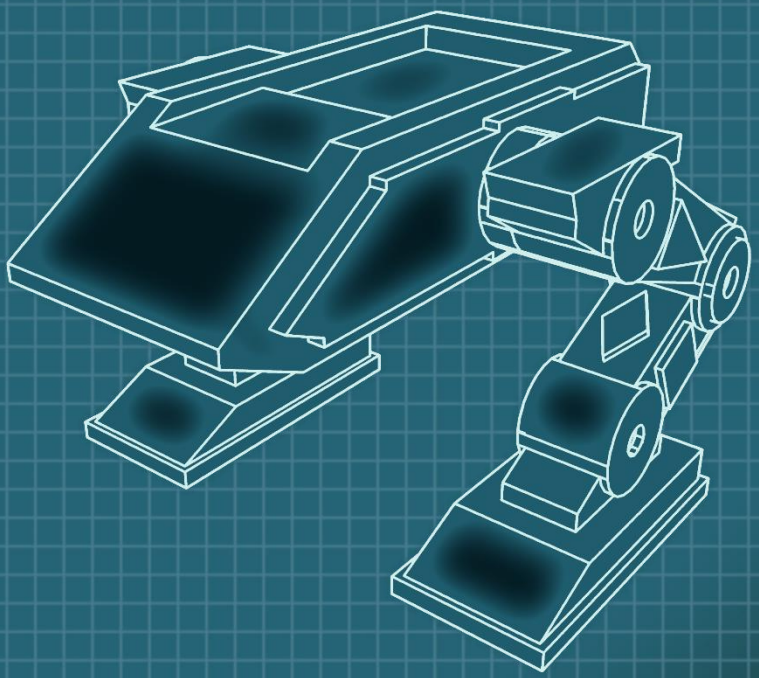
机甲帝国

首届 c++智能体争霸赛

吴健雄学院科技部　先声网



开发者手册

By 狄学长

开发者手册

目录

1 闪电上手

打开 RobotAIFactory 工程

完成 RobotAI.cpp 中的 Update(…)和 ChooseArmor(…)的函数体，完成一些个性化其他函数体

代码中均有详细注释

设置项目属性（以 VS2012 为例）：

1. 解决方案配置为 Release 版本
2. 项目》RobotAIFactory 属性》配置属性》常规》公共语言运行时支持，设置为“公共语言运行时支持(/clr)”

生成（Build）工程（首选 VS2012），在 Release 文件夹中找到 RobotAIFactory.dll，这就是你的 AI。

ArmorClient.exe 用于加载 dll 并展开战斗和生成录像。

ArmorFlash.exe 作为门户，可以播放战斗录像。

将 RobotAIFactory.dll 重命名为你的学号，在活动 QQ 群上传

2 开发详解

2.1 资源包的内容

RobotAIFactory 文件夹：

RobotAIFactory 工程就是你开发自己的机甲 AI 的地方，使用 c++的 IDE，按照代码模板进行编码，最终生成你自己的 AI .dll 文件

ArmorClient 文件夹：

ArmorClient.exe 是机甲战斗的调试器，加载 AI.dll 可以进行战斗，生成调试记录和生成战斗录像。也可以自己模拟联赛。

同目录下的一系列.dll 不是 AI，而是支持 EXE 运行的 DLL 文件，请保持它与 ArmorClient.exe 置于同一目录下。若你的机器是 32 位且 ArmorClient.exe 无法运行，可以尝试用 dll32 中的 dll 将 ArmorClient 目录下的 dll 覆盖掉。

