// Strategy4Blue.cpp

//

#include "stdafx.h"

#include <string.h>

#include<ctime>

#include "Strategy4Blue.h"

Vector3D BallPosition[10];

int cnt = 0;

int state = 0;

int LastMajor = 1, LastAssist = 2, LastProtect = 4;

// Set the team name for blue side.

STRATEGY4BLUE\_API void SetBlueTeamName(char\* teamName)

{

// MUST change the name into your own team name!

strcpy(teamName, "Romantic");

}

//命名队名

// Set the positions and rotations of your team when your team places robots first.当你的队伍首先放置机器人时，设置你的队伍的位置和旋转。

// [IN] gameState : the state of game gameState:游戏状态

// [OUT] robots : robots data for placing,including position and rotation 用于放置的机器人数据，包括位置和旋转

STRATEGY4BLUE\_API void SetFormerRobots(PlayMode gameState, Robot robots[])

{

switch (gameState) {

case 1:

break;

case 2:

break;

case 3://我方半场上半自由球

robots[0].pos.x = 215;

robots[0].pos.y = 100;

robots[0].rotation = -90;

robots[1].pos.x = 140;

robots[1].pos.y = 85;//195 80

robots[1].rotation = 80;

robots[3].pos.x = 195;

robots[3].pos.y = 80;//198 80

robots[3].rotation = 130;

robots[2].pos.x = 168;

robots[2].pos.y = 85;//160 150

robots[2].rotation = 102;

robots[4].pos.x = 195;

robots[4].pos.y = 150;//140 50

robots[4].rotation = -180;

state = 1;

break;

case 4://我方半场下半自由球

robots[0].pos.x = 215;

robots[0].pos.y = 80;

robots[0].rotation = -90;

robots[1].pos.x = 140;

robots[1].pos.y = 95;//195 30

robots[1].rotation = -80;

robots[3].pos.x = 195;

robots[3].pos.y = 96;//198 100

robots[3].rotation = -130;

robots[2].pos.x = 168;

robots[2].pos.y = 98;//160 100

robots[2].rotation = -95;

robots[4].pos.x = 195;

robots[4].pos.y = 30;//140 130

robots[4].rotation = 180;

state = 1;

break;

case 5:

break;

case 6://我方开球

robots[0].pos.x = 215;

robots[0].pos.y = 90;

robots[0].rotation = -90;

robots[3].pos.x = 135;

robots[3].pos.y = 90;//180 90

robots[3].rotation = 180;

robots[2].pos.x = 130;

robots[2].pos.y = 72;//120 80

robots[2].rotation = 150;

robots[1].pos.x = 130;

robots[1].pos.y = 108;//125 100

robots[1].rotation = -150;

robots[4].pos.x = 100;

robots[4].pos.y = 95;//95 90

robots[4].rotation = -30;

state = 1;

break;

case 7://敌方点球

robots[0].pos.x = 215;

robots[0].pos.y = 90;

robots[0].rotation = -90;

robots[1].pos.x = 90;

robots[1].pos.y = 120;//90 160

robots[1].rotation = 180;

robots[2].pos.x = 90;

robots[2].pos.y = 160;//90 120

robots[2].rotation = 180;

robots[3].pos.x = 90;

robots[3].pos.y = 80;//90 80

robots[3].rotation = 180;

robots[4].pos.x = 90;

robots[4].pos.y = 50;//90 50

robots[4].rotation = 180;

state = 0;

break;

case 8:

break;

case 9:

break;

case 10:

break;

case 11:

break;

case 12://我方守门员开球

robots[0].pos.x = 215;

robots[0].pos.y = 90;

robots[0].rotation = 117.5;

robots[1].pos.x = 160;

robots[1].pos.y = 40;//190 100 13310085

robots[1].rotation = 90;

robots[2].pos.x = 135;

robots[2].pos.y = 80;//170 65

robots[2].rotation = 85;//160 30 90

robots[3].pos.x = 185;

robots[3].pos.y = 75;//150 40

robots[3].rotation = 180;

robots[4].pos.x = 218;//130;

robots[4].pos.y = 131;//100;//130 130

robots[4].rotation = 180;//85;

state = 0;

break;

}

}

// Set the positions and rotations of your team when your team places robots last. 当你的队伍最后放置机器人时，设置你的队伍的位置和旋转。

// [IN] gameState : the state of game gameState:游戏状态

// [IN] formerRobots : the data of first placing team formerRobots:第一名队伍的数据

// [IN] ball : the position of the ball 球的位置

// [OUT] laterRobots : robots data for placing,including position and rotation laterRobots:放置机器人的数据，包括位置和旋转

STRATEGY4BLUE\_API void SetLaterRobots(PlayMode gameState, Robot formerRobots[],

Vector3D ball, Robot laterRobots[])

{

// This is just for a demo. You may set your own data.

switch (gameState) {

case 1://敌方半场上半场自由球

laterRobots[0].pos.x = 215;

laterRobots[0].pos.y = 90;

laterRobots[0].rotation = -90;

laterRobots[3].pos.x = 120;

laterRobots[3].pos.y = 100;//155 130

laterRobots[3].rotation = 180;

laterRobots[2].pos.x = 120;

laterRobots[2].pos.y = 155;//140 110

laterRobots[2].rotation = 180;

laterRobots[1].pos.x = 120;

laterRobots[1].pos.y = 60;//95 80

laterRobots[1].rotation = 0;

laterRobots[4].pos.x = 80;

laterRobots[4].pos.y = 150;//120 90

laterRobots[4].rotation = 180;

state = 1;

break;

case 2://敌方半场下半场自由球

laterRobots[0].pos.x = 215;

laterRobots[0].pos.y = 90;

laterRobots[0].rotation = -90;

laterRobots[3].pos.x = 120;

laterRobots[3].pos.y = 80;//165 80

laterRobots[3].rotation = 180;

laterRobots[2].pos.x = 120;

laterRobots[2].pos.y = 120;//130 60

laterRobots[2].rotation = 180;

laterRobots[1].pos.x = 120;

laterRobots[1].pos.y = 40;//95 100

laterRobots[1].rotation = 180;

laterRobots[4].pos.x = 85;

laterRobots[4].pos.y = 30;//85 30

laterRobots[4].rotation = 180;

state = 1;

break;

case 3:

break;

case 4:

break;

case 5://敌方开球

laterRobots[0].pos.x = 215;

laterRobots[0].pos.y = 90;

laterRobots[0].rotation = -90;

laterRobots[3].pos.x = 190;

laterRobots[3].pos.y = 85;//190 90

laterRobots[3].rotation = 180;

laterRobots[2].pos.x = 130;

laterRobots[2].pos.y = 110;//150 90

laterRobots[2].rotation = -135;

laterRobots[1].pos.x = 130;

laterRobots[1].pos.y = 70;//130 60

laterRobots[1].rotation = 135;

laterRobots[4].pos.x = 138;

laterRobots[4].pos.y = 90;//130 120

laterRobots[4].rotation = 180;

state = 1;

break;

case 6:

break;

case 7:

break;

case 8://我方点球

laterRobots[0].pos.x = 215;

laterRobots[0].pos.y = 90;

laterRobots[0].rotation = -90;

laterRobots[1].pos.x = 180;

laterRobots[1].pos.y = 90;//180 90

laterRobots[1].rotation = 180;

laterRobots[2].pos.x = 130;

laterRobots[2].pos.y = 110;//115 110

laterRobots[2].rotation = 180;

/\*laterRobots[4].pos.x = 130;

laterRobots[4].pos.y = 70;//115 70

laterRobots[4].rotation = 180;\*/

laterRobots[4].pos.x = 114;

laterRobots[4].pos.y = 40;//115 70

laterRobots[4].rotation = 145.7;

/\*

laterRobots[3].pos.x = 94.1;

laterRobots[3].pos.y = 110.75;//66 98.5

laterRobots[3].rotation = -158.9;\*/

/\*

laterRobots[3].pos.x = 93.59;

laterRobots[3].pos.y = 70.45;//66 98.5

laterRobots[3].rotation = 158.9;\*/

laterRobots[3].pos.x = 37.5;

laterRobots[3].pos.y = 175;//66 98.5

laterRobots[3].rotation = -90;

state = 0;

break;

case 9:

break;

case 10:

break;

case 11://敌方守门员开球

laterRobots[0].pos.x = 215;

laterRobots[0].pos.y = 90;

laterRobots[0].rotation = -90;

laterRobots[4].pos.x = 140;

laterRobots[4].pos.y = 70;//140 50

laterRobots[4].rotation = -160;

laterRobots[2].pos.x = 150;

laterRobots[2].pos.y = 145;//140 130

laterRobots[2].rotation = 180;

laterRobots[3].pos.x = 185;

laterRobots[3].pos.y = 28;//120 60

laterRobots[3].rotation = 180;

laterRobots[1].pos.x = 140;

laterRobots[1].pos.y = 100;//120 90

laterRobots[1].rotation = -150;

state = 0;

break;

case 12:

break;

}

}

// Set the position of the ball when your team get the Goal Kick. 当你的球队得到进球球时，设置球的位置。

// [IN] gameState : the state of game gameState:游戏状态

// [IN] pBall : the position of the ball pBall:球的位置

STRATEGY4BLUE\_API void SetBall(PlayMode gameState, Vector3D\* pBall)

{

// This is just for a demo. You may set your own data. 这只是一个演示。你可以设置你自己的数据。

if (PM\_GoalKick\_Blue == gameState)

{

pBall->x = 210;

pBall->y = 110;

}

}

// Strategy for your team using centimeter unit, cartesian coordinate system. 团队策略使用厘米单位，平面直角坐标系。

// The origin is at the left bottom of the field. 原点位于字段的左下方。

STRATEGY4BLUE\_API void RunStrategy(Environment\* pEnv)

{

/\*

// This is just for a demo. YOU MUST USE YOUR OWN STRATEGY HERE! 这只是一个演示。在这里你必须使用自己的策略!

srand((unsigned int)time(NULL));

int num = rand() % 100 + 1;

PlayMode gameState;

//if (state==1) {

int tid1 = 1;

int tid2 = 2;

int tid3 = 4;

int temp;

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D player1 = pEnv->home[tid1].pos;

Vector3D player2 = pEnv->home[tid2].pos;

Vector3D player3 = pEnv->home[tid3].pos;

BallPosition[++cnt] = currentBall;

if (cnt == 5)

{

cnt = 0;

}

if (currentBall.x <= 45)

{

tid1 = LastMajor;

tid2 = LastAssist;

tid3 = LastProtect;

major\_Attacking(pEnv, tid1, tid2, tid3);

assist\_Attacking(pEnv, tid1, tid2, tid3);

intercept\_protect(pEnv, tid1, tid2, tid3);

LastMajor = tid1;

LastAssist = tid2;

LastProtect = tid3;

}

else if (currentBall.x >= 165)

{

tid1 = LastMajor;

tid2 = LastAssist;

tid3 = LastProtect;

major\_Attacking(pEnv, tid1, tid2, tid3);

assist\_Attacking(pEnv, tid1, tid2, tid3);

intercept\_protect(pEnv, tid1, tid2, tid3);

LastMajor = tid1;

LastAssist = tid2;

LastProtect = tid3;

}

else

{

int id\_array[10] = { 1, 2, 4 };

int temp\_right[5];

int n = 0;

for (int i = 0; i < 3; ++i)

if (currentBall.x < pEnv->home[id\_array[i]].pos.x)

temp\_right[++n] = id\_array[i];

if (n == 0)

{

tid1 = LastMajor;

tid2 = LastAssist;

tid3 = LastProtect;

major\_Attacking(pEnv, tid1, tid2, tid3);

assist\_Attacking(pEnv, tid1, tid2, tid3);

intercept\_protect(pEnv, tid1, tid2, tid3);

LastMajor = tid1;

LastAssist = tid2;

LastProtect = tid3;

}

else

{

double minx = 1000;

int id\_player;

for (int i = 1; i <= n; ++i)

{

Vector3D temp\_player = pEnv->home[temp\_right[i]].pos;

double temp\_distance = calc\_distance(temp\_player.x, temp\_player.y, currentBall.x, currentBall.y);

if (temp\_distance < minx)

{

minx = temp\_distance;

id\_player = temp\_right[i];

}

}

tid1 = id\_player;

tid2 = tid3 = -1;

for (int i = 0; i < 3; ++i)

{

if (id\_array[i] != tid1)

{

if (tid2 == -1) tid2 = id\_array[i];

else tid3 = id\_array[i];

}

}

major\_Attacking(pEnv, tid1, tid2, tid3);

assist\_Attacking(pEnv, tid1, tid2, tid3);

intercept\_protect(pEnv, tid1, tid2, tid3);

LastMajor = tid1;

LastAssist = tid2;

LastProtect = tid3;

}

}

Goalie(pEnv);

CenterDefender1(pEnv, 3);

//}

else {\*/

//nopushtime(pEnv);

Goalie(pEnv);

LeftWing(pEnv, 1);

RightWing(pEnv, 2);

CenterDefender1(pEnv, 3);

CenterAttacker(pEnv, 4);

/\*}

Vector3D ball = pEnv->currentBall.pos;

Vector3D player1 = pEnv->home[3].pos;

if (ball.x < 40 && player1.x < 80) {

//if (gameState == PM\_PenaltyKick\_Blue)

//{

plent(pEnv, 3, 4);

}

//Goalie(pEnv);

//plent(pEnv, 3, 4);\*/

}

void plent(Environment\* pEnv, int id1, int id2) {

Vector3D ball = pEnv->currentBall.pos;

Vector3D player1 = pEnv->home[id1].pos;

Vector3D player2 = pEnv->home[id2].pos;

if (ball.x < 40 && player1.x < 80) {

Position(pEnv, id1, 37.5, 100, 1);

Position(pEnv, id2, 0,70);

Velocity(&(pEnv->home[id2]), 125, 125);

stop2(pEnv, id1, 37.5, 90);

if (player2.y >= 75 && player2.y <= 85)

Velocity(&(pEnv->home[id2]), 125, 75);

}

}

// #############################################################################################

// #############################################################################################

void CenterDefender2(Environment\* pEnv, int id1) {

Vector3D ball = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id1].pos;

if (ball.x < 40 && player.x < 100) {

Penalty\_Kike1(pEnv, 3);

if (pEnv->currentBall.pos.x > 50)

CenterDefender(pEnv, 3, 0);

}

else {

CenterDefender(pEnv, 3, 0);

}

}

void CenterDefender(Environment\* pEnv, int id1, int id2) //id1--后卫 id2--守门员

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D player3 = pEnv->home[id1].pos; // 后卫

Vector3D player0 = pEnv->home[id2].pos; // 守门员

double A[2] = { 0 };

PredictBall\_cnt(pEnv, 3, A);

// 需要预处理

if (A[0] >= 190)

A[0] = 190;

double D = calc\_distance(ball.x, ball.y, player3.x, player3.y);

double s\_player = 1e10; // 记录间距

int d\_player = -1; // 记录编号

// 求得敌方球员中距离球最近的距离

for (int i = 1; i <= 4; i++) {

double dnt = calc\_distance(ball.x, ball.y, pEnv->opponent[i].pos.x, pEnv->opponent[i].pos.y);

if (dnt < s\_player) {

s\_player = dnt;

d\_player = i;

}

}

Vector3D playerd = pEnv->opponent[d\_player].pos;

double x1 = playerd.x;

double y1 = playerd.y;

// 边线区域

if ((ball.y > 0 && ball.y < 8) || (ball.y > 172 && ball.y < 180))

{

if (ball.x < 150) {

Position(pEnv, id1, 170, fabs(ball.y - 10));

}

else if (ball.x >= 190) {

Position(pEnv, id1, 190, fabs(ball.y - 30));

stop(pEnv, id1, 190, fabs(ball.y - 30));

