URWPGSim2D 开发人员手册

V 1.2 Revised 20120101

北京大学智能控制实验室　**2012** 年 **1** 月

目　　录

1. 概述

本手册最新版本请从中国水中机器人大赛官方网站（http://robot.pku.edu.cn）仿真组资源下载页面获取。

1.1. 读者对象

本手册为可能参与 URWPGSim2D 软件设计、开发、测试和维护的人员和可能参与中国水中机器人大赛 2D 仿真比赛策略编写的人员编制，URWPGSim2D 涉及的所有技术细节会尽最大可能记录于此。

1.2. 名词解释

1. URWPGSim2D：Underwater Robot Water Polo Game Simulator 2 Dimension Edition，水中机器人水球比赛仿真器 2D 版本。
2. MRDS：Microsoft Robotics Developer Studio，微软机器人开发套件。
3. CCR：Concurrency and Coordination Runtime，并发协调运行时，MRDS 用于解决机器人软件开发中并发问题的技术和基础软件库。
4. DSS：Decentralized Software Services，分布式软件服务，MRDS 用于解决机器人软件开发中异步问题的技术和基础软件库。
5. Simulation Mission：Mission，仿真使命，即仿真比赛或实验项目，模拟机器鱼比赛或实验项目的对象。
6. Simulation Environment：SimEnvironment，仿真环境，仿真使命运行所处的虚拟环境，其中包括仿真场地（模拟比赛或实验用水池的对象）、零个或多个仿真水球（模拟比赛或实验用水球的对象）、零个或多个仿真方形障碍物（模拟比赛或实验用方形障碍物的对象）、零个或多个仿真圆形障碍物（模拟比赛或实验用圆形障碍物的对象）。
7. Simulation RoboFish：RoboFish，仿真机器鱼，模拟比赛或实验用机器鱼的对象。
8. Simulation Loop：仿真循环，仿真使命运行过程中所有仿真动作顺序执行一遍的过程。
9. Simulation Cycle：仿真周期，理论上考虑为比单个仿真循环所耗时间（与运行软硬件环境有关，不能精确确定，在相同环境下，每次运行也不精确相同）预估最大值稍大的确定时间间隔（如 100 毫秒）。当前（2011325）平台仿真循环在推荐的软硬件配置（见2开发环境）下所耗时间大约在 10-20 毫秒之间，为简化线程同步，在仿真使命的公共参数类中设置了一个成员MsPerCycle，保存一个初始化仿真使命时传入的整数值，称为“每周期毫秒数”，实际运行时的仿真周期值不确定，为MsPerCycle值加上当前周期仿真循环所耗时间，通常MsPerCycle都取 100 毫秒，仿真周期值大约在 110-120 毫秒之间。仿真使命倒计时以根据MsPerCycle计算出来的总周期数递减的方式进行，故界面上显示的倒计时牌并不是按精确的世界时间递减。
10. Simulation Action：仿真动作，包括将策略计算出来的决策命令拷贝到每支队伍每条仿真机器鱼对象本身的决策字段（后续计算都是直接使用仿真机器鱼对象自身的决策命

令)、对所有动态对象（目前包括每支队伍的所有仿真机器鱼、仿真环境中仿真水球列表的所有仿真水球）进行运动学计算、对所有对象（包括每支队伍的所有仿真机器鱼、仿真环境中所有对象）相互进行碰撞处理（包括碰撞检测和碰撞响应）。

2. 开发环境

2.1. 硬件环境

URWPGSim2D在PC机或工作站上进行开发，其硬件配置要求如表 2-1所示。

表 2-1 URWPGSim2D 运行硬件配置表

核心配件	最低配置	推荐配置
CPU	Intel P4 2.0GHz 或同档次 AMD CPU	Intel E7300 2.66GHz 或以上
内存	256MB	2GB 或以上
显卡	支持 DirectX 9.0，Pixel Shader 3.0，显存 128M 或以上	
硬盘	10GB	80GB 或以上

2.2. 软件环境

操作系统：Windows XP Professional SP3，Windows Vista 或 Windows 7。

.Net 框架：.Net Framework 3.5 with SP1。

IDE：Microsoft Visual Studio Team System 2008 Team Suite with SP1，或 Microsoft Visual Studio 2008 Professional with SP1。

编程语言：C#（CSharp） V3.0。

MRDS：Microsoft Robotics Developer Studio 2008 R3。

附件：Microsoft XNA Framework Redistributable 3.1，Microsoft Excel 2003 Com Library。

2.3. 搭建标准开发环境

搭建开发环境所需软件，在中国水中机器人大赛官方网站仿真组资源下载页面部分提供本地下载，没有本地下载的则提供官方下载链接。

1. PC 机或工作站安装 Windows XP Professional SP3 操作系统。
2. 按照默认设置安装DotNet3.5SP1（该软件包集成了SP1，且安装时不需要联网，官方网站提供的安装包安装时需要联网）和XNA3.1。
3. 按照至少保留C#开发相关组件的要求安装（建议除SQL Server数据库外的部分完全安装）Microsoft Visual Studio Team System 2008 Team Suite中文版with SP1并破解。

4. 按照默认设置安装TortoiseSVN1.6.5和VS2008 的VisualSVN插件并破解。

2.4. 开始开发

从中国水中机器人大赛官方网站仿真组资源下载页面获取最新的URWPGSim2D源代码SVN地址及相应的用户名和密码，打开VS2008 的VisualSVN菜单选择“Get Solution from Subversion”，输入得到的SVN地址，弹出用户名和密码输入框时输入相应的值即可获得最新版本源代码。

为方便使用 Visual Studio 的快捷键进行调试，需要先将 ConductorSvr 项目设为启动项目。操作方法是，在解决方案浏览器中，右键点击 ConductorSvr 项目名称，选择“设为启动项目”。

2.5. 补充说明

MRDS 不需要安装，URWPGSim2D 所要用到的由 MRDS 提供的全部程序集（即 DLL 文件）已经包含在源代码包的 URWPGSim2D\bin 目录里。

