تیم شبیه سازی دوبعدی فوتبال موعود

کیان زمانی، صبحان بهاریان، سینا محمدی، امیر رضا شهانگیر، پوریا عزیمی

توضیحات خلاصه : تیم شبیه سازی دوبعدی موعود متشکل از دانش آموزان پایه دهم رشته ریاضی دبیرستان موعود می باشند. اعضای تیم از ابتدای مهرماه سال ۱۳۹۶ آموزش برنامه نویسی و لینوکس را آغاز نموده اند. در حال حاضر دو هفته می باشد که با لیگ شبیه سازی دوبعدی دانش آموزی و بیس اراعه شده آشنا شده اند.
در این گزارش فنی به طور خلاصه الگوریتم مارک و شوت و مکان یابی بازیکنان توضیح داده خواهد شد.

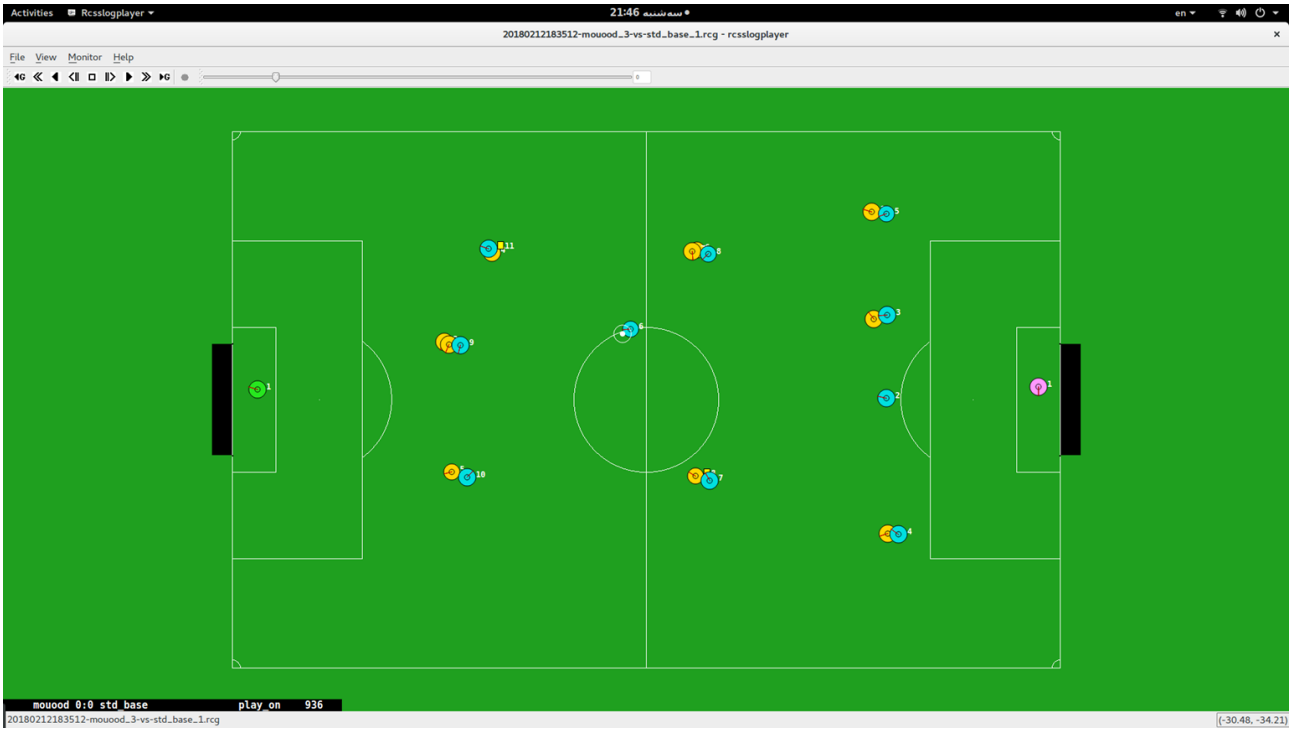
الگوریتم مارک

در کد تیم شبیه سازی دوبعدی موعود الگوریتم مارک به این صورت می باشد که در زمانی که توپ در دست بازیکنان حریف می باشد هر بازیکن خودی نزدیک ترین بازیکن حریف به خودش را مارک می کند. برای اینکه این مارک باعث شود بازیکن حریف توانایی فرار عمق نداشته باشد بازیکنان حریفی که نزدیک به خط آفساید ما هستند پشت سرشان قرار میگیریم تا حرکت عمقی انجام ندهند. اما بازیکنان دیگر بین توپ و بازیکن حریف قرار میگیرند. طوری که پاس مستقیم نگیرند.

```
if(opp_min < self_min && opp_min < mate_min){
    if(self_min < opp_min){
        Body_Intercept().execute( agent );
        return true;
    }else{
        Vector2D self_pos = wm.self().pos();
        double min_dist = 1000;
        int min_opp = 0;
        for (int i=2;i<=11;i++){
            if(wm.theirPlayer(i)==NULL ||wm.theirPlayer(i)->unum()<=0)
                continue;
            Vector2D opp_pos = wm.theirPlayer(i)->pos();
            if(opp_pos.dist(self_pos) < min_dist){
                min_dist = opp_pos.dist(self_pos);
                min_opp = i;
            }
        }
        if(min_opp>0){
            if (wm.theirPlayer(min_opp)->pos().x < wm.ourDefenseLineX() + 10){
                Body_GoToPoint2010(wm.theirPlayer(min_opp)->pos() - Vector2D(2,0),0.7,100).execute(agent);
            }else{
                Vector2D tar = wm.theirPlayer(min_opp)->pos();
                tar += (wm.ball().inertiaPoint(opp_min) - wm.theirPlayer(min_opp)->pos()).setLength(1.0);
                Body_GoToPoint2010(tar,0.7,100).execute(agent);
            }

            return true;
        }
    }
}
```

بازیکنی که نزدیک ترین مکان به بازیکن صاحب توپ حریف را داشته باشد با استفاده از کلاس اینترسپت سعی در گرفتن توپ می نماید.