}

if (fabs(player3.y - ball.y) < 5) {

if (ball.x >= 85)

Position(pEnv, id1, ball.x, ball.y);

}

}

else // 中间位置

{

// 定点

if (ball.x <= 110 && A[1] <= 110) {

Position(pEnv, id1, 190, fabs(ball.y - 15));

stop(pEnv, id1, 190, fabs(ball.y - 15));

}

else if (ball.x >= 190) {

Position(pEnv, id1, 190, fabs(ball.y - 30));

stop(pEnv, id1, 190, fabs(ball.y - 30));

}

else {

Position(pEnv, id1, 190, A[1]);

stop(pEnv, id1, 190, A[1]);

// 这里需要关注速度

if (fabs(player3.x - ball.x) < 25 && ball.x + 4 < player3.x)

Position(pEnv, id1, A[0], A[1]);

}

if (player3.x > 165 && player3.x < 202 && D < 15) {

if (ball.x + 2 < player3.x) // 限制向内旋转

if (ball.y - 2 > player3.y)

Velocity(&(pEnv->home[id1]), -120, 120);

else if (ball.y + 2 < player3.y)

Velocity(&(pEnv->home[id1]), 120, -120);

}

}

}

void speed(Environment\* pEnv, int id1, double x1, double y1) {

Vector3D player = pEnv->home[id1].pos; // 后卫

double x2 = player.x;

double y2 = player.y;

double D = calc\_distance(x1, y1, x2, y2);

if (D < 15 && player.x > x1)

Velocity(&(pEnv->home[id1]), 125, 125);

}

void rotate1(Environment\* pEnv, int id)

{

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id].pos;

double D = calc\_distance(player.x, player.y, currentBall.x, currentBall.y);

if (currentBall.y > player.y && D <= 8)

{

Velocity(&(pEnv->home[id]), -125, 125);

}

}

void rotate2(Environment\* pEnv, int id)

{

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id].pos;

double D = calc\_distance(player.x, player.y, currentBall.x, currentBall.y);

if (currentBall.y < player.y && D <= 8)

{

Velocity(&(pEnv->home[id]), 125, -125);

}

}

void assist\_Attacking(Environment\* pEnv, int id1, int id2, int id3)

{

double A[3] = { 0 };

Vector3D player1 = pEnv->home[id1].pos;

Vector3D player2 = pEnv->home[id2].pos;

Vector3D player3 = pEnv->home[id3].pos;

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D predictedBall = pEnv->predictedBall.pos;

PredictBall\_cnt(pEnv, 3, A);

double dist = calc\_distance(player2.x, player2.y, currentBall.x, currentBall.y);

if (currentBall.x <= 35)

{

if (currentBall.y < 50)

{

Position(pEnv, id2, 40, currentBall.y + 60);//80参数可修改

}

else if (currentBall.y > 130)

{

Position(pEnv, id2, 40, currentBall.y - 60);//80参数可修改

}

else//敌方守门框

{

double k1 = (player1.y - currentBall.y) / (player1.x - currentBall.x);

double k2 = (player2.y - currentBall.y) / (player2.x - currentBall.x);

double y1\_dist = player1.y - player1.x \* k1;

double y2\_dist = player2.y - player2.x \* k2;

int flag = 0;

if (y1\_dist >= 70 && y1\_dist <= 110) flag = 1;

if (y2\_dist < 70 || y2\_dist > 110) flag = 1;

if (!flag && currentBall.x < player2.x)

{

if (dist < 10) {

Position(pEnv, id2, currentBall.x, currentBall.y);

}

else {

Position(pEnv, id2, A[0], A[1]);

}

}

else

{

if (currentBall.y < 90)

{

Position(pEnv, id2, 40, 130);

}

else

{

Position(pEnv, id2, 40, 50);

}

}

}

}

else if (currentBall.x > 35 && currentBall.x <= 110)

{

if (currentBall.y < 30)

{

Position(pEnv, id2, player1.x + 20, max(player1.y + 20, 25));//参数可调 15 20

}

else if (currentBall.y > 150)

{

Position(pEnv, id2, player1.x + 20, min(player1.y - 20, 155));//参数可调

}

else

{

Position(pEnv, id2, currentBall.x + 20, currentBall.y + 20);//参数可调

}

}

else if (currentBall.x > 110 && currentBall.x < 165)

{

if (currentBall.y < 30)

{

Position(pEnv, id2, min(player1.x + 20, 160), max(player1.y + 20, 25));//参数可调 y15

}

else if (currentBall.y > 150)

{

Position(pEnv, id2, min(player1.x + 20, 160), min(player1.y - 20, 155));//参数可调 y15

}

else

{

Position(pEnv, id2, min(player1.x + 20, 160), currentBall.y + 20);//参数可调

}

}

else

{

Position(pEnv, id2, 170, 50);

}

}

void intercept\_protect(Environment\* pEnv, int id1, int id2, int id3)

{

Vector3D player1 = pEnv->home[id1].pos;

Vector3D player2 = pEnv->home[id2].pos;

Vector3D player3 = pEnv->home[id3].pos;

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D predictedBall = pEnv->predictedBall.pos;

int dist = calc\_distance(player3.x, player3.y, currentBall.x, currentBall.y);

if (currentBall.x <= 35)

{

if (currentBall.y < 50)

{

Position(pEnv, id3, player1.x + 20, max(player1.y + 20, 30));

}

else if (currentBall.y > 130)

{

Position(pEnv, id3, player1.x + 20, min(player1.y - 20, 150));

}

else

{

Position(pEnv, id3, 45, currentBall.y \* 2 - player1.y);

}

}

else if (currentBall.x > 35 && currentBall.x <= 110)

{

if (currentBall.y < 90)

{

Position(pEnv, id3, currentBall.x + 35, max(currentBall.y + 50, 55));

}

else

{

Position(pEnv, id3, currentBall.x + 35, min(currentBall.y - 50, 125));

}

}

else if (currentBall.x > 110 && currentBall.x < 165)

{

if (currentBall.y < 30)

{

Position(pEnv, id3, min(player1.x + 35, 150), min(player1.y + 50, 55));

}

else if (currentBall.y > 150)

{

Position(pEnv, id3, min(player1.x + 35, 150), min(player1.y - 50, 125));

}

else

{

if (currentBall.y < 90)

{

Position(pEnv, id3, min(currentBall.x + 30, 150), currentBall.y + 30);

}

else

{

Position(pEnv, id3, min(currentBall.x + 30, 150), currentBall.y - 30);

}

}

}

else

{

Position(pEnv, id3, 165, 130);

}

}

double calc\_slope(double x1, double y1, double x2, double y2) { // 计算两点间的斜率

return (y2 - y1) \* 1.0 / (x2 - x1);

}

void major\_Attacking(Environment\* pEnv, int id1, int id2, int id3) // 主攻球员采取的策略

{

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D player1 = pEnv->home[id1].pos;

Vector3D player2 = pEnv->home[id2].pos;

Vector3D player3 = pEnv->home[id3].pos;

Vector3D predictedBall = pEnv->predictedBall.pos;

Vector3D opponent\_player0 = pEnv->opponent[0].pos;

int dist = calc\_distance(player1.x, player1.y, currentBall.x, currentBall.y);//球员与球的距离

int dist1 = calc\_distance(currentBall.x, currentBall.y, predictedBall.x, predictedBall.y);

int dist2 = calc\_distance(opponent\_player0.x, opponent\_player0.y, player1.x, player1.y);

double k1 = calc\_slope(currentBall.x, currentBall.y, player1.x, player1.y);

double y\_dist1 = player1.y - k1 \* player1.x; // 球与主攻机器人连线的延长线与门框的交点

if (player1.x <= 35) // 球在对方外门框线以内

{

if (currentBall.y <= 45) // 球在下边线区域

{

if (player1.x < currentBall.x) //如下采取的是截球

{

if (player1.y < currentBall.y)//左下角

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125); // 逆时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 8, currentBall.y + 8);

}

}

else//左上角

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45); // 顺时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 8, currentBall.y - 8);// y + 8

}

}

}

else

{

if ((y\_dist1 >= 70 && y\_dist1 <= 110)) // 如果交点在门框内

{

Position(pEnv, id1, currentBall.x, currentBall.y); // 直接采取冲球的方式

}

else

{

if (currentBall.x > 15) {

if (player1.y < currentBall.y) //右下角

{

if (dist < 13)// 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);//x + 10, y - 10

}

}

else//右上角

{

if (dist < 13) // 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y - 10);

}

}

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

}

else if (currentBall.y >= 135)//球在上边线区域

{

if (player1.x < currentBall.x)

{

if (player1.y < currentBall.y)//左下角

{

if (dist < 13)

{

Velocity(&(pEnv->home[id1]), 45, 125); // 逆时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

else//左上角

{

if (dist < 13)

{

Velocity(&(pEnv->home[id1]), 125, 45); // 顺时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

}

else

{

if ((y\_dist1 >= 70 && y\_dist1 <= 110)) // 如果交点在门框内

{

Position(pEnv, id1, currentBall.x, currentBall.y); // 直接采取冲球的方式

}

else

{

if (currentBall.x > 15) {

if (player1.y < currentBall.y) // 右下角

{

if (dist < 13)// 修改前18

{

Velocity(&(pEnv->home[id1]), 45, 125);//逆

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

else//右上角

{

if (dist < 13) // 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

}

else // 球在中间区域

{

if (y\_dist1 >= 70 && y\_dist1 <= 110) // 当交点在门框内时

{

if (currentBall.x < player1.x) // 当机器人在球的右边，就撞球

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else // 当机器人在球的左边，应该采取截球

{

if (dist < 18) // 修改前18

{

if (player1.y > currentBall.y)

{

Velocity(&(pEnv->home[id1]), 48, 125);

}

else

{

Velocity(&(pEnv->home[id1]), 125, 48);

}

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

}

else // 交点不在门框时，判断对方守门员位置，调整自己位置

{

/\*if (opponent\_player0.y <= 90) Position(pEnv, id1, max(20, currentBall.x), currentBall.y - 5);

else Position(pEnv, id1, max(20, currentBall.x), currentBall.y + 5);\*/

Position(pEnv, id1, currentBall.x, currentBall.y);

/\*if (player1.y < currentBall.y) // 旋转进球 修改前

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}\*/

}

}

}

else if (currentBall.x <= 110) // 当球在对方半场

{

if (currentBall.y >= 50 && currentBall.y <= 130) // 球在中间区域

{

if (currentBall.x < player1.x) // 机器人在球的左边

{

if (player1.y < currentBall.y)

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}

}

else

{

//Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);修改前

if (player1.y < currentBall.y)

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}

}

}

else // 球在两边区域

{

if (currentBall.y <= 15) //

{

if (dist < 15)

{

Velocity(&(pEnv->home[id1]), 125, 15);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 15);

}

}

else if (currentBall.y >= 165)

{

if (dist < 15)

{

Velocity(&(pEnv->home[id1]), 25, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 15);

}

}

else if (currentBall.y <= 50)

{

if (currentBall.x < player1.x)

{

if (dist < 5)

{

Velocity(&(pEnv->home[id1]), 125, -15);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 3);

}

}

else

{

Position(pEnv, id1, min(currentBall.x + 15, 105), currentBall.y + 15);

}

}

else

{

if (currentBall.x < player1.x)

{

if (dist < 5)

{

Velocity(&(pEnv->home[id1]), -15, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 3);

}

}

else

{

Position(pEnv, id1, min(currentBall.x + 15, 105), currentBall.y - 15);

}

}

}

}

else if (currentBall.x <= 185)//考虑边界 后卫的判断

{

Vector3D player\_CD = pEnv->home[3].pos;

int dist3 = calc\_distance(player\_CD.x, player\_CD.y, currentBall.x, currentBall.y);

if (currentBall.y <= 10)

{

if (dist3 <= 10)

{

Position(pEnv, id1, currentBall.x + 15, currentBall.y + 8);// 10 8

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

else if (currentBall.y >= 170)

{

if (dist3 <= 10)

{

Position(pEnv, id1, currentBall.x + 15, currentBall.y - 8); // 8 -5

}

else

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

else

{

if (currentBall.x <= player1.x)

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else

{

Position(pEnv, id1, currentBall.x + 12, min(currentBall.y + 10, 170));// 7

}

}

}

else

{

if (currentBall.y <= 50)

{

if (currentBall.x <= player1.x)//==

{

Position(pEnv, id1, min(currentBall.x, 185), currentBall.y);

}

else

{

if (currentBall.y > player1.y)

{

Position(pEnv, id1, currentBall.x - 10, max(currentBall.y - 15, 10));

}

else

{

Position(pEnv, id1, 210, 40);

rotate2(pEnv, id1);

}

}

}

else if (currentBall.y >= 130)

{

if (currentBall.x <= player1.x)

{

Position(pEnv, id1, min(currentBall.x, 185), currentBall.y);

}

else

{

if (currentBall.y < player1.y)

{

Position(pEnv, id1, currentBall.x - 10, min(currentBall.y + 15, 170));

}

else

{

Position(pEnv, id1, 210, 140);

rotate1(pEnv, id1);

}

}

}

else

{

if (currentBall.x < player1.x)

{

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else

{

if (currentBall.y > 90)

{

Position(pEnv, id1, 180, 110);

}

else

{

Position(pEnv, id1, 180, 70);

}

}

}

}

}

// 新增：守门员位置限制辅助函数

void ClampGoaliePosition(double& x, double& y) {

if (x > 218) x = 218;

if (x < 212) x = 212;

if (y < 60) y = 60;

if (y > 120) y = 120;

}

void Goalie(Environment\* pEnv)

{

double A[2] = { 0 };

Vector3D CurrentBall = pEnv->currentBall.pos;

Vector3D player0 = pEnv->home[0].pos;

double ballx = CurrentBall.x;

double bally = CurrentBall.y;

double Y = calculateAngleBisector(216, CurrentBall.x, CurrentBall.y, 216, 70, 216, 110);

double D = calc\_distance(ballx, bally, player0.x, player0.y);

double ball\_player = 1e10;

for (int i = 1; i <= 4; i++) {

double dnt = calc\_distance(ballx, bally, pEnv->opponent[i].pos.x, pEnv->opponent[i].pos.y);

if (dnt < ball\_player) {

ball\_player = dnt;

}

}

PredictBall\_cnt(pEnv, 3, A);

if (ballx < 180) // 对方半场

{

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

else if (ballx >= 180) {

int flag = 0;

for (int i = 1; i <= 4; i++) {

if (pEnv->opponent[i].pos.x > 165) {

flag++;

}

}

if (flag >= 1) {

if (A[1] > 120) {

if (bally > 120 && ballx < 215) {

double tx = 215, ty = (Y + 90) / 2 + 5;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 118;

ClampGoaliePosition(tx, ty);

Velocity(&(pEnv->home[0]), 120, 125);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty - 3);

}

}

else if (A[1] < 60) {

if (bally < 60 && ballx < 215) {

double tx = 215, ty = (Y + 90) / 2 - 5;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 67;

ClampGoaliePosition(tx, ty);

Velocity(&(pEnv->home[0]), 125, 125);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

}

else {

double tx = 215, ty = A[1];

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

else { // 无敌人情况--用于开球

if (D < ball\_player && bally > player0.y && bally < 120) {

double tx = ballx, ty = bally;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

}

else {

if (ballx < 190) {

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else {

if (A[1] > 120) {

if (bally > 120 && ballx < 217) {

double tx = 215, ty = Y + 5;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 112.8;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

}

else if (A[1] < 60) {

if (bally < 60 && ballx < 217) {

double tx = 215, ty = Y - 5;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 67.2;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

}

else {

double tx = 215, ty = A[1];

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

}

}

}

}

void Goalie1(Environment\* pEnv)//守门员 by Ai

{

double vc = 125;

double A[2] = { 0 };

