Phase Moon Team
Description Paper

Ariana Hajighasemi*, Arshida Moosavi[1], Niyayesh Naeimi[2], and Raeika Maroufi[2]

[1]Salam Hgih School, Tehran, Iran
arianahajighasemi@yahoo.com

Abstract. In this paper we represent a description of Phase Moon team in Robocup 2021 junior soccer simulation. Simulated robots are programed to play soccer in a virtual field using artificial intelligence and some algorithms for defending, attacking and etc. The program is written in python language which is so popular and powerful today.

Keywords: programming, AI, robotics

1 Introduction

Phase Moon team founded in 2017 and participated in some competitions including Iran Open 2018 until 2021, Juniorcup 2018 and 2021, Salamcup 2017 until 2021 and Robocup Asia Pacific 2018. At first our team began its work with Junior Soccer Openweight league and in Salamcup 2021 which was a presentation-competition we reached to first place. Then we moved into Robocup simulation challenge and we reached to first place in Iran Open 2021. And now for Robocup 2021 we are going to participate in Junior Soccer Simulation challenge to get more experience.

1. Ariana Hajighasemi
2. Arshida Moosavi
3. Niyayesh Naeimi
4. Raeika Maroufi

2 Software and Programming

In this year Robocup Committee introduced an awesome simulator software made with Webots engine. Webots is an open source and multi-platform desktop application used to simulate robots. It provides a complete development environment to model, program and simulate robots. It has been designed for a professional use, and it is widely used in industry, education and research.

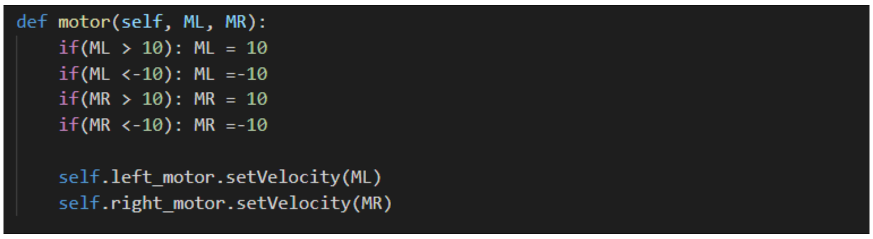
2.1 Programing Language

In our project because the simulation software (rcj-soccer-sim) is based on python programming language we had to write our code in this language. Python is one of the most popular programming languages in the world and it's easy to use. The main property that makes this language so popular is that it lets you work quickly and integrate systems more effectively. We can use different libraries for different jobs simply by importing them in our code like math library for math calculations and etc.

2.2 Functional Programming

Sometimes we repeat a part of a code many times in a program. In this case declaring a function will help us to avoid repeating that part of the code. We defined various functions in our program to make it cleaner and also easier to understand. The first function that we defined is readData() which gets all information that the simulator gives us and This function assign it into some variables that we will use in our algorithms and calculations.

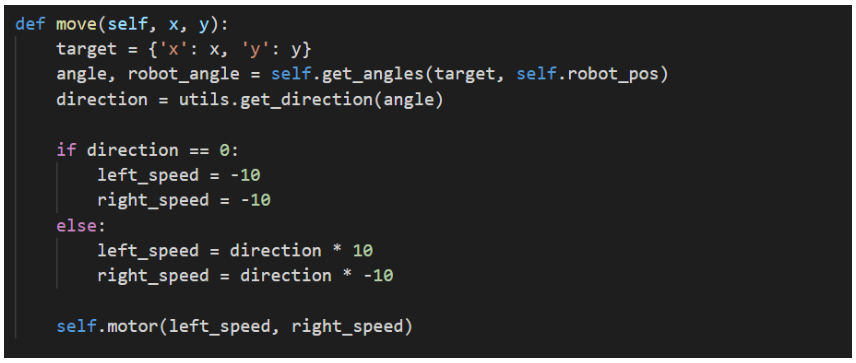
Motor function. This function takes two arguments as inputs that will represent left and right motors velocity and set these values to robot's motors velocity. As it was set for Webots that the maximum velocity is 10 and the minimum velocity is -10 we wrote some conditions to filter the value of each motor's velocity. The motor function is shown below.



```python
def motor(self, ML, MR):
    if(ML > 10): ML = 10
    if(ML <-10): ML =-10
    if(MR > 10): MR = 10
    if(MR <-10): MR =-10

    self.left_motor.setVelocity(ML)
    self.right_motor.setVelocity(MR)
```