3. 整体设计

3.1. 功能设计

URWPGSim2D 的主要作用是作为水中机器人竞赛平台和水中机器人科研平台，要求方便扩展竞赛和实验项目，方便独立编写竞赛和实验项目的策略算法。

URWPGSim2D 包括服务端（URWPGSim2DServer）和客户端（URWPGSim2DClient）两大部分。服务端模拟水中环境，控制和呈现仿真过程及结果，向客户端发送实时仿真环境和过程信息；半分布式客户端模拟水中机器人队伍，全分布式客户端模拟单个水中机器人，加载比赛或实验策略，完成决策计算过程，向服务端发送决策结果。

3.2. VS2008 Solution & Project结构

URWPGSim2D软件由一个名为URWPGSim2D.sln的VS2008 C# Solution生成，该Solution包含至少 9 个（可扩展到更多）Project，由依赖关系决定的生成次序如表 3-1所示。所有Project均使用名为URWPGSim2D.snk的密钥文件进行签名以生成具有强名称的程序集。

URWPGSim2D1.0.11.316 包括 9 个自有组件和 MRDS 及其他第三方组件。

表 3-1 URWPGSim2D 自有组件表

组件名称	对应的文件	描述	相应 Project	依赖的组件
Core	URWPGSim2D.Core.dll	运动学和动力学计算模块；随机扰动模块；碰撞响应模块	Core	无

Common	URWPGSim2D.Common.dll、config.xml	仿真机器鱼模型；仿真环境（场地/水球/障碍物/通道等）模型；仿真使命模型；碰撞检测模块；系统配置模块；辅助函数模块	Common	Core
Match	URWPGSim2D.Match.dll	仿真使命具体实现模块（添加比赛项目的地方）	Match	Core 、Common
StrategyLoader	URWPGSim2D.StrategyLoader.dll	实现策略动态加载的辅助模块	StrategyLoader	Common
Gadget	URWPGSim2D.Gadget.dll	绘制轨迹，显示实时信息等辅助功能模块	Gadget	无
Sim2DSvr	Sim2DSvr.manifest.xml、URWPGSim2D.Sim2DSvr.Y2010.M11.dll 、URWPGSim2D.Sim2DSvr.Y2010.M11.Proxy.dll 、URWPGSim2D.Sim2DSvr.Y2010.M11.Transform.dll	服务端 DSS 服务和界面模块	Sim2DSvr	Common、Match 、StrategyLoader、Gadget
Sim2DClt	Sim2DClt.manifest.xml 、URWPGSim2D.Sim2DClt.Y2010.M11.dll 、URWPGSim2D.Sim2DClt.Y2010.M11.Proxy.dll 、URWPGSim2D.Sim2DClt.Y2010.M11.Transform.dll	客户端 DSS 服务和界面模块	Sim2DClt	Common、Sim2DSvr、StrategyLoader
ConductorSvr	URWPGSim2DServer.exe	服务端启动引导程序	ConductorSvr	Sim2DSvr
ConductorClt	URWPGSim2DClient.exe	客户端启动引导程序	ConductorClt	Sim2DClt

URWPGSim2D Solution 目录（解决方案目录，Solution Directory）名称任意，默认为 URWPGSim2D（从 SVN 服务器上下载时，可以取任意名称）。Solution 中包含 9 个 Project，所有 Project 目录平行位于 Solution 目录下。Solution 目录中除各 Project 目录外，还有两个目录 URWPGSim2D 和 Strategy，分别为输出目录和策略目录。

URWPGSim2D 软件在部署方面的设计目标是全绿色，不写注册表，不往系统目录拷贝文件。所有要用到的第三方组件，尽量全部置于程序运行目录下，实现即拷即用。输出目录 URWPGSim2D 下设一级子目录 bin 作为程序运行目录。bin 目录名称不能改变，这与 MRDS

DSS 服务的运行机制有关，DSS 服务组件都是生成为 dll 文件的形式，默认的部署方法是部署在 MRDS 的安装目录下的 bin 子目录中，通过 bin 目录下的 DssHost.exe 启动，启动时 DssHost.exe 会自动在 bin 目录的上级目录中寻找名为 store 的目录，以查询服务相关的 Cache 信息，没找到则自动新建该目录。所以输出目录 URWPGSim2D 下至少有个 bin 子目录。

所有 Project 的输出目录均设置为“..\URWPGSim2D\bin”，全部第三方组件也置于该目录下。所有 Project 引用的程序集（这里全为单个 dll 文件）的 Copy Local 属性设为 False。于是各 Project 引用的程序集的路径目录部分就全部固定为相对目录“..\URWPGSim2D\bin”，无论如何生成新版本，引用都能得到及时更新。

策略目录 Strategy 是 Strategy Solution 所在目录。Strategy Solution 提供了大部分仿真使命的策略模板。编写仿真比赛或自编实验项目策略，请直接在 Strategy Solution 中使用已有项目的策略 Project 或添加新项目的策略 Project。

