**实验四 复杂的动作决策**

**一、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。

**二、实验设备**

硬件环境：PC机；

软件环境：操作系统linux。

**三、实验内容**

**1.在WM中增添函数判断守门员位置，朝空隙较大的一方射门**

这里应该使用角度来比较空隙大小。将自身和守门员、球门上界、球门下界分别连线，比较夹角。将球向比较大的夹角踢。

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中添加函数，如果无人阻挡则向前带球，否则闪避带球。**

首先添加函数isOpponentAtAngleEx(),判断在视野中指定夹角和距离构成的扇形中是否存在对手球员。若没有则向前带球。闪避带球则是分别判断当前视野的左45°和右45°范围内有无对方球员。如有则向另一方向带球以实现躲避。

位置：worldmodelhighlevel.cpp

**//2020.5.7**

**/\*这个函数判断在扇区内有没有敌方球员，若有则ture 否则false**

**\***

**\*/**

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;

}

位置：basicplayer.cpp

SoccerCommand BasicPlayer::dribble\_Dodge(VecPosition posAgent)//闪避带球

{

SoccerCommand soc;

//Circle cir(posAgent,7);

//ObjectT o = WM->getClosestInSetTo(OBJECT\_SET\_OPPONENTS,posAgent);

//VecPosition p = WM->getGlobalPosition(o);

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

AngDeg ang = (VecPosition(52.5,0) - posAgent).getDirection();

soc = dribble(ang, DRIBBLE\_WITHBALL );

if( WM->isOpponentAtAngleEx( ang - 45, ang , 7 ) )

{

ang += 45;

}

else if( WM->isOpponentAtAngleEx( ang, ang + 45 , 7 ) )

{

ang -= 45;

}

soc = dribble( ang , DRIBBLE\_SLOW );

return soc;

}