GameData.lua 是游戏的数据脚本，请保持它与 ArmorClient.exe 置于同一目录下。

ArmorFlash 文件夹：

ArmorFlash.exe 是机甲帝国的前端门户，里面有关于机甲的游戏性质的介绍，外观预览等；同时也是战斗录像的播放器。它的 swf 版本在网页上也可以找到。

BGM50.mp3 是背景音乐，请保证它与 ArmorFlash.exe 置于同一目录下。

AI 文件夹：

里面是一些狄学长写的可供测试的弱智 AI，你可以体会一下践踏他们的感觉。

BattleRecord 文件夹：

里面是一些战斗录像，由弱智 AI 们倾情出演。

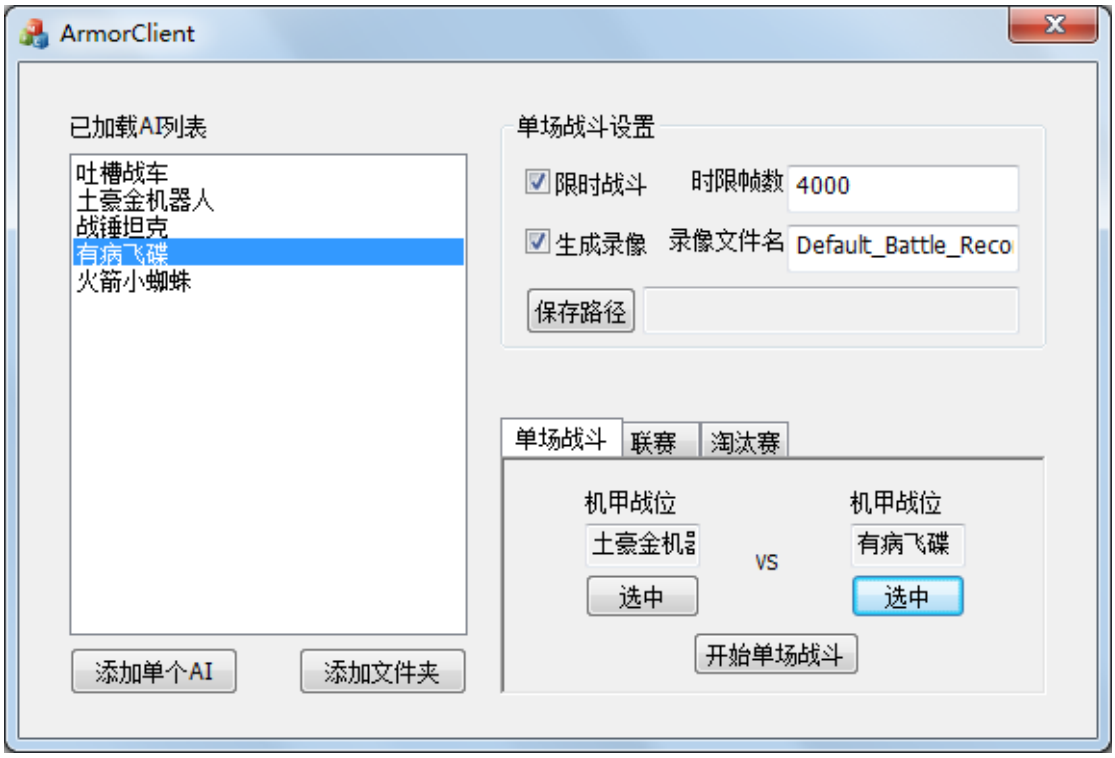
Document 文件夹：

里面是《开发者文档》和《游戏和赛制文档》

虽然我们尽力避免更新，但首届比赛经验不足难免出现无法预计的问题。我们如果更新，会在 QQ 群通知，并且我们尽量不影响到大家已有的编程（保证接口不变）。

2.2 ArmorClient.exe 简要帮助

ArmorClient.exe 是用于调试 AI，并生成机甲 AI 之间战斗录像.txt 文件和统计信息等的程序。使用起来并不困难，这里做简要介绍。



这是程序的主界面。左侧是已加载的 AI 列表。

加载 AI 的两种方式：单个添加，和文件夹添加

【注意 1】文件夹添加是添加整个文件夹中的所有.dll 文件，请确保该文件夹中的.dll 全部为合法 AI。

【注意 2】程序在若干环境不同的 PC 上进行了测试，基本能顺利进行。有的机器加载 AI 列表中可能显示乱码，但不影响其他操作。有的机器在添加文件夹时会程序终止，那么就请一个一个加载吧。大家可以在 QQ 群中向我们反应问题，我们也将努力优化。如果你的机器上程序运行有问题，装上 VS2012 可能会是最完美的解决方案。

