**实验四 复杂的动作决策**

**1. 实验目的**

进一步了解 demeer5 并能熟悉的修改 demeer5 的内容以达到对场上的球员的控制：

1. 能理解 UVA 程序中原来的 demeer5 中的全部内容
2. 能通过修改 demeer5 中的具体函数内容实现对场上球员的控制
3. 能通过底层动作的简单组合控制场上队员做出一些复杂动作决策
4. 对 WorldModel 有初步的认识，学会在 WorldModel,basicplayer 里添加新函

数

**2. 实验设备**

硬件环境：PC

软件环境：操作系统linux

**3. 实验内容**

1. 在球队程序中找到 player.c 并打开;
2. 在 player.c 中找到 demmer5 函数;
3. 阅读此段程序，并结合 monitor 观察球员的具体行为（你将发现可以踢到球的球员会将球朝球门的方向踢去，而不能踢到球的队员中如果是离球最近的队员就去截球，否者则按阵型跑位）;

(4) 修改 demmer5 函数改变队员的行为具体步骤如下：

①在 demeer5 函数中找到

else if( WM->getFastestInSetTo( OBJECT\_SET\_TEAMMATES, OBJECT\_BALL, &iTmp )` == WM->getAgentObjectType() && !WM->isDeadBallThem() )

{ // 如果是最快到达球的队员

Log.log( 100, "I am fastest to ball; can get there in %d cycles", iTmp );

soc = intercept( false ); // 截球

if( soc.commandType == CMD\_DASH && // 当体力低时

WM->getAgentStamina().getStamina() <

SS->getRecoverDecThr()\*SS->getStaminaMax()+200 )

{

soc.dPower = 30.0 \* WM->getAgentStamina().getRecovery(); // 慢速移动

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) ); //脖子转向球

}

else // 当体力高时

{

ACT->putCommandInQueue( soc ); //正常移动速度

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );//脖子转向球

}

}

此函数的内容是，当球不可踢时，如果是离球最近的队员就执行截球命令（截球函数 intercept()在 BasicPlayer.c 中定义）观察在队员体力值小于多少时，带球速度会变慢。

②现在我们来通过修改函数，来改变非持球队员的决策，在以上函数之前加上此段代码：

else if(WM->getAgentObjectType()==OBJECT\_TEAMMATE\_9) //如果是 9 号队员

{

soc = SoccerCommand(CMD\_TURN,60); //转身体

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( alignNeckWithBody( ) ); //脖子随着身体一起转动

}

然后编译运行程序，观察球员的行为我们会发现当 9 号不持球时，身体在一直的转动（此动作可用来找球）

1. 在 WorldMoled 里填加状态函数

①打开 WorldModel.h，在里面预定义函数，即写入

bool isOpponentAtAngleEx( AngDeg angA , AngDeg angB ,double dDist )；

该函数用来判断当前球员角度在 angA~angB 之间距离小于 dDist 的范围内是否有对方队员。

②.找到并打开 WorldModel.c 在里面填加一个新函数

bool WorldModel::isOpponentAtAngleEx( AngDeg angA , AngDeg angB ,double dDist )

{

VecPosition posAgent = getAgentGlobalPosition(); VecPosition posOpp;

AngDeg angOpp;

int iIndex;

for( ObjectT o = iterateObjectStart( iIndex, OBJECT\_SET\_OPPONENTS ); o != OBJECT\_ILLEGAL;

o = iterateObjectNext ( iIndex, OBJECT\_SET\_OPPONENTS ) )

{

posOpp = getGlobalPosition( o );

angOpp = ( posOpp - posAgent ).getDirection() ;

if( angA<=angOpp && angOpp <=angB && posAgent.getDistanceTo( posOpp ) < dDist )

return true;

}

iterateObjectDone( iIndex ); return false;

}

③.将 if( WM->isBallKickable())内的内容改为：

if( WM->isBallKickable())

{

double ang = (VecPosition(52.5,0)-posAgent).getDirection(); if ( WM->isOpponentAtAngleEx(ang-45, ang, 6) )

ang+=45;

else if ( WM->isOpponentAtAngleEx(ang,ang+45,6) ) ang-=45;

SoccerCommand soc = dribble ( ang , DRIBBLE\_SLOW ); ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) );

}

任务：

然后编译运行程序，观察球员的行为，试分析球员的行为

根据以上描述完成练习：通过基本动作的组合实现球员的以下行为

①判断守门员的位置，朝球门空隙较大的一方射门，（通过在 WorldModel 里建立新状态来判断，球门哪一方空隙较大，守门员的位置为 VecPosition posGoalie = WM->getGlobalPosition(WM->getOppGoalieType());球门位置坐标为（52.5，0），可尝试朝（52.5，6.5）（52.5，-6.5）两点射门）

else if ( WM->isBallKickable())//if kickable//如果球已知，而且当前球在我脚下(可踢)

{

if(WM->isInTheirPenaltyArea(WM->getBallPos())) {//朝空隙大的一方射门

VecPosition posGoalie;

posGoalie = WM->getGlobalPosition(WM->getOppGoalieType());

if( WM->maxAngShoot( posGoalie, posAgent ))

soc = kickTo(VecPosition(52.5,6.0),SS->getBallSpeedMax());

else

soc = kickTo(VecPosition(52.5,-6.0),SS->getBallSpeedMax());