Move function. In so many states we need to tell the robot to move toward a coordination position. For example, to fallow the ball in game we tell the robot to move toward the ball position. Because we were going to do this action many times, we made a function which takes two arguments for co-ordinational position, x and y. To move towards the given position, we calculate the angle between the robot and to target position using a function that was declared before. After calculating the angle, we made a variable called direction to see which position is the target standing, front, right or left. Three values are assigned to this variable, 0 for front position, 1 for right position and -1 for left position.



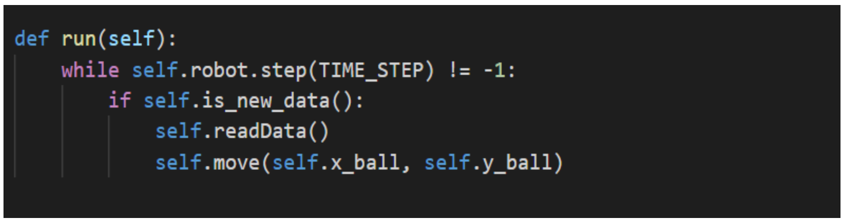
```
def move(self, x, y):
    target = {'x': x, 'y': y}
    angle, robot_angle = self.get_angles(target, self.robot_pos)
    direction = utils.get_direction(angle)

    if direction == 0:
        left_speed = -10
        right_speed = -10
    else:
        left_speed = direction * 10
        right_speed = direction * -10

    self.motor(left_speed, right_speed)
```

Predict Neutral Spots function. It wasn't necessary to create a new function for this job but to make the code cleaner we defined a function to predict that which position will be chosen for placing the ball in lack of progress situation. This function returns us the number of neutral spots that might be chosen by the referee for placing the ball and we can tell one of our robots (if necessary) to go into that position and wait for the ball to spawn in that position. The algorithm to find the neutral spot is similar to referee code in simulator.

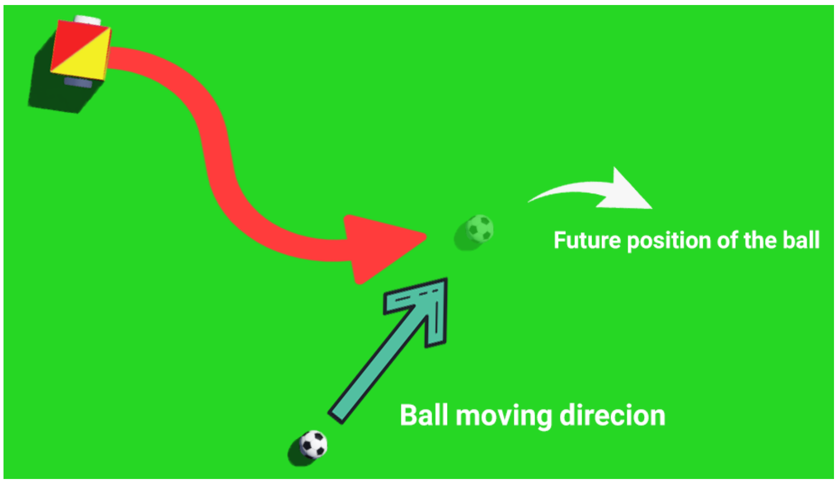
Run function. This function is called by Webots simulator to run the codes and algorithm we want to run for each robot. So, we need a continuing loop to process all the data that Webots gives us and send required commands to robots.



```
def run(self):
    while self.robot.step(TIME_STEP) != -1:
        if self.is_new_data():
            self.readData()
            self.move(self.x_ball, self.y_ball)
```

2.3 Artificial Intelligence

All Robots need a way to think and make decisions by processing the information which is gotten from the world around. We programed the robots to estimate the best decision in any situation. As an example, the robot will calculate the future position of the ball using its velocity and moving direction and instead of moving toward the ball it moves toward the future position of the ball and try to push it in a direction that will goes to the opponent goal.