AI 列表中，单击 AI，再单击单场战斗标签页的选中，即可将此 AI 送上擂台等待战斗。

AI 列表中，双击 AI，可浏览其统计成就信息。

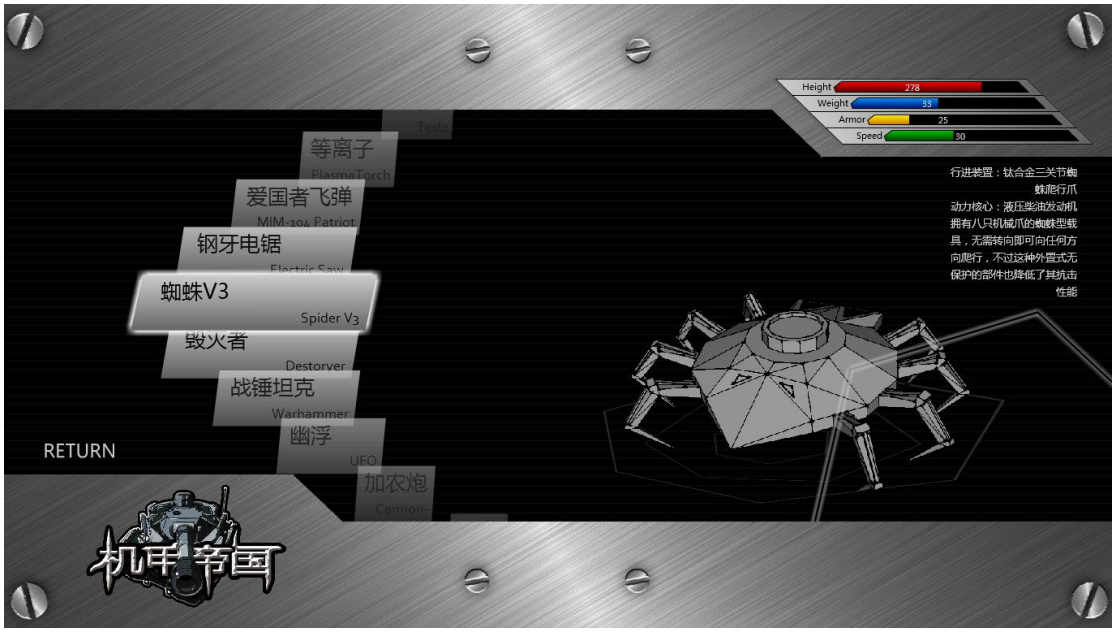
联赛和淘汰赛模式的参赛者是全部已加载 AI。

其他的比如战斗模式应该一看就懂吧（战斗时限别设太大啊，这里不怎么健壮）

2.3 ArmorFlash.exe 简要帮助

ArmorFlash.exe 充当一个门户和战斗播放器，主要是游戏性的效果展示。
其实这个用起来应该不需要什么帮助的吧。。。

【播放战斗】是通过加载 ArmorClient 生成的战斗录像.txt，来播放战斗。
【机甲介绍】点了就知道啦。。。
【参数预览】供开发者看一下自己的机甲在战场上最终是什么样。特别是颜色的设置，可以把在这里面设定得满意了的值记下，在 AI 代码中的颜色部分返回这个值。
【关于赛事】点了就知道啦。。。
【离开战场】点了就知道啦。。。



2.4 RobotAIFactory 详解

RobotAIFactory 工程就是开发机甲 AI 的主战场。首先使用 c++的集成开发环境打开它。

【注意 1】项目的配置（以 VS2012 为例）

1. 解决方案配置为 Release 版本
2. 项目》RobotAIFactory 属性》配置属性》常规》公共语言运行时支持，设置为“公共语言运行时支持(/clr)”

