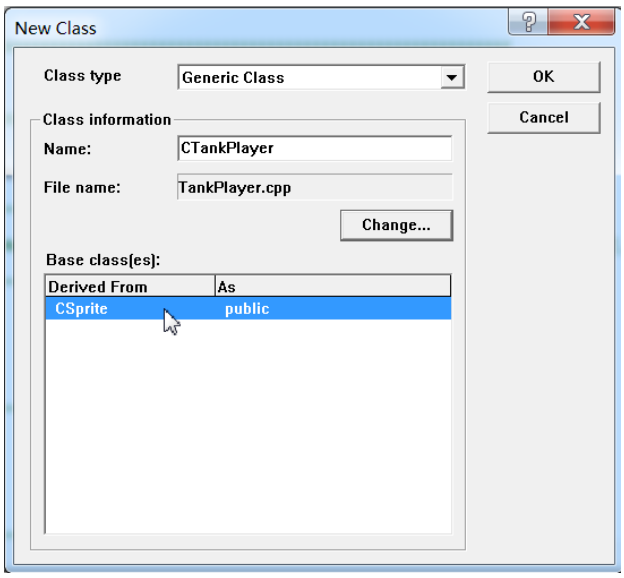
【实验指引】

1、通过类向导创建 CTankPlayer 类，其继承于 CSprite 类。以 VC++ 6.0 为例：

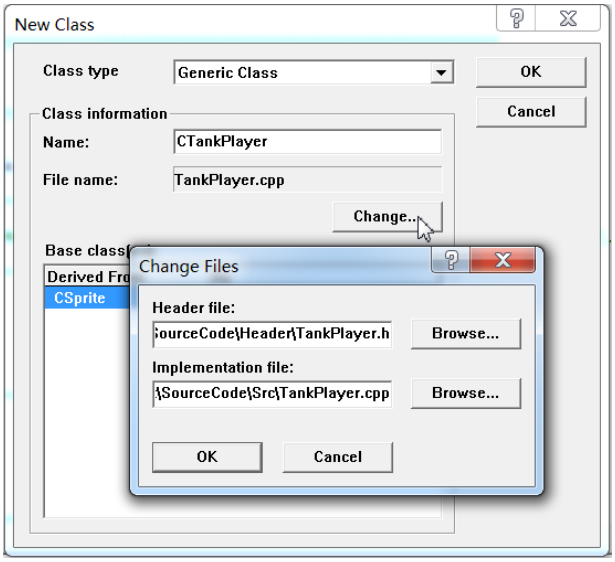
第一步、点击菜单“插入”-〉“新建类”。



第二步、在“New Class”对话框中输入类名和父类名。

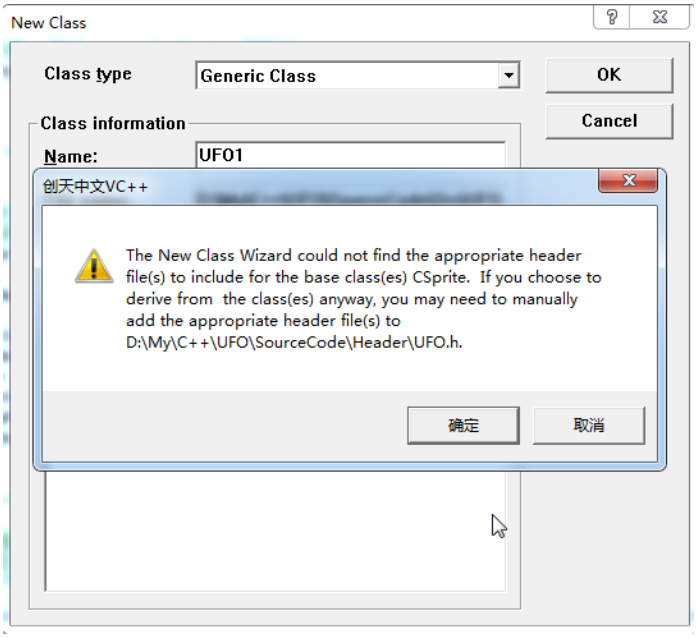


第三步、点击“Change”按钮，在新对话框中修改 CTankPlayer 类的头文献和 cpp 文献的途径。将头文献保存到项目文献夹的\SourceCode\Header 文献夹中，将 cpp 文献保存到项目文献夹下的\SourceCode\Src 文献夹中。