3.3. 整体结构

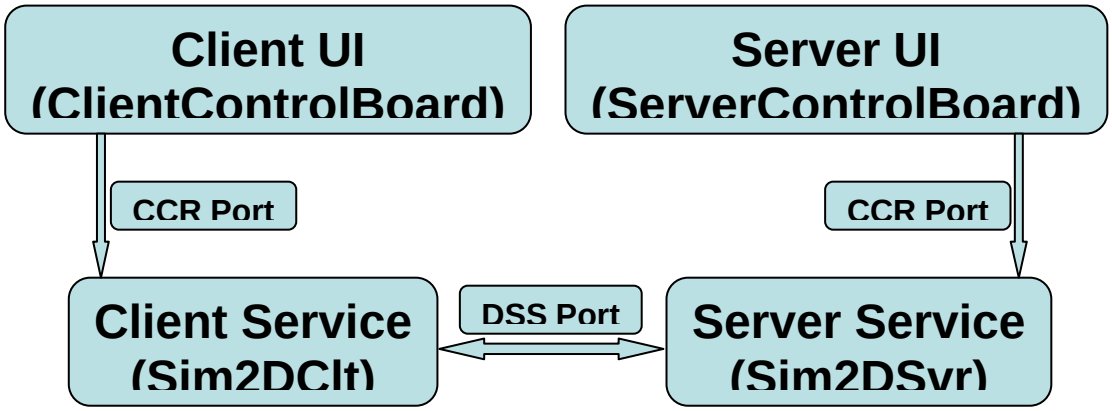

图 3-1 URWPGSim2D 整体结构图

3.4. 服务端结构

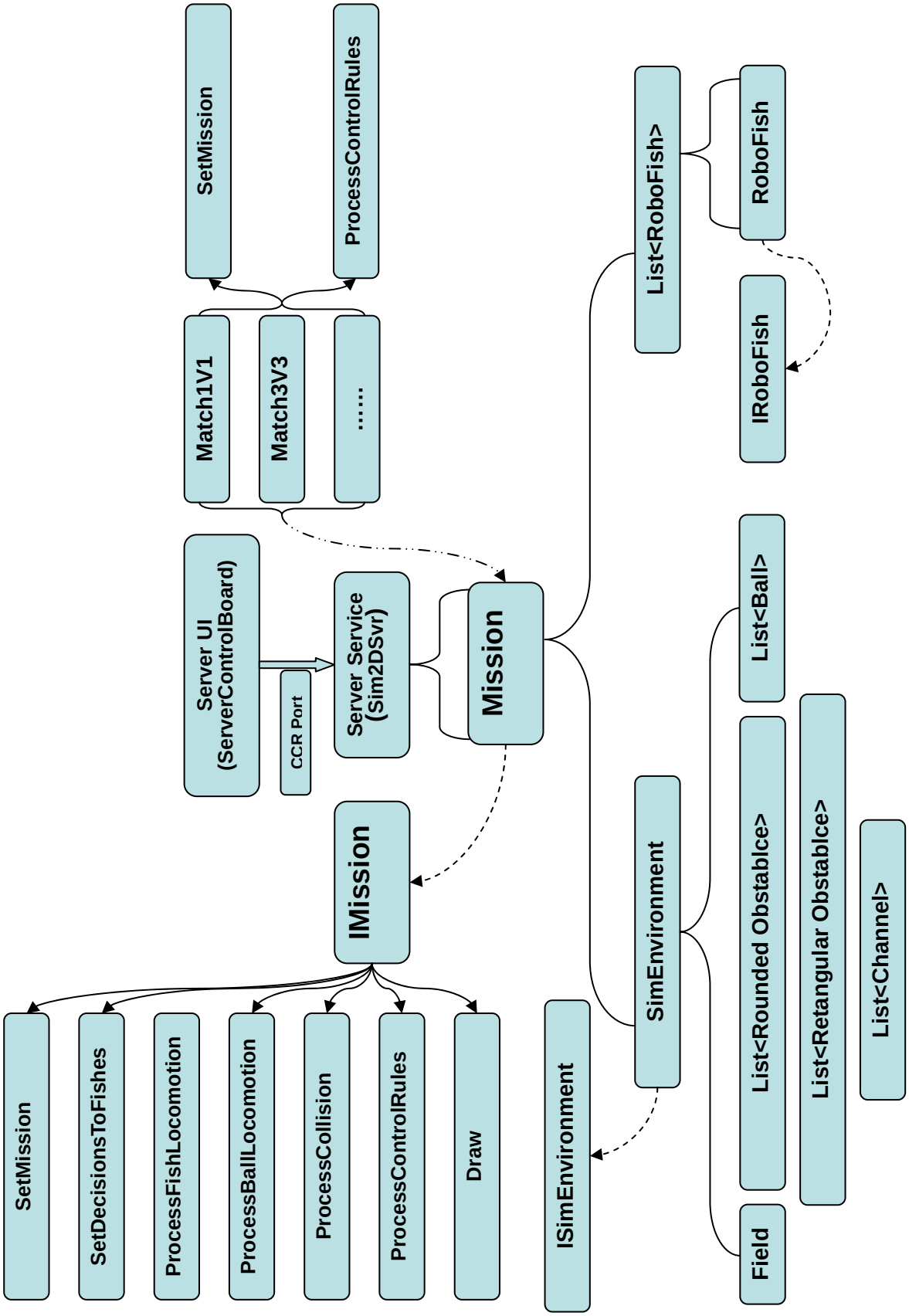

图 3-2 URWPGSim2D 服务端模型结构图

3.4.1. 以仿真使命为中心

URWPGSim2D以仿真使命为中心的模型结构如图 3-2所示。

URWPGSim2D 的设计采用面向对象思想。从对象建模的角度看，包括仿真机器鱼、仿真环境和仿真使命（比赛或实验项目）三类模型，以仿真使命模型为中心。仿真使命包括仿真机器鱼队伍列表和仿真环境。三类模型各包括一个顶层模型分别为仿真机器鱼基类 RoboFish、仿真环境基类 SimEnvironment 和仿真使命基类 Mission。类 RoboFish 定义了各种具体仿真使命所需仿真机器鱼的公共特性（用类的属性表示）和公共行为（用类的方法表示）。类 SimEnvironment 定义了各种具体仿真使命所需仿真环境的公共特性（仿真环境不需要定义行为）。类 Mission 定义了各种具体仿真使命的公共特性和公共行为。

仿真机器鱼和仿真环境的公共行为很少或不需要定义，而仿真使命的公共行为则很多。故基类 RoboFish 和 SimEnvironment 虽然从形式上实现了 IRoboFish 和 ISimEnvironment 接口，但这两个接口没有定义任何方法；而基类 Mission 实现了 IMission 接口，该接口定义了诸多方法。部分方法在基类 Mission 中提供具体实现，另一部分方法则在基类 Mission 中只提供虚函数实现，由具体仿真使命类提供重载（override）的具体实现。

定义了一个泛型类 Team<TFish>用于将仿真机器鱼组成队伍。该类有一个 TFish 类型的列表（List<TFish>）成员 Fishes，保存一支队伍的全部仿真机器鱼对象；有一个队伍公共参数（TeamCommonPara）类型的成员 Para，保存一支队伍的各项特性参数如队伍名称、仿真机器鱼数量、当前得分等。