PredictBall\_cnt(pEnv, 3, A);

double B\_G\_Angle = atan(pEnv->currentBall.pos.y - pEnv->home[0].pos.y) / (pEnv->currentBall.pos.x - pEnv->home[0].pos.x);

double Ball\_Angle = atan((pEnv->lastBall.pos.y - pEnv->currentBall.pos.y) / (pEnv->lastBall.pos.x - pEnv->currentBall.pos.x));

double Ball\_FB = pEnv->lastBall.pos.x - pEnv->currentBall.pos.x;

double Ball\_UD = pEnv->lastBall.pos.y - pEnv->currentBall.pos.y;

double Ball\_Goalie\_x = fabs(pEnv->currentBall.pos.x - pEnv->home[0].pos.x);

double Ball\_Goalie\_y = fabs(pEnv->currentBall.pos.y - pEnv->home[0].pos.y);

double Pre\_Ball\_Angle = atan((A[1] - pEnv->currentBall.pos.y) / (A[0] - pEnv->currentBall.pos.x));

double Y = calculateAngleBisector(216, pEnv->currentBall.pos.x, pEnv->currentBall.pos.y, 216, 70, 216, 110);

double ball\_player = 1e10;

int n = 0;

for (int i = 1; i <= 4; i++) {

double dnt = calc\_distance(pEnv->currentBall.pos.x, pEnv->currentBall.pos.y, pEnv->opponent[i].pos.x, pEnv->opponent[i].pos.y);

if (dnt < ball\_player) {

ball\_player = dnt;

n = i;

}

}

int flag = 0;

for (int i = 1; i <= 4; i++) {

if (pEnv->opponent[i].pos.x > 165) {

flag++;

}

}

if (pEnv->currentBall.pos.x == pEnv->lastBall.pos.x)

{

if (pEnv->currentBall.pos.x <= 200 && pEnv->currentBall.pos.x > 180)

{

if (106 >= pEnv->currentBall.pos.y && 76 <= pEnv->currentBall.pos.y)

{

double tx = 215, ty = A[1];

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

}

}

else

{

double tx = pEnv->currentBall.pos.x, ty = pEnv->currentBall.pos.y;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

}

}

else

{

if (pEnv->currentBall.pos.x < 150)

{

double tx = 215, ty = 90;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

else if (flag > 1)

{

if (pEnv->currentBall.pos.x < 185)

{

vc = 100;

if (pEnv->currentBall.pos.y > 140)

{

double tx = 215, ty = 100;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

else if (pEnv->currentBall.pos.y < 40)

{

double tx = 215, ty = 80;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

else

{

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

}

else

{

if (pEnv->currentBall.pos.y <= 55)

{

double tx = 215, ty = 75;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

else if (125 <= pEnv->currentBall.pos.y)

{

double tx = 215, ty = 105;

ClampGoaliePosition(tx, ty);

Position\_Goalie(pEnv, 0, tx, ty, vc);

stop2(pEnv, 0, tx, ty);

}

else

{

if (80 < pEnv->currentBall.pos.y && pEnv->currentBall.pos.y < 100)

{

double tx = 215, ty = pEnv->currentBall.pos.y;

ClampGoaliePosition(tx, ty);

Angle(&pEnv->home[0], B\_G\_Angle);

Position\_Goalie(pEnv, 0, tx, ty, vc);

}

else if (65 < pEnv->currentBall.pos.y && pEnv->currentBall.pos.y <= 80)

{

double tx = 215, ty = pEnv->currentBall.pos.y + 5;

ClampGoaliePosition(tx, ty);

Angle(&pEnv->home[0], B\_G\_Angle);

Position\_Goalie(pEnv, 0, tx, ty, vc);

}

else if (100 <= pEnv->currentBall.pos.y && pEnv->currentBall.pos.y <= 110)

{

double tx = 215, ty = pEnv->currentBall.pos.y - 5;

ClampGoaliePosition(tx, ty);

Angle(&pEnv->home[0], B\_G\_Angle);

Position\_Goalie(pEnv, 0, tx, ty, vc);

}

}

}

}

else

{

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop(pEnv, 0, tx, ty);

}

}

}

void Position\_Goalie(Environment\* pEnv, int id, double x, double y, double vc)//守门员

{

double tx = x, ty = y;

ClampGoaliePosition(tx, ty);

Position\_Goalie(&(pEnv->home[id]), tx, ty, 110);

}

double stop3(Environment\* pEnv, int id1, double x1, double y1)//by Ai

{

Vector3D player = pEnv->home[id1].pos;

double x2 = player.x;

double y2 = player.y;

double D = calc\_distance(x1, y1, x2, y2);

if (id1 == 0) //速度缓冲

{

if (D < 1)

{

return 0;//速度

}

else if (D < 14)

{

return 80;

}

else if (D < 5)

{

return 50;

}

else

{

return 125;

}

}

}

void Position\_Goalie(Robot\* robot, double x, double y, double vc)

{

double desired\_angle = 0, theta\_e = 0, d\_angle = 0;

double vl = 0, vr = 0;

double dx, dy, d\_e, Ka = 0.3;

dx = x - robot->pos.x;

dy = y - robot->pos.y;

d\_e = sqrt(dx \* dx + dy \* dy);

if (dx == 0 && dy == 0)

desired\_angle = 90;

else

desired\_angle = 180. / PI \* atan2((double)(dy), (double)(dx));

theta\_e = desired\_angle - robot->rotation;

while (theta\_e > 180) theta\_e -= 360;

while (theta\_e < -180) theta\_e += 360;

if (theta\_e > 95 || theta\_e < -95)

{

theta\_e += 180;

if (theta\_e > 180)

theta\_e -= 360;

if (theta\_e > 80)

theta\_e = 80;

if (theta\_e < -80)

theta\_e = -80;

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.1;

vr = (-vc \* (1.0 / (1.0 + exp(-4.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (-vc \* (1.0 / (1.0 + exp(-4.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else if (theta\_e < 85 && theta\_e > -85)

{

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.3;

vr = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else

{

vr = (+1.38 \* theta\_e);

vl = (-1.38 \* theta\_e);

}

Velocity(robot, vl, vr);

}

void Goalie2(Environment\* pEnv)

{

int flag = 0;

double A[2] = { 0 };

PredictBall\_cnt(pEnv, 3, A);

double ballx = A[0];

double bally = A[1];

Vector3D CurrentBall = pEnv->currentBall.pos;

Vector3D player0 = pEnv->home[0].pos;

double ball\_player = 1e10;

double Y = calculateAngleBisector(220, ballx, bally, 220, 70, 216, 110);

double D = calc\_distance(A[0], A[1], player0.x, player0.y);

double D67 = calc\_distance(132.54, 90.0, CurrentBall.x, CurrentBall.y);

double D5 = calc\_distance(165.0, 90.0, CurrentBall.x, CurrentBall.y);

double D4 = calc\_distance(188.27, 90.0, CurrentBall.x, CurrentBall.y);

double D3 = calc\_distance(195, 90.0, CurrentBall.x, CurrentBall.y);

if (CurrentBall.x < 110) {

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else {

for (int i = 1; i <= 4; i++) {

if (pEnv->opponent[i].pos.x > 165) {

flag++;

}

}

for (int i = 1; i <= 4; i++) {

if (pEnv->opponent[i].pos.x > 110) {

flag++;

}

}

if (flag >= 1) {

if (D67 <= 90 && D5 >= 58.52) {

if (A[1] >= 90) {

double tx = 215, ty = 100;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 80;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

else if (D5 < 58.52 && D4 >= 37.5) {

double tx = 215, ty = Y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else if (D4 < 37.5 && D3 >= 32) {

double tx = 215, ty = A[1];

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else if (D3 < 32) {

if (A[1] >= 120) {

double tx = 215, ty = 110;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else if (A[1] <= 60) {

double tx = 215, ty = 70;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = A[1];

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

else if (D67 > 90) {

if (A[1] >= 90) {

double tx = 215, ty = 112;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

else {

double tx = 215, ty = 68;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

}

else {

if (CurrentBall.x >= 200) {

double tx = CurrentBall.x, ty = CurrentBall.y;

ClampGoaliePosition(tx, ty);

Position(pEnv, 0, tx, ty);

stop2(pEnv, 0, tx, ty);

}

}

}

}

void RestrictedArea(Environment\* pEnv)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D predictedBall = pEnv->predictedBall.pos;

Vector3D Leftwing\_pos = pEnv->home[1].pos;//左翼

Vector3D Rightwing\_pos = pEnv->home[2].pos;//右翼

Vector3D Defender\_pos = pEnv->home[4].pos;//前锋

Vector3D Pos[3] = { Leftwing\_pos ,Rightwing\_pos ,Defender\_pos };

int Sign\_attack = 4; //标记控球机器人

int Sign\_wait\_up = 2;//标记靠上等待机器人

int Sign\_wait\_down = 2;//标记靠下等待机器人

int id[3] = { 1,2,4 };

//距球距离

int dis\_left = Distance(ball.x, Leftwing\_pos.x, ball.y, Leftwing\_pos.y);

int dis\_right = Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y);

int dis\_defender = Distance(ball.x, Defender\_pos.x, ball.y, Defender\_pos.y);

int distance[3] = { dis\_left ,dis\_right ,dis\_defender };

int min = 250;

//找出距球最近的机器人

for (int i = 0; i < 3; i++)

{

if (distance[i] <= min)

min = distance[i];

Sign\_attack = id[i];

}

//获取两个waiter的id

int waiter[2], k = 0;

for (int i = 0; i < 3; i++)

{

if (id[i] != Sign\_attack)

{

waiter[k] = id[i];

k++;

}

}

//比较等待机器人位置，决定站位

if (Pos[waiter[0]].y > Pos[waiter[1]].y)

{

Sign\_wait\_up = waiter[0];

Sign\_wait\_down = waiter[1];

}

else

{

Sign\_wait\_up = waiter[1];

Sign\_wait\_down = waiter[0];

}

}

double angle(double x1, double y1, double x2, double y2) {

return atan((y1 - y2) / (x1 - x2));

}

double calculateAngleBisector(double x, double x1, double y1, double x2, double y2, double x3, double y3)

{

double k1 = angle(x1, y1, x2, y2);

double k2 = angle(x1, y1, x3, y3);

double k = tan((k1 + k2) / 2.0);

return k \* (x - x1) + y1;

}

double calc\_distance(double x1, double y1, double x2, double y2) { // 计算两点之间的距离

return sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

}

void stop(Environment\* pEnv, int id1, double x1, double y1)

{

Vector3D player3 = pEnv->home[id1].pos; // 后卫

double x2 = player3.x;

double y2 = player3.y;

if (calc\_distance(x1, y1, x2, y2) < 5)

Velocity(&(pEnv->home[id1]), 0, 0);

}

void stop2(Environment\* pEnv, int id1, double x1, double y1)

{

Vector3D player = pEnv->home[id1].pos;

double x2 = player.x;

double y2 = player.y;

double D = calc\_distance(x1, y1, x2, y2);

if (id1 == 3) {

if (D < 10)

Velocity(&(pEnv->home[id1]), 0, 0);

}

}

//左翼

void LeftWing(Environment\* pEnv, int id)

{

int id1 = id;

Vector3D ball = pEnv->currentBall.pos;

if (ball.x > 85 && ball.x < 195 && ball.y > 15 && ball.y < 165)

{

int id2 = 0;

Vector3D ball = pEnv->currentBall.pos;

Vector3D player3 = pEnv->home[id1].pos; // 后卫

Vector3D player0 = pEnv->home[id2].pos; // 守门员

double A[2] = { 0 };

PredictBall\_cnt(pEnv, 3, A);

// 需要预处理

if (A[0] >= 190)

A[0] = 190;

double D = calc\_distance(ball.x, ball.y, player3.x, player3.y);

double s\_player = 1e10; // 记录间距

int d\_player = -1; // 记录编号

// 求得敌方球员中距离球最近的距离

for (int i = 1; i <= 4; i++) {

double dnt = calc\_distance(ball.x, ball.y, pEnv->opponent[i].pos.x, pEnv->opponent[i].pos.y);

if (dnt < s\_player) {

s\_player = dnt;

d\_player = i;

}

}

Vector3D playerd = pEnv->opponent[d\_player].pos;

double x1 = playerd.x;

double y1 = playerd.y;

// 边线区域

if ((ball.y > 0 && ball.y < 8) || (ball.y > 172 && ball.y < 180))

{

if (ball.x < 150) {

Position(pEnv, id1, 170, fabs(ball.y - 10));

}

else if (ball.x >= 190) {

Position(pEnv, id1, 190, fabs(ball.y - 30));

stop(pEnv, id1, 190, fabs(ball.y - 30));

}

if (fabs(player3.y - ball.y) < 5) {

if (ball.x >= 85)

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, ball.x, ball.y);

}

}

else // 中间位置

{

// 定点

if (ball.x <= 110 && A[1] <= 110) {

Position(pEnv, id1, 190, fabs(ball.y - 15));

stop(pEnv, id1, 190, fabs(ball.y - 15));

}

else if (ball.x >= 190) {

Position(pEnv, id1, 190, fabs(ball.y - 30));

stop(pEnv, id1, 190, fabs(ball.y - 30));

}

else {

Position(pEnv, id1, 190, A[1]);

stop(pEnv, id1, 190, A[1]);

// 这里需要关注速度

if (fabs(player3.x - ball.x) < 25 && ball.x + 4 < player3.x) {

Position(pEnv, id1, A[0], A[1]);

}

}

if (player3.x > 165 && player3.x < 202 && D < 15) {

if (ball.x + 2 < player3.x) // 限制向内旋转

if (ball.y - 2 > player3.y)

Velocity(&(pEnv->home[id1]), -120, 120);

else if (ball.y + 2 < player3.y)

Velocity(&(pEnv->home[id1]), 120, -120);

}

}

}

else if (ball.x > 110 && ball.x < 195 && ball.y < 15)

{

Position(pEnv, id1, ball.x - 30, max(ball.y + 30, 60));

}

else if (ball.x > 110 && ball.x < 195 && ball.y > 165)

{

Position(pEnv, id1, ball.x - 30, min(ball.y - 50, 80));

}

else if (ball.x > 195)

{

if (ball.y > 90)

{

Position(pEnv, id1, 165, ball.y - 45);

}

else

{

Position(pEnv, id1, 165, ball.y + 45);

}

}

else if ((ball.x > 15 && ball.x < 35) && ball.y < 70)

{

Position(pEnv, id1, 25, ball.y + 40);

}

else if ((ball.x > 15 && ball.x < 35) && ball.y > 110)

{

Position(pEnv, id1, 25, ball.y - 40);

}

else if (ball.x < 15)

{

if (ball.y > 70 && ball.y < 110)

{

Position(pEnv, id1, ball.x, ball.y);

Velocity(&(pEnv->home[id1]), 125, 125);

}

else if (ball.y <= 70)

{

Position(pEnv, id1, 30, 115);

}

else

{

Position(pEnv, id1, 30, 60);

}

}

else if (ball.x > 35 && ball.x < 110 && ball.y < 15)

{

Position(pEnv, id1, ball.x + 30, ball.y + 60);

}

else if (ball.x > 35 && ball.x < 110 && ball.y > 165)

{

Position(pEnv, id1, ball.x + 30, ball.y - 60);

}

else

{

Vector3D currentBall = pEnv->currentBall.pos;

Vector3D player1 = pEnv->home[id1].pos;

Vector3D predictedBall = pEnv->predictedBall.pos;

