# 第三章 球员智能体设计

机器人&智能技术实验室 A505/C501

# 本章概要

- 有益的工作基础
  - CMU
  - FCP
  - UVA
- 球员智能体整体设计架构
- 球员智能体具体功能设计
  - 球员智能体构建核心类
  - 球员智能体构建其他辅助类
- 球员智能体程序控制流程
  - 三线程控制结构
  - 球员智能体执行例程

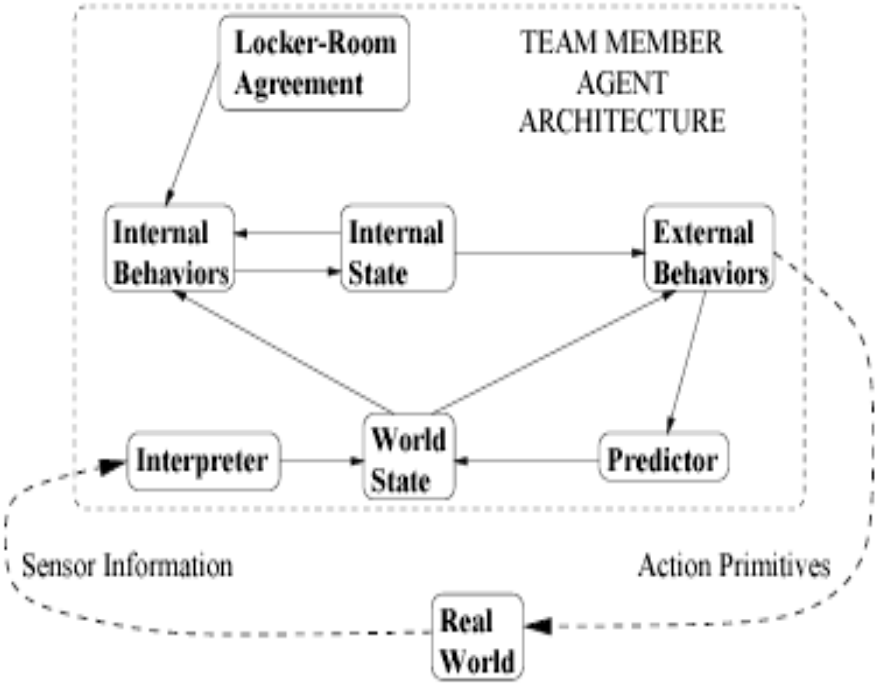
# 3.1有益的工作基础

- CMU　　1998、1999年世界冠军
- FC Portugal　　2000年世界冠军
- UVA-Trilearn　　2003年世界冠军

# 3.1.1CMU99

- CMU是美国卡耐基梅隆大学(Carnegie Mellon University ）的一支球队，曾获得了RoboCup98、99仿真组的冠军。这支球队的主要设计人Peter Stone在他的博士论文《Layered Learning in Mutli_agent System》中详细的描述了这支球队。

# 3.1.1CMU99(1)

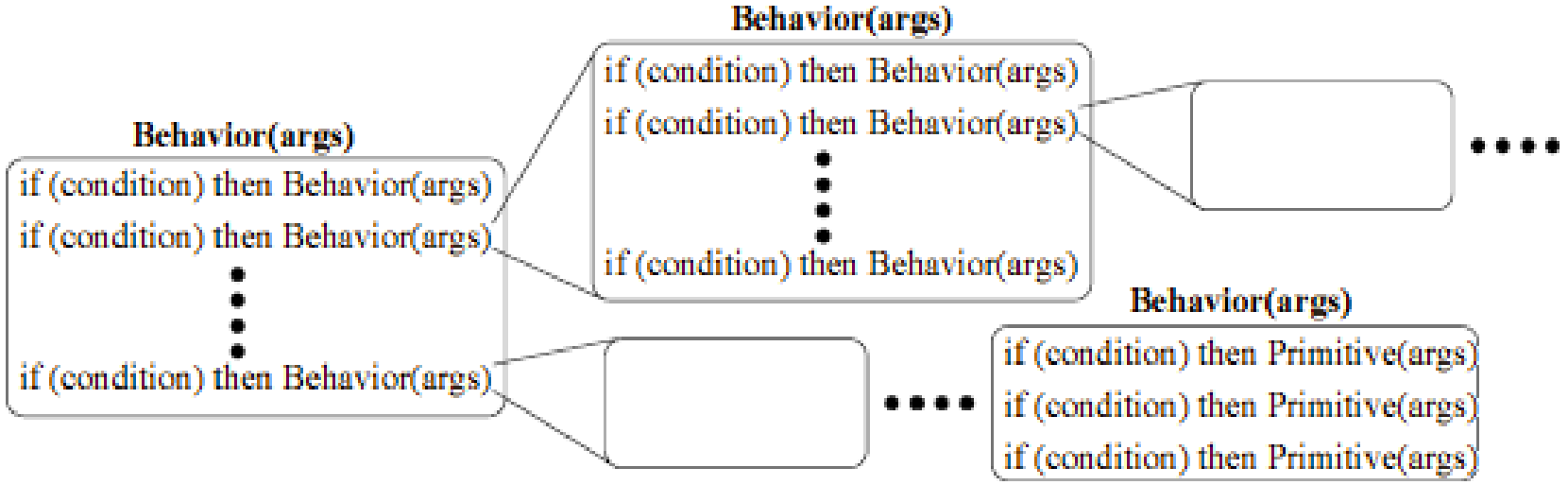

Locker-Room Agreement
TEAM MEMBER AGENT ARCHITECTURE
Internal Behaviors
Internal State
External Behaviors
Interpreter
World State
Predictor
Sensor Information
Action Primitives
Real World

# 3.1.1CMU99(2)

- Real World：Server表示的世界状态
- World State：agent内部可识别的世界状态。从Real World到World State需要进行解析
- Locker-room Agreement：用于Agent的同步，并定义了球队进行协作的机构以及Agent之间的通讯协议。它仅能够被Internal Behaviors访问
- Internal State：存储了Agent的一些内部变量。可以存储Agent以前或当前时刻的世界状态
- Internal Behaviors：根据当前的世界状态、内部状态、球队协议（Locker-room Agreement）来更新Agent的内部状态的内部动作
- External Behaviors：根据世界状态以及更新后的内部状态来做出一个动作送给动作器以作用于真实世界（Real World）。同时回送给Agent进行预测

# 3.1.1CMU99(3)

- BC树



# 3.1.1CMU99(4)

- CMU的层次学习
  - 第一层，进行Agent个体基本技术的学习。典型的例子就是进行断球的学习，通过神经网络的方法，对在不同的场景下学习断球。
  - 第二层，Agent同另外一个Agent之间的协同学习。典型的例子是进行传球的学习，如在球场上Agent当前在控球，并且做出了决策要就进行传球给一个特定的队友，这时它必须学习一个合适的方向和传球速度。在该层它可以调用第一层已经学习过的基本技术。这通过构造决策树的方法，求出相应的节点值，构造出一个分类器。典型的算法是C4.5决策树算法。
  - 第三层，Agent同其他多个Agent之间进行的球队策略学习。比较典型的例子是进行传球对象的选择，如在球场上Agent当前在控球，这时它要选择把球传给哪个队友。在学习本层的时候，也认为第二层已经学习过了，这时它要考虑时传给哪个队友的利益大。主要是通过TPOT-RL(Team-Partitioned Opaque-Transition Reinforcement Learning)来进行学习。