仿真环境包括仿真场地、仿真水球、仿真障碍物和仿真通道等元素。仿真环境基类 SimEnvironment 有一个仿真场地 Field 类型的成员 FieldInfo，保存仿真场地对象；有一个仿真水球 Ball 类型的列表（List<Ball>）成员 Balls，保存当前仿真环境中的全部仿真水球对象；有一个仿真圆形障碍物 RoundedObstacle 类型的列表（List<RoundedObstacle>）成员 ObstaclesRound，保存当前仿真环境中的全部仿真圆形障碍物对象；有一个仿真方形障碍物 RectangularObstacle 类型的列表（List<RectangularObstacle>）成员 ObstaclesRect，保存当前仿真环境中的全部仿真方形障碍物对象；有一个仿真通道 Channel 类型的列表（List<Channel>）成员 Channels，保存当前仿真环境中的全部仿真通道对象（20110709 注：考虑仿真通道没有实际使用过，要用也可以由两个平行摆放的仿真矩形障碍物构成，因此废弃掉仿真通道的概念）。

仿真使命基类 Mission 有一个仿真机器鱼基类 RoboFish 组成的队伍列表（List<Team<RoboFish>>）成员 TeamsRef；有一个仿真环境基类 SimEnvironment 类型的成员 EnvRef；有一个仿真使命公共参数 MissionCommonPara 类型的成员 CommonPara。

具体仿真使命类（如 3VS3 比赛项目的仿真使命类 Match3V3）继承基类 Mission，相应的具体仿真机器鱼类（如 Fish3V3）继承基类 RoboFish，具体仿真环境类（如 Environment3V3）继承基类 SimEnvironment。

3.4.2. 以仿真循环为主线

仿真使命启动运行后，仿真循环将周期性地持续进行，直到设定的仿真时间耗完、人为/程序决定暂停/停止。

仿真循环将所有仿真动作执行一遍，包括如下步骤。

1. 分配决策值 SetDecisionsToFishes。
2. 对仿真机器鱼运动学参数进行计算 ProcessFishLocomotion。
3. 对仿真水球运动学参数进行计算 ProcessBallLocomotion。
4. 处理场上所有对象间的碰撞包括检测和响应碰撞 ProcessCollision。
5. 处理当前仿真使命特有的规则如比赛犯规计分等 ProcessControlRules。
6. 处理策略调用，区分本地调用和远程调用。
7. 处理界面动态数据更新。

3.5. 版本控制

3.5.1. 源码管理

URWPGSim2D 开发过程采用 SVN 进行源代码版本控制。SVN 服务器使用 VisualSVN-Server-2.0.7，SVN 客户端使用 TortoiseSVN-1.6.5（TortoiseSVN-1.6.5.16974-win32-svn-1.6.5），配合 Visual Studio 2008 的插件 VisualSVN-1.7.6 破解版。

3.5.2. 产品管理

URWPGSim2D 产品版本号采用四段版本号“Majo.Minor.[Revision[.Build]]”。Major 为主版本号，从 1 开始，遇上功能巨大变动时才增加，一般每次增 1；Minor 为次版本号，从 0 开始，遇上功能较大变化时增加，一般每次增 1；Revision 为修订号，取 SVN 服务器中源代码库 Revision 号；Build 为 6 位生成日期，由年月日按照 YMMDD 方式构成。如 1.1.103.10713 表示 11 年 7 月 13 日使用 Revision 号为 103 的源代码版本生成，尚未进行很大的功能扩充，进行过 1 次较大功能变化。

4. 策略编写

4.1. 策略是什么

内容上看，策略是用于控制参与当前仿真使命的仿真机器鱼完成任务的算法代码。形式上看，策略是可在服务端和客户端加载的 dll 文件。

4.2. 策略调用方式

4.2.1. 本地模式

服务端 URWPGSim2DServer 界面上“Referee→Strategy”选择“Local”。本地模式下，仿真使命所有参与队伍的策略都在服务端加载，和 URWPGSim2DServer.exe 运行于同一进程空间，由于仿真循环也运行在 URWPGSim2DServer.exe 进程空间中，所有策略和仿真循环运行于同一进程空间。于是所有队伍的所有仿真机器鱼控制指令及仿真循环运行指令均来自同一进程，因此本地模式是集中式仿真模式。

4.2.2. 远程模式

服务端 URWPGSim2DServer 界面上“Referee→Strategy”选择“Remote”。远程模式下，必须启动当前仿真使命参与队伍数量同样多的 URWPGSim2DClient.exe 进程，用于加载各支队伍的策略。这些进程可以与服务端进程处于同一台电脑，也可以分别位于不同电脑，只要这些电脑处于同一局域网（理论上可以运行于广域网上，尚未测试）内。每支队伍的策略，和加载它的 URWPGSim2DClient.exe 进程运行于同一进程空间。于是每支队伍的所有仿真机器鱼控制指令独立于仿真循环运行指令，各支队伍的仿真机器鱼控制指令也互相独立，但队伍内部各仿真机器鱼的控制指令则来自同一进程。因此远程模式是半分布式仿真模式。

若 URWPGSim2DClient.exe 进程与 URWPGSim2DServer.exe 进程处于不同电脑，则运行 URWPGSim2DClient.exe 进程电脑上的../URWPGSim2D/bin/**Sim2DClt.manifest.xml** 文件中的<dssp:Service>http://**localhost**:50000/Sim2DSvr</dssp:Service>配置节，localhost 需要修改成 URWPGSim2DServer.exe 进程所在电脑的 IP 或主机名。

4.2.3. 异步调用

无论是本地模式还是远程模式，都是采用异步方式。

本地模式，调用代码在 Sim2DSvrService.NextStepProcessDetail 方法中，服务端 URWPGSim2DServer.exe 进程通过为每支队伍启动一个 Arbiter.Receiver，异步调度 CCR 线程池里的线程来为每支队伍分别执行 GetLocalDecision 运行决策算法获得决策值。GetLocalDecision 与仿真循环是异步关系，执行结果是将获得的决策数组填充在一片公共空间 DecisionRef 中，而仿真循环则通过 SetDecisionsToFishes 方法从 DecisionRef 将最近获得的决策值分配给相应队伍的仿真机器鱼。对于 2 支及 2 支以上队伍参与的仿真使命，每个仿真周期各支队伍的 GetLocalDecision 执行顺序并不确定。