}

else

soc = dribble(0.0,DRIBBLE\_SLOW); //带球

ACT->putCommandInQueue( soc ); // 放入命令队列

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ) ); // 把脖子转向球，也就是一直看着球

Log.log( 100, "kick ball" );

}

**射门判断函数maxAngShoot**

**/\*以下这个函数可以向空隙较大的一方射门，采用夹角比较**

**\* 参数：守门员位置，自己位置**

**\* 返回值：bool型，true为向上踢，false为向下踢**

**\* 需要在worldmodel.h里添加内容**

**\*/**

bool WorldModel::maxAngShoot( VecPosition posGoalie, VecPosition posAgent )//朝较大的夹角射门

{

//VecPosition posGoalie;

AngDeg ang\_goalie, angup, angdown;

//posGoalie = WM->getGlobalPosition(WM->getOppGoalieType());

ang\_goalie = (posGoalie - posAgent).getDirection();

angup = (VecPosition(52.5,6.0) - posAgent).getDirection();

angdown = (VecPosition(52.5,-6.0) - posAgent).getDirection();

if(fabs(angup - ang\_goalie) > fabs(angdown - ang\_goalie))

return true;

else

return false;

}

②在 BasicPlay 里填加一个带球函数，要求如果无人阻挡（带球将要经过的路线附近没有对方球员）就朝球门方向带球，否则想办法避开对方球员带球前进（要求只要作出闪避的动作即可，不要求效果）。

打开 WorldModel.h，在里面预定义函数，即写入

bool isOpponentAtAngleEx( AngDeg angA , AngDeg angB ,double dDist )；

该函数用来判断当前球员角度在 angA~angB 之间距离小于 dDist 的范围内是否有对方队员。

找到并打开 WorldModel.c 在里面添加一个新函数：

bool WorldModel::isOpponentAtAngleEx( AngDeg angA , AngDeg angB ,double dDist )

{

VecPosition posAgent = getAgentGlobalPosition(); VecPosition posOpp;

AngDeg angOpp; int iIndex;

for( ObjectT o = iterateObjectStart( iIndex, OBJECT\_SET\_OPPONENTS ); o != OBJECT\_ILLEGAL;

o = iterateObjectNext ( iIndex, OBJECT\_SET\_OPPONENTS ) )

{

posOpp = getGlobalPosition( o );

angOpp = ( posOpp - posAgent ).getDirection() ;

if( angA<=angOpp && angOpp <=angB && posAgent.getDistanceTo( posOpp ) < dDist )

return true;

}

iterateObjectDone( iIndex ); return false;

}

在 playerteams.cpp 中添加

else if( WM->isBallKickable())

{

Circle cir(posAgent,2.5);

int num=WM->getNrInSetInCircle(OBJECT\_SET\_OPPONENTS,cir); if(num<2)

{

double ang = (VecPosition(52.5,0)-posAgent).getDirection(); if ( WM->isOpponentAtAngleEx(ang-45, ang, 6) )

ang+=45;

else if ( WM->isOpponentAtAngleEx(ang,ang+45,6) ) ang-=45;

SoccerCommand soc = dribble ( ang , DRIBBLE\_SLOW );

}

else

{

soc=dribble(0.0,DRIBBLE\_FAST);

}

ACT->putCommandInQueue( soc );

ACT->putCommandInQueue( turnNeckToObject( OBJECT\_BALL, soc ));

}

③尝试修改视觉函数使得球员能更多的获得场上信息（要求不影响球员的动作）。

先在 Basicplayer.h 中函数声明

SoccerCommand view1122333 (SoccerCommand soc );

在 Basicplayer.cpp 中 view1122333 (SoccerCommand soc )函数定义如下：

/\*将视觉分成 1122333 的视觉模式（“1”代表 60 度，“2”代表 120，“3”代表 180）\*/

SoccerCommand BasicPlayer::view1122333(SoccerCommand soc)

{

//7 --> 共分成 7 份

VecPosition posAgent = WM->getAgentGlobalPosition(); AngDeg angBody = WM->getAgentGlobalBodyAngle(); AngDeg angTurn=30;

switch( WM->getCurrentCycle()%7 )

{

case 0:

angTurn -= 60;

case 1:

ACT->putCommandInQueue(SoccerCommand(CMD\_CHANGEVIEW, VA\_NARROW, VQ\_HIGH));

ACT->putCommandInQueue(turnNeckToPoint(posAgent+VecPosition( 5, VecPosition::normalizeAngle(angBody+angTurn),POLAR), soc)); break;

case 2:

case 3:

ACT->putCommandInQueue(SoccerCommand(CMD\_CHANGEVIEW, VA\_NORMAL, VQ\_HIGH));

ACT->putCommandInQueue(turnNeckToPoint(posAgent+VecPosition(5, angBody, POLAR), soc));

break;

case 4:

case 5:

case 6:

ACT->putCommandInQueue(SoccerCommand(CMD\_CHANGEVIEW, VA\_WIDE, VQ\_HIGH));

break;

default:;

}

}

再在 playerTeams.cpp 球可踢的条件下增加

ACT->putCommandInQueue(view1122333(soc));

就可看到明显的视觉变化。