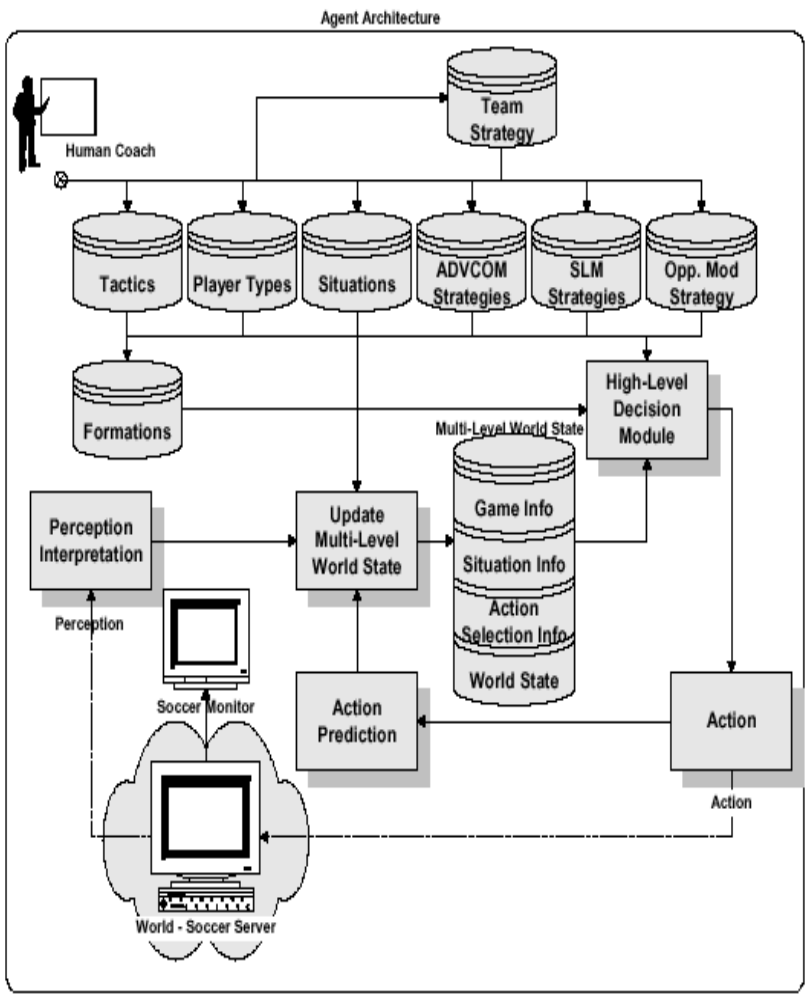
# 3.1有益的工作基础

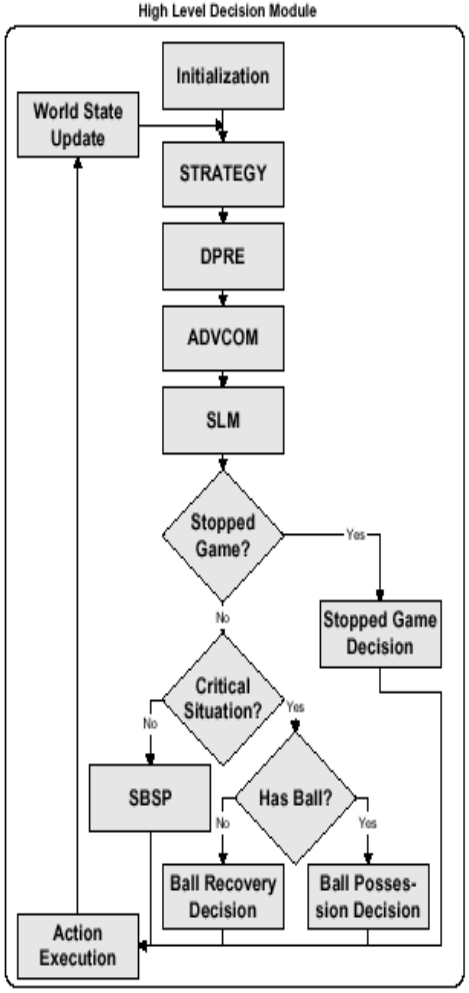
- CMU 1998、1999年世界冠军
- FC Portugal 2000年世界冠军
- UVA-Trilearn 2003年世界冠军

# 3.1.2 FC Portugal 2000

- FC Portugal由葡萄牙的里斯本大学和波尔图大学合作完成的一支球队。获得了2000年世界冠军。
- 它是在CMUnited99公开的底层源代码的基础上，对多智能体的合作方面做出了巨大的贡献（在这之前，RoboCup仿真参赛队的阵型以及站位都很混乱
- FC Portugal在球队策略、战术、阵型、球员类型、站位机制以及角色的动态转换机制等方面都有自己的特点。

# 3.1.2 FC Portugal 2000（1）



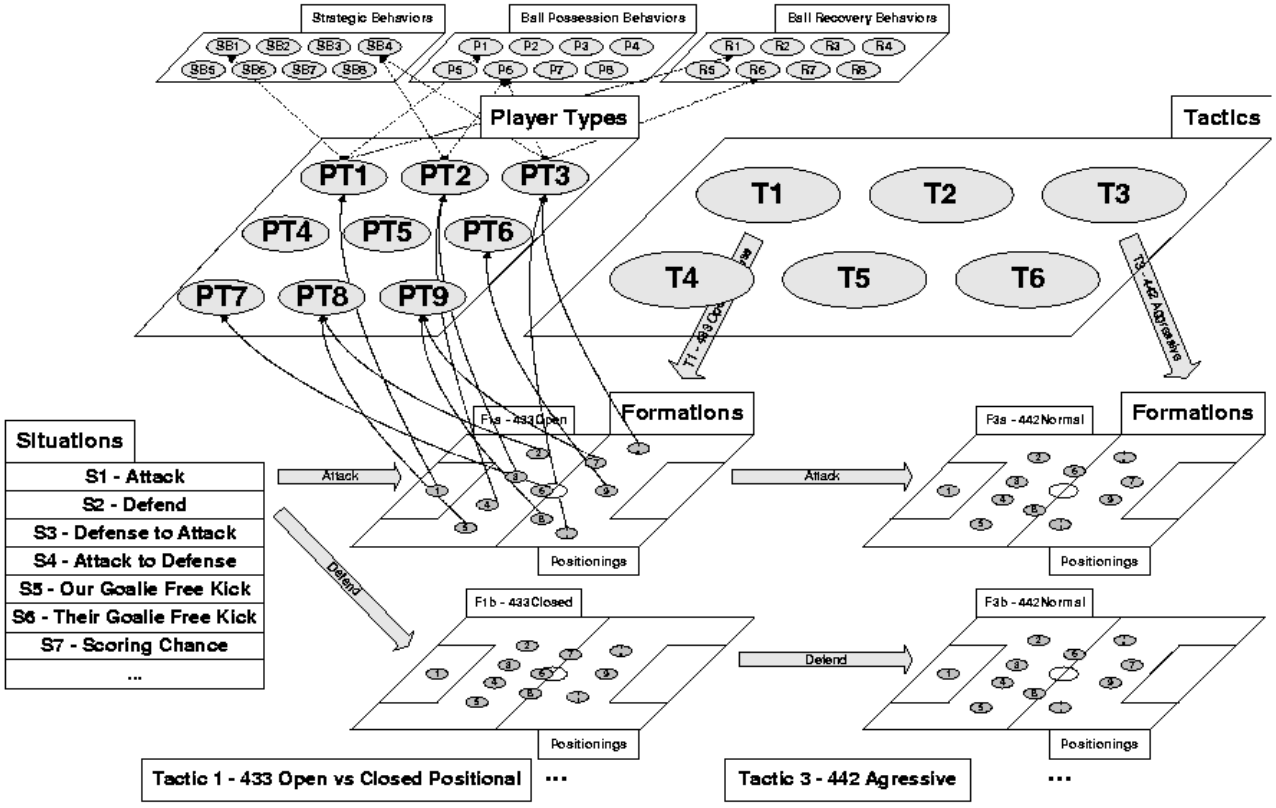



# 3.1.2 FC Portugal 2000（2）

- FC Portugal的信息模型是一个四层结构的数据模型：
  - 全局形势信息－ 高层信息比如比分、时间、比赛策略（射门，成功传球，控球等）和对方行为，用来确定每一时刻的战术；
  - 形势信息－ 与阵型选择相关的信息，和SBSP，SLM和ADVCOM机制相关的信息；
  - 动作选择信息－ 一套高层参数，用来确定动态形势，选择适当的控球或开球行为；
  - 世界状态－ 底层信息，包括球员和足球的位置和速度。

# 3.1.2 FC Portugal 2000（3）

- CMUnited提出了
并且根据比赛结
变阵型；FC Por
并提出了战术和
- FC Portugal球队
球员类型（定义
开球行为）和一
（433，442，
等）。
- 阵型是在不同的
如防守、攻击、
球门球等，对每
点和球员类型。
的位置和动作的



# 3.1.2 FC Portugal 2000（4）

- SBSP (Situation Based Strategic Positioning,基于场上形势的策略站位)
  - SBSP就是Agent能够根据当前球场上的形势，包括现在球队正在使用的阵形、战术以及球员的类型来确定球员在球场上的基本位置；再通过球的位置、速度、球场上的形势（如本方是在进攻、本方是在防守、双方的得失球等）以及球员的策略特性来修正基本位置，得到球员的应该跑向的位置，也就是球员的策略跑位点。球员的策略特性包括对球的吸引力、球场球员可以允许站的位置、在场上某些区域的特定位置特性、粘球的倾向、越位线的设置以及在某些特定形势下对场上特定目标的注意力等方面。

# 3.1.2 FC Portugal 2000（5）

- DPRE（Dynamic Positioning and Role Exchange ）动态站位和角色变换。
  - 是基于Peter Stone关于根据协议来进行智能体角色的变换的继续研究。在FC Portugal中，球员不仅能够改变它们的站位（站位由阵型来决定的），而且还可以在当前阵型下改变球员类型。当然DPRE只有有利于球队时才使用的。是否有利，只要通过一个效用函数来进行评价的。效用函数要考虑一下因素：球员位置到它们的战略位置的距离、每个站位的重要性和在当前形势下的站位是否恰当。