远程模式，涉及比较复杂的 CCR 和 DSS 通信过程，调用入口代码在 Sim2DSvrService.NextStepProcessDetail 方法中，服务端 URWPGSim2DServer.exe 进程通过 SpawnIterator 异步调度 Sim2DSvrService.MissionParaNotification 方法执行，向所有客户端通知当前仿真使命的 Mission 对象值（全部仿真环境和过程信息均在此对象中）。客户端 URWPGSim2DClient.exe 进程的 Sim2DCltService 服务实例收到 MissionParaNotification 后，通过 SpawnIterator 异步调度 Sim2DCltService.AnnounceDecisionToServer 方法执行，获取决策数组以 ClientAnnounceDecision 类型的消息发送给服务端。服务端 URWPGSim2DServer.exe 进程的 Sim2DSvrService 服务实例收到 ClientAnnounceDecision 后，CCR 调度器异步调度线程池里的线程来执行 ClientAnnounceDecisionHandler，将收到的决策数组填充到公共空间 DecisionRef 中，供仿真循环通过 SetDecisionsToFishes 分配给相应队伍的仿真机器鱼。前述过程还不包括客户端启动时，主动连接服务端，请求和服务端建立 Subscribe/Notify 即订阅/通知关系的过程。关于 URWPGSim2D 涉及的通信细节，另有介绍。

4.3. 编写指南

以下描述中%URWPGSim2D%为 URWPGSim2D Solution Directory，如 D:\My Documents\Visual Studio 2008\Projects\URWPGSim2D。

4.3.1. 编程相关

编写策略，首先需要建立基于**.Net Framework 3.5** 使用 **C#**编程语言的 **Windows 类库**

（Class Library）类型的 **VS2008** 项目（Project）及相应的解决方案（Solution）。

项目的命名空间必须为**URWPGSim2D.Strategy**；项目中必须有一个类名称为**Strategy**；Strategy类必须继承自**MarshalByRefObject类**并实现**IStrategy接口**，为**GetTeamName**和**GetDecision**方法提供具体实现，接口的具体信息参见第7条推荐的程序结构；项目的Reference至少添加Microsoft.Dss.Base.dll、Microsoft.Xna.Framework.dll、URWPGSim2D.Common.dll、URWPGSim2D.StrategyLoader四个。

1. 可以直接使用%URWPGSim2D%\Strategy\下的 Strategy Solution 即 Strategy.sln，然后在 Strategy Solution 中添加策略 Project，或直接使用现有 Strategy 模板 Project。

2. 也可新建策略 Project（如 Strategy3VS3）的同时新建 Solution，并让 Project 在 Solution 的下级目录中，Solution 目录可以放在任意位置，Solution 名称可以任意，推荐使用 Strategy。

3. 修改项目属性（Project Properties）等。有了 Strategy Project，譬如为编写 Match3VS3 仿真使命而建立的 Strategy Project 名为 Strategy3VS3，还需要修改部分项目属性。在解决方案浏览器（Solution Browser）中，选择 Strategy Project 如 Strategy3VS3，右键选择属性（Properties）。在弹出的页面中，应用程序（Application）→默认命名空间（Default namespace）选项修改为“URWPGSim2D.Strategy”。

4. 修改类名（Class name）等。默认建立的 cs 文件（如 Class1.cs）重命名为 StrategyProjectName.cs（如 Strategy3VS3.cs）。再打开该文件，将其中的 namespace 名称（如 ClassLibrary1）修改为 URWPGSim2D.Strategy；class 名称（如 Class1）修改为 Strategy : MarshalByRefObject, IStrategy。

5. 类 Strategy 中必须重载 MarshalByRefObject 的 InitializeLifetimeService 方法，其实现只有一条语句 return null，以使得 Strategy 的对象在主应用程序域以外被加载时不会产生超时问题。不照此实现，使命运行过程中，如果发生暂停（比如调试时断点中断或正式运行时弹出对话框）超过一定时间，策略对象会被自动销毁；再次调用策略时，策略已经不存在。（屏蔽掉 try…catch 块，在调试时下断点等上五分钟左右继续运行，会弹出类似“对象’05e33_483c_4a09_80ea_1ac761a571b6/pcbpomcu+nihdarschxpvuhm_17.rem’经断开连接或不在服务器上”的 RemotingException 提示。）

6. 类 Strategy 的实例在加载完毕后一直存在于内存中，除非仿真使命运行过程中更换策略，因此，可以自定义私有成员变量保存必要信息。

7. 推荐的程序结构如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using xna = Microsoft.Xna.Framework;
using URWPGSim2D.Common;
using URWPGSim2D.StrategyLoader;
```

```
namespace URWPGSim2D.Strategy
{
    public class Strategy : MarshalByRefObject, IStrategy
    {
        #region reserved code never be changed or removed
        /// <summary>
        /// override the InitializeLifetimeService to return null instead of a valid ILease implementation
        /// to ensure this type of remote object never dies
        /// </summary>
        /// <returns>null</returns>
        public override object InitializeLifetimeService()
        {
            return null; // makes the object live indefinitely
        }
        #endregion

        /// <summary>
        /// 决策类当前对象对应的仿真使命参与队伍的决策数组引用 第一次调用GetDecision时分配空间
        /// </summary>
        private Decision[] decisions = null;

        /// <summary>
        /// 获取队伍名称 在此处设置参赛队伍的名称
        /// </summary>
        /// <returns>队伍名称字符串</returns>
        public string GetTeamName()
        {
            return "3VS3 Test Team";
        }

        /// <summary>
        /// 获取当前仿真使命（比赛项目）当前队伍所有仿真机器鱼的决策数据构成的数组
        /// </summary>
        /// <param name="mission">服务端当前运行着的仿真使命 Mission 对象</param>
        /// <param name="teamId">当前队伍在服务端运行着的仿真使命中所处的编号
        /// 用于作为索引访问 Mission 对象的 TeamsRef 队伍列表中代表当前队伍的元素</param>
        /// <returns>当前队伍所有仿真机器鱼的决策数据构成的 Decision 数组对象</returns>
        public Decision[] GetDecision(Mission mission, int teamId)
        {
            // 决策类当前对象第一次调用 GetDecision 时 Decision 数组引用为 null
            if (decisions == null)
            {// 根据决策类当前对象对应的仿真使命参与队伍仿真机器鱼的数量分配决策数组空间
                decisions = new Decision[mission.CommonPara.FishCntPerTeam];
            }
```