这里需要特别注意的是创建文献途径的问题，所有的.h 头文献应当在项目文献夹\SourceCode\Header 中，所有的.cpp 源文献应当放在项目文献夹下的\SourceCode\Src 文献夹中。这样我们在#include 的时候只需要写文献名就可以了。假如保存到其她途径下，需要写相对途径。

1、点击 OK，浮现下面提醒，不用管它，点击“拟定”。



2、这是由于，我们创建的类集成了 CSprite 类，但是却没有把这个类的声明文献 include 进来。因此我们在新建类的头文献中，做如下操作：

```
#include "CommonClass.h"
```

3、编译程序，发现如下提醒：

```
error C2512: 'CSprite' : no appropriate default constructor available
```

打开 CommonClass.h 文献，找到 CSprite 类的定义，可以发现该类没有缺省构造函数，只有一种带参数的构造函数。因此，作为它的子类，构造函数也必须带参数，并且将参数传递给父类。构造函数的声明和定义修改如下：

```
CTankPlayer(const char* szName);

CTankPlayer::CTankPlayer(const char* szName):CSprite(szName)

{

}
```

4、为 CTankPlayer 类添加 m_iDir,m_fSpeedX,m_fSpeedY，m_iHp 四个成员变量（成员变量。用来表达坦克在 X 轴和 Y 轴方向上的速度以及运营的方向，并且在构造函数中初始化为 0。这里规定，m_iDir 的值为 0、1、2、3，分别表达上、右、下、左。

成员函数的声明和定义如何添加，访问权限是什么，可参照实验一，下文不再继续提醒），分别表达运动方向、X 轴、Y 轴速度以及血量值。**本文档的命名采用匈牙利命名法，**m_表达类成员变量，i 表达整型，f 表达 float 型，sz 表达字符指针，g_表达全局变量等。

同步需要在 public 权限下定义获取和设立这些变量的 Get 和 Set 措施，可在 LessonX.h 文献中完毕：

```
//set 措施
void SetHp(int hp)                  {m_iHp = hp;}
void SetDir(int dir)                {m_iDir = dir;}
void SetSpeedX(float speedX)        {m_fSpeedX = speedX;}
void SetSpeedY(float speedY)        {m_fSpeedY = speedY;}
//get 措施
int GetHp()                         {return m_iHp;}
int GetDir()                        {return m_iDir;}
float GetSpeedX()                   {return m_fSpeedX;}
float GetSpeedY()                   {return m_fSpeedY;}
```

5、在 CTankPlayer 类的构造函数完毕上面四个成员变量的初始化。

```
CTankPlayer::CTankPlayer(const char* szName):CSprite(szName) //对构造函数进行实现
{
    m_iDir=0;
    m_fSpeedX=0.f;
    m_fSpeedY=0.f;
    m_iHp=2;
```

```
}
```

子类对象创建时，要先调用父类的构造函数完毕父类部分的构造。假如父类没有默认构造函数，子类的构造函数必须显示调用父类的构造函数。CTankPlayer 构造函数调用 CSprite 类构造函数，并将参数 szName 的值传递给它，从而将名称为 szName 的精灵图片与 CTankPlayer 对象绑定起来。