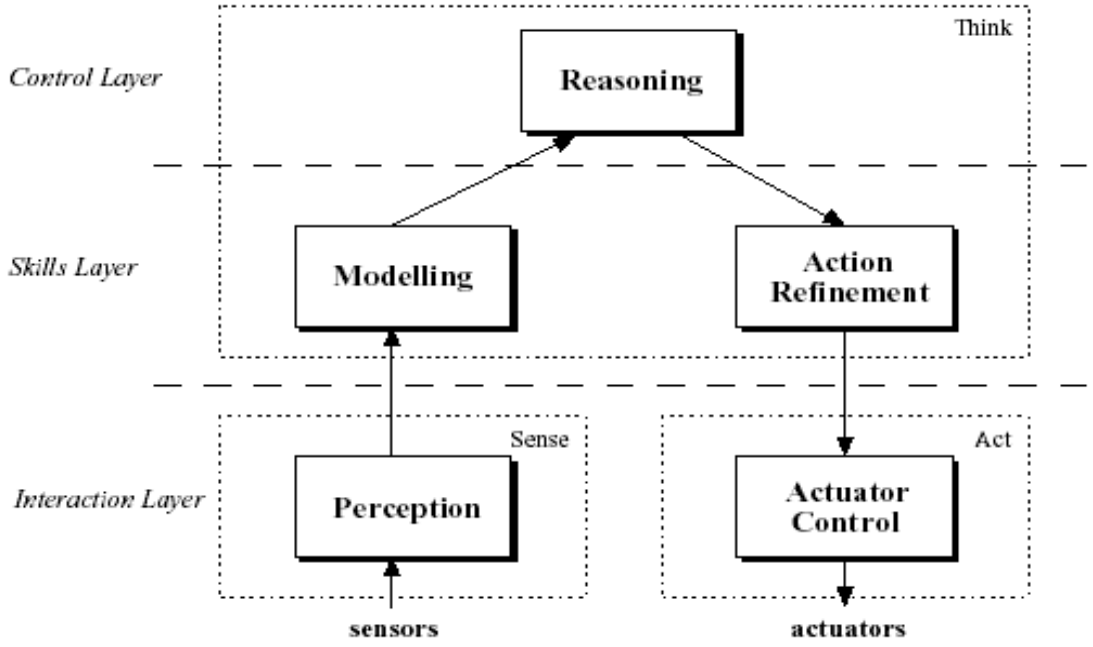
# 3.1有益的工作基础

- CMU　　1998、1999年世界冠军
- FC Portugal　　2000年世界冠军
- UVA-Trilearn　　2003年世界冠军

# 3.1.4 UVA-Trilearn

- UVA-Trilearn是荷兰阿姆斯特丹大学的一支球队，获得了2003年的世界冠军。欧洲的老牌强队，它的Agent结构设计成三层的分层结构。
- UVA-Trilearn的MAS结构特点不是很明显，它主要把异构球员的方法应用到Agent系统，所以该球的攻击力比较强悍。

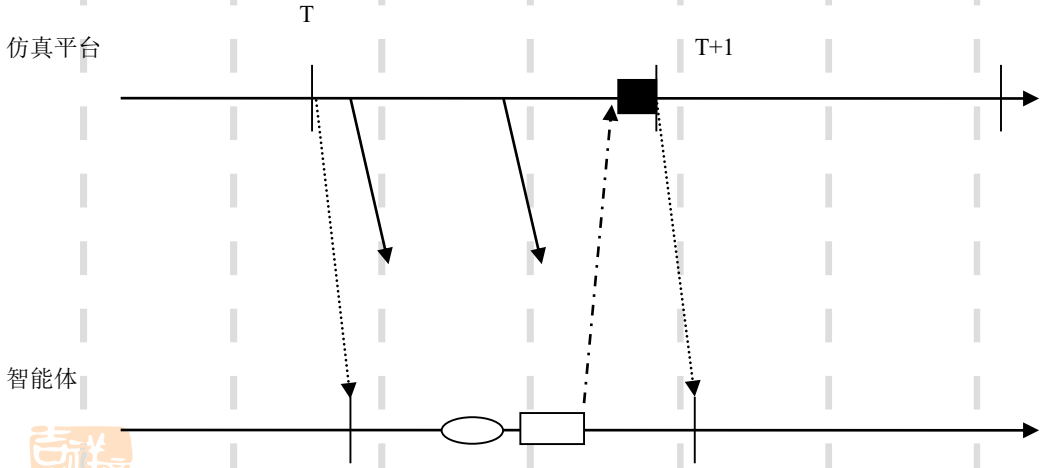
# 3.1.5 UVA-Trilearn（1）

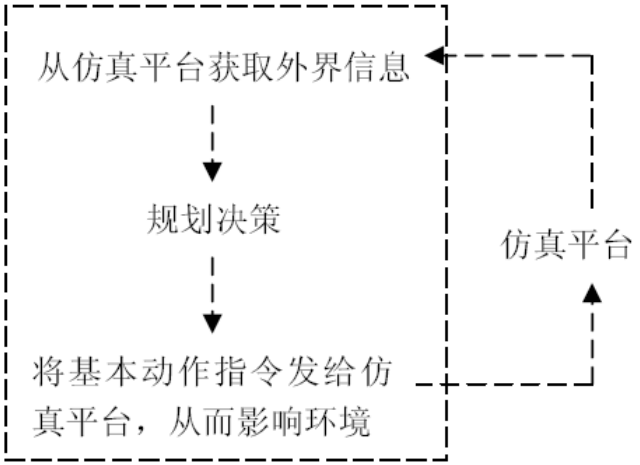


# 本章概要

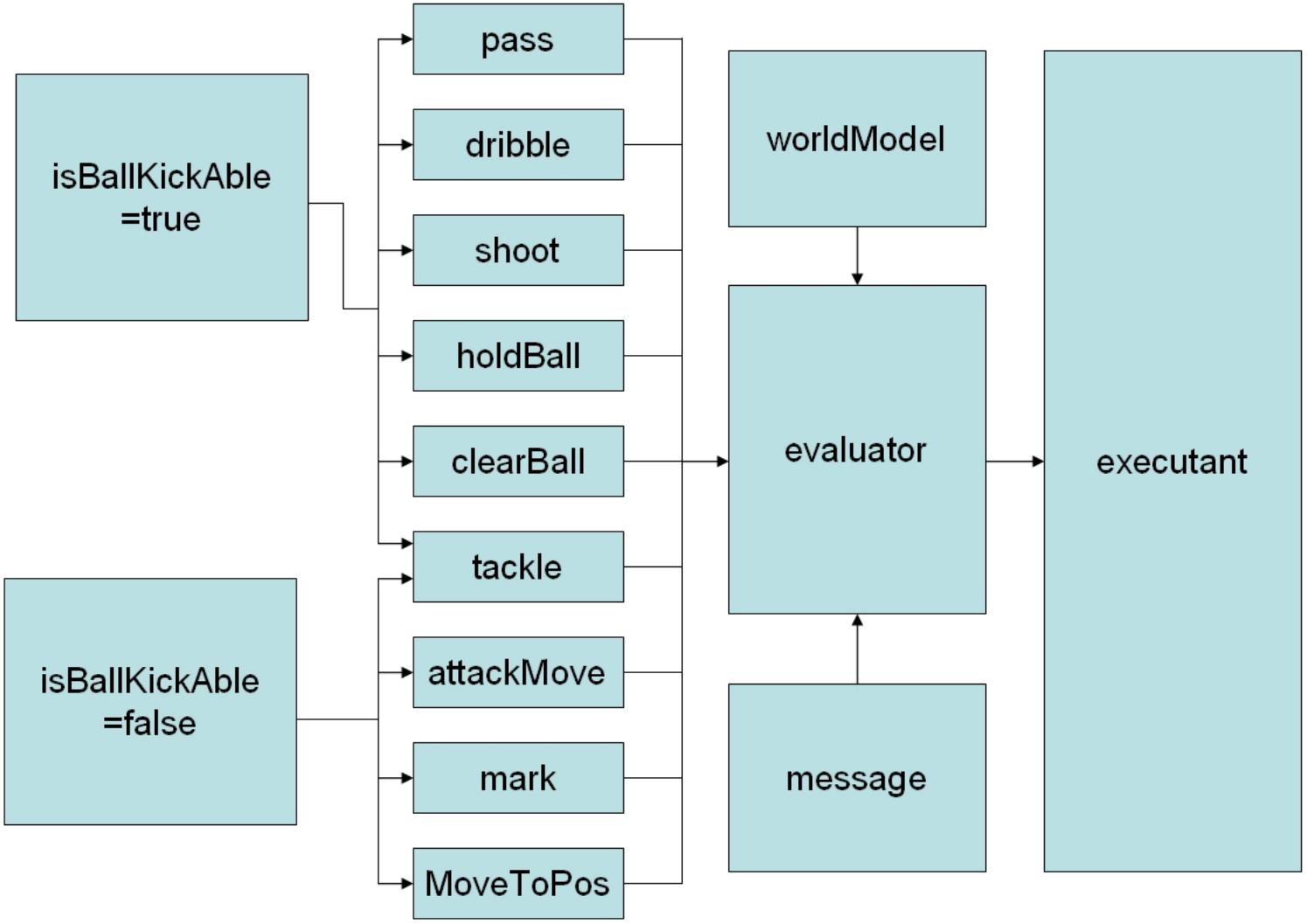
- 有益的工作基础
  - 记98-10年历届世界冠军的有益经验
- 球员智能体整体设计架构
- 球员智能体具体功能设计
  - 球员智能体构建核心类
  - 球员智能体构建其他辅助类
- 球员智能体程序控制流程
  - 三线程控制结构
  - 球员智能体执行例程