【注意 2】IDE 首选 VS2012，也可以选择 VS2013（初步通过测试可行）；正式开始编写 AI 之前，请直接生成空模板，在 Release 文件夹中找到生成的 RobotAIFactory.dll，使用 ArmorClient.exe 加载，并与其它 AI 生成战斗，使用 ArmorFlash.exe 观看战斗。若整个过程顺利进行，则说明你的 IDE 版本是可用的。如果其中发生错误，我们首先推荐你安装 VS2012 进行编写。如果你不想这么做，你也可以将自己的代码上传到 QQ 群上并注明，我们的工作人员将帮助你生成 dll 后发送给你，这中间需要一些时间。

RobotAIFactory 工程里面的文件列表如下

文件名	描述
RobotAI.h	你自己的 AI 类的头文件
RobotAI.cpp	你自己的 AI 类的 cpp 文件，编写 AI 的主阵地
RobotAI_Interface.h	AI 类的接口，有一些功能函数宏可以调用，不要改动
RobotAIstruct.h	一些与 AI 编写相关的结构体以供参考，不要改动
struct.h	一些底层功能性代码一共参考，不要改动
GlobalFunction.cpp	一些可能会用到的基本函数以供参考
stdafx.h	用来包含头文件和标准库，可以添加自己需要的头文件
Targetver.h **Dllmain.cpp** **RobotAIFactory.cpp** **Stdafx.cpp**	这些文件既不需要看，也不需要改动，直接忽略吧

你还可以自己添加需要的头文件和 cpp 文件，比如编写某个类，实现某些函数等。

1.1.1 RobotAI.h & RobotAI.cpp

这两个文件是最为重要的，它们对应你自己的机甲类 RobotAI，将实现你的机甲的主要功能。

1.1.1.1 帧操纵函数 Update(…)

Update 函数是每帧对机甲进行操纵的函数，如果这个函数体里什么都不写，你的机甲就将呆在原地挨打。

三个参数的含义：

RobotAI_Order& order 机甲操纵指令，你在函数体中给它赋值以操纵机甲在这一帧的行为（在开发者手册 RobotAIstuct.h 一节中有它的详细介绍）

const RobotAI_BattlefieldInformation& info 战场信息（在开发者手册 RobotAIstuct.h 一节中有它的详细介绍）

int myID 自己机甲在 info 中 robot 数组对应的下标

```
    void RobotAI::Update(RobotAI_Order& order,const RobotAI_BattlefieldInformation& info,int
myID)
    {
     //帧操纵函数
     //功能：在每一帧被调用，完成你的机甲在这一帧的动作决策
     //参数：order ...  机甲操纵指令，你在函数体中给它赋值以操纵机甲在这一帧的行为
     //        info ...  战场信息
     //        myID ... 自己机甲在info中robot数组对应的下标
     //        (这几个参数的详细说明在开发手册可以找到，你也可以在RobotAIstruct.h中直接找到它们
的代码)



    }
```

1.1.1.2 选择装备函数 ChooseArmor

ChooseArmor 函数将在战斗开始被调用，用以挑选相应的武器和引擎。

这个函数必不可少，不过很简单，只要对两个枚举类型的参数赋值即可。

weapontypename 和 enginetypename 的内容可以在 RobotAIstruct.h 一节中找到。

```
void RobotAI::ChooseArmor(weapontypename& weapon,enginetypename& engine,bool a)
{
//挑选装备函数
//功能：在战斗开始时为你的机甲选择合适的武器炮塔和引擎载具
//参数：weapon    ... 代表你选择的武器，在函数体中给它赋值
//       engine   ... 代表你选择的引擎，在函数体中给它赋值
//tip:   括号里的参数是枚举类型 weapontypename 或 enginetypename
//       开发文档中有详细说明，你也可以在RobotAIstruct.h中直接找到它们的代码
//tip:   最后一个bool是没用的。。那是一个退化的器官

weapon=WT_Cannon; //啊，我爱加农炮
engine=ET_Spider; //啊，我爱小蜘蛛
}
```