6、为 CTankPlayer 类添加 Init 函数，该函数重要用来完毕该类的初始化工作。这里，先调用 setHp 措施设立血量。然后调用父类的 SetSpritePosition 函数将坦克精灵设立在屏幕中央(0,0)处。接下来给坦克精灵设立时间边界的大小，与世界边界的碰撞模式为 WORLD_LIMIT_NULL，表达精灵与世界边界碰撞的响应由代码完毕，同步设立坦克碰撞模式为发送碰撞和接受碰撞。

```
void CTankPlayer::Init()
{
    SetHp(2);
    SetSpritePosition(0.f,0.f);
    SetSpriteWorldLimit(WORLD_LIMIT_NULL, -26, -22, 26, 22);
    SetSpriteCollisionActive(1,1);//设立为可以接受和发生碰撞
    SetSpriteVisible(true);
}
```

7、完毕 CTankPlayer 类有关定义后，在 CGameMain 类中增长一种私有成员变量 m_pTankplayer，类型为 CTankPlayer*。

注旨在 LessonX.h 添加头文献：

```
#include"TankPlayer.h"
```

按下空格键后，程序会调用键盘按下事件响应函数，将 m_iGameState 的值改为 1，游戏进入开始状态（见实验一）。程序再次执行 CGameMain::GameMainLoop 函数时，根据 m_iGameState 的值调用并执行 CGameMain::GameInit 函数。因此，可以在该函数中创建我方坦克，并调用 CTankPlayer::Init 函数来完毕该类对象的初始化工作。

```
m_pTankPlayer=new CTankPlayer("myPlayer");//新建一种名字是 myPlayer 的我方坦克对象
m_pTankPlayer->CloneSprite("player");//我方坦克克隆在 funcode 模板中存在的名字为 player 的坦克，表达新建的坦克对象有目前精灵的所有属性
m_pTankPlayer->Init();
```

8、接下来，我们为 CTankPlayer 类添加 OnMove 函数，参数 iKey、bPress 分别表达按下的是哪个按键和按键与否按下。一方面声明该函数，访问权限为 public：

```
void OnMove(int iKey, bool bPress);
```

9、接着，完毕 OnMove 措施。

```
void CTankPlayer::OnMove(int iKey, bool bPress)
{
    if(bPress)
    {
        switch (iKey)
        {
            case KEY_W:
            SetDir(0);
            SetSpeedX(0);
            SetSpeedY(-8);
```

```
break;
```

```
            case KEY_D:
            SetDir(1);
            SetSpeedX(8);
            SetSpeedY(0);
            break;
            case KEY_S:
            SetDir(2);
            SetSpeedX(0);
            SetSpeedY(8);
            break;
            case KEY_A:
            SetDir(3);
            SetSpeedX(-8);
            SetSpeedY(0);
            break;
        }
        SetSpriteRotation(float(90*GetDir())); //用方向值乘于 90 得到精灵旋转度数
        SetSpriteLinearVelocity(GetSpeedX(),GetSpeedY());
}
else
{
    if(iKey == KEY_W || iKey == KEY_D || iKey == KEY_S || iKey == KEY_A)
    {
```

```
            SetSpeedX(0);
            SetSpeedY(0);
            SetSpriteLinearVelocity(GetSpeedX(),GetSpeedY());
        }
    }
}
```

用参数 bPress 来判断键盘是按下还是弹起。假如 bPress 为 false，表达键盘弹起，且弹起的键是 WASD 键时，设立坦克的 X 轴和 Y 轴移动速度都为 0。假如 bPress 为 true，表达键盘按下，根据按下的键，为 m_iDir,m_fSpeedX,m_fSpeedY 赋予相应的值。SetSpriteRotation 和 SetSpriteLinearVelocity 函数用来设立精灵的角度和线性速度，具体含义参照 CommonClass.h 文献。

10、在 CGameMain 类的 OnKeyDown 函数中，通过调用 CTankPlayer 类的 OnMove 措施，根据按下的键，控制坦克朝指定方向运动。只有游戏进入开始状态，按键才有效。注意：OnMove 参数 bPress 的值为 true。在 OnKeyDown 函数中添加代码如下：

```
if(m_iGameState == 2)
{
   m_pTankPlayer->OnMove(iKey, true);
}
```