```
            #region 决策计算过程 需要各参赛队伍实现的部分
            #endregion
            // 请从这里开始编写代码

            #endregion

            return decisions;
        }
    }
}
```

8. GetTeamName 接口用于**设置队伍名称**。将其中的“3VS3 Test Team”（不含引号）替换成队伍名称即可。

9. GetDecision 接口用于**生成 Strategy 对象对应的队伍里所有仿真机器鱼的决策数据**。决策过程较复杂的情况下，建议将各种子过程封装成 Strategy 类的 Private 方法，由 GetDecision 或子过程调用。

10. 由于 Strategy 类引用了 URWPGSim2D.Common.dll、URWPGSim2D.StrategyLoader 这两个组件，它们均具有强名称（使用%URWPGSim2D%\URWPGSim2D.snk 进行了签名，强名称相关介绍详见 MSDN），因此 URWPGSim2D 主程序（包括前述两个组件）版本号更新后，策略 dll 文件，**必须对相应的 Strategy 类重新添加引用，重新生成**，才能正常加载。

4.3.2. 业务相关

1. 策略编写工作直接目标是给当前队伍决策数组 decisions 各元素填充决策值，即给 Strategy 类当前对象对应的仿真使命参与队伍各仿真机器鱼生成控制指令，仿真机器鱼的**控制指令**是一个 **Decision 类型**的值。

2. 决策数据类型 Decision 包括两个 int 成员，**VCode** 为**速度档位**值，**TCode** 为**转弯档位**值。

3. VCode 取值范围 **0-14** 共 15 个整数值，每个整数对应一个速度值，速度值整体但非严格递增。有个别档位值对应的速度值低于比它小的档位值对应的速度值，速度值数据来源于实验。

4. TCode 取值范围 **0-14** 共 15 个整数值，每个整数对应一个角速度值。整数 **7 对应直游**，角速度值为 0，整数 **6-0**，**8-14** 分别对应**左转**和**右转**，偏离 7 越远，角速度值越大。

5. 任意两个速度/转弯档位之间切换，都需要若干个仿真周期，才能达到稳态速度/角速度值。目前运动学计算过程决定稳态速度/角速度值接近但小于目标档位对应的速度/角速度值。

6. **禁止使用原地打转功能**，即不能让仿真机器鱼速度为 0 而角速度不为 0；平台做了限制，速度值低于 1 档的稳态速度时，角速度被强制置 0；所以不要试图把速度降到 0，然后只给角速度进行转弯，这样做的结果是仿真机器鱼会原地不动。

7. 场地坐标系及点和向量定义。以矩形场地中心为坐标原点，**正右为正 X 轴，正下为正**

Z 轴；从正 X 到负 X 轴，顺时针为 0~π，逆时针为 0~-π。考虑与 MRDS 中的三维坐标系一致，水平面用 XOZ 表示，Y 轴作为第三维。程序中涉及向量和点的定义，都使用 XNA 库中的 Vector3 类型，用到其中的 X 和 Z 维，Y 维均置为 0。二维点和向量与三位点和向量之间的转换，二维的 X 与三维的 X 对应，二维的 Y 与三维的 Z 对应。

8. 对抗性仿真使命的策略，**一定要判断己方所处半场，根据所处半场确定目标球门**。根据目标球门设计仿真机器鱼的行为及相关决策算法。

9. GetDecision 接口的 int 类型的参数 teamId，表示当前 Strategy 对象对应的队伍在服务端运行着的仿真使命所有参与队伍中的编号（从 0 开始）。teamId 的初始值，Local 模式时，由策略加载顺序确定；Remote 模式时，由客户端启动顺序确定。2 支队伍参与的对抗性仿真使命，交换半场后，teamId 由服务端或客户端策略调用入口处相应半场处理代码进行值的交换，传到 GetDecision 的 teamId 已经正确代表当前时刻当前 Strategy 对象对应的队伍编号。

10. GetDecision 接口的 Mission 类型的参数 mission，包含服务端当前运行着的仿真使命公开给策略使用的全部信息。mission.TeamsRef[teamId]指向当前 Strategy 对象对应的仿真使命参与队伍 Team<RoboFish>对象，通过它可以访问到己方队伍及其全部仿真机器鱼的所有公开信息。当前运行着的仿真使命为 2 支队伍参与的对抗性项目时，mission.TeamsRef[(1 + teamId) % 2]指向对方队伍的 Team<RoboFish>对象，通过它可以访问到对方队伍及其全部仿真机器鱼的所有公开信息。

11. 表 4-1是策略中可以使用的全部参数，所有成员的引用，均以mission.开头，如mission.CommonPara.TeamCount；其中“CommonPara.*”一类的成员表示随后若干右对齐的成员名称，是以CommonPara.*的形式存在的，如CommonPara.MsPerCycle，其完整引用为mission.CommonPara.MsPerCycle。MissionCommonPara、Team<RoboFish>、TeamCommonPara、RoboFish、Field、RetangularObstacle、RoundedObstacle等类型还有其他成员，不向策略公开，那是只与服务端处理有关的变量，不要试图使用它们，因为服务端向客户端或本地策略传递Mission类型参数的过程包含一个序列化和反序列化的过程，这些不向策略公开的参数，相应的成员没有序列化和反序列化代码支持。

表 4-1 策略中可以使用的参数表

成员名称	类型	意义
CommonPara	MissionCommonPara	当前仿真使命公共参数
CommonPara.*		
FishCntPerTeam	int	每支队伍仿真机器鱼数量
MsPerCycle	int	仿真周期毫秒数
RemainingCycles	int	当前剩余仿真周期数
TeamCount	int	当前仿真使命参与队伍数量
TotalSeconds	int	当前仿真使命运行时间秒数
TeamsRef	List<Team<RoboFish>>	当前仿真使命参与队伍列表
TeamsRef[teamId]	Team<RoboFish>	决策类当前对象对应的仿真使命参与队伍（当前队伍）
TeamsRef[teamId]		

成员名称	类型	意义
.*		
Para	TeamCommonPara	当前队伍公共参数
Fishes	List<RoboFish>	当前队伍仿真机器鱼列表
TeamsRef[teamId].Para.*		
FishCount	int	当前队伍仿真机器鱼数量 （与 CommonPara.FishCntPerTeam 的值一致）
MyHalfCourt	HalfCourt（enum）	当前队伍当前所处半场 （HalfCourt.LEFT(0) / Halft.RIGHT(1)）
Score	int	当前队伍当前得分值
Name	string	当前队伍名称
TeamsRef[teamId].Fishes[i].*		
PositionMm	xna.Vector3	当前队伍第 i 条鱼位置（鱼体前端刚体中心）
