

**Ocean Scream AI编写文档**

**For Java**

# 1.概述

1. 主程序与AI控制程序之间使用Socket通信。
2. 我们为参赛选手编写了SDK，封装了必要的数据结构和网络通信逻辑。选手使用SDK开放的接口编写AI程序。
3. 选手在使用Ocean Scream SDK时唯一需要做的是编写一个名为AI的类，该类继承自AbstractAI抽象类，每一个AI的逻辑部分需要在Iteration()方法中实现。该方法会在整个游戏的进行过程中不间断的被循环执行。
4. 选手在编写AI时可以自行创建新的类，但是新的类只能作为AI类的内部成员，即用户需要保证全部代码处于且只处于AI.java文件中。
5. 选手最终提交的文件为单独的AI.java文件。
6. Ai在游戏进行的过程中可以无限制的向游戏主程序发送指令，但是游戏主程序在处理AI程序提交的指令时会有100ms的时间间隔，如主程序在0ms时刻接收了AI程序A提交的1条指令，而之后在50ms时AI程序A又向主程序提交了第二条指令，则第2条指令不会被立即执行，而是加入到等待队列中，在100ms的时刻才会被游戏主程序处理。
7. 由于主程序端含有每秒10条的指令数限制，AI代码应建立相应的计时机制。

# 2.常量定义

所有常量均带有public static声明标示符,定义于OSInterface.

|  |  |  |
| --- | --- | --- |
| 常数名 | 说明 | 值 |
| float MaxArmor | 船只最大护甲值 | 1000 |
| float Acceleration | 船只加速度每秒 | 10 |
| float MaxSpeed | 船只的最大速率每秒 | 25 |
| float AngularRate | 船只转向角速度角度每秒 | 60 |
| float ShipBoundingRadius | 船只碰撞半径 | 15 |
| float IslandBoundingRadius | 岛屿碰撞半径若要实现地图切换则不一定是常数 | 32 |
| float CannonSpan | 单侧炮台射击时间间隔秒 | 4 |
| float CannonAngle | 单侧炮台射击范围扇形角度角度 | 90 |
| float CannonRange | 炮台射击最大距离 | 250 |
| float[]ResourceRestoreRate | 占领资源点的回复速度每秒 | {0,5,10,15,25,50} |
| float ResourceRadius | 资源点占领半径若要实现地图切换则不一定是常数 | 196 |
| int MapWidth | 地图宽度 | 2048 |
| int MapHeight | 地图高度 | 2048 |
| float RangeOfView | 视野大小 | 400 |

# 3.辅助类

## 3.1 ShipInfo

该类储存了地图中一条船的详细信息，即关于该船的各种属性。具体使用参考后面的实例。

|  |  |
| --- | --- |
| int ID | 得到船只的ID |
| int Faction | 得到当前船只所属阵营ID |
| float Armor | 得到船只剩余护甲(即生命值) |
| float PositionX | 得到船只的位置的X坐标 |
| float PositionY | 得到船只的位置的Y坐标 |
| float VelocityX | 得到船只当前速度向量的X分量 |
| float VelocityY | 得到船只当前速度向量的Y分量 |
| float CurrentSpeed | 得到船只的当前速率 |
| float DirectionX | 得到船只朝向的单位向量的X分量 |
| float DirectionY | 得到船只朝向的单位向量的Y分量 |
| float Rotation | 得到船只的旋转角量，值在-180和180之间，顺时针为正 |
| boolean IsMoving | 仅给予己方船只的数据得到船只是否在尝试向前移动 |
| boolean IsBlocked | 仅给予己方船只的数据得到船只是否尝试移动但被挡住了 |
| boolean IsRotating | 仅给予己方船只的数据得到船只是否正在执行朝目标点或角量旋转 |
| float[] CooldownRemain | 仅给予己方船只的数据  得到炮击的剩余冷却时间，0是右炮台，1是左炮台 |

## 3.2 ResourceInfo

该类储存了地图中一个资源点(小岛)的详细信息，即关于该资源点的各种属性。具体使用参考后面的实例。

|  |  |
| --- | --- |
| int ID | 资源点ID |
| int Faction | 资源点目前归属阵营ID |
| float PositionX | 资源点的X坐标 |
| float PositionY | 资源点的Y坐标 |

# 4.主接口类:OSInterface类

提供AI向游戏主程序发送指令的全部方法接口，并提供获取当前AI阵营及游戏运行状态的方法。

## 4.1公有属性：

|  |  |
| --- | --- |
| int Faction | 返回AI所控制的阵营ID。 |
| boolean Running | 返回主程序端游戏的运行状态，为false则代表主程序端游戏已经结束。 |
| List<ResourceInfo> Resource | 返回储存所有资源点的List |
| List<ShipInfo> Ship | 返回储存所有双方船只的List |