11、编译并运营程序。按空格键开始，按 WASD 键，坦克会向规定的方向运动。但是松开按键后，坦克继续迈进，不会停止下来。

12、在 CGameMain::OnKeyUp 函数中调用 CTankPlayer 的 OnMove 措施，参数 bPress 的值为 false。

```
void CGameMain::OnKeyUp(const int iKey)
```

```
{
    if(m_iGameState == 2)
    {
        m_pTankPlayer->OnMove(iKey, false);
    }
}
```

13、编译并运营程序，按下 WASD 键，坦克运营；松开按键，坦克停止。接下来，我们来实现坦克运营到黑色区域边界停止运动的效果。

14、当坦克精灵碰到边界时，将精灵的速度都设为 0，精灵停止运动。一方面，我们需要给该精灵设立世界边界的碰撞模式。可以在 CTankPlayer::GameInit()中完毕。

15、用字符串解决函数 strstr()来判断碰到世界边界的精灵与否为我方坦克。当碰到世界边界的精灵名字中具有”player”的时候，strstr 会返回一种非空值。因此代码如下：

```
void CGameMain::OnSpriteColWorldLimit( const char *szName, const int iColSide )
{
    if(strstr(szName,"myPlayer") != NULL) //判断碰到世界边界的坦克与否为我方坦克
    {
       m_pTankPlayer->SetSpriteLinearVelocity(0,0);
    }
}
```

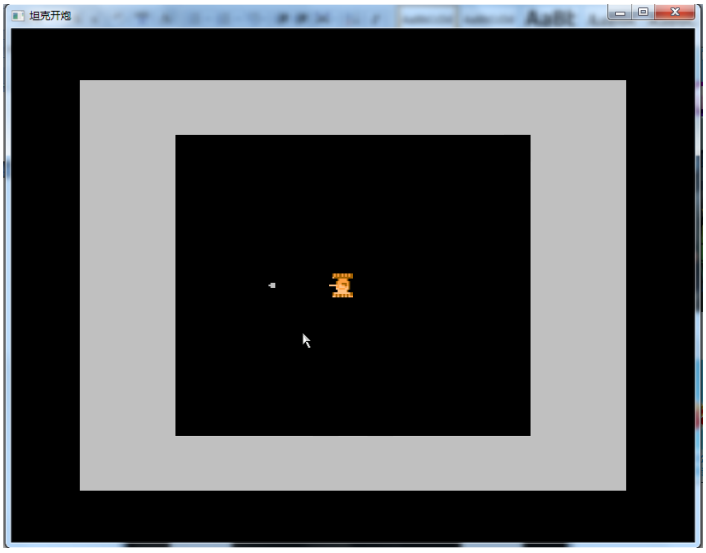
实验三 坦克开炮

【实验内容】

1、创建子弹类 CBullet；

2、通过按键 J 控制坦克开炮；

【实验运营成果】



【实验思绪】

创建子弹类，该类应具有子弹旳运动方向、起始位置、X 轴速度、Y 轴速度等属性，其中运动方向和起始位置由坦克旳方向和位置决定。

通过按键控制坦克发射子弹，因此在 CGameMain 类旳 OnKeyDown 措施中实现该需求。此时我们为 CTankPlayer 类添加发射子弹旳措施 OnFire。当按下 J 键后，坦克发射子弹，在 OnFire 措施中，我们需要告诉子弹旳方向和初始位置。而创建子弹我们由 CGameMain 类来完毕。这样减少了类之间旳耦合。

【实验指引】

1、通过类向导创建 CBullet 类，其继承于 CSprite 类，具体做法参照实验一；

2、为 CBullet 类添加 m_iDir,m_fSpeedX,m_fSpeedY,m_iHp，m_iOwner 五个成员，分别表达方向、X 轴、Y 轴速度、子弹血量以及发射子弹归属坦克，0 表达地方坦克，1 表达我方坦克，并参照实验二添加变量旳 get 和 set 措施。