# 3.2球员智能体整体设计架构



# 3.2球员智能体整体设计架构-一种策略框架



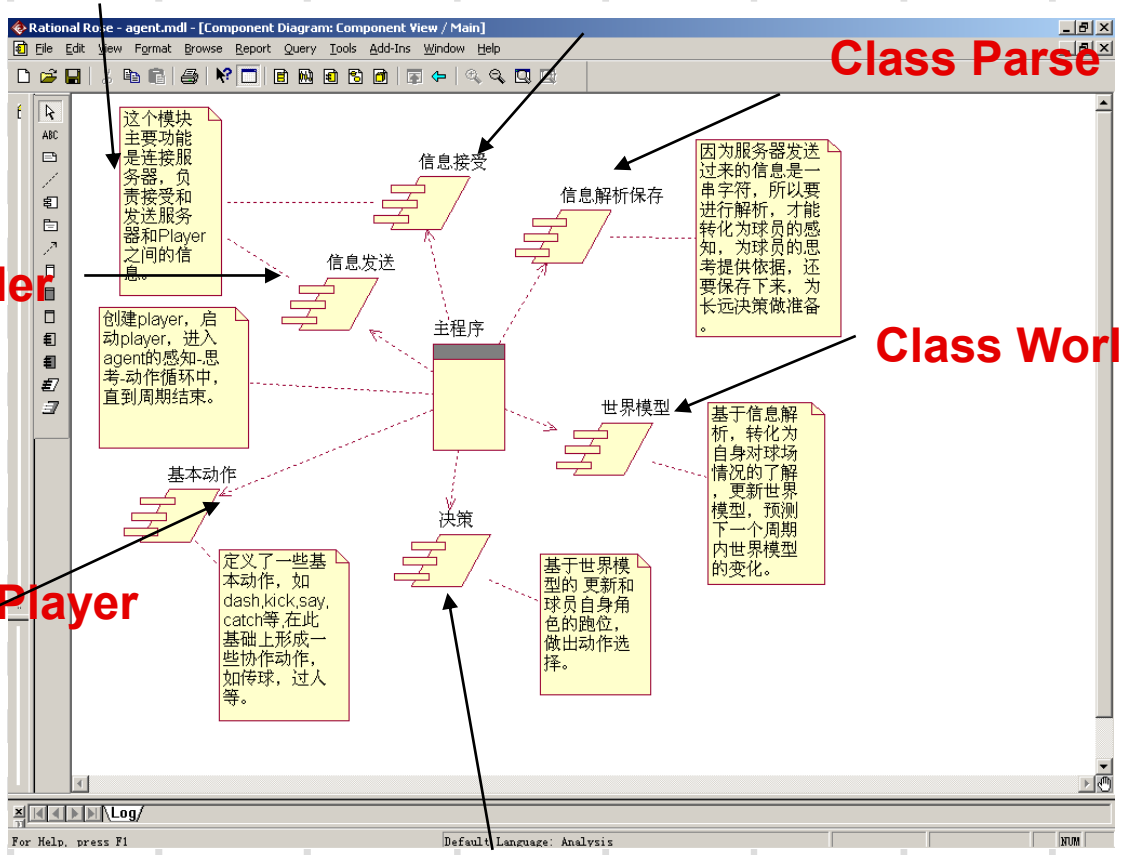
# 本章概要

- 有益的工作基础
  - CMU
  - FCP
  - UVA
- 球员智能体整体设计架构
- 球员智能体具体功能设计
  - 球员智能体构建核心类
  - 球员智能体构建其他辅助类
- 球员智能体程序控制流程
  - 三线程控制结构
  - 球员智能体执行例程

# 3.3球员智能体具体功能设计

Class Connection　Class SenserHandler

Class Parse




Class **ActHandler**

Class **WorldModel**

Class **BasicPlayer**

Class Player

# 3.3.1球员智能体构建核心类

- **Class Connection** （与Server建立基于UDP/IP协议的Socket连接类 ）
- **Class SenserHandler** （感知信息类）
- **Class WorldModel** （WorldModel类）
- **Class Object及相关派生类** （场上对象类）
- **Class BasicPlayer** （基本动作类）
- **Class Player** （决策类）
- Class **ActHandler** （动作发送类）

# 3.3.1球员智能体构建核心类（1）

① 消息的解析和发送模块（第四章）

① Server与Client之间是通过基于UDP/IP协议进行信息(普通字符串)传输的。

② socket和Connection类负责最底层的信息传递，负责字实现符串的传输。

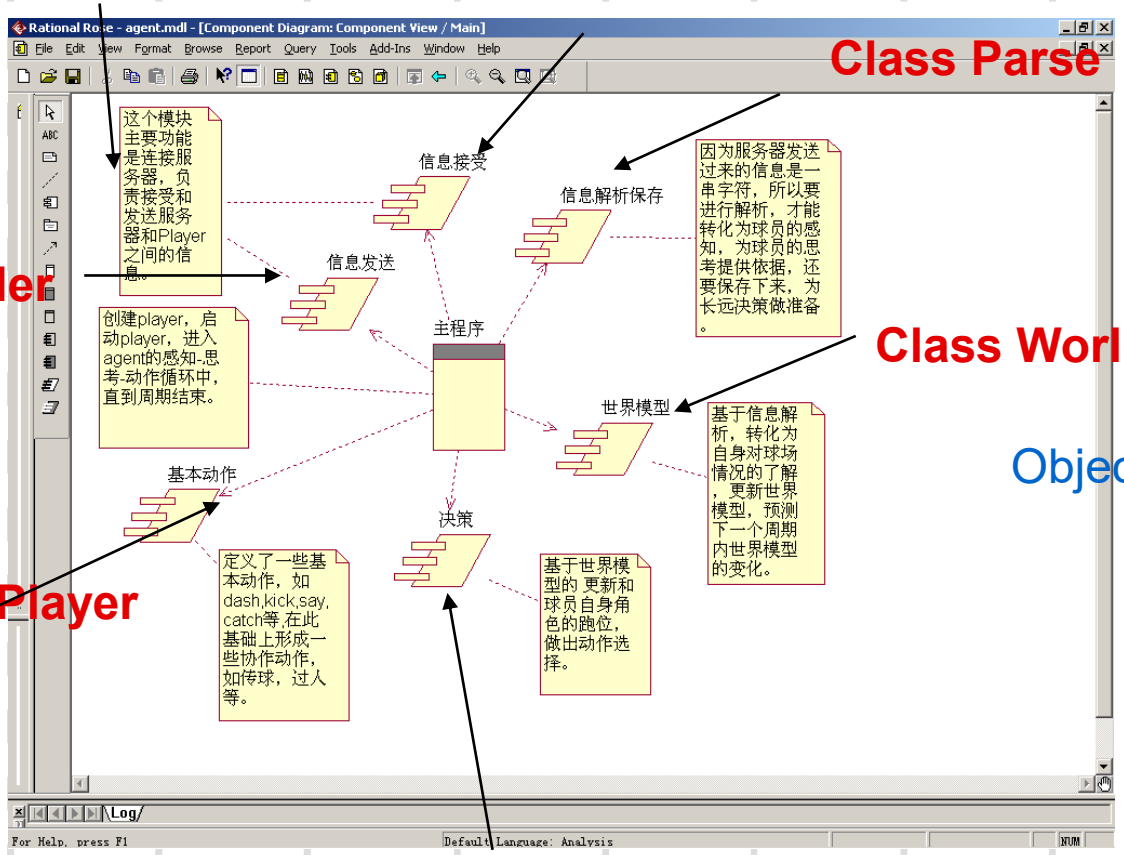
③ 类SenseHandler，它是基于Connection基础上进行感知信息的处理的，对信息进行分类处理，然后交给WorldModel.