1.1.1.3 个性信息函数组

获取机甲名称，开发者（团队）名称，武器和引擎的颜色 RGB 偏移值。非常简单，真的只需要看代码注释就行了，我这里就不写了。。。

1.1.1.4 触发函数组

这组函数是会在特定时刻被触发的。不是必须完成，你完全可以置之不理。

onBattleStart 一场战斗开始时被调用，可能可以用来初始化

onBattleEnd 一场战斗结束时被调用，可能可以用来析构你动态分配的内存空间（如果你用了的话）

onHit 被子弹击中时被调用

1.1.1.5 自定义函数和成员变量

最后，你也可以在 RobotAI 类中添加你自己的函数和相应的实现。方法是在 RobotAI.h 添加成员变量声明和函数声明，在 RobotAI.cpp 中添加函数定义。

1.1.2 RobotAI_Interface.h

AI 类的接口，不需要也不能做任何改动。

该头文件的开头部分定义了一些可以在 RobotAI.cpp 等中调用的功能函数的宏名

```
//-------------------------------------------------------------
//功能函数指针的宏名，使用宏名可以使调用以下功能函数更符合习惯
//-------------------------------------------------------------

//输出调试string
//用法：TRACE("操，老子挨了一枪");
#define TRACE (*trace)


//返回武器、引擎的相应静态数据信息（下面这一大坨）
//用法：get_weapon_name(WT_Cannon)
//tip:   括号里的输入参数是枚举类型 weapontypename 或 enginetypename
//       开发文档中有详细说明，你也可以在RobotAIstruct.h中直接找到它们的代码
#define get_weapon_name (*getWeaponName)
#define get_weapon_damage (*getWeaponDamage)
#define get_weapon_ammo (*getWeaponAmmo)
#define get_weapon_coolingTime (*getWeaponCoolingTime)
#define get_weapon_inaccuracy (*getWeaponInaccuracy)
#define get_weapon_rotationSpeed (*getWeaponRotationSpeed)
#define get_engine_name (*getEngineName)
#define get_engine_maxSpeed (*getEngineMaxSpeed)
#define get_engine_maxHp (*getEngineMaxHp)
#define get_engine_rotationSpeed (*getEngineRotationSpeed)
#define get_engine_acceleration (*getEngineAcceleration)

//-------------------------------------------------------------------------
```

TRACE(string debug_info) 输出调试信息到调试窗口（仅限单场战斗）
使用例子：TRACE(“老子挨了一枪\n”);

get 系列方法：获得相应装备的数据
使用例子：get_weapon_rotationSpeed(WT_Cannon)，得到加农炮的炮塔旋转速度。

1.1.3 RobotAIstruct.h

1.1.3.1 机甲操纵码 RobotAI_Order

在 Update 函数中，RobotAI_Order 的一个实例 order 作为参数之一被传入，在函数体中对 order 进行赋值可以实现对机甲在这一帧的行为进行操纵。

```
struct RobotAI_Order
{
//用于RobotAI中Update(..)方法的对机器人下达的操作命令
int fire;    //控制武器开火与否
int wturn;   //控制武器旋转与否
int run; //引擎操纵码之一，影响速度，具体功能因所选引擎而异
int eturn;   //引擎操纵码之一，影响旋转，具体功能因所选引擎而异

RobotAI_Order(){fire=0;wturn=0;run=0;eturn=0;}
};
```