مکان یابی بازیکنان : برای بهتر شدن ترکیب بازی مقداری آن را به صورت عددی تغییر دادیم و به اعداد زیر رسیدیم که حالت بهتری نسبت به بیس بوده است.

```
rcsc::Vector2D Bhv_BasicMove::getPosition(const rcsc::WorldModel & wm, int self_unum){
    int ball_step = 0;
    if ( wm.gameMode().type() == GameMode::PlayOn
         || wm.gameMode().type() == GameMode::GoalKick_ )
    {
        ball_step = std::min( 1000, wm.interceptTable()->teammateReachCycle() );
        ball_step = std::min( ball_step, wm.interceptTable()->opponentReachCycle() );
        ball_step = std::min( ball_step, wm.interceptTable()->selfReachCycle() );
    }

    Vector2D ball_pos = wm.ball().inertiaPoint( ball_step );

    dlog.addText( Logger::TEAM,
                  __FILE__": HOME POSITION: ball pos=(%.1f %.1f) step=%d",
                  ball_pos.x, ball_pos.y,
                  ball_step );

    std::vector<Vector2D> positions(12);
    double min_x_rectengle[12]={0,-52,-52,-52,-52,-52,-10,-15,-15,0,0,0};
    double max_x_rectengle[12]={0,-48,-10,-10,-10,-10,40,30,30,50,50,50};
    double min_y_rectengle[12]={0,-2,-20,-10,-30,10,-10,-30, 0,-20,-30, 0};
    double max_y_rectengle[12]={0,+2, 10, 20,-10,30, 10, 0, 30, 20, 0, 30};

    for(int i=1; i<=11; i++){
          double xx_rectengle = max_x_rectengle[i] - min_x_rectengle[i];
          double yy_rectengle = max_y_rectengle[i] - min_y_rectengle[i];
          double x_ball = ball_pos.x + 52.5;
          x_ball /= 105.5;
          double y_ball = ball_pos.y + 34;
          y_ball /= 68.0;
          double x_pos = xx_rectengle * x_ball + min_x_rectengle[i];
          double y_pos = yy_rectengle * y_ball + min_y_rectengle[i];
          positions[i] = Vector2D(x_pos,y_pos);
    }
```

تصمیم گیری در زمان حمله : در زمان حمله با تغییر جزیی استراتژی به نتایج بهتری دست پیدا کردیم. برای این مورد بازیکن در صورتی که حریف نزدیکش نباشد به دریبل زدن با سرعت کمتری نسبت به بیس ادامه می دهد در صورتی که بازیکن حریف نزدیکش شود توپ را به بازیکنان دیگر پاس میدهد.

```
bool
Bhv_BasicOffensiveKick::execute( PlayerAgent * agent )
{
    dlog.addText( Logger::TEAM,
                  __FILE__": Bhv_BasicOffensiveKick" );

    const WorldModel & wm = agent->world();

    if(shoot(agent)){
        return true;
    }

    const PlayerPtrCont & opps = wm.opponentsFromSelf();
    const PlayerObject * nearest_opp
        = ( opps.empty()
            ? static_cast< PlayerObject * >( 0 )
            : opps.front() );
    const double nearest_opp_dist = ( nearest_opp
                                      ? nearest_opp->distFromSelf()
                                      : 1000.0 );
    if(nearest_opp_dist < 3){
        if(pass(agent))
            return true;
    }
    if(dribble(agent)){
        return true;
    }

    if(pass(agent))
        return true;

    if ( nearest_opp_dist > 2.5 )
    {
        dlog.addText( Logger::TEAM,
                      __FILE__": hold" );
        agent->debugClient().addMessage( "OffKickHold" );
        Body_HoldBall().execute( agent );
        return true;
    }
    clearball(agent);
    return true;
```

شوت : در الگوریتم شوت یکی از تیرک های دروازه انتخاب می شود سپس در صورتی که در سکتور توپ تا نقطه مورد نظر نزدیک به تریک بازیکن حریف نبود آن نقطه به عنوان مکان شوت در نظر گرفته می شود.

```
bool Bhv_BasicOffensiveKick::shoot( rcsc::PlayerAgent * agent ){
    const WorldModel & wm = agent->world();
    Vector2D ball_pos = wm.ball().pos();
    Vector2D center_goal = Vector2D(52.5,0);
    if(ball_pos.dist(center_goal) > 25)
            return false;
    Vector2D left_goal = Vector2D(52.5,6);
    Vector2D right_goal = Vector2D(52.5,-6);

    if(left_goal.dist(ball_pos) < right_goal.dist(ball_pos)){
        Sector2D a(ball_pos,1,left_goal.dist(ball_pos),(left_goal - ball_pos).th() - 10,(left_goal - ball_pos).th()+10);
        if(wm.existOpponentIn(a,1,true)){
            return false;
        }
        Body_SmartKick(left_goal,3,0.1,2).execute(agent);
    }else{
        Sector2D a(ball_pos,1,right_goal.dist(ball_pos),(right_goal - ball_pos).th() - 10,(right_goal - ball_pos).th()+10);
        if(wm.existOpponentIn(a,1,true)){
            return false;
        }
        Body_SmartKick(right_goal,3,0.1,2).execute(agent);
    }
    return true;
}
```