2.4 Code Format

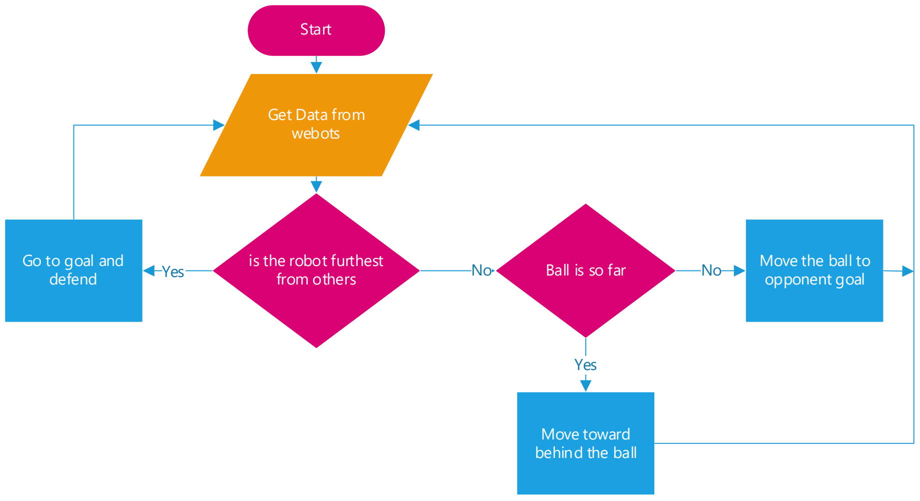
Each robot has its own specific python file that contains a class which have the whole codes and algorithms for that robot. A main python file will use all these three classes for each robot and run their codes. There might be some functions that we will use in three files for all robots. Repeating the definition of this function in each robot's file doesn't make sense. So, we have a utils file which contains all functions that might be used in all robot files. The structure of Webots code is shown below.

```
controllers/
├── robot/
│    └── robot.py
│    └── robot1.py
│    └── robot2.py
│    └── robot3.py
|    └── utils.py
```