操纵码详解

操纵码成员变量	值>0		值==0		值<0
fire	武器开火		无动作		
wturn	武器顺时针旋转		无动作		武器逆时针旋转
run(毁灭者)	全速前进		立即停止		全速后退
run(战锤坦克)	加速前进		刹车		
run(幽浮)	沿当前 rotation 加速		自然减速		沿当前 rotation 加速
	1	2	3	4	其他值
run(蜘蛛 V3)	左	右	上	下	无动作
	值>0		值==0		值<0
eturn(毁灭者)	原地顺时针旋转		停止旋转		原地逆时针旋转
eturn(战锤坦克)	右拐（顺时针）		无动作		左拐（逆时针）
eturn(幽浮)	加速方向顺时针旋转		无动作		加速方向逆时针旋转
eturn(蜘蛛 V3)	无动作				

【注意 1】操纵码的赋值与事件的发生并不等价。例如，当武器弹药耗尽或冷却时间未归零时，即使 fire>0 也无法进行开火生成子弹；再如，当引擎达到最高速时（战锤坦克和幽浮），或行进方向被阻挡时，即使 run>0 也无法使机甲行进到相应位置。

【注意 2】毁灭者的特性：当 eturn!=0 时（即旋转时），无法前后移动，即使 run!=0。（即，移动和旋转不能同时执行，旋转优先级大于移动）

【注意 3】武器炮塔的角度是独立的，与引擎载具的角度无关，即不会因为引擎载具的转动而转动。

【注意 4】只要机甲与军火库接触，并且军火库已经冷却完成，不需任何操作，机甲将补满弹药，军火库重新冷却。

【提示】如何理解四种引擎载具的移动方式：蜘蛛 V3 是最普通的 RPG 人物走法：上下左右四个方向，无需加速和旋转；毁灭者是只能沿当前朝向前后移动，无需加速，旋转半径为 0，且同一时刻旋转与移动不能同时发生；战锤坦克的移动方式最接近现实中的汽车：只能沿朝向前进，需要加速，旋转的角速度正比于当前径向速度；幽浮就是飞碟那样的漂浮状，需要加速，当前速度方向与加速度方向无关，旋转时旋转的是加速度方向而非当前速度方向。

1.1.3.2 战场信息 RobotAI_BattlefieldInformation& info

AI 可以从这些结构体中获得战场上的全部信息。
结构体的形式是很清晰的，直接阅读代码或使用代码提示更加容易理解。

```
//每帧的战场总信息结构体
struct RobotAI_BattlefieldInformation
{
//使用最基本的数组存储

int num_robot;    //机甲数量（本届争霸赛衡为2了，包括已经挂了的）

RobotAI_RobotInformation robotInformation[Info_MaxRobots];


int num_bullet;   //当前子弹数量（便于循环访问于何时终止）

RobotAI_BulletInformation bulletInformation[Info_MaxBullets];


int num_obstacle; //地图上的障碍物数量（便于循环访问于何时终止）

Circle obstacle[Info_MaxObstacles];


int num_arsenal;  //地图上的军火库数量（便于循环访问于何时终止）

RobotAI_ArsenalInformation arsenal[Info_MaxArsenals];



Box boundary;     //战场边界，Box结构体
};
```

需要注意:
Circle 就是圆形；(x,y)圆心坐标，r 半径；
Box 就是矩形；(x,y)矩形中心坐标，width 水平宽度，height 高度；
Robot（机甲）的形状都是 Circle；
Bullet（子弹）的形状都是 r=0 的 Circle；
Obstacle（障碍物）的形状都是 Circle；
Arsenal（军火库）的形状都是 Circle；
boundary（战场边界）是 Box；
光棱和磁暴的子弹是瞬间的射线，所以不会在 bulletInformation 中出现。

