Ocean Scream AI 编写文档

For C#

1.概述

1. 主程序与 AI 控制程序之间使用 Socket 通信。选手发送了指令到游戏主程序后，会被放在一个请求队列里，然后主程序以 100ms/条指令的速度去处理在队列里的指令。此间 AI 的 Iteration 函数会被继续循环运行。
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. AI 可以自己选择是否更新游戏数据（船只、岛屿的信息，剩余的时间等）。调用 Interface.Data()向游戏发送更新指令，更新完所有游戏数据。Data 指令也要占 100ms 的处理时间。
5. 选手在编写 AI 时可以自行创建新的类，但是自定义类必须处于 MSTCOS.SDK 命名空间下，且必须处于且只处于 AI.cs 文件中。
6. AI 程序不允许用多线程，不允许用第三方库，不允许进行文件读写，不允许进行调用 cmd、修改注册表等操作。
7. 选手最终提交的文件为单独的 AI.cs 文件。
8. 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。
9. 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。AI可以调用SDK中的TimeLeft获取离游戏结束还有多长时间，单位是毫秒，选手可以利用TimeLeft来计时。

2.常量定义

所有常量均带有 public static readonly 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	45
float ShipBoundingRadius	船只碰撞半径	15
float IslandBoundingRadius	岛屿碰撞半径	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	350

float[] ResourceRestoreRate[6]	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	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
bool IsMoving	仅给予己方船只的数据得到船只是否在尝试向前移动
bool 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
bool 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。
bool 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。
int TimeLeft	返回游戏剩余时间的秒数
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 为己方船的编号，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 为己方船的编号，(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。

调用该指令游戏程序会同时执行 StartMoving 和到目标点的 StartRotatingTo. 当船行驶到距目标点 25 时则执行 StopMoving，船开始以 10 个单位每秒的速率减速，直到停止。需要注意的是，由于船不是瞬间停住的，调用 MoveTo 指令移向目标点，最后船实际停靠点并不与目标点重合，而会产生一定误差。误差大小根据减速瞬间的速率决定，假设此时速率为 v0,则完全停止需要实际 t=v0/10，所以从减速到停止船行驶了 0.5*10*t*t，则误差为(0.5*10*t*t-25)。因此若船减速瞬间速率为 25，则整个减速过程行驶了 31.25，停止时船中心与目标点的距离为 6.25.

如果在 MoveTo 的过程中执行了新的旋转类指令船仍然继续移动，但取消在目标点附近自动 StopMoving 的状态。

如果在 MoveTo 的过程中执行了 StartMoving 指令船仍然继续移动，但取消在目标点附近自动 StopMoving 的状态，且保留船之前的旋转类指令。

MoveTo 指令是不精确的指令，只是为了方便对精确性要求不高的操作，要实现更精确的操作请自行计算并调用其他函数。

4.2.3 Data 指令

接口原型：public void Data()

Data指令是更新指令，调用之后会更新Ship、Resource、TimeLeft的数据，也就是说在发送Data指令之前，Ship、Resource、TimeLeft的数据都是过时的。Data指令被程序处理之后，Ship、Resource、TimeLeft才是当前的最新状态。使用Ship、Resource、TimeLeft不会产生新指令，但是使用Data会产生新指令，也就是说消耗掉100ms的指令处理时间。

4.2.4 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，即同时调用 StopRotating()和 StopMoving()
StopMoving(int ship)	减速直至停止，加速度为-10，不是瞬间停止。
StopRotating(int ship)	立即停止旋转。

StartMoving(int ship)	先加速前进，加速度为 10，直到速率为 25 后以该恒定速率前进。调用 MoveTo 后再调用该命令会忽略 MoveTo 的到达目标附近时调用的 StopMoving 命令,但保留 StartRotatingTo 命令。
StartRotatingTo(int ship, float x, float y)	船会以最小角度转动使船头对准目标点
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.C#语言 AI 示例

注：dll 是用 VS2010 编译的。比赛时也是要用 VS2010 编译。

5.1 使用 Visual Studio 2010 IDE

1. 创建一个控制台程序（Console Application），将项目工程命名为 AI。
2. 在项目上点击右键并选择“添加引用”，进入 MSTCOSSDK.dll 所在的目录并选择 MSTCOSSDK.dll。
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示例 AI 代码如下：

   AI.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using MSTCOS.SDK;

namespace MSTCOS.SDK
{
    public class AI : AbstractAI
    {
        public AI(OSInterface Interface)
            : base(Interface)
        {
        }
```

```
        List<ShipInfo> ships, a, b;

        private float dist(ShipInfo a, ShipInfo b)
        {
            return (a.PositionX - b.PositionX) * (a.PositionX - b.PositionX) + (a.PositionY -
b.PositionY) * (a.PositionY - b.PositionY);
        }

        public override void Iteration()
        {
            Interface.Data();   //更新数据
            ships = Interface.Ship;   //获取己方所有船只及可见范围内的敌方船只

            a = new List<ShipInfo>();
            b = new List<ShipInfo>();

            for (i = 0; i < ships.Count; i++)     //筛选敌我船只，我方船只放入a,敌方船只放入b
                if (ships[i].Faction == Interface.Faction) a.Add(ships[i]);
                else b.Add(ships[i]);

            Interface.MoveTo(a[0].ID, 100, 100 );//我方0号战舰向(100,100)前进
            Interface.StartMoving(a[1].ID);//我方1号战舰前进
        }

        public static void Main(String[] args)
        {
            String[] temp = new String[2];
            temp[0] = "buaacs";
            temp[1] = "111.222.33";
            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，战队名只能为英文即ASCII码，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 使用 csc 命令行编译工具

1. 编写好 AI.cs 文件。一个简单的待完善的示例 AI 代码请参考 5.1。

2. 然后，将 MSTCOSSDK.dll 和 AI.cs 放到同一个目录下，请确保 csc 所在的目录处于 Path 环境变量中，一般为 C:\Windows\Microsoft.Net\Framework\版本号，并使用下面的指令编译 AI 客户端：
   csc /r:MSTCOSSDK.dll /out:AI.exe AI.cs
3. 如果一切正常的话会在当前目录下生成 AI.exe 文件。

5.3 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.exe 即可向游戏连接 AI，注意 AI.exe 所在目录必须有 MSTCOSSDK.dll。
3.指挥你的舰队打败敌人吧！

北航微软技术俱乐部