Vector3D opponent\_player0 = pEnv->opponent[0].pos;

int dist = calc\_distance(player1.x, player1.y, currentBall.x, currentBall.y);//球员与球的距离

int dist1 = calc\_distance(currentBall.x, currentBall.y, predictedBall.x, predictedBall.y);

int dist2 = calc\_distance(opponent\_player0.x, opponent\_player0.y, player1.x, player1.y);

double k1 = calc\_slope(currentBall.x, currentBall.y, player1.x, player1.y);

double y\_dist1 = player1.y - k1 \* player1.x; // 球与主攻机器人连线的延长线与门框的交点

if (currentBall.x <= 35) // 球在对方外门框线以内

{

if (currentBall.y <= 45) // 球在下边线区域

{

if (player1.x < currentBall.x) //如下采取的是截球

{

if (player1.y < currentBall.y)//左下角

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125); // 逆时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 8, currentBall.y + 8);

}

}

else//左上角

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45); // 顺时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 8, currentBall.y - 8);// y + 8

}

}

}

else

{

if ((y\_dist1 >= 70 && y\_dist1 <= 110)) // 如果交点在门框内

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y); // 直接采取冲球的方式

}

else

{

if (currentBall.x > 15) {

if (player1.y < currentBall.y) //右下角

{

if (dist < 13)// 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);//x + 10, y - 10

}

}

else//右上角

{

if (dist < 13) // 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x + 10, currentBall.y - 10);

}

}

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

}

else if (currentBall.y >= 135)//球在上边线区域

{

if (player1.x < currentBall.x)

{

if (player1.y < currentBall.y)//左下角

{

if (dist < 13)

{

Velocity(&(pEnv->home[id1]), 45, 125); // 逆时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

else//左上角

{

if (dist < 13)

{

Velocity(&(pEnv->home[id1]), 125, 45); // 顺时针旋转

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

}

else

{

if ((y\_dist1 >= 70 && y\_dist1 <= 110)) // 如果交点在门框内

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y); // 直接采取冲球的方式

}

else

{

if (currentBall.x > 15) {

if (player1.y < currentBall.y) // 右下角

{

if (dist < 13)// 修改前18

{

Velocity(&(pEnv->home[id1]), 45, 125);//逆

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

else//右上角

{

if (dist < 13) // 修改前18

{

Velocity(&(pEnv->home[id1]), 125, 45);//顺

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

}

}

else // 球在中间区域

{

if (y\_dist1 >= 70 && y\_dist1 <= 110) // 当交点在门框内时

{

if (currentBall.x < player1.x) // 当机器人在球的右边，就撞球

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else // 当机器人在球的左边，应该采取截球

{

if (dist < 18) // 修改前18

{

if (player1.y > currentBall.y)

{

Velocity(&(pEnv->home[id1]), 48, 125);

}

else

{

Velocity(&(pEnv->home[id1]), 125, 48);

}

}

else

{

Position(pEnv, id1, currentBall.x + 10, currentBall.y + 10);

}

}

}

else // 交点不在门框时，判断对方守门员位置，调整自己位置

{

/\*if (opponent\_player0.y <= 90) Position(pEnv, id1, max(20, currentBall.x), currentBall.y - 5);

se Position(pEnv, id1, max(20, currentBall.x), currentBall.y + 5);\*/

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

/\*if (player1.y < currentBall.y) // 旋转进球 修改前

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}\*/

}

}

}

else if (currentBall.x <= 110) // 当球在对方半场

{

if (currentBall.y >= 50 && currentBall.y <= 130) // 球在中间区域

{

if (currentBall.x < player1.x) // 机器人在球的左边

{

if (player1.y < currentBall.y)

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}

}

else

{

//Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);修改前

if (player1.y < currentBall.y)

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 45, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 5);

}

}

else

{

if (dist < 18)

{

Velocity(&(pEnv->home[id1]), 125, 45);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 5);

}

}

}

}

else // 球在两边区域

{

if (currentBall.y <= 15) //

{

if (dist < 15)

{

Velocity(&(pEnv->home[id1]), 125, 15);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 15);

}

}

else if (currentBall.y >= 165)

{

if (dist < 15)

{

Velocity(&(pEnv->home[id1]), 25, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 15);

}

}

else if (currentBall.y <= 50)

{

if (currentBall.x < player1.x)

{

if (dist < 5)

{

Velocity(&(pEnv->home[id1]), 125, -15);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y - 3);

}

}

else

{

Position(pEnv, id1, min(currentBall.x + 15, 105), currentBall.y + 15);

}

}

else

{

if (currentBall.x < player1.x)

{

if (dist < 5)

{

Velocity(&(pEnv->home[id1]), -15, 125);

}

else

{

Position(pEnv, id1, currentBall.x + 5, currentBall.y + 3);

}

}

else

{

Position(pEnv, id1, min(currentBall.x + 15, 105), currentBall.y - 15);

}

}

}

}

/\*else if (currentBall.x <= 185)//考虑边界 后卫的判断

{

Vector3D player\_CD = pEnv->home[3].pos;

int dist3 = calc\_distance(player\_CD.x, player\_CD.y, currentBall.x, currentBall.y);

if (currentBall.y <= 10)

{

if (dist3 <= 10)

{

Position(pEnv, id1, currentBall.x + 15, currentBall.y + 8);// 10 8

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

else if (currentBall.y >= 170)

{

if (dist3 <= 10)

{

Position(pEnv, id1, currentBall.x + 15, currentBall.y - 8); // 8 -5

}

else

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

}

else

{

if (currentBall.x <= player1.x)

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else

{

Position(pEnv, id1, currentBall.x + 12, min(currentBall.y + 10, 170));// 7

}

}

}

else

{

if (currentBall.y <= 50)

{

if (currentBall.x <= player1.x)//==

{

Position(pEnv, id1, min(currentBall.x, 185), currentBall.y);

}

else

{

if (currentBall.y > player1.y)

{

Position(pEnv, id1, currentBall.x - 10, max(currentBall.y - 15, 10));

}

else

{

Position(pEnv, id1, 210, 40);

rotate2(pEnv, id1);

}

}

}

else if (currentBall.y >= 130)

{

if (currentBall.x <= player1.x)

{

Position(pEnv, id1, min(currentBall.x, 185), currentBall.y);

}

else

{

if (currentBall.y < player1.y)

{

Position(pEnv, id1, currentBall.x - 10, min(currentBall.y + 15, 170));

}

else

{

Position(pEnv, id1, 210, 140);

rotate1(pEnv, id1);

}

}

}

else

{

if (currentBall.x < player1.x)

{

Attack2(&(pEnv->home[id1]), pEnv);

Position(pEnv, id1, currentBall.x, currentBall.y);

}

else

{

if (currentBall.y > 90)

{

Position(pEnv, id1, 180, 110);

}

else

{

Position(pEnv, id1, 180, 70);

}

}

}

}\*/

}

}

//Attack2(&(pEnv->home[id]), pEnv);

void Goalie(Environment\* pEnv, int id)

{

Vector3D CurrentBall = pEnv->currentBall.pos;

Vector3D player0 = pEnv->home[0].pos;

double ballx = CurrentBall.x;

double bally = CurrentBall.y;

double vc = 100;//速度

double A[4] = { 0 };

double B[4] = { 0 };

PredictBall\_cnt(pEnv, 4, A);//预测 4 周期

PredictBall\_cnt(pEnv, 3, B);//预测 2 周期

// 0 x 1 y

double B\_G\_Angle = 0;//球与守门员的角度

B\_G\_Angle = atan(pEnv->currentBall.pos.y - pEnv->home[0].pos.y) / (pEnv->currentBall.pos.x - pEnv->home[0].pos.x);

double Ball\_Angle = 0;//球的方向角

Ball\_Angle = atan((pEnv->lastBall.pos.y - pEnv->currentBall.pos.y) / (pEnv->lastBall.pos.x - pEnv->currentBall.pos.x));

double Ball\_FB = 0;//球的前后趋势

Ball\_FB = pEnv->lastBall.pos.x - pEnv->currentBall.pos.x;//大于0向右，小于0向左

double Ball\_UD = 0;//球的上下趋势

Ball\_UD = pEnv->lastBall.pos.y - pEnv->currentBall.pos.y;//大于0向下，小于0向上

// 守门员与预测的距离 长距离

double D = calc\_distance(A[0], A[1], player0.x, player0.y);

// 守门员与预测的距离 短距离

double D1 = calc\_distance(B[0], B[1], player0.x, player0.y);

double Pre\_Ball\_Angle;//预测方向角

Pre\_Ball\_Angle = atan((A[1] - pEnv->currentBall.pos.y) / (A[0] - pEnv->currentBall.pos.x));

// 球坐标点和门框两端点构成的三角形，以球坐标为顶角构成的角平分线与215的交点

double Y = calculateAngleBisector(215, A[0], A[1], 215, 70, 215, 110);

double ball\_player = 1e10;// 求得敌方球员中距离球最近的编号

int D2 = 1e10;// 求得敌方球员中距离球最近的距离

for (int i = 1; i <= 4; i++) // 求得敌方球员中距离球最近的距离

{

double dnt = calc\_distance(pEnv->currentBall.pos.x, pEnv->currentBall.pos.y, pEnv->opponent[i].pos.x, pEnv->opponent[i].pos.y);

if (dnt < D2)

{

D2 = dnt;

ball\_player = i;

}

}

int flag = 0;

for (int i = 1; i <= 4; i++) // 统计进入范围的敌方球员个数

{

if (pEnv->opponent[i].pos.x > 165)

{

flag++;

}

}

if (pEnv->currentBall.pos.x == pEnv->lastBall.pos.x)

{

if (pEnv->currentBall.pos.x <= 200 && pEnv->currentBall.pos.x > 180)//点球

{

if (106 >= pEnv->currentBall.pos.y && 76 <= pEnv->currentBall.pos.y)//球门

{

Position\_Goalie(pEnv, 0, 215, A[1], 125);

}

}

else //发球

{

Position\_Goalie(pEnv, 0, pEnv->currentBall.pos.x, pEnv->currentBall.pos.y, vc);

}

}

else

{

if (pEnv->currentBall.pos.x <= 110)// 左半场

{

Position(pEnv, 0, 215, Y);

stop2(pEnv, 0, 215, Y);

}

else // x>110

{

if (pEnv->currentBall.pos.x <= 205)

{

if (pEnv->currentBall.pos.y > 130)//5区上方

{

Position\_Goalie(pEnv, 0, 215, Y, vc);

stop2(pEnv, 0, 215, Y);

}

else if (pEnv->currentBall.pos.y < 50)//5区下方

{

Position\_Goalie(pEnv, 0, 215, Y, vc);

stop2(pEnv, 0, 215, Y);

}

else

{

if (pEnv->currentBall.pos.x <= 185)//4区域

{

if (pEnv->currentBall.pos.x <= 150)

{

if (A[1] <= 130 && A[1] >= 50)

{

Position(pEnv, 0, 215, A[1]);

stop2(pEnv, 0, 215, A[1]);

}

else

{

Position(pEnv, 0, 215, 90);

stop2(pEnv, 0, 215, 90);

}

}

else

{

if (B[1] <= 130 && B[1] >= 50)

{

Position(pEnv, 0, 215, B[1]);

stop2(pEnv, 0, 215, B[1]);

}

else

{

Position(pEnv, 0, 215, 90);

stop2(pEnv, 0, 215, 90);

}

}

}

else//3区域

{

if (B[1] > 130)

{

Position(pEnv, 0, 215, 130);

stop2(pEnv, 0, 215, 130);

}

else if (B[1] > 50)

{

Position\_Goalie(pEnv, 0, 215, B[1], 125);

// 旋转

Velocity(&(pEnv->home[0]), 125, 125);

stop2(pEnv, 0, 215, B[1]);

}

else

{

Position(pEnv, 0, 215, 50);

stop2(pEnv, 0, 215, 50);

}

}

}

}

else//12区内

{

if (B[1] <= 30)//下方

{

Position\_Goalie(pEnv, 0, 215, 68, 105);

stop2(pEnv, 0, 215, 70);

}

else if (B[1] <= 65)//2下

{

Position\_Goalie(pEnv, 0, 215, 75, 110);

stop2(pEnv, 0, 215, 75);

}

else if (B[1] <= 115)// 1中间部分

{

if (D1 <= 3.2 && D1 < D2)

{

// 旋球

Position\_Goalie(pEnv, 0, 215, B[1], 125);

// 旋转

Velocity(&(pEnv->home[0]), 125, 125);

stop2(pEnv, 0, 215, B[1]);

}

else {

Position\_Goalie(pEnv, 0, 215, B[1], 125);

stop2(pEnv, 0, 215, B[1]);

}

}

else if (B[1] <= 150)//2上

{

Position\_Goalie(pEnv, 0, 215, 105, 110);

stop2(pEnv, 0, 215, 105);

}

else//上

{

Position\_Goalie(pEnv, 0, 215, 107, 105);

stop2(pEnv, 0, 215, 100);

}

}

}

}

}

//右翼

void RightWing(Environment\* pEnv, int id)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D Rightwing\_pos = pEnv->home[id].pos;

Vector3D Leftwing\_pos = pEnv->home[1].pos;

Vector3D Attacker\_pos = pEnv->home[4].pos;

Vector3D Goalie\_pos = pEnv->home[0].pos;

Vector3D Defender\_pos = pEnv->home[3].pos;

Vector3D predictedBall;

double A[3];

PredictBall\_cnt(pEnv, 4, A);

predictedBall.x = A[0];

predictedBall.y = A[1];

double ang = 0;

//防止碰我方守门员

if (Distance(Goalie\_pos.x, Rightwing\_pos.x, Goalie\_pos.y, Rightwing\_pos.y) <= 8)

{

Position(pEnv, id, Rightwing\_pos.x - 15, Rightwing\_pos.y);

}

//若进入我方禁区，撤出

if (Rightwing\_pos.x >= 200 && ball.y >= 60 && ball.y <= 120)

{

if (ball.y >= 90)

{

Position(pEnv, id, 185, 60);

stop(pEnv, 0, 185, 60);

}

else

{

Position(pEnv, id, 185, 120);

stop(pEnv, 0, 185, 120);

}

}

//---------上下边线处理------------

//上边界

else if (ball.y >= 168)

{

//敌方半场上边界

if (ball.x <= 110)

{

if (AttackerClose(pEnv) && Attacker\_pos.x > ball.x)//前锋在球旁且在球右边

{

//前锋控球则在旁边，防止碰撞,球过来则逆时针旋转

Position(pEnv, id, ball.x + 17, ball.y - 15);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9 && ball.x < Rightwing\_pos.x)

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), -200, 200);

}

}

else if (predictedBall.x - ball.x > 0)//若球向我方半场，拦截

{

if (ball.x <= Rightwing\_pos.x)//机器人在球右

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9 && ball.x < 85)//距离近且不在中间区域

Velocity(&(pEnv->home[id]), -200, 200);

}

else if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 15, ball.y - 15);

}

else//左上

{

Position(pEnv, id, ball.x - 10, ball.y - 15);

}

}

else//向敌方半场

{

if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 10, ball.y - 10);

}

else if (ball.x > Rightwing\_pos.x && ball.y <= Rightwing\_pos.y)//左上

{

Position(pEnv, id, ball.x - 10, ball.y - 15);

}

else//右边

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 10 && ball.x < 85)

Velocity(&(pEnv->home[id]), -200, 200);

}

}

}

//我方半场上边界,主要拦截

else

{

if (predictedBall.x - ball.x > 0)//若球向我方半场，拦截

{

if (ball.x < Rightwing\_pos.x)//机器人在球右

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 10 && ball.x > 135)//距离近且不在中间区域

Velocity(&(pEnv->home[id]), -125, 125);

}

else if (ball.x >= Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 10, ball.y - 10);

}

else//左上

{

Position(pEnv, id, ball.x - 10, ball.y - 12);

}

}

else//向敌方半场则推进

{

if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 10, ball.y - 10);

}

else if (ball.x > Rightwing\_pos.x && ball.y <= Rightwing\_pos.y)//左上

{

Position(pEnv, id, ball.x - 10, ball.y - 10);

}

else//右边

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 10 && ball.x > 170)//距离近且不在中间区域