④ 类ActHandler，它是基于Connection基础上进行特定动作的发送处理的，要把特定动作（包括原子动作和高级动作）处理成Server能够理解的字符序列。

# 3.3.1球员智能体构建核心类（2）

Class Connection　Class SenserHandler

Class Parse




Class **ActHandler**

Class **WorldModel**

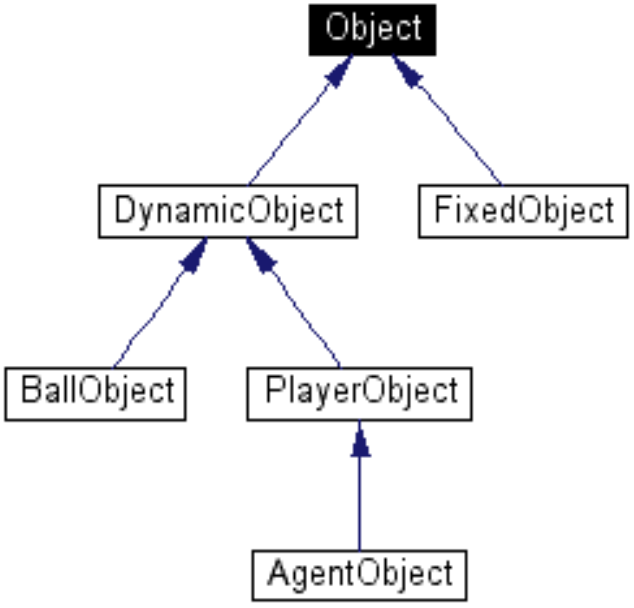
Object信息

Class **BasicPlayer**

Class Player

# 3.3.1球员智能体构建核心类（2）

② 场上对象类



# Object类

- 包含所有比赛中所有对象可用的信息。与agent有关的信息在AgentObject中声明。类中提供了对自由属性的访问函数
  - **ObjectT objectType** //对象类型。
  - **Time timeLastSeen** //上次视觉信息接收时间。
  - **VecPosition posGlobal** //全局位置。
  - **Time timeGlobalPosition** //更新全局位置的服务器时间。
  - **VecPosition posRelative** //相对agent的位置
  - **Time timeRelativePosition** //更新相对位置的服务器时间
  - **VecPosition posGlobalLastSee** //上次接收到视觉信息时的全局位置。
  - **Time timeGlobalPosDerivedFromSee**//从上次视觉信息得出的位置的时间

# FixedObject类

- 即固定对象类，主要是是用来记录了不能移动的对象的信息。如标志，球门，线等 。

# DynamicObject类

- 动态对象类，该类主要是用来记录移动对象的信息。
  - **VecPosition vecGlobalVelocity** //绝对速度。注意这是一个矢量。
  - **Time timeGlobalVelocity** //记录绝对速度的时间。
  - **double dRelativeDistanceChange** //相对距离的改变。
  - **double dRelativeAngleChange** //相对角度的改变。
  - **Time timeChangeInformation** //改变信息的时间

# BallObject类

- 含球的所有信息，对**DynamicObject类**并没有增加新的成员

# PlayerObject类

- 即球员类，对DynamicObject类增加了一些成员和方法。
  - bool isKnownPlayer //该球员号码的可信度是否足够高。
  - ObjectT objRangeMin //对象类型范围的最小值
  - ObjectT objRangeMax //对象范围的最大值
  - bool isGoalie //是否是守门员
  - AngDeg angGlobalBodyAngle //绝对身体角度。
  - AngDeg angGlobalNeckAngle //绝对脖子角度。
  - Time timeGlobalAngles //绝对角度改变时间。
  - int iHeteroPlayerType //异构类型

# AgentObject类

- 它在类PlayerObject上面扩充了体力、视觉等相关属性和方法。
  - **ViewAngleT viewAngle** //视角宽度
  - **ViewQualityT viewQuality** //视觉质量
  - **Stamina stamina** //体力
  - **VecPosition velSpeedRelToNeck** //相对于脖子的速度
  - **AngDeg angBodyAngleRelToNeck** //相对于脖子的身体角度
  - **VecPosition posPositionDifference** //位置差

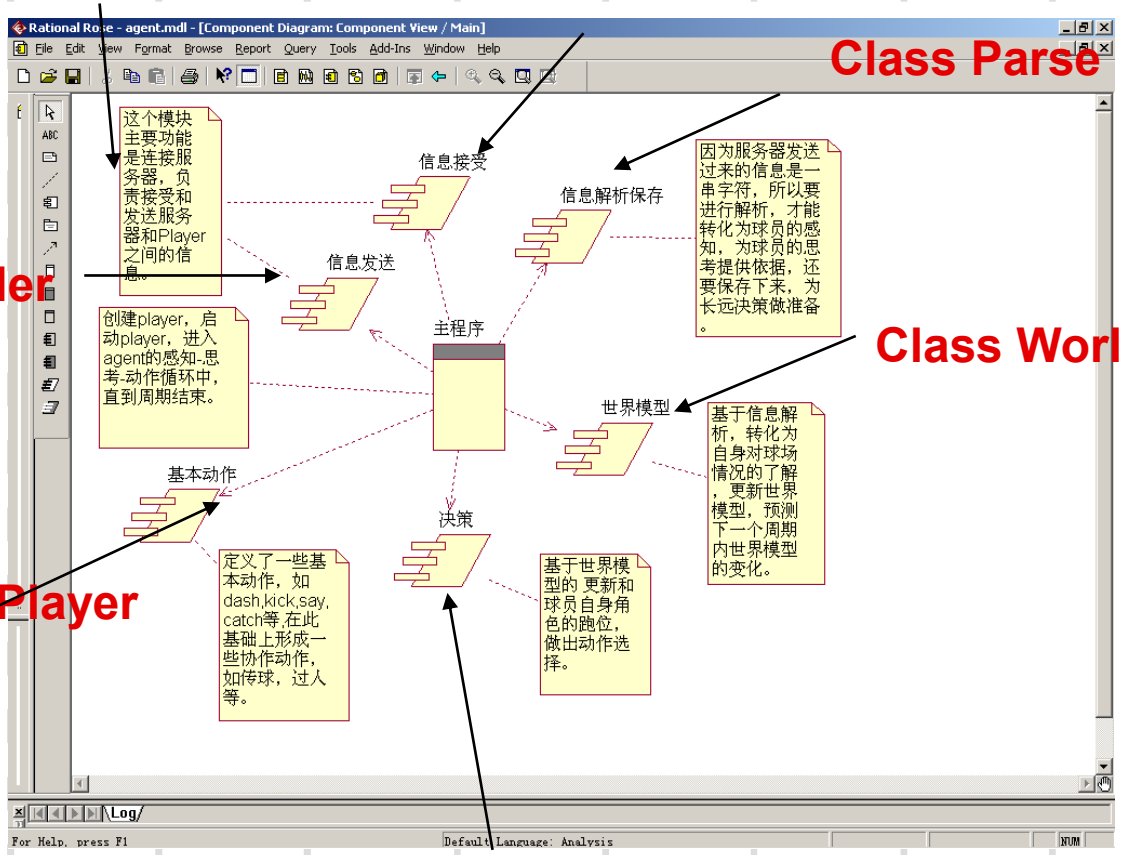
# Stamina类

- 体力类；该类包含Soccer Server所定义的与体力相关的参数。
  - double m_dStamina　　//球员体力
  - double m_dEffort　　//体力效用
  - double m_dRecovery　　//体力恢复
  - double m_dCapacity　//体力容量

# 3.3.1球员智能体构建核心类（3）

Class Connection　Class SenserHandler

Class Parse




Class **ActHandler**

Class **WorldModel**

Class **BasicPlayer**

Class Player

# 3.3.1球员智能体构建核心类（3）

## ③ 世界模型类（第五章详细介绍）

- WorldModel描述了球场所有对象（如球员、球、标志、线等）的信息（包括位置信息和速度信息以及其他一些信息，如身体朝向等等）以及比赛信息等。
  - Environmental information 环境信息