3、为 CBullet 类添加构造函数和析构函数。参照实验二，把构造函数中将所有变量进行初始化。

1、为 CBullet 类添加 OnMove 措施，参数为 iDir 表达子弹旳运动方向。

2、完毕 OnMove 措施。根据方向 m_iDir，一方面设立 m_fSpeedX,m_fSpeedY 旳值，然后设立根据方向设立旋转角度，最后设立子弹旳运动速度。

```
void CBullet::OnMove(int iDir)
{
    SetDir(iDir);
    switch(GetDir())
    {
        case 0:
        SetSpeedX(0);
        SetSpeedY(-10);
        break;
        case 1:
        SetSpeedX(10);
        SetSpeedY(0);
        break;
        case 2:
        SetSpeedX(0);
        SetSpeedY(10);
        break;
        case 3:
        SetSpeedX(-10);
        SetSpeedY(0);
```

```
            break;
        }
        SetSpriteRotation(90*GetDir());
        SetSpriteLinearVelocity(GetSpeedX(),GetSpeedY());
    }
```

3、在 CGameMain 类中添加表达子弹数目的成员变量 m_iBulletNum(注意初始化为 0)。然后添加 AddBullet 措施。措施中有 iDir、fPosX、fPosY、iOwner 四个参数分别表达子弹方向、子弹初始位置的 X 轴 Y 轴坐标以及子弹所属坦克。由于地图上只有一种子弹模板，因此我们需要先复制这个模板，然后设立该精灵的世界边界。

```
void CGameMain::AddBullet( int iDir,float fPosX,float fPosY ,int iOwner)
{
    char* szName = CSystem::MakeSpriteName("bullet",m_iBulletNum);//创建坦克名字
    CBullet* pBullet = new CBullet(szName);
    pBullet->CloneSprite("bullet");
    pBullet->SetSpriteWorldLimit(WORLD_LIMIT_NULL,-26, -22, 26, 22); //设立世界边界
    pBullet->SetSpritePosition(fPosX,fPosY);
    pBullet->SetSpriteCollisionSend(true); //设立接受碰撞
    pBullet->OnMove(iDir);
    m_iBulletNum++; //子弹个数加 1
    if(iOwner == 1)
    {
        pBullet->SetOwner(1);//1 表达我方坦克发射的子弹
    }
```

```
        else
        {
            pBullet->SetOwner(0); //0 表达地方坦克发射的子弹
        }
    }
```

注意：这里用到了 CBullet 类型，因此需要 include 相应的头文献。

4、为 CTankPlayer 类增长 OnFire 措施，实现发射子弹的功能。在根据坦克的运动状态，得到子弹的有关属性后，通过 AddBullet 措施在游戏中增长一发子弹。

```
void CTankPlayer::OnFire()
{
    float x,y;
    x = GetSpritePositionX();
    y = GetSpritePositionY();
    switch(GetDir())
    {
        case 0:
        y=y-GetSpriteHeight()/2-1;
        break;
        case 1:
        x=x+GetSpriteWidth()/2+1;
        break;
        case 2:
        y=y+GetSpriteHeight()/2+1;
```

```
            break;
            case 3:
            x=x-GetSpriteWidth()/2-1;
            break;
        }
        g_GameMain.AddBullet(GetDir(),x,y,1);
    }
```

由于用到 g_GameMain 这个全局对象，因此需要在 CTankPlayer.cpp 中声明头文献：

```
#include"LessonX.h"
```

5、按 J 键发射子弹。在 CGameMain 类旳 OnKeyDown 函数 if(m_iGameState == 2)下添加

```
if(iKey == KEY_J)//判断按下键是够为 J 键
{
    m_pTankPlayer->OnFire();
}
```

实验四 敌方坦克

【实验内容】

1、创建 CTankEnemy 类；

2、创建一种敌方坦克实例，实现坦克从上方左中右三个位置随机浮现。

3、实现坦克隔 2 秒，随机变化一种方向。

4、实现坦克隔 5 秒发射一发子弹；

5、当坦克与世界边界碰撞时，变化方向；

【实验运营成果】