Velocity(&(pEnv->home[id]), -125, 125);

}

}

}

}

//下边界

else if (ball.y <= 14)

{

//敌方半场下边界,准备在上半场等待

if (ball.x <= 110)

{

if (ball.x >= 35)

{

Position(pEnv, id, ball.x + 30, ball.y + 100);

stop(pEnv, 0, ball.x + 30, ball.y + 100);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, ball));

}

else

{

Position(pEnv, id, 45, ball.y + 100);

stop(pEnv, 0, 45, ball.y + 100);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, ball));

}

}

else if (ball.x < 185)//我方半场下边界,右翼在球右上方准备

{

Position(pEnv, id, ball.x + 35, ball.y + 35);

}

}

//----------end上下边线----------------

//---------进攻区域---------------------

else if (ball.x < 90)

{

//左边线

if (ball.x < 10)

{

if (ball.y > 115)//上边

{

if (ball.x <= Rightwing\_pos.x && ball.y >= Rightwing\_pos.y)//机器人在球右下

{

Position(pEnv, id, ball.x + 10, ball.y + 10);

}

else if (ball.x <= Rightwing\_pos.x && ball.y <= Rightwing\_pos.y)//右上

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, ball.x, ball.y, 180);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 10 && ball.y > 135)//门口不旋转，直接推

Velocity(&(pEnv->home[id]), -200, 200);

}

else if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 15, ball.y - 15);

}

else//左上

{

Position(pEnv, id, ball.x + 15, ball.y + 10);

}

}

else if (ball.y < 65)//下边

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, 60, 95);

stop(pEnv, 0, 60, 95);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, predictedBall));

}

}

//敌方禁区

else if (ball.x < 45 && ball.y >= 40 && ball.y <= 140)

{

if (Distance(ball.x, pEnv->home[id].pos.x, ball.y, pEnv->home[id].pos.y) <= 40)

{

double k1;

k1 = sqrt((-15 - predictedBall.x) \* (-15 - predictedBall.x) + (70 - predictedBall.y) \* (70 - predictedBall.y));

double k2;

k2 = sqrt((-15 - predictedBall.x) \* (-15 - predictedBall.x) + (115 - predictedBall.y) \* (115 - predictedBall.y));

double k;

k = sqrt((predictedBall.x - Rightwing\_pos.x) \* (predictedBall.x - Rightwing\_pos.x) + (predictedBall.y - Rightwing\_pos.y) \* (predictedBall.y - Rightwing\_pos.y));

int falg = 0;

if (Rightwing\_pos.x > ball.x && (k >= k2 && k <= k1))//中间射门区域

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, ball.x, ball.y, 180);

stop(pEnv, id, 8, ball.y);

}

else

{

if (ball.y >= 90)

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9 && ball.x < Rightwing\_pos.x)

Velocity(&(pEnv->home[id]), 180, -180);

}

else

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9 && ball.x < Rightwing\_pos.x)

Velocity(&(pEnv->home[id]), -180, 180);

}

}

}

else

{

Position(pEnv, id, 52, ball.y + 10);

stop(pEnv, 0, 50, ball.y + 10);

}

}

//球在上半场中间区域

else if (ball.y >= 90)

{

//若前峰能够控球,则在右下辅助

if (AttackerClose(pEnv) && Attacker\_pos.x > ball.x)

{

Position(pEnv, id, ball.x + 15, ball.y - 25);

}

else//不能控球则右翼去拿球

{

if (ball.x <= Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//机器人在球右下

{

Position(pEnv, id, ball.x + 8, ball.y + 8);

}

else if (ball.x <= Rightwing\_pos.x && ball.y <= Rightwing\_pos.y)//右上

{

Attack2(&(pEnv->home[id]), pEnv);

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9)

Velocity(&(pEnv->home[id]), -180, 180);

}

else if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 15, ball.y - 10);

}

else//左上

{

Position(pEnv, id, ball.x + 10, ball.y + 10);

}

}

}

else//球在下半场

{

if (ball.x <= 35)

{

Position(pEnv, id, ball.x + 40, ball.y + 60);

}

else

{

Position(pEnv, id, ball.x + 40, 100);

}

}

}

//-----------end进攻区域--------------

//-----------防守区域---------------------

else if (ball.x > 115)

{

//禁区

if (ball.x >= 185 && ball.y >= 50 && ball.y <= 130)

{

if (ball.y < 90)

{

Position(pEnv, id, 165, 90);

stop(pEnv, 0, 165, 90);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, ball));

}

else

{

Position(pEnv, id, 165, 90);

stop(pEnv, 0, 165, 90);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, ball));

}

}

//下半场

else if (ball.y < 90)

{

if (ball.y >= 50)//上区域

{

if (Rightwing\_pos.x >= ball.x)//在球右方

{

Position(pEnv, id, ball.x, ball.y);

}

else

{

Position(pEnv, id, 175, 90);

stop(pEnv, 0, 175, 90);

}

}

else//下区域

{

if (ball.x < 175)//

{

if (Rightwing\_pos.x >= ball.x)//在球右边

{

Position(pEnv, id, ball.x, ball.y);

}

else//在球左边

{

Position(pEnv, id, 185, 47);

stop(pEnv, 0, 180, 50);

// Angle(&(pEnv->home[2]), GetAngle(Rightwing\_pos, ball));

}

}

else//右下边线

{

if (ball.y >= Rightwing\_pos.y)//在球下面,撤走

{

Position(pEnv, id, ball.x - 25, ball.y, 180);

}

else if (OurRobotBesideBall(pEnv) && (Defender\_pos.y > ball.y || Leftwing\_pos.y > ball.y))

{

Position(pEnv, id, ball.x - 25, ball.y + 3, 180);

}

else

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9)

{

Velocity(&(pEnv->home[id]), 200, -200);

}

}

}

}

}

//上半场

else

{

if (ball.x >= 200)//右上边线

{

if (ball.y <= Rightwing\_pos.y)//在球上面,撤走

{

Position(pEnv, id, ball.x - 25, ball.y, 180);

}

else if (OurRobotBesideBall(pEnv) && (Defender\_pos.y < ball.y || Leftwing\_pos.y < ball.y))

{

Position(pEnv, id, ball.x - 25, ball.y - 3, 180);

}

else

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9)

{

Velocity(&(pEnv->home[id]), -125, 125);

}

}

}

else

{

if (ball.x <= Rightwing\_pos.x && ball.y >= Rightwing\_pos.y)//机器人在球右下

{

Position(pEnv, id, ball.x, ball.y);

if (Distance(ball.x, Rightwing\_pos.x, ball.y, Rightwing\_pos.y) <= 9)

{

Velocity(&(pEnv->home[id]), -125, 125);

}

}

else if (ball.x <= Rightwing\_pos.x && ball.y <= Rightwing\_pos.y)//右上

{

Position(pEnv, id, ball.x + 8, ball.y - 8);

}

else if (ball.x > Rightwing\_pos.x && ball.y > Rightwing\_pos.y)//左下

{

Position(pEnv, id, ball.x + 8, ball.y - 8);

}

else//左上

{

Position(pEnv, id, ball.x - 8, ball.y - 8);

}

}

}

}

//-----------end防守区域---------------------

//中间判断区域

else if (ball.x >= 90 && ball.x <= 120)

{

//下半场

if (ball.y < 85)

{

if (predictedBall.x - ball.x > 0)//球向右

{

Position(pEnv, id, 137, 90);//先待命转向

stop(pEnv, 0, 137, 90);

}

else//球向左

{

Position(pEnv, id, ball.x + 10, ball.y + 15);

}

}

//上半场

else

{

if (predictedBall.x - ball.x > 0)//向右

{

if (Rightwing\_pos.x - ball.x > 0)//在球右边

{

Position(pEnv, id, ball.x, ball.y);

}

else

{

Position(pEnv, id, ball.x + 10, ball.y + 8);

}

}

else//向左

{

if (Rightwing\_pos.x - ball.x > 0)

{

Position(pEnv, id, ball.x, ball.y);

}

else

{

Position(pEnv, id, ball.x + 10, ball.y + 10);

}

}

}

}

}

void Anglelr(Environment\* pEnv, Robot\* robot) {

double theta\_e = 0, vl = 20, vr = -30;

Vector3D ball = pEnv->currentBall.pos;

Vector3D player = pEnv->home[3].pos;

double ang;

ang = GetAngle(ball, player);

theta\_e = ang - robot->rotation;

while (theta\_e > 180) theta\_e -= 360;

while (theta\_e < -180) theta\_e += 360;

if (theta\_e < -90) theta\_e += 180;

else if (theta\_e > 90) theta\_e -= 180;

if (fabs(theta\_e) > 50)

{

vl = -9. / 90.0 \* theta\_e;

vr = 9. / 90.0 \* theta\_e;

}

else if (fabs(theta\_e) > 20)

{

vl = -11.0 / 90.0 \* theta\_e;

vr = 11.0 / 90.0 \* theta\_e;

}

Velocity(robot, vl, vr);

}

void CenterDefender1(Environment\* pEnv, int id) {

Vector3D ball = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id].pos;

if (ball.x < 40 && player.x < 100) {//calc\_distance(ball.x, ball.y, 37.5, 90) < 3

plent(pEnv,id,id);

}

else {

CenterDefender(pEnv, id);

}

}

Vector3D last\_player;

void CenterDefender(Environment\* pEnv, int id)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D last\_ball = pEnv->lastBall.pos;

Vector3D preball = pEnv->predictedBall.pos;

Vector3D player = pEnv->home[id].pos;

double A[2];

PredictBall\_cnt(pEnv, 3, A);

int Vel[4] = { 100,110,117.5,125 };

double Dp = calc\_distance(player.x, player.y, ball.x, ball.y);//现在球和人的距离

double Db = calc\_distance(last\_ball.x, last\_ball.y, ball.x, ball.y);//上个周期球和目前的距离

double Dp\_p = calc\_distance(last\_player.x, last\_player.y, player.x, player.y);//上个周期人和目前的距离

double Dd\_p = calc\_distance(pEnv->home[0].pos.x, pEnv->home[0].pos.y, player.x, player.y);//人和守门员的距离

if (player.x > 203 && player.y < 115 && player.y>65)

{

if (player.y > 90)

Position(pEnv, id, player.x - 7, player.y - 10);

else

Position(pEnv, id, player.x - 7, player.y + 10);

}

//===========

else if (player.x < 90)

{

if (Dp < 11)

Position(pEnv, id, ball.x, ball.y);

else

{

if (player.y < 90)

Position(pEnv, id, player.x + 13, player.y + 10);

else

Position(pEnv, id, player.x + 13, player.y + 10);

}

}

//===========删除；

else {

if (ball.x < 75) {//球在对面半场

if (ball.y < 30)//球在下部分

{

Position(pEnv, id, 150, 25);//到这个点待命；

stop(pEnv, id, 150, 25);

}

else if (ball.y >= 30 && ball.y < 60)

{

Position(pEnv, id, 150, 45);//到这个点待命；

}

else if (ball.y >= 60 && ball.y < 120)

{

Position(pEnv, id, 150, 90);//到这个点待命；

}

else if (ball.y >= 120 && ball.y < 150)

{

Position(pEnv, id, 150, 135);

}

else

{

Position(pEnv, id, 150, 155);

stop(pEnv, id, 150, 155);

}

}

else if (ball.x >= 75 && ball.x < 160)

{

if (ball.y <= 10)

{

if (ball.x < player.x)

{

if (Db < 11) {

Position(pEnv, id, ball.x + 4, ball.y - 4, Vel[3]);

if (Dp < 15)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, 200, -200);

}

}

}

else {

Position(pEnv, id, player.x, 3, Vel[3]);

}

}

else {

Position(pEnv, id, ball.x + 20, ball.y + 15, Vel[3]);

}

}

else if (ball.y >= 170)

{

if (ball.x < player.x)

{

if (Db < 11) {

Position(pEnv, id, ball.x + 4, ball.y + 4, Vel[3]);

if (Dp < 15)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, -200, 200);

}

}

}

else {

Position(pEnv, id, player.x, 177, Vel[3]);

}

}

else {

Position(pEnv, id, ball.x + 20, ball.y - 15, Vel[3]);

}

}

else if (ball.y > 10 && ball.y < 90)

{

if (ball.x < player.x)

{

Position(pEnv, id, ball.x, ball.y, Vel[1]);

if (Dp <= 8) {

Position(pEnv, id, ball.x, ball.y, Vel[2]);

}

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, -300, 300);

}

}

else if (ball.x > player.x)

{

Position(pEnv, id, 180, 60, Vel[2]);

}

}

else if (ball.y < 170 && ball.y >= 90)

{

if (ball.x < player.x)

{

Position(pEnv, id, ball.x, ball.y, Vel[1]);

if (Dp <= 8) {

Position(pEnv, id, ball.x, ball.y, Vel[2]);

}

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, 300, -300);

}

}

else if (ball.x > player.x)

{

Position(pEnv, id, 180, 120);

}

}

}

/\*===========================================================================================\*/

else if (ball.x >= 160 && ball.x < 205)

{

if (ball.y < 30)

{

if (ball.x < player.x)

{

if (fabs(ball.y - player.y) < 8)

{

Position(pEnv, id, ball.x, ball.y, Vel[1]);

if (Dp < 10)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3)

{

Velocity(pEnv->home + id, 300, -300);

}

}

}

}

else

{

if (ball.x > 185)

{

Position(pEnv, id, 210, ball.y + 13, Vel[2]);

}

else

{

Position(pEnv, id, ball.x + 15, ball.y + 12, Vel[1]);

}

}

}

else if (ball.y > 150)

{

if (ball.x < player.x)

{

if (fabs(ball.y - player.y) < 8)

{

Position(pEnv, id, ball.x, ball.y, Vel[1]);

if (Dp < 10)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3)

{

Velocity(pEnv->home + id, -300, 300);

}

}

}

}

else

{

if (ball.x > 185)

{

Position(pEnv, id, 210, ball.y - 13, Vel[2]);

}

else

{

Position(pEnv, id, ball.x + 15, ball.y + 12, Vel[1]);

}

}

}

else

{

if (ball.x < 185)

{

if (ball.x < player.x)

{

Position(pEnv, id, ball.x, ball.y);

if (Dp < 10) {

if (player.y > ball.y)

{

Velocity(pEnv->home + id, 200, -200);

}

else

{

Velocity(pEnv->home + id, -200, 200);

}

}

}

else

{

Position(pEnv, id, 193, ball.y, Vel[2]);

}

}

}

}

else//ball.x>=205;

{

if (ball.y < 50)

{

Position(pEnv, id, ball.x, ball.y + 13, Vel[3]);

if (fabs(ball.x - player.x) < 8)

{

if (ball.y < player.y)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, 200, -200);

}

}

else

{

Position(pEnv, id, ball.x - 13, ball.y + 10, Vel[2]);

}

}

}

else if (ball.y > 130)

{

Position(pEnv, id, ball.x, ball.y - 13, Vel[3]);

if (fabs(ball.x - player.x) < 8)

{

if (ball.y > player.y)

{

Position(pEnv, id, ball.x, ball.y, Vel[2]);

if (Db < 3 && Dp < 10)

{

Velocity(pEnv->home + id, -200, 200);

}

}

else

{

Position(pEnv, id, ball.x - 13, ball.y - 10, Vel[2]);

}

}

}

else

{

if (ball.y > 90)

{

Position(pEnv, id, 180, pEnv->home[0].pos.y - 13);

}

else

{

Position(pEnv, id, 180, pEnv->home[0].pos.y + 13);

}

}

}

}

last\_player = player;

}

void Penalty\_Kike1(Environment\* pEnv, int id)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id].pos;

double x1 = 40.0;

double y1 = 0.46 \* 40 + 153;

Position(pEnv, 3, x1, y1, 125);