BodyDirectionRad	float	当前队伍第 i 条鱼鱼体方向（鱼体前端刚体长边方向）
VelocityMmPs	float	当前队伍第 i 条鱼速度值
VelocityDirectionRad	float	当前队伍第 i 条鱼速度方向 （与 BodyDirectionRad 的值一致）
AngularVelocityRadPs	float	当前队伍第 i 条鱼角速度值
BodyLength	int	当前队伍第 i 条鱼鱼体前端刚体长度
BodyWidth	int	当前队伍第 i 条鱼鱼体前端刚体宽度
CollisionModelRadiusMm	int	当前队伍第i条鱼碰撞检测外层圆形模型半径即总长度的一半
CollisionModelBodyRadiusMm	int	当前队伍第 i 条鱼前端刚体矩形外接圆半径
CollisionModelTailRadiusMm	int	当前队伍第 i 条鱼尾部碰撞检测圆形模型半径
PolygonVertices[0]	xna.Vector3	当前队伍第 i 条鱼鱼头位置
EnvRef	SimEnvironment	当前仿真使命环境对象
EnvRef.*		
FeildInfo	Field	当前仿真使命的仿真场地对象
Balls	List<Ball>	当前仿真使命的全部仿真水球列表
Balls[i]	Ball	当前仿真使命第 i 个仿真水球对象
ObstaclesRect	List<RetangularObstacle>	当前仿真使命的全部仿真方形障碍物列表
ObstaclesRect[i]	RetangularObstacle	当前仿真使命第 i 个仿真方形障碍物对象
ObstaclesRound	List<RoundedObstacle>	当前仿真使命的全部仿真圆形障碍物列表

成员名称	类型	意义
ObstaclesRound[i]	RoundedObstacle	当前仿真使命第 i 个仿真圆形障碍物对象
EnvRef.FieldInfo.*		
FieldLengthXMm	int	当前仿真场地 X 方向长度
FieldLengthZMm	int	当前仿真场地 Z 方向长度
GoalDepthMm	int	当前仿真场地球门深度（X 方向长度）
GoalWidthMm	int	当前仿真场地球门宽度（Z 方向长度）
ForbiddenZoneLengthXMm	int	当前仿真场地禁区 X 方向长度
ForbiddenZoneLengthZMm	int	当前仿真场地禁区 Z 方向长度
LeftMm	int	当前仿真场地左边界 X 坐标
RightMm	int	当前仿真场地右边界 X 坐标
TopMm	int	当前仿真场地上边界 Z 坐标
BottomMm	int	当前仿真场地下边界 Z 坐标
EnvRef.Balls[i].*		
PositionMm	xna.Vector3	当前仿真使命第 i 个水球位置
RadiusMm	int	当前仿真使命第 i 个水球半径
VelocityMmPs	float	当前仿真使命第 i 个水球速度值
VelocityDirectionRad	float	当前仿真使命第 i 个水球速度方向
EnvRef.ObstaclesRect[i].*		
PositionMm	xna.Vector3	当前仿真使命第 i 个方形障碍物位置
LengthMm	int	当前仿真使命第 i 个方形障碍物长度
WidthMm	int	当前仿真使命第 i 个方形障碍物宽度
DirectionRad	float	当前仿真使命第 i 个方形障碍物方向（长度朝向）
EnvRef.ObstaclesRound[i].*		
PositionMm	xna.Vector3	当前仿真使命第 i 个圆形障碍物位置
RadiusMm	int	当前仿真使命第 i 个圆形障碍物半径
HtMissionVariables	Hashtable	当前仿真使命需要传递给策略的特有参数哈希表 表中键值对，键字符串为变量名，值字符串为变量值

12. 具体仿真使命特有的参数（如水球 3VS3 等对抗性项目中指示当前比赛阶段值的标志量 CompetitionPeriod）可以通过 HtMissionVariables["键名"]得到，然后可通过 Convert.To*** 转换成其值的原始数据类型使用。如水球 3VS3 项目的当前比赛阶段值通过如下方式获得：int matchPeriod = Convert.ToInt32(mission.HtMissionVariables["CompetitionPeriod"]);。每个具体仿真使命需要传递给策略的特有参数由仿真使命设计人员确定，并在仿真使命

规则文档中描述其意义和用法。

4.3.3. 调试相关

1. 策略可以先在 Local 模式下粗调，逻辑基本正确了，再到 Remote 模式下进行适应性测试和参数修正。**比赛策略一定要在 Remote 模式下进行过测试；对抗性比赛策略一定要测试过半场交换后的行为是否与预期一致。**

2. 为方便调试，需要在项目属性（Project Properties）中设置调试（Debug）→启动操作（Start action）→启动外部程序（Start external program）选项，欲在 Local 模式调试，设为%URWPGSim2D%\URWPGSim2D\bin\URWPGSim2DServer.exe；欲在 Remote 模式调试，设为%URWPGSim2D%\URWPGSim2D\bin\URWPGSim2DClient.exe。需要注意的是，刚从 SVN 服务器上获取到代码，bin 目录下是没有这两个文件的，其他自有组件除了 URWPGSim2D.Core.dll 之外也是没有的，需要先“生成解决方案（Build Solution）”。

3. 由于一个 Solution 只可以设置一个启动项目，而一个 Solution 里可能有若干个策略 Project，所以调试时，一般可以在需要调试的策略 Project 名称上点右键，**选择“调试”→“启动新实例”**。

4. 调试 2 支及以上队伍参与的仿真使命策略时，一定要事先生成一个或多个备用策略（最简单的即可），**被调试的策略，只能作为一支队伍的策略加载一次**，否则在策略代码中下的断点，每个周期都会中断多次，不利于跟踪。

5. 调试时出现找不到组件一类的错误时，先检查被调试项目的引用，不正确则先删除，再重新从%URWPGSim2D%\URWPGSim2D\bin\下添加（在“引用”目录上右键，“添加引用”→“浏览”，找到前述目录，选择需要的程序集）。主程序版本号更新后，重新生成策略 dll，也可能需要先删除引用，再重新添加；这取决于原来引用的程序集所在目录与新版本程序集所在目录是否一致。

5. 标准函数

5.1. PoseToPose函数

该函数位于 URWPGSim2D.StrategyHelper 命名空间（URWPGSim2D.StrategyHelper.dll）下的 Helpers 类（静态类）中。

5.1.1. 函数功能介绍

PoseToPose 函数，也称作位姿到位姿控制函数，其作用是实现仿真机器鱼从当前位姿到目标位姿的精确控制。位姿到位姿的控制分为两个阶段：第一阶段，控制仿真机器鱼快速游动到临时目标点；第二阶段，控制仿真机器鱼游动至目标点。其中，临时目标点为最终目标位姿反向延长线上的某一点，其距离阈值可以调节。

5.1.2. 函数参数说明

5.1.2.1. 函数原型

