RoboSoccer Attack
Strategy

Authors
Himank Agrawal - 201101119
Kunal Chawla - 201101170
Anuj Kosambi - 201101135
Agent: Currantes
Team Name: Rowdy Rooster

Functions:

1. **void givePass**(PlayerAgent * agent ,int passTaker,int passReceiver,double ballspeed,Vector2D);
   passtaker player gives pass to the player receiver with the given initial speed and to the given point of receiving.
   This function take a kick of with the given initial speed and to the direction of the given receiving point. It also use addSayMessage function to call the receiver to collect the ball at the given point.
2. **void collectPass**(PlayerAgent * agent)
   It check for the last receiver from AudioMemory.Pass list and it also derived the pass point of receiving the ball from it. If the receiver player is close enough to collect ball then it start running towards the ball else it goes towards the receiving point. After capturing the ball, it changes the lastRole of the receiver to the normal player.
3. **Vector2D best_catching_point**(Vector2D ball_pos,Vector2D player_pos,int thr,double ratio,double vel1,double vel2)
   If the player at position (ball_pos) passes ball in the direction with angle thr(degree) and with avg speed vel1, then our function gives best catching point for another player at position(player_pos) with maximum speed vel2.
4.
   class POINTS
   {
   public:
   double dis;
   int player_num;
   Vector2D tackle_point;
   };
5. **POINTS passAttack**(PlayerAgent* agent)
   given agent it find the best point to pass and player number who will collect the pass. Srtategy used in this function is explained in term paper.
6. **pair<Vector2D,double> get_safe_point**(Vector2D ball_point,int angle_start,int angle_end,int angle_thr,double sector_radius,vector<Vector2D> opponentslist,PlayerAgent * agent,int recr_steps)
   This function runs recursively up to some steps(recr_steps) and returns the safest

point in the region defined by . angle_start and angle_end with radius of sector_radius, taking into account the positions of opponent (opponentslist) which changes relative to ball at every recursion level.

7. **bool dribbleAttack**(PlayerAgent * agent)
   this function checks returns false if it is not safe to dribble, otherwise it dribbles in safe direction and returns true. if the closest opponent is very far than it dribble toward goal otherwise it dribbles in the safest direction using get_safe_point function
8. **void makeAttack**(PlayerAgent* agent)
   this function chooses the type of attack according to player position and scenario of field.

Helper functions:

1. double getfirstspeed(double avg_vel,double distance_travel);
   return initial speed ball required to travel given distance with given avg speed.
2. bool isoutfield(Vector2D point);
   return true if given point is out of the field.
3. bool isinregion(Vector2D point,Vector2D bl,Vector2D tr);
   return true if Rectangle with bl (bottom_left) and tr (top_right) points contains given point;