Velocity(&(pEnv->home[3]), 125, 125);

if (ball.x < 35) {

Velocity(&(pEnv->home[3]), -180, 180);

stop(pEnv, id, x1 - 5, y1 - 5);

}

}

void Penalty\_Kike2(Environment\* pEnv, int id)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D player = pEnv->home[id].pos;

double x1 = 40.0;

double y1 = 0.46 \* 40 + 70;

Position(pEnv, 3, x1, y1, 125);

Velocity(&(pEnv->home[3]), 102, 125);

if (ball.x < 35) {

Velocity(&(pEnv->home[3]), 180, -180);

stop(pEnv, id, x1 - 5, y1 - 5);

}

}

void nopushtime(Environment\* pEnv)

{

Vector3D ball = pEnv->currentBall.pos;

int cnt = 0;

std::vector<int>currobot;

for (int i = 1; i <= 4; i++)

{

double dist;

Vector3D robot = pEnv->home[i].pos;

dist = sqrt((robot.x - ball.x) \* (robot.x - ball.x) + (robot.y - ball.y) \* (robot.y - ball.y));

if ((ball.x < 10 && dist < 15) || (ball.x > 210 && dist < 15) || (ball.y > 170 && dist < 15) || (ball.y < 10 && dist < 15))

{

cnt++;

currobot.push\_back({ i });

}

}

if (cnt >= 2)

{

return;

}

else

{

int l = 0, r = 0, fre = 0, lat = 0;

std::vector<int>left;

std::vector<int>right;

std::vector<int>qian;

std::vector<int>ho;

for (auto& t : currobot)

{

Vector3D robot = pEnv->home[t].pos;

if (robot.x < ball.x)

{

l++;

left.push\_back(t);

}

else if (robot.x > ball.x)

{

r++;

right.push\_back(t);

}

}

for (auto& t : currobot)

{

Vector3D robot = pEnv->home[t].pos;

if (robot.y > ball.y)

{

fre++;

qian.push\_back(t);

}

else if (robot.y < ball.y)

{

lat++;

ho.push\_back(t);

}

}

if (ball.y < 10 && ball.x>15 && ball.x < 205)

{

for (auto& m : left)

{

Position(pEnv, m, ball.x, ball.y + 14);

}

int flag = 0;

for (auto& m : right)

{

flag++;

if (flag > 1)

{

Position(pEnv, m, ball.x, ball.y + 14);

}

}

}

else if (ball.y > 170 && ball.x > 15 && ball.x < 205)

{

for (auto& m : left)

{

Position(pEnv, m, ball.x, ball.y - 14);

}

int flag = 0;

for (auto& m : right)

{

flag++;

if (flag > 1)

{

Position(pEnv, m, ball.x, ball.y - 14);

}

}

}

else if ((ball.x > 170 && ball.y < 90) || (ball.x < 10 && ball.y>90))

{

for (auto& m : ho)

{

if (ball.x < 10)

Position(pEnv, m, ball.x + 15, ball.y);

else

{

Position(pEnv, m, ball.x - 15, ball.y);

}

}

int flag = 0;

for (auto& m : qian)

{

flag++;

if (flag > 1)

{

if (ball.x < 10)

Position(pEnv, m, ball.x + 15, ball.y);

else

{

Position(pEnv, m, ball.x - 15, ball.y);

}

}

}

}

else if ((ball.x > 170 && ball.y < 90) || (ball.x < 10 && ball.y > 90))

{

for (auto& m : qian)

{

if (ball.x < 10)

Position(pEnv, m, ball.x + 15, ball.y);

else

{

Position(pEnv, m, ball.x - 15, ball.y);

}

}

int flag = 0;

for (auto& m : ho)

{

flag++;

if (flag > 1)

{

if (ball.x < 10)

Position(pEnv, m, ball.x + 15, ball.y);

else

{

Position(pEnv, m, ball.x - 15, ball.y);

}

}

}

}

}

}

void CenterAttacker(Environment\* pEnv, int id)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D palyer = pEnv->home[4].pos;

Vector3D lastball = pEnv->lastBall.pos;

Vector3D preball = pEnv->currentBall.pos;

double dist = sqrt((ball.y - palyer.y) \* (ball.y - palyer.y) + (ball.x - palyer.x) \* (ball.x - palyer.x));

if (ball.x > 10 && ball.x <= 40)//敌方禁区

{

if (ball.y < 40 && ball.y>0)//下边线

{

if (ball.y < 10 && palyer.x>ball.x)//球已经在底线，继续推底线，加点旋转

{

//Velocity(&(pEnv->home[id]), 45, 125);//逆时针增加控球机会

Position(pEnv, id, 0, 0);//走到下底线

if (fabs(lastball.y - ball.y) < 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Velocity(&(pEnv->home[id]), 260, 260);

}

//if (ball.x < 7 && dist < 10)//球到底角，把球推上去

//{

// Velocity(&(pEnv->home[id]), 100, 0);

//}

}

else//球不在底线，讨论求和机器人的相对位置

{

if (palyer.x < ball.x)//前锋跑球前面去了

{

if (palyer.y <= ball.y)//球在我的右上方

{

Position(pEnv, id, ball.x + 8, ball.y + 15);//尽快移动到球的右上方

}

else if (palyer.y > ball.y)//球在我的右下方

{

Position(pEnv, id, ball.x + 8, ball.y + 10);//也移动到球的右上方

}

}

else if (palyer.x >= ball.x)//前锋在球的左方,很适合推球

{

if (palyer.y > ball.y)//最适合下边线区的推球位置

{

double k;

k = (ball.y - palyer.y) / (ball.x - palyer.x);

if ((k \* (0 - ball.x) + ball.y) < 0)//可以把球推到下底线

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

else

{

Position(pEnv, id, ball.x + 5, ball.y + 5);//离球近点

Velocity(&(pEnv->home[id]), 125, 125);

}

}

else if (palyer.y < ball.y)//球在我左上方

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

}

}

}

else if (ball.y >= 40 && ball.y <= 140)//中间直接射门

{

if (palyer.x > 10) {

double k1;

k1 = sqrt((-15 - ball.x) \* (-15 - ball.x) + (70 - ball.y) \* (70 - ball.y));

double k2;

k2 = sqrt((-15 - ball.x) \* (-15 - ball.x) + (115 - ball.y) \* (115 - ball.y));

double k;

k = sqrt((ball.x - palyer.x) \* (ball.x - palyer.x) + (ball.y - palyer.y) \* (ball.y - palyer.y));

int falg = 0;

if (palyer.x >= ball.x && (k >= k2 && k <= k1))//中间最佳射门区域

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

else

{

huxian(pEnv, id);

}

}

else

{

Position(pEnv, id, preball.x + 8, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

}

else if (ball.y > 140)//上边线

{

if (ball.y > 170 && palyer.x > ball.x)//球已经在底线，继续推底线，加点旋转

{

Position(pEnv, id, 0, 180);//走到上底线

if (ball.x < 15)//球到底角，把球推上去

{

Velocity(&(pEnv->home[id]), 125, 100);

Position(pEnv, id, ball.x, ball.y);

if (fabs(lastball.y - ball.y) <= 8)//球被挤了,把球旋出去让右翼完成射门

{

Velocity(&(pEnv->home[id]), 200, 200);

}

}

}

else//球不在底线，讨论求和机器人的相对位置

{

if (palyer.x < ball.x)//前锋跑球前面去了

{

if (palyer.y < ball.y)//球在我的右上方

{

Position(pEnv, id, ball.x + 8, ball.y - 8);//尽快移动到球的右下方

}

else if (palyer.y > ball.y)//球在我的右下方

{

Position(pEnv, id, ball.x + 8, ball.y - 8);//也移动到球的右上方

}

}

else if (palyer.x >= ball.x)//前锋在球的左方,很适合推球

{

if (palyer.y > ball.y)

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

else if (palyer.y < ball.y)//球在我左上方

{

double k;

k = (ball.y - palyer.y) / (ball.x - palyer.x);

if ((k \* (0 - ball.x) + ball.y) > 180)//可以把球推到上底线

{

Position(pEnv, id, ball.x, ball.y);

Velocity(&(pEnv->home[id]), 125, 125);

}

else

{

Position(pEnv, id, ball.x + 5, ball.y);//离球近点

}

}

}

}

}

}

else if (ball.x < 10)

{

if (ball.y < 90)

{

if (ball.y <= palyer.y)

{

Position(pEnv, id, ball.x + 7, ball.y - 10);

}

else if (ball.y > palyer.y)

{

Velocity(&(pEnv->home[id]), 125, 100);

Position(pEnv, id, ball.x, ball.y);

if (fabs(lastball.y - ball.y) <= 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Position(pEnv, id, ball.x, ball.y - 5);

Velocity(&(pEnv->home[id]), 200, 200);

}

}

}

else if (ball.y >= 90 && ball.y < 180)

{

if (ball.y < palyer.y)

{

Velocity(&(pEnv->home[id]), 100, 125);

Position(pEnv, id, ball.x, ball.y);

if (fabs(lastball.y - ball.y) <= 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Position(pEnv, id, ball.x, ball.y + 5);

Velocity(&(pEnv->home[id]), 200, 200);

}

}

else if (ball.y >= palyer.y)

{

Position(pEnv, id, ball.x + 7, ball.y + 10);

}

}

}

else if (ball.x >= 40 && ball.x <= 80)

{

if (palyer.x < ball.x)//球在人的后方，不好控球

{

if (ball.y < palyer.y)//球在人的右下方

{

Position(pEnv, id, ball.x + 10, ball.y + 15);//移动到球的右上方

}

else if (ball.y >= palyer.y)

{

Position(pEnv, id, ball.x + 10, ball.y - 15);

}

}

else if (palyer.x >= ball.x)

{

if (fabs(ball.y - palyer.y) <= 7)

{

Position(pEnv, id, 15, ball.y);

}

else if (ball.y - palyer.y > 7)//球在我的左上方

{

double k;

k = (ball.y - palyer.y) / (ball.x - palyer.x);

if (ball.y + k \* (-ball.x) < 180)

{

Position(pEnv, id, ((180 - ball.y) + k \* ball.x) / k, 180);

if (fabs(lastball.y - ball.y) <= 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Velocity(&(pEnv->home[id]), -180, 180);

}

}

else

{

Position(pEnv, id, ball.x + 5, ball.y - 5);

}

}

else if (palyer.y - ball.y > 7)

{

double k;

k = (ball.y - palyer.y) / (ball.x - palyer.x);

if (ball.y + k \* (-ball.x) > 0)

{

Position(pEnv, id, ((0 - ball.y) + k \* ball.x) / k, 0);

if (fabs(lastball.y - ball.y) <= 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Velocity(&(pEnv->home[id]), 180, -180);

}

}

else

{

Position(pEnv, id, ball.x + 10, ball.y - 10);

}

}

}

}

else if (ball.x >= 80 && ball.x <= 160)

{

if (ball.x < palyer.x)//球在我的前面

{

Position(pEnv, id, ball.x, ball.y);

if (fabs(lastball.y - ball.y) <= 8 && dist < 8)//球被挤了,把球旋出去让右翼完成射门

{

Velocity(&(pEnv->home[id]), -180, 180);

}

else if (fabs(lastball.y - ball.y) <= 8 && dist >= 8)

{

if (ball.y < 30)

{

Position(pEnv, id, ball.x + 20, min(palyer.y + 20, ball.y + 20));//参数可调 15 20

}

else if (ball.y > 150)

{

Position(pEnv, id, ball.x + 20, min(palyer.y - 20, ball.x - 20));//参数可调

}

else

{

Position(pEnv, id, ball.x + 20, ball.y + 20);//参数可调

}

}

}

else

{

if (ball.y < 90)

{

Position(pEnv, id, ball.x + 20, preball.y + 15);

}

else if (ball.y > 90)

{

Position(pEnv, id, ball.x + 20, preball.y - 15);

}

}

}

else if (ball.x > 160)

{

if (ball.x > palyer.x)

{

if (ball.y < 90)

{

Position(pEnv, id, 155, ball.y + 15);

}

else if (ball.y >= 90 && ball.y <= 180)

{

Position(pEnv, id, 155, ball.y - 20);

}

}

else if (ball.x < palyer.x)

{

Position(pEnv, id, ball.x, ball.y);

if (fabs(lastball.y - ball.y) <= 8 && dist >= 8 && (ball.y < 10 || ball.y>170))

{

if (ball.y < 90)

{

Position(pEnv, id, ball.x, ball.y + 20);

}

else if (ball.y >= 90)

{

Position(pEnv, id, ball.x, ball.y - 20);

}

}

}

}

}

void MoonFollowOpponent(Robot\* robot, OpponentRobot\* opponent)

{

Position(robot, opponent->pos.x, opponent->pos.y);

}

//调速度

void Velocity(Robot\* robot, double vl, double vr)

{

if (vl > 125)vl = 125;

if (vl < -125)vl = -125;

if (vr > 125)vr = 125;

if (vr < -125)vr = -125;

if (true)

{//速度的特别控制

if (vl == 0 && vr != 0)

vl = 0.00001;

if (vr == 0 && vl != 0)

vr = 0.00001;

}

robot->velocityLeft = vl;

robot->velocityRight = vr;

}

//调整角度

void Angle(Robot\* robot, double desired\_angle)

{

double theta\_e = 0, vl = 0, vr = 0;

theta\_e = desired\_angle - robot->rotation;

while (theta\_e > 180) theta\_e -= 360;

while (theta\_e < -180) theta\_e += 360;

if (theta\_e < -90) theta\_e += 180;

else if (theta\_e > 90) theta\_e -= 180;

if (fabs(theta\_e) > 50)

{

vl = -9. / 90.0 \* theta\_e;

vr = 9. / 90.0 \* theta\_e;

}

else if (fabs(theta\_e) > 20)

{

vl = -11.0 / 90.0 \* theta\_e;

vr = 11.0 / 90.0 \* theta\_e;

}

Velocity(robot, vl, vr);

}

//让编号为id的球员移动到(x,y)

void Position(Environment\* pEnv, int id, double x, double y, double Vel)

{

Position(&(pEnv->home[id]), x, y);

}

//让球员robot移动到(x,y)

void Position(Robot\* robot, double x, double y, double Vel)

{

double desired\_angle = 0, theta\_e = 0, d\_angle = 0;

double vl = 0, vr = 0, vc = 125;//130

double dx, dy, d\_e, Ka = 0.8;//0.8

dx = x - robot->pos.x;

dy = y - robot->pos.y;

d\_e = sqrt(dx \* dx + dy \* dy);

if (dx == 0 && dy == 0)

desired\_angle = 90;

else

desired\_angle = 180. / PI \* atan2((double)(dy), (double)(dx));

theta\_e = desired\_angle - robot->rotation;

while (theta\_e > 180) theta\_e -= 360;

while (theta\_e < -180) theta\_e += 360;

if (theta\_e > 95 || theta\_e < -95)

{

theta\_e += 180;

if (theta\_e > 180)

theta\_e -= 360;

if (theta\_e > 80)

theta\_e = 80;

if (theta\_e < -80)

theta\_e = -80;

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 2.2;//2.2

vr = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.1) + Ka \* theta\_e);

vl = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.1) - Ka \* theta\_e);

}

else if (theta\_e < 85 && theta\_e > -85)

{

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 2.3;//2.3

vr = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.1) + Ka \* theta\_e);

vl = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.1) - Ka \* theta\_e);

}

else

{

vr = (+1.38 \* theta\_e);//1.38

vl = (-1.38 \* theta\_e);

}

Velocity(robot, vl, vr);

}

//调速度

void Goalie1(Robot\* robot, Environment\* pEnv)