```
public static void PoseToPose(ref Decision decision, RoboFish fish, xna.Vector3 destPtMm, float destDirRad, float angThreshold, float disThreshold, int msPerCycle, ref int times)
```

5.1.2.2. 参数说明表

表 5-1 PoseToPose 函数参数说明表

参数名称	参数类型	参数说明
decision	ref Decision	PoseToPose 计算得到的决策变量值（输出参数）
fish	RoboFish	执行 PoseToPose 的仿真机器鱼对象
destPtMm	xna.Vector3	目标位置坐标（目标点）
destDirRad	float	目标方向弧度值（目标方向）
angThreshold	float	关键调节参数（中间方向与鱼体方向度数差值绝对值）上限，默认 30 度
disThreshold	float	关键调节参数（临时目标点与最终目标点距离）阈值
msPerCycle	int	每个仿真周期的毫秒数，即 mission.CommonPara.MsPerCycle
times	ref int	参见5.1.3（输出参数）

5.1.3. 函数调用方法

1. 在策略代码 Strategy 类中添加整型成员变量，并初始化为 0，作为调用 PoseToPose 函数的最后一个参数 times 的输入，如 int times;。该变量用于记录算法进入第二阶段的时间。

2. 在策略代码 Strategy 类的成员函数（方法）中调用 PoseToPose。使用推荐参数的调用代码如下，可根据实际调试情况进行调整。

```
StrategyHelper.Helpers.PoseToPose(ref decisions[i], mission.TeamsRef[teamId].Fishes[i], targetPoint, targetDirection, 30.0f, 8 * b.RadiusMm, mission.CommonPara.MsPerCycle, ref times);
```

3. 需要调用者自行判断是否已经完成位姿控制目标。完成一次位姿控制目标后，若需要再次调用PoseToPose进行新的位姿到位姿控制，需要将第1步中定义的times参数的输入变量（Strategy类的成员变量）清零。

5.2. Dribble函数

该函数位于 URWPGSim2D.StrategyHelper 命名空间（URWPGSim2D.StrategyHelper.dll）下的 Helpers 类（静态类）中。

5.2.1. 函数功能介绍

Dribble 函数，也称作带球函数，其作用是实现仿真机器鱼的带球控制。在某些比赛项目中，若想控制仿真机器鱼带球，则可调用此函数。

5.2.2. 函数参数说明

5.2.2.1. 函数原型

```
public static void Dribble(ref Decision decision, RoboFish fish, xna.Vector3 destPtMm, float destDirRad,float angleTheta1, float angleTheta2, float disThreshold, int VCode1, int VCode2, int cycles, int msPerCycle, bool flag)
```

5.2.2.2. 参数说明表

表 5-2 Dribble 函数参数说明表

参数名称	参数类型	参数说明
decision	ref Decision	Dribble 计算得到的决策变量值（输出参数）
fish	RoboFish	执行 PoseToPose 的仿真机器鱼对象
destPtMm	xna.Vector3	目标位置坐标（目标点）
destDirRad	float	目标方向弧度值（目标方向）
angleTheta1	float	鱼体方向与目标方向角度差的阈值一；角度差在此阈值范围内，则赋给仿真机器鱼一个合理的速度档位（见参数 disThreshold 说明）
angleTheta2	float	鱼体方向与目标方向角度差的阈值二；角度差小于此阈值，则仿真机器鱼直线游动；角度差大于此阈值，则仿真机器鱼调整游动方向
disThreshold	float	距离阈值；距离大于此阈值，仿真机器鱼以速度档位 VCode1 游动；距离小于此阈值，仿真机器鱼以速度档位 VCode2 游动
VCode1	int	直游档位 1（默认 6 档）
VCode2	int	直游档位 2（默认 4 档）
cycles	int	速度和转弯档位之间切换所需周期数经验值；建议取值范围在 5-20 之间；此参数作用是防止机器鱼“转过”
msPerCycle	int	每个仿真周期的毫秒数，即 mission.CommonPara.MsPerCycle
flag	bool	机器鱼坐标选择标准；true 为 PositionMm，即鱼体绘图中心；false 为 PolygonVertices[0]，即鱼头点

5.2.3. 函数调用方法

在策略代码 Strategy 类的成员函数（方法）中调用 Dribble。使用推荐参数的调用代码如下，可根据实际调试情况进行调整。

```
StrategyHelper.Helpers.Dribble(ref decisions[i], mission.TeamsRef[teamId].Fishes[i], targetPoint, targetDirection, 5, 10, 150, 6, 4, 15, 100, true);
```