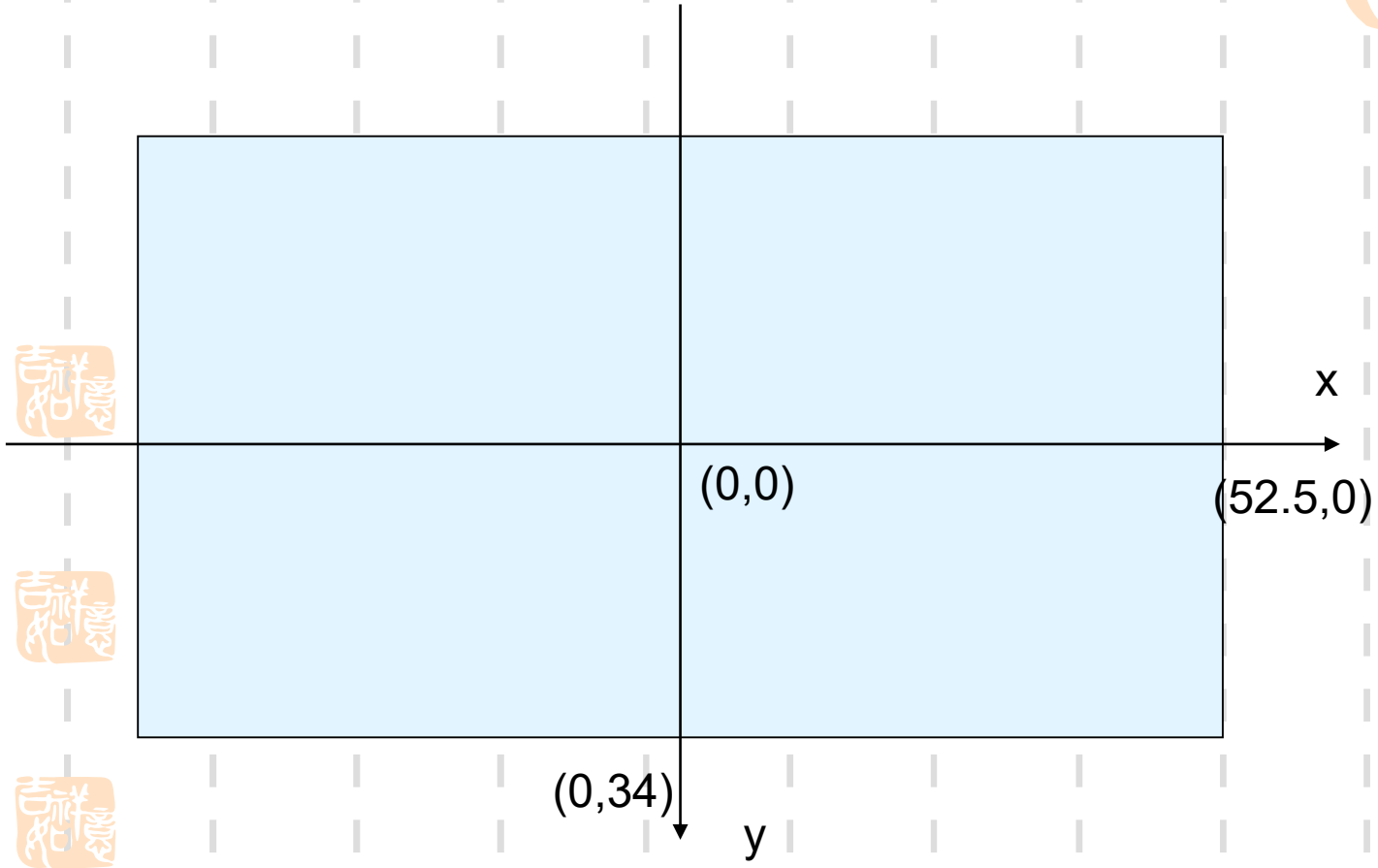
    包含：服务器参数，异构球员（每一种异构球员的具体参数）
  - Match information 比赛信息

    包含：Time, PlayerNumber, Side, TeamName, PlayMode, GoalDiff（比分差）
  - Object information 对象信息（*******）

    包含各类对象且有继承关系
  - Action information 动作信息

    包含：每种动作执行的次数，动作队列，已经执行的动作

# 3.3.1球员智能体构建核心类（3）



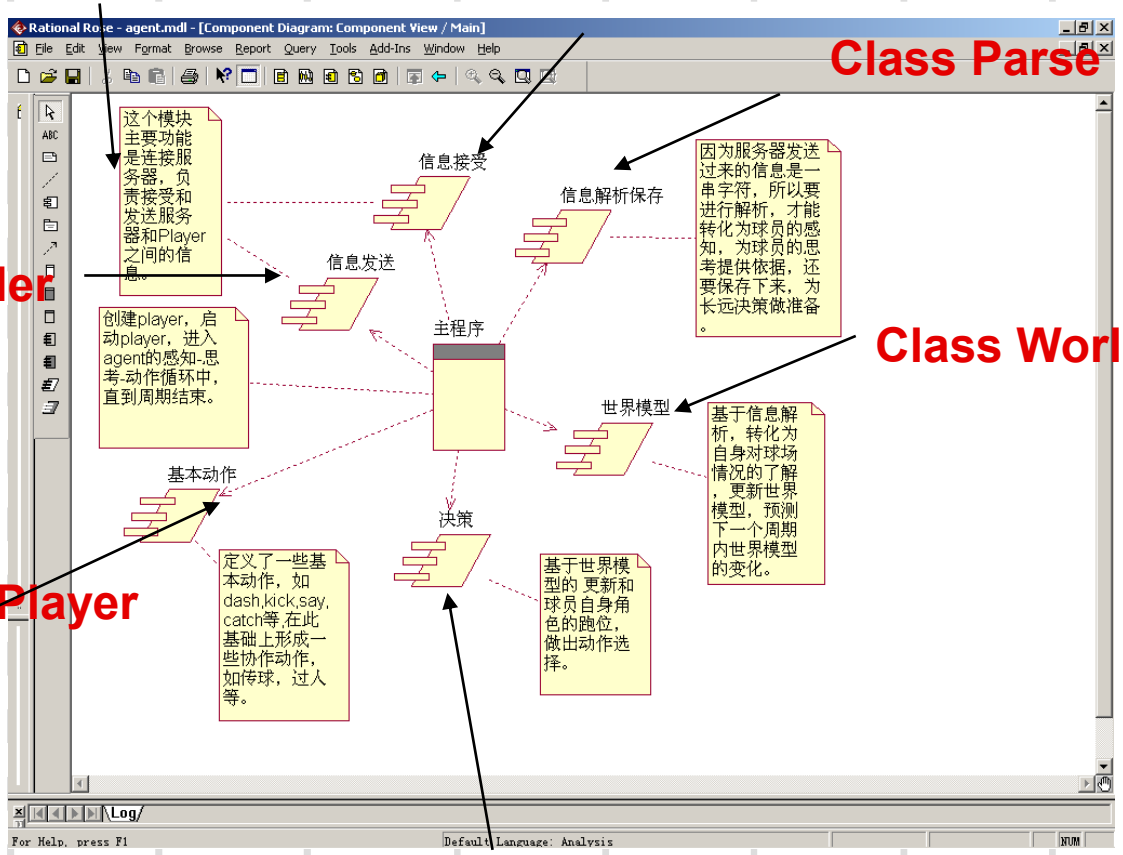
# 3.3.1球员智能体构建核心类（3）

- 类**WorldModel**所包含的方法可分为四类：
  - Retrieval methods
    从球员的世界模型中获得对象的信息
  - Update methods
    从来自server的感觉信息更新球员的世界模型
  - Prediction methods
    根据先前的感觉信息预测世界的将来的状态
  - High-Level methods
    从基本的世界模型信息得出高层的结论

# 3.3.1球员智能体构建核心类（4）

Class Connection　Class SenserHandler

Class Parse




Class **ActHandler**

Class **WorldModel**

Class **BasicPlayer**

Class **Player**

# 3.3.1球员智能体构建核心类（4）

④ 基本动作类（第六章详细介绍）

- Class BasicPlayer。该类主要作用：该类定义球员智能体使用的skill。如何选择时机执行这些skill在该类里并没有给出，而在Player类(决策类)里给出。
- 动作类型::
  - LOW-LEVEL SKILLS
  - INTERMEDIATE SKILLS
  - HIGH-LEVEL SKILLS

# 3.3.1球员智能体构建核心类（4）

- LOW-LEVEL SKILLS（低级动作）
  - SoccerCommand alignNeckWithBody ( );
  - SoccerCommand turnBodyToPoint (VecPosition pos, int iPos= 1);
  - SoccerCommand turnBackToPoint ( VecPosition pos, int iPos=1 );
  - SoccerCommand turnNeckToPoint ( VecPosition pos,SoccerCommand com);
  - SoccerCommand searchBall ( );
  - SoccerCommand dashToPoint ( VecPosition pos );
  - SoccerCommand freezeBall ( );
  - SoccerCommand kickBallCloseToBody ( AngDeg ang );
  - SoccerCommand accelerateBallToVelocity( VecPosition vel );
  - SoccerCommand catchBall ( );
  - SoccerCommand communicate ( char *str );
  - SoccerCommand teleportToPos ( VecPosition pos );

# 3.3.1球员智能体构建核心类（4）

- INTERMEDIATE SKILLS（中间动作）
  - SoccerCommand turnBodyToObject (ObjectTo );
  - SoccerCommand turnNeckToObject ( ObjectT o, SoccerCommand com );
  - SoccerCommand moveToPos ( VecPosition posTo, DAngeg angWhenToTurn, double dDistDashBack = 0.0, bool bMoveBack = false);
  - SoccerCommand interceptClose ( );
  - SoccerCommand interceptCloseGoalie ( );
  - SoccerCommand kickTo ( VecPosition posTarget, double dEndSpeed );
  - SoccerCommand turnWithBallTo( AngDeg ang, AngDeg angKickThr, double dFreezeThr );
  - SoccerCommand moveToPosAlongLine ( VecPosition pos, AngDeg ang, double dDistThr, int iSign, AngDeg angThr, AngDeg angCorr );