3 Strategy and Algorithms

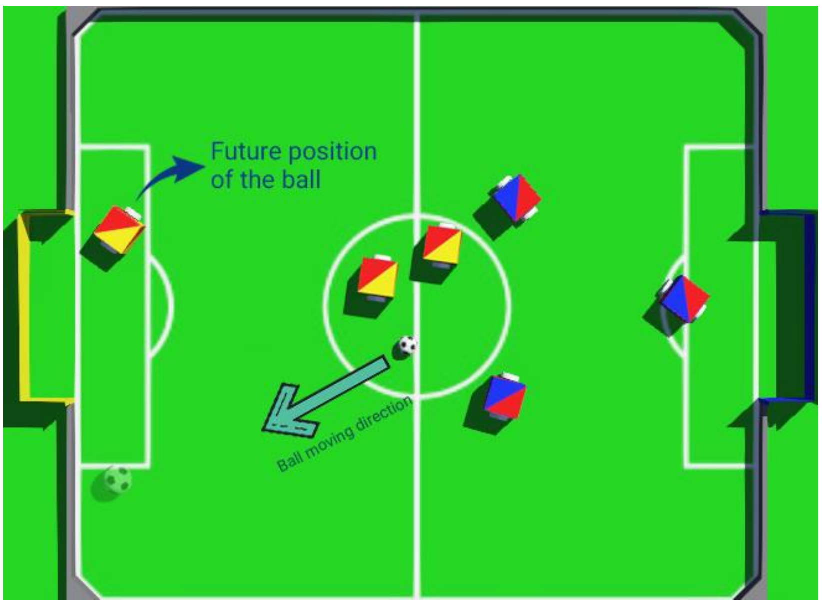
3.1 Main Playing Algorithm

As it was mentioned in section 2.3 the robot in every single moment process all the information, took from Webots and makes the best decision using its artificial intelligence. The main playing algorithm is that one robot which is furthest to the ball than others go to our goal for defending. The two other robots will first go behind the ball and then move the ball into opponent goal. In some cases, like lack of progress for ball position robots will go to predicted neutral spots and wait for the ball to spawn in that position.


Start
Get Data from webots
is the robot furthest from others
Yes
Go to goal and defend
No
Ball is so far
No
Move the ball to opponent goal
Yes
Move toward behind the ball

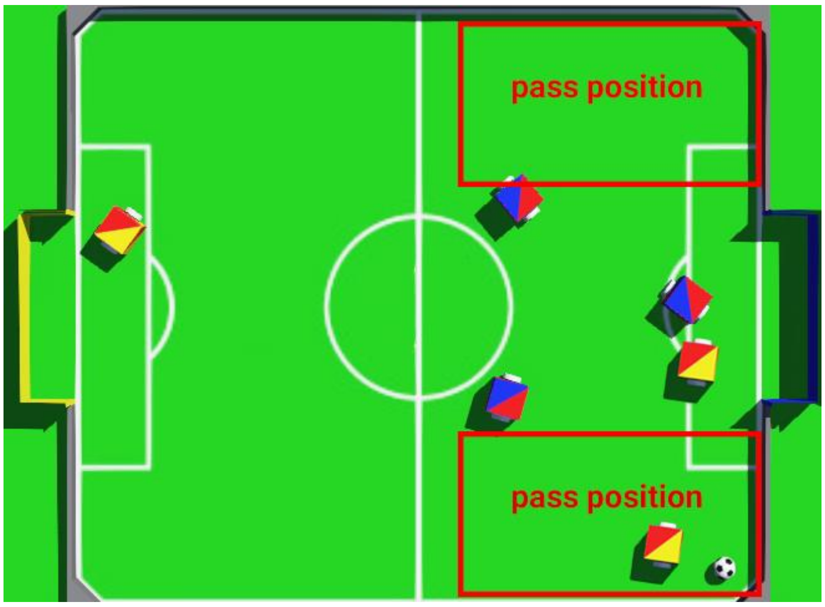
3.2 Goal Keeper

The goal keeper robot will first calculate the ball move function and predict the future ball position in front of the goal. And then it goas toward that position and when the ball gets near to the robot and the distance be greater than a value the robots role will change to forward and it goas toward the ball.



3.3 Pass to another robot

It is possible to have one of our forward robots in a position to take the pass from another robot. To do this we need to know if the ball is in the pass position or not. These positions are shown in the figure below. If the ball is in the pass position the other robot will go in front of opponent's goal and wait for receiving the pass from the other robot.



3.4 Lack Of Progress Positions

There are 7 neutral spots that the simulator will chose to put robots or the ball in lack of progress situation. As we mentioned in 2.2 section, we have a function to predict the neutral spot chosen by the referee. After finding this neutral position we tell the robot to go behind that position if the ball stays resident in anywhere.

4 Conclusion and Future Works

In conclusion we found that working in a simulated robot will allow us to apply new ideas and improve our programming skills. Also, we are planning to work on real robots maybe in major league in future competition

References

1. This project uses Webots (http://www.cyberbotics.com), an open-source mobile robot simulation software developed by Cyberbotics Ltd.
2. Python Software Foundation. Python Language Reference, version 2.7. Available at http://www.python.org
3. MATLAB and Signal Processing Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States. http://www.mathworks.com/
4. Robocup junior soccer https://github.com/RoboCupJuniorTC/awesome-rcj-soccer