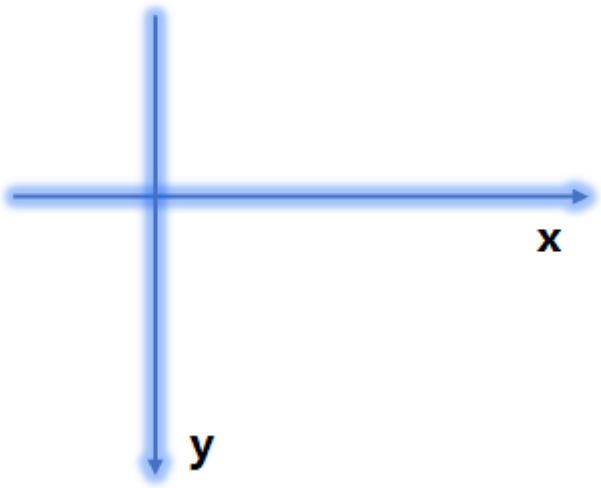
1.1.3.3 枚举类型

weapontypename：武器炮塔，9 种
enginetypename：引擎载具，4 种
bullettypename：子弹，9 种
详见代码

1.1.4 struct.h & GlobalFunction.cpp

主要是一些通用结构体和函数，可以在代码中使用，也可以不使用而自己开发完整的代码，比如高手们。

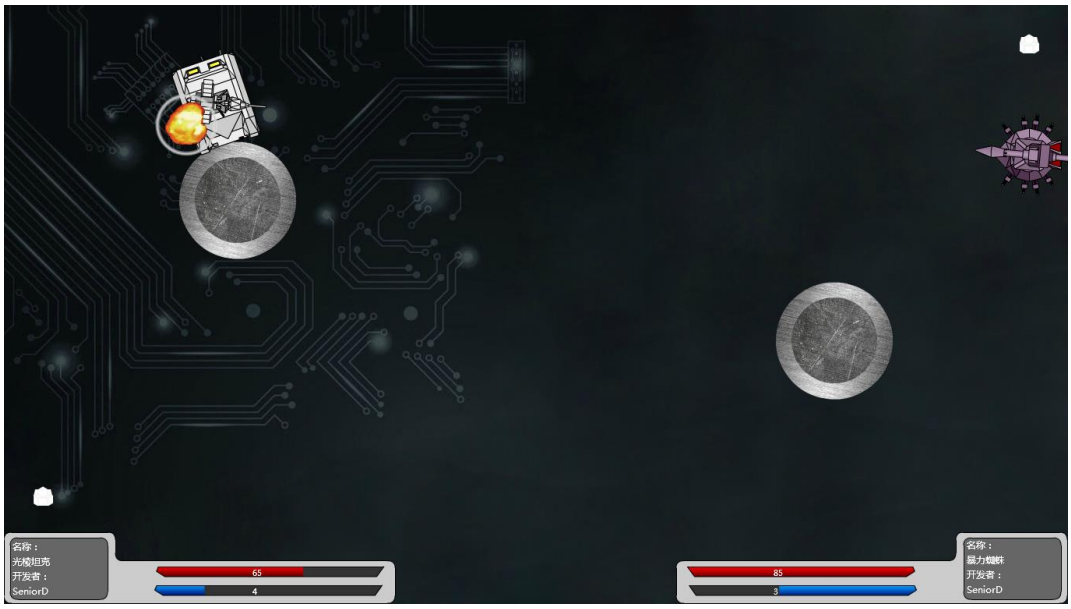
2.5 坐标系统



如图所示。地图的左上角坐标为(0,0)
这样一来，象限数从右下角开始顺时针数过来就是：一、二、三、四。在计算角度时（如用 atan2 时可能需要知道）

2.6 地图

本届大赛使用一张固定的地图以减小 AI 开发难度。地图的信息如下：



战场尺寸：1366*680
对于 Box boundary 而言
boundary.width=1366
boundary.height=680
boundary.x=683
boundary.y=340

出生点 0 坐标(50,50)
出生点 1 坐标(1316,630)

障碍物：2 个
障碍物 0：坐标(300,250)，半径 75
障碍物 1：坐标(1066,430)，半径 75

军火库：2 个
军火库 0：坐标(50,630)，半径 5
军火库 1：坐标(1316,50)，半径 5

冷却时间：1000

当然也可以在代码中的 info 获取这些数据，动态规划路径