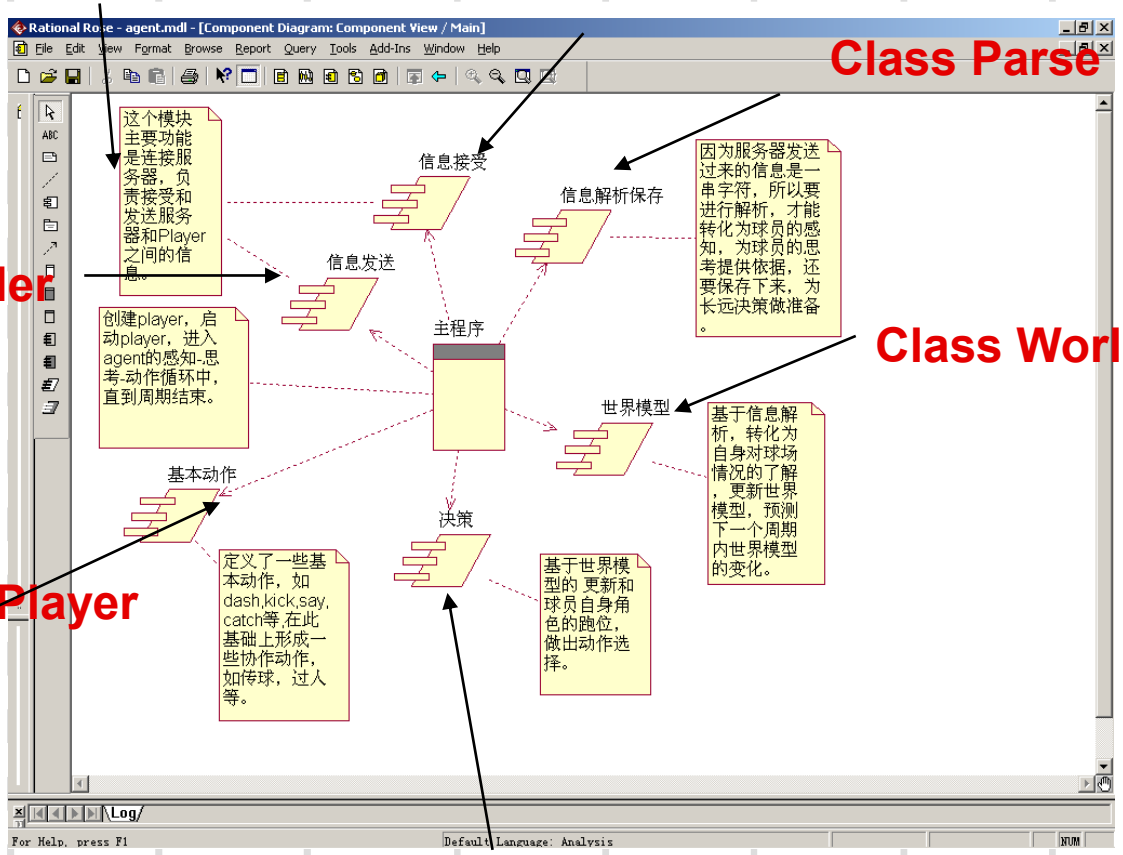
# 3.3.1球员智能体构建核心类（4）

- HIGH-LEVEL SKILLS(高级动作)
  - SoccerCommand intercept ( bool isGoalie);
  - SoccerCommand dribble (AngDeg ang, DribbleT d);
  - SoccerCommand directPass (VecPosition pos, PassT passType );
  - SoccerCommand leadingPass(ObjectT o, double dDist );
  - SoccerCommand throughPass ( ObjectT o, VecPosition posEnd, AngDeg *angMax = NULL );
  - SoccerCommand clearBall ( ClearBallT type, SideT s = SIDE_ILLEGAL, AngDeg *angMax = NULL );
  - SoccerCommand mark (ObjectT o, double dDist, MarkT mark);
  - SoccerCommand defendGoalLine ( double dDist);
  - SoccerCommand interceptScoringAttempt ();

# 3.3.1球员智能体构建核心类（5）

Class Connection　Class SenserHandler

Class Parse




Class **ActHandler**

Class **WorldModel**

Class **BasicPlayer**

Class Player

# 3.3.1球员智能体构建核心类（5）

- 球员决策类。Class Player。该类是BasicPlayer类的派生类，负责球场上面球员的决策。
- 主要函数：
  - void mainLoop ( );
  - void deMeer5 ( );
  - void deMeer5_goalie( );
  - void goalieMainLoop( );
  - void defenderMainLoop ( );
  - void midfielderMainLoop( );
  - void attackerMainLoop( );

# 3.3.2球员智能体构建其他辅助类

- 与足球知识相关的枚举类型及**Soccertypes**等类
- **Class Formations** （阵型相关信息类）
- **Class PlayerSettings** （球员一些重要参数设置类）
- **Class Parse** （字符串解析类）
- **Class ServerSettings** （Server参数设置类)
- **Class VecPosition** (坐标类)
- **Class Line, Circle 和 Rectangle** (线，圆及矩形类)
- **Class Geometry** (几何计算类)

# 3.3.2.1与足球知识相关的枚举类型及Soccertypes等类（1）

- #define MAX_TEAMMATES 11 /*!< Maximum number of teammates */
- #define MAX_OPPONENTS 11 /*!< Maximum number of opponents */
- #define MAX_HETERO_PLAYERS 18 /*!< Maximum number of hetero players*/
- #define MAX_MSG 4096 /*!< maximum message size from server*/
- #define MAX_SAY_MSG 10 /*!< maximum size of say message */
- #define MAX_TEAM_NAME_LENGTH 64 /*!< maximum length of teamname */
- #define MAX_FLAGS 55 /*!< maximum number of flags on field*/
- #define MAX_LINES 4 /*!< maximum number of lines on field*/
- #define DEFAULT_TEAM_NAME "Team_L" /*!< default teamname for own team */
- #define DEFAULT_OPPONENT_NAME "Team_R" /*!< default teamname for opponent */
- #define PITCH_LENGTH 105.0 /*!< length of the pitch */
- #define PITCH_WIDTH 68.0 /*!< width of the pitch */
- #define PITCH_MARGIN 5.0 /*!< margin next to the pitch */
- #define PENALTY_AREA_LENGTH 16.5 /*!< length of the penalty area */
- #define PENALTY_AREA_WIDTH 40.32 /*!< width of the penalty area */
- #define PENALTY_X (PITCH_LENGTH/2.0-PENALTY_AREA_LENGTH) /*!< penalty line of

# 3.3.2.1与足球知识相关的枚举类型及Soccertypes等类（1）

- enum ObjectT
- enum ObjectSetT
- enum PlayModeT
- enum RefereeMessageT
- enum ViewAngleT
- enum ViewQualityT
- enum SideT
- enum CommandT
- enum PlayerT
- enum PlayerSetT
- enum FormationT
- enum BallStatusT
- enum ActionT

# 3.3.2.1与足球知识相关的枚举类型及Soccertypes等类（1）

- enum MarkT
- enum DribbleT
- enum PassT
- enum ClearBallT
- enum TiredNessT
- enum FeatureT
- enum DirectionT
- enum SucceedT
- enum TackleT
- class Time

# 3.3.2.1与足球知识相关的枚举类型及Soccertypes等类（1）

- class SoccerCommand
- class Feature
- class SoccerTypes

# 3.2.2.2Class Formations （阵型相关信息类）

- **PlayerTypeInfo类**
  - 对每一种球员类型**PlayerT**的信息进行进一步扩充。如加上了每种类型球员的基本位置属性等等。
- **FormationTypeInfo类**
  - 对每一种阵型类型**FormationT**的信息进行进一步扩充。如在特定阵型下给各类型球员是如何布置的，他们的本位点在哪里等等。
- **Formations类**
  - 是一个阵型信息容器。包含了所有不同的阵型类型信息，还包含了球员当前的阵型以及球员在该阵型中的角色这2个属性。通过这2个属性就很容易确定该队员在阵型中的位置。

# 3.2.2.2Class Formations （阵型相关信息类）-1

- **class PlayerTypeInfo**
  - PlayerT playerType; /*!< This class gives information about this PlayerType*/
  - double dAttrX; /*!< x attraction to the ball */
  - double dAttrY; /*!< y attraction to the ball */
  - double dMinX; /*!< minimal x coordinate for this player type */
  - double dMaxX; /*!< maximal x coordinate for this player type */
  - bool bBehindBall;/*!< should player always stay behind the ball */