注意：游戏中的己方战舰只能看见距离自己400的敌方战舰，即战争迷雾，所以调用

Interface.Ship返回的船只为己方所有船只与己方所有船只可见范围内的敌方船只。小岛不受战争迷雾影响，所以Resource返回的是全部的资源点。

## 4.2 接口方法

### 4.2.1 Attack指令

接口原型：public void Attack(int sourceShip, int targetShip)

在Iteration()方法中调用OSInterface.Attack(sourceShip,targetShip)便会向游戏主程序发送相应的Attack指令，如果符合攻击要求，游戏会执行sourceShip攻击targetShip指令，其中sourceShip为已方船的编号，targetShip为攻击目标船的编号。Attack指令中的目标参数可接受0，使船只失去目标，便于选手精确控制火力（避免船在低伤害位置自动开火）.

若sourceShip，targetShip的数值不合法，比如超出了地图中请求的数目，或者sourceShip所代表的船所属阵营不同于AI的控制阵营，则攻击失败。

无论Attack指令执行是否成功执行，均会消耗一条指令限额，因此建议在执行指令前对指令参数的合法性进行完备的检查。

### 4.2.2 MoveTo指令

接口原型：public void MoveTo(int sourceShip, float x, float y)

在Iteration()方法中调用OSInterface.MoveTo(sourceShip,x,y)便会向游戏主程序发送相应的MoveTo指令，如果符合移动要求，游戏会执行把sourceShip移动到(x,y)的指令，其中sourceShip为已方船的编号，(x,y)为要将船到达的坐标。

若sourceShip，x，y的数值不合法，比如超出了地图中请求的数目，或者sourceShip所代表的船所属阵营不同于AI的控制阵营，则移动失败，。

无论MoveTo指令执行是否成功执行，均会消耗一条指令限额，因此建议在执行指令前对指令参数的合法性进行完备的检查。

### 4.2.3其他指令

以下方法返回类型均为void.

|  |  |
| --- | --- |
| 方法 | 说明 |
| Stop(int ship) | 停止移动和旋转 |
| StopRotating(int ship) | 停止旋转 |
| StartMoving(int ship) | 开始向前移动，忽略MoveTo的到达目标附近时停止移动的命令 |
| StartRotatingTo(int ship, float x, float y) | 开始朝目标点旋转 |
| StartRotating(int ship, float target) | 开始朝目标角量旋转。目标角量以上方为0,顺时针为正，大小在-180到180之间。 |

# 5.Java语言AI示例

### 5.1使用Eclipse等IDE

1. 创建一个java工程。
2. 在项目属性中的JAVA BUILD PATH中选择Add External JARs，然后进入Glorywar.jar所在的目录选择Glorywar.jar。
3. 创建一个名为AI的类，并在我们提供的模板的基础上编写iteration()方法。一个简单的待完善的AI代码示例如下：

AI.java

package MSTCOSSDK;

import java.util.ArrayList;

import java.util.List;

public class AI extends AbstractAI {

public AI(OSInterface Interface) {

super(Interface);

}

List<ShipInfo> ships, a, b;

private float dist(ShipInfo a, ShipInfo b)

{

float tmp=(a.PositionX - b.PositionX) \* (a.PositionX - b.PositionX) + (a.PositionY -b.PositionY) \* (a.PositionY - b.PositionY);

return (float) Math.sqrt(tmp);

}

@Override

public void Iteration()

{

ships = Interface.Ship; //获取己方所有船只及可见范围内的敌方船只

a = new ArrayList<ShipInfo>();

b = new ArrayList<ShipInfo>();

for (int i = 0; i < ships.size(); i++)

if (ships.get(i).Faction == Interface.Faction) a.add(ships.get(i));

else b.add(ships.get(i)); //筛选敌我船只，我方船只放入a,敌方船只放入b

Interface.MoveTo(a.get(0).ID,100,100); //我方0号战舰向(100,100)前进

Interface.StartMoving(a.get(1).ID); //我方1号战舰前进

}

public static void main(String args[])

{

String[] temp =new String[2];

temp[0] = "buaa";

temp[1] = "255.0.255";

SDK.startGame(temp);

}

}注意：

1.main函数里必须用调用SDK.StartGame(String[] temp)，其中第一个字符串为战队名，第二个字符串为舰队的颜色，格式为R.G.B，其中0<=R,G,B<=255。

然后将这个文件编译运行即可。

## 5.2 AI程序使用方法

1. 打开游戏主程序。

2. 选择进入Player VS AI或AI VS AI，再运行你的AI程序即可向游戏连接AI。

3.指挥你的舰队打败敌人吧！

**北航微软技术俱乐部**

By郭哲聪2012.4.14