{

double velocityLeft = 0, velocityRight = 0;

double Tx = pEnv->goalBounds.right - pEnv->currentBall.pos.x;

double Ty = pEnv->fieldBounds.top - pEnv->currentBall.pos.y;

double Ax = pEnv->goalBounds.right - robot->pos.x;

double Ay = pEnv->fieldBounds.top - robot->pos.y;

if (Ay > Ty + 0.9 && Ay > 27)

{

velocityLeft = -100;

velocityRight = -100;

}

else if (Ay < Ty - 0.9 && Ay < 43) {

velocityLeft = 100;

velocityRight = 100;

}

else {

velocityLeft = 0;

velocityRight = 0;

}//控制球的位置

if (Ay < 27 && Tx > 10) {

velocityLeft = 100;

velocityRight = 100;

}

//传球的配合

double Tr = robot->rotation;

if (Tr < 0.001)

Tr = Tr + 360;

if (Tr > 360.001)

Tr = Tr - 360;

if (Tr > 270.5)

velocityRight = velocityRight + fabs(Tr - 270);

else if (Tr < 269.5)

velocityLeft = velocityLeft + fabs(Tr - 270);

robot->velocityLeft = velocityLeft;

robot->velocityRight = velocityRight;

if (Ay > Ty - 0.9 && Ay < 43)

{

velocityLeft = 100;

velocityRight = 100;

}

if (Ay < 27)

{

velocityLeft = 100;

velocityRight = 100;

}

if (Ay > 43)

{

velocityLeft = -100;

velocityRight = -100;

}

if (Tr < 0.001)

Tr = Tr + 360;

if (Tr > 360.001)

Tr = Tr - 360;

if (Tr > 270.5)

velocityRight = velocityRight + fabs(Tr - 270);

else if (Tr < 269.5)

velocityLeft = velocityLeft + fabs(Tr - 270);

robot->velocityLeft = velocityLeft;

robot->velocityRight = velocityRight;

}

//进攻

void Attack2(Robot\* robot, Environment\* pEnv)

{

Vector3D t = pEnv->currentBall.pos;

double r = robot->rotation;

if (r < 0) r += 360;

if (r > 360) r -= 360;

double vl = robot->velocityLeft, vr = robot->velocityRight;

if (t.y > pEnv->fieldBounds.top - 2.5) t.y = pEnv->fieldBounds.top - 2.5;

if (t.y < pEnv->fieldBounds.bottom + 2.5) t.y = pEnv->fieldBounds.bottom + 2.5;

if (t.x > pEnv->fieldBounds.right - 3) t.x = pEnv->fieldBounds.right - 3;

if (t.x < pEnv->fieldBounds.left + 3) t.x = pEnv->fieldBounds.left + 3;

double dx = robot->pos.x - t.x;

double dy = robot->pos.y - t.y;

double dxAdjusted = dx;

double angleToPoint = 0;

if (fabs(robot->pos.y - t.y) > 7 || t.x > robot->pos.x)

dxAdjusted -= 5;

if (dxAdjusted == 0)

{

if (dy > 0)

angleToPoint = 270;

else

angleToPoint = 90;

}

else if (dy == 0)

{

if (dxAdjusted > 0)

angleToPoint = 360;

else

angleToPoint = 180;

}

else

angleToPoint = atan(fabs(dy / dx)) \* 180.0 / PI;

if (dxAdjusted > 0)

{

if (dy > 0)

angleToPoint -= 180;

else if (dy < 0)

angleToPoint = 180 - angleToPoint;

}

if (dxAdjusted < 0)

{

if (dy > 0)

angleToPoint = -angleToPoint;

else if (dy < 0)

angleToPoint = 90 - angleToPoint;

}

if (angleToPoint < 0) angleToPoint = angleToPoint + 360;

if (angleToPoint > 360) angleToPoint = angleToPoint - 360;

if (angleToPoint > 360) angleToPoint = angleToPoint - 360;

double c = r;

double angleDiff = fabs(r - angleToPoint);

if (angleDiff < 40)

{

vl = 100;

vr = 100;

if (c > angleToPoint)

vl -= 10;

if (c < angleToPoint)

vr -= 10;

}

else

{

if (r > angleToPoint)

{

if (angleDiff > 180)

vl += 360 - angleDiff;

else

vr += angleDiff;

}

if (r < angleToPoint)

{

if (angleDiff > 180)

vr += 360 - angleDiff;

else

vl += angleDiff;

}

}

robot->velocityLeft = vl;

robot->velocityRight = vr;

//NearBound2(robot, vl, vr, pEnv);

}

//根据机器人的位置和朝向，对速度进行调整，以确保机器人不会靠近场地边界

void NearBound2(Robot\* robot, double vl, double vr, Environment\* pEnv)

{

Vector3D a = robot->pos;

double r = robot->rotation;

if (a.y > pEnv->fieldBounds.top - 15 && r > 45 && r < 130)

{

if (vl > 0)

vl /= 3;

if (vr > 0)

vr /= 3;

}

if (a.y < pEnv->fieldBounds.bottom + 15 && r < -45 && r > -130)

{

if (vl > 0) vl /= 3;

if (vr > 0) vr /= 3;

}

if (a.x > pEnv->fieldBounds.right - 10)

{

if (vl > 0)

vl /= 2;

if (vr > 0)

vr /= 2;

}

if (a.x < pEnv->fieldBounds.left + 10)

{

if (vl > 0)

vl /= 2;

if (vr > 0)

vr /= 2;

}

robot->velocityLeft = vl;

robot->velocityRight = vr;

}

//回防防守

void Defend(Robot\* robot, Environment\* pEnv, double low, double high)

{

double vl = 0, vr = 0;

Vector3D z = pEnv->currentBall.pos;

double Tx = pEnv->goalBounds.right - z.x;

double Ty = pEnv->fieldBounds.top - z.y;

Vector3D a = robot->pos;

a.x = pEnv->goalBounds.right - a.x;

a.y = pEnv->fieldBounds.top - a.y;

if (a.y > Ty + 0.9 && a.y > low)

{

vl = -100;

vr = -100;

}

if (a.y < Ty - 0.9 && a.y < high)

{

vl = 100;

vr = 100;

}

if (a.y < low)

{

vl = 100;

vr = 100;

}

if (a.y > high)

{

vl = -100;

vr = -100;

}

double Tr = robot->rotation;

if (Tr < 0.001)

Tr += 360;

if (Tr > 360.001)

Tr -= 360;

if (Tr > 270.5)

vr += fabs(Tr - 270);

else if (Tr < 269.5)

vl += fabs(Tr - 270);

NearBound2(robot, vl, vr, pEnv);

}

//判断在以球员robot为圆心，r为半径的圆范围内是否有己方球员（如果有返回true,否则返回false）

bool Find\_Players\_around\_Player(Robot\* robot, double r, Environment\* pEnv) {

Vector3D robot\_pos = robot->pos;

for (int i = 0; i < 5; i++) {

Vector3D player\_pos = pEnv->home[i].pos;

double D\_robot\_player = sqrt(pow(fabs(robot\_pos.x - player\_pos.x), 2) + pow(fabs(robot\_pos.y - player\_pos.y), 2));

if (D\_robot\_player < r) return false;

}

return true;

}

//让球员robot尝试去抢球,策略为：将球员robot移动到球的位置上

void tryToStealBall(Robot\* robot, Environment\* pEnv) {

double x = pEnv->currentBall.pos.x, y = pEnv->currentBall.pos.y;

Position(robot, x, y);

}

double Distance(double x1, double x2, double y1, double y2)

{

return sqrt((x1 - x2) \* (x1 - x2) + (y1 - y2) \* (y1 - y2));

}

double GetAngle(Vector3D a, Vector3D b)//b位于a方向

{

double k;

double angle;

if (a.x == b.x)//垂直于x轴

{

angle = 90;

if (b.y > a.y)

return angle;

else return -angle;

}

else

{

k = (b.y - a.y) / (b.x - a.x);

angle = 180 \* 1.0 / PI \* atan(k);

if (angle > 0)

{

if (a.x < b.x)

return angle;

else

return angle - 180;

}

else

{

if (a.x < b.x)

return angle;

else

return angle + 180;

}

}

}

/\*void KickBall(Environment\* pEnv, Robot\* robot, int id, double x, double y) {

Vector3D predictedBall = pEnv->predictedBall.pos;

Vector3D ball = pEnv->currentBall.pos;

Vector3D t;

Vector3D d\_pos;

d\_pos.x = x;

d\_pos.y = y;

double dx, dy, k;

double tx, ty, length;// tx，ty为射门点,length为距离

double tdx, tdy;

double le = 8;//斜边长

double rt\_angle;

double angle;

dx = predictedBall.x - x;

dy = predictedBall.y - y;

k = dy / dx;

tdx = sqrt(le \* le / (k \* k + 1));

tdy = fabs(k \* tdx);

if (dx < 0)

tx = predictedBall.x - tdx;

if (dx >= 0)

tx = predictedBall.x + tdx;

if (dy < 0)

ty = predictedBall.x - tdx;

if (dy >= 0)

ty = predictedBall.x + tdx;

dx = robot->pos.x - tx;

dy = robot->pos.y - ty;

length = sqrt(dx \* dx + dy \* dy);

t.x = tx;

t.y = ty;

rt\_angle = GetAngle(robot->pos, t);

angle = GetAngle(t, d\_pos);

if (length > 2.5)//距离过远

to(robot, id, tx, ty);

else if (rt\_angle != angle)//转动

Angle(robot, angle);

else //射门

Position(robot, ball.x, ball.y);

}\*/

//弧线射门

void huxian(Environment\* pEnv, int whichrobot)

{

Vector3D freball = pEnv->predictedBall.pos;

Vector3D paly1 = pEnv->home[whichrobot].pos;

Vector3D ball = pEnv->currentBall.pos;

Vector3D lastball = pEnv->lastBall.pos;

Vector3D ansball;

int t = 3;

//动态预测球的位置

ansball.x = ball.x + t \* (ball.x - lastball.x);

ansball.y = ball.y + t \* (ball.y - lastball.y);

//确定有效的预射门范围(蓝队)

double y1, y2;

y1 = (lastball.x - paly1.x) \* (70 - paly1.y) / (0 - paly1.y) + paly1.y;

y2 = (ball.x - paly1.x) \* (110 - paly1.y) / (0 - paly1.x) + paly1.x;

//确定目标点 sx,sy

double sx, sy;

double bc;

bc = sqrt((ansball.x - paly1.x) \* (ansball.x - paly1.x) + (ansball.y - paly1.y) \* (ansball.y - paly1.y));

double k\_bc;

k\_bc = (paly1.y - ansball.y) / (paly1.x - ansball.x);

double k\_ds;

k\_ds = 1.0 / k\_bc;

double k\_ab;

k\_ab = (90 - ansball.y) / (0 - ansball.x);

sx = (k\_ab \* -(ansball.x) + ansball.y + k\_ds \* (ansball.x + paly1.x) / 2 - (ansball.y + paly1.y) / 2) / (k\_ds - k\_ab);

sy = k\_ab \* (sx - ansball.x) + ansball.y;

if (paly1.y > y1 && paly1.y < y2) Position(pEnv, whichrobot, sx, sy);

else Position(pEnv, whichrobot, ball.x, ball.y);

}

bool AccessTo\_A(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x < 15 && pEnv->home[i].pos.y < 115 && pEnv->home[i].pos.y > 65)

{

num++;

}

}

if (num >= 1)return false;

return true;

}

//判断禁区超人数

bool AccessTo\_B(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x < 37.5 && pEnv->home[i].pos.y < 130 && pEnv->home[i].pos.y > 50)

{

num++;

}

}

if (num >= 2)return false;

return true;

}

//后卫

//旋转方向

bool whererotation(Environment\* pEnv, int id)

{

double desired\_angle;

Vector3D palyer = pEnv->home[id].pos;

Vector3D curball = pEnv->currentBall.pos;

Vector3D preball = pEnv->predictedBall.pos;

double cur\_angle = pEnv->home[id].rotation;//现在的机器人角度

if (curball.x - palyer.x == 0)//没有斜率

{

if ((cur\_angle > 0 && cur\_angle < 90) || (cur\_angle > -180 && cur\_angle < -90))

{

//Velocity(robot, vl, vr + 20);//改变右轮的速度，逆时针旋转

return 1;

}

else

{

//Velocity(robot, vl + 20, vr);//改变左轮的速度，顺时针旋转

return 0;

}

}

else {

double k = (curball.y - palyer.y) / (curball.x - palyer.x);

desired\_angle = atan(k) \* 360;//目标角度

if ((desired\_angle - cur\_angle > 0 && desired\_angle - cur\_angle < 90) || (desired\_angle > 180 && cur\_angle < 270))//头朝向机器人和球得到一条直线的右方向

{

//Velocity(robot, vl, vr + 20)//逆时针

return 1;

}

else

{

//Velocity(robot, vl + 20, vr);//顺时针

return 0;

}

}

}

bool Securityoffensive(Environment\* pEnv) {

Vector3D ball = pEnv->currentBall.pos;

double D;

int at = 0;

for (int i = 1; i < 5; i++)

{

D = (pEnv->home[i].pos.x - ball.x) \* (pEnv->home[i].pos.x - ball.x) +

(pEnv->home[i].pos.y - ball.y) \* (pEnv->home[i].pos.y - ball.y);

if (D < 64)

at++;

}

return at >= 1;

}

void PredictBall\_cnt(Environment\* pEnv, int cnt, double\* A)//周期预测，cnt表示第几个周期，最多预测三个周期

{

double dx = pEnv->currentBall.pos.x - pEnv->lastBall.pos.x;

double dy = pEnv->currentBall.pos.y - pEnv->lastBall.pos.y;

double predictedX = pEnv->currentBall.pos.x;

double predictedY = pEnv->currentBall.pos.y;

// 这里假设cnt的最大值是3，最小值是1

if (cnt < 1) cnt = 1;

if (cnt > 3) cnt = 3;

// 对每个周期进行预测

for (int i = 0; i < cnt; ++i) {

// 预测下一个周期的位置

predictedX += dx;

predictedY += dy;

// 检查是否反弹于x轴方向的边界

if (predictedX < 0 || predictedX > 220) {

dx = -dx; // 反转x方向的速度以模拟反弹

if (predictedX < 0) {

predictedX = -predictedX; // 如果越界，位置反弹

}

else if (predictedX > 220) {

predictedX = 440 - predictedX; // 如果越界，位置反弹

}

}

// 检查是否反弹于y轴方向的边界

if (predictedY < 0 || predictedY > 180) {

dy = -dy; // 反转y方向的速度以模拟反弹

if (predictedY < 0) {

predictedY = -predictedY; // 如果越界，位置反弹

}

else if (predictedY > 180) {

predictedY = 360 - predictedY; // 如果越界，位置反弹

}

}

}

// 将最终的预测数据存储到A数组

A[0] = predictedX;

A[1] = predictedY;

}

void spin(Environment\* pEnv, int id1, double x1, double y1) {

Vector3D player = pEnv->home[id1].pos; // 后卫

double x2 = player.x;

double y2 = player.y;

double D = calc\_distance(x1, y1, x2, y2);

if (D < 15)

Velocity(&(pEnv->home[id1]), 125, -125);

}

void Rotate(Robot\* robot, int d, int gear)

{

if (d == 0)//逆时针

{

switch (gear)

{

case 1:

Velocity(robot, -45, 50); break;

case 2:

Velocity(robot, -80, 85); break;

case 3:

Velocity(robot, -90, 95); break;

case 4:

Velocity(robot, -125, 125); break;

default:

break;

}

}

else//顺时针

{

switch (gear)

{

case 1:

Velocity(robot, 50, -45); break;

case 2:

Velocity(robot, 85, -80); break;

case 3:

Velocity(robot, 95, -90); break;

case 4:

Velocity(robot, 125, -125); break;

default:

break;

}

}

}

//判断球周围是否有我方球员

bool OurRobotBesideBall(Environment\* pEnv)