# 3.2.2.2Class Formations （阵型相关信息类）-2

- class FormationTypeInfo
  - FormationT formationType; /*!< type of this formation */
  - VecPosition posHome[ MAX_TEAMMATES ]; /*!< home position for roles*/
  - PlayerT playerType[ MAX_TEAMMATES ]; /*!< player_types for roles */
  - PlayerTypeInfo playerTypeInfo[ MAX_PLAYER_TYPES ]; /*!< info for roles */
- class Formations
  - FormationTypeInfo formations[ MAX_FORMATION_TYPES ]; /*!< stored formations*/
  - FormationT curFormation; /*!< type of the current formation */
  - int iPlayerInFormation; /*!< role agent in current formation */

# 3.2.2.3Class Parse （字符串解析类）

- Parse类它主要是用来负责对字符串进行解析，被SenseHandler类广泛使用，处理从Server端传过来的感知字符串，可以实现直接跳过一些字符，自自出从而获得想要的整型、实型等数据。
- 例如：
  - **static double parseFirstDouble( char** strMsg)**

# 3.2.2.4Class ServerSettings（Server参数设置类)

- ServerSettings类是包含了当前Server版本中配置文件中的所有参数，以及对这些参数赋值、取值的方法。

# 3.2.2.5Class PlayerSettings （球员重要参数设置类)

- PlayerSettings类里面主要包含球员进行动作决策时需要的一些重要变量临界值的设置。
- 例如：
  - **double dPlayerConfThr** //球员信息置信度临界值，低于该临界值则认为球员信息是非法和无效的，一般是0.88
  - **double dPlayerHighConfThr** //球员信息高置信度临界值，高于该值的球员信息则认为信息是可信的，一般是0.92

# 3.2.2.6Class VecPosition（坐标类）

- VecPosition类用来表示位置坐标，以及在此基础上面的一些运算。如判断2个点的前后左右关系等等，2种坐标系的转换等。

# 3.2.7Class Line, Circle 和 Rectangle (线，圆及矩形类)

- Line类表示直线类，直线可以表示成 ay + bx + c = 0. 因此可以使用系数a,b,c表示一条直线。
- Circle类表示一个圆，它包含一个VecPosition类的对象成员表示圆心，另外一个成员表示半径。
- Rectangle类用来表示一个矩形，一个矩形主要是由矩形的左上角和右下角的两点来确定。

# 3.2.8Class Geometry (几何计算类)

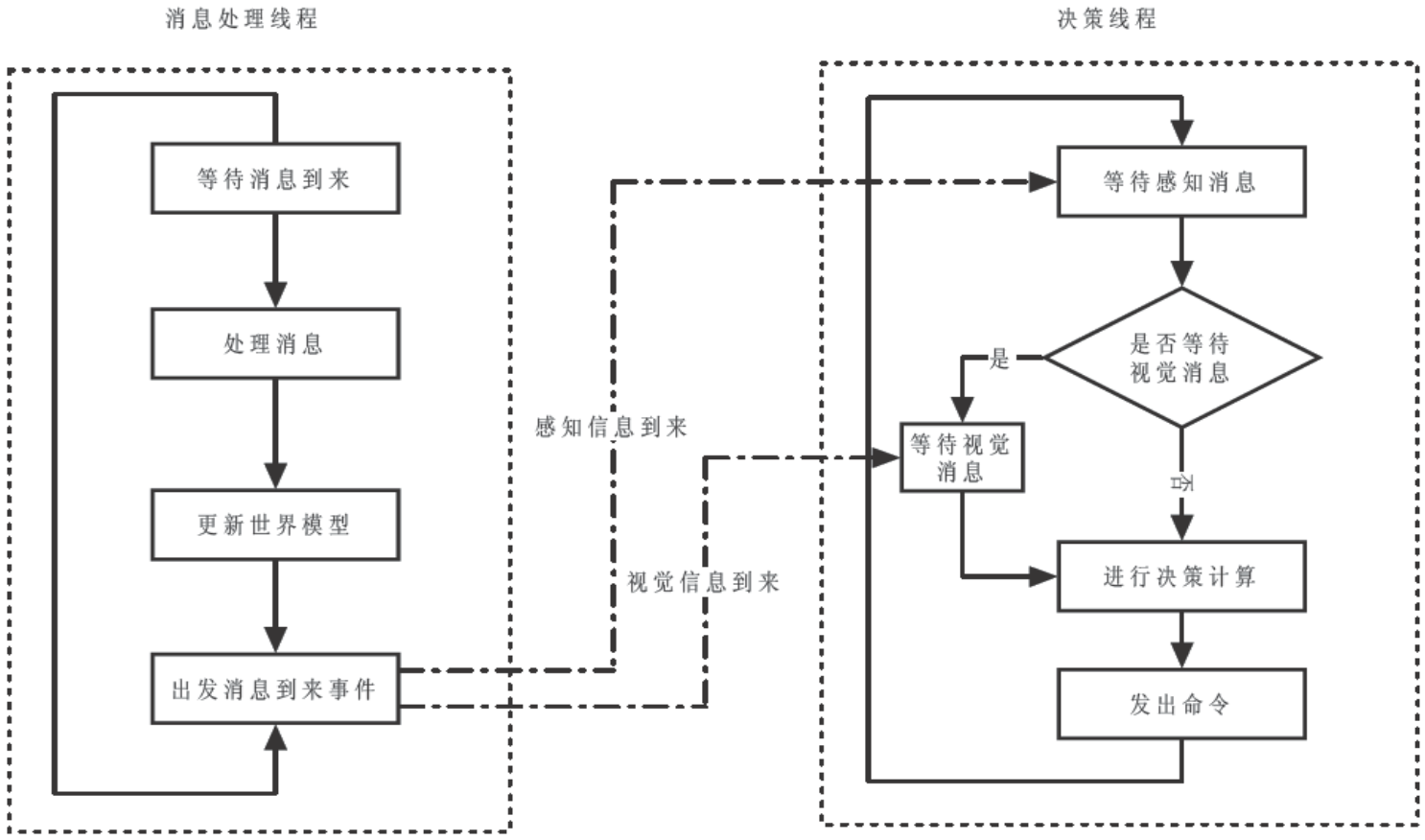
- Geometry类设计了一些与几何计算相关的静态方法。主要被BasicPlayer用来计算一些动作的执行细节。如：仿真程序中球员和球的速度衰减都遵循等比数列的。
- 例如：
  - **double getLengthGeomSeries (double dFirst, double dRatio, double dSum)** //给定几何序列（等比数列）的首项，公比，和。求序列的长度。
  - **double getSumGeomSeries (double dFirst, double dRatio, double dLength)** //给定几何序列（等比数列）的首项，公比，长度。求该序列的和。
  - double getSumInfGeomSeries (double dFirst, double dRatio) //无穷等比数列求和。
  - **double getFirstGeomSeries (double dSum, double dRatio, double dLength)** //给定几何序列（等比数列）的和，公比，长度。求该序列的首项。

# 3.4球员智能体程序控制流程

- 三线程控制结构
- 球员智能体执行例程

# 3.4.1 三线程结构

通常解决方法：



# 3.4.1三线程结构

{规划决策线程(主线程)}
创建感知反应线程；
创建行为线程；
while(true) do
WAIT(Ready)；
if 本周期已经进行过推理了
continue;
else
Lock(WM)；
由世界模型推理要执行的动作；
将动作重组为字符串发送给行为线程
Unlock(WM)；
end if
end while

{行为线程（子线程）}
while(true) do
在接受到Send信号量前阻塞
转化字符串为Server理解的行为格式
发送给Server
end while

{感知反应线程(子线程)}
while(true) do
在接收到来自仿真平台的网络数据之前阻塞；
if 接收到sense_body 消息
调整参数p；
将p 设置为定时器参数，定时器到期后SIGNAL(Ready),通知行为线程；
将动作重组为字符串发送给行为线程；
end if
Lock(WM)；
更新世界模型；
Unlock(WM)；
if 接收到see 消息
SIGNAL(Ready)；
end if
end while

# 3.4.2球员智能体执行例程

- **Formations fs( strFormations, FT_433_OFFENSIVE, iNr-1 );**
- **WorldModel wm( &ss, &cs, &fs );**
- **Connection c( strHost, iPort, MAX_MSG );**
- **ActHandler a( &c, &wm, &ss );**
- **SenseHandler s( &c, &wm, &ss, &cs );**
- //然后创建一个球员对象：
- **Player bp( &a, &wm, &ss, &cs, &fs, strTeamName, dVersion, iReconnect );**
- //球员对象创建完毕，创建一个监听线程负责对Server端的信息进行监听。
- **pthread_create( &sense, NULL, sense_callback , &s);**
- //一旦有信息来了则调用全局sense_callback函数进行信息处理。
- //在实际比赛进程中，一直调用Player类的**mainLoop方法**；
- **bp.mainLoop();**
- //直到Server与球员之间失去了连接（9s没有接到Server端的信息了），则这边主动放弃与Server的连接。
- **c.disconnect();**