{

Vector3D ball = pEnv->currentBall.pos;

for (int i = 0; i < 5; i++) {

Vector3D player\_pos = pEnv->home[i].pos;

double D\_robot\_ball = sqrt(pow(fabs(ball.x - player\_pos.x), 2) + pow(fabs(ball.y - player\_pos.y), 2));

if (D\_robot\_ball > 6) return false;

}

return true;

}

bool OpponentRobotBesideBall(Environment\* pEnv)

{

Vector3D ball = pEnv->currentBall.pos;

for (int i = 0; i < 5; i++) {

Vector3D Opponent\_pos = pEnv->home[i].pos;

double D\_robot\_ball = sqrt(pow(fabs(ball.x - Opponent\_pos.x), 2) + pow(fabs(ball.y - Opponent\_pos.y), 2));

if (D\_robot\_ball > 3) return false;

}

return true;

}

//前锋是否距球近

bool AttackerClose(Environment\* pEnv)

{

Vector3D ball = pEnv->currentBall.pos;

Vector3D Attacker\_pos = pEnv->home[4].pos;

double D\_robot\_ball = sqrt(pow(fabs(ball.x - Attacker\_pos.x), 2) + pow(fabs(ball.y - Attacker\_pos.y), 2));

if (D\_robot\_ball > 8) return false;

return true;

}

bool AccessTo\_E(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x < 37.5 && pEnv->opponent[i].pos.y < 130 && pEnv->opponent[i].pos.y > 50)

{

num++;

}

}

if (num >= 3)return false;

return true;

}

//判断对方前场人数 35<=x<=110

bool AccessTo\_F(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x <= 110 && pEnv->opponent[i].pos.x >= 35)

{

num++;

}

}

if (num >= 3)return false;

return true;

}

//判断我方后场人数 110<x<182.5

bool AccessTo\_C(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 110 && pEnv->opponent[i].pos.x < 182, 5)

{

num++;

}

}

if (num >= 3)return false;

if (num < 3)return true;

}

/\*

//判断我方大禁区人数 x>182.5 50<y<130

bool AccessTo\_D(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 182.5 && pEnv->opponent[i].pos.y < 130 && pEnv->opponent[i].pos.y > 50)

{

num++;

}

}

if (num >= 3)return false;

if (num < 3)return true;

}

int robot1(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x < 35 && pEnv->home[i].pos.y > 135)

{

num++;

}

}

return num;

}

int robot2(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x < 35 && pEnv->home[i].pos.y < 135 && pEnv->home[i].pos.y > 45)

{

num++;

}

}

return num;

}

int robot3(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x < 35 && pEnv->home[i].pos.y < 45)

{

num++;

}

}

return num;

}

int robot4(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 35 && pEnv->home[i].pos.x < 110 && pEnv->home[i].pos.y < 90)

{

num++;

}

}

return num;

}

int robot5(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 35 && pEnv->home[i].pos.x < 110 && pEnv->home[i].pos.y > 90)

{

num++;

}

}

return num;

}

int robot6(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 110 && pEnv->home[i].pos.x < 165 && pEnv->home[i].pos.y >150)

{

num++;

}

}

return num;

}

int robot7(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 110 && pEnv->home[i].pos.x < 165 && pEnv->home[i].pos.y > 90 && pEnv->home[i].pos.y < 150)

{

num++;

}

}

return num;

}

int robot8(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 110 && pEnv->home[i].pos.x < 165 && pEnv->home[i].pos.y > 30 && pEnv->home[i].pos.y < 90)

{

num++;

}

}

return num;

}

int robot9(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x > 110 && pEnv->home[i].pos.x < 165 && pEnv->home[i].pos.y < 30)

{

num++;

}

}

return num;

}

int robot10(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->home[i].pos.x >= 175 && pEnv->home[i].pos.y >= 135 && pEnv->home[i].pos.y <= 45)

{

num++;

}

}

return num;

}

int opponent1(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x < 35 && pEnv->opponent[i].pos.y > 135)

{

num++;

}

}

return num;

}

int opponent2(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x < 35 && pEnv->opponent[i].pos.y < 135 && pEnv->opponent[i].pos.y > 45)

{

num++;

}

}

return num;

}

int opponent3(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x < 35 && pEnv->opponent[i].pos.y < 45)

{

num++;

}

}

return num;

}

int opponent4(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 35 && pEnv->opponent[i].pos.x < 110 && pEnv->opponent[i].pos.y < 90)

{

num++;

}

}

return num;

}

int opponent5(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 35 && pEnv->opponent[i].pos.x < 110 && pEnv->opponent[i].pos.y > 90)

{

num++;

}

}

return num;

}

int opponent6(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 110 && pEnv->opponent[i].pos.x < 165 && pEnv->opponent[i].pos.y >150)

{

num++;

}

}

return num;

}

int opponent7(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 110 && pEnv->opponent[i].pos.x < 165 && pEnv->opponent[i].pos.y > 90 && pEnv->opponent[i].pos.y < 150)

{

num++;

}

}

return num;

}

int opponent8(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 110 && pEnv->opponent[i].pos.x < 165 && pEnv->opponent[i].pos.y > 30 && pEnv->opponent[i].pos.y < 90)

{

num++;

}

}

return num;

}

int opponent9(Environment\* pEnv)

{

int num = 0;

for (int i = 1; i < 5; i++)

{

if (pEnv->opponent[i].pos.x > 110 && pEnv->opponent[i].pos.x < 165 && pEnv->opponent[i].pos.y < 30)

{

num++;

}

}

return num;

}\*/

static bool f1 = 0, f2 = 0, f3 = 0, fmeng = 0, fadj = 0, ok1 = 0, ok2 = 0;

void Goalie3(Environment\* pEnv) // Ա

{

//if (fabs(pEnv->home[0].pos.x - 215) <= 1)

//{

// //Velocity(&pEnv->home[0], 0, 0);

// Angle(&pEnv->home[0], 0);

//}

Vector3D CurrentBall = pEnv->currentBall.pos;

Vector3D lastBall = pEnv->lastBall.pos;

Vector3D player0 = pEnv->home[0].pos;

double ballx = CurrentBall.x;

double bally = CurrentBall.y;

double lastballx = lastBall.x;

double lastbally = lastBall.y;

if (fabs(pEnv->home[0].rotation + 78.69) <= 5 && fabs(pEnv->currentBall.pos.x - 210) <= 5 && fabs(pEnv->currentBall.pos.y - 110) <= 5)//

{

PositionV(pEnv, 0, pEnv->currentBall.pos.x, pEnv->currentBall.pos.y, 150);

fmeng = 1;

return;

}

if (fmeng && ballx - lastballx < 0 || ballx < 165)

{

fmeng = 0;

}

if ((fabs(pEnv->home[0].pos.x - 215) <= 5 && (fabs(pEnv->home[0].rotation + 90) <= 10 || fabs(pEnv->home[0].rotation - 90) <= 10)))

{

fadj = 1;

}

else {

fadj = 0;

//fmeng = 0;

}

if (!fmeng && !fadj && !(fabs(pEnv->home[0].pos.x - 215) <= 5 && (fabs(pEnv->home[0].rotation + 90) <= 5 || fabs(pEnv->home[0].rotation - 90) <= 5)))

{

//Velocity(&pEnv->home[0], -90, 90);

adj(pEnv);

return;

}

move(pEnv);

}

void PositionV(Environment\* pEnv, int id, double x, double y, double v)

{

PositionV(&(pEnv->home[id]), x, y, v);

}

void PositionV(Robot\* robot, double x, double y, double v)

{

double desired\_angle = 0, theta\_e = 0, d\_angle = 0;

double vl = 0, vr = 0, vc = v;

double dx, dy, d\_e, Ka = 10.0 / 90.0;

dx = x - robot->pos.x;

dy = y - robot->pos.y;

d\_e = sqrt(dx \* dx + dy \* dy);

if (dx == 0 && dy == 0)

desired\_angle = 90;

else

desired\_angle = 180. / PI \* atan2((double)(dy), (double)(dx));

theta\_e = desired\_angle - robot->rotation;

while (theta\_e > 180)

theta\_e -= 360;

while (theta\_e < -180)

theta\_e += 360;

if (d\_e > 100.)

Ka = 17. / 90.;

else if (d\_e > 50)

Ka = 19. / 90.;

else if (d\_e > 30)

Ka = 21. / 90.;

else if (d\_e > 20)

Ka = 23. / 90.;

else

Ka = 25. / 90.;

if (theta\_e > 95 || theta\_e < -95)

{

theta\_e += 180;

if (theta\_e > 180)

theta\_e -= 360;

if (theta\_e > 80)

theta\_e = 80;

if (theta\_e < -80)

theta\_e = -80;

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.1;

vr = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else if (theta\_e < 85 && theta\_e > -85)

{

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.1;

vr = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else

{

vr = (+.17 \* theta\_e);

vl = (-.17 \* theta\_e);

}

Velocity(robot, vl, vr);

}

void move(Environment\* pEnv)

{

// 存储预测3个周期后的球坐标

double A[2] = { 0 };

PredictBall\_cnt(pEnv, 3, A);

Vector3D CurrentBall = pEnv->currentBall.pos;

Vector3D lastBall = pEnv->lastBall.pos;

Vector3D player0 = pEnv->home[0].pos;

double ballx = CurrentBall.x;

double bally = CurrentBall.y;

double lastballx = lastBall.x;

double lastbally = lastBall.y;

// 球坐标点和门框两端点构成的三角形，以球坐标为顶角构成的角平分线与215的交点

double Y = calculateAngleBisector(216, CurrentBall.x, CurrentBall.y, 216, 70, 216, 110);

// 守门员与球的距离

double D = calc\_distance(ballx, bally, player0.x, player0.y);

double ball\_player = 1e10;

if (ballx < 110)

{

PositionG(pEnv, 0, 215, 90);

//stop(pEnv, 0, 215, 90);

}

else {

//if (ballx < 175)

//{

// if (bally > 120)

// {

// PositionG(pEnv, 0, 215, 112);

// //stop(pEnv, 0, 215, 112);

// }

// else if (bally < 60)

// {

// PositionG(pEnv, 0, 215, 68);

// //stop(pEnv, 0, 215, 68);

// }

// else {

// PositionG(pEnv, 0, 215, Y);

// //stop(pEnv, 0, 215, bally);

// }

//}

//else {

// if (bally > 120)

// {

// PositionG(pEnv, 0, 215, 112);

// //stop(pEnv, 0, 215, 112);

// }

// else if (bally < 60)

// {

// PositionG(pEnv, 0, 215, 68);

// //stop(pEnv, 0, 215, 68);

// }

// else {

// PositionG(pEnv, 0, 215, A[1]);

// //stop(pEnv, 0, 215, bally);

// }

//}

if (ballx < 175)

{

if (A[1] > 120)

{

PositionG(pEnv, 0, 215, 112);

//stop(pEnv, 0, 215, 112);

}

else if (A[1] < 60)

{

PositionG(pEnv, 0, 215, 68);

//stop(pEnv, 0, 215, 68);

}

else {

PositionG(pEnv, 0, 215, Y);

//stop(pEnv, 0, 215, bally);

}

}

else {

if (A[1] > 120)

{

PositionG(pEnv, 0, 215, 112);

//stop(pEnv, 0, 215, 112);

}

else if (A[1] < 60)

{

PositionG(pEnv, 0, 215, 68);

//stop(pEnv, 0, 215, 68);

}

else {

PositionG(pEnv, 0, 215, A[1]);

//stop(pEnv, 0, 215, bally);

}

}

}

}

void adj(Environment\* pEnv)

{

//Angle(&pEnv->home[0], 0);

double y = pEnv->home[0].pos.y;

if (fabs(pEnv->home[0].pos.x - 215) > 5)

{

Angle(&pEnv->home[0], 0);

f3 = 1;

}

/\*else {

move(pEnv);

return;

}\*/

if (f3 && (fabs(pEnv->home[0].rotation) <= 5 || fabs(pEnv->home[0].rotation - 180) <= 5 || fabs(pEnv->home[0].rotation + 180) <= 5))

{

f1 = 1;

f3 = 0;

}

if (f1)

{

Position(pEnv, 0, 215, y);

stop(pEnv, 0, 215, y);

f1 = 0;

}

if (fabs(pEnv->home[0].pos.x - 215) > 2 && fabs(pEnv->home[0].pos.x - 215) < 10)

{

PositionV(pEnv, 0, 215, y, 30);

}

else if (fabs(pEnv->home[0].pos.x - 215) <= 2)

{

//Velocity(&pEnv->home[0], -90, 90);

//Velocity(&pEnv->home[0], 0, 0);

if (fabs(pEnv->home[0].rotation + 90) <= 20 || fabs(pEnv->home[0].rotation - 90) <= 20)

{

//Velocity(&pEnv->home[0], -90, 90);

move(pEnv);

return;

}

Angle(&pEnv->home[0], -90);

}

}

void PositionG(Environment\* pEnv, int id, double x, double y)

{

PositionG(&(pEnv->home[id]), x, y); // ˶ ñ Ϊid Ļ ƶ x, y

}

void PositionG(Robot\* robot, double x, double y) // ˶ ʵ

{

double desired\_angle = 0, theta\_e = 0, d\_angle = 0; // desirea\_angleΪ Ƕȣ theta\_eΪҪת ĽǶȣ d\_angle ʱû õ ֪ ġ

double vl, vr, vc = 175, v\_adjust = 120;

double dx, dy, d\_e, Ka = 10.0 / 90.0; // d\_eΪŷ Ͼ 룬KaΪ ϵ

dx = x - robot->pos.x; // Ŀ x ˵ ǰx Ĳ ֵs

dy = y - robot->pos.y; // Ŀ y ˵ ǰy Ĳ ֵ

d\_e = sqrt(dx \* dx + dy \* dy); // ŷʽ

if (d\_e <= 2)

{

Velocity(robot, 0, 0);

Angle(robot, -90.0);

return;

}

else

{

if (dx == 0 && dy == 0)

desired\_angle = 90; // ˺ Ŀ ͬһ 㣬 趨 Ƕ Ϊ90

else

desired\_angle = 180. / PI \* atan2((double)(dy), (double)(dx)); // 򣬼 Ƕȡ

theta\_e = desired\_angle - robot->rotation; // Ҫת ٶ

while (theta\_e > 180)

theta\_e -= 360;

while (theta\_e < -180)

theta\_e += 360; // Ϊ Ҷ Դﵽ Ҫ ĽǶȣ ⲿ ڰ ת Ŀ 㵽С 180 ʹ֮ ĴﵽĿ

if (d\_e > 100.)

Ka = 17. / 90.;

else if (d\_e > 50)

Ka = 19. / 90.;

else if (d\_e > 30)

Ka = 21. / 90.;

else if (d\_e > 20)

Ka = 23. / 90.;

else

Ka = 25. / 90.; // ŷʽ ϵ

if (theta\_e > 95 || theta\_e < -95) // Ƕ ת Ҫ 95

{

theta\_e += 180;

if (theta\_e > 180)

theta\_e -= 360;

if (theta\_e > 80)

theta\_e = 80;

if (theta\_e < -80)

theta\_e = -80;

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.1; // ϽǶ Ӱ ϵ

vr = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (-vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else if (theta\_e < 85 && theta\_e > -85) // Ƕ ת ҪС 85

{

if (d\_e < 5.0 && fabs(theta\_e) < 40)

Ka = 0.1; // ϽǶ Ӱ ϵ

vr = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) + Ka \* theta\_e);

vl = (vc \* (1.0 / (1.0 + exp(-3.0 \* d\_e)) - 0.3) - Ka \* theta\_e);

}

else // Ҫת ĽǶȾ ֵ [85, 95]

{

vr = (+.17 \* theta\_e);

vl = (-.17 \* theta\_e);

} // ĳ ַ ٶ

Velocity(robot, vl, vr); // е

}

}